

Lecturas Importantes

1. Fundamentos de programación de R, de Vincent Zoonekynd
http://zoonek2.free.fr/UNIX/48_R/02.html.

Preguntas

1. (a) Escribe una función, para calcular la siguiente integral:

$$f : x \rightarrow \frac{H(x)}{\int_x^\infty H(t)dt} \quad \text{donde} \quad H(t) = 1 - \Phi_{\mu,\sigma}(t) = \int_t^\infty \phi_{\mu,\sigma}(t)dt,$$

donde $\phi_{\mu,\sigma}$ es la función densidad de la distribución $N(\mu, \sigma^2)$.

- (b) Calcula las probabilidades binomiales, usando una función recursiva. Sea $X \sim \text{binomial}(n, p)$ y sea $f(x, k, p) = \mathbb{P}(X = x) = \binom{k}{x} p^x (1-p)^{k-x}$, para $0 \leq x \leq k$ y $0 \leq p \leq 1$. Es fácil mostrar que:

$$f(0, k, p) = (1-p)^k;$$
$$f(x, k, p) = \frac{(k-x+1)p}{x(1-p)} f(x-1, k, p) \quad \text{para } x \geq 1.$$

Usa, este resultado para escribir una función recursiva `binom.pmf(x, k, p)`, que retorna $f(x, k, p)$. Puedes verificar que esta función trabaja, comparando, los resultados con la función `dbinom(x, k, p)`.

- (c) La función `runif(n)` simula n variables aleatorias idénticamente distribuidas a `Uniforme(0,1)`. Así `1 + runif(n)`², simula n copias idénticamente distribuidas de $Y = 1 + X^2$. Estima y dibuja el pdf de Y usando una muestra aleatoria simulada. Experimenta con el ancho de los intervalos para obtener un gráfico de buen aspecto: debe ser razonablemente detallada, pero también razonablemente suave. ¿Qué tamaño debe tener su muestra para obtener una aproximación decente?.
2. (a) Considere el siguiente escenario:

Pedro es un estudiante de universidad y todos sus amigos creen que el es valiente y que muchas de las chicas lo encuentran atractivo. Pedro hace poco ha ganado las regionales del ICPC y por ello su confianza es alta; en una fiesta encuentra a una chica que considera atractiva y le pregunta si quiere bailar con el. Ella parece ser más inteligente que Pedro y le dice:

Hay M hombres y W mujeres en esta fiesta además de nosotros. Tienes C caramelos en tu mano y los distribuirás aleatoriamente entre la gente (solo si es posible). Ahora si le pides a los hombres de esta fiesta sus caramelos, llamemos a esta cantidad CC y si CC se puede distribuir exacta e igualmente en 2 grupos, bailaré contigo.

Pedro quiere saber la probabilidad de poder bailar con la chica. Determina un algoritmo para resolver el problema conociendo los valores de M, W, C . Por ejemplo si $M = 1$ $W = 2$ $C = 2$, entonces la respuesta es 0.5555556.

- (b) En Rusia, por el día de la Bandera, el dueño de una tienda decidió decorar la ventada de su tienda con franjas de colores blanco, azul y rojo. El quiere cumplir con dos consignas:
 - No se pueden colocar franjas del mismo color de forma adyacente.

- Una franja de color azul siempre se debe colocar entre una blanca y una roja o entre una roja y una blanca (Ejemplo: BAR - RAB).

Determina un algoritmo para expresar la cantidad de formas en las que el dueño puede decorar su tienda conociendo sólo la cantidad de franjas que caben en la ventana (N). $N \geq 1$. Por ejemplo si $N = 3$, entonces tu respuesta es 4 ya que se pueden formar 4 banderas; BRB, RAB, BAR, RBR.

- (a) Escribe un programa para leer, cada uno de los archivos `edad.txt` y `dientes.txt` y luego escribe una lista amalgamada del archivo `edad_dientes.txt`, de la siguiente forma:

ID	Edad	Num_dientes
1	18	28
2	19	27
3	17	32
.	.	.
.	.	.
.	.	.

- (b) La función `order(x)` retorna una permutación de $1:\text{length}(x)$, con los orden de los elementos de x . Por ejemplo:

```
> x <- c(1.1, 0.7, 0.8, 1.4)
> (y <- order(x))

[1] 2 3 1 4
```

```
> x[y]

[1] 0.7 0.8 1.1 1.4
```

Usando `order`, modifica tu programa del ejercicio anterior para que el archivo de salida sea ordenado por su segunda columna.

- (c) ¿Cuáles son las ventajas y desventajas de un enfoque muy flexible (versus un enfoque menos flexible) para la regresión o la clasificación? Bajo qué circunstancias, podría preferirse un enfoque más flexible a un enfoque menos flexible?
 - (d) Describe las diferencias entre un enfoque de aprendizaje estadístico paramétrico y otro no paramétrico. ¿Cuáles son las ventajas de un enfoque paramétrico para la regresión o clasificación (en oposición a un enfoque no paramétrico)? ¿Cuáles son sus desventajas?
- (a) Ejecuta este código en tu mente y predice como será la salida. A continuación, ejecute el código en R y compruebe sus predicciones.

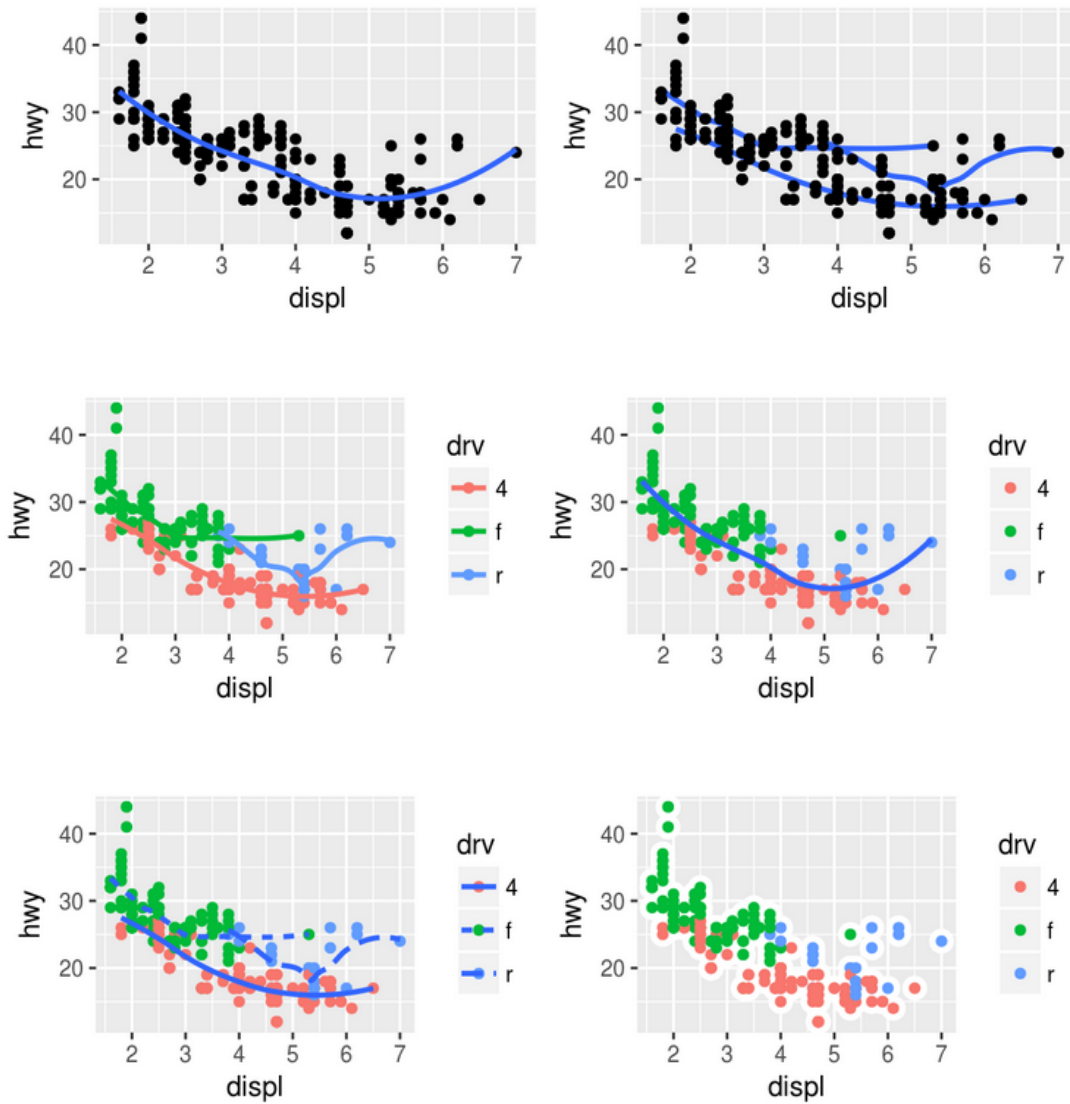
```
> ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
+   geom_point() +
+   geom_smooth(se = FALSE)
```

- (b) ¿Qué hace `show.legend = FALSE`? ¿Qué sucede si lo retiras? ¿Qué hace el argumento `se` a `geom_smooth()`?
- (c) ¿Cuál es el geometría predeterminada asociada con `stat_summary()`?
- (d) ¿Qué hace `geom_col()`? ¿Cómo es de diferente de `geom_bar()`?
- (e) ¿Qué variables calcula `stat_smooth()`? ¿Qué parámetros controlan su comportamiento?
- (f) ¿Los siguientes gráficos son diferentes? Explica tus respuesta.

```
> ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
+   geom_point() +
+   geom_smooth()
>
```

```
> ggplot() +
+   geom_point(data = mpg, mapping = aes(x = displ, y = hwy)) +
+   geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy))
```

(g) Escribe el código de R, para generar los siguientes gráficos:



5. Responde y escribe código fuente si el ejercicio lo amerita, a las siguientes preguntas:

- (a) Los datos de iris corresponden a las medidas en centímetros de las variables `length`, `width sepal` y `length`, `width petal` es decir ancho y altura de los pétalos y sépalos respectivamente, de 50 flores cada una de tres especies de iris. Hay cuatro variables numéricas correspondientes al sépalo y pétalo y un factor `Species`. Muestra una tabla de medias para `Species` (donde las medias se deben calcularse por separado para cada una de las tres `Species`).
- (b) The National Institute of Standards and Technology tiene una página web que lista los primeros 500 dígitos del número irracional π . Podemos leer esos dígitos en R, por medio del script:

```
> pidigits =
+ read.table("http://www.itl.nist.gov/div898/strd/univ/data/PiDigits.dat",
+ skip=60)
```

Usa la función `table` para construir una tabla de frecuencias de los dígitos del 1 al 9.

- (c) La función `dim()` devuelve las dimensiones (un vector que tiene el número de filas entonces el número de columnas) de matrices y data frames. Utilice esta función para encontrar el número de filas de los data frames de `tinting`, `possum` y `possumsites` del paquete `DAAG`.
- (d) Supongamos que se está interesado en mostrar 3 miembros de la familia de curvas beta, donde la densidad con parámetros a y b , denotados por $Beta(a, b)$ es dado por

$$f(y) = \frac{1}{B(a,b)} y^{a-1} (1-y)^{b-1}, \quad 0 < y < 1.$$

Se puede dibujar una sola densidad beta, con parámetros $a = 5$ y $b = 2$, usando la función `curve`:

```
> curve(dbeta(x, 5, 2), from=0, to=1)
```

- Usa tres aplicaciones de la función `curve` para mostrar las densidades $Beta(2, 6)$, $Beta(4, 4)$ y $Beta(6, 2)$ en un sólo gráfico.
- Usa el siguiente comando de R, para colocar un título al gráfico de las ecuaciones de densidad beta

```
> title(expression(f(y)==frac(1,B(a,b))*y^{a-1}*(1-y)^{b-1}))
```

- Usa la función `text`, para etiquetar cada una de las curvas betas con sus correspondientes valores de los parámetros a y b .
- Redibuja el gráfico usando diferentes colores o tipos de líneas para las tres curvas de densidad.

- (e) ¿Qué es lo hace esta función estadística?. ¿Qué mejoras crees que se debería hacer?.

```
> bc <- function(lambda) {
+   if (lambda == 0) {
+     function(x) log(x)
+   } else {
+     function(x) (x ^ lambda - 1) / lambda
+   }
+ }
```

- (f) Crea una función que crea funciones que calculan el i -ésimo momento central de un vector numérico. Puedes verificar, usando el siguiente código:

```
> m1 <- moment(1)
> m2 <- moment(2)
>
> x <- runif(100)
> stopifnot(all.equal(m1(x), 0))
> stopifnot(all.equal(m2(x), var(x) * 99 / 100))
```

- (g) La idea de la integración numérica es simple: encontrar el área bajo una curva, aproximando esa curva por simples componentes. Dos aproximaciones se implementan a continuación:

```
> puntomedio <- function(f, a, b) {  
+   (b - a) * f((a + b) / 2)  
+ }  
>  
> trapecio <- function(f, a, b) {  
+   (b - a) / 2 * (f(a) + f(b))  
+ }
```

Para hacer más precisos estas aproximaciones, dividiremos el rango en pedazos más pequeños e integraremos cada pieza usando una de las reglas simples. Esto se llama integración compuesta. Implementaremos dos nuevas funciones, a partir de esta definición:

```
> puntomedio_compuesto <- function(f, a, b, n = 10) {  
+   puntos <- seq(a, b, length = n + 1)  
+   h <- (b - a) / n  
  
+   area <- 0  
+   for (i in seq_len(n)) {  
+     area <- area + h * f((puntos[i] + puntos[i + 1]) / 2)  
+   }  
+   area  
+ }  
>  
> trapecio_compuesto <- function(f, a, b, n = 10) {  
+   puntos <- seq(a, b, length = n + 1)  
+   h <- (b - a) / n  
  
+   area <- 0  
+   for (i in seq_len(n)) {  
+     area <- area + h / 2 * (f(puntos[i]) + f(puntos[i + 1]))  
+   }  
+   area  
+ }
```

Te darás cuenta de que hay una gran cantidad de duplicación entre `puntomedio_compuesto()` y `trapecio_compuesto()`. Aparte de la regla interna utilizada para integrar en un rango, son básicamente los mismos. A partir de estas funciones específicas puede extraer una función de integración compuesta más general:

```
> compuesto <- function(f, a, b, n = 10, rule) {  
+   puntos <- seq(a, b, length = n + 1)  
  
+   area <- 0  
+   for (i in seq_len(n)) {  
+     area <- area + rule(f, puntos[i], puntos[i + 1])  
+   }  
  
+   area  
+ }  
> composite(sin, 0, pi, n = 10, rule = puntomedio)  
> composite(sin, 0, pi, n = 10, rule = trapecio)
```

Esta función toma dos funciones como argumentos: la función a integrar y la regla de integración. Ahora podemos agregar reglas aún mejores para integrar rangos más pequeños:

```

> simpson <- function(f, a, b) {
+   (b - a) / 6 * (f(a) + 4 * f((a + b) / 2) + f(b))
+ }
>
> boole <- function(f, a, b) {
+   pos <- function(i) a + i * (b - a) / 4
+   fi <- function(i) f(pos(i))
+
+   (b - a) / 90 *
+     (7 * fi(0) + 32 * fi(1) + 12 * fi(2) + 32 * fi(3) + 7 * fi(4))
+ }
>
> composicion(sin, 0, pi, n = 10, rule = simpson)
> composite(sin, 0, pi, n = 10, rule = boole)

```

Resulta que las reglas del punto medio, trapecio, Simpson y Boole son ejemplos de una familia más general llamada reglas de Newton-Cotes. (Son polinomios de creciente complejidad.) Podemos usar esta estructura común para escribir una función que puede generar cualquier regla general de Newton-Cotes:

```

> newton_cotes <- function(coef, open = FALSE) {
+   n <- length(coef) + open
+
+   function(f, a, b) {
+     pos <- function(i) a + i * (b - a) / n
+     points <- pos(seq.int(0, length(coef) - 1))
+
+     (b - a) / sum(coef) * sum(f(points) * coef)
+   }
+ }
>
> boole <- newton_cotes(c(7, 32, 12, 32, 7))
> milne <- newton_cotes(c(2, -1, 2), open = TRUE)
> compuesto(sin, 0, pi, n = 10, rule = milne)

```

En lugar de crear funciones individuales (por ejemplo, `puntomedio()`, `trapecio()`, `simpson()`, etc.), podríamos almacenarlas en una lista. Si se hace así, ¿cómo cambiaría el código?. ¿Puedes crear la lista de funciones de una lista de coeficientes para las fórmulas de Newton-Cotes?.

(h) ¿ Por qué, las dos siguientes invocaciones de `lapply()` son equivalentes?

```

> trims <- c(0, 0.1, 0.2, 0.5)
> x <- rcauchy(100)
>
> lapply(trims, function(trim) mean(x, trim = trim))
> lapply(trims, mean, x = x)

```