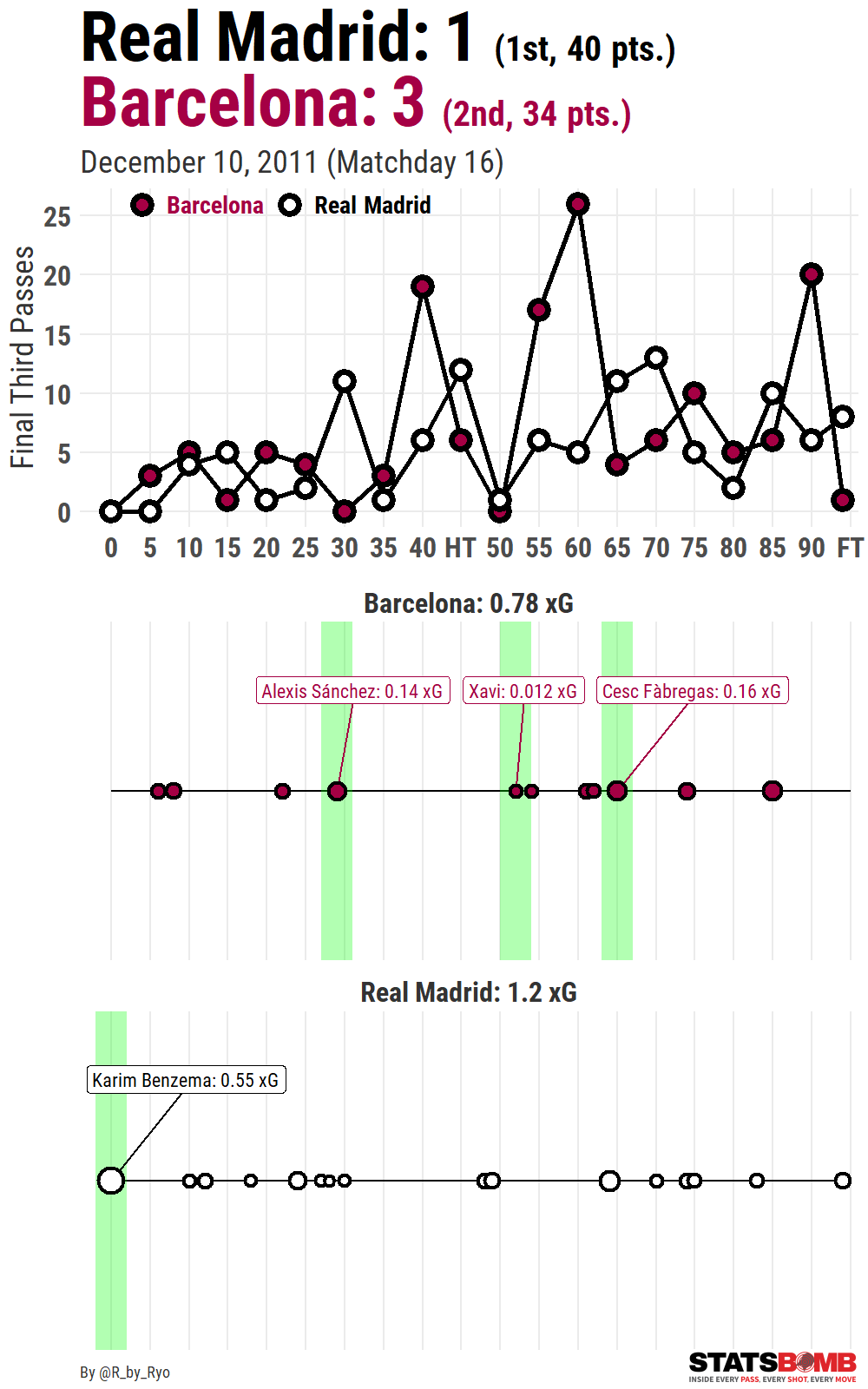
This will be **Part 1** of what I hope to be a multi-part series of  
plotting soccer event-level data with R! This is more of a tutorial blog  
post rather than a deep analytical piece but I will give some context to  
the examples to set the scene! I can’t give an exact number of how many  
parts as I am still getting to grips with this kind of data and I feel  
like I’ve only scratched the surface. You can read some of the other  
stuff I’ve done, preview blog posts for the [Asian  
Cup](https://ryo-n7.github.io/2019-01-11-visualize-asian-cup/) and the  
[Copa  
America](https://ryo-n7.github.io/2019-06-18-visualize-copa-america/),  
along with the code to all the standalone viz I’ve done on my  
[soccer\_ggplot GitHub  
repository](https://github.com/Ryo-N7/soccer_ggplots).

I’ll mostly be using the **Messi Data Biography** data but the steps I  
show below are applicable to the other data available as well! I will be  
working with the **free** data sets so some things may differ compared  
to the full data available. Also note that it is possible to create the  
viz in this blog post using data from other providers of event-level  
data such as Opta. The difference in code will mainly be in the data  
ingestion and cleaning phases but the gist of the {ggplot2} code should  
be similar.

As an example and motivation, one of the visualizations we are going to  
create is shown below:



Let’s get started!

**Getting the Data**

A few important steps before you even start using R:

* Please read the [StatsBomb/Open-Data Github  
  repository](https://github.com/statsbomb/open-data)
* Register your details  
  [here](https://www.statsbomb.com/resource-centre)
* Read and agree to the [User  
  Agreement](https://github.com/statsbomb/open-data/blob/master/LICENSE.pdf)
* Install the {StatsBombR} package via  
  devtools::install\_github("statsbomb/StatsBombR")
* You might also want to take a look at the documentation found  
  [here](https://github.com/statsbomb/open-data/tree/master/doc)

Once that’s done we can start coding!

**Packages**

Here’s all the packages I’ll be using (note: I like using {pacman} so I  
don’t have to repeat library() a billion times):

if (!require("pacman")) {

install.packages("pacman")

}

pacman::p\_load(tidyverse, ## mainly dplyr, purrr, and tidyr

StatsBombR, SBpitch, soccermatics,

extrafont, ggupset, tibbletime,

ggtext, ggrepel, glue,

patchwork, cowplot, gtable, grid,

magick)

## loading fonts

loadfonts(device = "win", quiet = TRUE)

After loading the  
[{StatsBombR}](https://github.com/statsbomb/StatsBombR) library (note:  
that I already did this above but just showing it again below for  
demonstrational purposes) we first want to take a look at the output of  
the FreeCompetitions() function which gives you a data frame of all  
the competitions available for free from StatsBomb. Do note that this  
part will be different if you are a customer using the API.

library(StatsBombR)

comps <- FreeCompetitions()

glimpse(comps)

If you View() or glimpse() the data frame you’ll see that the  
competition\_id we need is **11** for the Lionel Messi data. We use  
this to filter() the comps data frame and then call FreeMatches()  
to get a data frame of the available matches. Finally pass that data  
frame to StatsBombFreeEvents() to access the data, this can take a  
while if you don’t have a good internet connection!

messi\_matches\_raw <- comps %>%

filter(competition\_id == 11) %>%

FreeMatches()

messi\_data\_raw <- StatsBombFreeEvents(MatchesDF = messi\_matches\_raw)

**Clean All and Add Season Labels**

Now that we’ve got the raw data we can clean it and add some extra  
information using the allclean() function. This function takes care  
of:

* cleanlocations(): cleans the location variables in the data
* Goalkeeper: Add goalkeeper data from the freeze frame
* Shot: Adds more shot information
* Freeze frame: Extracts info from freeze frames, i.e. density
* Defensive: Defensive information

We can also add in the actual season names by joining with the “comps”  
data frame and joining it by the “season\_id”.

messi\_data\_clean <- messi\_data\_raw %>%

allclean() %>%

left\_join(comps %>% select(season\_id, season\_name), by = "season\_id")

The player names in the data are the full names and for lots of  
Spanish/Portuguese players in the data that means their **FULL** names.  
To make the names shorter and so that labels on plots can be more  
legible it’s a good idea to clean the “name” variables up a bit. There  
is a function, JoinPlayerNickName() that allows you to do that,  
however, you need a username and password for the StatsBomb API, which I  
don’t have sooo… I have several options:

* Manually clean the names…
* Find a nice list of player names and left\_join() after cleaning
  + Example: Use transfermarkt data
* Use the [{fuzzyjoin}](https://github.com/dgrtwo/fuzzyjoin) package:  
  Join a name even if there are n number of differences

In the end I just did it manually… around 10 full minutes of hard  
concentration and it was done. Added bonus is that now I am intimately  
familiar with the full names of every Barcelona player in the past  
decade!

messi\_data\_clean <- messi\_data\_clean %>%

## player name

mutate([player.name](http://player.name" \t "_blank) = case\_when(

[player.name](http://player.name) == "Oleguer Presas Renom" ~ "Oleguer",

[player.name](http://player.name) == "Xavier Hernández Creus" ~ "Xavi",

[player.name](http://player.name) == "Carles Puyol i Saforcada" ~ "Carles Puyol",

[player.name](http://player.name) == "Anderson Luís de Souza" ~ "Deco",

[player.name](http://player.name) == "Rafael Márquez Álvarez" ~ "Rafa Márquez",

[player.name](http://player.name) == "Giovanni van Bronckhorst" ~ "Gio v.Bronckhorst",

[player.name](http://player.name) == "Samuel Eto'o Fils" ~ "Samuel Eto'o",

[player.name](http://player.name) == "Víctor Valdés Arribas" ~ "Víctor Valdés",

[player.name](http://player.name) == "Juliano Haus Belletti" ~ "Juliano Belletti",

[player.name](http://player.name) == "Ludovic Giuly" ~ "Ludovic Giuly",

[player.name](http://player.name) == "Andrés Iniesta Luján" ~ "Andrés Iniesta",

[player.name](http://player.name) == "Ronaldo de Assis Moreira" ~ "Ronaldinho",

[player.name](http://player.name) == "Lionel Andrés Messi Cuccittini" ~ "Lionel Messi",

[player.name](http://player.name) == "Fernando Navarro i Corbacho" ~ "Fernando Navarro",

[player.name](http://player.name) == "Sylvio Mendes Campos Junior" ~ "Sylvinho",

[player.name](http://player.name) == "Damià Abella Pérez" ~ "Damià",

[player.name](http://player.name) == "Rubén Iván Martínez Andrade" ~ "Ronaldinho",

[player.name](http://player.name) == "Ronaldo de Assis Moreira" ~ "Rubén",

[player.name](http://player.name) == "Thiago Motta" ~ "Thiago Motta",

[player.name](http://player.name) == "Mark van Bommel" ~ "Mark van Bommel",

[player.name](http://player.name) == "Henrik Larsson" ~ "Henrik Larsson",

[player.name](http://player.name) == "José Edmílson Gomes de Moraes" ~ "Edmílson",

[player.name](http://player.name) == "Gabriel Francisco García de la Torre" ~ "Gabri",

[player.name](http://player.name) == "Santiago Ezquerro Marín" ~ "Santi Ezquerro",

[player.name](http://player.name) == "Maximiliano Gastón López" ~ "Maxi López",

[player.name](http://player.name) == "Gianluca Zambrotta" ~ "Gianluca Zambrotta",

[player.name](http://player.name) == "Eiður Smári Guðjohnsen" ~ "Eiður Guðjohnsen",

[player.name](http://player.name) == "Lilian Thuram" ~ "Lilian Thuram",

[player.name](http://player.name) == "Javier Pedro Saviola Fernández" ~ "Javier Saviola",

[player.name](http://player.name) == "Gnégnéri Yaya Touré" ~ "Yaya Touré",

[player.name](http://player.name) == "Bojan Krkíc Pérez" ~ "Bojan",

[player.name](http://player.name) == "Eric-Sylvain Bilal Abidal" ~ "Eric Abidal",

[player.name](http://player.name) == "Gabriel Alejandro Milito" ~ "Gabriel Milito",

[player.name](http://player.name) == "Giovani dos Santos Ramírez" ~ "Giovani dos Santos",

[player.name](http://player.name) == "Víctor Vázquez Solsona" ~ "Víctor Vázquez",

[player.name](http://player.name) == "Thierry Henry" ~ "Thierry Henry",

[player.name](http://player.name) == "José Manuel Pinto Colorado" ~ "José Manuel Pinto",

[player.name](http://player.name) == "Daniel Alves da Silva" ~ "Dani Alves",

[player.name](http://player.name) == "Sergio Busquets i Burgos" ~ "Sergio Busquets",

[player.name](http://player.name) == "Seydou Kéita" ~ "Seydou Kéita",

[player.name](http://player.name) == "José Martín Cáceres Silva" ~ "Martín Cáceres",

[player.name](http://player.name) == "Gerard Piqué Bernabéu" ~ "Gerard Piqué",

[player.name](http://player.name) == "Aliaksandr Hleb" ~ "Aliaksandr Hleb",

[player.name](http://player.name) == "Pedro Eliezer Rodríguez Ledesma" ~ "Pedro",

[player.name](http://player.name) == "Sergio Rodríguez García" ~ "Rodri",

[player.name](http://player.name) == "Rafael Romero Serrano" ~ "Fali",

[player.name](http://player.name) == "José Manuel Rueda Sampedro" ~ "José Manuel Rueda",

[player.name](http://player.name) == "Zlatan Ibrahimovic" ~ "Zlatan Ibrahimovic",

[player.name](http://player.name) == "Dmytro Chygrynskiy" ~ "Dmytro Chygrynskiy",

[player.name](http://player.name) == "Maxwell Scherrer Cabelino Andrade" ~ "Maxwell",

[player.name](http://player.name) == "Jeffren Isaac Suárez Bermúdez" ~ "Jeffren",

[player.name](http://player.name) == "Víctor Sánchez Mata" ~ "Víctor Sánchez",

[player.name](http://player.name) == "Thiago Alcântara do Nascimento" ~ "Thiago Alcântara",

[player.name](http://player.name) == "David Villa Sánchez" ~ "David Villa",

[player.name](http://player.name) == "Javier Alejandro Mascherano" ~ "Javier Mascherano",

[player.name](http://player.name) == "Andreu Fontàs Prat" ~ "Andreu Fontàs",

[player.name](http://player.name) == "Ibrahim Afellay" ~ "Ibrahim Afellay",

[player.name](http://player.name) == "Manuel Agudo Durán" ~ "Nolito",

[player.name](http://player.name) == "Marc Bartra Aregall" ~ "Marc Bartra",

[player.name](http://player.name) == "Adriano Correia Claro" ~ "Adriano",

[player.name](http://player.name) == "Martín Montoya Torralbo" ~ "Martín Montoya",

[player.name](http://player.name) == "Jonathan dos Santos Ramírez" ~ "Jonathan dos Santos",

[player.name](http://player.name) == "Francesc Fàbregas i Soler" ~ "Cesc Fàbregas",

[player.name](http://player.name) == "Alexis Alejandro Sánchez Sánchez" ~ "Alexis Sánchez",

[player.name](http://player.name) == "Juan Isaac Cuenca López" ~ "Isaac Cuenca",

[player.name](http://player.name) == "Gerard Deulofeu Lázaro" ~ "Gerard Deulofeu",

[player.name](http://player.name) == "Cristian Tello" ~ "Cristian Tello",

[player.name](http://player.name) == "Sergi Roberto Carnicer" ~ "Sergi Roberto",

[player.name](http://player.name) == "Marc Muniesa Martínez" ~ "Marc Muniesa",

TRUE ~ [player.name](http://player.name)

)) %>%

## [pass.recipient.name](http://pass.recipient.name)

mutate([pass.recipient.name](http://pass.recipient.name" \t "_blank) = case\_when(

[pass.recipient.name](http://pass.recipient.name) == "Oleguer Presas Renom" ~ "Oleguer",

[pass.recipient.name](http://pass.recipient.name) == "Xavier Hernández Creus" ~ "Xavi",

[pass.recipient.name](http://pass.recipient.name) == "Carles Puyol i Saforcada" ~ "Carles Puyol",

[pass.recipient.name](http://pass.recipient.name) == "Anderson Luís de Souza" ~ "Deco",

[pass.recipient.name](http://pass.recipient.name) == "Rafael Márquez Álvarez" ~ "Rafa Márquez",

[pass.recipient.name](http://pass.recipient.name) == "Giovanni van Bronckhorst" ~ "Gio v.Bronckhorst",

[pass.recipient.name](http://pass.recipient.name) == "Samuel Eto'o Fils" ~ "Samuel Eto'o",

[pass.recipient.name](http://pass.recipient.name) == "Víctor Valdés Arribas" ~ "Víctor Valdés",

[pass.recipient.name](http://pass.recipient.name) == "Juliano Haus Belletti" ~ "Juliano Belletti",

[pass.recipient.name](http://pass.recipient.name) == "Ludovic Giuly" ~ "Ludovic Giuly",

[pass.recipient.name](http://pass.recipient.name) == "Andrés Iniesta Luján" ~ "Andrés Iniesta",

[pass.recipient.name](http://pass.recipient.name) == "Ronaldo de Assis Moreira" ~ "Ronaldinho",

[pass.recipient.name](http://pass.recipient.name) == "Lionel Andrés Messi Cuccittini" ~ "Lionel Messi",

[pass.recipient.name](http://pass.recipient.name) == "Fernando Navarro i Corbacho" ~ "Fernando Navarro",

[pass.recipient.name](http://pass.recipient.name) == "Sylvio Mendes Campos Junior" ~ "Sylvinho",

[pass.recipient.name](http://pass.recipient.name) == "Damià Abella Pérez" ~ "Damià",

[pass.recipient.name](http://pass.recipient.name) == "Rubén Iván Martínez Andrade" ~ "Ronaldinho",

[pass.recipient.name](http://pass.recipient.name) == "Ronaldo de Assis Moreira" ~ "Rubén",

[pass.recipient.name](http://pass.recipient.name) == "Thiago Motta" ~ "Thiago Motta",

[pass.recipient.name](http://pass.recipient.name) == "Mark van Bommel" ~ "Mark van Bommel",

[pass.recipient.name](http://pass.recipient.name) == "Henrik Larsson" ~ "Henrik Larsson",

[pass.recipient.name](http://pass.recipient.name) == "José Edmílson Gomes de Moraes" ~ "Edmílson",

[pass.recipient.name](http://pass.recipient.name) == "Gabriel Francisco García de la Torre" ~ "Gabri",

[pass.recipient.name](http://pass.recipient.name) == "Santiago Ezquerro Marín" ~ "Santi Ezquerro",

[pass.recipient.name](http://pass.recipient.name) == "Maximiliano Gastón López" ~ "Maxi López",

[pass.recipient.name](http://pass.recipient.name) == "Gianluca Zambrotta" ~ "Gianluca Zambrotta",

[pass.recipient.name](http://pass.recipient.name) == "Eiður Smári Guðjohnsen" ~ "Eiður Guðjohnsen",

[pass.recipient.name](http://pass.recipient.name) == "Lilian Thuram" ~ "Lilian Thuram",

[pass.recipient.name](http://pass.recipient.name) == "Javier Pedro Saviola Fernández" ~ "Javier Saviola",

[pass.recipient.name](http://pass.recipient.name) == "Gnégnéri Yaya Touré" ~ "Yaya Touré",

[pass.recipient.name](http://pass.recipient.name) == "Bojan Krkíc Pérez" ~ "Bojan",

[pass.recipient.name](http://pass.recipient.name) == "Eric-Sylvain Bilal Abidal" ~ "Eric Abidal",

[pass.recipient.name](http://pass.recipient.name) == "Gabriel Alejandro Milito" ~ "Gabriel Milito",

[pass.recipient.name](http://pass.recipient.name) == "Giovani dos Santos Ramírez" ~ "Giovani dos Santos",

[pass.recipient.name](http://pass.recipient.name) == "Víctor Vázquez Solsona" ~ "Víctor Vázquez",

[pass.recipient.name](http://pass.recipient.name) == "Thierry Henry" ~ "Thierry Henry",

[pass.recipient.name](http://pass.recipient.name) == "José Manuel Pinto Colorado" ~ "José Manuel Pinto",

[pass.recipient.name](http://pass.recipient.name) == "Daniel Alves da Silva" ~ "Dani Alves",

[pass.recipient.name](http://pass.recipient.name) == "Sergio Busquets i Burgos" ~ "Sergio Busquets",

[pass.recipient.name](http://pass.recipient.name) == "Seydou Kéita" ~ "Seydou Kéita",

[pass.recipient.name](http://pass.recipient.name) == "José Martín Cáceres Silva" ~ "Martín Cáceres",

[pass.recipient.name](http://pass.recipient.name) == "Gerard Piqué Bernabéu" ~ "Gerard Piqué",

[pass.recipient.name](http://pass.recipient.name) == "Aliaksandr Hleb" ~ "Aliaksandr Hleb",

[pass.recipient.name](http://pass.recipient.name) == "Pedro Eliezer Rodríguez Ledesma" ~ "Pedro",

[pass.recipient.name](http://pass.recipient.name) == "Sergio Rodríguez García" ~ "Rodri",

[pass.recipient.name](http://pass.recipient.name) == "Rafael Romero Serrano" ~ "Fali",

[pass.recipient.name](http://pass.recipient.name) == "José Manuel Rueda Sampedro" ~ "José Manuel Rueda",

[pass.recipient.name](http://pass.recipient.name) == "Zlatan Ibrahimovic" ~ "Zlatan Ibrahimovic",

[pass.recipient.name](http://pass.recipient.name) == "Dmytro Chygrynskiy" ~ "Dmytro Chygrynskiy",

[pass.recipient.name](http://pass.recipient.name) == "Maxwell Scherrer Cabelino Andrade" ~ "Maxwell",

[pass.recipient.name](http://pass.recipient.name) == "Jeffren Isaac Suárez Bermúdez" ~ "Jeffren",

[pass.recipient.name](http://pass.recipient.name) == "Víctor Sánchez Mata" ~ "Víctor Sánchez",

[pass.recipient.name](http://pass.recipient.name) == "Thiago Alcântara do Nascimento" ~ "Thiago Alcântara",

[pass.recipient.name](http://pass.recipient.name) == "David Villa Sánchez" ~ "David Villa",

[pass.recipient.name](http://pass.recipient.name) == "Javier Alejandro Mascherano" ~ "Javier Mascherano",

[pass.recipient.name](http://pass.recipient.name) == "Andreu Fontàs Prat" ~ "Andreu Fontàs",

[pass.recipient.name](http://pass.recipient.name) == "Ibrahim Afellay" ~ "Ibrahim Afellay",

[pass.recipient.name](http://pass.recipient.name) == "Manuel Agudo Durán" ~ "Nolito",

[pass.recipient.name](http://pass.recipient.name) == "Marc Bartra Aregall" ~ "Marc Bartra",

[pass.recipient.name](http://pass.recipient.name) == "Adriano Correia Claro" ~ "Adriano",

[pass.recipient.name](http://pass.recipient.name) == "Martín Montoya Torralbo" ~ "Martín Montoya",

[pass.recipient.name](http://pass.recipient.name) == "Jonathan dos Santos Ramírez" ~ "Jonathan dos Santos",

[pass.recipient.name](http://pass.recipient.name) == "Francesc Fàbregas i Soler" ~ "Cesc Fàbregas",

[pass.recipient.name](http://pass.recipient.name) == "Alexis Alejandro Sánchez Sánchez" ~ "Alexis Sánchez",

[pass.recipient.name](http://pass.recipient.name) == "Juan Isaac Cuenca López" ~ "Isaac Cuenca",

[pass.recipient.name](http://pass.recipient.name) == "Gerard Deulofeu Lázaro" ~ "Gerard Deulofeu",

[pass.recipient.name](http://pass.recipient.name) == "Cristian Tello" ~ "Cristian Tello",

[pass.recipient.name](http://pass.recipient.name) == "Sergi Roberto Carnicer" ~ "Sergi Roberto",

[pass.recipient.name](http://pass.recipient.name) == "Marc Muniesa Martínez" ~ "Marc Muniesa",

TRUE ~ [pass.recipient.name](http://pass.recipient.name)

))

I only changed it for these two variables but you could do it for more  
using the scoped variants of mutate() such as mutate\_at() or  
mutate\_if() to change the values of variables that adhere to certain  
conditions.

**Save Cleaned Data**

Now that we’ve got a clean data set it might be a good idea to save it.  
I use the here::here() function for setting the path root to the  
top-level of the current project directory and then jumping into the  
“data” folder. Read this blog post  
[here](https://malco.io/2018/11/05/why-should-i-use-the-here-package/)  
for more info on why it’s useful to do so.

saveRDS(messi\_data\_clean, file = here::here("data/messi\_data\_clean.RDS"))

To get data for the other data sets it’s a matter of finding and  
filtering for the correct “competition\_id”. For the Women’s World Cup  
data that’ll be **72** and for the Men’s World Cup last year it’ll be  
**43**. The other data cleaning steps are the same.

With a nice clean data set ready, we can move on to reshaping the data  
for analysis and plotting!

**xG Timeline**

**Data**

To get the data for a single match, in this case an **“El Clasico”**  
match from the 2011/2012 season, we filter() for its “match\_id”  
number. Our main statistic of interest for the next two plots is going  
to be xG in the “shot.statsbomb\_xg” variable. If the value for it is  
NA we can safely set the value to **0**, otherwise we just keep the  
value for that row.

We also create a separate data set that sums up the total xG for both  
teams and creates a nice label using the {glue} package. The  
“team\_label” variable will come in handy in the plots. After joining  
that data frame in, we also create a “player\_label” variable to store  
the “[player.name](http://player.name)” and “shot.statsbomb\_xg” values for rows where a  
**Goal** was scored. This variable will also be used as labels in the  
plots.

clasico\_1112 <- messi\_data\_clean %>%

filter(match\_id == 69334) %>%

mutate(shot.statsbomb\_xg = if\_else([is.na](http://is.na)(shot.statsbomb\_xg),

0, shot.statsbomb\_xg))

clasico\_1112\_xg <- clasico\_1112 %>%

group\_by([team.name](http://team.name)) %>%

summarize(tot\_xg = sum(shot.statsbomb\_xg) %>% signif(digits = 2)) %>%

mutate(team\_label = glue::glue("{[team.name](http://team.name)}: {tot\_xg} xG"))

clasico\_1112 <- clasico\_1112 %>%

left\_join(clasico\_1112\_xg, by = "[team.name](http://team.name)") %>%

mutate(player\_label = case\_when(

[shot.outcome.name](http://shot.outcome.name) == "Goal" ~ glue::glue("{[player.name](http://player.name)}: {shot.statsbomb\_xg %>% signif(digits = 2)} xG"),

TRUE ~ ""))

**Plot**

There’s several components to this plot. First, there is a timeline  
going across the plot showing the total minutes of the game, this is  
done with geom\_segment() and setting the x and xend to **0** and  
**95** respectively while the y and yend arguments are kept to zero  
as there shouldn’t be any movement along the y-axis. Second, there are  
green segments highlighting when an actual goal was scored in the game,  
done via the geom\_rect() function and passing data where the  
“[shot.outcome.name](http://shot.outcome.name)” variable had the value, **Goal**. I added a small  
two minute buffer on either side of the goal time to create a  
rectangular highlight. Last but certainly not least, are the  
geom\_point()s of different sizes (depending on the value of XG)  
showing the xG events throughout the match.

If you’ve worked with fonts and {ggplot2} before you might think it’s  
weird that I’m calling windowsFonts() below. Normally I wouldn’t, but  
I can’t seem to get the fonts to show up properly when I stitch multiple  
plots together (in a later section) so I had to resort to doing it this  
way. If you want to just create a standalone plot then the  
windowsFonts() code isn’t needed and you can call the font family in  
theme() as you would normally (after doing the {extrafont} stuff at  
the beginning). This is something peculiar with fonts and certain  
Operating Systems and you may experience different problems or none at  
all on your computer.

windowsFonts(robotoc = windowsFont("Roboto Condensed"))

clasico\_xg\_timelineplot <- clasico\_1112 %>%

ggplot() +

geom\_segment(x = 0, xend = 95,

y = 0, yend = 0) +

geom\_rect(data = clasico\_1112 %>% filter([shot.outcome.name](http://shot.outcome.name) == "Goal"),

aes(xmin = minute - 2, xmax = minute + 2,

ymin = -0.005, ymax = 0.005),

alpha = 0.3, fill = "green") +

geom\_label\_repel(data = clasico\_1112 %>% filter([shot.outcome.name](http://shot.outcome.name) == "Goal"),

aes(x = minute, y = 0,

color = [team.name](http://team.name), label = player\_label),

nudge\_x = 4, nudge\_y = 0.003, family = "robotoc",

show.legend = FALSE) +

geom\_point(data = clasico\_1112 %>% filter(shot.statsbomb\_xg != 0),

shape = 21, stroke = 1.5,

aes(x = minute, y = 0,

size = shot.statsbomb\_xg, fill = [team.name](http://team.name))) +

scale\_color\_manual(values = c("Barcelona" = "#a50044",

"Real Madrid" = "black")) +

scale\_fill\_manual(values = c("Barcelona" = "#a50044",

"Real Madrid" = "white")) +

facet\_wrap(vars(team\_label), ncol = 1) +

scale\_x\_continuous(breaks = seq(0, 95, by = 5),

labels = c(seq(0, 40, by = 5), "HT",

seq(50, 90, by = 5), "FT"),

limits = c(-3, 95),

expand = c(0.01, 0)) +

scale\_y\_continuous(limits = c(-0.005, 0.005),

expand = c(0, 0)) +

scale\_size(range = c(2, 6)) +

labs(caption = "By @R\_by\_Ryo") +

theme\_minimal() +

theme(legend.position = "none",

strip.text = element\_text(size = 16, family = "robotoc",

face = "bold", color = "grey20"),

plot.caption = element\_text(family = "robotoc", color = "grey20",

hjust = 0),

axis.title = element\_blank(),

axis.text = element\_blank(),

panel.grid.minor = element\_blank(),

panel.grid.major.y = element\_blank())

clasico\_xg\_timelineplot

Alone and without x-axis labels it doesn’t amount to much but in  
combination with the next two plots it’ll all come together nicely.

**xG Accumulated Plot**

While the previous plot highlights certain xG events throughout the  
match it doesn’t give you a sense of the ebb and flow of the game  
through the lens of xG. This next plot adds up the total xG of teams  
over time which can show periods of dominance and the spread of high/low  
xG shots across a match.

**Data**

Similar to the previous plot except we’re just taking the cumulative sum  
over time using the cumsum() function. We put a lag() on it so that  
both teams start off with 0 xG at minute 0. To help with the labels for  
our plot we left\_join() the same data frame except only for rows where  
there was a goal. Then we create slightly different versions of the  
minutes (“minute\_goal”) and rollsum (“rollsum\_goal”) variables for the  
goals so they line up properly on the plot. For the actual label we use  
glue::glue() to glue together the values of the “[player.name](http://player.name)” variable  
and the “sumxg” variable (only for rows where the shot outcome is equal  
to **Goal**).

clasico\_rollsum <- clasico\_1112 %>%

group\_by(minute, [team.name](http://team.name), period) %>%

summarize(sumxg = sum(shot.statsbomb\_xg)) %>%

ungroup() %>%

group\_by([team.name](http://team.name)) %>%

mutate(rollsum = lag(cumsum(sumxg)),

rollsum = if\_else([is.na](http://is.na)(rollsum), 0, rollsum)) %>%

select([team.name](http://team.name" \t "_blank), minute, rollsum, sumxg) %>%

mutate(rollsum = case\_when(

row\_number() == n() & sumxg != 0 ~ rollsum + sumxg,

TRUE ~ rollsum

))

clasico\_rollsum <- clasico\_rollsum %>%

left\_join(clasico\_1112 %>% filter([shot.outcome.name](http://shot.outcome.name) == "Goal") %>% select(minute, [shot.outcome.name](http://shot.outcome.name), [team.name](http://team.name), [player.name](http://player.name)),

by = c("minute", "[team.name](http://team.name)")) %>%

mutate(rollsum\_goal = rollsum + sumxg,

minute\_goal = minute + 1,

player\_label = case\_when(

[shot.outcome.name](http://shot.outcome.name) == "Goal" ~ glue::glue("{[player.name](http://player.name)}: {sumxg %>% signif(digits = 2)} xG"),

TRUE ~ ""))

glimpse(clasico\_rollsum)

## Observations: 189

## Variables: 9

## Groups: [team.name](http://team.name) [2]

## $ [team.name](http://team.name) "Barcelona", "Real Madrid", "Barcelona", "Re...

## $ minute 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7,...

## $ rollsum 0.00000000, 0.00000000, 0.00000000, 0.585871...

## $ sumxg 0.00000000, 0.58587144, 0.00000000, 0.000000...

## $ [shot.outcome.name](http://shot.outcome.name) NA, "Goal", NA, NA, NA, NA, NA, NA, NA, NA, ...

## $ [player.name](http://player.name) NA, "Karim Benzema", NA, NA, NA, NA, NA, NA,...

## $ rollsum\_goal 0.00000000, 0.58587144, 0.00000000, 0.585871...

## $ minute\_goal 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8,...

## $ player\_label "", "Karim Benzema: 0.59 xG", "", "", "", ""...

**Plot**

If you’re familiar with R, this is a simple line plot. However, there’s  
still a lot of work to be done to make it look nice. By setting the  
breaks, labels, and limits in scale\_x\_continous() we can properly  
label the time along the x-axis. For the y-axis, I use the sec\_axis()  
function to attach labels for each team’s total xG at the end of the  
line, more specifically the y-axis on the opposite side. To save on some  
space we can also move the legend into a more accessible place in the  
plot by specifying the coordinates in the legend.position argument of  
theme(). We can also use the new  
[{ggtext}](https://github.com/clauswilke/ggtext) package to add more  
styling to the text in the labels and titles using CSS/HTML. Using  
geom\_point() and geom\_label\_repel() we can add markers to signify  
when goals were scored along with who the goal scorer was and the xG  
value of the shot.

tot\_clasico\_df <- clasico\_1112\_xg %>%

pull(tot\_xg)

clasico\_rollsumxg\_plot <- clasico\_rollsum %>%

ggplot(aes(x = minute, y = rollsum,

group = [team.name](http://team.name), color = [team.name](http://team.name))) +

geom\_line(size = 2.5) +

geom\_label\_repel(data = clasico\_rollsum %>% filter([shot.outcome.name](http://shot.outcome.name) == "Goal"),

aes(x = minute\_goal, y = rollsum\_goal,

color = [team.name](http://team.name), label = player\_label),

nudge\_x = 6, nudge\_y = 0.15, family = "Roboto Condensed",

show.legend = FALSE) +

geom\_point(data = clasico\_rollsum %>% filter([shot.outcome.name](http://shot.outcome.name) == "Goal"),

aes(x = minute\_goal, y = rollsum\_goal, color = [team.name](http://team.name)), show.legend = FALSE,

size = 5, shape = 21, fill = "white", stroke = 1.25) +

scale\_color\_manual(values = c("Barcelona" = "#a50044",

"Real Madrid" = "#000000"),

labels = c("**Barcelona**",

"**Real Madrid**")) +

scale\_fill\_manual(values = c("Barcelona" = "#a50044",

"Real Madrid" = "#000000")) +

scale\_x\_continuous(breaks = c(seq(0, 90, by = 5), 94),

labels = c(seq(0, 40, by = 5), "HT",

seq(50, 90, by = 5), "FT"),

expand = c(0.01, 0),

limits = c(0, 94)) +

scale\_y\_continuous(sec.axis = sec\_axis(~ ., breaks = tot\_clasico\_df)) +

labs(title = "**Real Madrid: 1 (1st, 40 pts.)**  
 **Barcelona: 3 (2nd, 34 pts.)**",

subtitle = "December 10, 2011 (Matchday 16)",

x = NULL,

y = "Expected Goals") +

theme\_minimal() +

theme(text = element\_text(family = "Roboto Condensed"),

plot.title = element\_markdown(size = 40, family = "Roboto Condensed"),

plot.subtitle = element\_text(size = 18, family = "Roboto Condensed",

color = "grey20"),

axis.title = element\_text(size = 18, color = "grey20"),

axis.text = element\_text(size = 16, face = "bold"),

panel.grid.minor = element\_blank(),

legend.text = element\_markdown(size = 16),

legend.position = c(0.2, 0.95),

legend.direction = "horizontal",

legend.title = element\_blank())

clasico\_rollsumxg\_plot

Real Madrid had higher xG throughout the match (boosted considerably by  
the first goal less than 30 seconds in which had an xG value of 0.54)  
yet it was Barcelona who scored 3 goals from an xG of 0.78 to win the  
game.

**Final Third Passes**

In this plot we look at a rolling sum (using a window of 5 minutes) of  
the passes that were made by each team in the final third of the field.

**Data**

We group\_by() each team and the minute to count the number of events  
that had a value of “Pass” with the condition that they happened in the  
final third of the field (“location.x” >= 80).

roll\_final\_pass <- clasico\_1112 %>%

group\_by([team.name](http://team.name" \t "_blank), minute) %>%

mutate(count = case\_when(

[type.name](http://type.name) == "Pass" & location.x >= 80 ~ 1L,

TRUE ~ 0L

)) %>%

select([team.name](http://team.name" \t "_blank), minute, count) %>%

ungroup()

The main problem here is that not every minute is included in the data  
due to a variety of factors, for this game, there isn’t any “Pass” data  
for the 93rd minute for either team and no pass data for Barcelona in  
the entirety of the 14th minute. So even if we apply our rolling sum  
function it wouldn’t be accurate as it’ll won’t be taking into account  
the rows for those missing minutes. We just need to create another data  
frame that has every combination of the minutes throughout the match for  
each team. I use tidyr::crossing() here but there are other ways to do  
this.

first\_min <- clasico\_1112$minute %>% unique() %>% first()

last\_min <- clasico\_1112$minute %>% unique() %>% last()

minute <- c(first\_min:last\_min)

[team.name](http://team.name) <- c("Real Madrid", "Barcelona")

crossing(minute, [team.name](http://team.name)) %>% slice(26:32)

## # A tibble: 7 x 2

## minute [team.name](http://team.name)

##

## 1 12 Real Madrid

## 2 13 Barcelona

## 3 13 Real Madrid

## 4 14 Barcelona

## 5 14 Real Madrid

## 6 15 Barcelona

## 7 15 Real Madrid

Now there’s a row for the missing minutes as well and now we can take  
this crossed data frame and join it with the passing data frame. Then we  
sum up the number of passes for each minute interval and then apply a  
rolling\_sum() function. This custom function is created using  
tibbletime::rollify(). To use this function, specify an input function  
to be used for the rolling window, in our case sum() and the window to  
be of length 5. In the final line we filter() the data so we only take  
the rows for each 5 minute interval and the last row (the 94th minute).

rolling\_sum <- tibbletime::rollify(.f = sum, window = 5)

roll\_clasico\_pass <- crossing(minute, [team.name](http://team.name)) %>%

left\_join(roll\_final\_pass, by = c("minute", "[team.name](http://team.name)")) %>%

group\_by([team.name](http://team.name" \t "_blank), minute) %>%

summarize\_all(sum) %>%

ungroup() %>%

mutate(count = ifelse([is.na](http://is.na)(count), 0, count)) %>%

group\_by([team.name](http://team.name)) %>%

mutate(rollsum = rolling\_sum(count),

rollsum = ifelse([is.na](http://is.na)(rollsum), 0, rollsum)) %>%

group\_by([team.name](http://team.name)) %>%

select(-count) %>%

filter(row\_number() %% 5 == 1 | row\_number() == n())

roll\_clasico\_pass %>% head(5)

## # A tibble: 5 x 3

## # Groups: [team.name](http://team.name) [1]

## [team.name](http://team.name) minute rollsum

##

## 1 Barcelona 0 0

## 2 Barcelona 5 3

## 3 Barcelona 10 5

## 4 Barcelona 15 1

## 5 Barcelona 20 5

**Plot**

This is similar to the previous plot but with the addition of  
geom\_point() to add markers for the number of final third passes at  
the 5 minute intervals we just created. We change the shape of the  
points to **21** (a hollow circle) so that we can fill the inside with  
the color of each team specified in scale\_fill\_manual(). We also set  
the stroke to **2.5** so that the outline of the circle is a bit  
thicker.

windowsFonts(robotoc = windowsFont("Roboto Condensed"))

finalthird\_rollingplot <- roll\_clasico\_pass %>%

ggplot(aes(x = minute, y = rollsum,

group = [team.name](http://team.name))) +

geom\_line(data = roll\_clasico\_pass,

size = 1.2) +

geom\_point(data = roll\_clasico\_pass,

aes(fill = [team.name](http://team.name)),

size = 3.5, shape = 21, stroke = 2.5) +

scale\_x\_continuous(breaks = seq(0, 95, by = 5),

labels = c(seq(0, 40, by = 5), "HT",

seq(50, 90, by = 5), "FT"),

limits = c(-3, 95),

expand = c(0.01, 0)) +

scale\_y\_continuous(breaks = seq(0, 30, by = 5),

labels = seq(0, 30, by = 5)) +

scale\_fill\_manual(values = c("Barcelona" = "#a50044",

"Real Madrid" = "white"),

labels = c("**Barcelona**",

"**Real Madrid**")) +

labs(title = "**Real Madrid: 1 (1st, 40 pts.)**  
 **Barcelona: 3 (2nd, 34 pts.)**",

subtitle = "December 10, 2011 (Matchday 16)",

x = NULL,

y = "Final Third Passes") +

theme\_minimal() +

theme(text = element\_text(family = "robotoc"),

plot.title = element\_markdown(size = 40, family = "robotoc"),

plot.subtitle = element\_text(size = 18, family = "robotoc",

color = "grey20"),

axis.title = element\_text(size = 18, color = "grey20"),

axis.text = element\_text(size = 16, face = "bold"),

panel.grid.minor = element\_blank(),

legend.text = element\_markdown(size = 14),

legend.position = c(0.25, 0.95),

legend.direction = "horizontal",

legend.title = element\_blank())

finalthird\_rollingplot

As a standalone plot it’s nice as you can see which team was on the  
offensive at different points throughout the game. However, it might be  
even more useful if we can look at this data in combination with some of  
the other plots we created previously which leads us to the next  
section…

**All Together Now!**

You can combine several of the plots we made above to create a nice  
infographic that summarizes the game using this kind of data. I’m sure  
you’ve seen some of these online such as  
[this](https://twitter.com/WomensFootyStat/status/1147930874647896067)  
from [Women’s Footy Stat](https://twitter.com/WomensFootyStat) among  
others. The two packages I normally use are  
[{patchwork}](https://github.com/thomasp85/patchwork/) and  
[{cowplot}](https://github.com/wilkelab/cowplot/) for this kind of job  
but with the {ggtext} formatting as well as how wonky fonts work on  
Windows and R I had to resort to using {grid} and {gtable} to combine  
the plots without the text getting messed up on rendering.

library(gtable)

library(grid)

png(filename = here::here("Lionel Messi/output/clasico\_match\_plot\_RAW.png"),

width = 1000, height = 1600, res = 144, bg = "white")

one <- ggplotGrob(finalthird\_rollingplot)

two <- ggplotGrob(clasico\_xg\_timelineplot)

gg <- rbind(one, two, size = "last")

gg$widths <- unit.pmax(one$widths, two$widths)

grid.newpage()

grid.draw(gg)

dev.off()

## png

## 2

If you don’t want to include the {ggtext} stuff then using  
cowplot::plot\_grid() with the arguments align set to **v** for  
vertical alignment, **h** for horizontal alignment, and axis set to  
**l** for left margin alignment works just fine.

## ...delete all {ggtext} code and resave ggplot objects...

clasico\_match\_plot <- plot\_grid(finalthird\_rollingplot,

clasico\_xg\_timelineplot, ncol = 1,

align = "hv", axis = "l")

ggsave(plot = clasico\_match\_plot,

filename = here::here("Lionel Messi/output/clasico\_match\_plotRAW.png"),

height = 14, width = 10)

Nice! However, we’ve got one last thing to do which is to add the  
StatsBomb logo to our plot as per their user agreement. For this I’ll  
use a special function, add\_logo() (mainly based on the  
[{magick}](https://cran.r-project.org/web/packages/magick/vignettes/intro.html)  
package) created by [Thomas Mock](https://twitter.com/thomas_mock) that  
I always use for appending logos onto plots.

add\_logo <- function(plot\_path, logo\_path, logo\_position, logo\_scale = 10){

# Requires magick R Package <https://github.com/ropensci/magick>

# Useful error message for logo position

if (!logo\_position %in% c("top right", "top left", "bottom right", "bottom left")) {

stop("Error Message: Uh oh! Logo Position not recognized\n Try: logo\_positon = 'top left', 'top right', 'bottom left', or 'bottom right'")

}

# read in raw images

plot <- magick::image\_read(plot\_path)

logo\_raw <- magick::image\_read(logo\_path)

# get dimensions of plot for scaling

plot\_height <- magick::image\_info(plot)$height

plot\_width <- magick::image\_info(plot)$width

# default scale to 1/10th width of plot

# Can change with logo\_scale

logo <- magick::image\_scale(logo\_raw, as.character(plot\_width/logo\_scale))

# Get width of logo

logo\_width <- magick::image\_info(logo)$width

logo\_height <- magick::image\_info(logo)$height

# Set position of logo

# Position starts at 0,0 at top left

# Using 0.01 for 1% - aesthetic padding

if (logo\_position == "top right") {

x\_pos = plot\_width - logo\_width - 0.01 \* plot\_width

y\_pos = 0.01 \* plot\_height

} else if (logo\_position == "top left") {

x\_pos = 0.01 \* plot\_width

y\_pos = 0.01 \* plot\_height

} else if (logo\_position == "bottom right") {

x\_pos = plot\_width - logo\_width - 0.01 \* plot\_width

y\_pos = plot\_height - logo\_height - 0.001 \* plot\_height

} else if (logo\_position == "bottom left") {

x\_pos = 0.01 \* plot\_width

y\_pos = plot\_height - logo\_height - 0.01 \* plot\_height

}

# Compose the actual overlay

magick::image\_composite(plot, logo, offset = paste0("+", x\_pos, "+", y\_pos))

}

We input the finished plot that we just saved as well as the path to the  
StatsBomb logo that I have saved in an “img” folder. We can then set the  
logo\_position and the logo\_scale (relative to the plot) and save it  
using magick::image\_write().

plot\_logo <- add\_logo(

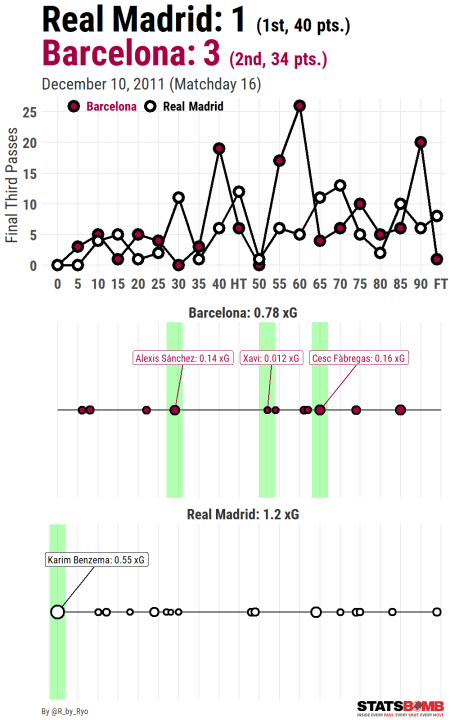
plot\_path = here::here("Lionel Messi/output/clasico\_match\_plot\_RAW.png"),

logo\_path = here::here("img/stats-bomb-logo.png"),

logo\_position = "bottom right",

logo\_scale = 5)

plot\_logo



## Save Plot

magick::image\_write(

image = plot\_logo,

path = here::here("Lionel Messi/output/clasico\_match\_plot\_FIN.png"))

With the plot done, let me give you a bit of context to this game. After  
Real Madrid scored a goal within the first minute, it became a tight  
game with either side not really being able to string many passes in the  
final third. However, Alexis Sanchez was able to score the equalizer  
against the run of play from a Messi through ball around the 30th minute  
([Video of Sanchez’s  
goal](https://www.youtube.com/watch?v=XkoKJ-Ao4i4)), during a period  
where Real Madrid had a lot of final third passes and created several  
chances in quick succession (albeit of low xG values). Barca’s two later  
goals came from sustained pressure of their own in the final third.  
Although Barcelona were able to close the gap on Real Madrid to just 3  
points there was still half a season to go and defeats to Osasuna and  
Real Madrid in the return fixture proved to be their undoing.

With the StatsBomb data available and the plot-stitching R packages  
shown above you can make similar plots or combine any two, three, or  
even four plots to provide an overview of a match or season! You could  
also add in text-only ggplot objects and combine it with the plots to  
make an infographic, the possibilities are endless!

**Pass Partner Plots**

These next few plots explore the passing partnerships between all the  
Barcelona players. Rather than a full pass network graph this is simply  
looking at things from a more micro-level by counting up the frequency  
in which two players exchanged passes with each other. From a  
visualization standpoint there are problems with using a standard bar  
chart due to the long labels needed along the x-axis. One option is to  
put the player names on the y-axis however it’s not always the best to  
do so. Another way is to use **“upset plots”** which visualizes the set  
intersections by a matrix located around the main plot.

I used the [{ggupset}](https://github.com/const-ae/ggupset/) package but  
there are alternatives such as {UpsetR} which provides some additional  
features. The choice is a matter of preference, as for me, I liked how  
seamlessly {ggupset} worked with the existing {ggplot2} API. Before we  
get to plotting we need to manipulate the data we have to that we get  
the right variables to pass to the plotting functions.

**Data: All Passes Received in the Box**

If you check the data you’ll see that the majority of the values in the  
[pass.outcome.name](http://pass.outcome.name) variable are set to NA and you might think that  
there’s a lot of missing data. However, the empty values are all  
actually “Complete” passes. To make this more explicit we can use the  
fct\_explicit\_na() function to set those NAs to “Complete” while also  
turning the variable into a factor.

Following that we filter() for event types that are specifically  
“Pass”-es that have a “Complete” outcome from the team “Barcelona” that  
only come from open play and where the passes end up in the opposition’s  
box. You can find out the exact coordinates to set up the filtering for  
passes into the box (and any other area of the pitch you have in mind)  
by taking a look at page 34 of StatsBomb’s [Open Data Specification  
(version  
1.1)](https://github.com/statsbomb/open-data/blob/master/doc/StatsBomb%20Open%20Data%20Specification%20v1.1.pdf).

From there we select() the variables we want to keep, you can use a  
select helper function such as contains() to grab all variables  
containing the string that you supply, in this case all of the “pass”  
variables, “pass.angle”, “pass.length”, etc.

Then for each season we count the number of passes between a player  
(“[player.name](http://player.name)”) and the recipient of the pass (“[pass.recipient.name](http://pass.recipient.name)”)  
and call this variable “pass\_num”. After making sure we ungroup() we  
edit the “[player.name](http://player.name)” variable so that it includes both the name and  
the number of passes they made (the “pass\_num” variable we just  
created).

Finally, {ggupset} expects the variable that we are creating the plot  
for to be in a list form. So we create a new list variable “pass\_duo”  
whose elements contain the passer’s name (“[player.name](http://player.name)”) and the pass  
recipient’s name (“[pass.recipient.name](http://pass.recipient.name)”).

pass\_received\_all\_box <- messi\_data\_clean %>%

mutate([pass.outcome.name](http://pass.outcome.name" \t "_blank) = fct\_explicit\_na([pass.outcome.name](http://pass.outcome.name), "Complete")) %>%

filter([type.name](http://type.name" \t "_blank) == "Pass",

[team.name](http://team.name) == "Barcelona",

[pass.outcome.name](http://pass.outcome.name) == "Complete",

## Only passes from open play

![play\_pattern.name](http://play_pattern.name) %in% c("From Corner", "From Free Kick",

"From Throw In"),

## Only passes that ended up inside the box:

pass.end\_location.x >= 102 & pass.end\_location.y <= 62 &

pass.end\_location.y >= 18) %>%

select([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name),

season\_id, season\_name,

[position.name](http://position.name), [position.id](http://position.id),

location.x, location.y,

pass.end\_location.x, pass.end\_location.y,

contains("pass")) %>%

group\_by(season\_name) %>%

add\_count([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name), name = "pass\_num") %>%

ungroup() %>%

mutate([player.name](http://player.name" \t "_blank) = glue::glue("{[player.name](http://player.name)}: {pass\_num}")) %>%

mutate(pass\_duo = map2([player.name](http://player.name), [pass.recipient.name](http://pass.recipient.name), ~c(.x, .y))) %>%

select([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name), pass\_num,

season\_name, pass\_duo)

Now we can get to the actual plotting!

As we have data for multiple seasons, instead of repeating the {ggplot2}  
code for every year we can create a “base plot” for every season and  
store it inside the data frame via nesting. To do the nesting, you need  
to group\_by() the season and then call nest(). As you can see below  
this creates a column called “data” which holds all the variables and  
values from each of the seasons listed in “season\_name”.

pass\_received\_all\_box %>%

group\_by(season\_name) %>%

nest()

## # A tibble: 8 x 2

## season\_name data

## >

## 1 2004/2005 [46 x 4]

## 2 2005/2006 [96 x 4]

## 3 2006/2007 [127 x 4]

## 4 2007/2008 [152 x 4]

## 5 2008/2009 [249 x 4]

## 6 2009/2010 [267 x 4]

## 7 2010/2011 [296 x 4]

## 8 2011/2012 [253 x 4]

With the way the data frame is set up now, you can use mutate() to  
create a new variable column containing the plots for each season! If  
you want to do this, especially if you also want to programatically add  
in the season name to each of the plots you need to use the  
purrr::map2() function. By passing the “data” and “season\_name”  
variables to the function we can ensure that they can be used in the  
code to create the plots. Here we’re passing the “data” as vector .x  
and the “season\_name” as vector .y, these notations are the ones  
we’ll use to refer to these variables inside the actual {ggplot2}  
function call itself.

Using the ~ to denote that the following code is the function we want  
to use, we start building out plot. As can be seen the first argument  
“data” is set to the .x argument that we set before as the data for a  
specific season. The main code for the upset pot comes in  
scale\_x\_upset() where you can set the number of intersections to plot,  
in this case **10** for ten different passer-pass receiver pairings.  
Within theme\_combmatrix() you can set the usual theme elements for a  
plot as well as upset matrix specific aspects such as the line and  
point’s color and size as well as spacing for the text.

all\_pass\_nested\_box <- pass\_received\_all\_box %>%

group\_by(season\_name) %>%

nest() %>%

mutate(plot = map2(

.x = data, .y = season\_name,

~ ggplot(data = .x, aes(x = pass\_duo)) +

geom\_bar(fill = "#a70042") +

scale\_x\_upset(n\_intersections = 10,

expand = c(0.01, 0.01)) +

scale\_y\_continuous(expand = c(0.04, 0.04)) +

labs(title = glue::glue("

Total Completed Passes Into The Box

Between All Players ({.y})"),

subtitle = "'Name: Number' = Passer, 'No Number' = Pass Receiver",

x = NULL, y = "Number of Passes") +

theme\_combmatrix(

text = element\_text(family = "Roboto Condensed",

color = "#004c99"),

plot.title = element\_text(family = "Roboto Condensed", size = 20,

color = "#a70042"),

plot.subtitle = element\_text(family = "Roboto Condensed", size = 16,

color = "#004c99"),

axis.title = element\_text(family = "Roboto Condensed", size = 14,

color = "#004c99"),

axis.text.x = element\_text(family = "Roboto Condensed", size = 12,

color = "#004c99"),

axis.text.y = element\_text(family = "Roboto Condensed", size = 12,

color = "#004c99"),

panel.background = element\_rect(fill = "white"),

combmatrix.panel.point.size = 4,

combmatrix.panel.point.color.fill = "#a70042",

combmatrix.panel.line.color = "#a70042",

panel.grid = element\_line(color = "black"),

panel.grid.major.x = element\_blank(),

axis.ticks = element\_blank())))

glimpse(all\_pass\_nested\_box)

## Observations: 8

## Variables: 3

## $ season\_name "2004/2005", "2005/2006", "2006/2007", "2007/2008"...

## $ data > Ronaldinho: 5 , Ronaldinho: 5 ...

## $ plot [

Now you can check out the 8th element of the “plot” variable which  
corresponds to the 2011/2012 season:

all\_pass\_nested\_1112 <- all\_pass\_nested\_box$plot[[8]] +

scale\_y\_continuous(labels = seq(0, 15, by = 5),

breaks = seq(0, 15, by = 5),

limits = c(0, 15))

ggsave(plot = all\_pass\_nested\_1112,

filename = here::here("Lionel Messi/output/allpass\_1112\_plotRAW.png"),

height = 6, width = 8)

plot\_logo <- add\_logo(

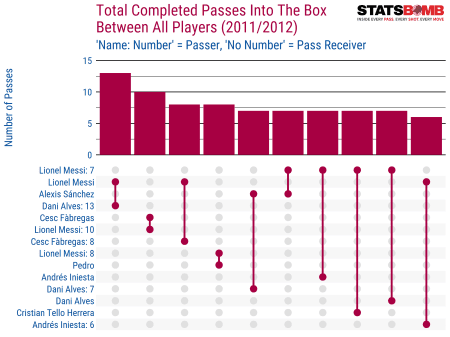
plot\_path = here::here("Lionel Messi/output/allpass\_1112\_plotRAW.png"),

logo\_path = here::here("img/stats-bomb-logo.png"),

logo\_position = "top right",

logo\_scale = 5)

plot\_logo



## Save Plot

magick::image\_write(

image = plot\_logo,

path = here::here("Lionel Messi/output/allpass\_1112\_plotFIN.png"))

You can add in whatever other {ggplot2} functions in as needed but this  
way you don’t have to type out the entire code block for each season;  
you can just tweak and adjust the “base plot” we created in the “plot”  
variable of the nested data frame!

As can be seen above a little more work needs to be done concerning the  
axis-labels. Although Messi is labelled having made 7 passes, they are 7  
passes **EACH** to Alexis Sanchez, Iniesta, Cristian Tello, and Dani  
Alves so the total really should read as 28.

**Data: Shot Assists**

Just to show another example let’s look at shot assists instead. Besides  
the differences inside filter() the code is the same (minus label and  
title parts too of course):

## Data

messi\_all\_shot\_assist <- messi\_data\_clean %>%

mutate([pass.outcome.name](http://pass.outcome.name" \t "_blank) = fct\_explicit\_na([pass.outcome.name](http://pass.outcome.name), "Complete")) %>%

filter([team.name](http://team.name" \t "_blank) == "Barcelona",

![is.na](http://is.na)(pass.shot\_assist),

![play\_pattern.name](http://play_pattern.name) %in% c("From Corner", "From Free Kick",

"From Throw In")) %>%

select([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name),

season\_id, season\_name,

[position.name](http://position.name), [position.id](http://position.id),

location.x, location.y,

pass.end\_location.x, pass.end\_location.y,

contains("pass")) %>%

group\_by(season\_name) %>%

add\_count([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name), name = "pass\_num") %>%

ungroup() %>%

mutate([player.name](http://player.name" \t "_blank) = glue::glue("{[player.name](http://player.name)}: {pass\_num}")) %>%

mutate(pass\_duo = map2([player.name](http://player.name), [pass.recipient.name](http://pass.recipient.name), ~c(.x, .y))) %>%

select([player.name](http://player.name" \t "_blank), [pass.recipient.name](http://pass.recipient.name), pass\_num,

season\_name, pass\_duo)

## Nest plots

messi\_nested\_all\_shot\_assist <- messi\_all\_shot\_assist %>%

group\_by(season\_name) %>%

nest() %>%

mutate(plot = map2(

data, season\_name,

~ ggplot(data = .x, aes(x = pass\_duo)) +

geom\_bar(fill = "#a70042") +

scale\_x\_upset(n\_intersections = 10,

expand = c(0.01, 0.01)) +

scale\_y\_continuous(expand = c(0.04, 0.04)) +

labs(title = glue::glue("Shot Assists ({.y})"),

subtitle = "'Name: Number' = Passer, 'No Number' = Pass Receiver",

caption = "Source: StatsBomb",

x = NULL, y = "Number of Passes") +

theme\_combmatrix(

text = element\_text(family = "Roboto Condensed",

color = "#004c99"),

plot.title = element\_text(family = "Roboto Condensed", size = 20,

color = "#a70042"),

plot.subtitle = element\_text(family = "Roboto Condensed", size = 16,

color = "#004c99"),

axis.title = element\_text(family = "Roboto Condensed", size = 14,

color = "#004c99"),

axis.text.x = element\_text(family = "Roboto Condensed", size = 12,

color = "#004c99"),

axis.text.y = element\_text(family = "Roboto Condensed", size = 12,

color = "#004c99"),

panel.background = element\_rect(fill = "white"),

combmatrix.panel.point.size = 4,

combmatrix.panel.point.color.fill = "#a70042",

combmatrix.panel.line.color = "#a70042",

panel.grid = element\_line(color = "black"),

panel.grid.major.x = element\_blank(),

axis.ticks = element\_blank())))

## Plot 2011/2012

messi\_nested\_all\_shot\_assist$plot[[8]] +

scale\_y\_continuous(labels = seq(0, 12, by = 2),

breaks = seq(0, 12, by = 2),

limits = c(0, 12))

It might be a good idea to combine these plots with other visualizations  
such as a pass frequency table and/or a pass network map. For looking at  
completed passes into the box you could create some pass maps  
highlighting Zone 14 or the Half-Spaces like the ones [Between the  
Posts](https://betweentheposts.net/match-plots/) create for their match  
reports. For the shot assists plot above we might also want to put it  
side-by-side with an xG plot to show whose passes created high xG value  
chances.

I only used upset plots for two different elements (the passer and the  
pass receiver) but the advantages of this visualization method becomes  
more pronounced with even more set intersections so there remains more  
room for applying these to other types of soccer viz. For example you  
could extend this to look at the most frequent passing sequences between  
3 players or 4 or even 5. From the first example one of the top passing  
sequences between 3 players might be something like Victor Valdes –  
Busquets – Xavi/Iniesta. The matrix underneath the plot may become a bit  
unwieldy without some filtering and tweaking by setting different values  
for n\_intersections, n\_sets, and others in scale\_x\_upset().

**Conclusion**

In this blog post I went over some simple plots (dot, line, and bar  
charts) you can do using {ggplot2} with the free StatsBomb data. There’s  
plenty more to do with this data and I’m still experimenting and  
learning everyday. A good way to practice is to take something someone  
has done and then recreate that visualization by applying it to a  
slightly different data set with your favorite programming language.  
This is exactly how I learned to do things; I’ll see something on  
Twitter and try to remake that viz using Liverpool data or J-League data  
instead!

Some people you may want to follow for inspiration:

* [Eliot McKinley](https://twitter.com/etmckinley/) who’s done stuff  
  like [Pass Sonars](https://github.com/etmckinley/PassSonar) and  
  [Action Density  
  Plots](https://github.com/etmckinley/Action-Density), and [Freeze  
  Frame Plots](https://github.com/etmckinley/Freeze-Frame)
* [FC\_rstats](https://twitter.com/FC_rstats) who has a lot of  
  tutorials available on  
  [Github](https://github.com/FCrSTATS/Visualisations)
* [Between the Posts’ Match Summary  
  Plots](https://betweentheposts.net/match-plots/)
* And many many more…

If I can refine/improve upon any of the above then I’ll show my own  
version of these in a future part. Anything new whether its a standalone viz or a full blog post will be linked with the code [here](https://github.com/Ryo-N7/soccer_ggplots) on my **soccer\_ggplot** GitHub  
repository!

If you want to iterate this process over many players/teams/seasons  
there are ways to do so using the purrr::map() family of functions or  
for loops along with the nest() approach I used for the pass partner  
plots. You might also be interested in creating automated parameterized  
reports with RMarkdown using this data, some resources include [“The  
lazy and easily distracted report writer” by Mike Smith at  
RStudioConf::2019](https://github.com/MikeKSmith/RStudioConf2019) and  
[Chapter 15: Parameterized reports of YiHui Xi’s RMarkdown: The  
Definitive  
Guide](https://bookdown.org/yihui/rmarkdown/parameterized-reports.html).  
You may also want to have a unified theme to all your plots, I talk  
about creating your own {ggplot2} themes  
[here](https://ryo-n7.github.io/2019-05-16-introducing-tvthemes-package/)  
and there’s other great resources like  
[this](https://rfortherestofus.com/2019/08/themes-to-improve-your-ggplot-figures/) that you might want to read.

Part 2 will be more xG plots and also on plotting out the data on soccer  
pitches using packages like  
[{ggsoccer}](https://github.com/Torvaney/ggsoccer),  
[{SBpitch}](https://github.com/FCrSTATS/SBpitch),  
[{soccermatics}](https://github.com/JoGall/soccermatics), and more!