

Trabajar con StatsBomb Data en R



¿Qué es R y por qué usarlo?

¿Qué es R y por qué usarlo?

R es un lenguaje de programación especialmente útil para el manejo de conjuntos grandes de datos. En el ámbito que nos ocupa (estadística avanzada en fútbol) nos permite tratar los conjuntos de datos para diferentes fines tales como la creación de métricas así como visualizaciones de los mismos.

R se puede descargar en este enlace:

<https://cran.r-project.org/mirrors.html>

En StatsBomb trabajamos regularmente con R (entre otros lenguajes de programación) en nuestro día a día, particularmente en el departamento de análisis. Empezar a trabajar con hojas de cálculo puede ser una posibilidad válida al comienzo, pero a medida que los conjuntos de datos crecen se vuelven más difíciles de manejar haciendo casi imposible realizar una disección detallada de los mismos sin un lenguaje de programación.

Una vez superada la curva de aprendizaje, R es ideal para trabajar y analizar los datos de manera eficiente y sencilla.



RStudio

La versión básica de R es un tanto engorrosa. Esto ha llevado a la creación de varios entornos de desarrollo integrados (IDEs). Estos “wrappers” son softwares desarrollados a partir de la versión inicial y tratan de hacer la mayoría de tareas dentro de R más sencillas y manejables para el usuario. El más popular de estos es **RStudio**:

<https://www.rstudio.com/products/rstudio/>

Es recomendable instalar **RStudio** u otro IDE similar para que el proceso de trabajo con los datos de StatsBomb más simple y limpio.

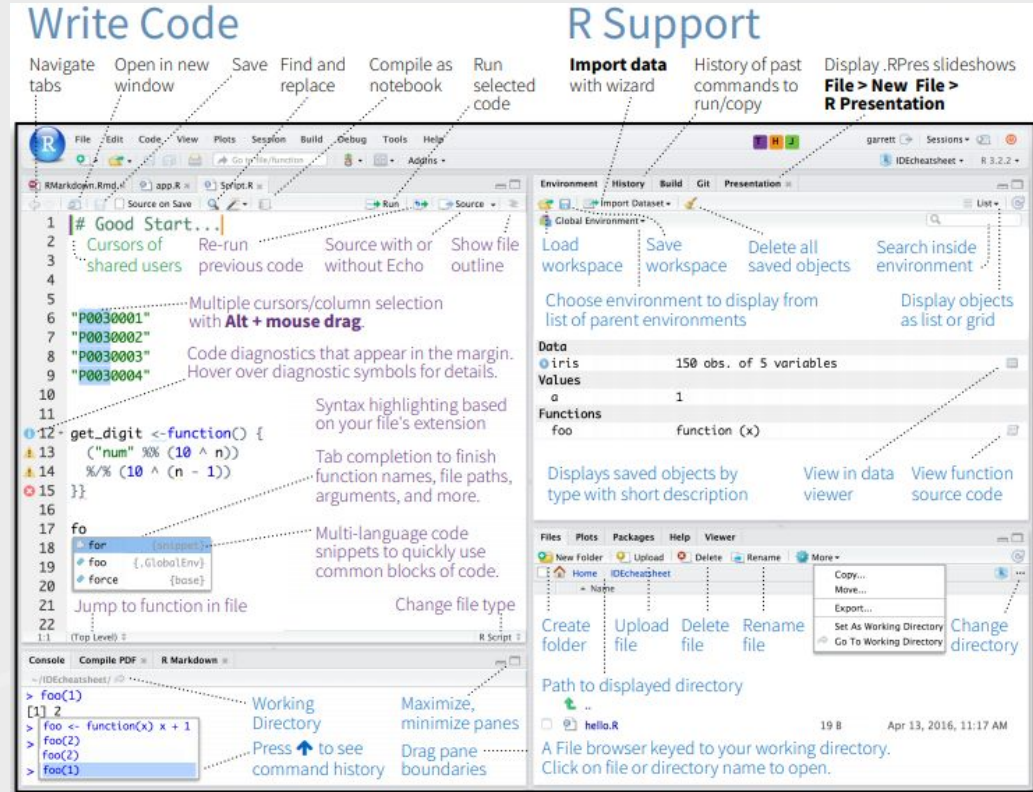


Abrir un Proyecto Nuevo en R

Esto es lo que verá el usuario al cargar por primera vez **RStudio**.

En caso de no tener clara la función de cada opción o sección de **RStudio** es recomendable echar un vistazo a alguna de las hojas de consejos y tutoriales relativos a R en <https://www.rstudio.com/resources/cheatsheets/>.

Es sencillo encontrar una gran cantidad de recursos con explicaciones y respuestas detalladas a cualquier pregunta que pueda surgir respecto a R.



Paquetes & ‘StatsBombR’

¿Qué es un Paquete de R?

Los paquetes son conjuntos de código y funciones organizadas que se pueden descargar fácilmente. Se utilizan para simplificar la realización de algunas tareas. Para instalar un paquete en R simplemente hay que ejecutar *install.packages('NombreDelPaquete')*.

Los paquetes que utilizaremos y que será necesario tener instalados son los siguientes:

‘devtools’: La mayoría de paquetes se encuentran en [CRAN](https://cran.r-project.org/). Sin embargo, también se pueden encontrar muchos paquete útiles en Github. Devtools permite descargar los paquetes directamente desde Github.

‘ggplot2’: El paquete más popular para llevar a cabo la visualización de datos en R.

‘StatsBombR’: El paquete propio de StatsBomb para analizar nuestros datos.

‘StatsBombR’ tiene *‘dependencies’*. Estos son otros paquetes de los que el paquete original depende para funcionar correctamente. Al instalar StatsBombR se instalarán automáticamente sus *‘dependencies’* para que eso no suponga una tarea adicional. Una vez esté instalado, el paquete se puede cargar ejecutando *library(NombreDelPaquete)*.



¿Qué es 'StatsBombR'?

Derrick Yam (ex-Data Scientist de StatsBomb) desarrolló StatsBombR, un paquete dedicado a hacer uso de los datos originales de StatsBomb de manera más sencilla e intuitiva. Se puede descargar en este enlace de Github donde se incluye además información sobre su uso:

<https://github.com/statsbomb/StatsBombR>

Para instalar el paquete en R es necesario instalar primero el paquete 'devtools' ejecutando la siguiente línea:

```
install.packages("devtools")
```

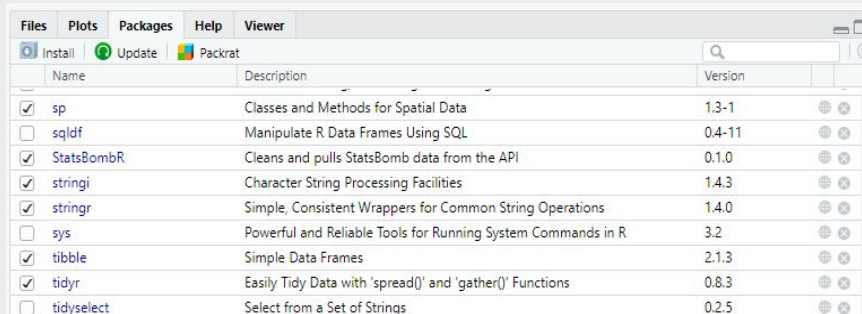
Para instalar **StatsBombR** ejecutar a continuación:

```
devtools::install_github("statsbomb/StatsBombR")
```



STATSBOMB  **R**

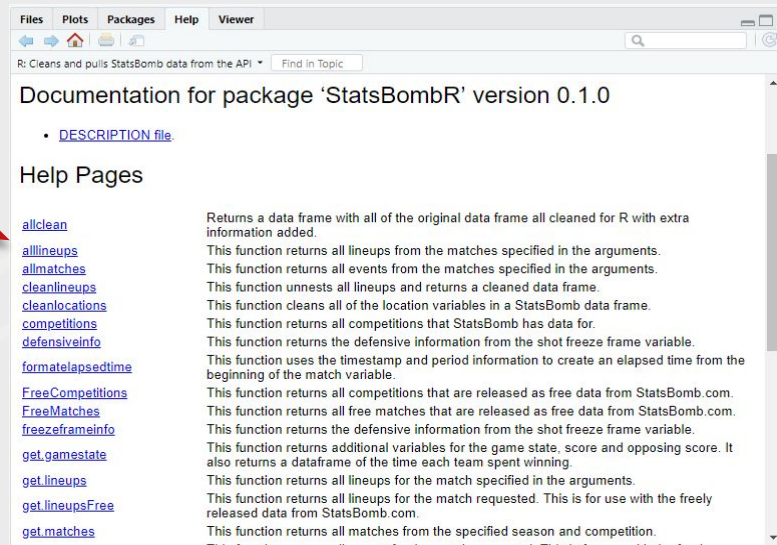
Información Adicional Sobre los Paquetes



| Files | Plots | Packages | Help | Viewer |
|--|--|----------|------|--------|
| Install | Update | Packrat | | |
| Name | Description | Version | | |
| <input checked="" type="checkbox"/> sp | Classes and Methods for Spatial Data | 1.3-1 | | |
| <input type="checkbox"/> sqldf | Manipulate R Data Frames Using SQL | 0.4-11 | | |
| <input checked="" type="checkbox"/> StatsBombR | Cleans and pulls StatsBomb data from the API | 0.1.0 | | |
| <input checked="" type="checkbox"/> stringi | Character String Processing Facilities | 1.4.3 | | |
| <input checked="" type="checkbox"/> stringr | Simple, Consistent Wrappers for Common String Operations | 1.4.0 | | |
| <input type="checkbox"/> sys | Powerful and Reliable Tools for Running System Commands in R | 3.2 | | |
| <input checked="" type="checkbox"/> tibble | Simple Data Frames | 2.1.3 | | |
| <input checked="" type="checkbox"/> tidyr | Easily Tidy Data with 'spread()' and 'gather()' Functions | 0.8.3 | | |
| <input type="checkbox"/> tidyselect | Select from a Set of Strings | 0.2.5 | | |

Para encontrar más información sobre las diferentes funciones dentro de un paquete sólo hay que hacer click en el nombre del paquete como se ve en la imagen.

Esto nos mostrará la información del paquete incluyendo los detalles de sus funciones.



Files Plots Packages Help Viewer

R: Cleans and pulls StatsBomb data from the API Find in Topic

Documentation for package 'StatsBombR' version 0.1.0

- [DESCRIPTION file.](#)

Help Pages

| | |
|-----------------------------------|---|
| allclean | Returns a data frame with all of the original data frame all cleaned for R with extra information added. |
| alllineups | This function returns all lineups from the matches specified in the arguments. |
| allmatches | This function returns all events from the matches specified in the arguments. |
| cleanlineups | This function unnests all lineups and returns a cleaned data frame. |
| cleanlocations | This function cleans all of the location variables in a StatsBomb data frame. |
| competitions | This function returns all competitions that StatsBomb has data for. |
| defensiveinfo | This function returns the defensive information from the shot freeze frame variable. |
| formatelapsedtime | This function uses the timestamp and period information to create an elapsed time from the beginning of the match variable. |
| FreeCompetitions | This function returns all competitions that are released as free data from StatsBomb.com. |
| FreeMatches | This function returns all free matches that are released as free data from StatsBomb.com. |
| freezeinfo | This function returns the defensive information from the shot freeze frame variable. |
| get.gamesstate | This function returns additional variables for the game state, score and opposing score. It also returns a dataframe of the time each team spent winning. |
| get.lineups | This function returns all lineups for the match specified in the arguments. |
| get.lineupsFree | This function returns all lineups for the match requested. This is for use with the freely released data from StatsBomb.com. |
| get.matches | This function returns all matches from the specified season and competition. |



Importar StatsBomb Data

Funciones para Importar los Datos

Para manejar los datos de StatSBomb en R es necesario familiarizarse antes con varias funciones importantes dentro de **StatsBombR**.

FreeCompetitions() – Muestra todas las competiciones disponibles en los datos gratuitos.

Almacenar el output de esta o cualquier otra función en lugar de tenerlo en la consola de R es posible hacerlo ejecutando lo siguiente:

Comp <- FreeCompetitions(). Así, al ejecutar *Comp* (o cualquier palabra utilizada para tal caso) dará el output de *FreeCompetitions()*

Matches <- FreeMatches(Comp) - Muestra todos los partidos disponibles dentro de las competiciones seleccionadas.

StatsBombData <- StatsBombFreeEvents(MatchesDF = Matches, Parallel = T) – Importar todos los datos de evento para los partidos seleccionados.



Importar los Datos

A continuación vamos a ver un ejemplo de cómo importar datos en R. Primero, abrimos un nuevo script para tener el código accesible de la siguiente manera **File -> New File -> R Script**. Se puede guardar en cualquier momento.

```
library(tidyr)  
library(StatsBombR)1
```

```
Comp <- FreeCompetitions() %>%  
filter(competition_id==11 & season_name=="2005/2006")2
```

```
Matches <- FreeMatches(Comp)3
```

```
StatsBombData <- StatsBombFreeEvents(MatchesDF = Matches, Parallel = T)4
```

```
StatsBombData = allclean(StatsBombData)5
```

⁴ : En este punto se ha creado una 'dataframe' (esencialmente una tabla u hoja de datos) llamada StatsBombData (o el nombre elegido para tal caso) con todos los datos de evento gratuitos para la temporada 05/06 de la Liga.

¹ : *tidyr* importa varios paquetes diferentes. Los más importantes para esta tarea son *dplyr* y *magrittr*. *StatsBombR* importa StatsBombR.

² : Importa las competiciones disponibles para el usuario y se filtran utilizando la función 'filter' de *dplyr* para obtener la temporada 05/06 de La Liga en este caso.

³ : Importa todos los partidos de la competición seleccionada.

⁵ : Extrae toda la información relevante previamente descrita.



Trabajar con los datos

Conocer los Datos

En nuestro Github (el mismo lugar donde están los datos) se pueden encontrar documentos adicionales con las especificaciones de StatsBomb Data. Estos están disponibles para ver o descargar y contienen explicaciones a las dudas que puedan surgir sobre los distintos tipos de eventos o cuestiones similares.

Los documentos incluyen:

[Open Data Competitions v2.0.0.pdf](#) – Cubre los objetos contenidos en la información de las competiciones (*FreeCompetitions()*).

[Open Data Matches v3.0.0.pdf](#) – Describe la información de partido para descargar (*FreeMatches()*).

[Open Data Lineups v2.0.0.pdf](#) – Describe la estructura de la información de alineación (*getlineupsFree()*).

[Open Data Events v4.0.0.pdf](#) -- Incluye los significados de los nombres en las columnas dentro de los datos de evento.

[StatsBomb Event Data Specification v1.1.pdf](#) – Descripción detallada de todos los eventos en los datos.



Ejemplos de Uso de los Datos

Una vez que tenemos disponible el archivo StatsBombData vamos a ver varios modos en los que se puede utilizar al mismo tiempo que nos familiarizamos con R. Los ejemplos irán incrementando en grado de dificultad.

Ejemplo 1: Tiros y Goles – Un punto de partida simple pero fundamental. Veremos cómo extraer los números de tiros y goles de cada equipo, primero los totales y luego los de cada partido.

Ejemplo 2: Crear Gráficos de los Tiros - Una vez que tenemos los datos de tiros y goles ¿cómo podemos crear un gráfico a partir de ellos?

Ejemplo 3: Tiros cada 90 minutos – Extraer los tiros para jugadores es relativamente sencillo una vez que sabemos hacerlo para equipos. ¿Pero cómo podemos ajustar los números cada 90 minutos?

Ejemplo 4: Representar Pases Gráficamente – Filtrar los datos extrayendo un subconjunto de datos y visualizarlos sobre un campo empleando para ello *ggplot2*.



Ejemplo 1: Tiros y Goles

```
shots_goals = StatsBombData %>%  
  group_by(team.name) %>%1  
  summarise(shots = sum(type.name=="Shot", na.rm = TRUE),  
  goals = sum(shot.outcome.name=="Goal", na.rm = TRUE))2
```

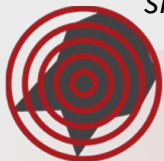
Vamos a desgranarlo paso a paso:

¹: Este código agrupa los datos por equipo, de tal forma que cualquier operación que realicemos en ellos será ejecutada por cada equipo. I.e. extraerá los tiros y goles para cada equipo de manera individual.

²: **Summarise** toma cualquier operación ejecutada y genera una tabla nueva y separada con ello. La mayoría de usos de *summarise* suelen ser después de *group_by*.

shots = sum(type.name=="Shot", na.rm = TRUE) crea una nueva columna llamada 'shots' que suma todas las filas bajo la columna 'type.name' que contienen la palabra 'Shot'. *na.rm = TRUE* pide ignorar cualquier NA dentro de esa columna.

shot.outcome.name=="Goal", na.rm = TRUE) hace lo mismo con los goles.



Ejemplo 1: Tiros y Goles

En este punto deberíamos tener una tabla como esta.

Para realizar el mismo cálculo por partido en lugar de los totales solo tenemos que cambiarlo de la siguiente manera:

```
shots_goals = StatsBombData %>%  
  group_by(team.name) %>%  
  summarise(shots = sum(type.name=="Shot", na.rm =  
    TRUE)/n_distinct(match_id),  
    goals = sum(shot.outcome.name=="Goal", na.rm =  
    TRUE)/n_distinct(match_id))
```

Añadir `'/n_distinct(match_id)'` implica que estamos dividiendo el número de tiros/goles entre el número de partidos para cada equipo.

Totals

| team.name | shots | goals |
|----------------------------|-------|-------|
| Chelsea LFC | 392 | 38 |
| Manchester City WFC | 389 | 51 |
| Arsenal WFC | 373 | 68 |
| Reading WFC | 295 | 31 |
| Birmingham City WFC | 254 | 25 |
| Everton LFC | 230 | 14 |
| West Ham United LFC | 217 | 25 |
| Brighton & Hove Albion WFC | 207 | 16 |
| Liverpool WFC | 177 | 17 |
| Bristol City WFC | 153 | 15 |
| Yeovil Town LFC | 129 | 10 |

Per Game

| team.name | shots | goals |
|----------------------------|-----------|-----------|
| Chelsea LFC | 20.631579 | 2.0000000 |
| Manchester City WFC | 19.450000 | 2.5500000 |
| Arsenal WFC | 18.650000 | 3.4000000 |
| Reading WFC | 14.750000 | 1.5500000 |
| Birmingham City WFC | 13.368421 | 1.3157895 |
| Everton LFC | 12.105263 | 0.7368421 |
| Brighton & Hove Albion WFC | 10.894737 | 0.8421053 |
| West Ham United LFC | 10.850000 | 1.2500000 |
| Liverpool WFC | 9.315789 | 0.8947368 |
| Bristol City WFC | 7.650000 | 0.7500000 |
| Yeovil Town LFC | 6.789474 | 0.5263158 |



Ejemplo 2: Gráficos de los Datos

```
library(ggplot2)
```

```
ggplot(data = shots_goals, aes(x = reorder(team.name, shots), y = shots))1 +  
  geom_bar(stat = "identity", width = 0.5)2 +  
  labs(y="Shots")3 +  
  theme(axis.title.y = element_blank())4 +  
  scale_y_continuous(expand = c(0,0))5 +  
  coord_flip()6 +  
  theme_SB()7
```

¹: Aquí estamos diciendo a ggplot qué datos estamos utilizando y qué queremos en los ejes x/y del gráfico. 'Reorder' ordena los nombres de los equipos en función de los tiros.

³: Cambia el nombre del eje de tiros.

²: Pide a ggplot formatearlo como un gráfico de barras.

⁴: Elimina el título del eje.

⁵: Aquí podemos reducir el espacio entre las barras y el límite del gráfico.

⁶: Rota el gráfico completo colocando las barras en sentido horizontal.

⁷: *theme_SB()* es el tema visual interno para gráficos de ggplot incluido en StatsBombR. Su uso es opcional.



Ejemplo 2: Gráficos de los Datos

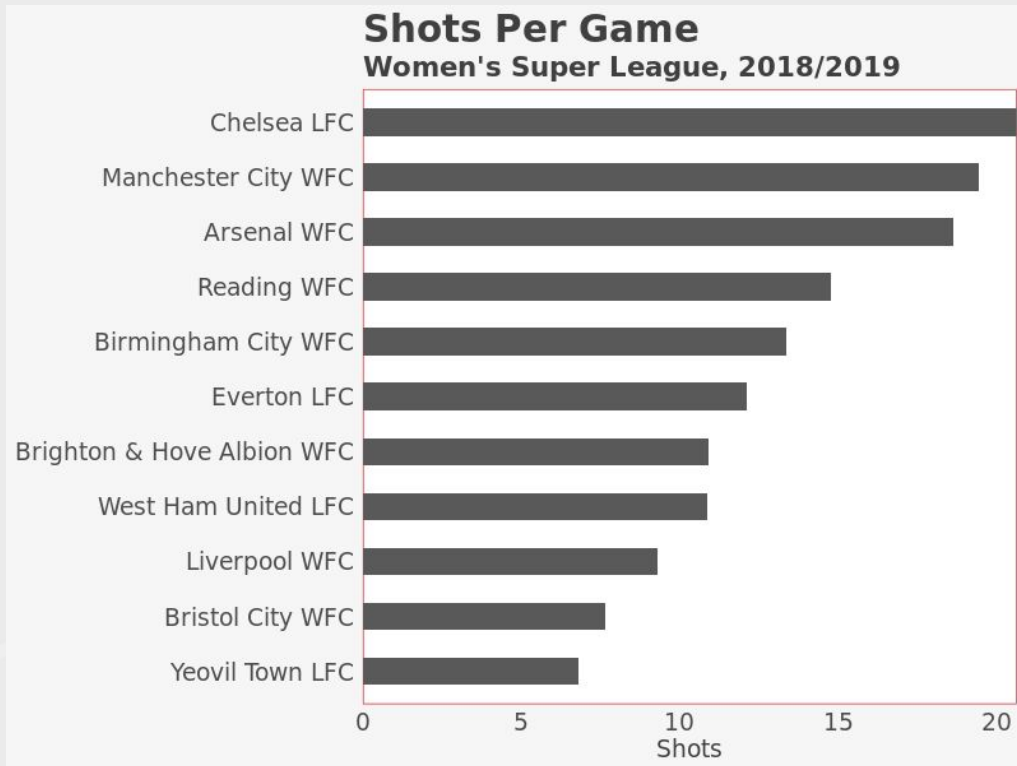
Lo anterior debería generar un gráfico como este.

El diseño obtenido es básico y diáfano. Puede ser modificado de diferentes maneras para conseguir un visual más atractivo.

Cualquier elemento de un gráfico *ggplot* desde el texto a los datos en sí puede ser modificado de numerosas maneras abriendo la puerta a la creatividad del usuario.

Más información sobre el tipo de diseños que se pueden conseguir:

<https://ggplot2.tidyverse.org/reference/>



Ejemplo 3: Tiros Cada 90 Minutos

```
player_shots = StatsBombData %>%  
  group_by(player.name, player.id) %>%  
  summarise(shots = sum(type.name=="Shot", na.rm = TRUE)) 1
```

```
player_minutes = get.minutesplayed(StatsBombData) 2
```

```
player_minutes = player_minutes %>%  
  group_by(player.id) %>%  
  summarise(minutes = sum(MinutesPlayed)) 3
```

```
player_shots = left_join(player_shots, player_minutes) 4
```

```
player_shots = player_shots %>% mutate(nineties = minutes/90) 5
```

```
player_shots = player_shots %>% mutate(shots_per90 = shots/nineties) 6
```

¹ : Similar al cálculo para los equipos. Incluimos aquí 'player.id' ya que será importante después.

² : Esta función obtiene los minutos de cada jugador en cada partido en la muestra.

³ : Agrupamos lo anterior sumando los minutos en cada partido para obtener el total de minutos disputados por cada jugador.

⁴ : *left_join* combina las tablas de tiros y de minutos con el player.id actuando como punto de referencia.

⁵ : *mutate* es una función dplyt que crea una nueva columna. En este caso estamos creando una columna que divide los minutos totales entre 90 dando como resultado el número de 90s del jugador en la temporada.

⁶ : Finalmente dividimos los tiros totales entre el número de 90s para obtener la columna de tiros cada 90 minutos (shots per 90).



Ejemplo 3: Tiros Cada 90 Minutos

En este punto tendremos los tiros cada 90 minutos para todas las jugadoras de la WSL.

A continuación, se puede filtrar la tabla eliminando a las jugadoras con insuficiente muestra mediante la función `'filter'` (dplyr).

El mismo proceso puede ser aplicado a todo tipo de eventos dentro de StatsBomb Data. Diferentes tipos de pases, acciones defensivas, etc.

| player.name | player.id | shots | minutes | nineties | shots_per90 |
|------------------|-----------|-------|-------------|-------------|-------------|
| Vivianne Miedema | 15623 | 112 | 1881.406033 | 20.90451148 | 5.35769516 |
| Missy Bo Kearns | 24237 | 1 | 18.450033 | 0.20500037 | 4.87803997 |
| Pauline Bremer | 20725 | 7 | 131.822717 | 1.46469685 | 4.77914593 |
| Francesca Kirby | 4641 | 54 | 1027.349783 | 11.41499759 | 4.73061861 |
| Georgia Stanway | 4643 | 71 | 1410.878733 | 15.67643037 | 4.52909229 |
| Bethany England | 15550 | 51 | 1053.991783 | 11.71101981 | 4.35487266 |
| Jordan Nobbs | 10192 | 35 | 738.513717 | 8.20570796 | 4.26532362 |
| Danielle Carter | 24281 | 5 | 105.765800 | 1.17517556 | 4.25468346 |
| Ellen White | 10180 | 31 | 689.171150 | 7.65745722 | 4.04834126 |
| Nikita Parris | 4654 | 74 | 1705.892717 | 18.95436352 | 3.90411421 |
| Fara Williams | 10251 | 76 | 1848.079417 | 20.53421574 | 3.70113965 |
| Erin Cuthbert | 4660 | 54 | 1392.178833 | 15.46865370 | 3.49093082 |
| Nadia Nadim | 4650 | 7 | 183.808083 | 2.04231204 | 3.42748800 |
| Janine Beckie | 4992 | 12 | 323.301367 | 3.59223741 | 3.34053645 |
| Lauren Hemp | 15555 | 24 | 647.574350 | 7.19527056 | 3.33552433 |



Ejemplo 4: Representar Pases Gráficamente

Finalmente, vamos a trazar los pases de un jugador en el campo. Para esto necesitaremos en primer lugar una visualización de un campo de fútbol. Es posible crear uno propio una vez estemos familiarizados con ggplot que pueda ser utilizado además para diferentes propósitos. Más adelante veremos opciones para ello. De momento, hay opciones ya formateadas que podemos utilizar.

La que utilizaremos aquí es cortesía de [FC rStats](#). Este usuario de Twitter ha creado varios paquetes públicos de R para analizar datos de fútbol. El paquete que nos ocupa se llama '[SBPitch](#)' y sirve exactamente para eso. En '*Paquetes Adicionales*' veremos otras alternativas para crear campos de juego.

Para instalar *SBPitch* ejecutamos:

```
devtools::install_github("FCrSTATS/SBpitch")
```

Vamos a representar los pases completados por Messi dentro del área en la Liga 05/06. Trazar todos los pases sería farragoso y poco útil por tanto elegimos un subconjunto. Es importante asegurarse de utilizar las funciones explicadas anteriormente para importar los datos.



Ejemplo 4: Representar Pases Gráficamente

```
library(SBpitch)
```

```
passes = messidata %>%
```

```
  filter(type.name=="Pass" & is.na(pass.type.name) & player.id==5503)1 %>%
```

```
  filter(pass.end_location.x>=102 & pass.end_location.y<=62 & pass.end_location.y>=18)2
```

```
create_Pitch() +
```

```
  geom_segment(data = passes, aes(x = location.x, y = location.y,  
    xend = pass.end_location.x, yend = pass.end_location.y),
```

```
    lineend = "round", size = 0.6, arrow = arrow(length = unit(0.08, "inches")))3 +
```

```
  labs(title = "Lionel Messi, Completed Box Passes", subtitle = "La Liga, 2005/2006")4 +
```

```
  coord_fixed(ratio = 105/100)5
```

⁴ : Crea un título y subtítulo para el gráfico. Entre otras opciones se puede añadir una leyenda usando *caption* =.

¹ : Filtrar los pases de Messi.

is.na(pass.type.name) filtrar solo los pases completados.

² : Filtrar los pases dentro del área. Las coordenadas de las zonas del campo en SBD se pueden encontrar en nuestro [event spec](#).

³ : Obtenemos una flecha desde un punto de origen (location.x/y inicio del pase) a un punto final (pass.end_location.x/y, final del pase). *Lineend*, *size* y *length* son las opciones de customización disponibles aquí.

⁵ : Ajusta el gráfico a la relación de aspecto elegida para que no quede estirado o poco estético.



Ejemplo 4: Representar Pases Gráficamente

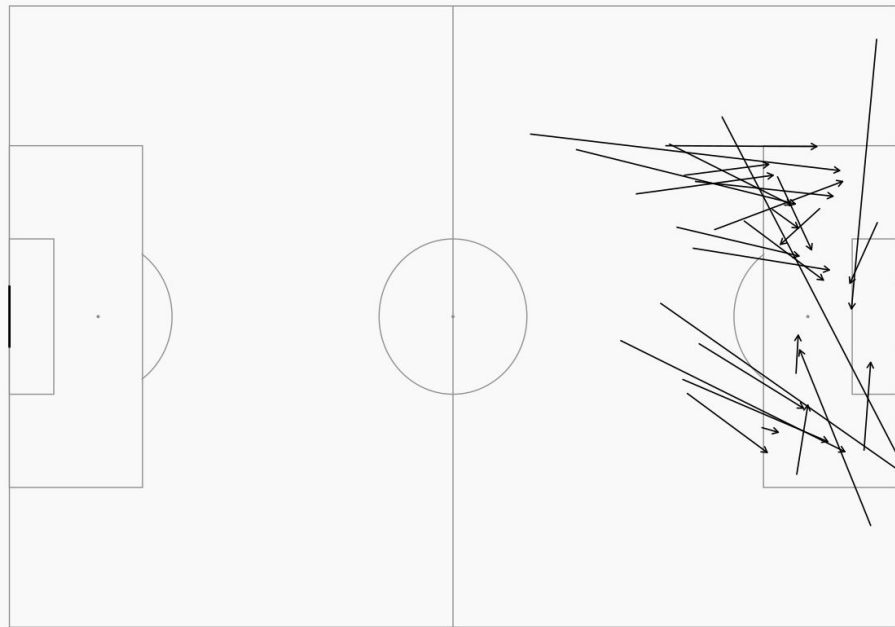
El resultado será un gráfico tal que así. De nuevo, esta es una versión básica a partir de la cual se pueden implementar todo tipo de mejoras visuales.

La opción *theme()* permite cambiar el tamaño, posición, fuente y otros aspectos de los títulos así como otros apartados estéticos del gráfico.

Es posible añadir *colour=* a *geom_segment()* para colorear los las flechas de cada pase del modo escogido.

En el siguiente enlace se pueden encontrar diferentes posibilidades disponibles para customizar los gráficos: [ggplot Cheat Sheet](#).

Lionel Messi, Completed Box Passes
La Liga, 2005/2006



Funciones útiles en StatsBombR

Existen docenas de funciones dentro de StatsBombR para realizar diferentes tareas. Se puede consultar la lista completa [aquí](#). No todas las funciones están disponibles en los datos gratuitos. Algunas solo son accesibles para nuestros clientes (vía API). Una pequeña muestra de las más útiles:

get.playerfootedness() – Devuelve la pierna hábil (preferida) de un jugador a partir de nuestros datos de pases (incluyen la pierna con la que se realiza el pase).

get.opposingteam() – Devuelve una columna opuesta para cada equipo en cada partido.

get.gamestate() – Devuelve la información de cuánto tiempo acumula cada equipo en cada uno de los posibles *Game States* (ganando/empatando/perdiendo).

annotate_pitchSB() – Nuestra solución para trazar un campo de juego en ggplot.



Paquetes adicionales

La comunidad ha desarrollado múltiples paquetes para R. Es probable que cualquier cuestión o tarea que se quiera llevar a cabo en R tenga desarrollado un paquete específico para ella. Nombrar todos sería imposible pero aquí va una pequeña selección de algunos que son relevantes para trabajar con StatsBomb Data:

[Ben Torvaney, ggsoccer](#) - Alternativa para trazar campos de juego con StatsBomb Data.

[Joe Gallagher, soccermatics](#) – Otra alternativa para dibujar campos de juego incluyendo además atajos sencillos para crear mapas de calor entre otras funciones.

[ggrepel](#) – Solución para problemas de texto superpuesto en las gráficas.

[gganimate](#) – Opción sencilla para crear gráficos animados con ggplot en R.



¡Esperamos que disfruten de los datos!

Preguntas y sugerencias:

Euan Dewar euan.dewar@statsbomb.com

Traducido por: Pablo P. Rodríguez