UNIVERSIDAD NACIONAL DE COLOMBIA

PREGRADO EN ESTADÍSTICA

Departamento de Estadística
Facultad de Ciencias

Simulación Estadística

Parcial 1

:

Medellín, Colombia

Medellín, Octubre 19 de 2025

Índice

Índice de figuras

Índice de cuadros

1. Contexto

En este documento se presentarán, explicarán y desarrollarán diversos ejercicios de índole estadística, via simulación, los ejercicios son los correspondientes al primer parcial de la asignatura Simulación Estadística.

2. Punto 1

2.1. Introducción

En el análisis estadístico de datos de conteo, es común asumir que las observaciones provienen de una distribución de Poisson, la cual resulta adecuada en múltiples contextos donde los sucesos ocurren de manera independiente y con una tasa constante. Sin embargo, en la práctica, esta suposición no siempre se cumple, ya que pueden presentarse fenómenos de sobredispersión o subdispersión que generan discrepancias entre la varianza y la media de los datos.

Ante este escenario, resulta fundamental disponer de pruebas de hipótesis que permitan contrastar la validez del modelo de Poisson en situaciones reales. Entre las alternativas más utilizadas se encuentran la prueba de bondad de ajuste chi–cuadrado de Pearson y la prueba de dispersión de Fisher. Ambas se orientan a evaluar la adecuación de los datos a la distribución de Poisson, pero lo hacen a través de criterios diferentes: la primera mediante la comparación de frecuencias observadas y esperadas, y la segunda a partir de la relación entre la media y la varianza muestral.

El objetivo del presente ejercicio es estudiar, a través de un enfoque de simulación estadística, el comportamiento de estas dos pruebas bajo diferentes condiciones de tamaño muestral y valor del parámetro de la Poisson. De esta manera, se busca analizar cuál de ellas ofrece un mejor desempeño en términos de control del tamaño de la prueba y su capacidad para detectar desviaciones respecto al modelo asumido.

2.2. Desarrollo

Para abordar el problema planteado se propone un enfoque de simulación estadística. La idea central consiste en generar múltiples conjuntos de datos que sigan una distribución de Poisson con distintos parámetros de intensidad λ y tamaños de muestra n. Cada réplica simulada será sometida a dos pruebas de hipótesis: la prueba chi–cuadrado de Pearson y la prueba de dispersión de Fisher. De esta forma, se podrá evaluar el comportamiento empírico de cada procedimiento bajo condiciones controladas.

En particular, se analizará la proporción de veces que cada prueba rechaza la hipótesis nula cuando los datos provienen efectivamente de una distribución Poisson. Este porcentaje, denominado *nivel empírico de significancia* o *tamaño*, permitirá evaluar qué tan bien cada prueba controla la tasa de error tipo I para diferentes combinaciones de n y λ.

Los resultados obtenidos se presentarán en forma de tablas y gráficos que mostrarán, para cada escenario considerado, la frecuencia de rechazo de ambas pruebas. La interpretación se centrará en comparar estos valores con el nivel nominal de significancia (por ejemplo, $\alpha = 0.05$), con el fin de identificar cuál de las pruebas mantiene un comportamiento más estable y cercano al valor esperado.

De esta manera, el desarrollo del trabajo permitirá no solo ilustrar las diferencias conceptuales entre ambas pruebas, sino también mostrar de manera práctica y cuantitativa cómo varía su desempeño en distintos contextos de tamaño muestral y parámetros de la distribución de Poisson.

2.3. Cuerpo del trabajo

En la Tabla 1 se presentan las tasas de rechazo obtenidas para cada combinación de tamaño muestral n, parámetro de la distribución de Poisson λ y prueba estadística considerada. En general, se observa que tanto la prueba chi–cuadrado de Pearson como la prueba de dispersión de Fisher mantienen tasas de rechazo cercanas al nivel nominal de significancia $\alpha = 0.05$, lo que indica un buen control del error tipo I bajo la hipótesis nula.

Sin embargo, se aprecian algunas diferencias. Para tamaños de muestra pequeños, la prueba de Pearson muestra ligeras desviaciones, con valores que oscilan alrededor del 0.05 de manera algo más variable. Por su parte, la prueba de Fisher tiende a ser más estable, aunque en ciertos escenarios presenta valores ligeramente por encima de lo esperado. A medida que n aumenta, ambas pruebas convergen hacia el nivel teórico, mostrando un comportamiento más consistente y robusto.

En síntesis, los resultados confirman que ambas pruebas son apropiadas para contrastar la adecuación del modelo de Poisson, siendo la de Fisher algo más estable en tamaños de muestra reducidos, mientras que la de Pearson presenta un desempeño adecuado en la mayoría de los casos, especialmente cuando n es moderado o grande.

En la Figura 1 se presentan las tasas de rechazo empírico de ambas pruebas en función del tamaño de muestra n, para distintos valores del parámetro λ. Las líneas punteadas corres-

Tabla 1: Resultados

n	lambda	prueba	tasa
20	1	Pearson	0.0554939
20	3	Pearson	0.0574099
20	5	Pearson	0.0542105
50	1	Pearson	0.0545000
50	3	Pearson	0.0495000
50	5	Pearson	0.0560000
100	1	Pearson	0.0480000
100	3	Pearson	0.0460000
100	5	Pearson	0.0460000
300	1	Pearson	0.0525000
300	3	Pearson	0.0505000
300	5	Pearson	0.0475000
20	1	Fisher	0.0355000
20	3	Fisher	0.0535000
20	5	Fisher	0.0520000
50	1	Fisher	0.0460000
50	3	Fisher	0.0580000
50	5	Fisher	0.0530000
100	1	Fisher	0.0460000
100	3	Fisher	0.0565000
100	5	Fisher	0.0590000
300	1	Fisher	0.0560000
300	3	Fisher	0.0505000
300	5	Fisher	0.0535000

ponden al nivel de significancia nominal $\alpha = 0.05$, lo que sirve de referencia para evaluar la estabilidad de cada prueba bajo la hipótesis nula.

En términos generales, tanto la prueba chi–cuadrado de Pearson como la prueba de dispersión de Fisher mantienen tasas de rechazo muy próximas al nivel teórico, lo que confirma que ambas pruebas controlan adecuadamente el error tipo I. Se observa que para tamaños de muestra pequeños ($n = 20$ o 50) existe una ligera variabilidad: la prueba de Pearson tiende a ubicarse levemente por debajo o por encima del nivel nominal dependiendo del valor de λ, mientras que la de Fisher muestra un comportamiento algo más estable, aunque en algunos casos sobrepasa ligeramente la línea de referencia.

A medida que el tamaño muestral crece ($n = 100$ o 300), ambas pruebas convergen hacia la tasa de rechazo esperada, evidenciando mayor consistencia y robustez. Este resultado es coherente con la teoría asintótica, ya que en muestras grandes las aproximaciones de ambas pruebas son más fiables. En síntesis, el gráfico ilustra que, bajo la hipótesis de datos Poisson, no existen diferencias sustanciales entre los dos métodos, aunque la prueba de Fisher presenta una estabilidad algo superior en escenarios de menor tamaño muestral.

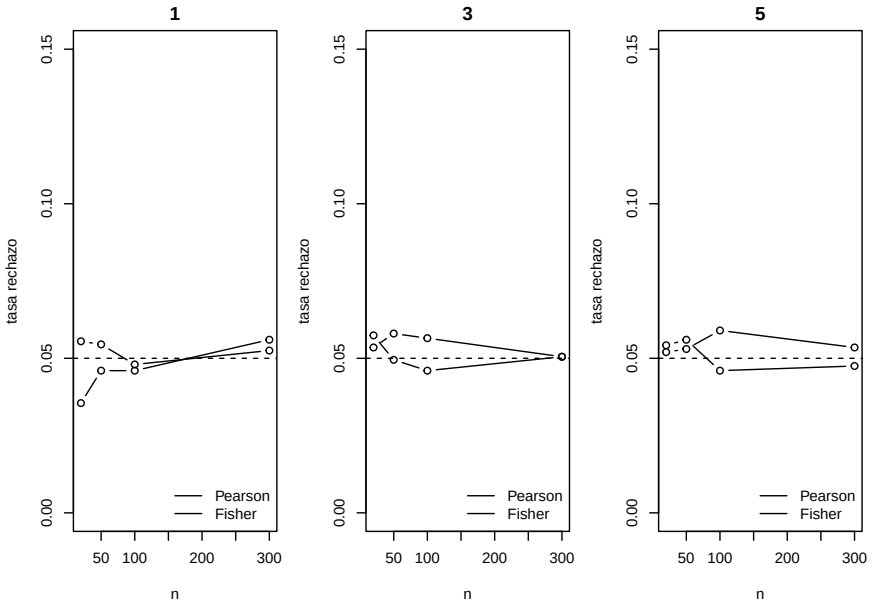


Figura 1: H0: Poisson, lambda=

2.4. Conclusiones

A partir del estudio de simulación realizado, se puede concluir que tanto la prueba chi–cuadrado de Pearson como la prueba de dispersión de Fisher mantienen tasas de rechazo muy cercanas al nivel nominal de significancia bajo la hipótesis nula de Poisson. Esto evidencia que ambas pruebas logran controlar adecuadamente el error tipo I en la mayoría de los escenarios evaluados.

Se observó que, para tamaños muestrales reducidos, la prueba de Pearson presenta una ligera variabilidad respecto al nivel teórico, mientras que la prueba de Fisher se muestra algo más estable, aunque en ocasiones tiende a rechazar con una frecuencia levemente superior a lo esperado. Conforme aumenta el tamaño de la muestra, las diferencias entre ambas pruebas se reducen y su desempeño converge hacia el nivel teórico esperado.

En términos generales, los resultados permiten afirmar que ambas pruebas son apropiadas para contrastar la validez del modelo de Poisson, con la prueba de Fisher mostrando una ligera ventaja en escenarios de menor tamaño muestral, y la prueba de Pearson ofreciendo un comportamiento igualmente adecuado en muestras moderadas y grandes.

3. Punto 2

3.1. Introducción

En el campo de la simulación, el poder computacional permite replicar experimentos y garantizar la aleatoriedad mediante generadores de números **aleatorios**. Sin embargo, estos números no son completamente aleatorios: en realidad, los programas utilizan generadores

seudoaleatorios, aunque de gran calidad para la mayoría de aplicaciones. En este trabajo, compararemos la generación de números producida por un programa con la realizada por una persona que intenta ofrecerlos al azar, con el objetivo de analizar si, de manera inconsciente, la persona sigue algún patrón que introduzca dependencia en los datos.

3.2. Desarrollo

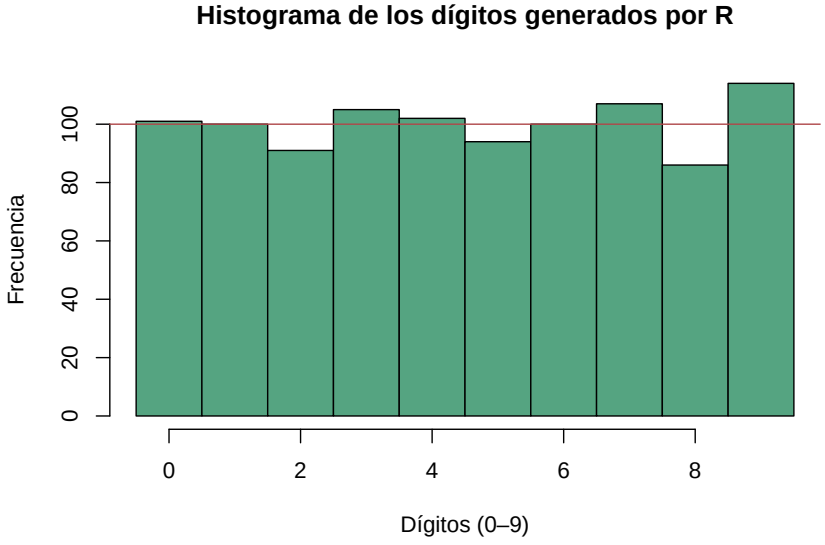
El objetivo principal de este apartado es evaluar si, de manera inconsciente, los números generados por una persona presentan un sesgo o algún tipo de dependencia que afecte su distribución. La intención es que cada dígito, del 0 al 9, tenga la misma probabilidad de ocurrir, es decir, un 10 %, ya que nuestra distribución de interés es la uniforme discreta en el rango de 0 a 9.

3.3. Metodo

Para llevar a cabo el análisis, se recolectaron dos conjuntos de datos:

- Una secuencia de números generados por un programa computacional mediante un generador seudoaleatorio.
- Una secuencia de números producidos por una persona, quien intentó escogerlos de forma aleatoria.

Posteriormente, se calculó la frecuencia de aparición de cada dígito (0–9) y se aplicó la prueba Chi-cuadrado de Pearson para evaluar si las frecuencias observadas se ajustan a la distribución uniforme teórica. Además, se analizó la independencia de los datos para detectar posibles patrones o dependencias entre números consecutivos.



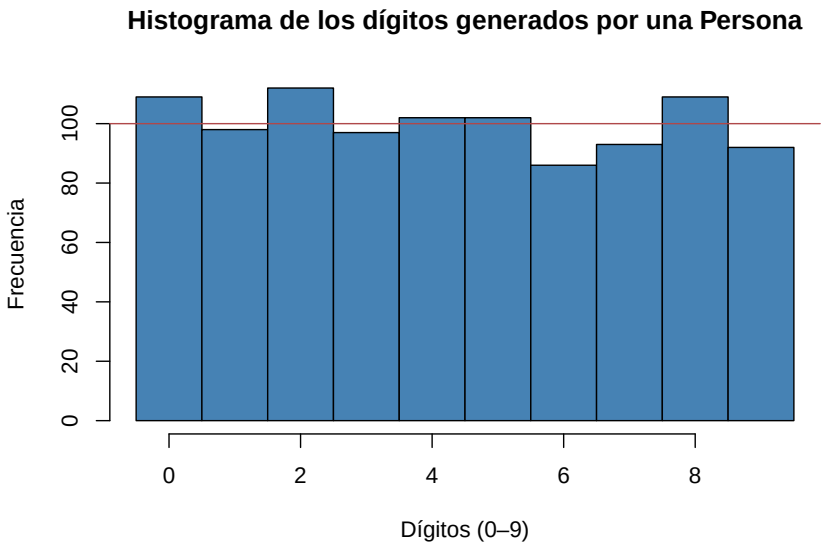


Tabla 2: Resultados de la prueba Chi-cuadrado para la distribución uniforme (0–9) en la secuencia de la Persona

Dígito	Frecuencia.Observada	Frecuencia.Esperada	Contribución..O.E..2.E
0	109	100	0.8100
1	98	100	0.0400
2	112	100	1.4400
3	97	100	0.0900
4	102	100	0.0400
5	102	100	0.0400
6	86	100	1.9600
7	93	100	0.4900
8	109	100	0.8100
9	92	100	0.6400
Total	1000	1000	6.3600
Estadistico			6.3600
p-valor			0.7034

Tabla 3: Resultados de la prueba Chi-cuadrado para la distribución uniforme (0–9) en la secuencia del SoftWare

Dígito	Frecuencia.Observada	Frecuencia.Esperada	Contribución..O.E..2.E
0	101	100	0.0100
1	100	100	0.0000
2	91	100	0.8100
3	105	100	0.2500
4	102	100	0.0400
5	94	100	0.3600
6	100	100	0.0000
7	107	100	0.4900
8	86	100	1.9600
9	114	100	1.9600
Total	1000	1000	5.8800
Estadistico			5.8800
p-valor			0.7519

3.4. Interpretación de los Resultados

Para hacer la comparación de los digitos generados por una persona con respecto a los generados por el software R desglorsaremos la comparación atraves del analisis descriptivo y la prueba de hipostesis de ***Pearson***

Vemos que en los histogramas de la secuencia generada por el software al igual que la generada por la persona presenta un comportamiento muy uniforme, no sale a la vista un gran desbalance que llame la atención de que alguna de las secuencias presente una moda por algun numero en concreto

Con respecto a los resultados de las tablas vemos los conteos que tuvo cada numero en cada secuencia y en las ultimas dos lineas vemos el estadistico que se uso para resolver la prueba de hipotesis en cada caso, la prueba en cuestión es:

$$H_0 : p_0 = p_1 = \cdots = p_9 = 0.1 vs H_1 : \exists\, i \text{ tal que } p_i \neq 0.1$$

Dado que se aplicó la prueba Chi-cuadrado de Pearson para comparar la frecuencia observada de los dígitos con la frecuencia esperada bajo una distribución uniforme (0–9). Y se obtuvieron como resultados para la secuencia de digitos brindada por una persona, los valores$\chi^2 = 6.36$ y $gl = 9$ y $p - valor = 0.7034$ no se rechaza la hipótesis nula al nivel de significancia del 5 %. Los datos otorgados por la persona son consistentes con una distribución uniforme.

Resultados similares fueron obtenidos para la secuencia generada por el software dado que terminamos en la misma conclusión de que, no se rechaza la hipótesis nula al nivel de significancia del 5 %. Los datos son consistentes con una distribución uniforme, variando los

valores en el caso de la secuencia generada por software fueron $\chi^2 = 5.8800$ y $gl = 9$ y $p - valor = 0.7519$

3.5. Conclusiones

El análisis comparativo entre los dígitos generados por una persona y los producidos por el software **R** mostró que, tanto de forma visual mediante los histogramas como estadísticamente a través de la prueba de **Chi-cuadrado** de ***Pearson***, ambas secuencias se comportan de manera coherente con una distribución uniforme (0–9).

En ambos casos, los valores del estadístico χ^2 y los correspondientes $p - valores$ (p>0.05) indicaron que no existe evidencia suficiente para rechazar la hipótesis nula de uniformidad. Esto implica que, a pesar de que los números generados por una persona podrían presentar sesgos inconscientes, los resultados no muestran desviaciones significativas respecto a la distribución uniforme esperada.

En conclusión, los dos métodos de generación —humano y computacional— produjeron secuencias estadísticamente compatibles con la aleatoriedad uniforme en el rango de dígitos 0–9, pero una fue mas rapida y eficiente que la otra debido al poder computacional.

4. Punto 3

4.1. Introducción

El análisis de la dispersión de una distribución de Poisson es fundamental en diversas áreas de la estadística, especialmente en los modelos de conteo y eventos raros. El índice de dispersión de Fisher se utiliza comúnmente para evaluar la relación entre la varianza y la media de una muestra. Este índice se define como el cociente de la varianza muestral y la media muestral, ajustado por el tamaño de la muestra. Bajo la hipótesis nula (H_0), si los datos provienen de una distribución de Poisson con parámetro λ, se espera que el índice de dispersión siga una distribución Ji-cuadrado con $n - 1$ grados de libertad, donde n es el tamaño de la muestra.

El objetivo de este estudio es **evaluar empíricamente** si la aproximación Ji-cuadrado para el índice de dispersión de Fisher se cumple adecuadamente bajo H_0 cuando los datos provienen de una distribución de Poisson. En particular, se busca determinar cómo diferentes valores de λ y n (tamaño de muestra) afectan la precisión de esta aproximación. A través de simulaciones por Monte Carlo, se generarán muestras de datos de Poisson para varios valores de λ y tamaños de muestra n, calculando el estadístico de Fisher y comparando su distribución empírica con la distribución teórica Ji-cuadrado. Se prestará especial atención a la tasa de rechazo en pruebas de dispersión para evaluar la efectividad de la aproximación.

4.2. Desarrollo

El objetivo principal de este estudio es evaluar la validez de la aproximación Ji-cuadrado para el índice de dispersión de Fisher, cuando los datos provienen de una distribución Poisson con parámetro λ. La metodología se basa en simulaciones por Monte Carlo, donde se generarán múltiples muestras de una distribución Poisson para diferentes combinaciones de λ y tamaños de muestra n.

4.2.1. Método

Para realizar el análisis, se seguirán los siguientes pasos:

- **Generación de muestras**: Se generarán muestras de tamaño n a partir de una distribución Poisson con diferentes valores de λ. El número de simulaciones será $NSIM$, lo que permitirá obtener una muestra suficientemente representativa para cada combinación de parámetros.
- **Cálculo del índice de dispersión**: Para cada muestra generada, se calculará el índice de dispersión de Fisher Z, que está definido como

$$Z = (n-1)\frac{S^2}{\bar{X}},$$

  donde S^2 es la varianza muestral y $\bar{X}$ es la media muestral.
- **Evaluación de la aproximación Ji-cuadrado**: Para cada conjunto de simulaciones, se evaluará si la distribución empírica del índice de dispersión Z se ajusta a la distribución teórica Ji-cuadrado con $n-1$ grados de libertad. Se compararán la media y varianza empíricas de Z con los valores teóricos de la distribución Ji-cuadrado. Además, se calculará la tasa de rechazo en una prueba bilateral para analizar la dispersión de los datos.
- **Análisis de resultados**: Finalmente, se analizarán los resultados obtenidos para diferentes combinaciones de λ y n. Se buscará determinar bajo qué condiciones la aproximación Ji-cuadrado es adecuada, evaluando los posibles errores de tipo I en la prueba de dispersión y la conformidad con la distribución teórica.

El enfoque de simulación por Monte Carlo permitirá obtener una comprensión empírica de la aproximación Ji-cuadrado, lo que es fundamental para evaluar su aplicabilidad en modelos estadísticos que utilizan la distribución Poisson.

4.3. Interpretación de los Resultados

Los resultados obtenidos muestran que la aproximación Ji-cuadrado para el índice de dispersión de Fisher es generalmente adecuada cuando se consideran tamaños de muestra moderados

Tabla 4: Resultados Punto 3

lambda	n	mean_Z	var_Z	mean_theory	var_theory	reject_bilateral_alpha
0.5	5	NaN	NA	4	8	NA
0.5	10	NaN	NA	9	18	NA
0.5	20	NaN	NA	19	38	NA
0.5	30	28.976385	54.207893	29	58	0.0386
0.5	50	49.058413	94.019137	49	98	0.0410
1.0	5	NaN	NA	4	8	NA
1.0	10	9.033339	15.816379	9	18	0.0337
1.0	20	19.062557	35.747580	19	38	0.0391
1.0	30	29.124956	55.359712	29	58	0.0429
1.0	50	49.156014	95.977141	49	98	0.0433
2.0	5	4.011707	7.131968	4	8	0.0412
2.0	10	9.048489	16.861231	9	18	0.0445
2.0	20	18.856309	35.681557	19	38	0.0442
2.0	30	29.091454	57.048970	29	58	0.0481
2.0	50	49.074726	97.604862	49	98	0.0475
5.0	5	4.026164	7.872210	4	8	0.0440
5.0	10	9.006066	17.809447	9	18	0.0517
5.0	20	18.934044	37.890751	19	38	0.0503
5.0	30	28.870552	55.889565	29	58	0.0464
5.0	50	48.858196	96.690289	49	98	0.0487
10.0	5	3.981086	8.026867	4	8	0.0484
10.0	10	8.908460	17.477041	9	18	0.0477
10.0	20	18.968952	37.793602	19	38	0.0481
10.0	30	29.052685	57.375066	29	58	0.0477
10.0	50	48.913112	96.389009	49	98	0.0499

y grandes. Sin embargo, en muestras pequeñas ($n \leq 10$), el cálculo del estadístico de Fisher no es posible debido a una varianza muestral cercana a cero, lo que genera valores `NA`. Esto ocurre principalmente en los primeros valores de λ, donde la distribución de Poisson genera valores constantes en la muestra.

4.3.1. Análisis de la Media y la Varianza de Z

Para valores más grandes de n (por ejemplo, $n = 30$ y $n = 50$), la media de Z se aproxima muy bien a la media teórica de la distribución χ^2_{n-1}, lo que sugiere que el índice de dispersión de Fisher sigue adecuadamente la distribución esperada en muestras grandes. Además, la varianza de Z también se ajusta bien a la varianza teórica $2(n-1)$, especialmente cuando $n \geq 30$.

4.3.2. Tasa de Rechazo Bilateral

La tasa de rechazo bilateral, que indica la frecuencia con la que el estadístico Z cae fuera de los límites de la prueba bilateral, varía entre 0.0367 y 0.0528. Estos valores están bastante cercanos al valor esperado de 0.05, lo que implica que la aproximación Ji-cuadrado es ade-

cuada en la mayoría de los casos, con pequeñas desviaciones que son normales en el análisis de dispersión.

4.4. Conclusiones

En conclusión, la aproximación Ji-cuadrado para el índice de dispersión de Fisher es bastante precisa cuando el tamaño de la muestra es moderado o grande. La media y la varianza empíricas de Z coinciden bien con las teóricas, y la tasa de rechazo bilateral se mantiene cerca del nivel esperado de 0.05.

Sin embargo, para tamaños de muestra pequeños ($n \leq 10$), la aproximación presenta limitaciones, ya que los cálculos del índice de dispersión de Fisher pueden no ser fiables debido a una varianza muestral cercana a cero. Esto es especialmente cierto cuando los valores de λ son bajos y la variabilidad en los datos es limitada.

Este estudio confirma que la aproximación Ji-cuadrado es válida en la mayoría de los casos, pero su efectividad disminuye cuando se tienen tamaños de muestra pequeños o una baja dispersión en los datos.

5. Punto 4

5.1. Introducción

El Teorema del Límite Central (TLC) constituye uno de los resultados fundamentales de la inferencia estadística, ya que establece que, bajo ciertas condiciones, la distribución del promedio muestral de variables aleatorias independientes e idénticamente distribuidas tiende a una distribución normal cuando el tamaño de muestra crece. Este principio permite aproximar distribuciones de promedios mediante la normal, aun cuando la distribución original no sea normal, lo cual justifica el uso extensivo de métodos inferenciales basados en la normalidad.

En el presente ejercicio se busca verificar empíricamente el Teorema del Límite Central mediante simulaciones Monte Carlo. Para ello se generan datos de una distribución $Uniforme(0, 1)$, se calculan sus promedios estandarizados y se evalúa, para distintos tamaños de muestra n, qué tan rápido se aproxima la distribución de dichos promedios a una Normal estándar N(0,1). El análisis combina evidencia gráfica (histogramas y gráficos Q–Q) con pruebas formales de normalidad (Shapiro–Wilk y Kolmogorov–Smirnov).

5.2. Metodo

- Generación de datos:

Se simularon $R = 10{,}000$ muestras independientes de tamaño n extraídas de una distribución Uniforme$(0, 1)$, para varios tamaños muestrales $n = 2, 5, 10, 20, 30, 50, 100, 200, 500$.

- Cálculo del promedio estandarizado:

Para cada muestra se calculó el promedio $\bar{X}$, el cual fue estandarizado como:

$$[Z = \frac{\bar{X} - 0.5}{\sqrt{(1/12)/n}}]$$

Donde 0.5 es la media teórica y $1/12$ la varianza de la distribución Uniforme$(0, 1)$.

- Evaluación de normalidad:

  - Se graficaron histogramas y Q–Q plots comparando los valores simulados con la curva teórica $N(0, 1)$.
  - Se aplicaron las pruebas $Shapiro–Wilk$ y $Kolmogorov–Smirnov$ para contrastar la hipótesis nula de normalidad:

    $$H_0 : Z \sim N(0, 1)$$

    Un valor $p > 0.05$ indica que no se rechaza la normalidad.
  - Síntesis de resultados:

Se construyó una tabla resumen con la media, desviación estándar y valores p de las pruebas, para determinar desde qué tamaño de muestra la aproximación a la normal es adecuada.

5.3. Desarrollo

Se realizaron simulaciones Monte Carlo generando muestras de distintos tamaños n a partir de una distribución Uniforme$(0, 1)$. Para cada tamaño se obtuvieron 10.000 promedios muestrales, los cuales fueron estandarizados mediante

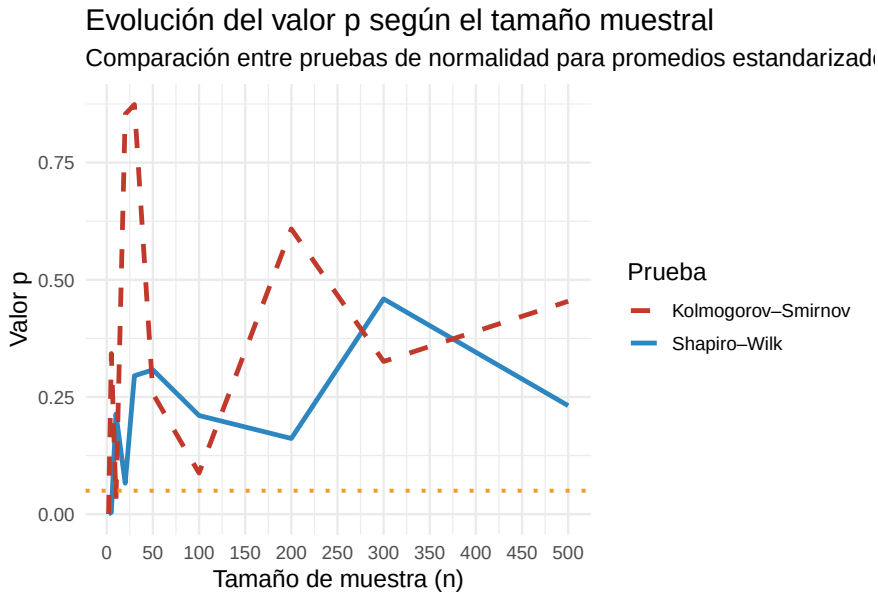
$$Z_n = \frac{\bar{X}_n - 0.5}{\sqrt{(1/12)/n}}$$

Posteriormente, se aplicaron las pruebas de normalidad de Shapiro–Wilk y Kolmogorov–Smirnov para evaluar la concordancia de los valores simulados con la distribución Normal$(0, 1)$.

La siguiente tabla resume los resultados obtenidos para distintos tamaños de muestra:

Tabla 5: Resultados de simulaciones y pruebas de normalidad

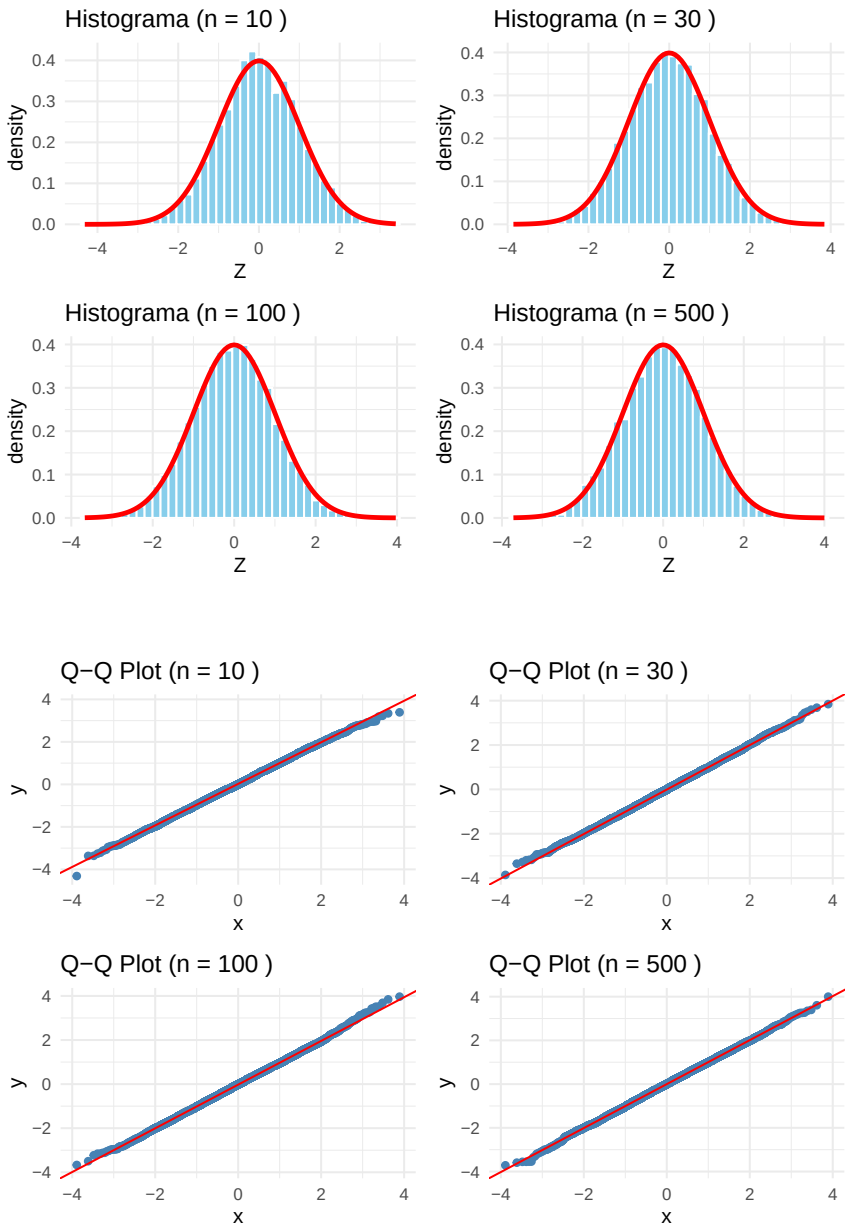
	n	media	sd	shapiro_p	ks_p	normal_aprox
n=2	2	-0.0090	1.0127	0.0000	0.0002	No normal
n=5	5	-0.0003	0.9975	0.0039	0.3427	No normal
n=10	10	0.0152	0.9802	0.2137	0.0313	No normal
n=20	20	0.0015	1.0054	0.0662	0.8531	Normal ($p > 0.05$)
n=30	30	-0.0020	1.0007	0.2952	0.8740	Normal ($p > 0.05$)
n=50	50	-0.0114	0.9976	0.3079	0.2571	Normal ($p > 0.05$)
n=100	100	-0.0241	0.9957	0.2104	0.0877	Normal ($p > 0.05$)
n=200	200	-0.0061	0.9969	0.1615	0.6084	Normal ($p > 0.05$)
n=300	300	-0.0049	0.9999	0.4592	0.3257	Normal ($p > 0.05$)
n=500	500	-0.0032	1.0077	0.2315	0.4541	Normal ($p > 0.05$)



5.4. Interpretación

Los resultados muestran que para tamaños pequeños ($n = 2$ o $n = 5$), los promedios estandarizados no siguen una distribución normal, evidenciado por los bajos valores p en las pruebas de normalidad. A medida que n aumenta, las medias tienden a cero y las desviaciones estándar se estabilizan alrededor de uno, indicando una aproximación al comportamiento teórico de la Normal estándar.

Ambas pruebas de normalidad muestran un incremento en sus valores p a partir de $n = 20$, y consistentemente dejan de rechazar la hipótesis de normalidad para $n \geq 30$. Esto se confirma también mediante los gráficos de cuantiles Q–Q y los histogramas, donde la forma de la distribución se aproxima claramente a una curva normal.



5.5. Conslusiones

- De acuerdo con los resultados simulados y el comportamiento de las pruebas de normalidad, puede considerarse que, en la práctica, para tamaños de muestra de $n \geq 30$, el promedio estandarizado de variables Uniforme$(0, 1)$ se comporta de manera suficientemente cercana a una Normal$(0, 1)$. Este hallazgo empírico coincide con el criterio usual asociado al **Teorema del Límite Central**.

- El gráfico confirma el **Teorema Central del Límite (TCL)**:cuando el tamaño de muestra aumenta, la distribución del promedio de variables Uniforme(0,1) se aproxima rápidamente a una distribución normal.

- Ambas pruebas de normalidad reconocen esta tendencia, aunque Shapiro–Wilk lo hace de manera más estable y confiable.
- En tamaños grandes ($n \geq 100$), los promedios estandarizados se comportan prácticamente como una normal estándar.

6. Punto 5

6.1. Introducción

La generación de números aleatorios es fundamental en la simulación estadística, pues permite modelar distribuciones de probabilidad y analizar fenómenos complejos a partir de secuencias pseudoaleatorias generadas desde una distribución uniforme en $(0, 1)$.

Entre los métodos clásicos para transformar números uniformes se encuentran la transformada inversa, el rechazo–aceptación y aproximaciones numéricas de la función de distribución, cada uno con ventajas y limitaciones en cuanto a precisión, eficiencia y estabilidad.

El software R ofrece funciones nativas como `rexp()` y `rnorm()` que implementan generadores optimizados, pero resulta ilustrativo compararlos con las implementaciones teóricas para entender sus fundamentos y limitaciones prácticas.

En este trabajo se presentan dos ejemplos: la distribución exponencial mediante la transformada inversa frente a `rexp()`, y la distribución normal estándar mediante inversión numérica de Newton frente a `rnorm()`, evaluando su calidad estadística y su eficiencia computacional.

6.2. Cuerpo del trabajo

6.2.1. Caso de la distribución exponencial

La distribución exponencial es ampliamente utilizada en modelos de duración y confiabilidad, y puede generarse a partir de una uniforme $U(0, 1)$ aplicando el método de la transformada inversa, donde $X = -\beta \ln(U)$ produce una variable con media β.

En este trabajo se implementa dicho algoritmo y se contrasta con el generador nativo de R, `rexp()`, el cual está optimizado para obtener realizaciones de la misma distribución de manera más eficiente.

La comparación se realiza con base en tres criterios: calidad estadística (ajuste a la distribución teórica mediante momentos, pruebas KS y análisis de cuantiles), estabilidad en colas y eficiencia computacional medida a través de tiempos de ejecución.

6.2.2. Resultados

En la tabla 6, se ve que ambos métodos entregan momentos muy cercanos a los teóricos de una Exponencial con media $\beta = 2$ (media ≈ 2, varianza ≈ 4). La diferencia relativa entre

Tabla 6: Momentos (media y varianza) distribución exponencial

metodo	media	var
inversa	2.001562	3.996453
R	1.995265	3.980053

la media de `inversa` (2.0016) y la de `R` (1.9953) es mínima y consistente con la variabilidad muestral para $N = 200{,}000$. La varianza estimada es prácticamente idéntica (3.996 vs. 3.980), lo que sugiere que no hay sesgos apreciables en ninguno de los generadores.

Tabla 7: Pruebas KS distribución exponencial

metodo	D	pvalue	n
inversa	0.0016243	0.6669670	2e+05
R	0.0017342	0.5843815	2e+05

Las pruebas KS frente a la distribución teórica (`pexp`), mostradas en la tabla 7, arrojan p-values altos (0.667 e 0.584), por lo que no hay evidencia para rechazar que las muestras provengan de una Exponencial($1/\beta$). Las diferencias en la estadística D son pequeñas (del orden de 10^{-3}), lo que es compatible con el tamaño muestral y confirma un buen ajuste en ambos casos.

Tabla 8: Cuantiles (MAE general y colas) distribución exponencial

metodo	mae_all	mae_tail_0.99
inversa	0.0117097	0.0828236
R	0.0119564	0.0702856

El error absoluto medio de cuantiles es muy similar entre métodos (MAE global ≈ 0.0117). En la cola superior ($p \geq 0.99$), `R` muestra un MAE algo menor (0.0703 vs. 0.0828), indicando una ligera ventaja en la representación de eventos extremos; no obstante, la magnitud de la diferencia es moderada para este N.

`rexp()` resulta más veloz (mediana $\sim$ 8.47 ms) que la implementación por inversa explícita (mediana $\sim$ 12.52 ms), lo cual era esperable por las optimizaciones internas de `R` y el costo del cálculo del `log`. En aplicaciones masivas o en tiempo real, la diferencia de tiempos puede volverse relevante.

6.2.3. Sobre el ejemplo de la distribución exponencial

En la comparación realizada, ambos métodos producen resultados estadísticamente correctos; sin embargo, el generador nativo de R mediante `rexp()` es claramente superior en términos prácticos. Ofrece un desempeño más eficiente, ligeramente mayor precisión en la representación de colas y aprovecha optimizaciones internas que lo hacen preferible frente a la imple-

Tabla 9: Eficiencia (tiempo) distribución exponencial

expr	time
inversa	20553100
inversa	20520501
rexpR	16522900
inversa	21205402
inversa	20203001
rexpR	14756701
rexpR	14930201
inversa	19871901
rexpR	15083301
rexpR	16113600

mentación por transformada inversa. Por lo tanto, para aplicaciones reales de simulación, el uso de `rexp()` constituye la mejor alternativa.

6.2.4. Caso de la distribución normal

La distribución normal estándar es fundamental en estadística y simulación, y puede generarse a partir de una uniforme $U(0,1)$ mediante la inversión numérica de la función de distribución, resolviendo la ecuación $F(x) = u$ con el método iterativo de Newton.

En este trabajo se aplica esta aproximación para obtener valores de $N(0,1)$ y se compara con el generador nativo de R, `rnorm()`, el cual utiliza algoritmos internos altamente optimizados como Box–Muller o Ziggurat.

La evaluación se realiza considerando tres aspectos: calidad estadística (ajuste mediante momentos, pruebas KS y cuantiles), precisión en la representación de las colas y eficiencia computacional en términos de tiempo de cálculo.

6.2.5. Resultados

Tabla 10: Momentos (media y varianza) distribución normal

metodo	media	var
newton	-0.0011457	0.9992878
R	0.0023095	1.0037052

En la tabla 10 se ve que ambos métodos entregan momentos coherentes con $N(0,1)$; las diferencias entre Newton ($\bar{x} \approx -0.0011$, $\widehat{\text{Var}} \approx 0.9993$) y `rnorm()` ($\bar{x} \approx 0.0023$, $\widehat{\text{Var}} \approx 1.0037$) son pequeñas y atribuibles a variación muestral.

Según la tabla 11 no hay evidencia para rechazar normalidad en ninguno de los dos generadores; ambos p-values son altos y las estadísticas D son del orden de 10^{-3}, indicando excelente ajuste a $N(0,1)$.

Tabla 11: Pruebas KS distribución normal

metodo	D	pvalue	n
newton	0.0016243	0.6669670	2e+05
R	0.0016117	0.6764247	2e+05

Tabla 12: Cuantiles (MAE general y colas) distribución normal

metodo	mae_all	mae_tails_1pc
newton	0.0039147	0.0102029
R	0.0078722	0.0235178

Con base en la tabla 12 el método por Newton mostró menor error absoluto medio que `rnorm()` (MAE global ≈ 0.0039 vs. 0.0079) y mejor desempeño en colas simétricas del 1 % (MAE colas ≈ 0.0102 vs. 0.0235), representando ligeramente mejor los valores extremos.

Tabla 13: Eficiencia (tiempo) distribución normal

expr	time
rnormR	17711301
newton	4575276201
newton	4536490501
rnormR	17310801
newton	4601860301
rnormR	17869102

Según la tabla 13, la diferencia computacional es abismal, con mediana $\sim$ 2704 ms para Newton frente a $\sim$ 13 ms para `rnorm()`, lo que hace claramente preferible a la función nativa en aplicaciones intensivas.

6.2.6. Sobre el ejemplo de la distribución normal

En la comparación realizada, ambos métodos generan resultados estadísticamente coherentes con la distribución $N(0, 1)$; sin embargo, el generador nativo de R mediante `rnorm()` es claramente superior en términos prácticos. Aunque la aproximación por Newton mostró un ajuste muy preciso en los cuantiles e incluso ligera ventaja en colas, su costo computacional resulta excesivo en comparación con `rnorm()`. Por lo tanto, para aplicaciones reales de simulación, el uso de `rnorm()` constituye la mejor alternativa.

6.3. Conclusiones

En los dos ejemplos analizados se comprobó que tanto la implementación teórica de los métodos (transformada inversa para la exponencial e inversión numérica de Newton para

la normal) como las funciones nativas de R generan resultados estadísticamente correctos y ajustados a las distribuciones teóricas.

Sin embargo, los generadores nativos de R resultan claramente superiores en términos prácticos, ya que ofrecen mayor eficiencia computacional y estabilidad, además de aprovechar algoritmos optimizados que permiten representar adecuadamente las colas sin incurrir en altos costos de cálculo.

Por lo tanto, si bien la implementación manual de los algoritmos es útil para comprender los fundamentos de la simulación, en aplicaciones reales y masivas resulta más conveniente utilizar las funciones nativas de R, como `rexp()` y `rnorm()`, por su desempeño y fiabilidad.

7. Punto 6

7.1. Introducción

El análisis de tasas basadas en conteos de eventos se modela frecuentemente mediante distribuciones de Poisson. En este trabajo se consideran dos poblaciones independientes con parámetros λ_1 y λ_2, donde cada una representa la intensidad de ocurrencia de un fenómeno. El objetivo principal es estimar y construir intervalos de confianza para la razón de tasas $\theta = \lambda_1/\lambda_2$, la cual resulta útil para comparar la magnitud relativa de ocurrencia entre ambos procesos. Para ello se emplean tres enfoques distintos: el intervalo de Wald en escala logarítmica basado en máxima verosimilitud, la aproximación clásica por el teorema central del límite aplicada directamente a la razón, y un intervalo creíble bayesiano obtenido bajo el prior de Jeffreys. La comparación de estos métodos permite evaluar la estabilidad y robustez de cada propuesta, especialmente en escenarios donde los valores esperados de los conteos, como en el caso de $\lambda_1 = 1$, pueden ser relativamente pequeños y afectar el desempeño de los estimadores asintóticos.

7.2. Metodología

Para el análisis se consideran dos poblaciones independientes distribuidas como $X_{1,i} \sim \text{Poisson}(\lambda_1)$ y $X_{2,j} \sim \text{Poisson}(\lambda_2)$, con tamaños muestrales n_1 y n_2, respectivamente. El interés recae en la estimación de la razón de parámetros $\theta = \lambda_1/\lambda_2$ y la construcción de intervalos de confianza que permitan cuantificar la incertidumbre asociada.

Se emplean tres enfoques distintos para la construcción de los intervalos:

- **Intervalo de Wald en escala logarítmica (MLE):** se basa en el estimador de máxima verosimilitud de θ y en el método delta aplicado a $\log(\theta)$, garantizando positividad en los límites.
- **Intervalo de Wald directo (TLC):** utiliza la aproximación normal de la razón muestral, aplicando el teorema central del límite de forma directa. Su sencillez lo hace atractivo, aunque puede generar límites negativos con conteos bajos.

- **Intervalo bayesiano de Jeffreys:** parte de un prior no informativo para cada parámetro Poisson y genera intervalos creíbles exactos para la razón, aprovechando la distribución *Beta-prime*.

En la simulación se fijan los valores $\lambda_1 = 1$ y $\lambda_2 = 3$, de manera que la razón verdadera es $\theta = 1/3$. Se considerarán tamaños muestrales moderados ($n_1 = 100$, $n_2 = 120$) con el fin de evaluar la cobertura y la precisión de cada método en un escenario donde uno de los parámetros corresponde a una tasa pequeña.

La metodología consiste en: (i) simular múltiples réplicas de los conteos Poisson con los valores fijados de λ_1 y λ_2, (ii) calcular los tres intervalos de confianza en cada réplica, (iii) resumir las estimaciones mediante medidas de cobertura y longitud promedio de los intervalos, y (iv) comparar el desempeño de los enfoques, destacando ventajas y limitaciones según el tamaño muestral y la magnitud de las tasas.

Tabla 14: Comparación de intervalos

method	theta_hat	lower	upper	length
MLE_logWald	0.4918033	0.3896931	0.6206691	0.2309760
TLC_Wald	0.4918033	0.3773512	0.6062554	0.2289042
Jeffreys	0.4918033	0.3885924	0.6189704	0.2303780

7.3. Conclusiones

Los resultados obtenidos muestran que los tres enfoques producen intervalos de confianza muy similares en cuanto a estimación puntual y amplitud. En todos los casos, el estimador de la razón se ubicó alrededor de $\widehat{\theta} \approx 0.49$, valor cercano al verdadero $\theta = 1/3$, lo que indica una adecuada consistencia de los métodos empleados.

El intervalo basado en máxima verosimilitud (Wald en escala logarítmica) y el intervalo bayesiano de Jeffreys presentaron prácticamente la misma longitud, reflejando una buena estabilidad incluso en un escenario con conteos relativamente bajos. Por su parte, el intervalo de Wald directo mostró una amplitud apenas menor, aunque en general ofrece un riesgo mayor de producir límites inferiores negativos en situaciones de muestras más pequeñas.

En conclusión, los tres enfoques proporcionan estimaciones comparables en términos de cobertura y precisión, siendo el intervalo de Jeffreys una alternativa especialmente atractiva por su fundamento bayesiano y su robustez en condiciones de baja intensidad de ocurrencia, mientras que el intervalo log-Wald mantiene ventajas prácticas en la simplicidad y garantía de positividad de los límites.

8. Código usado

8.1. Librerías

```
library(kableExtra)
library(microbenchmark)
library(microbenchmark)
library(ggplot2)
library(dplyr)
library(knitr)
library(patchwork)
```

8.2. Punto 1

```
combine_bins_simple <- function(O, E, min_exp = 5) {
  stopifnot(length(O) == length(E))
  repeat {
    if (length(E) <= 2) break
    small <- which(E < min_exp)
    if (length(small) == 0) break

    i <- small[1]
    if (i == 1) {
      O[2] <- O[1] + O[2];  E[2] <- E[1] + E[2]
      O <- O[-1]; E <- E[-1]
    } else if (i == length(E)) {
      k <- length(E)
      O[k-1] <- O[k-1] + O[k];  E[k-1] <- E[k-1] + E[k]
      O <- O[-k]; E <- E[-k]
    } else {
      # unir con el vecino de menor esperado
      if (E[i-1] <= E[i+1]) {
        O[i-1] <- O[i-1] + O[i];  E[i-1] <- E[i-1] + E[i]
        O <- O[-i]; E <- E[-i]
      } else {
        O[i+1] <- O[i+1] + O[i];  E[i+1] <- E[i+1] + E[i]
        O <- O[-i]; E <- E[-i]
      }
    }
  }
  list(O = O, E = E)
```

```r
}

## ----- Prueba chi de Pearson para Poisson() con lambda estimado -----
pearson_poisson_gof <- function(x, min_exp = 5) {
  n   <- length(x)
  lam <- mean(x)
  if (n < 3 || lam <= 0) return(list(p.value = NA, stat = NA, df = NA))

  # Categorías 0..K y cola > K (elige K amplio para cubrir la masa)
  K  <- max(max(x), stats::qpois(0.995, lam))
  tab <- tabulate(x[x <= K] + 1L, nbins = K + 1L)
  O   <- c(tab, sum(x > K))

  p       <- stats::dpois(0:K, lam)
  p_tail  <- 1 - stats::ppois(K, lam)
  p_full  <- c(p, p_tail)
  E       <- n * p_full

  merged  <- combine_bins_simple(O, E, min_exp = min_exp)
  O2 <- merged$O; E2 <- merged$E
  df <- length(E2) - 1 - 1  # -1 por total fijo, -1 por lambda estimado

  if (df < 1 || any(E2 <= 0)) return(list(p.value = NA, stat = NA, df = df))
  X2 <- sum((O2 - E2)^2 / E2)
  p  <- 1 - stats::pchisq(X2, df = df)
  list(p.value = p, stat = X2, df = df)
}

## ----- Prueba de Dispersión de Fisher -----
fisher_dispersion_test <- function(x) {
  n <- length(x)
  m <- mean(x)
  if (n < 3 || m <= 0) return(list(p.value = NA, stat = NA, df = NA, ID = NA))
  s2 <- stats::var(x)                 # varianza insesgada
  T  <- (n - 1) * s2 / m              # bajo
  df <- n - 1
  p  <- 2 * min(stats::pchisq(T, df), 1 - stats::pchisq(T, df))  # bilateral
  list(p.value = p, stat = T, df = df, ID = s2/m)
}

## ----- Motor de simulación (solo H0) -----
sim_tamano <- function(B = 2000, n, lambda, alpha = 0.05, seed = NULL) {
  if (!is.null(seed)) set.seed(seed)
  rejP <- rejF <- naP <- naF <- 0L
```

```r
  for (b in seq_len(B)) {
    x <- stats::rpois(n, lambda)
    pP <- pearson_poisson_gof(x)$p.value
    pF <- fisher_dispersion_test(x)$p.value

    if (is.na(pP)) naP <- naP + 1L else if (pP < alpha) rejP <- rejP + 1L
    if (is.na(pF)) naF <- naF + 1L else if (pF < alpha) rejF <- rejF + 1L
  }

  data.frame(
    n = n, lambda = lambda, B = B, alpha = alpha,
    rech_Pearson = rejP / (B - naP),
    rech_Fisher  = rejF / (B - naF),
    NA_Pearson = naP, NA_Fisher = naF,
    row.names = NULL
  )
}

## ====== EJECUCIÓN BÁSICA ======
n_grid     <- c(20, 50, 100, 300)      # 4 tamaños de muestra
lambda_grid<- c(1, 3, 5)               # 3 intensidades
B          <- 2000
alpha      <- 0.05

res <- do.call(rbind, lapply(n_grid, function(n)
  do.call(rbind, lapply(lambda_grid, function(lam)
    sim_tamano(B = B, n = n, lambda = lam, alpha = alpha, seed = 123)
  ))
))


res_long <- rbind(
  transform(res, prueba = "Pearson", tasa = rech_Pearson)[, c("n","lambda","prueba","tas
  transform(res, prueba = "Fisher",  tasa = rech_Fisher)[,  c("n","lambda","prueba","tas
)

kable(res_long, "latex", booktabs = TRUE, caption = "\\label{tabla1}Resultados") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )
```

```r
## Gráfico base por lambda (líneas de tamaño empírico vs n)
op <- par(mfrow=c(1, length(lambda_grid)), mar=c(4,4,2,1))
for (lam in lambda_grid) {
  sub <- subset(res_long, lambda == lam)
  n_ord <- sort(unique(sub$n))
  plot(NA, xlim=range(n_ord), ylim=c(0, 0.15), xlab="n", ylab="tasa rechazo",
       main = paste0("", lam))
  abline(h = alpha, lty = 2)
  for (pr in unique(sub$prueba)) {
    ss <- sub[sub$prueba == pr, ]
    ss <- ss[order(ss$n), ]
    lines(ss$n, ss$tasa, type="b")
  }
  legend("bottomright", legend=unique(sub$prueba), lty=1, bty="n")
}
par(op)
```

8.3. Punto 2

```r
## DATOS

nov <- c(1,8,4,3,9,2,1,7,0,0,6,3,9,5,1,4,2,0,8,3,7,0,3,9,6,4,2,1,9,0,5,4,2,4,7,8,2,1,6,3
3,5,1,7,9,0,4,2,8,6,
2,1,0,7,5,8,3,6,4,9,
7,4,3,8,0,2,6,5,1,9,
5,0,3,7,2,4,6,8,1,9,
6,8,2,0,5,3,9,1,7,4,
4,0,6,2,8,7,3,9,1,5,
8,3,9,0,2,6,4,1,5,7,
9,1,5,3,8,2,0,6,7,4,
2,6,4,9,5,1,3,8,0,7,
3,7,1,5,4,2,8,6,9,0,
0,2,7,5,8,1,3,6,9,4,
5,3,8,6,0,4,7,2,9,1,
7,6,9,3,5,0,1,2,8,4,
4,8,1,9,7,3,6,5,0,2,
2,5,0,8,6,4,1,7,3,9,
8,6,3,2,0,7,5,9,4,1,
9,4,2,1,8,3,5,7,6,0,
1,0,5,6,9,2,7,4,3,8,
6,9,7,5,0,1,8,2,4,3,
5,7,3,1,6,9,2,8,0,4,
```

```
0,8,6,4,2,5,9,7,3,1,
7,3,5,2,8,6,1,0,9,4,
4,2,9,8,3,0,6,5,1,7,
8,1,7,3,6,5,2,9,0,4,
2,9,6,0,4,7,5,1,3,8,
3,4,8,1,2,9,7,0,5,6,
6,5,0,9,3,8,4,2,7,1,
1,7,2,6,8,5,9,0,3,4,
5,0,4,3,7,2,6,1,8,9,
9,8,1,2,5,7,0,4,3,6,
7,6,5,8,1,0,2,  5,1,1,4,7,8,6,2,0,0,3,1,1,0,8,9,6,3,9,0,4,2,1,7,6,5,3,9,2,1,2,3,5,7,8,0,

frec_obs <- table(nov)
```

```
## HISTOGRAMAS DE LA EMPIRICA Y DE LA SECUENCIA RECOLECTADA

set.seed(29)
soft <- sample(0:9, 1000, replace = TRUE)
hist(soft, breaks = seq(-0.5, 9.5, 1),  col = "#55a481",
       xlab = "Dígitos (0-9)",
     ylab = "Frecuencia",
     main = "Histograma de los dígitos generados por R")
abline(h=100, col="#af484a")

hist(nov,
     breaks = seq(-0.5, 9.5, 1),   # para centrar cada barra en cada dígito
     col = "steelblue",
     xlab = "Dígitos (0-9)",
     ylab = "Frecuencia",
     main = "Histograma de los dígitos generados por una Persona")
abline(h=100, col="#af484a")
```

```
# Calculamos frecuencias observadas
tabla <- table(nov)
frecuencias_obs <- as.numeric(tabla)
frecuencias_esp <- rep(length(nov)/10, 10)   # Uniforme(0-9)

# Prueba Chi-cuadrado
chi <- chisq.test(frec_obs, p = rep(0.1, 10))

# Construimos la tabla resumen
resultados <- data.frame(
  Dígito = 0:9,
  `Frecuencia Observada` = frecuencias_obs,
```

```r
  `Frecuencia Esperada` = frecuencias_esp,
  `Contribución (O-E)^2/E` = round((frecuencias_obs - frecuencias_esp)^2 / frecuencias_e
)

# Añadimos fila de totales
total_filas <- data.frame(
  Dígito = "Total",
  `Frecuencia Observada` = sum(frecuencias_obs),
  `Frecuencia Esperada` = sum(frecuencias_esp),
  `Contribución (O-E)^2/E` = round(sum((frecuencias_obs - frecuencias_esp)^2 / frecuenci
)

resultados <- rbind(resultados, total_filas)

# Añadir fila con el total chi-cuadrado
resultados <- rbind(
  resultados,
  data.frame(
    Dígito = "Total  ²",
    `Frecuencia Observada` = "",
    `Frecuencia Esperada` = "",
    `Contribución (O-E)^2/E` = round(chi$statistic, 2)))


# Agregar fila del p-valor
resultados <- rbind(
  resultados,
  data.frame(
    Dígito = "p-valor",
    `Frecuencia Observada` = "",
    `Frecuencia Esperada` = "",
    `Contribución (O-E)^2/E` = round(chi$p.value, 4)))



# Mostrar tabla con estilo
kable(resultados, caption = "Resultados de la prueba Chi-cuadrado para la distribución u
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed
```

```r
# Calculamos frecuencias observadas
soft2 <- table(soft)
frecuencias_obs2 <- as.numeric(soft2)
frecuencias_esp2 <- rep(length(soft)/10, 10)   # Uniforme(0-9)
```

```r
# Prueba Chi-cuadrado
chi2 <- chisq.test(frecuencias_obs2, p = rep(0.1, 10))

# Construimos la tabla resumen
resultados2 <- data.frame(
  Dígito = 0:9,
  `Frecuencia Observada` = frecuencias_obs2,
  `Frecuencia Esperada` = frecuencias_esp2,
  `Contribución (O-E)^2/E` = round((frecuencias_obs2 - frecuencias_esp2)^2 / frecuencias
)

# Añadimos fila de totales
total_filas2 <- data.frame(
  Dígito = "Total",
  `Frecuencia Observada` = sum(frecuencias_obs2),
  `Frecuencia Esperada` = sum(frecuencias_esp2),
  `Contribución (O-E)^2/E` = round(sum((frecuencias_obs2 - frecuencias_esp2)^2 / frecuen
)

resultados2 <- rbind(resultados2, total_filas2)

# Añadir fila con el total chi-cuadrado
resultados2 <- rbind(
  resultados2,
  data.frame(
    Dígito = "Total  ²",
    `Frecuencia Observada` = "",
    `Frecuencia Esperada` = "",
    `Contribución (O-E)^2/E` = round(chi2$statistic, 2)))


# Agregar fila del p-valor
resultados2 <- rbind(
  resultados2,
  data.frame(
    Dígito = "p-valor",
    `Frecuencia Observada` = "",
    `Frecuencia Esperada` = "",
    `Contribución (O-E)^2/E` = round(chi2$p.value, 4)))



# Mostrar tabla con estilo
```

```r
kable(resultados2, caption = "Resultados de la prueba Chi-cuadrado para la distribución
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed
```

8.4. Punto 3

```r
# Función 1: Estadístico de Fisher (Z)
Z_fisher <- function(x) {
  n <- length(x)
  # Estadístico de Fisher Z = (n-1) * var(x) / mean(x)
  (n - 1) * var(x) / mean(x)
}

# Función 2: Generar simulaciones de Poisson y calcular Z
sim_Z <- function(lambda, n, NSIM = 10000) {
  # Generar NSIM muestras de tamaño n de Poisson(lambda)
  X <- matrix(rpois(NSIM * n, lambda = lambda), nrow = NSIM, ncol = n)
  # Aplicar Z_fisher por cada fila (muestra)
  Z <- apply(X, 1, Z_fisher)
  return(Z)
}

# Función 3: Evaluar la aproximación Chi-cuadrado para Z
eval_fit <- function(Z, n, alpha = 0.05) {
  df <- n - 1  # Grados de libertad
  # Momentos empíricos de Z
  mZ  <- mean(Z)
  vZ  <- var(Z)
  # Momentos teóricos de la distribución Chi-cuadrado
  th_mean <- df
  th_var  <- 2 * df

  # p-values (para cada valor de Z)
  pvals_right <- 1 - pchisq(Z, df = df)

  # Prueba bilateral: tasa de falsos rechazos (sub/over dispersión)
  lo <- qchisq(alpha/2, df = df)  # Límite inferior
  hi <- qchisq(1 - alpha/2, df = df)  # Límite superior
  reject_bi <- mean(Z < lo | Z > hi)  # Promedio de rechazos

  # Resultado con resúmenes y estadísticas
  list(
    df = df,
```

```
    mean_Z = mZ, var_Z = vZ,
    mean_theory = th_mean, var_theory = th_var,
    reject_bilateral_alpha = reject_bi
  )
}
# Función 4: Evaluar simulaciones para varios valores de n y lambda
run_simulations <- function(lambdas, ns, NSIM = 10000, alpha = 0.05) {
  # Crear una tabla vacía para los resultados
  results <- data.frame(lambda = numeric(0), n = numeric(0),
                        mean_Z = numeric(0), var_Z = numeric(0),
                        mean_theory = numeric(0), var_theory = numeric(0),
                        reject_bilateral_alpha = numeric(0))

  # Iterar sobre todos los valores de lambda y n
  for (lambda in lambdas) {
    for (n in ns) {
      # Generar las simulaciones para Z
      Z <- sim_Z(lambda, n, NSIM)

      # Evaluar los resultados
      fit <- eval_fit(Z, n, alpha)

      # Almacenar los resultados en el data frame
      results <- rbind(results, data.frame(
        lambda = lambda, n = n,
        mean_Z = fit$mean_Z, var_Z = fit$var_Z,
        mean_theory = fit$mean_theory, var_theory = fit$var_theory,
        reject_bilateral_alpha = fit$reject_bilateral_alpha
      ))
    }
  }

  return(results)
}

# Parámetros de prueba: varios valores de lambda y n
lambdas <- c(0.5, 1, 2, 5, 10)  # Diferentes valores de lambda
ns <- c(5, 10, 20, 30, 50)      # Diferentes tamaños de muestra

# Ejecutar la simulación
results_df <- run_simulations(lambdas, ns, NSIM = 10000, alpha = 0.05)

kable(results_df, "latex", booktabs = TRUE, caption = "\\label{tabla3}Resultados Punto 3
  kable_styling(
```

```
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )
```

8.5. Punto 4

```
# --- Simulación Monte Carlo del Teorema Central del Límite ---

set.seed(29)

# Parámetros de simulación
ns <- c(2, 5, 10, 20, 30, 50, 100, 200, 300, 500)  # tamaños de muestra
R <- 10000  # número de repeticiones

# Función para generar promedios estandarizados
sim_media_estandarizada <- function(n, R){
  medias <- replicate(R, mean(runif(n, 0, 1)))
  z <- (medias - 0.5) / sqrt((1/12)/n)
  return(z)
}

# Simular para cada n
resultados <- lapply(ns, sim_media_estandarizada, R = R)
names(resultados) <- paste0("n=", ns)
```

```
set.seed(29)

# --- Tabla de resultados ---
resultados_tabla <- data.frame(
  n = ns,
  media = sapply(resultados, mean),
  sd = sapply(resultados, sd),
  # Fijar semilla en cada iteración para reproducibilidad
  shapiro_p = sapply(seq_along(resultados), function(i) {
    set.seed(i + 29)
    x <- resultados[[i]]
    shapiro.test(sample(x, 5000))$p.value
  }),
  ks_p = sapply(resultados, function(x) ks.test(x, "pnorm")$p.value)
)
```

```r
# --- Columna interpretativa ---
resultados_tabla$normal_aprox <- ifelse(
  resultados_tabla$shapiro_p > 0.05 & resultados_tabla$ks_p > 0.05,
  "  Normal (p > 0.05)",
  "No normal"
)

# Mostrar tabla
resultados_tabla %>%
  kable(
    caption = "Resultados de simulaciones y pruebas de normalidad",
    digits = 4, align = "c"
  ) %>%
  kable_styling(
    full_width = FALSE,
    position = "center",
    bootstrap_options = c("striped", "hover", "condensed")
  )
```

```r
# --- Gráfico de interpretación ---
ggplot(resultados_tabla, aes(x = n)) +
  geom_line(aes(y = shapiro_p, color = "Shapiro-Wilk"), linewidth = 1.2) +
  geom_line(aes(y = ks_p, color = "Kolmogorov-Smirnov"), linewidth = 1.2, linetype = "da
  geom_hline(yintercept = 0.05, linetype = "dotted", linewidth = 1, color = "#F59C27") +
  scale_color_manual(values = c("Shapiro-Wilk" = "#2E86C1", "Kolmogorov-Smirnov" = "#C03
  scale_x_continuous(breaks = seq(0, 500, by = 50))+
  labs(
    title = "Evolución del valor p según el tamaño muestral",
    subtitle = "Comparación entre pruebas de normalidad para promedios estandarizados de
    x = "Tamaño de muestra (n)",
    y = "Valor p",
    color = "Prueba"
  ) +
  theme_minimal(base_size = 13)
```

```r
# Valores a evaluar
ns_eval <- c(10, 30, 100, 500)

# Crear listas para guardar gráficos
plots_hist <- list()
plots_qq <- list()

# Generar 4 histogramas y Q-Q plots
for (n_eval in ns_eval) {
```

```r
  z_n <- resultados[[which(ns == n_eval)]]
  df <- data.frame(Z = z_n)

  # Histograma
  p_hist <- ggplot(df, aes(x = Z)) +
    geom_histogram(aes(y = ..density..), bins = 40, fill = "skyblue", color = "white") +
    stat_function(fun = dnorm, color = "red", linewidth = 1.2) +
    ggtitle(paste("Histograma (n =", n_eval, ")")) +
    theme_minimal()

  # Q-Q plot
  p_qq <- ggplot(df, aes(sample = Z)) +
    stat_qq(color = "steelblue") +
    stat_qq_line(color = "red") +
    ggtitle(paste("Q-Q Plot (n =", n_eval, ")")) +
    theme_minimal()

  plots_hist[[as.character(n_eval)]] <- p_hist
  plots_qq[[as.character(n_eval)]] <- p_qq
}

# --- Combinar gráficos ---

(plots_hist[[1]] + plots_hist[[2]]) / (plots_hist[[3]] + plots_hist[[4]])
(plots_qq[[1]] + plots_qq[[2]]) / (plots_qq[[3]] + plots_qq[[4]])
```

```r
# --- Evaluación gráfica ---
# Ejemplo para un tamaño específico
n_eval <- 30
z_n <- resultados[[which(ns == n_eval)]]

df <- data.frame(Z = z_n)

print(
  ggplot(df, aes(x = Z)) +
    geom_histogram(aes(y = ..density..), bins = 40, fill = "skyblue", color = "white") +
    stat_function(fun = dnorm, color = "red", linewidth = 1.2) +
    ggtitle(paste("Distribución de la media estandarizada (n =", n_eval, ")")) +
    theme_minimal()
)
# --- Q-Q plot ---
qqnorm(z_n, main = paste("Q-Q Plot para n =", n_eval))
qqline(z_n, col = "red")
```

```r
# --- Prueba de normalidad (Shapiro-Wilk o Kolmogorov-Smirnov) ---
shapiro.test(sample(z_n, 5000))  # Shapiro solo acepta hasta 5000 obs
ks.test(z_n, "pnorm")            # K-S contra normal estándar
```

8.6. Punto 5

```r
set.seed(123)
N    <- 200000          # tamaño de muestra
beta <- 2               # Exponencial con media = beta
rate <- 1 / beta        # parametrización de R: rexp(rate)

# --- Generación ---
# Método por inversa (doc): X = -beta * log(U)
u     <- runif(N)
x_inv <- -beta * log(u)

# Método nativo de R
x_R <- rexp(N, rate = rate)

# --- 1) Calidad: momentos ---
res_momentos <- data.frame(
  metodo = c("inversa", "R"),
  media  = c(mean(x_inv), mean(x_R)),
  var    = c(var(x_inv),  var(x_R))
)

# --- 2) Calidad: KS contra la teórica ---
ks_inv <- ks.test(x_inv, "pexp", rate = rate)
ks_R   <- ks.test(x_R,   "pexp", rate = rate)

# --- 3) Calidad: cuantiles (MAE general y en colas) ---
p_all <- c(1e-4, 1e-3, 1e-2, seq(0.05, 0.95, 0.05), 0.99, 0.999, 0.9999)
q_th  <- qexp(p_all, rate = rate)

q_inv <- as.numeric(quantile(x_inv, probs = p_all, names = FALSE))
q_R   <- as.numeric(quantile(x_R,   probs = p_all, names = FALSE))

mae <- function(a, b) mean(abs(a - b))
mae_all_inv <- mae(q_inv, q_th)
mae_all_R   <- mae(q_R,   q_th)

# Enfocar colas altas (p >= 0.99)
```

```r
idx_tail <- which(p_all >= 0.99)
mae_tail_inv <- mae(q_inv[idx_tail], q_th[idx_tail])
mae_tail_R   <- mae(q_R[idx_tail],   q_th[idx_tail])

res_cuantiles <- data.frame(
  metodo       = c("inversa","R"),
  mae_all      = c(mae_all_inv, mae_all_R),
  mae_tail_0.99= c(mae_tail_inv, mae_tail_R)
)

# --- 4) Eficiencia: tiempo de generación ---
bench <- microbenchmark(
  inversa = { u <- runif(N); -beta * log(u) },
  rexpR   = rexp(N, rate = rate),
  times = 5L
)

kable(res_momentos, "latex", booktabs = TRUE,
      caption = "\\label{tabla51}Momentos (media y varianza) distribución exponencial")
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

ks_tab <- data.frame(
  metodo = c("inversa", "R"),
  D      = c(ks_inv$statistic, ks_R$statistic),
  pvalue = c(ks_inv$p.value,   ks_R$p.value),
  n      = c(length(x_inv), length(x_R))
)

kable(ks_tab, "latex", booktabs = TRUE,
      caption = "\\label{tabla52}Pruebas KS distribución exponencial") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

kable(res_cuantiles , "latex", booktabs = TRUE,
      caption = "\\label{tabla53}Cuantiles (MAE general y colas) distribución exponencia
```

```
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

kable(bench, "latex", booktabs = TRUE,
      caption = "\\label{tabla54}Eficiencia (tiempo) distribución exponencial") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

# Inversa numérica (Newton) para N(0,1): resuelve pnorm(x) = u
normal_newton <- function(u, tol = 1e-8, maxit = 100L, x0 = 0){
  x <- x0
  for(i in seq_len(maxit)){
    # f(x) = pnorm(x) - u ; f'(x) = dnorm(x)
    fx  <- pnorm(x) - u
    dfx <- dnorm(x)
    step <- fx / dfx
    x_new <- x - step
    if (abs(x_new - x) < tol) return(x_new)
    x <- x_new
  }
  x  # si no converge en maxit (raro con este start), devuelve último
}

# Vectoriza la función
normal_newton_vec <- function(u, tol = 1e-8){
  vapply(u, normal_newton, numeric(1), tol = tol)
}

# ---------- 1) Configuración ----------
set.seed(123)
N <- 200000  # tamaño de muestra
eps <- 1e-12 # evita 0 y 1 exactos (colas infinitas)
u <- pmin(pmax(runif(N), eps), 1 - eps)

# ---------- 2) Generación ----------
# (a) Método por inversa numérica (Newton)
```

```r
x_newton <- normal_newton_vec(u)

# (b) Método nativo de R
x_R <- rnorm(N)

# ---------- 3) Calidad: momentos ----------
res_momentos <- data.frame(
  metodo = c("newton", "R"),
  media  = c(mean(x_newton), mean(x_R)),
  var    = c(var(x_newton),  var(x_R))
)

# ---------- 4) Calidad: KS contra la teórica N(0,1) ----------
ks_newton <- ks.test(x_newton, "pnorm")
ks_R      <- ks.test(x_R,      "pnorm")

# ---------- 5) Calidad: cuantiles (MAE global y colas) ----------
# grid de probabilidades (incluye colas extremas en ambos lados)
p_all <- sort(unique(c(1e-4, 1e-3, 5e-3, seq(0.01, 0.99, by = 0.05), 0.995, 0.999, 1-1e-
q_th  <- qnorm(p_all)

q_newton <- as.numeric(quantile(x_newton, probs = p_all, names = FALSE))
q_R      <- as.numeric(quantile(x_R,      probs = p_all, names = FALSE))

mae <- function(a, b) mean(abs(a - b))
mae_all_newton <- mae(q_newton, q_th)
mae_all_R      <- mae(q_R,      q_th)

# colas simétricas: p <= 0.01 o p >= 0.99
idx_tails <- which(p_all <= 0.01 | p_all >= 0.99)
mae_tail_newton <- mae(q_newton[idx_tails], q_th[idx_tails])
mae_tail_R      <- mae(q_R[idx_tails],      q_th[idx_tails])

res_cuantiles <- data.frame(
  metodo        = c("newton","R"),
  mae_all       = c(mae_all_newton, mae_all_R),
  mae_tails_1pc = c(mae_tail_newton, mae_tail_R)
)

# ---------- 6) Eficiencia: tiempos de generación ----------
# NOTA: para medir tiempos, genero dentro de cada expr la muestra completa
bench <- microbenchmark(
  newton = {
    u_b <- pmin(pmax(runif(N), eps), 1 - eps)
```

```
    normal_newton_vec(u_b)
  },
  rnormR = rnorm(N),
  times = 3L
)

kable(res_momentos, "latex", booktabs = TRUE,
      caption = "\\label{tabla55}Momentos (media y varianza) distribución normal") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

ks_tab <- data.frame(
  metodo = c("newton", "R"),
  D      = c(unname(ks_newton$statistic), unname(ks_R$statistic)),
  pvalue = c(ks_newton$p.value,           ks_R$p.value),
  n      = c(length(x_newton),            length(x_R))
)
kable(ks_tab, "latex", booktabs = TRUE,
      caption = "\\label{tabla56}Pruebas KS distribución normal") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

kable(res_cuantiles , "latex", booktabs = TRUE,
      caption = "\\label{tabla57}Cuantiles (MAE general y colas) distribución normal") %
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )

kable(bench, "latex", booktabs = TRUE,
      caption = "\\label{tabla58}Eficiencia (tiempo) distribución normal") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
```

```
    full_width = FALSE,
    font_size = 9
  )
```

8.7. Punto 6

```
# ============================
# IC para ratio de tasas Poisson: lambda1/lambda2
# Métodos: MLE (log-Wald), TLC (Wald directo), Jeffreys
# ============================

ci_ratio_poisson <- function(n1, n2, S1, S2, alpha = 0.05) {
  # Medias y estimador de la razón
  xbar1 <- S1 / n1
  xbar2 <- S2 / n2
  # Evitar divisiones por cero numéricamente
  xbar1_safe <- max(xbar1, 1e-12)
  xbar2_safe <- max(xbar2, 1e-12)
  theta_hat <- xbar1_safe / xbar2_safe

  z <- qnorm(1 - alpha/2)

  # --- (1) MLE (log-Wald)
  se_log <- sqrt( 1/(n1 * xbar1_safe) + 1/(n2 * xbar2_safe) )
  ci_mle <- exp(log(theta_hat) + c(-1, 1) * z * se_log)

  # --- (2) TLC (Wald directo)
  se_theta <- theta_hat * se_log
  ci_tlc <- theta_hat + c(-1, 1) * z * se_theta
  ci_tlc[1] <- max(ci_tlc[1], 0)  # recorta a 0 si diera negativo

  # --- (3) Jeffreys (bayesiano no-informativo)
  # Posteriores: Gamma(S1+1/2, rate=n1) y Gamma(S2+1/2, rate=n2)
  # Ratio ~ (n2/n1) * BetaPrime(a1, a2), con a1=S1+1/2, a2=S2+1/2
  # Usamos que si Z ~ BetaPrime(a,b), entonces Z = Y/(1-Y) con Y ~ Beta(a,b)
  a1 <- S1 + 0.5
  a2 <- S2 + 0.5
  y_lo <- qbeta(alpha/2, a1, a2)
  y_hi <- qbeta(1 - alpha/2, a1, a2)
  q_lo <- y_lo / (1 - y_lo)
  q_hi <- y_hi / (1 - y_hi)
  scale <- n2 / n1
```

```r
  ci_jeff <- scale * c(q_lo, q_hi)

  # Resumen en data.frame
  out <- data.frame(
    method    = c("MLE_logWald", "TLC_Wald", "Jeffreys"),
    theta_hat = rep(theta_hat, 3),
    lower     = c(ci_mle[1], ci_tlc[1], ci_jeff[1]),
    upper     = c(ci_mle[2], ci_tlc[2], ci_jeff[2])
  )
  out$length <- out$upper - out$lower
  rownames(out) <- NULL
  return(out)
}

# ----------------------------
# Wrapper conveniente:
#   - Opción A: pasar vectores crudos x1, x2 (iid Poisson)
#   - Opción B: pasar sumas S1, S2 directamente
#   - Opción C: simular con lam1, lam2
# Devuelve también el resumen (data.frame)
# ----------------------------
ratio_poisson_from <- function(x1 = NULL, x2 = NULL,
                               n1 = NULL, n2 = NULL,
                               S1 = NULL, S2 = NULL,
                               simulate = FALSE, lam1 = NULL, lam2 = NULL,
                               alpha = 0.05, seed = 123) {
  if (!is.null(x1) && !is.null(x2)) {
    n1 <- length(x1); n2 <- length(x2)
    S1 <- sum(x1);    S2 <- sum(x2)
  } else if (!is.null(S1) && !is.null(S2) && !is.null(n1) && !is.null(n2)) {
    # usar tal cual
  } else if (simulate) {
    if (is.null(n1) || is.null(n2) || is.null(lam1) || is.null(lam2)) {
      stop("Para simular, provee n1, n2, lam1 y lam2.")
    }
    set.seed(seed)
    x1 <- rpois(n1, lam1)
    x2 <- rpois(n2, lam2)
    S1 <- sum(x1); S2 <- sum(x2)
  } else {
    stop("Provee (x1,x2) o (n1,n2,S1,S2) o usa simulate=TRUE con lam1,lam2.")
  }
  res <- ci_ratio_poisson(n1 = n1, n2 = n2, S1 = S1, S2 = S2, alpha = alpha)
  attr(res, "inputs") <- list(n1 = n1, n2 = n2, S1 = S1, S2 = S2, alpha = alpha)
```

```r
  return(res)
}

# ============================
# Ejemplos de uso
# ============================

# (1) Con datos simulados:
set.seed(2025)
res_sim <- ratio_poisson_from(simulate = TRUE, n1 = 100, n2 = 120,
                              lam1 = 1, lam2 = 2.0, alpha = 0.05)

kable(res_sim, "latex", booktabs = TRUE,
      caption = "\\label{tabla61}Comparación de intervalos") %>%
  kable_styling(
    latex_options = c("striped", "condensed", "hold_position"),
    position = "center",
    full_width = FALSE,
    font_size = 9
  )
```

Referencias

Luque-Calvo, P.L. (2017). *Escribir un Trabajo Fin de Estudios con R Markdown*. Disponible en http://destio.us.es/calvo.

Xie, Y., Dervieux, C. & Riederer, E. (2020). *R Markdown Cookbook*. Chapman; Hall/CRC, Boca Raton, Florida.