Módulo 1. Cruce de bases de datos y funciones

BARÇA INNOVATION HUB
Universitas

- Funciones
- Referencias

Funciones

Se ha descrito con anterioridad el concepto de función, y también se han visto múltiples ejemplos en el material audiovisual; pero vamos a desarrollar con más detalle esta funcionalidad de RStudio, ya que tiene gran potencial para el tratamiento de datos y permite gran flexibilidad para el usuario.

- **Funciones**: elementos que desarrollan tareas en R. Exactamente igual que en Excel, usamos el nombre de la función, por ejemplo, SUM para las sumas, añadimos los valores que queramos sumar y nos da un resultado después de ejecutar el código.

Esta definición forma parte del primer módulo del primer curso, simplemente para describir con términos sencillos cuál es el objetivo de la función. Podemos visualizar esta definición de manera esquemática:

Figura 1. Definición de función

Fuente: elaboración propia

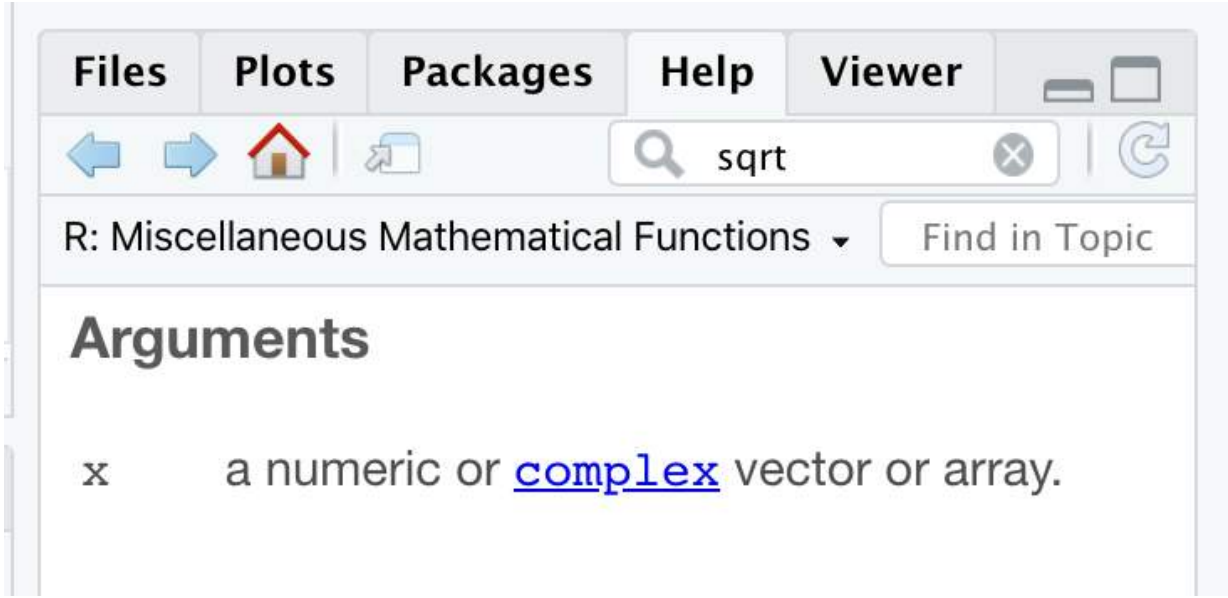
Describimos *input* como aquellos valores (hemos visto que pueden ser columnas de datos, valores únicos, listas, etc.) que indicaremos en las funciones y *output* como el resultado final.

Si queremos dar una definición más técnica a las funciones, podríamos decir que es aquel código que utiliza *inputs* o argumentos para realizar cálculos y devolver un *output* o resultado.

Veamos con más detalle las características de las funciones.

Vamos a utilizar la función sqrt(). Esta función permite calcular la raíz cuadrada del valor numérico que le indiquemos.

Figura 2. Función sqrt()

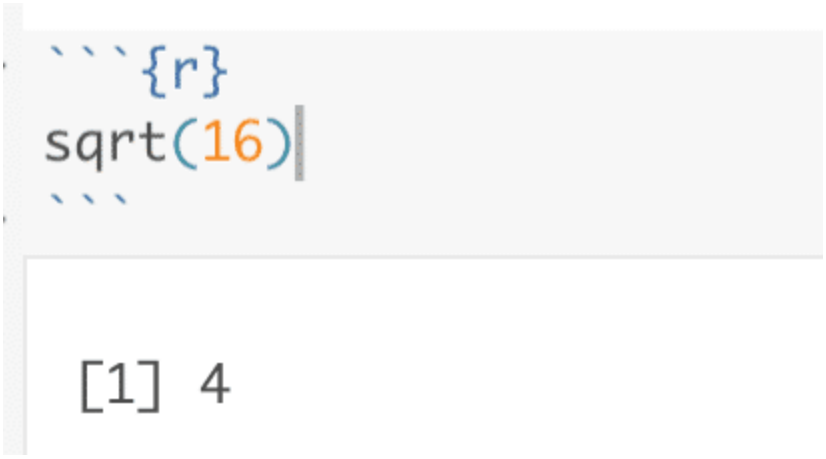


Fuente: captura de pantalla de RStudio (Posit, 2011).

En la documentación de la función, vemos cómo esta función únicamente necesita un argumento o *input* para devolver este resultado. El *input* puede ser un valor numérico o estructuras que contengan múltiples valores.

Para utilizar la función deberemos escribir en el código el nombre de esta y, entre paréntesis, definir los argumentos que queramos utilizar.

Figura 3. Función sqrt()



Fuente: captura de pantalla de RStudio (Posit, 2011)

En este ejemplo, utilizando la función sqrt(), podemos ver como el input es «16» y la función automáticamente nos da el output «4».

Existen dos tipos de funciones:

- **Incorporadas en RStudio.** Forman parte de las funcionalidades de RStudio y pueden ser usadas manteniendo la estructura que indica la documentación de cada una de ellas.

- **Creadas por el usuario**. Crearemos funciones que ayuden a automatizar los cálculos que queramos realizar de manera repetitiva.

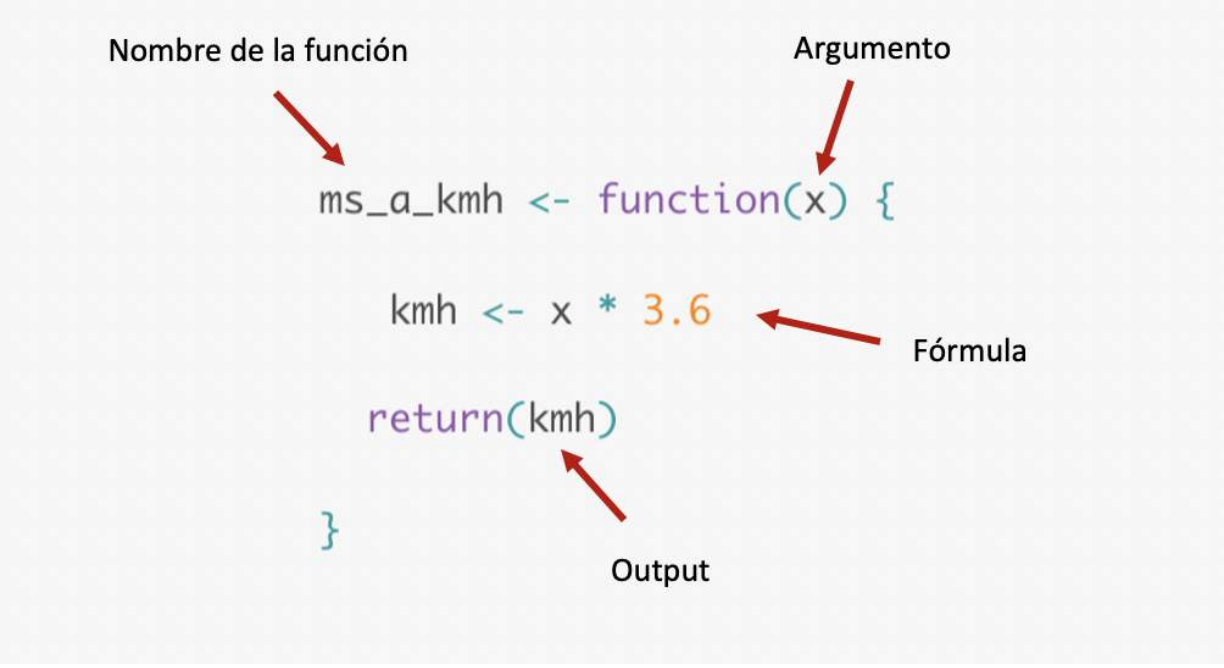
RStudio permite crear funciones siguiendo el patrón que queramos definir, de esta manera, podremos utilizarlas a lo largo de nuestro código para hacer cálculos que creamos necesarios y que no estén dentro de las funcionalidades de RStudio.

Veamos un ejemplo: queremos transformar los valores de metros por segundo a kilómetro por hora, ya que es muy común tener valores en distintas unidades, dependiendo del *software* que utilicemos; además, si queremos comunicar la información, expresar velocidades en kilómetros por hora es más comprensible para ciertos miembros del *staff*.

Para crear funciones debemos utilizar la estructura que requiere RStudio:

- Debemos escoger un nombre para la función
- Utilizar la estructura function(){}
- Especificar el argumento/s
- Crear la fórmula que necesitemos, utilizando los argumentos definidos
- Especificar el *output*, ya que la fórmula o código que utilicemos puede tener múltiples partes.

Figura 3. Creación de funciones en RStudio



Fuente: adaptación propia con base en captura de pantalla de RStudio (Posit, 2011)

En la imagen podemos ver cómo hemos decidido el nombre de la función «ms_a_kmh» y hemos especificado que únicamente utilizará un argumento «x».

- Este argumento puede especificarse con cualquier otro valor (puede ser «x» o «ms», por ejemplo).

- El argumento, tal y como lo hemos definido dentro del paréntesis, debe estar presente en la fórmula. Si en lugar de «x» lo hubiéramos llamado «ms», en la fórmula deberíamos reemplazar la «x» por «ms».

Finalmente, definimos que el *output* será el resultado del cálculo que indicamos en la fórmula.

De ahora en adelante, para utilizar esta función escribiremos «ms_a_kmh()» en el script para realizar los cálculos.

Figura 4. Creación de funciones en RStudio. Ejemplo

Fuente: captura de pantalla de RStudio (Posit, 2011)

En el ejemplo anterior, convertimos 8m/s (*input*) a km/h (*output*), utilizando la función previamente creada.

Ventajas de utilizar funciones:

- Permite automatizar cálculos, es decir, reutilizar código de manera más simple.
- Reduce la posibilidad de cometer errores escribiendo el código.

En los videos podremos ver ejemplos de creación de variables con múltiples argumentos y de características más complejas.

En R, las funciones:

- ◯ son elementos que desarrollan tareas específicas. Al igual que en Excel, usamos el nombre de la función, por ejemplo, sum() para realizar sumas.
- ◯ no pueden realizar tareas como en Excel. Por ejemplo, no podemos usar una función llamada SUM para realizar sumas. En lugar de eso, debemos escribir el código completo manualmente para sumar los valores sin la ayuda de funciones predefinidas.

SUBMIT

CARACTERÍSTICAS PRINCIPALES DEL CRUCE DE BASES DE DATOS

Destacamos en el curso anterior algunas de las características del proceso de extracción, transformación y carga de datos. Este proceso consiste en almacenar de forma adecuada los datos recogidos en un contexto particular para que puedan ser fácilmente accesibles y en cierta medida preparados para el análisis.

Cuando hablamos de almacenamiento, nos referimos a bases de datos. Estas bases de datos son conjuntos de datos organizados que habitualmente se guardan en la nube. Esta organización se realiza mediante tablas que pueden estar o no relacionadas entre ellas. En el contexto del *sport scientist*, es común utilizar este tipo de formatos de almacenamiento de datos para obtener la información en la frecuencia deseada (diario, semanal, etc.) y realizar los análisis pertinentes.

Estos formatos ofrecen un gran número de ventajas que repercuten en la eficiencia del trabajo y el acceso a mayor volumen de información:

- **Datos históricos acumulados.** Factor diferencial en organizaciones deportivas, disponer de datos de temporadas anteriores es una gran ventaja para evaluar procedimientos y cambios longitudinalmente. Una problemática frecuente es que los clubes cuando cambian *staff* pierden toda la información recogida por el cuerpo técnico anterior, ya que no disponen de sistemas de almacenamiento de datos automatizados.
- **Añadir nuevas variables/datos**: la estructura de las bases de datos permite flexibilidad para almacenar y crear nuevas variables según las necesidades de la organización.
- **Actualización de los datos**: en el caso de que haya cambios en el procesamiento de los datos, debido a actualizaciones de los distintos sistemas que estemos usando y las variables necesiten ser recalculadas, podremos realizar esta operación sobre los datos almacenados en el pasado.
- **Control de cambios**: cualquier cambio en las tablas que realicemos quedará registrado, así como las versiones anteriores. Esta funcionalidad permite que, si se ha cometido un error en la configuración, podremos volver a recuperar los datos de la versión anterior.
- **Acceso múltiple a la información**: distintos usuarios podrán usar la información al mismo tiempo, de manera segura y rápida.

Sin embargo, en muchos contextos no existe la posibilidad de tener acceso a una base de datos, pero disponemos de herramientas que permiten tener

funcionalidades parecidas, pero a menor escala. Hablamos de los libros de Excel, o archivos de texto. Si queremos garantizar el mejor funcionamiento de este tipo de formatos de almacenamiento, deberemos respetar los principios que destacamos en el curso anterior para la organización de las tablas de datos.

En la analítica de datos y en el contexto del rendimiento físico deportivo, es habitual la utilización de más de una tabla de datos para realizar análisis. Es muy probable que una sola tabla de datos no permita disponer de toda la información que necesitemos. Existe la posibilidad de elaborar una tabla con todos los datos que queramos utilizar a diario, pero este sistema puede presentar limitaciones a largo plazo. Veamos un ejemplo común en nuestro contexto.

Queremos disponer de una tabla de datos que nos permita hacer análisis relacionados con la carga física del jugador, en sesiones específicas o a lo largo de la temporada/temporadas. Una fuente de datos habitual para este tipo de análisis son los datos que nos provee el GPS. Cuando descargamos estos datos del *software* específico del sistema que estemos utilizando, la información que provee dicha descarga a menudo no es completa. Obtendremos información como la siguiente:

- Fecha
- Ejercicio/partido
- Duración
- Posición predeterminada
- Nombre del jugador
- Variable locomotora 1 (por ejemplo: distancia en metros)
- Variable locomotora 2 (por ejemplo: m/min)
- Variables 3,4,5...

Estos datos son suficientes para ciertos tipos de análisis, lo vimos en cursos anteriores, pero en el momento que necesitemos información complementaria, será necesario añadir otras variables. Imaginemos que

queremos hacer un estudio longitudinal a lo largo de la temporada, donde queremos analizar los cambios en desempeño condicional a lo largo de las jornadas, o la comparativa según la competición o el nivel del rival. Para obtener esa información, en la tabla de datos deberíamos añadirla manualmente después de cada sesión. Lo mismo pasaría si las posiciones en las que juega cada uno de los jugadores fueran variables a lo largo de las jornadas, aunque en el *software* de los dispositivos GPS tuviéramos posiciones indicadas para cada jugador, para obtener la información adecuada deberíamos editarlo manualmente después de cada partido.

La lista de ejemplos es múltiple, si el jugador está lesionado o no disponible, la temporada, la localización del encuentro, etc.

La alternativa para que el proceso sea menos costoso es disponer de múltiples tablas donde acceder a la información y utilizar una herramienta como RStudio para unir esas tablas cuando sea conveniente, y de esta manera poder utilizarlas para los análisis que queramos realizar.

Cuando hablamos de almacenamiento, nos referimos a:

- ◯ memoria RAM
- ◯ tarjetas gráficas

- ◯ Base de datos
- ◯ lenguajes de programación

SUBMIT

Es importante volver a remarcar que, en el caso de que no dispongamos de una estructura como una base de datos profesional, y en su defecto utilicemos libros de Excel, estos se mantengan en el formato más simple y limpio posible, y que la introducción manual de los datos sea la mínima.

Este ejemplo pretende ilustrar la necesidad de unir o cruzar múltiples tablas de datos para los análisis que queramos realizar y de esta manera ser más eficientes con el tiempo del que disponemos y disponer de una mayor cantidad de información que permitirá dar valor a nuestro trabajo.

RStudio permite hacer el cruce de tablas de datos de múltiples:

- añadiendo más variables a la tabla original;

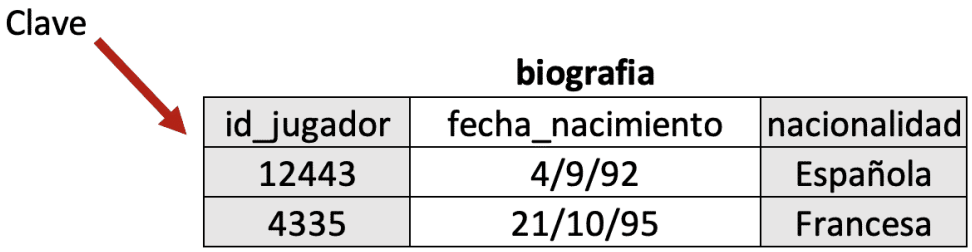
- filtrando observaciones de una tabla a partir de las observaciones de otra de las tablas (Grolemund y Wickham, 2017)

Para comprender el funcionamiento del cruce de bases de datos, es necesario hablar del concepto «clave» o «*key*».

Las claves son esas variables dentro de una tabla que identifican una observación única. Es necesario remarcar la palabra única, es decir, las claves indicarán que esas observaciones no están repetidas.

En la imagen que se presenta a continuación vemos un ejemplo simplificado de una tabla llamada «biografía». En ella observamos cómo hay 3 columnas, La «clave» será la columna id_jugador; ya que ella muestra observaciones únicas, un mismo jugador no puede tener múltiples fechas de nacimiento o nacionalidades.

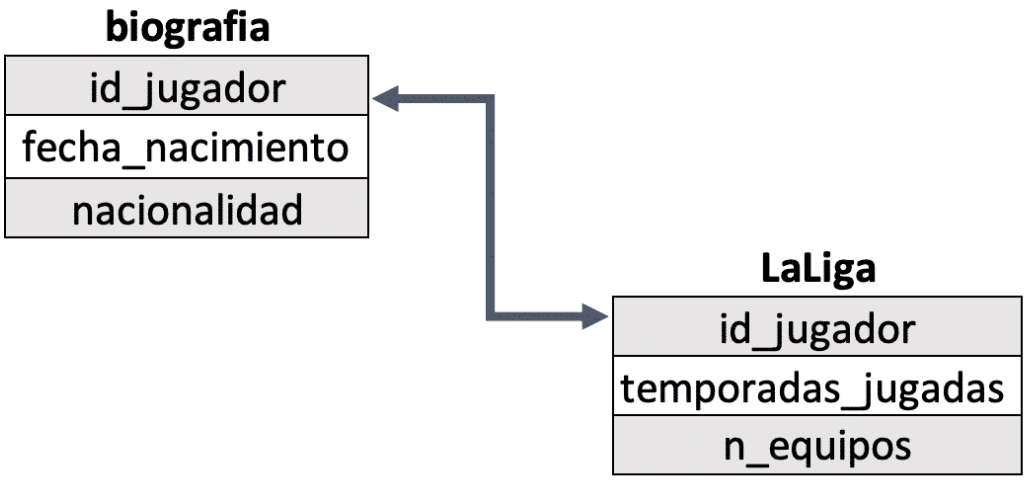
Figura 5. Ejemplo de clave



Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

Esta clave permitirá la asociación/cruce con otra tabla que también tenga la misma clave y, de esta manera, se podrán unir. En el ejemplo siguiente vemos cómo se une con una tabla llamada «LaLiga» que podría tener datos acumulados de los distintos jugadores que han competido en una competición.

Figura 6. Unión de tablas. Ejemplo

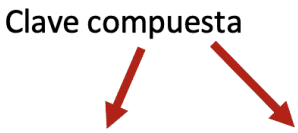


Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

Sin embargo, en nuestro contexto es muy común tener tablas con claves compuestas, por el tipo de datos que recogemos. Estas claves compuestas indican que hay múltiples variables/columnas que son necesarias para distinguir observaciones (filas) no repetidas.

En el ejemplo siguiente vemos cómo las variables «id_jugador» y «fecha_test» son necesarias para identificar observaciones únicas, ya que los jugadores realizan los test múltiples veces y, si seleccionáramos solo la columna «id_jugador», tendríamos más de una observación.

Figura 7. Clave compuesta . Ejemplo



test_fisicos

id_jugador	fecha_test	altura_salto	30_15
12443	2/2/23	32	18
4335	2/2/23	34	20
4389	2/2/23	28	18.5
12443	4/4/23	34	19

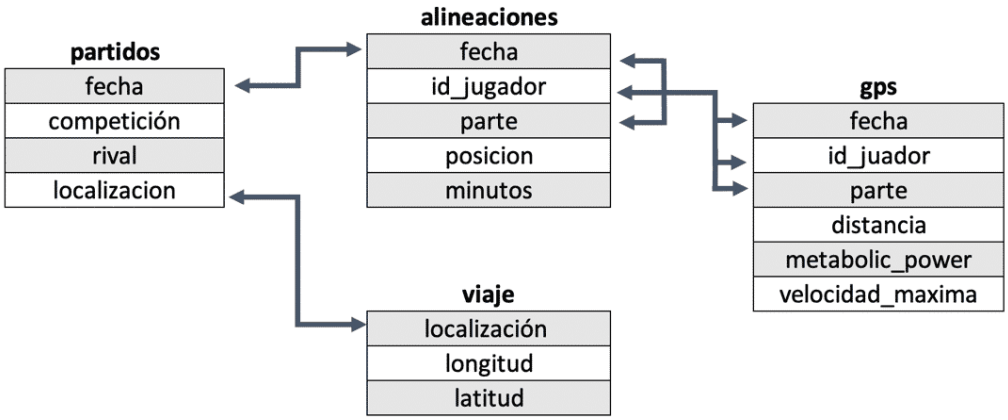
Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

Para relacionar estos datos con otra tabla de datos, también sería necesario relacionarlo con claves compuestas de otra tabla.

La siguiente imagen pretende ejemplificar relaciones múltiples entre distintas claves, las cuales tienen una variedad de claves o claves compuestas que permiten hacer relaciones entre ellas.

Con las 4 tablas que se muestran podemos ver cómo podríamos obtener información sobre las alineaciones durante la competición, datos sobre la carga física mediante los datos GPS e información complementaria que puede ser de interés para análisis complementarios, como los desplazamientos o la distancia viajada y contexto sobre la competición.

Figura 8. Relaciones múltiples entre distintas claves



Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

RStudio permite hacer el cruce de tablas de datos de múltiples:

- [ ] añadiendo más variables a la tabla original.
- [ ] dividiendo las observaciones entre diferentes tablas.
- [ ] RStudio no permite hacer el cruce de tablas de datos de múltiples sin instalar complementos adicionales.
- [ ] filtrando observaciones de una tabla a partir de las observaciones de otra tabla.

SUBMIT

Tipos de unión de tablas de datos

Mediante RStudio vamos a utilizar tres tipos principales de unión de tablas de datos. De nuevo, se utilizan funciones para ello que permiten realizar estos cruces con gran facilidad. Cada tipo de función conseguirá un resultado distinto y, por lo tanto, debemos determinar cuál es el objetivo que pretendemos con el cruce de tablas de datos antes de decidir qué función utilizar.

El material de video será esencial para la comprensión práctica de cada una de las funciones, las descripciones que se muestran a continuación permiten ser una referencia complementaria.

Para ejemplificar cada una de las funciones, vamos a partir de un ejemplo (Atlassian, 2019), donde queremos unir las siguientes dos tablas:

Figura 9. Ejemplo de funciones

biografia

id_jugador	fecha_nacimiento	nacionalidad
12443	4/9/92	Española
4335	21/10/95	Francesa

LaLiga

id_jugador	temp_jugadas	n_equipos
124433	5	3
67899	8	1

Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

- left_join(): esta función permite mantener todas las observaciones de la primera tabla (biografía) y añadir las variables y observaciones de los valores que coinciden con la segunda tabla (LaLiga).

- inner_join(): esta función pertenece al tipo de uniones que filtran datos, es decir, solo mantiene las observaciones que comparten valores (id_jugador) entre tablas.

- full_join(): esta función incluye todas las observaciones y variables de las dos tablas.

Figura 10. Ejemplo de funciones

left_join()

id_jugador	fecha_nacimiento	nacionalidad	temp_jugadas	n_equipos
12443	4/9/92	Española	5	3
4335	21/10/95	Francesa		

inner_join()

id_jugador	fecha_nacimiento	nacionalidad	temp_jugadas	n_equipos
12443	4/9/92	Española	5	3

full_join()

id_jugador	fecha_nacimiento	nacionalidad	temp_jugadas	n_equipos
12443	4/9/92	Española	5	3
4335	21/10/95	Francesa		
67899			8	1

Fuente: elaboración propia con base en captura de pantalla de RStudio (Posit, 2011)

Este ejemplo deja claro que es necesario decidir qué unión queremos realizar en cada caso particular, ya que los resultados serán completamente distintos y, por lo tanto, los consecuentes análisis derivados de estos resultados estarán condicionados por la forma en la que hemos unido los datos.

CONTINUAR

Referencias

Atlassian, (2019). *SQL join types explained visually*. https://www.atlassian.com/data/sql/sql-join-types-explained-visually

Grolemund, G., & Wickham, H. (2017). *R for Data Science*. O'Reilly Media.

Posit, (2011). RStudio [*software* para lenguaje de programación].

CONTINUAR