Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

GRADO EN ESTADÍSTICA

TennisBI: Aplicación Shiny para la visualización de datos de tenis

Autor: **Alberto González de la Plaza**
Tutora: **María Teresa González Arteaga**
2022-2023

Agradecimientos

A mi familia, en especial a mis padres por apoyarme y ayudarme en todo lo que pueden.

A mis amigos, por los ánimos recibidos y la motivación para crear cosas nuevas.

A mi tutora María Teresa González Arteaga, por compartir conmigo sus conocimientos de R y Shiny. Gracias por darme la libertad para escoger dos temas que me apasionan, la visualización de datos y el deporte.

A Julio Pastor Benito por ayudarme en la tarea del despliegue de la aplicación.

A toda la familia del CD Parquesol por generar ilusión y alegría en mí y en todo el barrio.

Gracias a todos

Resumen

El propósito principal de este proyecto es crear una aplicación de visualización de datos de tenis que permita representar mucha información de una manera clara y organizada. Para implementar esta herramienta llamada *TennisBI*, se ha desarrollado una aplicación web mediante la librería Shiny de R, ya que permite crear páginas web y a su vez utilizar funciones de analítica de datos de R.

Adicionalmente, se incluyen procedimientos de análisis de datos para enriquecer la información mostrada. Para ello, se crea una medida de similaridad entre tenistas utilizando una técnica usual en Estadística y en análisis de datos. Esta métrica permite encontrar a qué tenista en activo se parecen los jugadores más jóvenes de la actualidad en términos tenísticos.

En el documento también se define que herramientas han sido utilizadas y se explica que es un proceso de visualización de datos desde la extracción de los mismos hasta la implementación de la página web.

Palabras clave: Tenis, Shiny, visualización de datos, app, R, análisis de datos, similaridad.

Abstract

The main purpose of this project is to create a data visualization application that allows to display a lot of information in a clear and organised format. To implement this application called *TennisBI*, a web application has been developed using R Shiny library, because it allows the creation of web pages and the use of R data analytics functions.

Furthermore, it includes data analysis procedures to enrich the information displayed. For this purpose, a similarity measure between tennis players is created using a common technique in statistics and data analysis. This metric makes it possible to find which current active tennis players are similar to the youngest players in tennis terms.

The document also defines the tools that have been used and it explains the data visualization process from data extraction to the implementation of the web app.

Key words: Tennis, Shiny, data visualization, app, R, data analysis, similarity.

Índice general

Índice de figuras

Capítulo 1

Introducción

En los últimos tiempos, las aplicaciones web se han vuelto herramientas populares ya que se pueden ejecutar en cualquier sistema operativo necesitando únicamente un navegador web. Eso implica que no se requiera la instalación de ningún programa ni aplicación separada para su ejecución.

En el lenguaje de programación R, ampliamente utilizado por los estadísticos, existe un paquete llamado Shiny que permite desarrollar aplicaciones web. Este paquete, al estar integrado en el propio lenguaje, facilita la ejecución de procedimientos estadísticos y análisis que son posibles gracias a R.

Hoy en día existe un crecimiento exponencial de los datos. Este crecimiento afecta a todos los sectores como puede ser medicina, industria y en sectores totalmente diferentes como puede ser el deporte. Toda esta cantidad de datos es muy importante saber analizarla y representarla adecuadamente para generar información valiosa.

Este TFG presenta *TennisBI* [1], una aplicación de visualización de datos de tenis. *TennisBI* muestra información de tenistas, de torneos de tenis, de enfrentamientos entre tenistas y de jóvenes jugadores de tenis. También compara a los tenistas por países.

El tenis es uno de los deportes más conocidos en el mundo, en él por cada partido genera miles de datos. Esta información pueden ser utilizada por comentaristas de televisión y así enriquecer la narración. También pueden ser utilizados por preparadores físicos y entrenadores profesionales para elaborar un enfoque más personalizado en el entrenamiento y la preparación de los jugadores.

1.1. Objetivos

El objetivo principal de este trabajo es utilizar los datos de los partidos de tenis disponibles así como otras fuentes de datos abiertas para crear una aplicación de visualización de datos. También se pretende crear un algoritmo de similaridad entre tenistas para encontrar a qué tenista actual se parecen los jóvenes jugadores.

Los objetivos concretos que se afrontan durante la ejecución del trabajo son:

- Preparar los datos para presentarlos correctamente al usuario.
- Desarrollar una aplicación de visualización de datos utilizando Shiny que permita representar información de tenistas, torneos, enfrentamientos, jóvenes tenistas y comparación por países. La aplicación debe mostrar la misma información para el tenis masculino y femenino.
- Presentar diferentes gráficos descriptivos que sean de rápida interpretación.

[1] `http://shiny1.eio.uva.es:3838/albertogonza/app/`

- Diseñar una interfaz moderna, atractiva y usable.
- Crear un algoritmo de similaridad entre tenistas para encontrar a qué tenista actual se parecen los jóvenes jugadores de tenis teniendo en cuenta sus características durante los partidos.

1.2. Estructura de la memoria

La presente memoria tiene el siguiente esquema:

- **Capítulo 1. Introducción**. Presenta el contexto que enmarca el proyecto, los objetivos a alcanzar y la estructura del resto del documento.
- **Capítulo 2. Competiciones de tenis**. Se describen las competiciones de tenis por ser la temática de la aplicación junto con la historia y los elementos del tenis.
- **Capítulo 3. Datos**. Se explican los datos utilizados en la aplicación. Desde la extracción de los datos desde las fuentes, siguiendo con la exploración para observar la calidad de los datos disponibles y finalizando con la transformación de los mismos, en la que se aplica una serie de reglas para garantizar su calidad.
- **Capítulo 4. Algoritmo de similaridad**. Se realiza una explicación del desarrollo del algoritmo de similaridad entre jugadores de tenis.
- **Capítulo 5. Herramientas utilizadas**. Se explican las herramientas en el desarrollo de la aplicación: R, RStudio, Shiny y un resumen del resto de las librerías utilizadas.
- **Capítulo 6. Estructura y funcionamiento de una aplicación Shiny**. Se detalla la estructura y funcionamiento de una aplicación Shiny.
- **Capítulo 7. Aplicación *TennisBI***. Se realiza una descripción detallada de la aplicación *TennisBI* junto con una explicación de cómo se ha desarrollado.
- **Capítulo 8. Conclusiones**. Se presentan las conclusiones obtenidas del trabajo y se enumeran algunas de las posibles líneas de trabajo futuro a partir de este proyecto.

Capítulo 2

Competiciones de tenis

El tenis es un deporte de raqueta que se juega tanto individualmente como en parejas. Se disputa en una cancha rectangular dividida en dos mitades por una red. El objetivo del juego es golpear la pelota con una raqueta y hacerla pasar por encima de la red hacia el campo del oponente, evitando que éste la devuelva correctamente.

Un partido de tenis se divide en sets y juegos. Un set se gana al alcanzar primero un número determinado de juegos, generalmente seis, con una ventaja de al menos dos juegos sobre el oponente. El partido se gana al vencer un número específico de sets, generalmente tres en el tenis masculino y dos en el tenis femenino.

El tenis es un deporte muy conocido en todo el mundo con millones de jugadores amateur y con torneos profesionales de alto nivel por ciudades de todo el mundo.

2.1. Historia del tenis

Origen y primeros años

La mayoría de los historiadores creen que el tenis se originó en los claustros monásticos del norte de Francia en el siglo XII, pero la pelota se golpeaba con la palma de la mano. Durante gran parte del final de la Edad Media e inicios de la Modernidad la pelota se golpeaba con las manos con unos guantes. Sin embargo, a partir del siglo XV, en el Renacimiento, los guantes empezaron a reforzarse con cuerdas para reducir el impacto y el dolor en las manos al jugar. Los refuerzos en los guantes fueron rápidamente sustituidos por palas de madera. No fue hasta el siglo XVI cuando se creó la primera raqueta con un mango largo y una cuerda hecha de tripa de oveja.

En el siglo XVII el tenis llegó a ser conocido como "el deporte de los reyes" ya que fue practicado por varios monarcas franceses como Francisco I, Enrique II, Carlos IX y Enrique IV que no sólo practicaron sino que promovieron el tenis entre la población. A partir de este momento el tenis desbordó las fronteras de su país natal y se extendió por el mundo. [1]

Inicio del tenis moderno

El comandante Walter Clopton Wingfield en 1874, definió las primeras reglas del tenis sobre hierba basándose en el badminton, un deporte de raqueta en el que se enfrentan dos jugadores o dos parejas. Los jugadores se sitúan en las mitades opuestas de una pista rectangular dividida por una red. Un año más tarde, J.H. Walsh y Henry Jones adquirieron una pradera en Wimbledon, donde fundaron el All England Club y establecieron las primeras canchas de tenis. Fue en 1877 cuando se jugó el primer torneo de Wimbledon masculino y la primera edición femenina de Wimbledon fue en 1884, 7 años después.

Justo en ese momento, el tenis se expandió más allá de sus fronteras hasta llegar a todo el mundo. Hasta tal punto llegó su popularidad que el tenis se incorporó a los Juegos Olímpicos de Atenas en 1896.

También surgieron el resto de torneos que forman el Grand Slam, el Abierto de Estados Unidos en 1881, Abierto de Francia (más tarde llamado Roland Garros) en 1891 y el Abierto de Austrialia 1905. [2]

A partir de 1912, el tenis comenzó a ganar cada vez un mayor reconocimiento, puesto que se creó la Federación Internacional de Tenis (ILTF), cuyo objetivo no era otro que el de institucionalizar este deporte.

En 1926, el promotor CC Pyle estableció el primer circuito profesional con un grupo de jugadores estadounidenses y franceses jugando partidos de exhibición para audiencias de pago. Los primeros profesionales más notables fueron el estadounidense Vinnie Richards y la francesa Suzanne Lenglen. [3, 4]

La Era Abierta (The Open Era)

La Era Abierta comenzó en 1968 cuando los torneos de Grand Slam acordaron permitir que los jugadores profesionales compitieran con los aficionados lo que llevó a un aumento significativo en la competencia y el nivel de juego. Antes de 1968, solo los aficionados podían competir en torneos de Grand Slam y otros eventos organizados por la ILTF, incluida la Copa Davis.

Los años de transición al tenis profesional estuvieron plagados de disputas políticas y juicios por el control de lo que se había convertido en un deporte de grandes sumas de dinero. Tanto los jugadores masculinos como femeninos formaron asociaciones: los hombres formaron la "Association of Tennis Professionals" (ATP) en 1972 y las mujeres formaron la "Women's Tennis Association (WTA)" en 1973.

En 1990, la Asociación de Profesionales del Tenis, dirigida por Hamilton Jordan, se convirtió en el organismo rector del tenis profesional masculino. Establecieron el ATP Tour y agruparon los nueve eventos más prestigiosos como "Super Nine". Doce de los torneos que fueron menos prestigiosos que los primeros nueve eventos fueron renombrados como "Championship Series - Double Week". Ganar un torneo Super Nine valía aproximadamente la mitad de los puntos que ganar un torneo de Grand Slam. El formato continuó hasta 2000, momento en el que los Super Nine pasaron a llamarse Masters Series, ocupando el rango por debajo de los Grand Slams y la Championship Series pasó a llamarse simplemente International. En 2009, los eventos Masters pasaron a llamarse ATP World Tour Masters 1000. Los torneos International se convirtieron en ATP World Tour 500 y los eventos restantes se convirtieron en ATP World Tour 250.

El circuito WTA se organizó de manera diferente que el circuito ATP, siendo los torneos más prestigiosos los cuatro Grand Slam, seguido de los torneos Premier (Premier Mandatory, Premier Five y Premier) y por último los torneos International. [5]

2.2. Reglamento

Se realiza una breve descripción de las reglas del tenis explicando con que elementos se juega y los sistemas de puntuación involucrados en el deporte de la raqueta. [6, 7]

Reglas generales del tenis

- Una pelota debe caer dentro de los límites para que el juego continúe; si un jugador golpea la pelota fuera de los límites, perderá el punto.
- Los jugadores/equipos no pueden tocar la red o los postes ni cruzar al campo contrario.
- Los jugadores/equipos no pueden transportar la pelota ni atraparla con la raqueta.
- Los jugadores no pueden golpear la pelota dos veces.
- Los jugadores deben esperar a que la pelota pase la red antes de devolverla.
- El jugador que no devuelva una pelota antes de que bote dos veces pierde el punto.
- Si la pelota golpea o toca a los jugadores, cuenta como penalización.

- Si la raqueta sale de la mano o se produce abuso verbal, se aplica una penalización.
- Cualquier pelota que bote en las líneas de demarcación se considera buena.
- Un saque debe botar primero antes de que el jugador que lo recibe pueda devolverlo.

Equipamiento

- Pelota de tenis : Según la Federación Internacional de Tenis (ITF), una pelota de tenis típica debe pesar entre 56 y 59,4 gramos con un diámetro de 6,54 a 6,86 centímetros. Deben ser de color amarillo o blanco, aunque la mayoría de las bolas son amarillas.
- Raqueta de tenis : Una raqueta de tenis no puede exceder los 73,7 cm de largo y los 31,7 cm de ancho. La raqueta debe tener un marco que encierre cuerdas resistentes, generalmente hechas de nailon, entretejidas en un patrón cruzado y unido, y un mango.

  El jugador debe golpear la pelota con el centro de la raqueta, la parte con cuerdas que también se conoce como superficie de golpeo.

Puntuación

- Puntos: Unidad de medida más pequeña. Los puntos se incrementan desde 0-15-30-40-juego.
- Juegos: Los juegos constan de 4 puntos cada una, y se gana cuando un jugador alcanza los 4 puntos con al menos 2 juegos de ventaja.
- Sets: Un set consta de 6 juegos y lo gana el jugador/equipo que llegue primero a 6 juegos con al menos 2 puntos de ventaja.
- Set de ventaja: Si se alcanza un tanteo de 6-6 y se utilizan las reglas del set de ventaja, un jugador/equipo sólo puede ganar un set con una ventaja de 2 juegos.
- Partidos: Un partido se juega normalmente al mejor de 3 o al mejor de 5 sets.
- "Deuce": Se produce si se alcanza una puntuación de 40-40. Para ganar el juego, un jugador/equipo debe ganar 2 puntos consecutivos para llevarse el juego. Si un jugador gana un punto, tiene ventaja, pero si pierde el punto siguiente, el marcador vuelve a deuce.
- Juego de tie-break: Si se alcanza un marcador de juego de 6-6 y se utilizan las reglas de set de tie-break, los jugadores deben jugar un juego de tie-break para decidir quién gana el set. En un juego de tie-break, un jugador/equipo debe alcanzar 7 puntos con una ventaja de dos puntos para ganar. Para el formato de saque de un juego de tie-break, el jugador 1 saca para el primer punto, el jugador 2 saca para los dos puntos siguientes, el jugador 1 saca para los dos puntos siguientes...

Modalidades

El tenis puede jugarse en tres modalidades distintas:

- Modalidad individual: Un tenista juega contra otro tenista.
- Modalidad doble: Dos tenistas juegan juntos contra otro equipo con dos jugadores.
- Modalidad dobles mixtos: Igual que en dobles pero los equipos están integrados por un hombre y una mujer.

2.3. Organización y torneos

La ITF y las asociaciones nacionales que la constituyen gobiernan el tenis en todo el mundo, supervisan competiciones internacionales como la Copa Davis y la Copa Federación y el tenis en los Juegos Olímpicos. Los circuitos profesionales fueron gobernados desde finales de la década de 1970 por los consejos internacionales de tenis profesional masculino y femenino. Estos grupos, compuestos por representantes

de la ITF, jugadores y torneos, supervisan el calendario internacional, la implementación de reglas y códigos de conducta, y la capacitación y supervisión de los oficiales del circuito. Los consejos trabajan en estrecha colaboración con ATP y WTA, que brindan una serie de servicios y beneficios a los jugadores y torneos y mantienen clasificaciones que brindan la base para ingresar a torneos y cabezas de serie.

Hasta 1974, cuando Sudáfrica ganó por defecto a India, solo cuatro naciones habían ganado la Copa Davis: Australia, Gran Bretaña, Francia y Estados Unidos. Los campeonatos de esos cuatro países son los tradicionales torneos "major" que componen el Grand Slam. Wimbledon en Gran Bretaña es el más antiguo, ya que se juega sobre el césped del All England Club desde 1877. El campeonato francés, se juega en el Stade Roland-Garros en Auteuil, en las afueras de París, se juega en tierra batida (arcilla). El campeonato de Estados Unidos se jugaron sobre césped desde sus inicios en 1881 hasta 1974, los siguientes tres años se jugaron en una superficie de arcilla sintética en el West Side Tennis Club en Nueva York, y en 1978 el torneo se trasladó a las canchas de pista dura del USA National Tennis Center. El campeonato de Australia se jugó sobre hierba en varias ciudades hasta 1968, cuando se trasladaron a Melbourne se utlizó pista dura.

Los principales eventos por equipos son la Copa Davis, la Copa Federación y la Copa Wightman (mujeres estadounidenses contra británicas).[8]

El ATP Tour consta de tres categorías de torneos: [9]

1. El primer nivel del tenis masculino es el ATP Tour el cual consta de los Grand Slams, los ATP Tour Masters 1000, ATP Tour 500, y ATP Tour 250.

2. El segundo nivel consta del ATP Challenger Tour.

3. El tercer nivel es el ITF World Tennis Tour.

El WTA Tour también consta de tres categorías: [10]

1. El primer nivel del tenis femenino son los llamados torneos Premier (Premier Mandatory, Premier Five y Premier). En el año 2020 se cambiaron de nombre a (WTA 1000, WTA 500 y WTA 250).

2. El segundo nivel son los torneos pertenecientes a WTA 125k Series.

3. El tercer nivel es el ITF World Tennis Tour.

2.4. Ranking

En la actualidad los jugadores están ordenados en un tipo de clasificación llamado Entry Ranking, en el cual se suman los puntos obtenidos por los jugadores en las últimas 52 semanas, o sea, aproximadamente un año. Por lo tanto, al finalizar cada semana, se le restan a cada jugador los puntos obtenidos en esa misma semana del año anterior, y se le suman los ganados en la semana actual.

El Ranking ATP es la clasificación que la Asociación de Tenistas Profesionales (ATP) realiza de los mejores tenistas masculinos en categoría individual y dobles desde el año 1973. Por otra parte el Ranking WTA son las clasificaciones definidas por la Asociación de Tenis Femenino (WTA), introducidas en noviembre de 1975.

Los puntos son otorgados de acuerdo al nivel de los torneos y a lo lejos que llegue un jugador en ellos. A lo largo de los años, el método para calcular los puntos de la clasificación ha variado muchas veces. Actualmente el método es el definido en la Figura 2.1. [11, 12]

Tournament category	W	F	SF	QF	R16	R32	R64	R128	Q
ATP Tour									
Grand Slam	2000	1200	720	360	180	90	45	10	25
ATP Finals	+900 (1500 max)	+400 (1000 max)	200 for each round robin match win (600 max)						
ATP Masters 1000	1000	600	360	180	90	45	10 (25)	(10)	25 (12)
ATP 500 Series	500	300	180	90	45	(20)			20 (10)
ATP 250 Series	250	150	90	45	20	(5)			12 (5)
ATP Challenger Tour									
Challenger 125	125	75	45	25	11	5			5
Challenger 110	110	65	40	20	9	5			4
Challenger 100	100	60	35	18	8	5			4
Challenger 90	90	55	33	17	8	5			4
Challenger 80	80	50	30	16	7	3			4
Challenger 50	50	30	17	9	4				3
ITF Men's World Tennis Tour									
Futures M25	25	16	8	3	1				
Futures M15	15	8	4	2	1				

Category	W	F	SF	QF	R16	R32	R64	R128	Q	Q3	Q2	Q1
Grand Slam (S)	2000	1300	780	430	240	130	70	10	40	30	20	2
WTA Finals (S)	1500*	1080*	750*	(+125 per Round Robin Match; +125 per Round Robin Win)								
WTA 1000 – Mandatory (96S)	1000	650	390	215	120	65	35	10	30	–	20	2
WTA 500 (56S)	470	305	185	100	55	30	1	–	25	–	13	1
WTA 250 (32S,32Q)	280	180	110	60	30	1	–	–	18	14	10	1
WTA 125K series (S)	160	95	57	29	15	1	-	-	6	-	4	1
ITF $100,000+H (S)	150	90	55	28	14	1	-	-	6	–	4	–

Figura 2.1: Ranking ATP (imagen superior) y Ranking WTA (imagen inferior) extraído de ([11, 12])

2.5. Terminología de un partido de tenis

Un partido de tenis comienza con el árbitro lanzando una moneda. El jugador que gana el sorteo puede elegir **sacar** (serve), **recibir-restar** (return) o elegir el lado por el que quiere empezar el partido.

El jugador que saca debe pararse detrás de la línea de base de su lado de la pista de tenis y dentro de los límites de la marca central y la línea lateral.

Para un **servicio** exitoso, el tenista sacador debe lanzar la pelota hacia arriba con la mano que no juega y golpearla con la raqueta antes de que rebote. La pelota debe cruzar la red y caer dentro del área de servicio que está marcada en el lado diagonalmente opuesto de la cancha para que se considere un servicio legal.

A cada jugador se le permiten dos servicios por punto. Si un jugador golpea la red, o el primer rebote de la pelota ocurre fuera del área de servicio, se llama **falta de servicio** y el tenista qué estaba sacando tiene un segundo servicio para iniciar el punto.

Si el pie del tenista sacador toca la línea de base o se sale de los límites de la línea lateral, se conoce como falta de pie y el tenista sacador tendrá un segundo servicio.

Si un jugador también comete una falta en su segundo servicio, se llama **doble falta** y el jugador que recibe obtiene ese punto.

Sin embargo, si el tenista sacador de un jugador golpea la red y cae dentro del área de servicio, el servidor aún tendrá dos servicios para iniciar el punto. Esta situación se conoce como "**let**".

El tenista sacador tiene que alternar entre las mitades verticales de la cancha de tenis para cada punto.

Si el servidor logra conectar un servicio legal y el receptor no puede devolver la pelota, se conoce como un **ace** y el servidor obtiene el punto. [7]

Capítulo 3

Datos

En este capítulo se describe la implementación del proceso de extracción, transformación y carga (ETL):

- **Extracción**. Se explica el proceso de recopilación y almacenamiento de los datos desde los orígenes.
- **Transformación**. Se aplica una serie de reglas para garantizar la calidad de los datos.
- **Carga**. Se transfieren los datos ya transformados al entorno de explotación para poder elaborar análisis posteriores.

El esquema que se lleva desde la extracción de las fuentes hasta la finalización creando la aplicación. Se encuentra en la Figura 3.1.

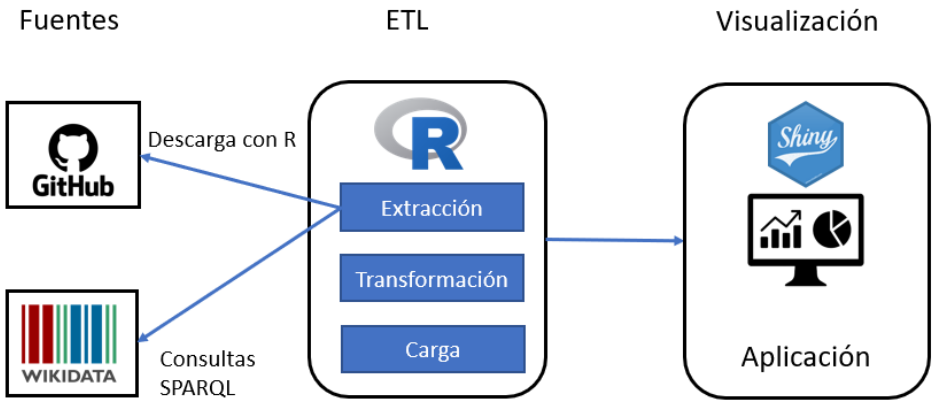

Figura 3.1: Esquema desde la extracción de los datos hasta la creación de la aplicación de visualización

3.1. Extracción de los datos iniciales

El proceso de extracción de datos es la etapa que determina cuáles son las fuentes de datos que se utilizan. Abarcando desde su descarga del sistema de origen a su almacenamiento en un formato preparado para ser transformado adecuadamente en la siguiente etapa.

3.1.1. Fuentes originales

La fuente de datos que se utiliza está procesada y proporcionada por Jeff Sackmann y disponible en Github. Consta de dos repositorios uno para los datos ATP (tenis masculino) y otro para la WTA (tenis femenino). [13, 14]

En el repositorio de tenis masculino *tennis_atp* existen 154 ficheros CSV que se pueden agrupar en seis grupos:

- Ficheros de partidos ATP de modalidad individual: Información sobre los partidos de modalidad individual del circuito ATP para un año concreto, *atp_matches_1968.csv* - *atp_matches_2023.csv*.
- Fichero de partidos amateur: Información de los partidos de competiciones individuales antes de 1968 (antes de la Era Abierta). Se dispone de un único archivo llamado *atp_matches_amateur.csv* con registros desde 1867 hasta 1967.
- Ficheros de partidos ATP de modalidad doble: Información sobre los partidos de modalidad doble del circuito ATP para cada año concreto, *atp_matches_doubles_2000.csv* - *atp_matches_doubles_2023.csv*.
- Ficheros de partidos Challenger de modalidad individual: Información sobre los partidos de modalidad individual del circuito Challenger (segundo más importante después de ATP) para cada año concreto, *atp_matches_qual_chall_1978.csv* - *atp_matches_qual_chall_2023.csv*.
- Fichero de descripción de tenistas: Información genérica de los tenistas que aparecen en el resto de ficheros. Se dispone de un único archivo llamado *atp_players.csv*.
- Ficheros del ranking ATP: Varios ficheros organizados por décadas desde los 70 hasta la actualidad en los que incluyen información semanal del ranking ATP, *atp_rankings_70s.csv* - *atp_rankings_current.csv*.

Por otro lado en el repositorio de tenis femenino *tennis_wta* existen los 183 ficheros CSV que se pueden agrupar en cuatro grupos:

- Ficheros de partidos WTA de modalidad individual: Información sobre los partidos de modalidad individual del circuito WTA para cada año concreto, *wta_matches_1968.csv* - *wta_matches_2023.csv*.
- Ficheros de partidos de clasificación ITF de modalidad individual: Información sobre los partidos de modalidad individual clasificatorios ITF para cada año concreto. *wta_matches_qual_itf_1920.csv* - *wta_matches_qual_itf_2023.csv*.
- Ficheros de descripción de tenistas: Información genérica de las tenistas que aparecen en el resto de ficheros. Se dispone de un único archivo llamado *wta_players.csv*.
- Ficheros del ranking WTA: Varios ficheros organizados por décadas desde los 80 hasta la actualidad en los que incluyen información semanal del ranking WTA, *wta_rankings_80s.csv* - *wta_rankings_current.csv*.

La tarea de descargar los datos desde los orígenes puede convertirse en algo tediosa si queremos tener los datos actualizados regularmente. Jeff Sackmann actualiza los ficheros después de cada torneo. Por ello la tarea de extracción de datos se debe intentar automatizar. Por este motivo se implementó una función en R que descarga en local los archivos CSV de los repositorios.

Para que no haya conflictos al clonar los repositorios de Github se deben eliminar primero todos los archivos del directorio local.

```
downloadData<- function(){
  # Elimina el directorio local y todos los ficheros que hubiera dentro
  unlink("../data/atp/", recursive = TRUE, force = TRUE)
  unlink("../data/wta/", recursive = TRUE, force = TRUE)

  # Clona el repositorio en el directorio local
  system2("git", c("clone", "https://github.com/JeffSackmann/tennis_atp.git", "../data/
    atp/"))

  system2("git", c("clone", "https://github.com/JeffSackmann/tennis_wta.git", "../data/
    wta/"))
}
```

3.1.2. Recopilación de archivos de datos

La información disponible en los repositorios de GitHub es muy completa con 337 archivos CSV. Los repositorios contienen información desde los comienzos de las competiciones de modalidad individual de los circuitos más importantes (ATP y WTA). También se dispone de información de partidos de modalidad doble (solo masculino) o incluso de partidos de circuitos inferiores como Challenger (masculino) o clasificatorios ITF (femenino). Por ello se debe escoger únicamente los archivos utilizados por la aplicación.

Por tanto teniendo en cuenta los objetivos definidos en 1.1, se toman las siguientes decisiones:

- Se seleccionan archivos de partidos de modalidad individual de los circuitos más importantes dentro del mundo del tenis (ATP y WTA). Estos archivos corresponden a *atp_matches_XXXX.csv* y *wta_matches_XXXX.csv* siendo XXXX los años a seleccionar. Como la información de partidos de modalidad doble únicamente está en el repositorio masculino y uno de los requisitos es que la aplicación tenga la misma funcionalidad para tenis masculino como para femenino, se decide descartar los partidos de modalidad doble. Dentro de los partidos de modalidad individual se escogen únicamente a partir del año 2000 hasta la actualidad. El motivo de seleccionar datos sólo del siglo XXI es que muchas variables no aparecen en los archivos antiguos, como por ejemplo, número de aces o número de dobles falta.

  El dataset resultante de fusionar todos los archivos de partidos de la ATP y WTA lo llamo ***dataMatches***.

- Se seleccionan ficheros de descripción de tenistas, que se encuentran en los archivos *atp_players.csv* y *wta_players.csv*.

  El dataset resultante de fusionar la información de tenistas masculinos y femeninos lo llamo ***dataPlayers***.

El resto de archivos de los repositorios no son utilizados. La posición del ranking no es necesaria obtenerla de los ficheros del ranking ya que también aparece en los ficheros de partidos de modalidad individual.

3.1.3. Identificador y descripción de cada dato

Para los ficheros referentes a los partidos de modalidad individual *atp_matches_XXXX.csv* y *wta_matches_XXXX.csv* se disponen de 49 variables y gracias al archivo *matches_data_dictionary.txt* también situado en el mismo repositorio se puede saber la descripción de cada variable. Cada fila de los ficheros referentes a los partidos de modalidad individual corresponde a un partido concreto.

Estas son las variables del dataset ***dataMatches***, resultante de fusionar todos los ficheros de partidos de modalidad individual tanto masculinos como femeninos:

- `tourney_id`: Identificador del torneo.
- `tourney_name`: Nombre del torneo.
- `surface`: Superficie en la que se ha jugado el partido. Puede tomar los siguientes valores: Clay (Arcilla), Grass (Hierba), Hard (Pista dura), Carpet (Moqueta).
- `draw_size`: Tamaño del cuadro del torneo, es decir número de participantes.
- `tourney_level`: Nivel del torneo. Puede tomar los siguientes valores:
  - Para tenis masculino: "G" = Grand Slams, "M" = Masters 1000s, "A" = Otros torneos, "C" = Challengers, "S" = ITFs, "F" = Tour finals y otros eventos de final de temporada, and "D" = Davis Cup.
  - Para tenis femenino: A mayores de los descritos para el tenis masculino existe nuevos códigos como "P" = Premier, "PM" = Premier Mandatory, and "I" = International, "T1" = Tier I (antiguos torneos WTA), "D"= Billie Jean King Cup, Wightman Cup y Bonne Bell Cup.

- `tourney_date`: Fecha de inicio del torneo.
- `match_num`: Identificador de partido dentro de un torneo concreto.
- `winner_id`: Identificador de tenista ganador del partido.
- `winner_seed`: Posición preclasificada del tenista ganador en el torneo.
- `winner_entry`: Describe como ha accedido el tenista ganador del partido al torneo. Puede tomar los siguientes valores: 'WC' = wild card, 'Q' = qualifier, 'LL' = lucky loser, 'PR' = protected ranking, 'ITF' = ITF entry.
- `winner_name`: Nombre del tenista ganador del partido.
- `winner_hand`: Indica la mano hábil del tenista ganador del partido.
- `winner_ht`: Altura en centímetros del tenista ganador del partido.
- `winner_ioc`: Código del país del tenista ganador del partido con la codificación de tres letras IOC (International Olympic Committee) [15].
- `winner_age`: Edad del tenista ganador del partido.
- `loser_id`: Identificador de tenista perdedor del partido.
- `loser_seed`: Posición preclasificada del tenista perdedor en el torneo.
- `loser_entry`: Describe como ha accedido el tenista perdedor del partido al torneo. Puede tomar los siguientes valores: 'WC' = wild card, 'Q' = qualifier, 'LL' = lucky loser, 'PR' = protected ranking, 'ITF' = ITF entry.
- `loser_name`: Nombre del tenista perdedor del partido.
- `loser_hand`: Indica la mano hábil del tenista perdedor del partido.
- `loser_ht`: Altura en centímetros del tenista perdedor del partido.
- `loser_ioc`: Código del país del tenista perdedor del partido con la codificación de tres letras IOC (International Olympic Committee) [15].
- `loser_age`: Edad del tenista perdedor del partido.
- `score`: Resultado del partido.
- `best_of`: Número de sets máximos que se puede jugar en ese torneo, puede ser al mejor de 3 o al mejor de 5 sets.
- `round`: Puede tomar los siguientes valores ordenados de manera decreciente: "F"= Final, "SF"= Semifinal, "QF" = Cuartos de final, "R16"= Octavos de final, "R32" = Dieciseisavos de final...
- `minutes`: Duración del partido en minutos.
- `w_ace`: Número de saques directos del tenista ganador del partido.
- `w_df`: Número de dobles falta del tenista ganador del partido.
- `w_svpt`: Número de puntos al servicio del tenista ganador.
- `w_1stIn`: Número de primeros servicios efectuados del tenista ganador.
- `w_1stWon`: Número de primeros servicios ganados del tenista ganador.
- `w_2ndWon`: Número de segundos servicios ganados del tenista ganador.
- `w_SvGms`: Número de juegos al servicio del tenista ganador.
- `w_bpSaved`: Número de puntos de rotura en contra salvados del tenista ganador.

- `w_bpFaced`: Número de puntos de rotura totales en contra del tenista ganador.
- `l_ace`: Número de saques directos del tenista perdedor del partido.
- `l_df`: Número de dobles falta del tenista perdedor del partido.
- `l_svpt`: Número de puntos al servicio del tenista perdedor.
- `l_1stIn`: Número de primeros servicios efectuados del tenista perdedor.
- `l_1stWon`: Número de primeros servicios ganados del tenista perdedor.
- `l_2ndWon`: Número de segundos servicios ganados del tenista perdedor.
- `l_SvGms`: Número de juegos al servicio del tenista perdedor.
- `l_bpSaved`: Número de puntos de rotura en contra salvados del tenista perdedor.
- `l_bpFaced`: Número de puntos de rotura totales en contra del tenista perdedor.
- `winner_rank`: Puesto en el ranking (ATP o WTA) antes del torneo del tenista ganador.
- `winner_rank_points`: Puntos en el ranking (ATP o WTA) antes del torneo del tenista ganador.
- `loser_rank`: Puesto en el ranking (ATP o WTA) antes del torneo del tenista perdedor.
- `loser_rank_points`: Puntos en el ranking (ATP o WTA) antes del torneo del tenista ganador.

Para los ficheros referentes a descripción de tenistas *atp_players.csv* y *wta_players.csv*, fusionados en ***dataPlayers***.Cada fila corresponde a un tenista concreto. Constan de 8 variables y esta es su descripción:

- `player_id`: Identificador de tenista.
- `name_first`: Nombre del tenista.
- `name_last`: Primer apellido del tenista.
- `hand`: Mano hábil del tenista.
- `dob`: Fecha de nacimiento del tenista.
- `ioc`: Código del país del tenista con la codificación de tres letras IOC (International Olympic Committee) [15]
- `height`: Altura en centímetros del tenista.
- `wikidata_id`: Identificador del tenista en Wikidata [16].

3.2. Exploración de los datos

Antes de tratar los datos se realiza una exploración de los mismos para comprobar si existen valores ausentes, valores atípicos, correlación entre variables...

En esta subsección se exploran los nuevos conjuntos de datos ***dataMatches*** y ***dataPlayers***. Son los resultados de fusionar los archivos de partidos y de tenistas respectivamente.

3.2.1. Valores ausentes (missing values)

La primera exploración que se realiza es calcular el porcentaje de valores ausentes de las variables del dataset de partidos y de información de tenistas. Mediante el siguiente script en R se obtienen el porcentaje de valores ausentes de las variables y sólo se muestran las variables que tienen algún valor ausente:

```
matchesNan<- round((colMeans(is.na(dataMatches)))*100,1)
print(sort(matchesNan[matchesNan>0],decreasing = T))
playersNan<- round((colMeans(is.na(dataPlayers)))*100,1)
print(sort(playersNan[playersNan>0],decreasing = T))
```

Obteniéndose los siguiente resultados:

loser_seed	winner_seed	minutes	w_SvGms	l_SvGms	w_df
71.0	41.7	35.2	32.4	32.4	15.2
l_df	w_ace	w_svpt	w_1stIn	w_1stWon	w_2ndWon
15.2	15.1	15.1	15.1	15.1	15.1
w_bpSaved	w_bpFaced	l_ace	l_svpt	l_1stIn	l_1stWon
15.1	15.1	15.1	15.1	15.1	15.1
l_2ndWon	l_bpSaved	l_bpFaced	w_1stOut	l_1stOut	tourney_level
15.1	15.1	15.1	15.1	15.1	14.4
loser_ht	loser_rank	loser_rank_points	winner_ht	winner_rank	winner_rank_points
2.6	0.4	0.4	0.3	0.2	0.2

Figura 3.2: Porcentaje de missing values por cada variable en ***dataMatches*** (información de partidos)

height	hand	dob
2.9	1.3	0.4

Figura 3.3: Porcentaje de missing values por cada variable en ***dataPlayers*** (información de tenistas)

En los ficheros de partidos existen 30 variables con algún valor ausente pero sólo tiene grandes porcentajes en las variables de la posición preclasificada de los tenistas en cada torneo (`winner_seed` y `loser_seed`). Ambas variables no se han tenido en cuenta para ningún gráfico debido a su alto contenido en missing values.

Por otra parte en los datos de información de tenistas sólo encontramos 3 variables con valores ausentes y son la altura, la mano hábil y la fecha de nacimiento. La altura posee casi de 3 % de missing values pero no causa problema ya que en ***dataMatches*** existen las variables `winner_ht` y `loser_ht` con menos missing values.

Para las variables con missing values no se ha realizado ninguna técnica de imputación ni se ha eliminado los valores ausentes.

3.2.2. Valores atípicos (Outliers)

Para identificar posibles valores atípicos (outliers) se ha decidido visualizar mediante un diagrama de cajas por cada variable numérica para observar si existen valores atípicos.

En la Figura 3.4 se pueden observar los diagramas de caja de 9 variables de ***dataMatches***. Se observa que existen valores muy grandes pero no han sido por errores de medición, sino por partidos que han durado en exceso. Este aumento de la duración es porque ninguno de los tenistas lograba ganar el último set por más de dos puntos de diferencia. Esto provoca que todas las variables de ese partido (número de saques, número de dobles falta...) aumenten considerablemente. En la Figura 3.4 sólo se representa un subconjunto de 9 variables, pero en la etapa de exploración se han visualizado mediante diagramas de caja todas las variables de ***dataMatches***. Para el resto de variables también aparecen outliers por el mismo motivo de la duración de los partidos.

Al no tratarse de errores de medición sino que han sido producto de partidos muy largos no se han eliminado los valores atípicos.

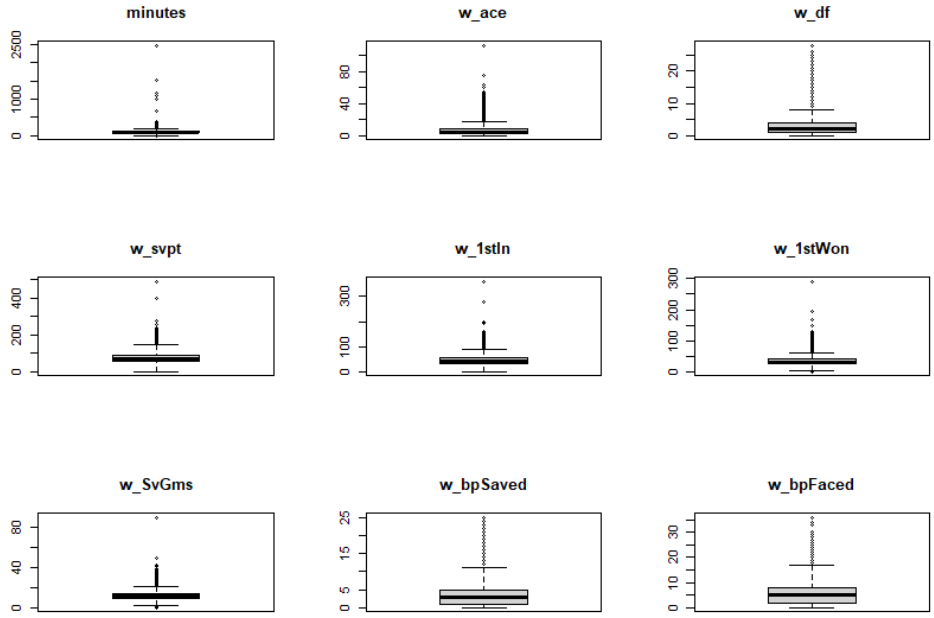

Figura 3.4: Diagramas de cajas para la búsqueda de valores atípicos en ***dataMatches*** (en la etapa de exploración se han probado con todas las variables)

3.2.3. Correlación variables

Para observar la correlación entre las variables se representa mediante un gráfico de correlación (correlation plot) con todas las variables numéricas de ***dataMatches***. Sólo se han incluido en la Figura 3.5 las 9 variables que se representan en la Figura 3.4 referente a los outliers.

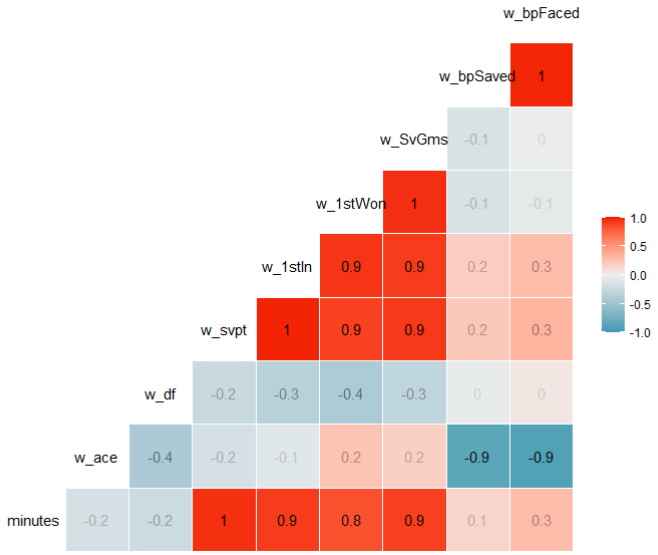

Figura 3.5: Correlaciones de algunas variables de ***dataMatches***

Se observa como hay variables con correlaciones positivas muy altas como `w_svpt`, `w_1stIn`, `w_1stWon` y `w_SvGms` o `w_bpSaved` y `w_bpFaced`. También se observa como las variables `w_svpt`, `w_1stIn`, `w_1stWon` y `w_SvGms` están altamente correladas con la variable de duración del partido `minutes` lo que explica los outliers encontrados en el apartado anterior. Lo mismo ocurre con las variables del tenista perdedor

del partido `l_svpt`, `l_1stIn`, `l_1stWon` y `l_SvGms` las cuales están altamente correladas con la variable `minutes`.

3.3. Transformación de los datos

En esta subsección se detallan los cambios realizados para disponer los datos en formato adecuado.

3.3.1. Fusión de todos los archivos en dos únicos datasets

Los datos después de prepararlos se guardan en archivos CSV que se actualizan periódicamente. A estos ficheros se acceden fácilmente mediante la aplicación Shiny para elaborar los posteriores análisis y visualizaciones. Los dos archivos de datos que se crean con todo el conjunto de archivos disponibles son:

- ***dataMatches***: Dataset resultante de fusionar todos los archivos de partidos tanto masculinos (ATP) como femeninos (WTA) disponibles. Para construirlo correctamente es necesario crear dos variables nuevas. La variable `year`, para identificar el año en el que se jugó cada partido y la variable `circuit`, que identifica si el partido es ATP (masculino) o WTA (femenino).

  Se ha creado una función para fusionar los ficheros en uno único que recibe por argumento el circuito (ATP o WTA). Mediante la siguiente expresión regular (regex):
  "(",circuit,"_matches_[0-9][0-9][0-9][0-9](.csv))" se seleccionan todos los nombres de ficheros del repositorio de donde se encuentran los datos descargados que contengan en el nombre atp_matches (wta_matches si circuit = "wta") seguido de cuatro dígitos y que finaliza con .csv. Tras ello se seleccionan los ficheros más actuales ya que sólo van a utilizarse archivos correspondientes a partidos del siglo XXI.

```
mergeMatches<-function(circuit="atp"){
  list<-list.files(paste('../data/',circuit,sep=""), pattern=paste("(",circuit,"_
    matches_[0-9][0-9][0-9][0-9](.csv))",sep=""), full.names=TRUE)

  # Selecciona todos los datasets de partidos del siglo XXI
  actual_year <- as.integer(format(Sys.Date(), "%Y"))

  list<- sort(list,decreasing = T)[1:(actual_year-1999)]
  print(list)
  dataMatches<- data.frame()
  for (file in list){
    dataMatches<- rbind(dataMatches,read.csv(file))
  }
  return(dataMatches)

}

# data_matches
dataATPMatches<-mergeMatches("atp")
dataATPMatches$circuit<- "ATP"
dataWTAMatches<-mergeMatches("wta")
dataWTAMatches$circuit<- "WTA"
dataMatches<- rbind(dataATPMatches,dataWTAMatches)
dataMatches<-transformDataMatches(dataMatches)
```

- ***dataPlayers***: Dataset resultante de fusionar la información de tenistas masculinos y femeninos. Se añade la variable `circuit` que identifica si el partido es ATP (masculino) o WTA (femenino) igual que en ***dataMatches***.

```
mergePlayers<-function(dataMatches){
  datosPlayersATP <- read.csv("../data/atp/atp_players.csv")
  datosPlayersATP$circuit<-"ATP"
  datosPlayersWTA <- read.csv("../data/wta/wta_players.csv")
  datosPlayersWTA$circuit<-"WTA"
```

```
6
7   datosPlayers<-rbind(datosPlayersATP,datosPlayersWTA)
8
9   return(datosPlayers)
10 }
11 dataPlayers<- mergePlayers(dataMatches)
```

3.3.2. Filtrado de datos

Después de fusionar en dos únicos datasets toda la información la siguiente transformación que se ha realizado ha sido filtrar los datos para reducir el número de partidos y de tenistas que utiliza la aplicación. Se filtran los datos para reducir el tamaño de la información almacenada. Otro motivo también es para quedarse únicamente con los partidos de torneos individuales.

En cuanto a los datos de partidos (***dataMatches***) se ha realizado un filtrado eliminando todos los partidos de torneos realizados por equipos. Estos son para el circuito masculino la Copa Davis, United Cup, Atp Cup, Laver Cup. Para el circuito femenino es la BJK Cup (Billie Jean Cup) y la Fed Cup. Se han eliminado un total de 13825 partidos.

Por tanto a fecha de 17/04/2023 se dispone un total de 120398 partidos en la aplicación como se observa en la Figura 3.6.

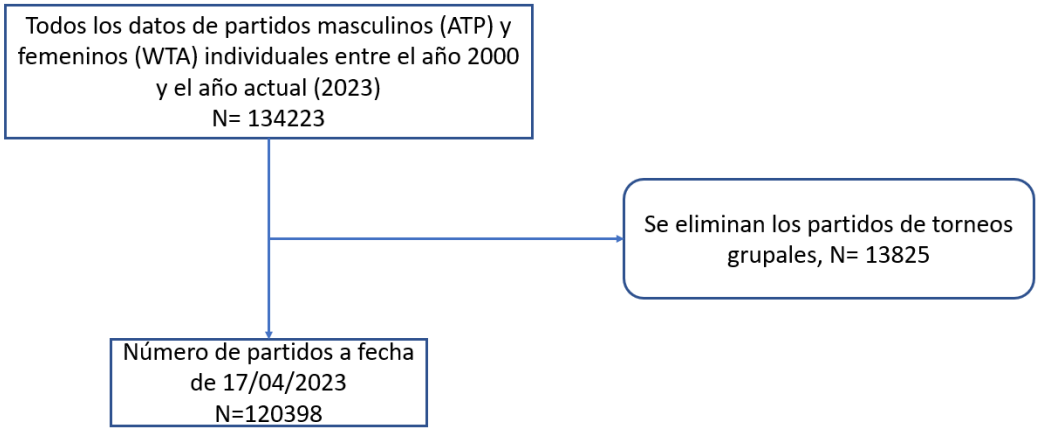

Figura 3.6: Diagrama de flujo del proceso de selección de observaciones de ***dataMatches***

```
filterData<-function(dataMatches){
    # Elimino los partidos de competiciones grupales
   teamTourneys<- "Davis|United Cup|Atp Cup|Laver Cup|BJK Cup|Fed Cup"
  individualTourneys<-dataMatches$tourney_name[!grepl(teamTourneys, dataMatches$tourney
    _name)]
  dataMatches<- dataMatches[dataMatches$tourney_name %in% individualTourneys,]

  print("Eliminados partidos de torneos por equipos")
  print("Dimension data matches despues de filtro: ")
  print(dim(dataMatches))

  return(dataMatches)
}
```

Para los datos de información de tenistas se han eliminado todos los tenistas que no tuvieran ningún partido ganado en el dataset ***dataMatches*** después de haber ejecutado todos los filtros, se eliminan un total de 122380 tenistas. Por tanto a fecha de 17/04/2023 se dispone un total de 2065 tenistas en la aplicación como se observa en la Figura 3.7

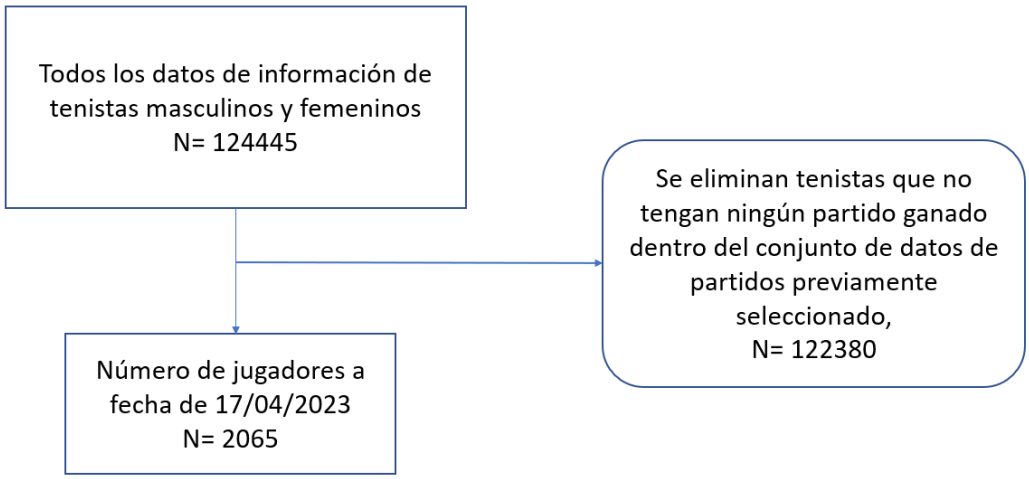

Figura 3.7: Diagrama de flujo del proceso de selección de observaciones de ***dataPlayers***

Con R se implementa en una única línea seleccionando los tenistas que tengan su nombre completo en la columna de `winner_name` de ***dataMatches***. Para ello primero debemos crear la variable de nombre completo (`name_complete` formado por nombre (`name_first`) y primer apellido (`name_last`).

```
  dataPlayers$name_complete<-paste(dataPlayers$name_first,dataPlayers$name_last)

dataPlayers<- dataPlayers[dataPlayers$name_complete %in% unique(dataMatches$winner_name
    ),]
```

3.3.3. Recodificar variables

Recodificar una variable categórica significa cambiar las etiquetas o niveles de una variable categórica existente para que sean más comprensibles o significativas para el análisis.

Durante la etapa de exploración se observó ciertas incoherencias en algunas variables categóricas. Como la aparición de "Us Open" y "US Open" para referirse al Grand Slam estadounidense o "s-Hertogenbosch","'s-Hertogenbosch","s Hertogenbosch" para referirse al torneo de la ciudad de Rosmalen, Países Bajos. En otros casos la recodificación se ha utilizado para traducir al español o para poner la descripción en vez de códigos como es en el caso de `tourney_level` y de `hand` cambiando las etiquetas "L" y "R" por "Izquierda" y "Derecha".

- **Recodificaciones en *dataMatches***: Se han recodificado tres variables en el dataset resultante de la fusión de partidos, ***dataMatches***.
  - Para la variable nombre de torneo (`tourney_name`) primero mediante la función *trimws* se han eliminado todos los espacios al final del nombre del torneo que causaba que existiera "Belgrade" y "Belgrade " con un espacio al final refiriéndose al mismo torneo. Después mediante la función *grep* se utiliza para buscar todos los nombres de torneos que aparezca la cadena "Hertogenbosch" ya que aparecía con nombres muy similares y era necesario unificarlo en un único nombre. También con la función *gsub* se sustituye la aparición de "Us Open" por la de "US Open".
  - Para la variable de nivel de torneo (`tourney_level`) las modalidades aparecían codificadas con letras y en el archivo matches_data_dictionary.txt se puede conocer el significado de cada código y con la función *factor* se le pasa como argumento la columna, luego en levels se le indican los códigos y en labels el significado del código.
  - La variable superficie (`surface`) ha sido traducida de inglés a español.

```
refactorVariables<-function(dataMatches){
  # Variable nombre de torneo:

  # Elimina el espacio al final que se ha incluido por error
  dataMatches$tourney_name <- trimws(dataMatches$tourney_name)
  # s-Hertogenbosch sale escrito de muchas formas diferentes
  dataMatches$tourney_name[grepl("Hertogenbosch", dataMatches$tourney_name)] <- "s
    -Hertogenbosch"
  # Aparece Us Open y US Open
  dataMatches$tourney_name <- gsub("Us Open", "US Open", dataMatches$tourney_name,
     ignore.case = TRUE)

  # Nivel torneo:
  dataMatches$tourney_level<-factor(dataMatches$tourney_level,levels = c("A","D","
    F","G","M","P","PM","I","S"),labels=c("Torneo ATP","Davies Cup","Tour Finals","
    Grand Slam","Master 1000","Premier","Premier Mandatory","Internacional","ITFs")
    )

  # Superficie:
  dataMatches$surface<-factor(dataMatches$surface,levels = c("Hard","Clay","Grass"
    ,"Carpet"),labels=c("Pista dura","Arcilla","Hierba","Moqueta"))

  return(dataMatches)

}

```

- **Recodificaciones en *dataPlayers***: La única recodificación es en la variable de mano hábil (`hand`) cambiar las etiquetas "L" y "R" por "Izquierda" y "Derecha".

```
# codigo perteneciente a la funcion transformDataPlayers() que transforma data_
    players

# Refactor variable hand
  dataPlayers$hand<-factor(dataPlayers$hand,levels = c("L","R"),labels=c("
    Izquierda","Derecha"))

```

3.3.4. Creación nuevas variables

- **Nuevas variables en *dataMatches***: Se han creado cuatro variables en el dataset resultante de la fusión de partidos, ***dataMatches***.
  - `date`: Fecha del comienzo del torneo (`tourney`) en formato "%Y %m %d".
  - `year`: Año en el que se jugó partido.
  - `w_1stOut`: Número de fallos en el primer saque del tenista ganador del partido. Se crea restando el número de puntos al servicio (`w_svpt`) menos el número de primeros servicios efectuados (`w_1stIn`), ambas variables del tenista ganador.
  - `l_1stOut`: Número de fallos en el primer saque del tenista perdedor del partido. Se crea restando el número de puntos al servicio (`l_svpt`) menos el número de primeros servicios efectuados (`l_1stIn`), ambas variables del tenista ganador.

```
dataMatches$date <- as.Date(as.character(dataMatches$tourney_date),'%Y%m%d')
  dataMatches$year <- format(dataMatches$date,'%Y')

# numero de fallos en el primer saque del tenista ganador
  dataMatches$w_1stOut <- dataMatches$w_svpt-dataMatches$w_1stIn
  # numero de fallos en el primer saque del tenista perdedor
  dataMatches$l_1stOut <- dataMatches$l_svpt-dataMatches$l_1stIn

```

- **Nuevas variables en *dataPlayers***: Se ha creado la variable nombre completo de un tenista (`name_complete`) que fusiona le nombre (`name_first`) y el primer apellido (`name_last`) con un espacio entre medias ya que así es como aparece en ***dataMatches*** en las variables de `winner_name` y `loser_name`. Esto en R se implementa mediante la función *paste* en la que se indica las dos columnas que se desean juntar y el separador por defecto es el espacio.

```
1  dataPlayers$name_complete<-paste(dataPlayers$name_first,dataPlayers$name_last)
2
```

Capítulo 4

Algoritmo de similaridad

La aplicación construida, *TennisBI* consta de un apartado de los tenistas jóvenes (jóvenes promesas) en la actualidad. En este apartado, a cada joven promesa se le compara con el resto de tenistas profesionales y se pronostica a qué jugador actual es "más similar". Se define como tenista joven o joven promesa a aquellos tenistas con edad en el año actual inferior a 25 años. Se define como tenista o jugador actual a aquellos tenistas que han jugado al menos un partido en los dos últimos años.

El propósito es crear un algoritmo de similaridad que permita comparar tenistas mediante únicamente habilidades tenísticas como efectividad en el saque, efectividad al resto, habilidad en las diferentes superficies...

4.1. Metodología

La función que se crea para calcular los tenistas más similares a las jóvenes promesas del tenis, utiliza **análisis de componentes principales (ACP) escalado y centrado**.

En el análisis descriptivo de las variables se observó como existen variables altamente correladas (ver Figura 3.2.3). El ACP es una técnica utilizada para describir un conjunto de datos en términos de nuevas variables ("componentes") no correlacionadas. Las componentes principales se ordenan por la cantidad de varianza original que describen. Esta técnica es útil para reducir la dimensionalidad de un conjunto de datos. [17]

Con el análisis de componentes principales obtenemos vectores que son variables independientes unas de otras. Esto es un requisito para utilizar la similitud coseno. Al realizar ACP escalado y centrado se consigue que las variables originales tengan el mismo peso para el análisis ya que las variables disponibles tienen diferentes escalas.

En estudios previos donde se realizaron propuestas de búsquedas de deportistas "similares" para datos de fútbol y de tenis, se utiliza la similitud coseno para establecer el grado de similaridad entre jugadores (ver Fórmula 4.1). [18, 19]

La similitud coseno mide la similitud entre dos vectores A y B en un espacio que posee un producto interior. Se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir si ambos vectores apuntan a un mismo lugar. Cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a uno. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1. La similitud coseno se suele emplear como indicador de cohesión y nunca como una distancia ya que no cumple la desigualdad triangular. [20]

La fórmula de la similitud coseno es la siguiente:

$$\text{similitud\_coseno}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}} \tag{4.1}$$

4.2. Etapas creación algoritmo de similaridad

4.2.1. Creación de nuevas variables (Feature Engineering)

En el dataset de partidos se disponen de múltiples variables tanto del tenista ganador como del tenista perdedor del partido como se indica en el apartado 3.1.3. Algunas de estas variables aportan información por sí solas pero en algunas otras es necesario transformarlas para crear nuevas variables útiles para el algoritmo de similaridad.

Se definen las siguientes nuevas variables para cada jugador teniendo en cuenta los datos de partidos de los dos últimos años:

- `ace_mean`: Media de número de saques directos (`ace`) por partido.
- `ace_entre_df`: Número de saques directos (`ace`) por partido dividido entre número de dobles falta (`df`) por partido.

$$\texttt{ace\_entre\_df} = \frac{ace}{df}$$

- `%_1stIn`: Porcentaje de acierto del primer saque. Se calcula dividiendo el número de aciertos en el primer saque (`1stIn`) dividido el número total de saques (`svpt`).

$$\texttt{\%\_1stIn} = \frac{1stIn}{svpt} \cdot 100\,\%$$

- `%_1stWon_saque`: Porcentaje de victorias de un juego en el primer saque cuando el tenista está sacando. Se calcula dividiendo el número de puntos ganados en el primer saque (`1stWon`) entre el número de aciertos en el primer saque (`1stIn`).

$$\texttt{\%\_1stWon\_saque} = \frac{1stWon}{1stIn} \cdot 100\,\%$$

- `%_2ndWon_saque`: Porcentaje de victorias de un juego en el segundo saque cuando el tenista está sacando. Se calcula dividiendo el número de puntos ganados en el segundo saque (`2ndWon`) entre el número de fallos en el primer saque (`1stOut`).

$$\texttt{\%\_2ndWon\_saque} = \frac{2ndWon}{1stOut} \cdot 100\,\%$$

- `%_1stWon_resto`: Porcentaje de victorias de un juego en el primer saque cuando el tenista está restando (recibiendo el saque). Se calcula restando 1 menos la división del número de puntos ganados en el primer saque del rival (`rival_1stWon`) entre el número de aciertos en el primer saque del rival (`rival_1stIn`). Se realiza el complementario de la probabilidad de victoria del rival en el primer saque cuando está sacando.

$$\texttt{\%\_1stWon\_resto} = (1 - \frac{rival\_1stWon}{rival\_1stIn}) \cdot 100\,\%$$

- `%_2ndWon_resto`: Porcentaje de victorias de un juego en el segundo saque cuando el tenista está restando (recibiendo el saque). Se calcula restando 1 menos la división del número de puntos ganados en el segundo saque del rival (`rival_2ndWon`) entre el número de fallos en el primer saque del rival

(`rival_1stOut`). Se calcula la probabilidad del complemento de victoria del rival en el segundo saque cuando está sacando.

$$\texttt{\%\_2ndWon\_resto} = (1 - \frac{rival\_2ndWon}{rival\_1stOut}) \cdot 100\,\%$$

- `%_bp_Salvados`: Porcentaje de puntos de rotura en contra salvados. Se calcula dividiendo el número de puntos de rotura de saque / bolas de break salvados en contra(`bpSaved`) entre el número total de puntos de rotura de saque en contra (`bpFaced`).

$$\texttt{\%\_bp\_Salvados} = \frac{bpSaved}{bpFaced} \cdot 100\,\%$$

- `%_bp_Rotos`: Porcentaje de puntos de rotura a favor completados. Se calcula restando 1 menos la división del número de puntos de rotura de saque / bolas de break salvados en contra(`rival_bpSaved`) del rival entre el número total de puntos de rotura de saque en contra (`rival_bpFaced`) del rival.

$$\texttt{\%\_bp\_Rotos} = (1 - \frac{rival\_bpSaved}{rival\_bpFaced}) \cdot 100\,\%$$

El análisis de componentes principales sólo permite variables numéricas, pero en los datos disponibles existe la variable superficie (`surface`) la cuál es muy interesante para el análisis y es necesaria realizar una transformación para poder utilizarla en el análisis de componentes principales.

Por este motivo hemos creado un coeficiente para cada superficie la cual se obtiene diviendo el porcentaje de victorias en cada superficie entre el porcentaje de victorias global. Por tanto como a partir de 2009 eliminaron las pistas de moqueta se han creado las siguientes tres variables:

- `coef_Hierba`: Coeficiente del tenista en la superficie hierba. Se calcula diviendo el porcentaje de victorias en hierba entre el porcentaje total de victorias.

$$\texttt{coef\_Hierba} = \frac{\%\_Victoria\_hierba}{\%\_Victoria\_total}$$

- `coef_Arcilla`: Coeficiente del tenista en la superficie arcilla. Se calcula diviendo el porcentaje de victorias en arcilla entre el porcentaje total de victorias.

$$\texttt{coef\_Arcilla} = \frac{\%\_Victoria\_arcilla}{\%\_Victoria\_total}$$

- `coef_Pista_dura`: Coeficiente del tenista en la superficie pista dura. Se calcula diviendo el porcentaje de victorias en pista dura entre el porcentaje total de victorias.

$$\texttt{coef\_Pista\_dura} = \frac{\%\_Victoria\_pista\_dura}{\%\_Victoria\_total}$$

Todas estas nuevas variables estás construidas de tal manera que niveles altos de las variables significa que un tenista es bueno en esa variable. Esto se elige así para poder añadir todas las variables en un gráfico de radar y tener una rápida interpretación. (ver Figura 4.1).

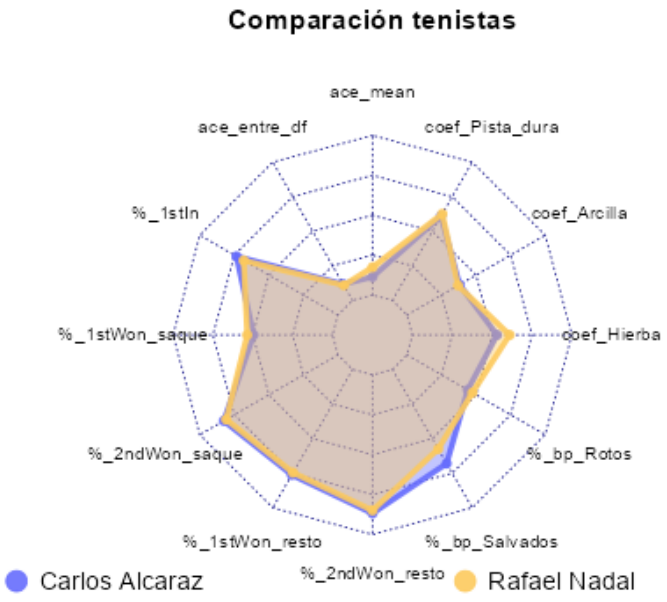

Figura 4.1: Gráfico de radar con las variables utilizadas en el algoritmo de similaridad para comparar a Carlos Alcaraz y Rafael Nadal

4.2.2. Obtención del tenista más similar

Una vez construidas las variables que se consideran en el algoritmo lo siguiente que hay que realizar para obtener el jugador más similar es implementarlo en R.

Para realizar el ACP se utiliza la función *prcomp* de la librería *stats*. Esta función recibe como primer argumento una matriz numérica con los valores de las variables. Para que el ACP sea escalado es necesario indicarlo con el argumento *scale*. [21]

Para realizar la similitud coseno al resultado obtenido del ACP se utiliza la función *cosine* de la librería *lsa*. [22]

Para evitar que un tenista se asigne a sí mismo como jugador más similar, se asigna a toda la diagonal de la matriz de similaridades el valor de -Infinito.

Por último se devuelve mediante la función *which.max* el identificador donde se encuentra la observación más similar.

```
getSimilarityPlayer<-function(df){
  # df:  primera observacion la joven promesa y despues Jugadores profesionales

  #Componentes principales(quitamos primera columna del nombre)
  pca<-prcomp(as.matrix(sapply(df[,-1], as.numeric)),scale. = TRUE)

  # Similitud coseno
  similarities<-cosine(t(pca$x))

  # Poner -Inf en la diagonal para que no se escoja a si mismo como mas parecido
  diag(similarities)<- -Inf

  # Cogemos el id de la observacion mas similar
  return(df[which.max(similarities[,1]),])
}


```

Capítulo 5

Herramientas utilizadas

En este capítulo se presentan las herramientas utilizadas durante el desarrollo del presente Trabajo de Fin de Grado.

5.1. R

R es un lenguaje y un entorno para realizar computación estadística y para representar gráficos. R es un proyecto GNU que originalmente era el lenguaje S, desarrollado en los Laboratorios Bell (antes AT&T, ahora Lucent Technologies) por John Chambers.[23]

R está disponible como Software Libre bajo los términos de la Licencia Pública General GNU de la Fundación para el Software Libre. Se compila y ejecuta en una amplia variedad de plataformas UNIX y sistemas similares (incluyendo FreeBSD y Linux), Windows y MacOS. [24]

R es un conjunto integrado de instalaciones de software para la manipulación de datos, el cálculo y la visualización gráfica. Además, incluye una instalación efectiva de manejo y almacenamiento de datos. También cuenta con un conjunto de operadores para cálculos en arrays, especialmente matrices. También presenta un lenguaje de programación bien desarrollado, simple y efectivo. Este lenguaje interpretado incluye condicionales, bucles, funciones recursivas definidas por el usuario y facilidades de entrada y salida.

Asimismo, R ofrece una colección grande, coherente e integrada de herramientas intermedias para el análisis de datos. Varios ejemplos de estas herramientas son los modelos lineales y no lineales, test estadísticos clásicos, análisis de series temporales, funciones para realizar clasificación, funciones para realizar clustering...

Por otro lado, proporciona facilidades gráficas para el análisis y visualización de datos. Es uno de los puntos fuertes del lenguaje ya que permite producir gráficos bien diseñados, incluyendo símbolos matemáticos y fórmulas cuando sea necesario. El usuario conserva el control total sobre los gráficos.

R se puede extender fácilmente a través de librerías o paquetes. Hay alrededor de ocho paquetes suministrados con la distribución R y muchos más están disponibles a través del CRAN (Comprehensive R Archive Network) el cual contiene las versiones más recientes y anteriores de la distribución de R, documentación y paquetes de R, actualmente en mayo de 2023 tiene 19606 paquetes disponibles lo que hace que R sea muy completo. [25]

Para tareas de computación intensiva, se puede vincular código C, C++ y Fortran y llamarlo en tiempo de ejecución. Los usuarios avanzados pueden escribir código C para manipular objetos R directamente.

Algunas de las principales ventajas de R respecto a otro tipos de software:

- Software de acceso libre y código abierto, gracias a esto no tiene limitadas sus funciones.

- R es sencillo de comprender, es un lenguaje de alto nivel que sigue el paradigma de orientación a objetos.
- R permite la manipulación de datos de manera comprensiva y sencilla.
- R permite crear gráficos de alta calidad exportables en diversos formatos.
- R permite guardar los comandos utilizados en forma de scripts para su posterior reutilización.
- R está disponible para los sistemas operativos más utilizados actualmente (Windows, macOS y GNU/Linux).

Una utilidad importante en R es el paquete RMarkdown que permite escribir informes dinámicos y hacerlo en varios formatos (PDF, HTML y RTF).

Además, desde R se pueden crear aplicaciones web interactivas (apps) con el paquete Shiny [26]. Este paquete se va a utilizar en en el presente TFG para la creación de la aplicación web.

La interfaz que el programa base muestra es muy básica y permite utilizar todas las funcionalidades del programa. Pero resulta excesivamente simple y se ha preferido utilizar RStudio que posee una interfaz más moderna y amigable, RStudio se explica en detalle en el siguiente apartado.

R permite crear y editar scripts en el propio programa, los scripts son ficheros con órdenes de R para poder ejecutarlas en cualquier momento.

A fecha de mayo de 2023, se encuentra en la versión 4.3.0, aunque para el presente trabajo se utiliza la versión 4.2.1.

5.2. RStudio

RStudio es un IDE (entorno de desarrollo integrado) gratuito y de código abierto para R. Su interfaz está organizada para que el usuario pueda ver claramente gráficos, tablas de datos, código R y generar todos al mismo tiempo. También ofrece una función que permite a los usuarios importar archivos CSV, Excel, SAS (*.sas7bdat), SPSS (*.sav) y Stata (*.dta) en R sin tener que escribir el código para hacerlo. [27] Otros formatos menos usuales se pueden leer utilizando librerías de R.

RStudio permite la edición de scripts de R (ficheros que contienen instrucciones en lenguaje R), la creación de cuadernos, ficheros RMarkdown (que permiten la ejecución de chunks de código R o incluso otros lenguajes como C o Python, su salida y texto, así como exportación a PDF, HTML...), crear paquetes y librerías o incluso aplicaciones web, entre otras funcionalidades, de manera sencilla. La característica más importante de RStudio es su sencilla y moderna interfaz, que permite la interacción con el directorio de trabajo, el entorno y la consola de comandos de una manera cómoda.

La Figura 5.1 muestra los cuatro paneles más habituales de la interfaz de RStudio. A continuación se describen brevemente: [28]

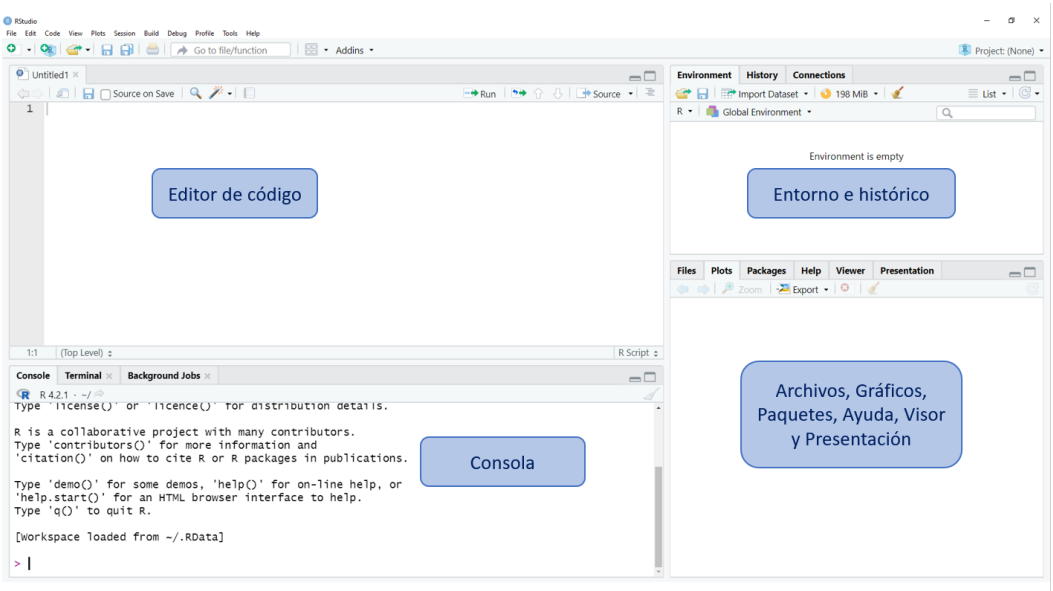

Figura 5.1: Interfaz de RStudio

- **Editor de código** (panel superior izquierda): Permite a los usuarios ver y editar varios archivos relacionados con el código, como .R, .rmd, .qmd, .py, .css o archivos de texto generales como .txt o .md. Cada archivo adicional que se abra se agrega como una nueva pestaña dentro del panel editor de código.
- **Entorno e histórico** (panel superior derecha): En la pestaña de **Entorno** se presentan una serie de objetos (variables, listas, dataframes...) que se han creado en el editor de código o leído de algún fichero. También muestra la memoria utilizada actualmente por la sesión de R activa y posee un icono de escoba para eliminar todos los objetos del entorno actual. En la pestaña de **histórico** se recogen todos los comandos que se ejecutaron en la sesión actual junto con la opción de búsqueda.
- **Consola** (panel inferior izquierda): Muestra el resultado al ejecutar el código de los diferentes scripts. Además, se puede escribir y ejecutar código directamente.
- **Archivos, Gráficos, Paquetes, Ayuda, Visor** y **Presentación** (panel inferior derecha): La pestaña de **Archivos** indica el directorio de trabajo y permite navegar por el explorador de archivos. La pestaña **Gráficos** proporciona la visualización de los diferentes gráficos ejecutados por el código. La pestaña **Paquetes** permite ver los paquetes R actualmente instalados y tiene una barra de búsqueda para buscar en la biblioteca actual de paquetes del CRAN. Hay un botón para la instalación de nuevos paquetes o la actualización de paquetes seleccionados existentes. La pestaña **Ayuda** permite un acceso cómodo a la ayuda de R pudiendo así buscar cualquier información, elemento o función. También se puede acceder a la ayuda de determinada función introduciendo en la **Consola** *?functionName* o *help(functionName)*. La pestaña **Visor** muestra contenido interactivo cuando éste es generado con el código. La pestaña **Presentation** se utiliza para mostrar diapositivas HTML generadas a través de paquete *revealjs*.

5.3. Shiny

Shiny es un paquete R que permite crear aplicaciones web interactivas que pueden ejecutar código R en el backend. Con Shiny, puede alojar aplicaciones independientes en una página web, añadir gráficos interactivos en documentos R Markdown o crear cuadros de mando (dashboards). También puede ampliar sus aplicaciones Shiny con CSS, widgets HTML y acciones de JavaScript.

Shiny permite codificar potentes aplicaciones web interactivas completamente en R. Para ello se debe crear en R una interfaz de usuario y un servidor, y Shiny compila su código en HTML, CSS y JavaScript necesarios para mostrar su aplicación en la web.

El framework Shiny permite recopilar valores de entrada de una página web, hacer que esas entradas sean fácilmente disponibles para la aplicación en R y tener los resultados del código R escritos como valores de salida en la página web. En su forma más simple, una aplicación Shiny requiere una función de servidor para realizar los cálculos y una interfaz de usuario. Las aplicaciones Shiny tienen dos componentes, una definición de interfaz de usuario (ui.R) y un script de servidor (server.R).[29]

En el Capítulo 6 se explica el paquete Shiny con más detalle.

5.4. ggplot2

La librería *ggplot2* es un paquete que proporciona comandos útiles para crear gráficos complejos a partir de datos en un dataframe. Mediante código se especifica qué variables trazar, cómo se muestran y las propiedades visuales generales. Por lo tanto, solo necesitamos cambios mínimos si los datos subyacentes cambian o si decidimos cambiar de un diagrama de barras a un diagrama de dispersión. Esto ayuda a crear gráficos de calidad con cantidades mínimas de ajustes y retoques.

Los gráficos de *ggplot2* se construyen capa por capa agregando nuevos elementos. Agregar capas de esta manera permite una gran flexibilidad y personalización de las tramas.

Para construir un gráfico con *ggplot2*, se utiliza la siguiente plantilla básica que se puede usar para diferentes tipos de gráficos:

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +  <GEOM_FUNCTION>() + theme()
```

Utiliza la función *ggplot* y enlaza el gráfico a un conjunto de datos específico utilizando el argumento data. Con la función *aes* se seleccionan las variables que se trazarán y se especifican cómo presentarlas en el gráfico, por ejemplo, como posiciones x/y o características como tamaño, forma, color... Después se añaden con las funciones *geom* que indica como representar los datos en el gráfico (puntos, líneas, barras). Para añadir un *geom* al gráfico hay que utilizar el operador "+". [30] La función *theme* sirve para cambiar los aspectos del estilo visual que son independientes a los datos de entrada. Varios ejemplos de estos aspectos son el tamaño de letra, el color de fuente o el color del fondo.

El paquete *ggplot2* permite construir casi cualquier tipo de gráfico los cuales se pueden personalizar de múltiples maneras diferentes, una gran cantidad de gráficos que se pueden realizar aparecen en la Figura A.1 del anexo.

5.5. Plotly

Plotly es un paquete de R de código abierto para crear gráficos interactivos basados en la biblioteca de gráficos JavaScript llamada *plotly.js*[31]

Plotly permite generar con código sus propios gráficos interactivos pero también permite con la función *ggplotly* añadir interactividad a un gráfico previamente creado con *ggplot2*.

Esta librería proporciona una amplia gama de tipos de gráficos, incluyendo gráficos de dispersión, gráficos de barras, gráficos de líneas, gráficos de áreas y mucho más. Además de su versatilidad en la creación de gráficos estáticos. Plotly brinda la capacidad de crear visualizaciones interactivas, lo que permite a los usuarios explorar y profundizar en los datos al interactuar con los gráficos. Esto incluye características como zoom, selección de datos, tooltips y leyendas personalizables. Con Plotly, los usuarios de R tienen una poderosa herramienta para visualizar y comunicar efectivamente información compleja y significativa a través de gráficos atractivos y dinámicos. [32]

Capítulo 6

Estructura y funcionamiento de una aplicación Shiny

En el capítulo anterior se realizó una breve descripción del paquete Shiny. En el presente capítulo se detalla la estructura y funcionamiento de una aplicación Shiny.

Para realizar una aplicación Shiny es necesario comprender la estructura que debe tener y comprender la división entre el código que genera la interfaz de usuario (*frontend*) y el código que contiene toda la lógica del programa (*backend*). Por tanto, en la primera parte se explica la estructura de una aplicación Shiny. En la segunda, se describe la programación reactiva de Shiny y cuáles son las estructuras reactivas más comunes. Por último, se muestra una guía para aprender a manipular el aspecto básico de una aplicación.

Para descargar el paquete Shiny del CRAN, se debe ejecutar la siguiente línea de código R desde la consola:

```
1 install.packages("shiny")
```

Una vez instalada, para cargar la librería hay que ejecutar la siguiente orden:

```
1 library(shiny)
```

6.1. Estructura de una aplicación Shiny

Las aplicaciones Shiny se dividen en dos partes [33, 34]:

- **UI**: La interfaz de usuario controla el diseño y la apariencia de la aplicación Shiny. La **UI** es el front-end de la aplicación.
- **Server**: El servidor contiene toda la lógica del programa. Es una función con dos argumentos, *input*, que obtiene las entradas del usuario y *output*, que envía a la interfaz los resultados. El **Server** es el back-end de la aplicación.

Ambas funciones, **UI** y **Server** se pueden escribir en R puro, pero también se puede incorporar código JavaScript, CSS o HTML.

La aplicación se sirve al cliente (usuario de la aplicación) a través de un host (Protocolo de Internet o dirección IP) y un número de puerto. Luego, el servidor mantiene abierta una conexión web para recibir solicitudes. La sesión R detrás de la aplicación se asegurará de que esta solicitud se traduzca en la interactividad deseada y envíe la respuesta, generalmente un objeto actualizado, como un gráfico o una tabla. [35]

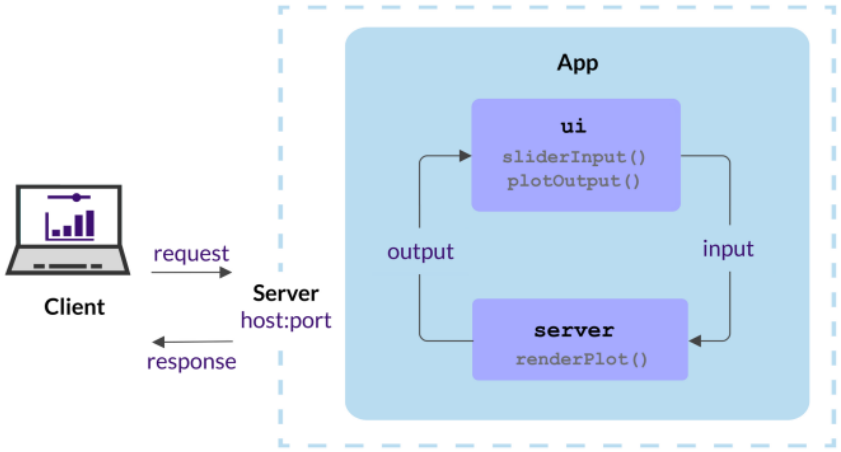

Figura 6.1: Esquema del comportamiento de una aplicación Shiny (extraído de [35])

En la Figura 6.1 se explica mediante un esquema el funcionamiento de la aplicación Shiny y la división entre la interfaz de usuario (**UI**) donde se define la apariencia de la aplicación y el servidor (**Server**) el cuál contiene toda la lógica del programa.

```
library(shiny)

ui <- fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)),
    mainPanel(
      plotOutput(outputId = "distPlot")
    )
  )
)

server <- function(input, output) {
  output$distPlot <- renderPlot({
    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")
  })
}

shinyApp(ui, server)
```


Figura 6.2: Ejemplo de código y aplicación resultante (extraído de [36])

En la Figura 6.2 se muestra el código y el resultado de ejecutar ese código que crea una aplicación Shiny. Se puede observar que primero se carga la librería Shiny, después se definen la interfaz de usuario (**UI**) y el servidor (**Server**). La orden que une los dos componentes es:

```
shinyApp(ui,server)
```

6.1.1. UI. Parámetros de entrada y de salida

Las aplicaciones Shiny se crean mediante objetos que recogen la entrada del usuario (*inputs*) y objetos que visualizan la salida (*outputs*):

- **Inputs**: Son las funciones que recogen la entrada del usuario. Todos tienen en común dos argumentos, *inputId* el cual da nombre al parámetro a nivel interno y *label* que define el título del elemento en el entorno gráfico. Cada tipo de input puede tener argumentos adicionales. Existen múltiples tipos diferentes de *inputs*, los más conocidos aparecen en la Figura 6.3

- **Outputs**: Son las funciones que muestran los parámetros de salida. Todos tienen en común un argumento llamado *outputId* que indica el nombre del objeto a mostrar. Existen múltiples tipos diferentes de *outputs*, los más conocidos aparecen en la Figura 6.4

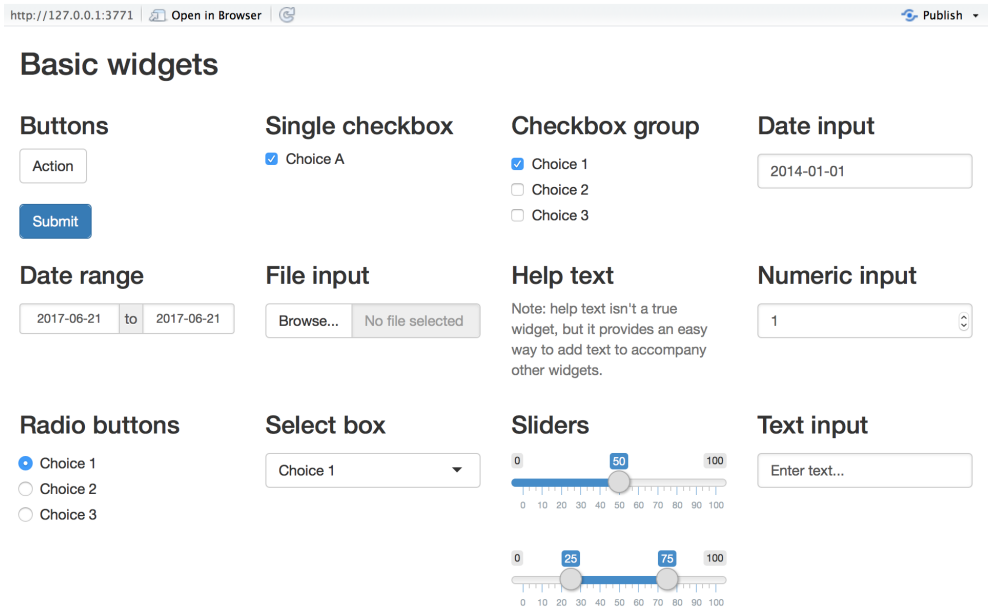

Figura 6.3: Listado de inputs (extraído de [37])

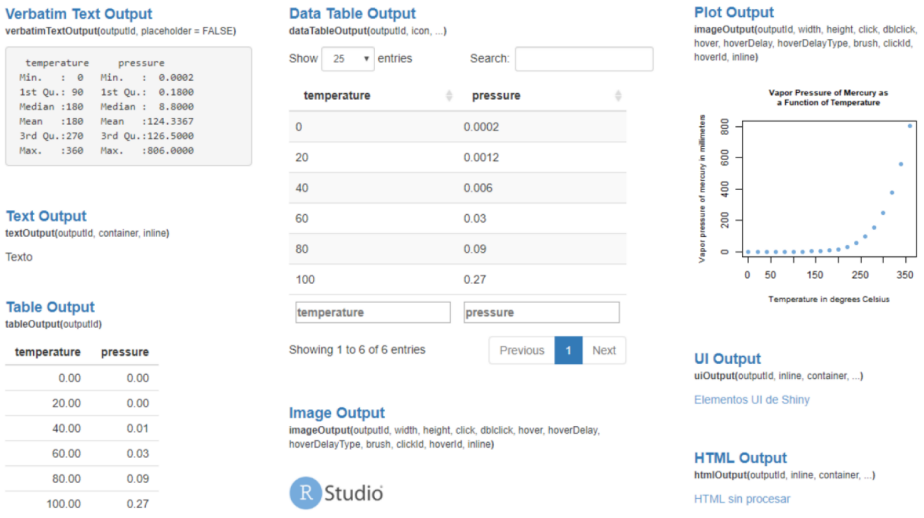

Figura 6.4: Listado de outputs (extraído de [36])

6.1.2. Server. Conexión entre los inputs y los outputs

Los outputs se definen en el **Server** y en la **UI** se configura su visualización. Para crear los parámetros de salida se deben realizar tres reglas:

1. Primera regla: Guardar los parámetros de salida en **output**$ acompañado del nombre del objeto el cuál servirá de referencia en el UI a la hora de visualizarlo.

2. Segunda regla: Construir objetos para renderizarlos con una función *render()*. Los objetos se renderizan con una función de renderización para poder visualizarlos en la aplicación. Hay una relación directa entre las funciones que visualizan los parámetros de salida **Output()* y las funciones que los renderizan *render*()* como se puede observar en la Figura 6.5.

3. Tercera regla: Acceso al valor de los inputs mediante **input$**. Por último para conectar los inputs y los outputs y que los segundos dependan del valor de los primeros, es necesario acceder a los parámetros de entrada mediante input$. de esta manera cuando cambie la entrada todo lo que esta unido a ese valor se actualiza y el gráfico cambia al instante, esto se conoce como reactividad.

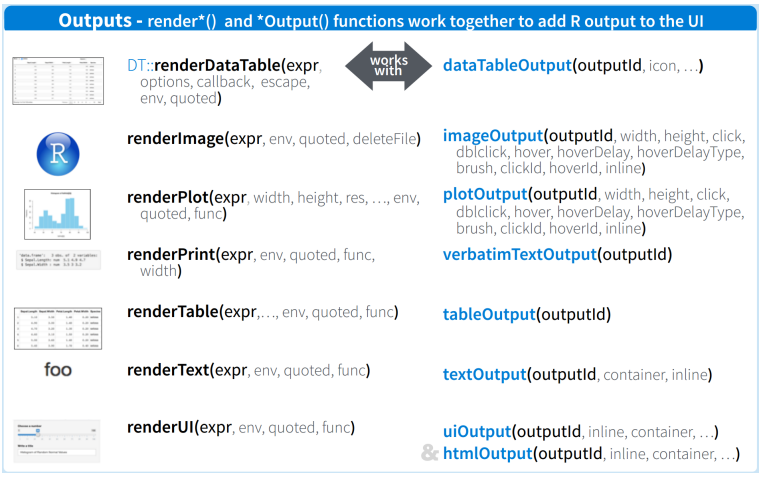

Figura 6.5: Relación entre outputs y funciones de renderización (extraído de [38])

Existen dos maneras para guardar una aplicación Shiny. La primera es mediante un único script como el ejemplo de Shiny 6.2 llamado app.R. La segunda opción consiste en separar UI y server en dos scripts dentro de un mismo directorio llamados ui.r y server.r respectivamente.

6.2. Programación reactiva

Shiny sigue el paradigma de la programación reactiva. En este paradigma de programación, las salidas se actualizan automáticamente cada vez que las entradas se modifican. En Shiny, hay tres tipos de objetos en la programación reactiva: fuentes reactivas, conductores reactivos y extremos reactivos[39]:

- ***Reactive sources*** (fuentes reactivas): Vienen determinadas por los componentes del argumento *input* con un nombre especificado en la interfaz de usuario (**UI**). Su modificación en tiempo de ejecución altera las salidas.
- ***Reactive conductors*** (conductores reactivos): Son los componentes de la aplicación Shiny que conectan las fuentes con las salidas. Estos conductores se declaran con las funciones reactivas *reactive* o *observe*.
- ***Reactive endpoints*** (puntos finales reactivos): Son los componentes de la aplicación que almacenan los resultados enviados a la interfaz gráfica. Al renderizar los resultados, se desencadena el cálculo de la salida. Se caracterizan por ser parte del argumento *output*.

6.3. Diseño de una aplicación Shiny

Shiny ofrece múltiples herramientas diseñar una aplicación.

6.3.1. Guía de diseño de aplicaciones

Shiny incluye una serie de facilidades para diseñar los componentes de una aplicación. Esta guía describe algunos de los tipos de diseños de una aplicación Shiny [40]:

- Diseño de barra lateral (Sidebar layout): Contiene una barra lateral (*sidebarPanel()*) con los parámetros de entrada junto con un área (*mainPanel()*) con el contenido de salida. Por defecto la barra lateral se sitúa a la izquierda pero se puede modificar para que esté a la derecha.

```
    sidebarLayout(

      sidebarPanel(
        # Inputs excluidas para que sea mas breve
      ),
      mainPanel(
        # Outputs excluidas para que sea mas breve
      )
    )

```

- Diseños de cuadrícula: Las filas son creadas por la función *fluidRow()* e incluyen columnas definidas por la función *column()*.
- Diseño basado en pestañas y en listas de navegación: Muy útiles para aplicaciones que necesitan subdividir su interfaz de usuario en secciones discretas. Esto se puede lograr utilizando la funciones *tabsetPanel()* y *navlistPanel()*.
- Creación de aplicaciones con múltiples páginas utilizando la función *navbarPage()*. La función *navbarPage()* crea una aplicación con una barra de navegación en la parte superior.

Algunos ejemplos de diseños se muestran en la Figura 6.6.

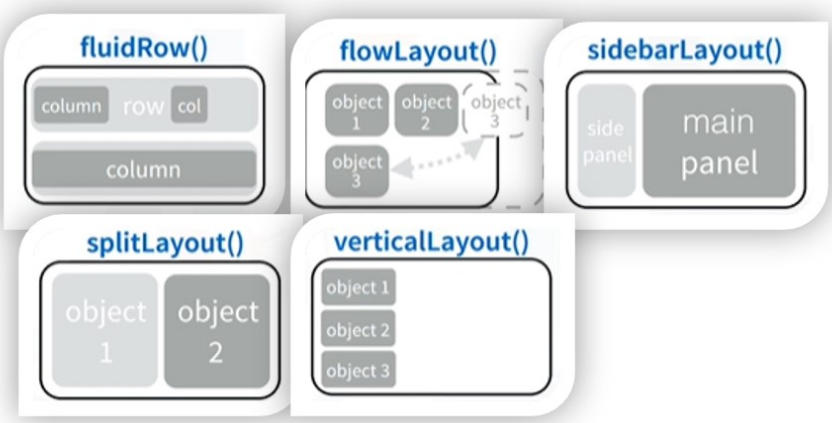

Figura 6.6: Ejemplos de diseños de aplicación (extraído de [41])

Añadir contenido estático

En la interfaz de usuario pueden introducirse elementos de HTML mediante la función de R *tags* las cuales permiten introducir múltiples tags entre los que destacan para crear secciones (*h1,..h6*), para agrupar contenido (*p, hr*), editar texto (*strong, br, code*) y añadir imágenes (*img*). Cada función en la lista crea una etiqueta HTML que puede usar para diseñar una aplicación Shiny.

6.3.2. Diseño de un Shiny Dashboard

Shinydashboard [42] es un paquete de R que genera de una forma sencilla cuadros de mando (dashboards) interactivos y visualmente atractivos. Diseñado específicamente para aplicaciones web basadas en Shiny, *shinydashboard* ofrece una amplia gama de herramientas y componentes predefinidos que permiten a los desarrolladores construir rápidamente interfaces de usuario modernas y personalizadas. Además, *shinydashboard* permite una fácil personalización mediante el uso de temas y estilos CSS, lo que garantiza que cada panel de control tenga una apariencia única y se ajuste a los requisitos específicos de cada proyecto. Esta es la interfaz de usuario mínima posible para crear un dashboard en Shiny:

```
library(shiny)
library(shinydashboard)
```

```
3
4  ui <- dashboardPage(
5    dashboardHeader(),
6    dashboardSidebar(),
7    dashboardBody()
8  )
9
10 server <- function(input, output) { }
11
12 shinyApp(ui, server)
13
```

El código anterior genera el siguiente resultado, (ver Figura 6.7).

Figura 6.7: Ejemplo de un dashboard y sus partes

Un dashboard tiene tres partes principales: un encabezado, una barra lateral y un cuerpo: [43]

- **Encabezado**: El encabezado se genera con la función *dashboardHeader()*. Este puede contener el título de la aplicación mediante el argumento title y también puede contener menús desplegables que se generan con la función *dropdownMenu()*. Hay tres tipos de menús: mensajes, notificaciones y tareas.
- **Barra lateral**: Una barra lateral se utiliza normalmente para una navegación rápida entre las diferentes pestañas de la aplicación. Puede contener las diferentes pestañas de la aplicación y entradas Shiny, como sliders y entradas de texto.
- **Cuerpo**: El cuerpo de una aplicación de *shinydashboard* puede contener cualquier contenido normal de Shiny. El componente básico de la mayoría de los dashboards es un archivo *box*. Las cajas a su vez pueden contener cualquier contenido como gráficos, texto, entradas...

  Existen múltiples tipos de cajas con diferentes estilo como la caja básica *box*, la caja con varias pestañas *tabBox* y las cajas para mostrar un valor numérico o de texto junto con un icono llamados *infoBox* y *valuebox*.

Capítulo 7

TennisBI: aplicación Shiny de visualización de datos de tenis

En este capítulo se presenta *TennisBI*, una aplicación Shiny de visualización de datos de tenis. *TennisBI* muestra información de tenistas, de torneos de tenis, de enfrentamientos entre tenistas y de jóvenes jugadores de tenis. También compara a los tenistas por países.

Este es el enlace para acceder a la visualización: `http://shiny1.eio.uva.es:3838/albertogonza/app/`

En el Anexo C se muestran muchas imágenes de la aplicación.

7.1. Pantalla de Carga

Durante la pantalla de carga de la aplicación se ha creado una animación con el logo de *TennisBI* el cuál se va rellenando a medida que se va cargando la aplicación, se ha realizado con la librería *waiter*.

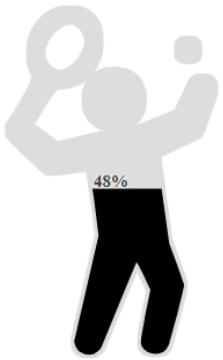

Figura 7.1: Pantalla de Carga con el logo de *TennisBI*

7.2. Página de *Inicio*

En la Figura 7.2 se muestra la página de ***Inicio*** de la aplicación.

Figura 7.2: Página de ***Inicio***

En ella se puede observar una breve descripción de los contenidos de la aplicación junto con un carrusel de tres imágenes (ver Figura C.3) las cuáles se van deslizando cada cierto tiempo de manera automática. El usuario puede en todo momento cambiar de imagen mediante dos botones en forma de flecha o mediante los puntos de debajo del carrusel de imágenes. Estas imágenes explican de una manera dinámica para el usuario qué características tiene la aplicación.

7.3. Encabezado de página

TennisBI ofrece todas sus visualizaciones y análisis tanto para tenis masculino (circuito ATP) como femenino (circuito WTA). En el encabezado de página se dispone de un botón para cambiar de circuito y que afecta a toda la aplicación. Esto evita la necesidad de incluir un botón en cada página para elegir el circuito. Este botón se sitúa a la derecha del encabezado y a la izquierda se sitúa el nombre de la aplicación junto con el logo, (ver Figura C.4).

7.4. Menú de navegación

En la Figura 7.3 se observa el menú de navegación, es un menú lateral a la izquierda de la pantalla el cuál puede ser plegable y mostrarse únicamente los iconos para dar mayor visibilidad a los gráficos de la aplicación. Para aumentar la usabilidad se han incluido iconos representativos de cada pestaña para facilitar al usuario la tarea de cambiar entre páginas.

Figura 7.3: Menú lateral (izquierda desplegado y en la derecha plegado)

Existen siete pestañas, una para cada una de las páginas de la aplicación, siendo estas sus descripciones:

- ***Inicio***: Muestra una breve descripción sobre el contenido que se va a encontrar el usuario en la aplicación.
- ***Tenista***: Muestra información gráfica sobre tenistas. El usuario elige un tenista. Tiene los apartados de información general, palmarés con los títulos obtenidos, ranking ATP o WTA, información por superficie, información de rivales y últimos partidos. Más detalles en el apartado 7.5.
- ***Torneo***: Muestra información gráfica sobre torneos. El usuario elige un torneo. Tiene los apartados de información general, estadísticas, duración de los partidos, últimos ganadores, información por año y animación títulos. Más detalles en el apartado 7.6.
- ***Enfrentamiento***: Muestra información gráfica comparando dos tenistas actuales. Definiendo como tenista actual aquellos que han jugado al menos un partido en los dos últimos años. El usuario elige dos tenistas. Tiene los apartados de comparación de victorias, comparación de los rankings ATP o WTA, comparación de variables tenísticas, comparación por superficie y últimos partidos entre los dos tenistas. Más detalles en el apartado 7.7.
- ***Futuras Promesas***: Muestra información gráfica sobre las futuras promesas del tenis junto con los jugadores actuales más similares. Definiendo como jugador promesa aquel con edad en el año actual inferior a 25 años. Tiene un apartado de comparación con el tenista más similar y después un ranking de tenistas más similares. Más detalles en el apartado 7.8.
- ***Países***: Muestra información por países. Tiene un apartado de comparación de los diez mejores tenistas por país y también un apartado de comparación de número de tenistas profesionales por país. Más detalles en el apartado 7.9.
- ***Información***: Muestra información sobre la aplicación y el origen de los datos. Más detalles en el apartado 7.10.

7.5. Página de *Tenista*

En esta página se muestra información gráfica sobre tenistas individuales. Tiene los apartados de información general, palmarés con los títulos obtenidos, ranking ATP o WTA, información por superficie, información de rivales y últimos partidos. Se han filtrado los tenistas sólo para esta página para que sólo se puedan elegir tenistas que hayan ganado al menos un título (ATP o WTA).

Se ha construido utilizando estructuras de *tabsetPanel* de la librería Shiny. Se crea una pestaña para cada apartado y además se representa mucha más información de una manera clara y estructurada. [26]

En la parte superior de la página para todos los apartados se incluye un selector de entrada única (*selectizeInput()*) para elegir el tenista del que se quiere visualizar la información, junto al selector de tenista se encuentra un slider (*sliderInput()*) para elegir el período de años del que se quiere visualizar la información. Este slider es doble y permite elegir el año inicial y final del rango desde el año 2000 hasta el año actual. Cualquier cambio en alguno de los selectores afecta a todos los gráficos de la página de ***Tenista*** de manera automática.

Por último, a la derecha de las entradas de datos se encuentran una bandera indicando la nacionalidad del tenista, esto se ha desarrollado con la función *flag()* de la librería *shinyflags*. [44]. Para ello se ha necesitado transformar la variable con la nacionalidad (`ioc`) ya que la función requería estar en código ISO [45]. Dicha transformación se ha implementado con la función *countrycode()* de la librería *countrycode*.[46]

En la esquina superior derecha de la página de ***Tenista*** se ha incluido un gráfico interactivo para visualizar el porcentaje de victorias del tenista. Se trata de un **gráfico de donut** interactivo con una etiqueta en medio con el valor del porcentaje de victorias. Posee un tooltip: cuando pasas el ratón por encima de la zona verde de victorias se muestra el número total de victorias y si se pasa por la zona roja del gráfico se muestra el número total de derrotas. El gráfico sin interactividad se ha creado con la librería *ggplot2* [47] y para lograr la interactividad ha sido con la función *girafe()* de la librería *ggiraph*. [48]

7.5.1. Apartado información general

La primera pestaña al abrir la página de ***Tenista*** es la de *información general (Inf. General)*, (ver Figura 7.4), la cuál incluye una tabla con el nombre, mano hábil, altura en centímetros, fecha de nacimiento, número de partidos, número de torneos profesionales (ATP o WTA) ganados, número de Grand Slams ganados y la superficie destacada del tenista. La superficie destacada del tenista es la superficie en la que tiene mejor porcentaje de victorias.

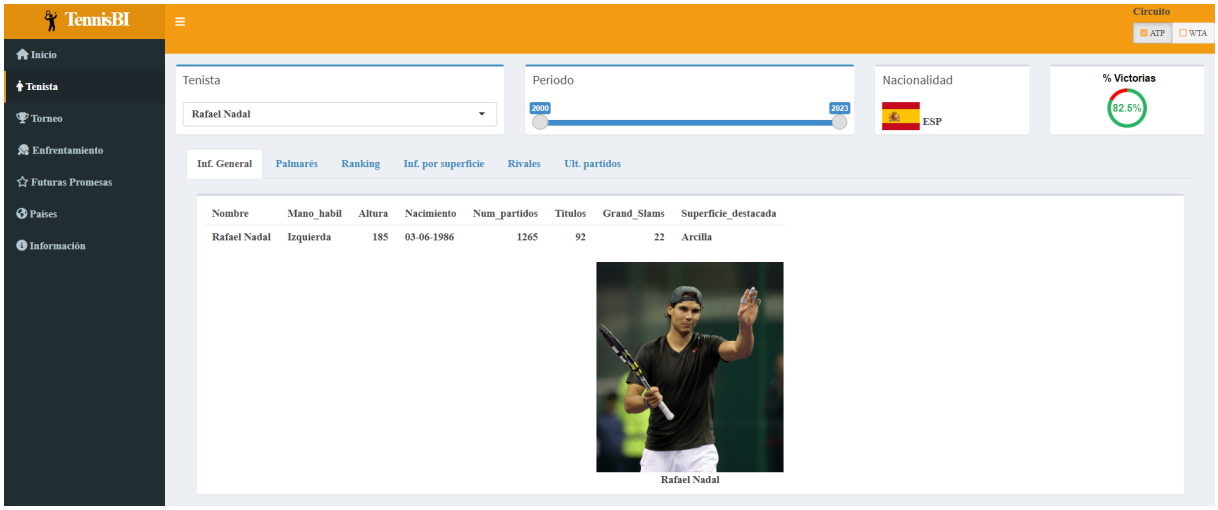

Figura 7.4: Apartado Información general de la página ***Tenista***

Debajo de la tabla se encuentra una imagen del tenista seleccionado. En los ficheros originales de información de tenistas, **dataPlayers**, disponen de una columna llamada `wikidata_id` con el identificador del tenista en Wikidata. Con este identificador se realiza una consulta SPARQL para obtener el enlace a la imagen del tenista en Wikidata. [16]

Wikidata es una base de datos colaborativa y abierta que almacena información estructurada en forma de entidades, atributos y valores. Fue creada por la Fundación Wikimedia.

Las consultas SPARQL son una forma de acceder y manipular la información almacenada en Wikidata. SPARQL es un lenguaje de consulta estandarizado para la web semántica que permite realizar consultas complejas sobre conjuntos de datos RDF (Resource Description Framework). Wikidata utiliza RDF como formato de almacenamiento de datos, lo que permite que los usuarios realicen consultas con SPARQL para obtener información de forma eficiente y flexible.

Por ello para obtener el enlace a la imagen de los tenistas que está almacenada en Wikidata se utiliza la librería *WikidataQueryServiceR* que permite realizar consultas SPARQL en R. [49]

Con el siguiente código en R obtiene de un identificador de tenista en la Wikidata el enlace a su imagen en la Wikidata.

```
  wikidata_id<-data[data$name_complete==playerName,"wikidata_id"]

query<- paste('SELECT ?image WHERE { BIND(wd:',wikidata_id,' AS ?item) ?item wdt:P18 ?
    image.BIND(STRAFTER(wikibase:decodeUri(STR(?image)), "http://commons.wikimedia.org/
    wiki/Special:FilePath/") AS ?fileTitle)}',sep="")

image<- query_wikidata(query,format="smart")
        src <- as.vector(image)$image[1]

```

7.5.2. Apartado palmarés

Este apartado muestra información sobre todos los torneos ganados del tenista en el periodo seleccionado, (ver Figura 7.5).

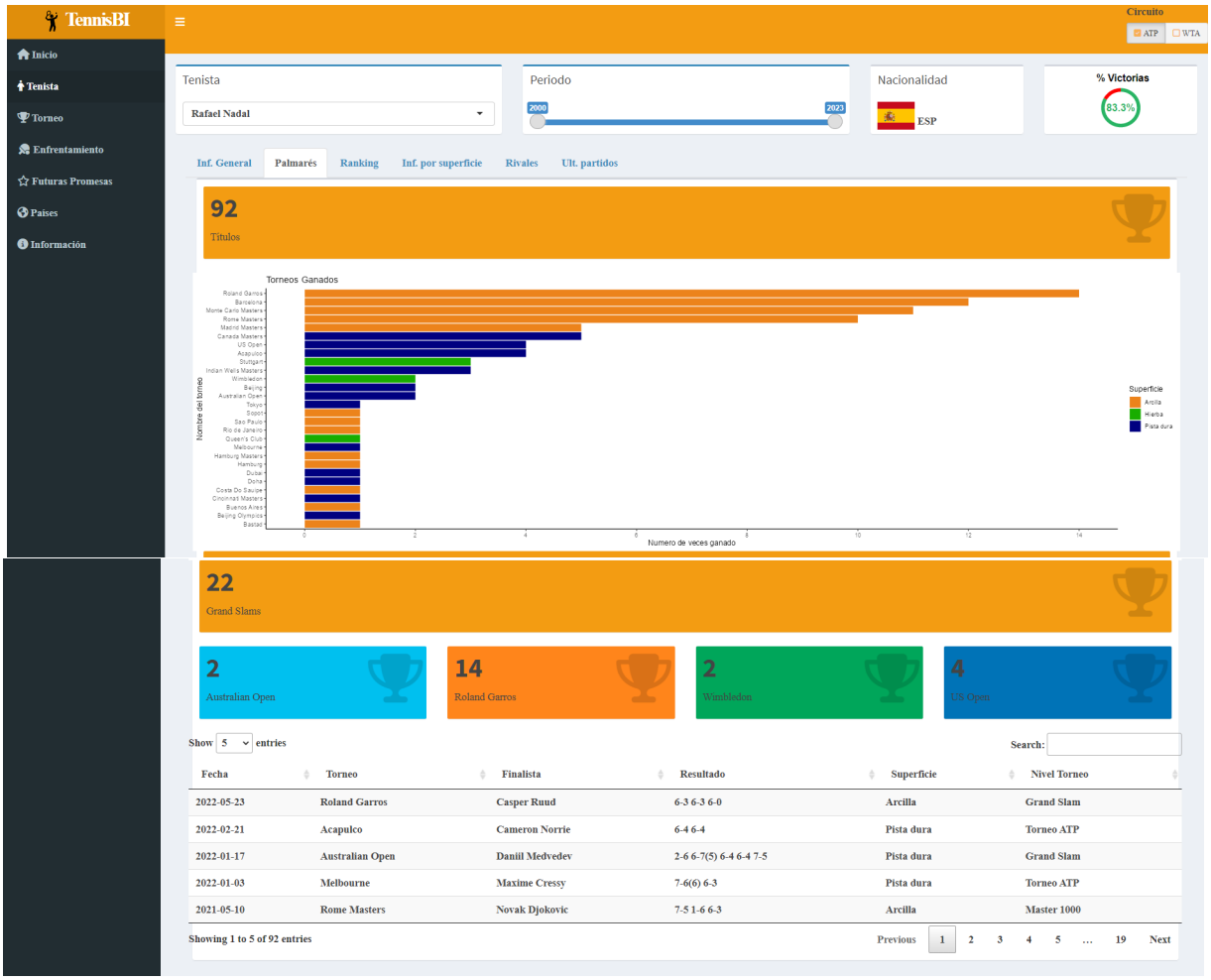

Figura 7.5: Apartado palmarés de la página ***Tenista***

En la parte superior incluye una etiqueta con el total de títulos ganados. Tras ello hay un gráfico que explica de forma detallada qué torneos y cuantas veces ha ganado cada torneo. Esto se ha implementado mediante un **gráfico de barras** con la librería *ggplot2* [47] en el que en el eje X indica la cantidad de veces que se ha ganado un torneo, en el eje Y se indica el nombre del torneo y para la leyenda de los colores de las barras se indica la superficie de la pista (hierba, pista dura, arcilla o moqueta), se trata de un gráfico interactivo gracias a la librería *plotly*. [31] Los torneos aparecen ordenados de mayor a menor valor de la variable.

Después del gráfico de torneos ganados se ha incluido una etiqueta para representar el número de Grand Slams ganados y luego cuatro etiquetas una para cada Grand Slam.

Por último al final se encuentra una tabla con la información de todos los torneos que ha ganado indicando fecha, nombre del torneo, finalista, resultado, superficie y nivel del torneo. Esta tabla se ha construido con la librería *DT* [50] la cual permite al usuario ordenar las columnas, buscar mediante teclado, mostrar más filas (por defecto la aplicación muestra 5) y navegar entre páginas. La tabla está ordenada por fecha (más recientes primero).

7.5.3. Apartado ranking

Esta sección muestra mediante un **gráfico de líneas** la evolución temporal del ranking del tenista, (ver Figura C.8). Se implementa con la librería *ggplot2* y *plotly* para que sea interactivo. En el eje X se muestra la fecha y en el eje Y la posición el ranking de manera invertida ya que cuánto menor es el ranking es mejor el tenista. El número de ticks en el eje Y se actualiza cuando se cambia el tenista o el periodo. Se han incluido dos líneas horizontales doradas una continua cuando ranking = 1 para observar cuando ha

sido el mejor tenista del mundo y luego una línea discontinua cuando ranking = 10 para observar cuando el tenista ha estado en el top 10.

7.5.4. Apartado información por superficie

En este apartado se segrega la información de victorias según las superficies y se desarrollan dos gráficos que pueden cambiar mediante un selector *selectizeInput()*, (ver Figura C.9).

- Total de victorias y derrotas para cada una de las cuatro superficies: Se implementa mediante un **gráfico de barras apiladas** no interactivo con la librería *ggplot2*. En el eje X se encuentra la superficie del torneo y en el eje Y el total de victorias y derrotas, el color verde indica las victorias y el color rojo indica las derrotas.
- Porcentaje de victorias y derrotas para cada una de las cuatro superficies: Se implementa mediante un **gráfico de barras apiladas de porcentaje** no interactivo con la librería *ggplot2*. En el eje X se encuentra la superficie del torneo y en el eje Y el porcentaje de victorias y derrotas, el color verde indica las victorias y el color rojo indica las derrotas.

7.5.5. Apartado rivales

En este apartado se muestra información de los rivales con los que más ha jugado, los rivales a los que más ha ganado y los rivales con los que más ha perdido, (ver Figura C.10).

En la parte superior se crea un selector (*selectizeInput()*) para elegir la variable que se quiere representar ya sea número de partidos jugados, número de victorias o número de derrotas contra el tenista seleccionado.

Se crea un **gráfico de barras (bar chart)** interactivo con la librería *ggplot2* y *plotly*. En el eje X se representa la variable seleccionada por el selector y en el eje Y se indican los nombres de los tenistas rivales. Esta visualización está filtrada únicamente para que aparezcan diez rivales. El color se ha elegido en función de la variable seleccionada eligiendo el color azul para los tenistas con los que ha jugado más partidos, color verde para los tenistas a los que más ha ganado y por último color rojo para los tenistas con los que más ha perdido. Las barras aparecen ordenadas de mayor a menor valor de la variable.

7.5.6. Apartado últimos partidos

En este último apartado se muestran los últimos partidos jugados por el tenista en formato tabla con la siguiente información: Fecha, nombre del torneo, indicador de si ganó o perdió el partido, el nombre del rival, el resultado, la superficie de la pista y la ronda del torneoC.11. Para crear una comprensión rápida del usuario se han coloreado de verde las filas de los partidos que ha ganado el tenista y de rojo los partidos que ha perdido. Esta tabla aparece ordenada por fecha (más recientes primero) pero en la etapa de exploración se observó que todos los partidos de un mismo torneo se añaden con la misma fecha (fecha de comienzo del torneo) por ese motivo para los partidos con la misma fecha se vuelve a ordenar según la ronda del torneo.

7.6. Página de *Torneo*

La página de ***Torneo*** representa información relacionada con torneos profesionales de tenis.

En la parte superior de la página para todos los apartados incluye un selector de entrada única (*selectizeInput()*) para elegir el torneo del que quiere visualizar la información.

Tiene los apartados de información general, estadísticas , duración de los partidos, últimos ganadores, información por año y una animación de los ganadores de cada año, se ha vuelto a utilizar las estructuras de *tabsetPanel*.

7.6.1. Apartado de Inf. General

En este apartado se representa la información general del torneo en formato tabla. Se incluye nombre del torneo, superficie de la pista, nivel del torneo y total de partidos que se han jugado allí. Debajo de la tabla, se incluye una imagen de una pista de tenis con el color de la superficie del torneo seleccionado, (ver Figura C.12).

7.6.2. Apartado de Estadísticas

En la parte superior de este apartado se incluye un selector (*selectizeInput()*) para escoger entre el top de tenistas con más títulos, el top de tenistas con más partidos ganados o el top tenistas con más efectividad. Siendo la efectividad $= \frac{victorias}{partidos} \cdot 100\,\%$, (ver Figura C.13).

Se ha vuelto a optar como gráfico un **gráfico de barras** interactivo con *ggplot2* y *plotly*, representando en el eje X la variable seleccionada y en el eje Y el nombre de los tenistas. Esta visualización está filtrada únicamente para que aparezcan diez tenistas. Como paleta de color se ha escogido una que establece el color verde a los tenistas con mayor valor en la variable seleccionada y rojo a los que tienen menor valor. Las barras aparecen ordenadas de mayor a menor valor de la variable.

7.6.3. Apartado de Duración

Se ha incluido un apartado para comparar la duración de los partidos del torneo frente a la duración de los partidos de torneos del mismo nivel, torneos en la misma superficie y una comparación global con todos los torneos, (ver Figura 7.6).

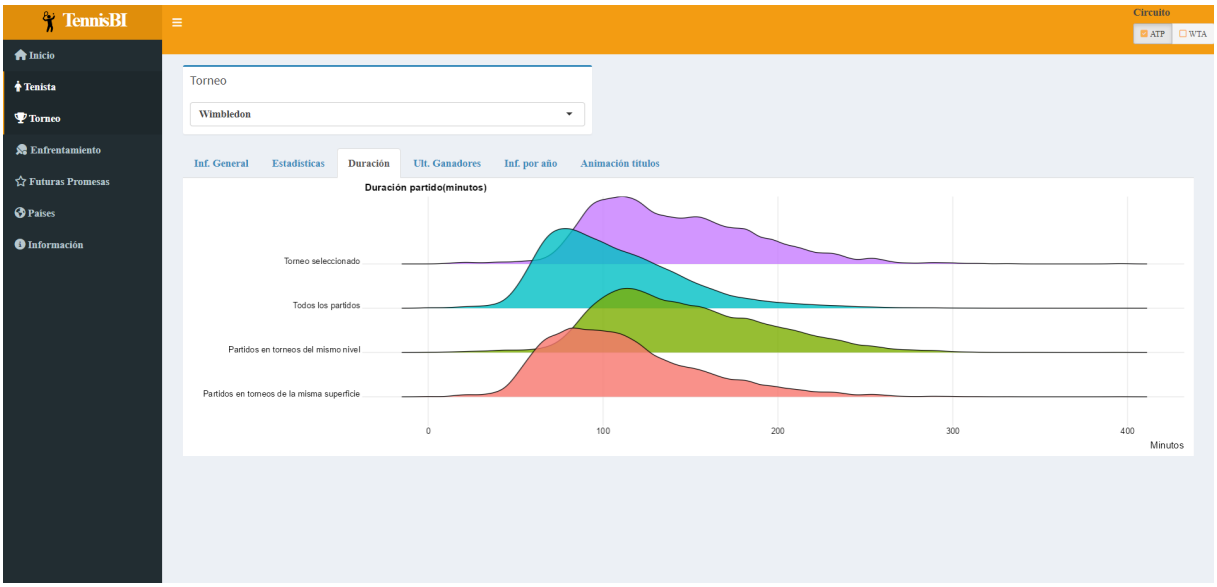

Figura 7.6: Apartado Duración de la página ***Torneo***

Para visualizar esta información se podría haber elegido un gráfico de densidad pero al haber cuatro densidades parecidas no se iba a poder comparar correctamente es por ello que se ha elegido un gráfico menos común pero que permite comparar mejor varias densidades el cuál es el **gráfico de Ridgeline (Ridgeline plot)** el cuál consiste en representar todas las densidades pero en vez de superpuestas, se representan una encima de otra sin sobreponerse.

En el eje X se representa la duración del partido en minutos y en el eje Y indica si se trata de partidos del torneo seleccionado, partidos de torneos del mismo nivel que el torneo seleccionado, partidos de torneos de la misma superficie que el torneo seleccionado y por último partidos de todos los torneos.

7.6.4. Apartado de Ult. Ganadores

En este apartado se incluye los últimos ganadores del torneo en formato tabla indicando la fecha, el ganador del torneo, el perdedor de la final, el resultado y la duración en minutos de la final. La tabla se encuentra ordenada por fecha (más reciente primero), (ver Figura C.15).

7.6.5. Apartado de Inf. por año

Cuando se selecciona este apartado en la parte superior de la página por encima de los paneles de los apartados, a la derecha del selector de torneo aparece un slider (*sliderInput()*) para elegir un año concreto.

A su vez aparecen tres nuevos subapartados creados nuevamente con *tabsetPanel()*.

- **Podium**: Se ha creado un podium en el que en el centro se sitúa el ganador, a la izquierda el segundo clasificado y a la derecha los dos semifinalistas simulando un podio olímpico, (ver Figura 7.7). Para simular este podio se ha creado un **gráfico de barras** no interactivo con la librería *ggplot2*, como eje Y se ha indicado el número de partidos ganados así el ganador será el que se sitúe más alto, luego el finalista y por último los semifinalistas. Se ha escogido el color oro para el ganador, plata para el finalista y bronce para los semifinalistas.

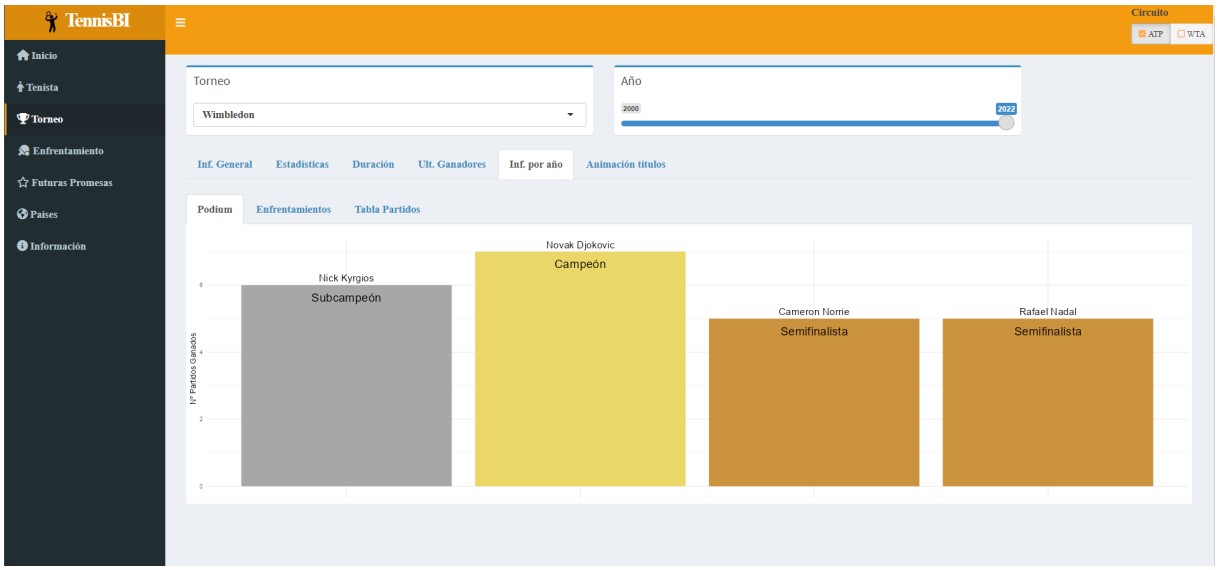

Figura 7.7: Subapartado Podio del apartado Información por año de la página ***Torneo***

- **Enfrentamientos**: Se ha creado un gráfico representando todos los enfrentamientos desde cuartos de final, (ver Figura 7.8).

  Se ha utilizado la función *draw_bracket()* de la librería *SwimmerR* que no está disponible en el CRAN de R y ha debido descargarse de un repositorio de Github. [51] Esta función está desarrollada para crear el *bracket* de enfrentamientos en el torneo universitario de baloncesto de Estados Unidos llamado "March Madness". [52]

  La función de *draw_bracket()* recibe como argumentos el campeón (*champion*), la ronda 3 (*round_three*) en nuestro caso los finalistas, la ronda 2 (*round_two*) en nuestro caso los semifinalistas y todos los participantes (*teams*). Está función está pensada para que en el argumento de *teams* introducirle los participantes ordenados por la semilla, pero en nuestros ficheros originales estaba casi siempre ausente (`winner_seed` y `loser_seed`). Se ha ajustado la función a los datos disponibles construyendo desde arriba hacia abajo el bracket, comenzando con el ganador, después finalista, después semifinalistas...

  Se ha programado una función llamada *getRivalsInRound()* que dado los datos de un torneo, el nombre de un tenista y la ronda te devuelve el nombre del contrincante.

```
    getRivalsInRound<-function(data,name,round){
  return(data[data$round==round & data$winner_name==name,"loser_name"])
}

```

De manera iterativa se ha ido creando los vectores de de semifinalistas y cuartofinalistas de la siguiente manera:

```
    champion <- data[data$round=="F","winner_name"]
    finalists <- c(champion, getRivalsInRound(data,champion,"F"))
    semifinalLosers<-sapply(finalists,getRivalsInRound,data=data,round="SF")
    semifinalists <- c(finalists,semifinalLosers[2],semifinalLosers[1] )
    quarterFinalLosers<-sapply(semifinalists,getRivalsInRound,data=data,round="QF"
    )
    quarterfinalists <- c(semifinalists,quarterFinalLosers[4],
                          quarterFinalLosers[3],
                          quarterFinalLosers[2],
                          quarterFinalLosers[1])
    draw_bracket(teams = quarterfinalists,
                 round_two = semifinalists,
                 round_three = finalists,
                 champion = champion,
                 title=title,
                 text_size = 1)

```

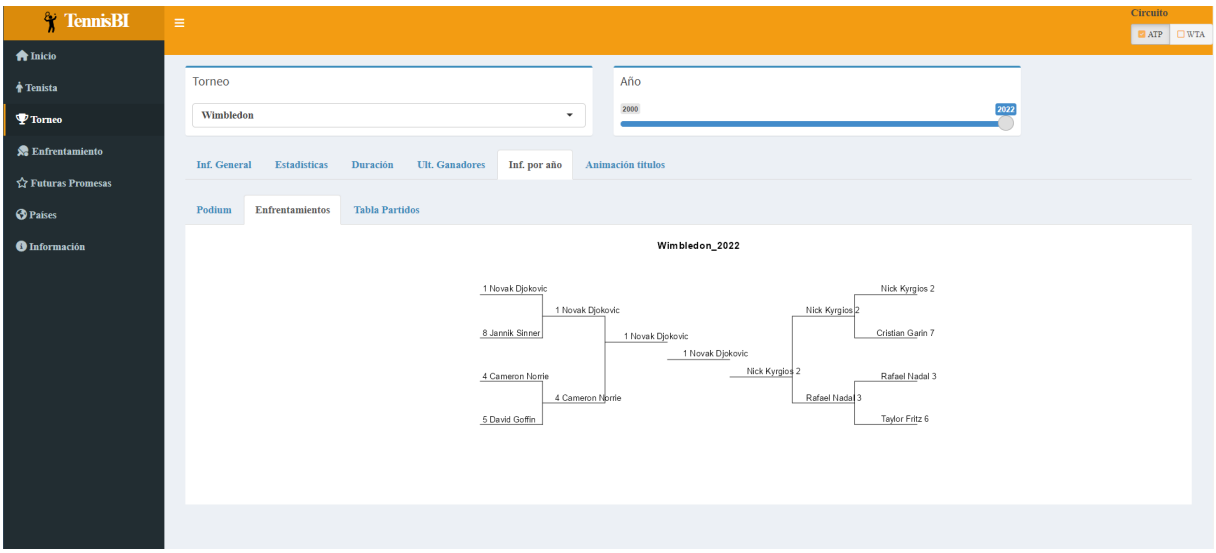

Figura 7.8: Subapartado Enfrentamientos del apartado Información por año de la página ***Torneo***

- **Tabla partidos**: Representa en formato tabla todos los partidos del torneo en el año seleccionado ordenados por ronda del torneo (ronda más altas como la final aparecen primero), en la tabla aparece la fecha, el nombre del ganador y del perdedor del partido, el resultado del partido, la ronda del torneo y la duración del partido en minutos, (ver Figura C.18).

7.6.6. Apartado de Animación títulos

Se crea un **gráfico de barras** animado representando los ganadores del torneo cada año. En el que el eje X se representa el número de títulos que ha ganado un tenista del torneo seleccionado. En el eje Y se indica el nombre del tenista y la variable con la que crea la animación es el año. Por lo tanto con este gráfico podemos observar la evolución de los máximos ganadores del torneo seleccionado según avanza el tiempo.

Se utiliza la librería *gganimate* [53] que permite crear animaciones en gráficos, (ver Figura C.19).

7.7. Página de *Enfrentamiento*

Esta página permite comparar dos tenistas actuales mediante múltiples variables. Definiendo como tenista actual aquellos que han jugado al menos un partido en los dos últimos años. Para ello se ha distribuido la página en tres columnas mediante la función *column()* de Shiny.

En la primera columna a la izquierda es la relacionada con el tenista 1. Se incluye en la parte superior de la columna un selector (*selectizeInput()*) para elegir tenista. Debajo del selector se incluye una navegación de pestañas (*tabsetPanel()*) para escoger entre mostrar la imagen del tenista o información básica del tenista en formato tabla vertical como es su nacionalidad, mano hábil, altura, nacimiento, número de partidos, porcentaje de victorias, ranking actual, títulos profesionales, número de Grand Slams y superficie destacada. Esta columna aparece con un borde y un fondo azul ya que será el color para representar al tenista 1 en algunos gráficos.

En la columna central se crea una navegación de pestañas (*tabsetPanel()*) con seis apartados apareciendo por defecto el de Victorias. Los apartados son Victorias, Ranking, Comparación, Información por superficie y partidos.

Por último, en la tercera columna es la relacionada con el tenista 2, tiene el mismo formato que la primera columna.

7.7.1. Apartado Victorias

En este gráfico se representa el porcentaje de victorias del tenista 1 frente al tenista 2 mediante un **gráfico de donut (donut chart)** igual que el utilizado en la página de ***Tenista*** 7.5, (ver Figura C.20 y Figura C.21).

7.7.2. Apartado Ranking

Este gráfico se ha creado de la misma manera que el desarrollado en el apartado de ranking en la página de ***Tenista*** 7.5.3 pero se han superpuesto los rankings de ambos tenistas. Tenista 1 es la línea de color azul y tenista 2 es la línea de color amarillo al igual que sus respectivos fondos de columna. Al utilizar *plotly* se puede seleccionar pulsando en la leyenda por si solo se quiere ver el ranking de uno de los tenistas, (ver Figura 7.9).

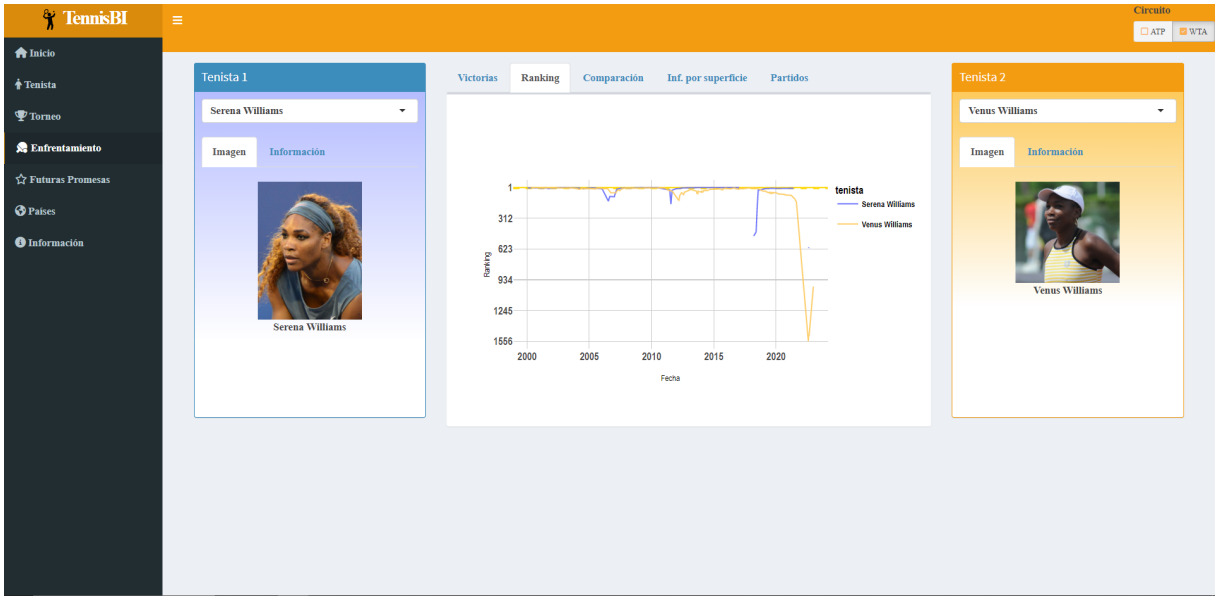

Figura 7.9: Apartado Ranking de la página ***Enfrentamiento***

7.7.3. Apartado Comparación

Dentro de este apartado mediante *tabsetPanel* se han creado otros cuatro subapartados. Están relacionados con las variables creadas en el apartado 4.2.1. Los cuatro subapartados son los siguientes:

- **Gráfico**: Para comparar multiples variables numéricas de dos tenistas se ha utilizado un **gráfico de radar (radar chart / spider chart)** como el mostrado en la Figura 4.1.

  En él se muestra las doce variables creadas, de color azul se muestra los valores del tenista 1 y en color amarillo los del tenista 2, ver Figura 7.10. Cuanto mayor valor tiene una variable más alejado del centro se encontrará el punto correspondiente. Para desarrollar este gráfico se ha utilizado la librería *fmsb*. [54]

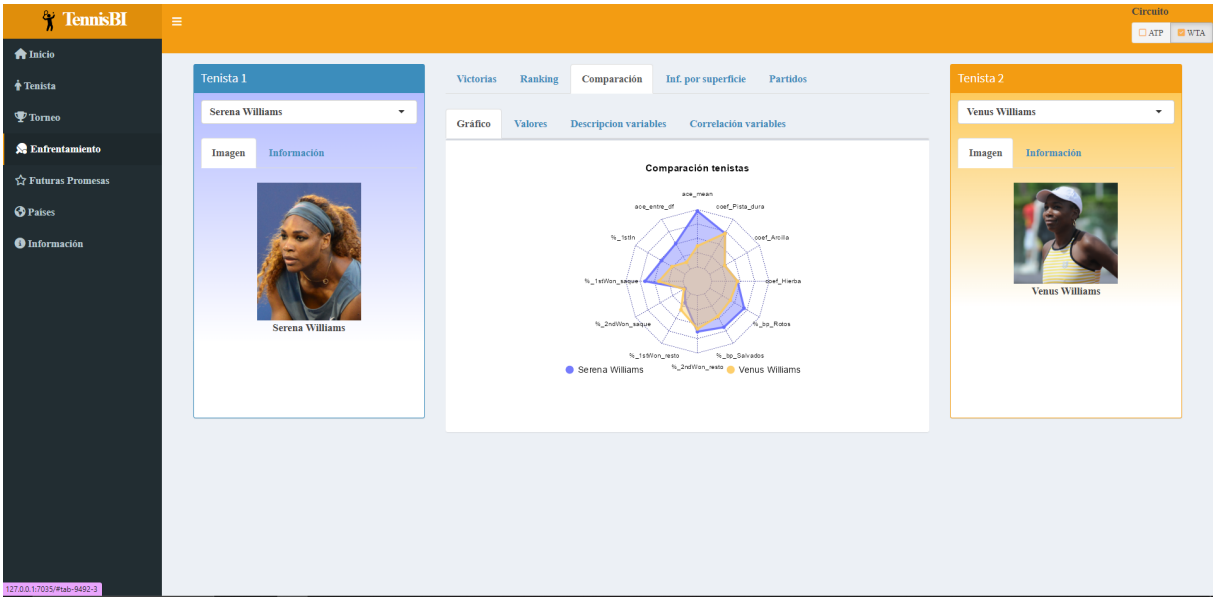

Figura 7.10: Subapartado Gráfico del apartado Comparación de la página ***Enfrentamiento***

- **Valores**: Al no poder añadir interactividad en el gráfico anterior se creó un apartado para mostrar los valores del gráfico de radar previo mediante una **tabla**, (ver Figura C.24).
- **Descripción variables**: Se ha creado una tabla con las descripciones las variables utilizadas en el apartado de **Comparación**, (ver Figura C.25).
- **Correlación variables**: Se ha creado un **gráfico de correlación** para representar la correlación entre las variables utilizadas en el apartado de **Comparación**. Para implementar este gráfico se ha utilizado la librería *GGally*[55], (ver Figura C.26).

7.7.4. Apartado Inf. por superficie

Para comparar las victorias y derrotas en cada superficie entre los dos tenistas se han creado dos gráficos al igual que en el apartado de Inf. por superficie de la página de ***Tenista*** 7.5.4 para comparar el total de victorias del tenista 1 frente al tenista 2 en cada superficie y el porcentaje de victorias y derrotas del tenista 1 frente al tenista 2, (ver Figura C.27).

7.7.5. Apartado Partidos

Se ha creado un tabla que muestra el resultado de los últimos partidos del tenista 1 frente al tenista 2, indicando la fecha, nombre del torneo, quién ha ganado, el resultado del partido y la ronda del torneo. Para crear una comprensión rápida del usuario se han coloreado de color azul las filas de los partidos que ha ganado el tenista 1 y de amarillo las filas de los partidos que ha ganado el tenista 2. La tabla se encuentra ordenada por fecha (más reciente primero), (ver Figura C.28).

7.8. Página de *Futuras promesas*

En esta página está centrada en las futuras promesas del tenis, definiendo futura promesa como aquellos tenistas con edad en el año actual inferior a 25 años.

En ella se puede observar información del tenista promesa, comparación respecto a su jugador más similar obtenido con el algormitmo de similaridad definido en el Capítulo 4 y top de tenistas más similares al tenista promesa.

La distribución de la página de ***Futuras promesas*** está elaborada con la misma configuración que la página de ***Enfrentamiento*** del apartado 7.7 con tres columnas.

La primera columna tiene un selector con los jóvenes tenistas y tiene debajo un navegador para elegir entre la imagen del joven tenista o una tabla con su información básica.

En la columna central se han creado dos apartados apareciendo por defecto el de tenista más similar. El otro apartado es el de Top tenistas más similares.

En la tercera columna se muestra al tenista más similar teniendo en cuenta el algoritmo de similaridad definido en el Capítulo 4. Tiene un navegador para elegir entre la imagen del tenista similar o una tabla con la información básica.

7.8.1. Apartado Tenista más similar

Contiene los mismos cuatro subapartados (Gráfico, Valores, Descripción variables y Correlación variables) que el apartado 7.7.3 de **Comparación** en la página de ***Enfrentamiento***, (ver Figuras C.29, C.30, C.31 y C.32).

La información representada se obtiene a partir de información de partidos de los dos últimos años.

7.8.2. Apartado Top Tenistas más similares

Mediante un **gráfico de barras (bar chart)** interactivo creado con la librería *ggplot2* y *plotly* se ha creado un diagrama de barras mostrando los diez tenistas más similares a la futura promesa seleccionada, (ver Figura 7.11).

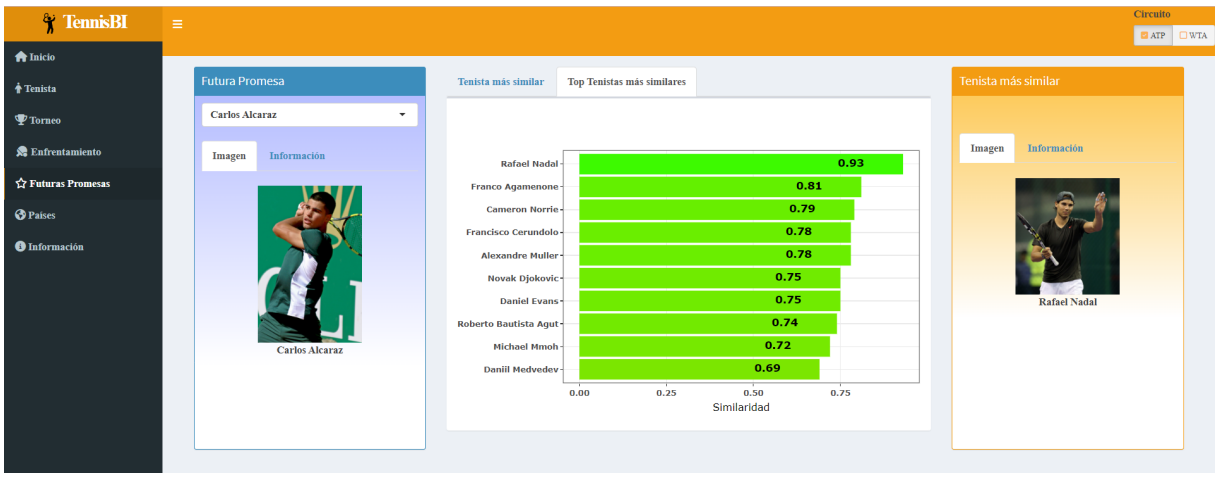

Figura 7.11: Apartado Tenistas más similares de la página ***Futuras promesas***

7.9. Página de *Países*

En esta página se presenta la información agrupada por países, se incluyen dos apartados mediante dos pestañas creadas con *tabsetPanel()*.

7.9.1. Apartado Mejores Tenistas por País

En esta pestaña se muestra información tenística sobre los diez mejores tenistas de un país en un periodo de tiempo variable, (ver Figura 7.12).

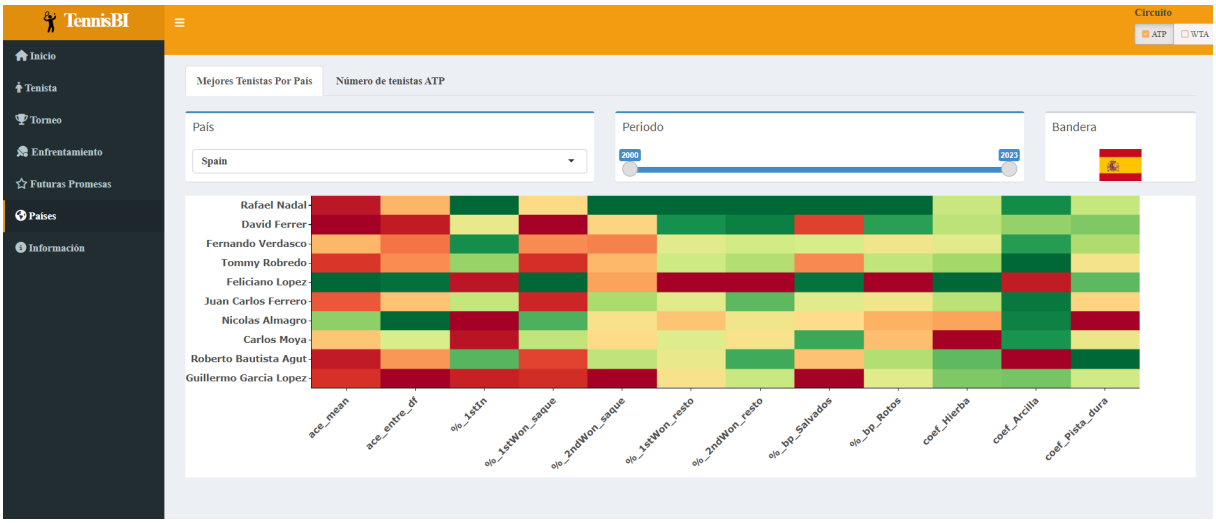

Figura 7.12: Apartado Mejores tenistas por país de la página ***Países***

Para ello en la parte superior existe un selector único de país mediante un *selectizeInput()* junto al selector de país se encuentra un slider (*sliderInput()*) para elegir el periodo de años del que se quiere visualizar la información. Este slider es doble y permite elegir el año inicial y final del rango desde el año 2000 hasta el año actual. Cualquier cambio en alguno de los selectores afecta al gráfico de manera automática. También se muestra la bandera del país seleccionado.

Debajo del selector del país, slider de periodo de años y bandera se encuentra el gráfico de los diez tenistas con más victorias del país seleccionado en los años seleccionados.

De cada tenista se representan las doce variables definidas en el apartado 4.2.1 y para visualizarlas correctamente en un único gráfico se ha optado por el gráfico llamado **mapa de calor (heatmap)**. Como filas del mapa de calor se encuentran los tenistas y cada columna es una variable. La escala de color muestra con un color cercanos al rojo un valor bajo en la variable y colores cercanos al verde con valores altos en las variables.

El gráfico es interactivo y se ha desarrollado con la librería *heatmaply* [56], para no sobrecargar el gráfico con el valor de cada tenista en cada variable se ha optado por crear un tooltip o etiqueta interactiva que aparece cuando se pasa el ratón por una celda la cual indica el valor.

Las variables aparecen ordenadas de tal manera que primero aparecen las variables relacionadas con el saque (`ace_mean`, `ace_entre_df`, `%_1stIn`, `%_1stWon_saque`, `%_2ndWon_saque`), después las variables relacionadas con el resto (`%_1stWon_resto`, `%_2ndWon_resto`), después las variables relacionadas con los puntos de break (`%_bp_Salvados`, `%_bp_Rotos`) y por último los coeficientes en cada superficie (`coef_Hierba`, `coef_Arcilla` y `coef_Pista_dura`).

7.9.2. Apartado Número de tenistas ATP/WTA

En este apartado se ha creado un mapa interactivo global mostrando el número de tenistas profesionales por cada país, (ver Figura 7.13).

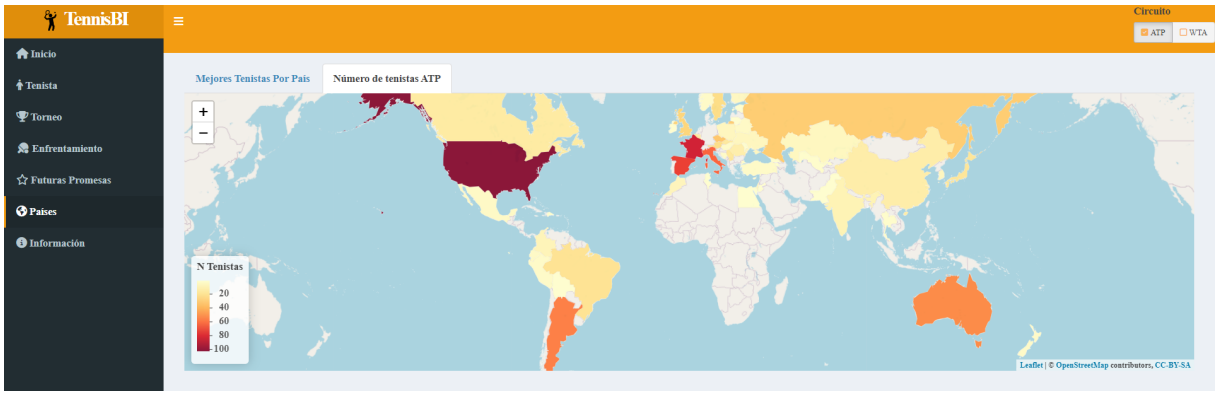

Figura 7.13: Apartado Número tenistas de la página ***Países***

Se trata de un mapa interactivo creado con la librería *leaflet* [57] que permite con un tooltip ver el número de tenistas cuando se pasa el ratón por encima de un país. Para crear el mapa primero es necesario descargar un archivo con los datos cartográficos (geoData) disponibles en la API de *Thematic mapping org*[58].

7.10. Página de *Información*

Para finalizar la aplicación se ha creado una página de ***Información*** sobre la aplicación en la que se explica con que herramientas se ha implementado la aplicación (R y RStudio), que librerías más importantes se han utilizado (shiny, shinydashboard, ggplot2, plotly, tidyverse...), también se ha incluido una breve descripción de los datos añadiendo un enlace al repositorio de Github, (ver Figura C.36).

Capítulo 8

Conclusiones

En esta última sección se exponen las conclusiones derivadas de la realización del proyecto y se presentan las líneas de trabajo a futuro.

Los objetivos planteados en el apartado 1.1 se han alcanzado totalmente ya que se ha logrado crear *TennisBI* [1], una aplicación de visualización de datos de competiciones de tenis con una interfaz moderna y amigable cumpliendo con los requisitos propuestos. Esta aplicación utiliza múltiples gráficos sin excesiva complejidad para el usuario que ayuda a una interpretación rápida de las visualizaciones. Además se muestra la misma información tanto para tenis masculino como para tenis femenino.

Se ha conseguido realizar un proceso automático de ETL, desde la extracción de datos de un repositorio de GitHub mediante R a todas las tareas de transformación y limpieza de los datos.

También se ha definido una medida de similaridad para encontrar tenistas similares teniendo en cuenta variables de los partidos jugados previamente.

8.1. Dificultades encontradas

En el transcurso del proyecto han aparecido ciertos obstáculos que se han debido solventar, algunos de ellos han estado relacionado con la calidad de los datos. Pese que los datos originales están bastante tratados y limpios se han encontrado varias variables con una gran cantidad de outliers como por ejemplo la variable de duración de partidos.

Otra dificultad ha sido la herramienta utilizada para la creación de la aplicación, Shiny la cuál es muy completa y da mucha libertad al desarrollador. Esto hace que se necesite un periodo de adaptación para aprender a utilizarla. Especialmente para poder representar mucha información de manera que no este sobrecargado. Esto ha sido posible gracias a los paneles dentro de cada página.

8.2. Conocimientos aplicados y aprendizajes obtenidos

El desarrollo de este Trabajo de Fin de Grado ha supuesto una puesta en práctica mediante un caso práctico de una gran cantidad de contenidos vistos en diferentes asignaturas, en particular:

- Fundamentos de Programación: Básico para cualquier proyecto en el que haya que programar.
- Estadística Descriptiva: Para el correcto uso de los gráficos utilizados
- Computación Estadística: Importante para conocer el funcionamiento de R, utilizado en todo el proyecto.
- Análisis de datos y análisis multivariante: Para tratar los datos de una manera correcta y conocer algoritmos como el análisis de componentes principales.

[1] `http://shiny1.eio.uva.es:3838/albertogonza/app/`

- También ciertas asignaturas del Grado en Ingeniería Informática que también estoy cursando como pueden ser Interacción Persona Computadora o Diseño y Evaluación de Sistemas Interactivos, para la gestión de sistemas con interacción con usuarios y crear aplicaciones con usabilidad.

Las competencias adquiridas o reforzadas conseguidas a lo largo del Trabajo de Fin de Grado:

- Se ha aprendido a desarrollar un proyecto de visualización de datos de principio a fin, comenzando con la obtención de los datos hasta finalizar creando la aplicación de visualización.
- Se ha aumentado los conocimientos de programación en R especialmente de realizar gráficos con la librería *ggplot2*.
- Se ha aprendido a crear aplicaciones web usables y amigables, en concreto con la librería Shiny de R.
- Se ha aumentado los conocimientos de SPARQL para realizar consultas a la Wikidata y poder obtener las imágenes de los tenistas.

8.3. Trabajo Futuro

El proyecto expuesto en este documento cuenta con diversas ampliaciones futuras, sobre todo en el ámbito de creación de nuevas visualizaciones y crear nuevos modelos estadísticos ya que la información generada por el tenis es inmensa, sólo hace falta analizarla profundamente. Las principales líneas a seguir son:

- En la pestaña de enfrentamiento crear un gráfico que muestre la probabilidad de victoria de cada tenista en la página de ***Enfrentamiento***. Para ello habría que crear un modelo estadístico que tuviera en cuenta variables como ranking actual, superficie del torneo, estilo de tenis de cada tenista...
- Acceso a información de tenis en tiempo real, esto permitiría observar información de partidos en directo y además juntarlo con predicciones a tiempo real en el que el marcador influye en la probabilidad de victoria.
- Crear una página de records que muestre diferente tipos de record. Tenista con más victorias, más títulos, racha de más partidos ganando, edad más joven en llegar al número 1...
- Crear una página de superficie comparando las superficies entre sí teniendo en cuenta muchas variables como número de saques directos, duración de los partidos...
- Ampliar la información de la aplicación añadiendo nuevas fuentes con más riqueza de datos.
- Crear una métrica para evaluar tenistas a partir de los datos teniendo en cuenta su nivel, sus lesiones, su concentración en momentos importantes como finales o bolas de break.
- Probar nuevas similaridades para comprobar si se obtienen resultados parecidos a los obtenidos con la similaridad coseno.

Anexos

Anexo A

Imágenes TFG

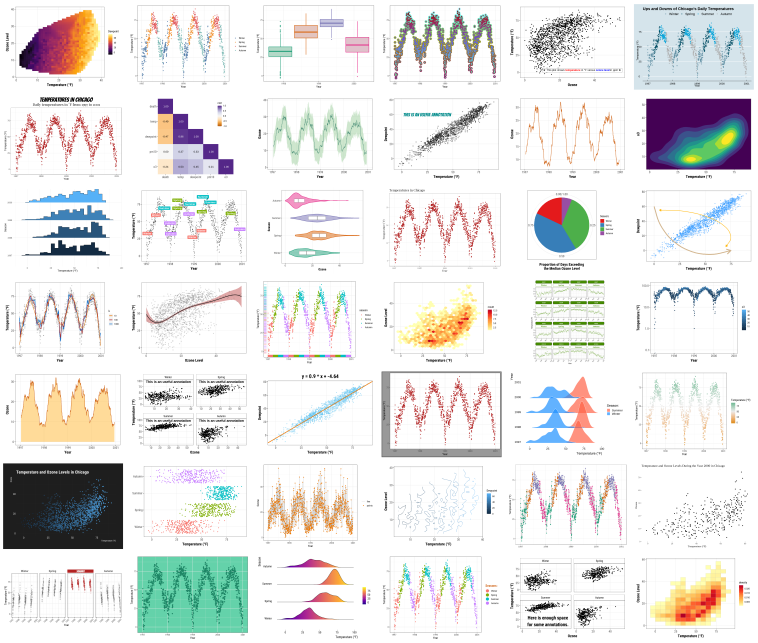

Figura A.1: Tipos de gráficos en *ggplot2*

Anexo B

Contenido digital

El repositorio se encuentra en la cuenta personal del alumno en el GitLab de la Escuela de Ingeniería Informática [1]. En el repositorio, se puede encontrar:

- app.R: Archivo R que lanza el programa.
- src: Directorio con el código fuente de la aplicación.
  - functions.R: Archivo R con las funciones utilizadas a lo largo de la aplicación.
  - libraries.R: Archivo R para cargar las librerias utilizadas.
  - plots.R: Archivo R con el código para crear los gráficos utilizados.
  - server.R: Archivo R con el código del servidor de la aplicación.
  - ui.r: Archivo R con el código de la interfaz de usuario de la aplicación.
  - updateData.R: Archivo R con el código para extraer y actualizar los datos.

  Los datos no se pueden adjuntar al repositorio por restricciones de almacenamiento.

[1] `https://gitlab.inf.uva.es/albegon/tfg_est_alberto_gonzalez`

Anexo C

Imágenes de la aplicación *TennisBI*

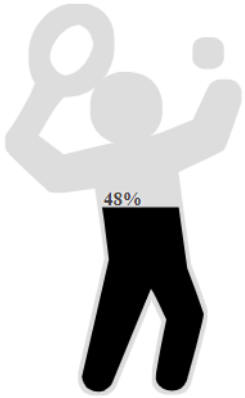

Figura C.1: Pantalla de Carga con el logo de *TennisBI*

Figura C.2: Página de ***Inicio***

Figura C.3: Imágenes del carrusel de la página de ***Inicio***

Figura C.4: Encabezado de la aplicación

Figura C.5: Menú lateral (izquierda desplegado y en la derecha plegado)

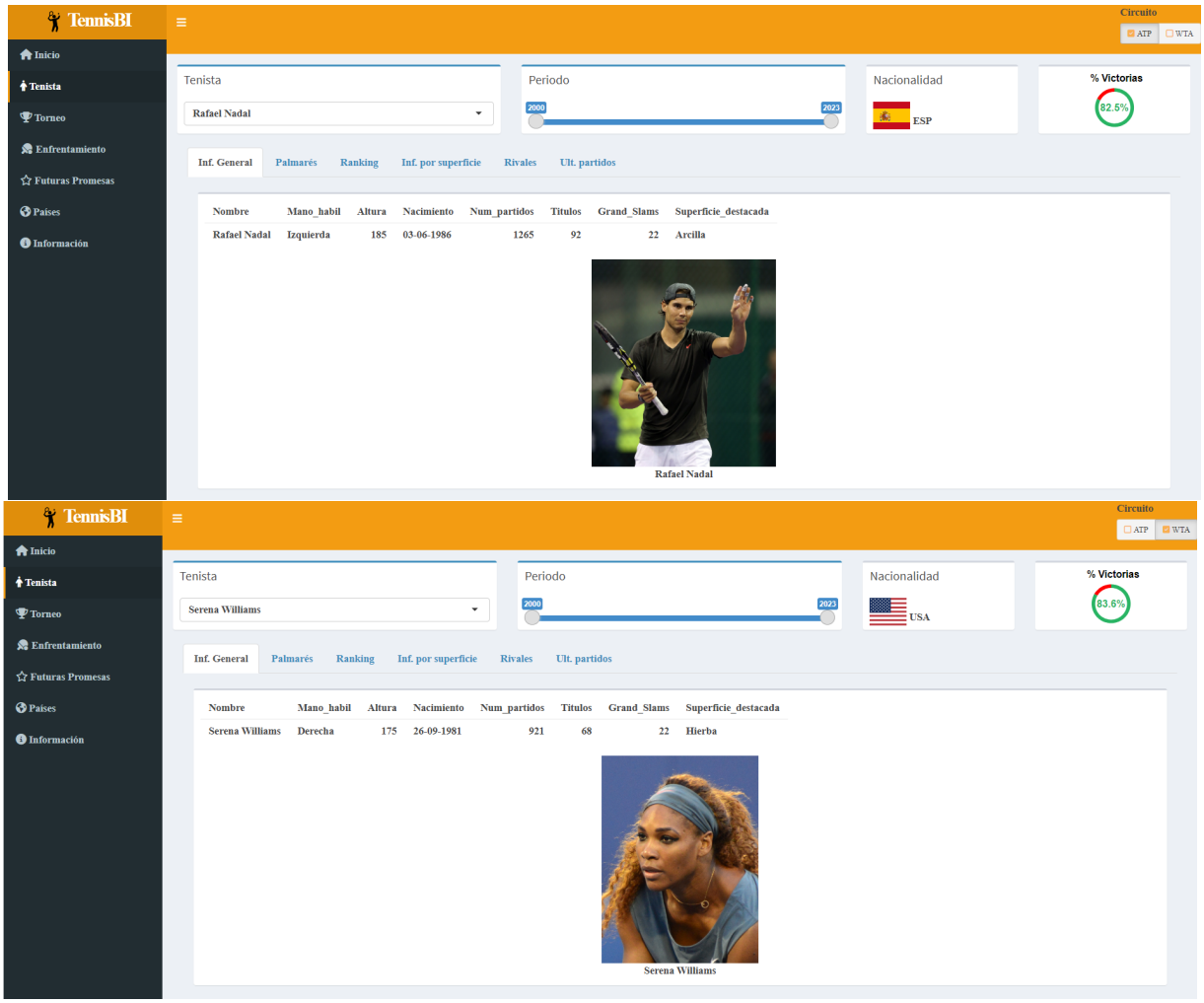

Figura C.6: Apartado Información general de la página ***Tenista***, tanto como para un tenista masculino (imagen superior) o femenino (imagen inferior)

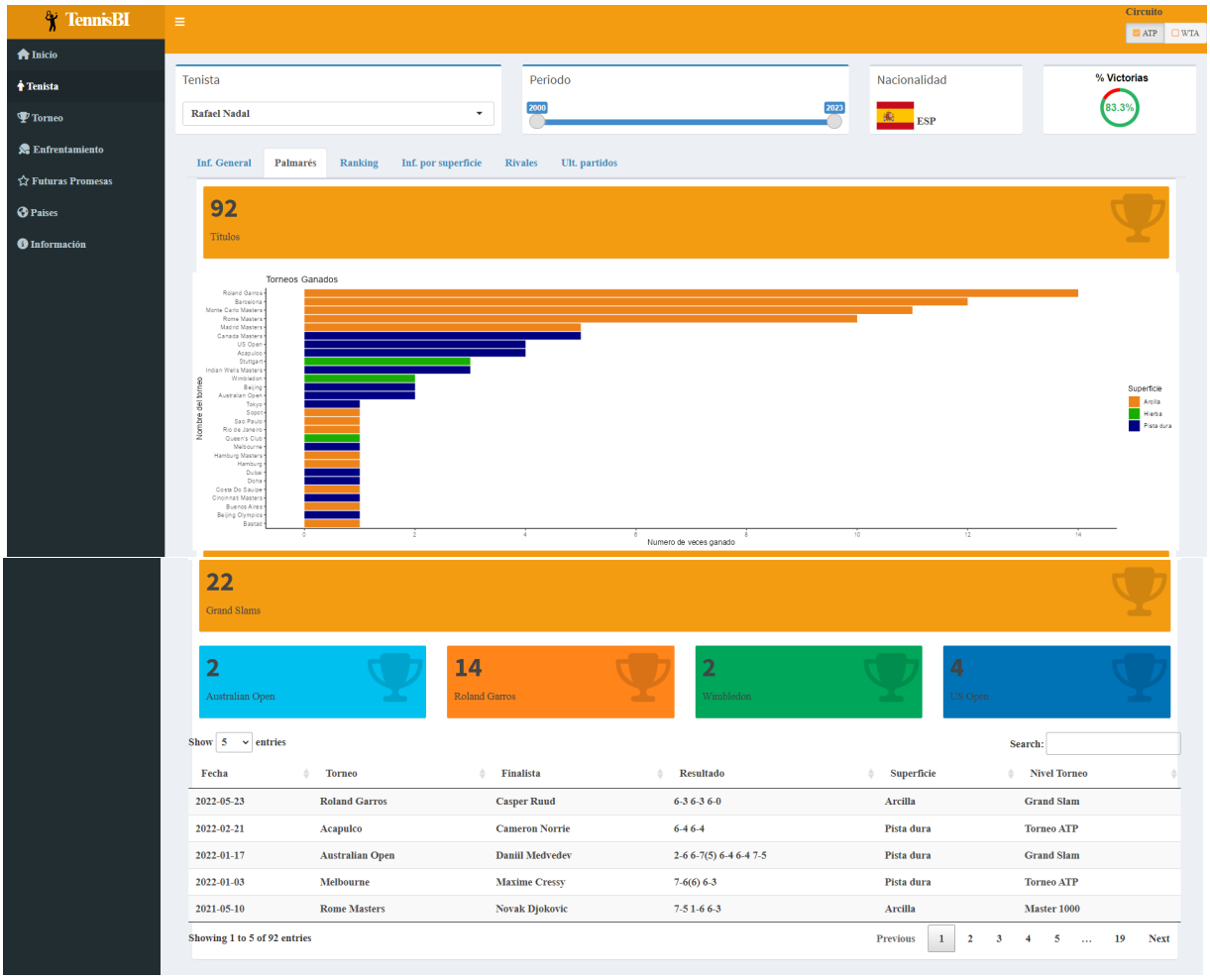

Figura C.7: Apartado palmarés de la página ***Tenista***

Figura C.8: Apartado ranking de la página ***Tenista***

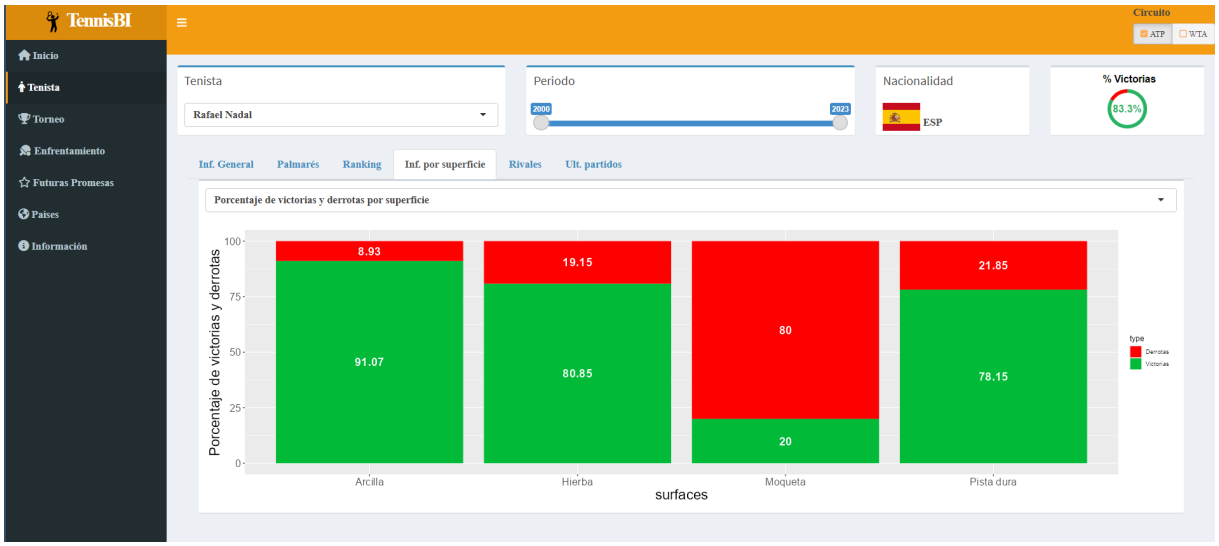

Figura C.9: Apartado Información por superficie de la página ***Tenista***

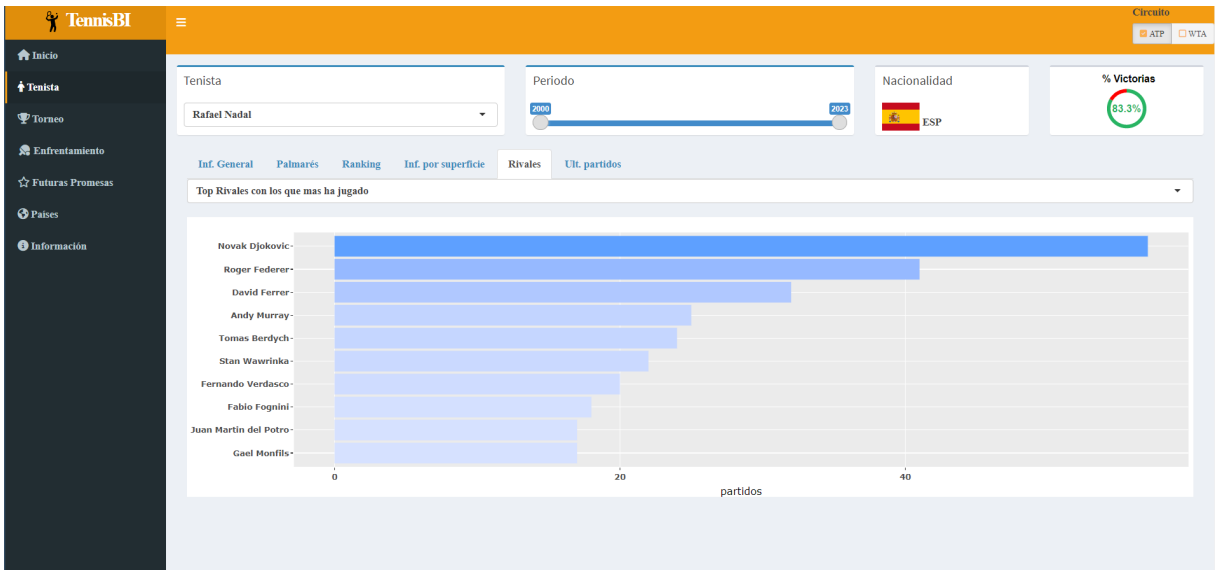

Figura C.10: Apartado Rivales de la página ***Tenista***

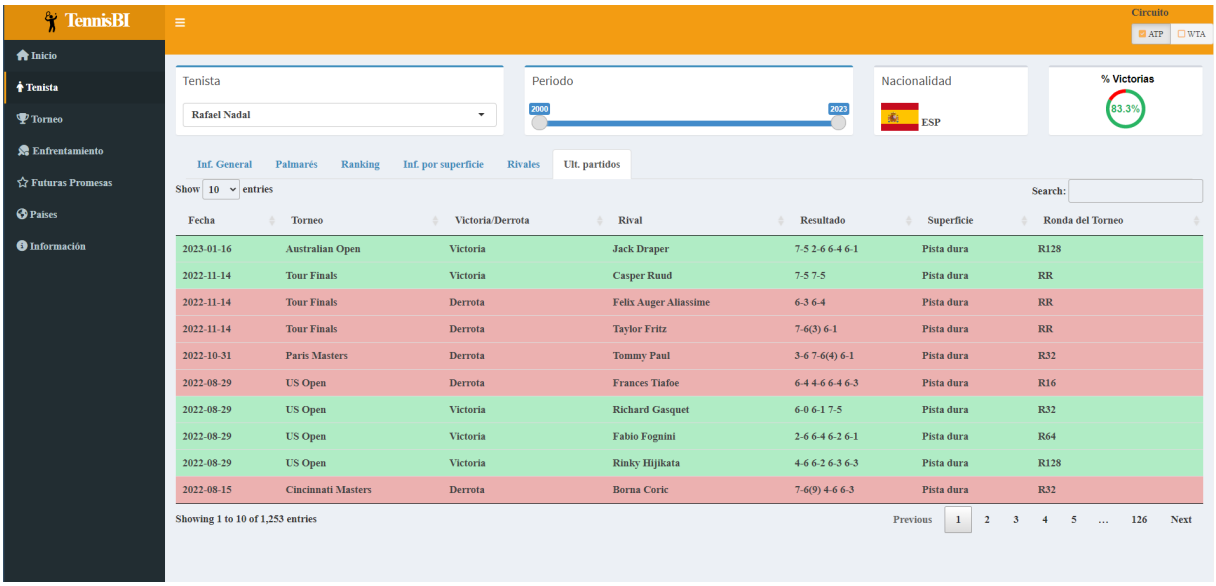

Figura C.11: Apartado Últimos partidos de la página ***Tenista***

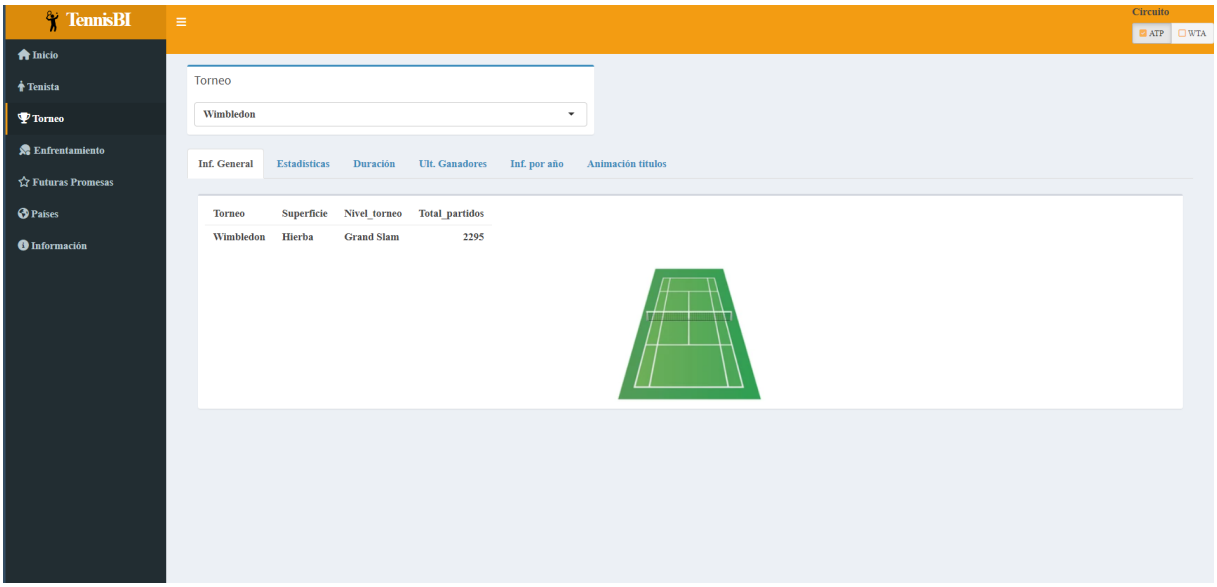

Figura C.12: Apartado Información general de la página ***Torneo***

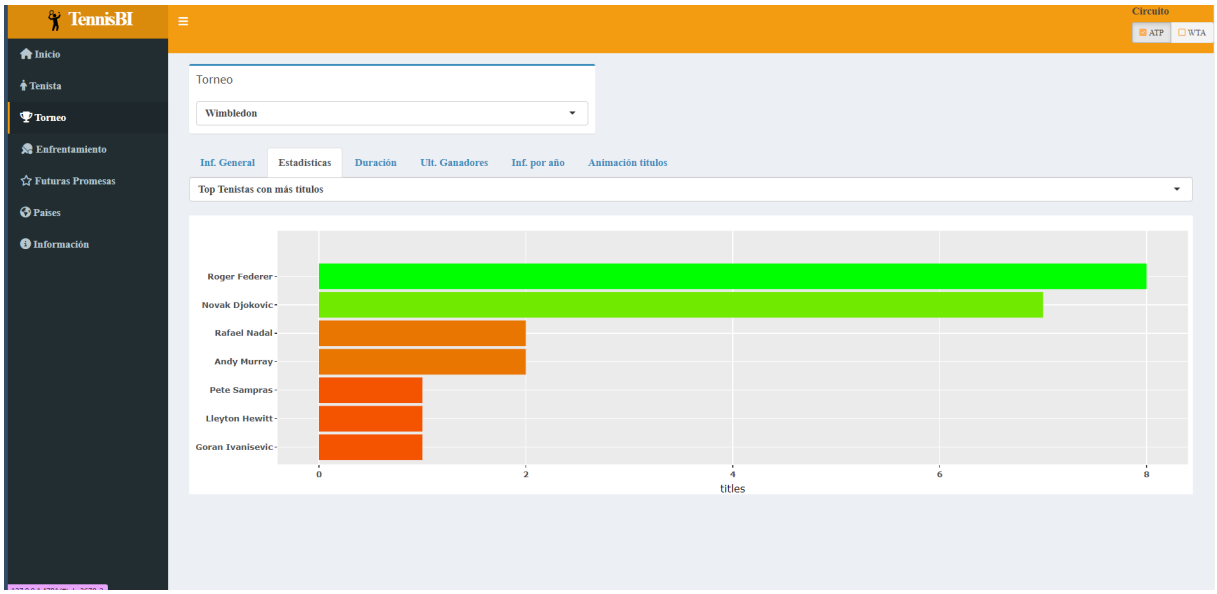

Figura C.13: Apartado Estadísticas de la página ***Torneo***

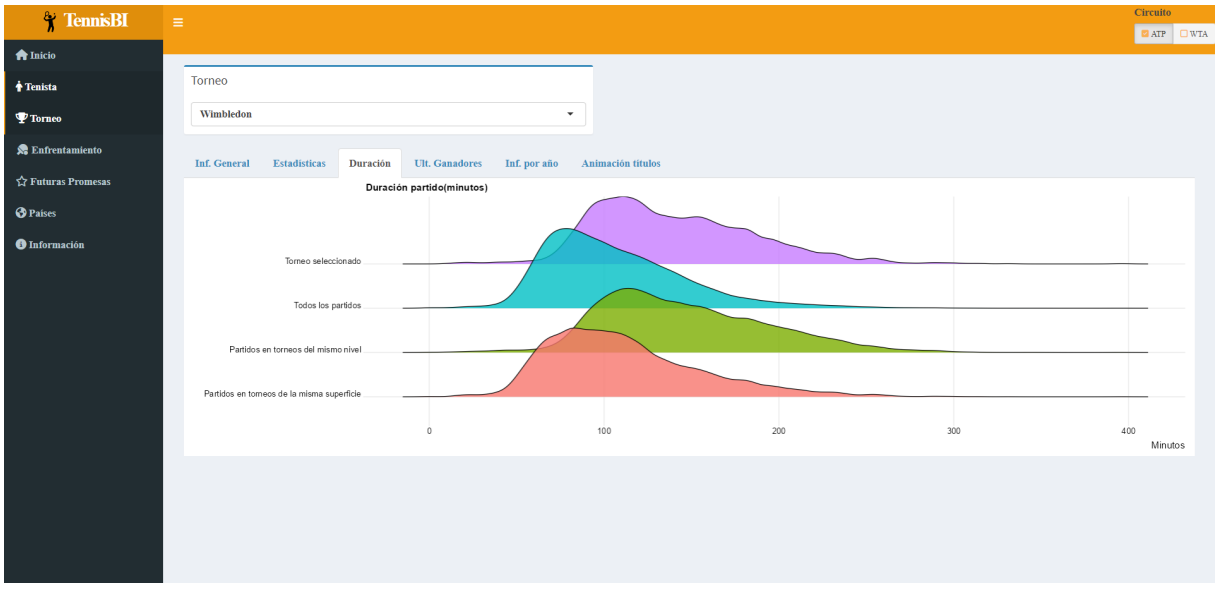

Figura C.14: Apartado Duración de la página ***Torneo***

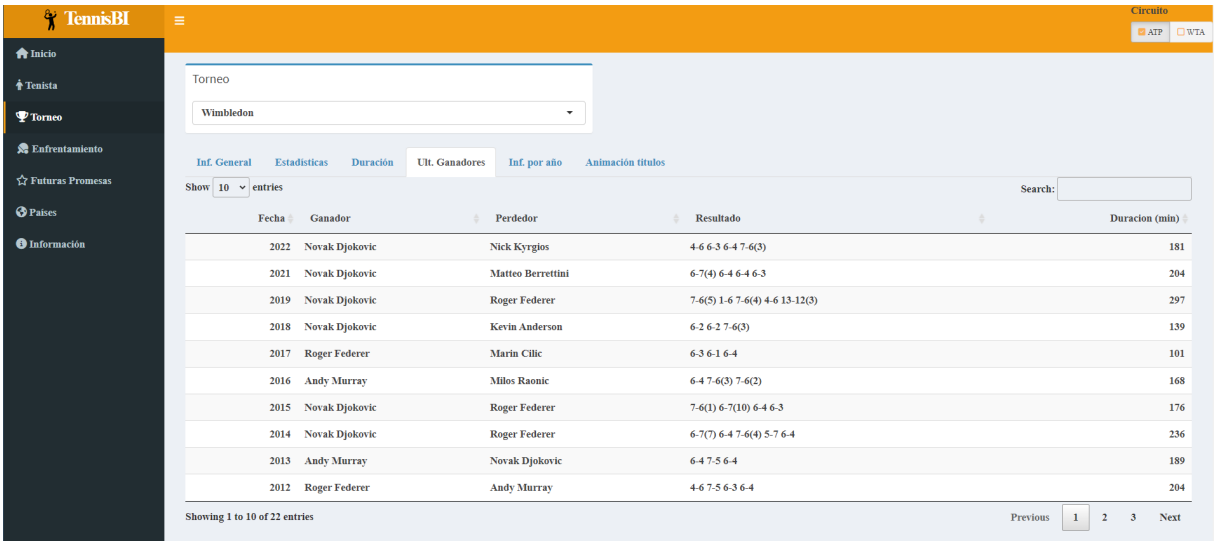

Figura C.15: Apartado Últimos ganadores de la página ***Torneo***

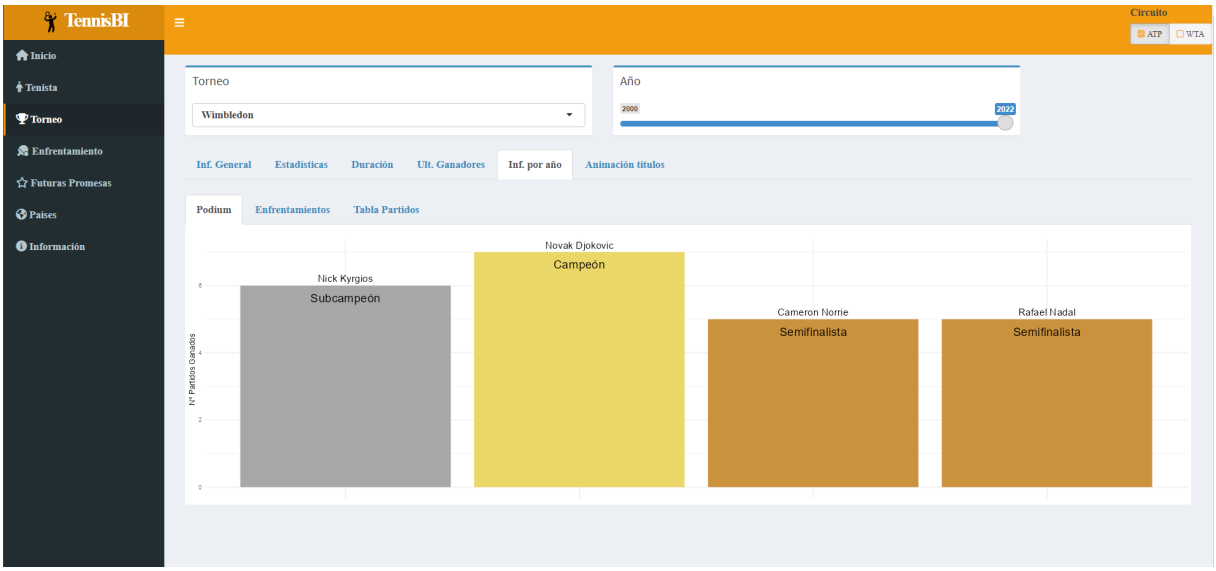

Figura C.16: Subapartado Podio del apartado Información por año de la página ***Torneo***

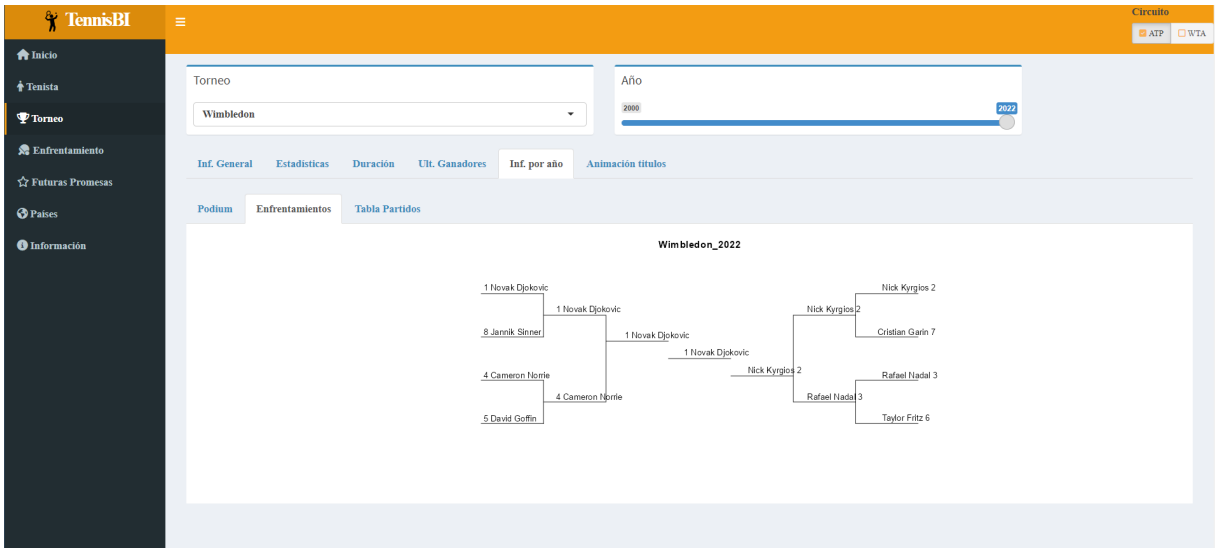

Figura C.17: Subapartado Enfrentamientos del apartado Información por año de la página ***Torneo***

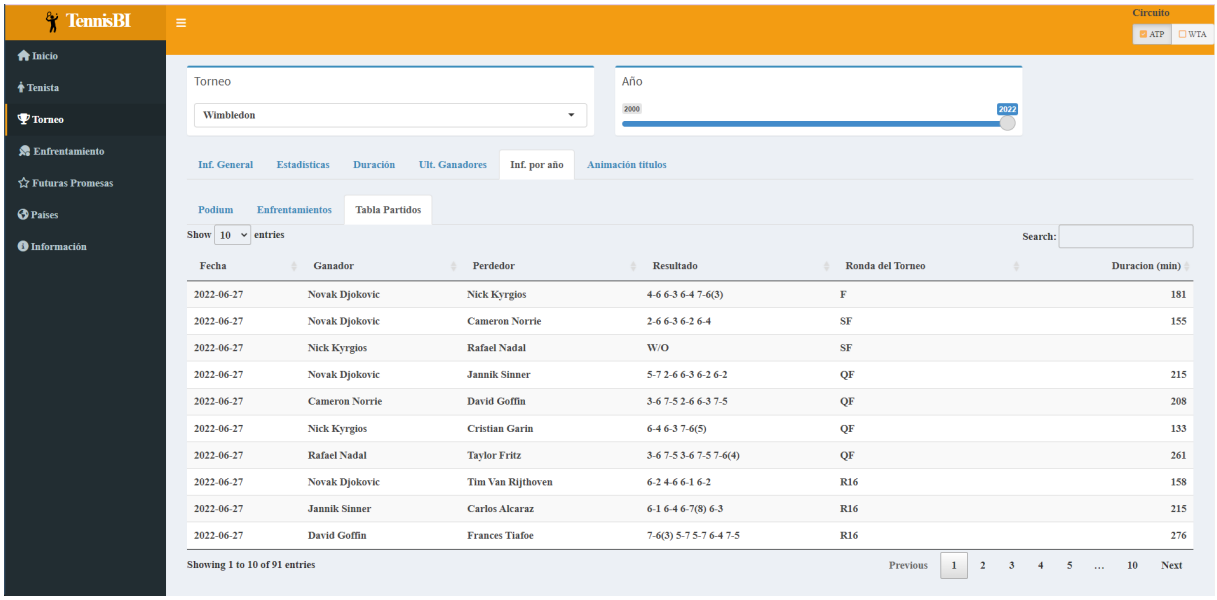

Figura C.18: Subapartado Tabla Partidos del apartado Información por año de la página ***Torneo***

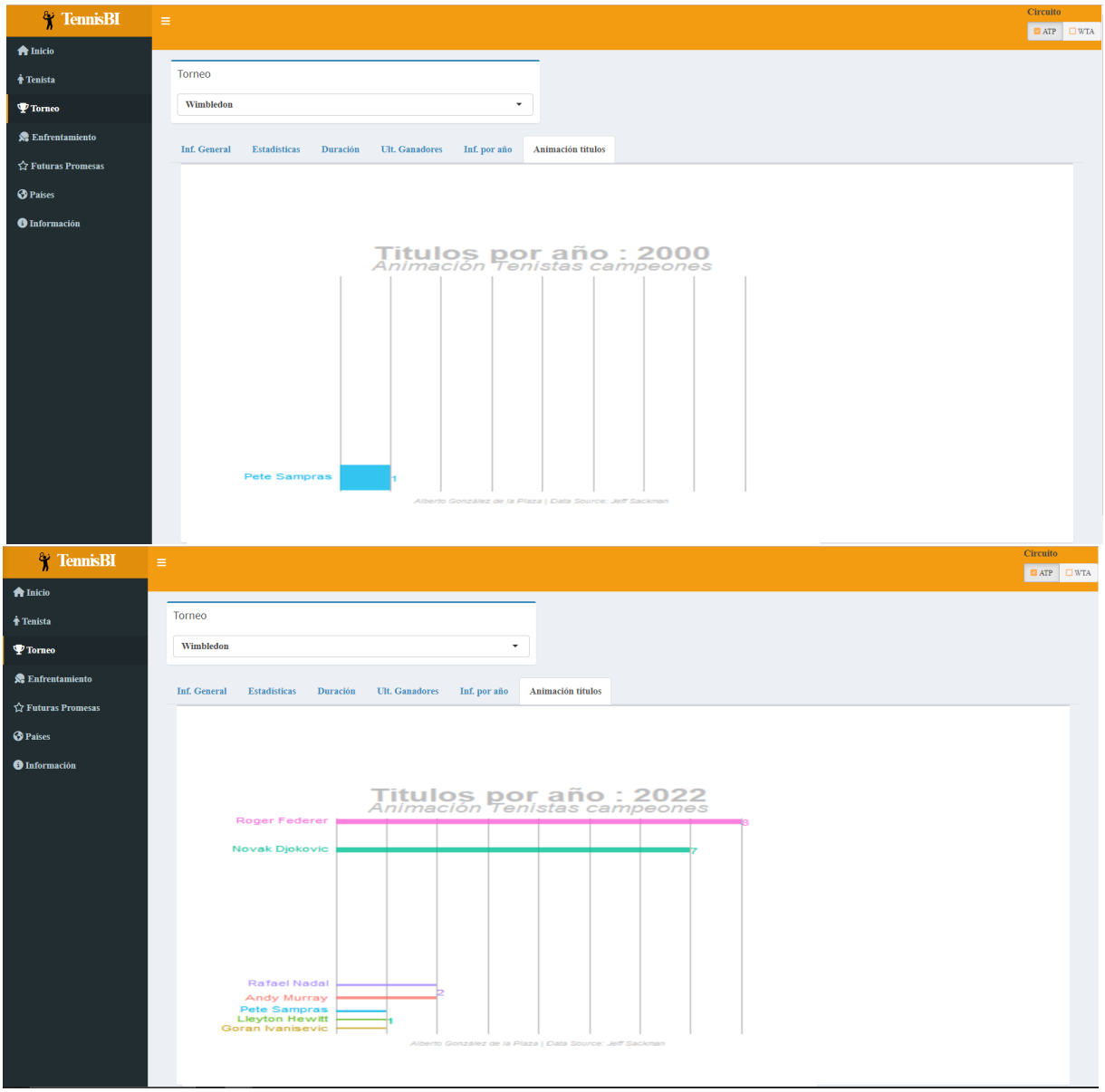

Figura C.19: Apartado Animación por año de la página ***Torneo*** (inicio y final de la animación)

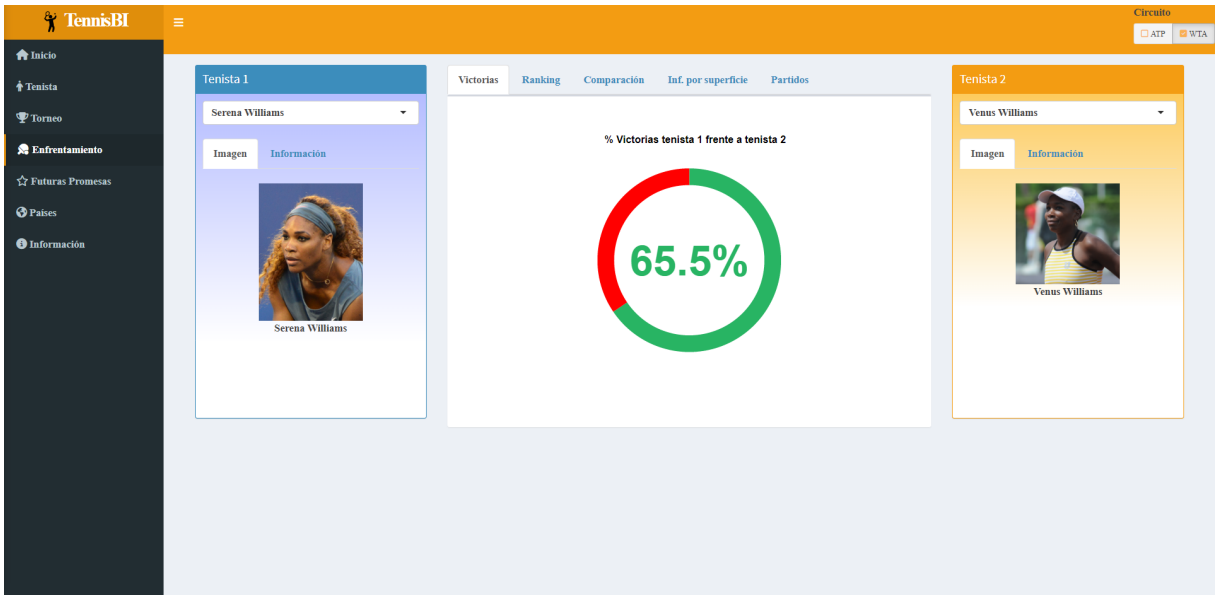

Figura C.20: Apartado Victorias de la página ***Enfrentamiento***, también se muestra la Imagen de ambas tenistas

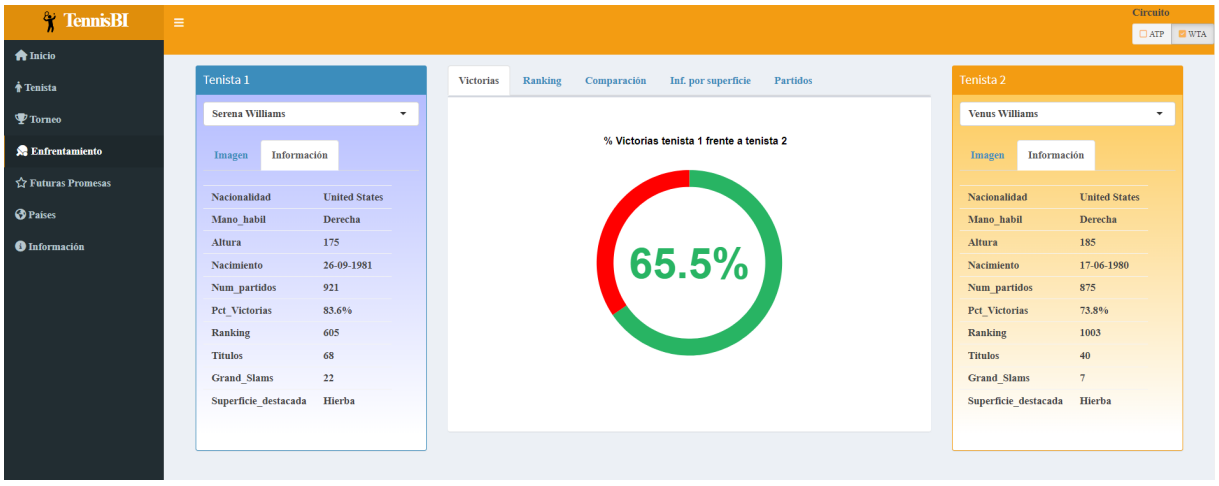

Figura C.21: Apartado Victorias de la página ***Enfrentamiento***, también se muestra la Información de ambas tenistas

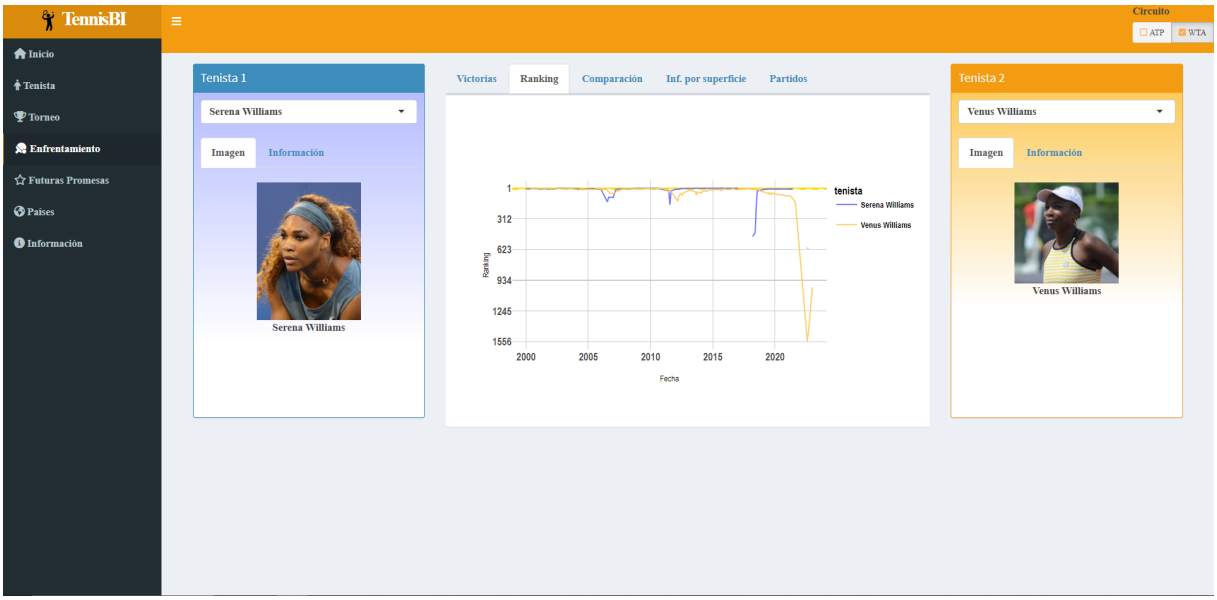

Figura C.22: Apartado Ranking de la página ***Enfrentamiento***

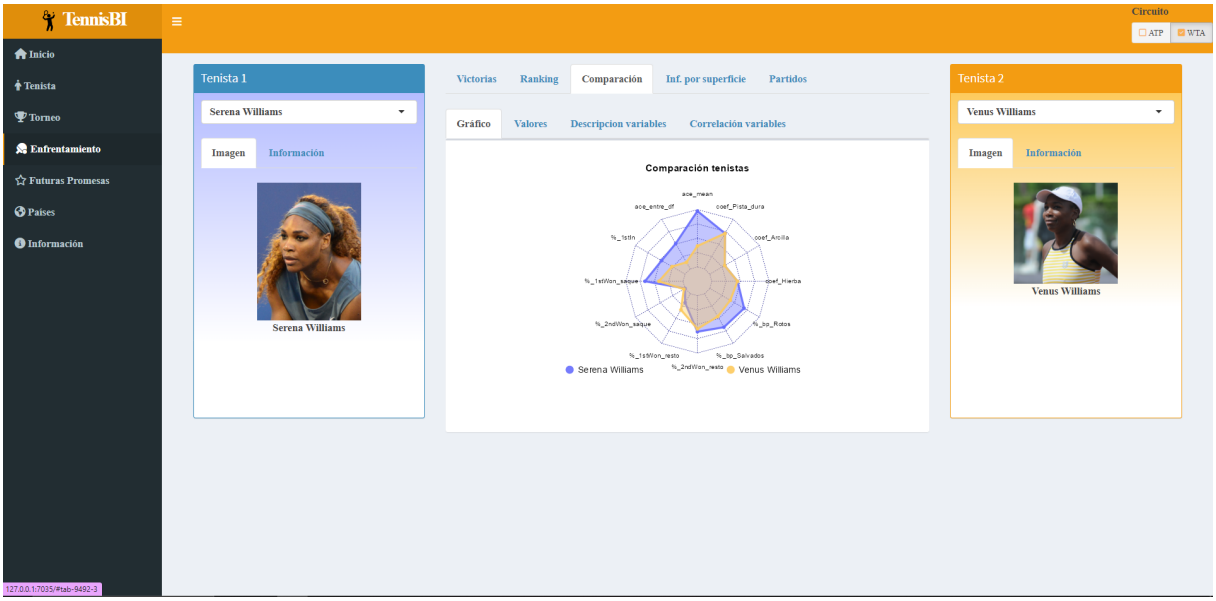

Figura C.23: Subapartado Gráfico del apartado Comparación de la página ***Enfrentamiento***

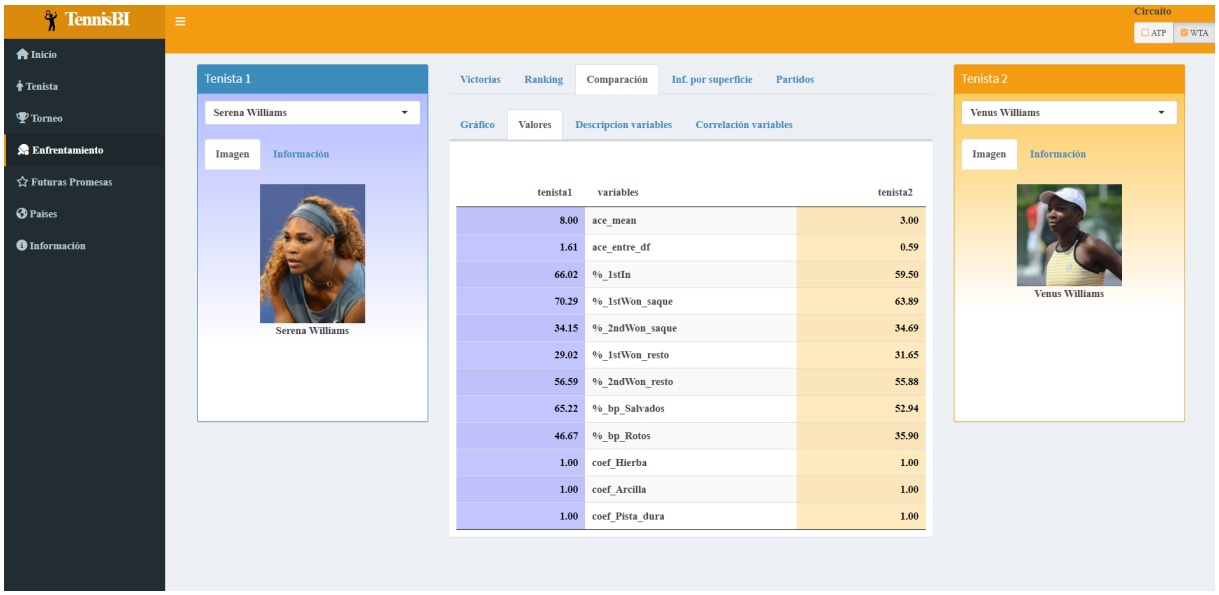

Figura C.24: Subapartado Valores del apartado Comparación de la página ***Enfrentamiento***

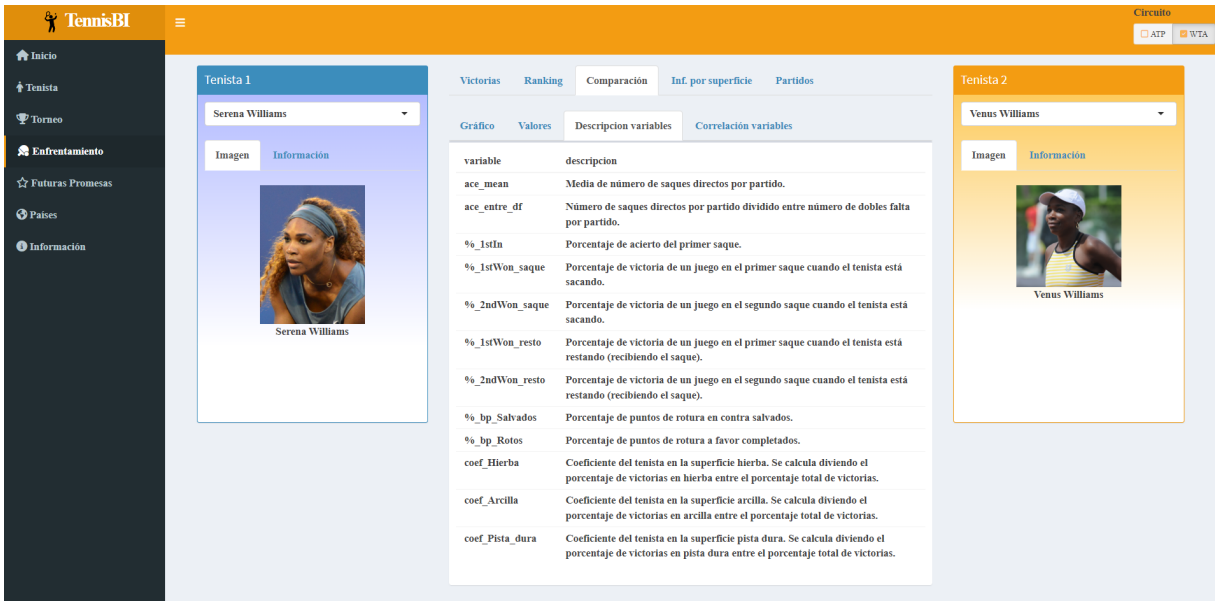

Figura C.25: Subapartado Descripción variables del apartado Comparación de la página ***Enfrentamiento***

Figura C.26: Subapartado Correlación variables del apartado Comparación de la página ***Enfrentamiento***

Figura C.27: Apartado Información por superficie de la página ***Enfrentamiento***

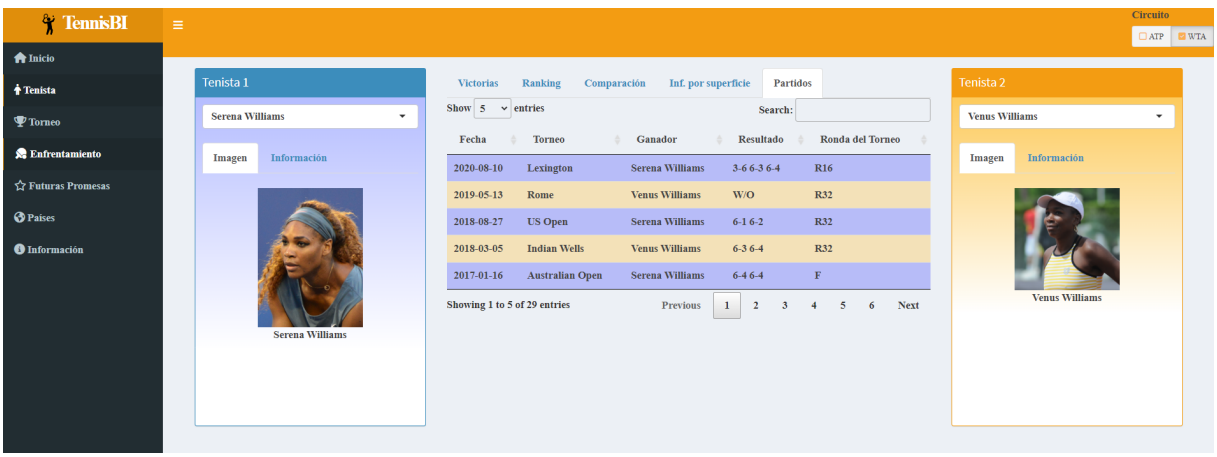

Figura C.28: Apartado Partidos de la página ***Enfrentamiento***

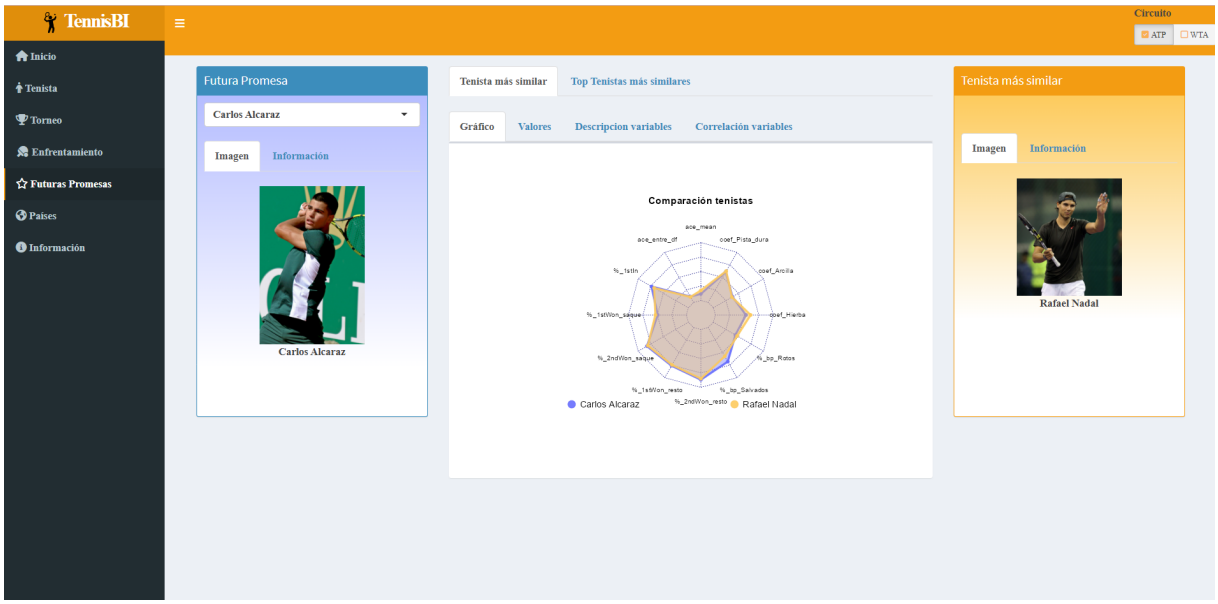

Figura C.29: Subapartado Gráfico del apartado Tenista más similar de la página ***Futuras promesas***

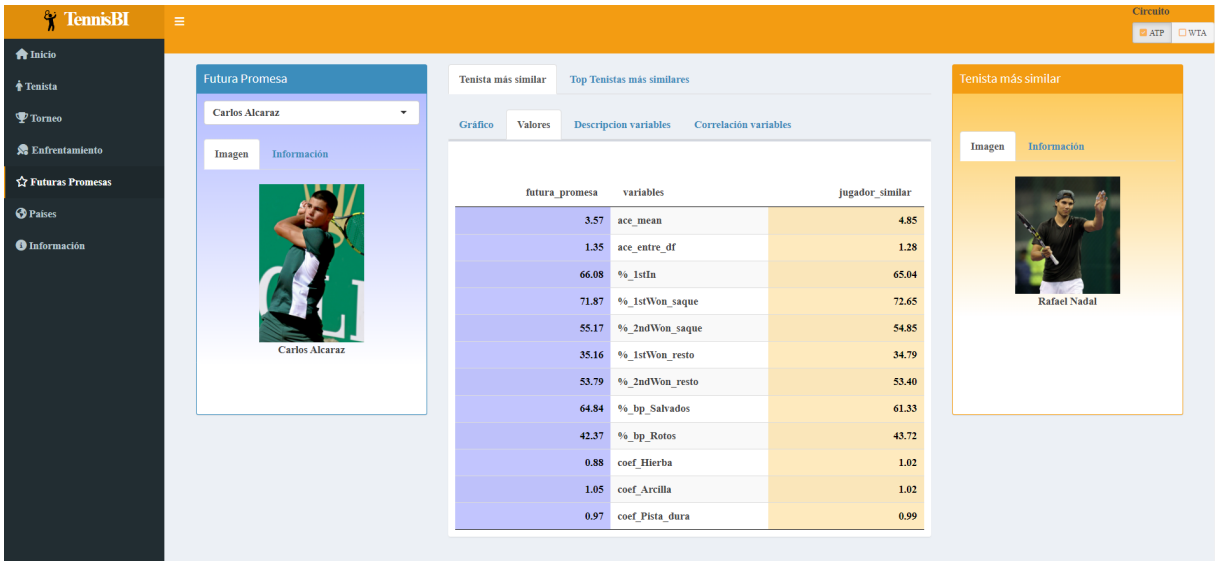

Figura C.30: Subapartado Valores del apartado Tenista más similar de la página ***Futuras promesas***

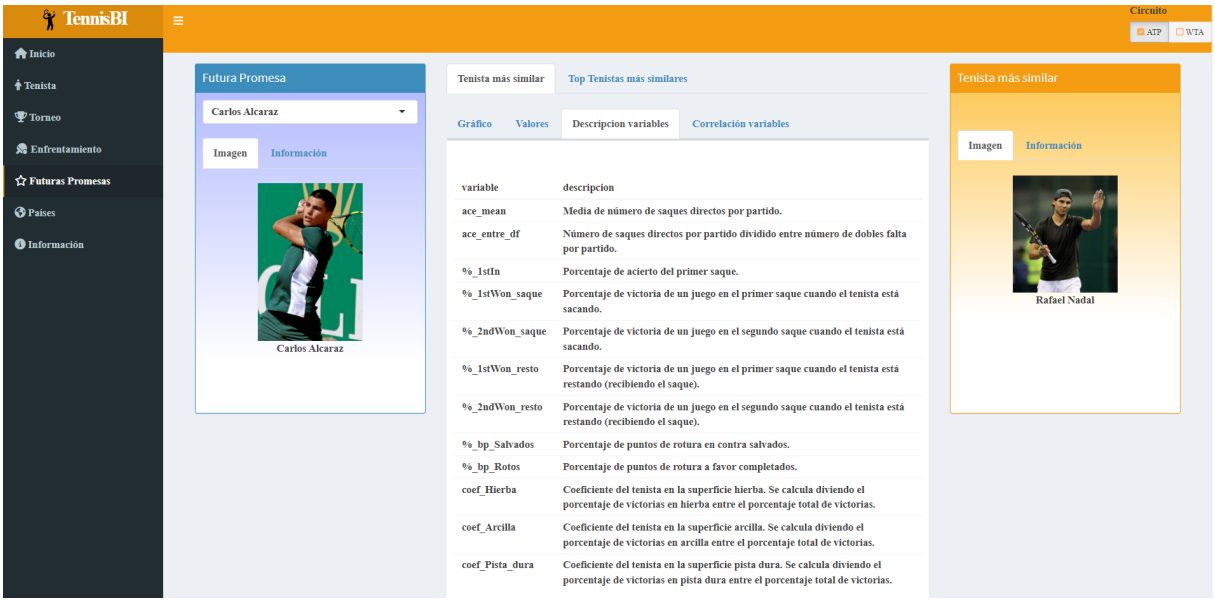

Figura C.31: Subapartado Descripción variables del apartado Tenista más similar de la página ***Futuras promesas***

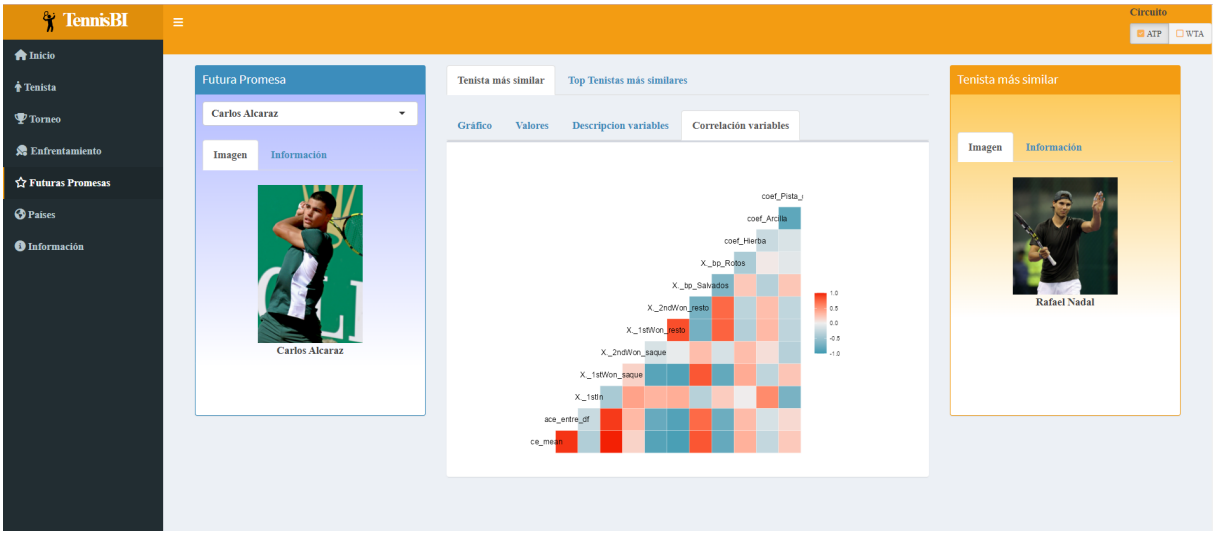

Figura C.32: Subapartado Correlación variables del apartado Tenista más similar de la página ***Futuras promesas***

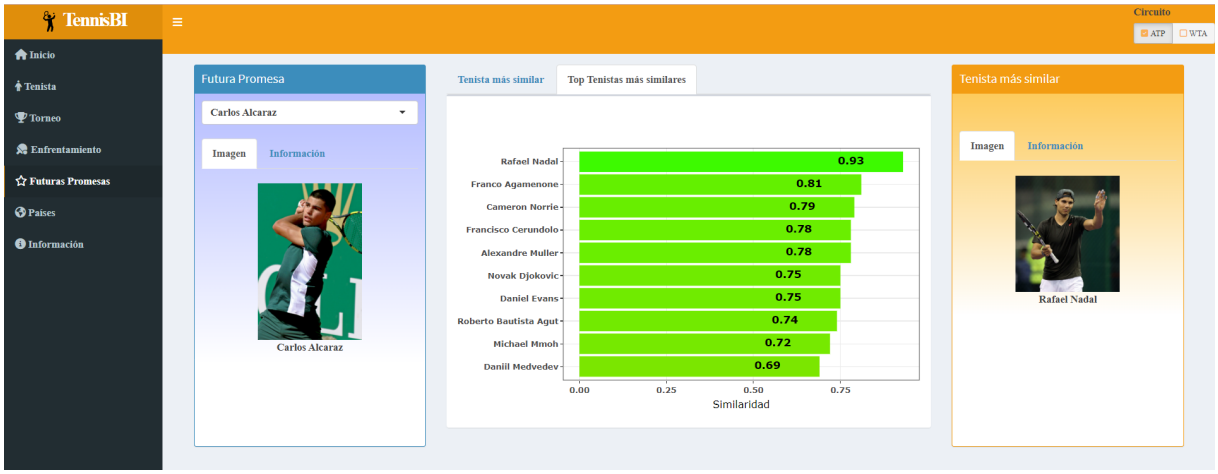

Figura C.33: Apartado Tenistas más similares de la página ***Futuras promesas***

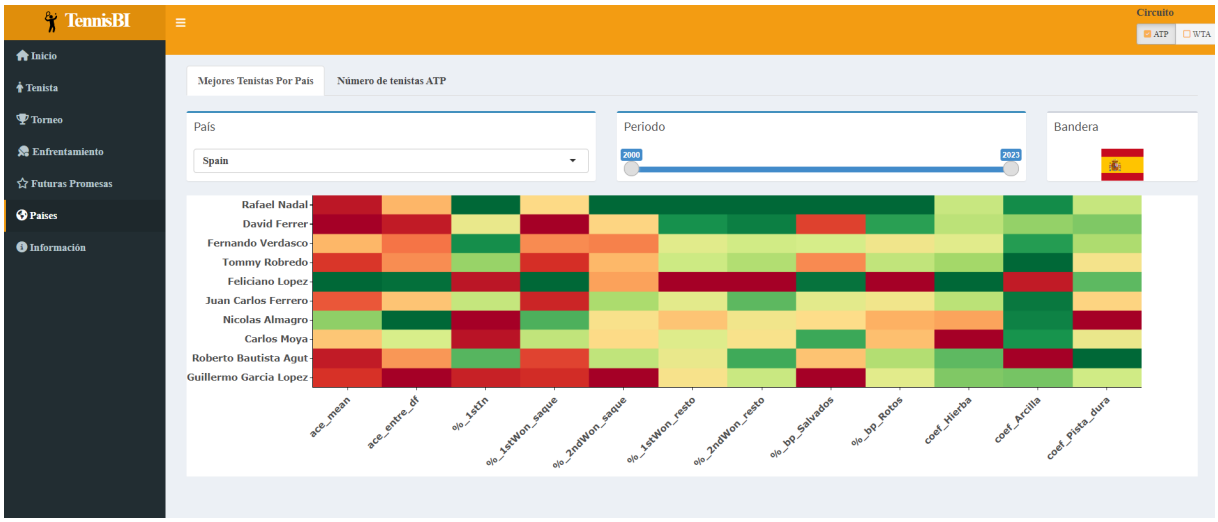

Figura C.34: Apartado Mejores tenistas por país de la página ***Países***

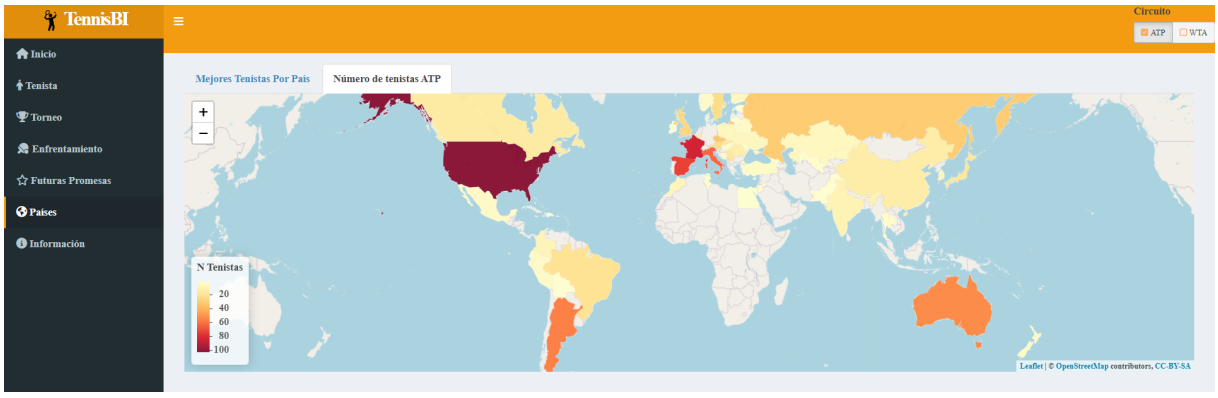

Figura C.35: Apartado Número tenistas de la página ***Países***

Figura C.36: Página ***Información***

Bibliografía

[1] Meollo. *Historia y origen del tenis*. URL: https://meollo.net/es/historia-y-origen-del-tenis/ (Último acceso 26-05-2023).

[2] Wikipedia. *History of tennis*. URL: https://en.wikipedia.org/wiki/History_of_tennis#Origin (Último acceso 26-05-2023).

[3] Superprof Blog. *¿Cuál es la historia y el origen del tenis?* URL: https://www.superprof.es/blog/historia-deporte-tenis/ (Último acceso 26-05-2023).

[4] Morys George Lyndhurst Bruce. "Organization and tournaments". En: *Britannica* (). URL: https://www.britannica.com/sports/tennis/Play-of-the-game (Último acceso 26-05-2023).

[5] Wikipedia. *WTA Tour*. URL: https://en.wikipedia.org/wiki/WTA_Tour (Último acceso 26-05-2023).

[6] Pro Tennis Tips. *Tennis Rules*. URL: http://protennistips.net/tennis-rules/ (Último acceso 26-05-2023).

[7] Rahul Venkat. "Tennis rules, scoring system and all you need to know about the racket sport". En: *Olympics News* (). URL: https://olympics.com/en/news/tennis-rules-regulations-how-to-play-basics (Último acceso 26-05-2023).

[8] Barry Steven Lorge. "Professional and open tennis". En: *Britannica* (). URL: https://www.britannica.com/sports/tennis/Organization-and-tournaments (Último acceso 26-05-2023).

[9] Wikipedia. *Association of Tennis Professionals*. URL: https://en.wikipedia.org/wiki/Association_of_Tennis_Professionals (Último acceso 26-05-2023).

[10] Wikipedia. *Women's Tennis Association*. URL: https://en.wikipedia.org/wiki/Women%27s_Tennis_Association (Último acceso 26-05-2023).

[11] Wikipedia. *Ranking ATP*. URL: https://es.wikipedia.org/wiki/Ranking_ATP (Último acceso 26-05-2023).

[12] Wikipedia. *WTA rankings*. URL: https://en.wikipedia.org/wiki/WTA_rankings (Último acceso 26-05-2023).

[13] Jeff Sackmann. *tennis_atp*. URL: https://github.com/JeffSackmann/tennis_atp (Último acceso 26-03-2023).

[14] Jeff Sackmann. *tennis_wta*. URL: https://github.com/JeffSackmann/tennis_wta (Último acceso 26-03-2023).

[15] International Olympic Committee. *NATIONAL OLYMPIC COMMITTEES*. URL: https://olympics.com/ioc/national-olympic-committees (Último acceso 02-04-2023).

[16] Comunidad Wikimedia. *Wikidata*. URL: https://www.wikidata.org (Último acceso 02-04-2023).

[17] Comunidad Wikimedia. *Análisis de componentes principales*. URL: https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales (Último acceso 25-04-2023).

[18] Ramiro Gómez Nuño. "Nuevo paquete para el análisis de datos de juego y jugadores de fútbol: GASB". Statistics Thesis. Universidad de Valladolid, 2021. URL: https://uvadoc.uva.es/handle/10324/50487 (Último acceso 24-04-2023).

[19] William Knottenbelt, Padmanaba Srinivasan y Raghavan Subramanian. "Thinking the GOAT: Imitating Tennis Styles". En: *MIT Sloan Sports Analytics Conference* (ene. de 2023). URL: http://hdl.handle.net/10044/1/102851 (Último acceso 24-04-2023).

[20] Comunidad Wikimedia. *Similitud coseno*. URL: https://es.wikipedia.org/wiki/Similitud_coseno (Último acceso 24-04-2023).

[21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2022. URL: https://www.R-project.org/.

[22] Fridolin Wild. *lsa: Latent Semantic Analysis*. R package version 0.73.3. 2022. URL: https://CRAN.R-project.org/package=lsa.

[23] R Foundation. *The R Project for Statistical Computing*. URL: https://www.r-project.org/ (Último acceso 20-05-2023).

[24] R Foundation. *What is R?* URL: https://www.r-project.org/about.html (Último acceso 20-05-2023).

[25] R Foundation. *Contributed Packages*. URL: https://cran.r-project.org/web/packages/ (Último acceso 20-05-2023).

[26] Winston Chang et al. *shiny: Web Application Framework for R*. R package version 1.7.4. 2022. URL: https://CRAN.R-project.org/package=shiny.

[27] Kent State University. *STATISTICAL AND QUALITATIVE DATA ANALYSIS SOFTWARE: ABOUT R AND RSTUDIO*. URL: https://libguides.library.kent.edu/statconsulting/r (Último acceso 20-05-2023).

[28] Shiny Posit. *Pane Layout*. URL: https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html (Último acceso 20-05-2023).

[29] Domino Datalab. *What is Shiny (R)?* URL: https://www.dominodatalab.com/data-science-dictionary/shiny-in-r (Último acceso 20-05-2023).

[30] Data Carpentry. *Data visualization with ggplot2*. URL: https://datacarpentry.org/R-ecology-lesson/04-visualization-ggplot2.html (Último acceso 20-05-2023).

[31] Carson Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman y Hall/CRC, 2020. ISBN: 9781138331457. URL: https://plotly-r.com.

[32] Plotly. *Plotly R Open Source Graphing Library*. URL: https://plotly.com/r/ (Último acceso 20-05-2023).

[33] Javier Martín de Benito. "ElecCyL: aplicación Shiny para el estudio de diferentes sistemas de reparto aplicados a las elecciones autonómicas en Castilla y León". Trabajo de Fin de Grado. Universidad de Valladolid, 2022. URL: https://uvadoc.uva.es/handle/10324/57959 (Último acceso 21-05-2023).

[34] Andrea Condado Gómez. "Aplicaciones Shiny para explorar datos de resultados electorales". Trabajo de Fin de Grado. Universidad de Valladolid, 2021. URL: https://uvadoc.uva.es/handle/10324/43809 (Último acceso 21-05-2023).

[35] Peter Solymos. "Plotly R Open Source Graphing Library". En: *R bloggers* (abr. de 2021). URL: https://www.r-bloggers.com/2021/04/the-anatomy-of-a-shiny-application/ (Último acceso 21-05-2023).

[36] Tania Martín Fernández. "Diseño y utilización de una herramienta de visualización de datos a escala municipal". Trabajo de Fin de Grado. Universidad de Valladolid, 2021. URL: https://uvadoc.uva.es/handle/10324/50495 (Último acceso 21-05-2023).

[37] Shiny Posit. *Shiny - Add control widgets*. URL: https://shiny.posit.co/r/getstarted/shiny-basics/lesson3/ (Último acceso 21-05-2023).

[38] Loan Robinson. *Outputs - render() R-Shiny*. URL: https://bookdown.org/loankimrobinson/rshinybook/basic-back-server-output.html (Último acceso 21-05-2023).

[39] Shiny Posit. *Reactivity - An overview*. URL: https://shiny.posit.co/r/articles/build/reactivity-overview/ (Último acceso 21-05-2023).

[40] Shiny Posit. *Application layout guide*. URL: https://shiny.posit.co/r/articles/build/layout-guide/ (Último acceso 21-05-2023).

[41] Weicheng Zhu. *Shiny Tutorial - UI*. URL: http://dreamhunter.me/shinyTutorial/ui.html (Último acceso 21-05-2023).

[42] Winston Chang y Barbara Borges Ribeiro. *shinydashboard: Create Dashboards with 'Shiny'*. R package version 0.7.2. 2021. URL: https://CRAN.R-project.org/package=shinydashboard.

[43] shinydashboard. *Background: Shiny and HTML*. URL: https://rstudio.github.io/shinydashboard/structure.html (Último acceso 21-05-2023).

[44] Pierre Formont. *shinyflags: Insert Country Flags In Shiny Applications*. R package version 0.1.0. 2023.

[45] Agencia tributaria. *Código países*. URL: https://www.agenciatributaria.es/static_files/Sede/Procedimiento_ayuda/GC07/Codigo_paises.pdf (Último acceso 25-04-2023).

[46] Vincent Arel-Bundock, Nils Enevoldsen y CJ Yetman. "countrycode: An R package to convert country names and country codes". En: *Journal of Open Source Software* 3.28 (2018), pág. 848. URL: https://doi.org/10.21105/joss.00848.

[47] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

[48] David Gohel y Panagiotis Skintzos. *ggiraph: Make 'ggplot2' Graphics Interactive*. R package version 0.8.7. 2023. URL: https://CRAN.R-project.org/package=ggiraph.

[49] Mikhail Popov. *WikidataQueryServiceR: API Client Library for 'Wikidata Query Service'*. R package version 1.0.0. 2020. URL: https://CRAN.R-project.org/package=WikidataQueryServiceR.

[50] Yihui Xie, Joe Cheng y Xianying Tan. *DT: A Wrapper of the JavaScript Library 'DataTables'*. R package version 0.23. 2022. URL: https://CRAN.R-project.org/package=DT.

[51] Gregory A. Pilgrim. *SwimmeR: Package for working with Swimming Data*. 2019. URL: https://github.com/gpilgrim2670/SwimmeR.

[52] NCAA. *NCAA bracket for March Madness*. URL: https://www.ncaa.com/march-madness-live/bracket (Último acceso 10-05-2023).

[53] Thomas Lin Pedersen y David Robinson. *gganimate: A Grammar of Animated Graphics*. R package version 1.0.8. 2022. URL: https://CRAN.R-project.org/package=gganimate.

[54] Minato Nakazawa. *fmsb: Functions for Medical Statistics Book with some Demographic Data*. R package version 0.7.5. 2023. URL: https://CRAN.R-project.org/package=fmsb.

[55] Barret Schloerke et al. *GGally: Extension to 'ggplot2'*. R package version 2.1.2. 2021. URL: https://CRAN.R-project.org/package=GGally.

[56] Galili et al. "heatmaply: an R package for creating interactive cluster heatmaps for online publishing". En: *Bioinformatics* (2017). DOI: 10.1093/bioinformatics/btx657. eprint: https://academic.oup.com/bioinformatics/article-pdf/doi/10.1093/bioinformatics/btx657/21358327/btx657.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btx657.

[57] Joe Cheng, Bhaskar Karambelkar y Yihui Xie. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*. R package version 2.1.2. 2023. URL: https://CRAN.R-project.org/package=leaflet.

[58] thematicmapping.org. *Thematic Mapping API*. URL: https://thematicmapping.org/downloads/ (Último acceso 13-05-2023).