An Analysis of Pass Type Effectiveness of Italy and England in Euro 2020

Z Martinez and Maya McArthur

2022-07-30

Introduction

Data is undoubtedly present within all aspects of life, sports being no exception. The emergence and rapid growth of the data science industry has given rise to opportunities to leverage data to produce compelling results that highlight otherwise inconspicuous gaps and patterns. In the sports world, where stats and metrics are king (queen?), data science has the potential to truly change the game with new methods of analyzing player performance, team strategies, scouting decisions, outcome predictions, and even player health.

Although opportunities for analysis are plentiful in sports, one cannot run analyses without adequate data. Many companies have worked, and continue to work, to collect and provide detailed data to propel sports analysis, one being StatsBomb, "a UK-based football analytics and data visualization company" with an "ongoing commitment to developing football analysts of the future" by "provid[ing] the education, tools, and resources needed to learn the trade" (Arastey 2019; StatsBomb 2021). In addition to releasing UEFA Euro 2020 data, StatsBomb has released StatsBomb 360 data, which "captures a freeze-frame showing the location of all players in the frame for every event we collect" (StatsBomb 2021).

In this notebook, we utilize UEFA Euro 2020 data and StatsBomb 360 data provided by StatsBomb to analyze to what extent successful passes within the penalty box contributed to the success of the top two Euro 2020 teams: Italy and England.

The metric we use to measure the success of a cross pass is whether or not the receiver is a teammate of the passer. We employ a two-pronged approach to address this research question. First, we analyze the differences in effect on success based on these passes in a Sankey diagram. Second, we map out and compare successful cross passes within the penalty box for Italy and England, comparing their performance in their first round games compared to the championship game.

Required Packages and Defined Functions

Packages

- **StatsBombR**: package for working with and manipulating StatsBomb data in R.
- **tidyverse**: a compilation of core R packages including dplyr, ggplot2, tidyr, readr, purrr, tibble, stringr, and forcats.
- **ggsankey** : package for making sankey, alluvial, and sankey bump plots.
- **devtools**: package for installing ggsankey and StatsBombR.
- **ggpubr**: package for further customization of plots created using ggplot2.
- **dplyr** :
- **ggplot2** :

Defined Functions

- `plot_id()`: function that returns soccer field map layout with player positions based on inputted `id`.
- `extra()`: function that adjusts plot axes, margins, legends, font sizes, and colors.

```r
# FUNCTION FOR PLOTTING BY ID
plot_id <- function(df, identification, main){

chart = df %>%
filter(id==identification) %>%
mutate(Player_Type_Key = case_when(
    actor==TRUE & receiver == FALSE ~ "Actor",
    teammate==TRUE & receiver==FALSE ~ "Teammate",
    teammate==FALSE & keeper==FALSE & receiver==FALSE ~ "Opponent",
    keeper==TRUE & teammate==FALSE & receiver==FALSE ~ "Goalkeeper",
    teammate== TRUE & receiver== TRUE ~ "Receiver",
    keeper==TRUE & teammate==FALSE & receiver==TRUE ~ "Oppossing Receiver"))

passer.x = df %>% filter(actor == TRUE & id == identification) %>% pull(ff_location.x)
passer.y = df %>% filter(actor == TRUE & id == identification) %>% pull(ff_location.y)
receiver.x = df %>% filter(receiver == TRUE & id == identification) %>% pull(ff_location.x)
receiver.y = df %>% filter(receiver == TRUE & id == identification) %>% pull(ff_location.y)

ggplot() +
annotate("rect",xmin = 0, xmax = 120, ymin = 0, ymax = 80, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 0, xmax = 60, ymin = 0, ymax = 80, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 18, xmax = 0, ymin = 18, ymax = 62, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 102, xmax = 120, ymin = 18, ymax = 62, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 0, xmax = 6, ymin = 30, ymax = 50, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 120, xmax = 114, ymin = 30, ymax = 50, fill = NA, colour = "black", size = 0.6) +
annotate("rect",xmin = 120, xmax = 120.5, ymin =36, ymax = 44, fill = NA, colour = "black", size = 0.6)
annotate("rect",xmin = 0, xmax = -0.5, ymin =36, ymax = 44, fill = NA, colour = "black", size = 0.6) +
annotate("segment", x = 60, xend = 60, y = -0.5, yend = 80.5, colour = "black", size = 0.6)+
annotate("segment", x = 0, xend = 0, y = 0, yend = 80, colour = "black", size = 0.6)+
annotate("segment", x = 120, xend = 120, y = 0, yend = 80, colour = "black", size = 0.6)+
theme(rect = element_blank(),
line = element_blank()) +
# add penalty spot right
annotate("point", x = 108 , y = 40, colour = "black", size = 1.05) +
annotate("path", colour = "black", size = 0.6,
x=60+10*cos(seq(0,2*pi,length.out=2000)),
y=40+10*sin(seq(0,2*pi,length.out=2000)))+
# add centre spot
annotate("point", x = 60 , y = 40, colour = "black", size = 1.05) +
annotate("path", x=12+10*cos(seq(-0.3*pi,0.3*pi,length.out=30)), size = 0.6,
y=40+10*sin(seq(-0.3*pi,0.3*pi,length.out=30)), col="black") +
annotate("path", x=107.84-10*cos(seq(-0.3*pi,0.3*pi,length.out=30)), size = 0.6,
y=40-10*sin(seq(-0.3*pi,0.3*pi,length.out=30)), col="black") +
geom_point(data = chart, aes(x = ff_location.x, y = ff_location.y, fill=factor(Player_Type_Key)),
size = 6, alpha = 0.9, shape=21) +
coord_flip(xlim = c(80, 125)) +
labs(title = main) +
annotate("segment", x = passer.x, xend = receiver.x,
                        y = passer.y, yend = receiver.y,
                        colour = "black", size = 0.6, alpha = 0.7) +
scale_fill_discrete(name = "Key")
}
```

```
# FUNCTION FOR ADDITIONAL PLOT MODIFICATIONS
extra <- function(){
theme(axis.text.x=element_blank(),
axis.title.x = element_blank(),
axis.title.y = element_blank(),
axis.text.y=element_blank(),
plot.title = element_text(size = 14, colour = "black", hjust = 0.5),
legend.direction = "horizontal",
legend.title=element_text(size=12),
legend.text=element_text(size=10),
legend.margin=margin(3,3,3,3),
legend.box.margin=margin(5,5,5,5),
axis.ticks=element_blank(),
plot.background = element_rect(fill = "white"),
strip.text.x = element_text(size=13))
}
```

Checking the Data

We pulled in competition data to `Comp` using `FreeCometitions()` and matches from `FreeMatches(Comp)`. We also pulled in 360 data using `StatsBombFree360Events()` and event data using `StatsBombFreeEvents()`. After cleaning `events` with `allclean()` and including the opposing team with `get.opposingteam()`, we joined `events` and `data360`, filtering for `type.name == "Pass"` or `type.name == "Ball Receipt*"` to create `ffs`.

```
Comp <- FreeCompetitions()
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

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

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

```
Matches = Matches %>% filter(competition.competition_name=="UEFA Euro")
data360 <- StatsBombFree360Events(MatchesDF = Matches, Parallel = T)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

```
## Warning in if (MatchesDF == "ALL") {: the condition has length > 1 and only the
## first element will be used
```

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

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wi
```

```
## Warning in if (MatchesDF == "ALL") {: the condition has length > 1 and only the
## first element will be used
```

```
events <- allclean(events)
events <- get.opposingteam(events)
```

```
data360 = data360 %>% rename(id = event_uuid)

events = events %>% left_join(data360, by = c("id" = "id"))

events = events %>% rename(match_id = match_id.x) %>% select(-match_id.y)
```

```
ffs = events %>%
group_by(team.name) %>%
```

```
filter(type.name=="Pass" | type.name == "Ball Receipt*") %>%
select(id, match_id, team.name, OpposingTeam, player.name, type.name, minute, second, location.x, locati
```

```
ffs = ffs %>% unnest(freeze_frame) %>%
mutate(ff_location.x = (map(location, 1)), ff_location.y = (map(location, 2))) %>%
select(-location) %>%
mutate(ff_location.x = as.numeric(ifelse(ff_location.x == "NULL", NA, ff_location.x)), ff_location.y = a
```

Our key variables of interest are as follows: `id`, `match_id`, `team.name`, `OpposingTeam`, `player.name`, `type.name`, `minute`, `second`, `location.x`, `location.y`, `pass.end_location.x`, `pass.end_location.y`, `pass.type.name`, `pass.cross`, and `freeze_frame`.

Tidying the Data

Prong One: Preparing Data for Sankey Diagram For the Sankey diagram we will need: **ggsankey**, **ggplot**, and **dplyr** packages. For the diagram, we will be looking between the two finalists, Italy and England, to see the flow between `type.name` and `pass.type` that were made that lead to successful passes throughout the tournament.

- We add an additional filter to the events data frame and extract ball receipt. This variable will tell us who on the team received the ball
- We un-nest the freeze frames as we did for the crosses above
- From `ffs1` we create a new data frame called `ffs2` with a new variable titled receiver. We are able to do this using the mutate function and making two cases. We set it to **TRUE** if `type.name` variable is ___"Ball Receipt*"___ and **FALSE** if otherwise.
- We modify code from the `crosses` data manipulation. Instead of filtering for a specific `match_id`, we filter for when `team.name` is Italy or England. *Note: this chart does not include successful passes for when these teams are on defense.*
- In the same code we add a new variable called `successful.passes` using the same method we did when we created `receiver`. We denoted a successful pass as when `teammate` is **TRUE** and `receiver` is **TRUE**, and a successful pass is **FALSE** otherwise.
- On the first run of the diagram, we noticed a 2 sections that was gray and did not have any nodes connected to it. We deduced that this resulted from the `type.name` being **Pass**. For some reason, the line of code for `chart2` would not work if added to the pipeline in creating `chart1`. We filtered out the **N/A** values in `pass.type.name` or in `successful.passes` to get a clearer picture. ** We create a new data frame, `d_f`, using `make_long()`. This allows for the data set we want to use, `chart2`, to be converted to a 'wide' data frame that `ggsankey()` will understand.

```
ffs2 = events %>%
group_by(team.name) %>%
filter(type.name=="Pass"| type.name=="Ball Receipt*")%>%
select(id, match_id, team.name, OpposingTeam, player.name, type.name, minute, second, location.x, locati
```

```
ffs2 = ffs2 %>% unnest(freeze_frame) %>%
mutate(ff_location.x = (map(location, 1)), ff_location.y = (map(location, 2))) %>%
select(-location) %>%
mutate(ff_location.x = as.numeric(ifelse(ff_location.x == "NULL", NA, ff_location.x)), ff_location.y = a
```

```
ffs2 = ffs2 %>%
  mutate(receiver = case_when(type.name=="Ball Receipt*"~"TRUE",
                              type.name!= "Ball Receipt*"~"FALSE"))
```

```
#Italy vs England
chart1 = ffs2 %>%
filter(team.name=="Italy"|team.name=="England") %>%
mutate(Player_Type_Key = case_when(actor==TRUE & teammate==TRUE ~ "Actor",
```

```
teammate==TRUE ~ "Teammate",
teammate == FALSE & receiver== TRUE ~ "Receiver",
teammate==FALSE & keeper==FALSE ~ "Opponent",
keeper==TRUE & teammate==FALSE ~ "Goalkeeper"))%>%
  mutate(successful.pass = case_when(teammate==TRUE & receiver== TRUE ~"TRUE",
         teammate==FALSE & receiver==TRUE~ "FALSE"))
```

```
chart2 = chart1%>%
  filter(!is.na(pass.type.name)|!is.na(successful.pass))
```

```
df <- chart2 %>%
  make_long(team.name,type.name,pass.type.name, successful.pass)
```

Prong Two: Preparing Data for Mapping Successful Cross Passes

- To set the foundation for measuring and mapping the receiver, in addition to the already existing actor, teammate, and keeper variables, we created a new Boolean variable: `receiver.` To do this, we added a vector to `ffs` that presented **TRUE** if `type.name == "Ball Receipt*"` and **FALSE** otherwise.
- Next, we filtered for cross passes where `pass.cross == TRUE` in one data frame and where `receiver == TRUE` in another data frame; both data frames filtered for `team.name == "Italy"` or `team.name == "England"`. Then, we combined the two data frames using `rbind()` to create `ffsITA` and `ffsENG`, respectively. Finally, we filtered for passes within the penalty box using `pass.end_location.x` and `pass.end_location.y`.
- In `ffsITA` and `ffsENG`, we identified rows where `pass.cross == FALSE`, excluding the first instance where `type.name == "Ball Receipt*"` following a row where `pass.cross == TRUE`. *(Although we filtered for `pass.cross == TRUE` in the previous step, `pass.cross == TRUE` and `receiver == TRUE` cannot both exist in the same row due to the structure of the dataset. Therefore, we must once again manually filter out rows where `pass.cross == FALSE` except for the **first one** following a `pass.cross == TRUE`.)*
- Once we had a data frame containing `pass.cross == TRUE` and the first subsequent row where `type.name == "Ball Receipt*"`, we manually adjusted `id` for each row where `type.name == "Ball Receipt*"` to match its preceding `id` where `pass.cross == TRUE`. By doing this, we assigned the `actor`, `receiver`, `keeper`, and `teammates` to the same `id`, allowing for easier mapping with ggplot2.

```
# adding receiver boolean column
ffs$receiver <- ffs$type.name == "Ball Receipt*"
ffs <- ffs %>%
  select(-actor, -keeper, -teammate, -receiver, everything())

# replacing NAs w 0's in pass.end_location x and y for ball receipts
ffs <- ffs %>% mutate_at(c(11,12), ~replace_na(.,0))
```

```
# filtering for crosses with Italy on offense
ffs_crossITA <- ffs %>% filter(pass.cross == TRUE & team.name == "Italy")
# filtering for Italy receivers who are also actors
ffs_receiptITA <- ffs %>% filter(receiver == TRUE & actor == TRUE & team.name == "Italy")
# adding both and sorting
ffsITA <- rbind(ffs_crossITA, ffs_receiptITA) %>% arrange(match_id, minute, second)
# more filtering
ffsITA <- ffsITA %>% filter(pass.end_location.x == 0 |
               pass.end_location.x>102 &
               pass.end_location.y == 0 |
               pass.end_location.y>18 &
               pass.end_location.y<62) %>%
                     filter(is.na(pass.type.name) |
```

```
                    pass.type.name=="Recovery" |
                    pass.type.name=="Interception")
# rearranging columns
ffsITA %>%
  select(pass.cross, player.name, type.name, minute, second, teammate, actor, receiver, everything())
```

```
## # A tibble: 4,630 x 20
## # Groups:   team.name [1]
##    pass.cross player.name  type.name minute second teammate actor receiver id
##    <lgl>      <chr>        <chr>      <int>  <int> <lgl>    <lgl> <lgl>    <chr>
##  1 NA         Giorgio Chi~ Ball Rec~      0     14 TRUE     TRUE  TRUE     22dc~
##  2 NA         Gianluigi D~ Ball Rec~      0     18 TRUE     TRUE  TRUE     1584~
##  3 NA         Leonardo Bo~ Ball Rec~      0     21 TRUE     TRUE  TRUE     91af~
##  4 NA         Giorgio Chi~ Ball Rec~      0     26 TRUE     TRUE  TRUE     085d~
##  5 NA         Leonardo Bo~ Ball Rec~      0     29 TRUE     TRUE  TRUE     b7d6~
##  6 NA         Giorgio Chi~ Ball Rec~      0     33 TRUE     TRUE  TRUE     cb86~
##  7 NA         Lorenzo Ins~ Ball Rec~      0     38 FALSE    TRUE  TRUE     619e~
##  8 NA         Ciro Immobi~ Ball Rec~      0     58 TRUE     TRUE  TRUE     3c2a~
##  9 NA         Lorenzo Ins~ Ball Rec~      0     58 TRUE     TRUE  TRUE     315a~
## 10 NA         Gianluigi D~ Ball Rec~      1     34 TRUE     TRUE  TRUE     a003~
## # ... with 4,620 more rows, and 11 more variables: match_id <int>,
## #   team.name <chr>, OpposingTeam <chr>, location.x <dbl>, location.y <dbl>,
## #   pass.end_location.x <dbl>, pass.end_location.y <dbl>, pass.type.name <chr>,
## #   ff_location.x <dbl>, ff_location.y <dbl>, keeper <lgl>
```

```
# deleting rows not following a pass.cross == TRUE
ffsITA <- ffsITA[-c(1:19, 30, 32:121, 135:176, 195:272, 308, 310:312, 327:482,
          494:506, 519:545, 558:765, 783:805, 817:826, 840:891,
          908:946, 959:1027, 1036, 1038:1324, 1337:1364, 1381:1466,
          1502:1569, 1587:1606, 1626, 1628:1702, 1714:1715,
          1717:1757, 1772:1834, 1847, 1849:2053, 2067:2193, 2211:2283,
          2299, 2301:2354, 2370:2426,
          2443:2472, 2483:2554, 2569:2584, 2597, 2599:2605, 2623:2698,
          2712:2776, 2808, 2810:3165, 3179, 3181:3214, 3228:3747,
          3759:3862, 3774:3967, 3983:3985, 4001:4064, 4075:4124,
          4144:4165, 4182:4425, 4440, 4442:4455, 4469:4497, 4511:4531,
          4543:4630),]
```

```
# creating and adding vector of 1s
rep <- rep(1, times = 685)
ffsITA$rep <- rep
# changing vector of 1s to 0s where pass.cross == TRUE
ffsITA <- within(ffsITA, {
    f <- rep == 1 & pass.cross == TRUE
    rep[f] <- 0
})
```

```
# changing_id to combine ball receipts and cross pass id's
ffsITA[11,1] <- "f0b58e23-903e-4db8-ab7a-03d4831e3e69"
ffsITA[24,1] <- "ace0d420-070d-422d-b1c4-fa1dc9211bdd"
ffsITA[42,1] <- "acf030c6-ffc6-45ce-8108-ebd5d3fef333"
ffsITA[58,1] <- "b4fedd6c-9806-4646-bd66-0a5c0b16df6e"
ffsITA[78,1] <- "f6558a1c-d627-417b-835f-091578e99265"
ffsITA[92,1] <- "dfdeb894-a158-4ce1-8fa8-ad4284c5fc83"
ffsITA[103,1] <- "7be82117-b55f-4e97-a38d-14b61b9b7e94"
```

```
ffsITA[115,1] <- "8dd1f848-c9e0-49b8-a978-8226579af228"
ffsITA[127,1] <- "c9710113-d67c-49b5-b6ae-321dab290e31"
ffsITA[144,1] <- "c3946adc-d2ec-43c0-99c6-fdeafed46a9c"
ffsITA[155,1] <- "599eb33a-2cf0-43b9-8d75-2be49dcbd9d6"
ffsITA[168,1] <- "a238ecf4-88fb-4f61-afff-36b80108ab87"
ffsITA[184,1] <- "62ae7ead-4455-4622-9857-4466331e6140"
ffsITA[196,1] <- "5515ab29-645c-4760-aab9-bdd679aa2add"
ffsITA[205,1] <- "c2e7e32d-9dfb-47da-99f9-793040a79a70"
ffsITA[217,1] <- "cb7da4ac-733c-4444-983e-bd6c70de20ae"
ffsITA[233,1] <- "7d5f132d-f953-4cbb-8794-5fdaed616253"
ffsITA[248,1] <- "bc5d7baa-4e88-4b30-929c-7c39c89cc76a"
ffsITA[268,1] <- "b34ab63e-cf05-4927-b03d-3098887e3525"
ffsITA[285,1] <- "a4eecd95-6d6c-4dc3-ad6b-b004bf650beb"
ffsITA[317,1] <- "be99a45b-6483-4fa7-a791-5c268117bdc3"
ffsITA[331,1] <- "308d2357-cc7c-44f0-9bd3-cdda72ccf6f2"
ffsITA[344,1] <- "9d6931e7-a315-4652-87b7-f3cc7c2a0760"
ffsITA[357,1] <- "3f6252b5-db6c-4bef-b2f7-bee439e8e819"
ffsITA[374,1] <- "65146a02-f239-4597-b857-d7e104dc8390"
ffsITA[390,1] <- "6df67f44-501a-43c6-bac9-9659fb089a34"
ffsITA[405,1] <- "13d1ae0e-9121-4cae-9f91-51bab099dc8c"
ffsITA[421,1] <- "b4ff75e5-88d6-4cc0-88fa-e354dcafb9e4"
ffsITA[431,1] <- "d6942214-6398-41c2-9d96-30f08c80303b"
ffsITA[445,1] <- "6e2ec02c-3870-4ae4-9d21-5ff091df72c0"
ffsITA[458,1] <- "0c19561d-4fbe-4929-baa6-cc7f0d4efbb8"
ffsITA[475,1] <- "d4633b2b-34ba-4b57-a80f-0908ed03cdb4"
ffsITA[488,1] <- "4d722c2e-5653-4058-a490-7768da557b6c"
ffsITA[520,1] <- "9c15cc1a-5a1a-49c7-938f-baa0c7edcf81"
ffsITA[534,1] <- "08d25b7d-f774-4f69-9064-3b8db22d7d06"
ffsITA[547,1] <- "d1d2a72d-c4c9-4ef8-b6d6-6d99636922df"
ffsITA[558,1] <- "d4d6fec7-275f-4206-bb48-d36624f3bd23"
ffsITA[573,1] <- "c83b0fe3-58ba-4d3a-ae93-e797775f3b2e"
ffsITA[588,1] <- "8eaf1e40-6bce-4caf-a8e3-ac332453e5f4"
ffsITA[598,1] <- "9b5c2212-c4dc-4793-858c-41e989f128fe"
ffsITA[617,1] <- "6874f5b6-bbe4-4738-acb8-8abbfbb95a60"
ffsITA[633,1] <- "07e146f4-543f-4458-897b-176717d778ae"
ffsITA[648,1] <- "c4bfe891-56a4-4533-a172-6e0ebc3c0365"
ffsITA[661,1] <- "d337e9f5-d404-4975-be9d-d23cdf1df2ac"
ffsITA[674,1] <- "65fce300-6f9e-4ce5-99f4-2fc35c576822"
ffsITA[685,1] <- "ef124fe1-7b74-4c75-a8cd-c72351fbfbc5"
```

```
# cutting out unsuccessful cross passes (receiver is opponent)
ffsITA_successful <- ffsITA[-c(25:42, 116:127, 286:305, 358:374, 391:405,
                    446:475, 489:534, 548:573, 589:598, 618:685),]
ffsITA_UNsuccessful <- ffsITA[c(25:42, 116:127, 286:305, 358:374, 391:405,
                    446:475, 506:534, 548:573, 589:598, 618:685),]
```

```
# filtering for crosses with Italy on offense
ffs_crossENG <- ffs %>% filter(pass.cross == TRUE & team.name == "England")
# filtering for Italy receivers who are also actors
ffs_receiptENG <- ffs %>% filter(receiver == TRUE & actor == TRUE & team.name == "England")
# adding both and sorting
ffsENG <- rbind(ffs_crossENG, ffs_receiptENG) %>% arrange(match_id, minute, second)
# more filtering
ffsENG <- ffsENG %>% filter(pass.end_location.x == 0 |
```

```
               pass.end_location.x>102 &
               pass.end_location.y == 0 |
               pass.end_location.y>18 &
               pass.end_location.y<62) %>%
                     filter(is.na(pass.type.name) |
                     pass.type.name=="Recovery" |
                     pass.type.name=="Interception")
# rearranging columns
ffsENG %>%
  select(pass.cross, player.name, type.name, minute, second, teammate, actor, receiver, everything())
```

```
## # A tibble: 4,233 x 20
## # Groups:   team.name [1]
##    pass.cross player.name  type.name minute second teammate actor receiver id
##    <lgl>      <chr>        <chr>      <int>  <int> <lgl>    <lgl> <lgl>    <chr>
##  1 NA         John Stones  Ball Rec~      0      1 TRUE     TRUE  TRUE     0705~
##  2 NA         Raheem Ster~ Ball Rec~      0     35 TRUE     TRUE  TRUE     8379~
##  3 NA         Kieran Trip~ Ball Rec~      0     36 TRUE     TRUE  TRUE     bd7a~
##  4 NA         Tyrone Mings Ball Rec~      0     38 TRUE     TRUE  TRUE     077a~
##  5 NA         John Stones  Ball Rec~      0     42 TRUE     TRUE  TRUE     c9ad~
##  6 NA         Kyle Walker  Ball Rec~      0     45 TRUE     TRUE  TRUE     ac57~
##  7 NA         Jordan Pick~ Ball Rec~      0     47 TRUE     TRUE  TRUE     dc12~
##  8 NA         John Stones  Ball Rec~      0     55 TRUE     TRUE  TRUE     fc61~
##  9 NA         Jordan Pick~ Ball Rec~      0     58 TRUE     TRUE  TRUE     bf1a~
## 10 NA         Kalvin Phil~ Ball Rec~      1     16 TRUE     TRUE  TRUE     648b~
## # ... with 4,223 more rows, and 11 more variables: match_id <int>,
## #   team.name <chr>, OpposingTeam <chr>, location.x <dbl>, location.y <dbl>,
## #   pass.end_location.x <dbl>, pass.end_location.y <dbl>, pass.type.name <chr>,
## #   ff_location.x <dbl>, ff_location.y <dbl>, keeper <lgl>
```

```
# deleting rows not following a pass.cross == TRUE
ffsENG <- ffsENG[-c(1:531, 549:662, 678:692, 708:735,
                    751:799, 816:1073, 1089, 1091, 1108:1226, 1242:1490, 1507:1520,
                    1531:1693, 1710:1729, 1743:1798, 1815:1920, 1937:1952,
                    1967:2017, 2032, 2034:2322, 2340:2621,
                    2633:2758, 2771:2772, 2804:2851, 2866:2972, 2988:2998,
                    3009, 3011:3021, 3032:3054, 3069:3073, 3079:3148, 3164,
                    3166:3188, 3200:3223, 3240:3257, 3273:3290, 3306, 3308:3357,
                    3374, 3376:3415, 3430:3431, 3445:3485, 3503:3523, 3543:3637,
                    3669:3744, 3760:3774, 3791, 3793:3816, 3836:3917, 3930:4055,
                    4072:4163, 4180:4233),]
```

```
# creating and adding vector of 1s
rep <- rep(1, times = 661)
ffsENG$rep <- rep
# changing vector of 1s to 0s where pass.cross == TRUE
ffsENG <- within(ffsENG, {
    f <- rep == 1 & pass.cross == TRUE
    rep[f] <- 0
})
```

```
# changing_id to combine ball receipts and cross pass id's
ffsENG[17,1] <- "f947809e-59e8-4811-bf17-3943799605e1"
ffsENG[32,1] <- "e420a456-c703-4f72-be75-318c046e1a7b"
ffsENG[47,1] <- "890a8656-ac35-445b-a41d-d53d6eea13b8"
```

```
ffsENG[62,1] <- "c5a561b7-3baa-4c4f-b332-2a6b6cb685df"
ffsENG[78,1] <- "a906c961-7a4d-4e7e-b7c5-8aae24dba46b"
ffsENG[94,1] <- "f2d05437-9ad6-43ce-961c-0f35ec1c30c5"
ffsENG[110,1] <- "b2e05ec1-01dc-4284-9597-e12e5d2b0660"
ffsENG[125,1] <- "d2e61874-9093-4e46-bfe9-fef170d06883"
ffsENG[141,1] <- "17800bc4-5206-4758-ad7c-6f45758b3e10"
ffsENG[151,1] <- "8907b8de-ec0a-4e78-b971-0af704efc58f"
ffsENG[167,1] <- "c9fc1306-476d-4ba0-8eef-b5ed70713050"
ffsENG[180,1] <- "351881f0-ff11-4eb1-8bb2-aec270de60e8"
ffsENG[196,1] <- "f0de3f89-af52-49f5-8679-7e094cff6bd6"
ffsENG[212,1] <- "ea9c308c-7c87-4ec6-8039-dd94c7df5344"
ffsENG[226,1] <- "2c48a1dd-668b-457d-895b-edfc95e8536c"
ffsENG[241,1] <- "54714e89-571a-4d71-a820-2221c20bb25d"
ffsENG[258,1] <- "df47e1c0-1360-4a7e-80d7-1a7dc739bfd7"
ffsENG[269,1] <- "6049b572-c3e2-42ee-9c6f-6b49a9c8024c"
ffsENG[281,1] <- "56a9976d-5766-4bbd-acf5-b8d1c5a77013"
ffsENG[295,1] <- "c5b3c99d-c39e-436c-86ee-f93ee66f7888"
ffsENG[312,1] <- "88a644a5-8f04-4397-ac3e-031a92c549e4"
ffsENG[326,1] <- "c09bdb5b-4ff0-4db2-a011-6ddd7fcb329b"
ffsENG[341,1] <- "35e44698-7473-45b5-b433-cc047e55b11b"
ffsENG[352,1] <- "02966aa6-3e74-487f-a0b8-a14fc34243c1"
ffsENG[362,1] <- "a0fe1c91-de36-47bb-85db-6514b700549f"
ffsENG[376,1] <- "1c23fbf7-d764-41d0-8942-f89a0d991582"
ffsENG[381,1] <- "d01ab064-0e01-485d-b86c-e9e6d558fd14"
ffsENG[397,1] <- "fe68ea09-4065-4a84-8b5c-017a1f2a2919"
ffsENG[408,1] <- "41f7cd02-9851-4db6-a811-f67475537dfc"
ffsENG[424,1] <- "3169eac6-8a4b-402c-b974-9ed93f04f132"
ffsENG[439,1] <- "292d50e1-3339-48a1-be30-6b64ab473930"
ffsENG[455,1] <- "fbbe8777-345d-4504-84eb-ab731e500eda"
ffsENG[472,1] <- "2c420308-5aaa-4ffb-a33c-486f62bf1bc1"
ffsENG[486,1] <- "8e908909-3a81-472e-8a49-e58635ae54a6"
ffsENG[499,1] <- "282d4387-5706-494d-b94c-2599e09bbc18"
ffsENG[516,1] <- "316dfabd-8524-4bab-9dc1-82950d74d5f6"
ffsENG[535,1] <- "8e5a98b9-779d-4482-a4db-bef4a815f3c6"
ffsENG[550,1] <- "ec3b3a56-e557-49ee-9e90-7c5b3f589f05"
ffsENG[566,1] <- "fc4cd1bf-96ad-4397-aaca-6d09bee1112a"
ffsENG[581,1] <- "782da551-4871-4a1e-91e3-507b36cee275"
ffsENG[598,1] <- "f7146a03-4596-4010-8986-2cc4015ef63c"
ffsENG[617,1] <- "ff5d74ef-5f39-47dd-a367-2b48df00f3db"
ffsENG[629,1] <- "e7c80d53-76ac-43c9-8a13-99078e9c7857"
ffsENG[645,1] <- "18e362ba-655e-42e2-9ce3-de8c1565968a"
ffsENG[661,1] <- "d69c7311-c708-4c40-b2d6-9b0dcd0f33a1"
```

```
# cutting out unsuccessful cross passes (receiver is opponent)
ffsENG_successful <- ffsENG[-c(213:226, 242:258, 282:295, 313:341, 353:376, 398:408,
                    425:455, 517:550, 582:598, 618:629, 646:661),]
ffsENG_unsuccessful <- ffsENG[c(213:226, 242:258, 282:295, 313:341, 353:376, 398:408,
                    425:455, 517:550, 582:598, 618:629, 646:661),]
```

```
# grouping
successful_crossesENG = ffsENG_successful %>%
    group_by(team.name, OpposingTeam, id) %>%
    summarise(attackers = sum(teammate==TRUE &
                              ff_location.x>102 &
```

```
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              defenders = sum(teammate==FALSE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              att_n_def = attackers+defenders,
              att_v_def = attackers-defenders,
              receipt = sum(rep==1)
              ) %>%
    ungroup() %>% arrange(OpposingTeam)

unsuccessful_crossesENG = ffsENG_unsuccessful %>%
    group_by(team.name, OpposingTeam, id) %>%
    summarise(attackers = sum(teammate==TRUE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              defenders = sum(teammate==FALSE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              att_n_def = attackers+defenders,
              att_v_def = attackers-defenders,
              receipt = sum(rep==1)
              ) %>%
    ungroup() %>% arrange(OpposingTeam)
```

```
# grouping
successful_crossesITA = ffsITA_successful %>%
    group_by(team.name, OpposingTeam, id) %>%
    summarise(attackers = sum(teammate==TRUE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              defenders = sum(teammate==FALSE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              att_n_def = attackers+defenders,
              att_v_def = attackers-defenders,
              receipt = sum(rep==1)
              ) %>%
    ungroup() %>% arrange(OpposingTeam)

unsuccessful_crossesITA = ffsITA_UNsuccessful %>%
    group_by(team.name, OpposingTeam, id) %>%
    summarise(attackers = sum(teammate==TRUE &
                              ff_location.x>102 &
                              ff_location.y>18 &
                              ff_location.y<62, na.rm = TRUE),
              defenders = sum(teammate==FALSE &
                              ff_location.x>102 &
                              ff_location.y>18 &
```

```
                              ff_location.y<62, na.rm = TRUE),
              att_n_def = attackers+defenders,
              att_v_def = attackers-defenders,
              receipt = sum(rep==1)
              ) %>%
    ungroup() %>% arrange(OpposingTeam)
```

```
succ_crossesITA <- successful_crossesITA %>% filter(OpposingTeam == "Austria" | OpposingTeam == "Englan
unsucc_crossesITA <- unsuccessful_crossesITA %>% filter(OpposingTeam == "Austria" | OpposingTeam == "Eng
```

```
succ_crossesENG <- successful_crossesENG %>% filter(OpposingTeam == "Germany" | OpposingTeam == "Italy"
unsucc_crossesENG <- unsuccessful_crossesENG %>% filter(OpposingTeam == "Germany" | OpposingTeam == "It
```

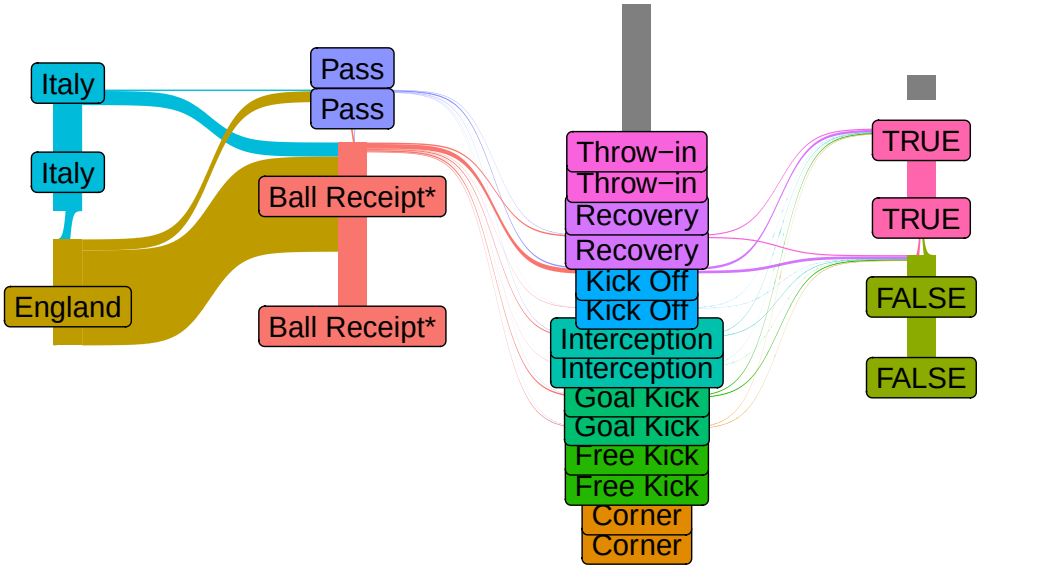
```
ffsITA_successful$id[10] <- "4d722c2e-5653-4058-a490-7768da557b6c"
ffsITA_successful$OpposingTeam[10] <- "Austria"

ffsITA_successful$id[23] <- "b4ff75e5-88d6-4cc0-88fa-e354dcafb9e4"
ffsITA_successful$OpposingTeam[23] <- "Austria"
```

Models, Visualizations, and Analysis

```
ggplot(data=df, aes(x = x,
               next_x = next_x,
               node = node,
               next_node = next_node,
               fill = factor(node),
               label= node)) +
  geom_sankey() +
  geom_sankey_label()+
  theme_sankey(base_size = 16)+
  theme(legend.position = "none")
```

Prong One: Sankey Diagram

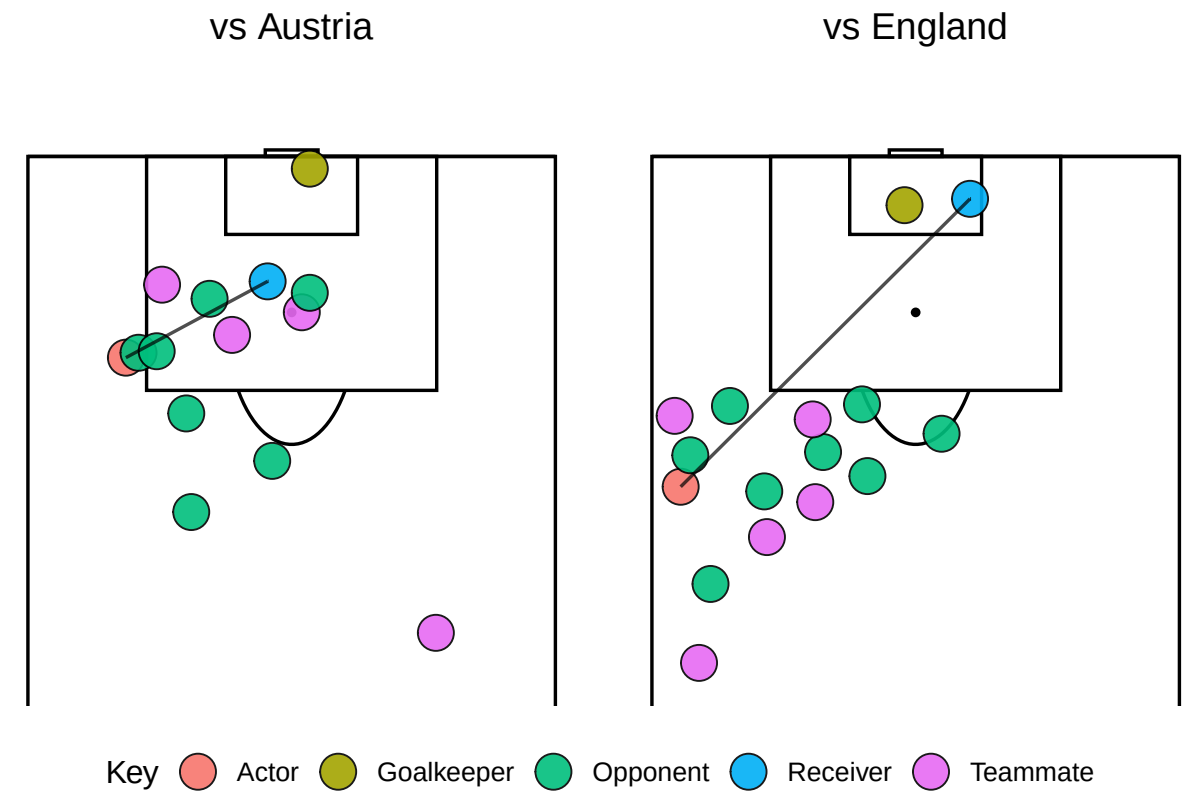
```
## Warning: Removed 4 rows containing missing values (geom_label).
```



We see that England had more ball receipts and passes than Italy did out of all their matches. We noticed that most of those passes and ball receipts stem from recovery and throw-in type passes. The rest were thinly distributed to the rest of the passes with corner type passes garnering the least from passes and ball receipts. it can be seen that recovery passes tend to lean more unsuccessful than successful passes. Throw-ins were split evenly between successful and unsuccessful pass. We can see that more of the goal kicks lead to unsuccessful than successful.AS noted in the data manipulation. The grey areas are empty due to the large quantities of N/A's in both variables.

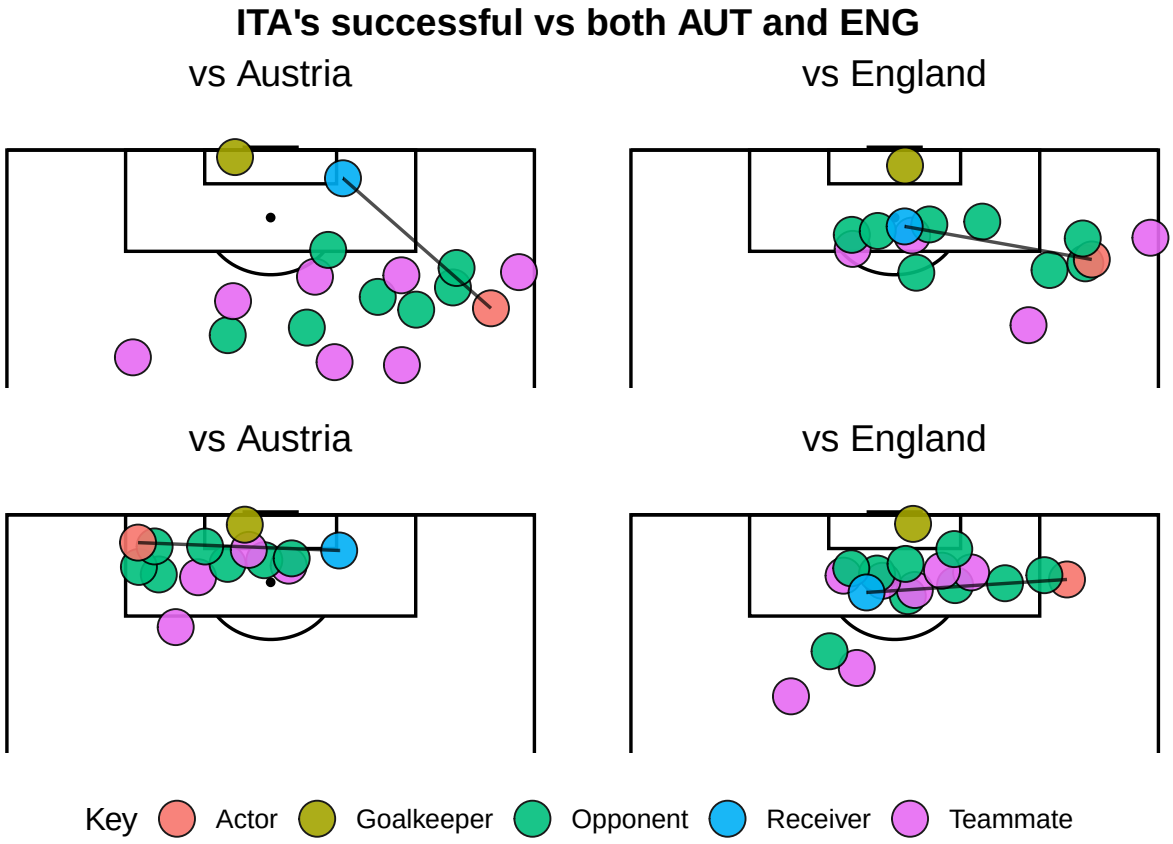
Prong Two: Maps of Successful Cross Passes We identified successful cross passes in the following matches: * Italy vs Austria * Italy vs England * England vs Germany * England vs Italy We chose these matches to compare similar cross passes between both finalists' (Italy and England) first and last games. Below, we map out similar cross passes between the games, identifying the success of each pass based on whether the recipient was a teammate of the passer or not.

```
p1 <- plot_id(ffsITA_successful, "4d722c2e-5653-4058-a490-7768da557b6c", "vs Austria") + extra()
p2 <- plot_id(ffsITA_UNsuccessful, "07e146f4-543f-4458-897b-176717d778ae", "vs England") + extra()
plot1 <- ggarrange(p1, p2, nrow=1, common.legend = TRUE, legend="bottom")
annotate_figure(plot1, top = text_grob("ITA's successful crosses vs AUT but unsuccessful crosses vs ENG
```



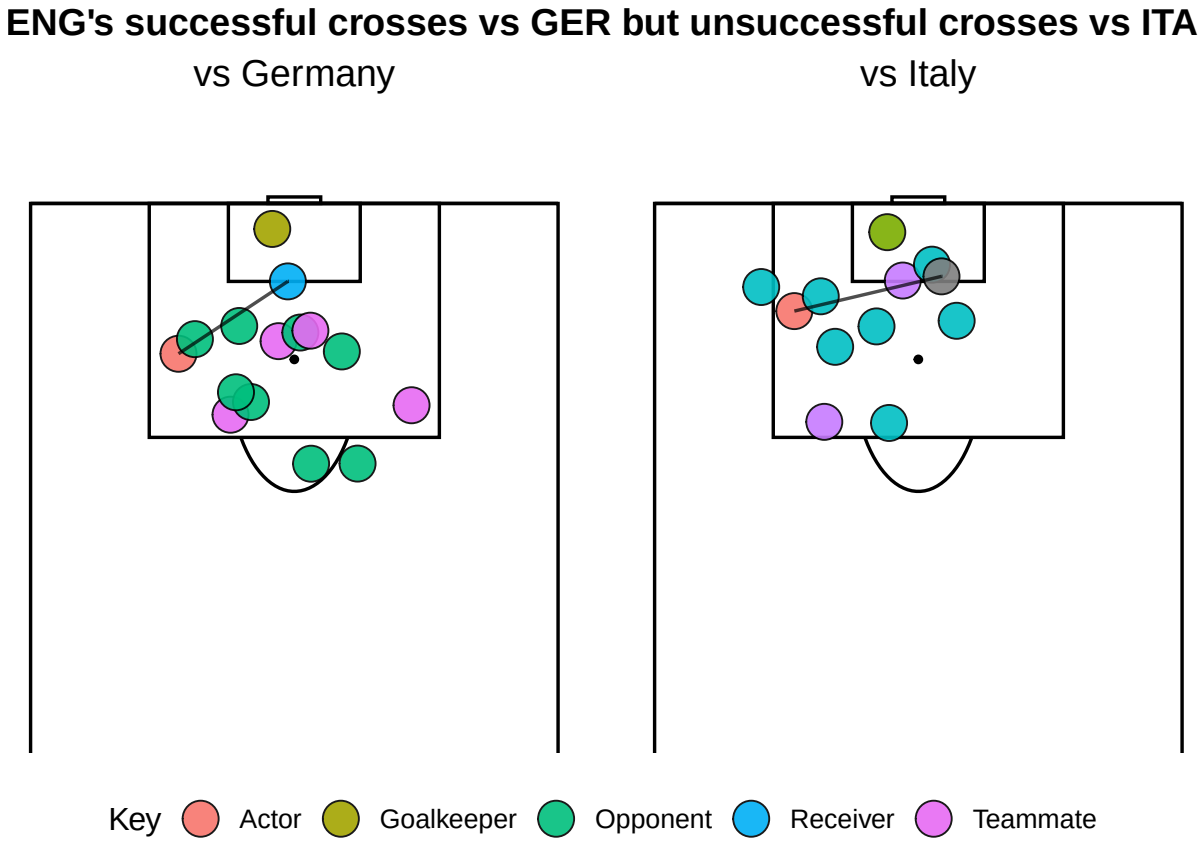
This first diagram above shows a successful cross by Italy in their match against Austria in the first round on the right. On the left, a similar cross made by Italy in the final round against England is shown, however, this cross was unsuccessful.

```
p4 <- plot_id(ffsITA_successful, "b4ff75e5-88d6-4cc0-88fa-e354dcafb9e4", "vs Austria")  + extra()
p5 <- plot_id(ffsITA_successful, "8eaf1e40-6bce-4caf-a8e3-ac332453e5f4", "vs England")  + extra()
p6 <- plot_id(ffsITA_successful, "6e2ec02c-3870-4ae4-9d21-5ff091df72c0", "vs Austria")  + extra()
p7 <- plot_id(ffsITA_successful, "6874f5b6-bbe4-4738-acb8-8abbfbb95a60", "vs England") + extra()
plot2 <- ggarrange(p4, p5, p6, p7, nrow=2, ncol=2, common.legend = TRUE, legend="bottom")
annotate_figure(plot2, top = text_grob("ITA's successful vs both AUT and ENG", color = "black", face =
```



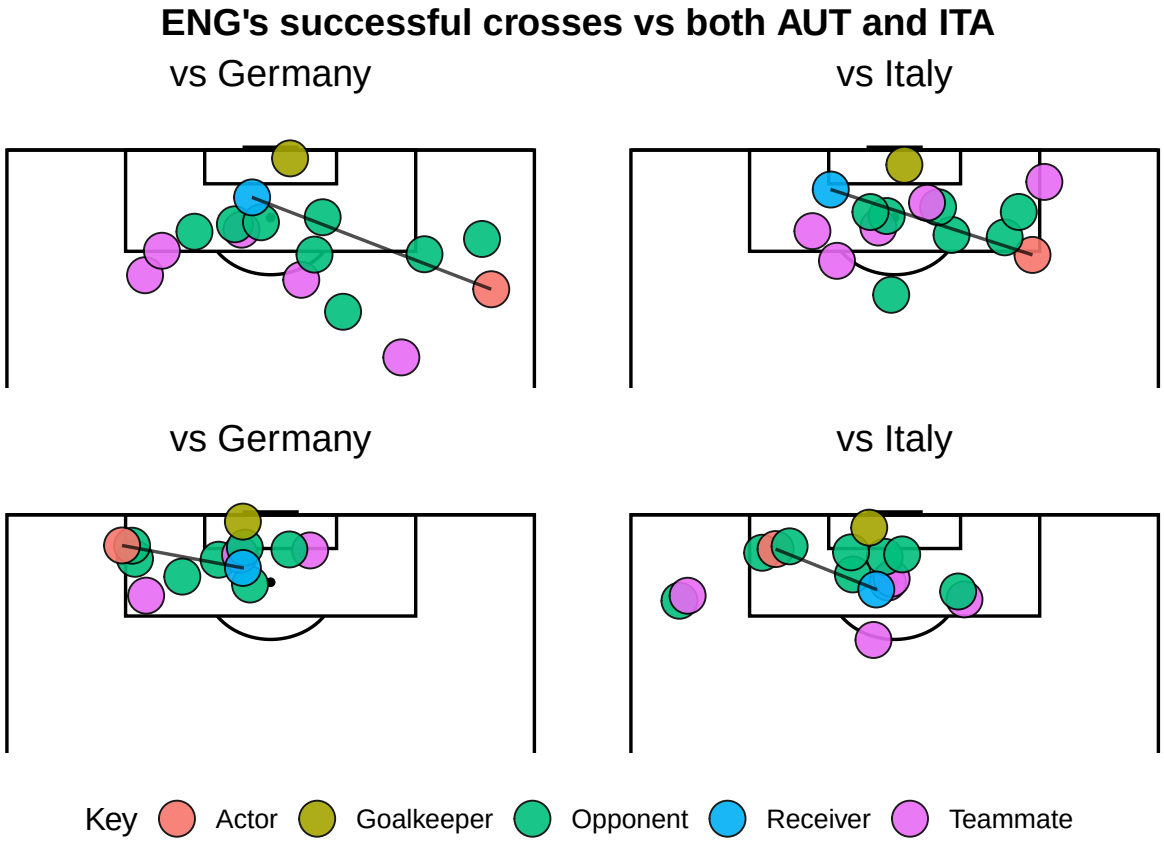
The second diagram, above, shows successful crosses made by Italy against Austria alongside successful crosses made by Italy against England, suggesting strong points in their offensive strategy and cross passing.

```
p8 <- plot_id(ffsENG_successful, "54714e89-571a-4d71-a820-2221c20bb25d", "vs Germany")  + extra()
p9  <- plot_id(ffsENG_unsuccessful, "e7c80d53-76ac-43c9-8a13-99078e9c7857", "vs Italy")  + extra()
plot4 <- ggarrange(p8, p9, nrow=1, common.legend = TRUE, legend="bottom")
annotate_figure(plot4, top = text_grob("ENG's successful crosses vs GER but unsuccessful crosses vs ITA
```



Similar to our first diagram, the above diagram, third in our set of four, shows a successful cross by England in their match against Germany in the first round on the right alongside a similar but unsuccessful cross against Italy.

```
p10 <- plot_id(ffsENG_successful, "ea9c308c-7c87-4ec6-8039-dd94c7df5344", "vs Germany") + extra()
p11 <- plot_id(ffsENG_successful, "782da551-4871-4a1e-91e3-507b36cee275", "vs Italy")  + extra()
p12 <- plot_id(ffsENG_successful, "351881f0-ff11-4eb1-8bb2-aec270de60e8", "vs Germany") + extra()
p13 <- plot_id(ffsENG_successful, "18e362ba-655e-42e2-9ce3-de8c1565968a", "vs Italy") + extra()
plot5 <- ggarrange(p10, p11, p12, p13,  nrow=2, ncol=2, common.legend = TRUE, legend="bottom")
annotate_figure(plot5, top = text_grob("ENG's successful crosses vs both AUT and ITA", color = "black",
```



Finally, our fourth diagram, above, shows successful crosses by England against Germany in the first round alongside similar successful cross passes made by England against Italy in the championship.

Crosses that were successful in the first round game but unsuccessful in the championship game indicate differences in strategy, skill level, and overall success of passing between games and as the team continued to face more challenging teams throughout the tournament. Crosses that were successful in both the first round and championship games indicate a strong suit in the team's offensive strategy. Mapping out successful and unsuccessful cross passes compared across first round and championship games gives insight into areas where the team should work to improve and the team's strong suits.

Conclusion

The availability of data such as these provided by StatsBomb allow for interesting and effective exploratory analysis such as those contained in this report. By analyzing which pass types made up what number of passes for each team with the Sankey diagram and comparing the effectiveness of similar crosses with the several maps presented, we have created methods able to highlight the strong points of each teams and what made them successful in the 2020 Euros.

References

[1] "StatsBomb: Advanced Football Analytics Through An Interactive Platform." Sport Performance Analysis, https://www.sportperformanceanalysis.com/article/statsbomb-advanced-football-analytics-through-an-interactive-visualisation-platform.

[2] StatsBomb. "StatsBomb Announce The Release Of Free StatsBomb 360 Data: Euro 2020 Available Now." StatsBomb | Data Champions, 17 Nov. 2021, https://statsbomb.com/news/statsbomb-announce-the-release-of-free-statsbomb-360-data-euro-2020-available-now/.

[3] Wickham H, Girlich M (2022). tidyr: Tidy Messy Data. https://tidyr.tidyverse.org, https://github.com/tidyverse/tidyr.