

Inferencia

Rodrigo Zepeda-Tello y Luis Carlos Bernal

2021-01-14

Contents

Chapter 1

Introducción

1.1 Estadística y muestras

La enciclopedia Stanford de filosofía establece la siguiente definición de estadística¹.

Estadística La estadística es una disciplina matemática y conceptual que se enfoca en la relación entre datos e hipótesis. Los datos son registros de observaciones o eventos en un estudio científico, por ejemplo, un conjunto de mediciones de individuos de una población. Los datos que son obtenidos se conoce como la muestra, datos muestrales, o simplemente los datos, y todas las posibles muestras posibles en un estudio forman una colección llamada el espacio muestra. Las hipótesis, por su parte, son enunciados generales sobre el sistema objetivo de la investigación científica, por ejemplo, expresar un hecho general sobre todos los individuos en la población. Una hipótesis estadística es un enunciado general que puede ser expresada como una distribución de probabilidad sobre el espacio muestral, es decir, ésta determina una probabilidad para cada una de las posibles muestras.

De manera breve, la estadística es una disciplina que se encarga de, a través de muestras (cuantificadas como datos), describir el mundo. Y hay muchas cosas por describir: asociaciones, causalidad, realizar predicciones, establecer mecanismos de funcionamiento de objetos, etc. Es así como se establece su objetivo el cual de acuerdo con ? es:

realizar una inferencia sobre la población con base en la información contenida en una muestra de dicha población y proveer una medida asociada de qué tan buena es la inferencia.

¹Traducción y subrayado de Rodrigo

Dentro de la definición previa y objetivos hay que destacar varios términos que son de importancia. La primera es la **población**, cualquier conjunto (no vacío) de objetos. Una **población** es lo más general posible, no necesariamente involucra personas o seres vivos. Algunos ejemplos de poblaciones incluyen: las personas que viven en Guatemala (si me interesa saber algo de los guatemaltecos en general), los árboles del Amazonas (si quiero saber cosas de ecología), los perros callejeros en Ciudad de México, los consumidores de una marca de cereal, los coches que transitan por Dubai, los granos de arena en una playa específica de Cancún, las células T dentro de los seres humanos o los metales pesados.

Más relevante que la población (para nuestros propósitos) es la **población objetivo**. El conjunto de elementos que formarán parte del estudio. Definir la **población objetivo** es complicado en algunas situaciones; por ejemplo, si se desea saber si *los mexicanos* están a favor o en contra de legalizar la marihuana hay que establecer quiénes son *los mexicanos*. ¿Cuentan las personas con nacionalidad mexicana que residen en el extranjero? ¿Cuentan los menores de edad? ¿Qué pasa con los extranjeros que son residentes? De nuevo, la población objetivo no necesariamente son personas, es sólo aquello que nos interesa medir.

Idealmente el estudio estadístico sería sobre la población objetivo. Por ejemplo, si nos interesa estudiar la evolución de los enfermos de VIH, la **población objetivo** serían los enfermos. Sin embargo, en el mundo real es imposible conseguir a toda la población objetivo (dentro de los enfermos, por ejemplo, están aquellos que aún no saben que tienen la enfermedad y no acudirían a nuestro estudio). La **población muestrada** resulta de esta dificultad. La **población muestrada** es el conjunto de elementos sobre los cuales se construyó la muestra para el análisis estadístico. En el caso de los enfermos de VIH la **población muestrada** podrían ser las personas que para una fecha específica habían sido diagnosticadas (y nos olvidamos de quienes desconocen su diagnóstico) o toda la población mexicana (y llevamos kits de diagnóstico con nosotros cuando diagnosticemos). En encuestas de consumo, por ejemplo, usualmente no se muestrean zonas remotas o de muy bajos recursos por lo que la **población muestrada** no coincide con la **población objetivo** (todos los consumidores) sino que son sólo los consumidores de mayor poder adquisitivo. En encuestas de elecciones si bien la población objetivo son *todas las personas que voten el día de la elección*, como la mayoría se hacen *antes* de la elección (exceptuando las de salida) entonces se aproxima la definición de *votante* buscando incluir sólo aquellos que estén registrados en el padrón electoral o bien aquellos que al ser encuestados digan que *sí* van a votar. Aquí la **población muestrada** tampoco coincide con la objetivo.

Una **muestra** es un subconjunto de la población muestrada. Si la muestra coincide con la población muestrada (es decir, muestreaste a todo el mundo) se dice que es un **censo**. Si se tiene un censo se conoce TODA la población por lo que no es necesario hacer ningún análisis de inferencia (ya sabes todo de todos). Puedes realizar predicciones o descripciones. Ejemplos de censos son las encuestas de fin de cursos, las calificaciones de todo un grupo o el registro

de todas las compras de todas las personas en una tienda en línea.

Ojo No hay que confundir la definición de **muestra** con la definición estadística de **muestra aleatoria** (ver más adelante) la cual es un tipo muy específico de muestra obtenida bajo reglas restrictivas.

Finalmente hay que definir **inferencia**, el propósito de estas notas. Para ello usaremos el ejemplo y una versión adaptada de la definición de ?. Considera que sabes dos verdades:

1. Llovió anoche
2. Cuando llueve el suelo se moja

por lo que esta mañana *infieres* que el suelo estará mojado y sales de tu casa con botas y no con chanclas. El proceso de *inferir* parece una consecuencia lógica de las premisas 1 y 2 pero no lo es exactamente: hoy es otro día y si hizo suficiente calor en la noche el agua pudo haberse evaporado del suelo. De ahí que definamos inferencia como:

Realizar un juicio el cual se explica a partir de premisas que suponemos verdaderas.

En particular la **inferencia estadística** será la rama de la estadística cuyo propósito es

Realizar juicios probabilísticos a partir de datos que suponemos verdaderos.

Aquí es necesario desglosar un poco la definición:

- Se habla de **juicios probabilísticos** pues nuestros juicios nunca van a ser tan certeros como *el suelo està mojado*. Más bien van a ser del estilo *hay una probabilidad muy alta de que el suelo esté mojado o nueve de cada diez veces el suelo estará mojado*.
- La **suposición de verdad** de los datos es muy relevante. Imagina el siguiente experimento: tu amiga borracha durante una fiesta se le ocurre que, de la nada, desarrolló poderes de psíquica y puede adivinar el futuro resultado de una moneda (cara o cruz). Tiras una moneda diez veces y todas las veces tu amiga hace una predicción correcta. *Considerando los datos como verdad concluirías que tu amiga es psíquica*. Una observación a profundidad de la moneda quizá te revele que es una moneda truqueada que siempre cae en cara. En ese momento cambiarías la **suposición de verdad** de los datos y la inferencia de que tu amiga es psíquica.

A lo largo de este libro aprenderemos lo básico para realizar inferencias estadísticas: observar datos y suponer verdades a partir de ellos. Tristemente la estadística nunca nos va a poder dar la verdad absoluta pero, si lo hacemos bien, es quizás lo más cerca que podamos estar de ella.

1.2 Modelos

La estadística funciona a partir de la construcción de **modelos**. Estos pretenden ser una forma de describir el mundo mediante teoría de la probabilidad y lo que se busca es utilizar dicha teoría para realizar inferencias. Estos modelos teóricos representan la forma en la que suponemos funciona la población. Para propósitos de estas notas diremos que los modelos viven *en el mundo de los modelos o mundo de las ideas*. Los datos observados, para poder distinguirlos, viven en *el mundo real*. Muchos de los modelos (no todos) se componen de **parámetros** que requieren para poder funcionar los cuales son estimados mediante **estadísticos** que se construyen a partir de los datos.

Para nuestros propósitos, los modelos que usaremos siempre construirán una población de la siguiente forma:

1.2.0.1 Población

Una población es un conjunto no vacío de variables (o vectores) aleatorias.

$$\mathcal{X} = \{X_1, X_2, \dots\}$$

Una población no necesariamente es finita. Por ejemplo, si nos interesa saber el tiempo que tarda un cliente en ser atendido en una llamada telefónica al banco quizás podemos suponer que la llamada telefónica tiene una duración descrita por un modelo Exponencial(λ). La población sería el conjunto infinito de todas las posibles llamadas telefónicas que se pueden realizar bajo este modelo. Por otro lado, un ejemplo finito de una población, son las caras de una moneda en un experimento donde busquemos, para una moneda específica, si caen más caras que cruces (cae más de un lado que del otro).

Una **muestra** es cualquier subconjunto (posiblemente infinito también) de la población.

1.2.0.2 Muestra

Una muestra \mathcal{M} de una población \mathcal{X} es cualquier subconjunto no vacío de \mathcal{X} . Es decir, \mathcal{M} es una muestra de \mathcal{X} si:

$$\mathcal{M} \subseteq \mathcal{X}$$

Pocas veces hablaremos de *muestras* de manera general y nos enfocaremos, sobre todo, en **muestras aleatorias**:

1.2.0.3 Muestra aleatoria

Una muestra aleatoria de tamaño n , $\mathcal{X}_{(n)}$, de una población \mathcal{X} es un subconjunto finito (de tamaño n), no vacío de \mathcal{X} donde sus elementos son **variables aleatorias independientes idénticamente distribuidas**. Es decir, $\mathcal{X}_{(n)}$ es una muestra de \mathcal{X} si:

1. **Es una muestra:** $\mathcal{X}_{(n)} \subseteq \mathcal{X}$,
2. **de tamaño n :** Cardinalidad($\mathcal{X}_{(n)}$) = n ,
3. **con variables independientes:** si $X_i, X_j \in \mathcal{X}_{(n)}$ entonces $\mathbb{P}(X_i \in A, X_j \in B) = \mathbb{P}(X_i \in A) \cdot \mathbb{P}(X_j \in B)$ para A, B conjuntos medibles,
4. **idénticamente distribuidas:** para todo $i = 1, 2, \dots, n$ se tiene que X_i tiene función de distribución acumulada F_X .

El punto 4. de la definición pide que todas las variables descritas tengan la misma distribución. Por ejemplo, podemos pedir que todas sean exponenciales con el mismo parámetro o todas sean gamma con los mismos parámetros. El punto es que todas las variables aleatorias estén descritas con el mismo modelo y sean independientes entre sí.

El punto 3. de la definición puede escribirse de otras formas más amigables, por ejemplo, si suponemos que las variables aleatorias son continuas y tienen densidad f_X entonces la independencia puede escribirse como:

$$f_X(x_i, x_j) = f_X(x_i) \cdot f_X(x_j)$$

mientras que si son discretas con función de masa de probabilidad p_X tenemos:

$$p_X(x_i, x_j) = p_X(x_i) \cdot p_X(x_j)$$

La **muestra observada** así como la **muestra aleatoria observada** es el conjunto de *datos* que realmente viste. Mientras que la **muestra** y la **muestra aleatoria** viven en el mundo de los *modelos* y son variables aleatorias (constructos teóricos, como sabes, bastante complejos), la **muestra observada** es lo que se midió. Antes de dar la definición veamos un ejemplo con un dado.

1.2.0.4 Tiro de un dado

Se realiza un experimento para saber si un dado es justo (todos los lados tienen la misma probabilidad). Para ello se tira el dado $n = 10$ veces y se registran los tiros: 2, 6, 1, 3, 3, 3, 5, 1, 3, 2.

Mundo del modelo

La población en este caso es el conjunto infinito de todos los posibles tiros del dado. De ese conjunto obtenemos una muestra aleatoria de tamaño

$n = 10$ (suponemos que los tiros son independientes entre sí) dada por:

$$X_{(n)} = \{X_1, X_2, X_3, \dots, X_{10}\}$$

donde X_i tiene la siguiente distribución:

$$\mathbb{P}(X_i = z) = \begin{cases} p_1 & \text{si } z = 1 \\ p_2 & \text{si } z = 2 \\ p_3 & \text{si } z = 3 \\ p_4 & \text{si } z = 4 \\ p_5 & \text{si } z = 5 \\ p_6 & \text{si } z = 6 \\ 0 & \text{en otro caso.} \end{cases}$$

donde $\sum_{k=1}^n p_k = 1$ y $p_k \geq 0$ para todo k . Lo que interesa en este estudio es *inferir* quiénes son las p_k para determinar si es más probable que caiga en un lado que en otro. Las p_k se conocen como parámetros.

Mundo real

Ya en la realidad en esos 10 tiros no observamos cualquier cosa, observamos valores específicos que hacen que la **muestra aleatoria observada** sea:

$$s_n = \{x_1, x_2, \dots, x_{10}\} = \{2, 6, 1, 3, 3, 3, 5, 1, 3, 2\}.$$

Por supuesto que repitiendo el experimento (volviendo a tirar el dado 10 veces) lo más probable es que la **muestra aleatoria observada** cambie (y veamos otros números) pero el modelo, reflejado en la **muestra aleatoria** (teórica), permanezca inmutable. Una forma de estimar las probabilidades podría ser mediante proporciones y calcular, por ejemplo, la probabilidad de que aparezca 1 como:

$$\hat{p}_1 = \frac{\text{Veces que aparece 1}}{n} = \frac{2}{10}$$

En este caso, \hat{p}_1 dado por $\frac{2}{10}$ es un **estimador observado** de la verdadera probabilidad p_1 que vive en el mundo de los modelos (y jamás podremos conocer)

Armados con el ejemplo anterior realicemos la definición de las muestras observadas:

1.2.0.5 Muestra observada

Una muestra observada es una colección no vacía de valores codificados como números reales los cuales corresponden a realizaciones de una muestra

\mathcal{X} . Usualmente la denotamos:

$$s = \{x_1, x_2, \dots\}$$

donde las x_i NO SON VARIABLES ALEATORIAS sino que son datos **fijos** ya observados.

1.2.0.6 Muestra aleatoria observada

Una muestra aleatoria observada es una colección no vacía de tamaño n de valores codificados como números reales los cuales corresponden a realizaciones de una muestra aleatoria $\mathcal{X}_{(n)}$. En particular suponemos que x_1 es el valor observado de la variable aleatoria X_1 , x_2 es el valor observado de la variable aleatoria X_2 y así sucesivamente. Generalmente la denotamos por:

$$s_{(n)} = \{x_1, x_2, \dots, x_n\}$$

donde las x_i NO SON VARIABLES ALEATORIAS sino que son datos **fijos** ya observados.

Veamos un segundo ejemplo:

1.2.0.7 Cantidad de personas que llegan a una tienda

En muchos casos la llegada de personas se supone que sigue una distribución Poisson. En este caso nos interesa estimar el número promedio de personas por día que hay en una tienda de la cual se han medido las siguientes cantidades (por día). Suponemos que las llegadas son independientes entre sí (la cantidad de gente que llegó un día no influye en la cantidad que llegó el otro).

Día	Número de personas
1	50
2	45
3	60
4	65
5	55
6	40

Mundo del modelo

La población en este caso es el conjunto infinito de todas las posibles formas en que en un día pueden llegar personas. De ese conjunto obtenemos una muestra aleatoria de tamaño $n = 6$ (suponemos que las observaciones son independientes entre sí) dada por:

$$X_{(n)} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$$

donde las $X_i \sim \text{Poisson}(\lambda)$ (todas con el mismo λ). Recordamos que la media de una Poisson es λ por lo que el **parámetro** que nos interesa estimar es λ .

Mundo real

A partir de las 6 llegadas observadas construimos **la muestra aleatoria observada**:

$$s_n = \{x_1, x_2, x_3, x_4, x_5, x_6\} = \{50, 45, 60, 65, 55, 40\}.$$

Una forma de estimar la media λ es mediante el siguiente **estimador observado**:

$$\hat{\lambda} = \frac{1}{6} \sum_{i=1}^6 x_i = 52.5$$

Ojo, esto no significa que λ sea 52.5. Significa que nuestra hipótesis de quién es λ es 52.5 y que esperaríamos la próxima vez en la tienda 52 ó 53 personas. En el mundo real *quién sabe cuánto vale λ* , nuestra hipótesis es que vale 52.5 pero eso no necesariamente es la realidad.

Como ya establecimos, muchas veces el modelo utiliza un **parámetro** el cual es desconocido. A partir de los datos construimos un **estimador observado** el cual es nuestra hipótesis del verdadero valor del parámetro. En general va a ser imposible que le atinemos al *verdadero* valor del parámetro pero la idea es que el **estimador observado** esté lo suficientemente cerca. En el ejemplo anterior nos gustaría, por ejemplo, que el verdadero parámetro quizás fuera $\lambda = 52$ ó $\lambda = 54$ pero nos sacaría mucho de onda que el parámetro real fuera $\lambda = 1000000$.

1.2.0.8 Distirbución paramétrica

Una función de distirbución acumulada es una **distribución paramétrica** con parámetro $\vec{\theta}$ si dada una colección de distribuciones

$$\{F_{\vec{\theta}}|\theta \in \Theta\}$$

determinar $\vec{\theta}$ determina la distribución. Es decir, la familia de distribuciones estándizada por $\vec{\theta}$. A $\vec{\theta}$ se le conoce como el **parámetro** o **vector de parámetros**.

La definición anterior suena muy compleja sin embargo los ejemplos ya los conocemos.

1.2.0.9 La normal

La distribución normal es una distribución paramétrica con

$$\vec{\theta} = (\mu, \sigma^2)^T$$

el vector de parámetros dado por la media y la varianza.

1.2.0.10 La exponencial

La distribución exponencial es una distribución paramétrica con

$$\theta = \lambda$$

el parámetro que establece la tasa de la exponencial.

1.2.0.11 La normal con varianza 1

La distribución normal con varianza 1 es una distribución paramétrica con

$$\theta = \mu$$

En este caso la varianza es conocida ($\sigma^2 = 1$) pero la media no por eso sólo la media es el parámetro.

Podemos entonces definir un **estimador**:

1.2.0.12 Estimador

Dada una distribución paramétrica F_θ con parámetro θ un estimador $\hat{\theta}$ de θ es una variable aleatoria que se construye como función de la muestra aleatoria:

$$\hat{\theta} : \mathcal{X}_{(n)} \rightarrow \Theta$$

Como $\hat{\theta}$ es una función de la muestra aleatoria entonces puede representarse como:

$$\hat{\theta} = \hat{\theta}(X_1, X_2, \dots, X_n)$$

Dado un conjunto de datos, el **estimador observado de θ** es el estimador $\hat{\theta}$ de θ evaluado en los datos.

1.2.0.13 Estimador observado

Dada una distribución paramétrica F_θ con parámetro θ con estimador $\hat{\theta}$ y datos observados $s_{(n)} = \{x_1, x_2, \dots, x_n\}$ el **estimador observado** corresponde a la evaluación de $\hat{\theta}$ en $s_{(n)}$; es decir:

$$\hat{\theta}(x_1, x_2, \dots, x_n)$$

Veamos ejemplos para entender mejor cómo funciona esto.

1.2.0.14 Tiros de una moneda

Se tiene una moneda que cae más de un lado que del otro. Interesa estimar p la probabilidad de que caiga cruz. Para ello se toma una **muestra aleatoria** de 5 tiros de la moneda:

$$X_{(n)} = \{X_1, X_2, \dots, X_5\}$$

Suponemos que los tiros son independientes. El modelo entonces implicaría que

$$X_i \sim \text{Bernoulli}(p)$$

para cada $i = 1, 2, \dots, 5$. Si codificamos cruz como 1 y cara como 0, la **muestra aleatoria observada** es:

$$s_{(n)} = \{1, 1, 1, 0, 1\} = \{x_1, x_2, \dots, x_5\}$$

donde tuvimos tres cruces continuas, luego una cara y finalmente una cruz. Una opción de estimador observado sería contar la proporción de cruces haciendo:

$$\hat{\theta}(x_1, \dots, x_5) = \frac{1}{n} \sum_{k=1}^n x_i = \frac{4}{5}$$

de donde diríamos que nuestra hipótesis de cuánto vale el parámetro p es $4/5$. Por otro lado, el **estimador teórico** es:

$$\hat{\theta}(X_1, \dots, X_5) = \frac{1}{n} \sum_{k=1}^n X_i$$

el cual tiene una distribución de probabilidad sencilla pues $\sum_{k=1}^n X_i \sim \text{Binomial}(n, p)$. Particularmente podemos calcular su valor esperado, por

ejemplo,

$$\mathbb{E}[\hat{\theta}(X_1, \dots, X_5)] = \mathbb{E}\left[\frac{1}{n} \sum_{k=1}^n X_i\right] = \frac{1}{n} \sum_{k=1}^n \mathbb{E}[X_i] = \frac{1}{n} \sum_{k=1}^n p = \frac{1}{n} np = p$$

lo cual implica que el estimador, en promedio, devolvería el parámetro que nos interesa (esta propiedad se conoce como *ser insesgado* y lo veremos más adelante).

1.2.0.15 Error de medición de una app

Una app que se dedica a medir la altura de edificios mediante la toma de videos tiene un error de medición con distribución normal y cuya varianza es 1. Interesa determinar el error de medición promedio, el parámetro μ . Para ello se toman videos y se miden edificios para obtener una colección de 7 errores de medición independientes en la siguiente **muestra aleatoria**:

$$X_{(n)} = \{X_1, X_2, \dots, X_5\}$$

El modelo es

$$X_i \sim \text{Normal}(\mu, 1)$$

para cada $i = 1, 2, \dots, 7$. Si los datos fueron:

Edificio	Error de medición
Bellas Artes	12.11
Torre Latinoamericana	40.54
Catedral Metropolitana	22.07
Palacio Nacional	15.22
Rectoría de la UNAM	45.18
Guerrero Chimalli	33.39
Estadio Azteca	41.76

la **muestra aleatoria observada** en este caso correspondió :

$$s_{(n)} = \{12.11, 40.54, 22.07, 15.22, 45.18, 33.39, 41.76\} = \{x_1, x_2, \dots, x_7\}$$

Una opción de estimador observado sería calcular la media muestral haciendo:

$$\hat{\theta}(x_1, \dots, x_7) = \frac{1}{n} \sum_{k=1}^n x_i = 30.03857$$

de donde diríamos que nuestra hipótesis de cuánto vale el parámetro μ es 30.03857. Por otro lado, el **estimador teórico** es:

$$\hat{\theta}(X_1, \dots, X_7) = \frac{1}{n} \sum_{k=1}^n X_i$$

tiene una distribución de probabilidad sencilla pues sabemos que $\sum_{k=1}^n X_i \sim \text{Normal}(\mu, \sigma^2)$. Particularmente podemos calcular su valor esperado, por ejemplo,

$$\mathbb{E}\left[\hat{\theta}(X_1, \dots, X_7)\right] = \mathbb{E}\left[\frac{1}{n} \sum_{k=1}^n X_i\right] = \frac{1}{n} \sum_{k=1}^n \mathbb{E}[X_i] = \frac{1}{n} \sum_{k=1}^n \mu = \frac{1}{n} n\mu = \mu$$

lo cual implica que el estimador, en promedio, devolvería el parámetro que nos interesa (este también es *insesgado*).

A modo de resumen y para concluir este capítulo introductorio, veamos un ejemplo más desarrollado de inferencia estadística.

1.3 Inferencia estadística: ejemplo.

Se tiene una caja con cinco pelotas de colores rojo R y azul A . Las pelotas son indistinguibles² entre sí salvo por el color. Se desconoce exactamente la proporción de colores de la caja (es decir no se sabe cuál de las siguientes opciones es: $\{R, R, R, R, R\}$, $\{R, R, R, R, A\}$, $\{R, R, R, A, A\}$, $\{R, R, A, A, A\}$, $\{R, A, A, A, A\}$, $\{A, A, A, A, A\}$) y eso es lo que se desea determinar. Para ello se extrae una bola, se anota que su color fue rojo, R , y se devuelve a la caja. Se extrae otra bola (que, pudo haber sido la misma que la inicial, recuerda que las bolas son indistinguibles y que la anterior se devolvió a la caja), se anota que su color fue rojo R y se devuelve a la caja. Finalmente en la tercera extracción sale una pelota azul A . Los datos observados (y ordenados) son los siguientes (R, R, A) . Hay dos estimadores posibles de la probabilidad de que salga rojo p que podemos calcular a partir de la **muestra aleatoria** ordenada (R, R, A)

²Pelotas distinguibles, por ejemplo, tendrían números distintos o serían de materiales diferentes o con marcas específicas para saber que una azul es distinta de la otra.

1.3.1 Estimador de momentos

Una opción es estimar la probabilidad de que salga rojo, p , mediante el conteo de cuántos rojos salieron divididos entre el total de extracciones. En este caso tendríamos el estimador evaluado en la muestra:

$$\hat{p}_M = \frac{2}{3}$$

Aquí una nota bien importante: es imposible (*¿por qué?*) que en la vida real la probabilidad p de que salga rojo sea $2/3$. El \hat{p} es un estimador pero que jamás va a coincidir con el valor de verdad.³. Sin embargo este estimador \hat{p} tiene características interesantes. Para verlas, definamos primero una **muestra aleatoria** de tamaño 3 de las pelotas en la urna:

$$X_{(n)} = \{X_1, X_2, X_3\}$$

donde marcaremos $X_i = 1$ si salió rojo y $X_i = 0$ si salió azul. Preguntarnos por la probabilidad de rojo es lo mismo que preguntarnos por la probabilidad de que $X_i = 1$. El estimador \hat{p} evaluado en la muestra, la suma ponderada de todos, (los rojos aportan 1 y los azules nada) está dado por:

$$\hat{p}(X_1, X_2, X_3) = \frac{1}{3}(X_1 + X_2 + X_3)$$

Notamos que para este caso los datos (muestra aleatoria observada) son $x_1 = 1$, $x_2 = 1$ y $x_3 = 0$. Por lo que el estimador observado es:

$$\hat{p}(x_1, x_2, x_3) = \frac{1}{3}(x_1 + x_2 + x_3) = \frac{2}{3}$$

Notamos que el estimador \hat{p} es una variable aleatoria que depende de la muestra (aleatoria). Una vez que se tiene la muestra el estimador \hat{p} colapsa en un único número real definido por la tabla. Pero antes de hacer el experimento (o bien si repetimos el experimento) el \hat{p} es una variable aleatoria que puede obtener múltiples valores distintos. Como es una variable aleatoria podemos entonces calcular su varianza, por ejemplo, así como su media:

$$\begin{aligned} \mathbb{E}[\hat{p}] &= \frac{1}{3}\mathbb{E}[X_1 + X_2 + X_3] \\ &= \frac{1}{3}(\mathbb{E}[X_1] + \mathbb{E}[X_2] + \mathbb{E}[X_3]) \\ &= \frac{1}{3}3p \\ &= p \end{aligned}$$

³Esto porque en la caja hay cinco pelotas y el denominador de la probabilidad p va a ser 5 por construcción del modelo. No hay forma de que el 5 se pueda simplificar en 3 o viceversa.

por lo que si hiciéramos el ejercicio de muestreo múltiples veces los estimadores \hat{p} que obtuviéramos le atinarían en promedio a p . Por otro lado la varianza está dada por:

$$\begin{aligned}\text{Var}Big[\hat{p}\Big] &= \frac{1}{9}\text{Var}[X_1 + X_2 + X_3] \\ &= \frac{1}{9}(\text{Var}[X_1] + \text{Var}[X_2] + \text{Var}[X_3]) \\ &= \frac{1}{9}3p(1-p) \\ &= \frac{1}{3}p(1-p)\end{aligned}$$

Podemos calcular más propiedades probabilísticas de \hat{p} pero el punto importante es que el \hat{p} tiene su propia distribución.

En R podemos simular este proceso de extracción de los \hat{p} :

```
library(ggplot2)

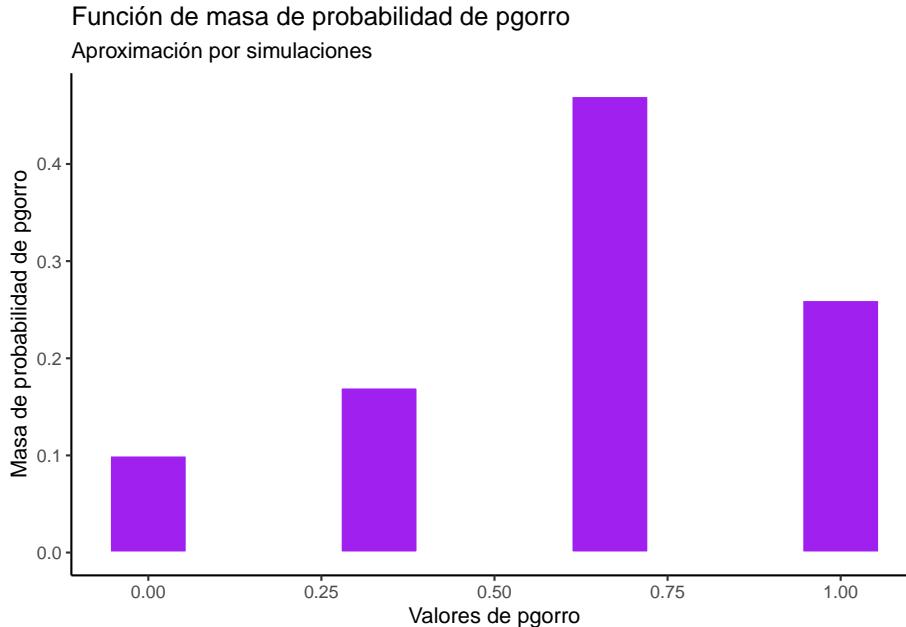
# Número de veces que haremos el experimento de extraer 3 pelotas
nsim      <- 100
tamaño.muestra <- 3

#La verdadera población
poblacion <- c("R", "R", "R", "A", "A")

#Aquí guardaremos los valores de pgorro
pgorro    <- rep(NA, nsim)

#Repetimos el proceso de muestreo n veces
for (i in 1:nsim){
  muestra      <- sample(poblacion, tamaño.muestra, replace = T)
  conteo_rojos <- length(which(muestra == "R"))
  pgorro[i]     <- conteo_rojos/tamaño.muestra
}

ggplot() +
  geom_histogram(aes(x = pgorro, y = ..count../100),
                 bins = 10, fill = "purple", color = "white") +
  theme_classic() +
  labs(
    x = "Valores de pgorro",
    y = "Masa de probabilidad de pgorro",
    title = "Función de masa de probabilidad de pgorro",
    subtitle = "Aproximación por simulaciones"
)
```



Esto implica que aunque tengamos datos fijos $\{R, R, RA, A\}$ como la extracción de la muestra es aleatoria el valor de \hat{p} va a variar por el simple proceso de selección aleatoria de la muestra. Darnos cuenta de qué tanto varía nuestro valor a estimar y cómo se aleja (o no) de la verdad va a ser un punto muy importante (en general queremos estimadores \hat{p} que no se alejen de los valores verdaderos).

Ahora, los estimadores \hat{p} no son únicos y se nos puede ocurrir otro estimador como ejemplo:

1.3.2 Estimador de máxima verosimilitud (maximización discreta)

Una segunda opción es buscar bajo qué valor de p (de muchos posibles) son más probables los datos observados. Este criterio se conoce como el criterio de **máxima verosimilitud**. La idea es ver, bajo cada una de las construcciones posibles de la caja (*i.e.* $\{R, R, R, R, R\}$, $\{R, R, R, R, A\}$, $\{R, R, R, A, A\}$, $\{R, R, A, A, A\}$, $\{R, A, A, A, A\}$, $\{A, A, A, A, A\}$) son más probables los datos observados (R, R, A) y elegir esa caja. Vamos a resolver este problema calculando bajo cada escenario de caja las probabilidades de obtener la combinación observada (R, R, A) .

La pregunta, antes de resolver el problema, es *¿y esto para qué sirve?* . Si bien los ejercicios de pelotas de colores y cajas son divertidos, estos por sí mismos no llegan muy lejos. Lo importante es que los ejercicios de urnas *son abstracciones*

de problemas reales. Por ejemplo, el problema de urnas ocurre de manera poblacional cuando nos interesa estimar una proporción. En un país hay personas que padecen diabetes R y sin diabetes A. Se sabe que en el país hay 100 millones de personas. La proporción exacta (dentro del país) es desconocida. Sin embargo podemos hacer una encuesta y obtener una muestra de 100 personas dentro de las cuales 20 padecieron diabetes y 80 no ($\{20R, 80A\}$). De aquí, usando el mismo razonamiento que usaremos con la caja de pelotas podemos buscar la combinación de personas con diabetes y sin diabetes en el país donde la combinación de $\{20R, 80A\}$ es la más probable.

Este mismo razonamiento puede cambiarse de múltiples formas. En una encuesta podemos tener más de dos opciones. Por ejemplo, si interesa determinar la cantidad de personas que votarían por el partido rojo R , por el azul A o por el negro B el modelo de pelotas en una caja ahora tiene tres tipos de pelota (y si hay n partidos habría n colores). Puede que los colores estén relacionados entre sí y extraer uno garantice la extracción de otro (por ejemplo si se entrevista gente en casas es muy probable que si se vive un niño en una casa *a fuerza* viva un adulto en ella mientras que la relación inversa no funciona: que un adulto habite una casa no determina que viva un niño en la misma). Otros cambios posibles son en el mecanismo de selección. Pudiera ser que las pelotas rojas R fueran más grandes que las azules A de tal forma que cuando se extrajeran hubiera mayor probabilidad de tener rojas en la muestra. Esto pasa, por ejemplo, cuando se hacen encuestas de productos. Generalmente sólo aquellas personas que tienen un sentimiento muy fuerte hacia un producto contestan la encuesta. De ahí que haya muchísimas reseñas diciendo que los productos son malísimos o buenísimos y nada intermedio: la gente que reseña algo con 3 estrellas son las pelotas azules que son más difíciles de extraer. Poco a poco veremos otros problemas con pelotas y urnas con sus análogos al mundo real. Por ahora resolvamos el que se especifica más arriba.

1.3.3 Ejemplo 5: Muestreo de urna con dos clases, con orden, con reemplazo

Considera una urna con cinco pelotas de colores rojo R y azul A . Se desconoce la proporción de pelotas en la urna. Las pelotas son indistinguibles entre sí salvo por el color. Se extrae de manera ordenada primero una bola roja, R , y se devuelve a la urna. Luego se extrae una segunda bola roja, R , y se devuelve a la urna. Finalmente se extrae una tercera bola y resulta azul: A . La combinación ordenada de pelotas extraídas es: (R, R, A) . Suponiendo todas las pelotas tienen la misma probabilidad de salir, cuál urna genera con mayor probabilidad los datos observados (y por tanto sería nuestra opción para decidir qué urna es la que tenemos): $\{R, R, R, R, R\}$, $\{R, R, R, R, A\}$, $\{R, R, R, A, A\}$, $\{R, R, A, A, A\}$, $\{R, A, A, A, A\}$ ó $\{A, A, A, A, A\}$.

1.3.4 Solución 5: Muestreo de urna con dos clases, con orden, con reemplazo

Hay dos combinaciones de posible urna que podemos descartar desde un inicio: la que sólo tiene rojos $\{R, R, R, R, R\}$ y la que sólo tiene azules $\{A, A, A, A, A\}$. Esto porque los datos observados nos muestran que obtuvimos tantos rojos como azules. En estas dos descartadas la probabilidad de obtener (R, R, A) es cero. Quedan como cajas posibles: la de cuatro rojas $\{R, R, R, R, A\}$, la de tres rojas $\{R, R, R, A, A\}$, la de tres azules $\{R, R, A, A, A\}$, la de una roja $\{R, A, A, A, A\}$. Comenzaré mi análisis con la primera que puse: los otros análisis son similares.

Análisis de $\{R, R, R, R, A\}$

Una de las formas más posibles de enlistar todos los escenarios es con un árbol. La forma larga (e impráctica) consiste en enlistar cada una de las formas de extraer las pelotas de la urna como muestra la siguiente imagen:

Esta es una forma “segura” de resolver un problema: puede ser largo pero si no se te olvida nada ¡siempre es una posibilidad! Observa que de todas las opciones (combinadas del primer nodo, el segundo y el tercero) se tienen 16 extracciones de la forma (R, R, A) de un total de 125 extracciones (5 opciones en el último nodo por 5 formas de extraer el segundo por 5 opciones primeras dan 125). La probabilidad entonces de extraer (R, R, A) bajo este esquema es:

$$\text{Probabilidad de } (R, R, A) \text{ dada la urna } \{R, R, R, R, A\} \text{ con reemplazo} = \frac{16}{125}$$

Análisis de $\{R, R, R, A, A\}$

Vale la pena para este segundo análisis podar un poco el árbol. Podemos darnos cuenta que en el caso anterior todas las ramas rojas son iguales por lo que quizás no vale la pena ponerlas todas. Al calcular la probabilidad de la extracción (R, R, A) dado que la urna es de la forma $\{R, R, R, A, A\}$ trabajaremos más a fondo el árbol. El árbol inicial es como sigue:

En este caso notamos que todos los caminos iniciados por rojo, R, son idénticos lo mismo que los caminos iniciados por azul A por lo que podemos simplificar el árbol copiando sólo las ramas distintas y anotando cada rama a cuántas representa (las azules son 2 de 5 mientras que las rojas 3 de 5 de ahí los números $2/5$ y $3/5$).

Para el caso que nos atañe en esta ocasión: obtener primero roja, luego otra roja y finalmente azul, (R, R, A) , la única opción es la

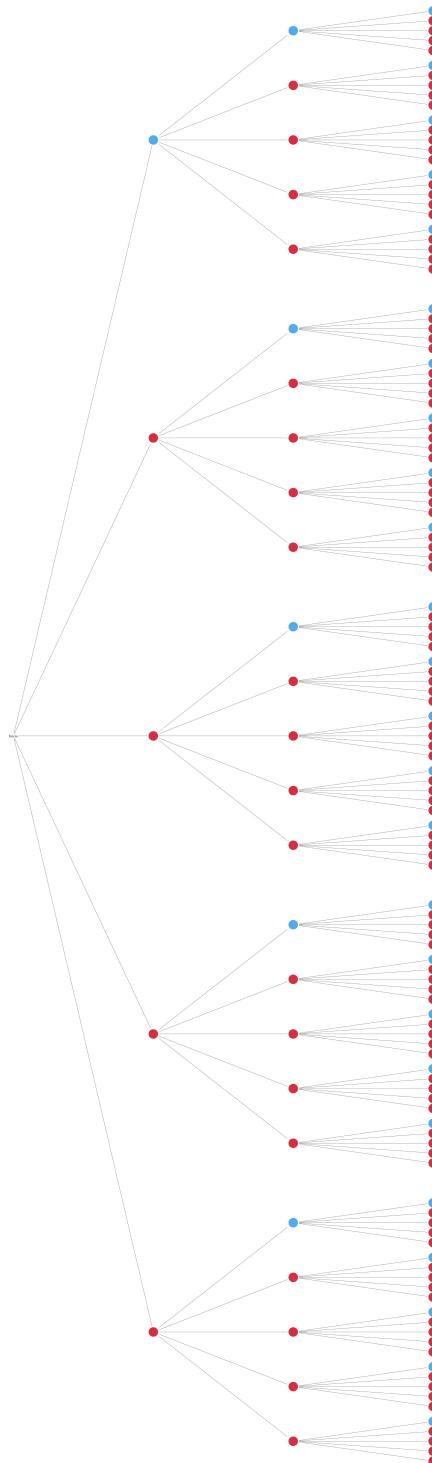
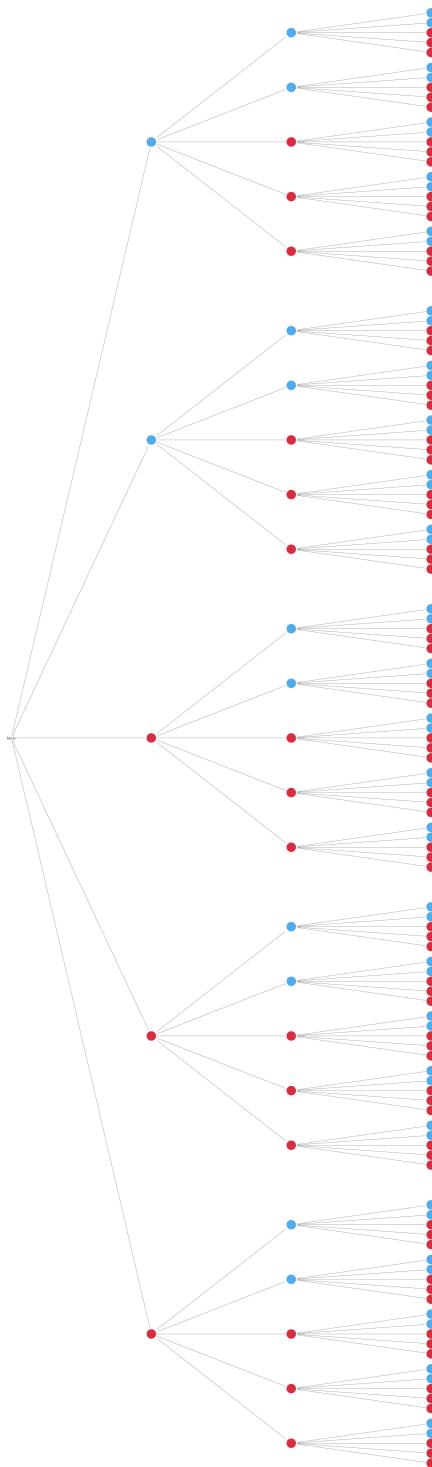


Figure 1.1: Árbol de decisión para el análisis de $\{R, R, R, R, A\}$ con reemplazo

Figure 1.2: Árbol de decisión en el caso $\{R, R, R, A, A\}$ con reemplazo

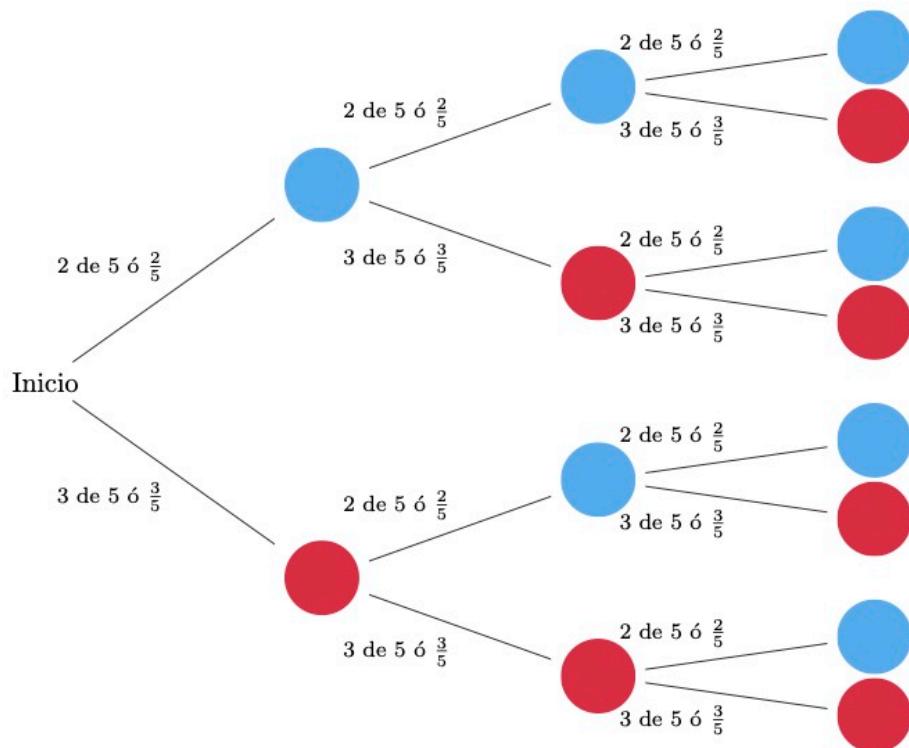


Figure 1.3: Árbol de decisión simplificado para el análisis de $\{R, R, R, A, A\}$ con reemplazo

rama de abajo con probabilidades dadas por:

$$\text{Probabilidad de } (R, R, A) \text{ dada la urna } \{R, R, R, A, A\} \text{ con reemplazo} = \frac{3 \times 3 \times 2}{5 \times 5 \times 5} = \frac{18}{125}$$

Esta forma reducida es equivalente a la que hubiéramos obtenido haciendo el análisis completo del árbol (por la primera imagen donde hay 18 opciones de 125 resultados). El producto de arriba refiere a el número de opciones para primera roja, por el número de opciones para segunda roja *dado* que la primera fue roja y la última son las opciones para una tercera azul *condicional* en que las dos primeras fueron rojas. En el denominador sólo multiplicamos los casos totales que son 5 ramas iniciales que ramifican en 5 nodos secundarios y 5 hojas finales.

Tómate unos minutos para intentar justificar por qué este conteo de multiplicaciones es equivalente a haber contado todas. Asegúrate de entenderlo bien antes de continuar ¡volveremos a ello más adelante!

Análisis de $\{R, R, A, A, A\}$

En este análisis usaremos el mismo truco de un árbol de decisiones reducido que usamos la vez pasada. Lo construiremos paso a paso para mostrar el proceso. De manera inicial tenemos dos opciones: roja (2 bolas de 5) ó azul (3 bolas de 5) por lo que a partir de la raíz construimos el árbol con las dos opciones y sus probabilidades

Dentro de la rama azul de nuevo tenemos la posibilidad de extraer rojas (2 de 5) o azules (3 de 5) por lo que se acoplan a la rama:

La lógica es idéntica si salió roja: hay $2/5$ de probabilidad de extraer una roja o $3/5$ de extraer una azul

Finalmente las últimas ramas del árbol se agregan de manera idéntica: $2/5$ de rojas y $3/5$ de azul en cada una de las posibles extracciones.

Notamos entonces que podemos hacer el cálculo multiplicando (como hicimos la vez pasada) para obtener:

$$\text{Probabilidad de } (R, R, A) \text{ dada la urna } \{R, R, A, A, A\} \text{ con reemplazo} = \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} = \frac{12}{125}$$

Análisis de $\{R, A, A, A, A\}$

Ya para este último caso los árboles de decisiones te son familiares por lo que puedes verificar que en este caso el árbol es el siguiente:

donde la probabilidad de (R, R, A) dado que la urna es $\{R, A, A, A, A\}$ está dada por:

$$\text{Probabilidad de } (R, R, A) \text{ dada la urna } \{R, A, A, A, A\} \text{ con reemplazo} = \frac{1}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} = \frac{4}{125}.$$

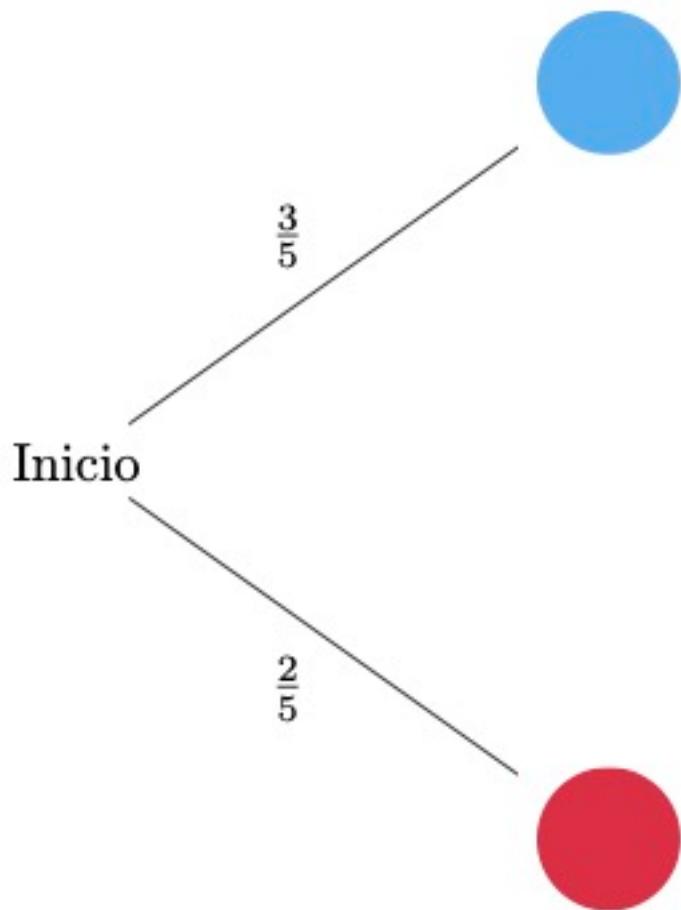


Figure 1.4: Versión 1 del árbol de decisión simplificado para $\{R, R, A, A, A\}$ con reemplazo

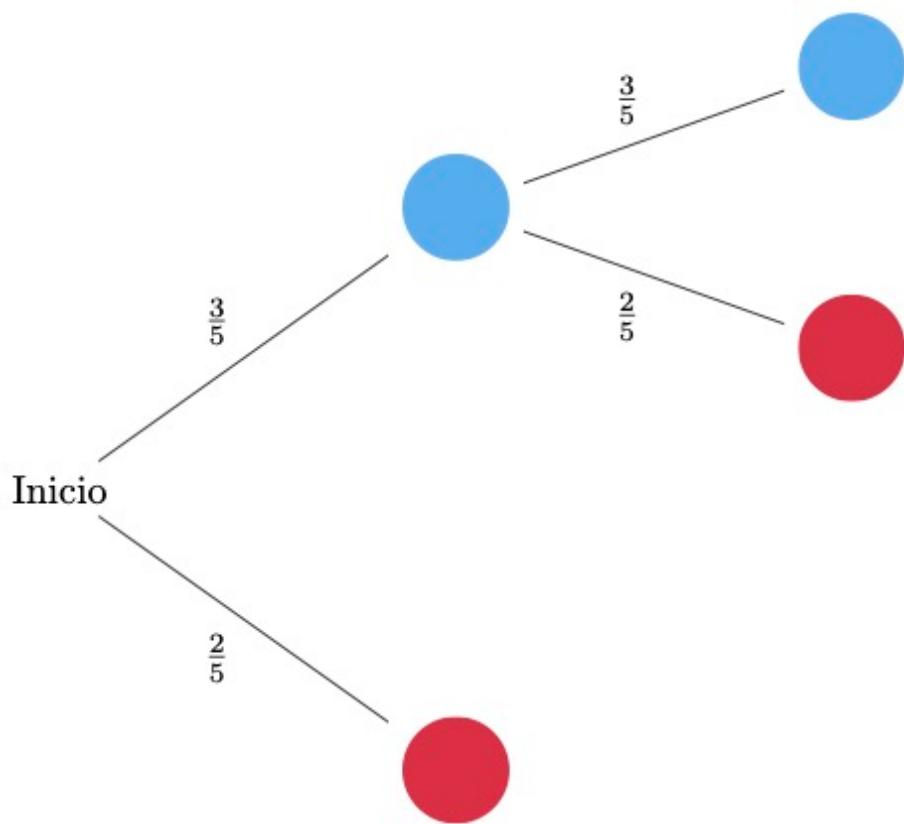


Figure 1.5: Versión 2 del árbol de decisión simplificado para $\{R, R, A, A, A\}$ con reemplazo

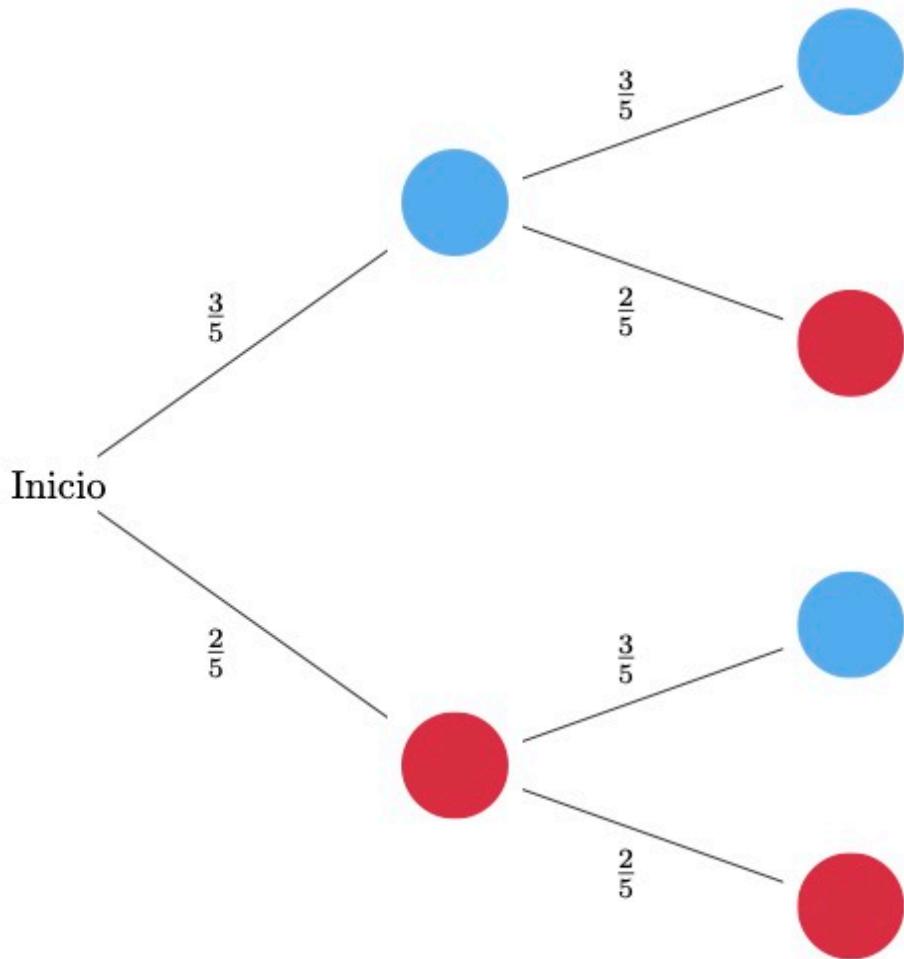


Figure 1.6: Versión 3 del árbol de decisión simplificado para $\{R, R, A, A, A\}$ con reemplazo

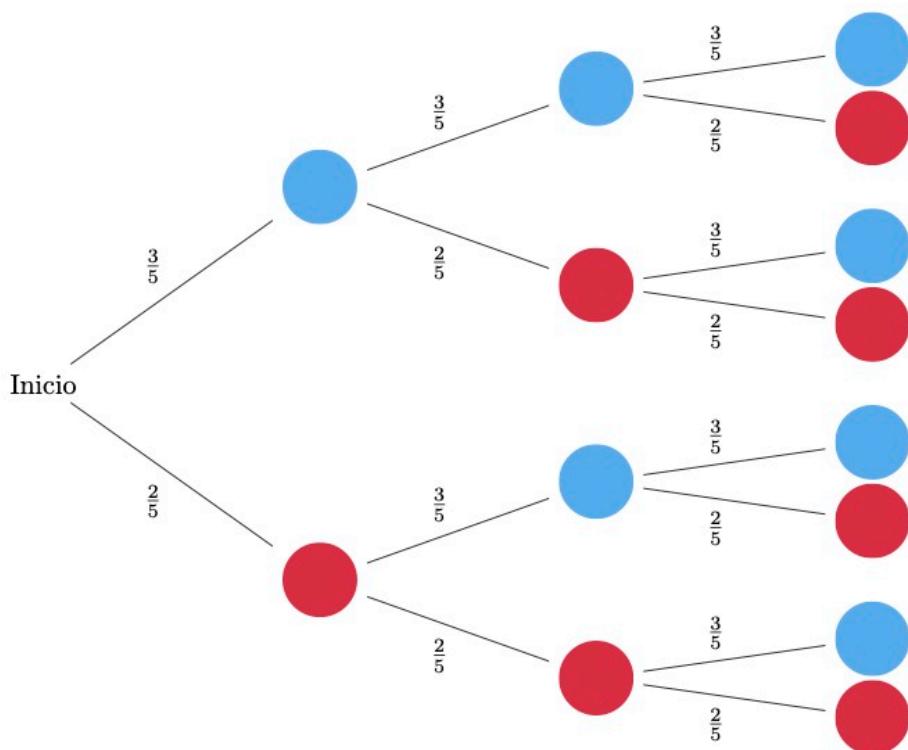


Figure 1.7: Versión final del árbol de decisión simplificado para $\{R, R, A, A, A\}$ con reemplazo

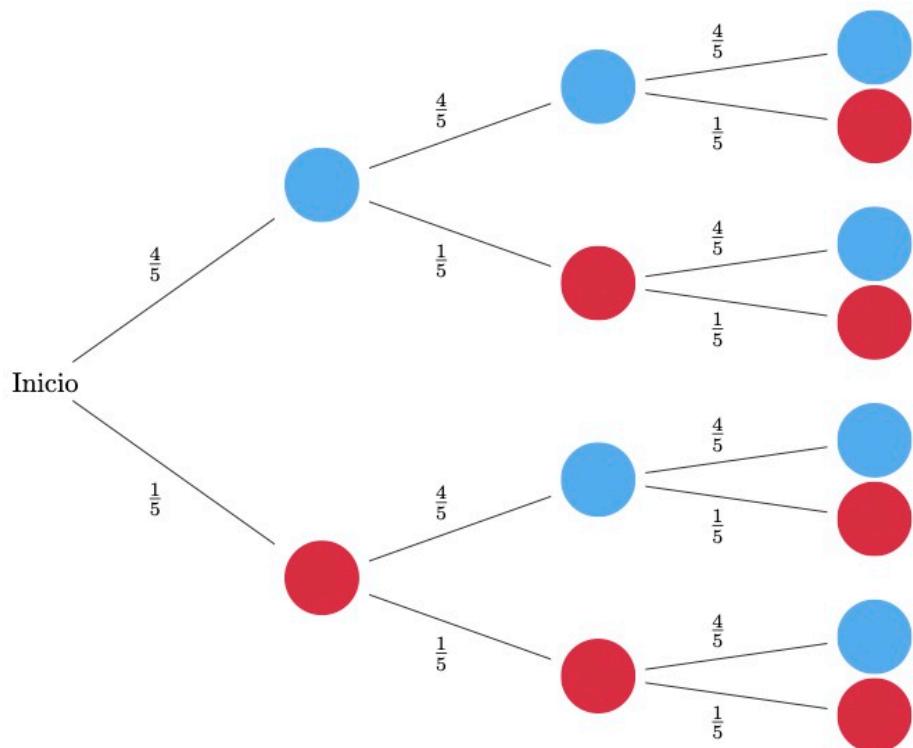


Figure 1.8: Árbol de decisión simplificado para el análisis de $\{R, A, A, A, A\}$ con reemplazo

Conclusión

Después de que analizamos las probabilidades bajo todas las combinaciones posibles de urna concluimos que la que tiene probabilidad más alta de generar en ese orden rojo, luego rojo y finalmente azul, (R, R, A) , es la urna $\{R, R, R, A, A\}$ pues si la urna tiene esa combinación de pelotas la probabilidad de extraer (R, R, A) es:

$$\text{Probabilidad de } (R, R, A) \text{ dada la urna } \{R, R, R, A, A\} \text{ con reemplazo} = \frac{18}{125} = 0.144.$$

Siguiendo la idea de que la urna que tenemos es la que tiene mayor probabilidad de generar los datos observados (esto se conoce como *criterio de máxima verosimilitud* en estadística) entonces estimaríamos que la urna que tenemos es $\{R, R, R, A, A\}$ por lo que $\hat{p} = \frac{3}{5}$ es el estimador de máxima verosimilitud de p .

OJO Dadas las observaciones (R, R, A) no podemos determinar a ciencia cierta cuál es la combinación de pelotas en la urna que tenemos. Por lo que siempre puede pasar que la combinación *real* sea distinta. Aquí el modelo nos dice por cuál optar (de manera lógica, aquella que genera con mayor probabilidad lo que observamos) pero la realidad ¡puede estar por otro lado!

Una vez que obtuvimos el estimador, \hat{p} , la segunda pregunta que nos interesará resolver es *qué tan buen estimador es \hat{p}* . Para ello también buscaríamos describir su distribución probabilística (su masa, su varianza, su valor esperado). Esto lo dejaremos para más adelante.

1.4 Resumen de la sección

En esta sección aprendimos que, en el mundo de las ideas, tenemos poblaciones de las cuales obtenemos subconjuntos (muestras). Las muestras observadas corresponden a los datos mientras que las muestras (sin el “observadas”) corresponden al modelo en el mundo de las ideas. Lo que buscamos es estimar los valores que necesitamos para determinar el modelo (parámetros) y esos valores se obtienen a través de estimadores (puede haber varios para el mismo parámetro). Los estimadores viven también en el mundo de los modelos y ahí tienen su distribución de probabilidad, sus masas y densidades, sus valores esperados y sus varianzas. Describir esto nos ayuda a saber cómo se comportan los estimadores y decidir cuáles son los buenos. Profundizaremos más en la siguiente sección donde haremos un ejemplo muy específico: ¿qué pasa si mi población está compuesta de variables aleatorias normales? Jugaremos con ello y veremos por qué es normal usar la normal.

Chapter 2

R

2.1 Programación en R

Una de las primeras cosas que necesitamos saber es que R (por más que sus más ávidos defensores digan lo contrario) no es para todo. Si tú ya conoces otro lenguaje (sea **Stata**, **Excel**, **SAS**, **Python**, **Matlab**, **Julia**, etc) sabrás utilizar muchas de sus opciones. Estoy seguro que, de conocer uno de estos, te será muchísimo más fácil seguir sacando promedios en tu lenguaje favorito que en R, realizar regresiones lineales es probablemente más sencillo en **Stata** mientras que las gráficas de barras para mí son más simples en **Excel**, **Python** excede en aplicaciones de inteligencia artificial mientras que **Matlab** es más veloz que R, **Julia** tiene muchas cosas de ecuaciones diferenciales que nadie más.

Lo que probablemente no sea más sencillo de hacer en otro lenguaje es realizar análisis estadístico, gráficas de todo tipo y modelos de simulación. Para eso, R es, indiscutiblemente, una de las mejores opciones para quienes no conocen de programación¹.

Finalmente, uno de los consejos más importantes que te puedo dar es que este curso no te va a servir si no practicas. Igual que como pasa con los idiomas uno no aprende R en una semana *sin practicarlo después*. Mi sugerencia es que, a la vez que sigues estas notas comiences a trabajar un proyecto *tuyo* específico junto con el buscador de Internet de tu preferencia a la mano y empieces a usar R en él. Practica².

¹Modelos de simulación más avanzados suelen hacerse en **C**, **C++** o **Fortran** por su velocidad; empero, es necesario conocer más de programación.

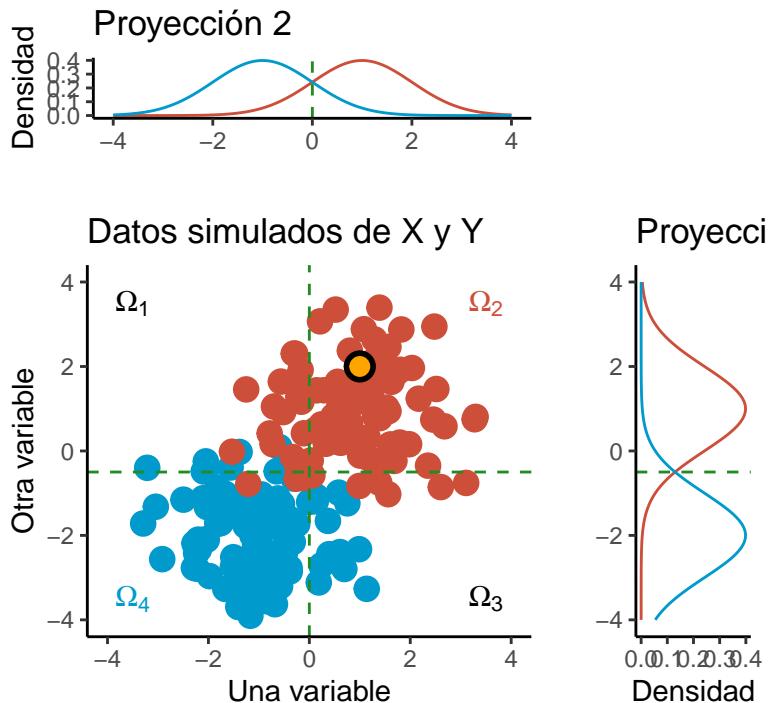
²La práctica hace al maestro



Figure 2.1: 'R' es un programa chido de estadística. FIN.

2.1.1 Puntos a favor de R

- Todo el mundo lo usa. Quizá éste es el punto más a favor. Si mucha gente lo conoce y lo utiliza, hay más opciones de ayuda. Los sitios de StackOverflow en inglés y en español son excelentes para pedir apoyo en R; los grupos de usuarios de Google son otra fuente muy buena. Entre más gente usa el programa; es más fácil obtener ayuda porque seguro alguien más tuvo hace ya tiempo el mismo problema que tú.
- Todas las personas que trabajan en estadística publican sus métodos y su código en R (eso, claro, cuando publican sus métodos). Es raro encontrar *un nuevo método estadístico* en el mundo y que no se pueda usar, de alguna forma, en R.
- Dentro de los lenguajes de programación R es de los más sencillos. Quienes lo hicieron realmente se preocuparon por su público (de no especialistas) y en general desarrollan para él.
- R es gratis. Y en esta época de austeridad, cualquier ahorro es bueno. Que sea gratis no significa que no esté respaldado: existen versiones de R respaldadas por grandes compañías como Microsoft
- Todo lo que se hace en R es público. R no tiene métodos secretos ni es una caja negra. Todo lo que hace cada una de las funciones de R, cualquiera lo puede revisar, por completo.
- En R puedes hacer libros o notas ¡como este! donde guardes todo tu trabajo, reportes automatizados e incluso documentos interactivos para facilitar el análisis de datos.
- R puede hacer gráficas bonitas:



RESULTADOS DE LA SIMULACIÓN

Por supuesto, no todo es miel sobre hojuelas con R. Particularmente, algunos de los problemas con el lenguaje:

- La curva de aprendizaje es mucho más empinada que para otros programas estadísticos (como Stata, SAS o SPSS) ¡particularmente si es tu primera vez programando!
- La mayor parte de las personas que trabajan en R no son programadores de verdad. Gran parte del código que te puedes encontrar **en el mundo real** está escrito con prisa para salir del aprieto sin mucha planeación, con pocos comentarios, falta de control de versiones y pocas herramientas de revisión. ¡Internet está lleno de criaturas espantosas escritas en R!
- R de ninguna manera es veloz por lo que algunos programas (lo veremos en simulación) pueden ser extremada (y dolorosamente) lentos.

2.1.2 Bienvenidx a R, Bunny-Wunnies Freak Out (sí, así se llama esta versión)

R es un lenguaje de cómputo y un programa estadístico libre, gratuito, de programación funcional (*¿qué es eso?*), orientado a objetos (*what??*) que mutó a

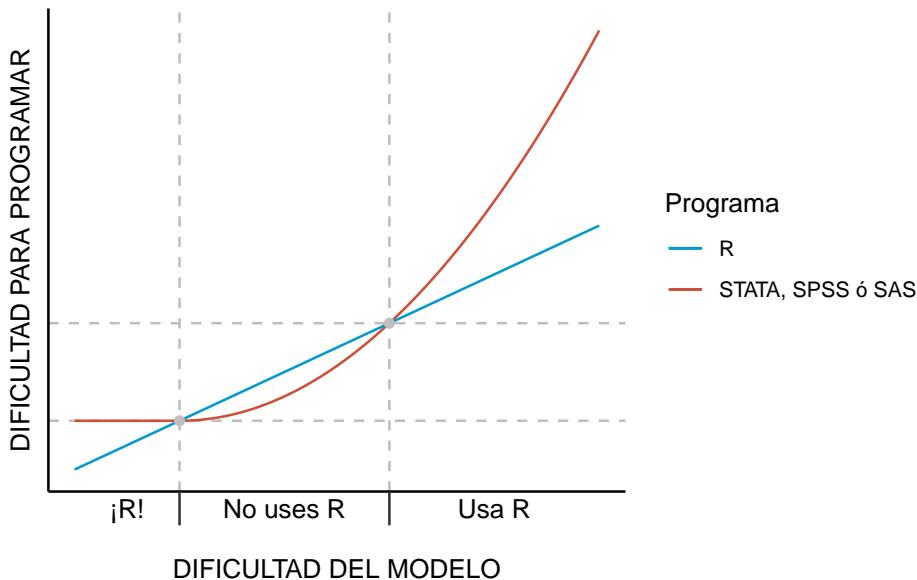


Figure 2.2: La curva de aprendizaje de ‘R’ es más empinada pero después de un rato vale la pena

partir de otros dos lenguajes conocidos como **Scheme** y **S**³. El primero de estos fue desarrollado en el MIT por Sussman y Steele mientras que el segundo surgió en los laboratorios Bell⁴ creado por Becker, Wilks y Chambers. R nació en junio de 1995 a partir del trabajo de Ross Ihaka y Robert Gentleman⁵.

Desde su creación, la mayor parte del desarrollo de R ha sido trabajo completamente voluntario de la Fundación R, del equipo de R Core y de miles de usuarios que han creado funciones específicas para R conocidas como paquetes (**packages**). Actualmente el repositorio más importante de R, CRAN, contiene más de 16000 paquetes con distintas funciones para hacer ¡lo que quieras!

Como todo el trabajo en R es voluntario hace falta:

1. Una homologación en los métodos. Puedes encontrar varias funciones *que supuestamente hacen exactamente lo mismo* (como es el caso de **emojifont**, **fontemoji** y **emoGG** para graficar usando emojis).
2. Estandarizar la notación. Algunos paquetes como aquellos del **tidyverse** (veremos más adelante) utilizan **pipes** (%>%); estos sólo funcionan en el **tidyverse** pero no fuera del mismo.

³De ahí que se llame R porque la R es una mejor letra que la S (todos lo sabemos) -Atte. Rodrigo, el autor de este documento.

⁴Mejor conocidos ahora como AT&T, la compañía celular que nunca tiene señal.

⁵Sus nombres empiezan con la letra R ¿coincidencia?

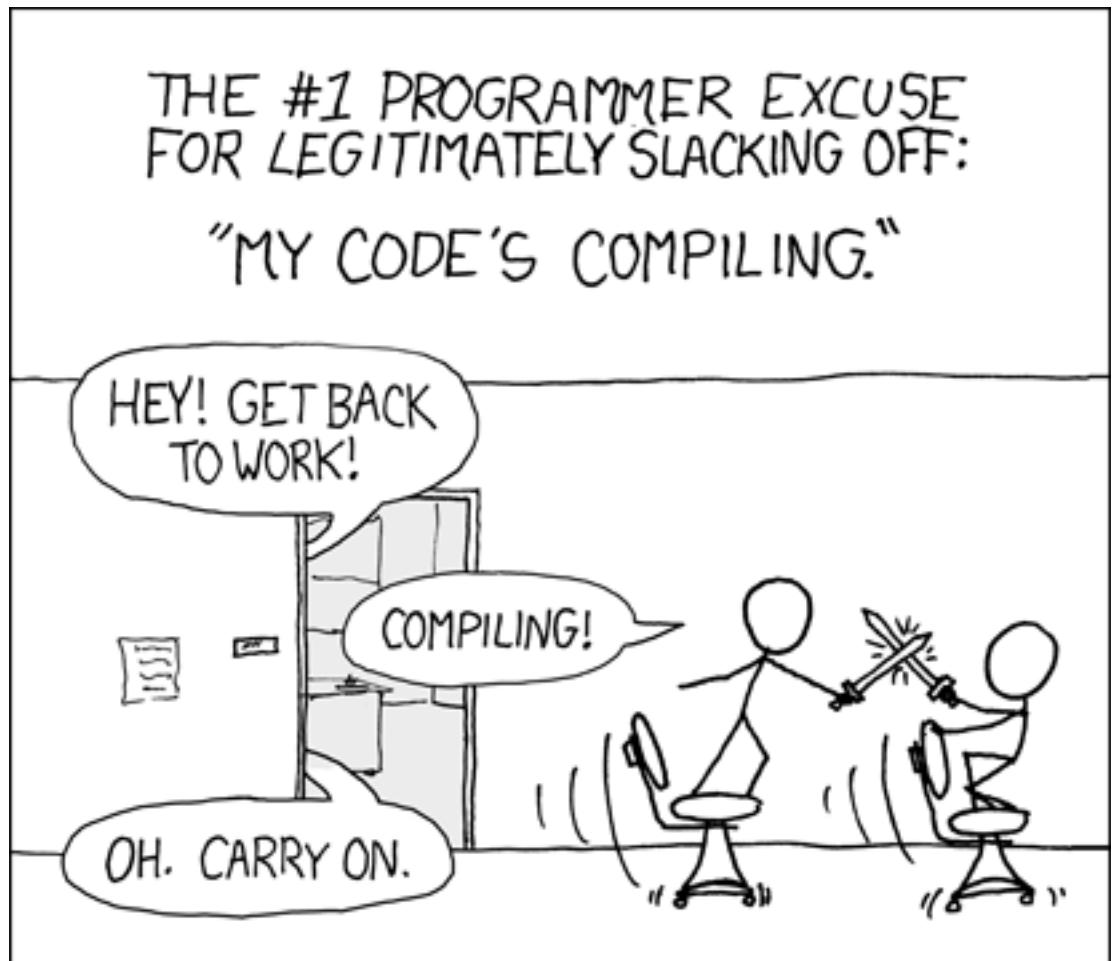


Figure 2.3: 'R' puede ser muy lento pero eso te da oportunidad de hacer otras cosas ;).

Sin embargo, también es una gran ventaja que sean los usuarios de R quienes guían su desarrollo. El lenguaje va mutando según peticiones de las personas que lo usan. Si hay algo que te gustaría R tuviera y aún no existe ¡lo puedes proponer!

2.2 Instalando cosas

2.2.1 Instalación de R

A lo largo de estas notas estaré trabajando con: R version 4.0.3 (2020-10-10) *Bunny-Wunnies Freak Out*. La más reciente versión de R la puedes encontrar en CRAN. Para ello ve al sitio y selecciona tu plataforma.

Nota usuarios de Mac En algunas Mac, al abrir R, aparece el siguiente mensaje de advertencia: `During startup - Warning messages: 1: Setting LC_CTYPE failed [...]` para solucionarlo ve a Aplicaciones y abre Terminal. Copia y pega en ella el siguiente texto: `defaults write org.R-project.R force.LANG en_US.UTF-8` Da enter, cierra la Terminal y reinicia R.

- En el caso de Windows da clic en [Download R for Windows](#) y luego en [install R for the first time](#). Finalmente, ejecuta el instalable que aparece al dar click en [Download R 4.0.3 for Windows](#).

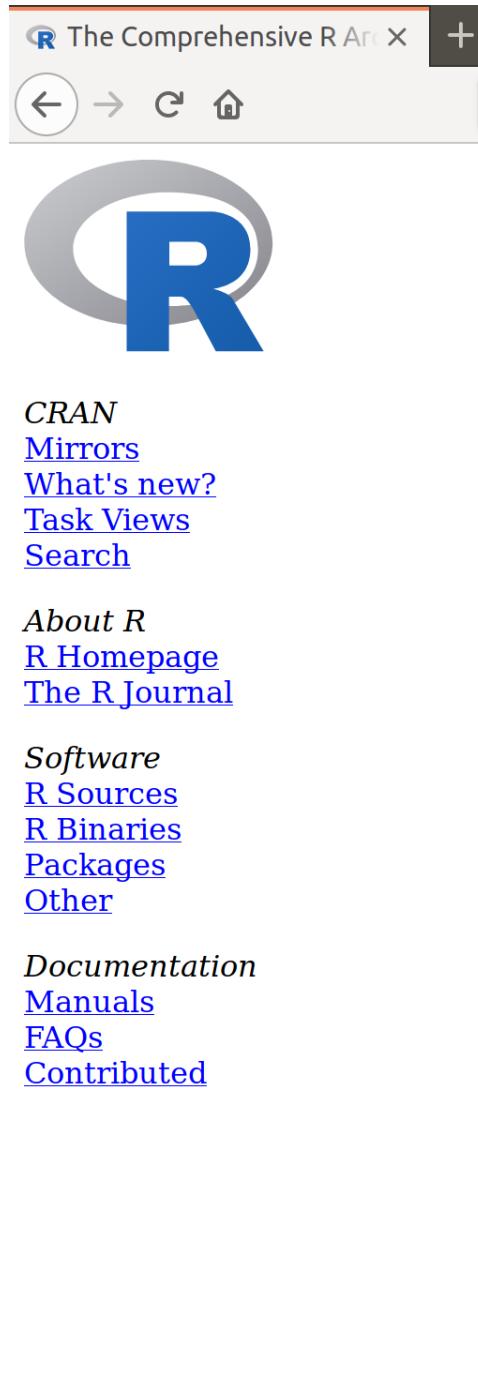
Para este curso pudiera ser que requirieras las herramientas de desarrollador Rtools.

- En el caso de Mac selecciona [Download R for \(Mac\) OS X](#) y luego elige `R-4.0.3.pkg`. En Mac puede que necesites instalar adicionalmente XQuartz (según tu versión de Mac). Si tu Mac es una versión suficientemente antigua, sigue las instrucciones específicas de CRAN.
- En el caso de Linux al elegir [Download R for Linux](#) tendrás la opción de buscar tu distribución específica. Al elegirla, aparecerán instrucciones para tu terminal de comandos; síguelas. En el caso de Linux, según los paquetes de R que elijamos instalar en la computadora requerirás instalar paquetería adicional para tu distribución de Linux. R te informará de la paquetería necesaria conforme la requiera.

Si tienes problemas para instalar puedes usar RStudio Cloud.

2.3 Instalación de RStudio

RStudio es una interfaz gráfica (IDE) para R. Puedes pensar a R como el *Bloc de Notas* y a RStudio como *Word*. El *Bloc* tiene todas las capacidades que necesitas para poder escribir; empero, es muchísimo mejor trabajar tus *papers* en *Word*. De la misma manera, R tiene todas las capacidades para hacer estadística pero un formato horrible y RStudio se ha convertido en la más popular forma de



The screenshot shows the official CRAN (Comprehensive R Archive Network) homepage. At the top, there's a navigation bar with icons for back, forward, search, and home, along with a URL field showing <https://cran.r-project.org>. Below the bar is the large R logo. To the right of the logo is a sidebar with several sections:

- Download and Install R**: Precompiled binary distributions are available for most platforms. Links include [Download R for Linux](#), [Download R for \(Mac\)](#), and [Download R for Windows](#).
- R is part of many Linux distributions**: A note stating that R is included in many Linux distributions, in addition to the link above.
- Source Code for all Platforms**: A note for Windows and Mac users mentioning that source code is available. It says "Windows and Mac users mostly want the source code. The source code means, you probably do not" and lists options for different releases.
- Documentation**: Links to [Manuals](#), [FAQs](#), and [Contributed](#) documentation.
- Questions About R**: A section with a bullet point: "If you have questions about R, please read our [Answers](#) page."

Figure 2.4: Oficialmente, la página de ‘R’ es de las páginas más feas del mundo.
¡No te dejes llevar por las apariencias!



Figure 2.5: RStudio es una empresa que se dedica a hacer cosas para R.

usar R. Por supuesto que no es la única; algunas alternativas son Atom con ide-r, Eclipse con StatET y RKWard. En general es posible seguir estas notas sin que tengas RStudio pero, si es tu primera vez programando, no lo recomiendo.

Si ya tienes experiencia con lenguajes como Python, Javascript, Java ó alguno de los mil C que existen, no tendrás ningún problema usando el editor de tu preferencia.

Para descargar RStudio ve a su página y da clic en **Download RStudio**. Baja tu pantalla hasta donde dice **Installers for Supported Platforms** y elige tu plataforma: **Windows**, **Mac OS X** ó tu sabor de **Linux** preferido. Una vez descargado el archivo, ábrelo y sigue las instrucciones que aparecen en pantalla.

2.4 Primeros pasos en R usando RStudio

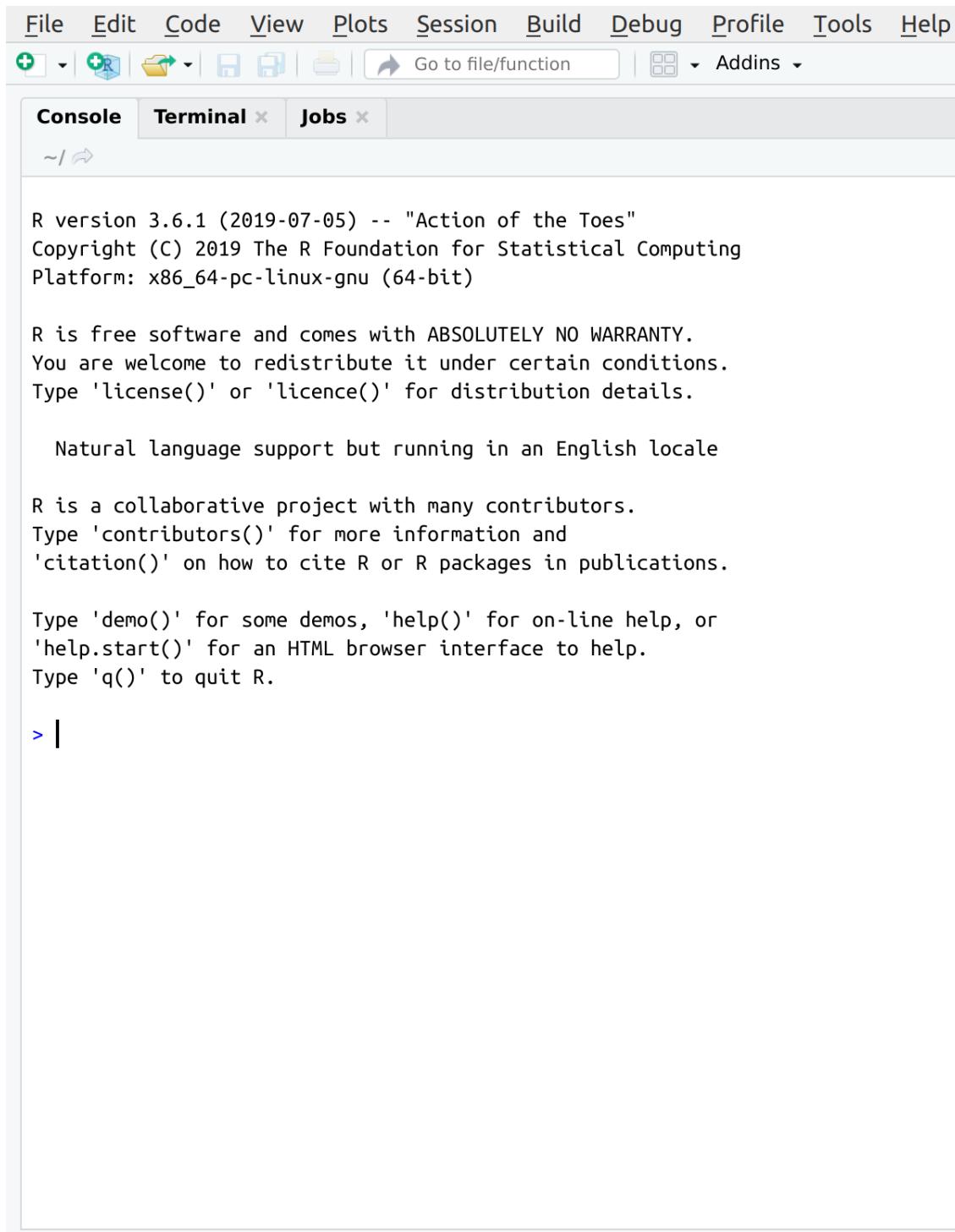
Una vez hayas instalado R y RStudio, abre RStudio⁶. Te enfrentarás a una pantalla similar a esta:

Si tu RStudio tiene sólo 3 páneles, como en mi caso, ve a la esquina superior izquierda (signo de hoja+) y elige un nuevo **R Script**

Tendrás, entonces, 4 páneles como se ve a continuación:

1. El primer panel (esquina inferior izquierda) es la **Consola**. Aquí es donde se ejecutan las acciones. Prueba escribir `2 + 3` en él y presiona enter. Aparece el resultado de la suma. Definitivamente, R es la calculadora que más trabajo cuesta instalar.
2. El segundo panel (esquina superior izquierda) es el panel con el **Script**. Aquí se escribe el programa pero no *se ejecuta*. Prueba escribir `10 + 9`. ¿Ves que no pasa nada? Lo que acabas de hacer es crear un programa que, cuando se ejecute, hará la suma de `10 + 9`. ¡Qué programa más aburrido! Sin embargo, no todo está perdido: presiona **CTRL+Enter** (**Cmd+Enter** en Mac) al final de la línea o bien da clic en **Run** y verás que, en la consola, aparece la instrucción y el resultado de la misma. El **Script** es una excelente fuente para tener un historial de lo que estás haciendo.
3. El tercer panel contiene el ambiente. Aquí aparecerán las variables que vayamos creando. Por ahora, para poner un ejemplo, importaremos el archivo `Example1.csv` (con valores simulados) disponible en Github dando clic en **Import Dataset** y **From Text (base)**. Selecciona el archivo y elige las opciones en la ventana de previsualización que hagan que se vea bien. Nota que una vez realizada la importación aparece en el panel derecho `Example1`. Al dar clic podrás ver la base de datos. Las bases de datos y variables que utilices durante tus análisis aparecerán en esa sección.
4. Para entender mejor lo que ocurre en el último de los páneles, lo mejor es trabajar con nuestra base. Escribe en la consola `plot(Example1)`. En el

⁶Si decidiste no instalar RStudio salta al final de esta sección.



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations like New, Open, Save, and Print. A search bar labeled "Go to file/function" is followed by an "Addins" dropdown. The main area has tabs for Console, Terminal, and Jobs, with "Console" selected. The console window displays the standard R startup message:

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"  
Copyright (C) 2019 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

A blue cursor arrow is visible at the bottom of the console window.

Figure 2.6: La primera vez que abres RStudio

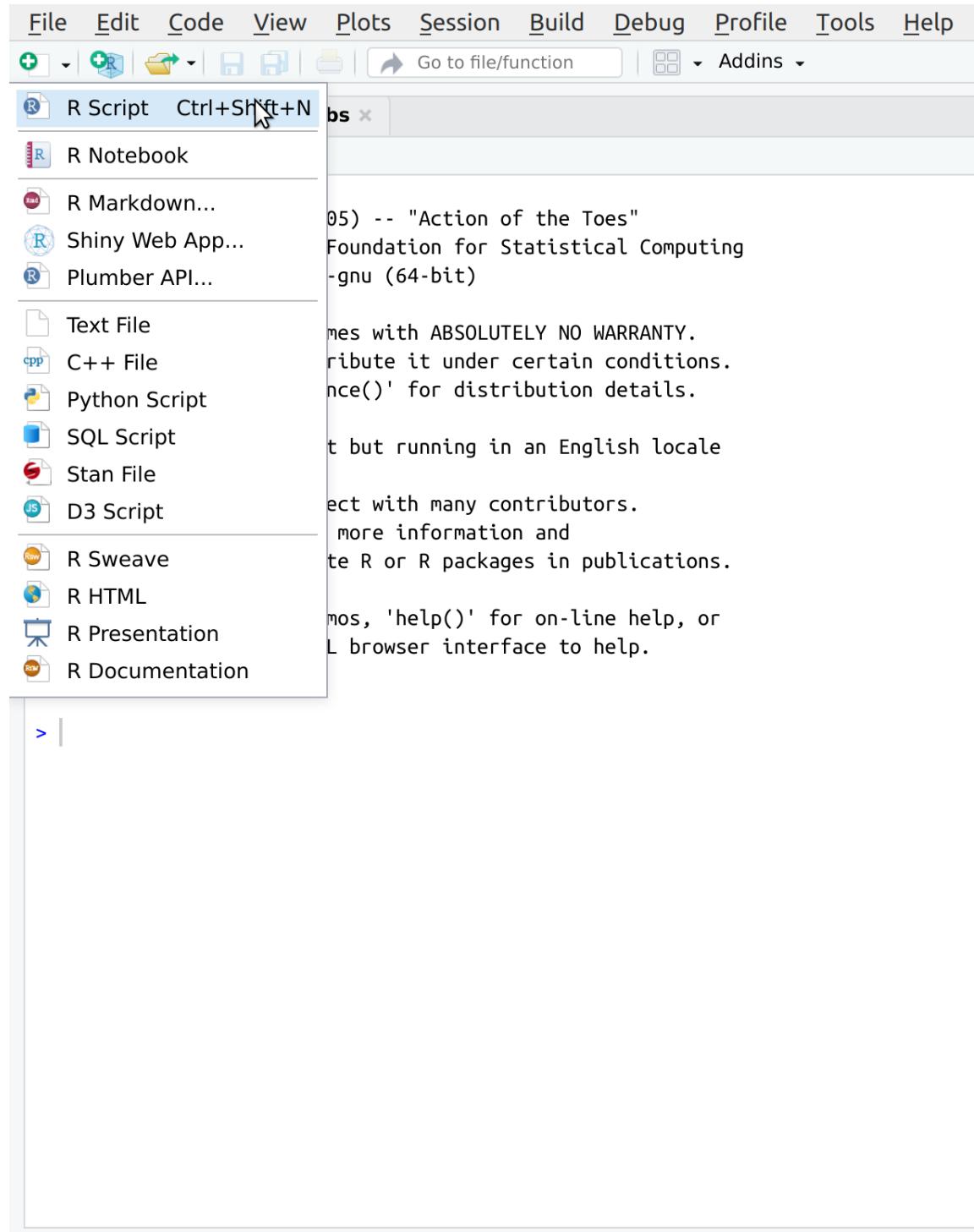


Figure 2.7: Elige hoja+ para crear un nuevo archivo

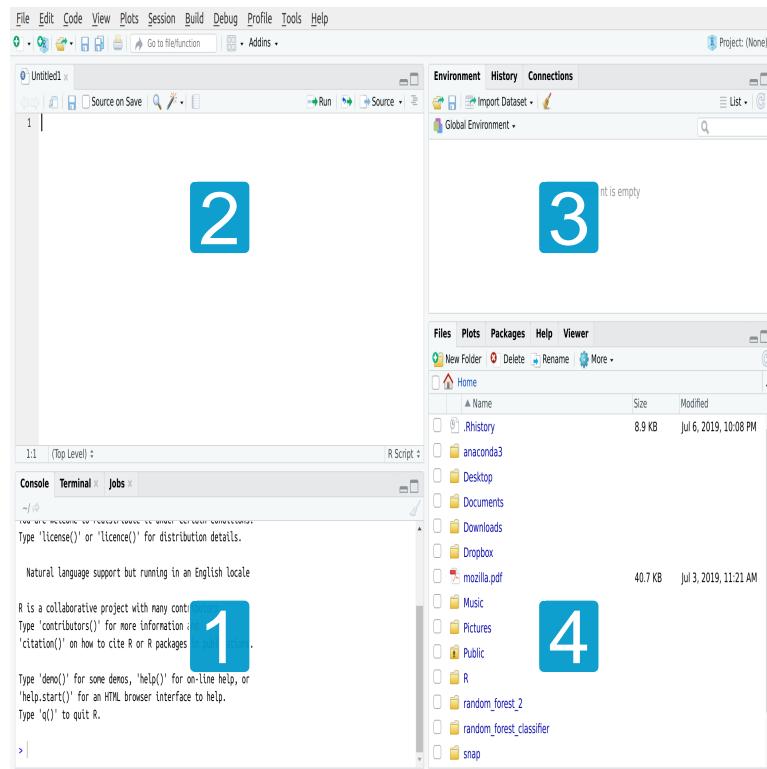


Figure 2.8: RStudio <3

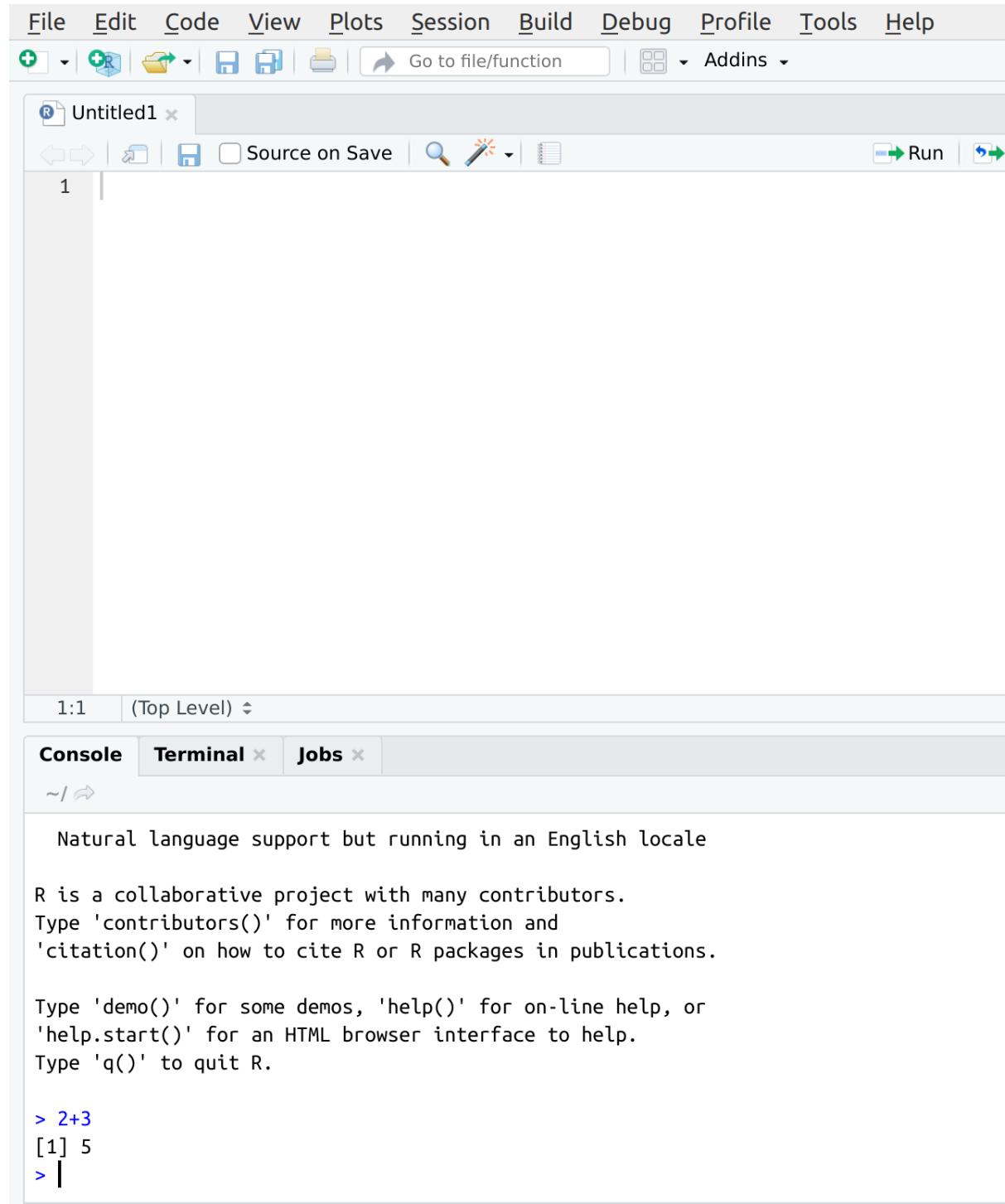


Figure 2.9: La consola de 'R' es la calculadora más difícil de instalar que existe.

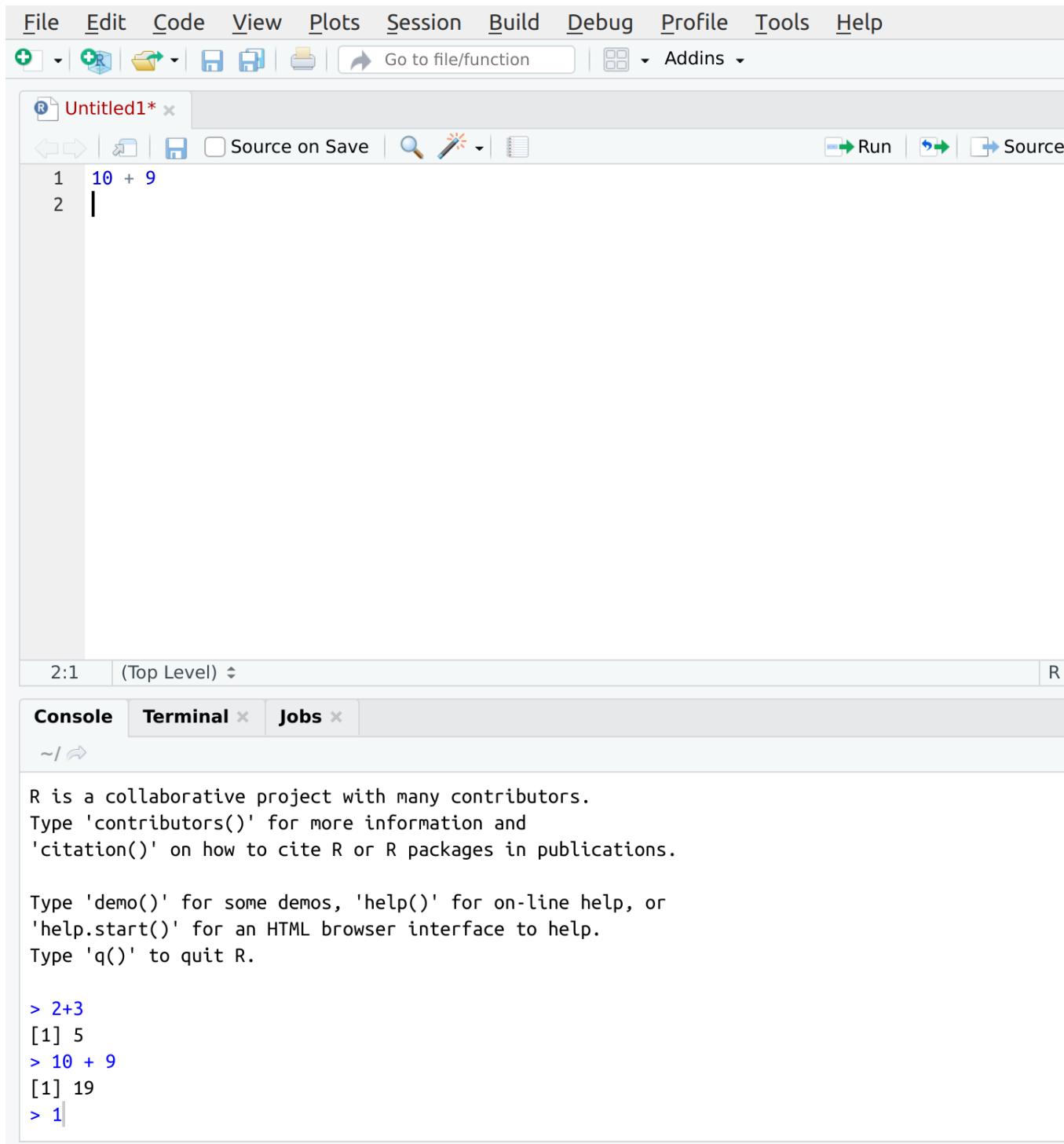


Figure 2.10: El ‘Script’ sirve para salvar las instrucciones en el orden en que las vas a ejecutar.

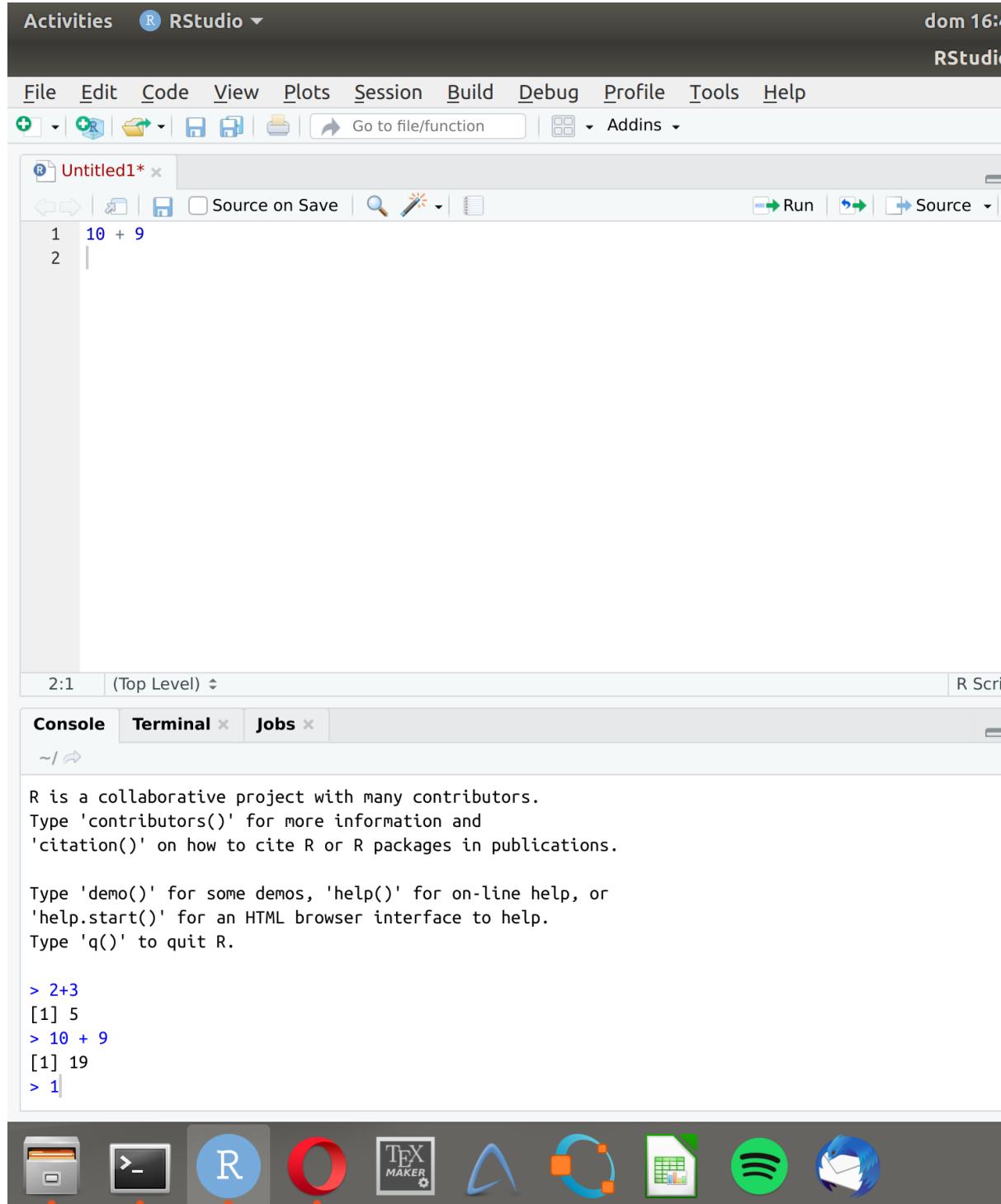


Figure 2.11: El ‘Ambiente’ muestra las variables (incluyendo bases de datos) que estás utilizando en este momento. A diferencia de otros programas estadísticos (o sea ‘Stata’) en ‘R’ es posible tener múltiples bases de datos abiertas a la vez.

cuarto pánel aparecerá una gráfica. El cuarto de los páneles para nosotros tendrá esa utilidad: mostrará las gráficas que hagamos así como la ayuda. Para ver la ayuda para las instrucciones de R puedes escribir `?.`. Prueba teclear `?plot` en la consola. El signo de interrogación es un `help()` que muestra las instrucciones para usar una función.

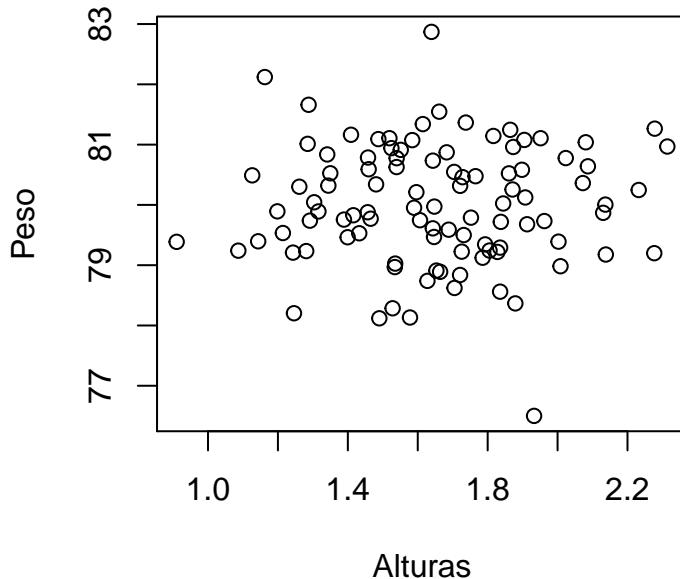


Figure 2.12: La gráfica que aparece de hacer un ‘plot’ de la base de datos de ejemplo.

Mi sugerencia personal es que escribas todo lo que haces en el `Script` y que sólo utilices la consola para verificar valores. De esta manera podrás almacenar todas las instrucciones ejecutadas y volver a ellas cuando se requieran. Por último te sugiero utilizar `# gatos` para comentar tu código. Así, el código anterior lo podrías ver en la consola como:

```
#Aquí pruebo cómo R hace las sumas
10 + 9
```

Comenta. Comenta. Comenta, por favor. Tu ser del futuro que regrese a sus archivos de R un mes después de haberlos hecho te lo agradecerá (y tu profe también).

Finalmente y como aclaración para estas notas, el código de R aparece como:

```
#Esto es código de R
7 - 2
```

Mientras que los resultados de evaluar en R se ven con `#:` [1] 5

Así, la evaluación con su resultado se ve de la siguiente forma:

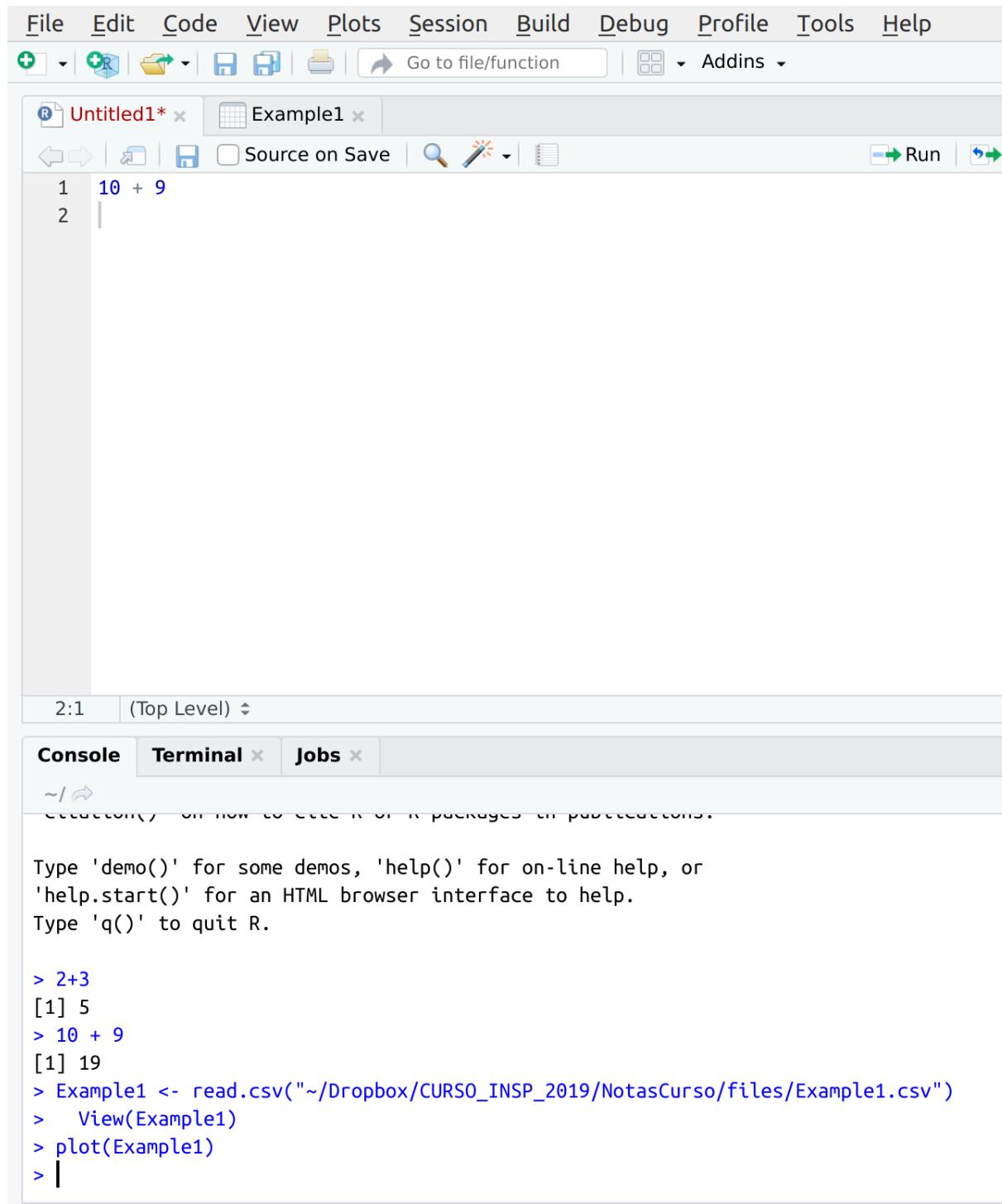


Figure 2.13: El cuarto panel muestra respectivamente las gráficas y la ayuda.

#Esto es código de R

7 - 2

[1] 5

2.5 Cálculos numéricos

R sirve como calculadora para las operaciones usuales. En él puedes hacer sumas,

#Esto es una suma en R

12 + 31

[1] 43

restas,

#Esto es una resta en R

3 - 4

[1] -1

multiplicaciones,

#Esto es una multiplicación en R

7*8

[1] 56

divisiones,

#Esto es una división en R

4/2

[1] 2

sacar logaritmos naturales \ln ,

#Para sacar logaritmo usas el comando \log

log(100)

[1] 4.60517

o bien logaritmos en cualquier base,⁷

#Puedes especificar la base del logaritmo con base

log(100, base = 10)

[1] 2

también puedes elevar a una potencia (por ejemplo hacer 6^3),

⁷Recuerda que un logaritmo base a te dice a qué potencia b tuve que elevar a para llegar a b . Por ejemplo $\log_{10}(100) = 2$ te dice que para llegar al 100 tuviste que hacer 10^2 .



Figure 2.14: Ada Lovelace (1815-1852), la primera en diseñar un algoritmo computacional ¡y sin tener computadoras!

#Así se calculan potencias

6^3

[1] 216

calcular la exponencial e ,

#Para exponenciales puedes usar `exp`
`exp(1)`

[1] 2.718282

o bien exponenciar cualquier variable e^{-3} ,

#O bien exponenciales específicas, e^{-3}
`exp(-3)`

[1] 0.04978707

también puedes usar el número π .

#Cálculo de π
`pi`

[1] 3.141593

No olvides que R usa el orden de las operaciones de matemáticas. Siempre es de izquierda a derecha con las siguientes excepciones:

1. Primero se evalúa lo que está entre paréntesis.
2. En segundo lugar se calculan potencias.
3. Lo tercero en evaluarse son multiplicaciones y divisiones.
4. Finalmente, se realizan sumas y restas.

Por ejemplo, en la siguiente ecuación

$$2 - 2 \cdot \frac{(3^4 - 9)}{(5 + 4)}$$

se resuelven primero los paréntesis $(3^4 - 9) = 81 - 9 = 72$ y $(5 + 4) = 9$; luego se resuelve la división: $\frac{72}{9} = 8$, se multiplica por el 2: $2 \cdot 8 = 16$ y finalmente se hace la resta: $2 - 16 = -14$.

2.5.1 Ejercicio

Determina, sin evaluar, los resultados de los siguientes segmentos de código:

#Primer ejercicio

$(9 - 3)^2 * (2 - 1) - 6$

```
#Segundo ejercicio
6 * 2 / (7 - 3) * 5

#Tercer ejercicio
2 * 3 ^ 2 * 2 / (5 - 4) * 1 / 10
```

Evalúa para comprobar tu respuesta.

2.5.2 Ejercicio

Calcula el área y el perímetro de un círculo de radio 5. Recuerda que la fórmula del área es $\pi \cdot r^2$ donde r es el radio; mientras que la del perímetro es: $\pi \cdot d$ donde d es el diámetro (= dos veces el radio).

2.5.3 Respuestas

Área = 78.5398163397448 Perímetro = 31.4159265358979

2.6 Variables

R es un programa orientado a objetos; esto quiere decir que R almacena la información en un conjunto de variables que pueden tener diferentes **clases** y opera con ellos según su clase. Por ejemplo, un conjunto de caracteres, entre comillas, es un **Character** (R lo piensa como texto)

```
#Un conjunto de caracteres es un char
"Hola"
```

[1] "Hola"

Un número (por ejemplo 2 tiene clase **numeric**)⁸. Hay que tener mucho cuidado con combinar floats con **Strings**:

```
#Código que sí funciona porque ambos son números
2 + 4
```

[1] 6

```
#Código que no funciona porque uno es carácter
2 + "4"
```

```
## Error in 2 + "4": non-numeric argument to binary operator
```

Si lo piensas, este último error ¡tiene todo el sentido! no puedes sumar un número a un texto. ¿O qué significaría 'Felices' * 4 ?

La magia de R comienza con que puedes almacenar valores en variables. Por ejemplo, podemos asignar un valor a una variable:

⁸Puede ser **float**, **int**, **double** pero no nos preocuparemos por eso.

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.					Working				
						IV_1	IV_2	IV_3	$0V_4$	$0V_5$	$0V_6$	$0V_7$	$0V_8$	$0V_9$	$0V_{10}$
						○	○	○	○	○	○	○	○	○	○
1	\times	$IV_2 \times IV_3$	IV_4, IV_5, IV_6	$\begin{cases} IV_2 = IV_2 \\ IV_3 = IV_3 \\ IV_4 = 2V_4 \end{cases}$	$= 2n$...	2	n	2n	2n	2n				
2	-	$IV_4 - IV_1$	$2V_4$	$\begin{cases} IV_1 = IV_1 \\ IV_5 = 2V_5 \end{cases}$	$= 2n - 1$	1	2n - 1						
3	+	$IV_5 + IV_1$	$2V_5$	$\begin{cases} IV_5 = 2V_5 \\ IV_1 = IV_1 \end{cases}$	$= 2n + 1$	1	2n + 1					
4	+	$2V_5 + 2V_4$	IV_{11}	$\begin{cases} 2V_5 = 0V_5 \\ 2V_4 = 0V_4 \end{cases}$	$= \frac{2n - 1}{2n + 1}$	0	0
5	\div	$IV_{11} + IV_2$	$2V_{11}$	$\begin{cases} IV_{11} = 2V_{11} \\ IV_2 = IV_2 \end{cases}$	$= \frac{1}{2} \cdot \frac{2n - 1}{2n + 1}$...	2	$\frac{1}{2}$
6	-	$0V_{13} - 2V_{11}$	IV_{13}	$\begin{cases} 2V_{11} = 0V_{11} \\ 0V_{13} = IV_{13} \end{cases}$	$= -\frac{1}{2} \cdot \frac{2n - 1}{2n + 1} = A_0$	
7	-	$IV_3 - IV_1$	IV_{10}	$\begin{cases} IV_3 = IV_3 \\ IV_1 = IV_1 \end{cases}$	$= n - 1 (= 3)$	1	...	n	$n - 1$
8	+	$IV_2 + 0V_7$	IV_7	$\begin{cases} IV_2 = IV_2 \\ 0V_7 = IV_7 \end{cases}$	$= 2 + 0 = 2$...	2	2		
9	\div	$IV_6 + IV_7$	$3V_{11}$	$\begin{cases} IV_6 = IV_6 \\ 0V_{11} = 3V_{11} \end{cases}$	$= \frac{2n}{2} = A_1$	2n	2	
10	\times	$IV_{21} \times 3V_{11}$	IV_{12}	$\begin{cases} IV_{21} = 3V_{21} \\ 3V_{11} = 3V_{11} \end{cases}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	
11	+	$IV_{12} + IV_{13}$	$2V_{13}$	$\begin{cases} IV_{12} = 0V_{12} \\ IV_{13} = 2V_{13} \end{cases}$	$= -\frac{1}{2} \cdot \frac{2n - 1}{2n + 1} + B_1 \cdot \frac{2n}{2}$	
12	-	$IV_{10} - IV_1$	$2V_{10}$	$\begin{cases} IV_{10} = 2V_{10} \\ IV_1 = IV_1 \end{cases}$	$= n - 2 (= 2)$	1	$n - 2$
13	-	$IV_6 - IV_1$	$2V_6$	$\begin{cases} IV_6 = 2V_6 \\ IV_1 = IV_1 \end{cases}$	$= 2n - 1$	1	2n - 1				
14	+	$IV_1 + IV_7$	$2V_7$	$\begin{cases} IV_1 = IV_1 \\ IV_7 = 2V_7 \end{cases}$	$= 2 + 1 = 3$	1	3			
15	\div	$2V_6 + 2V_7$	IV_8	$\begin{cases} 2V_6 = 2V_6 \\ 2V_7 = 2V_7 \end{cases}$	$= \frac{2n - 1}{3}$	2n - 1	3	$\frac{2n - 1}{3}$		
16	\times	$IV_8 \times 3V_{11}$	$4V_{11}$	$\begin{cases} IV_8 = 0V_8 \\ 3V_{11} = 4V_{11} \end{cases}$	$= \frac{2n}{2} \cdot \frac{2n - 1}{3}$	0	$\frac{2n}{2}$
17	-	$2V_6 - IV_1$	$3V_6$	$\begin{cases} IV_1 = IV_1 \\ IV_6 = 3V_6 \end{cases}$	$= 2n - 2$	1	2n - 2				
18	+	$IV_1 + 2V_7$	$3V_7$	$\begin{cases} IV_1 = IV_1 \\ 2V_7 = 3V_7 \end{cases}$	$= 3 + 1 = 4$	1	4			
19	\div	$3V_6 + 3V_7$	IV_9	$\begin{cases} 3V_6 = 3V_6 \\ 3V_7 = 3V_7 \end{cases}$	$= \frac{2n - 2}{4}$	2n - 2	4	$\frac{2n - 2}{4}$...	$\frac{2n - 2}{2}$
20	\times	$IV_9 \times 4V_{11}$	$5V_{11}$	$\begin{cases} IV_9 = 0V_9 \\ 4V_{11} = 5V_{11} \end{cases}$	$= \frac{2n}{2} \cdot \frac{2n - 1}{3} \cdot \frac{2n - 2}{4} = A_3$	0		
21	\times	$IV_{22} \times 5V_{11}$	$0V_{12}$	$\begin{cases} IV_{22} = IV_{22} \\ 0V_{12} = 2V_{12} \end{cases}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n - 1}{3} \cdot \frac{2n - 2}{3} = B_3 A_3$	
22	+	$2V_{12} + 2V_{13}$	$3V_{13}$	$\begin{cases} 2V_{12} = 0V_{12} \\ 2V_{13} = 3V_{13} \end{cases}$	$= A_0 + B_1 A_1 + B_3 A_2$	
23	-	$2V_{10} - IV_1$	$3V_{10}$	$\begin{cases} 2V_{10} = 3V_{10} \\ IV_1 = IV_1 \end{cases}$	$= n - 3 (= 1)$	1	$n - 3$
Here follows a repetition of Operations thirteen to twenty-three.															
24	+	$4V_{13} + 0V_{24}$	IV_{24}	$\begin{cases} 4V_{13} = 0V_{13} \\ 0V_{24} = IV_{24} \end{cases}$	$= B_7$	
25	+	$IV_1 + IV_3$	IV_3	$\begin{cases} IV_1 = IV_1 \\ IV_3 = IV_3 \end{cases}$	$= n + 1 = 4 + 1 = 5$	1	...	$n + 1$	0	0			
					by a Variable-card.										
					by a Variable card.										

Figure 2.15: El algoritmo diseñado por Ada Lovelace.

```
#Asignamos x = 10
x <- 10
```

Hay dos formas de asignar valores, una es con la flecha de asignación \leftarrow y otra con el signo de igual:

```
#Podemos asignar valores con el signo de =
y = 6
```

Nota que, cuando realizamos operaciones, la asignación es la última que se realiza:

```
#Aquí z = 106
z <- y + x^2
```

Los valores que fueron asignados en las variables, R los recuerda y es posible calcular con ellos:

```
#Podemos realizar una suma
x + y
```

```
[1] 16
#O bien podemos realizar una multiplicación
3*y - x
```

```
[1] 8
```

Podemos preguntarnos por el valor de las variables numéricas mediante los operadores == (sí, son dos iguales), != (que es un \neq), >, >=, <= y <:

```
#Podemos preguntarnos si x vale 4
x == 4
```

```
[1] FALSE
```

El operador de asignación también se puede utilizar al revés $2 \rightarrow x$
pero no lo hagas, por favor.

Nota que no estamos asignando el valor de x:

```
x
```

```
[1] 10
```

Podemos preguntarnos por diferencia:

```
x != 4
```

```
[1] TRUE
```

Así como por mayores, menores incluyendo posibles igualdades (*i.e.* los casos \geq y \leq)

```
#Nos preguntamos si x > y
x > y
```

[1] TRUE

```
#Nos preguntamos si x >= 10
x >= 10
```

[1] TRUE

```
#Nos preguntamos si y < 6
y < 6
```

[1] FALSE

```
#O bien si y <= 6
y <= 6
```

[1] TRUE

En todos los casos los resultados han sido TRUE ó FALSE. La clase de variables que toma valores TRUE ó FALSE se conoce como booleana. Hay que tener mucho cuidado con ellas porque, puedes acabar con resultados muy extraños:

```
#MALAS PRÁCTICAS, NO HAGAS ESTO
#Cuando lo usas como número TRUE vale 1
100 + TRUE
```

[1] 101

```
#MALAS PRÁCTICAS, NO HAGAS ESTO
#Cuando lo usas como número FALSE vale 0
6*FALSE
```

[1] 0

Aquí puedes encontrar una lista de malas prácticas en computación a evitar.

Finalmente, nota que es posible reescribir una variable y cambiar su valor:

```
#Aquí x vale 10, como antes
x
```

[1] 10

```
#Aquí cambianos el valor de x y valdrá 0.5
x <- 0.5
x
```

[1] 0.5

2.6.1 Ejercicios

Determina el valor que imprime R en cada caso, sin que corras los siguientes pedazos de código. Despues, verifica tu respuesta con R:

```
#Primer ejercicio
x <- 100
y <- 3
x > y

#Segundo ejercicio
z <- (4 - 2)^3
z <- z + z + z
z

#Tercer ejercicio
x <- 3
y <- 2
z <- x * y
x <- 5
y <- 10
z

#Cuarto ejercicio
variable1 <- 1000
variable2 <- 100
variable3 <- variable1/variable2 <= 10
variable3

#Quinto ejercicio
"2" - 2

#Sexto ejercicio
(0.1 + 0.1 + 0.1) == 0.3
```

2.6.2 NIVEL 3

Determina, sin correr el programa, qué regresa la consola en este caso

```
x <- 2
x <- 5 + x -> y -> x
x <- x^2
x
```

Comprueba con la consola tus resultados; puede que encuentres respuestas poco intuitivas.

2.7 Observaciones sobre la aritmética de punto flotante

Si hiciste el penúltimo ejercicio (el cual, obviamente hiciste y comprobaste con la consola) podrás haber notado una trampa. Analicemos qué ocurre; quizá hicimos mal la suma

```
#Veamos si este lado está mal
(0.1 + 0.1 + 0.1)
```

```
[1] 0.3
#O si éste es el que tiene la trampa
0.3
```

```
[1] 0.3
```

Aparentemente no hay nada malo ¿qué rayos le pasa a R? La respuesta está en la aritmética de punto flotante. Podemos pedirle a R que nos muestre los primeros 100 dígitos de la suma $0.1 + 0.1 + 0.1$:

```
#Veamos qué pasa con la suma
options(digits = 22) #Cambiemos dígitos
(0.1 + 0.1 + 0.1)    #Sumamos
```

```
[1] 0.3000000000000000444089
```

El comando `options(digits = 22)` especifica que R debe imprimir en la consola 22 dígitos. No más.

¡Ahí está el detalle! R no sabe sumar. En general, ningún programa de computadora sabe hacerlo. Veamos otros ejemplos:

```
4.1 - 0.1 #Debería dar 4
```

```
[1] 3.99999999999999555911
3/10      #Debería ser 0.3

[1] 0.29999999999999888978
log(10^(12345), base = 10) #Debería dar 12345
```

```
[1] Inf
```

El problema está en cómo las computadoras representan los números. Ellas escriben los números en binario. Por ejemplo, 230 lo representan como 11100110 mientras que el 7 es: 111. El problema de las computadoras radica en que éstas tienen una memoria finita por lo que números muy grandes como: 124765731467098372654176 la computadora hace lo mejor por representarlos eligiendo el más cercano:



Figure 2.16: Réplica de la Z3, la primer computadora con punto flotante (1941).

2.7. OBSERVACIONES SOBRE LA ARITMÉTICA DE PUNTO FLOTANTE61

```
#Nota la diferencia entre lo que le decimos a R  
#y lo que resulta  
x <- 124765731467098372654176  
x
```

```
[1] 124765731467098377420800
```

Un error de punto flotante en la vida real ocasionó en los años noventa, la explosión del cohete **Ariane** 5. Moraleja: hay que tener cuidado y respeto al punto flotante.

No olvides cambiar la cantidad de dígitos que deseas que imprima R en su consola de vuelta:

```
options(digits = 6) #Cambiamos dígitos
```

El mismo problema ocurre con números decimales cuya representación binaria es periódica; por ejemplo el $\frac{1}{10}$ en binario se representa como $0.0001100110011\overline{0011} \dots$. Como es el cuento de nunca acabar con dicho número, R lo trunca y almacena sólo los primeros dígitos de ahí que, cada vez que escribes 0.1, R en realidad almacene el 0.100000000000000055511 que es *casi lo mismo* pero no es estrictamente igual. Hay que tener mucho cuidado con esta inexactitud de las computadoras (inexactitud estudiada por la rama de Análisis Numérico) pues puede generar varios resultados imprevistos.

2.7.1 ¿Cómo checar un if?

En general lo que hacen las computadoras para comparar valores es que verifican que, en valor absoluto, el error sea pequeño. Recuerda que el valor absoluto de x , $|x|$, regresa siempre el positivo:

$$|4| = 4 \quad \text{y} \quad |-8| = 8$$

Para verificar que algo es más o menos 0.3 suele usarse el valor absoluto⁹ de la siguiente manera:

```
abs( (0.1 + 0.1 + 0.1) - 0.3 ) < 1.e-6
```

```
[1] TRUE
```

donde $1.e-6$ es notación corta para 0.000001 (también escrito como 1×10^{-6}). La pregunta que nos estamos haciendo es que si el error entre sumar 0.1+0.1+0.1 y 0.3 es muy pequeño < 0.000001 :

$$|(0.1 + 0.1 + 0.1) - 0.3| < 0.000001$$

⁹En R el comando **abs** toma el valor absoluto.

2.8 Leer y almacenar variables en R

Para terminar esta sección, aprenderemos cómo guardar variables en R. Para eso, el concepto de directorio es uno de los más relevantes. En general, en computación, el directorio se refiere a la dirección en tu computadora donde estás trabajando. Por ejemplo, si estás en una carpeta en tu escritorio de nombre “Ejercicios_R” probablemente tu directorio sea ‘~/Desktop/Ejercicios_R/’ (en Mac) o bien ‘~\Desktop\Ejercicios_R\’ en Windows¹⁰. La forma de saber tu directorio (en general) es ir a la carpeta que te interesa y con clic derecho ver propiedades (o escribir `ls` en la terminal Unix).

R tiene un directorio `default` que quién sabe dónde está (depende de tu instalación, generalmente está donde tu `Usuario`). Usualmente lo mejor es elegir un directorio para cada uno de los proyectos que hagas. Para ello si estás en `RStudio` puedes utilizar `Shift+Ctrl+H` (`Shift+Cmd+H` en Mac) o bien ir a `Session > Set Working Directory > Choose Directory` y elegir el directorio donde deseas trabajar tu proyecto. Pensando que elegiste el escritorio (`Desktop` en mi computadora) notarás que en la consola aparece el comando `setwd("~/Desktop")` (o bien con ‘\’ si eres Windows). Mi sugerencia es que copies ese comando en tu `Script` para que, la próxima vez que lo corras ya tengas preestablecido el directorio.

```
#Si eres Mac/Linux
setwd("~/Desktop")

#Si eres Windows
setwd("C:\Users\Rodrigo\Desktop") #Rodrigo = Mi usuario
```

Podemos verificar el directorio elegido con `getwd()`:

```
getwd()
```

En general es buena práctica en R establecer, hasta arriba del `Script`, el comando de directorio. Esto con el propósito de que, cuando compartas un archivo, la persona a quien le fue compartido el archivo pueda rápidamente elegir su propio directorio en su computadora.

Probemos guardar unas variables en un archivo dentro de nuestro directorio. Para ello utilizaremos el comando `save`.

```
#Crear las variables
x <- 200
y <- 100

#Los archivos de variables de R son rda
save(x,y, file = "MisVariables.rda")
```

¹⁰Windows usa backslash. Y hay toda una historia detrás de ello

Si vas a tu directorio, notarás que el archivo `MisVariables.rda` acaba de ser creado. De esta forma R puede almacenar objetos creados en R que sólo R puede leer (más adelante veremos cómo exportar bases de datos y gráficas). Observa que en tu ambiente (si estás en RStudio puedes verlas en el panel 3) deben aparecer las variables que hemos usado hasta ahora:

```
[1] "x" "y" "z" "tamaño.muestra" [5] "Example1" "muestra" "pgorro" "i"
[9] "nsim" "poblacion" "conteo_rojos"
```

Podemos probar sumar nuestras variables y todo funciona súper:

```
x + y #Funciona magníficamente
```

```
[1] 300
```

Limpiemos el ambiente. El comando equivalente al `clear all` en R es un poco más complicado de memorizar:

```
#EL clear all de R
rm(list = ls())
```

Ahora, si vuelves a ver el ambiente, éste estará vacío: ¡hemos limpiado el histórico! Nota que si intentamos operar con las variables, R ya no las recuerda:

```
x + y #Error
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

Así como hay que lavarse las manos antes de comer, es buen hábito limpiar todas las variables del ambiente de R antes de usarlo.

Podemos leer la base de datos usando `load`:

```
#Leemos las variables
load("MisVariables.rda")
```

```
#Una vez leídas podemos empezar a jugar con ellas
x + y #Ya funciona
```

```
[1] 300
```

Por último, es necesario resaltar la importancia del directorio. Para ello crea una nueva carpeta en tu escritorio de nombre `Mi_curso_de_R`. Mueve el archivo `"MisVariables.rda"` dentro de la carpeta. Borra todo e intenta leer de nuevo el archivo:

```
#Borraremos todo
rm(list = ls())
```

```
#Intentaremos leer el archivo de nuevo
load("MisVariables.rda")
```

Este error es porque R sigue pensando que nuestro directorio es el escritorio y está buscando el archivo ahí sin hallarlo. Para encontrarlo hay que cambiar el directorio a través de RStudio (ya sea **Ctrl+Shift+H** o **Session >Set Working Directory > Choose Directory**) o bien a través de comandos en R:

```
#Si eres Mac/Linux
setwd("~/Desktop/Mi_curso_de_R")

#Si eres Windows
setwd("C:/Users/Rodrigo/Desktop/Mi_curso_de_R") #Rodrigo = Mi usuario

#Aquí sí se puede leer
load("MisVariables.rda")
```

2.8.1 Ejercicio

Responde a las siguientes preguntas:

1. ¿Qué es el directorio y por qué es necesario establecerlo?
2. Si R me da el error '`No such file or directory`' ¿qué hice mal?
3. En RStudio, ¿qué hace `Session > Restart R`? ¿cuál es la diferencia con `rm(list = ls())`?
4. ¿Qué hace el comando `cat("\014")`? (*Ojo* puede que no haga nada). Si funciona, ¿cuál es la diferencia con `rm(list = ls())` y con `Restart R`?

2.9 Instalación de paquetes

Un paquete de R es un conjunto de funciones adicionales elaboradas por los usuarios, las cuales permiten hacer cosas adicionales en R. Para instalarlos requieres de una conexión a Internet (o bien puedes instalarlos a partir de un archivo, por ejemplo, mediante una USB). El comando de instalación es `install.packages` seguido del nombre del paquete. Por ejemplo (y por ocio) descarguemos el paquete `beepR` para hacer reproducir sonidos en la computadora¹¹. Para ello:

```
install.packages("beepR")
```

```
[...]
* DONE (beepR)
```

```
The downloaded source packages are in
  '/algun/lugar/downloaded_packages'
```

¹¹En los siguientes capítulos descargaremos paquetes más interesantes; pero no desprecies la utilidad de `beepR` yo lo he usado en múltiples ocasiones para que la computadora me avise que ya terminó de correr un código.

Esto significa que el paquete ha sido instalado. Nos interesa usar la función `beep` que emite un sonido (`??beep` para ver la ayuda). Si la llamamos así tal cual, nos da error:

```
beep(3)
```

R es incapaz de hallar la función porque aún no le hemos dicho dónde se encuentra. Para ello podemos llamar al paquete mediante la función `library` y decirle a R que incluya las funciones que se encuentran dentro de `beepr`:

```
library(beepr)
beep(3) #Esto produce un sonido
```

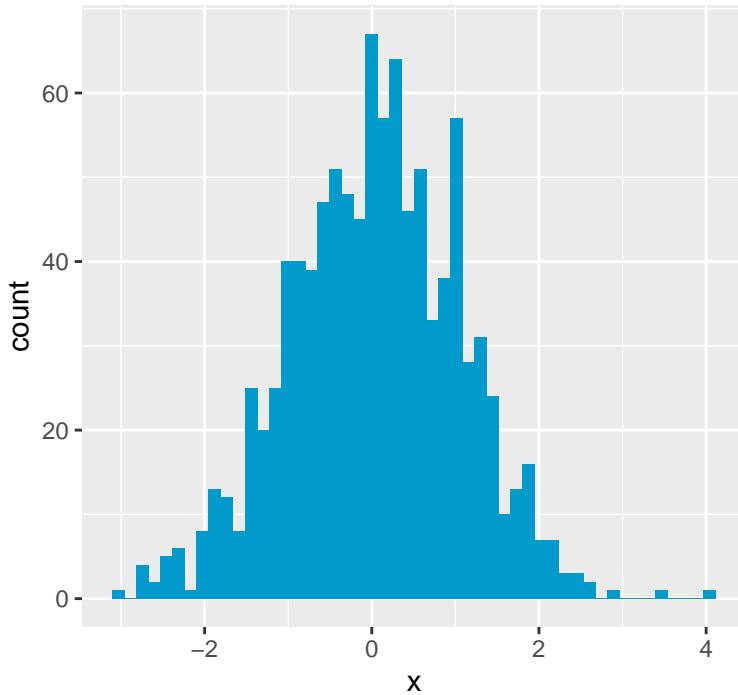
El comando `library` le dice a R ¡hey, voy a usar unas funciones que creó alguien más y que están dentro del paquete `beepr`! De esta manera, al correr `beep(3)`, R ya sabe dónde hallar la función y por eso no arroja error.

2.9.1 Ejercicios

NIVEL 1

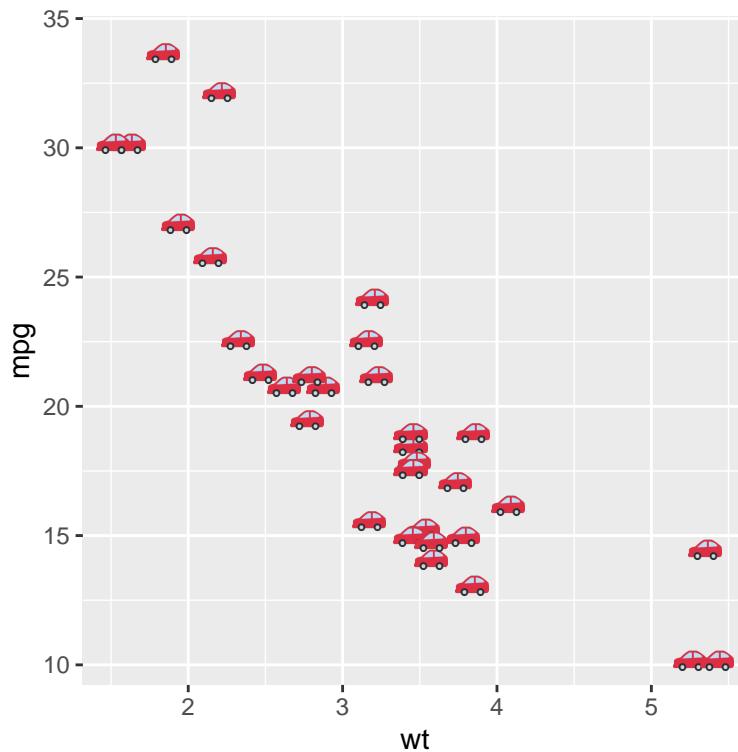
- Instala los paquetes `tidyverse` en R.
- De `tidyverse` haz lo necesario para que el siguiente bloque de código te arroje una gráfica:

```
#Aquí tienes que hacer algo
#
# RELLENA AQUÍ
#
#Esto genera un histograma
set.seed(1364752)
mis.datos <- data.frame(x = rnorm(1000))
ggplot(mis.datos, aes(x = x)) +
  geom_histogram(bins = 50, fill = "deepskyblue3") +
  ggtitle("Histograma generado por el código")
```

Histograma generado por el código**NIVEL 3**

1. Instala el paquete `devtools` (para hacerlo probablemente necesites instalar más cosas en tu computadora; averigua cuáles)
2. Usa `devtools` para instalar el paquete `emoGG` desde Github.
3. Verifica que tu instalación fue correcta haciendo la siguiente gráfica:

```
library(emoGG)
ggplot(mtcars, aes(wt, mpg)) + geom_emoji(emoji="1f697")
```



2.10 Comentarios adicionales sobre el formato

Así como en el español existen reglas de gramática para ponernos todos de acuerdo y entendernos entre todos, en R también existen *sugerencias* a seguir para escribir tu código. Las sugerencias que aquí aparecen fueron adaptadas de las que utiliza el equipo de Google.

1. No escribas líneas de más de 80 caracteres (si se salió de tu pantalla, mejor continúa en el siguiente renglón).
2. Coloca espacios entre operadores +, *, /, -, <-, =, <, <=, >, >=, == y usa paréntesis para agrupar:

```
#Esto no se ve muy bien
abs(3*5/(4-9)^2-60/100-888+0.1*8888-4/10*2) < 1.e-6
```

```
#Los espacios permiten distinguir el orden de las operaciones
abs( (3 * 5) / (4 - 9)^2 - 60 / 100 - 888
    + (0.1 * 8888) - (4 / 10) * 2 ) < 1.e-6
```

3. Intenta alinear la asignación de variables para legibilidad:

```
#Esto no tanto
altura <- 1.80
peso <- 80
edad <- 32

#Esto se ve bien
altura <- 1.80
peso   <- 80
edad   <- 32
```

4. Utiliza nombres que evoquen la variable que representas

```
#Cuando regreses a esto no sabrás ni qué
x <- 10
y <- 2
z <- 3.14
W <- z * x^y #¿Qué calculé?

#Es mejor especificar la variable
radio      <- 10
potencia   <- 2
pi_aprox   <- 3.14
area_circulo <- pi_aprox * radio^potencia
```

5. No utilices un nombre demasiado similar para cosas diferentes.

```
#Aquí, seguro eventualmente te vas a equivocar
altura <- 10  #Altura del edificio
Altura <- 1.8 #Mi altura
ALTURA <- 2000 #La altitud de la CDMX

#Siempre elegir nombres claros, aunque largos
altura.edificio <- 10  #Altura del edificio
altura.Rodrigo <- 1.8 #Mi altura
altura.CDMX    <- 2000 #La altitud de la CDMX
```

6. Comenta:

```
#¿Qué hace esto?
x <- 168
x <- x/100
y <- 71.2
print(y/x^2)

#Es mejor así
altura <- 168          #en centímetros
altura <- altura/100    #en metros
peso   <- 71.2          #peso en kg
```