ativ_3

April 22, 2023

[1]:
```
import pandas as pd
import socceraction.spadl as spadl
import matplotsoccer as mps
```

[2]:
```
import matplotlib.pyplot as plt
from mplsoccer import Pitch
```

[3]:
```
import scipy
import numpy as np
```

1 [CDAF] Atividade 3

1.1 Nome e matrícula

Nome: Igor Lacerda Faria da Silva Matrícula: 2020049173

1.2 Referências

- [1] https://figshare.com/collections/Soccer_match_event_dataset/4415000
- [2] https://socceraction.readthedocs.io/en/latest/api/generated/socceraction.spadl.wyscout.convert_to_act
- [3] https://github.com/TomDecroos/matplotsoccer
- [4] https://soccermatics.readthedocs.io/en/latest/gallery/lesson1/plot_PlottingShots.html
- [5] https://soccermatics.readthedocs.io/en/latest/gallery/lesson1/plot_PlottingPasses.html
- [6] https://soccermatics.readthedocs.io/en/latest/gallery/lesson1/plot_PassNetworks.html

1.3 Questão 1

- Baixe o dataset 'Wyscout Europa Top 5 2017/2018' em [1].
- Escolha uma partida e carregue os dados de eventos em um dataframe do pandas.
- Converta os dados de eventos para SPADL [2].

[4]:
```
PATH_DF = "data/events/events_Italy.json"
```

[5]:
```
df = pd.read_json(PATH_DF)
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 647372 entries, 0 to 647371
```

```
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   eventId       647372 non-null  int64
 1   subEventName  647372 non-null  object
 2   tags          647372 non-null  object
 3   playerId      647372 non-null  int64
 4   positions     647372 non-null  object
 5   matchId       647372 non-null  int64
 6   eventName     647372 non-null  object
 7   teamId        647372 non-null  int64
 8   matchPeriod   647372 non-null  object
 9   eventSec      647372 non-null  float64
 10  subEventId    647372 non-null  object
 11  id            647372 non-null  int64
dtypes: float64(1), int64(5), object(6)
memory usage: 59.3+ MB
```

[7]:
```
df.head()
```

[7]:
```
   eventId           subEventName                                     tags  \
0        8            Simple pass                    [{'id': 1801}]
1        8            Simple pass                    [{'id': 1801}]
2        7                  Touch                                       []
3        1  Ground attacking duel  [{'id': 504}, {'id': 703}, {'id': 1801}]
4        1  Ground attacking duel  [{'id': 503}, {'id': 703}, {'id': 1801}]

   playerId                                 positions  matchId  \
0      8327  [{'y': 52, 'x': 49}, {'y': 44, 'x': 43}]  2575959
1     20438  [{'y': 44, 'x': 43}, {'y': 17, 'x': 36}]  2575959
2      8306  [{'y': 17, 'x': 36}, {'y': 56, 'x': 78}]  2575959
3      8306  [{'y': 56, 'x': 78}, {'y': 15, 'x': 64}]  2575959
4      8306  [{'y': 15, 'x': 64}, {'y': 15, 'x': 72}]  2575959

             eventName  teamId matchPeriod  eventSec subEventId        id
0                 Pass    3158          1H  2.530536         85  180423957
1                 Pass    3158          1H  3.768418         85  180423958
2  Others on the ball    3158          1H  4.868265         72  180423959
3                 Duel    3158          1H  8.114676         11  180423960
4                 Duel    3158          1H  8.647892         11  180423961
```

[8]:
```
df["matchId"].unique()
```

[8]:
```
array([2575959, 2575960, 2575961, 2575962, 2575963, 2575964, 2575965,
       2575966, 2575967, 2575968, 2575969, 2575970, 2575971, 2575972,
       2575973, 2575974, 2575975, 2575976, 2575977, 2575978, 2575979,
       2575980, 2575981, 2575982, 2575983, 2575984, 2575985, 2575986,
       2575987, 2575988, 2575989, 2575990, 2575991, 2575992, 2575993,
```

2575994, 2575995, 2575996, 2575997, 2575998, 2575999, 2576000,
2576001, 2576002, 2576003, 2576004, 2576005, 2576006, 2576007,
2576008, 2576009, 2576010, 2576011, 2576012, 2576013, 2576014,
2576015, 2576016, 2576017, 2576018, 2576019, 2576020, 2576021,
2576022, 2576023, 2576024, 2576025, 2576026, 2576027, 2576028,
2576029, 2576030, 2576031, 2576032, 2576033, 2576034, 2576035,
2576036, 2576037, 2576038, 2576039, 2576040, 2576041, 2576042,
2576043, 2576044, 2576045, 2576046, 2576047, 2576048, 2576049,
2576050, 2576051, 2576052, 2576053, 2576054, 2576055, 2576056,
2576057, 2576058, 2576059, 2576060, 2576061, 2576062, 2576063,
2576064, 2576065, 2576066, 2576067, 2576068, 2576069, 2576070,
2576071, 2576072, 2576073, 2576074, 2576075, 2576076, 2576077,
2576078, 2576079, 2576080, 2576081, 2576082, 2576083, 2576084,
2576085, 2576086, 2576087, 2576088, 2576089, 2576090, 2576091,
2576092, 2576093, 2576094, 2576095, 2576096, 2576097, 2576098,
2576099, 2576100, 2576101, 2576102, 2576103, 2576104, 2576105,
2576106, 2576107, 2576108, 2576109, 2576110, 2576111, 2576112,
2576113, 2576114, 2576115, 2576116, 2576117, 2576118, 2576119,
2576120, 2576121, 2576122, 2576123, 2576124, 2576125, 2576126,
2576127, 2576128, 2576129, 2576130, 2576131, 2576132, 2576133,
2576134, 2576135, 2576136, 2576137, 2576138, 2576139, 2576140,
2576141, 2576142, 2576143, 2576144, 2576145, 2576146, 2576147,
2576148, 2576149, 2576150, 2576151, 2576152, 2576153, 2576154,
2576155, 2576156, 2576157, 2576158, 2576159, 2576160, 2576161,
2576162, 2576163, 2576164, 2576165, 2576166, 2576167, 2576168,
2576169, 2576170, 2576171, 2576172, 2576173, 2576174, 2576175,
2576176, 2576177, 2576178, 2576179, 2576180, 2576181, 2576182,
2576183, 2576184, 2576185, 2576186, 2576187, 2576188, 2576189,
2576190, 2576191, 2576192, 2576193, 2576194, 2576195, 2576196,
2576197, 2576198, 2576199, 2576200, 2576201, 2576202, 2576203,
2576204, 2576205, 2576206, 2576207, 2576208, 2576209, 2576210,
2576211, 2576212, 2576213, 2576214, 2576215, 2576216, 2576217,
2576218, 2576219, 2576220, 2576221, 2576222, 2576223, 2576224,
2576225, 2576226, 2576227, 2576228, 2576229, 2576230, 2576231,
2576232, 2576233, 2576234, 2576235, 2576236, 2576237, 2576238,
2576239, 2576240, 2576241, 2576242, 2576243, 2576244, 2576245,
2576246, 2576247, 2576248, 2576249, 2576250, 2576251, 2576252,
2576253, 2576254, 2576255, 2576256, 2576257, 2576258, 2576259,
2576260, 2576261, 2576262, 2576263, 2576264, 2576265, 2576266,
2576267, 2576268, 2576269, 2576270, 2576271, 2576272, 2576273,
2576274, 2576275, 2576276, 2576277, 2576278, 2576279, 2576280,
2576281, 2576282, 2576283, 2576284, 2576285, 2576286, 2576287,
2576288, 2576289, 2576290, 2576291, 2576292, 2576293, 2576294,
2576295, 2576296, 2576297, 2576298, 2576299, 2576300, 2576301,
2576302, 2576303, 2576304, 2576305, 2576306, 2576307, 2576308,
2576309, 2576310, 2576311, 2576312, 2576313, 2576314, 2576315,
2576316, 2576317, 2576318, 2576319, 2576320, 2576321, 2576322,

```
       2576323, 2576324, 2576325, 2576326, 2576327, 2576328, 2576329,
       2576330, 2576331, 2576332, 2576333, 2576334, 2576335, 2576336,
       2576337, 2576338])
```

[9]:
```
MATCH_ID = 2575959
df_match = df.query("matchId == @MATCH_ID")
```

[10]:
```
df_match.head()
```

[10]:

	eventId	subEventName	tags	playerId	positions	matchId	eventName	teamId	matchPeriod	eventSec	subEventId	id
0	8	Simple pass	[{'id': 1801}]	8327	[{'y': 52, 'x': 49}, {'y': 44, 'x': 43}]	2575959	Pass	3158	1H	2.530536	85	180423957
1	8	Simple pass	[{'id': 1801}]	20438	[{'y': 44, 'x': 43}, {'y': 17, 'x': 36}]	2575959	Pass	3158	1H	3.768418	85	180423958
2	7	Touch	[]	8306	[{'y': 17, 'x': 36}, {'y': 56, 'x': 78}]	2575959	Others on the ball	3158	1H	4.868265	72	180423959
3	1	Ground attacking duel	[{'id': 504}, {'id': 703}, {'id': 1801}]	8306	[{'y': 56, 'x': 78}, {'y': 15, 'x': 64}]	2575959	Duel	3158	1H	8.114676	11	180423960
4	1	Ground attacking duel	[{'id': 503}, {'id': 703}, {'id': 1801}]	8306	[{'y': 15, 'x': 64}, {'y': 15, 'x': 72}]	2575959	Duel	3158	1H	8.647892	11	180423961

[11]:
```
correct_columns = {
    "eventId": "type_id",
    "subEventName": "subtype_name",
    "playerId": "player_id",
    "matchId": "game_id",
    "eventName": "type_name",
    "teamId": "team_id",
    "eventSec": "milliseconds",
    "subEventId": "subtype_id",
    "id": "event_id",
}
df_match = df_match.rename(columns=correct_columns)
df_match["period_id"] = pd.factorize(df_match["matchPeriod"])[0] + 1
```

[12]:
```
df_match.head()
```

[12]:
```
   type_id           subtype_name                                      tags  \
0        8            Simple pass                       [{'id': 1801}]
1        8            Simple pass                       [{'id': 1801}]
2        7                  Touch                                      []
3        1  Ground attacking duel  [{'id': 504}, {'id': 703}, {'id': 1801}]
4        1  Ground attacking duel  [{'id': 503}, {'id': 703}, {'id': 1801}]

   player_id                                 positions  game_id  \
0       8327  [{'y': 52, 'x': 49}, {'y': 44, 'x': 43}]  2575959
1      20438  [{'y': 44, 'x': 43}, {'y': 17, 'x': 36}]  2575959
2       8306  [{'y': 17, 'x': 36}, {'y': 56, 'x': 78}]  2575959
3       8306  [{'y': 56, 'x': 78}, {'y': 15, 'x': 64}]  2575959
4       8306  [{'y': 15, 'x': 64}, {'y': 15, 'x': 72}]  2575959

            type_name  team_id matchPeriod  milliseconds subtype_id  \
0                Pass     3158          1H      2.530536         85
1                Pass     3158          1H      3.768418         85
2  Others on the ball     3158          1H      4.868265         72
3                Duel     3158          1H      8.114676         11
4                Duel     3158          1H      8.647892         11

    event_id  period_id
0  180423957          1
1  180423958          1
2  180423959          1
3  180423960          1
4  180423961          1
```

[13]:
```
df_match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1613 entries, 0 to 1612
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   type_id       1613 non-null   int64
 1   subtype_name  1613 non-null   object
 2   tags          1613 non-null   object
 3   player_id     1613 non-null   int64
 4   positions     1613 non-null   object
 5   game_id       1613 non-null   int64
 6   type_name     1613 non-null   object
 7   team_id       1613 non-null   int64
 8   matchPeriod   1613 non-null   object
 9   milliseconds  1613 non-null   float64
 10  subtype_id    1613 non-null   object
 11  event_id      1613 non-null   int64
 12  period_id     1613 non-null   int64
```

```
dtypes: float64(1), int64(6), object(6)
memory usage: 176.4+ KB
```

[14]:
```
TEAM_ID = 3158
OTHER_TEAM_ID = 3172
```

[15]:
```
df_spadl = spadl.wyscout.convert_to_actions(df_match, TEAM_ID)
```

[16]:
```
df_spadl
```

[16]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
0	2575959	1	0.002531	3158	8327	51.45	32.64	
1	2575959	1	0.003768	3158	20438	45.15	38.08	
2	2575959	1	0.005942	3158	8306	37.80	56.44	
3	2575959	1	0.008115	3158	8306	81.90	29.92	
4	2575959	1	0.008648	3158	8306	67.20	57.80	
…	…	…	…	…	…	…	…	…
1217	2575959	2	2.980286	3158	20879	25.20	13.60	
1218	2575959	2	2.983099	3158	8327	38.85	16.32	
1219	2575959	2	2.987436	3172	41034	77.70	16.32	
1220	2575959	2	2.991065	3172	50849	91.35	36.04	
1221	2575959	2	2.995171	3172	295176	61.95	59.16	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
0	45.15	38.08	180423957	0	0	1	
1	37.80	56.44	180423958	0	0	1	
2	81.90	29.92	NaN	0	21	1	
3	67.20	57.80	180423960	0	7	1	
4	75.60	57.80	180423961	0	7	1	
…	…	…	…	…	…	…	
1217	38.85	16.32	180425718	0	0	1	
1218	77.70	16.32	180425719	0	0	0	
1219	91.35	36.04	180425709	0	0	1	
1220	61.95	59.16	180425710	0	0	1	
1221	35.70	66.64	180425712	0	0	1	

	action_id
0	0
1	1
2	2
3	3
4	4
…	…
1217	1217
1218	1218
1219	1219
1220	1220
1221	1221

```
[1222 rows x 14 columns]
```

1.4 Questão 2

- Visualize uma sequência de 5 ações da partida usando matplotsoccer.actions [3].

[17]:
```
df_spadl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1222 entries, 0 to 1221
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   game_id            1222 non-null   int64
 1   period_id          1222 non-null   int64
 2   time_seconds       1222 non-null   float64
 3   team_id            1222 non-null   int64
 4   player_id          1222 non-null   int64
 5   start_x            1222 non-null   float64
 6   start_y            1222 non-null   float64
 7   end_x              1222 non-null   float64
 8   end_y              1222 non-null   float64
 9   original_event_id  1125 non-null   object
 10  bodypart_id        1222 non-null   int64
 11  type_id            1222 non-null   int64
 12  result_id          1222 non-null   int64
 13  action_id          1222 non-null   int64
dtypes: float64(5), int64(8), object(1)
memory usage: 133.8+ KB
```

[18]:
```
df_spadl.head(10)
```

[18]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
0	2575959	1	0.002531	3158	8327	51.45	32.64	
1	2575959	1	0.003768	3158	20438	45.15	38.08	
2	2575959	1	0.005942	3158	8306	37.80	56.44	
3	2575959	1	0.008115	3158	8306	81.90	29.92	
4	2575959	1	0.008648	3158	8306	67.20	57.80	
5	2575959	1	0.010376	3158	8306	75.60	57.80	
6	2575959	1	0.016241	3172	86366	65.10	55.76	
7	2575959	1	0.019153	3158	8306	64.05	57.80	
8	2575959	1	0.020873	3158	20518	38.85	57.80	
9	2575959	1	0.021504	3158	20438	43.05	47.60	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	action_id
0	45.15	38.08	180423957	0	0	1	0
1	37.80	56.44	180423958	0	0	1	1

```
2  81.90  29.92              NaN           0       21          1          2
3  67.20  57.80        180423960           0        7          1          3
4  75.60  57.80        180423961           0        7          1          4
5  75.60  51.00        180423962           0        0          1          5
6  59.85  59.84        180423979           0        0          1          6
7  38.85  57.80        180423968           0        0          1          7
8  43.05  47.60        180423969           0        0          1          8
9  35.70  57.12        180423970           0        0          1          9
```

[19]:
```
df_spadl = spadl.add_names(df_spadl)
```

[20]:
```
df_spadl["type_name"].unique()
```

[20]:
```
array(['pass', 'dribble', 'take_on', 'foul', 'freekick_short', 'cross',
       'goalkick', 'clearance', 'throw_in', 'interception', 'shot',
       'keeper_save', 'tackle', 'corner_short', 'corner_crossed',
       'shot_freekick', 'freekick_crossed'], dtype=object)
```

[21]:
```
# Acho que essa partida não teve gols
df_spadl.query("type_name == 'keeper_save'")
```

[21]:

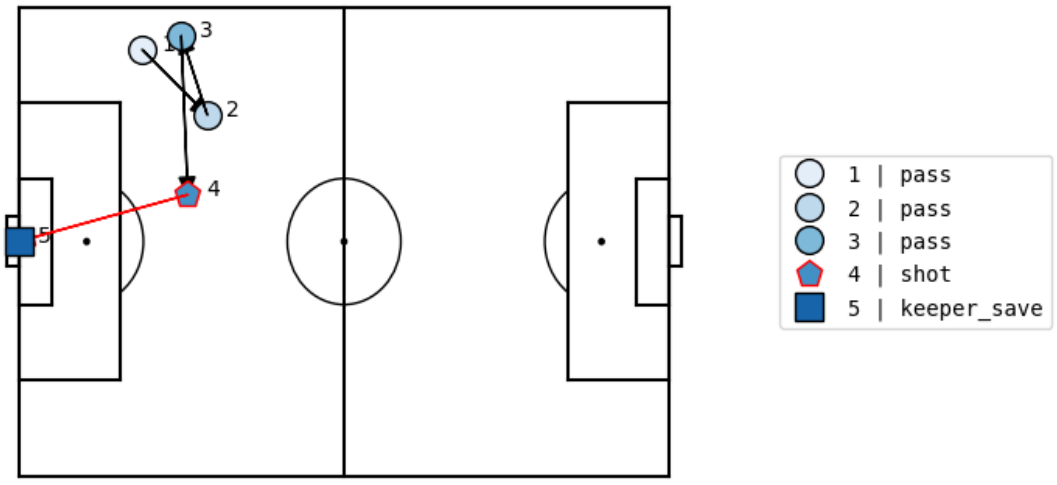
	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
44	2575959	1	0.195926	3158	214220	0.0	34.0	
341	2575959	1	1.463548	3158	214220	0.0	34.0	
544	2575959	1	2.485311	3158	214220	0.0	37.4	
646	2575959	2	0.188729	3158	214220	0.0	34.0	
928	2575959	2	1.480676	3158	214220	0.0	37.4	
1063	2575959	2	2.180659	3158	214220	0.0	30.6	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
44	0.0	34.0	180424035	2	14	1	
341	0.0	34.0	180424474	2	14	1	
544	0.0	37.4	180424785	2	14	1	
646	0.0	34.0	180424902	2	14	1	
928	0.0	37.4	180425354	2	14	1	
1063	0.0	30.6	180425535	2	14	1	

	action_id	type_name	result_name	bodypart_name
44	44	keeper_save	success	other
341	341	keeper_save	success	other
544	544	keeper_save	success	other
646	646	keeper_save	success	other
928	928	keeper_save	success	other
1063	1063	keeper_save	success	other

[22]:
```
df_action_sequence = df_spadl.loc[337:341]
df_action_sequence = spadl.add_names(df_action_sequence)
```

```
mps.actions(
    location=df_action_sequence[["start_x", "start_y", "end_x", "end_y"]],
    action_type=df_action_sequence.type_name,
    result=df_action_sequence.result_name == "success",
    zoom=False,
)
```



1.5 Questão 3

- Visualize os chutes da partida, desenvolvendo seu código em cima do dataframe SPADL. Faça um plot para cada time. Adapte de [4].
- Qual time as melhores chances da partida? Por quê?

[23]:
```
shot_list = ["shot", "shot_freekick", "shot_penalty"]
```

[24]:
```
df_shot = df_spadl.query("type_name in @shot_list")
```

[25]:
```
df_shot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24 entries, 43 to 1215
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   game_id            24 non-null     int64
 1   period_id          24 non-null     int64
 2   time_seconds       24 non-null     float64
 3   team_id            24 non-null     int64
```

```
 4   player_id          24 non-null     int64
 5   start_x            24 non-null     float64
 6   start_y            24 non-null     float64
 7   end_x              24 non-null     float64
 8   end_y              24 non-null     float64
 9   original_event_id  24 non-null     object
 10  bodypart_id        24 non-null     int64
 11  type_id            24 non-null     int64
 12  result_id          24 non-null     int64
 13  action_id          24 non-null     int64
 14  type_name          24 non-null     object
 15  result_name        24 non-null     object
 16  bodypart_name      24 non-null     object
dtypes: float64(5), int64(8), object(4)
memory usage: 3.4+ KB
```

[26]:
```
df_shot.head()
```

[26]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
43	2575959	1	0.193924	3172	21077	21.00	27.88	
64	2575959	1	0.250843	3158	8327	99.75	45.56	
153	2575959	1	0.633578	3158	20879	75.60	25.16	
318	2575959	1	1.359469	3158	23149	95.55	25.84	
328	2575959	1	1.412602	3172	295176	14.70	17.68	

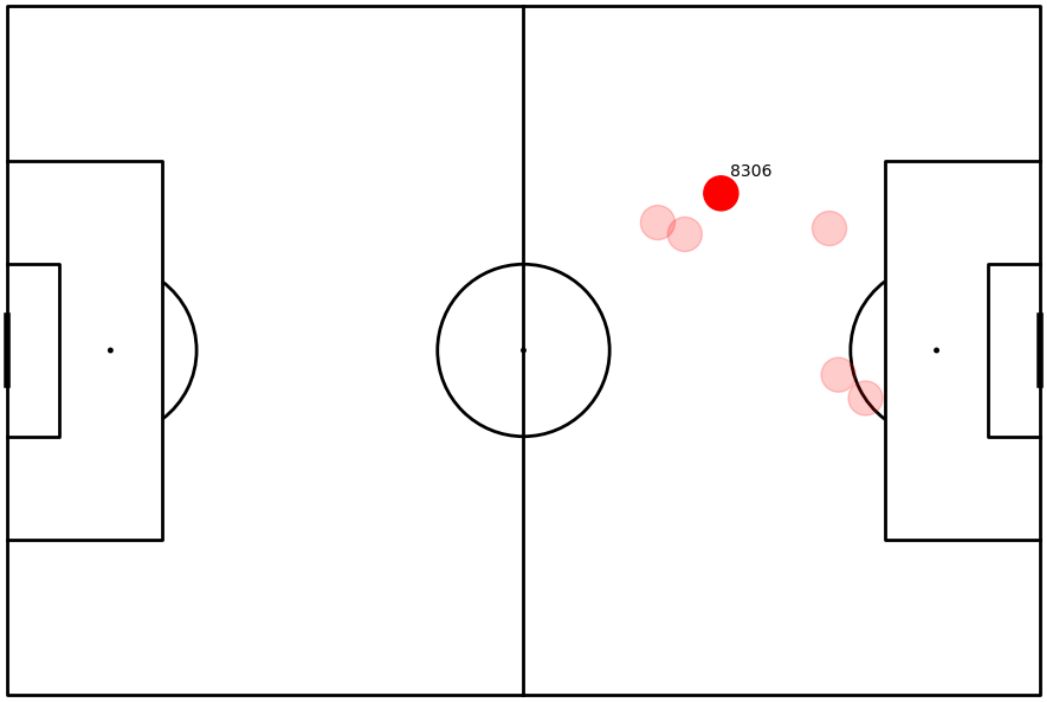
	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
43	0.0	34.00	180424028	0	11	0	
64	105.0	27.20	180424079	0	11	0	
153	75.6	25.16	180424208	0	11	0	
318	105.0	27.20	180424444	0	11	0	
328	0.0	40.80	180424409	0	11	0	

	action_id	type_name	result_name	bodypart_name
43	43	shot	fail	foot
64	64	shot	fail	foot
153	153	shot	fail	foot
318	318	shot	fail	foot
328	328	shot	fail	foot

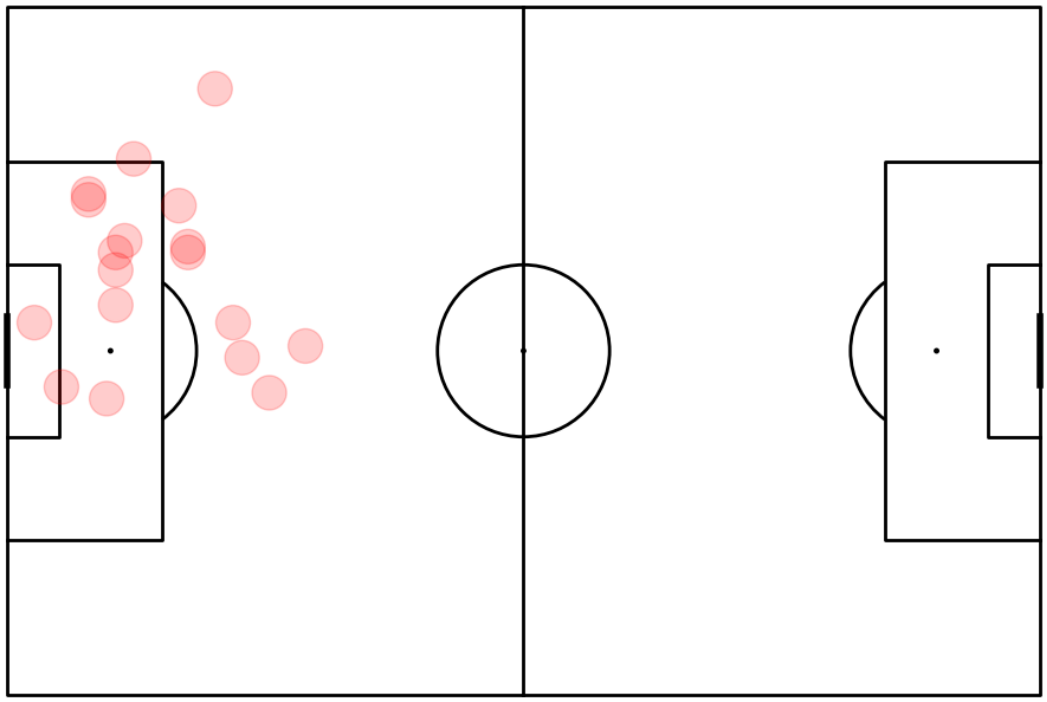
[27]:
```
def print_shots(shots: pd.DataFrame, id_team: str):
    pitch = Pitch(line_color="black")
    fig, ax = pitch.draw(figsize=(10, 7))
    # Plot the shots by looping through them.
    for _, shot in shots.iterrows():
        # get the information
        x = shot["start_x"]
        y = shot["start_y"]
        goal = shot["result_name"] == "success"
```

```
        # set circlesize
        circleSize = 2
        # plot first team
        if shot["team_id"] == id_team:
            if goal:
                shotCircle = plt.Circle((x, y), circleSize, color="red")
                plt.text(x + 1, y - 2, shot["player_id"])
            else:
                shotCircle = plt.Circle((x, y), circleSize, color="red")
                shotCircle.set_alpha(0.2)
            ax.add_patch(shotCircle)
    # set title
    fig.suptitle(f"{id_team} shots", color="white")
    fig.set_size_inches(10, 7)
    plt.show()
```

[28]:
```
print_shots(df_shot, TEAM_ID)
```



[29]:
```
print_shots(df_shot, OTHER_TEAM_ID)
```



1.5.1 Análise

O time 3172 (não sei quem é) teve ótimas oportunidades, mas errou todas suas chances. Por outro lado, o time 3158, mesmo tendo menos chances, conseguiu acertar um gol bem mais de longe e com menos tentativas, parabéns pra eles.

1.6 Questão 4

- Escolha um jogador da partida que você escolheu.
- Faça um heatmap de todas ações dele [3].
- Faça um heatmap de todas as ações ofensivas dele [3].
- Faça um heatmap de todas as ações defensivas dele [3].
- O que você pode inferir sobre o comportamento do jogador? O comportamento dele varia muito do ataque para a defesa?

[30]:
```
df_spadl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1222 entries, 0 to 1221
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   game_id            1222 non-null   int64
```

```
 1   period_id          1222 non-null   int64
 2   time_seconds       1222 non-null   float64
 3   team_id            1222 non-null   int64
 4   player_id          1222 non-null   int64
 5   start_x            1222 non-null   float64
 6   start_y            1222 non-null   float64
 7   end_x              1222 non-null   float64
 8   end_y              1222 non-null   float64
 9   original_event_id  1125 non-null   object
 10  bodypart_id        1222 non-null   int64
 11  type_id            1222 non-null   int64
 12  result_id          1222 non-null   int64
 13  action_id          1222 non-null   int64
 14  type_name          1222 non-null   object
 15  result_name        1222 non-null   object
 16  bodypart_name      1222 non-null   object
dtypes: float64(5), int64(8), object(4)
memory usage: 162.4+ KB
```

[31]:
```
df_spadl["player_id"].unique()
```

[31]:
```
array([  8327,  20438,   8306,  86366,  20518,   3475,  50849, 295176,
        21077,  49991,  92966,  20841,  41034,  20404, 214220,    114,
        23149,  44251,  20879,      0, 280419, 246175,  21620, 135150,
        20820,    625,   3463,  20418])
```

[32]:
```
PLAYER_ID = 8327
```

[33]:
```
df_player = df_spadl.query(f"player_id == {PLAYER_ID}")
```

[34]:
```
df_player.head()
```

[34]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	action_id	type_name	result_name	bodypart_name
0	2575959	1	0.002531	3158	8327	51.45	32.64	45.15	38.08	180423957	0	0	1	0	pass	success	foot
54	2575959	1	0.216335	3158	8327	75.60	17.68	58.80	59.16	NaN	0	21	1				
55	2575959	1	0.218884	3158	8327	58.80	59.16	57.75	68.00	180424056	0	0	0				
58	2575959	1	0.237519	3158	8327	74.55	50.32	67.20	48.28	180424061	0	0	1				
64	2575959	1	0.250843	3158	8327	99.75	45.56	105.00	27.20	180424079	0	11	0				

```
54          54   dribble     success          foot
55          55      pass        fail          foot
58          58      pass     success          foot
64          64      shot        fail          foot
```

[35]:
```
df_player = spadl.add_names(df_player)
```

[36]:
```
df_player.info()
```

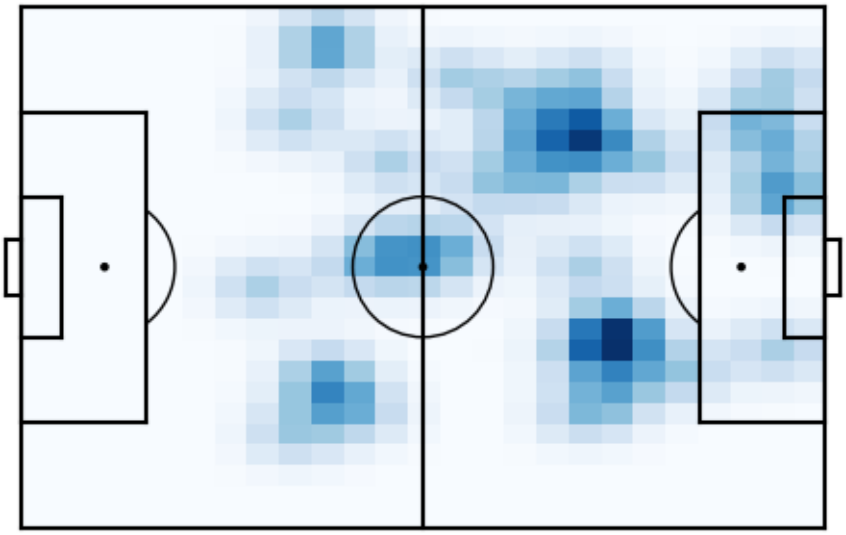
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37 entries, 0 to 1218
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   game_id            37 non-null     int64
 1   period_id          37 non-null     int64
 2   time_seconds       37 non-null     float64
 3   team_id            37 non-null     int64
 4   player_id          37 non-null     int64
 5   start_x            37 non-null     float64
 6   start_y            37 non-null     float64
 7   end_x              37 non-null     float64
 8   end_y              37 non-null     float64
 9   original_event_id  32 non-null     object
 10  bodypart_id        37 non-null     int64
 11  type_id            37 non-null     int64
 12  result_id          37 non-null     int64
 13  action_id          37 non-null     int64
 14  type_name          37 non-null     object
 15  result_name        37 non-null     object
 16  bodypart_name      37 non-null     object
dtypes: float64(5), int64(8), object(4)
memory usage: 5.2+ KB
```

[37]:
```
df_player["type_name"].unique()
```

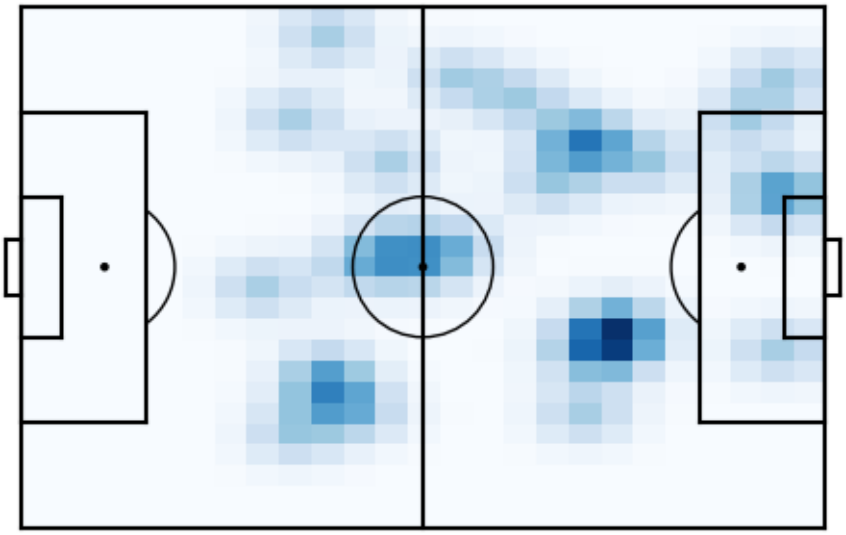
[37]:
```
array(['pass', 'dribble', 'shot', 'take_on', 'interception', 'cross'],
      dtype=object)
```

[38]:
```
def heatmap(df: pd.DataFrame):
    x = df["start_x"]
    y = df["start_y"]
    hm = mps.count(x, y, n=25, m=25)  # Construct a 25x25 heatmap from␣
 ↪x,y-coordinates
    hm = scipy.ndimage.gaussian_filter(hm, 1)
    mps.heatmap(hm)
```

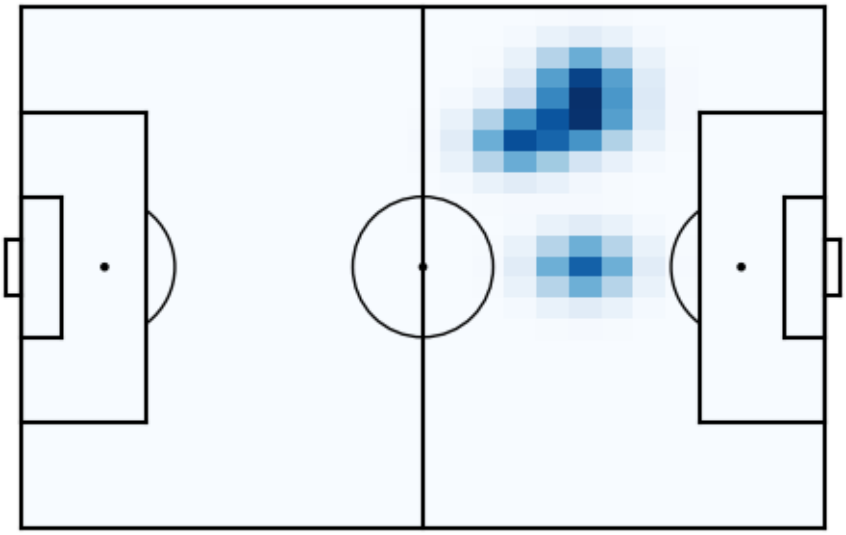
[39]:
```
heatmap(df_player)
```



[40]:
```
ATK_ACTION = [
    "pass",
    "cross",
    "throw_in",
    "freekick_crossed",
    "corner_crossed",
    "corner_short",
    "shot",
    "shot_penalty",
    "shot_freekick",
]

df_player_atk = df_player.query("type_name in @ATK_ACTION")
heatmap(df_player_atk)
```



[41]:
```
DEF_ACTION = [
    "take_on",
    "foul",
    "tackle",
    "interception",
    "keeper_save",
    "keeper_claim",
    "keeper_punch",
    "keeper_pick_up",
    "clearance",
]

df_player_def = df_player.query("type_name in @DEF_ACTION")
heatmap(df_player_def)
```



1.6.1 Análise

Esse jogador tem o perfil mais voltado ao ataque, fazendo ações defensivas somente quando necessário, para continuar um ataque.

1.7 Questão 5

- Para o mesmo jogador, crie um mapa de passes com os passes que ele efetuou na partida, desenvolvendo seu código em cima do dataframe SPADL. Adapte de [5].
- O mapa de passes trouxe alguma informação nova sobre o jogador?

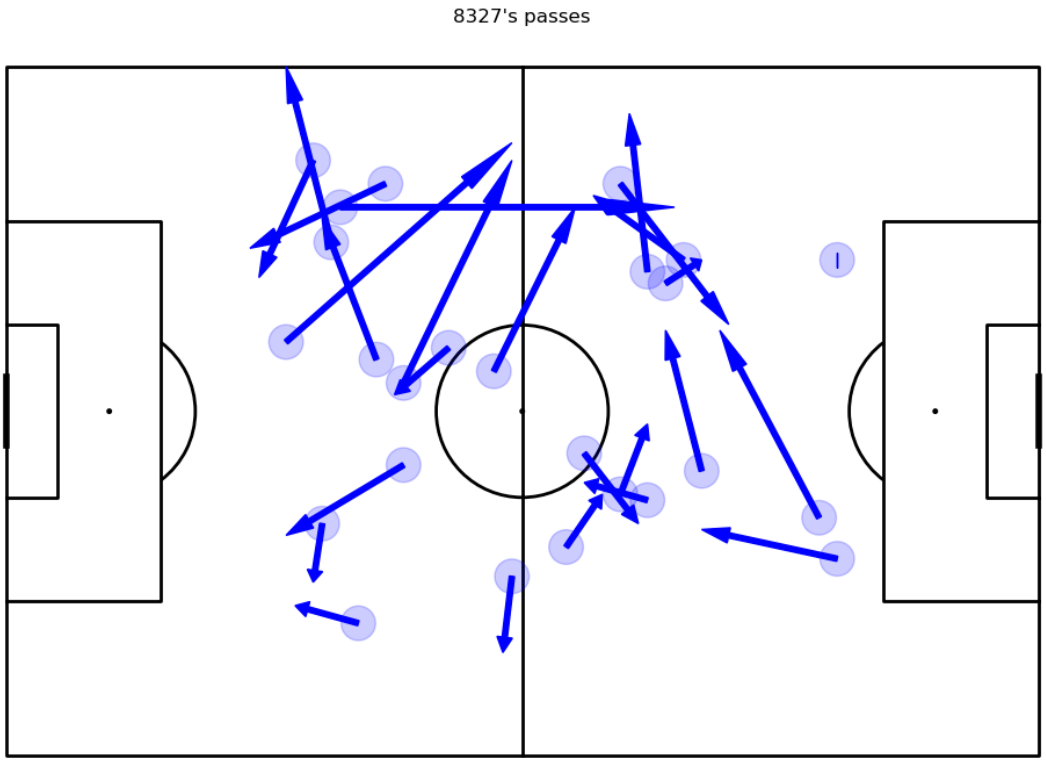
[42]:
```
df_player["type_name"].unique()
```

[42]:
```
array(['pass', 'dribble', 'shot', 'take_on', 'interception', 'cross'],
      dtype=object)
```

[43]:
```
pass_list = [
    "pass",
    "cross",
    "throw_in",
    "freekick_crossed",
    "freekick_short",
    "corner_crossed",
    "corner_short",
]
```

[44]:
```
df_player_pass = df_player.query("type_name in @pass_list")
```

[45]:
```
# drawing pitch
pitch = Pitch(line_color="black")
fig, ax = pitch.draw(figsize=(10, 7))

for _, thepass in df_player_pass.iterrows():
    x = thepass["start_x"]
    y = thepass["start_y"]
    # plot circle
    passCircle = plt.Circle((x, y), 2, color="blue")
    passCircle.set_alpha(0.2)
    ax.add_patch(passCircle)
    dx = thepass["end_x"] - x
    dy = thepass["end_y"] - y
    # plot arrow
    passArrow = plt.Arrow(x, y, dx, dy, width=3, color="blue")
    ax.add_patch(passArrow)

ax.set_title(f"{PLAYER_ID}'s passes")
fig.set_size_inches(10, 7)
plt.show()
```



1.7.1 Análise

O mapa de passes revela que esse jogador não é tão ofensivo quanto previsto pelos mapas de calor na questão anterior, fazendo mais passes próximo do meio de campo.

1.8 Questão 6

- Crie uma rede de passes de cada uma das equipes, desenvolvendo seu código em cima do dataframe SPADL. Adapte de [6].
- O que você consegue inferir sobre a formação de cada equipe? Quais jogadores de cada equipe possuem o maior grau (tem maior soma do peso das arestas)?

[46]:
```
df_spadl["type_name"].unique()
```

[46]:
```
array(['pass', 'dribble', 'take_on', 'foul', 'freekick_short', 'cross',
       'goalkick', 'clearance', 'throw_in', 'interception', 'shot',
       'keeper_save', 'tackle', 'corner_short', 'corner_crossed',
       'shot_freekick', 'freekick_crossed'], dtype=object)
```

[47]:
```
pass_list = [
    "pass",
    "cross",
    "throw_in",
    "freekick_crossed",
    "freekick_short",
    "corner_crossed",
    "corner_short",
]
```

[48]:
```
df_spadl.columns
```

[48]:
```
Index(['game_id', 'period_id', 'time_seconds', 'team_id', 'player_id',
       'start_x', 'start_y', 'end_x', 'end_y', 'original_event_id',
       'bodypart_id', 'type_id', 'result_id', 'action_id', 'type_name',
       'result_name', 'bodypart_name'],
      dtype='object')
```

[49]:
```
df_spadl.head()
```

[49]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
0	2575959	1	0.002531	3158	8327	51.45	32.64	
1	2575959	1	0.003768	3158	20438	45.15	38.08	
2	2575959	1	0.005942	3158	8306	37.80	56.44	
3	2575959	1	0.008115	3158	8306	81.90	29.92	
4	2575959	1	0.008648	3158	8306	67.20	57.80	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	action_id	\
0	45.15	38.08	180423957	0	0	1	0	
1	37.80	56.44	180423958	0	0	1	1	

```
2  81.90  29.92              NaN           0       21          1          2
3  67.20  57.80        180423960           0        7          1          3
4  75.60  57.80        180423961           0        7          1          4

  type_name result_name bodypart_name
0      pass     success          foot
1      pass     success          foot
2   dribble     success          foot
3   take_on     success          foot
4   take_on     success          foot
```

[50]:
```
df_pass_home = df_spadl.query(
    "type_name in @pass_list and result_name == 'success' and team_id ==␣
 ↪@TEAM_ID"
).copy()
```

[51]:
```
df_pass_away = df_spadl.query(
    "type_name in @pass_list and result_name == 'success' and team_id ==␣
 ↪@OTHER_TEAM_ID"
).copy()
```

[52]:
```
def define_recipient(df: pd.DataFrame):
    df["recipient_id"] = df["player_id"].shift(-1, fill_value=0).astype(int)
```

[53]:
```
define_recipient(df_pass_home)
```

[54]:
```
define_recipient(df_pass_away)
```

[55]:
```
def get_scatter_df(df: pd.DataFrame) -> pd.DataFrame:
    scatter_df = pd.DataFrame()
    for i, id in enumerate(df["player_id"].unique()):
        passx = df.loc[df["player_id"] == id]["start_x"].to_numpy()
        recx = df.loc[df["recipient_id"] == id]["end_x"].to_numpy()
        passy = df.loc[df["player_id"] == id]["start_y"].to_numpy()
        recy = df.loc[df["recipient_id"] == id]["end_y"].to_numpy()
        scatter_df.at[i, "player_id"] = id
        # make sure that x and y location for each circle representing the␣
 ↪player is the average of passes and receptions
        scatter_df.at[i, "x"] = np.mean(np.concatenate([passx, recx]))
        scatter_df.at[i, "y"] = np.mean(np.concatenate([passy, recy]))
        # calculate number of passes
        scatter_df.at[i, "no"] = df.loc[df["player_id"] == id].count().iloc[0]

    # adjust the size of a circle so that the player who made more passes
    scatter_df["marker_size"] = scatter_df["no"] / scatter_df["no"].max() * 1500
    return scatter_df
```

[56]:
```
scatter_df_home = get_scatter_df(df_pass_home)
```

[57]:
```
scatter_df_away = get_scatter_df(df_pass_away)
```

[58]:
```
def define_pairs(df: pd.DataFrame):
    df["pair_key"] = df.apply(
        lambda x: "_".join(sorted([str(x["player_id"]),␣
 ↪str(x["recipient_id"])])),
        axis=1,
    )
```

[59]:
```
define_pairs(df_pass_home)
```

[60]:
```
define_pairs(df_pass_away)
```

[61]:
```
def get_lines_df(df: pd.DataFrame):
    lines_df = df.groupby(["pair_key"]).start_x.count().reset_index()
    lines_df.rename({"start_x": "pass_count"}, axis="columns", inplace=True)
    # setting a treshold. You can try to investigate how it changes when you␣
 ↪change it.
    lines_df = lines_df[lines_df["pass_count"] > 2]
    return lines_df
```

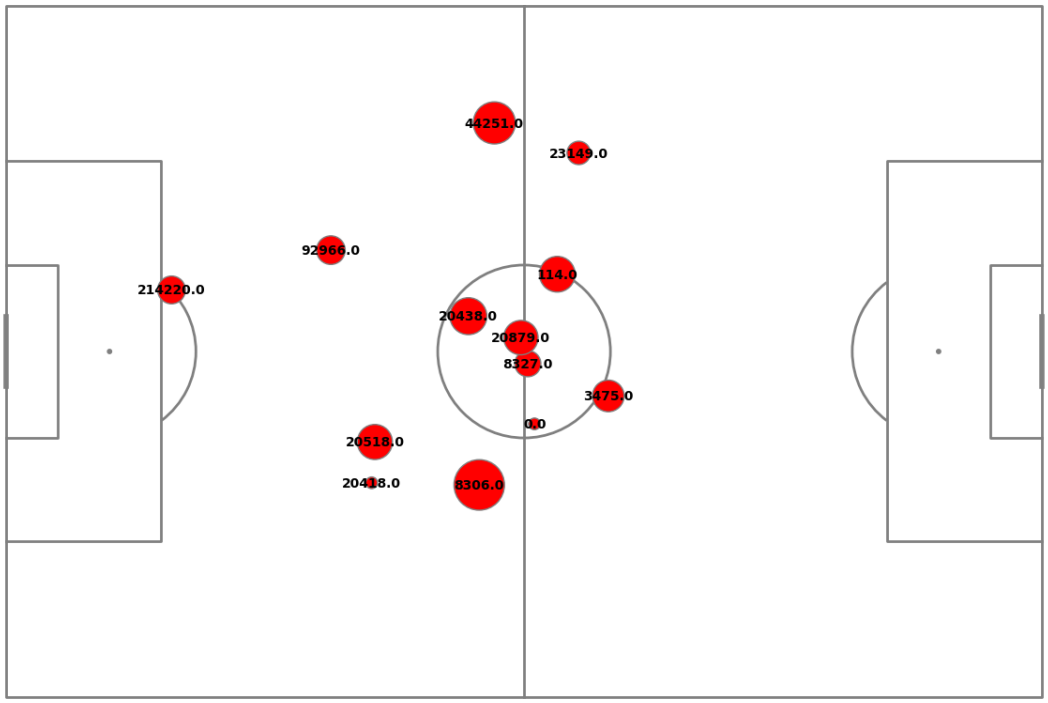
[62]:
```
lines_df_home = get_lines_df(df_pass_home)
```

[63]:
```
lines_df_away = get_lines_df(df_pass_away)
```

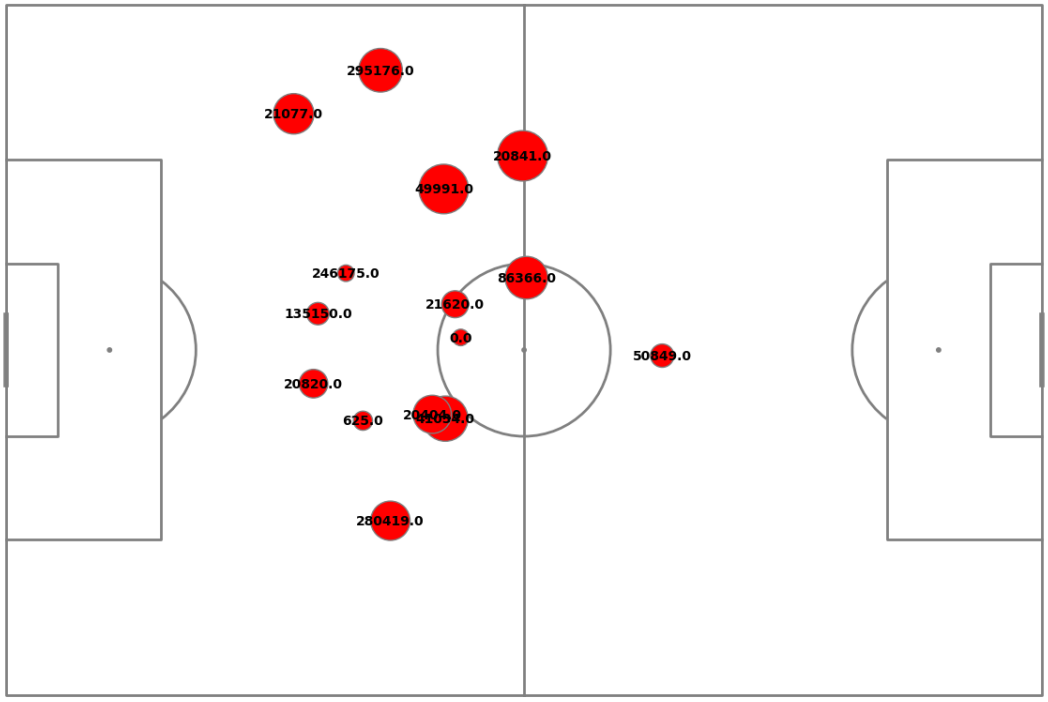
[64]:
```
def plot_nodes(scatter_df: pd.DataFrame):
    # Drawing pitch
    pitch = Pitch(line_color="grey")
    fig, ax = pitch.grid(
        grid_height=0.9,
        title_height=0.06,
        axis=False,
        endnote_height=0.04,
        title_space=0,
        endnote_space=0,
    )
    # Scatter the location on the pitch
    pitch.scatter(
        scatter_df.x,
        scatter_df.y,
        s=scatter_df.marker_size,
        color="red",
        edgecolors="grey",
        linewidth=1,
        alpha=1,
```

```
        ax=ax["pitch"],
        zorder=3,
    )
    # annotating player name
    for i, row in scatter_df.iterrows():
        pitch.annotate(
            row.player_id,
            xy=(row.x, row.y),
            c="black",
            va="center",
            ha="center",
            weight="bold",
            ax=ax["pitch"],
            zorder=4,
        )

    fig.suptitle("Nodes Locations", color="white")
    plt.show()
```

[65]:
```
plot_nodes(scatter_df_home)
```



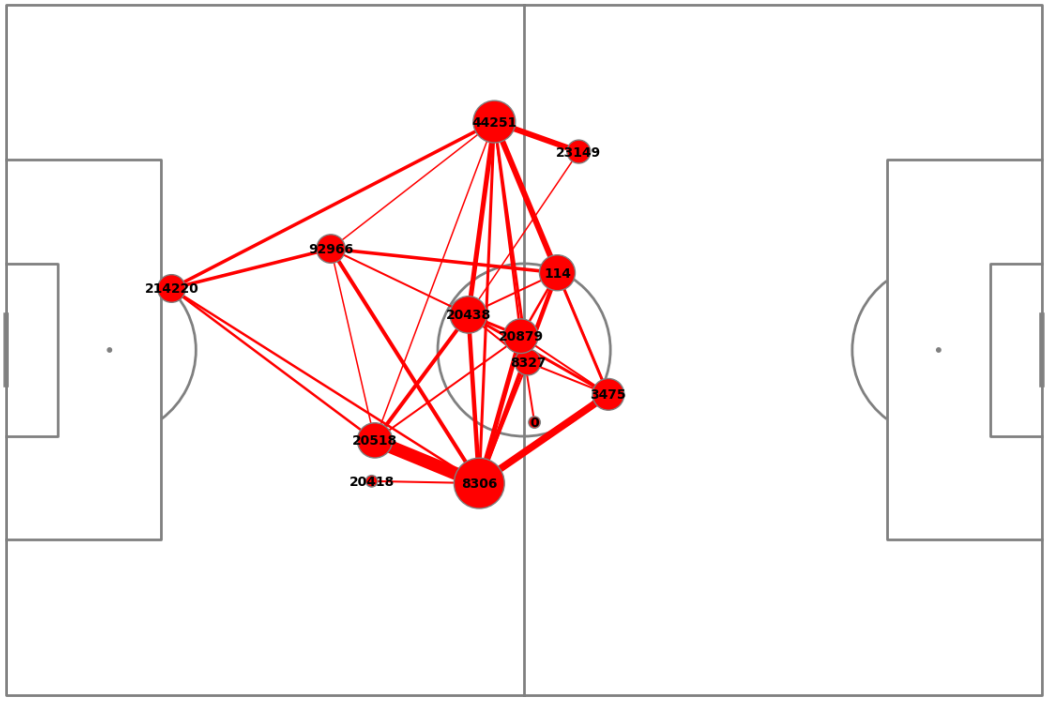
[66]:
```
plot_nodes(scatter_df_away)
```



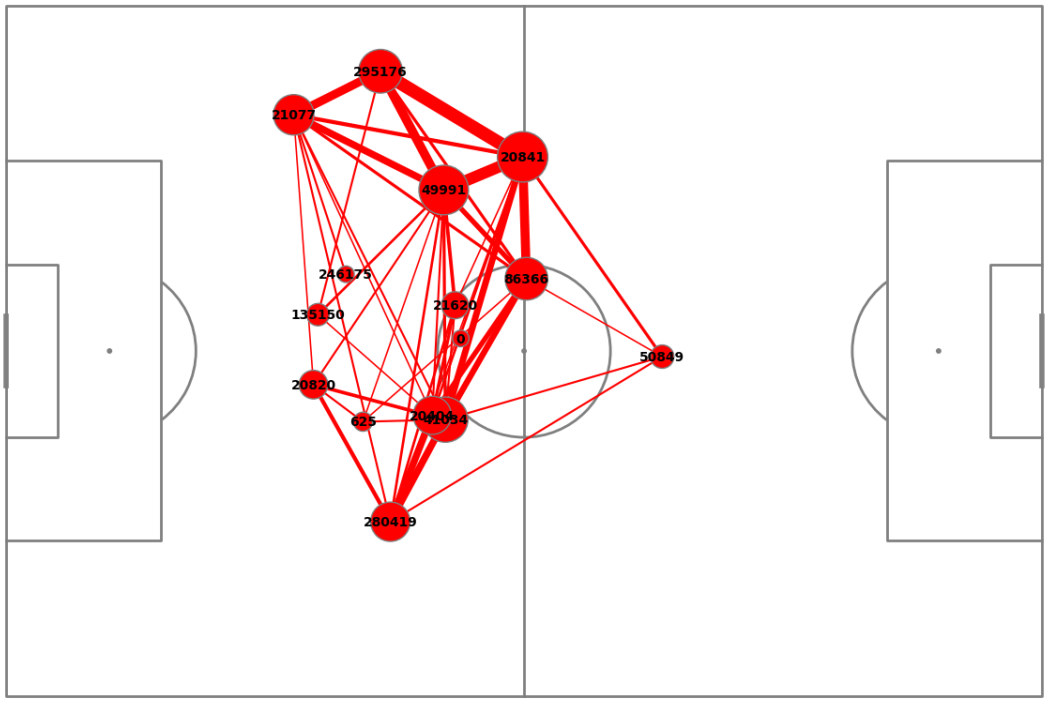
[67]:
```
def passing_netwrok(scatter_df: pd.DataFrame, lines_df: pd.DataFrame, team_id:␣
↪str):
    # plot once again pitch and vertices
    pitch = Pitch(line_color="grey")
    fig, ax = pitch.grid(
        grid_height=0.9,
        title_height=0.06,
        axis=False,
        endnote_height=0.04,
        title_space=0,
        endnote_space=0,
    )
    pitch.scatter(
        scatter_df.x,
        scatter_df.y,
        s=scatter_df.marker_size,
        color="red",
        edgecolors="grey",
```

```
        linewidth=1,
        alpha=1,
        ax=ax["pitch"],
        zorder=3,
    )
    for i, row in scatter_df.iterrows():
        pitch.annotate(
            int(row.player_id),
            xy=(row.x, row.y),
            c="black",
            va="center",
            ha="center",
            weight="bold",
            ax=ax["pitch"],
            zorder=4,
        )

    for i, row in lines_df.iterrows():
        player1 = int(row["pair_key"].split("_")[0])
        player2 = int(row["pair_key"].split("_")[1])
        # take the average location of players to plot a line between them
        player1_x = scatter_df.loc[scatter_df["player_id"] == player1]["x"]
        player1_y = scatter_df.loc[scatter_df["player_id"] == player1]["y"].
↪iloc[0]
        player2_x = scatter_df.loc[scatter_df["player_id"] == player2]["x"].
↪iloc[0]
        player2_y = scatter_df.loc[scatter_df["player_id"] == player2]["y"].
↪iloc[0]
        num_passes = row["pass_count"]
        # adjust the line width so that the more passes, the wider the line
        line_width = num_passes / lines_df["pass_count"].max() * 10
        # plot lines on the pitch
        pitch.lines(
            player1_x,
            player1_y,
            player2_x,
            player2_y,
            alpha=1,
            lw=line_width,
            zorder=2,
            color="red",
            ax=ax["pitch"],
        )

    fig.suptitle(f"Passing Network {team_id}", color="white")
    plt.show()
```

[68]:
```
passing_netwrok(scatter_df_home, lines_df_home, TEAM_ID)
```



[69]:
```
passing_netwrok(scatter_df_away, lines_df_away, OTHER_TEAM_ID)
```



- O que você consegue inferir sobre a formação de cada equipe? Quais jogadores de cada equipe possuem o maior grau (tem maior soma do peso das arestas)?

1.8.1 Análise

O time 3158 faz menos passes, com os principais jogadores sendo o 20518 e o 8306 (que fez gol). Por outro lado, a rede do time 3172 é mais "difusa", no sentindo de "uma grande quantidade de jogadores troca muitos passes". Por outro lado, esses passes estão mais próximos do meio de campo, o que não é muito bacana em termos de ataque. Nessa equipe, os jogadores com maior grau foram o 20841, 280419, 295176, entre outros.

[ ]: