luis_felipe_ramos_ferreira

April 23, 2023

1 [CDAF] Atividade 3

1.1 Nome e matrícula

Nome: Luís Felipe Ramos Ferreira Matrícula: 2019022553

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_ma
- [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].

[52]:
```
import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt
import warnings
from socceraction import spadl
import matplotsoccer
from mplsoccer import Pitch, Sbopen, VerticalPitch
warnings.filterwarnings('ignore')
```

A partida escolhida será o jogo da La Liga entre Barcelona e Real Marid, no Camp Nou.

Link para os gols da partida: https://www.youtube.com/watch?v=SPQbewtlChg

[53]:
```
BARCELONA_VS_REAL_MADRID_ID: int = 2565907
BARCELONA_ID: int = 676 # home team id
spain_df: pd.DataFrame = pd.read_json("../data/atv03/events/events_Spain.json")
match_df: pd.DataFrame = spain_df[spain_df["matchId"] ==␣
 ↪BARCELONA_VS_REAL_MADRID_ID]
```

```
players_df: pd.DataFrame = pd.read_json("../data/atv03/players/players.json",␣
 ↪encoding="iso-8859-1")
```

[54]:
```
match_df.rename(columns={"eventId": "type_id", "matchPeriod": "period_id",␣
 ↪"subEventId": "subtype_id", "matchId": "game_id", "teamId": "team_id",␣
 ↪"eventSec": "milliseconds", "playerId": "player_id", "id": "event_id"},␣
 ↪inplace=True)
match_df["period_id"] = pd.factorize(match_df["period_id"])[0] + 1
spadl_df: pd.DataFrame = spadl.wyscout.convert_to_actions(match_df,␣
 ↪BARCELONA_ID)
spadl_df
```

[54]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
0	2565907	1	0.003275	675	3321	52.50	33.32	
1	2565907	1	0.005109	675	14723	66.15	27.20	
2	2565907	1	0.007110	675	3306	73.50	15.64	
3	2565907	1	0.008912	675	3309	77.70	38.76	
4	2565907	1	0.011290	675	3915	96.60	27.20	
…	…	…	…	…	…	…	…	…
1440	2565907	2	2.968944	676	222770	5.25	34.68	
1441	2565907	2	2.982540	675	4498	0.00	68.00	
1442	2565907	2	2.983745	675	3306	12.60	34.68	
1443	2565907	2	2.984949	675	3306	12.60	40.80	
1444	2565907	2	2.986083	676	0	13.65	31.28	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
0	66.15	27.20	249644096	0	0	1	
1	73.50	15.64	249644097	0	0	1	
2	77.70	38.76	249644098	0	0	1	
3	96.60	27.20	249644099	0	0	1	
4	93.45	9.52	249644100	0	0	1	
…	…	…	…	…	…	…	
1440	0.00	68.00	249646245	0	18	0	
1441	12.60	34.68	249646064	0	5	1	
1442	12.60	40.80	NaN	0	21	1	
1443	12.60	40.80	249646066	0	8	1	
1444	0.00	68.00	249646247	0	4	0	

	action_id
0	0
1	1
2	2
3	3
4	4
…	…
1440	1440
1441	1441

```
1442       1442
1443       1443
1444       1444

[1445 rows x 14 columns]
```

[55]:
```python
# cleaning data to be compatible with matplotsoccer

spadl_df["result_name"] = spadl_df["result_id"].apply(lambda x: "success" if x␣
 ↪== 1 else "fail")
spadl_df["team_name"] = spadl_df["team_id"].apply(lambda x: "Barcelona" if x ==␣
 ↪BARCELONA_ID else "Real Madrid")
# spadl_df["player_name"] = spadl_df["player_id"].map(players_df.
 ↪set_index("wyId")["firstName"] + " " + players_df.
 ↪set_index("wyId")["lastName"])
spadl_df["player_name"] = spadl_df["player_id"].map(players_df.
 ↪set_index("wyId")["shortName"])

actiontypes: list[str] = [
    'pass',
    'cross',
    'throw_in',
    'freekick_crossed',
    'freekick_short',
    'corner_crossed',
    'corner_short',
    'take_on',
    'foul',
    'tackle',
    'interception',
    'shot',
    'shot_penalty',
    'shot_freekick',
    'keeper_save',
    'keeper_claim',
    'keeper_punch',
    'keeper_pick_up',
    'clearance',
    'bad_touch',
    'non_action',
    'dribble',
    'goalkick',
]
spadl_df["type_name"] = spadl_df["type_id"].apply(lambda x: actiontypes[x])
spadl_df.rename(columns={"milliseconds": "time_seconds"})

spadl_df
```

[55]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
0	2565907	1	0.003275	675	3321	52.50	33.32	
1	2565907	1	0.005109	675	14723	66.15	27.20	
2	2565907	1	0.007110	675	3306	73.50	15.64	
3	2565907	1	0.008912	675	3309	77.70	38.76	
4	2565907	1	0.011290	675	3915	96.60	27.20	
…	…	…	…	…	…	…		
1440	2565907	2	2.968944	676	222770	5.25	34.68	
1441	2565907	2	2.982540	675	4498	0.00	68.00	
1442	2565907	2	2.983745	675	3306	12.60	34.68	
1443	2565907	2	2.984949	675	3306	12.60	40.80	
1444	2565907	2	2.986083	676	0	13.65	31.28	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
0	66.15	27.20	249644096	0	0	1	
1	73.50	15.64	249644097	0	0	1	
2	77.70	38.76	249644098	0	0	1	
3	96.60	27.20	249644099	0	0	1	
4	93.45	9.52	249644100	0	0	1	
…	…	…	…	…	…	…	
1440	0.00	68.00	249646245	0	18	0	
1441	12.60	34.68	249646064	0	5	1	
1442	12.60	40.80	NaN	0	21	1	
1443	12.60	40.80	249646066	0	8	1	
1444	0.00	68.00	249646247	0	4	0	

	action_id	result_name	team_name	player_name	type_name
0	0	success	Real Madrid	K. Benzema	pass
1	1	success	Real Madrid	T. Kroos	pass
2	2	success	Real Madrid	Sergio Ramos	pass
3	3	success	Real Madrid	R. Varane	pass
4	4	success	Real Madrid	K. Navas	pass
…	…	…	…	…	…
1440	1440	fail	Barcelona	N\u00e9lson Semedo	clearance
1441	1441	success	Real Madrid	Lucas Vazquez	corner_crossed
1442	1442	success	Real Madrid	Sergio Ramos	dribble
1443	1443	success	Real Madrid	Sergio Ramos	foul
1444	1444	fail	Barcelona	NaN	freekick_short

[1445 rows x 18 columns]

1.4 Questão 2

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

As 5 ações escolhidas foram as 5 que resultaram no belo gol de Cristiano Ronaldo, após uma ótima jogada coletiva da equipe do Real Madrid.

[56]:
```
RONALDO_GOAL_ID: int = 231
actions_df: pd.DataFrame = spadl_df[RONALDO_GOAL_ID - 4: RONALDO_GOAL_ID + 1].
 ↪copy()
actions_df
```

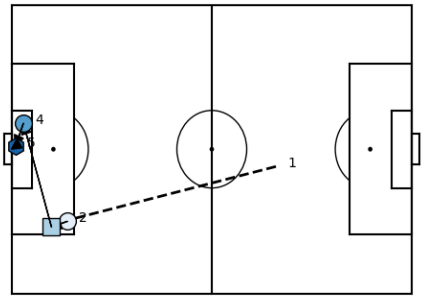
[56]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
227	2565907	1	0.829307	675	14723	69.30	29.92	
228	2565907	1	0.838265	675	3322	14.70	17.00	
229	2565907	1	0.839450	675	14723	10.50	15.64	
230	2565907	1	0.841186	675	3321	3.15	40.12	
231	2565907	1	0.842263	675	3322	1.05	34.68	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
227	13.65	17.00	249644259	0	21	0	
228	10.50	15.64	249644264	0	0	1	
229	3.15	40.12	249644265	0	1	1	
230	1.05	34.68	249644267	1	0	1	
231	0.00	34.00	249644266	0	11	1	

	action_id	result_name	team_name	player_name	type_name
227	227	fail	Real Madrid	T. Kroos	dribble
228	228	success	Real Madrid	Cristiano Ronaldo	pass
229	229	success	Real Madrid	T. Kroos	cross
230	230	success	Real Madrid	K. Benzema	pass
231	231	success	Real Madrid	Cristiano Ronaldo	shot

[57]:
```
matplotsoccer.actions(
    location=actions_df[["start_x", "start_y", "end_x", "end_y"]],
    action_type=actions_df.type_name,
    team=actions_df.team_name,
    result=actions_df.result_name == "success",
    label=actions_df[["time_seconds", "type_name", "player_name", "team_name"]],
    labeltitle=["time","actiontype","player","team"],
    zoom=False
)
```



	time	actiontype	player	team
1	0.82930732	dribble	T. Kroos	Real Madrid
2	0.8382647080000001	pass	Cristiano Ronaldo	Real Madrid
3	0.83945021	cross	T. Kroos	Real Madrid
4	0.8411863070000001	pass	K. Benzema	Real Madrid
5	0.842263114	shot	Cristiano Ronaldo	Real Madrid

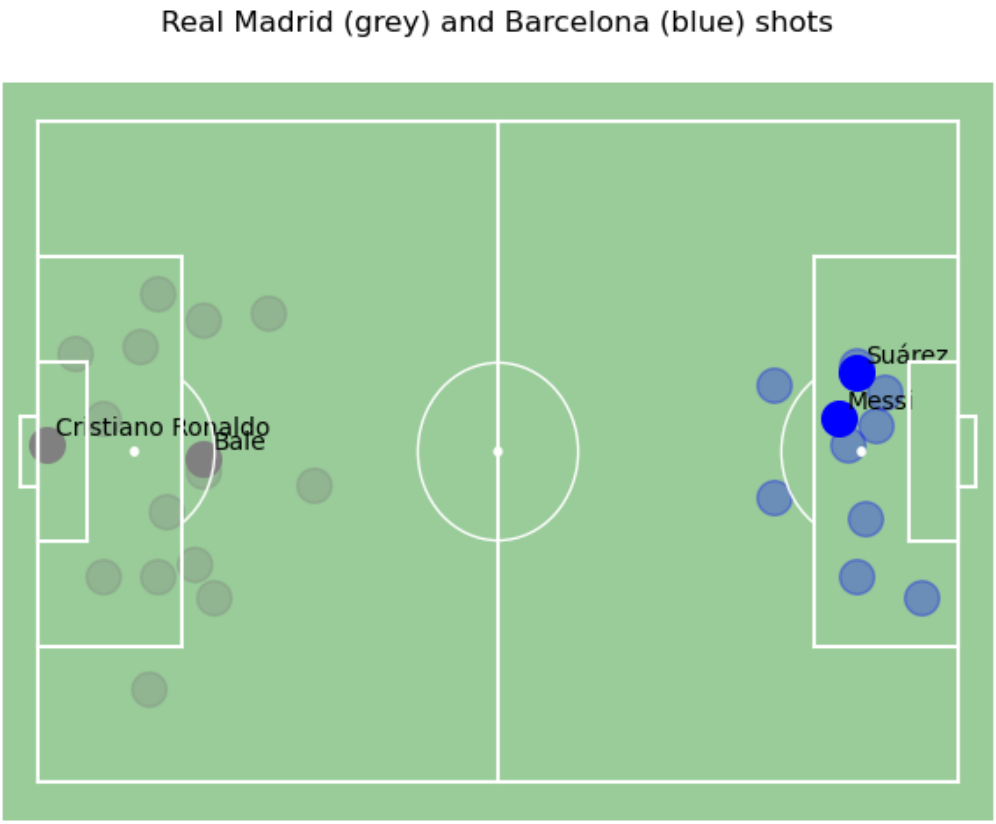
1.5 Questão 3

- 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?

Para visualização dos chutes, optei por utilizar a função de plot do *pitch* da biblioteca *matplotsoccer* por se tratar de uma visualização mais agradável e detalhada de um campo. Algumas modificações foram feitas em relação ao código disponibilizado nas referências, mas a ideia geral foi mantida

[58]:
```
team1, team2 = spadl_df.team_name.unique()
shots: pd.DataFrame = spadl_df.loc[spadl_df["type_name"].isin(
    ['shot',
    'shot_penalty',
    'shot_freekick'])].copy()

matplotsoccer.field("green",figsize=8, show=False)

# to make a better visualization
goal_scorers_ids: dict[int: str] = {
    7972: "Suárez",
    3322: "Cristiano Ronaldo",
    3359: "Messi",
    8278: "Bale"
}

# matplotsoccer spec
pitch_x_length: int = 105
pitch_y_length: int = 68

for i, shot in shots.iterrows():

    x: float = shot["start_x"]
    y: float = shot["start_y"]
    goal = shot["result_name"] == "success"
    team_name = shot["team_name"]

    circle_size = 2

    if (team_name == team1):
        if goal:
            plt.scatter(x, y, s=200, c="grey")
            plt.text(x + 1, y + 1, goal_scorers_ids[shot["player_id"]],␣
 ↪fontsize=10)
        else:
```

```
            plt.scatter(x, y, s=200, c="grey", alpha=.3)

    else:
        if goal:
            plt.scatter(x, y, s=200, c="blue")
            plt.text(x + 1, y + 1, goal_scorers_ids[shot["player_id"]])
        else:
            plt.scatter(x, y, s=200, c="blue", alpha=.3)

plt.title(f"{team1} (grey) and {team2} (blue) shots", fontsize = 12)
plt.show()
```



Analisando apenas o número bruto de chutes, o Real Madrid teve 15 oportunidades, enquanto o Barcelona teve 10. Pode-se ver no entanto que os 10 chutes do Barcelona tiveram uma concentração bem grande na região central da grande área, enquanto o Real Madrid arriscou muitas vezes de fora da área e também de regiões mais laterais da grande área.

Isso talvez possa refletir o estilo de jogo das equipes e suas principais formas de chegar ao gol. No entanto, considerando uma modelagem simples de *expected goals* em que os únicos parâmetros utilizados são a posição de onde o chute veio, nota-se que o Barcelona criou chances de maior

qualidade, chegando mais perto do gol do adversário do que o Real Madrid.

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?

[59]:
```
MESSI_ID: int = 3359
messi_df: pd.DataFrame = spadl_df.query("player_id == @MESSI_ID").copy()
messi_df.head()
```

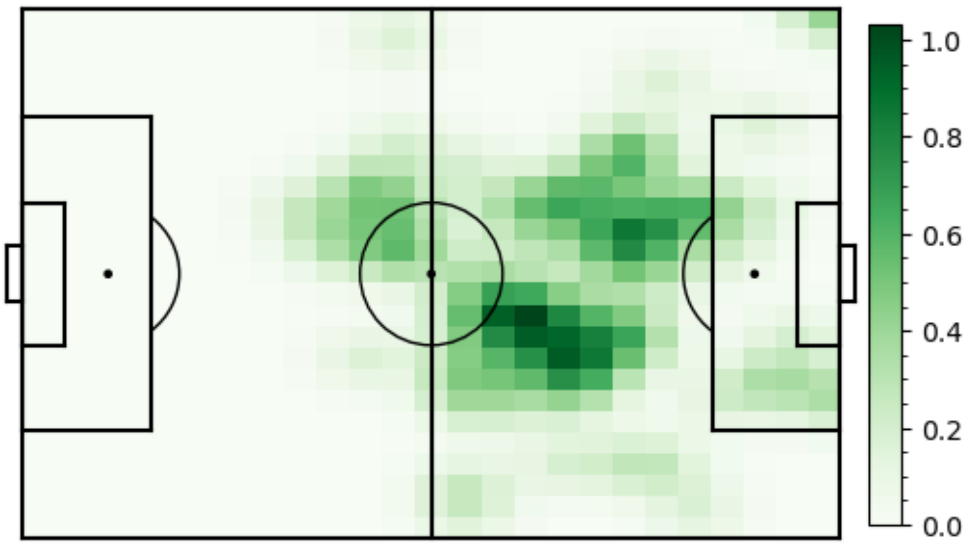
[59]:

	game_id	period_id	time_seconds	team_id	player_id	start_x	start_y	\
23	2565907	1	0.085701	676	3359	77.70	8.84	
27	2565907	1	0.092114	676	3359	70.35	8.84	
36	2565907	1	0.117242	676	3359	60.90	23.12	
38	2565907	1	0.122333	676	3359	68.25	22.44	
39	2565907	1	0.124687	676	3359	66.15	42.16	

	end_x	end_y	original_event_id	bodypart_id	type_id	result_id	\
23	84.00	6.12	249644574	0	0	1	
27	45.15	19.72	249644578	0	0	1	
36	78.75	14.96	249644587	0	0	1	
38	66.15	42.16	NaN	0	21	1	
39	72.45	38.76	249644590	0	0	1	

	action_id	result_name	team_name	player_name	type_name
23	23	success	Barcelona	L. Messi	pass
27	27	success	Barcelona	L. Messi	pass
36	36	success	Barcelona	L. Messi	pass
38	38	success	Barcelona	L. Messi	dribble
39	39	success	Barcelona	L. Messi	pass

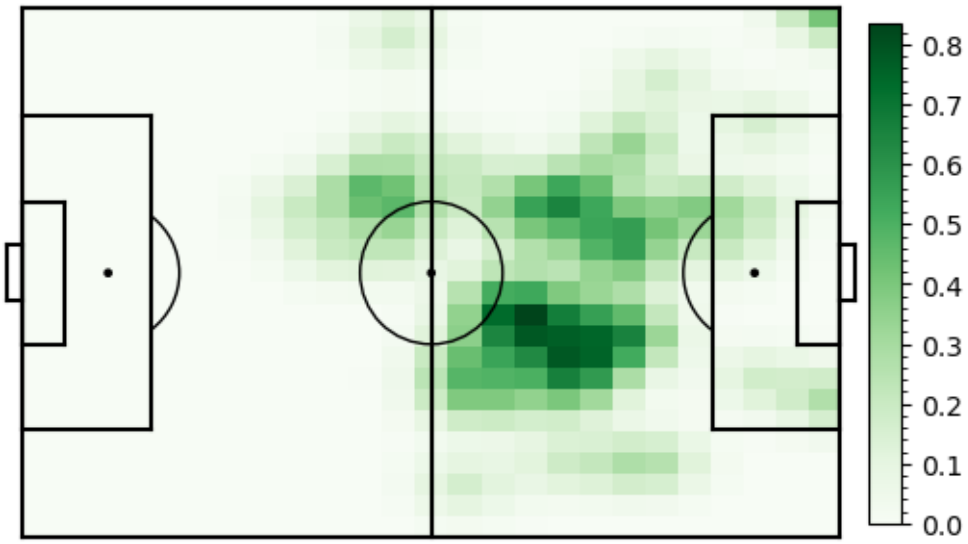
[60]:
```
# all actions heatmap
actions_heatmap = matplotsoccer.count(messi_df["start_x"], messi_df["start_y"],␣
↪n=25, m=25)
actions_heatmap = scipy.ndimage.gaussian_filter(actions_heatmap, 1)
matplotsoccer.heatmap(actions_heatmap, cmap="Greens", cbar=True)
```



[60]: <Axes: >

[61]:
```
# offensive actions heatmap
"""  we assume a offensive action is a action of one of these types:
    'pass',
    'cross',
    'throw_in',
    'freekick_crossed',
    'corner_crossed',
    'corner_short',
    'shot',
    'shot_penalty',
    'shot_freekick' """

messi_offensive_df: pd.DataFrame = messi_df[messi_df["type_name"].isin([
    'pass',
    'cross',
    'throw_in',
    'freekick_crossed',
    'corner_crossed',
    'corner_short',
    'shot',
    'shot_penalty',
    'shot_freekick'])]
```

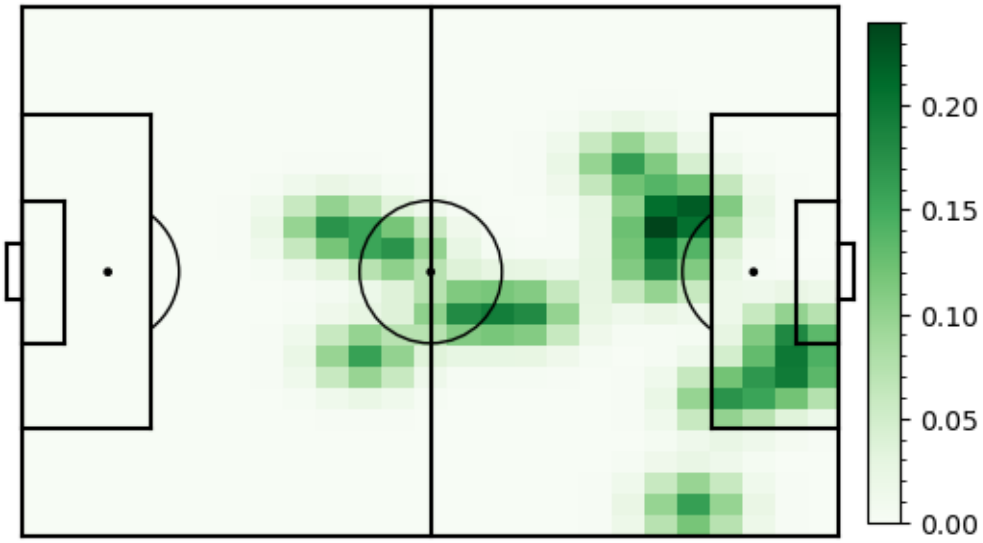
```
offensive_actions_heatmap = matplotsoccer.count(messi_offensive_df["start_x"],␣
 ↪messi_offensive_df["start_y"], n=25, m=25)
offensive_actions_heatmap = scipy.ndimage.
 ↪gaussian_filter(offensive_actions_heatmap, 1)
matplotsoccer.heatmap(offensive_actions_heatmap, cmap="Greens", cbar=True)
```



[61]: <Axes: >

[62]:
```
# defensive actions heatmap
"""  we assume a_defensive action is a action of one of these types:
    'take_on',
    'foul',
    'tackle',
    'interception',
    'keeper_save',
    'keeper_claim',
    'keeper_punch',
    'keeper_pick_up',
    'clearance' """

messi_defensive_df: pd.DataFrame = messi_df[messi_df["type_name"].isin([
    'take_on',
    'foul',
    'tackle',
    'interception',
```

```
    'keeper_save',
    'keeper_claim',
    'keeper_punch',
    'keeper_pick_up',
    'clearance'])]

defensive_actions_heatmap = matplotsoccer.count(messi_defensive_df["start_x"],␣
 ↪messi_defensive_df["start_y"], n=25, m=25)
defensive_actions_heatmap = scipy.ndimage.
 ↪gaussian_filter(defensive_actions_heatmap, 1)
matplotsoccer.heatmap(defensive_actions_heatmap, cmap="Greens", cbar=True)
```



[62]: <Axes: >

O comportamento do jogador Lionel Messi varia extremamente entre ações defensivas e ofensivas, uma vez que se trata de um atacante com grande qualidade ofensiva. Devido a isso, suas ações defensivas são quase ignoráveis. O *heatmap* plotado pode ser enganador, já que dá a impressão de que houveram mais ações defensivas do que ofensivas por parte do jogador. No entanto, nota-se que as escalas dos gráficos estão diferentes. Além disso, é importante citar como seu *heatmap* de jogadas gerais e de jogadas ofensivas é parecido, o que implica no fato de que a esmagadora maioria de suas jogadas é de cunho ofensivo (como se esperava).

Em relação ao seu comportamento, podemos ver que se trata de um jogador com posicionamento geral no lado direito do campo, que realiza movimentações e passes adentrando a área do adversário e movimentando a bola para o meio do campo. Nota-se também que se trata de um jogador com bom desempenho na bola parada, principalmente escanteios, já que sua concentração de ações nos

cantos de escanteio do campo é bem alta.

Em relação a seu aspecto defensivo, nota-se que suas ações são quase sempre no campo de ataque, provavelmente em tentativas de roubar a bola dos adversários ou interceptar passes, com o intuito de recuperar a bola para sua equipe e possivelmente gerar um contra-ataque. Nesse mesmo sentido defensivo, nota-se uma concentração grande de ações dentro do círculo central, área em que o jogador se posiciona muita das vezes em que a equipe adversária parte para o ataque.

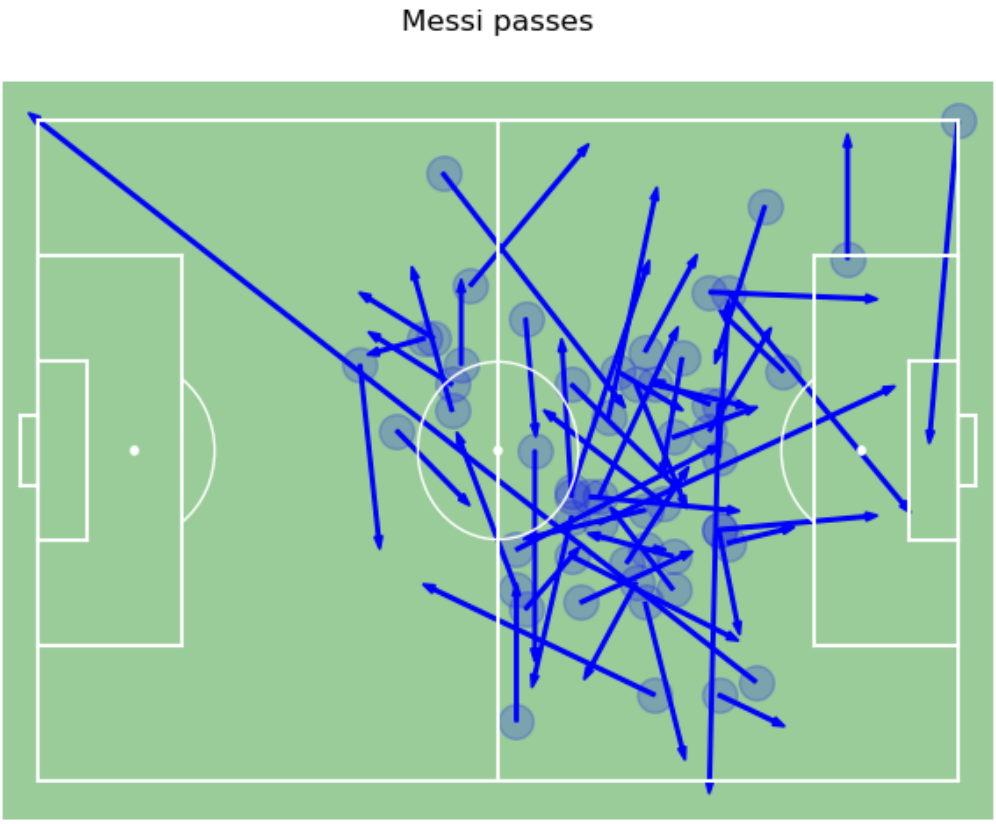
Em suma, nota-se que Messi é um jogador muito mais ofensivo do que defensivo, como era de se esperar, e suas ações se concentram principalmente em tentativas pelo lado direito do campo, na busca de redirecionar a bola para regiões centrais e/ou a grande área da equipe adversária.

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?

Assim como na questão 3, realizei algumas modificações em relação ao código disponibilizado nas referências apenas para melhorar a visualização.

[63]:
```
passes: pd.DataFrame = messi_df.loc[spadl_df["type_name"].isin(
    ['pass',
    'cross',
    'throw_in',
    'freekick_crossed',
    'freekick_short',
    'corner_crossed',
    'corner_short'])].copy()

matplotsoccer.field("green",figsize=8, show=False)

for i, thepass in passes.iterrows():
    x = thepass["start_x"]
    y = thepass["start_y"]

    plt.scatter(x, y, s=200, c="blue", alpha=.2)
    dx = thepass["end_x"] - x
    dy = thepass["end_y"] - y

    plt.arrow(x, y, dx, dy, width=.3, color="blue")

plt.title("Messi passes")
plt.show()
```



O mapa de passes do jogador ajudou a confirmar o que havia sido notado antes, isto é, Lionel Messi é um jogador que atua principalmente no lado direito do campo, gerando diversas movimentações entre esse setor e a região mais central do campo, tendo sempre como objetivo chegar mais próximo do gol.

Além disso, notamos sua grande qualidade em passes que envolvem lances progressivos de sua equipe entrando na área da equipe adversária. Isso demonstra sua grande qualidade de armador de jogadas, outra grande qulidade do jogador fora a capacidade ofensiva de chutes e dribles.

OBS: é importante notar algo nesse mapa de passes do jogador. Existe um passe mostrado em específico em que a bola foi redirecionado do campo de ataque até o canto de escanteio da área defensiva. A princípio, isso evidentemente causa estranheza. Ao buscar uma explicação, vi que o lance se tratava de um levantamento de bola na área que resultou no goleiro encaixando a bola e prosseguindo com o jogo. Por algum motivo que não fui capaz de compreender, na obtenção/tradução dos dados as coordenadas *end_x, end_y* desse lance foram computadas com valores fora do esperado, e esse passe no mínimo estranho foi observado. Naturalmente esse tipo de coisa seria tratada pelo engenheiro de dados responsável, mas achei interessante fazer essa observação e falar um pouco sobre como tais acontecimentos podem prejudicar uma boa análise dos dados. Neste caso em específico, o erro era muito óbvio, mas em outras situações isso pode passar despercebido e causar muitos problemas como geração de informação errônea para a equipe que irá interpretar os resultados.

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)?

[64]:
```
passes: pd.DataFrame = spadl_df.loc[spadl_df["type_name"].isin(
    ['pass',
    'cross',
    'throw_in',
    'freekick_crossed',
    'freekick_short',
    'corner_crossed',
    'corner_short'])].copy()

barcelona_passes_df: pd.DataFrame = passes.query('team_name == "Barcelona" and␣
 ↪result_name == "success"').copy().dropna().reset_index()
real_madrid_passes_df: pd.DataFrame = passes.query('team_name == "Real Madrid"␣
 ↪and result_name == "success"').copy().dropna().reset_index()
```

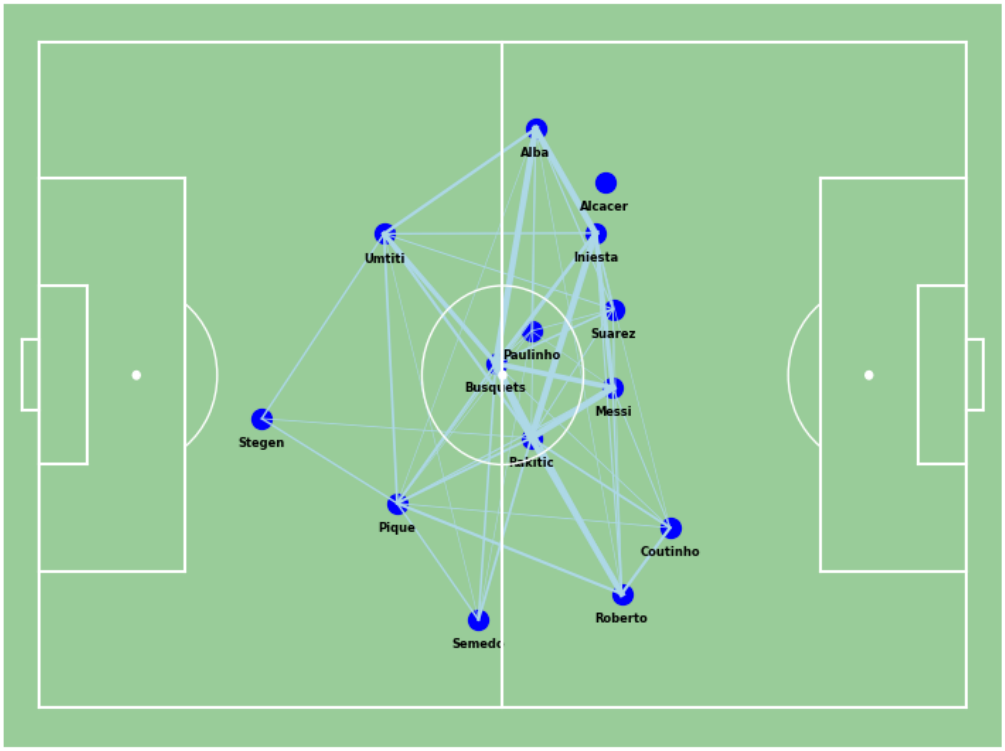
[69]:
```
barcelona_passes_df["pass_recipient_name"] = barcelona_passes_df["player_name"].
 ↪shift(-1)
barcelona_passes_df["player_name"] = barcelona_passes_df["player_name"].
 ↪apply(lambda x: str(x).split()[-1])
barcelona_passes_df["pass_recipient_name"] =␣
 ↪barcelona_passes_df["pass_recipient_name"].apply(lambda x: str(x).
 ↪split()[-1])
scatter_df = pd.DataFrame()
for i, name in enumerate(barcelona_passes_df["player_name"].unique()):
    passx = barcelona_passes_df.loc[barcelona_passes_df["player_name"] ==␣
 ↪name]["start_x"].to_numpy()
    recx = barcelona_passes_df.loc[barcelona_passes_df["pass_recipient_name"]␣
 ↪== name]["end_x"].to_numpy()
    passy = barcelona_passes_df.loc[barcelona_passes_df["player_name"] ==␣
 ↪name]["start_y"].to_numpy()
    recy = barcelona_passes_df.loc[barcelona_passes_df["pass_recipient_name"]␣
 ↪== name]["end_y"].to_numpy()
    scatter_df.at[i, "player_name"] = name
    #make sure that x and y location for each circle representing the player is␣
 ↪the average of passes and receptions
    scatter_df.at[i, "start_x"] = np.mean(np.concatenate([passx, recx]))
    scatter_df.at[i, "start_y"] = np.mean(np.concatenate([passy, recy]))
    #calculate number of passes
    scatter_df.at[i, "no"] = barcelona_passes_df.
 ↪loc[barcelona_passes_df["player_name"] == name].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)

#counting passes between players
barcelona_passes_df["pair_key"] = barcelona_passes_df.apply(lambda x: "_".
 ↪join(sorted([x["player_name"], x["pass_recipient_name"]])), axis=1)
lines_df = barcelona_passes_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]
```

[70]:
```
matplotsoccer.field("green", figsize=10, show=False)

players_passes_count: dict = dict.fromkeys(barcelona_passes_df["player_name"].
 ↪unique(), 0)

for i, row in lines_df.iterrows():
        player1 = row["pair_key"].split("_")[0]
        player2 = 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_name"] ==␣
 ↪player1]['start_x'].iloc[0]
        player1_y = scatter_df.loc[scatter_df["player_name"] ==␣
 ↪player1]['start_y'].iloc[0]
        player2_x = scatter_df.loc[scatter_df["player_name"] ==␣
 ↪player2]['start_x'].iloc[0]
        player2_y = scatter_df.loc[scatter_df["player_name"] ==␣
 ↪player2]['start_y'].iloc[0]
        num_passes = row["pass_count"]
        players_passes_count[player1] += num_passes
        players_passes_count[player2] += num_passes
        #adjust the line width so that the more passes, the wider the line
        line_width = (num_passes / lines_df['pass_count'].max() * 4)
        #plot lines on the pitch
        plt.plot([player1_x, player2_x], [player1_y, player2_y],
                        alpha=1, lw=line_width, zorder=2, color="lightblue")

plt.scatter(scatter_df.start_x, scatter_df.start_y, s=100, c="blue")
for i, row in scatter_df.iterrows():
    plt.annotate(row.player_name, xy=(row.start_x, row.start_y - 2.5),␣
 ↪c='black', va='center', ha='center', weight = "bold", size=6, zorder = 4)

print("Lista de jogadores com maior grau de passes até o menor grau de passes")
print(sorted(players_passes_count, key=players_passes_count.get, reverse=True))
```

```
plt.show()
```

```
Lista de jogadores com maior grau de passes até o menor grau de passes
['Rakitic', 'Busquets', 'Iniesta', 'Alba', 'Messi', 'Roberto', 'Pique',
'Umtiti', 'Suarez', 'Coutinho', 'Semedo', 'Paulinho', 'Stegen', 'Alcacer']
```



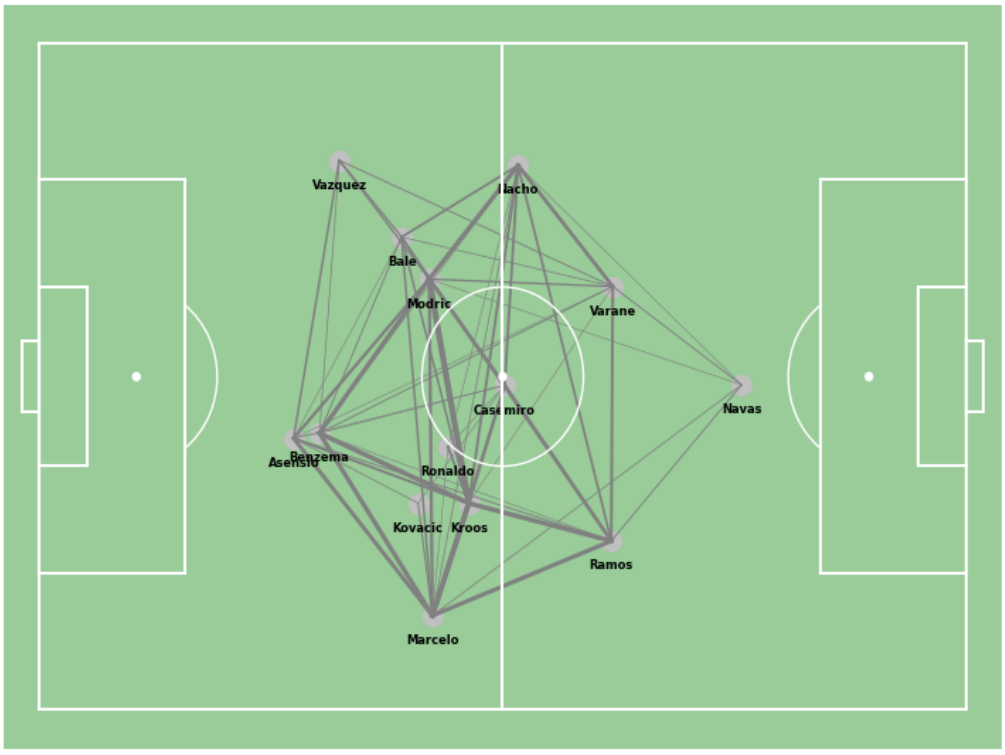
[71]:
```
real_madrid_passes_df["pass_recipient_name"] =␣
 ↪real_madrid_passes_df["player_name"].shift(-1)
real_madrid_passes_df["player_name"] = real_madrid_passes_df["player_name"].
 ↪apply(lambda x: str(x).split()[-1])
real_madrid_passes_df["pass_recipient_name"] =␣
 ↪real_madrid_passes_df["pass_recipient_name"].apply(lambda x: str(x).
 ↪split()[-1])
scatter_df = pd.DataFrame()
for i, name in enumerate(real_madrid_passes_df["player_name"].unique()):
    passx = real_madrid_passes_df.loc[real_madrid_passes_df["player_name"] ==␣
 ↪name]["start_x"].to_numpy()
    recx = real_madrid_passes_df.
 ↪loc[real_madrid_passes_df["pass_recipient_name"] == name]["end_x"].to_numpy()
```

```
    passy = real_madrid_passes_df.loc[real_madrid_passes_df["player_name"] ==␣
 ↪name]["start_y"].to_numpy()
    recy = real_madrid_passes_df.
 ↪loc[real_madrid_passes_df["pass_recipient_name"] == name]["end_y"].to_numpy()
    scatter_df.at[i, "player_name"] = name
    #make sure that x and y location for each circle representing the player is␣
 ↪the average of passes and receptions
    scatter_df.at[i, "start_x"] = np.mean(np.concatenate([passx, recx]))
    scatter_df.at[i, "start_y"] = np.mean(np.concatenate([passy, recy]))
    #calculate number of passes
    scatter_df.at[i, "no"] = real_madrid_passes_df.
 ↪loc[real_madrid_passes_df["player_name"] == name].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)

#counting passes between players
real_madrid_passes_df["pair_key"] = real_madrid_passes_df.apply(lambda x: "_".
 ↪join(sorted([x["player_name"], x["pass_recipient_name"]])), axis=1)
lines_df = real_madrid_passes_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]
```

[72]:
```
matplotsoccer.field("green", figsize=10, show=False)

players_passes_count: dict = dict.fromkeys(real_madrid_passes_df["player_name"].
 ↪unique(), 0)

plt.scatter(scatter_df.start_x, scatter_df.start_y, s=100, c="silver")
for i, row in scatter_df.iterrows():
    plt.annotate(row.player_name, xy=(row.start_x, row.start_y - 2.5),␣
 ↪c='black', va='center', ha='center', weight = "bold", size=6, zorder = 4)

for i, row in lines_df.iterrows():
        player1 = row["pair_key"].split("_")[0]
        player2 = 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_name"] ==␣
 ↪player1]['start_x'].iloc[0]
        player1_y = scatter_df.loc[scatter_df["player_name"] ==␣
 ↪player1]['start_y'].iloc[0]
        player2_x = scatter_df.loc[scatter_df["player_name"] ==␣
 ↪player2]['start_x'].iloc[0]
```

```
        player2_y = scatter_df.loc[scatter_df["player_name"] ==␣
↪player2]['start_y'].iloc[0]
        num_passes = row["pass_count"]
        players_passes_count[player1] += num_passes
        players_passes_count[player2] += num_passes
        #adjust the line width so that the more passes, the wider the line
        line_width = (num_passes / lines_df['pass_count'].max() * 3)
        #plot lines on the pitch
        plt.plot([player1_x, player2_x], [player1_y, player2_y],
                        alpha=1, lw=line_width, zorder=2, color="grey")

print("Lista de jogadores com maior grau de passes até o menor grau de passes")
print(sorted(players_passes_count, key=players_passes_count.get, reverse=True))

plt.show()
```

```
Lista de jogadores com maior grau de passes até o menor grau de passes
['Kroos', 'Modric', 'Marcelo', 'Ramos', 'Nacho', 'Benzema', 'Casemiro',
'Asensio', 'Bale', 'Varane', 'Vazquez', 'Navas', 'Ronaldo', 'Kovacic']
```



O jogador com o maior volume de passes na equipe do Barcelona foi Ivan Rakitić, enquanto na

equipe do Real Madrid o jogador foi Toni Kroos. Essa constatação faz sentido, dito que ambos são meio-campistas que possuem como qualidade um bom posicionamento e uma boa distribuição de bola pelo campo. O estilo de jogo de ambos times parece rodar em torno de jogadores desse tipo.

Mais especificamente, podemos ver que a equipe do Barcelona possui uma rede de passes muito intensa nos setores central e direito do campo, muito provavelmente devido ao fato de que é nessas áreas que atua Lionel Messi, o principal pilar de construção de jogadas do time. Vale destacar também outros jogadores que possuem um alto grau na rede de passes, como por exemplo Sérgio Busquets e Andrés Iniesta, ambos meio-campistas excpecionais.

Já sobre o Real Madrid, podemos notar uma rede de passes similar, com o jogo rodando em volta dos meio campistas, como Toni Kroos, Luka Modric e Nacho. É interessante notar na rede de passes do Real Marid o posicionamento de alguns jogadores como Marcelo. Apesar de ser um defensor/lateral-esquerdo, Marcelo possui um posicionamento e uma rede de passes intensa na parte ofensiva do campo, mostrando sua característica de subir constantemente pro ataque e realizar boas jogadas no setor esquerdo do campo, muitas das vezes com Cristiano Ronaldo.