19/21 = 9

Examen parcial

Maximiliano Alvarez, Daniel Camarena

1. Probabilidad

Puedes usar simulación o encontrar la respuesta de manera analítica (incluye procedimiento).

Una caja contiene 10 pares de aretes, si se seleccionan 8 aretes.

- **¿Cuál es el epacio de resultados?**

El espacio de resultados son todas las posibles maneras en las que podemos extraer 8 aretes de la caja que contiene los 10 pares de aretes.

```
# maneras de elegir los aretes de la caja
choose(20, 8)
```

```
## [1] 125970
```

- **¿Cuál es la probabilidad de que no se seleccione ningún par?**

```
# esta funcion samplea los aretes y encuentra los pares en la muestra elegida
# si la muestra elegida contiene al menos un par regresa falso, de lo contrario regresa
 verdadero

incomplete_pairs <- function(){
  earings<-c(1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,0,0)
  earings_sample<-sample(earings,8)
  earings_sample <- as.data.frame(earings_sample)
  earings_pairs <- earings_sample %>% group_by(earings_sample) %>%
                                  dplyr::summarise(n=n())
  pares<-earings_pairs$n
  if (2 %in% pares){
    result <- FALSE
  }else{
    result <- TRUE
  }
  return(result)
}

n <-10000
x <- rerun(n,incomplete_pairs())
prob <- Reduce("+", x)/n
print(paste("La probabilidad de no seleccionar ningun par es",prob * 100,"%"))
```

```
## [1] "La probabilidad de no seleccionar ningun par es 9.52 %"
```

- **¿Cuál es la probabilidad de que se seleccione exactamente un par completo?**

```
# esta funcion regresa 1 si la seleccion contiene unicamente un par de aretes, de lo con
trario regresa 0

one_pair <- function(){
  earings<-c(1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,0,0)
  earings_sample<-sample(earings,8)
  earings_sample <- as.data.frame(earings_sample)
  earings_pairs <- earings_sample %>% group_by(earings_sample) %>%
                                  dplyr::summarise(n=n())
  pares<-earings_pairs$n
  if (length(pares[pares==2])==1){
    result <- TRUE
  }else{
    result <- FALSE
  }
  return(result)
}

n <-10000
x <- rerun(10000, one_pair())
prob <- Reduce("+", x)/n
print(paste("La probabilidad de seleccionar exactamente un par es",prob * 100,"%"))
```

```
## [1] "La probabilidad de seleccionar exactamente un par es 42.87 %"
```

Bien

b. A y B juegan una serie de juegos. En cada juego A gana con probabilidad 0.4 y B con probabilidad 0.6 (independiente de lo ocurrido en los otros juegos). Paran de jugar cuando el número total de juegos ganados de un jugador es dos juegos más que el total de juegos ganados por el otro jugador:

- **¿Cuál es el espacio de resultados?**

El espacio de resultados esta definido por el número de juegos ganados por A y el numero de juegos ganados por B.

descripicón

- **Encuentra la probabilidad de que se jueguen 4 juegos en total.**

```
# generador de variable aleatoria que define quien gana un juego
win_game<-function(pa, pb){
  u <- runif(1)
  if(u<pa){
    result <- "A"
  }else{
    if(u>=pa & u<(pa+pb)){
      result <-"B"
    }
  }
  return(result)
}

# simula juegos, lleva contadores de qué jugador gana y lleva un contador de juegos juga
dos que es lo que regresa
simulate_games<-function(){
  pa<-0.4
  pb<-0.6
  countA<-0
  countB<-0
  countGames<-0
  winner <- win_game(pa,pb)
  countGames<-countGames+1
  if(winner=="A"){
    countA<-countA + 1
  }else{
    countB<-countB + 1
  }
  while(!abs(countA-countB)==2){
    winner <- win_game(pa,pb)
    countGames<-countGames+1
    if(winner=="A"){
      countA<-countA + 1
    }else{
     countB<-countB + 1
    }
  }
  return(countGames)
}

n<-100000
x<-rerun(n,simulate_games()) %>% flatten_dbl()
prob <- (length(x[x==4])/n)*100

print(paste("La probabilidad de que se juegen exactamente 4 juegos es",prob,"%"))
```

```
## [1] "La probabilidad de que se juegen exactamente 4 juegos es 25.046 %"
```

Bien

- **Encuentra la probabilidad de que A gane la serie.**

```
# generador de variable aleatoria que define quien gana un juego
win_game<-function(pa, pb){
  u <- runif(1)
  if(u<pa){
    result <- "A"
  }else{
    if(u>=pa & u<(pa+pb)){
      result <-"B"
    }
  }
  return(result)
}

# simula juegos, lleva contadores de qué jugador gana y regresa al ganador de cada serie
get_winner<-function(){
  pa<-0.4
  pb<-0.6
  countA<-0
  countB<-0
  winner <- win_game(pa,pb)
  if(winner=="A"){
    countA<-countA + 1
  }else{
    countB<-countB + 1
  }
  while(!abs(countA-countB)==2){
    winner <- win_game(pa,pb)
    if(winner=="A"){
      countA<-countA + 1
    }else{
     countB<-countB + 1
    }
  }
  if(countA > countB){
    winner <- "A"
  }else{
    winner <- "B"
  }
  return(winner)
}

n<-100000
x <- rerun(n,get_winner()) %>% flatten_chr()
prob <- (length(x[x=="A"])/n)*100
print(paste("La probabilidad de que gane A es", prob, "%"))
```

```
## [1] "La probabilidad de que gane A es 30.812 %"
```

Bien

2. Bootstrap

La base de datos *amis* (publicada por G. Amis) contiene información de velocidades de coches en millas por hora, las mediciones se realizaron en carreteras de Cambridgeshire, y para cada carretera se realizan mediciones en dos sitios. Mas aún, las mediciones se realizaron en dos ocasiones, antes y después de que se instalara una señal de alerta (de dismunición de velocidad), esta señal se erigió únicamente en uno de los dos sitios de cada carretera.

La cantidad de interés es el cambio medio relativo de velocidad en el cuantil 0.85. Se eligió esta cantidad porque el objetivo de la señal de alerta es disminuir la velocidad de los conductores más veloces.

Variables:

- *speed*: velocidad de los autos en mph,
- *period*: periodo en que se hicieron las mediciones, 1 indica antes de la señal, 2 cuando ya había señal,
- *warning*:
- *pair*: carretera en que se hizo la medición.

a. **¿Las observaciones conforman una muestra aleatoria?** Explica tu respuesta y en caso de ser negativa explica la estructura de los datos.

```
# cargamos los datos
amis <- read_csv("datos/amis.csv", col_types = "ddddd", col_names = c("obs", "speed", "p
eriod", "warning", "pair"), skip = 1)

# vista de los datos
kable(head(amis))
```

obs	speed	period	warning	pair
1	26	1	1	1
2	26	1	1	1
3	26	1	1	1
4	26	1	1	1
5	27	1	1	1
6	28	1	1	1

```
# observamos la composición de la muestra
kable(amis %>%
        group_by(warning, period) %>%
        dplyr::count())
```

warning	period	n
1	1	1100
1	2	1100
2	1	1100

warning	period	n
2	2	1100

```
# incorporando la carretera como una variable adicional de agrupación
kable(amis %>%
        group_by(pair, warning, period) %>%
        dplyr::count() %>%
        head(n = 10))
```

pair	warning	period	n
1	1	1	100
1	1	2	100
1	2	1	100
1	2	2	100
2	1	1	100
2	1	2	100
2	2	1	100
2	2	2	100
5	1	1	100
5	1	2	100

Con lo anterior podemos observar que cada tipo de observación, tomando en cuenta las variables *period* y *warning* para cada una de las distintas carreteras (*pair*), tiene el mismo número de observaciones dentro de la muestra.

Cuando sea necesario hacer muestreo, será necesario considerar estas distinciones, para mantener la 'equiprobabilidad' de los datos de la muestra, seleccionando tomando en consideración las variables discriminantes.

Bien

b. El estadístico de interés se puede escribir como

$$\eta = \frac{1}{m}\Sigma[(\eta_{a1} - \eta_{b1}) - (\eta_{a0} - \eta_{b0})]$$

donde η_{a1}, η_{b1} corresponden a los cuartiles 0.85 de la distribución de velocidad en los sitios en los que se colocó la señal de alerta, (a corresponde a las mediciones antes de la señal y b después) y η_{a0}, η_{b0} son los correspondientes para los sitios sin alerta, m denota el número de carreteras. **Calcula el estimador *plug-in* de η.**

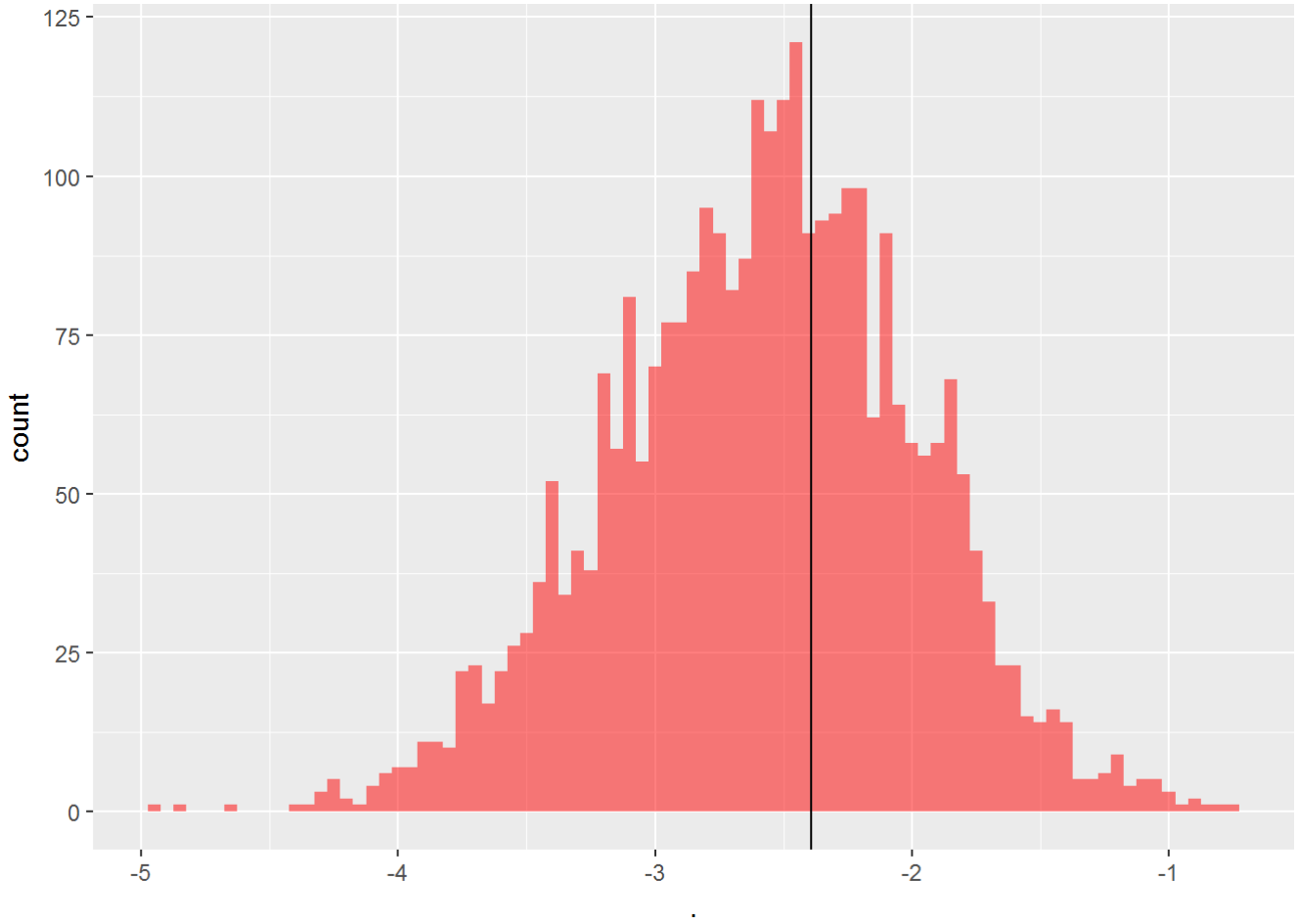
```
# manipulamos la base para poder obtener los cuantiles
cuartiles <- amis %>%
  group_by(pair, warning, period) %>%
  dplyr::summarise(q_85 = quantile(speed, probs = 0.85))

# tamaño de cada una de las muestras
n <- 100

# calculamos el número de carreteras
m <- length(unique(amis$pair))

# calculamos el estimador plug-in
eta_plugin <- cuartiles %>%
  mutate(q_85 = if_else(period == 2, -q_85, q_85)) %>%
  group_by(pair, warning) %>%
  dplyr::summarise(q_85 = sum(q_85)) %>%
  mutate(q_85 = if_else(warning == 1, -q_85, q_85)) %>%
  dplyr::summarise(q_85 = sum(q_85)) %>%
  dplyr::summarise(eta = sum(q_85) / m) %>%
  as.double()
```

Estimación?

c. **Genera $B = 3000$ replicaciones bootstrap de η y realiza un histograma.** Pistas:

1. considera tu respuesta de a) para tomar las muestras bootstrap,
2. hay muchas maneras de resolver esta pregunta, sin embargo, algunas funciones que te pueden resultar útiles son ldply y rdply (de plyr), do.call, group_by, mutate y summarise (de dplyr), spread (tidyr). También puedes usar los paquetes boot ó bootstrap si lo deseas.

```
# generamos la muestras y calculamos los cuantiles
eta_boot <- function(){
  muestra <- amis %>%
  group_by(pair, warning, period) %>%
  sample_n(., size = n, replace = TRUE) %>%
  dplyr::summarise(q_85 = quantile(speed, probs = 0.85))

  # calculamos el estimador eta
  eta <- muestra %>%
  mutate(q_85 = if_else(period == 2, -q_85, q_85)) %>%
  group_by(pair, warning) %>%
  dplyr::summarise(q_85 = sum(q_85)) %>%
  mutate(q_85 = if_else(warning == 1, -q_85, q_85)) %>%
  dplyr::summarise(q_85 = sum(q_85)) %>%
  dplyr::summarise(eta = sum(q_85) / m) %>%
  as.double()
}

# definimos el número de iteraciones
b <- 3000

# generamos las iteraciones necesarias de eta
eta <- rerun(b, eta_boot()) %>%
  flatten_dbl()

# creamos el histograma de los valores de eta
eta %>%
  as.data.frame(.) %>%
  ggplot(aes(.)) +
  geom_histogram(binwidth = .05, fill = "red", alpha = .5) +
  geom_vline(xintercept = eta_plugin, color = "black")
```



d. **Genera intervalos de confianza usando la aproximación normal y percentiles.** Comparalos y en caso de encontrar diferencias explica a que se deben.

```
# calculamos el intervalo de confianza normal
low_n <- eta_plugin - 1.96 * sd(eta)
high_n <- eta_plugin + 1.96 * sd(eta)
print(paste("Intervalo normal: [", round(low_n, 3), ", ", round(high_n, 3), "]"), sep =
"")
```

```
## [1] "Intervalo normal: [ -3.549 ,  -1.233 ]"
```

```
# calculamos el intervalo de confianza por percentiles
low_p <- quantile(eta, prob = 0.025)
high_p <- quantile(eta, prob = 0.975)
print(paste("Intervalo percentil: [", round(low_p, 3), ", ", round(high_p, 3), "]"), sep
 = "")
```

```
## [1] "Intervalo percentil: [ -3.764 ,  -1.468 ]"
```

Podemos observar algunas diferencias entre ambos intervalos de confianza. Esto se debe a que, como podemos apreciar en la gráfica anterior, la distribución está ligeramente sesgada a la Bien

3. Manipulación de datos y bootstrap

La bioequivalencia es un término utilizado en farmacocinética para evaluar la equivalencia terapéutica entre dos formulaciones de un medicamento que contiene el mismo principio activo o fármaco. Cuando una farmacéutica cambia una fórmula o desarrolla una nueva presentación de un medicamento ya disponible al público, requiere aprobación de la FDA para poder distribuirlo; para lograr esta aprobación debe demostrar que el nuevo medicamento es bioequivalente al medicamento ya aprobado, es así que se diseñan ensayos clínicos donde se compara la respuesta de los participantes al recibir las distintas formulaciones del medicamento.

En este ejercicio analizarás la bioequivalencia de una nueva tableta de Cimetidine de 800 mg en relación a tabletas aprovadas de 400 mg. En el ensayo clínico participaron 24 voluntarios saludables, cada uno se asignó de manera aleatoria para recibir la formulación de 800 mg (formulación a prueba) o dos tabletas de 400 mg (formulación referencia). Se recolectaron muestras de sangre antes de la dosis y durante las siguientes 24 horas.

- Lee los datos *cimetidine_raw*, las variables son:
- *formulation* indica si la observación corresponde a formulación de refernecia o de prueba,
- *subj* corresponde al sujeto,
- *seq* toma dos valores 1 indica que el sujeto se eavluó tomando la formulación de tratamiento primero y después la de referencia, 2 indica el caso contrario,
- *prd* indica el periodo (1 o 2) y
- las variables *HRXX* indican la medición (concentración de cimeditine en mCG/ml) para la hora XX (HR0 corresponde a la hora cero, HR05 a media hora, HR10 a una hora,…, HR240 a 24 horas).

Desafortunadamente estos datos crudos están truncados y para algunos sujetos (1, 3, 5, 8, 9, 12, 16, 17, 19, 22, 23) no tenemos las mediciones del tratamiento de referencia, sin embargo, podemos usar la información disponible para entender como se calculan las métricas. Para la parte final usaremos una tabla de datos agregados que sí está completa.

```
cimeditine <- read_delim("datos/cimeditine_raw.txt", delim = "\t", col_names = c("formul
ation","subj","seq","prd","HR0","HR05","HR10",
"HR15","HR20","HR30","HR40","HR60","HR80","HR100","HR120","HR180","HR240"), skip = 1)
```

- **¿Cumplen los principios de los datos limpios?** En caso de que no los cumplan limpialos y explica que fallaba. Imprime las primeras líneas de los datos limpios (si ya estaban limpios entonces los datos originales).

No cumplen los principios de los datos limpios, para que esto suceda necesitamos que cada columna en el dataset sea una variable y cada renglón sea una observación. La forma en la que son presentados estos datos no es así; debemos transformar algunas columnas en renglones (por ejemplo, la concentración por hora).

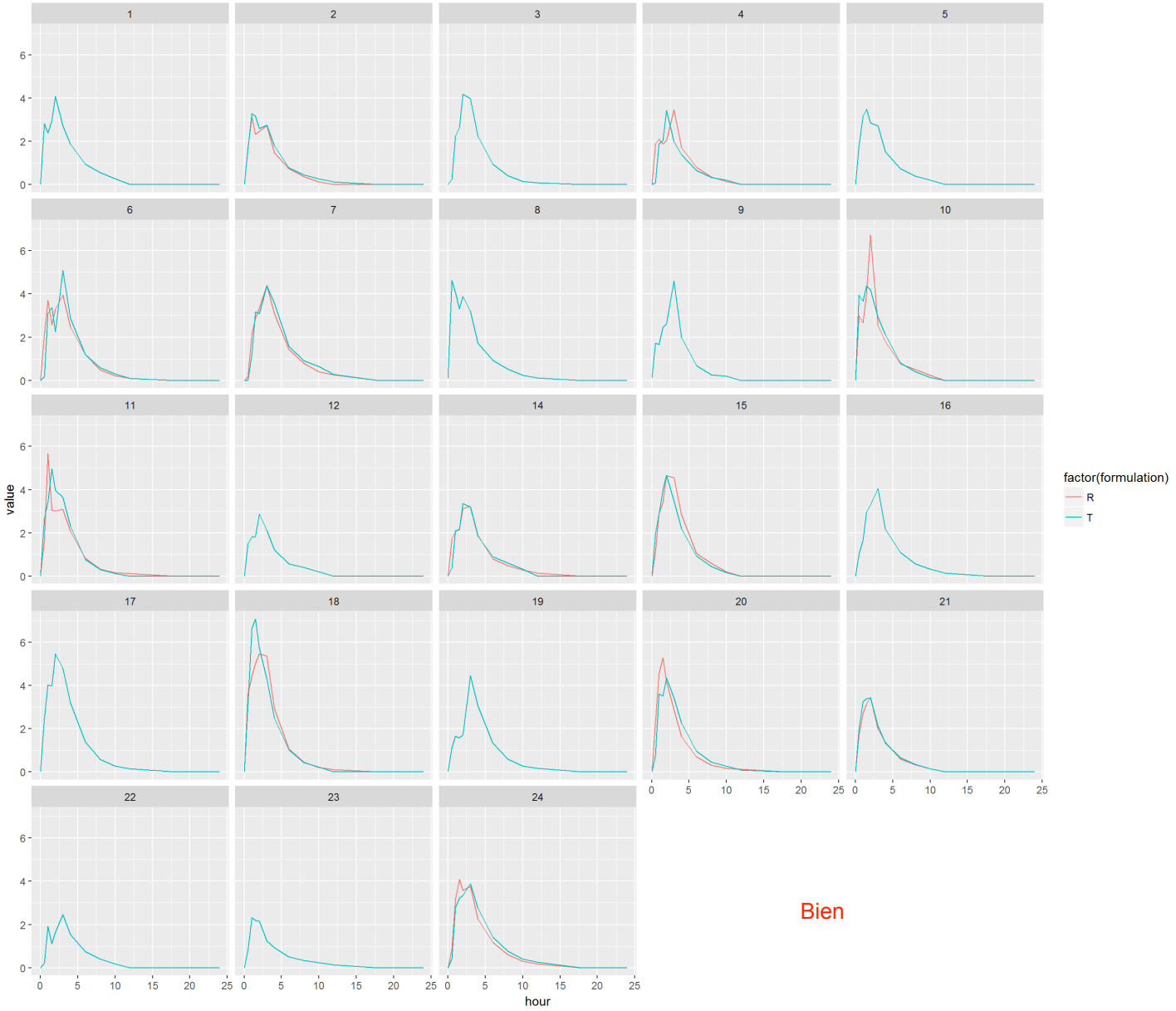
```
# limpiamos los datos
cimeditine_tidy <- cimeditine %>% gather(hour,value,5:17)
cimeditine_tidy$hour <-as.numeric(str_extract(cimeditine_tidy$hour, "[0-9]+"))/10
kable(head(cimeditine_tidy))
```

formulation	subj	seq	prd	hour	value
T	1	1	2	0	0.00
T	3	1	2	0	0.00

formulation	subj	seq	prd	hour	value
T	5	1	2	0	0.00
T	8	1	2	0	0.10
T	9	1	2	0	0.11
T	12	1	2	0	0.00

Grafica la concentración del medicamento por hora. Debes graficar en el eje horizontal la hora, en el eje vertical la concentración para cada persona, bajo cada tratamiento. Un ejemplo de lo que debes hacer es esta gráfica, con la diferencia que la curva de Wikipedia es el promedio sobre todos los individuos y tu graficarás una para cada uno.

```
# graficamos los datos, usamos facet_wrap para poner una grafica por cada sujeto
cimeditine_tidy %>%
  ggplot(aes(x = hour, y = value)) +
  geom_line(aes(color = factor(formulation))) +
  facet_wrap(~subj)
```



Bien

Podemos ver que en la mayoria de los casos el tratamiento y la referencia son similaries.

Para comprobar bioequivalencia la FDA solicita que el medicamento de prueba tenga un comportamiento similar al del medicamento de referencia. Para ello se compara la máxima concentración de medicamento (CMAX), el tiempo que tarda el individuo en alcanzar las concentración máxima en sangre (TMAX), y el área bajo la curva de concentración contra tiempo (AUC, que acabas de graficar). En particular para que la FDA concluya bioequivalencia se requiere que para cada medición (CMAX, TMAX y AUC) un intervalo de 90% de confianza para el cociente μT/μR de la media del tratamiento de prueba μT y la media de referencia μR esté contenido en el intervalo (80%, 125%). En este ejercicio usarás bootstrap para calcular un intervalo de confianza para el cociente de las medias de AUC.

- **Calcula el AUC para cada individuo en cada tratamiento disponible**, usa la fórmula de área trapezoidal:

```
# vamos a dividir los set en dos: Test (t) y referencia (r)

# vamos a usar lag para que sea más fácil
auc_1_t <- cimeditine_tidy  %>%
  filter(formulation == 'T') %>%
  select(formulation,subj,hour,value) %>%
  arrange(hour) %>%
  group_by(subj) %>%
  mutate(auc_value =lag(value,n=1,default=0, order_by=subj),
         auc_hour=lag(hour,n=1,default=0, order_by=subj),
         auc=(value+auc_value)*(hour-auc_hour)/2)

auc_2_t <- auc_1_t %>%
  na.omit() %>%
  select(formulation,subj,auc) %>%
  group_by(formulation,subj) %>%
  dplyr::summarise(auc=sum(auc))

auc_1_r <- cimeditine_tidy  %>%
  filter(formulation == 'R') %>%
  select(formulation,subj,hour,value) %>%
  arrange(hour) %>%
  group_by(subj) %>%
  mutate(auc_value =lag(value,n=1,default=0, order_by=subj),
         auc_hour=lag(hour,n=1,default=0, order_by=subj),
         auc=(value+auc_value)*(hour-auc_hour)/2)

auc_2_r <- auc_1_r %>% na.omit() %>%
  select(formulation,subj,auc) %>%
  group_by(formulation,subj) %>%
  dplyr::summarise(auc=sum(auc))

#unimos los dos sets
auc <- union(auc_2_t,auc_2_r)

kable(auc)
```

formulation	subj	auc
T	7	21.6950
T	10	17.9200
T	3	16.4500
T	5	14.0725
R	18	24.2850
R	21	12.5525
T	22	10.3100
R	15	19.7400

formulation	subj	auc
T	2	15.0350
R	14	15.1350
T	18	24.1000
T	12	11.0400
T	15	17.8200
R	2	12.8800
T	4	11.1300
T	6	18.9825
T	17	24.0150
R	11	16.7850
T	21	13.3350
R	7	20.5475
T	16	17.1950
R	10	18.1150
R	4	13.1225
T	11	18.0450
R	6	18.5400
T	19	17.5625
R	20	17.4650
T	20	17.8000
T	8	17.3550
T	9	13.4800
T	24	19.9375
T	14	14.4825
T	1	16.2400
T	23	9.6700
R	24	18.6875

Bien

- En estos últimos dos incisos usa los datos *cimeditine_boot.csv*, para el i-ésimo individuo subj tienes mediciones de AUC bajo tratamiento y referencia, denotemos a este par xi=(aucTi,aucRi), suponiendo que las xi se obuvieron de manera aleatoria de una distribución bivariada P, entonces la cantidad poblacional de interés es el parámetro θ=μT/μR. **Calcula el estimador plug-in de θ**.

```
# leemos los datos
cimeditine_boot <- read_csv("datos/cimeditine_boot.csv")

# los ponemos de manera tal que sea mas facil trabajarlos
cimeditine_boot <- cimeditine_boot %>% select(Subject,Formulation,AUC) %>%
  spread(Formulation, AUC)

n <- length(unique(cimeditine_boot$Subject))
mt <- mean(cimeditine_boot$T)
mr <- mean(cimeditine_boot$R)

theta_plugin <- mt / mr

print(paste("El estimador plug-in de theta es ",theta_plugin))
```

```
## [1] "El estimador plug-in de theta es  0.991585439151227"
```

Bien

- Usa bootstrap para generar un intervalo del 90% de confianza para θ, ¿la nueva tableta cumple el criterio de bioequivalencia de la FDA?

```
# funcion que calcula las thetas usando bootstrap
theta_boot<-function(){
  n <- length(unique(cimeditine_boot$Subject))
  test <- sample(cimeditine_boot$T,size=n, replace=TRUE)
  reference <- sample(cimeditine_boot$R,size=n, replace = TRUE)
  theta <- mean(test)/mean(reference)
  return(theta)
}

thetas_boot <- rerun(1000, theta_boot()) %>%
  flatten_dbl()

# intervalo de confianza usando metodo de percentiles
lci <- round(quantile(thetas_boot,prob=0.05),2)
hci <- round(quantile(thetas_boot,prob=0.95),2)
print(paste("[",lci,",",hci,"]", "y nuestro estimador plugin es",theta_plugin, "entonces
 vemos que sí es cubierto por el intervalo de confianza del 90% (percentiles) y por lo t
anto cumple con el criterio de bioequivalencia de la FDA"))
```

```
## [1] "[ 0.88 , 1.1 ] y nuestro estimador plugin es 0.991585439151227 entonces vemos qu
e sí es cubierto por el intervalo de confianza del 90% (percentiles) y por lo tanto cump
le con el criterio de bioequivalencia de la FDA"
```

```
# intervalo de confianza usando aproximación normal
li_norm <- round(theta_plugin - 1.645 * sd(thetas_boot),2)
ls_norm <- round(theta_plugin + 1.645 * sd(thetas_boot),2)
print(paste("[",li_norm,",",ls_norm,"]", "y nuestro estimador plugin es",theta_plugin,
"entonces vemos que si es cubierto por el intervalo de confianza del 90% (normal) y por
 lo tanto cumple con el criterio de bioequivalencia de la FDA"))
```

```
## [1] "[ 0.88 , 1.1 ] y nuestro estimador plugin es 0.991585439151227 entonces vemos qu
e si es cubierto por el intervalo de confianza del 90% (normal) y por lo tanto cumple co
n el criterio de bioequivalencia de la FDA"
```

4. Variables aleatorias y simulación

Recuerda que una variable aleatoria X tiene una distribución geométrica con parámetro p si

$$p_X(i) = P(X = i) = pq^{i-1}, i = 1, 2, \ldots, q = 1 - p$$

Bien

a. Recuerda la distribución geométrica **¿cuál es a relación entre la variable aleatoria binomial negativa y la geométrica?**

La distribución geométrica es un caso especial de la distribución binomial negativa, cuando $r = 1$.

Aún más general, si $y_1, \ldots, y_r$ son variables aleatorias independientes e idénticamente distribuidas con parámetro p, la suma $Z = \Sigma_{i=1}^{r} Y_i$ se distribuye binomial negativa con parámetros r, p.

b. **Utiliza el procedimiento descrito para generar observaciones de una variable aleatoria con distribución geométrica y la relación entre la geométrica y la binomial negativa para generar simulaciones de una variable aleatoria con distribución binomial negativa (parámetro p = 0.4, r = 20).** Utiliza la semilla 221285 (en R usa set.seed) y reporta las primeras 10 simulaciones obtenidas.

```
# definimos la función para generar geométricas a partir de una uniforme
rand_geom <- function(p){
  x <- as.integer(log(runif(1))/log(1-p))+1
  return(x)
}

# definimos la binomial negativa en términos de una geométrica
rand_bn <- function(r, p){
  x <- rerun(r, rand_geom(p)) %>%
    flatten_dbl() %>%
    sum()
  return(x)
}

# usamos la relación entre ambas variables para generar una variable aleatoria binomial
 negativa
p <- 0.4
r <- 20
n <- 10
set.seed(221285)

bin_neg <- rerun(n, rand_bn(r, p)) %>%
  flatten_dbl()

bin_neg
```

```
##  [1] 44 57 48 39 45 50 50 39 56 42
```

Bien

c. **Verifica la relación**

$$p_{j+1} = \frac{j(1-p)}{j+1-r} pj$$

y úsala para generar un nuevo algoritmo de simulación, vuelve a definir la semilla y reporta las primeras 10 simulaciones.

Para lo anterior, partimos de la definición de p_{j+1}

$$p_{j+1} = \frac{(j+1-1)!}{(j+1-r)!(r-1)!} p^r (1-p)^{j+1-r}$$

y, usando la propiedad de la función factorial $n! = n(n-1)!, n > 0$ obtenemos

$$p_{j+1} = \frac{j(j-1)!}{(j+1-r)(j-r)!(r-1)!} p^r (1-p)(1-p)^{j-r}$$

y lo anterior, podemos reacomodarlo y reagruparlo para obtener

$$p_{j+1} = \frac{j(1-p)}{(j+1-r)} \frac{(j-1)!}{(j-r)!(r-1)!} p^r (1-p)^{j-r} = \frac{j(1-p)}{j+1-r} p_j$$

con lo que queda verificada la relación.

```
# definimos la relación que obtuvimos previamente
rand_bn_rec <- function(r, p){
  i <- r
  u <- runif(1)
  pr <- p ^ r
  Fx <- pr
  while(u > Fx){
    pr <- ((i * (1 - p)) / (i + 1 - r)) * pr
    Fx <- Fx + pr
    i <- i + 1
  }
  x <- i
}

# usamos la relación recursiva para generar las variables
p <- 0.4
r <- 20
n <- 10
set.seed(221285)

bin_neg_rec <- rerun(n, rand_bn_rec(r, p)) %>%
  flatten_dbl()

bin_neg_rec
```

```
##  [1] 67 45 54 41 55 57 46 72 44 55
```

Bien

d. **Realiza 10,000 simulaciones usando cada uno de los algoritmos y compara el tiempo de ejecución.**

```
# definimos las variables relevantes
n <- 10000
p <- 0.4
r <- 20

# ejecutamos el primer algoritmo
system.time({rerun(n, rand_bn(r, p))})
```

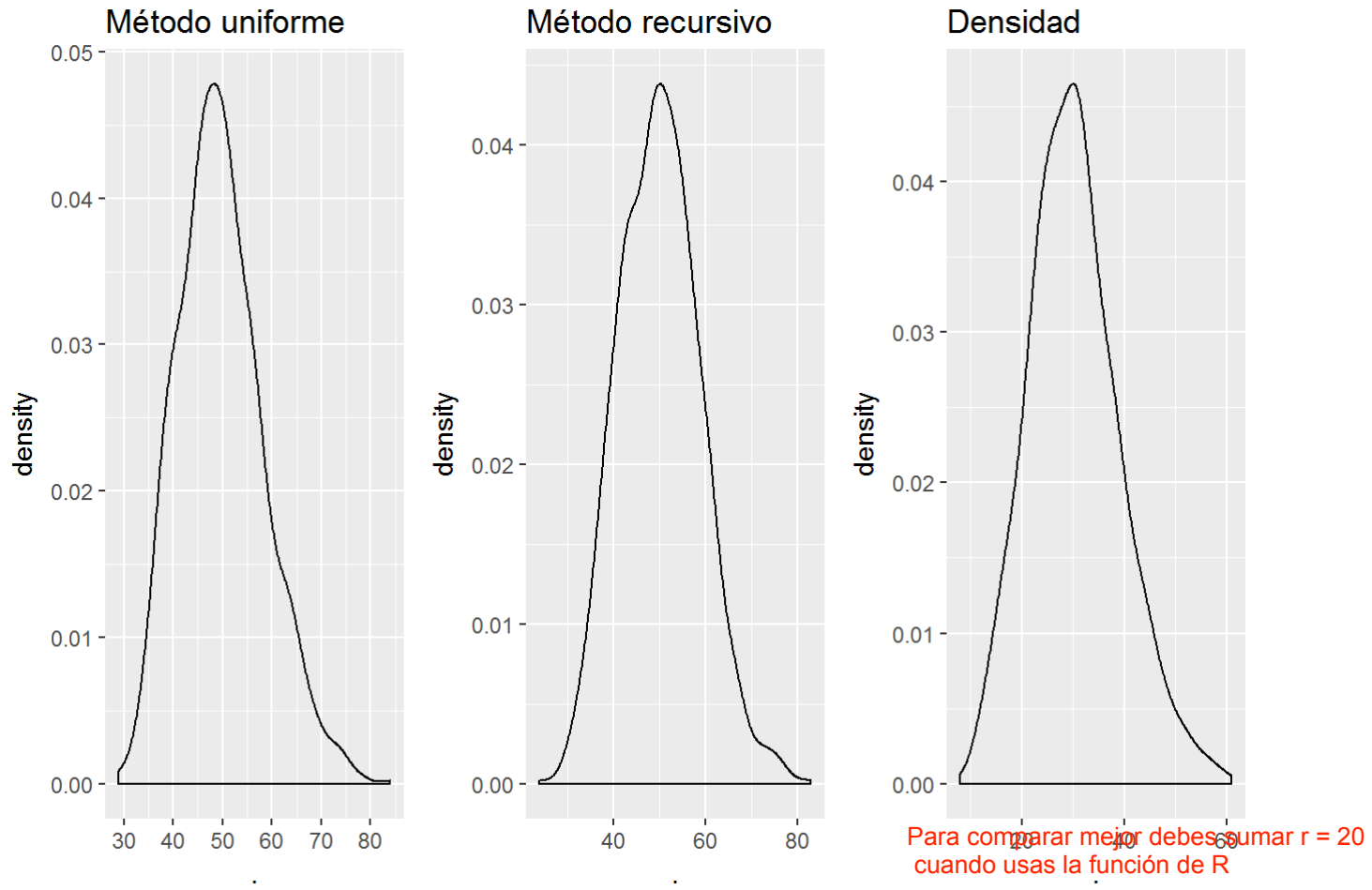
```
##    user  system elapsed
##  182.35    0.03  187.31
```

```
# ejecutamos el segundo algoritmo
system.time({rerun(n, rand_bn_rec(r, p))})
```

```
##    user  system elapsed
##    8.01    0.01    8.03
```

e. **Genera un histograma para cada algoritmo (usa 1000 simulaciones) y compáralo con la distribución construida usando la función de R dnbinom.**

```
n <- 1000
p <- 0.4
r <- 20

set.seed(221285)

# histograma del primer algoritmo
grafica_1 <- rerun(n, rand_bn(r, p)) %>%
  flatten_dbl() %>%
  as_data_frame() %>%
  ggplot(aes(.)) +
  geom_density() +
  ggtitle("Método uniforme")

# histograma del segundo algoritmo
grafica_2 <- rerun(n, rand_bn_rec(r, p)) %>%
  flatten_dbl() %>%
  as_data_frame() %>%
  ggplot(aes(.)) +
  geom_density() +
  ggtitle("Método recursivo")

# histograma de la distribución usando dnbinom
grafica_dens <- rnbinom(n, r, p) %>%
  as_data_frame() %>%
  ggplot(aes(.)) +
  geom_density() +
  ggtitle("Densidad")

grid.arrange(grafica_1, grafica_2, grafica_dens, ncol = 3)
```



5. Cobertura de intervalos de confianza

En este problema realizarás un ejercicio de simulación para comparar intervalos de confianza. Utiliza la función rpois (y la semilla 261285) para simular muestras de tamaño 20 de una distribución Poisson con parámetro $\lambda = 2.5$, el estadístico de interés es $\theta = exp(-2\lambda)$.

1. Genera una muestra aleatoria de tamaño 10 una distribución Poisson con $\lambda = 2.5$.
2. Genera 6000 muestras bootstrap y calcula intervalos de confianza del 95% para $\hat{\theta}$ usando 1) el método normal, 2) percentiles y 3) BC_a.
3. Revisa si el intervalo de confianza contiene el verdadero valor del parámetro, en caso de que no lo contenga registra si falló por la izquierda o falló por la derecha.

a. **Repite el proceso descrito 500 veces y llena la siguiente tabla:**

```
set.seed(261285)
lambda <- 2.5
theta_puntual <- exp(-2*lambda)
muestra <- rpois(10, lambda)
m <- 500

# la funcion  del estimador
estimador_theta <- function(x)(exp(-2*mean(x)))

# funcion que recibe una muestra y estima con una muestra de bootstrap
sample_theta <- function(x) {
  sample(x, replace=TRUE) %>% estimador_theta()
}

n <- 10

# intervalos de confianza usando la funcion vista en clase
calcula_intervalos <- function(n = 10){
    x <- rpois(n, lambda)
    theta_est<- estimador_theta(x)
    #theta_b <- rerun(m, sample_theta(x)) %>% map_dbl(~exp(-2*mean(.)))
    theta_b <- rerun(m, sample_theta(x)) %>% flatten_dbl()

     bca <- bcanon(x, nboot = m, theta = estimador_theta, alpha = c(0.05/2, 1-
00.5/2))$confpoints[,2]  #       intervalos BC_a
    intervalos <- data_frame(metodo = c("normal", "percent", "BC_a"),
        izq = c(theta_est - 1.96 * sd(theta_b), quantile(theta_b, probs = 0.05/2),
bca[1]),
        der = c(theta_est + 1.96 * sd(theta_b), quantile(theta_b, probs = 1-0.05/2),
bca[2])
    )
}

print(paste("El parámetro verdadero es", theta_puntual, "Los intervalos calculados con 6
000 replicaciones bootstrap son:"))
```

se necesitan más simulaciones, el ejercicio pone 6000

```
## [1] "El parámetro verdadero es 0.00673794699908547 Los intervalos calculados con 6000
replicaciones bootstrap son:"
```

```
sims_intervalos_1 <- rerun(1, calcula_intervalos())

kable(sims_intervalos_1)
```

metodo	izq	der
normal	-0.0166949	0.0277280
percent	0.0007466	0.0407622
BC_a	0.0004097	0.0082297

```
print(paste("El parámetro verdadero es", theta_puntual, "Los intervalos calculados con 6
000 replicaciones bootstrap son:"))
```

```
## [1] "El parámetro verdadero es 0.00673794699908547 Los intervalos calculados con 6000
replicaciones bootstrap son:"
```

```
print("Metodo Normal contiene el valor del parámetro")
```

```
## [1] "Metodo Normal contiene el valor del parámetro"
```

```
print("Metodo percentiles contiene el valor del parámetro")
```

```
## [1] "Metodo percentiles contiene el valor del parámetro"
```

```
print("Metodo BCA contiene el valor del parámetro")
```

```
## [1] "Metodo BCA contiene el valor del parámetro"
```

```
sims_intervalos_10 <- rerun(500, calcula_intervalos())

intervalos_ <- sims_intervalos_10 %>% bind_rows() %>%
  group_by(metodo) %>%
  dplyr::summarise("% falla izq" = 100 * mean(izq > theta_puntual),
            "% falla der" = 100 * mean(der < theta_puntual),
            "cobertura (simulaciones)" = 100 * (1-mean(izq>theta_puntual) - mean(der<the
ta_puntual)))

kable(intervalos_)
```

metodo	% falla izq	% falla der	cobertura (simulaciones)
BC_a	3.4	32	64.6
normal	0.4	6	93.6
percent	6.0	3	91.0

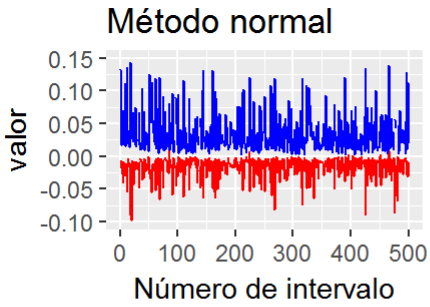
La columna cobertura es una estimación de la cobertura del intervalo basada en las simulaciones, para calcularla simplemente escribe el porcentaje de los intervalos que incluyeron el verdadero valor del parámetro. Recuerda usar la semilla.

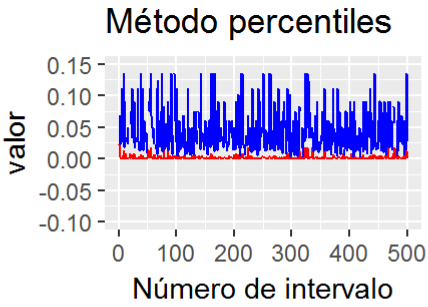
b. **Realiza una gráfica de páneles**, en cada panel mostrarás los resultados de uno de los métodos (normal, percentiles y BC_a), el eje x corresponderá al número de intervalo de confianza (1,…,500) y en el vertical graficarás los límites de los intervalos, es decir graficarás 2 líneas (usa geom_line) una corresponderá a los límites inferiores de los intervalos, y otra a los superiores.

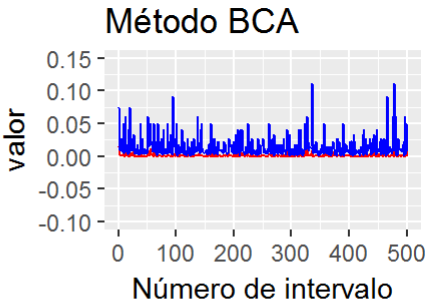
Nota: Un ejemplo en donde la cantidad $P(X=0)^2 = e^{-2\lambda}$ es de interés es como sigue: las llamadas telefónicas a un conmutador se modelan con un proceso Poisson y λ es el número promedio de llamadas por minuto, entonce $e^{-2\lambda}$ es la probabilidad de que no se reciban llamadas en 2 minutos.

```
tabla <- sims_intervalos_10 %>% bind_rows()
normales <- subset(tabla, tabla$metodo == "normal")
normales$numero <- seq.int(nrow(normales))
percentiles <- subset(tabla, tabla$metodo == "percent")
percentiles$numero <- seq.int(nrow(percentiles))
bca <- subset(tabla, tabla$metodo == "BC_a")
bca$numero <- seq.int(nrow(bca))

graf_normal <- ggplot(normales, aes(numero,izq)) +
  geom_line(aes(y = izq),colour="red") +
  geom_line(aes(y=der),color="blue") +
  scale_y_continuous(limits=c(-.1,.15)) +
  ggtitle( "Método normal") +
  ylab("valor") +
  xlab("Número de intervalo")

graf_percent<- ggplot(percentiles, aes(numero,izq)) +
  geom_line(aes(y = izq),colour="red") +
  geom_line(aes(y=der),color="blue") +
  scale_y_continuous(limits=c(-.1,.15)) +
  ggtitle( "Método percentiles") +
  ylab("valor") +
  xlab("Número de intervalo")

graf_bca <- ggplot(bca, aes(numero,izq)) +
  geom_line(aes(y = izq),colour="red") +
  geom_line(aes(y=der),color="blue") +
  scale_y_continuous(limits=c(-.1,.15))+
  ggtitle( "Método BCA") + ylab("valor") +
  xlab("Número de intervalo")

grid.arrange(graf_normal,graf_percent,graf_bca, ncol=3, heights=c(6,6,6),
widths=c(.8,.8,.8))
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```


Método normal
valor
0.15
0.10
0.05
0.00
-0.05
-0.10
0
100
200
300
400
500
Número de intervalo


Método percentiles
valor
0.15
0.10
0.05
0.00
-0.05
-0.10
0
100
200
300
400
500
Número de intervalo


Método BCA
valor
0.15
0.10
0.05
0.00
-0.05
-0.10
0
100
200
300
400
500
Número de intervalo

Bien