Clusterización y Métricas Avanzadas para Segmentación de Futbolistas y Modelos de Juego

TFM Máster Data Science | EFBS 2021/2022

ALEJANDRO FERNÁNDEZ RODRÍGUEZ

Agradecimientos

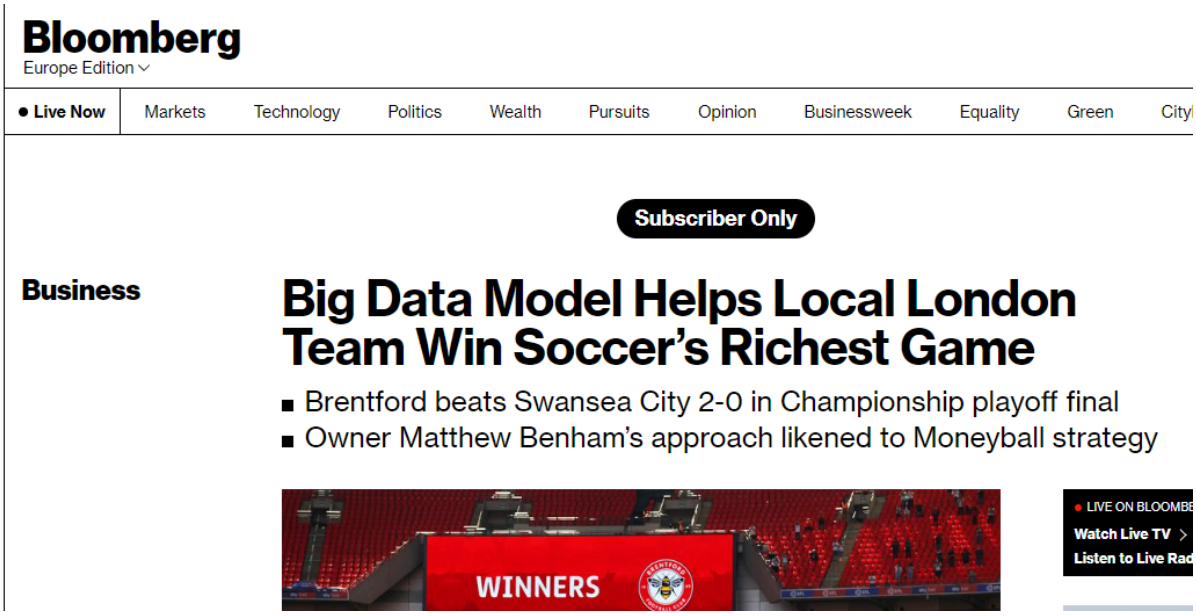
Friends of Tracking, MarkRStats, FCPYTHON, github.com/douglasbc/, towardsdatascience, twitter.com/Anuraag027/

A Mada, por motivarme y apoyarme

A Bárbara, por Serenity by Jan

Fuentes

Understat | FBRef+Statsbomb | Transfermarkt



FÚTBOL >

De la partícula de dios al balón

Físicos teóricos, matemáticos y programadores impulsan los departamentos de 'big data' de algunos de los principales clubes de fútbol de Europa, que les atribuyen mejoras en el juego y los fichajes

Introducción y Motivación del Proyecto

En el pasado mes de febrero se cumplió una década desde el estreno en España de una de las obras que han marcado el deporte mundial, en lo que a la ilustración de métodos basados en

datos se refiere. **Moneyball** se instaló para siempre como un concepto que, como pasa con el término "Big Data", no huye de lugares comunes al mismo tiempo que se acuña como palabra para denominar toda estrategia basada en el tratamiento y analítica de datos aplicado a nivel práctico en el entorno deportivo. La gesta de los Oackland A's a comienzos de siglo en la MLB y su fantástica rubricación en la gran pantalla, con Brad Pitt como protagonista, marcó sin duda un hito en la democratización de una nueva forma de entender el juego, alejándose de tópicos y juicios perceptivos y centrada en la maximización de los recursos a partir de lo que era, en aquel momento, una utopía: fiarlo todo a los datos, tratados, además, por personas que no pertenecían al sector, como físicos, ingenieros o economistas.

La disrupción, evidentemente, no fue inmediata y ha encontrado todo tipo de trabas. Mientras en EEUU el baloncesto, fútbol americano y beisbol profesional llevan una década instalados en este tipo de nueva "dictadura" (sus críticos, los más románticos, no dudan en decir que el juego ahora es mucho menos impredecible e imaginativo -no les falta razón, pero hablamos de sectores cuyos agentes son auténticos transatlánticos mediáticos y financieros, hoy por hoy lo que importa es competir, maximizar recursos limitados y ganar-). En Europa, el boom de los datos en el deporte rey se dio a mediados de la década pasada en Reino Unido, Alemania y los países escandinavos. El sector es aún inmaduro y con un margen a explorar descomunal, tanto a nivel cognitivo como tecnológico.

En este contexto, son muy concretos, pero también extremos, los ejemplos que encontramos sobre la aplicación de estos métodos al reclutamiento de futbolistas en nuestro continente. Matthew Benham, ex-pupilo de otro pionero como el dueño del Brighton Tony Bloom, se obsesionó con desarrollar modelos predictivos de apuestas que acabaron haciéndole millonario. Esos mismos modelos están hoy en producción en el club de su infancia, el Brentford, al que ha llevado con esta filosofía a la Premier League. Aún hoy, ya en la élite y con presupuestos superiores a los que prácticamente cualquier club en Europa puede manejar, siguen aplicando sus modelos. La semilla surgió con una premisa: dejar de fijarse en las métricas tradicionales de goles, pases, victorias o derrotas, y desarrollar nuevas métricas e indicadores -suyos son los archifamosos Expected Goals- que señalaban el progreso del equipo según otros factores. Esto permitió al Brentford detectar jugadores infravalorados que pasaban bajo el radar y construir un modelo de negocio alrededor de ellos para registrar ganancias récord. Exactamente igual que Billy Beane en *Moneyball*.

Tales plusvalías, y el descubrimiento de jugadores hoy de primer nivel mundial, como Said Benrahma, Ollie Watkins o Ivan Toney, jamás se habrían producido sin la explotación de datos a gran escala y el uso de algoritmos.

Brentford's Transfer Market Activity							
Player	**Purchased**			**Sold**			**Profit**
	Year	Club	Fee	Year	Club	Fee	
Ollie Watkins	2017	Exeter City	£1.8m	2020	Aston Villa	£33m	**£31.2m**
Said Benrahma	2018	OGC Nice	£2.7m	2020	West Ham	£30m	**£27.3m**
Neal Maupay	2017	Saint-Ettiene	£1.8m	2019	Brighton	£20m	**£18.2m**
Andre Gray	2014	Luton Town	£0.5m	2015	Burnley	£12m	**£11.5m**
Scott Hogan	2014	Rochdale	£0.8m	2017	Aston Villa	£12m	**£11.2m**
Chris Mepham	2016	Brentford B	£0	2019	Bournemouth	£11m	**£11m**
Ezri Konsa	2018	Charlton	£2.5m	2019	Aston Villa	£12m	**£9.5m**
Ryan Woods	2015	Shrewsbury	£1m	2018	Stoke	£6.5m	**£5.5m**
Nico Yennaris	2014	Arsenal	£0.2m	2019	Beijing Guoan	£5m	**£4.8m**
Jota	2014	Celta Vigo	£1.5m	2017	Birmingham	£6m	**£4.5m**
James Tarkowski	2014	Oldham	£0.3m	2016	Burnley	£4.5m	**£4.2m**
John Egan	2016	Gillingham	£0.4m	2018	Sheffield Utd	£4m	**£3.6m**
			£13.5m			£156m	**£142.5m**

Democratizando todavía más los métodos de Benham -con algoritmos y datos origen más modestos-, el objetivo del presente proyecto es generar un modelo que perfile las características de los jugadores de las cinco grandes ligas europeas según métricas avanzadas y, segmentando el modelo de juego de los equipos, poder asociar, a cada club, los futbolistas que, para cada posición en el campo, más se aproximen a su modelo de juego. El propósito es añadir la variante del estilo de juego de los equipos -el propio y los demás-, midiéndolo y cuantificándolo para poder encontrar a aquellos jugadores que más se aproximen, basándonos en métricas avanzadas, a los números del equipo que estemos analizando. Serán esos futbolistas los que, potencialmente, más se adecúen a las necesidades por las que dicho equipo acude al mercado, pues serán piezas que requerirán un proceso de adaptación más corto y conocerán mecanismos similares de juego, pues vendrán de conjuntos que, tácticamente, muestran simitudes.

Existen, disponibles para consulta pública, una gran cantidad de modelos que ofrecen la posibilidad de encontrar los jugadores que más se aproximan a las métricas de otro en concreto -en este mismo proyecto podrán encontrar uno, de hecho, que combinaremos con la analítica de los equipos para construir nuestro modelo final-. Dichos análisis pueden ser de gran validez a nivel de gestión en clubes cuando, en contexto de mercado, es necesario reemplazar a un jugador que ha dejado el club o ha bajado su rendimiento. Sin embargo, esas situaciones se dan con relativa poca asiduidad en el mundo real, siendo lo más habitual la necesidad de encontrar en el mercado la corrección a una debilidad actual o bien la mejora de alguna posición en concreto.

El plus de este modelo radica, pues, en su capacidad para añadir el factor de las características de los equipos a la hora de valorar cuál es el futbolista más adecuado en el que invertir en el mercado. Ello ayudaría, como explico más arriba, en el proceso de reclutamiento a reducir el grupo de jugadores que son candidatos a terminar siendo un fichaje, limitando el error de la parcela deportiva y asegurando que aspectos como la adaptación táctica sean más sencillos.

Modelos como el planteado no son de aplicación tan clara en clubes de gran magnitud. Los Real Madrid o Manchester City no precisan de modelos analíticos para descubrir donde está el valor,

pues su nivel atrae -y pueden permitirse a nivel económico- el mayor talento disponible. No así la mayoría de clubes, que saben que su mercado potencial es enorme y, además, cuentan con el hándicap de que un posible error penaliza mucho más, dado que los recursos económicos son mucho más limitados.

Las variables que se valoran en este modelo y sus resultados son de valor para una dirección deportiva, pero ni pueden ser un veredicto final -ayudarían a reducir la muestra y a ofrecer un background que facilite las decisiones- ni aseguran el éxito. Como ejemplo, el fichaje de Romelu Lukaku por el Chelsea, que sin duda habría sido recomendado también por este modelo. No sólo sus números puros le avalan como un extraordinario delantero, sino que el estilo de juego del Chelsea y de su anterior equipo, el Inter, tenían unas similitudes muy claras. Para más inri, Lukaku vivía el mejor momento de su carrera y el Chelsea había sido campeón de Europa con un equipo coral al que le faltaba exactamente un delantero como él. Era un *match made in heaven* por el que merecía la pena abonar más de 100 millones de euros + 12 millones al año por cinco temporadas, pero los buenos augurios se quedaron en septiembre. Este mes de julio, el belga, fichaje más caro de la historia del club, hará el camino de vuelta a Milán dejando a cambio una cifra irrisoria.

> Mi sueño es que la opinión no tenga ningún peso en las decisiones, que sean solo los datos
>
> James Cryne (Barnsley)

Tras los preámbulos, vayamos al análisis.

El presente notebook, tanto a nivel de código como de explicación, se estructura del siguiente modo:

- **ESTRUCTURA DEL PROYECTO**
- **IMPORTACIÓN Y LIMPIEZA DE DATOS**
- **SEGMENTACIÓN DE EQUIPOS**
    - `- Ejecución de la función de Clusterización`
    - `- Análisis de los Resultados`
- **PERFILANDO A LOS JUGADORES**
    - `- Ejecución de la función de Clusterización`
    - `- Análisis de los Resultados`
- **FUNCIONES DE DISTANCIA**
    - `- Player_Similarities`
    - `- Team_Mapping`
- **APLICACIÓN A CASOS REALES**

Estructura del Proyecto

Los datos surgen de dos fuentes que están en web, tanto fbref.com como understat.com. Además, se ha creado un procedimiento para descargar un dataset de kaggle que a su vez procede de Transfermarkt, para poder así tener la info del valor de mercado de los jugadores y algunos datos posicionales mas detallado. El hecho de tener varias fuentes implica valorar los índices de cada una de las webs, en especial para los jugadores.

En el presente trabajo, se valoran exclusivamente los futbolistas de campo que han tenido presencia en la temporada recién finalizada. De todos ellos, sólo aquellos que hayan disputado un mínimo de minutos figurarán en los modelos. Partimos con los datos de los jugadores de las cinco grandes ligas europeas y sus equipos. Habrá, pues, un dataset de equipos y otro de jugadores.

Todos los datos están normalizados a 90 minutos o bien en tasa porcentual. A continuación se explican algunas de las métricas avanzadas:

- **Expected Goals (xG)**: indicador estadístico que asigna una probabilidad de que una ocasión -disparo- sea gol en función de las características de la jugada
- **Expected Assists (xA)**: mide la probabilidad de que un pase se convierta en una asistencia de gol. El modelo premia a los jugadores que dan un pase en zonas de peligro, independientemente de si el receptor realiza o no un disparo.
- **Expected Goals Chain (xGChain)**: valora la participación del jugador en jugadas que acaban en ocasión.
- **Expected Goals Buildup (xGBuildup)**: valora la participación del jugador en jugadas que acaban en ocasión sin tener en cuenta la acción de tiro y pase previo.
- **PPDA**: medidor de intensidad y presión. Pases por acción defensiva
- **OPPDA**: medidor de intensidad y presión. Pases del rival por acción defensiva propia
- **Power** = xGBuildup * xGChain (salvo si se drata de un pivote o un defensa, en ese caso será =xGBuildup)

Además, a nivel de nomenclatura, aclarar que existen dos tipos de asignaciones de futbolistas a posiciones en el campo. **PosE** realiza una asignación general, mientras que la detallada se encuentra en la columna **POS**, como veremos mas abajo.

El proyecto cuenta con una serie de librerías, que se procede a detallar:

- **auto_plotting.py** : pensada para ejecutar de una vez todos los gráficos que me gustaría automatizar, cuyas funciones están definidas en squad_plotting.py y players_plotting.py
- **clustering.py** : ejecuta el modelo, que incluye las funciones de clustering + las de distancia
- **fbref_scrap.py**, **tmarkt_scraper.py**, **understat_match_scrap.py**, **understat_teams.py** y **understat_season_scrap.py** tienen las funciones de extracción de datos
- **update.py** ejecuta (de momento) todo el proceso de scraping y limpieza
- **modeling_functions.py** tiene algunas funciones para calcular métricas y mapear/limpiar los datos scrapeados
- **modeling.py** ejecuta esas funciones, une las tablas, limpia los datos y los prepara para el análisis
- **utils.py**: tiene la función que determina la temporada y crea las rutas en el path

Importación y Limpieza de Datos

In [1]:
```
import pandas as pd
import numpy as np
import utils as utils
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

```
from sklearn.preprocessing import OneHotEncoder
from yellowbrick.cluster.elbow import kelbow_visualizer
from sklearn.preprocessing import MinMaxScaler
import matplotlib.cm as cm
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
from sklearn.feature_selection import VarianceThreshold
from sklearn.ensemble import RandomForestClassifier
from rapidfuzz.distance import Levenshtein
from scipy.spatial import distance
import Players_plotting as pp
import Squad_plotting as sp
import time
```

Tras importar las librerías pertinentes, necesitaremos aplicar **utils** para obtener la temporada. Si no se especifica nada, tomará la actual.

In [2]:
```
seas = utils.get_season()
s = int(seas[0][:4])
actual = seas[1]

print('Season: ',s)

if actual==1:
    fbref_file= 'Current'
else:
    fbref_file=str(s)+'-'+str(int(s)+1)
```

```
Season:  2021
```

Se definen las rutas para facilitar la posterior importación.

In [3]:
```
understat = os.path.join(os.getcwd(),"understat_scraping","datasets")
transfermarkt = os.path.join(os.getcwd(),"tmarkt_scraping")
ruta = os.path.join(os.getcwd(),"DATA",fbref_file)
```

In [4]:
```
squad = pd.read_csv(ruta+'/squad_stats.csv',decimal=',',sep=';')
for i in squad.columns:
    if 'DIF_' in i:
        squad.drop(i,inplace=True,axis=1)
cols= squad.dtypes

print('Número de filas y columnas en Squad dataset: ',squad.shape)
squad['Comp'].value_counts()
```

Out[4]:
```
Número de filas y columnas en Squad dataset:  (98, 654)
La Liga           20
Ligue 1           20
Premier League    20
Serie A           20
Bundesliga        18
Name: Comp, dtype: int64
```

In [5]:
```
squad.head(5)
```

Out[5]:

	Squad	Rk	Comp	#Pl	Age	Poss	MP	Starts	Min	90s	...	poss_idx	succpress_idx

	Squad	Rk	Comp	#Pl	Age	Poss	MP	Starts	Min	90s	...	poss_idx	succpress_idx
0	Alaves	1	La Liga	33.0	24.0	41.8	38.0	418.0	3420.0	38.0	...	0.41	0.11
1	Angers	2	Ligue 1	30.0	22.0	48.7	38.0	418.0	3420.0	38.0	...	0.47	0.11
2	Arminia	3	Bundesliga	26.0	20.0	40.5	34.0	374.0	3060.0	34.0	...	0.40	0.13
3	Arsenal	4	Premier League	27.0	17.0	53.2	38.0	418.0	3420.0	38.0	...	0.55	0.10
4	Aston Villa	5	Premier League	31.0	21.0	46.5	38.0	418.0	3420.0	38.0	...	0.47	0.10

5 rows × 654 columns

Importamos los datos de equipos (98, los cinco de las grandes ligas europeas). Hacemos lo propio con los jugadores, y calculamos el número mínimo de minutos. Con la temporada ya acabada, este debe ser la media disputada por todos los jugadores por la mitad de la desviación típica.

In [6]:
```
players = pd.read_csv(ruta+'/player_stats.csv',decimal=',',sep=';')
min_minutes = int(players['Min'].mean()-0.5*players['Min'].std())


print('Mínimo de minutos para ser computado en los modelos analíticos: ',min_minutes
```

```
Mínimo de minutos para ser computado en los modelos analíticos:  907
```

In [7]:
```
players.isna().any().value_counts()
```

Out[7]:
```
False    261
True       2
dtype: int64
```

In [8]:
```
for i in players.columns:
    if players[i].isna().any()==True:
        print(i)
```

```
SQUAD_DIF_xPTS
SQUAD_DIF_deep
```

Las columnas que contienen DIF *miden la diferencia entre los valores de esta temporada y la anterior. Si comienzan por SQUAD* implica que corresponden a equipos. Es lógico que haya valores nulos en este tipo de métricas en el dataset de jugadores, pues hay equipos que la temporada pasada no estaban en la máxima categoría, así que no tienen métricas que restar a las de este año.

In [9]:
```
print('Número de filas y columnas en Players dataset: ',players.shape)
players.head()
```

```
Número de filas y columnas en Players dataset:  (2387, 263)
```

Out[9]:

	idx	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	...	xG-xga

	idx	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	...	xG-xga	
0	90f87966	908	Jose Fonte	POR	DF	Lille	Ligue 1	39	1983.0	38.0	...	-1.02	https://fb
1	2389cdc2	479	Matty Cash	POL	DF	Aston Villa	Premier League	25	1997.0	38.0	...	-1.23	https://fb
2	5313ed43	902	Seko Fofana	CIV	MF	Lens	Ligue 1	27	1995.0	38.0	...	-0.92	https://fb
3	2928dca2	548	Conor Coady	ENG	DF	Wolves	Premier League	29	1993.0	38.0	...	-1.54	https://fb
4	0ff630d0	2896	Akim Zedadka	ALG	DF	Clermont Foot	Ligue 1	27	1995.0	38.0	...	-1.35	https://fb

5 rows × 263 columns

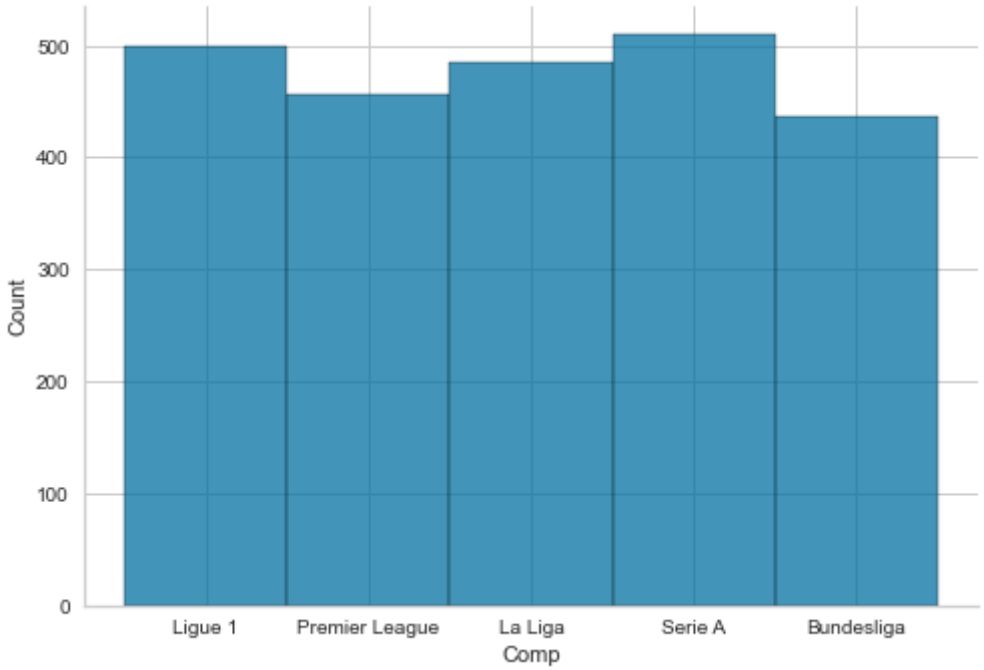
In [10]:
```
squad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98 entries, 0 to 97
Columns: 654 entries, Squad to Logo
dtypes: float64(613), int64(17), object(24)
memory usage: 500.8+ KB
```

In [11]:
```
players.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2387 entries, 0 to 2386
Columns: 263 entries, idx to DIF_npxG/Sh
dtypes: float64(246), int64(5), object(12)
memory usage: 4.8+ MB
```

In [12]:
```
sns.histplot(players.Comp)
sns.despine()
```



Segmentación de Equipos

A continuación se definen las variables que se valorarán de cara a segmentar el modelo de juego de los equipos. Dicho análisis se acometerá diferenciando entre cuatro categorías:

- Sistema, entendido como la organización del equipo en el terreno de juego
- Defensa y Presión
- Elaboración en Ataque
- Creación de Oportunidades y Definición

In [13]:
```
# Data filtering for each category in squads
general = ['Squad','Comp','stat']
system_cols = []
for i in squad.columns:
    if ('Use%_' in i and 'k' in i):
        system_cols.append(i)

def_cols = list(set(
    ['%ag_Press','Standard_Dist_ag','Pressures_hPress','Touches_htouch_ata_ag','pres
     'idxpos_p','idxpos_tk','ppda_coef','PassTypes_Live_ag','Expected_npxG/Sh_ag',
'PassTypes_Crs_ag','PassTypes_TB_ag','Total_Cmp%_ag','AerialDuels_Won','Carries_Prog
 'Carries_100touches_ag','Receiving_Prog_100Rec_ag','SCA_100touchesAtt3rd_ag','PPT3%
 'Passes_Launch%_ag','GoalKicks_Launch%_ag']))

buildup_cols = list(set(
    ['oppda_coef','idxpos_t','Carries_100touches_ag','Crs_100_Passes','Crs_ProgPass'
     'Carries_Prog_100_Inc','Carries_100touches','Receiving_Prog_100Rec','Poss',
     'PassTypes_Sw','PassTypes_TB','Pace','Touches_htouch_ata',
     'tch/press','Passes_Launch%','GoalKicks_Launch%']))

cca_cols = list(set(
    ['SCATypes_Sh','SCATypes_PassLive','SCATypes_PassDead','SCATypes_Def',
     'SCATypes_Drib','Performance_Crs','CrsPA','Drib_Rec100','KP_100_Prog','BodyPart
     'SCA_100touchesAtt3rd','Carries_Prog_100_Inc','Gini','xA/KP','Expected_npxG+xA_
     'CC_RecProg','Sh_T100',"xGChain_sh",'Crs_1_3'
     ]))


cat_dict= {'disposition':system_cols,
           'defensive_approach':def_cols,
           'buildup':buildup_cols,
           'attacking_approach':cca_cols}
```

In [14]:
```
scaler=MinMaxScaler()

def elbow_method(X, max_range_for_elbow):
    return kelbow_visualizer(KMeans(random_state=0), X, k=(1, max_range_for_elbow))
```

La función que viene a continuación realizará la clusterización de los equipos en base a las cuatro categorías mencionadas arriba. Más abajo analizaremos los resultados con objeto de perfilar cada uno de los clusters que se generen para todas las categorías.

En primer lugar, el dataset tomará las columnas citadas anteriormente para la categoría en cuestión, que se normalizarán vía MinMaxScaler. Se le aplicará el algoritmo VarianceThreshold, de cara a detectar si existen variables constantes o prácticamente constantes. Tras ello, se aplicará **PCA** hasta una varianza explicada del 80%. El método del codo nos ayudará a determinar el número óptimo de clusters que el algoritmo de clusterización **KMeans**.

Marcaremos específicamente que el modelo no segmente en más de cuatro grupos cada categoría.

Con ello, obtendremos la clusterización de los modelos de juego en base a esas categorías. Sin embargo, para poder analizarlo con mayor claridad, extraeremos las variables reales más explicativas (hasta un 90%) mediante un **RandomForestClassifier**. La función nos devolverá distintos plots que nos servirán de apoyo en nuestro análisis, el detalle del funcionamiento del proceso de entrenamiento de los procesos PCA y ElbowMethod y un dataset con la asignación a los clusters. Además, tendremos también un dataframe con la variabilidad explicada por cada una de las columnas.

In [15]:

```
def squad_clustering(categories):
    #np.random.seed(0)
    sns.set(style="whitegrid")
    squad_clustering = pd.DataFrame(squad[['Squad']],columns=['Squad'])
    fs=[]
    for i in categories:
        name = i
        print('-----Starting clustering for {}-----'.format(name))
        cols = cat_dict[i]
        data = squad[cols]
        data_norm = scaler.fit_transform(data)
        sel = VarianceThreshold(threshold=0.01)
        sel.fit(data_norm)
        print('Num. of variables that are not constant or cuasi-constant: ', sum(sel
        data_norm =  sel.transform(data_norm)
         # 2D PCA for the plot
        pca_app = 0
        for p in range(2,10):
            pca = PCA(n_components = p,random_state=0)
            print('PCA({}) for {}'.format(p,name))
            #reduced = pd.DataFrame(pca.fit_transform(data_norm))
            pca.fit(data_norm)
            print('Explained var = {}'.format(round(sum(pca.explained_variance_ratio
            if sum(pca.explained_variance_ratio_)>=0.9:
                pca_app = p
                print('Selected PCA: {}'.format(p))
                break
            else:
                continue

        plt.plot(range(0,p), list(pca.explained_variance_ratio_))
        plt.ylabel('Explained Variance')
        plt.xlabel('Principal Components')
        plt.xticks(range(0,p),rotation=60)
        plt.title('Explained Variance Ratio PCA{}'.format(pca_app))
        plt.show()

        pca = PCA(n_components = pca_app,random_state=0)
        reduced = pd.DataFrame(pca.fit_transform(data_norm))
        # We intend to have, as min, four clusters
        view = elbow_method(reduced,10)
        if view.elbow_value_>4:
            n = 4
        else:
            n = view.elbow_value_
#view.show()
        k = KMeans(n_clusters = n,random_state=0)

        reduced['cluster'] = k.fit_predict(reduced)
```

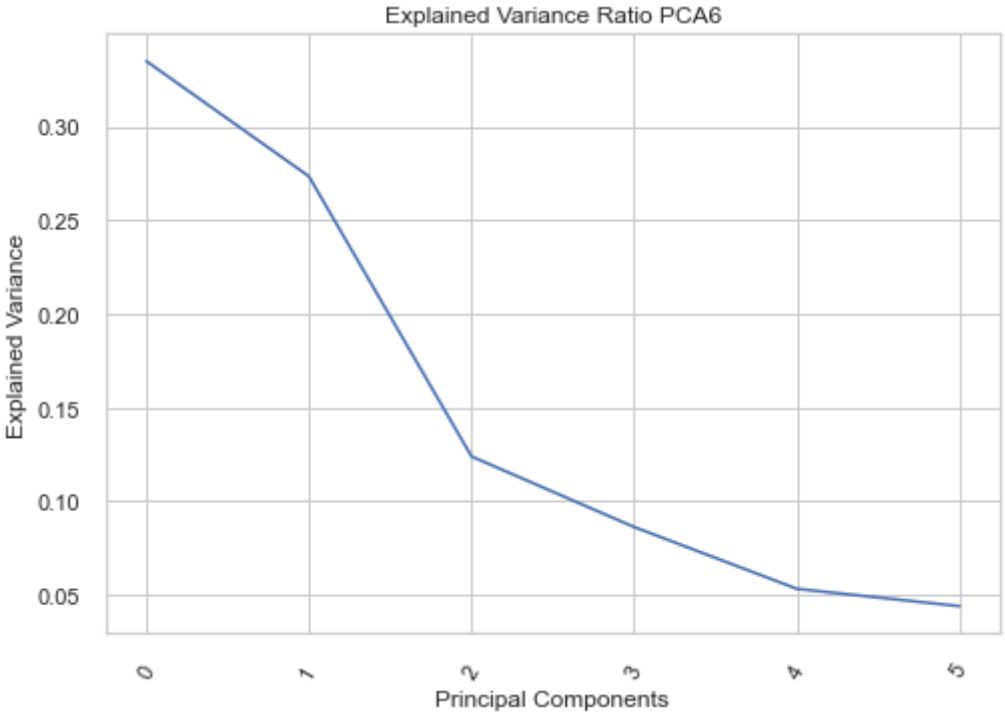
```
reduced['cluster'] = reduced['cluster'] +1
data = pd.merge(data,reduced[['cluster']],how='left',left_index=True,right_i
corr_matrix = data.corr()

upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np
upper
to_drop = [column for column in upper.columns if any(upper[column] > 0.9)]
data= data.drop(to_drop,axis=1)
data.drop('cluster',axis=1,inplace=True)
# Based on important information


clf = RandomForestClassifier(n_estimators=100, random_state=0)
clf.fit(scaler.fit_transform(data), reduced['cluster'])
feature_scores = pd.Series(clf.feature_importances_, index=data.columns).sor

f, ax = plt.subplots(figsize=(30, 20))
ax = sns.barplot(x=feature_scores, y=feature_scores.index)
ax.set_title("Feature scores of {} in squads".format(name),size=30)
ax.set_yticklabels(feature_scores.index,size=22)
ax.set_xlabel("Feature importance score",size=20)
ax.set_ylabel("Features",size=20)
plt.savefig(os.path.join(os.getcwd(),'Model','Teams')+"/Feature_scores_{}".f
plt.show()

feature_scores_to_drop = feature_scores[feature_scores.cumsum()>=.9]
data.drop(list(feature_scores_to_drop.index),inplace=True,axis=1)

cl=[]
for i in reduced.columns:
    if type(i)==int:
        i+=1
        cl.append('pc'+str(i))
    else:
        cl.append(i)
reduced.columns = cl
#reduced.columns=['pc1','pc2','cluster']
#centroids = k.cluster_centers_
reduced = pd.merge(reduced,squad[['Squad']],how='left',left_index=True,right

sns.set(style="white")
ax= sns.lmplot( x="pc1", y="pc2", hue='cluster', data = reduced, legend=True
 size = 8, scatter_kws={"s": 100},aspect=1.5,height=4, palette='muted')
ax=plt.gca()
ax.set_title('PCA({}) Clustering Distribution - {} in squads // Two Principa
texts = []
for x, y, s in zip(reduced.pc1, reduced.pc2, reduced.Squad):
    texts.append(plt.text(x, y, s))


#ax.set(ylim=(-5, 5))
#plt.scatter(centroids[:,0], centroids[:,1], c=range(centroids.shape[0]), s=
plt.tick_params(labelsize=10)
plt.xlabel("PC1", fontsize = 15)
plt.ylabel("PC2", fontsize = 15)
plt.tight_layout()
#ax.legend(frameon =True,shadow=True,loc="upper right")
plt.savefig(os.path.join(os.getcwd(),'Model','Teams')+"/PCA{}_{}clusters_{}.
plt.show()

#data_gr = data.groupby(by='cluster',as_index=False).mean()
data = pd.merge(data,reduced[['cluster']],how='left',left_index=True,right_i
squad_clustering[name+'_cluster'] = data[['cluster']]
```

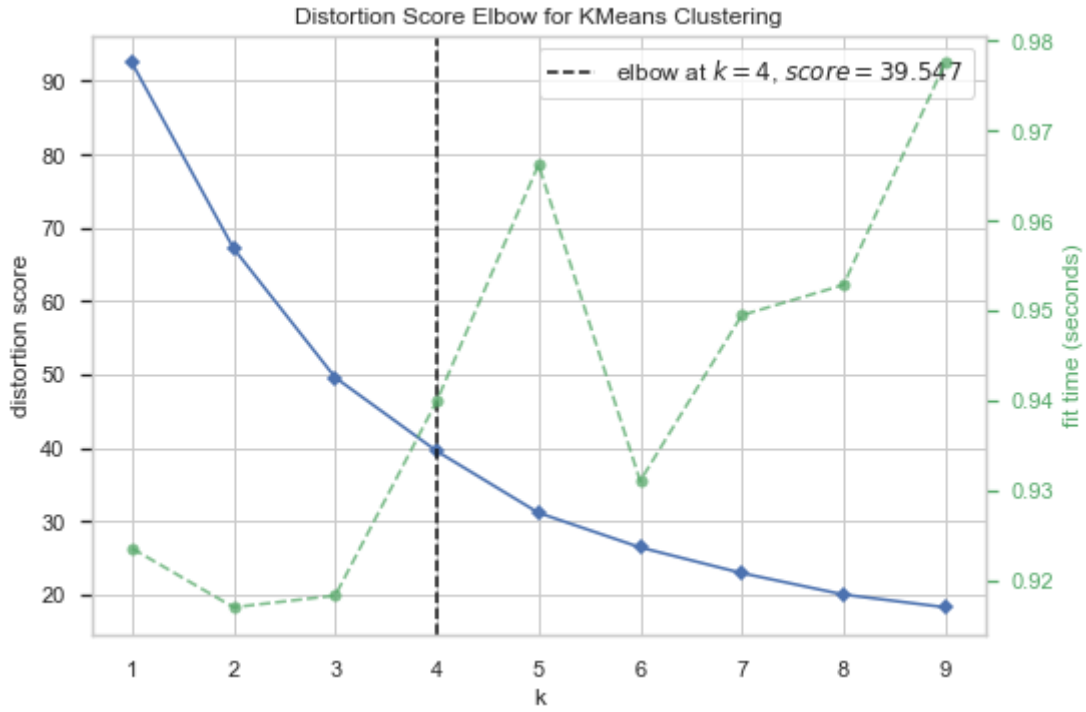
```
        fs.append(pd.DataFrame(feature_scores,columns=[name]))
    return squad_clustering,fs
```

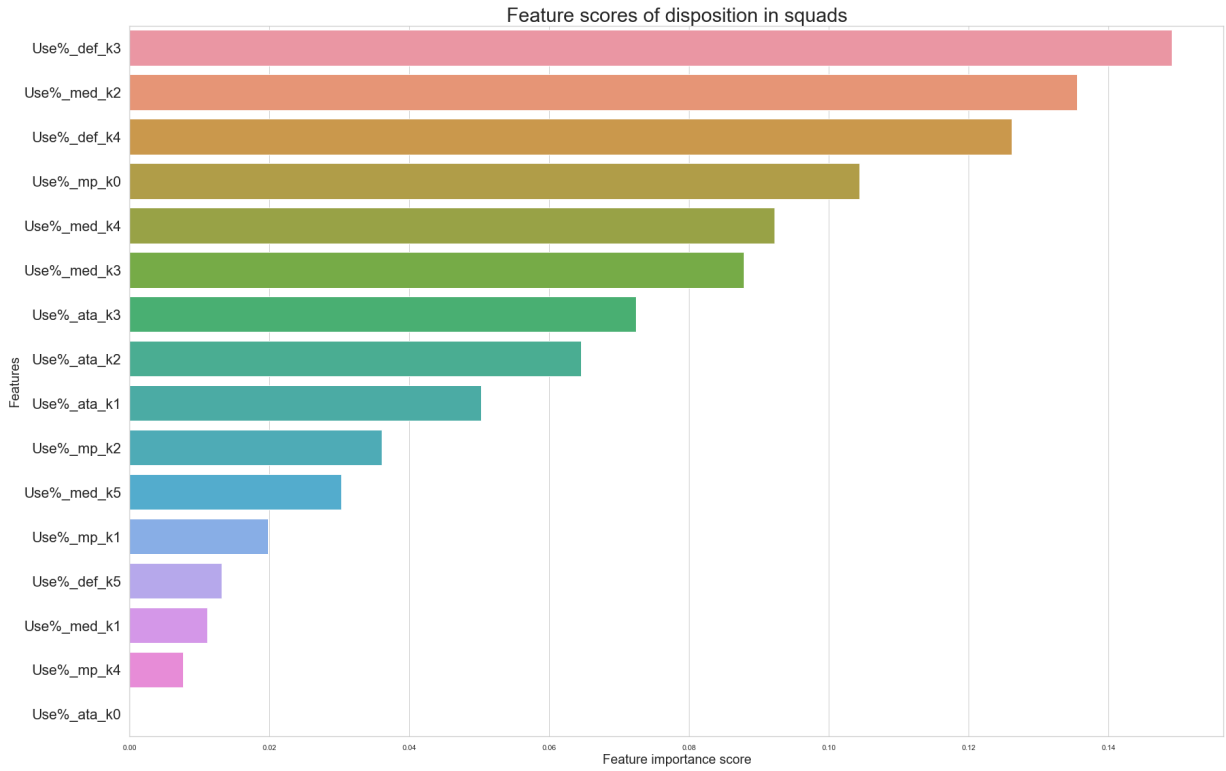
In [16]:

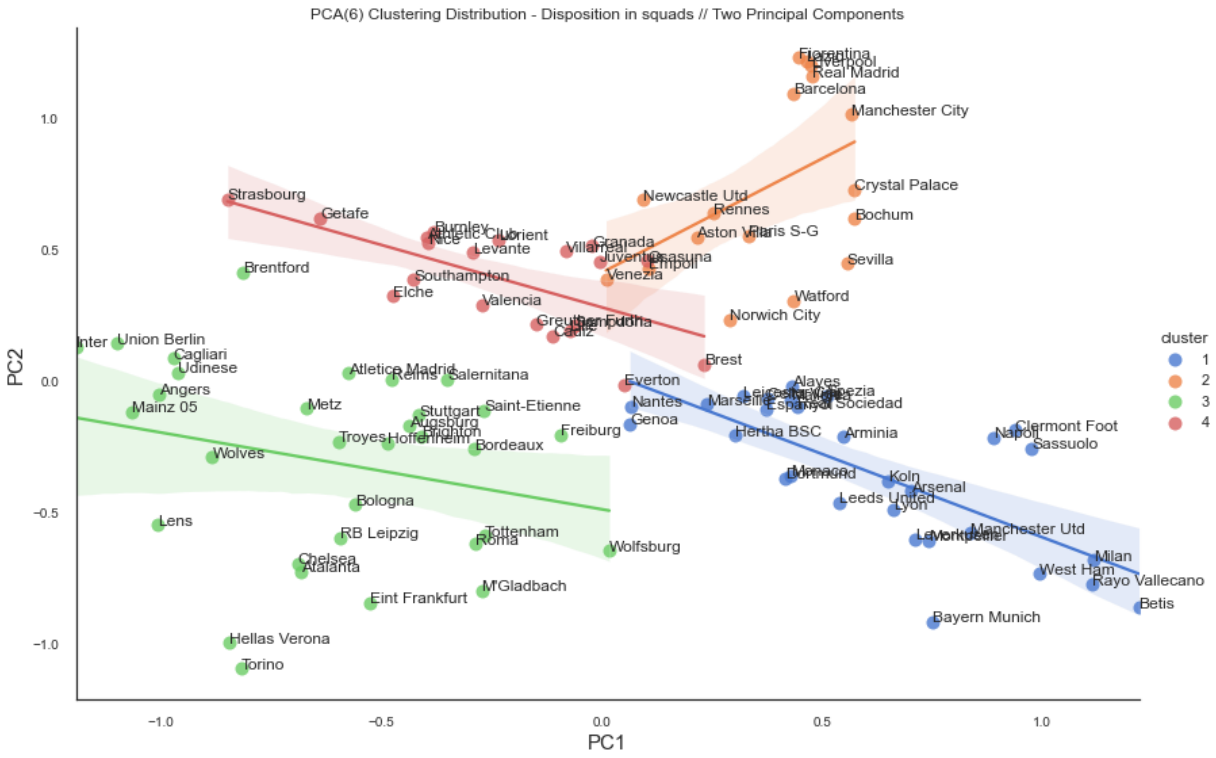
```
df,features = squad_clustering(list(cat_dict.keys()))
```

```
-----Starting clustering for disposition-----
Num. of variables that are not constant or cuasi-constant:  17  out of 17
PCA(2) for disposition
Explained var = 0.61
PCA(3) for disposition
Explained var = 0.73
PCA(4) for disposition
Explained var = 0.82
PCA(5) for disposition
Explained var = 0.87
PCA(6) for disposition
Explained var = 0.92
Selected PCA: 6
```

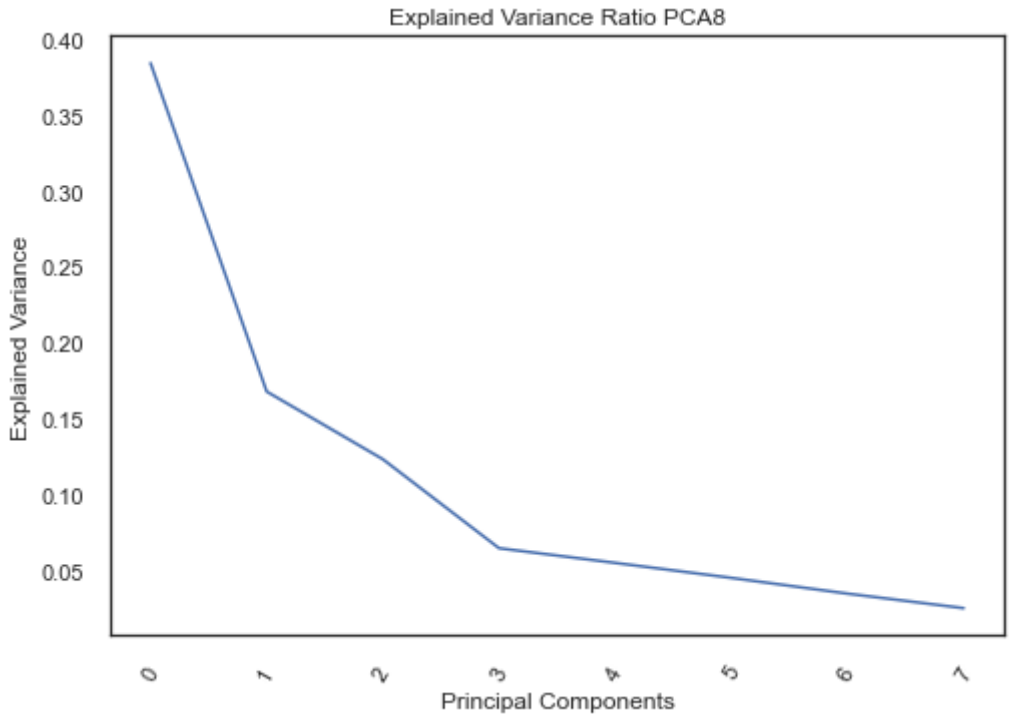


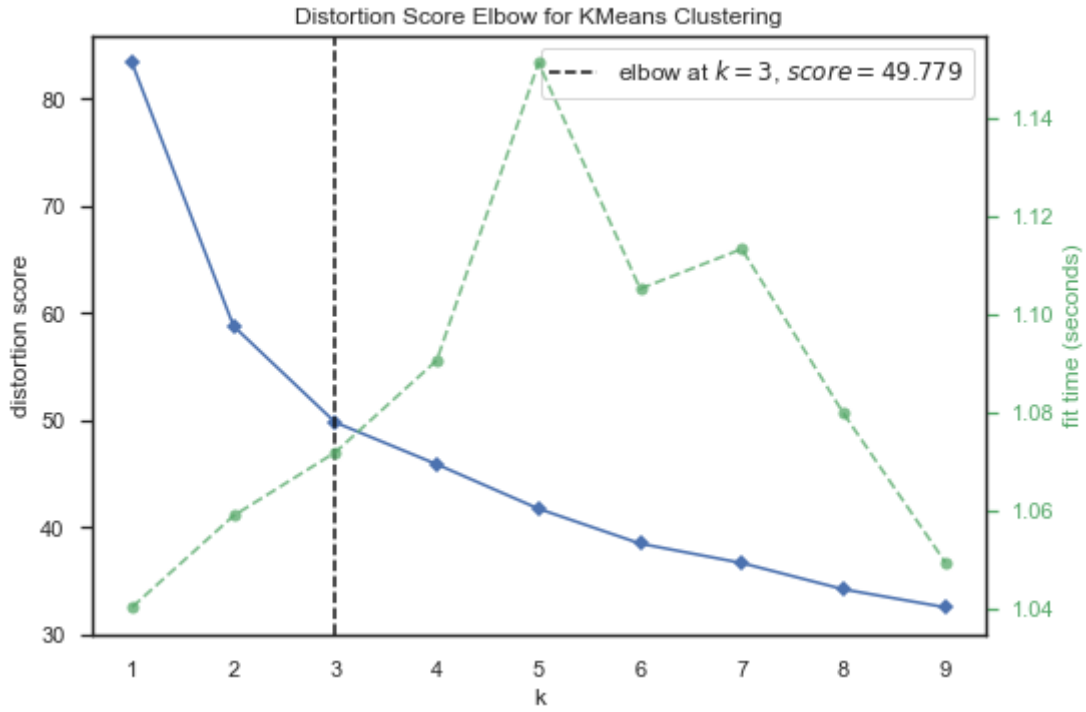

Distortion Score Elbow for KMeans Clustering
elbow at $k = 4$, $score = 39.547$
distortion score
fit time (seconds)
k
90
80
70
60
50
40
30
20
0.98
0.97
0.96
0.95
0.94
0.93
0.92
1
2
3
4
5
6
7
8
9

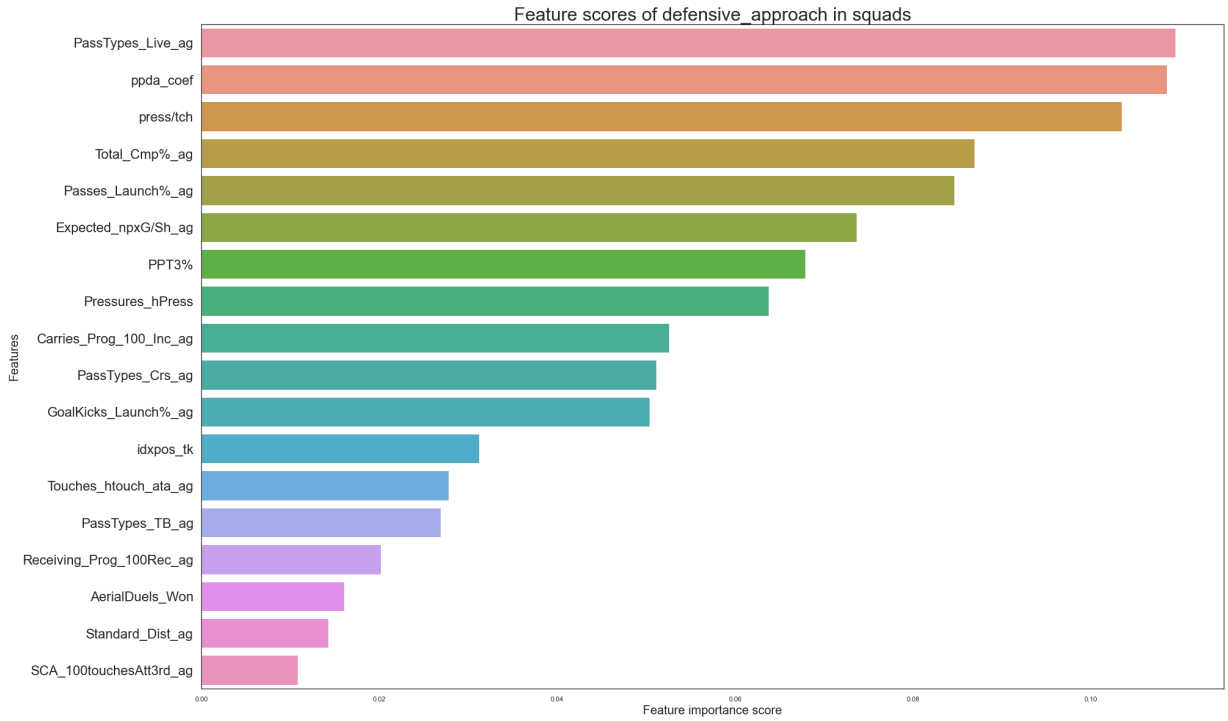

Feature scores of disposition in squads
Use%_def_k3
Use%_med_k2
Use%_def_k4
Use%_mp_k0
Use%_med_k4
Use%_med_k3
Use%_ata_k3
Use%_ata_k2
Use%_ata_k1
Use%_mp_k2
Use%_med_k5
Use%_mp_k1
Use%_def_k5
Use%_med_k1
Use%_mp_k4
Use%_ata_k0
Features
Feature importance score

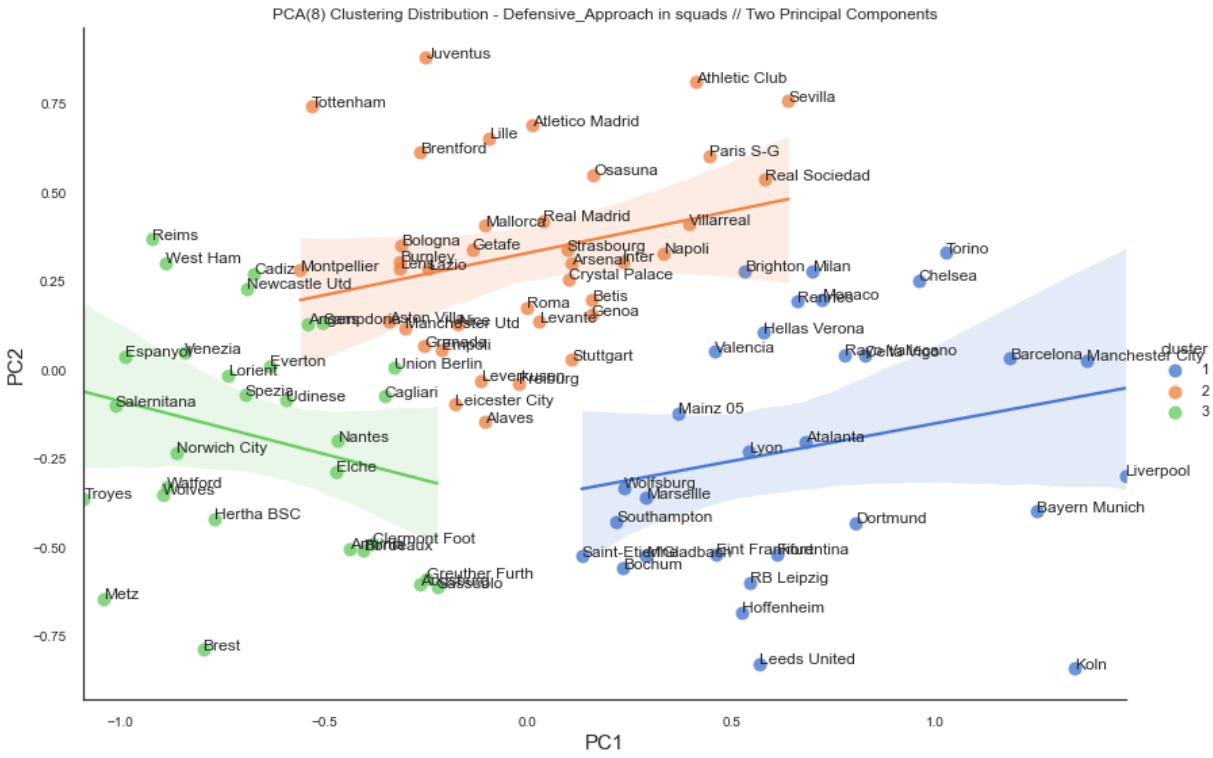



```
-----Starting clustering for defensive_approach-----
Num. of variables that are not constant or cuasi-constant:  21  out of 21
PCA(2) for defensive_approach
Explained var = 0.55
PCA(3) for defensive_approach
Explained var = 0.68
PCA(4) for defensive_approach
Explained var = 0.74
PCA(5) for defensive_approach
Explained var = 0.8
PCA(6) for defensive_approach
Explained var = 0.84
PCA(7) for defensive_approach
Explained var = 0.88
PCA(8) for defensive_approach
Explained var = 0.9
Selected PCA: 8
```

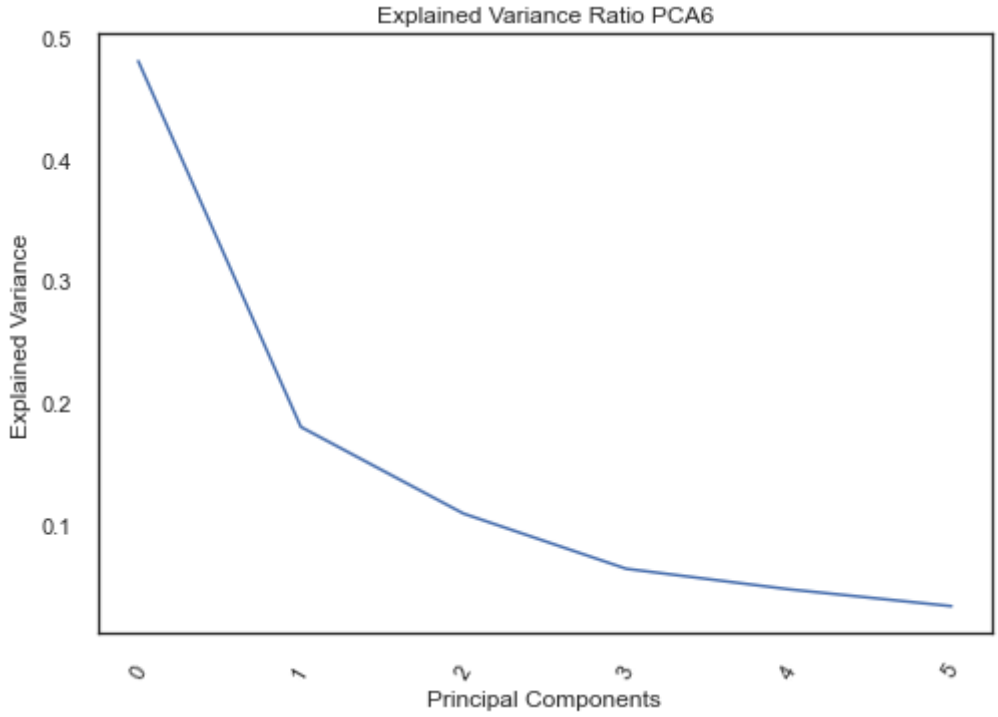


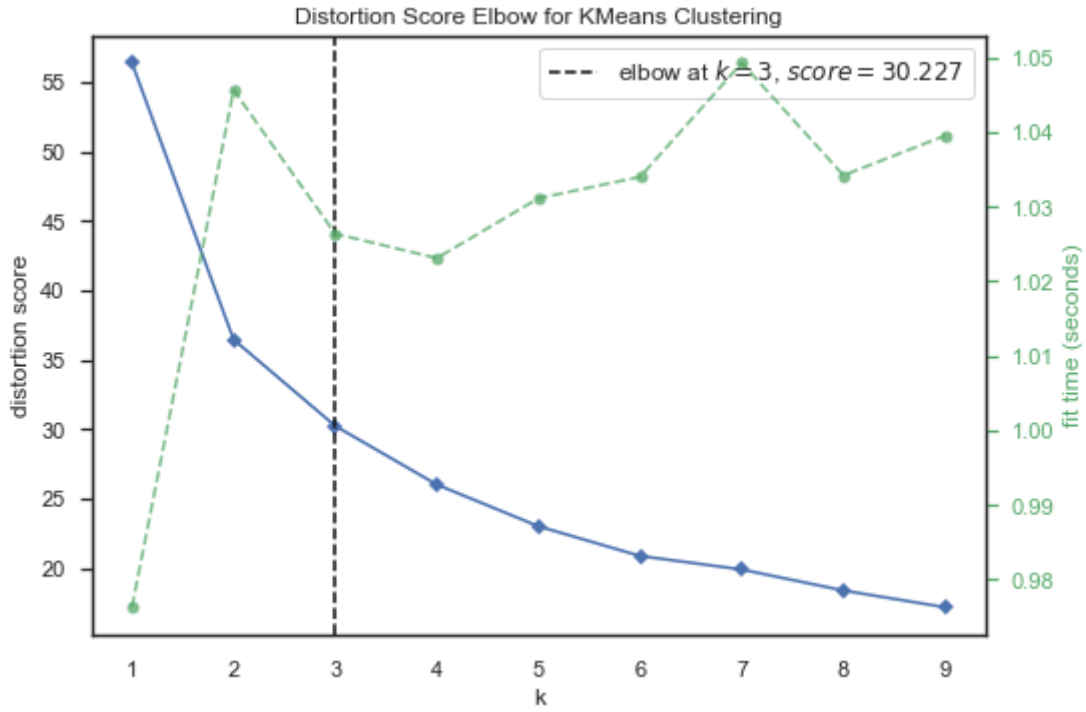

Distortion Score Elbow for KMeans Clustering
elbow at $k = 3$, $score = 49.779$
distortion score
fit time (seconds)
k
80
70
60
50
40
30
1.14
1.12
1.10
1.08
1.06
1.04
1
2
3
4
5
6
7
8
9

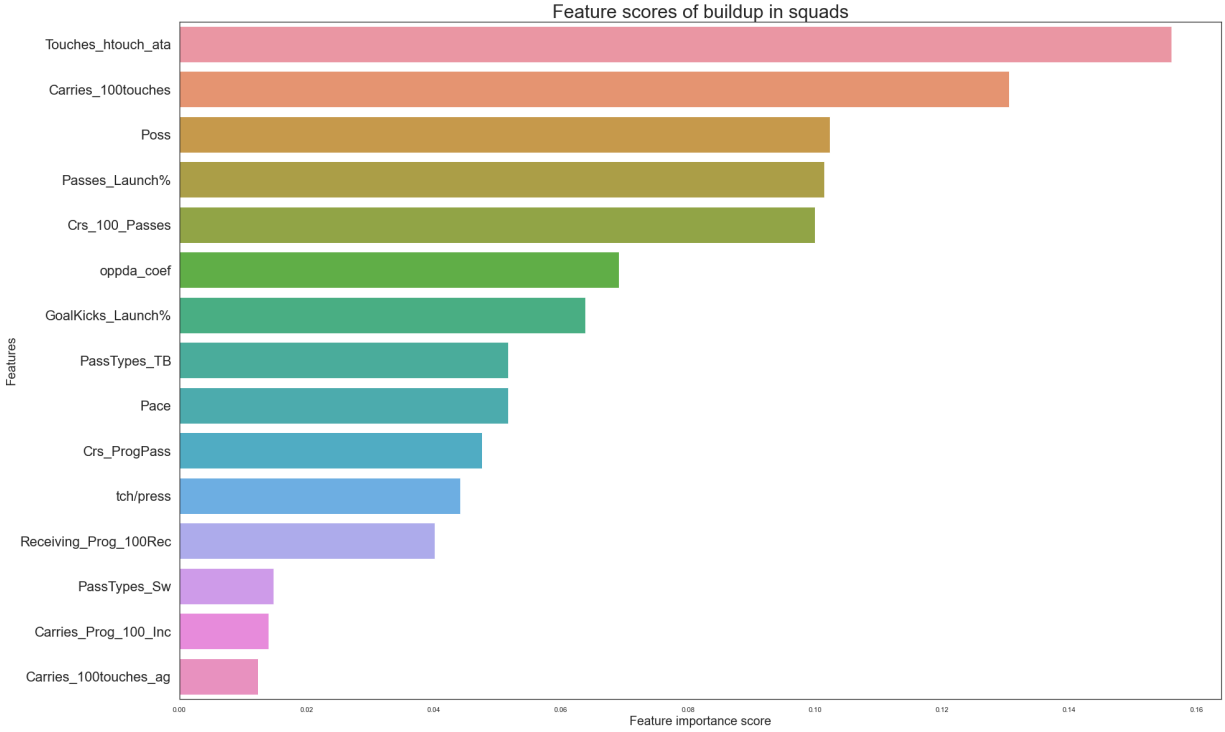

Feature scores of defensive_approach in squads
PassTypes_Live_ag
ppda_coef
press/tch
Total_Cmp%_ag
Passes_Launch%_ag
Expected_npxG/Sh_ag
PPT3%
Pressures_hPress
Carries_Prog_100_Inc_ag
PassTypes_Crs_ag
GoalKicks_Launch%_ag
idxpos_tk
Touches_htouch_ata_ag
PassTypes_TB_ag
Receiving_Prog_100Rec_ag
AerialDuels_Won
Standard_Dist_ag
SCA_100touchesAtt3rd_ag
Features
0.00
0.02
0.04
0.06
0.08
0.10
Feature importance score

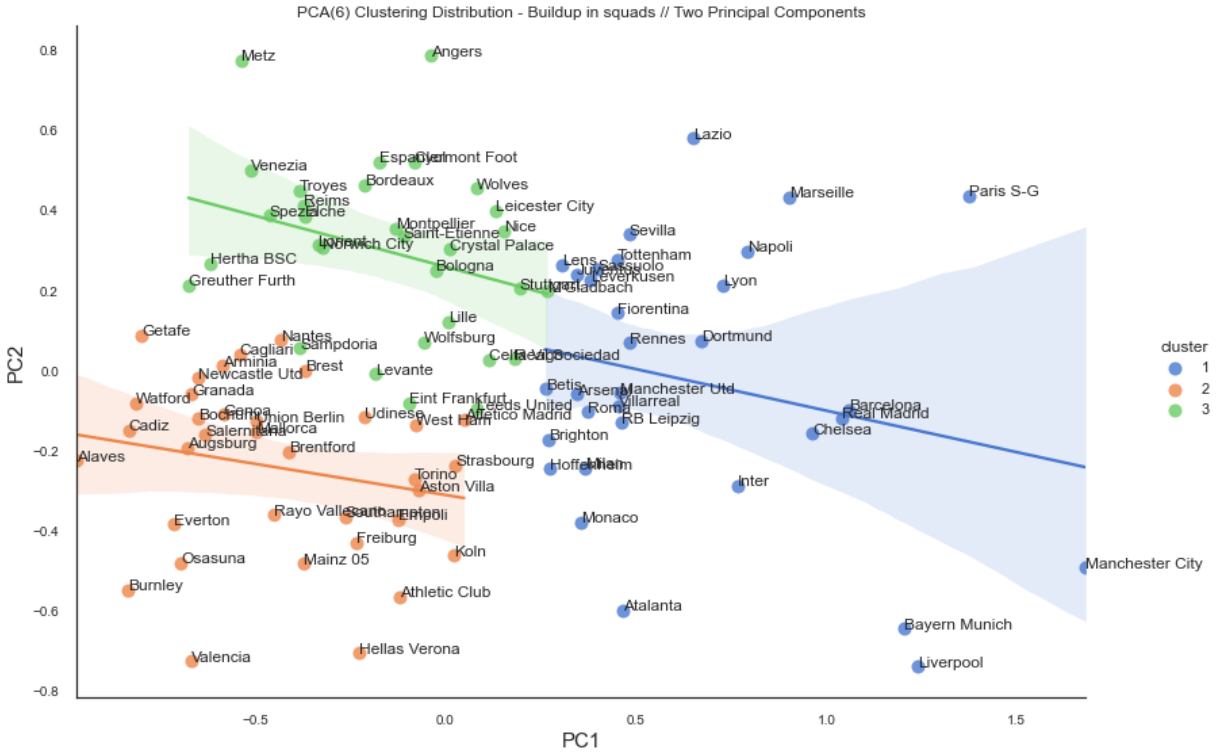



```
-----Starting clustering for buildup-----
Num. of variables that are not constant or cuasi-constant:  16  out of 16
PCA(2) for buildup
Explained var = 0.66
PCA(3) for buildup
Explained var = 0.77
PCA(4) for buildup
Explained var = 0.83
PCA(5) for buildup
Explained var = 0.88
PCA(6) for buildup
Explained var = 0.91
Selected PCA: 6
```

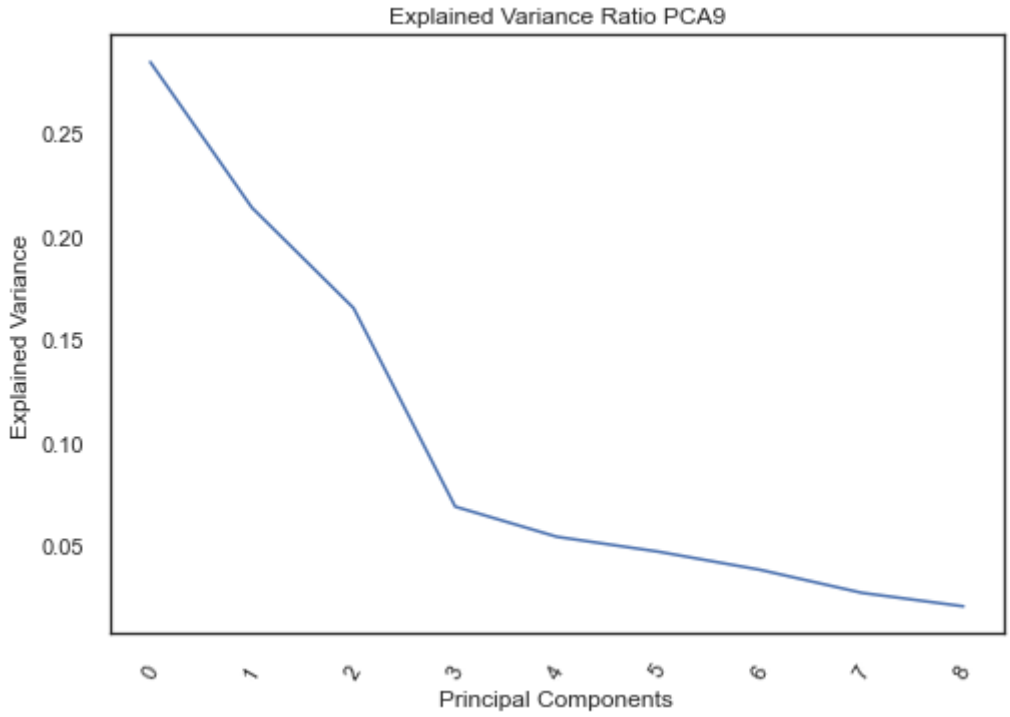


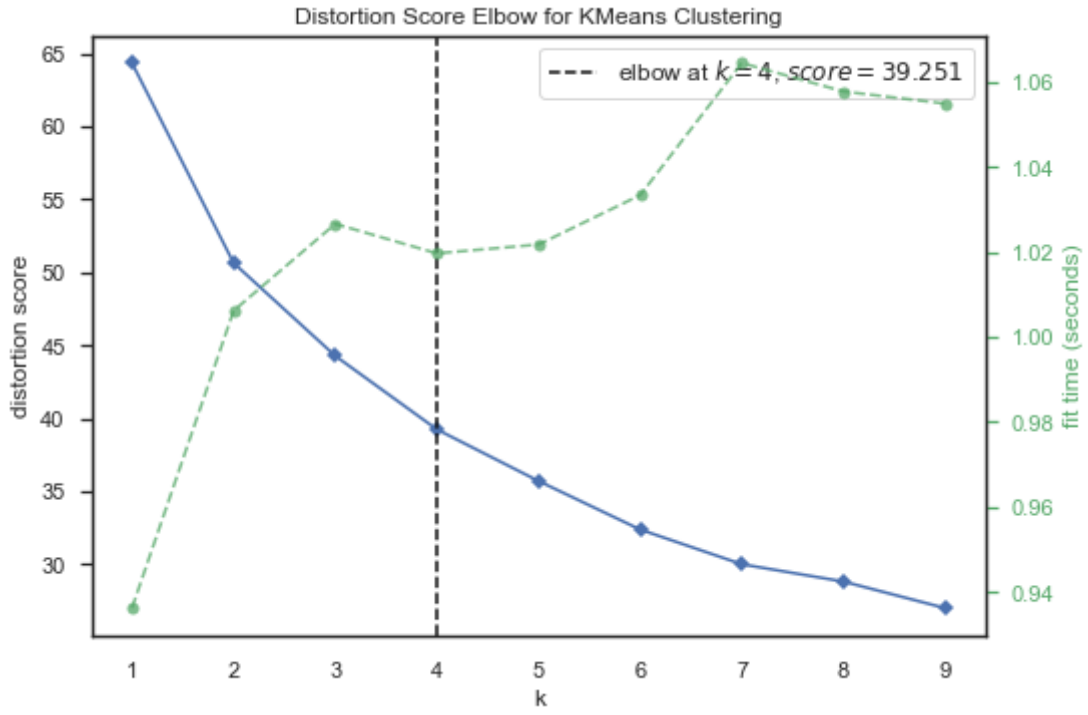

Distortion Score Elbow for KMeans Clustering
elbow at k = 3, score = 30.227
distortion score
fit time (seconds)
k
55
50
45
40
35
30
25
20
1.05
1.04
1.03
1.02
1.01
1.00
0.99
0.98
1
2
3
4
5
6
7
8
9

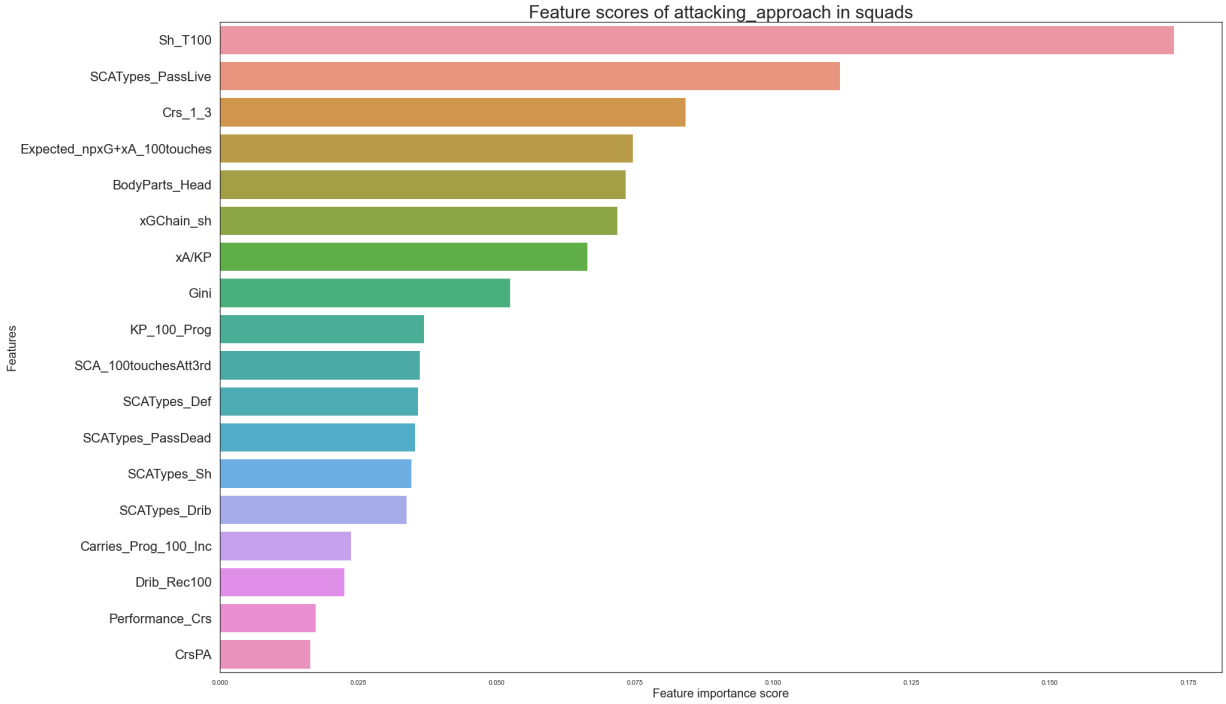

Feature scores of buildup in squads
Features
Touches_htouch_ata
Carries_100touches
Poss
Passes_Launch%
Crs_100_Passes
oppda_coef
GoalKicks_Launch%
PassTypes_TB
Pace
Crs_ProgPass
tch/press
Receiving_Prog_100Rec
PassTypes_Sw
Carries_Prog_100_Inc
Carries_100touches_ag
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Feature importance score

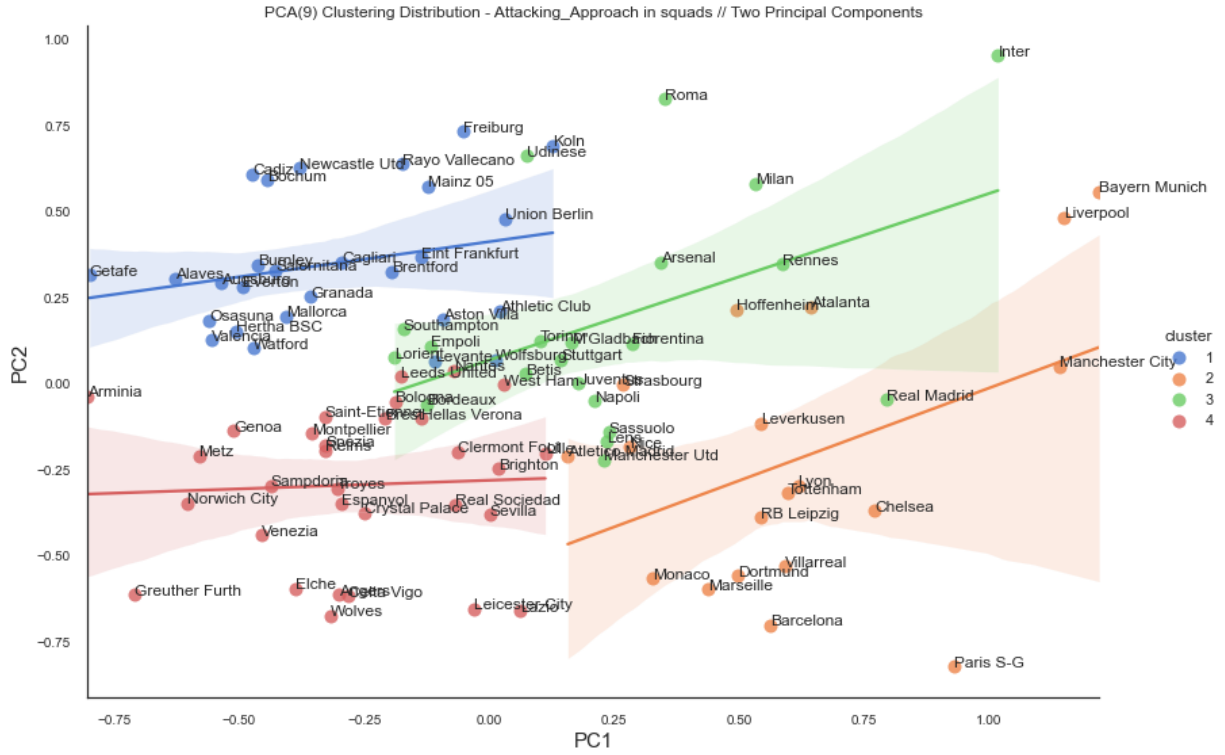



```
-----Starting clustering for attacking_approach-----
Num. of variables that are not constant or cuasi-constant:  19  out of 19
PCA(2) for attacking_approach
Explained var = 0.5
PCA(3) for attacking_approach
Explained var = 0.66
PCA(4) for attacking_approach
Explained var = 0.73
PCA(5) for attacking_approach
Explained var = 0.79
PCA(6) for attacking_approach
Explained var = 0.83
PCA(7) for attacking_approach
Explained var = 0.87
PCA(8) for attacking_approach
Explained var = 0.9
PCA(9) for attacking_approach
Explained var = 0.92
Selected PCA: 9
```


Explained Variance Ratio PCA9
Explained Variance
0.25
0.20
0.15
0.10
0.05
0
1
2
3
4
5
6
7
8
Principal Components


Distortion Score Elbow for KMeans Clustering
elbow at $k = 4$, $score = 39.251$
distortion score
fit time (seconds)
65
60
55
50
45
40
35
30
1.06
1.04
1.02
1.00
0.98
0.96
0.94
1
2
3
4
5
6
7
8
9
k





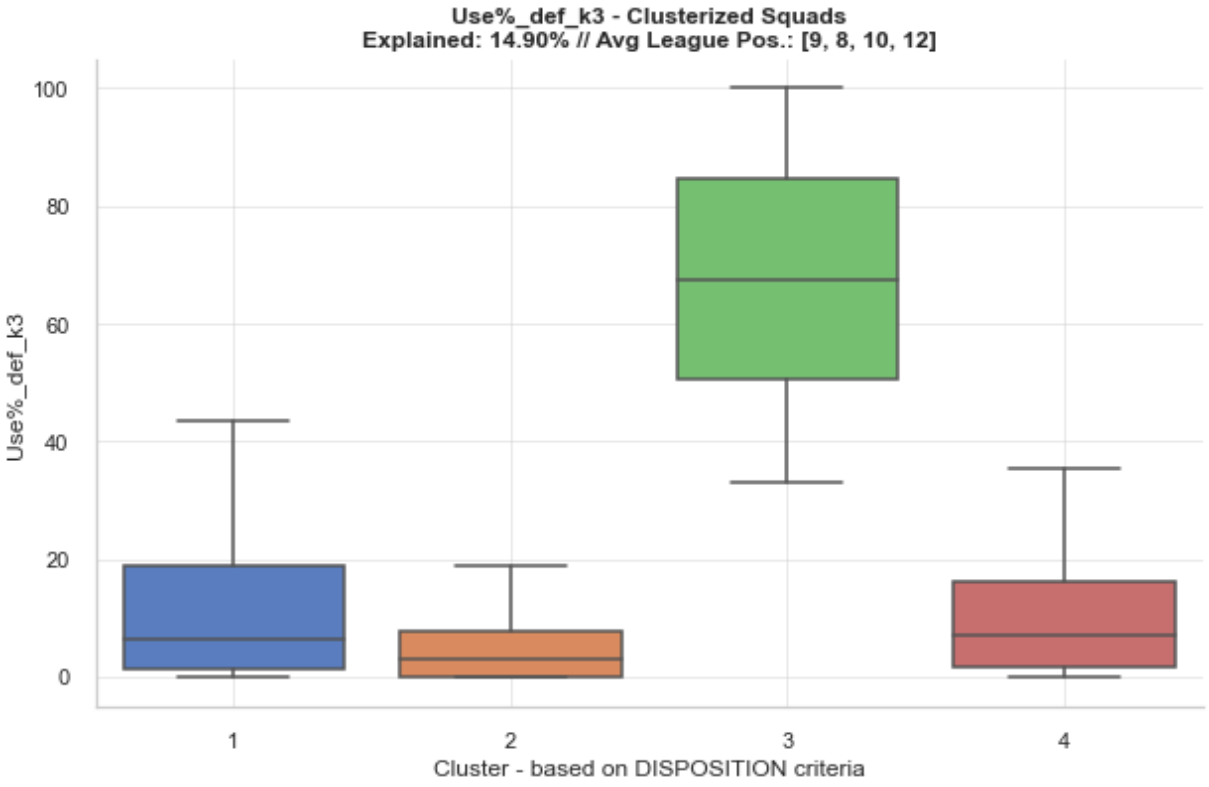
In [17]:
```
squad = pd.merge(squad,df,how='left',on='Squad')

for i in squad.columns:
    if '_cluster' in i:
        df[i] = df[i].astype(str)
```

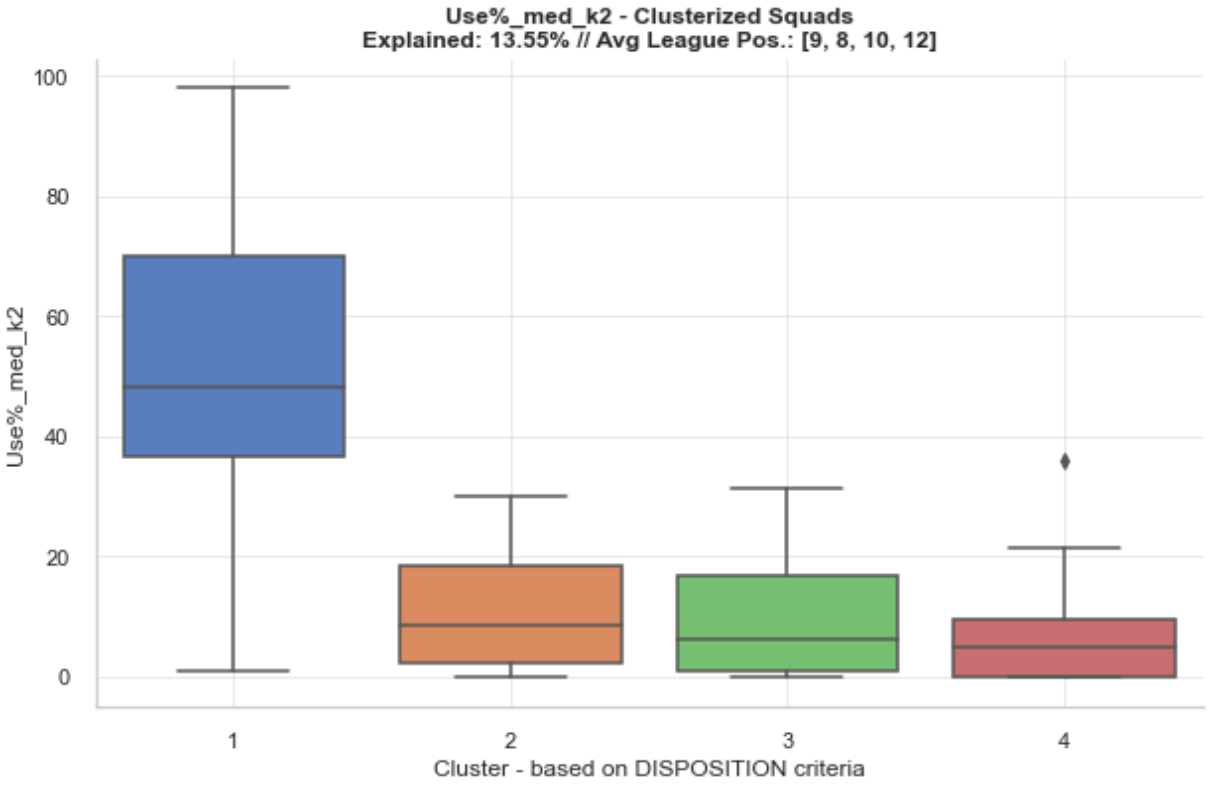
A partir del plotting siguiente, extraeremos una serie de Boxplots que nos permitirán conocer la composición media y dispersión de las medidas relevantes cada categoría. Dichas visualizaciones nos servirán de soporte para perfilar a los equipos.

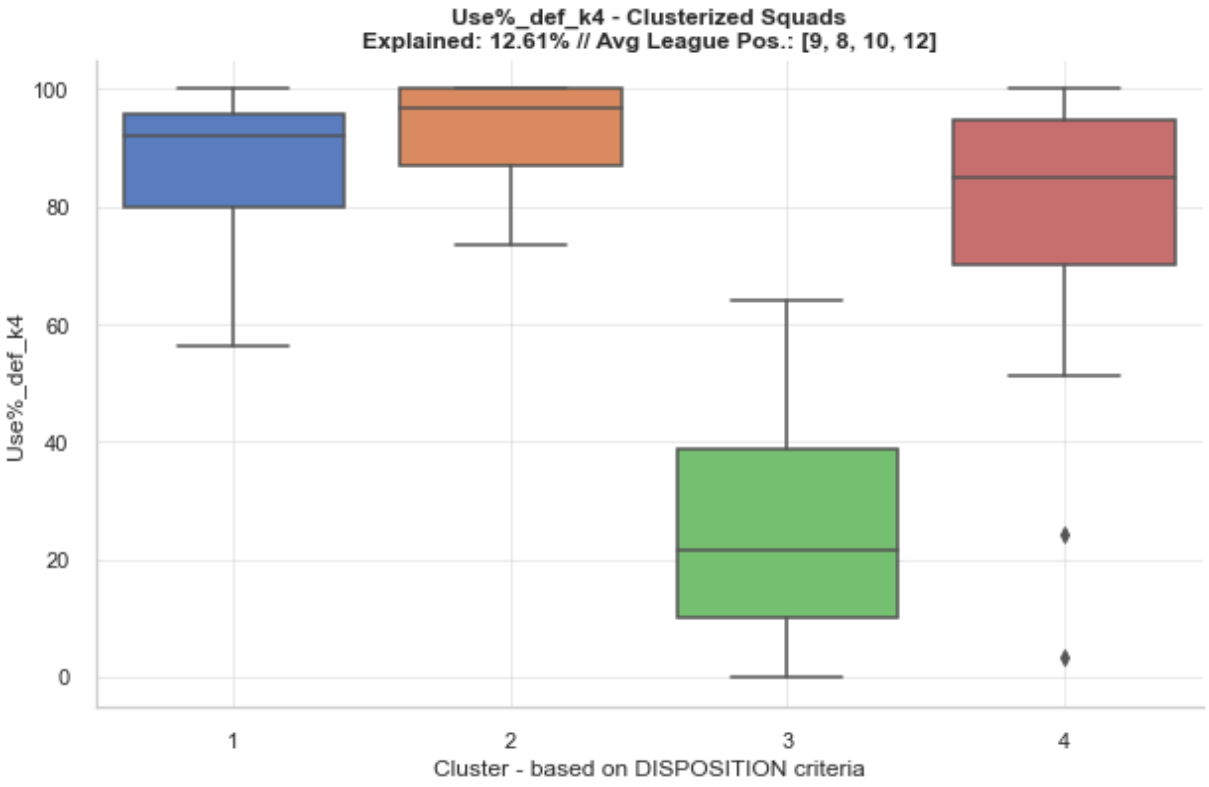
In [18]:
```
sns.set(style="whitegrid")

for f in features:
    clu = list(f.columns)[0]
    print('----------------------------------------------------------------------
```

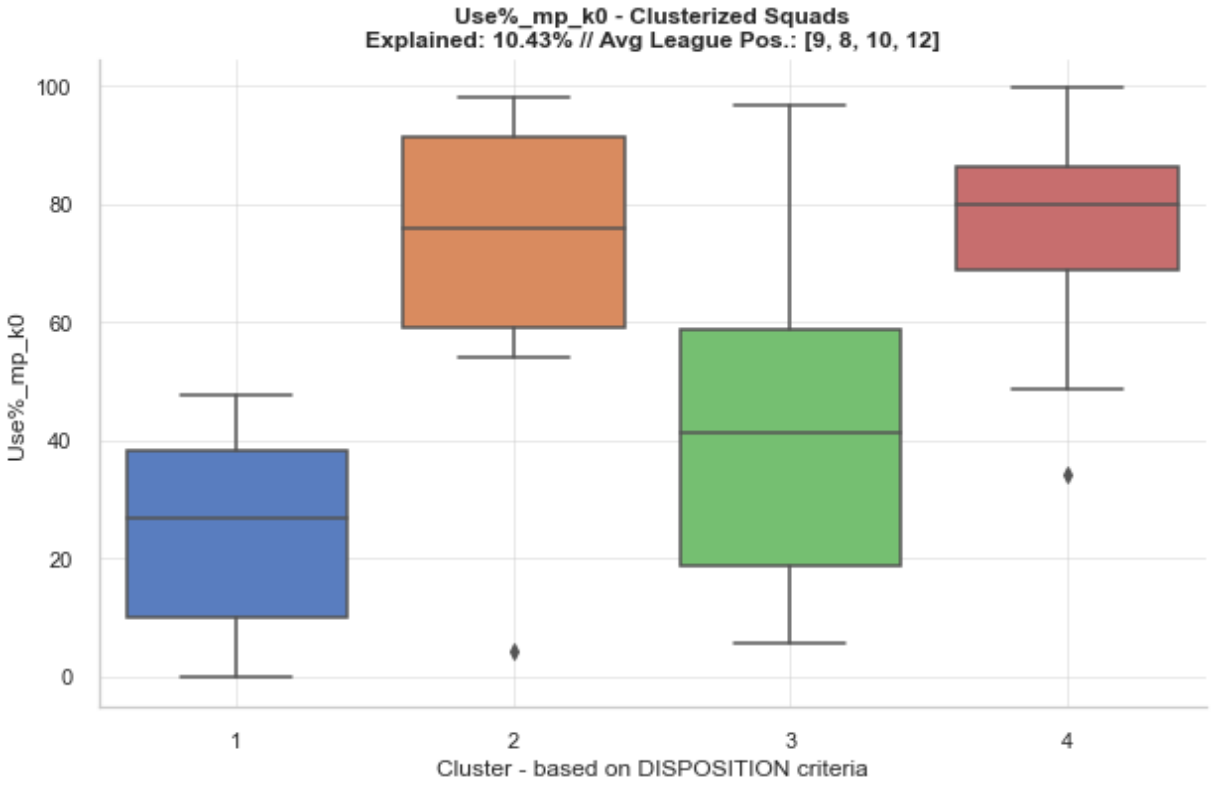
```
    print('-------------------------------------CATEGORY: {}-----------------------
    aa = squad.groupby(by=clu+'_cluster')['LgRk'].mean()
    aa = aa.astype(int)
    means = list(aa.values)
    f = f[:5]
    n=0
    for k in f.index:
        n+=1
        col=k
        val = round(f[clu][col],5)*100
        fig = plt.figure(figsize = (10, 6))
        ax = fig.add_subplot(111)

        sns.boxplot(x = "{}_cluster".format(clu),
                    y = col,
                    orient = "v",
                    data = squad,
                    ax = ax,
                    palette='muted')

        ax.set_xticklabels(ax.get_xticklabels(), rotation = 0)

        ax.set_title("{} - Clusterized Squads\nExplained: {:.2f}% // Avg League Pos.
                    weight='bold',size=12)
        ax.set_xlabel("Cluster - based on {} criteria".format(clu.upper()))
        ax.set_ylabel(col)
        sns.despine()
        plt.grid(True, alpha = 0.4)
        plt.savefig(os.path.join(os.getcwd(),'Model','Teams')+"/{}_cluster_metric{}.
        plt.show();
```

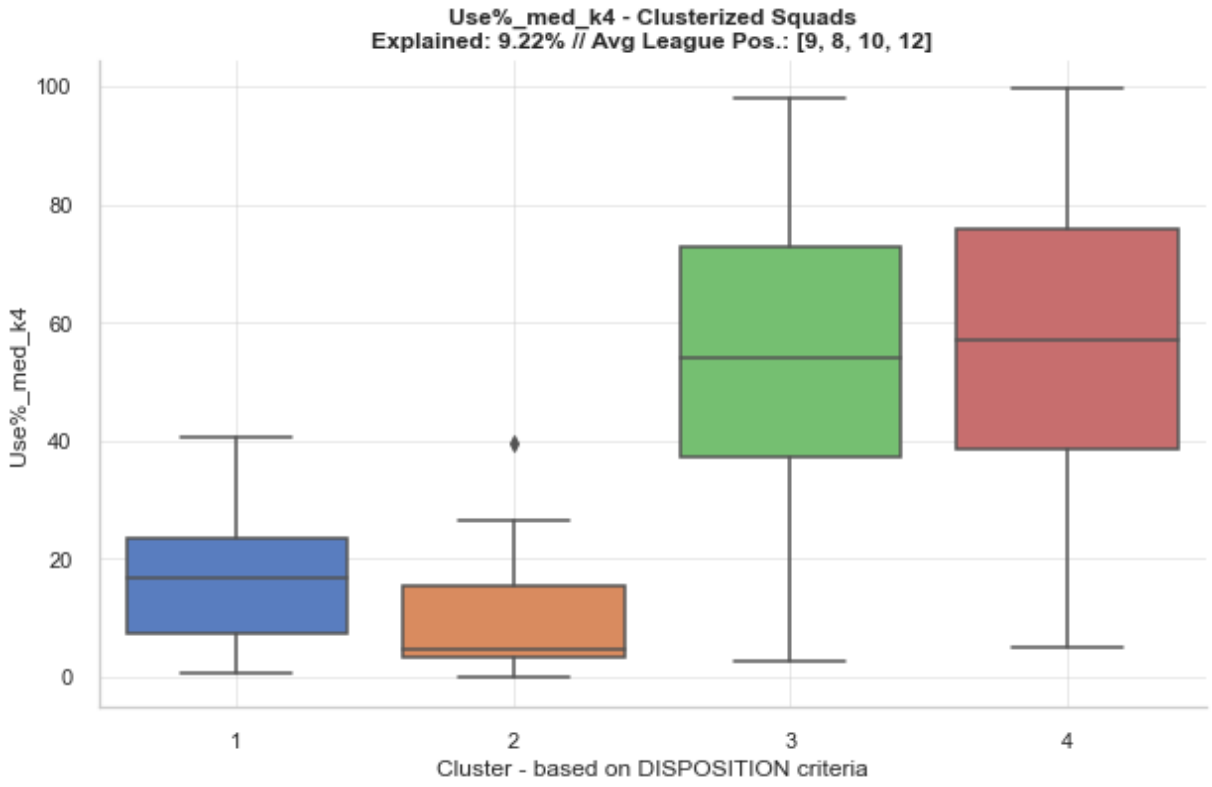
```
--------------------------------------------------------------------------------
------------------------
-------------------------------------CATEGORY: disposition--------------------------
------------------
```



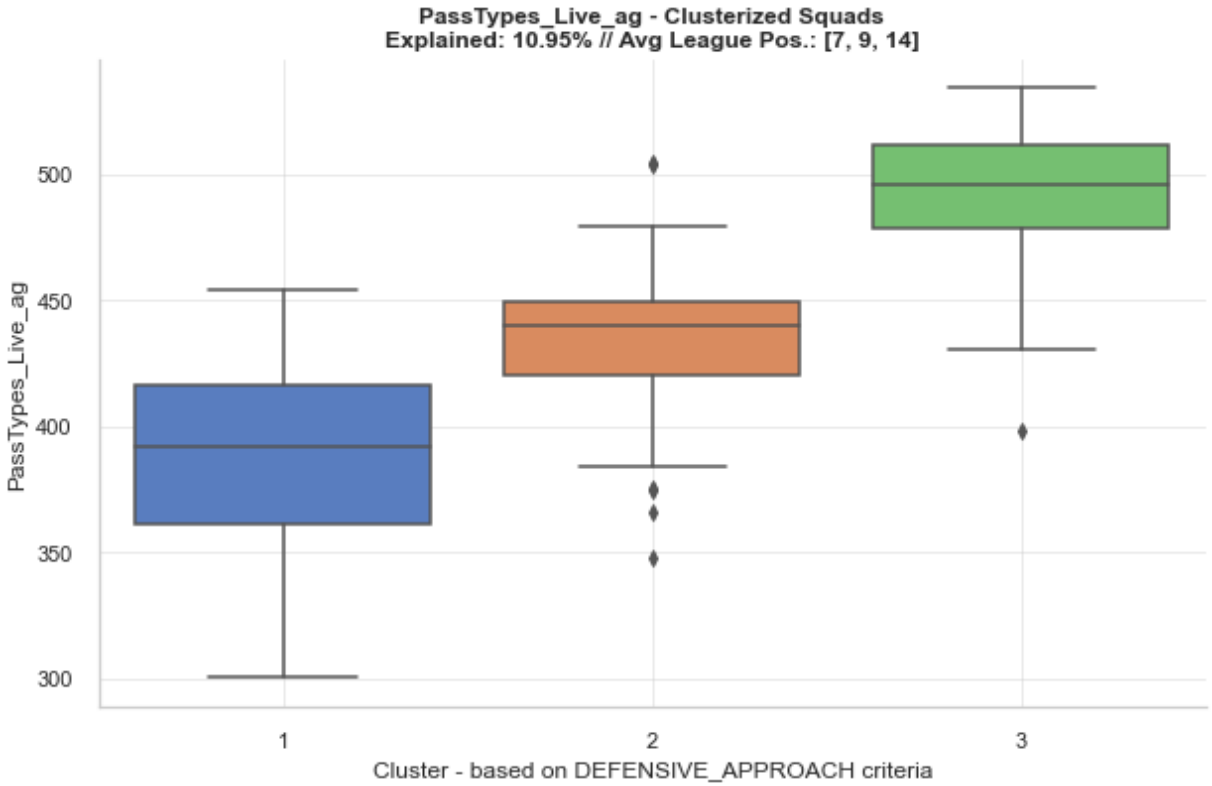

Use%_med_k2 - Clusterized Squads
Explained: 13.55% // Avg League Pos.: [9, 8, 10, 12]
Use%_med_k2
100
80
60
40
20
0
1
2
3
4
Cluster - based on DISPOSITION criteria

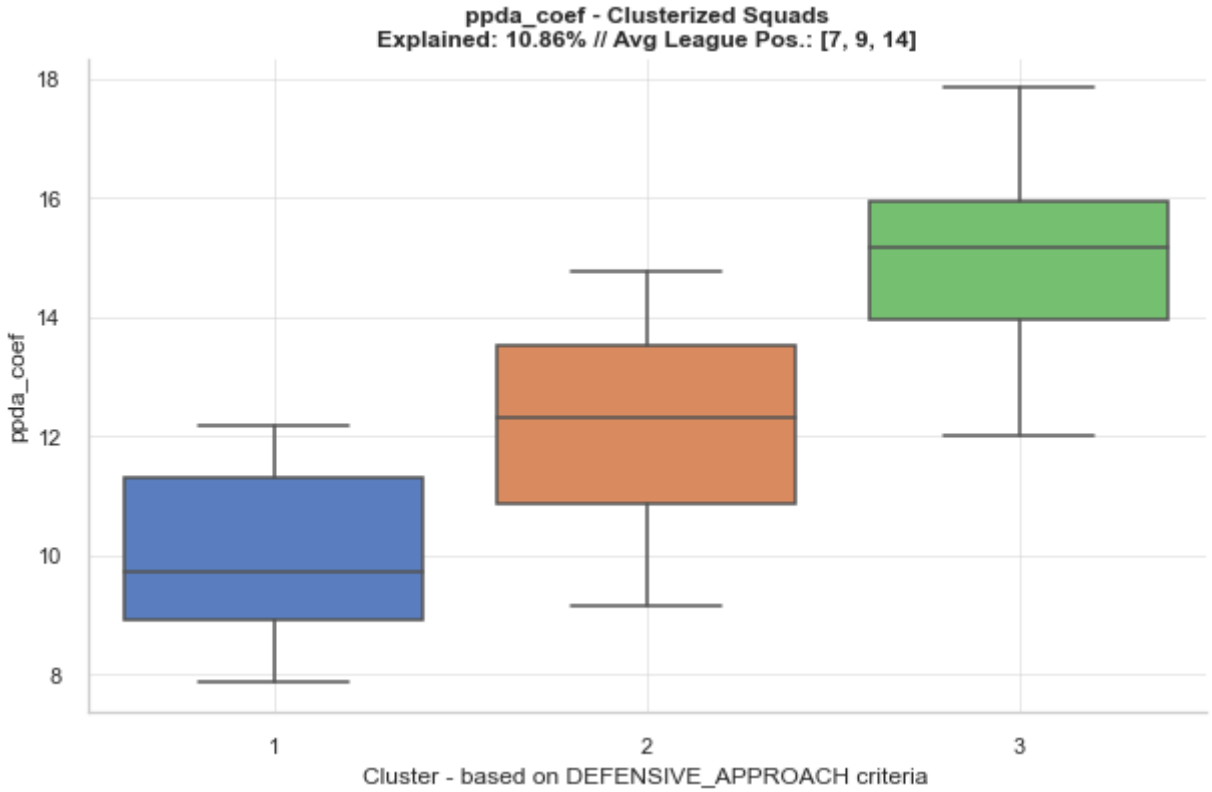

Use%_def_k4 - Clusterized Squads
Explained: 12.61% // Avg League Pos.: [9, 8, 10, 12]
Use%_def_k4
100
80
60
40
20
0
1
2
3
4
Cluster - based on DISPOSITION criteria

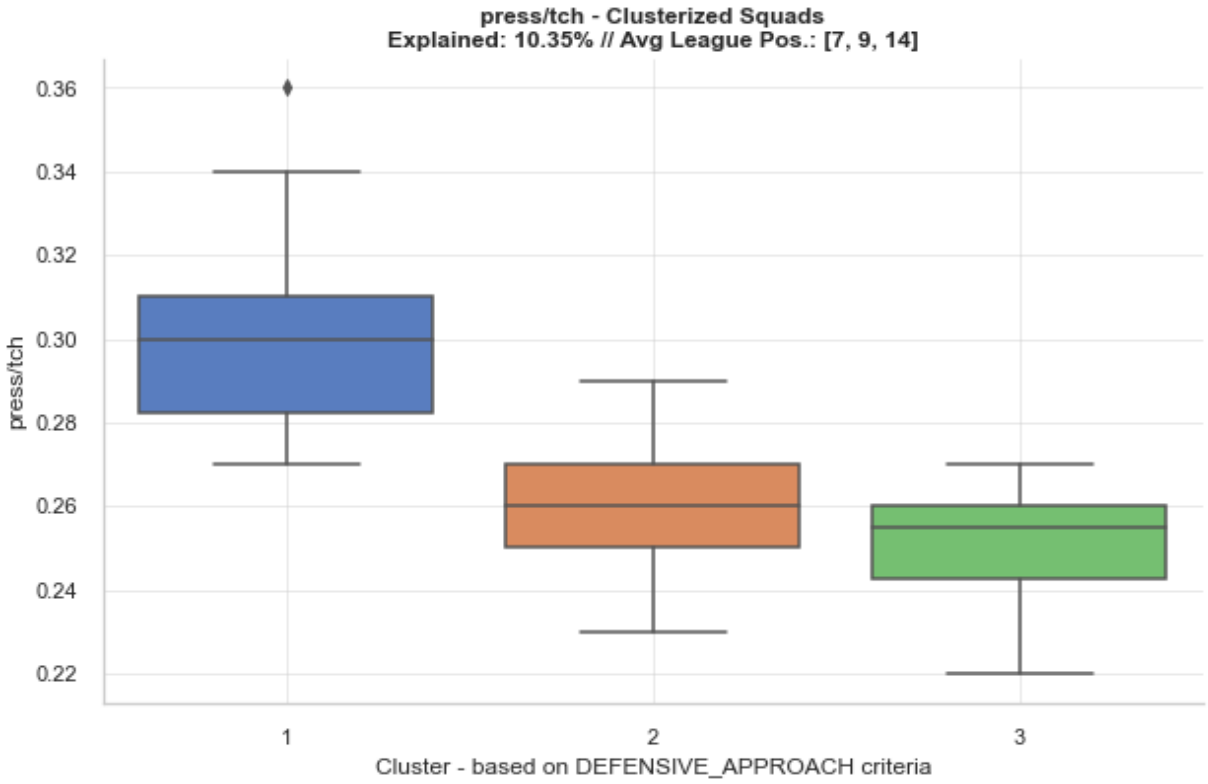


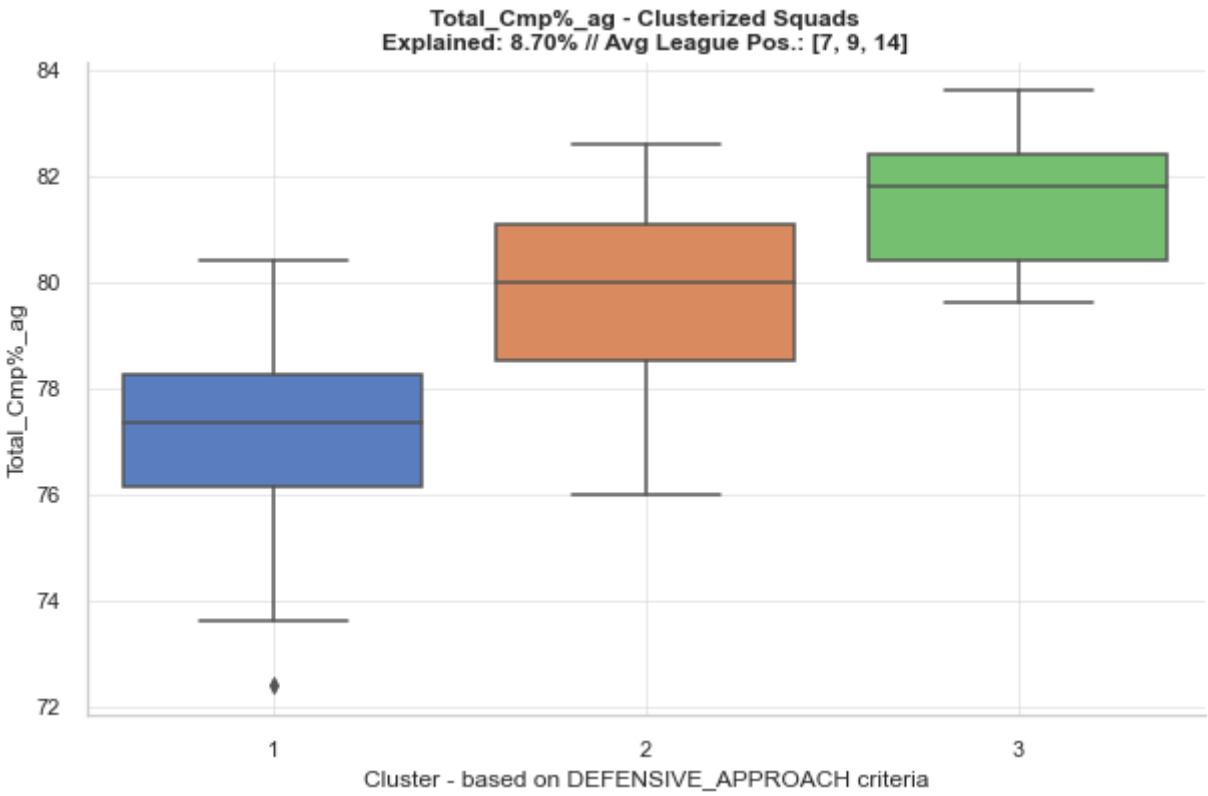


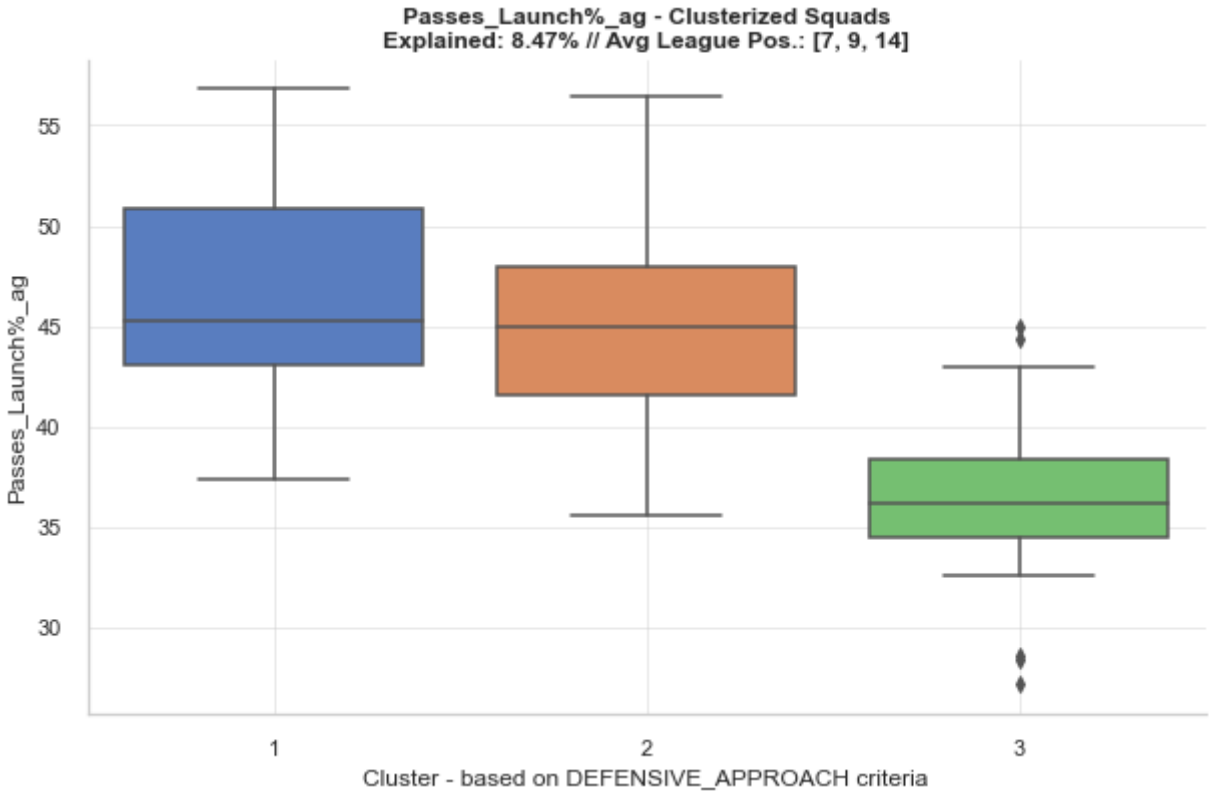
----------CATEGORY: defensive_approach----------


PassTypes_Live_ag - Clusterized Squads
Explained: 10.95% // Avg League Pos.: [7, 9, 14]
PassTypes_Live_ag
500
450
400
350
300
1
2
3
Cluster - based on DEFENSIVE_APPROACH criteria

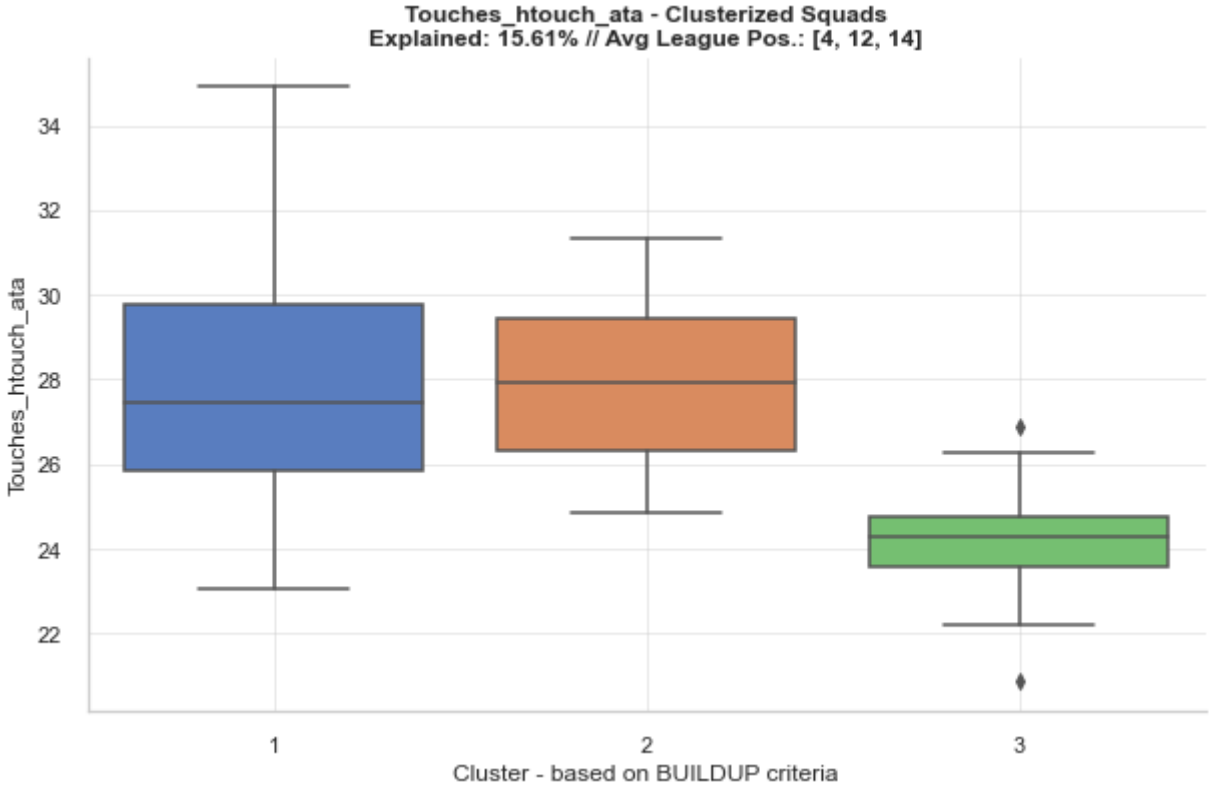

ppda_coef - Clusterized Squads
Explained: 10.86% // Avg League Pos.: [7, 9, 14]
ppda_coef
18
16
14
12
10
8
1
2
3
Cluster - based on DEFENSIVE_APPROACH criteria

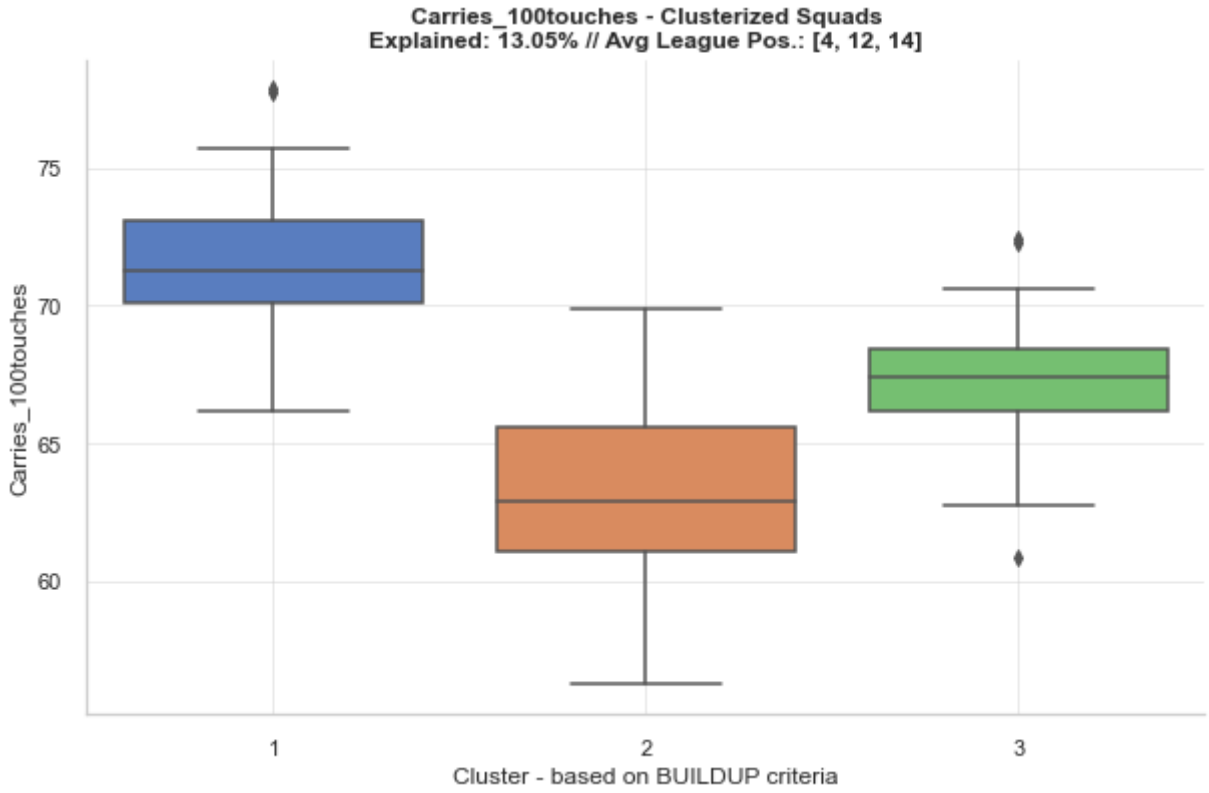

press/tch - Clusterized Squads
Explained: 10.35% // Avg League Pos.: [7, 9, 14]
press/tch
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
1
2
3
Cluster - based on DEFENSIVE_APPROACH criteria

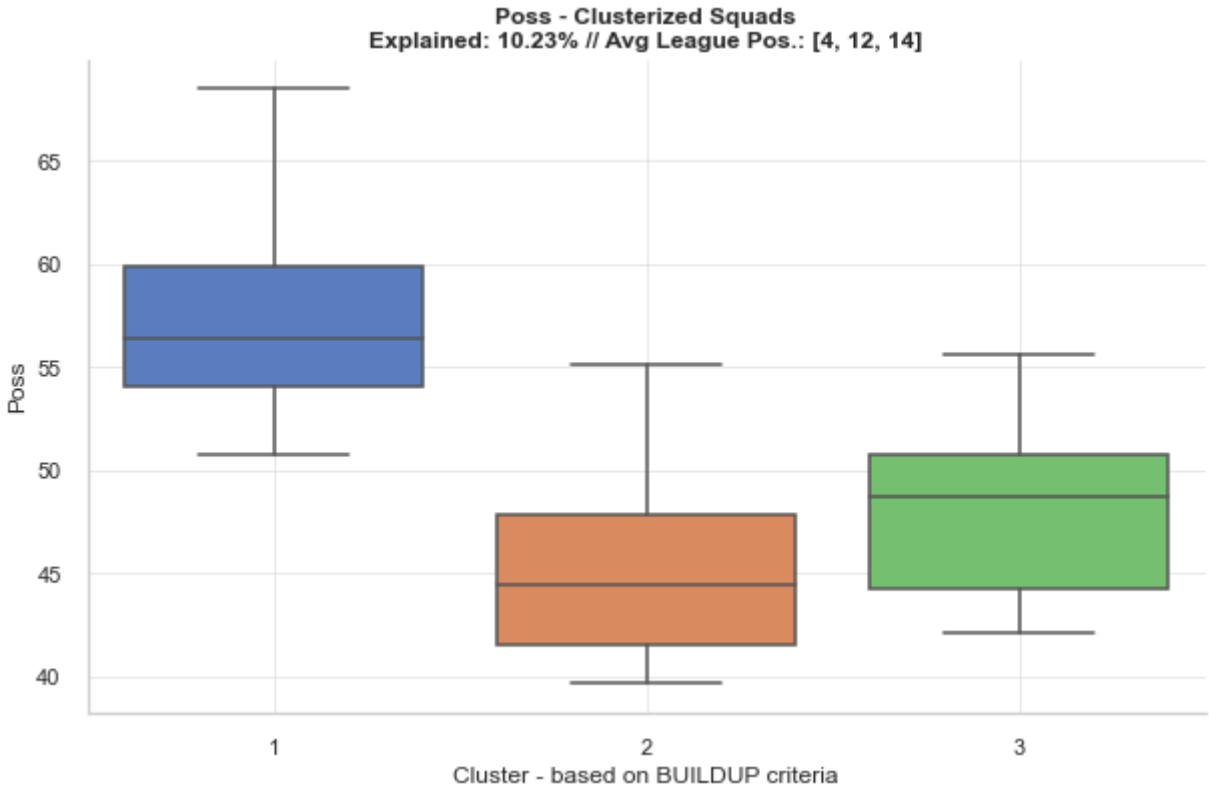

Total_Cmp%_ag - Clusterized Squads
Explained: 8.70% // Avg League Pos.: [7, 9, 14]
Total_Cmp%_ag
84
82
80
78
76
74
72
1
2
3
Cluster - based on DEFENSIVE_APPROACH criteria

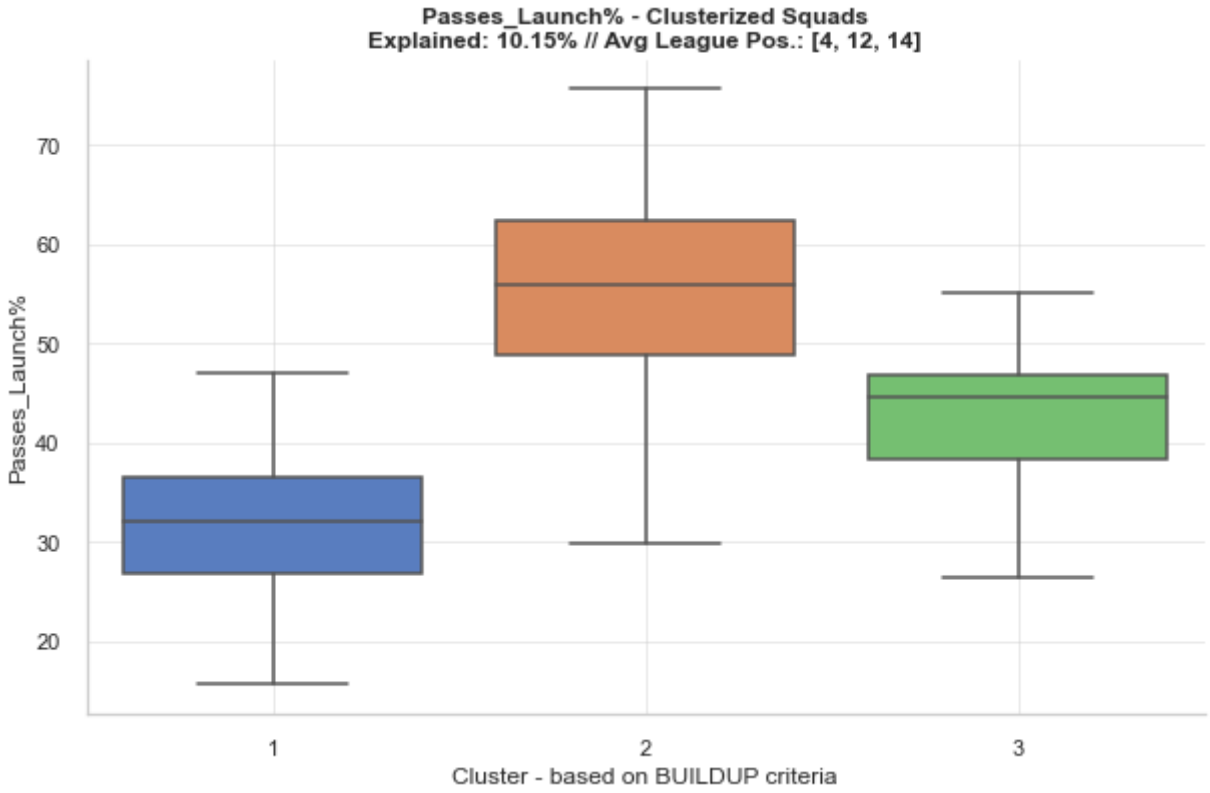


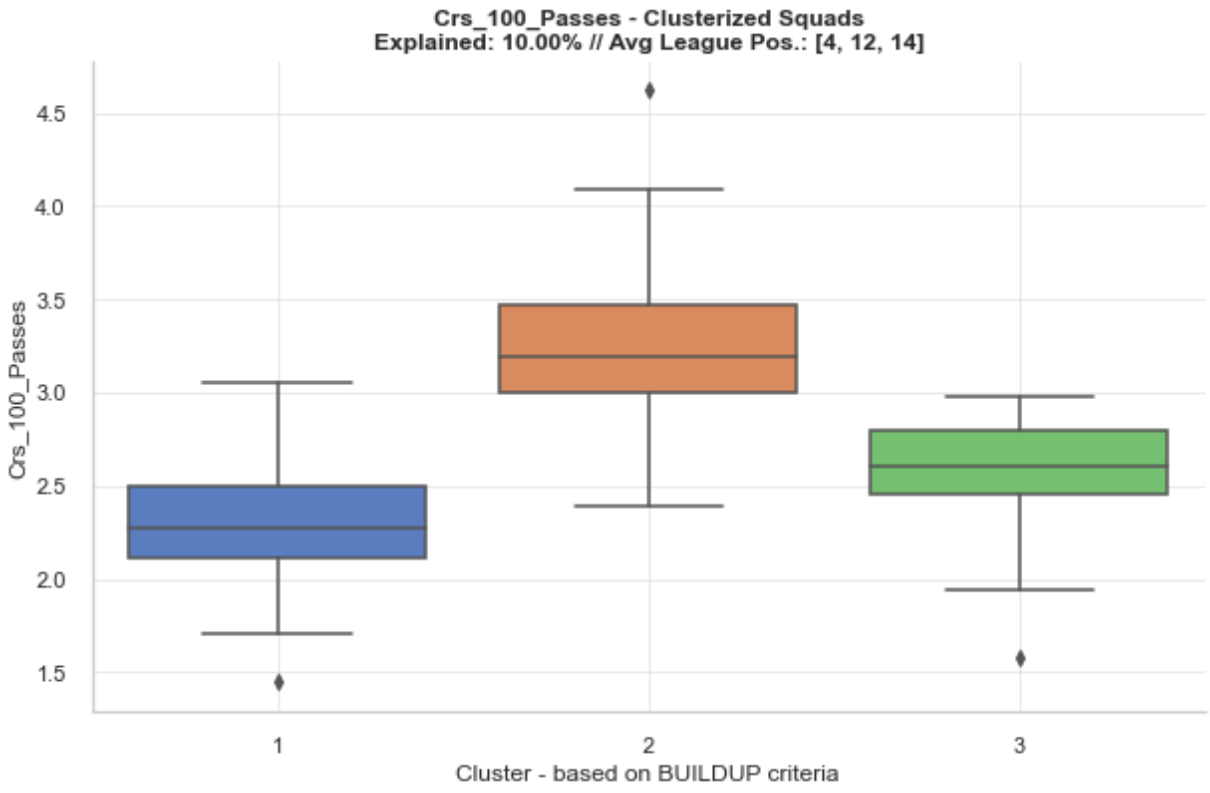
----------CATEGORY: buildup----------



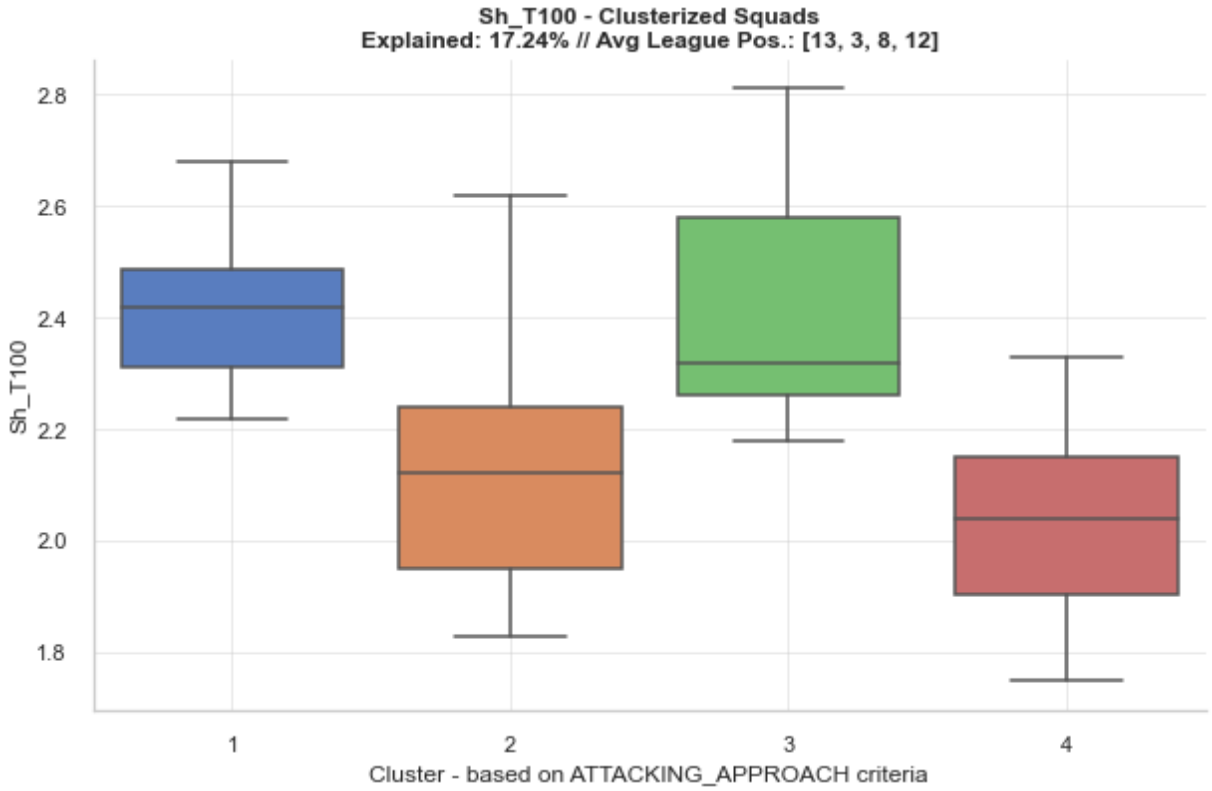

Carries_100touches - Clusterized Squads
Explained: 13.05% // Avg League Pos.: [4, 12, 14]
Carries_100touches
75
70
65
60
1
2
3
Cluster - based on BUILDUP criteria

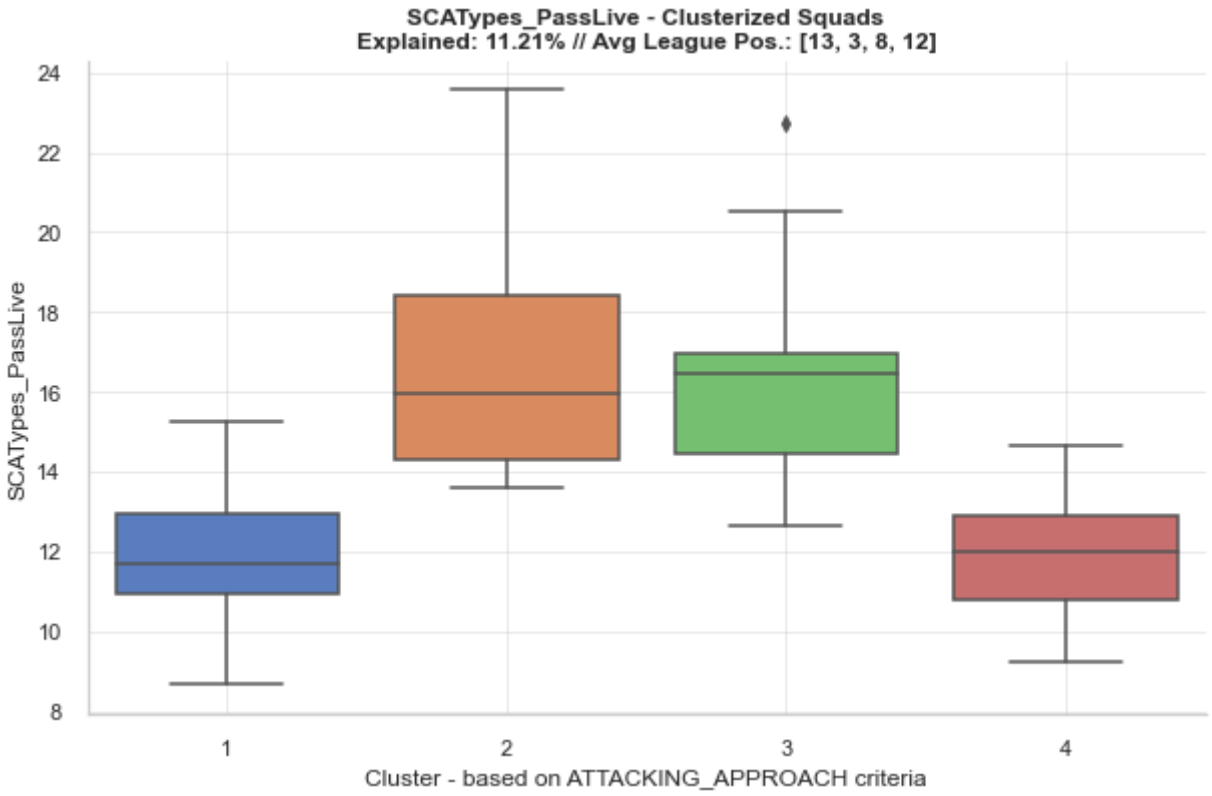

Poss - Clusterized Squads
Explained: 10.23% // Avg League Pos.: [4, 12, 14]
Poss
65
60
55
50
45
40
1
2
3
Cluster - based on BUILDUP criteria

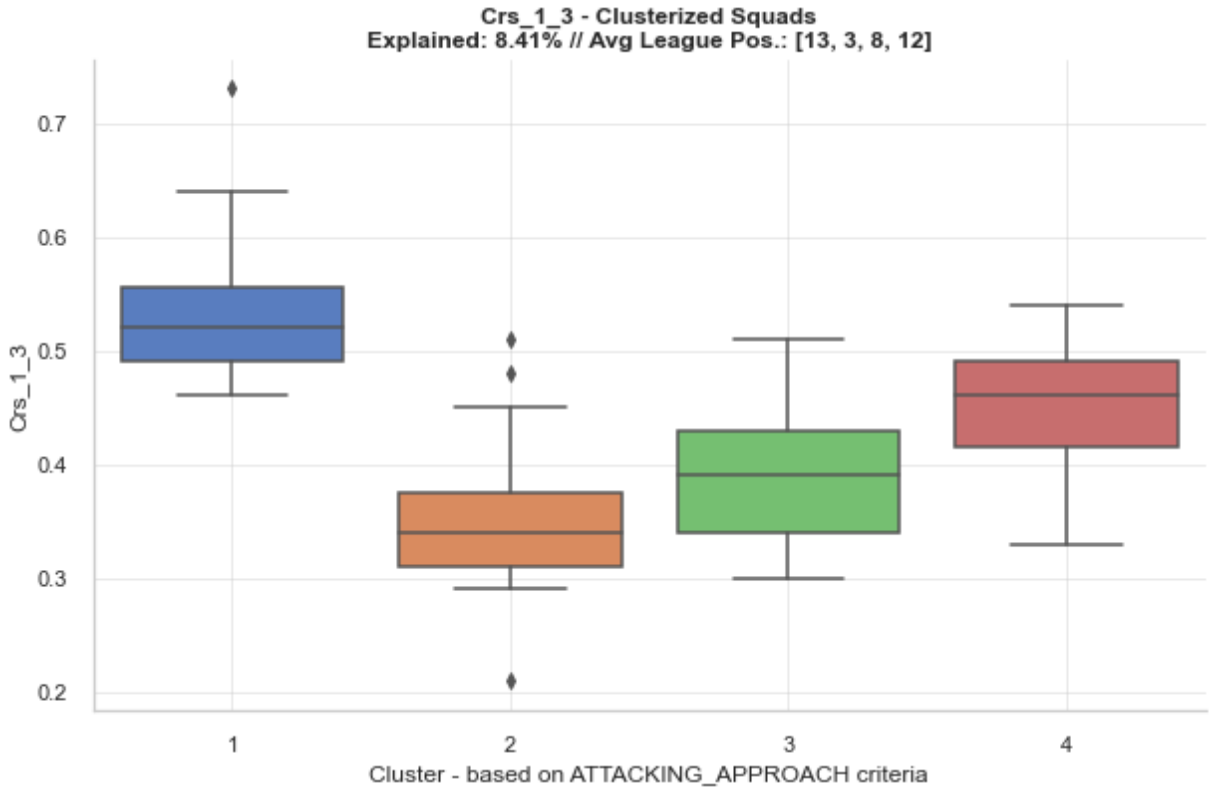


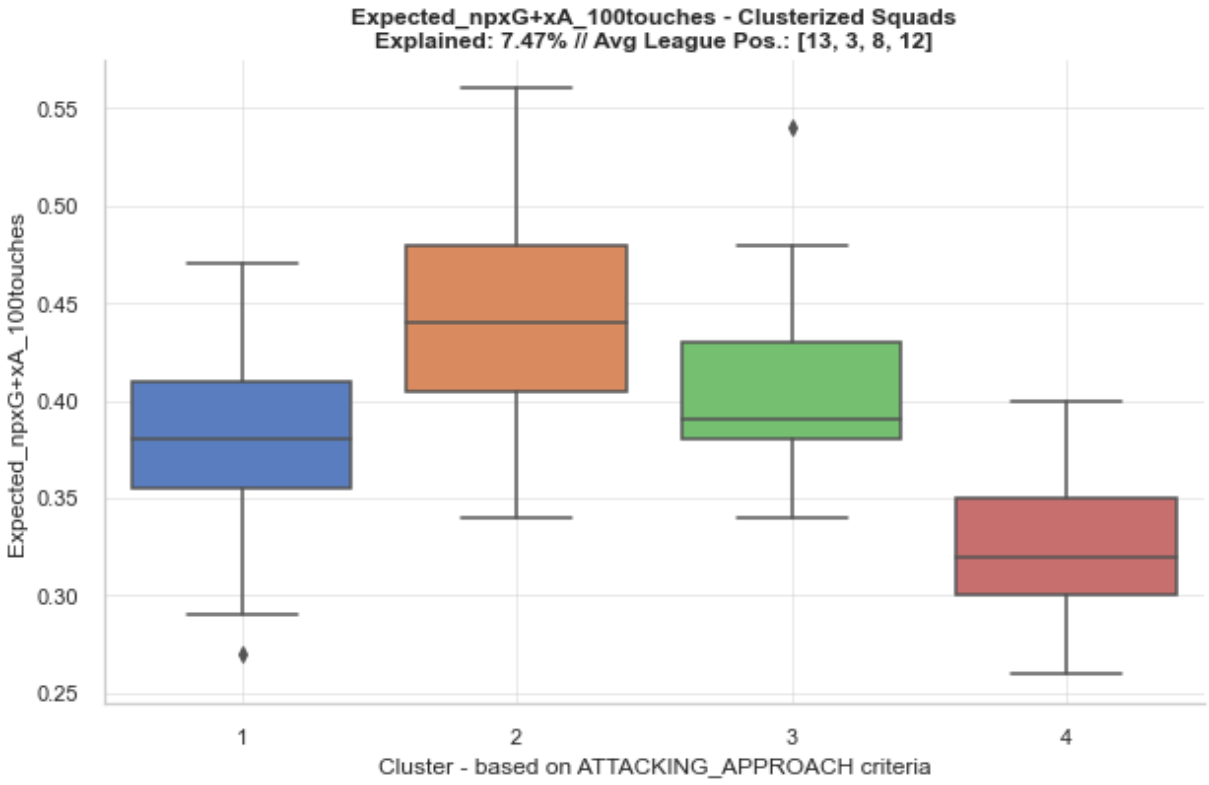


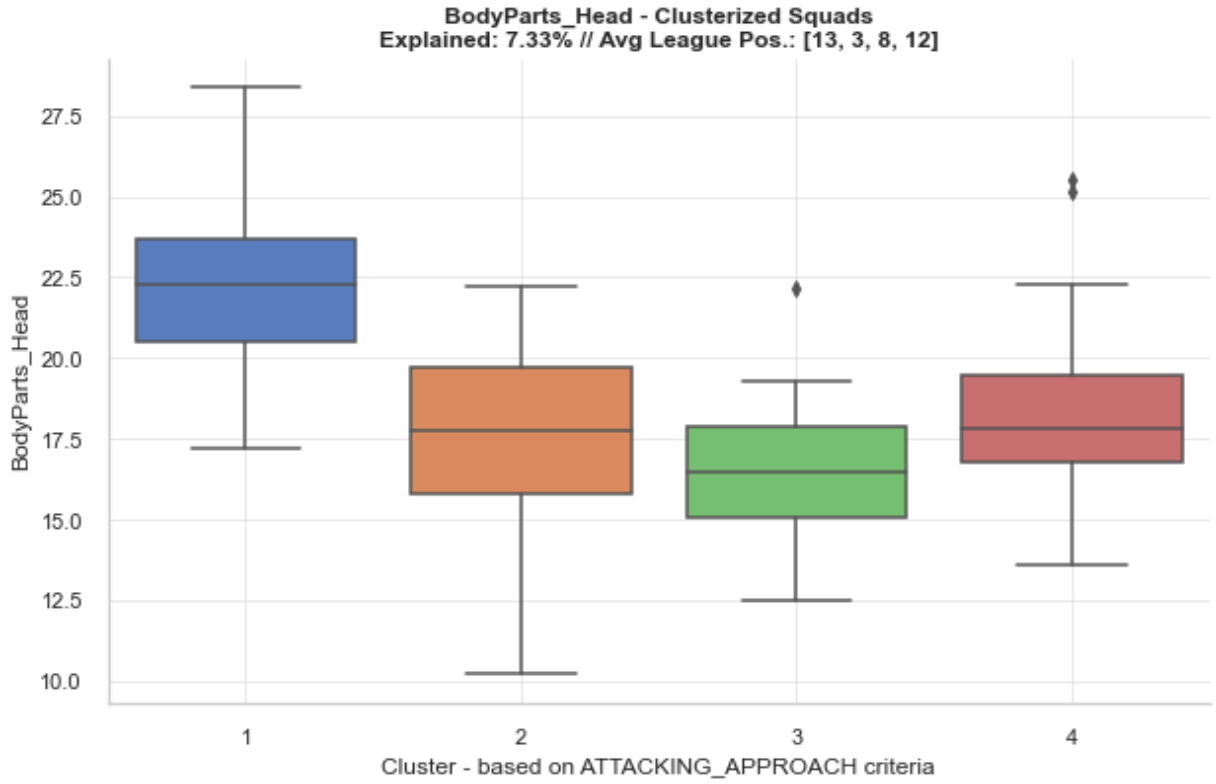
----------CATEGORY: attacking_approach----------


Sh_T100 - Clusterized Squads
Explained: 17.24% // Avg League Pos.: [13, 3, 8, 12]
Sh_T100
2.8
2.6
2.4
2.2
2.0
1.8
1
2
3
4
Cluster - based on ATTACKING_APPROACH criteria


SCATypes_PassLive - Clusterized Squads
Explained: 11.21% // Avg League Pos.: [13, 3, 8, 12]
SCATypes_PassLive
24
22
20
18
16
14
12
10
8
1
2
3
4
Cluster - based on ATTACKING_APPROACH criteria


Crs_1_3 - Clusterized Squads
Explained: 8.41% // Avg League Pos.: [13, 3, 8, 12]
Crs_1_3
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
4
Cluster - based on ATTACKING_APPROACH criteria


Expected_npxG+xA_100touches - Clusterized Squads
Explained: 7.47% // Avg League Pos.: [13, 3, 8, 12]
Expected_npxG+xA_100touches
0.55
0.50
0.45
0.40
0.35
0.30
0.25
1
2
3
4
Cluster - based on ATTACKING_APPROACH criteria



Segmentación de Equipos: Análisis del Resultado

Disposición

- Uno o dos delanteros (o dos mediapuntas), tres centrales – todos los sistemas de tres centrales. EJ: **CHELSEA, ATALANTA, ATLETICO**
- Dos delanteros o uno + un mp, cuatro defensas – 4-4-2/4-4-1-1 – EJ: **ATHLETIC, ELCHE, GETAFE**
- Un delantero, cuatro defensas, doble pivote – 4-2-3-1 – EJ: **BAYERN, DORTMUND**
- Un delantero, cuatro defensas – 4-3-3 o 4-1-4-1 – **MANCHESTER CITY, REAL MADRID, BARCELONA**

Defensa

- Intensidad mixta en presión (o cambiante a lo largo del campo), bloque medio-bajo – **REAL MADRID, SEVILLA, ROMA, NAPOLI, MAN UTD, REAL SOCIEDAD**
- Baja intensidad de presión, bloque bajo, pasividad en campo propio – **ELCHE, METZ, NORWICH, VENEZIA, WEST HAM, WOLVES, UNION**
- Alta intensidad en presión, pocos pases rivales por acción defensiva, presión alta, agresividad, poco tiempo de juego del rival, defensa mas lejos de porteria – **VALENCIA, MANCHESTER CITY, DORTMUND, BAYERN, BARCELONA**

Posesión

- Juego más elaborativo, más elaboración en tercio final, salida de balón en corto, acostumbrados a jugar ante rivales replegados y llegar arriba con balón controlado – **MAN CITY, REAL MADRID, BARCELONA, BAYERN, BETIS, LEIPZIG, CELTA**
- Juego más directo, más elaboración en tercio final (segunda jugada), contraataque y balón largo – **BURNLEY, CADIZ, VALENCIA, GETAFE, EVERTON, UNION**

- Juego más directo, salida mixta y elaboración mixta, centros a área, muy directos en tramo final de campo. Sin dominar siempre, juego más dinámico y rápido – **LEICESTER, ESPANYOL, REAL SOCIEDAD, NORWICH**

Finalización

- Término medio, acelera mucho el ataque en tramo final y no le importa un intercambio de golpes, facilidad para generar valor una vez se instala en área rival pero no siempre con la mejor selección de tiro – EJ: **REAL MADRID, MILAN, ROMA, BETIS**
- Ocasiones en pocos toques, mucho disparo desde fuera del área, más balón parado, centros, contragolpe y segunda jugada – EJ: **BRENTFORD, HERTHA, ASTON VILLA, ATHLETIC CLUB**
- Mayor paciencia, mejor selección de tiro, mas tiros procedentes de pase en juego vivo, facilidad para generar valor una vez se instala en área rival – **MANCHESTER CITY, BARCELONA, DORTMUND, VILLARREAL, LEIPZIG, BAYERN, LIVERPOOL, INTER**
- Paciencia, buen buildup, pocos tiradores con determinación pero calidad en la generación, y muy pocas individualidades que marquen la diferencia arriba – **REAL SOCIEDAD, SEVILLA, NORWICH, CELTA**

In [19]:
```
df['cluster_clas'] = df['disposition_cluster'] + df['defensive_approach_cluster'] +d
df.cluster_clas.value_counts()
```

Out[19]:
```
4221    5
1112    5
3334    5
1213    4
3112    4
3321    4
2112    3
3213    3
1324    3
4334    3
1234    3
1334    3
2113    2
1221    2
2334    2
1121    2
2214    2
2321    2
3131    2
1134    2
3221    2
4321    2
3233    1
1113    1
4121    1
4232    1
3323    1
3212    1
3123    1
1313    1
3133    1
4222    1
2212    1
2213    1
4123    1
```

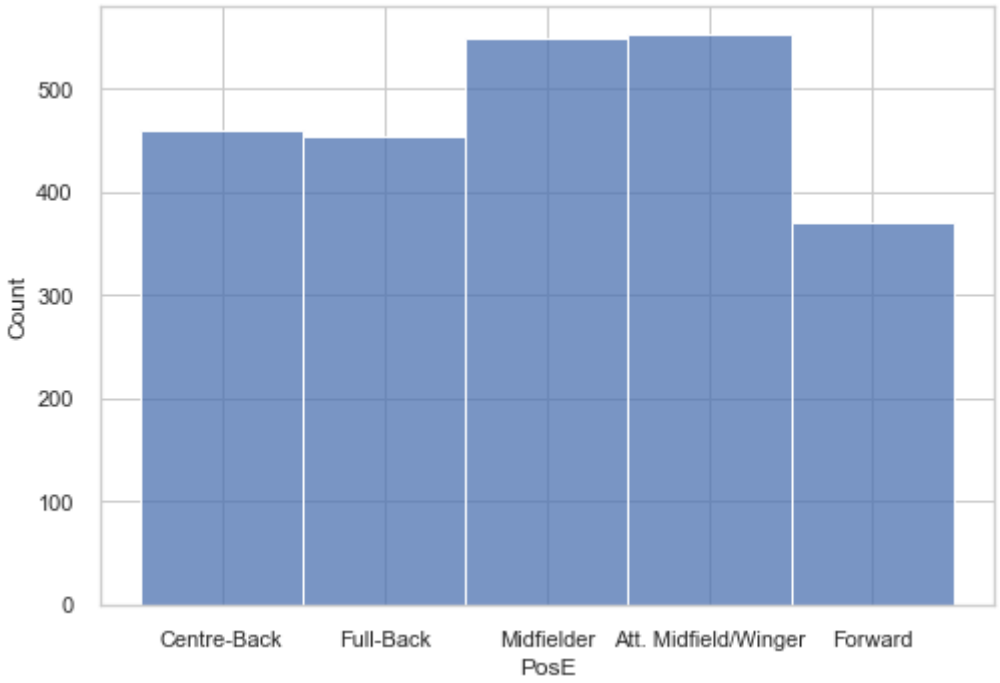
```
3134    1
3121    1
1331    1
4333    1
3114    1
2221    1
3222    1
2121    1
3234    1
3333    1
4324    1
2234    1
4234    1
2223    1
1224    1
3124    1
4213    1
4231    1
1212    1
4212    1
Name: cluster_clas, dtype: int64
```

Segmentación de Jugadores

A continuación, detallamos las posiciones principales y secundarias existentes en el dataset de jugadores, así como la asignación de variables a cada posición principal, que es el criterio por el cuál categorizaremos el proceso.

In [20]:
```
sns.histplot(players.PosE)
```

Out[20]:
```
<AxesSubplot:xlabel='PosE', ylabel='Count'>
```



In [21]:
```
for i in players.PosE.unique():

    print('----',i.upper(),'----')
    print(players[players.PosE==i]['POS'].value_counts())
    print('\n')
```

```
---- CENTRE-BACK ----
```

```
Centre-Back    459
Name: POS, dtype: int64


---- FULL-BACK ----
Right-Back    240
Left-Back     214
Name: POS, dtype: int64


---- MIDFIELDER ----
Central Midfield      334
Defensive Midfield    216
Name: POS, dtype: int64


---- ATT. MIDFIELD/WINGER ----
Attacking Midfield    210
Left Winger           178
Right Winger          165
Name: POS, dtype: int64


---- FORWARD ----
Centre-Forward    371
Name: POS, dtype: int64
```

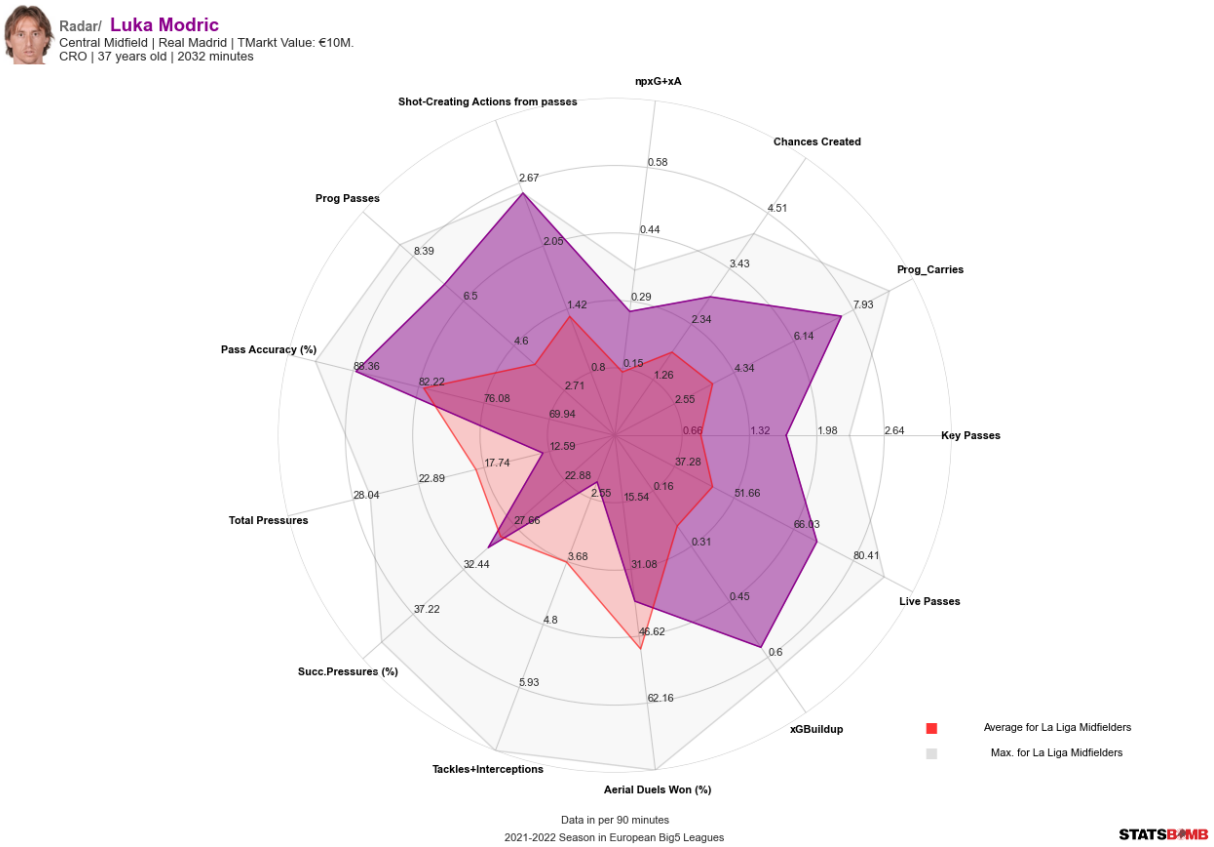
In [22]:

```
players = pd.merge(players,df,how='left',on='Squad')

players.drop('Age2',inplace=True,axis=1)
for i in players.columns:
    if 'DIF_' in i:
        players.drop(i,axis=1,inplace=True)


players['ID_understat'] = players['ID_understat'].astype(str)
players['ID_tmarkt'] = players['ID_tmarkt'].astype(str)

encoder = OneHotEncoder(sparse=False)
codif = encoder.fit_transform(players[['foot']])
players[encoder.get_feature_names()] = codif

for i in ['x0_Both','x0_Left','x0_Unknown']:
    if i in players.columns:
        players.drop(i,inplace=True,axis=1)
players['Wing_Natural'] = np.where(((players['x0_Right']==1) & (players.POS.str.cont
```

En el box de arriba se unen las tablas de cluster de equipos con el dataset bruto de jugadores. Es momento de realizar algunas transformaciones sobre los datos base, como marcar todos los IDs como strings y codificar el atributo de pierna buena de cada jugador. Ello nos permitirá extraer un dato clave en el caso de los jugadores de banda: si juegan a pie natural o a pie cambiado. Dicho dato nos servirá de utilidad al perfilar a los extremos.

Antes de proceder a la aplicación del clustering sobre los futbolistas, realizamos una pequeña exploración gráfica sobre algún jugador. De este modo, podemos observar los gráficos de radar disponibles, que emplearemos a lo largo del proyecto para comparar y medir los atributos de los jugadores

In [23]:
```
pp.sradar('Luka Modric')
```



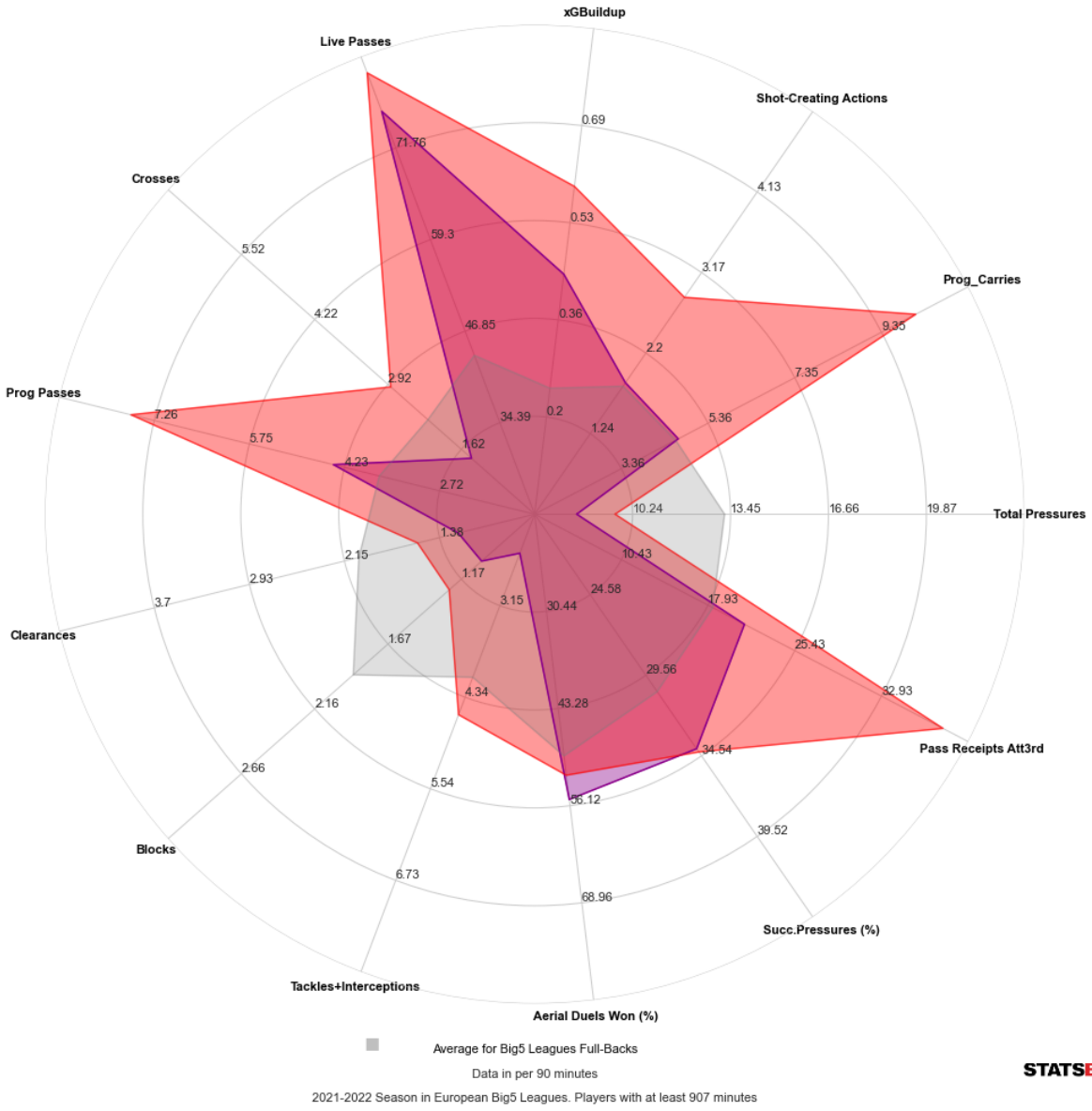
In [24]:
```
pp.radar_comp('Kyle Walker','Joao Cancelo')
```

Kyle Walker

Right-Back | Manchester City | TMarkt Value: €26M.
ENG | 32 years old | 1755 minutes

Joao Cancelo

Right-Back | Manchester City | TMarkt Value: €63M.
POR | 28 years old | 3227 minutes



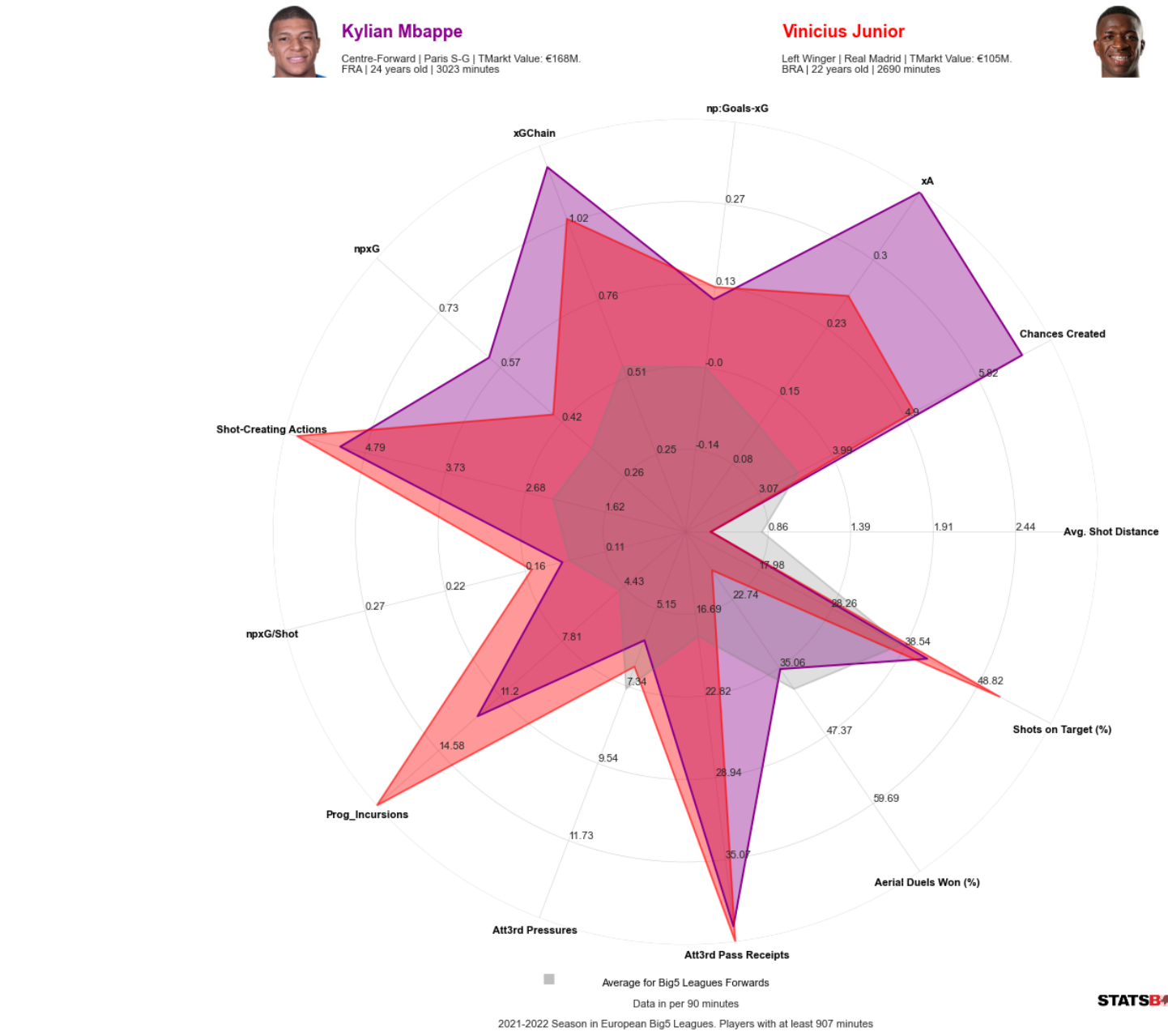
Data in per 90 minutes

2021-2022 Season in European Big5 Leagues. Players with at least 907 minutes

STATSBOMB

In [93]:

```
pp.radar_comp('Kylian Mbappe','Vinicius Junior')
```



In [26]:

```
cbs = list(set(['AerialDuels_Won%','Interceptions*','Blocks_Pass','Blocks_Sh','Clr/D
        'idxpos_t','idxpos_p','Tkl/DefAction100','VsDribbles_Tkl%','Touches_htouch_a
        'Int','Long_Cmp%','Medium_Cmp%','NonShAtt%','PassTypes_Crs','PassTypes_Live'
        'PassTypes_Sw','PassTypes_Press','PassTypes_TB','Performance_CrdY','Performa
        'Performance_Recov','Pressures_%','Pressures_hPress','Pressures_Succ','Press
        'Prog','SCATypes_Def','Short_Cmp%','Tackles%','Tackles_Def3rd%',
        'Tkl+Int','Carries_100touches','Carries_Prog_100_Inc','Touches_Live','Total_
        'Touches_DefPen%','1_3'
    ]))

mid = list(set([
    'KP','Carries_Prog','CC_Rec100','SCATypes_PassLive','Prog','xGBuildup_n',
    'Total_Cmp%','Pressures_Press','Pressures_%','Tkl+Int','AerialDuels_Won%','PassT
    'Touches_Att3rd','Touches_Def3rd','Touches_htouch_ata','Carries_100touches','Tou
    'Carries_Prog_100_Inc','Carries_1_3','Incursions1_3','Prog','1_3','KP_100_Prog',
    'Performance_Recov','Pressures_hPress','Pressures_Succ','Pressures_Press',
    'Tkl+Int','PassTypes_Sw','PassTypes_Press','PassTypes_TB','PassTypes_Crs','NonSh
    'idxpos_p','idxpos_t','Tkl/DefAction100','Interceptions*','Blocks_Pass','Blocks_
    'Expected_npxG/Sh','Sh_T100','xA/KP','Pass_Prog_100_p','Tackles%','Drib_Rec100',
    'Touches_DefPen%','PPT3%'
    ]))
```

```
flb=list(set(['Carries_Prog','Crs_ProgPass','Crs_100_Passes','PassTypes_Crs','Prog',
     'Carries_Prog_100_Inc','Total_Cmp%','Short_Cmp%','Tkl/DefAction100','Intercepti
     'Blocks_Pass','Blocks_Sh','NonShAtt%','idxpos_p','idxpos_t','PassTypes_Sw','Tkl
     'PassTypes_TB','Incursions1_3','Crs_1_3','Ground_Pass%','Touches_Att3rd',
     'Touches_Def3rd','Touches_htouch_ata','Pressures_Press','Pressures_%','Sh_T100'
     'Pressures_hPress','xGBuildup_n',"VsDribbles_Tkl%",'Touches_DefPen%','SCATypes_
     'Performance_Recov','Clr/DefPen_Touches','KP_100_Prog','xA/KP','Pass_Prog_100_p
     'press+tch','Tackles%']
            ))

attm=list(set(['Crs_ProgPass','Crs_100_Passes','PassTypes_Crs','Prog','Incursions1_3
      'SCA_100touchesAtt3rd','Pass_Prog_100_p','Carries_100touches','Receiving_Prog_
      'Carries_Prog_100_Inc','Pressures_hPress','Touches_htouch_ata','Dribbles_Succ%
      'Drib_Rec100','KP_100_Prog','Carries+','press+tch','Touches_AttPen%',
      'Expected_npxG+xA_100touches','Expected_npxG/Sh','XGChain_n','xGBuildup_n',
      'idxpos_t','idxpos_p','PPT3%','%ag_Press','NonShAtt%','PassTypes_TB','Incursio
      'Crs_1_3','Ground_Pass%','SCATypes_Drib','SCATypes_Sh','SCATypes_PassLive','To
      'Pressures_%','CC_RecProg','CC_Rec100','PPA','Receiving_Targ','Wing_Natural'
      ]))

fwd=list(set(['Crs_ProgPass','PassTypes_Crs','Prog','Incursions1_3','CC_Rec100',
      'SCA_100touchesAtt3rd','Pass_Prog_100_p','Receiving_Prog_100Rec',
      'Carries_1_3','Pressures_hPress','Touches_htouch_ata','Dribbles_Succ%',
      'Drib_Rec100','KP_100_Prog','press+tch','Touches_AttPen%',
      'Expected_npxG+xA_100touches','Expected_npxG/Sh','XGChain_n','xGBuildup_n',
      'idxpos_t','idxpos_p','PPT3%','%ag_Press','NonShAtt%','PassTypes_TB','Incursio
      'Crs_1_3','Ground_Pass%','SCATypes_Drib','SCATypes_Sh','SCATypes_PassLive','To
      'Pressures_%','CC_RecProg','CC_Rec100','BodyParts_Head','PPA','height'
      ]))

pos_dict= {'Centre-Back':cbs,
           'Midfielder':mid,
           'Att. Midfield/Winger':attm,
           'Full-Back':flb,
           'Forward':fwd}
```

La función que viene a continuación realizará la clusterización de los jugadores en base a las todas las posiciones mencionadas arriba. Cada posición principal tiene, a su vez, unas variables clave. Más abajo analizaremos los resultados con objeto de perfilar cada uno de los clusters que se generen para todas las posiciones.

En primer lugar, el dataset tomará las columnas citadas anteriormente para la posición en cuestión, que se normalizarán vía MinMaxScaler. Se le aplicará el algoritmo VarianceThreshold, de cara a detectar si existen variables constantes o prácticamente constantes. Aplicaremos a continuación PCA hasta el 90% de explicación y procederemos con KKeans.

A continuación, en primer lugar, mediremos la correlación entre las variables y descartaremos aquellas que presenten un valor elevado en la matriz. Se extraen, tras ello, las variables reales más explicativas respecto a los clusters otorgados por el kmeans aplicado mediante un **RandomForestClassifier**, con objeto de aclarar y simplificar el análisis.

Con ello, obtendremos la clusterización de los atributos de los futbolistas en base a su posición. La función nos devolverá distintos plots que nos servirán de apoyo en nuestro análisis, el detalle del funcionamiento del proceso de entrenamiento de los procesos PCA y ElbowMethod y un dataset con la asignación a los clusters. Además, tendremos también un dataframe con la variabilidad explicada por cada una de las columnas.

In [27]:

```
def player_clustering(position):
    #np.random.seed(0)
    fs={}
    frames={}
    ks = {}
    pcas={}
    positions=[]
    if type(position)==list:
        positions=position
    else:
        positions.append(position)
    for position in positions:

        pl = players[players.Min>=min_minutes]
        #pl_clustering = pd.DataFrame(players[['Player']],columns=['Player'])
        positioner = pl[pl.PosE==position]
        name = positioner.PosE.unique()[0]
        print('-----Starting clustering for {}s-----'.format(name))
        data_pos= pl[pl['PosE']==name]
        data_pos.reset_index(inplace=True)
        data_pos.drop('index',inplace=True,axis=1)
        data = data_pos[pos_dict[name]]

        data_norm = scaler.fit_transform(data)
        sel = VarianceThreshold(threshold=0.01)
        sel.fit(data_norm)
        print('Num. of variables that are not constant or cuasi-constant: ', sum(sel
        data_norm =  sel.transform(data_norm)
         # 2D PCA for the plot
        pca_app = 0
        print('Setting PCA for training')
        for p in range(2,100):
            pca = PCA(n_components = p,random_state=0,svd_solver='arpack')
            #print('PCA({}) for {}'.format(p,name))
            #reduced = pd.DataFrame(pca.fit_transform(data_norm))
            pca.fit(data_norm)
            #print('Explained var = {}'.format(round(sum(pca.explained_variance_rati
            if sum(pca.explained_variance_ratio_)>=0.9:
                pca_app = p
                print('Selected PCA: {}'.format(p))
                break
            else:
                continue

        reduced = pd.DataFrame(pca.transform(data_norm))
        # We intend to have, as min, four clusters
        plt.figure(figsize=(10,5))
        view = elbow_method(reduced,10)
        if view.elbow_value_>=5:
            n = 4
        else:
            n = view.elbow_value_
        #view.show()
        k = KMeans(n_clusters = n,random_state=0)
        pcas[position] = pca_app
        # We intend to have, as min, four clusters

        ks[position] = n
        reduced['cluster'] = k.fit_predict(reduced)
        reduced['cluster'] = reduced['cluster'] +1
        data = pd.merge(data,reduced[['cluster']],how='left',left_index=True,right_i
        corr_matrix = data.corr()

        upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np
```

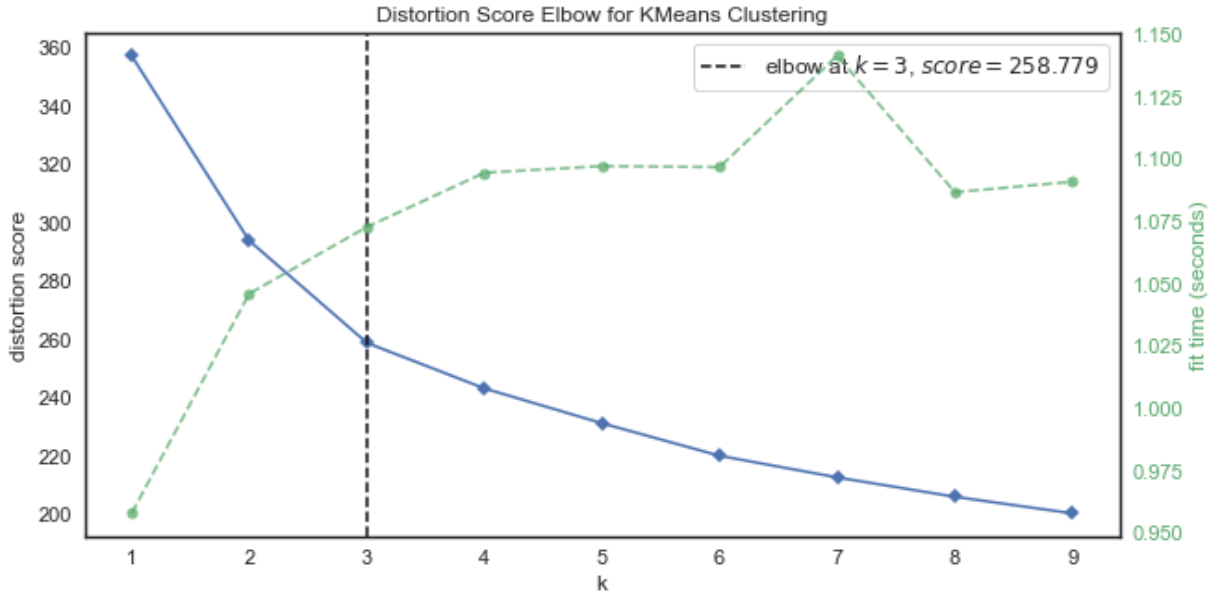
```
upper
to_drop = [column for column in upper.columns if any(upper[column] > 0.75)]
data= data.drop(to_drop,axis=1)
if 'cluster' in data.columns:
    data.drop('cluster',axis=1,inplace=True)
# Based on important information


clf = RandomForestClassifier(n_estimators=100, random_state=0)

#data = pd.merge(data,reduced[['cluster']],how='left',left_index=True,right_

clf.fit(scaler.fit_transform(data), reduced['cluster'])
feature_scores = pd.Series(clf.feature_importances_, index=data.columns).sor

f, ax = plt.subplots(figsize=(30, 20))
ax = sns.barplot(x=feature_scores, y=feature_scores.index)
ax.set_title("Feature scores of {} in Players clusterization".format(name.re
ax.set_yticklabels(feature_scores.index,size=22)
ax.set_xlabel("Feature importance score",size=20)
ax.set_ylabel("Features",size=20)
plt.savefig(os.path.join(os.getcwd(),'Model','Players')+"/Feature_scores_{}"
plt.show()

feature_scores_to_drop = feature_scores[feature_scores.cumsum()>=.9]
data.drop(list(feature_scores_to_drop.index),inplace=True,axis=1)

feature_scores = feature_scores[~feature_scores.index.isin(list(feature_scor


plt.plot(range(0,p), list(pca.explained_variance_ratio_))
plt.ylabel('Explained Variance')
plt.xlabel('Principal Components')
plt.xticks(range(0,p),rotation=60)
plt.title('Explained Variance Ratio PCA{}'.format(pca_app))
plt.show()



cl=[]
for i in reduced.columns:
    if type(i)==int:
        i+=1
        cl.append('pc'+str(i))
    else:
        cl.append(i)
reduced.columns = cl
if 'Player' in cl:
    reduced.drop('Player',inplace=True,axis=1)
#reduced.columns=['pc1','pc2','cluster']
#centroids = k.cluster_centers_
reduced = pd.merge(reduced,data_pos[['Player']],how='left',left_index=True,r

sns.set(style="white")
ax= sns.lmplot( x="pc1", y="pc2", hue='cluster', data = reduced, legend=True
 size = 8, scatter_kws={"s": 100},aspect=1.5,height=4, palette='muted')
ax=plt.gca()
ax.set_title('PCA({}) Clustering Distribution - {}s // Two Principal Compone
"""
texts = []
for x, y, s in zip(reduced.pc1, reduced.pc2, reduced.Player):
    if abs(x)>abs(reduced['pc1'].mean()) or abs(y)>abs(reduced['pc2'].mean()
        texts.append(plt.text(x, y, s))
"""
```

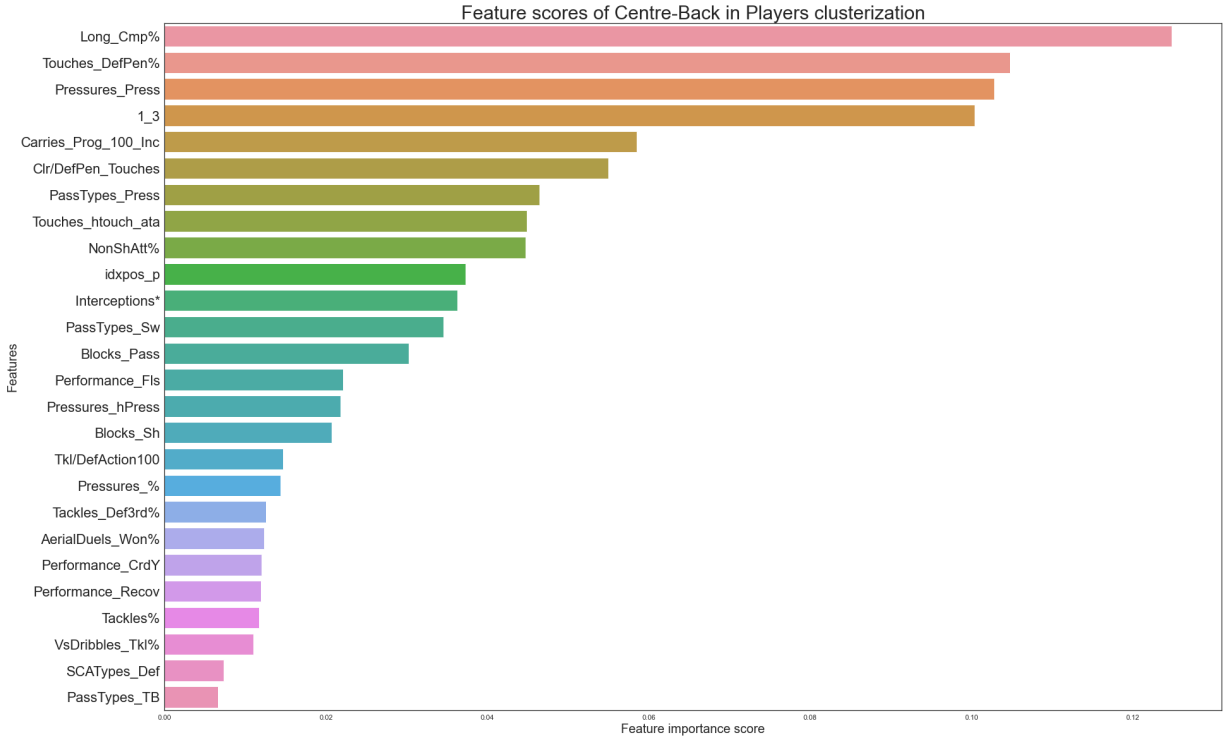
```
        plt.tick_params(labelsize=10)
        plt.xlabel("PC1", fontsize = 15)
        plt.ylabel("PC2", fontsize = 15)
        plt.tight_layout()
        plt.show();

        data = pd.merge(data,reduced[['cluster']],how='left',left_index=True,right_i
        #data_gr = data.groupby(by='cluster',as_index=False).mean()
        player_clustering = data_pos
        player_clustering['cluster'] = data[['cluster']]

        frames[position]= reduced
        fs[position]=pd.DataFrame(feature_scores,columns=[name])


    return frames, fs, ks, pcas
```

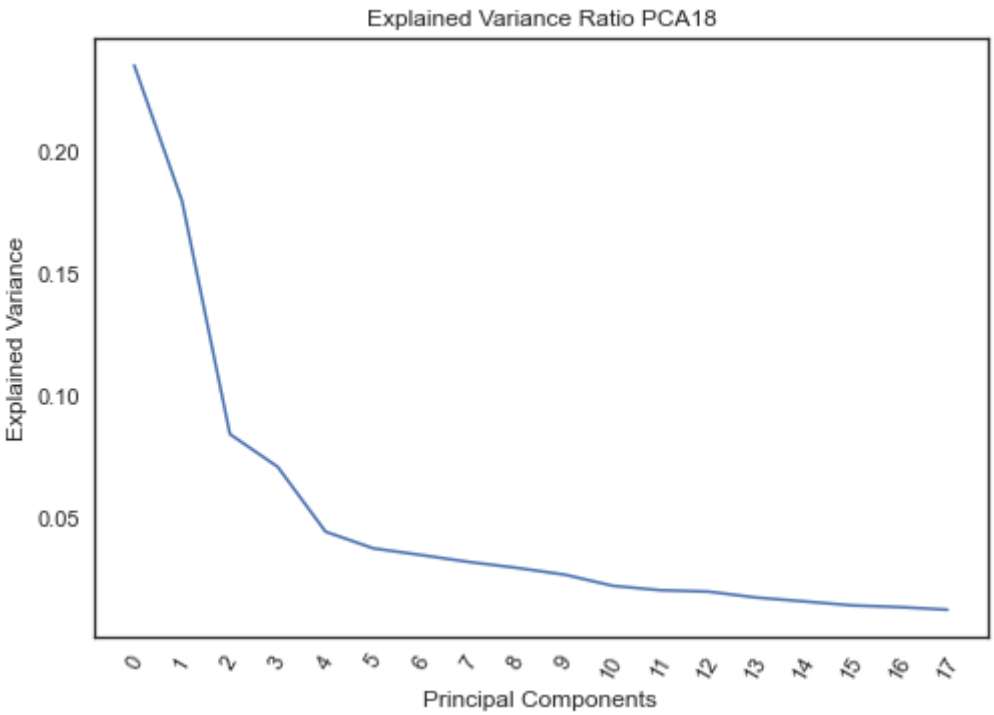
In [28]:

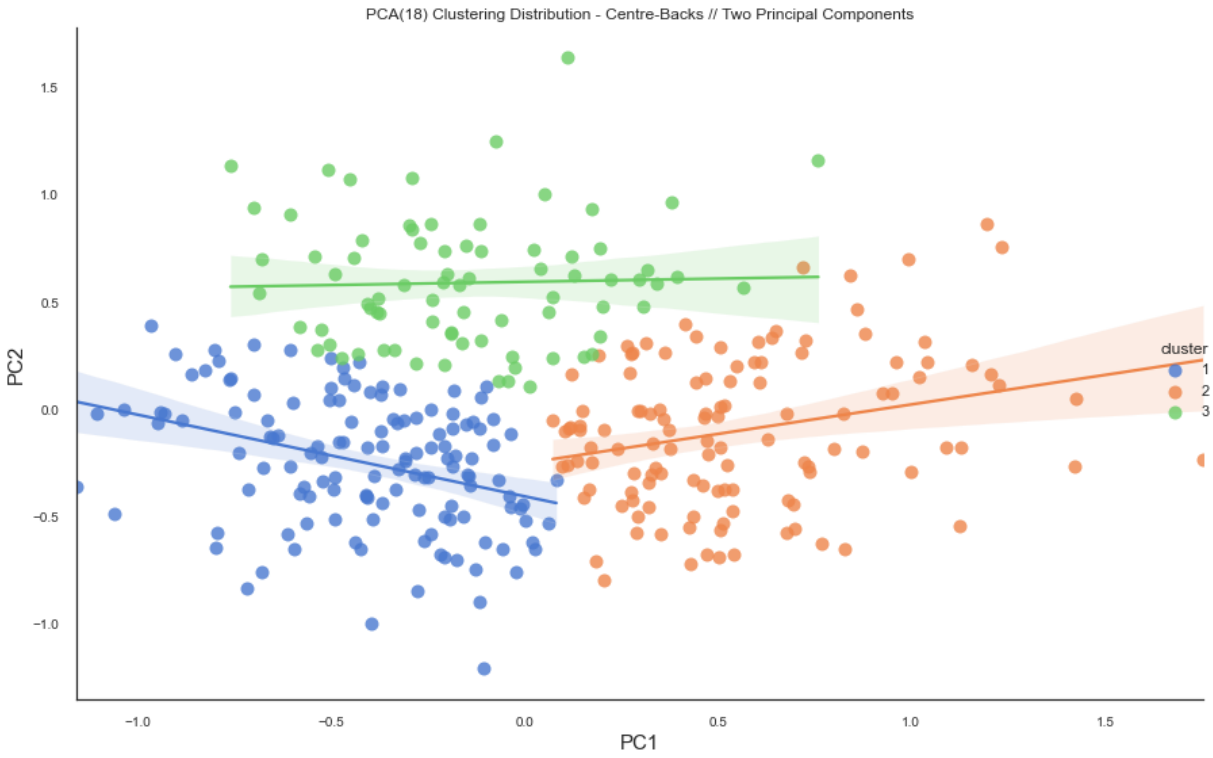
```
df,features,ks,pcas = player_clustering(list(players.PosE.unique()))
sns.set(style="whitegrid")
players2 = players[players['Min']>=min_minutes]
```

```
-----Starting clustering for Centre-Backs-----
Num. of variables that are not constant or cuasi-constant:  39  out of 39
Setting PCA for training
Selected PCA: 18
```

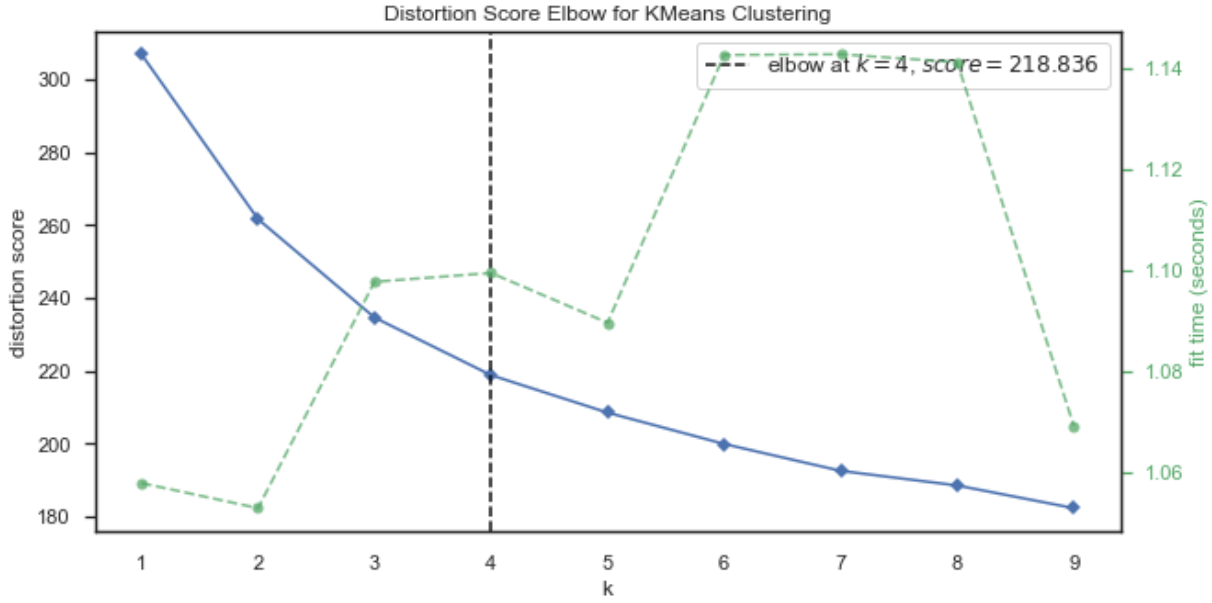


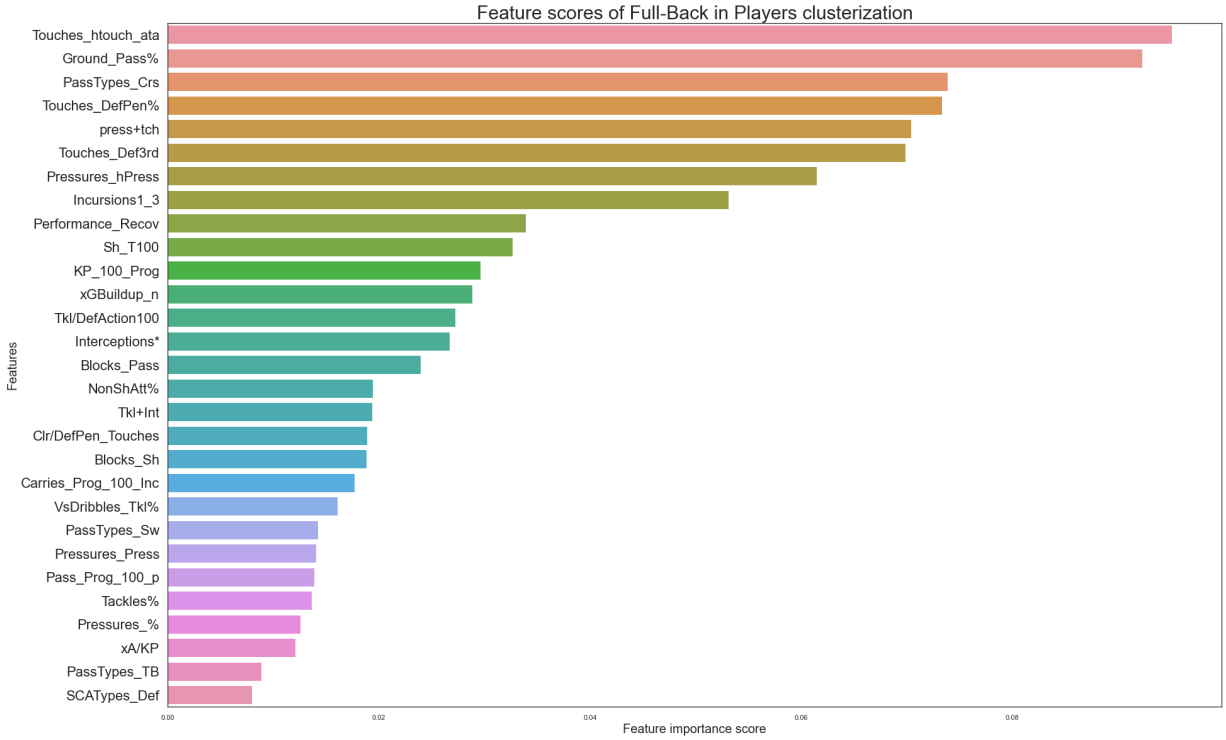

Feature scores of Centre-Back in Players clusterization
Long_Cmp%
Touches_DefPen%
Pressures_Press
1_3
Carries_Prog_100_Inc
Clr/DefPen_Touches
PassTypes_Press
Touches_htouch_ata
NonShAtt%
idxpos_p
Interceptions*
PassTypes_Sw
Blocks_Pass
Performance_Fls
Pressures_hPress
Blocks_Sh
Tkl/DefAction100
Pressures_%
Tackles_Def3rd%
AerialDuels_Won%
Performance_CrdY
Performance_Recov
Tackles%
VsDribbles_Tkl%
SCATypes_Def
PassTypes_TB
Features
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Feature importance score

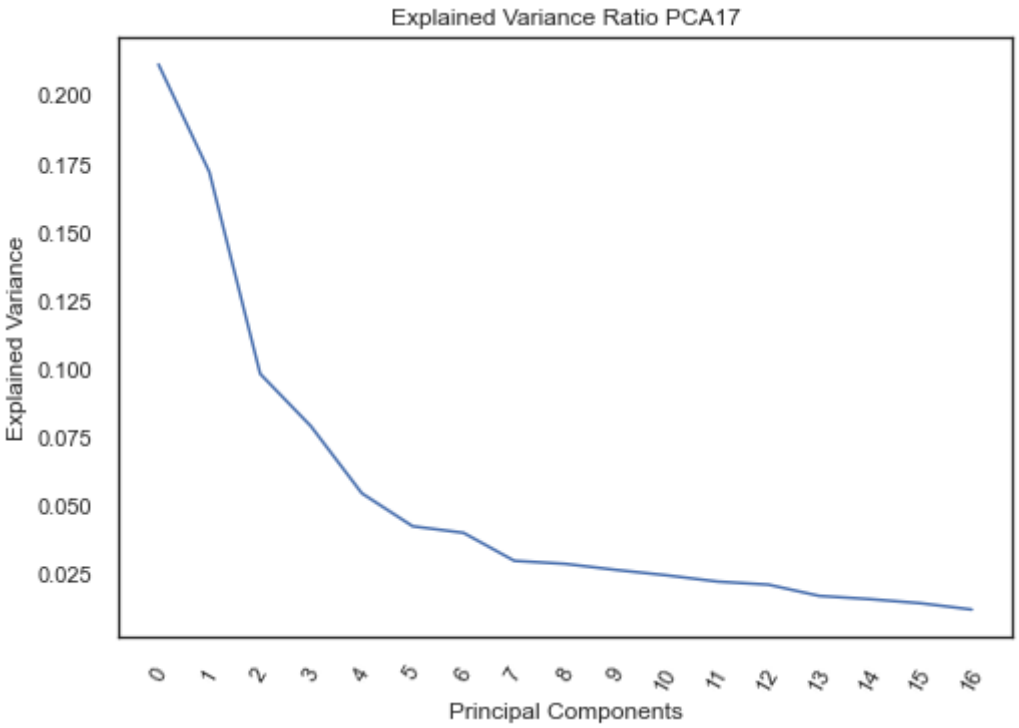

Explained Variance Ratio PCA18
Explained Variance
0.20
0.15
0.10
0.05
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
Principal Components

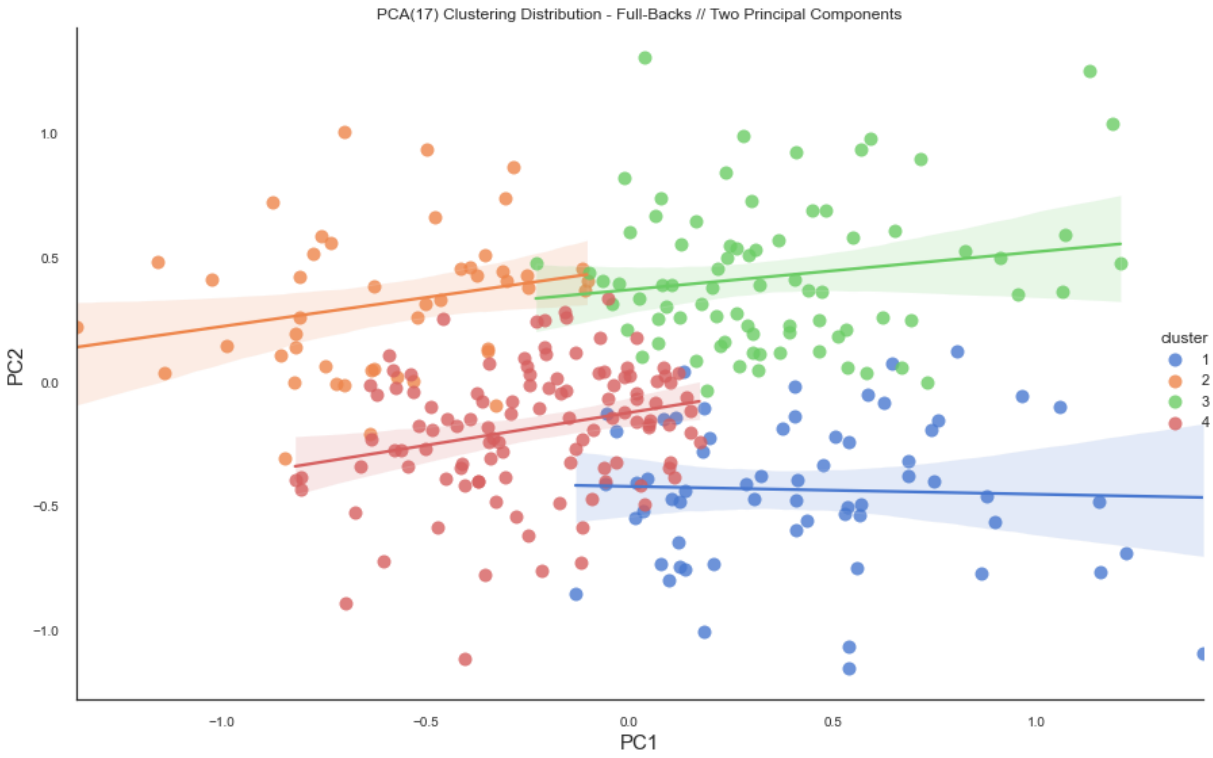



```
-----Starting clustering for Full-Backs-----
Num. of variables that are not constant or cuasi-constant:  40  out of 40
Setting PCA for training
Selected PCA: 17
```

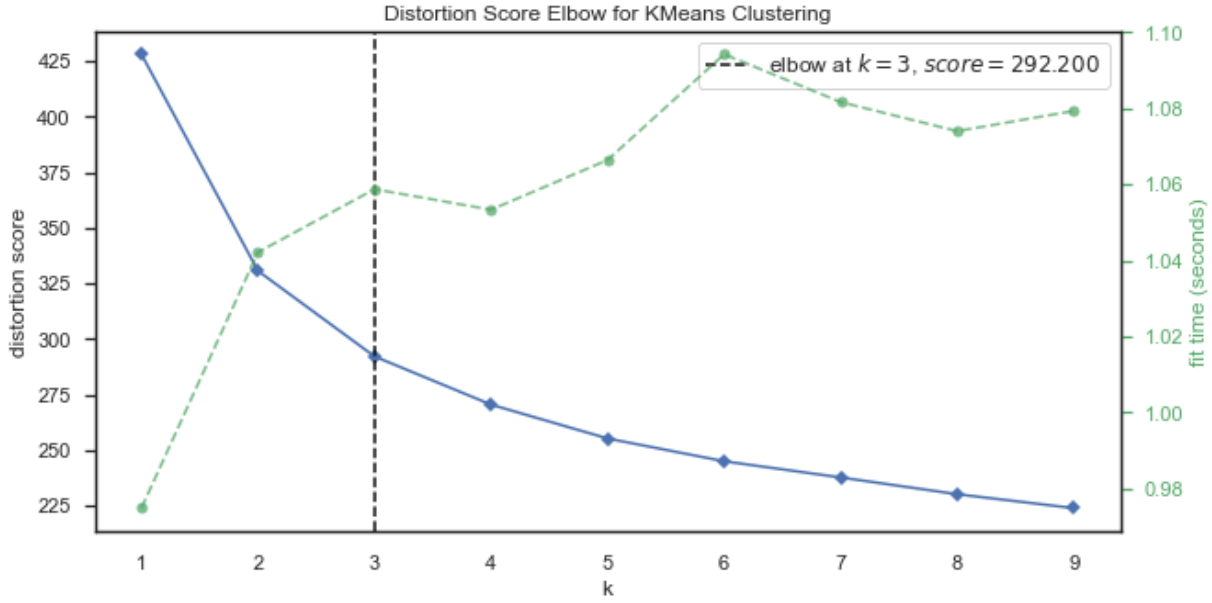


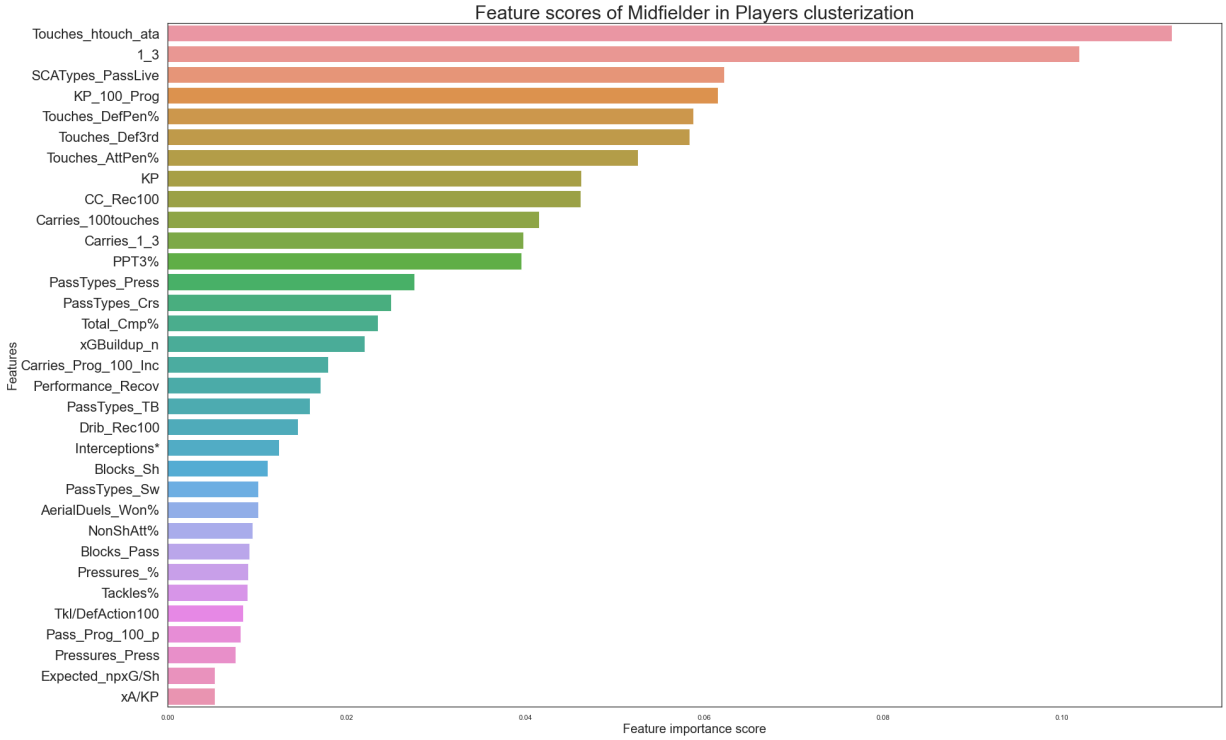

Feature scores of Full-Back in Players clusterization
Touches_htouch_ata
Ground_Pass%
PassTypes_Crs
Touches_DefPen%
press+tch
Touches_Def3rd
Pressures_hPress
Incursions1_3
Performance_Recov
Sh_T100
KP_100_Prog
xGBuildup_n
Tkl/DefAction100
Interceptions*
Blocks_Pass
NonShAtt%
Tkl+Int
Clr/DefPen_Touches
Blocks_Sh
Carries_Prog_100_Inc
VsDribbles_Tkl%
PassTypes_Sw
Pressures_Press
Pass_Prog_100_p
Tackles%
Pressures_%
xA/KP
PassTypes_TB
SCATypes_Def
Features
0.00
0.02
0.04
0.06
0.08
Feature importance score

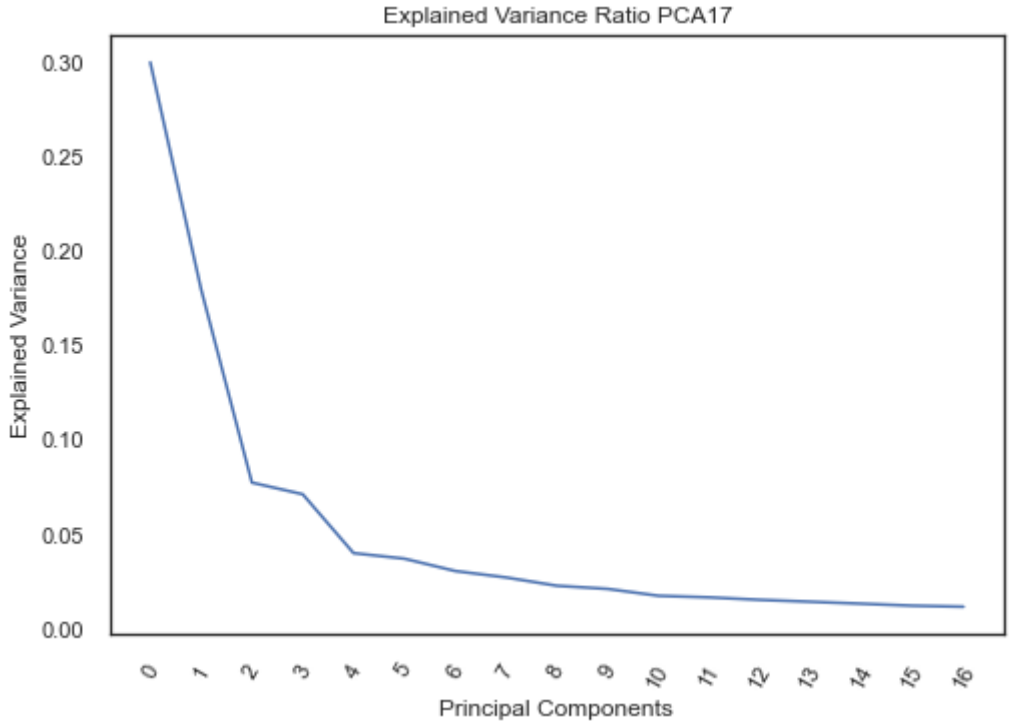

Explained Variance Ratio PCA17
Explained Variance
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Principal Components

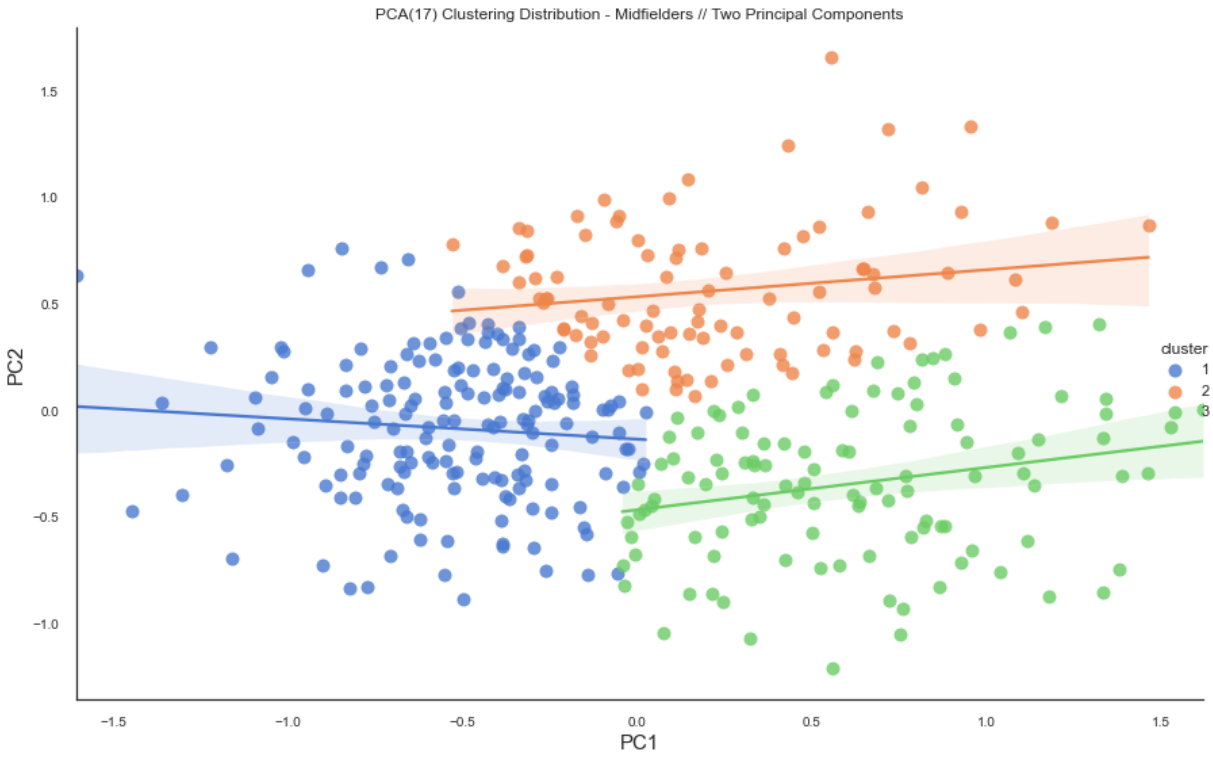



```
-----Starting clustering for Midfielders-----
Num. of variables that are not constant or cuasi-constant:  44  out of 44
Setting PCA for training
Selected PCA: 17
```

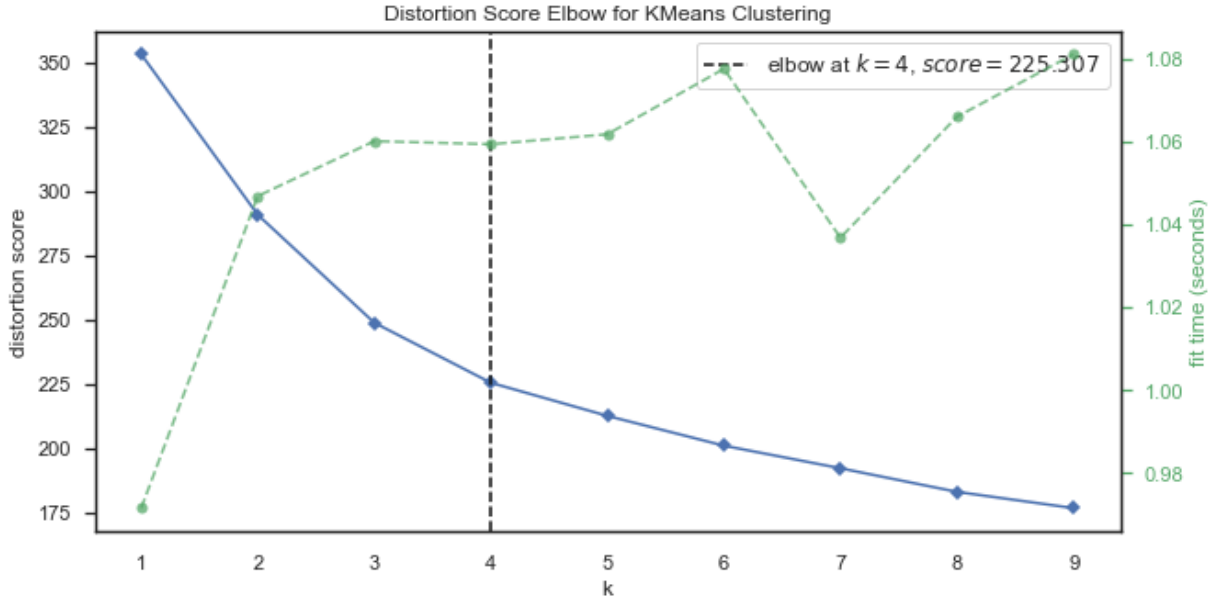


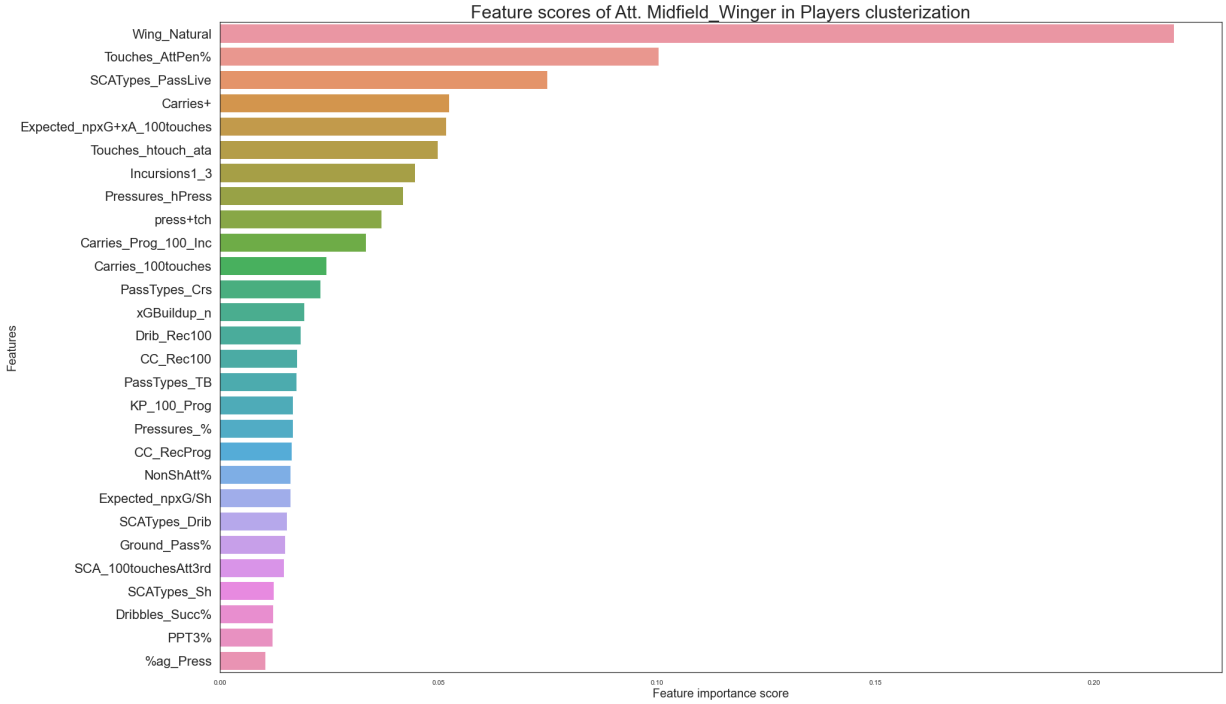

Feature scores of Midfielder in Players clusterization
Touches_htouch_ata
1_3
SCATypes_PassLive
KP_100_Prog
Touches_DefPen%
Touches_Def3rd
Touches_AttPen%
KP
CC_Rec100
Carries_100touches
Carries_1_3
PPT3%
PassTypes_Press
PassTypes_Crs
Total_Cmp%
xGBuildup_n
Carries_Prog_100_Inc
Performance_Recov
PassTypes_TB
Drib_Rec100
Interceptions*
Blocks_Sh
PassTypes_Sw
AerialDuels_Won%
NonShAtt%
Blocks_Pass
Pressures_%
Tackles%
Tkl/DefAction100
Pass_Prog_100_p
Pressures_Press
Expected_npxG/Sh
xA/KP
Features
0.00
0.02
0.04
0.06
0.08
0.10
Feature importance score

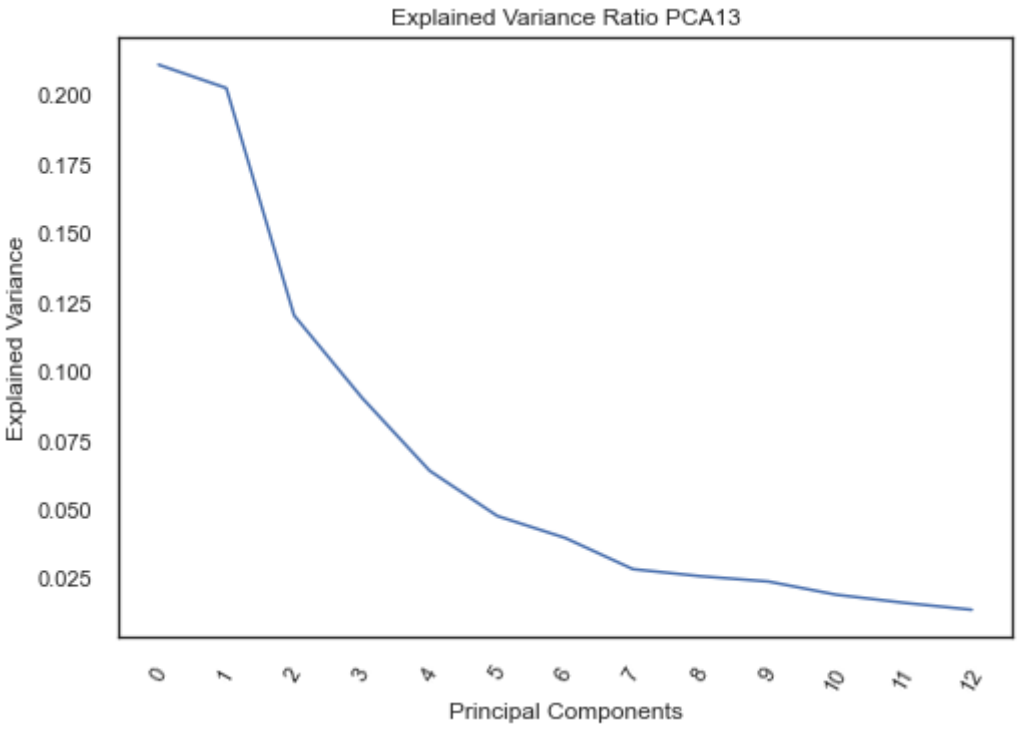

Explained Variance Ratio PCA17
Explained Variance
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Principal Components

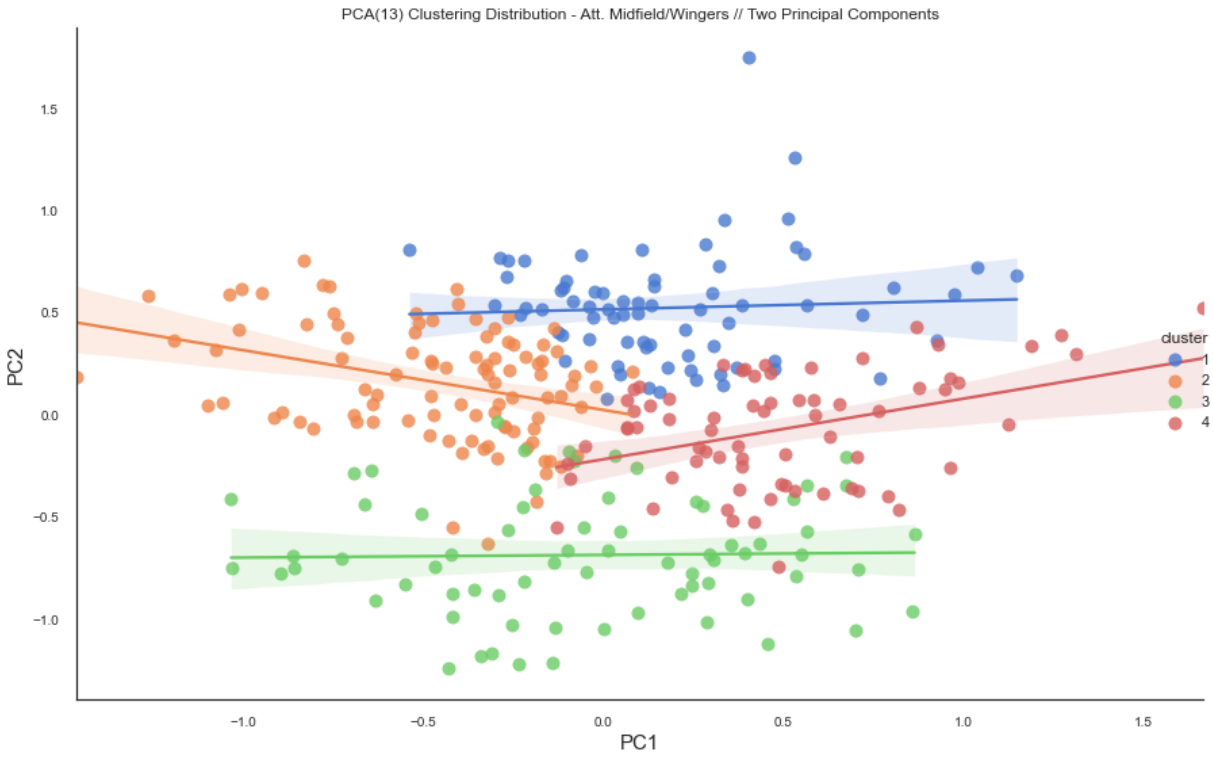



```
-----Starting clustering for Att. Midfield/Wingers-----
Num. of variables that are not constant or cuasi-constant:  39  out of 40
Setting PCA for training
Selected PCA: 13
```

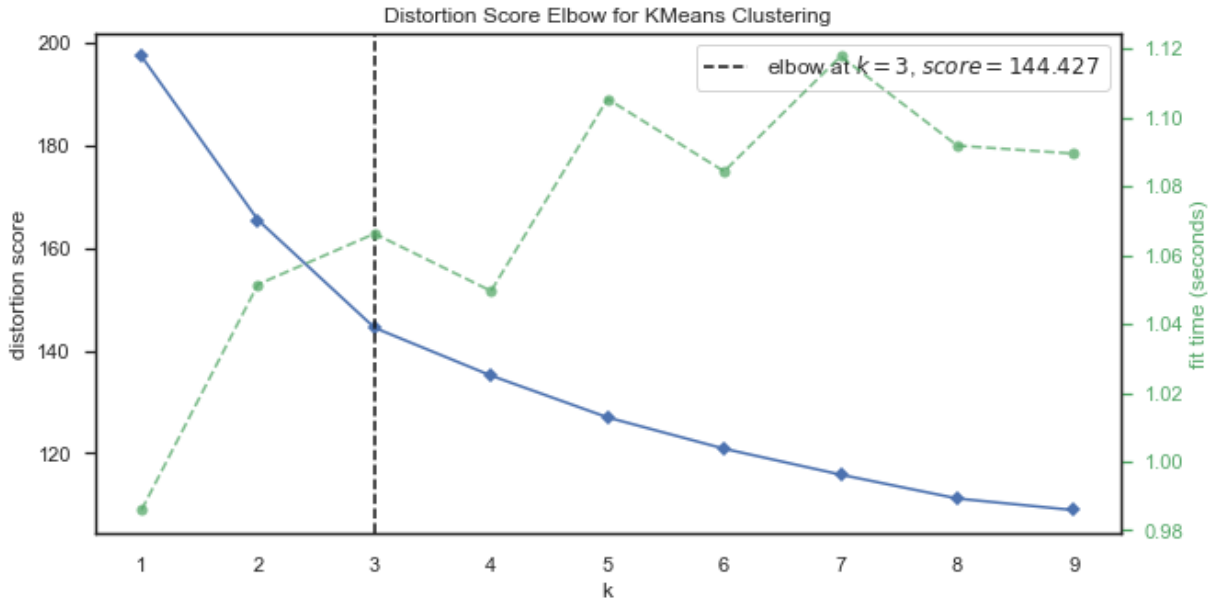


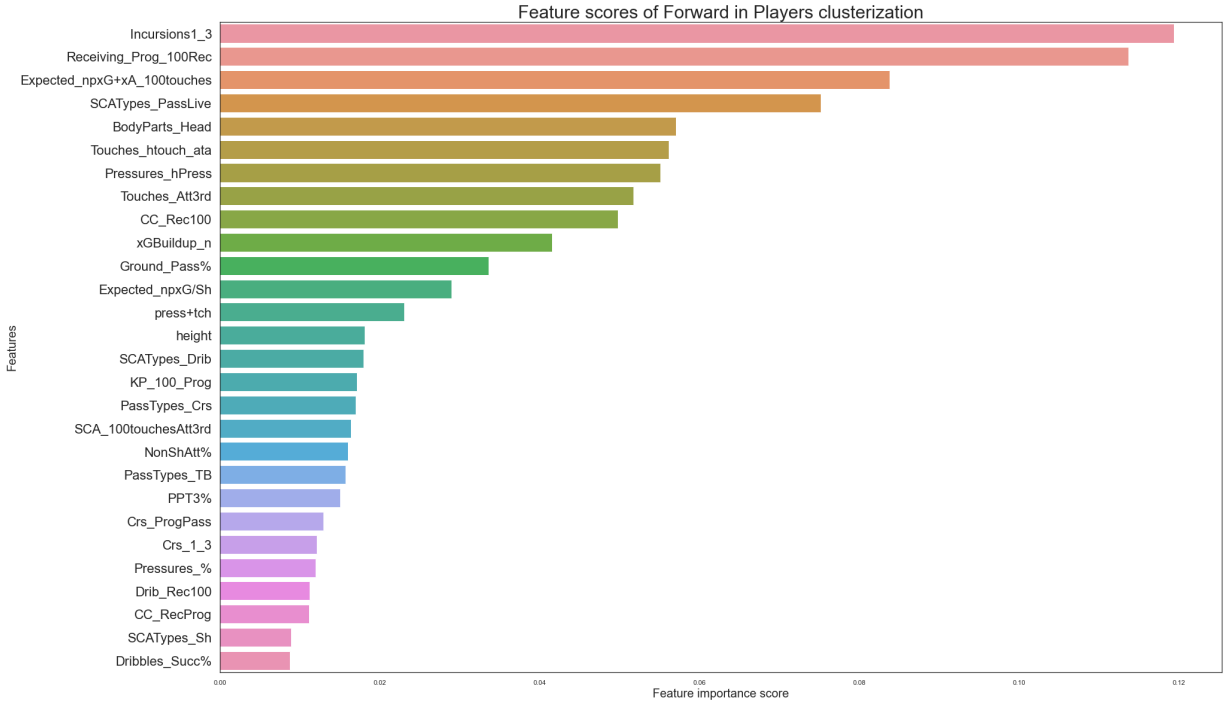

Feature scores of Att. Midfield_Winger in Players clusterization
Wing_Natural
Touches_AttPen%
SCATypes_PassLive
Carries+
Expected_npxG+xA_100touches
Touches_htouch_ata
Incursions1_3
Pressures_hPress
press+tch
Carries_Prog_100_Inc
Carries_100touches
PassTypes_Crs
xGBuildup_n
Drib_Rec100
CC_Rec100
PassTypes_TB
KP_100_Prog
Pressures_%
CC_RecProg
NonShAtt%
Expected_npxG/Sh
SCATypes_Drib
Ground_Pass%
SCA_100touchesAtt3rd
SCATypes_Sh
Dribbles_Succ%
PPT3%
%ag_Press
Features
0.00
0.05
0.10
0.15
0.20
Feature importance score

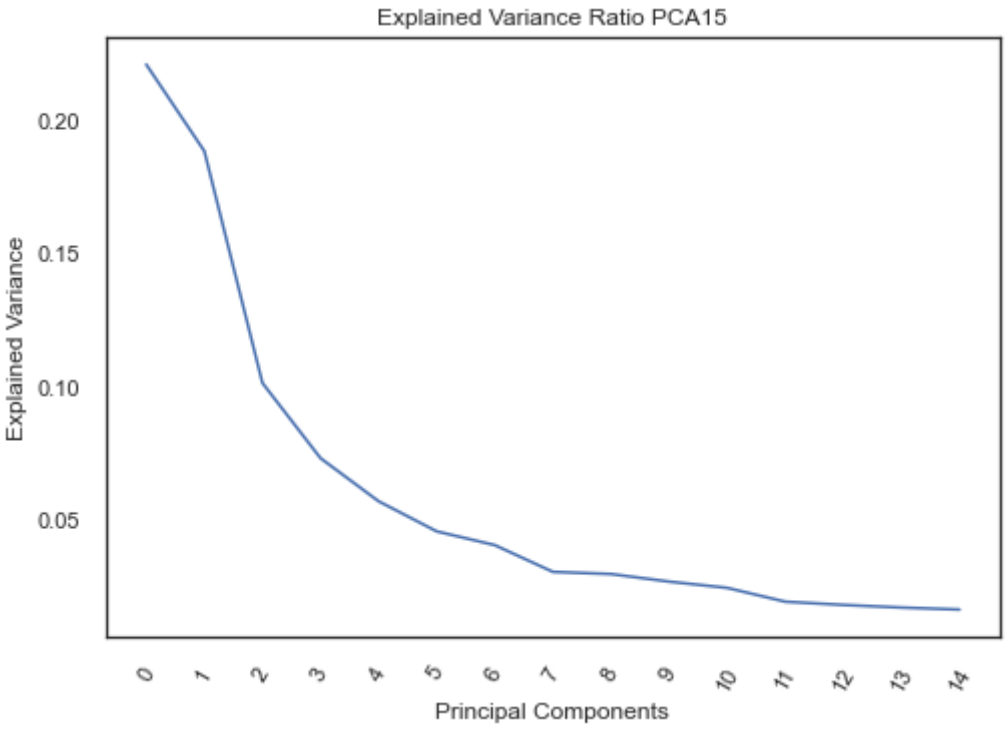

Explained Variance Ratio PCA13
Explained Variance
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
1
2
3
4
5
6
7
8
9
10
11
12
Principal Components

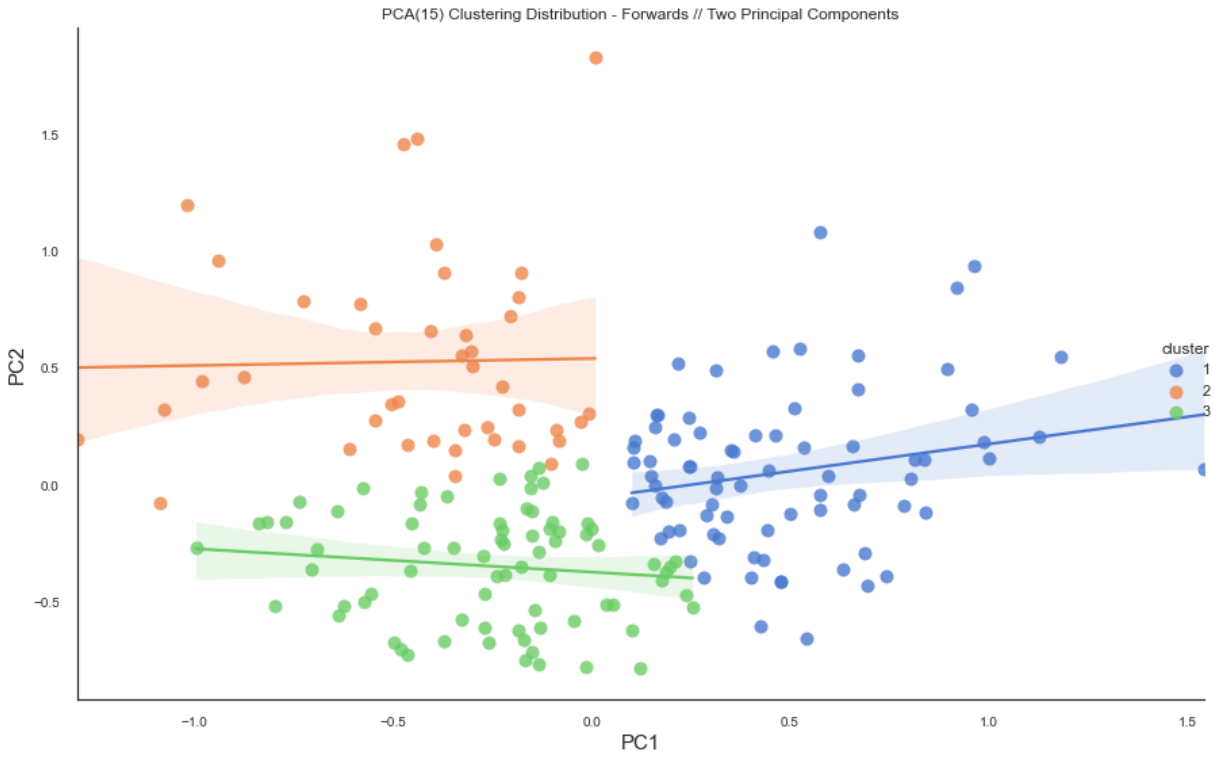



```
-----Starting clustering for Forwards-----
Num. of variables that are not constant or cuasi-constant:  37  out of 37
Setting PCA for training
Selected PCA: 15
```




Feature scores of Forward in Players clusterization
Incursions1_3
Receiving_Prog_100Rec
Expected_npxG+xA_100touches
SCATypes_PassLive
BodyParts_Head
Touches_htouch_ata
Pressures_hPress
Touches_Att3rd
CC_Rec100
xGBuildup_n
Ground_Pass%
Expected_npxG/Sh
press+tch
height
SCATypes_Drib
KP_100_Prog
PassTypes_Crs
SCA_100touchesAtt3rd
NonShAtt%
PassTypes_TB
PPT3%
Crs_ProgPass
Crs_1_3
Pressures_%
Drib_Rec100
CC_RecProg
SCATypes_Sh
Dribbles_Succ%
Features
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Feature importance score


Explained Variance Ratio PCA15
0.20
0.15
0.10
0.05
Explained Variance
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Principal Components



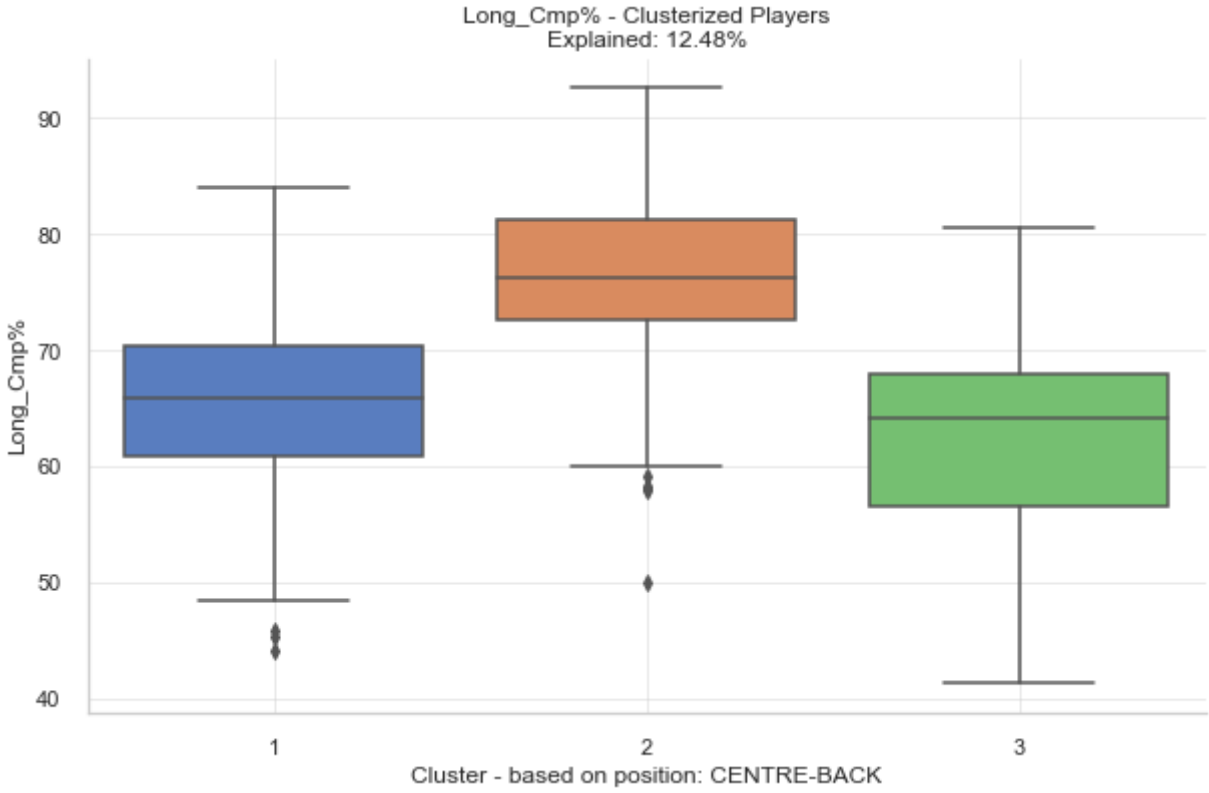
Tal y como sucedió en el modelado de equipos, extraeremos una serie de Boxplots que nos permitirán conocer la composición media y dispersión de las medidas relevantes cada posición. Dichas visualizaciones nos servirán de soporte para perfilar a los jugadores, como veremos abajo.

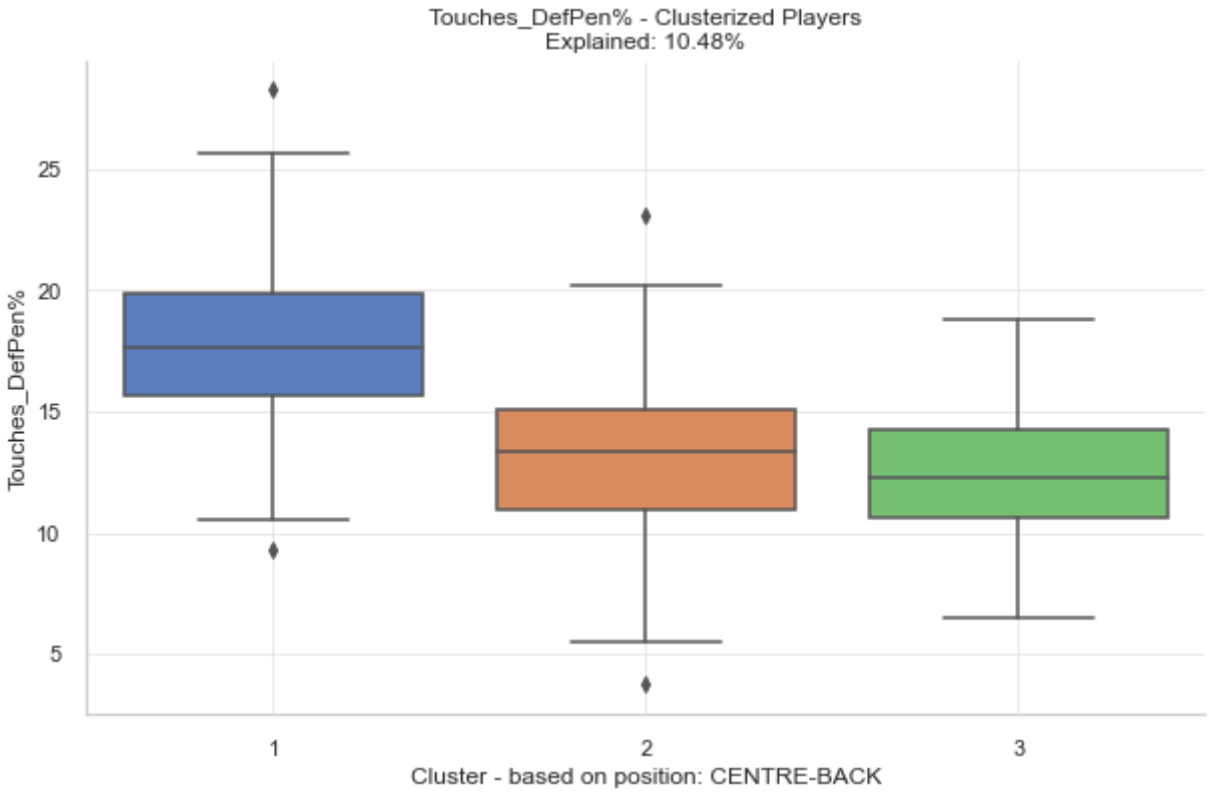
In [29]:

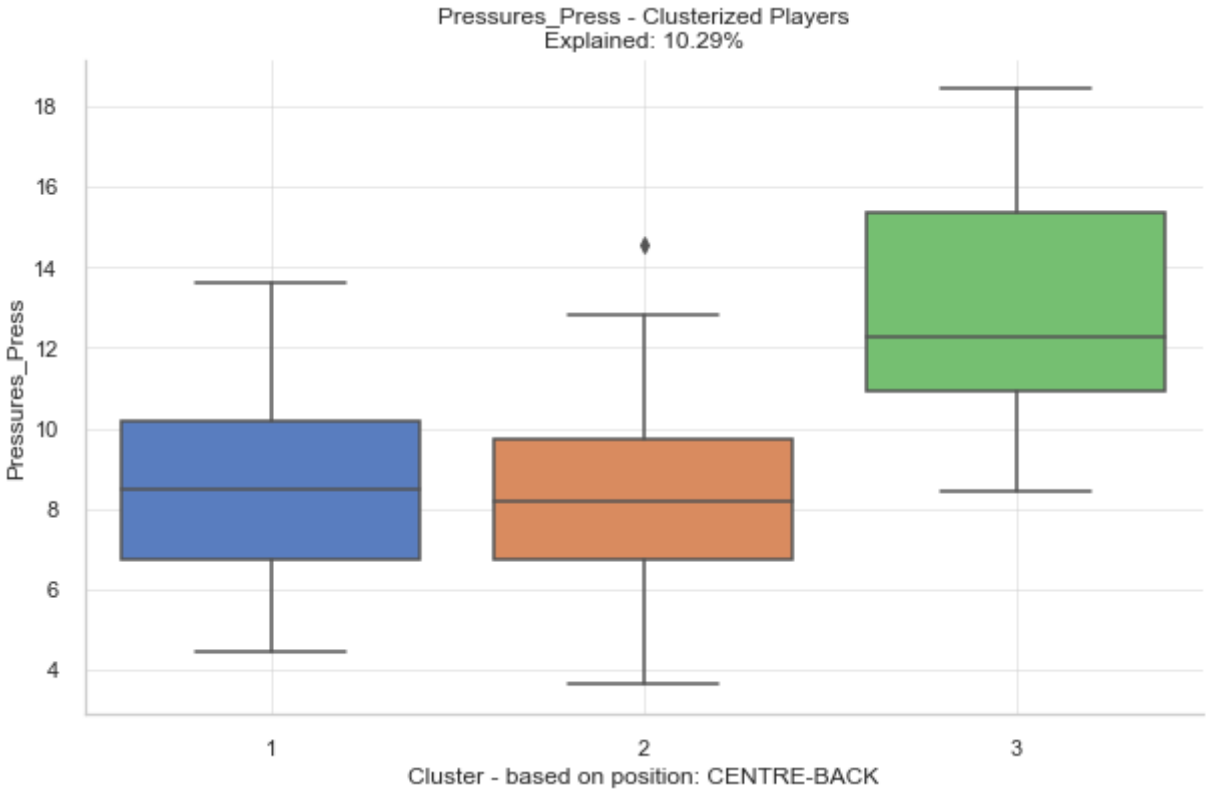
```
for f in features.keys():
    clu = f
    print('-----POSITION: {}-----'.format(clu))

    d = df[clu]
    pl = players.set_index('Player')

    ls=list(features[clu].index)
    ls.append('Player')
    d = pd.merge(d,players[ls],how='left',on='Player')
    f = features[clu][:8]
    n=0
    for k in f.index:
        n+=1
        col=k
        val = round(f[clu][col],5)*100
        fig = plt.figure(figsize = (10, 6))
        ax = fig.add_subplot(111)

        sns.boxplot(x = "cluster",
                    y = col,
                    orient = "v",
                    data = d,
                    ax = ax,
                    palette='muted')

        ax.set_xticklabels(ax.get_xticklabels(), rotation = 0)

        ax.set_title("{} - Clusterized Players\nExplained: {:.2f}%".format(col,val))
        ax.set_xlabel("Cluster - based on position: {}".format(clu.upper()))
        ax.set_ylabel(col)
        sns.despine()
        plt.grid(True, alpha = 0.4)
```

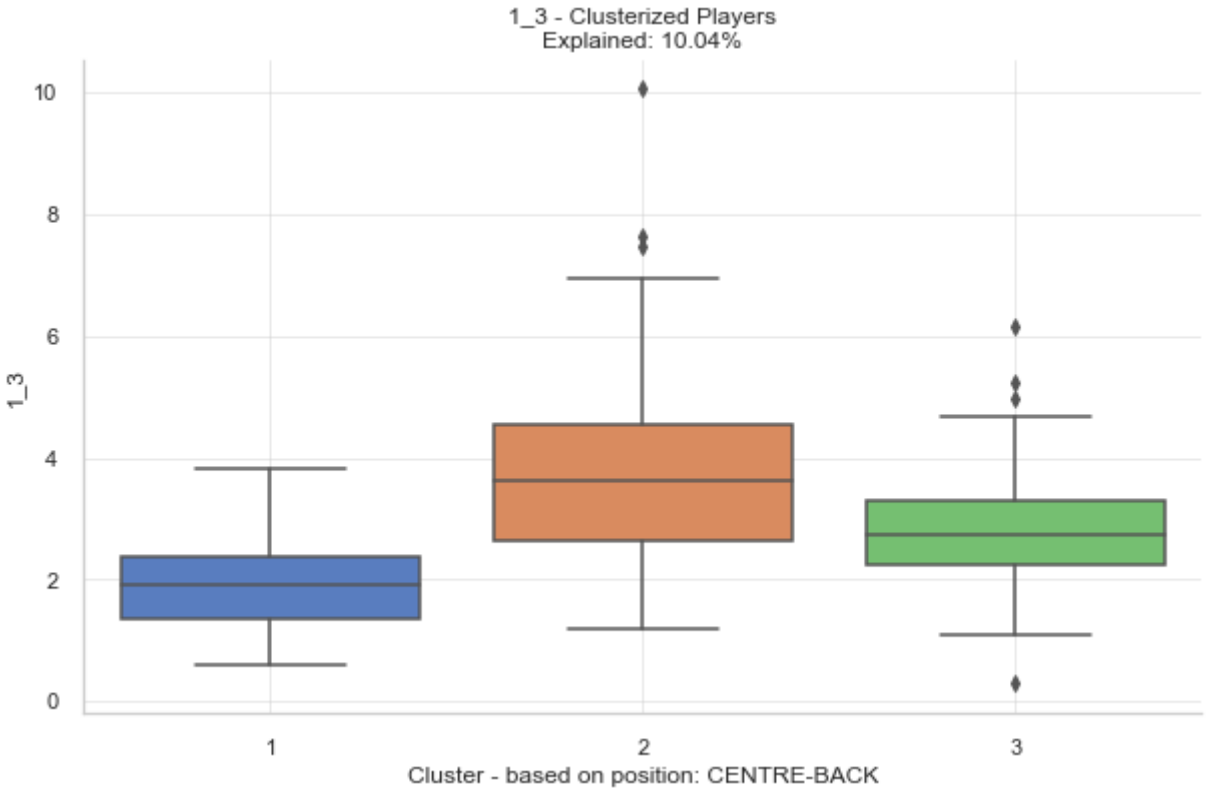
```
        plt.savefig(os.path.join(os.getcwd(),'Model','Players')+"/{}_cluster_metric{
        plt.show();
```

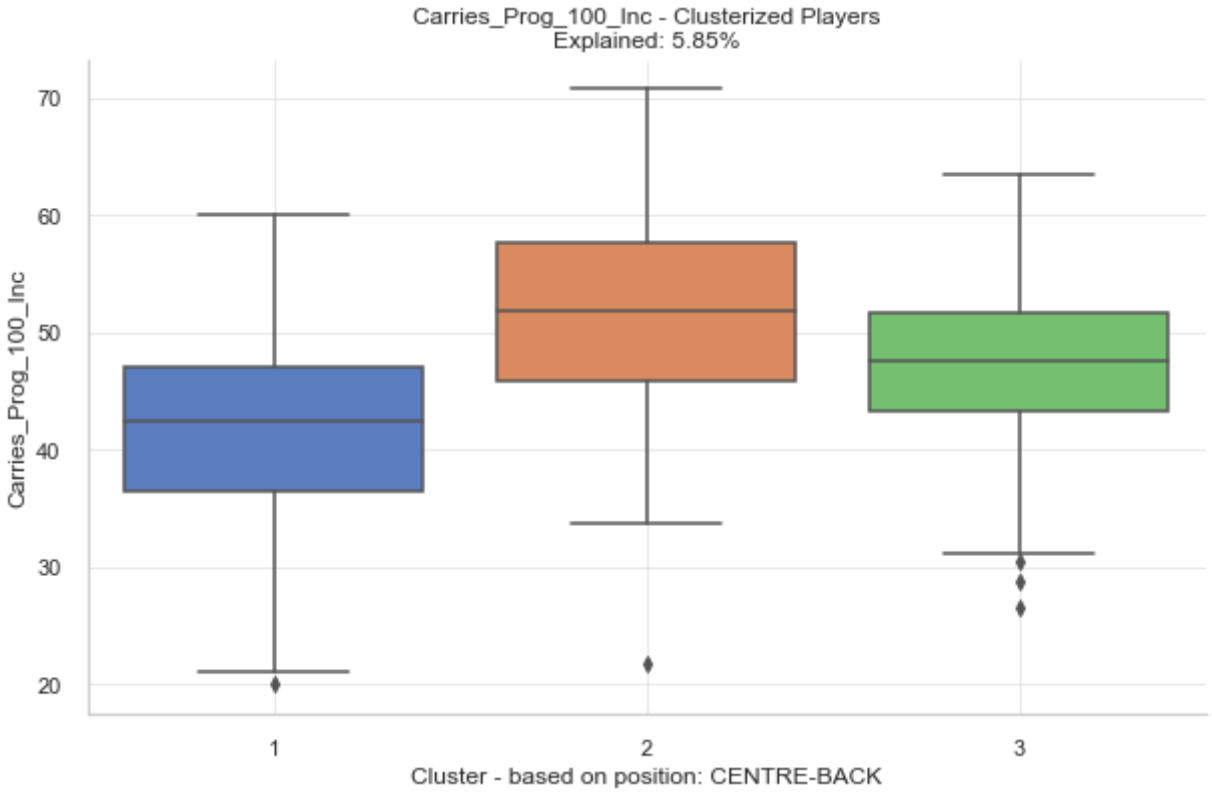
```
-----POSITION: Centre-Back-----
```

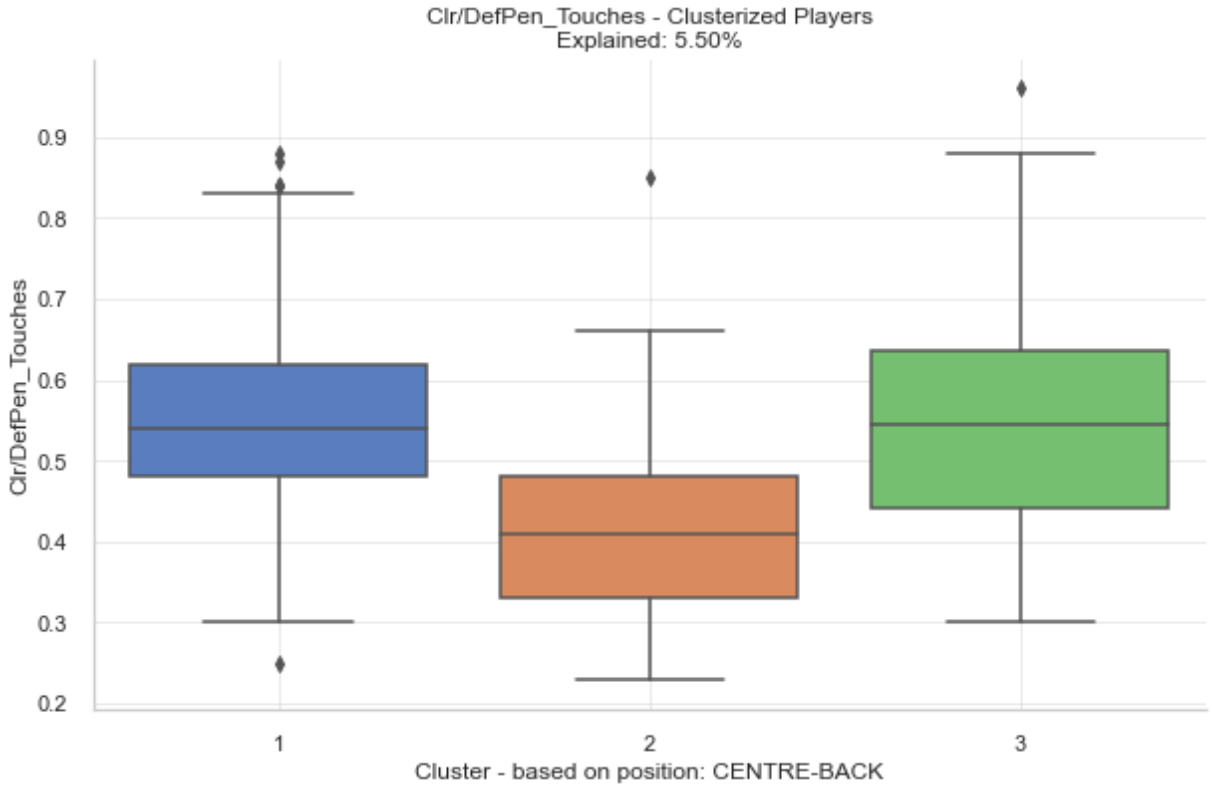


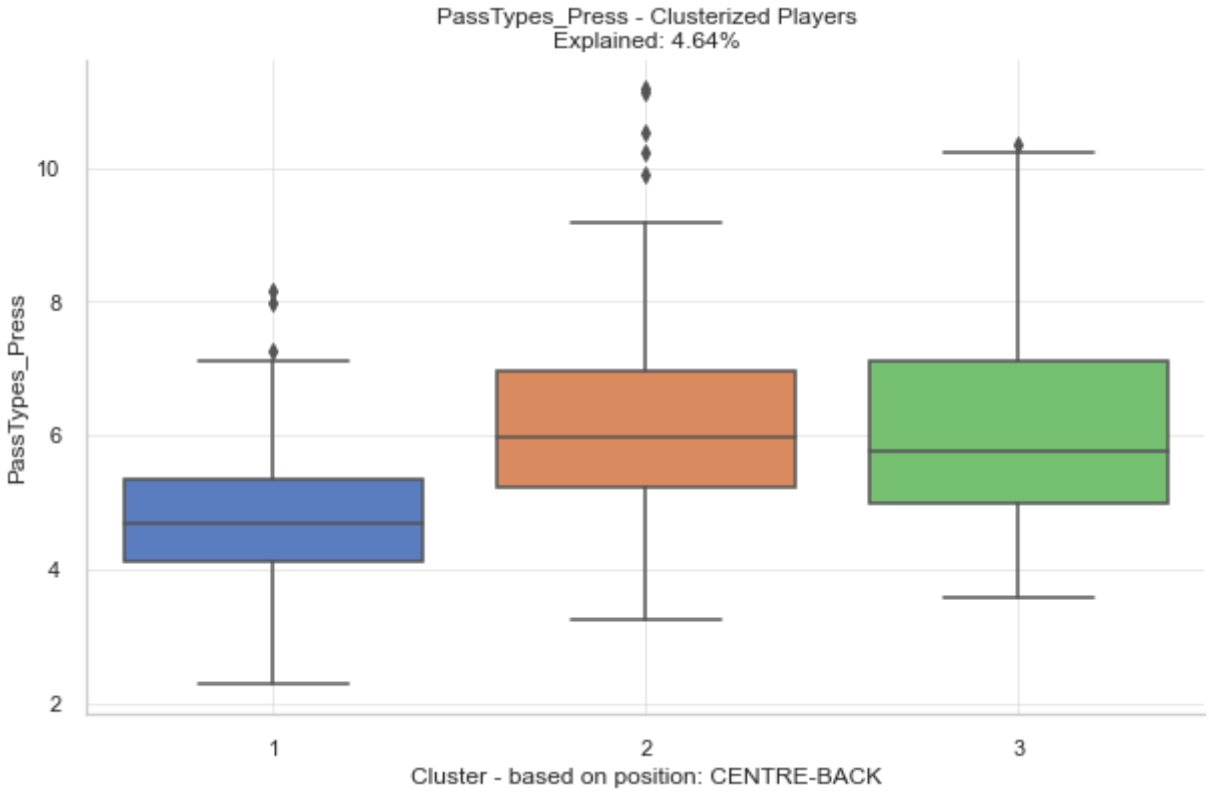


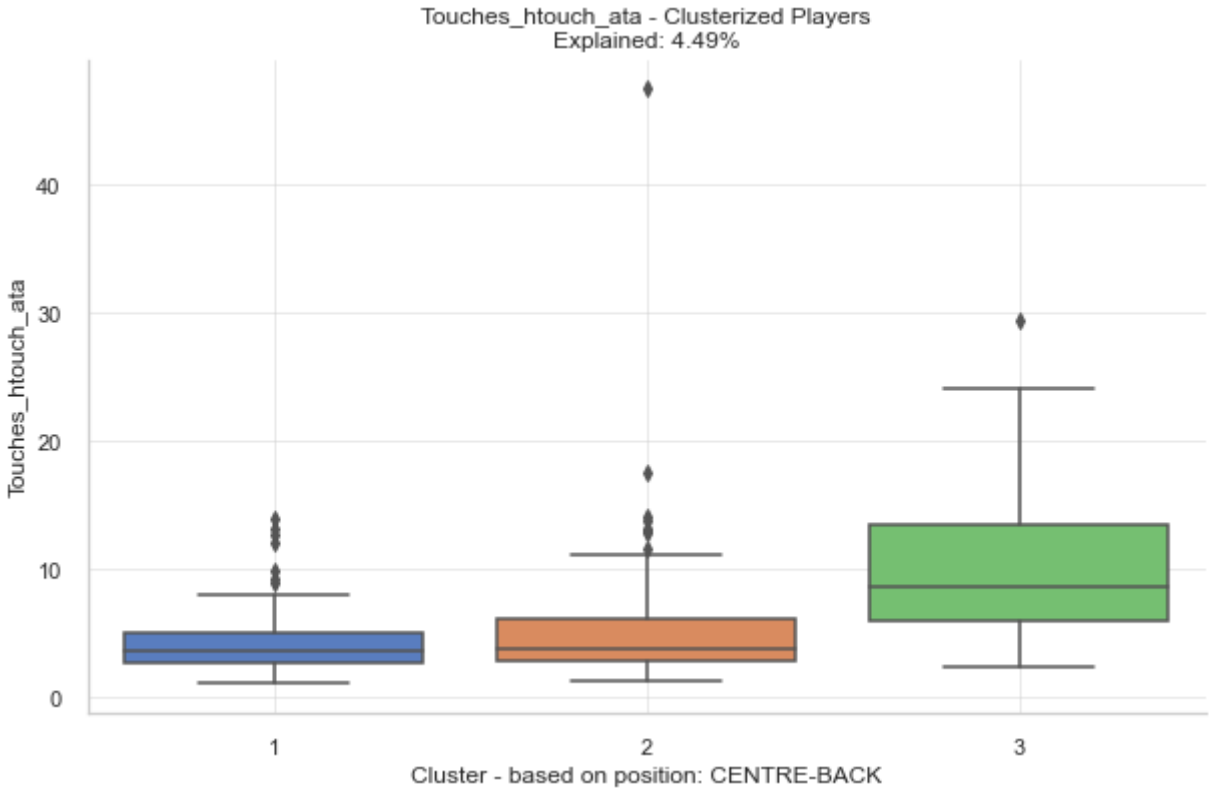

Pressures_Press - Clusterized Players
Explained: 10.29%
Pressures_Press
18
16
14
12
10
8
6
4
1
2
3
Cluster - based on position: CENTRE-BACK


1_3 - Clusterized Players
Explained: 10.04%
1_3
10
8
6
4
2
0
1
2
3
Cluster - based on position: CENTRE-BACK

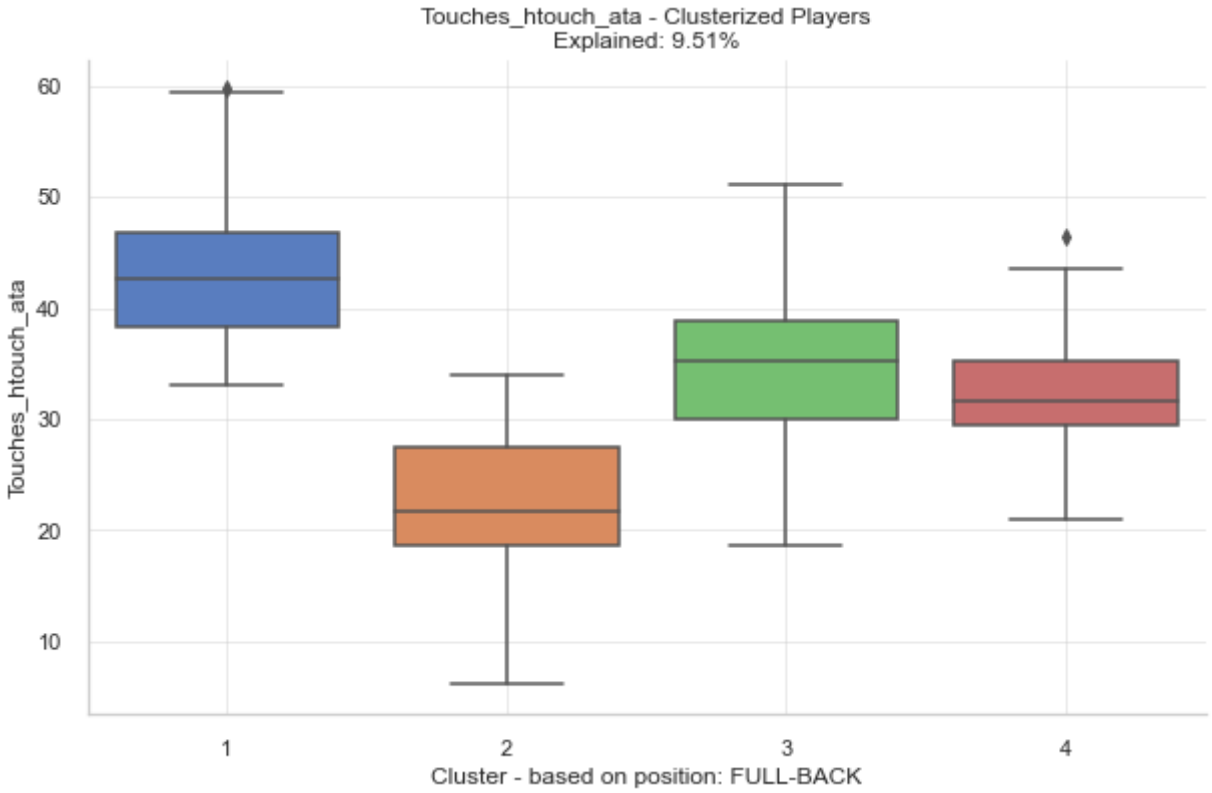

Carries_Prog_100_Inc - Clusterized Players
Explained: 5.85%
Carries_Prog_100_Inc
70
60
50
40
30
20
1
2
3
Cluster - based on position: CENTRE-BACK

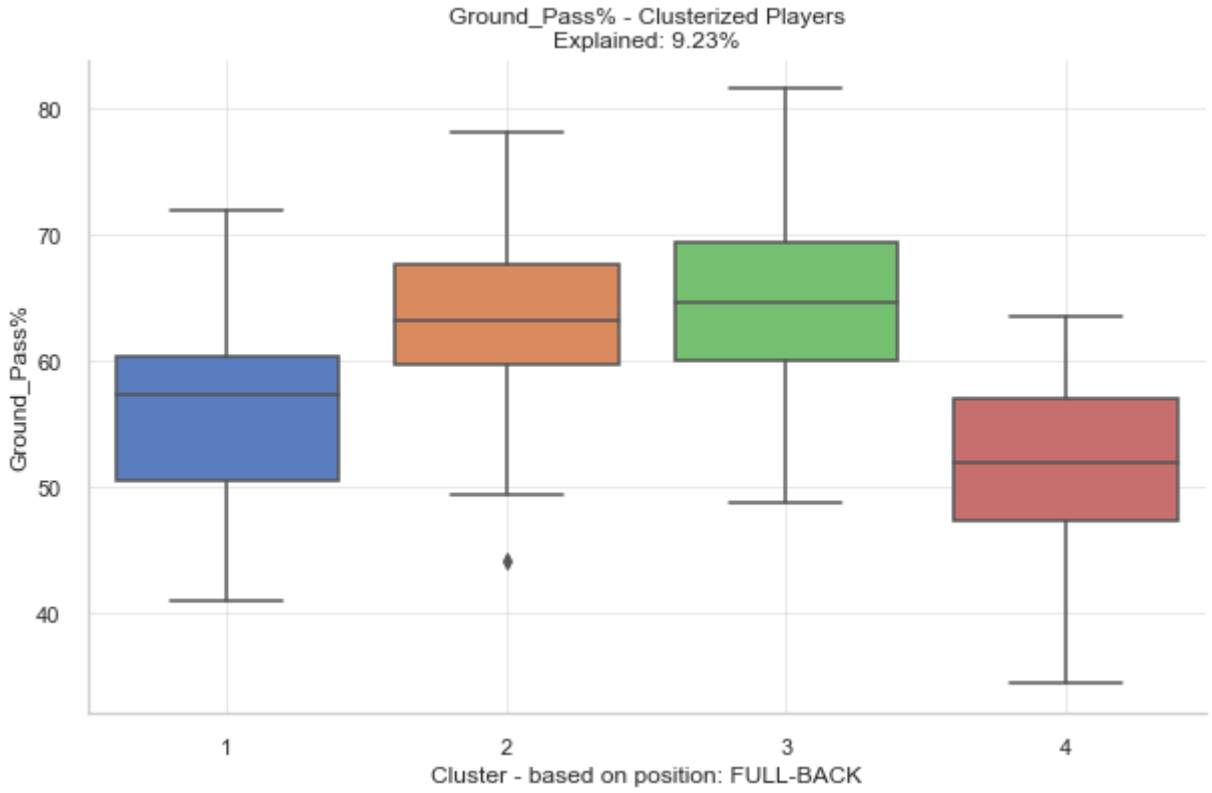

Clr/DefPen_Touches - Clusterized Players
Explained: 5.50%
Clr/DefPen_Touches
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
Cluster - based on position: CENTRE-BACK

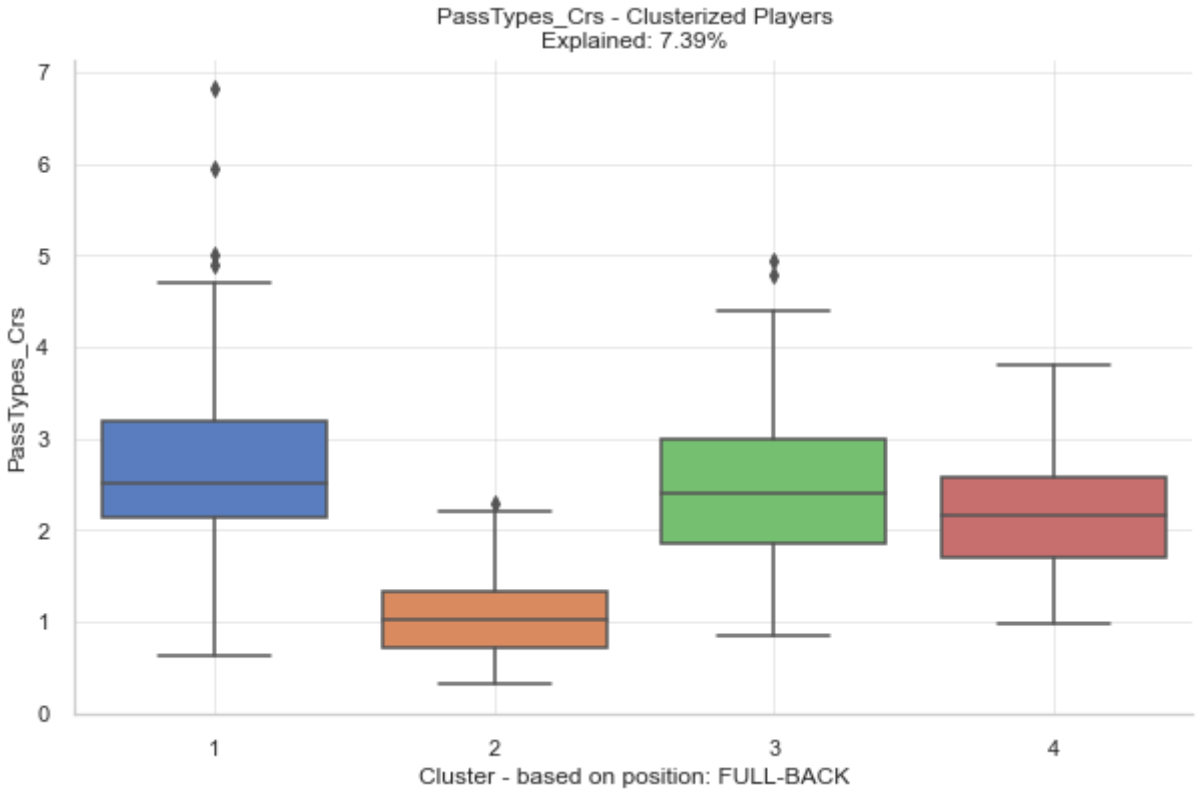


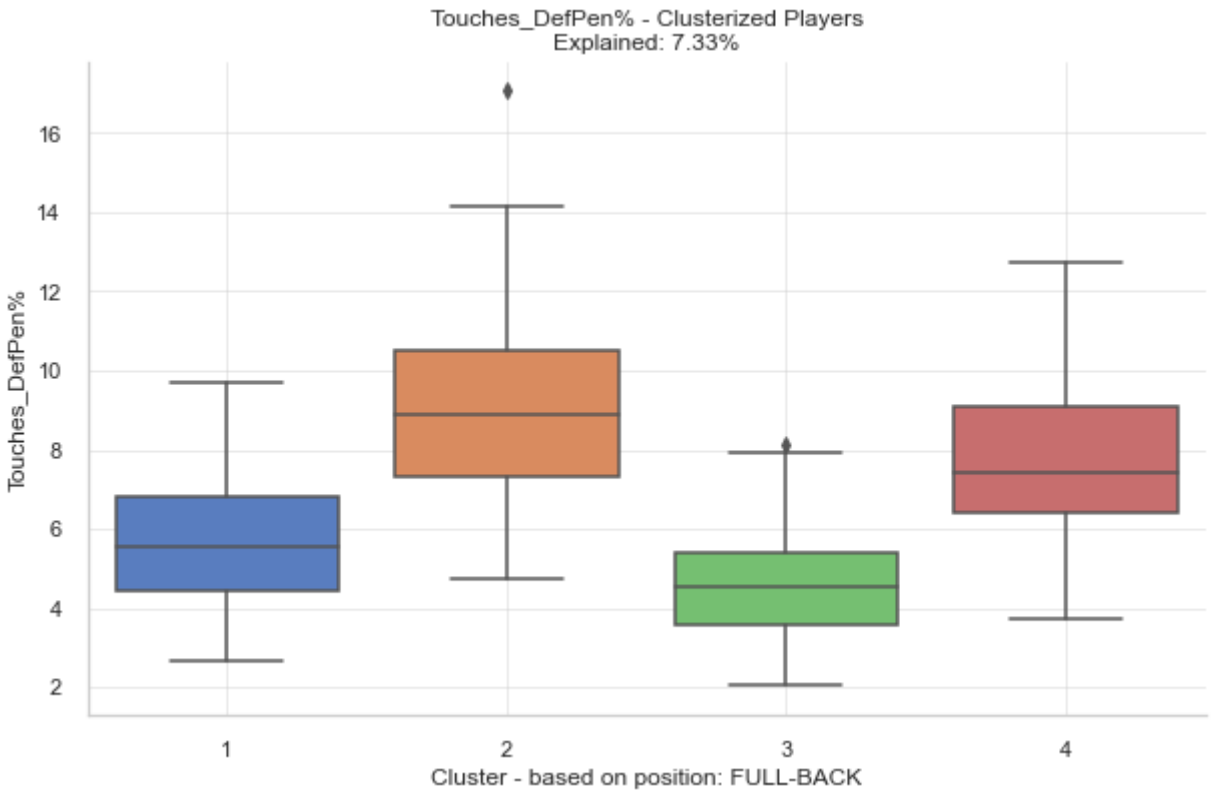


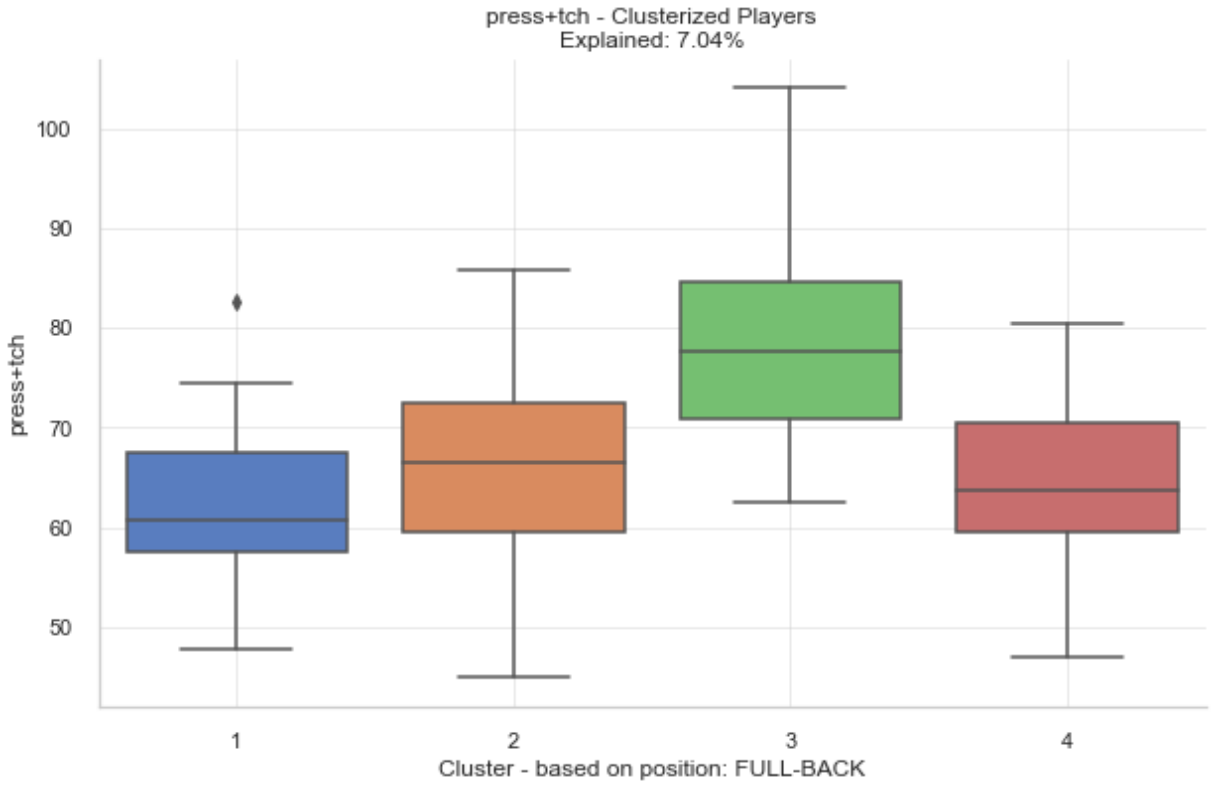
-----POSITION: Full-Back-----

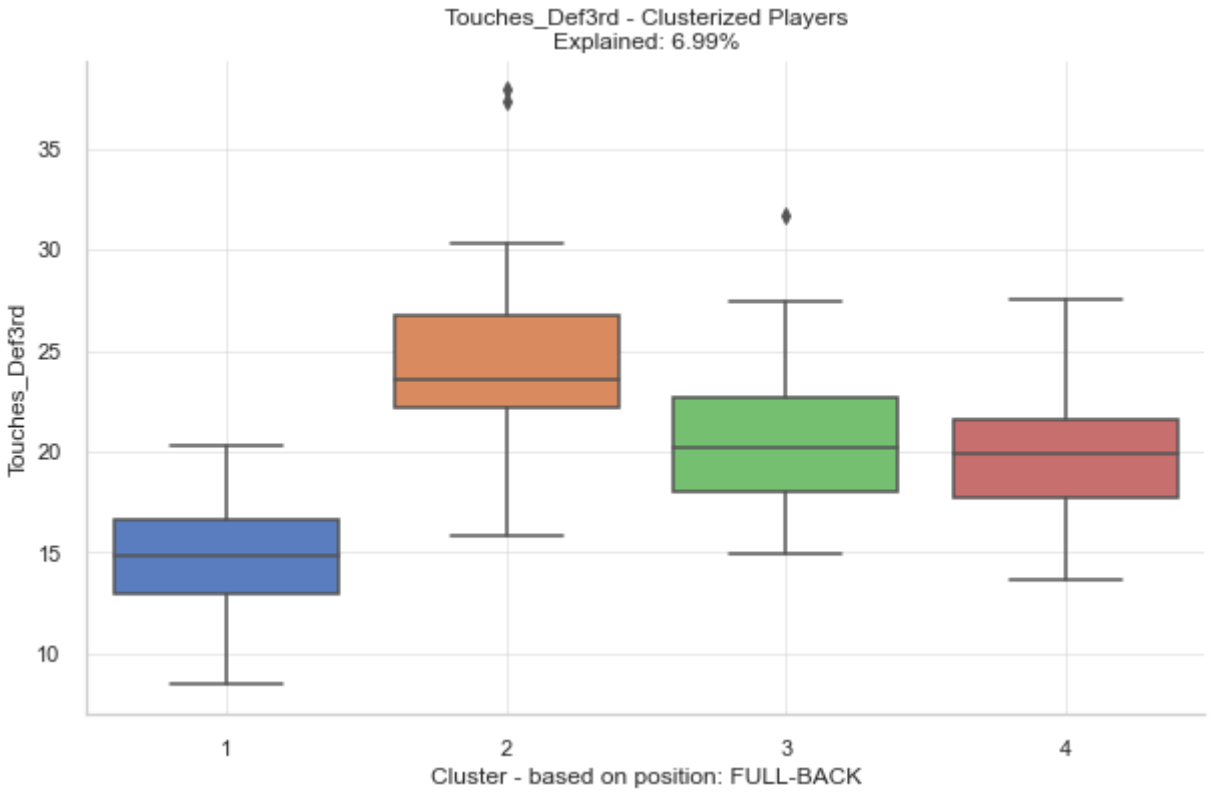

Touches_htouch_ata - Clusterized Players
Explained: 9.51%
Touches_htouch_ata
60
50
40
30
20
10
1
2
3
4
Cluster - based on position: FULL-BACK

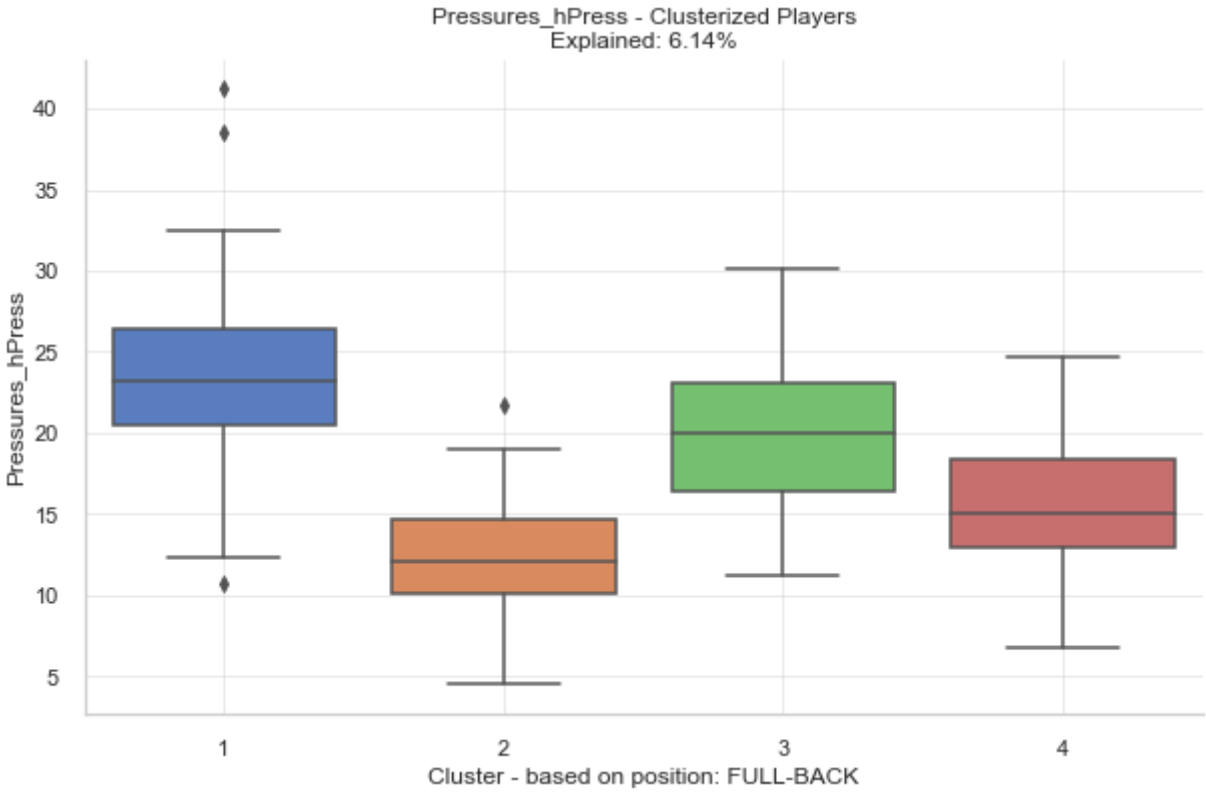

Ground_Pass% - Clusterized Players
Explained: 9.23%
Ground_Pass%
80
70
60
50
40
1
2
3
4
Cluster - based on position: FULL-BACK

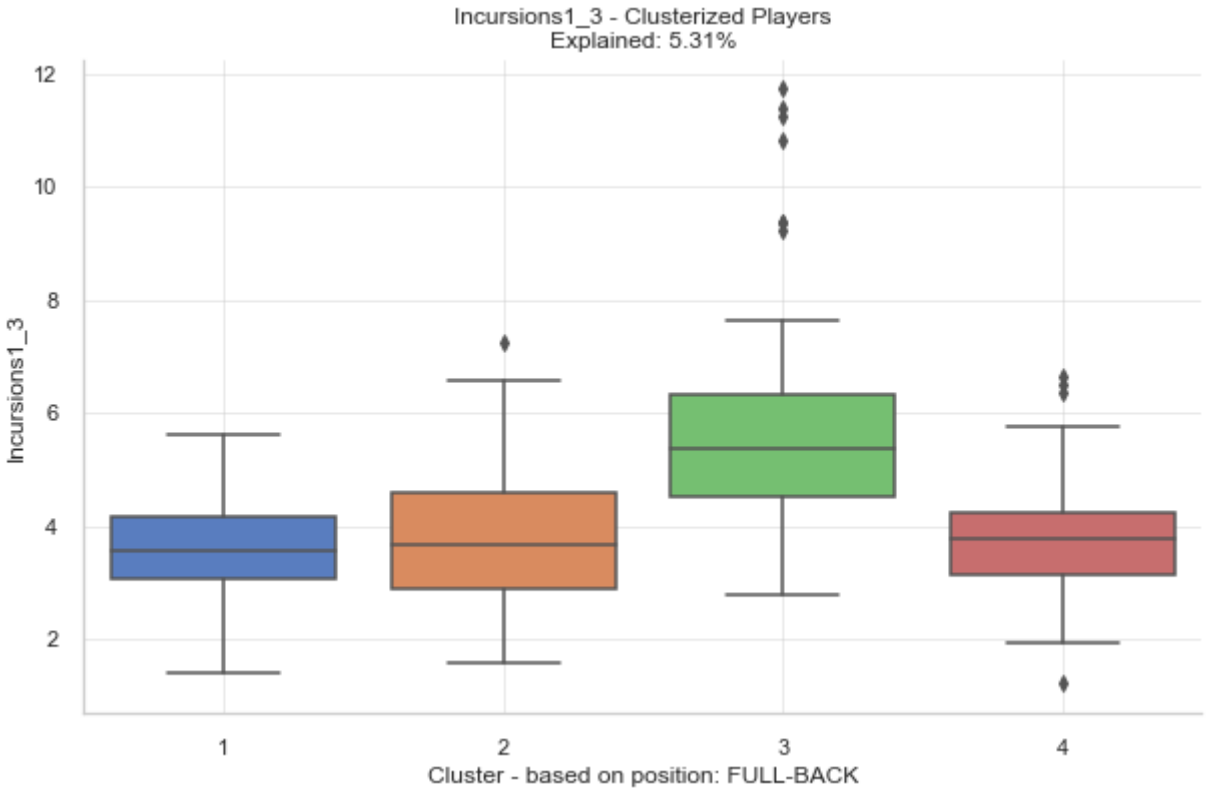

PassTypes_Crs - Clusterized Players
Explained: 7.39%
PassTypes_Crs
0
1
2
3
4
5
6
7
1
2
3
4
Cluster - based on position: FULL-BACK


Touches_DefPen% - Clusterized Players
Explained: 7.33%
Touches_DefPen%
2
4
6
8
10
12
14
16
1
2
3
4
Cluster - based on position: FULL-BACK

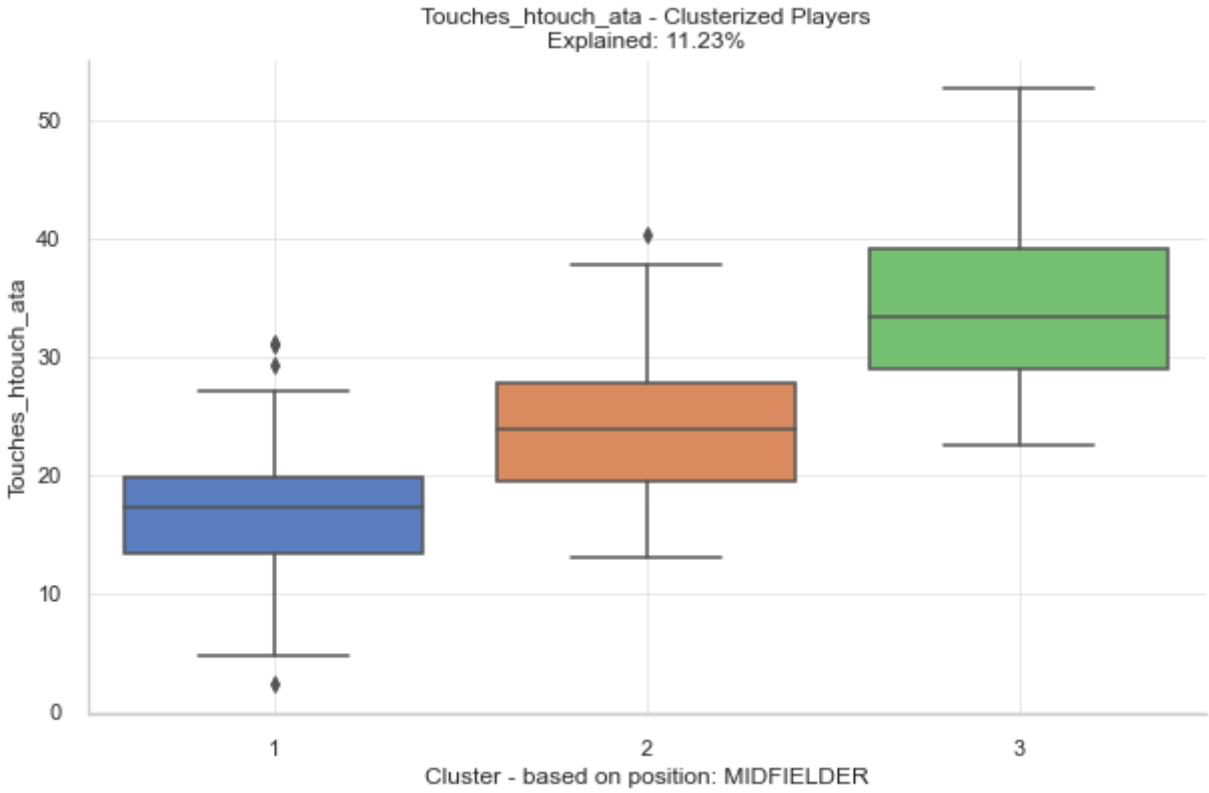

press+tch - Clusterized Players
Explained: 7.04%
press+tch
100
90
80
70
60
50
1
2
3
4
Cluster - based on position: FULL-BACK

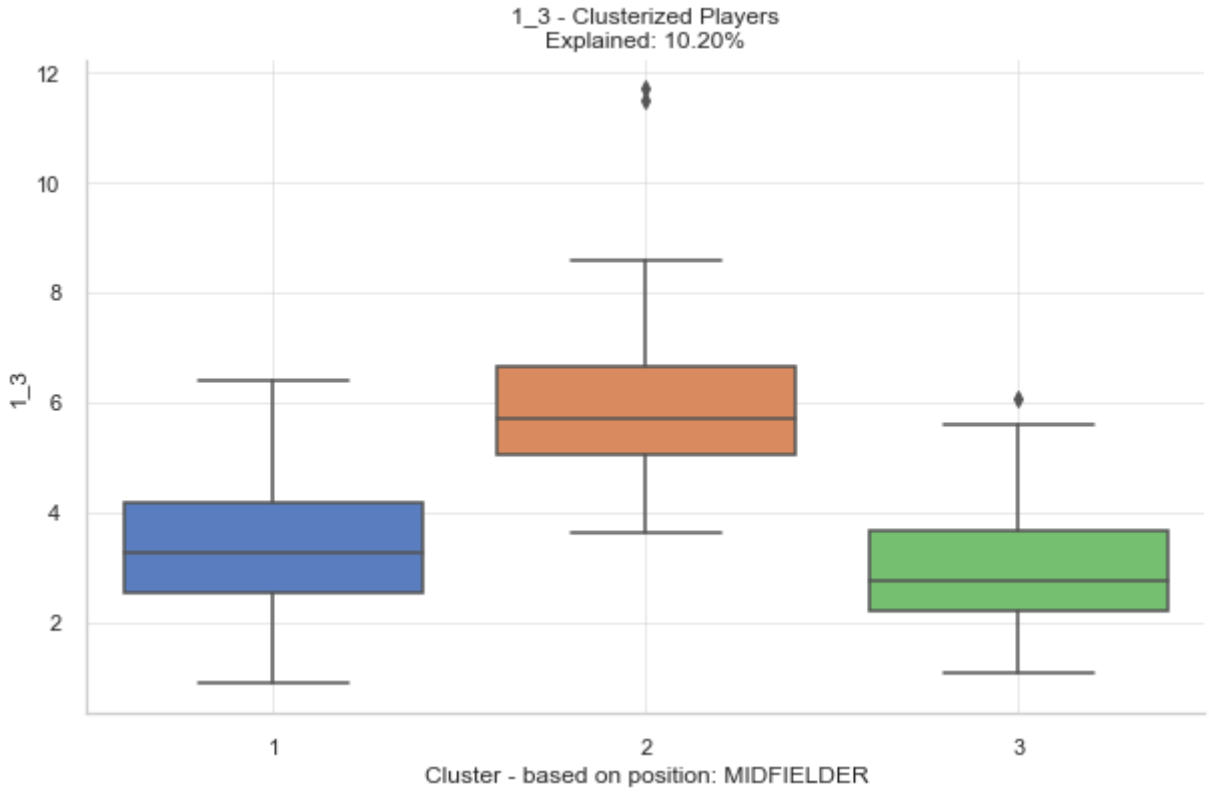

Touches_Def3rd - Clusterized Players
Explained: 6.99%
Touches_Def3rd
35
30
25
20
15
10
1
2
3
4
Cluster - based on position: FULL-BACK

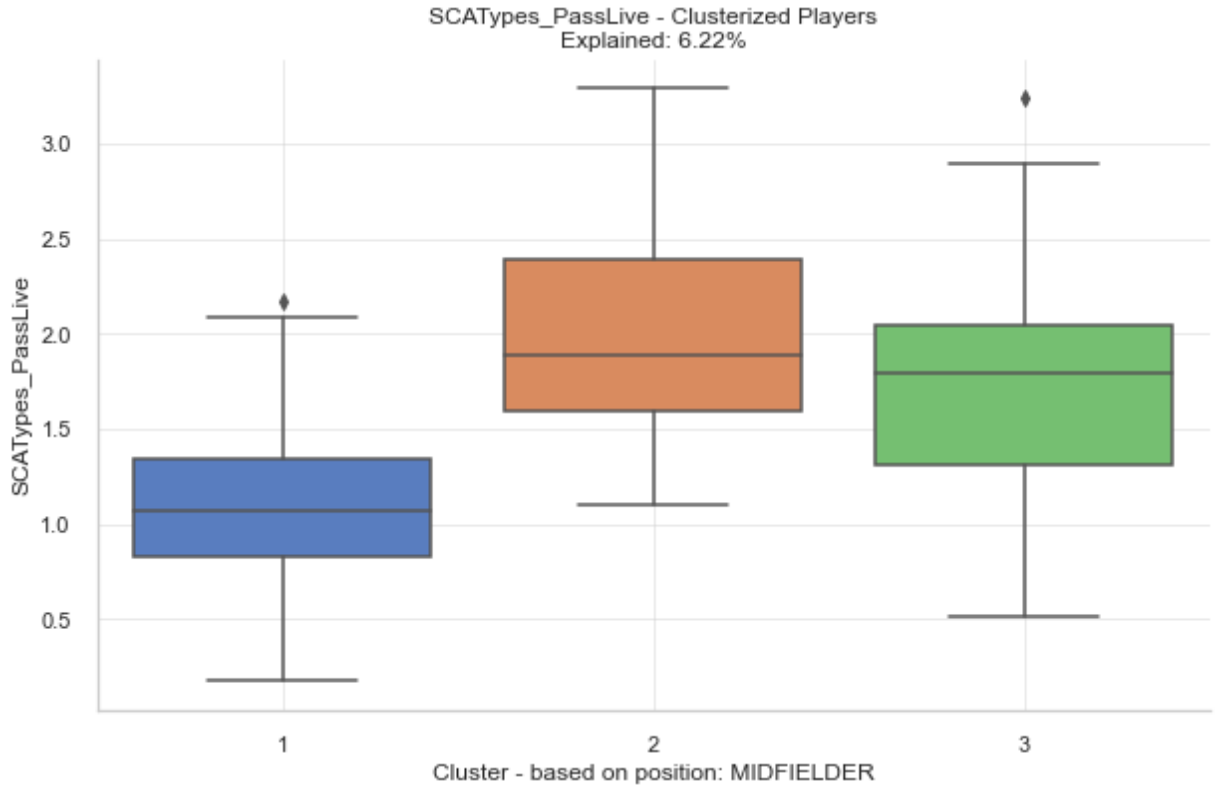


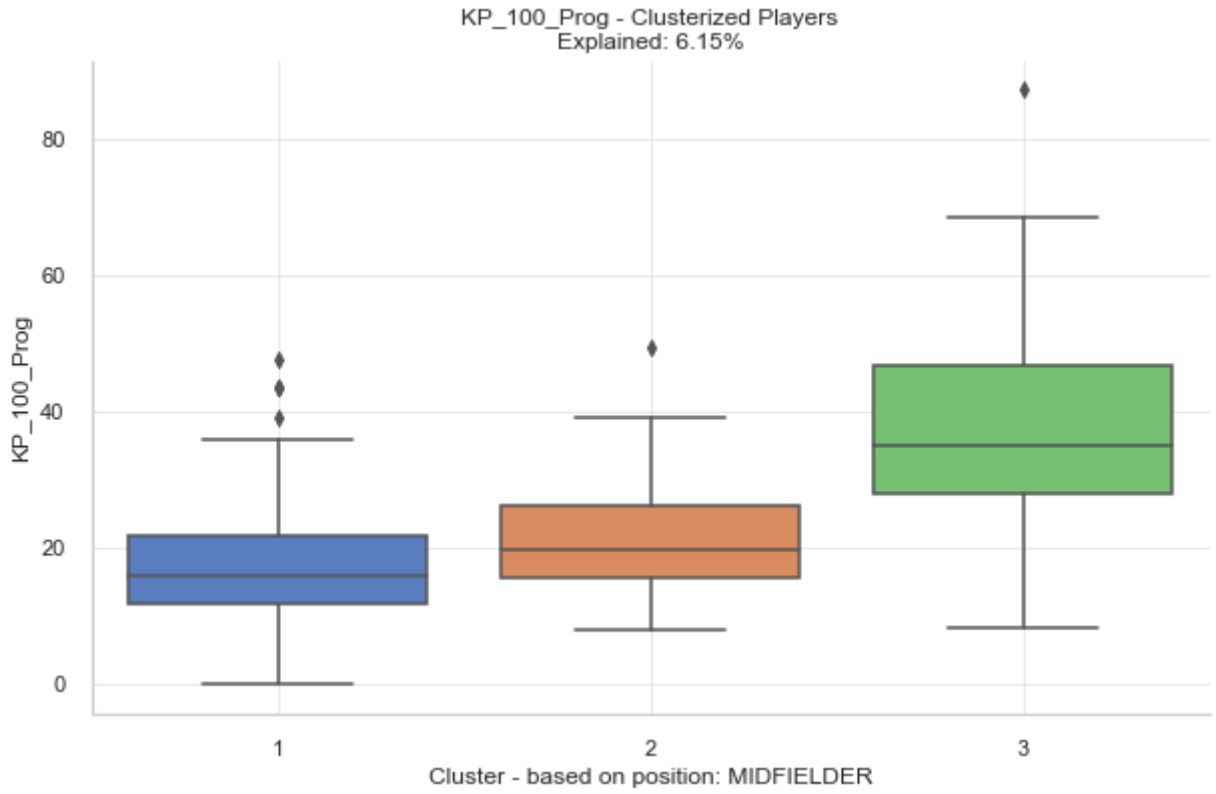


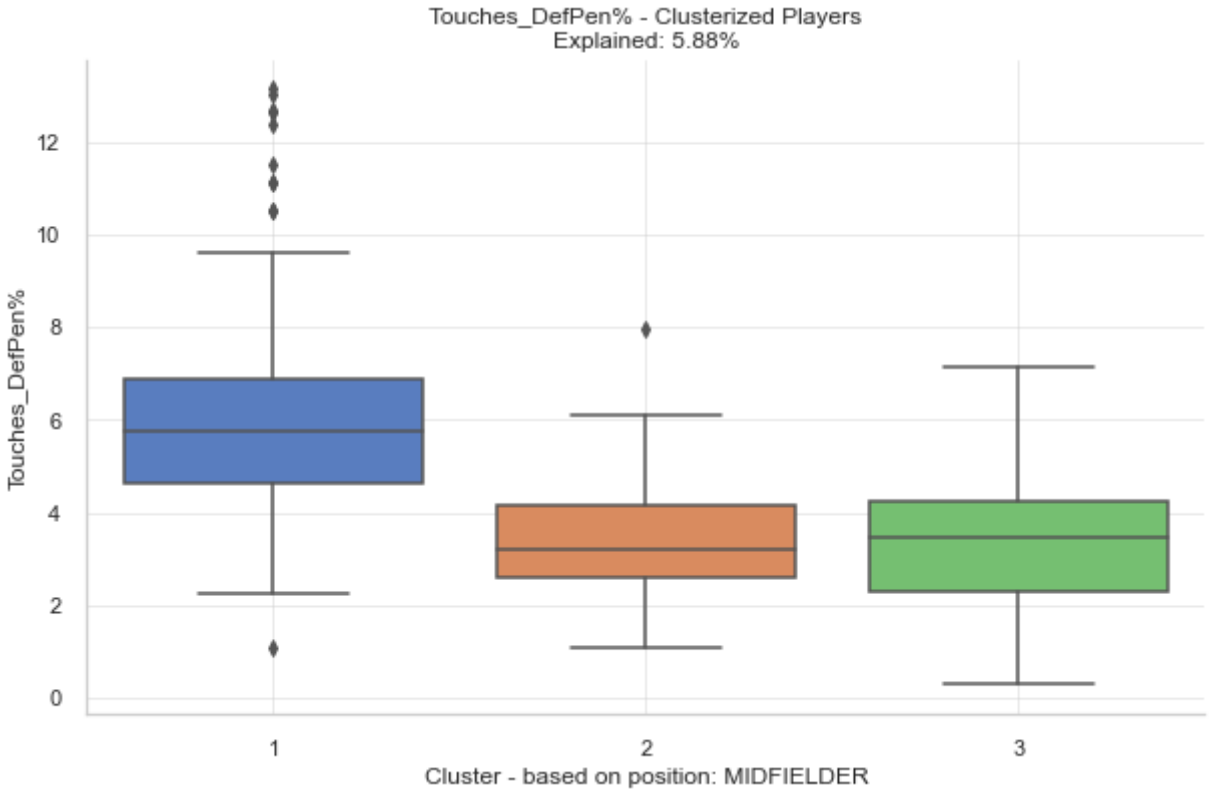
-----POSITION: Midfielder-----

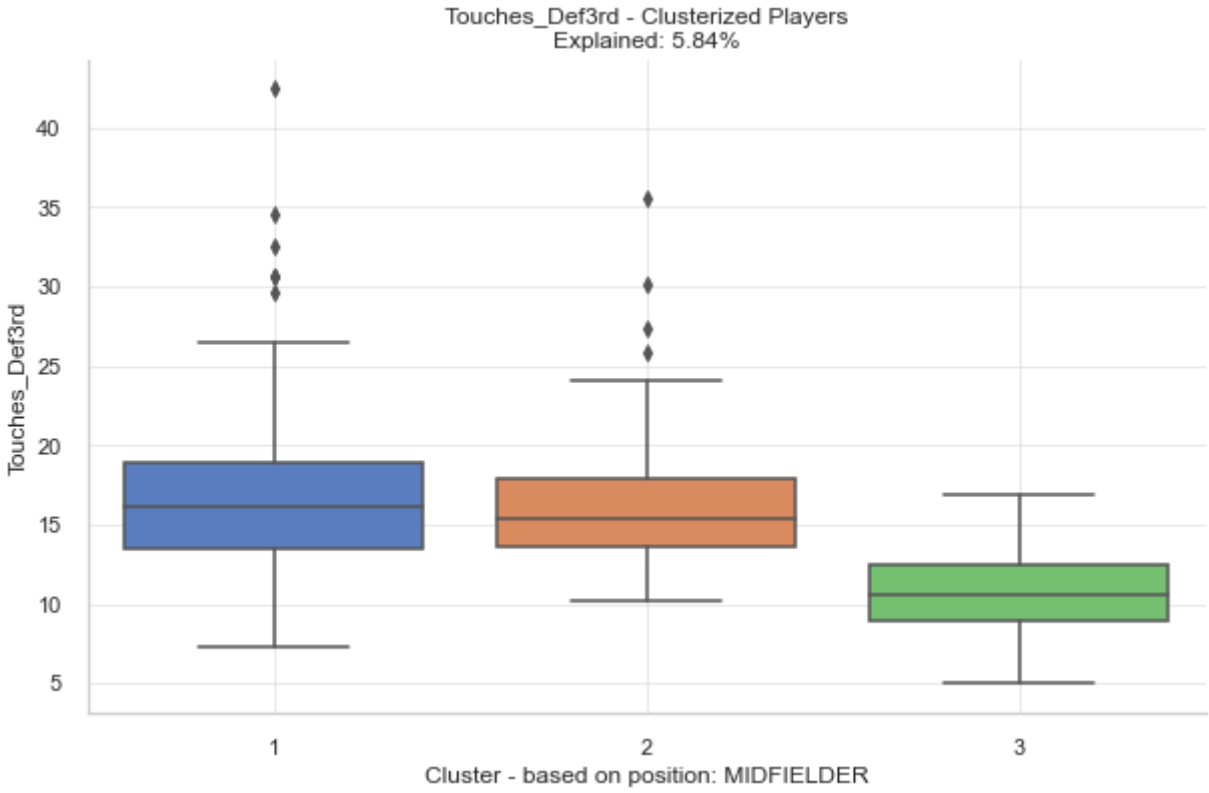

Touches_htouch_ata - Clusterized Players
Explained: 11.23%
Touches_htouch_ata
0
10
20
30
40
50
1
2
3
Cluster - based on position: MIDFIELDER

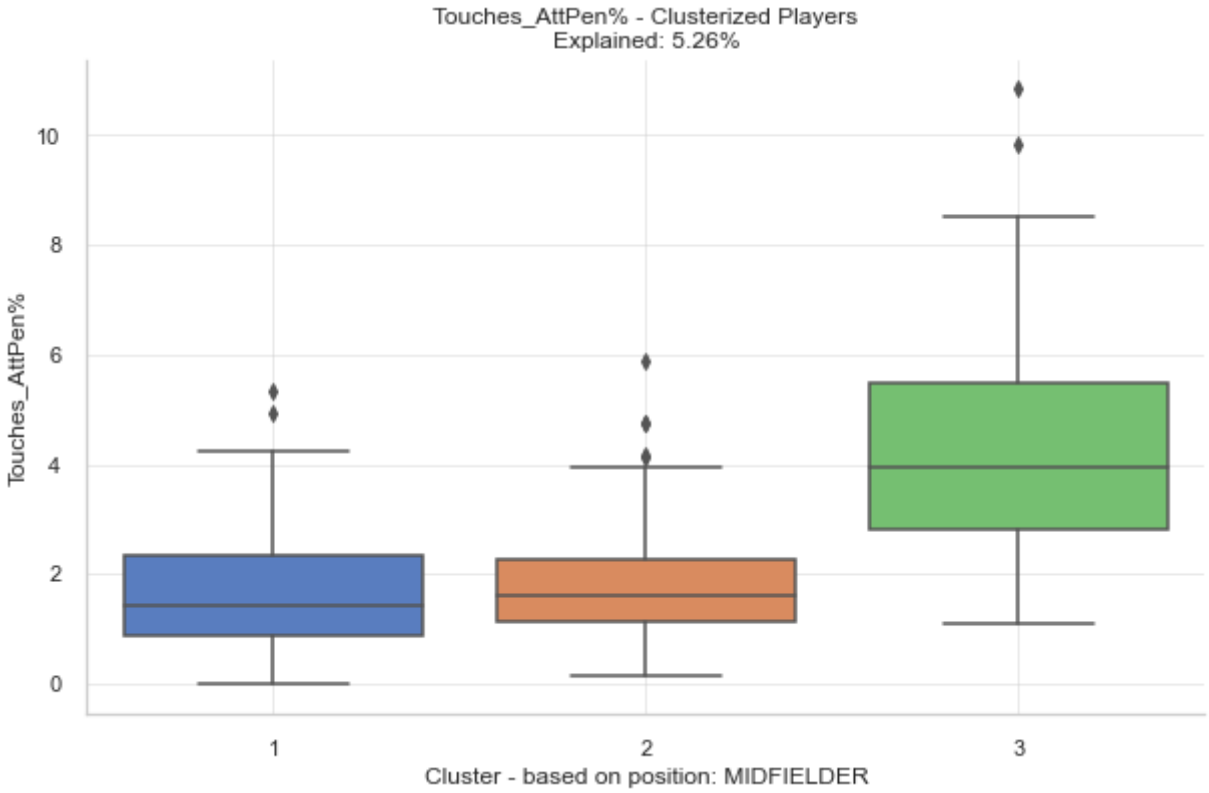

1_3 - Clusterized Players
Explained: 10.20%
1_3
2
4
6
8
10
12
1
2
3
Cluster - based on position: MIDFIELDER

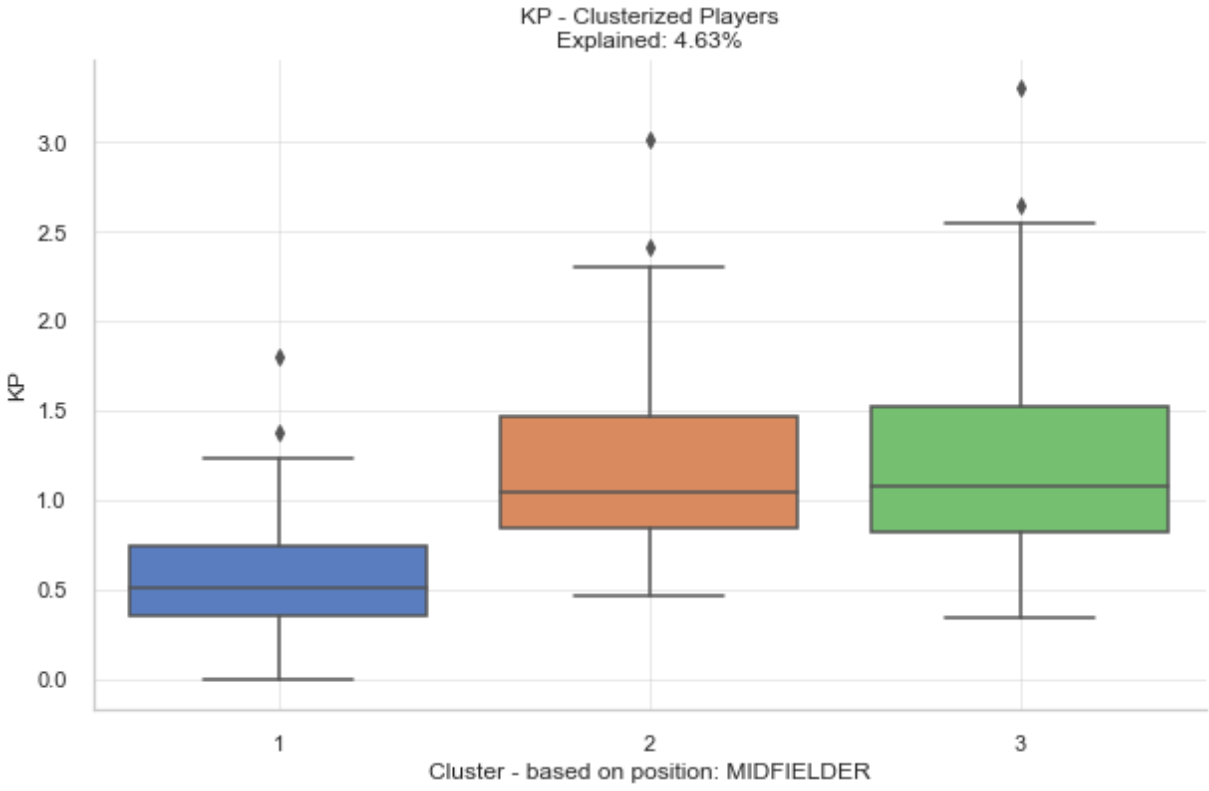

SCATypes_PassLive - Clusterized Players
Explained: 6.22%
SCATypes_PassLive
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
Cluster - based on position: MIDFIELDER


KP_100_Prog - Clusterized Players
Explained: 6.15%
KP_100_Prog
80
60
40
20
0
1
2
3
Cluster - based on position: MIDFIELDER

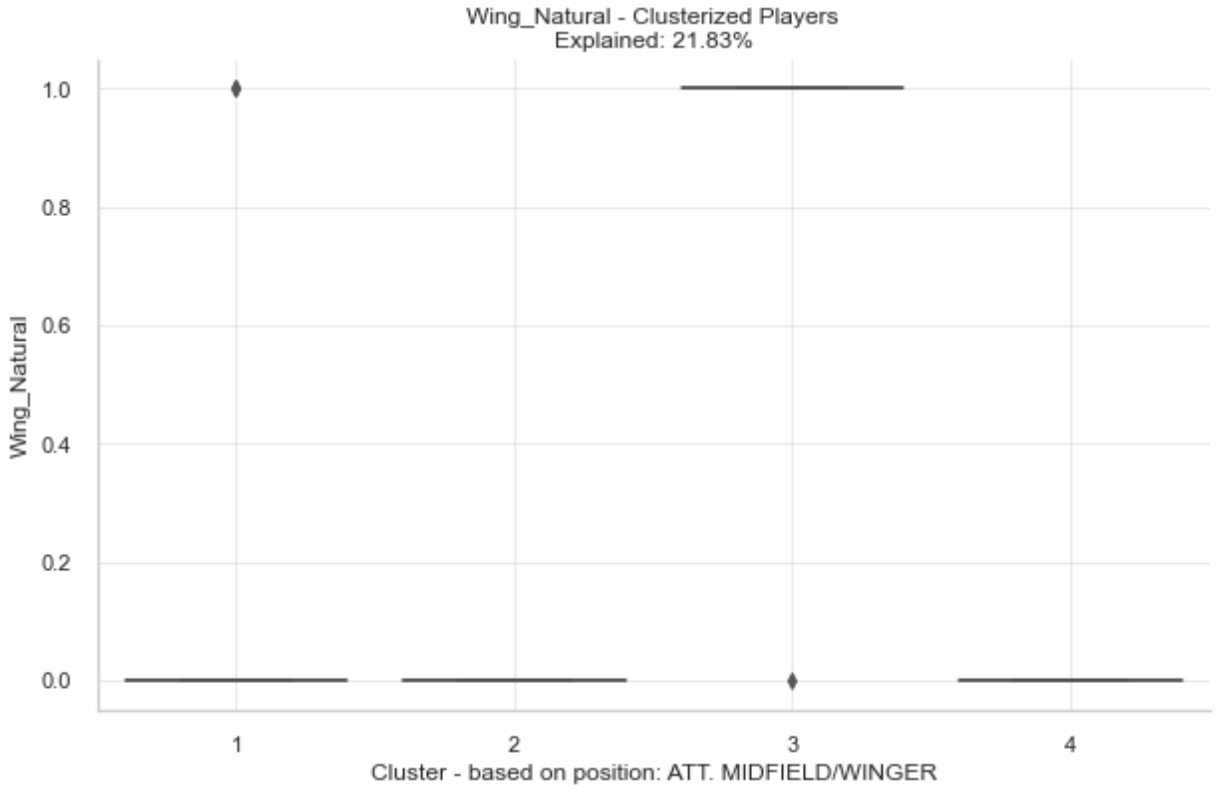

Touches_DefPen% - Clusterized Players
Explained: 5.88%
Touches_DefPen%
0
2
4
6
8
10
12
1
2
3
Cluster - based on position: MIDFIELDER

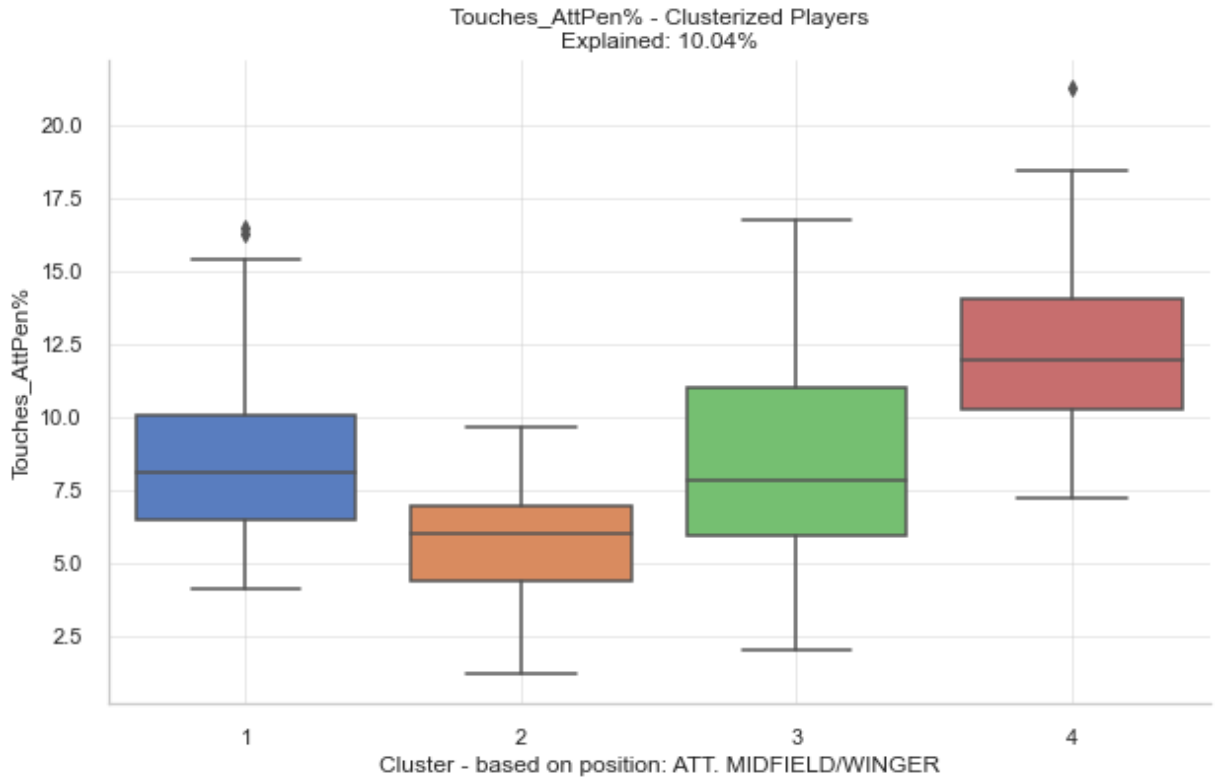

Touches_Def3rd - Clusterized Players
Explained: 5.84%
Touches_Def3rd
5
10
15
20
25
30
35
40
1
2
3
Cluster - based on position: MIDFIELDER

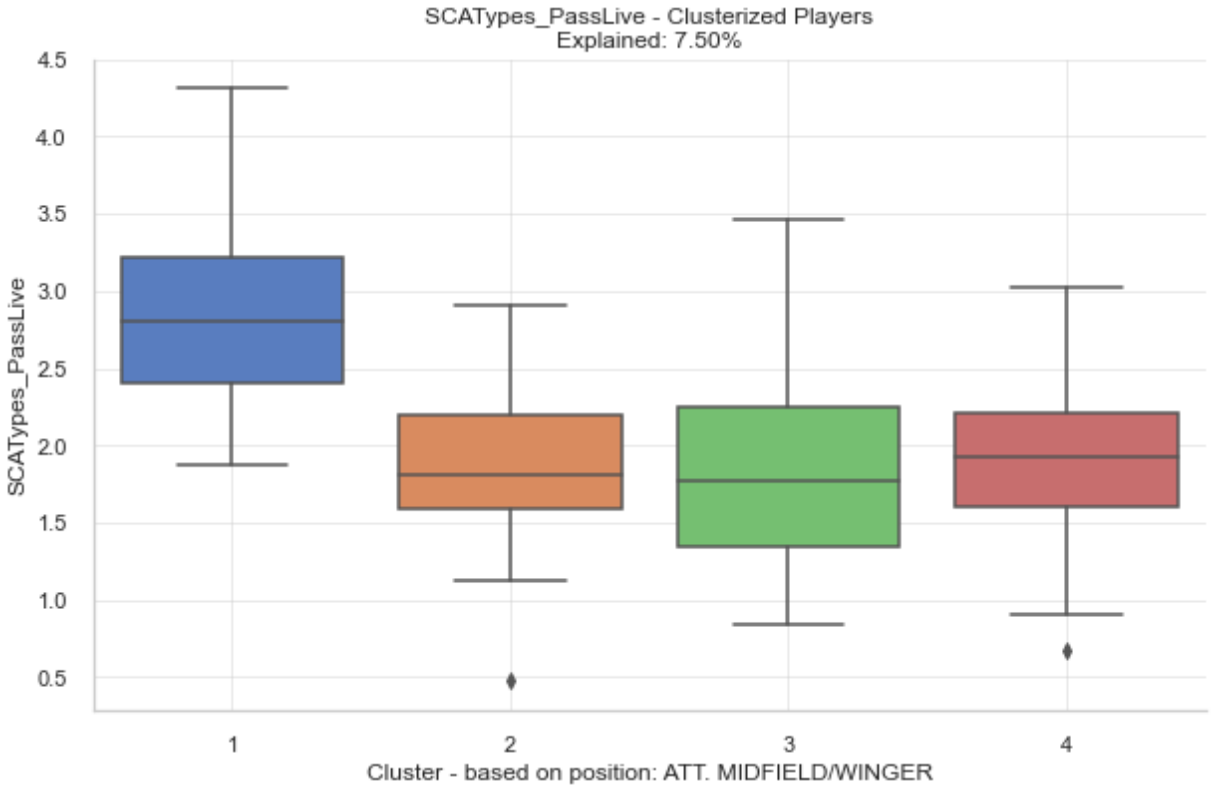


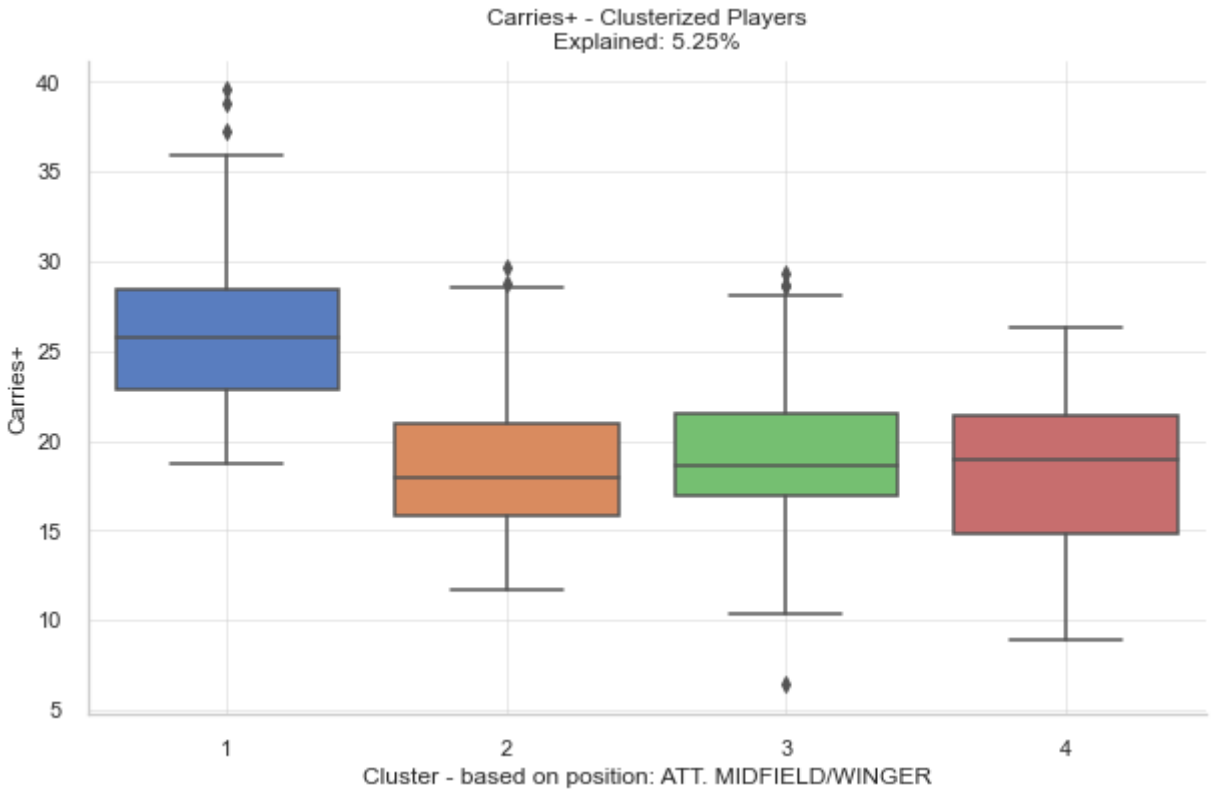


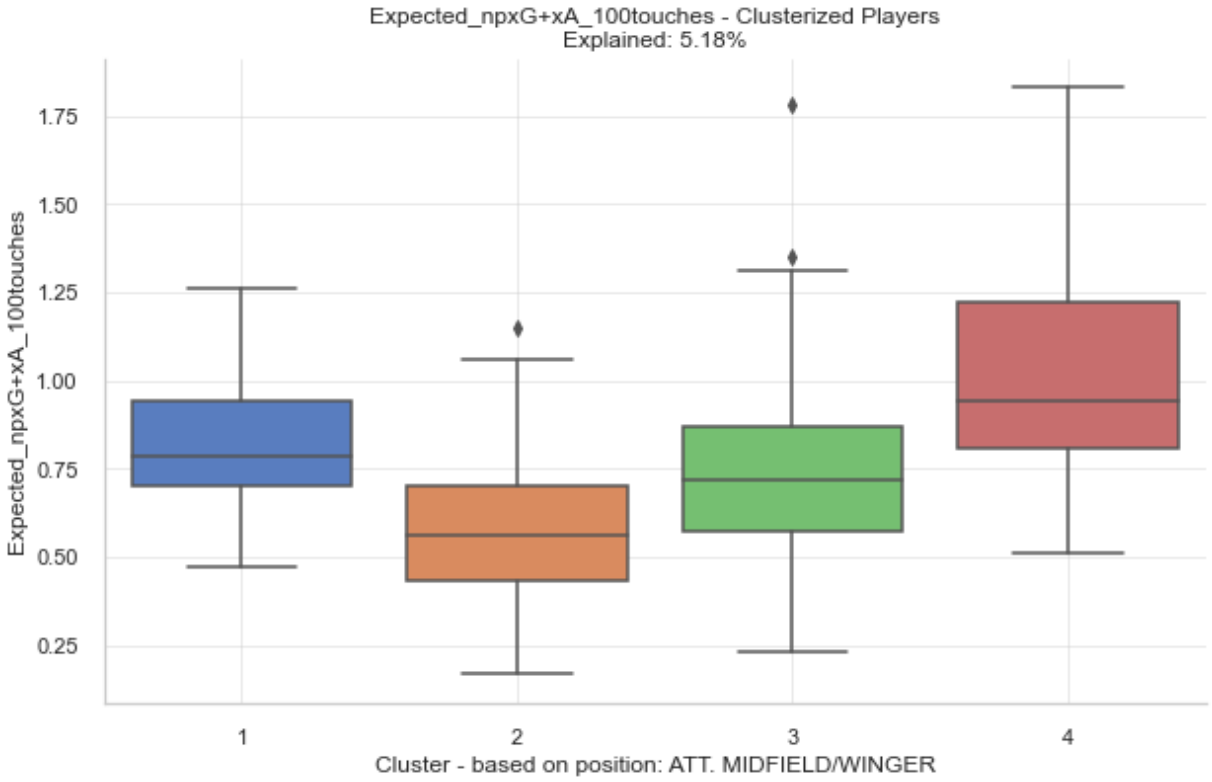
-----POSITION: Att. Midfield/Winger-----

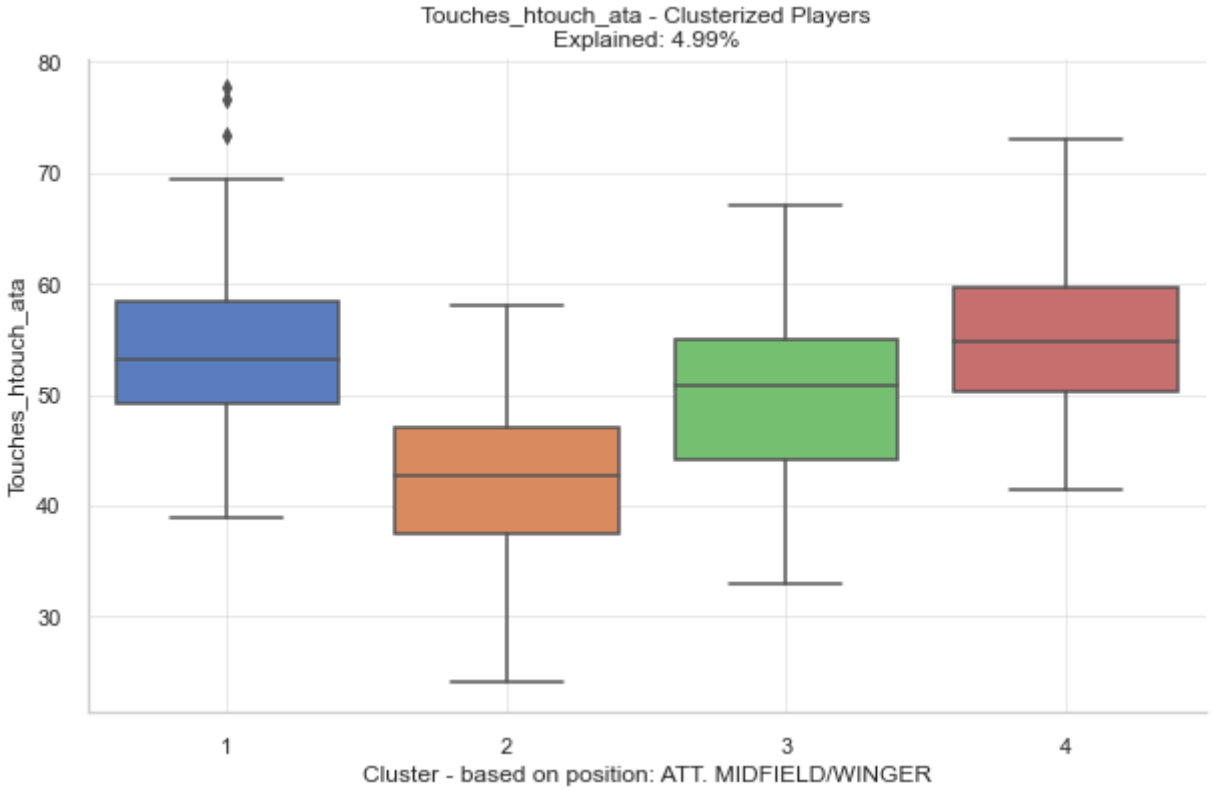

Wing_Natural - Clusterized Players
Explained: 21.83%
Wing_Natural
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER

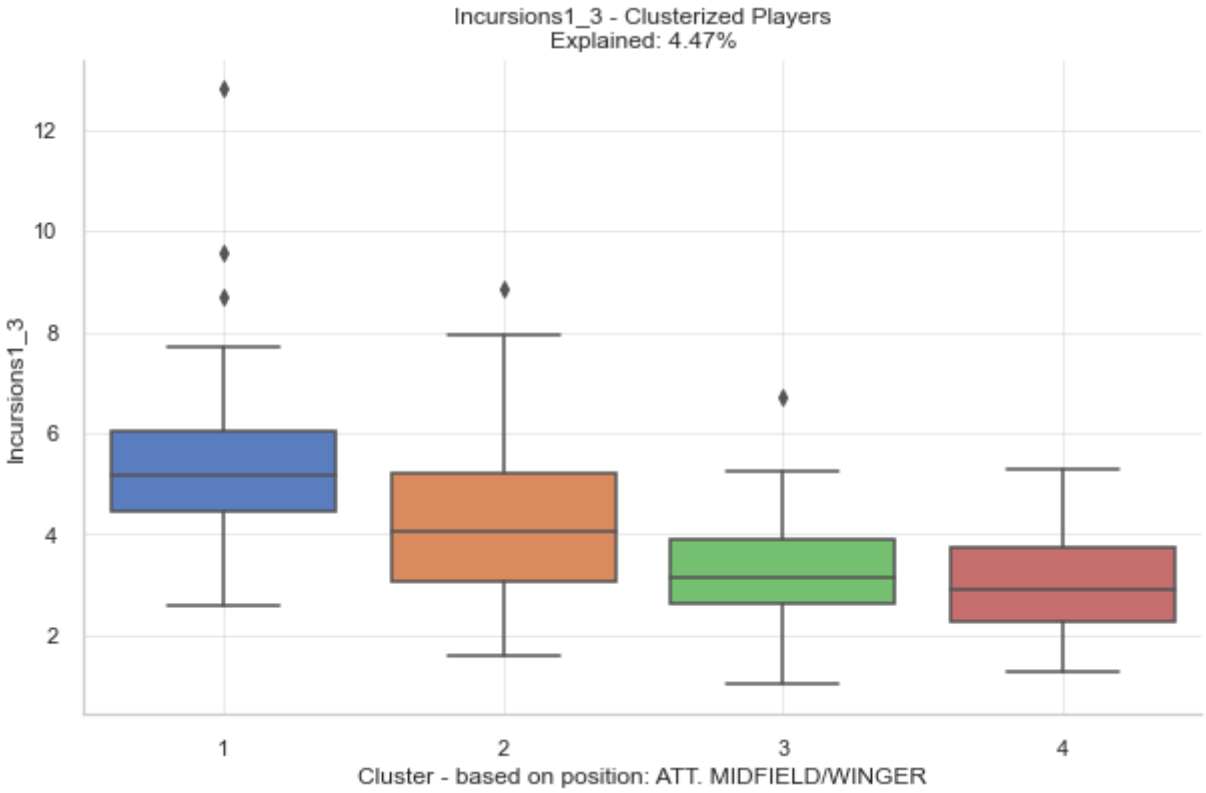

Touches_AttPen% - Clusterized Players
Explained: 10.04%
Touches_AttPen%
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER

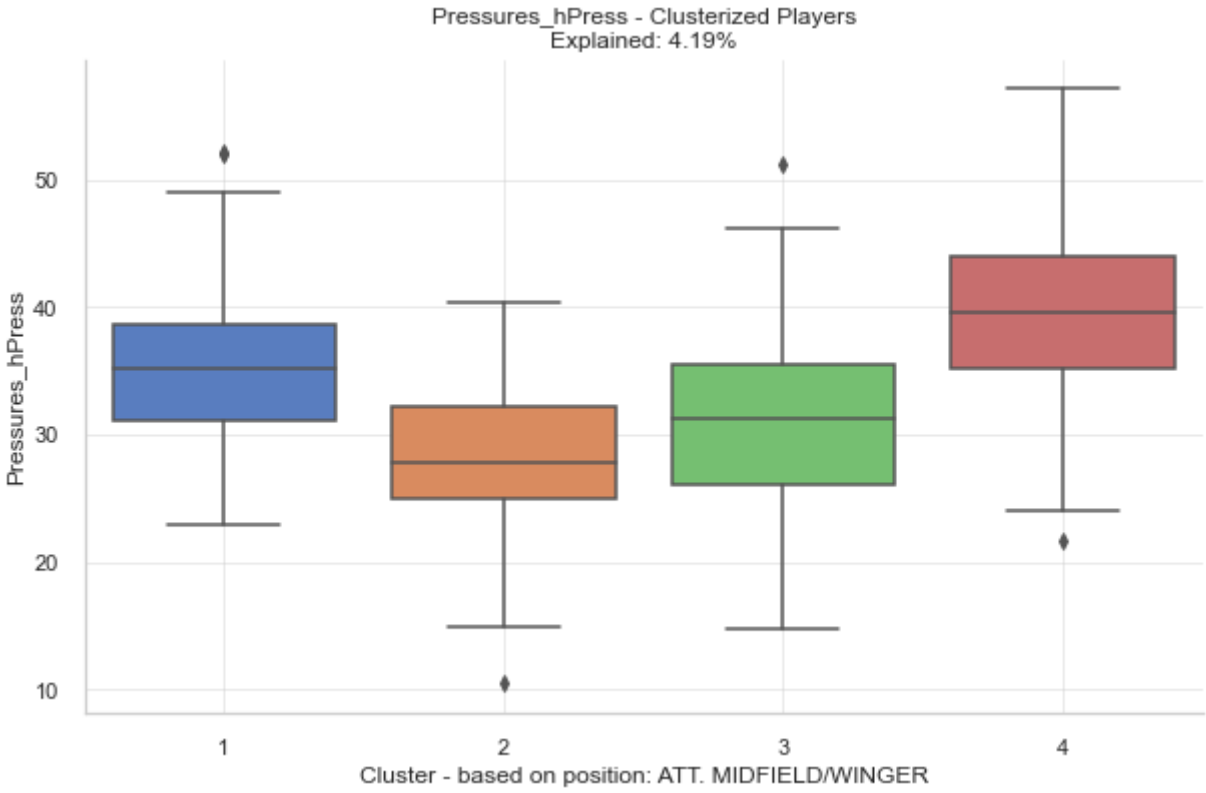

SCATypes_PassLive - Clusterized Players
Explained: 7.50%
SCATypes_PassLive
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER


Carries+ - Clusterized Players
Explained: 5.25%
Carries+
40
35
30
25
20
15
10
5
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER

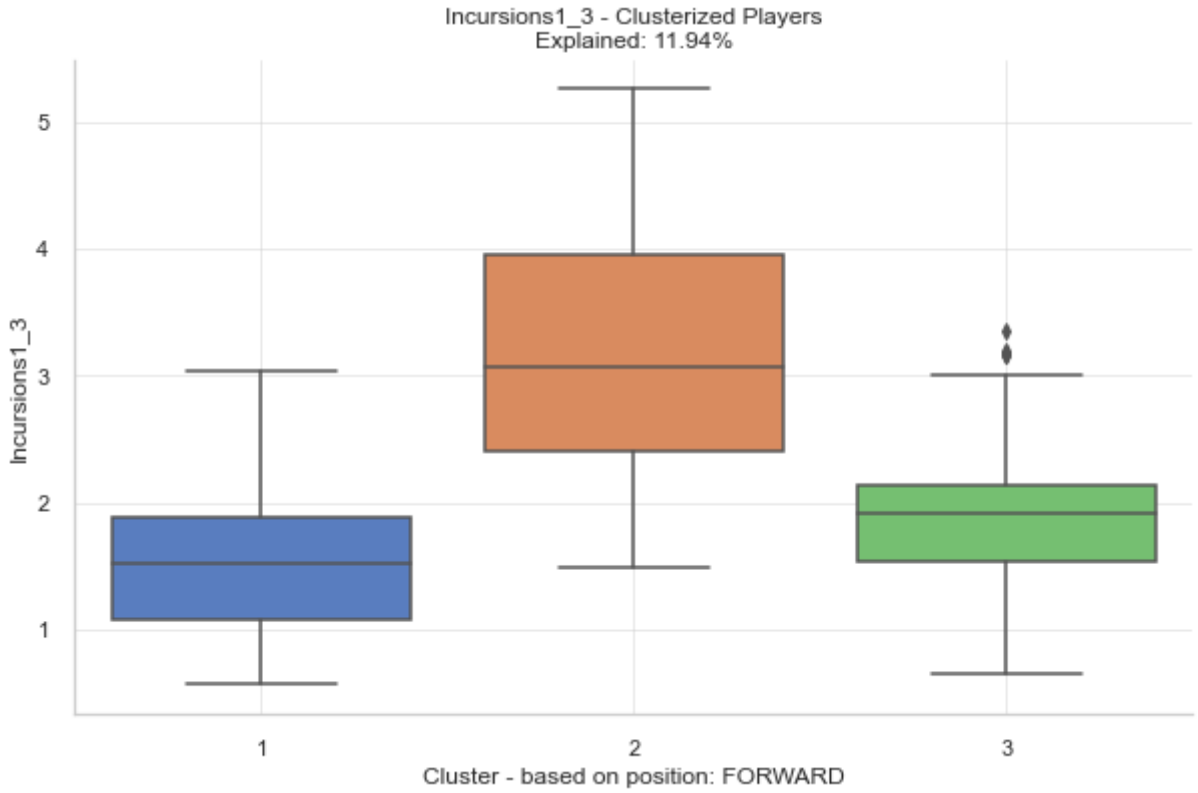

Expected_npxG+xA_100touches - Clusterized Players
Explained: 5.18%
Expected_npxG+xA_100touches
1.75
1.50
1.25
1.00
0.75
0.50
0.25
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER

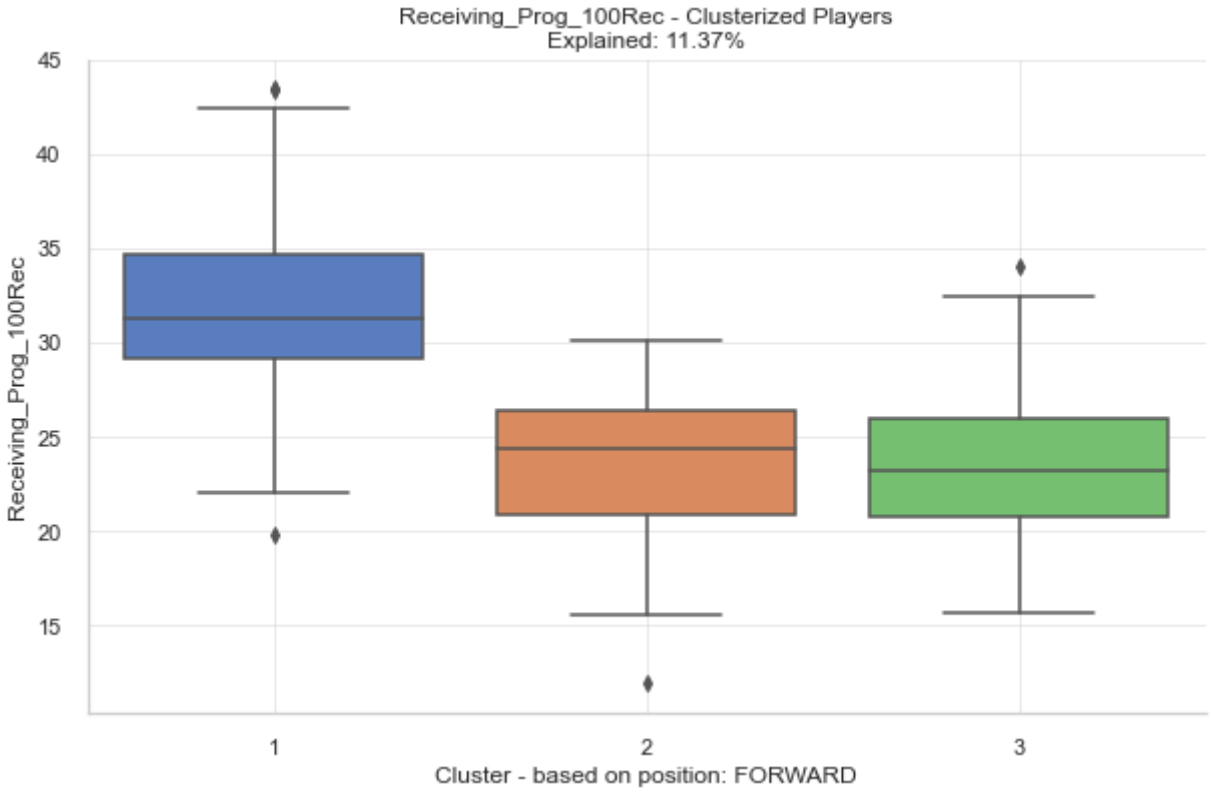

Touches_htouch_ata - Clusterized Players
Explained: 4.99%
Touches_htouch_ata
80
70
60
50
40
30
1
2
3
4
Cluster - based on position: ATT. MIDFIELD/WINGER

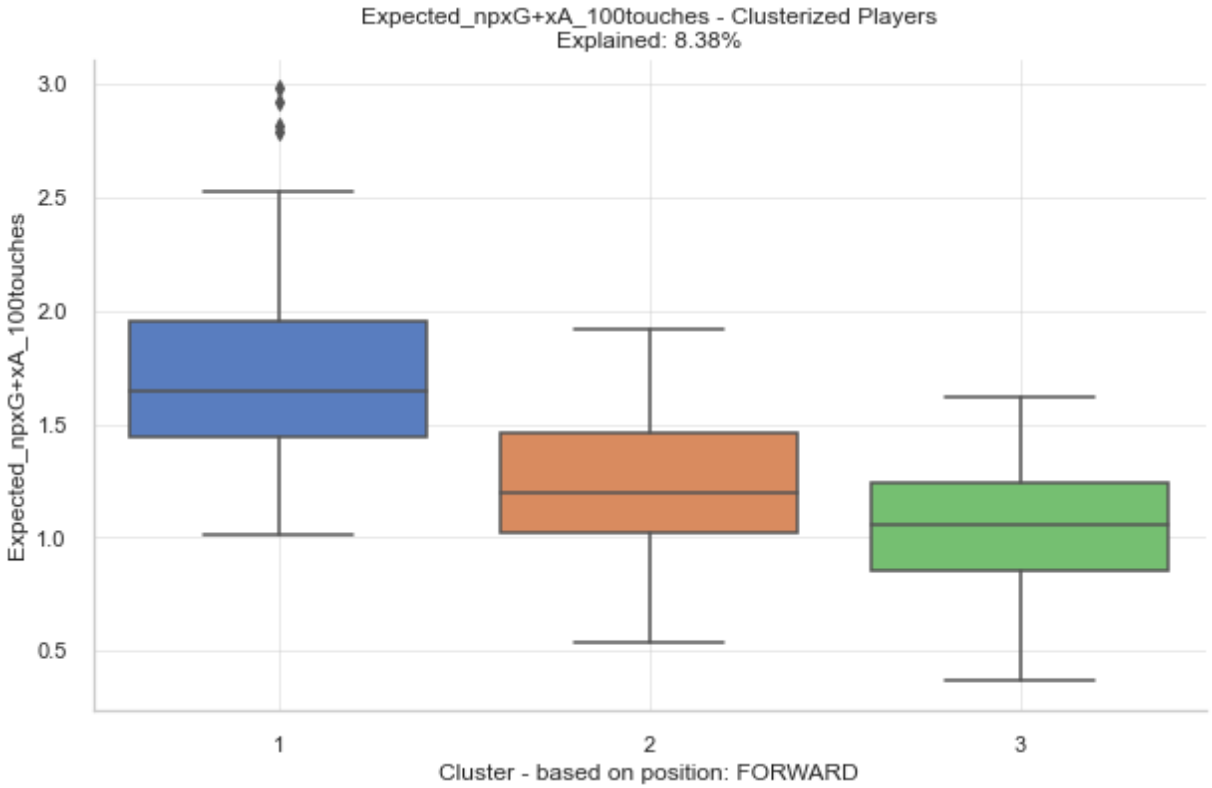


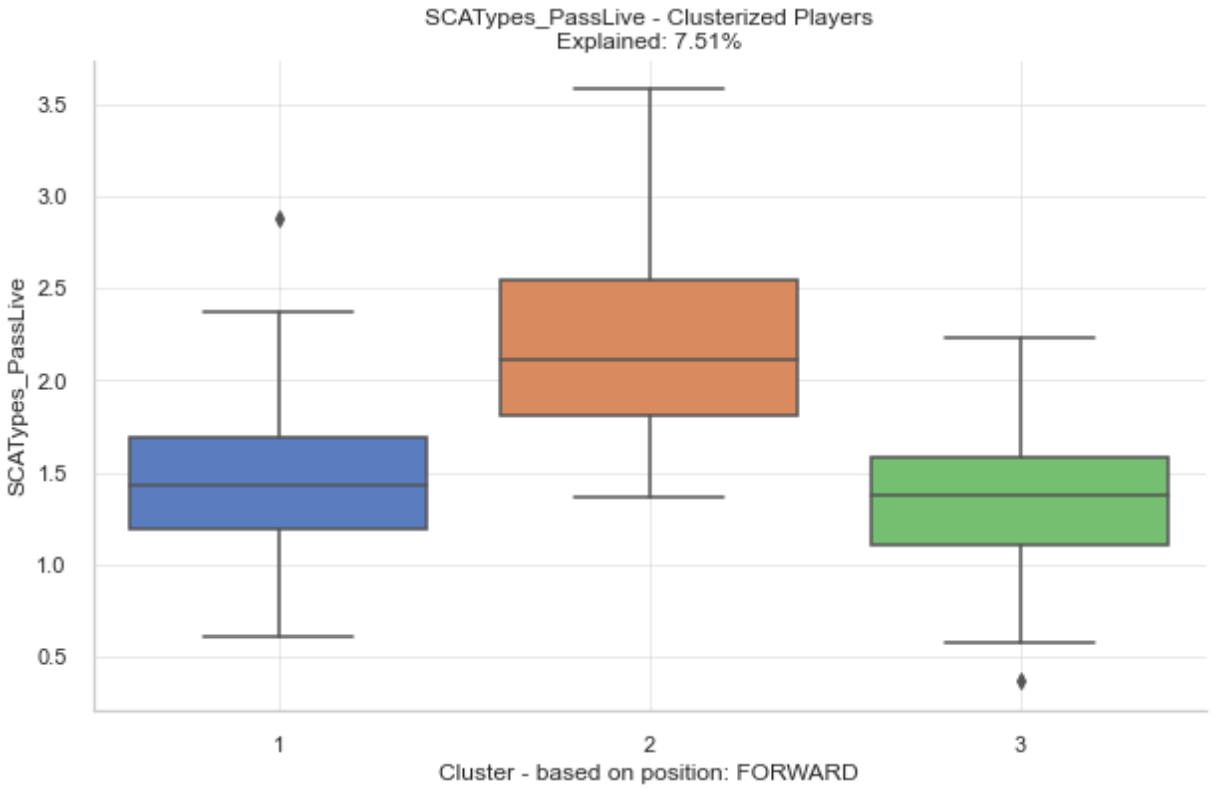


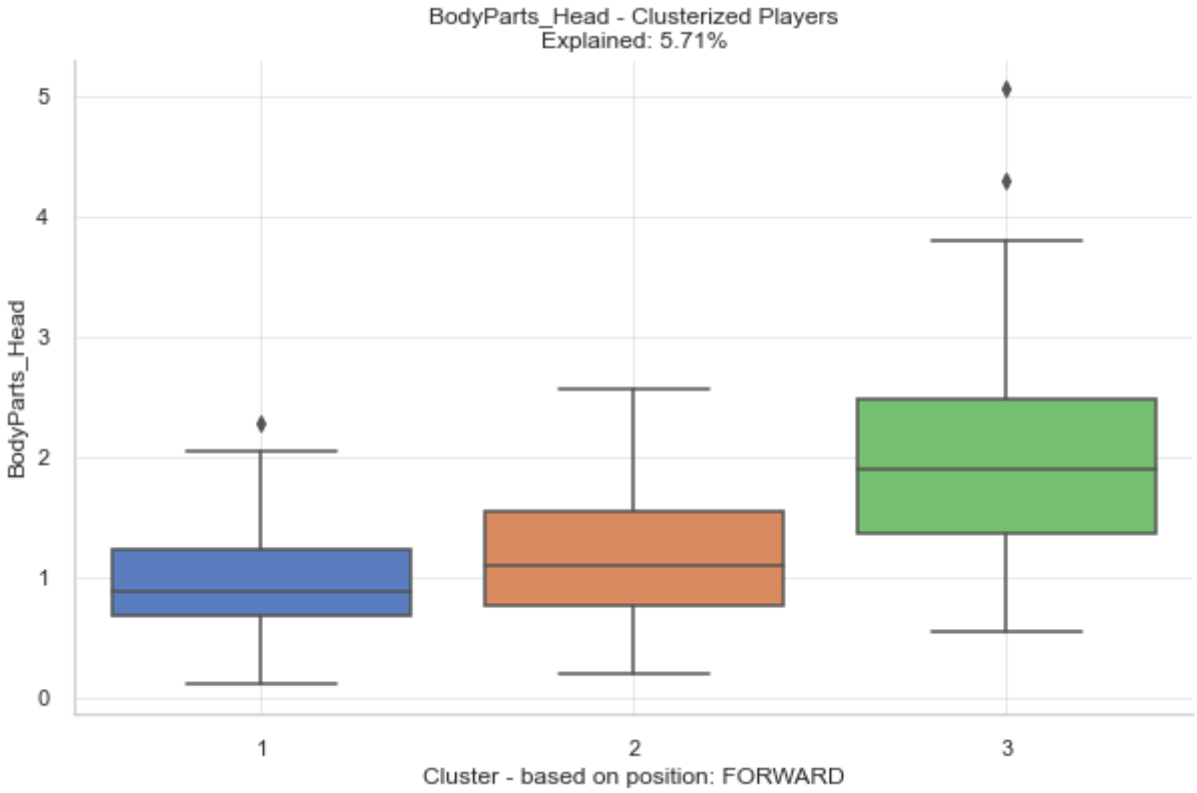
-----POSITION: Forward-----

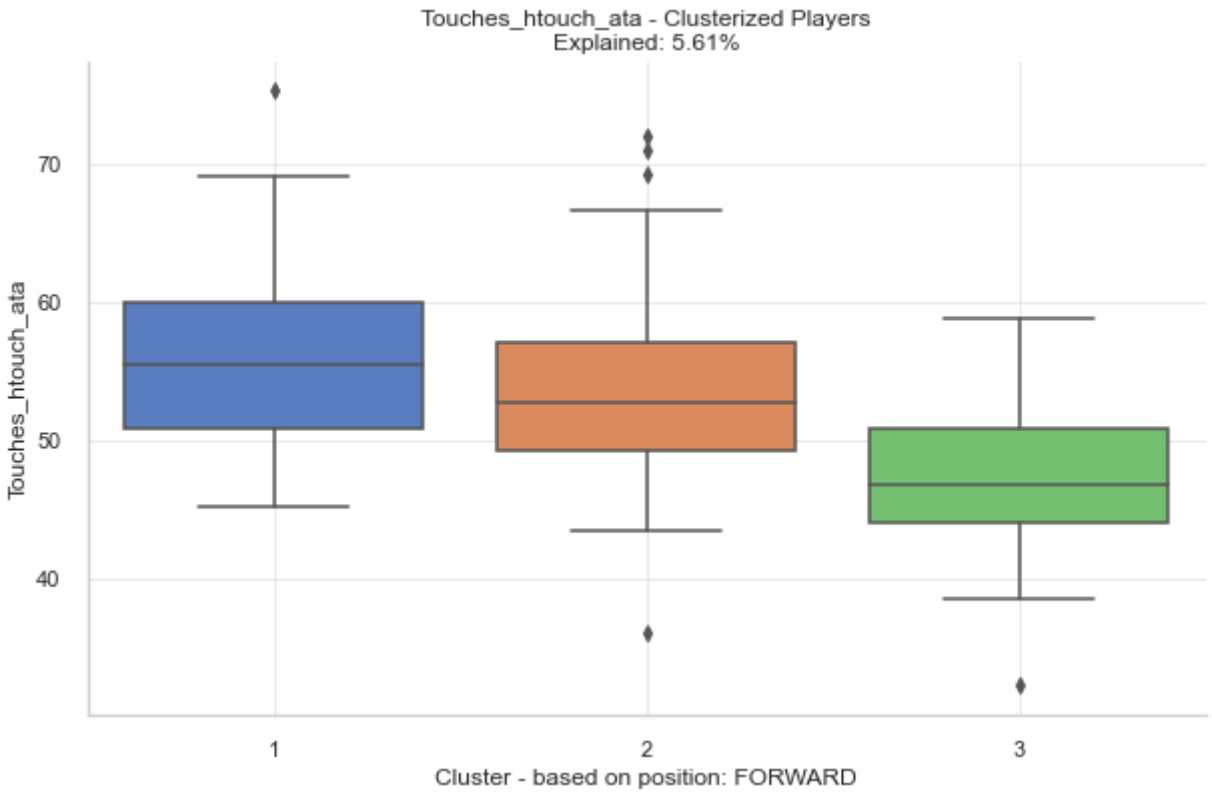

Incursions1_3 - Clusterized Players
Explained: 11.94%
Incursions1_3
5
4
3
2
1
1
2
3
Cluster - based on position: FORWARD

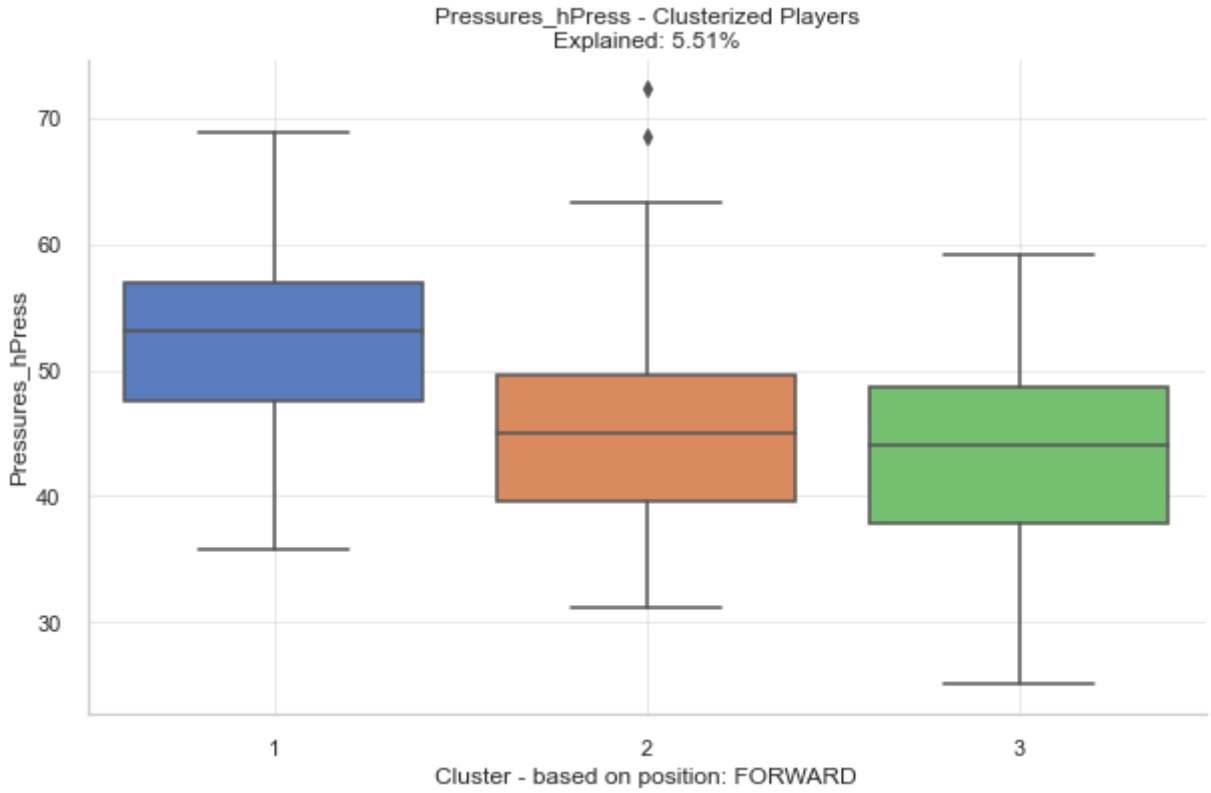

Receiving_Prog_100Rec - Clusterized Players
Explained: 11.37%
Receiving_Prog_100Rec
45
40
35
30
25
20
15
1
2
3
Cluster - based on position: FORWARD

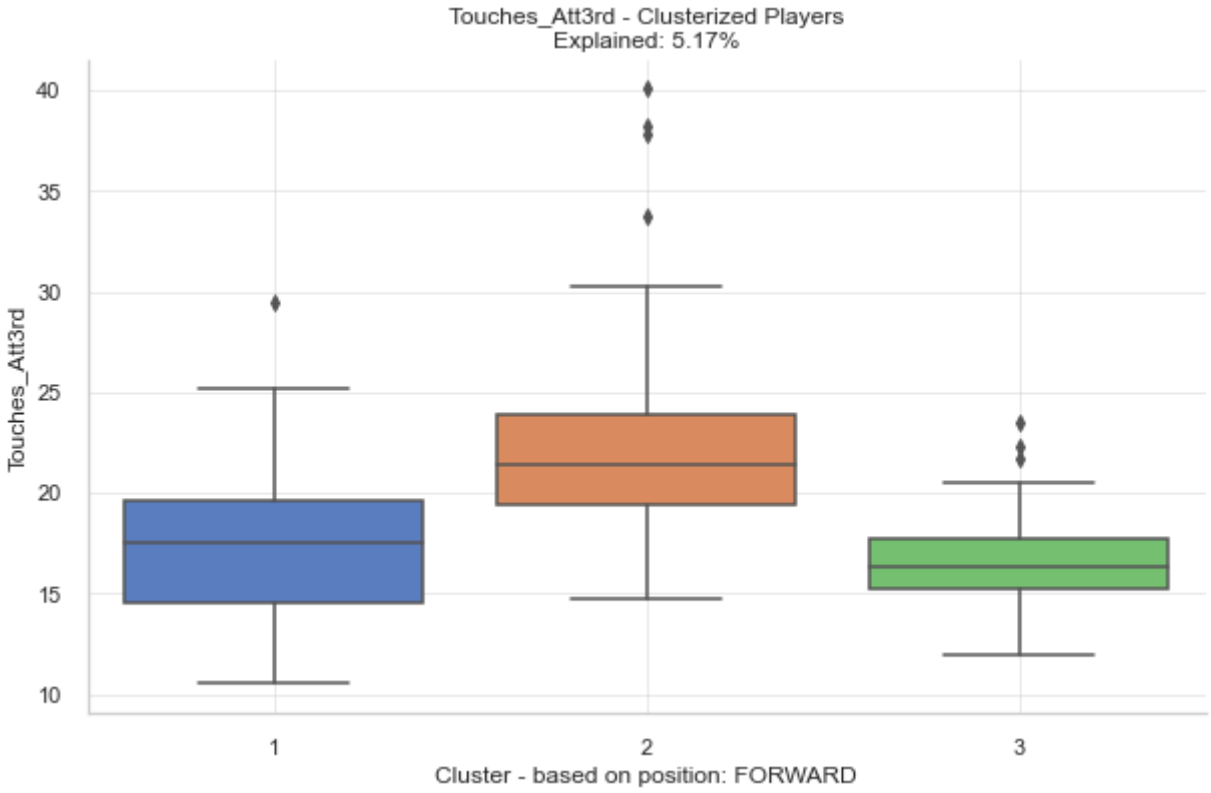

Expected_npxG+xA_100touches - Clusterized Players
Explained: 8.38%
Expected_npxG+xA_100touches
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
Cluster - based on position: FORWARD


SCATypes_PassLive - Clusterized Players
Explained: 7.51%
SCATypes_PassLive
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
Cluster - based on position: FORWARD


BodyParts_Head - Clusterized Players
Explained: 5.71%
BodyParts_Head
0
1
2
3
4
5
1
2
3
Cluster - based on position: FORWARD


Touches_htouch_ata - Clusterized Players
Explained: 5.61%
Touches_htouch_ata
40
50
60
70
1
2
3
Cluster - based on position: FORWARD





Segmentación de Jugadores: Análisis del Resultado

CENTRALES:

- Menor número de pases, pocas conducciones. Mixto en longitud de pase. Mucha defensa en área
- Más participación con balón, más pases y conducciones progresivas. Mucho pase en corto y gran porcentaje de acierto. Gran porcentaje en lases largos. Centrales muy buenos con los pies o de equipo dominante

- Menor participación con balón, a nivel medio en conducciones. Sobresaliente en presiones y pases bloqueados. Balones largos.

In [30]:
```
i='Centre-Back'
print('POSITION: {}'.format(i))
dfn = df[i]
dfn = pd.merge(dfn,players[['Player','Power']],how='left',on='Player')
dfn = dfn.sort_values(by='Power',ascending=False)
for j in list(sorted(dfn['cluster'.format(i)].unique())):
    print('Cluster: ',j)
    dfni= dfn[dfn['cluster'.format(i)]==j]
    print(dfni[['Player']].head(3).values)
    print('\n')
```

```
POSITION: Centre-Back
Cluster:  1
[['Yeray Alvarez']
 ['Robin Knoche']
 ['Luis Hernandez']]


Cluster:  2
[['Lucas Hernandez']
 ['Dayot Upamecano']
 ['Thiago Silva']]


Cluster:  3
[['Cristian Romero']
 ['Josko Gvardiol']
 ['Piero Hincapie']]
```

LATERALES:

- Participa muy arriba, centrador y poco defensor. Pocos toques en campo propio. Poca participación en buildup. CARRILERO EN DEFENSA DE 5
- Baja participación en campo rival con balón, pocos centros y mucha presencia en área propia. CENTRAL RECONVERTIDO
- Alto % de acierto en pases. Lateral que sube pero no finaliza. Poca presencia en área propia. Perfil ofensivo, progresivo. LATERAL OFENSIVO MODERNO/ORGANIZADOR
- Lateral que sube pero no finaliza. Mucha presencia en área propia. LATERAL CLASICO

In [31]:
```
i='Full-Back'
print('POSITION: {}'.format(i))
dfn = df[i]
dfn = pd.merge(dfn,players[['Player','Power']],how='left',on='Player')
dfn = dfn.sort_values(by='Power',ascending=False)
for j in list(sorted(dfn['cluster'.format(i)].unique())):
    print('Cluster: ',j)
    dfni= dfn[dfn['cluster'.format(i)]==j]
    print(dfni[['Player']].head(3).values)
    print('\n')
```

```
POSITION: Full-Back
Cluster:  1
[['Ivan Perisic']
 ['Davide Zappacosta']
```

```
 ['Marcos Alonso']]


Cluster:  2
[['Ben Davies']
 ['Danilo']
 ['Mert Muldur']]


Cluster:  3
[['Jordi Alba']
 ['Nuno Mendes']
 ['Benjamin Pavard']]


Cluster:  4
[['Takehiro Tomiyasu']
 ['Emerson']
 ['Alex Telles']]
```

CENTROCAMPISTAS

- Pivote defensivo posicional, pocas ocasiones generadas en juego abierto. Se mueve por el centro
- Posicionamiento medio, interior de posesión o mediocentro organizador equipo dominante. Muchos pases por partido y ocasiones creadas desde el pase. Se mueve por el centro
- Participa muy arriba con balón y suele buscar último pase. Mediapunta o interior muy ofensivo. Cae a banda

In [32]:

```
i='Midfielder'
print('POSITION: {}'.format(i))
dfn = df[i]
dfn = pd.merge(dfn,players[['Player','Power']],how='left',on='Player')
dfn = dfn.sort_values(by='Power',ascending=False)
for j in list(sorted(dfn['cluster'.format(i)].unique())):
    print('Cluster: ',j)
    dfni= dfn[dfn['cluster'.format(i)]==j]
    print(dfni[['Player']].head(3).values)
    print('\n')
```

```
POSITION: Midfielder
Cluster:  1
[['Casemiro']
 ['Danilo Pereira']
 ['Thiago Mendes']]


Cluster:  2
[['Rodri']
 ['Joshua Kimmich']
 ['Marcelo Brozovic']]


Cluster:  3
[['Nicolo Barella']
 ['Ilkay Gundogan']
 ['Manu Trigueros']]
```

MEDIAPUNTAS/EXTREMOS

- Participa muy arriba,crea muchas ocasiones, mucho impacto en tramo final. Alto volumen de participación. Pocos centros. EXTREMO INVERTIDO o SEGUNDO PUNTA
- Viene a recibir atrás, progresivo. Alto volumen de participación. Pocos centros. Perfil pasador, apenas genera con conducciones – MEDIAPUNTA / INTERIOR COMO FALSO EXTREMO
- Extremo a pie natural. No genera demasiada progresión en último tercio. Presionante en defensa. Centrador , Conductor mas que pasador – EXTREMO PURO
- No genera demasiada progresión en último tercio, bajo volumen de presión, finalizador de gran nivel. Conductor mas que pasador- DELANTERO EN BANDA

In [33]:
```
i='Att. Midfield/Winger'
print('POSITION: {}'.format(i))
dfn = df[i]
dfn = pd.merge(dfn,players[['Player','Power']],how='left',on='Player')
dfn = dfn.sort_values(by='Power',ascending=False)
for j in list(sorted(dfn['cluster'.format(i)].unique())):
    print('Cluster: ',j)
    dfni= dfn[dfn['cluster'.format(i)]==j]
    print(dfni[['Player']].head(3).values)
    print('\n')
```

```
POSITION: Att. Midfield/Winger
Cluster:  1
[['Leroy Sane']
 ['Neymar']
 ['Thomas Muller']]


Cluster:  2
[['Matteo Pessina']
 ['Christoph Baumgartner']
 ['Florian Kainz']]


Cluster:  3
[['Kingsley Coman']
 ['Moussa Diaby']
 ['Alexis Saelemaekers']]


Cluster:  4
[['Raheem Sterling']
 ['Mohamed Salah']
 ['Dejan Kulusevski']]
```

DELANTEROS

- Alto volumen de juego de espaldas o desmarques – Referencia, juega muy arriba y participa poco. 100% área. Poco juego de cabeza. Apenas generan ocasiones desde el pase – NUEVE PURO

- Muchas incursiones en tercio final, muy progresivo. Mucho tiempo fuera del área. Genera desde el pase, no es la referencia – SEGUNDO DELANTERO/DELANTERO EN BANDA/FALSO NUEVE

- Baja a recibir y participa en construcción. Participa muy poco en tercio final. Mucho juego con la cabeza, segunda jugada. Apenas generan ocasiones desde el pase – DELANTERO DE APOYO GRANDE

In [34]:
```
i='Forward'
print('POSITION: {}'.format(i))
dfn = df[i]
dfn = pd.merge(dfn,players[['Player','Power']],how='left',on='Player')
dfn = dfn.sort_values(by='Power',ascending=False)
for j in list(sorted(dfn['cluster'.format(i)].unique())):
    print('Cluster: ',j)
    dfni= dfn[dfn['cluster'.format(i)]==j]
    print(dfni[['Player']].head(3).values)
    print('\n')
```

```
POSITION: Forward
Cluster:  1
[['Robert Lewandowski']
 ['Andre Silva']
 ['Lautaro Martinez']]


Cluster:  2
[['Kylian Mbappe']
 ['Karim Benzema']
 ['Gabriel Jesus']]


Cluster:  3
[['Gianluca Scamacca']
 ['Lucas Holer']
 ['Marko Arnautovic']]
```

Funciones de Distancia

Los dos procesos que se exponen a continuación tienen como objetivo convertir todo el modelado analítico anterior en conocimiento práctico. Ambas tienen en común, por un lado, su propósito de servir a la dirección deportiva en la tarea de afinar y basar en datos el proceso de reclutamiento de jugadores; y, por otro, el uso de la distancia euclídea como medio para determinar las conclusiones. Sin embargo, el punto de partida y los significados son muy distintos, lo que las diferencia y también complementa, como veremos en los casos prácticos.

- La función **player_Similarities** mide el grado de similitud existente entre un jugador y el resto de sus homónimos. Toma un jugador y devuelve una tabla con la distancia euclídea respecto a los más parecidos.
- La función **team_Mapping**, por su parte, toma unos datos base correspondientes a un modelo de juego concreto y devuelve los jugadores cuyos atributos se encuentran más cercanos -y, por tanto, son más adecuados para adaptarse a ese contexto-. Toma una subposición y un equipo, y devuelve una tabla con la distancia euclídea respecto a los jugadores mas cercanos a esos datos base.

Player_Similarities encaja con el prototipo de función de medida de distancias, y sus resultados son sencillos de ver en cualquier web de analítica avanzada de jugadores. Esta función cobra relevancia en la práctica cuando nos interesa sustituir a un futbolista cuyo perfil y rol está muy definido. Un caso claro sucederá cuando un club netamente vendedor deba dar salida a una pieza clave y quiera buscar un sustituto lo más similar posible, para evitar redefinir su modelo de juego.

A continuación se expone la función y algunos ejemplos de su aplicación.

In [35]:
```
def player_similarities(p):
    print('Looking for similar profiles to {}'.format(p.upper()))
    player_name = p.title()

    data_player = players[players['Player']==p]
    idx = data_player.idx.unique()[0]
    name = data_player.PosE.unique()[0]
    name2 = data_player.POS.unique()[0]
    pl = players[(players.Min>=min_minutes) & (players.PosE==name)]
    data_pos = df[name]
    pl = pl.reset_index()
    pl = pl.drop('index',axis=1)


    reduced=data_pos

    reduced = pd.merge(reduced,pl[['idx','POS']],how='left',left_index=True,right_in
    y = reduced[reduced['idx']==idx]
    y.drop(['idx','Player','POS'],inplace=True,axis=1)
    pl = list(reduced.Player)
    pos = list(reduced.POS)
    reduced.drop(['idx','Player','POS'],inplace=True,axis=1)
    euc = []
    for i in reduced.values:
        euc.append(distance.euclidean(y.values,i))
    simil = pd.DataFrame(euc,index=[pl,pos],columns=['Similarity_Score'])
    simil = simil.reset_index()
    simil.columns = ['Player','POS','Similarity_Score']
    simil = simil[simil.Player!=player_name]
    if 't-Back' in name2:
        simil = simil[simil.POS==name2]

    simil = simil.sort_values(by='Similarity_Score',ascending=True)


    simil.drop('POS',axis=1,inplace=True)



    return simil
```

In [36]:
```
ps = player_similarities('Toni Kroos')
ps.head(10)
```

```
Looking for similar profiles to TONI KROOS
```

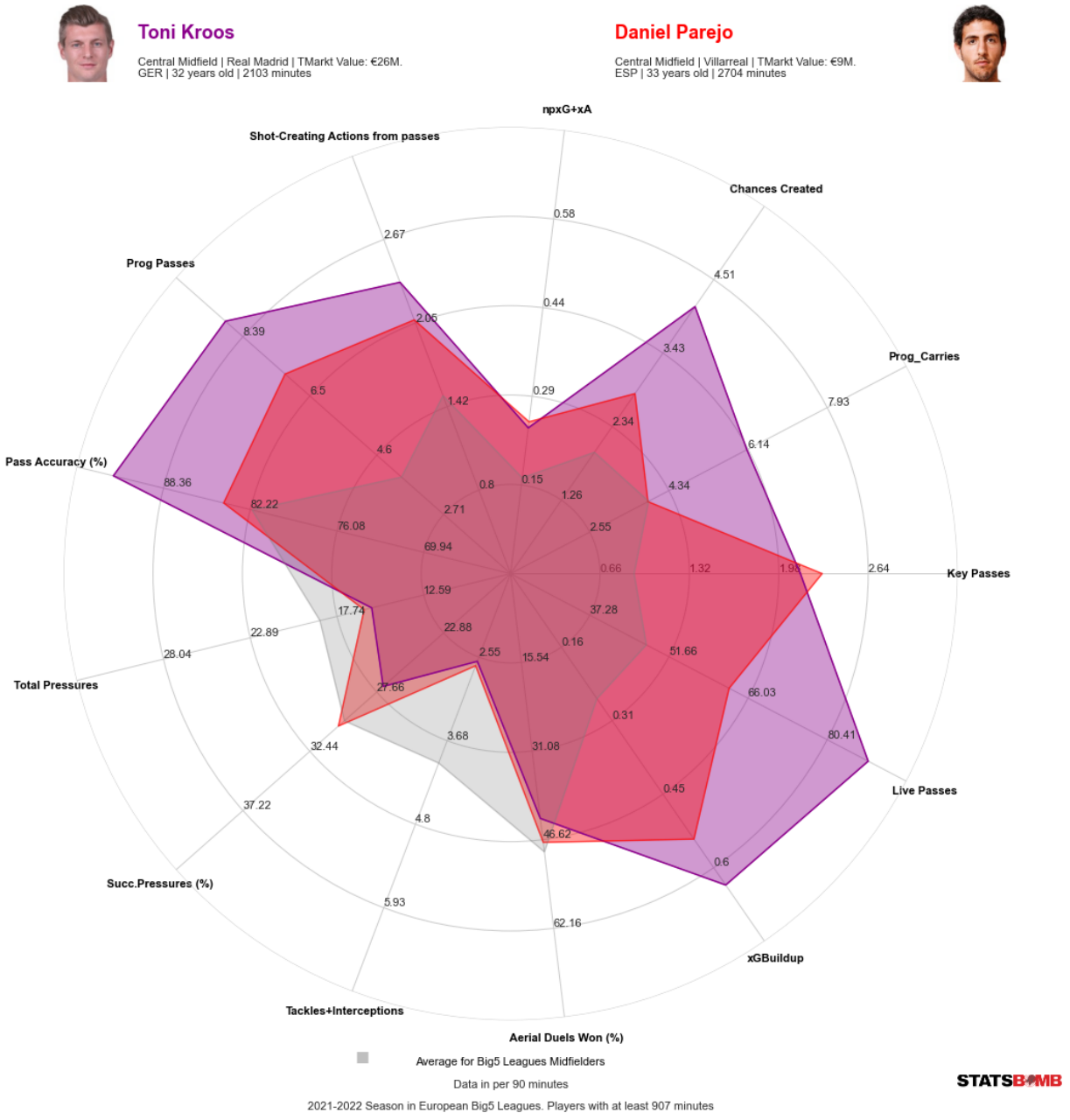
Out[36]:

	Player	Similarity_Score
225	Renato Sanches	0.969778
161	Luka Modric	1.035175

	Player	Similarity_Score
50	Daniel Parejo	1.177498
24	Marcelo Brozovic	1.228471
142	Rodrigo De Paul	1.272192
95	Fabian Ruiz Pena	1.279819
103	Granit Xhaka	1.340301
60	Jordan Henderson	1.342466
106	Jordan Veretout	1.359928
203	William Carvalho	1.365398

Procedemos a comparar en el radar a Toni Kroos con alguno de los jugadores con los que presenta mayor similitud.

In [37]:
```
pp.radar_comp('Toni Kroos','Daniel Parejo')
```



In [38]:
```
ps = player_similarities('Raphinha')
ps.head(10)
```

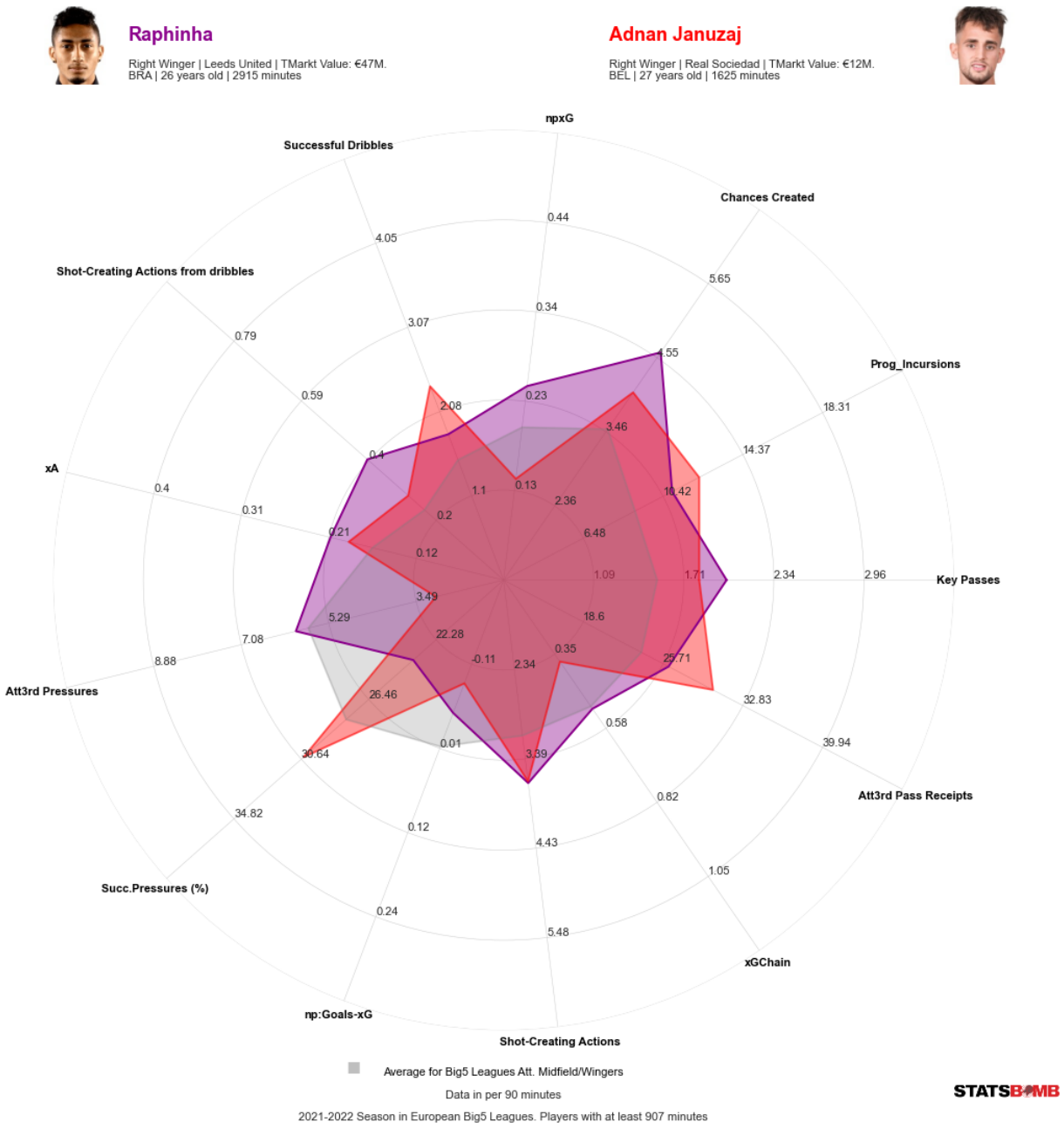
```
Looking for similar profiles to RAPHINHA
```

Out[38]:

	Player	Similarity_Score
20	Domenico Berardi	0.710781
189	Takefusa Kubo	0.822620
106	Nedim Bajrami	0.846301
73	Lorenzo Pellegrini	0.853288
181	Adnan Januzaj	0.859074
123	Sofiane Diop	0.878202
272	Calvin Stengs	0.892430
190	Ruslan Malinovskyi	0.905753
26	Nabil Fekir	0.910338
66	Sofiane Boufal	0.933621

En este caso, es el brasileño Raphinha el jugador que estamos comparando. El jugador de Porto Alegre dejará este verano una gran cantidad de dinero en las arcas del Leeds, tras dos años de excelente rendimiento. Los de Yorkshire podrían aprovechar la plusvalía generada y buscar un sustituto lo más similar posible a Raphinha. De la selección mostrada, destaca el belga Adnan Januzaj, que ha finalizado su contrato con la Real Sociedad y se podría incorporar libre de pago de traspaso. El jugador de ascendencia kosovar muestra, con un valor de mercado claramente inferior, unas métricas ofensivas comparables a las del genial extremo carioca

In [92]:
```
pp.radar_comp('Raphinha','Adnan Januzaj')
```



A nivel práctico, una dirección deportiva que coloque a los datos en el punto central de la toma de decisiones tratará de combinar ambos modelos. Mediante *team_mapping* podemos conocer, ante una necesidad que surja en la plantilla o una posible mejora, cuáles son los futbolistas de la posición en cuestión que más se aproximan al modelo de juego y a las condiciones tácticas que tenemos. Este modelo puede aportar información importante respecto al perfil necesario a incorporar, y puede ayudar a acotar a los posibles candidatos.

Si además dicha necesidad que hace ir al club al mercado de fichajes viene motivada por la salida de un futbolista importante, es conveniente combinar el resultado de esta función con el de la anterior, y evaluar de forma detallada desde la perspectiva técnica -ojeo, visionar partidos, informarse sobre su personalidad y entorno, valorar su historial de lesiones, tantear si es posible la negociación con el club al que pertenece, etc- las condiciones de aquellos jugadores que figuren en las listas resultantes y encajen en la política financiera del club.

Si realmente la incorporación que se pretende realizar viene a satisfacer un rol que no existía hasta hoy en la plantila, no tendría tanto sentido emplear la función *player_similarities*, mientras que *team_mapping* seguiría ofreciendo cierta información relevante.

In [40]:

```
def team_mapping(team,position):
    cluster_cols=[]
    for i in squad.columns:
        if 'cluster' in i or i=='Squad':
            cluster_cols.append(i)
    squad[cluster_cols] = squad[cluster_cols].astype(str)
    pl = pd.merge(players,squad[cluster_cols],how='left',on='Squad')
    clus = cluster_cols
    for i in clus:
        if i=='Squad':
            clus.remove('Squad')
    if 'Defensive' in position or 'Back' in position:
        clus.remove('attacking_approach_cluster')

    else:
        pass
    cluster_comb = squad[squad['Squad']==team].set_index(clus)
    cluster_comb.index=cluster_comb.index.map(''.join)
    cluster_comb = cluster_comb.index[0]
    grouped = players[(players['POS']==position) & (players['Min']>=min_minutes)]
    name = grouped.PosE.unique()[0]
    grouped = grouped.groupby(by=clus).mean()
    data_cluster = grouped[pos_dict[name]]
    data_cluster.index = data_cluster.index.map(''.join)
    data_cluster = data_cluster[data_cluster.index==cluster_comb]

    pca_app = pcas[name]

    pl = players[players['Min']>=min_minutes]
    data_pos= pl[pl['PosE']==name]
    data_pos.reset_index(inplace=True)
    data_pos.drop('index',inplace=True,axis=1)
    data = data_pos[pos_dict[name]]

    data = pd.concat([data,data_cluster])
    data.reset_index(inplace=True)
    data.drop('index',inplace=True,axis=1)


    data_norm = scaler.fit_transform(data)


    pca = PCA(n_components = pca_app,random_state=0)
    reduced = pd.DataFrame(pca.fit_transform(data_norm))

    y = reduced.tail(1)
    #y.drop('cluster',axis=1,inplace=True)
    reduced = reduced.head(reduced.shape[0])
    reduced = pd.merge(reduced,data_pos[['Player','idx','POS','Squad','Power','value
    #reduced['idx'] = reduced['idx'].fillna('{}-{}'.format(team,position))
    #reduced['Player'] = reduced['Player'].fillna('{}-{}'.format(team,position))
    if name=='Midfielder' and position=='Defensive Midfield':
        pass
    elif 'Winger' in position:
        reduced = reduced[(reduced['POS']=='Right Winger')| (reduced['POS']=='Left W
    else:
        reduced = reduced[reduced['POS']==position]
    pl = list(reduced.Player)
    idx = list(reduced.idx)
    sq = list(reduced.Squad)
    pw = list(reduced.Power)
    vl=list(reduced.value)
    ag=list(reduced.Age)
    ft=list(reduced.foot)
```

```
    reduced.drop(['idx','Player','POS','Squad','Power','value','Age','foot'],inplace
    euc = []
    for i in reduced.values:
        euc.append(distance.euclidean(y.values,i))
    simil = pd.DataFrame(euc,index=[pl,idx,sq,ag,ft,vl,pw],columns=['Team_Similarity

    simil = simil.sort_values(by='Team_Similarity_Index',ascending=True)

    simil = simil.reset_index()
    simil.columns = ['Player','idx','Squad','Age','foot','value','Power','Team_Simil
    simil['Team_Similarity_Index'] = round(simil['Team_Similarity_Index'],3)

    simil = simil[simil.Squad!=team]
    simil = simil.dropna()
    return simil
```

Esta función, como vemos, agrupa a todos los equipos pertenecientes a una misma serie de clusters de equipo y extrae un valor medio de todas sus métricas. Ese será el punto de referencia, para los equipos que pertenezcan a dichos clusters, a la hora de buscar jugadores similares.

El hecho de que el modelo devuelva jugadores que se aproximen a un retrato medio implica que vayan a existir similitudes a nivel de perfil, pero evidentemente no quiere decir que el nivel del jugador en cuestión se corresponda con el del equipo que se trata. Por ello, y también debido a que se debe tener en cuenta la realidad económica del propio club, es importante tener en cuenta las variables de valor de mercado, edad o las métricas de rendimiento más punteras. En las tablas de abajo mostraremos *Power*, que mide el grado de implicación del jugador en la generación de valor en el juego, medido éste como los xG. Evidentemente, esta métrica no es lo suficientemente precisa ni global como para definir el nivel de un futbolista, pero sí servirá de orientación en este caso.

Conviene poner el foco en las similitudes, y para ello mostraremos algunos ejemplos rápidos, pasando el equipo y la necesidad en cuestión, y en la que se nos devolverá una lista con los futbolistas más próximos a las métricas del modelo de juego del equipo indicado.

In [41]:
```
tm = team_mapping('Manchester City','Centre-Back').head(20)
tm
```

Out[41]:

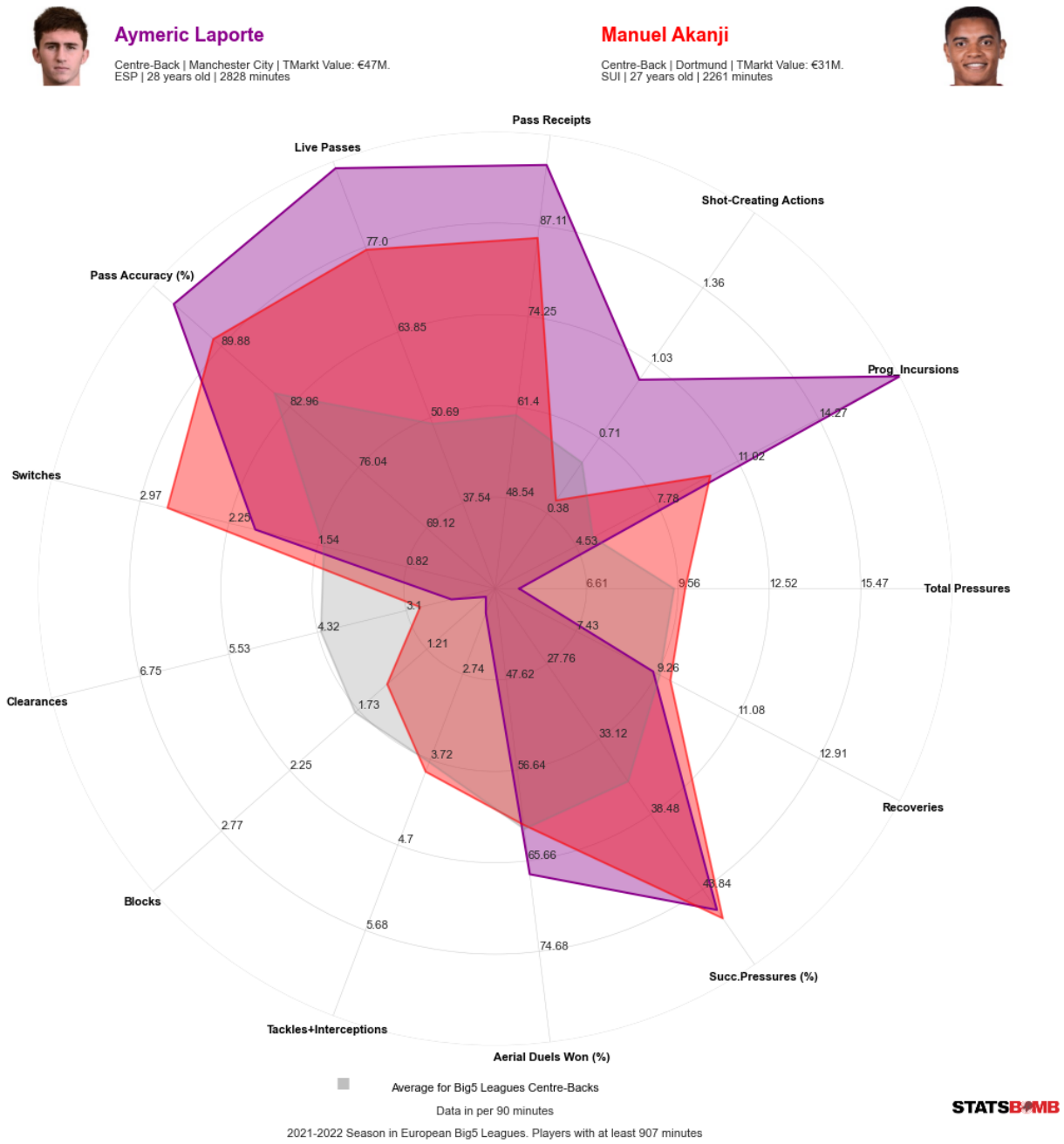
	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index
0	Dante	fd2bc67d	Nice	39.0	Left	0.0	0.39	0.587
1	Warmed Omari	4534a352	Rennes	22.0	Right	8.0	0.64	0.631
2	Jules Kounde	4d1666ff	Sevilla	24.0	Right	63.0	0.42	0.652
3	Pau Torres	532e1e4f	Villarreal	25.0	Left	52.0	0.51	0.654
4	Ronald Araujo	2bef2bca	Barcelona	23.0	Right	42.0	0.46	0.654
5	Duje Caleta-Car	94f8d10c	Marseille	26.0	Right	16.0	0.36	0.713
6	Francesco Acerbi	b96b595c	Lazio	34.0	Left	5.0	0.38	0.728
7	Axel Disasi	ad82197c	Monaco	24.0	Right	17.0	0.36	0.729
8	Nikola Milenkovic	bee704fc	Fiorentina	25.0	Right	24.0	0.34	0.736
9	Damien Da Silva	1ec7e1c5	Lyon	34.0	Right	1.0	0.20	0.746

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index
10	Manuel Akanji	89ac64a6	Dortmund	27.0	Right	31.0	0.55	0.753
11	Nayef Aguerd	288e1e13	Rennes	26.0	Left	12.0	0.54	0.768
12	Eric Garcia	2bed3eab	Barcelona	21.0	Right	19.0	0.72	0.774
13	Antonio Rudiger	18b896d6	Chelsea	29.0	Right	36.0	0.68	0.787
14	Gerard Pique	adfc9123	Barcelona	35.0	Right	5.0	0.72	0.795
15	Robin Le Normand	9a64e67e	Real Sociedad	26.0	Right	31.0	0.24	0.797
16	Thiago Silva	86e7deaf	Chelsea	38.0	Right	2.0	0.82	0.810
17	Virgil van Dijk	e06683ca	Liverpool	31.0	Right	57.0	0.65	0.813
18	Joel Matip	b217ef29	Liverpool	31.0	Right	19.0	0.61	0.819
19	Ibrahima Konate	5ed9b537	Liverpool	23.0	Right	31.0	0.13	0.823

Vemos en el ejemplo el listado de centrales que más se acercan a las métricas del modelo del Manchester City para esa posición. Como vimos antes en el perfilado, los centrales de ese cluster se caracterizan por tener un excelente juego de pies y una extrardinaria precisión en cambios de orientación. También deben ser rápidos, pues el conjunto de Pep Guardiola coloca la línea defensiva muy lejos de su portero, y con capacidad de anticipar.

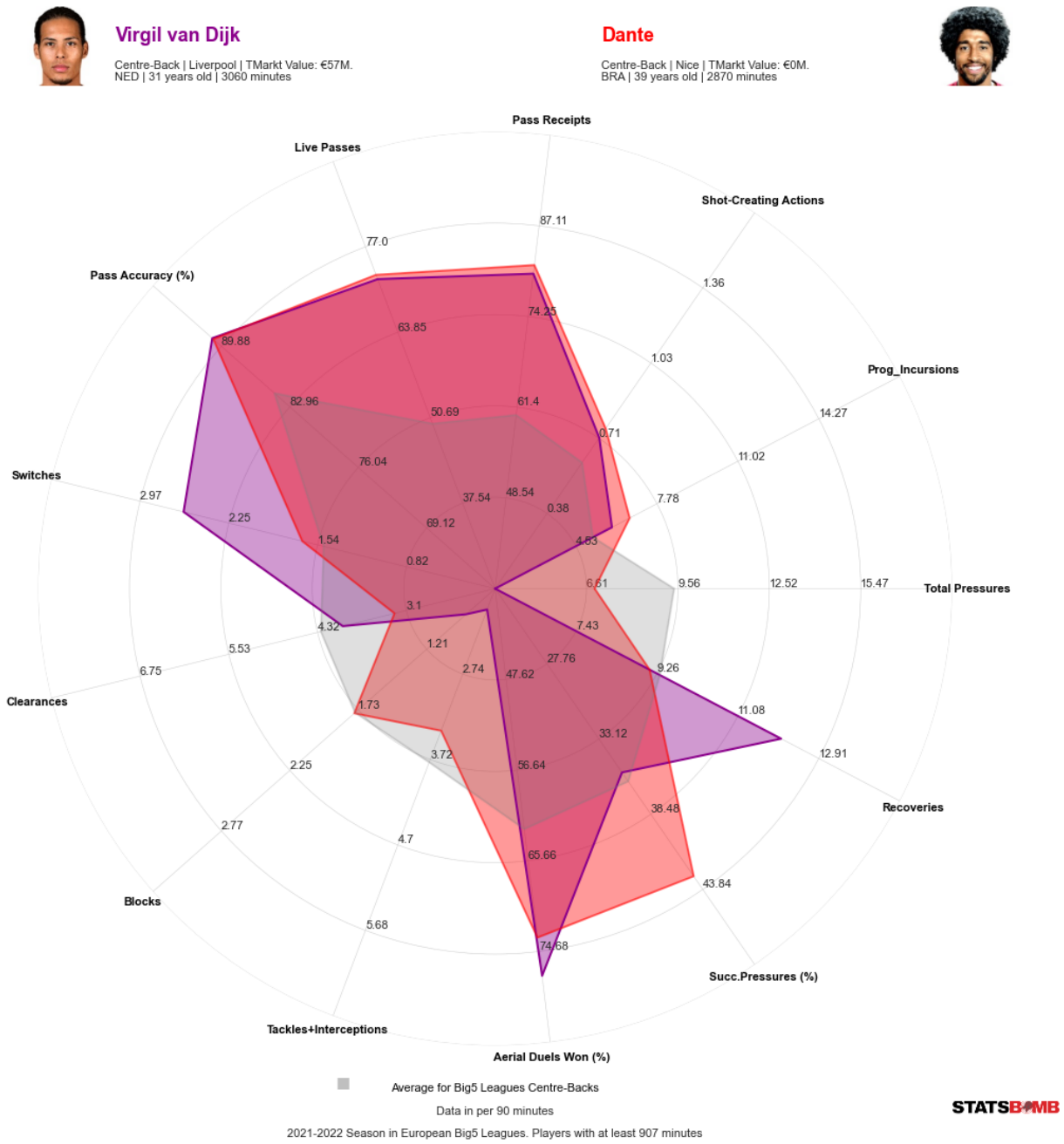
Tomamos como ejemplo, más abajo, a Manuel Akanji, y medimos en el radar su similitud con uno de los centrales *citizens*, el internacional español Aymeric Laporte

In [91]:
```
pp.radar_comp('Aymeric Laporte','Manuel Akanji')
```



El ejemplo anterior también sirve de ilustración para aclarar una pretendida característica de este modelo: en ningún momento se pretende -aunque se entiende que hay una correlación efectiva en éste, al menos de momento- que el nivel del jugador entre en la ecuación. En otras palabras, el modelo sólo valora la adecuación del jugador a un sistema, por lo que mide el perfil del futbolista, y no su calidad. Vemos que, de hecho, el brasileño Dante Bonfim figura en primer lugar pese a su edad -39 años- y las prestaciones negativas mostradas en el pasado en los grandes entornos -Bayern, Wolfsburgo o Brasil en 2014-, pese a su buena campaña 2021-22 en el Niza. El modelo valora, en este caso, el perfil de central cómodo con balón y jugando en línea adelantada que representa el central carioca, sin entrar en su capacidad de acierto, solidez, errores y otros valores relacionados con el nivel del jugador.

In [90]:
```
pp.radar_comp('Virgil van Dijk','Dante')
```



Para comprobar la ambivalencia del modelo, realizamos ahora otra búsqueda de centrales, pero en esta ocasión adaptados al estilo del Inter de Milán, un equipo que, si bien defiende arriba y exige también una salida limpia desde atrás, cuenta con una diferencia fundamental con el Manchester City: forma en una línea de 3 centrales desde hace años.

In [44]:
```
tm = team_mapping('Inter','Centre-Back').head(15)
tm
```

Out[44]:

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index
0	Kevin Danso	6e33125f	Lens	24.0	Right	9.0	0.46	0.501
1	Axel Disasi	ad82197c	Monaco	24.0	Right	17.0	0.36	0.546
2	Waldemar Anton	5e66fa06	Stuttgart	26.0	Right	6.0	0.42	0.573
3	Ronald Araujo	2bef2bca	Barcelona	23.0	Right	42.0	0.46	0.596
4	Caglar Soyuncu	21166ff4	Leicester City	26.0	Right	47.0	0.28	0.607

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index
5	Davinson Sanchez	da7b447d	Tottenham	26.0	Right	31.0	0.57	0.615
6	Mohamed Simakan	fcb27134	RB Leipzig	22.0	Right	19.0	0.31	0.628
7	Eder Militao	2784f898	Real Madrid	24.0	Right	63.0	0.54	0.628
8	Odilon Kossonou	1f3afdcb	Leverkusen	21.0	Right	21.0	0.39	0.632
9	Max Kilman	d0f72bf1	Wolves	25.0	Left	16.0	0.23	0.635
10	Alessio Romagnoli	3ead85f9	Milan	27.0	Left	21.0	0.31	0.642
11	Ben White	35e413f1	Arsenal	25.0	Right	42.0	0.39	0.647
12	Kevin Bonifazi	fb77bfef	Bologna	26.0	Right	4.0	0.11	0.648
13	Roger Ibanez	82efe6fa	Roma	24.0	Right	26.0	0.34	0.655
14	Anel Ahmedhodzic	eab957a3	Bordeaux	23.0	Right	6.0	0.19	0.660

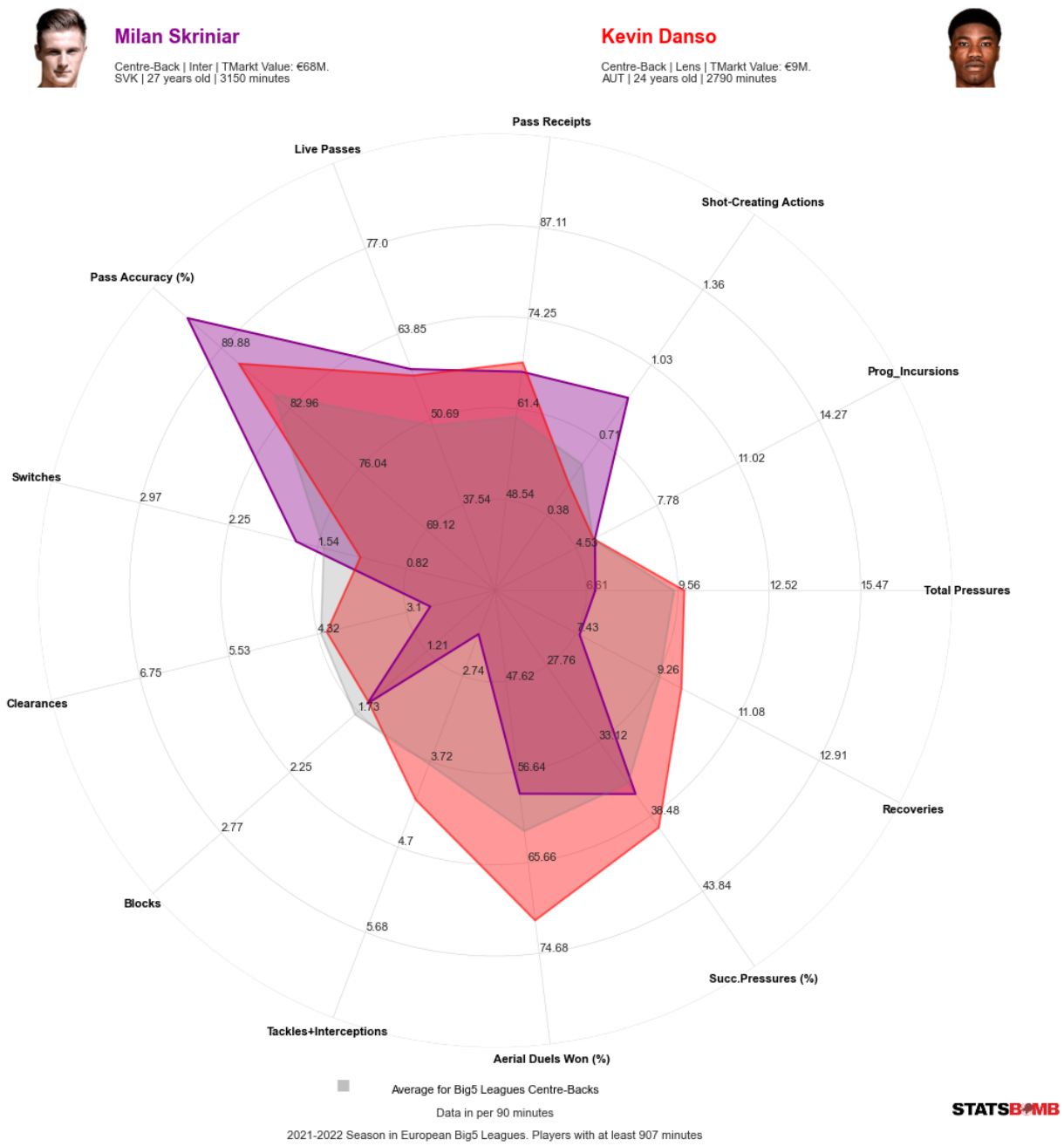
Observamos que el perfil que nos devuelve el modelo es consecuente con lo exigido, pues nos retorna futbolistas acostumbrados a jugar en defensa de tres -Kevin Danso, Davinson Sanchez, Mathijs de Ligt, Luca Ceppitelli o Max Kilman- o que han jugado en defensa de cuatro pero no como centrales (Araujo ha sido lateral en el Barcelona, Militao en Brasil y en el Porto, Ibáñez pivote, Soyuncu central en línea de 3 y White pivote en el Leeds).

In [45]:
```
for i,j in tm.iterrows():
    print(j['Player'],'-',j['Squad'],'-',squad[squad.Squad==j['Squad']].stat.values[
```

```
Kevin Danso - Lens - 3 defensas
Axel Disasi - Monaco - 4 defensas
Waldemar Anton - Stuttgart - 3 defensas
Ronald Araujo - Barcelona - 4 defensas
Caglar Soyuncu - Leicester City - 4 defensas
Davinson Sanchez - Tottenham - 3 defensas
Mohamed Simakan - RB Leipzig - 3 defensas
Eder Militao - Real Madrid - 4 defensas
Odilon Kossonou - Leverkusen - 4 defensas
Max Kilman - Wolves - 3 defensas
Alessio Romagnoli - Milan - 4 defensas
Ben White - Arsenal - 4 defensas
Kevin Bonifazi - Bologna - 3 defensas
Roger Ibanez - Roma - 4 defensas
Anel Ahmedhodzic - Bordeaux - 3 defensas
```

Vemos que, de hecho, un sorprendente Kevin Danso -de un valor más que asumible por un club de las dimensiones del Inter-, primera opción según el modelo, aguanta de forma más que decente la comparación con Milan Skriniar -68 millones de euros de valor-, el líder de la zaga del conjunto lombardo

In [89]:
```
pp.radar_comp('Milan Skriniar','Kevin Danso')
```



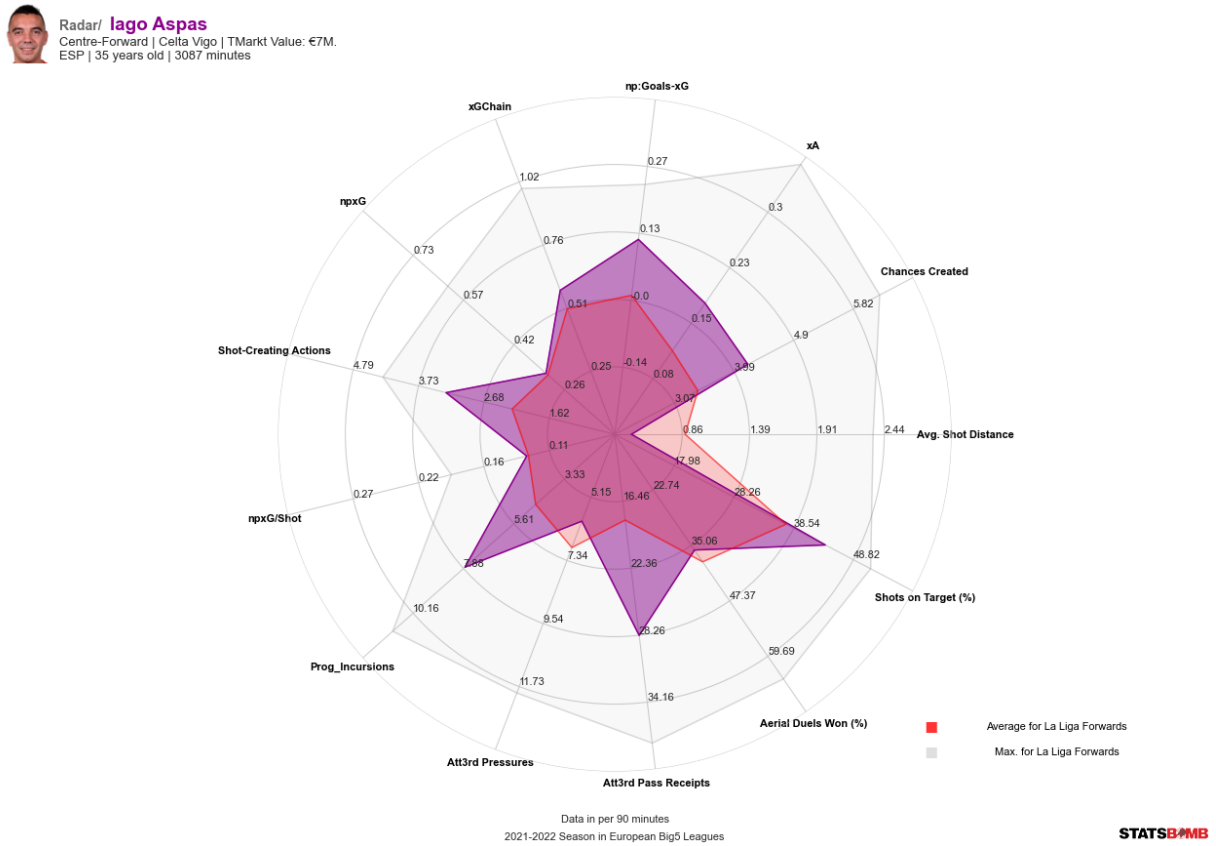
Casos

A continuación, y a modo de conclusión de la materia del trabajo, expondremos al detalle dos casos reales, en los que, a partir de una necesidad de reclutamiento actual surgida en un club, utilizaremos el modelo para tratar de resolver -o acotar- la problemática inicial y ofrecer una propuesta filtrada a la dirección deportiva.

1 - Un socio para Iago Aspas

Iago Aspas acumula casi una década comandando al Celta de Vigo desde el frente de ataque. Descontando los años de brillantez de Nolito y Fabián Orellana, ya lejanos, el delantero de Moaña apenas ha contado con colaboración cercana al nivel excelso que lleva demostrando en este tiempo (máximo goleador español del campeonato de Liga en cuatro ocasiones). Ya con 35 años, se observa que tanto su equipo como él necesitan a su lado un delantero que pueda combinar una notable capacidad anotadora con la activación, a nivel de presión, que exige a su equipo el entrenador argentino Eduardo Coudet.

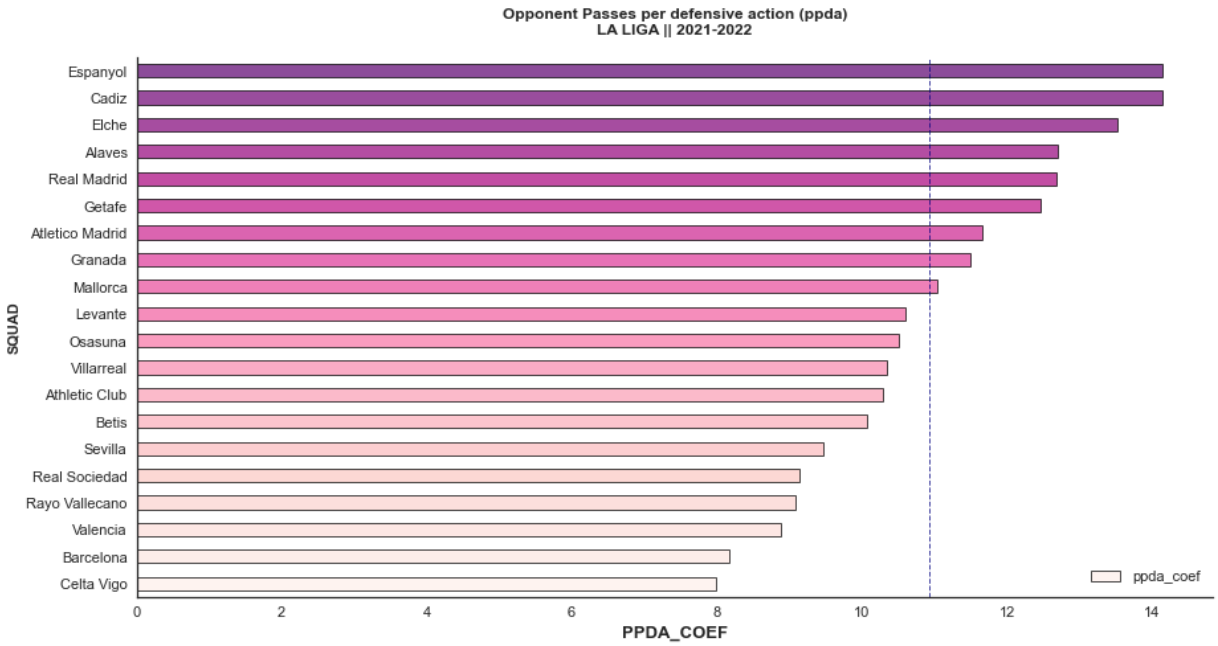
In [47]:
```
pp.sradar('Iago Aspas')
```



La tarea será, por tanto, ofrecerle a la dirección deportiva una lista corta de posibles candidatos a reforzar esta vacante. Se busca un futbolista en una edad deportiva interesante y asequible (recordemos las limitaciones económicas del Celta) que pueda adecuarse a jugar como referencia en ataque -se entiende que, a nivel deportivo, Iago Aspas se siente más cómodo en un papel generador y no tanto en la finalización- y que esté acostumbrado a mantener un alto nivel de esfuerzo defensivo que pueda liberar al moañés y provocar que el Celta mantenga los estándares de presión (ver gráfico a continuación) que caracterizan al equipo gallego desde la llegada de Coudet.

Abajo se muestran dos visualizaciones que provocan la necesidad del club de buscar un perfil trabajador en la punta. El Celta es el equipo que menos pases "cómodos" permite del campeonato español y, además, el que actúa con más agresividad en cada zona.

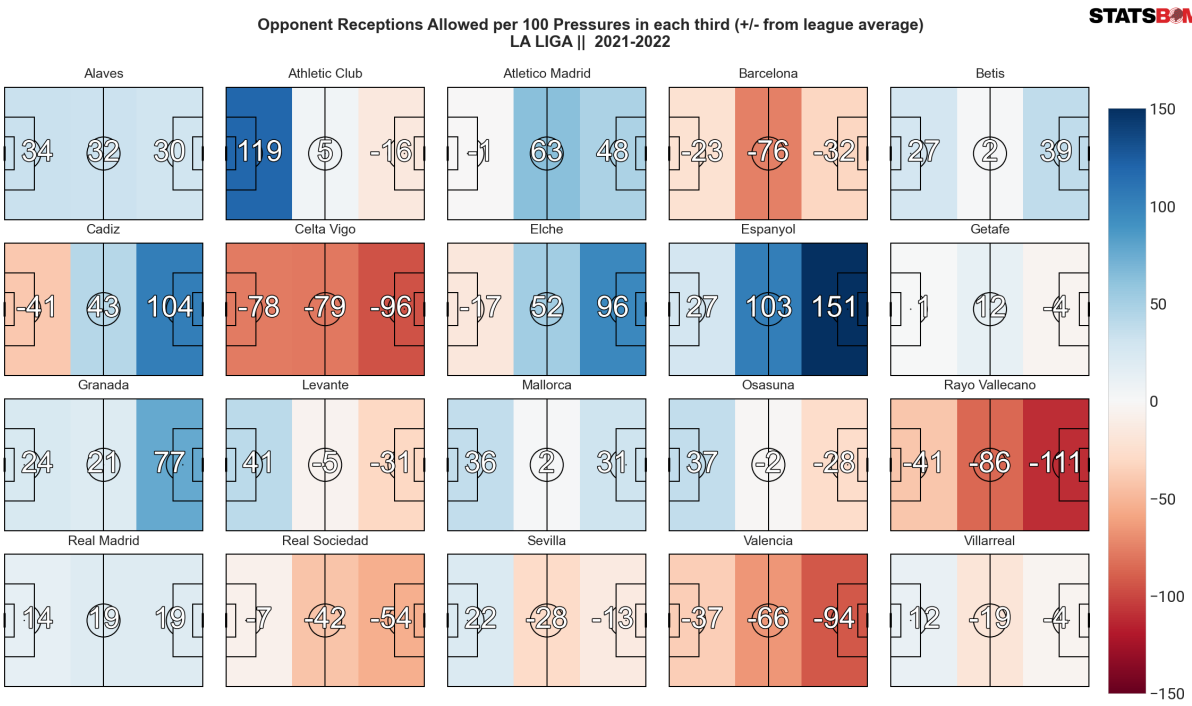
In [48]:
```
sp.barh(col='ppda_coef',metric='Opponent Passes per defensive action (ppda)',cmap='R
```



In [49]:
```
from Squad_plotting import pressing,dif_tpda,liga,bundesliga,ligue1,premier,seriea,p
```

In [50]:
```
sp.PPDA_third(data=pressing,cols=dif_tpda,league='La Liga',
                            metric='Opponent Receptions Allowed per 100 Pressures in e
                            minmax=(-150,150),name='TPDA_')
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```



In [51]:

```
tm = team_mapping('Celta Vigo','Centre-Forward')
tm = tm[(tm.value<=10) & (tm.Age<=30)]
tm = tm[(tm['Team_Similarity_Index']<np.percentile(tm['Team_Similarity_Index'],25))]
tm
```

Out[51]:

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index
1	Lucas Holer	ca618d23	Freiburg	28.0	Right	6.0	0.19	0.495
7	Moanes Dabbur	eac044ad	Hoffenheim	30.0	Right	6.0	0.27	0.591
8	Luis Javier Suarez	eeba07f8	Granada	25.0	Right	10.0	0.05	0.591
9	Caleb Ekuban	7afbd2c3	Genoa	28.0	Right	3.0	0.04	0.619
16	Branimir Hrgota	710fbdd1	Greuther Furth	29.0	Left	2.0	0.04	0.685
21	Florian Kruger	f9f92b5f	Arminia	23.0	Unknown	2.0	0.02	0.723
24	Rey Manaj	7703afb6	Spezia	25.0	Both	1.0	0.04	0.738
25	Borja Mayoral	64e8ed6d	Getafe	25.0	Right	9.0	0.01	0.738
26	Keita Balde	509a4ccb	Cagliari	27.0	Both	5.0	0.01	0.753
27	Ishak Belfodil	ff182fdc	Hertha BSC	30.0	Right	1.0	0.08	0.755
30	Kevin Lasagna	09538fdb	Hellas Verona	30.0	Left	3.0	0.02	0.764
32	Anthony Lozano	38699c72	Cadiz	29.0	Right	4.0	0.03	0.774
33	El Bilal Toure	47c609d5	Reims	21.0	Both	6.0	0.03	0.779
35	Vedat Muriqi	0cadc46d	Mallorca	28.0	Left	6.0	0.04	0.801

Eliminamos a todos los jugadores resultantes de la primera selección que estén en el mismo cluster que Iago Aspas, pues, al pretender formar una dupla complementaria con el moañés, nos

interesa descartar a todos aquellos delanteros que no sean "nueves" puros

In [52]:
```
dfc = df['Forward']
clu = dfc[dfc.Player=='Iago Aspas']['cluster'].values[0]
print('El cluster de DELANTEROS en el que se encuentra IAGO ASPAS es: ',clu)
tm = pd.merge(tm,dfc[['Player','cluster']],how='left',on='Player')
tm = tm[tm['cluster']!=clu]
tm
```

```
El cluster de DELANTEROS en el que se encuentra IAGO ASPAS es:  2
```

Out[52]:

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index	cluster
0	Lucas Holer	ca618d23	Freiburg	28.0	Right	6.0	0.19	0.495	3
2	Luis Javier Suarez	eeba07f8	Granada	25.0	Right	10.0	0.05	0.591	3
3	Caleb Ekuban	7afbd2c3	Genoa	28.0	Right	3.0	0.04	0.619	3
5	Florian Kruger	f9f92b5f	Arminia	23.0	Unknown	2.0	0.02	0.723	3
6	Rey Manaj	7703afb6	Spezia	25.0	Both	1.0	0.04	0.738	3
7	Borja Mayoral	64e8ed6d	Getafe	25.0	Right	9.0	0.01	0.738	3
8	Keita Balde	509a4ccb	Cagliari	27.0	Both	5.0	0.01	0.753	3
9	Ishak Belfodil	ff182fdc	Hertha BSC	30.0	Right	1.0	0.08	0.755	3
10	Kevin Lasagna	09538fdb	Hellas Verona	30.0	Left	3.0	0.02	0.764	1
11	Anthony Lozano	38699c72	Cadiz	29.0	Right	4.0	0.03	0.774	3
12	El Bilal Toure	47c609d5	Reims	21.0	Both	6.0	0.03	0.779	3
13	Vedat Muriqi	0cadc46d	Mallorca	28.0	Left	6.0	0.04	0.801	3

Incorporamos al análisis dos métricas relevantes de presión -**%ag_Press** hace referencia al número de presiones por cada 100 recepciones del rival, y **PPT3%** corresponde a las presiones realizadas por cada 100 recepciones rivales sólo en el tercio propio oponente-, de cara a poder cerciorarnos de que los futbolistas seleccionados se adapten a las exigencias de actividad planteadas. Además, añadimos la estadística de xGs sin penaltis y el acierto (Goles - xG), para poder medir el nivel de los candidatos.

No son preocupantes los dígitos negativos en la métrica de acierto respecto a calidad de las oportunidades (un -0.15 indica que, por cada siete partidos, un jugador deja de marcar un gol que sí debería de haber anotado). El 0.23 que muestra Borja Mayoral indica, sin embargo, que es posible que el jugador no sea capaz de sostener la racha de acierto que ha registrado esta temporada.

In [53]:
```
sel = pd.merge(tm,players[['idx','Expected_npxG','Expected_np:G-xG','%ag_Press','PPT
sel['%ag_Press'] =  round(sel['%ag_Press'] / players[(players.Min>=min_minutes) & (p
sel['PPT3%'] =  round(sel['PPT3%'] / players[(players.Min>=min_minutes) & (players['
sel
```

Out[53]:

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index	cluster	Exp
0	Lucas Holer	ca618d23	Freiburg	28.0	Right	6.0	0.19	0.495	3	
1	Luis Javier Suarez	eeba07f8	Granada	25.0	Right	10.0	0.05	0.591	3	
2	Caleb Ekuban	7afbd2c3	Genoa	28.0	Right	3.0	0.04	0.619	3	
3	Florian Kruger	f9f92b5f	Arminia	23.0	Unknown	2.0	0.02	0.723	3	
4	Rey Manaj	7703afb6	Spezia	25.0	Both	1.0	0.04	0.738	3	
5	Borja Mayoral	64e8ed6d	Getafe	25.0	Right	9.0	0.01	0.738	3	
6	Keita Balde	509a4ccb	Cagliari	27.0	Both	5.0	0.01	0.753	3	
7	Ishak Belfodil	ff182fdc	Hertha BSC	30.0	Right	1.0	0.08	0.755	3	
8	Kevin Lasagna	09538fdb	Hellas Verona	30.0	Left	3.0	0.02	0.764	1	
9	Anthony Lozano	38699c72	Cadiz	29.0	Right	4.0	0.03	0.774	3	
10	El Bilal Toure	47c609d5	Reims	21.0	Both	6.0	0.03	0.779	3	
11	Vedat Muriqi	0cadc46d	Mallorca	28.0	Left	6.0	0.04	0.801	3	

In [54]:
```
print('Media npxG en Delanteros: ',players[(players.Min>=min_minutes) & (players['Po
```

```
Media npxG en Delanteros:  0.3430769230769231
```

Podemos ver que el modelo es consecuente, pues nos ha devuelto, esencialmente y con las excepciones del Choco Lozano e Ishak Belfodil, a jugadores acostumbrados a ejercicios de presión. Para obtener la selección final, eliminaremos a aquellos jugadores que se encuentren por debajo de la media en alguna de las dos métricas de presiones. Ello nos deja una selección de cinco futbolistas con números decentes en xG y en actividades defensivas, además de asequibles y con perfil de nueve puro. Salvo Lucas Holer, ninguno de ellos se caracteriza por su participación en el juego, pero todos ellos son rematadores. Kevin Lasagna pertenece al cluster 1, por lo que su perfil de nueve se acentúa todavía más -mucha más presencia en el área-.

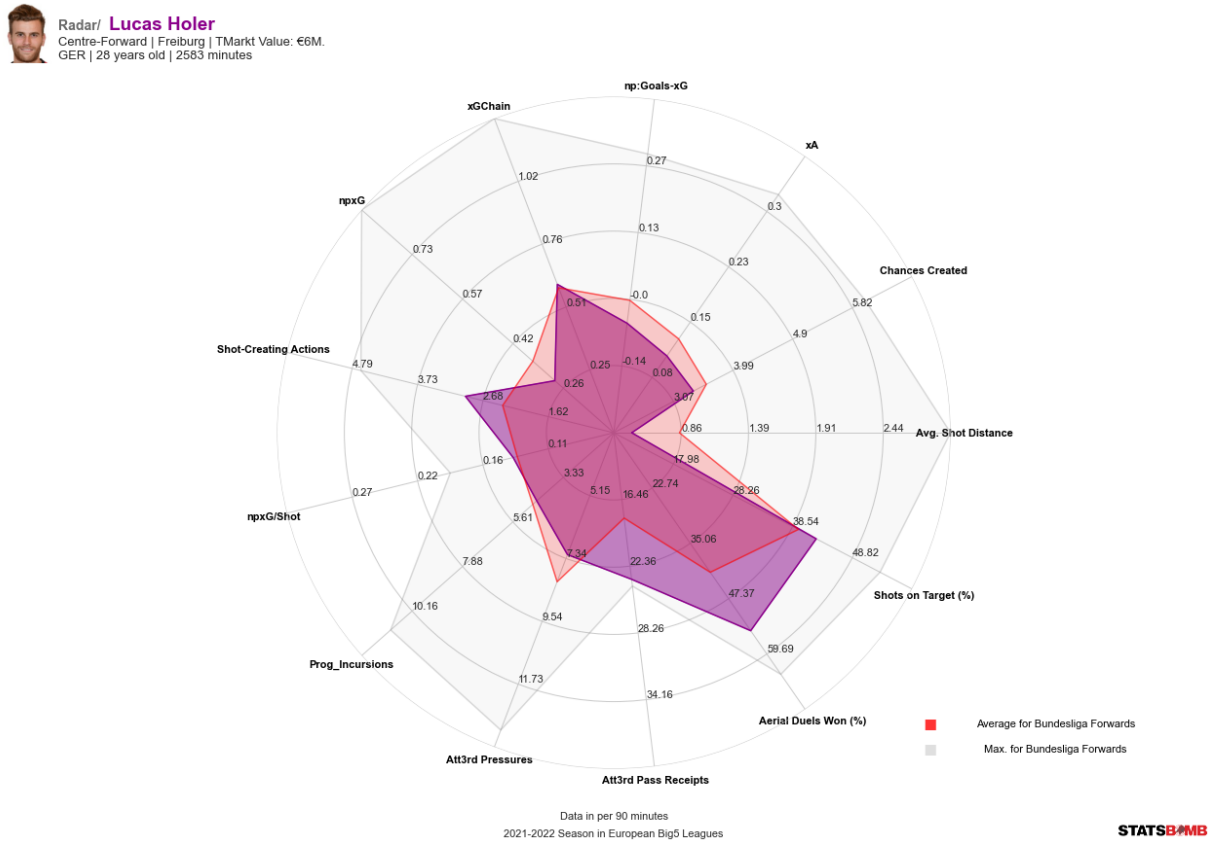
In [55]:
```
sel = sel[((sel['PPT3%']>1)|(sel['%ag_Press']>1)) & (sel['Expected_npxG']>=0.2)]
sel
```

Out[55]:

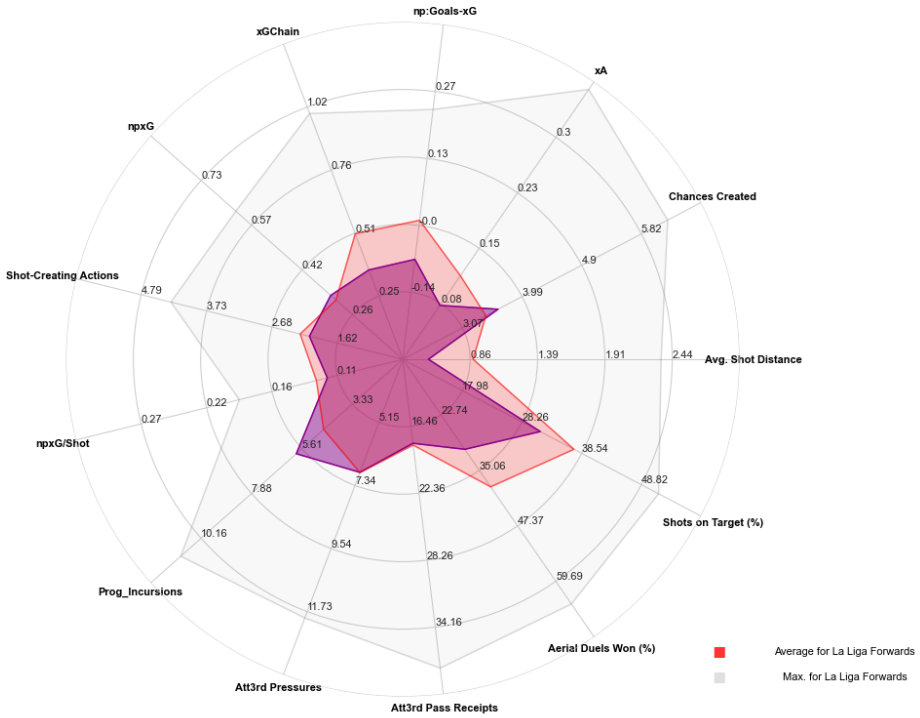
	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index	cluster	Expected_
0	Lucas Holer	ca618d23	Freiburg	28.0	Right	6.0	0.19	0.495	3	
1	Luis Javier Suarez	eeba07f8	Granada	25.0	Right	10.0	0.05	0.591	3	
4	Rey Manaj	7703afb6	Spezia	25.0	Both	1.0	0.04	0.738	3	
6	Keita Balde	509a4ccb	Cagliari	27.0	Both	5.0	0.01	0.753	3	
8	Kevin Lasagna	09538fdb	Hellas Verona	30.0	Left	3.0	0.02	0.764	1	

In [56]:

```
for i in list(sel.Player.head(3)):
    pp.sradar(i)
```



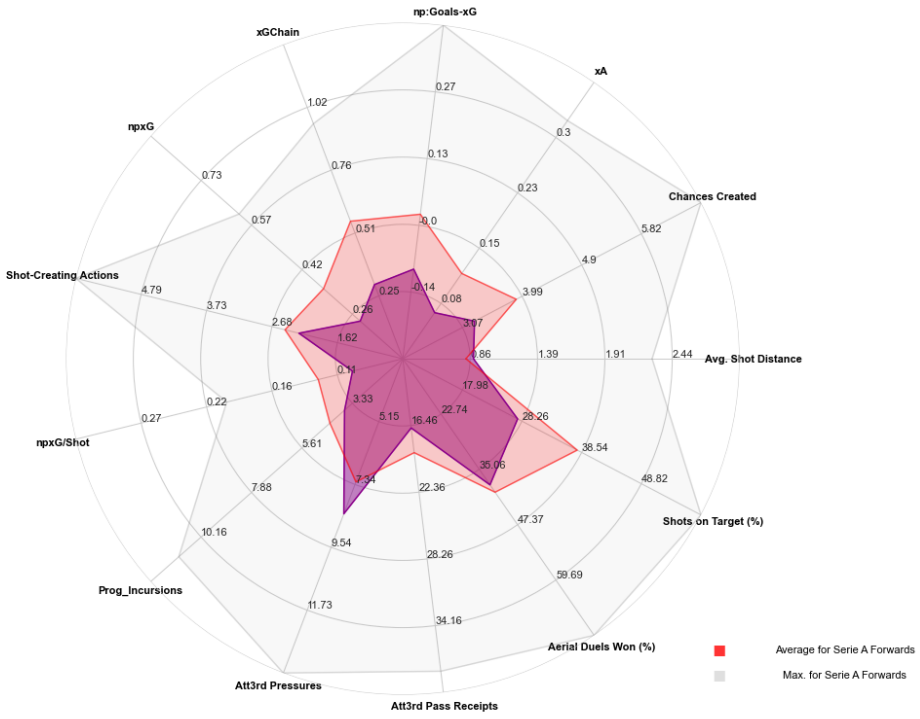
Radar/ **Luis Javier Suarez**
Centre-Forward | Granada | TMarkt Value: €10M.
COL | 25 years old | 2782 minutes



Data in per 90 minutes
2021-2022 Season in European Big5 Leagues

STATSBOMB

Radar/ **Rey Manaj**
Centre-Forward | Spezia | TMarkt Value: €1M.
ALB | 25 years old | 1768 minutes



Data in per 90 minutes
2021-2022 Season in European Big5 Leagues

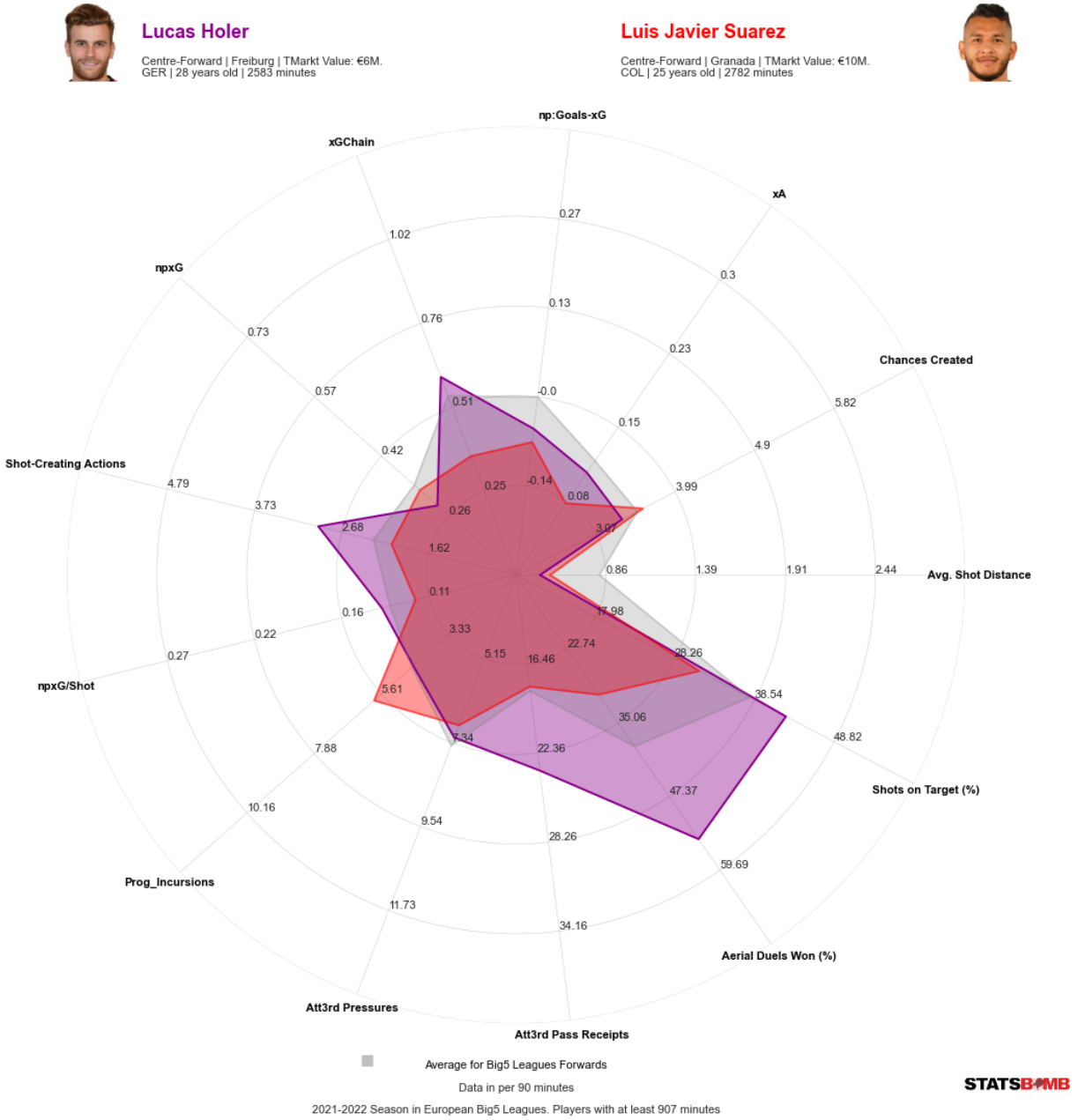
STATSBOMB

Nuestro análisis ha dejado fuera a la que parece ser, de acuerdo a los rumores, la primera opción para la dirección deportiva de Luis Campos, el madrileño Borja Mayoral. Sin embargo, otorga un nombre, como el de Keita Balde, al que el mercado puede estar infravalorando. El delantero de origen senegalés puede aportar desmarques a la altura de la calidad de los pases de Iago Aspas. Igualmente, los más interesantes parecen los dos primeros de la lista, Lucas Holer y Luis Suárez -que pertenece a un descendido Granada-. El colombiano tiene caché en España, conoce el campeonato, es agresivo y se siente cómodo finalizando y conduciendo, mientras que

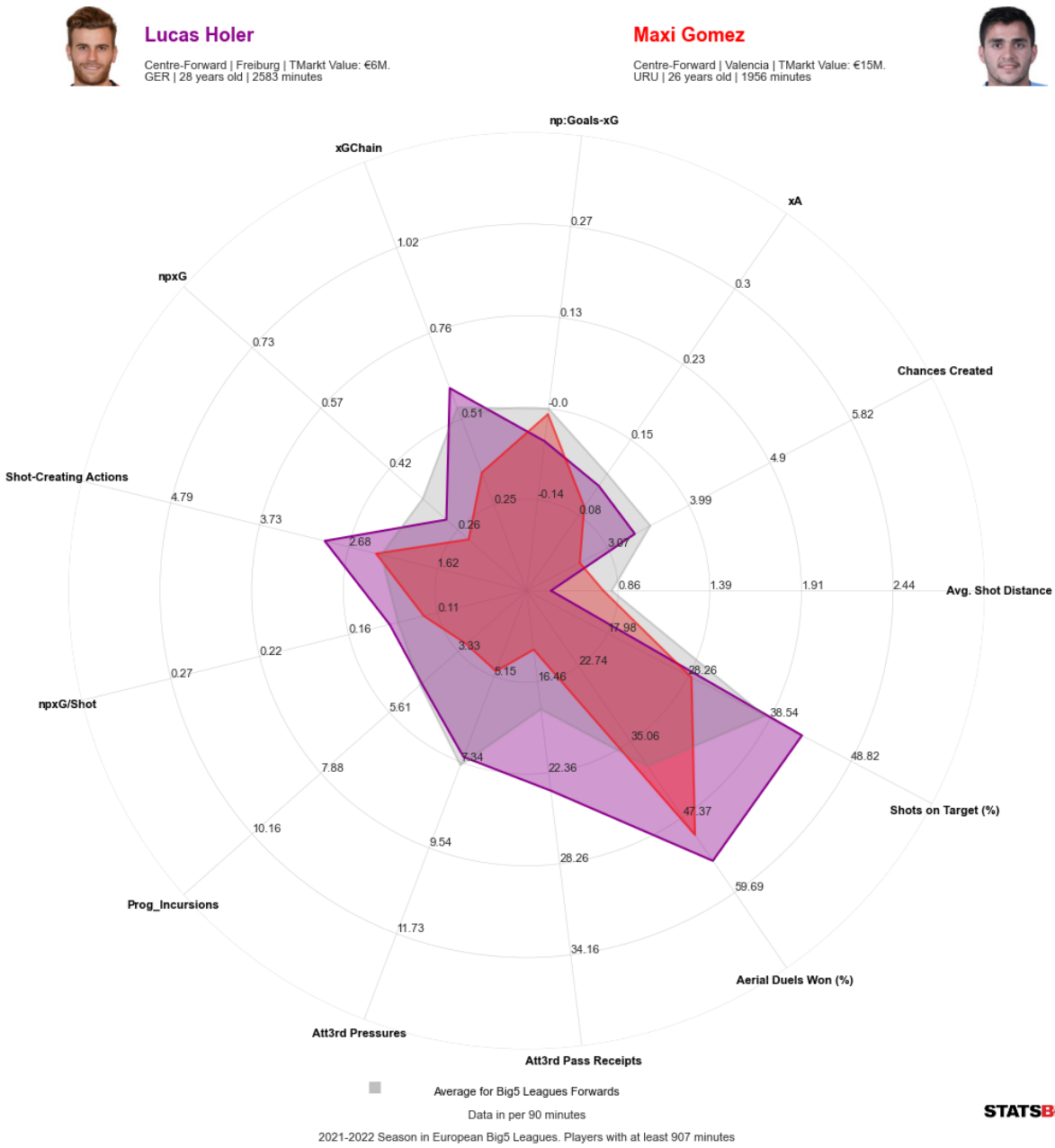
el alemán, que es mucho más participativo y productivo en otras lides distintas a la finalización, agrega un componente del que no dispone el Celta: el juego aéreo.

De hecho, como vemos más abajo, el alemán muestra similitudes con los dos últimos acompañantes de lago Aspas: el uruguayo Maxi Gómez y el gallego Santi Mina

In [85]:
```
pp.radar_comp('Lucas Holer','Luis Javier Suarez')
```



In [88]:
```
pp.radar_comp('Lucas Holer','Maxi Gomez')
```



Éxodo de Centrales

En el presente verano, el Chelsea afronta un doble desafío: debe rehacer un equipo campeón el pleno cambio de propiedad mientras afronta la salida en masa de gran parte de la defensa protagonista de la conquista de la Champions League por parte de los blues en 2021. Con la marcha de Antonio Rudiger y Andreas Christensen, ambos sin dejar ni un euro en las arcas del club, el conjunto londinense -que juega con defensa de tres- se queda tan sólo con tres centrales para este curso. Además, los veteranos Thiago Silva y César Azpilicueta abandonarán el club este verano o el próximo.

Con Malang Sarr y Trevor Chalobah como centrales jóvenes y físicos, el objetivo del club debe ser encontrar en el mercado, al menos, un central agresivo, presionante y veloz como Rudiger y otro con habilidad con los pies, buen posicionamiento y alto acierto en pases progresivos, como Christensen.

In [59]:

```
tm = team_mapping('Chelsea','Centre-Back')
tm = tm[(tm.Power>=tm.Power.mean()) & (tm['Team_Similarity_Index']<np.percentile(tm[
tm = tm[(tm.Age<30)]
tm.shape[0]
```

Out[59]:
```
45
```

In [60]:
```
dfc = df['Centre-Back']
clu = dfc[dfc.Player=='Antonio Rudiger']['cluster'].values[0]
print('El cluster de CENTRALES en el que se encuentra ANTONIO RUDIGER es: ',clu)
tm = pd.merge(tm,dfc[['Player','cluster']],how='left',on='Player')
tm = tm[tm['cluster']==clu]
tm
```

```
El cluster de CENTRALES en el que se encuentra ANTONIO RUDIGER es:  2
```

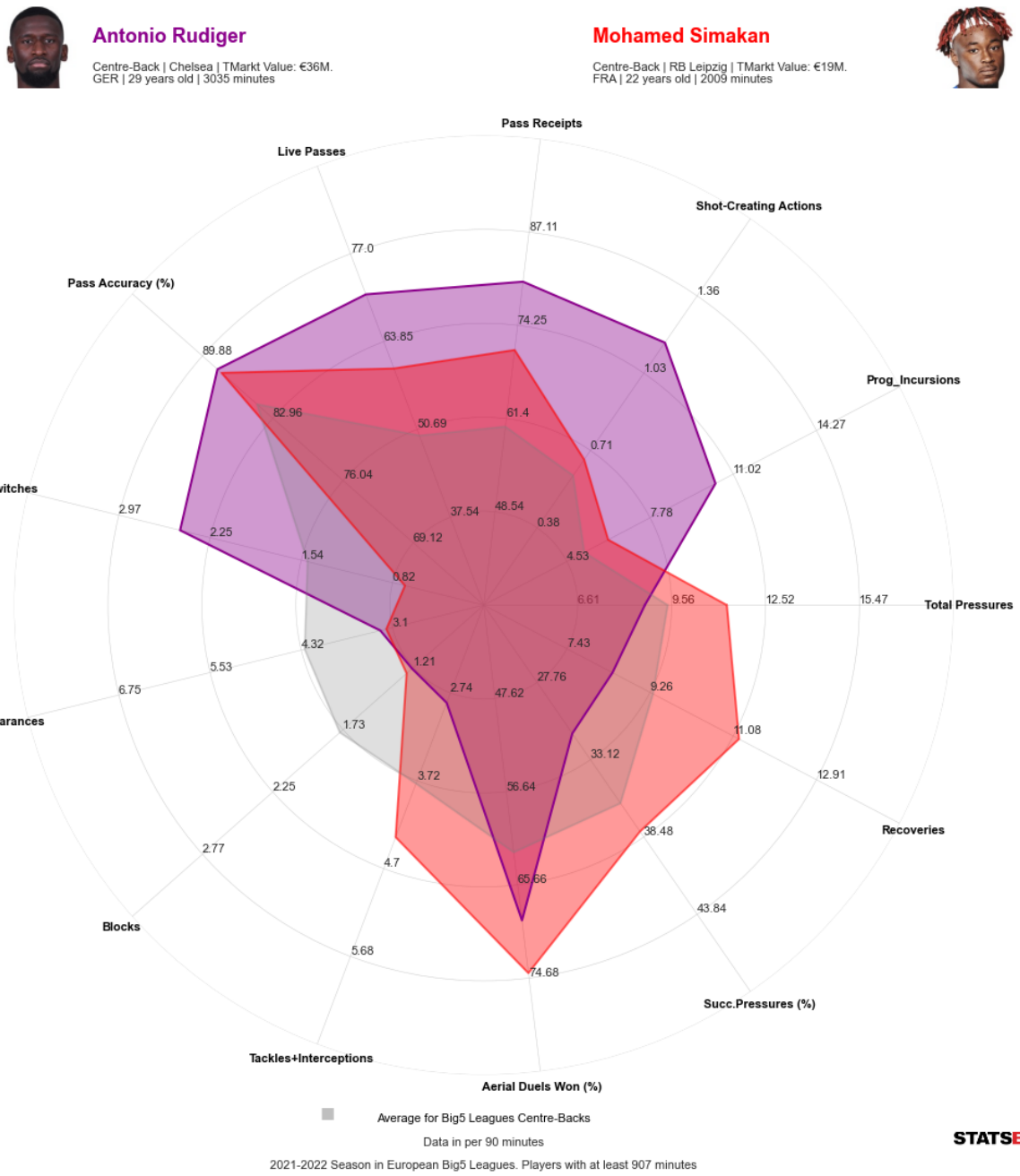
Out[60]:

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index	cluster
0	Mohamed Simakan	fcb27134	RB Leipzig	22.0	Right	19.0	0.31	0.489	2
1	Ben White	35e413f1	Arsenal	25.0	Right	42.0	0.39	0.584	2
2	Waldemar Anton	5e66fa06	Stuttgart	26.0	Right	6.0	0.42	0.621	2
3	Alessio Romagnoli	3ead85f9	Milan	27.0	Left	21.0	0.31	0.621	2
7	Hiroki Ito	204dded0	Stuttgart	23.0	Left	3.0	0.35	0.697	2
8	Davinson Sanchez	da7b447d	Tottenham	26.0	Right	31.0	0.57	0.706	2
9	Axel Disasi	ad82197c	Monaco	24.0	Right	17.0	0.36	0.707	2
10	Jean-Clair Todibo	88f130ed	Nice	23.0	Right	21.0	0.45	0.718	2
11	Caglar Soyuncu	21166ff4	Leicester City	26.0	Right	47.0	0.28	0.746	2
14	Aritz Elustondo	f7dc2ae5	Real Sociedad	28.0	Right	21.0	0.31	0.763	2
15	Matthias Ginter	f521b80a	M'Gladbach	28.0	Right	25.0	0.36	0.791	2
16	Roger Ibanez	82efe6fa	Roma	24.0	Right	26.0	0.34	0.792	2
17	Eder Militao	2784f898	Real Madrid	24.0	Right	63.0	0.54	0.807	2
18	Jules Kounde	4d1666ff	Sevilla	24.0	Right	63.0	0.42	0.807	2
19	Kevin Danso	6e33125f	Lens	24.0	Right	9.0	0.46	0.821	2
21	Adam Webster	c40b6180	Brighton	27.0	Right	19.0	0.27	0.841	2
22	John Brooks	e27da244	Wolfsburg	29.0	Left	12.0	0.31	0.856	2
23	Nikola Milenkovic	bee704fc	Fiorentina	25.0	Right	24.0	0.34	0.859	2
24	Alessandro Bastoni	75b86fb3	Inter	23.0	Left	63.0	0.66	0.860	2

	Player	idx	Squad	Age	foot	value	Power	Team_Similarity_Index	cluster
25	Castello Lukeba	5b9512c5	Lyon	20.0	Left	21.0	0.54	0.869	2
26	Jonathan Tah	bd142efb	Leverkusen	26.0	Right	26.0	0.53	0.883	2
27	Igor	a4e85758	Fiorentina	24.0	Left	8.0	0.38	0.899	2
28	Edmond Tapsoba	a73c8f95	Leverkusen	23.0	Right	42.0	0.66	0.903	2
29	Ronald Araujo	2bef2bca	Barcelona	23.0	Right	42.0	0.46	0.904	2
32	Benoit Badiashile	06df8256	Monaco	21.0	Left	31.0	0.46	0.931	2
33	Mario Hermoso	555f3a0b	Atletico Madrid	27.0	Left	26.0	0.47	0.932	2
36	Eric Garcia	2bed3eab	Barcelona	21.0	Right	19.0	0.72	0.954	2
38	Victor Lindelof	f5deef4c	Manchester Utd	28.0	Right	25.0	0.35	0.959	2
39	Nico Elvedi	48035304	M'Gladbach	26.0	Right	24.0	0.31	0.960	2
40	Pau Torres	532e1e4f	Villarreal	25.0	Left	52.0	0.51	0.973	2
41	Guillermo Maripan	d9764034	Monaco	28.0	Right	15.0	0.57	0.976	2
42	Manuel Akanji	89ac64a6	Dortmund	27.0	Right	31.0	0.55	0.989	2
44	Tiago Djalo	b2336681	Lille	22.0	Right	12.0	0.32	0.990	2

In [84]:

```
pp.radar_comp('Antonio Rudiger','Mohamed Simakan')
```



In [62]:
```
clu = dfc[dfc.Player=='Andreas Christensen']['cluster'].values[0]
print('El cluster de CENTRALES en el que se encuentra ANDREAS CHRISTENSEN es: ',clu)
```

```
El cluster de CENTRALES en el que se encuentra ANDREAS CHRISTENSEN es:  2
```

In [63]:
```
chelsea =[]
for i in list(tm.Player):
    chelsea.append(i)
```

In [64]:
```
ps1 = player_similarities('Antonio Rudiger')
ps1 = ps1[(ps1['Similarity_Score']<np.percentile(ps1['Similarity_Score'],10)) & (ps1
ps1
```

```
Looking for similar profiles to ANTONIO RUDIGER
```

Out[64]:

	Player	Similarity_Score
50	Pau Torres	0.807442
133	Matthias Ginter	0.891952

	Player	Similarity_Score
61	Jules Kounde	0.902362
195	Mario Hermoso	0.925316
138	Alessandro Bastoni	0.932416
144	Ronald Araujo	0.999259
229	Davinson Sanchez	1.042812
171	Eric Garcia	1.049553
257	Adam Webster	1.054808
27	Eder Militao	1.062610
44	Ben White	1.122223
259	Alessio Romagnoli	1.131243
35	Nikola Milenkovic	1.139413

In [77]:

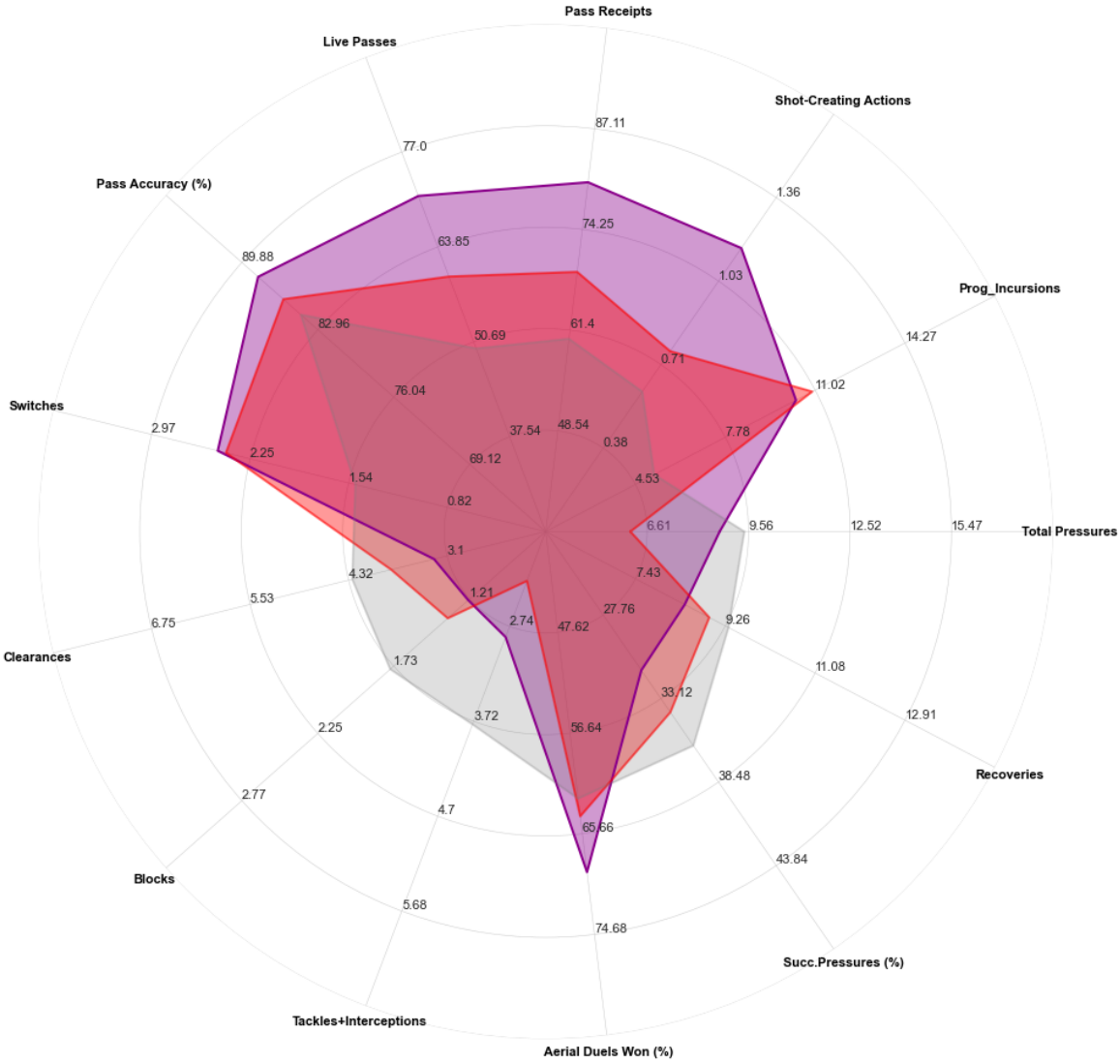
```
for i in list(ps1.Player.head()):
    pp.radar_comp('Antonio Rudiger',i)
    time.sleep(1)
```

Antonio Rudiger

Centre-Back | Chelsea | TMarkt Value: €36M.
GER | 29 years old | 3035 minutes

Pau Torres

Centre-Back | Villarreal | TMarkt Value: €52M.
ESP | 25 years old | 2855 minutes



Average for Big5 Leagues Centre-Backs

Data in per 90 minutes

2021-2022 Season in European Big5 Leagues. Players with at least 907 minutes

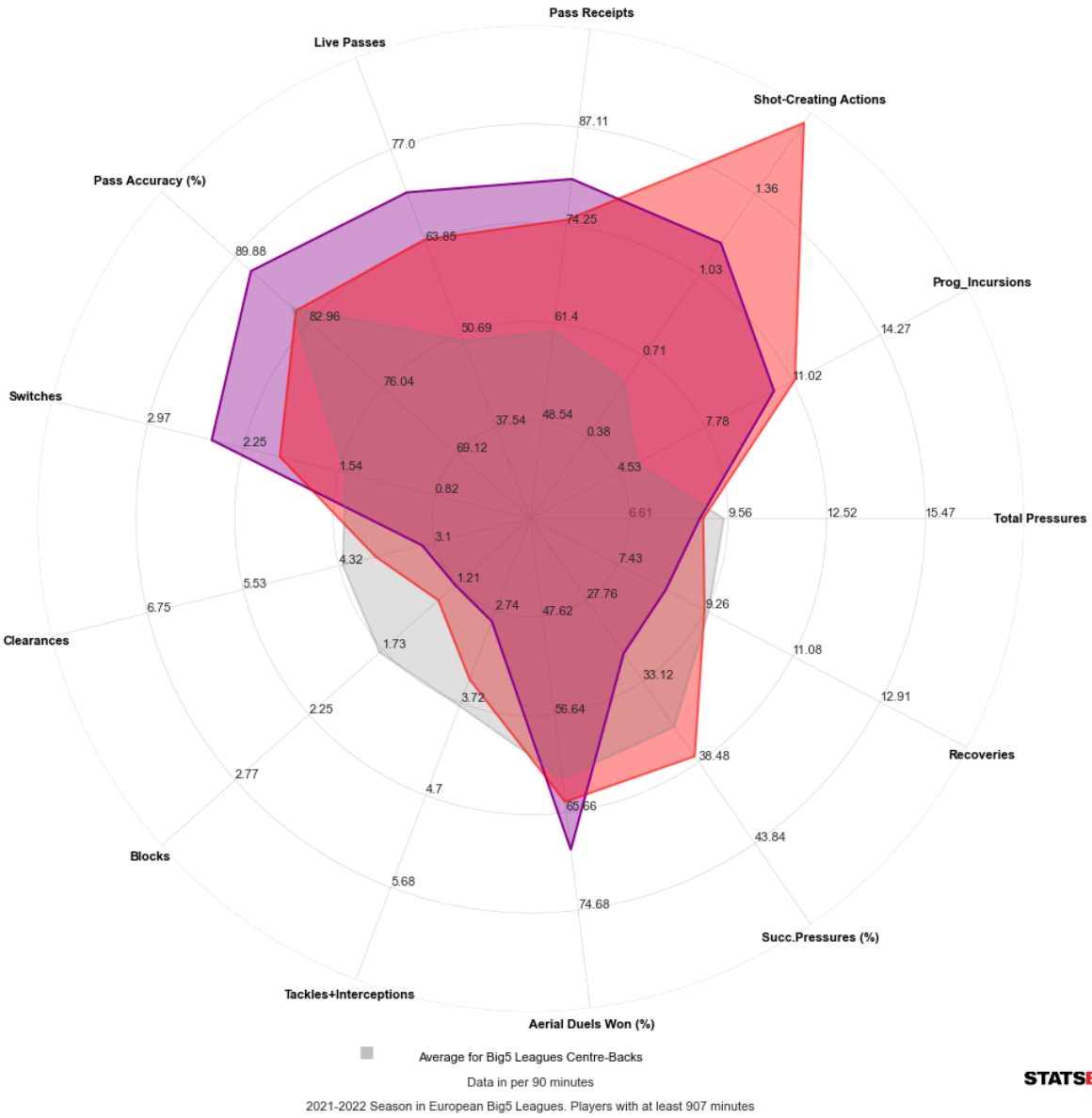
STATSBOMB

Antonio Rudiger

Centre-Back | Chelsea | TMarkt Value: €36M.
GER | 29 years old | 3035 minutes

Matthias Ginter

Centre-Back | M'Gladbach | TMarkt Value: €25M.
GER | 28 years old | 2355 minutes



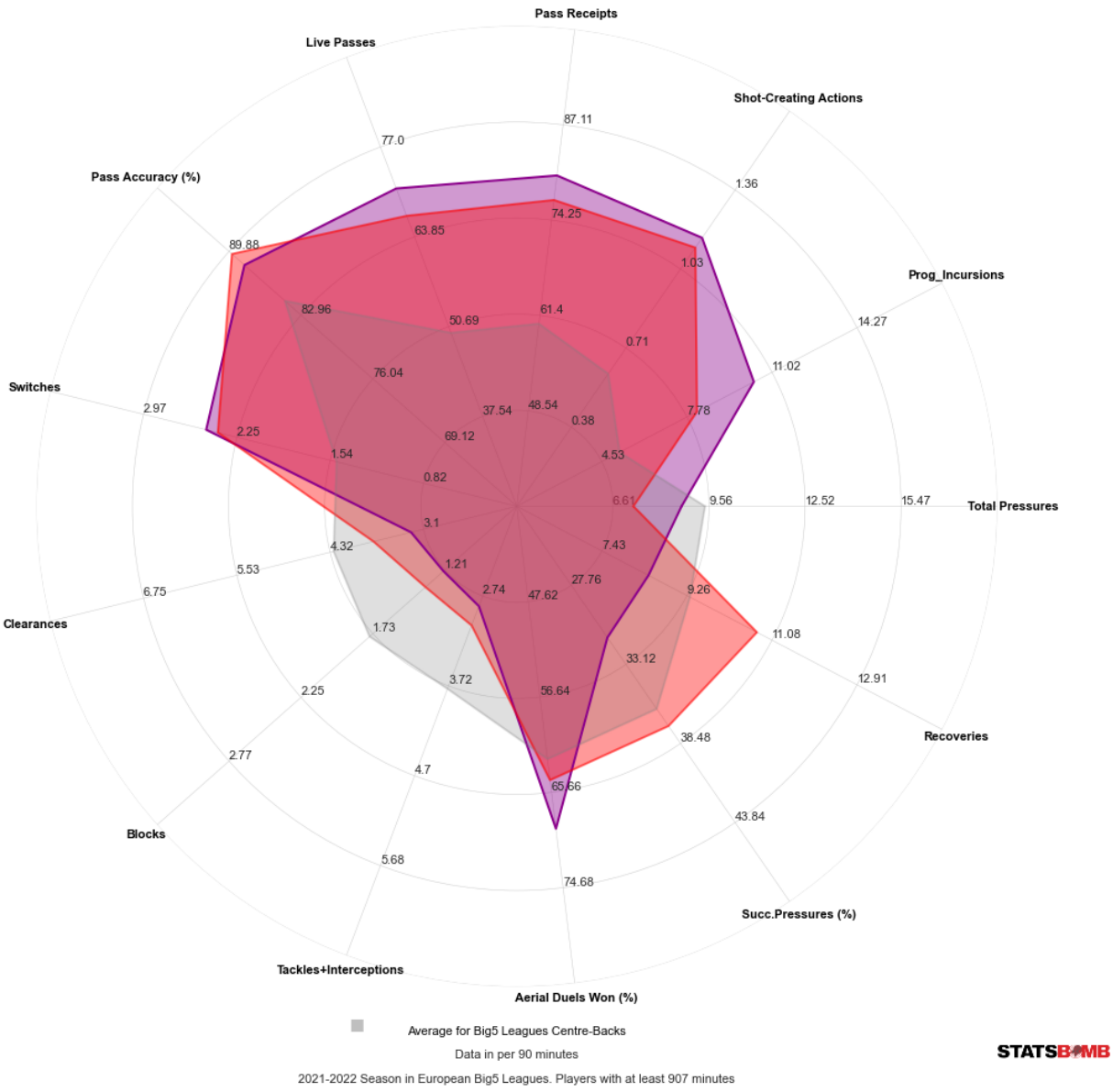
Average for Big5 Leagues Centre-Backs

Data in per 90 minutes

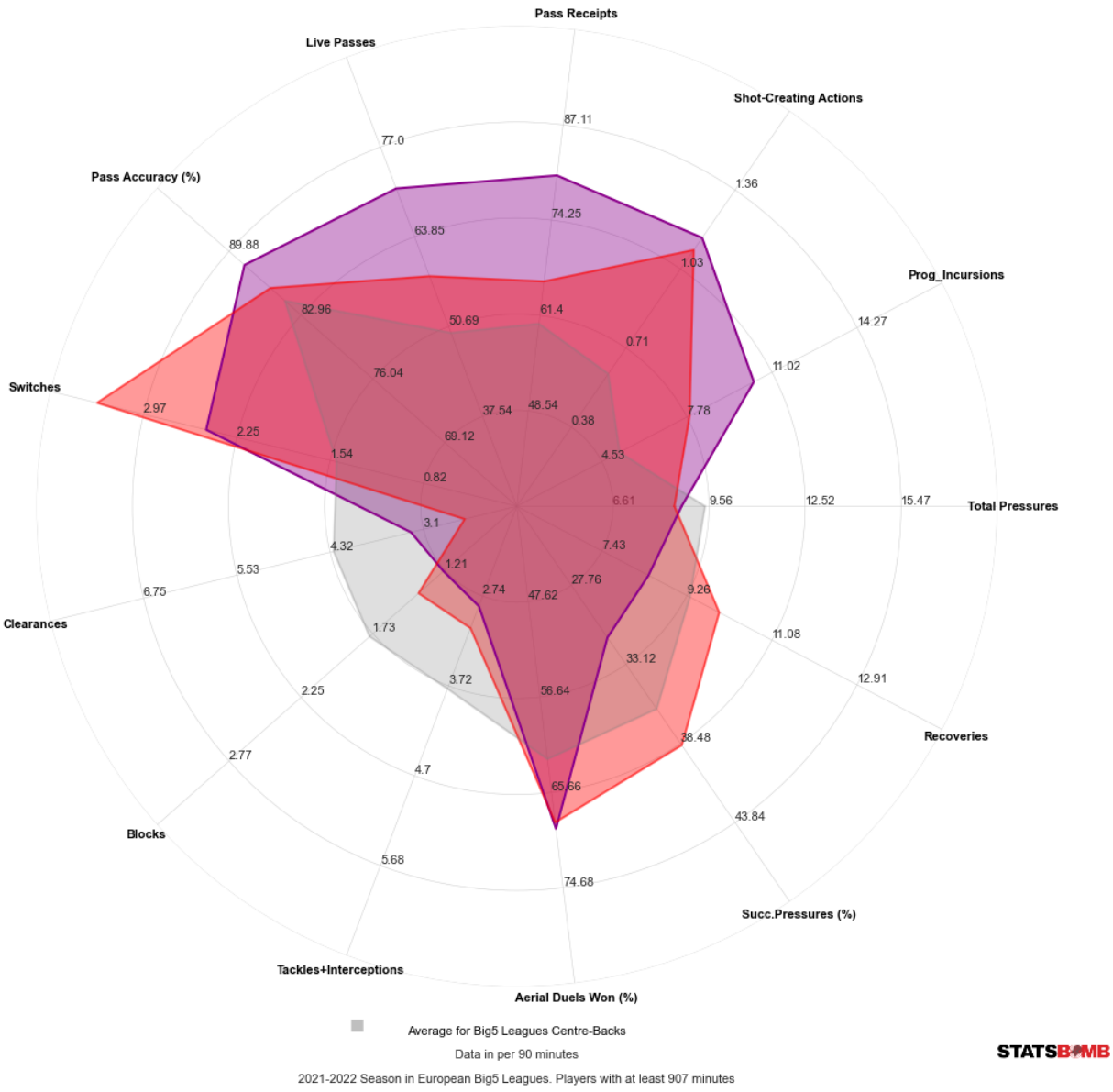
2021-2022 Season in European Big5 Leagues. Players with at least 907 minutes

STATSBOMB

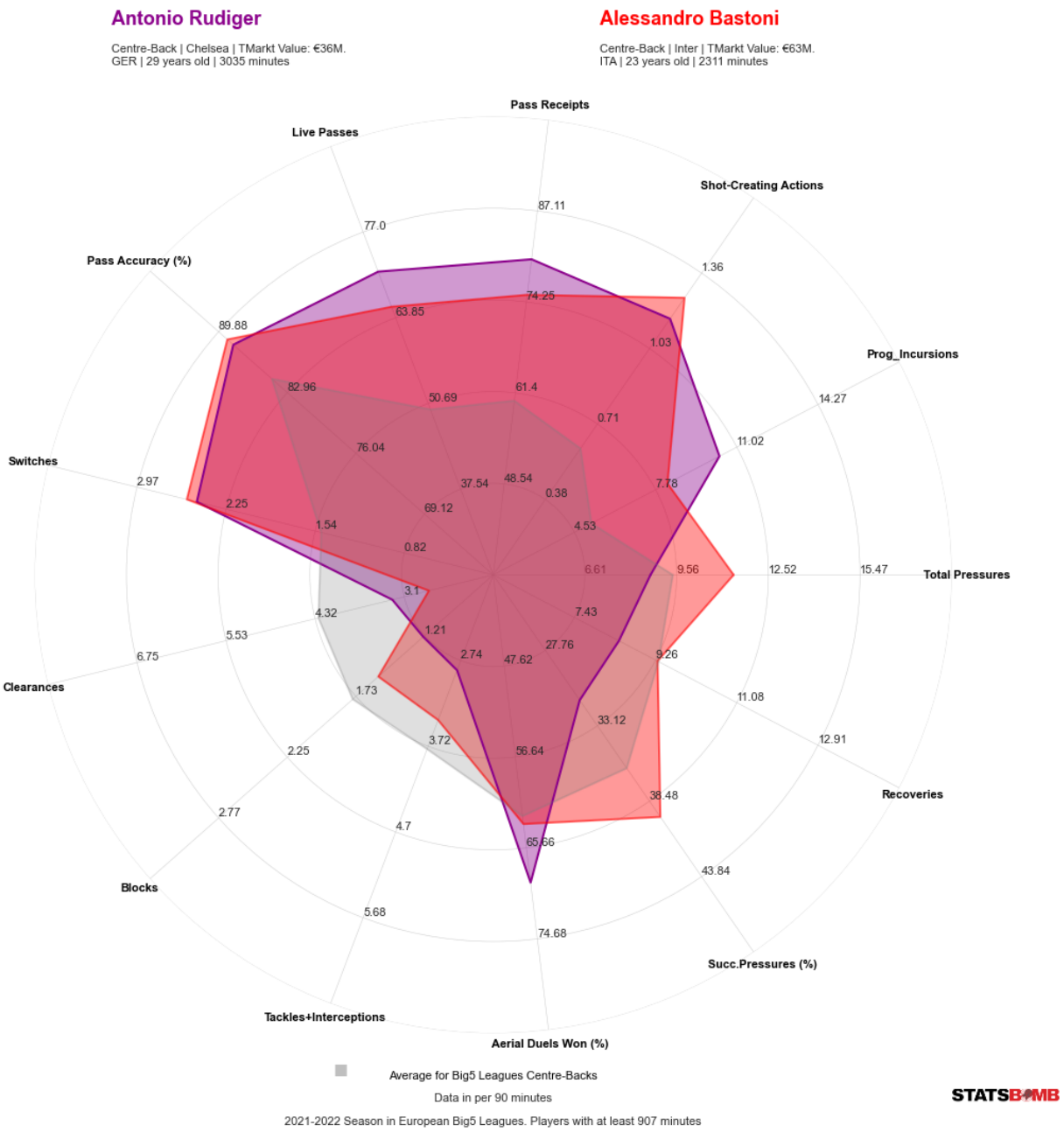
Could not load player photo 1
Could not load player photo 2



Could not load player photo 2

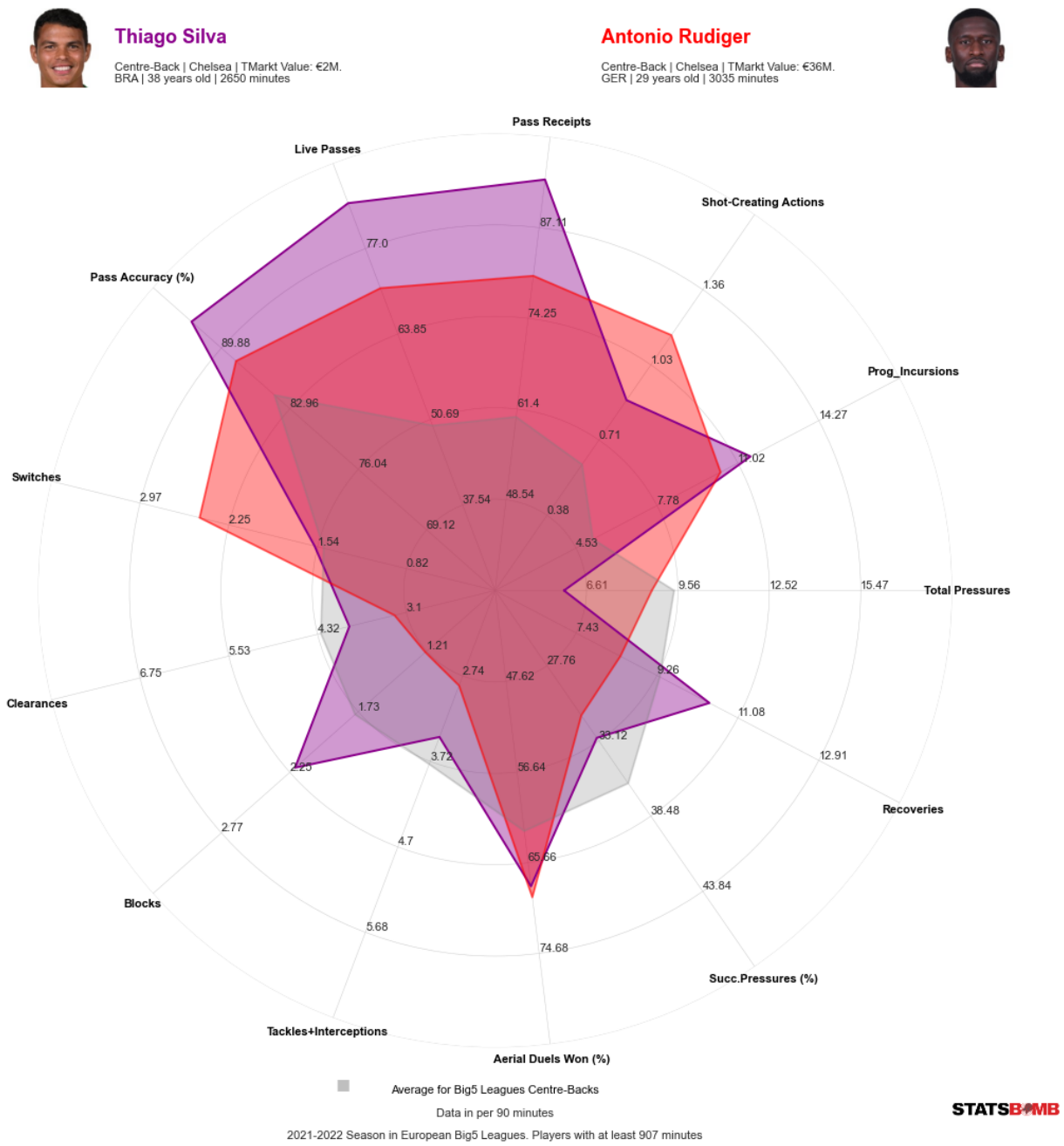


Could not load player photo 2



Se observa que centrales como Jules Koundé o Alessandro Bastoni se acercan mucho, no sólo al perfil, sino también al nivel de Toni Rudiger. Cualquiera de los dos sería una buena opción para la secretaría técnica del club londinense. Si se pretendiera específicamente un central zurdo, observamos que aparecen nombres como Pau Torres o Mario Hermoso -que sería una opción low cost-. Si bien a ambos les falta la agresividad y el liderazgo que caracteriza a Rudiger, debemos destacar las buenas actuaciones del central madrileño cuando ha tenido que jugar en defensa de tres en el Atlético de Madrid, y que al castellonense siempre se le ha visto cómodo con un central veterano -Albiol- al lado y ostenta una depuradísima salida de balón.

In [76]:
```
pp.radar_comp('Thiago Silva','Antonio Rudiger')
```



Además de tomar más responsabilidades en la construcción del juego, el hecho de ser el central de la base, en el caso de Thiago Silva, fomenta que el brasileño esté cerca de liderar las estadísticas de bloqueos e intercepciones, mientras que no se prodiga mucho en la anticipación -bajo nivel de presiones- y, en sus incursiones, sólo un pequeño porcentaje se debe a conducciones.

In [67]:

```
ps2 = player_similarities('Andreas Christensen')
ps3 = player_similarities('Thiago Silva')
ps2 = ps2[(ps2['Similarity_Score']<np.percentile(ps2['Similarity_Score'],10)) & (ps2
ps3 = ps3[(ps3['Similarity_Score']<np.percentile(ps3['Similarity_Score'],10)) & (ps3
ps2 = pd.concat([ps2,ps3])
ps2 = ps2.sort_values(by='Similarity_Score',ascending=True)
ps2 = ps2.drop_duplicates('Player',keep='first')
ps2 = ps2[ps2.Similarity_Score<1]
ps2
```

```
Looking for similar profiles to ANDREAS CHRISTENSEN
Looking for similar profiles to THIAGO SILVA
```

Out[67]:

	Player	Similarity_Score

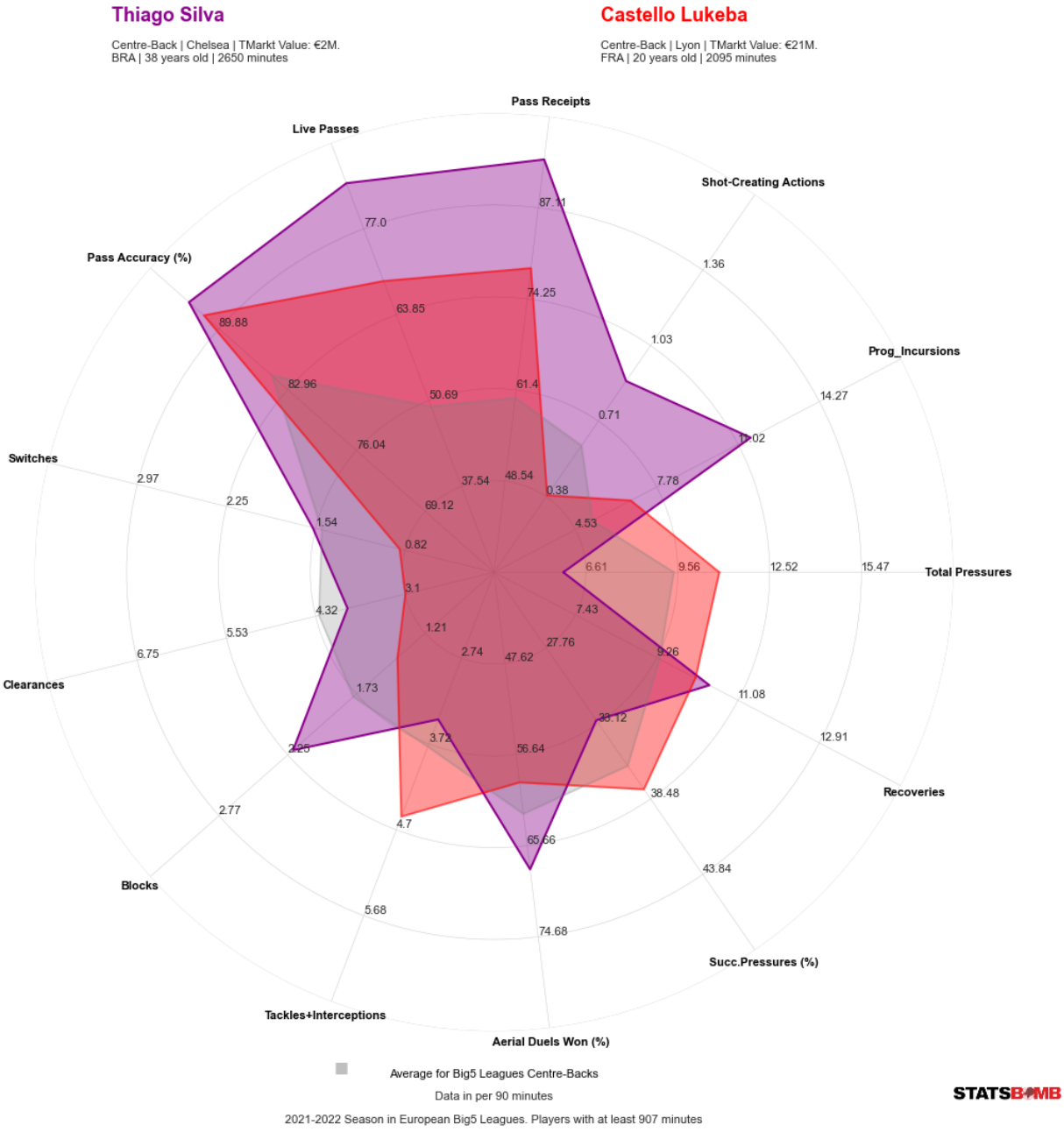
	Player	Similarity_Score
163	Castello Lukeba	0.607885
206	Edmond Tapsoba	0.852871
134	Hiroki Ito	0.867391
18	Jean-Clair Todibo	0.875805
181	Mohamed Simakan	0.901254
149	Igor	0.952636
50	Pau Torres	0.958852
259	Alessio Romagnoli	0.963394
41	Roger Ibanez	0.969754
44	Ben White	0.971936
89	Waldemar Anton	0.984667
61	Jules Kounde	0.989985
42	Jonathan Tah	0.998055

Para obtener al segundo central en cuestión, de perfil más conservador, buscamos entre aquellos que, mostrando adecuación al sistema de juego del Chelsea, muestran parecidos con alguno de los dos futbolistas que han despempeñado ese rol desde la llegada de Thomas Tuchel al banquillo de los blues: Thiago Silva y Andreas Christensen.

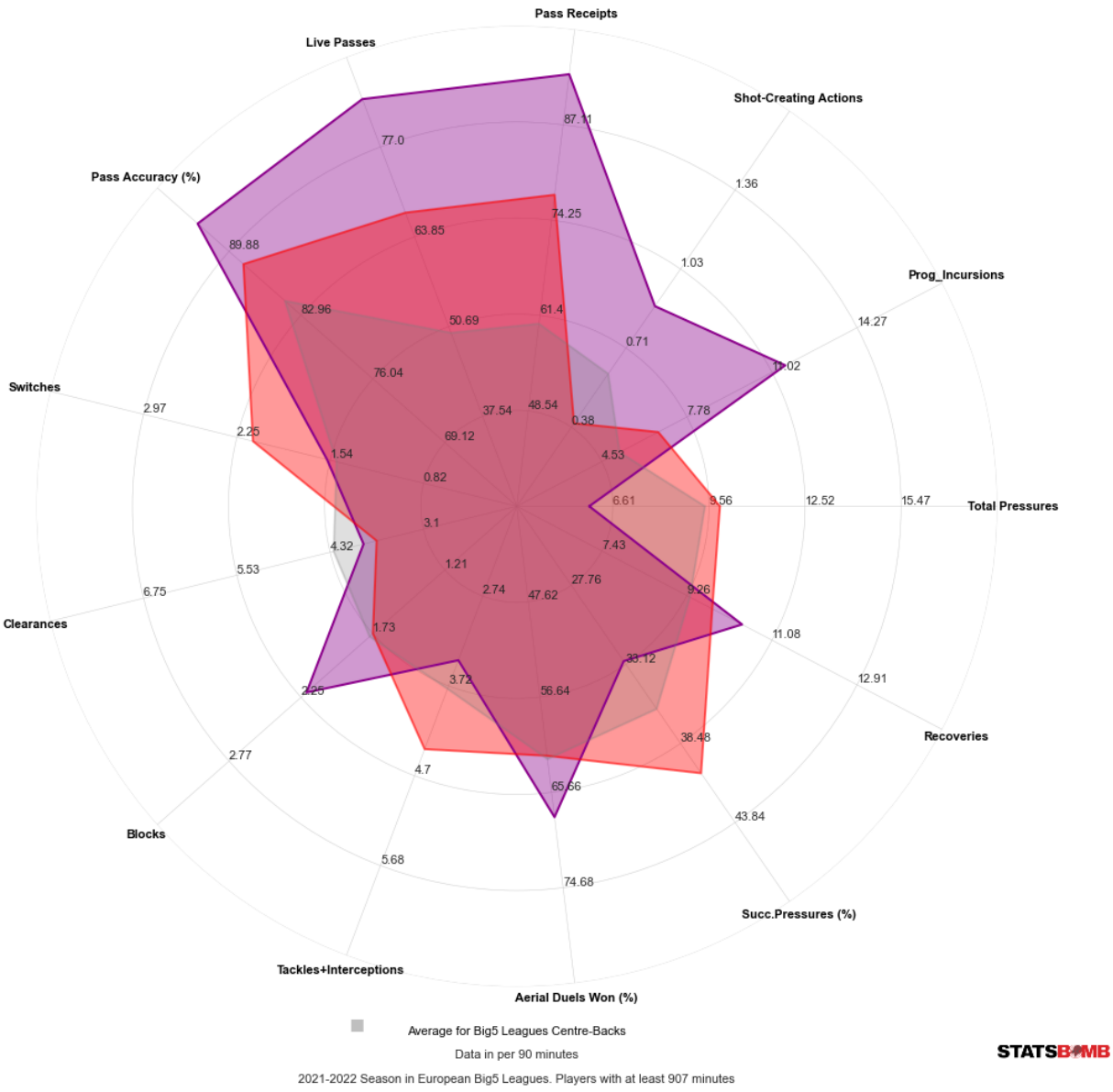
La shortlist resultante muestra jugadores de gran sensibilidad en el pase unido a un físico exhuberante. Destacan Edmond Tapsoba, Mohamed Simakan, Ben White -parece complicado que el Arsenal pueda vendérselo al Chelsea- o los sorprendentes Castello Lukeba e Hiroki Ito.

In [73]:
```
for i in list(ps2.Player.head()):
    pp.radar_comp('Thiago Silva',i)
    time.sleep(1)
```

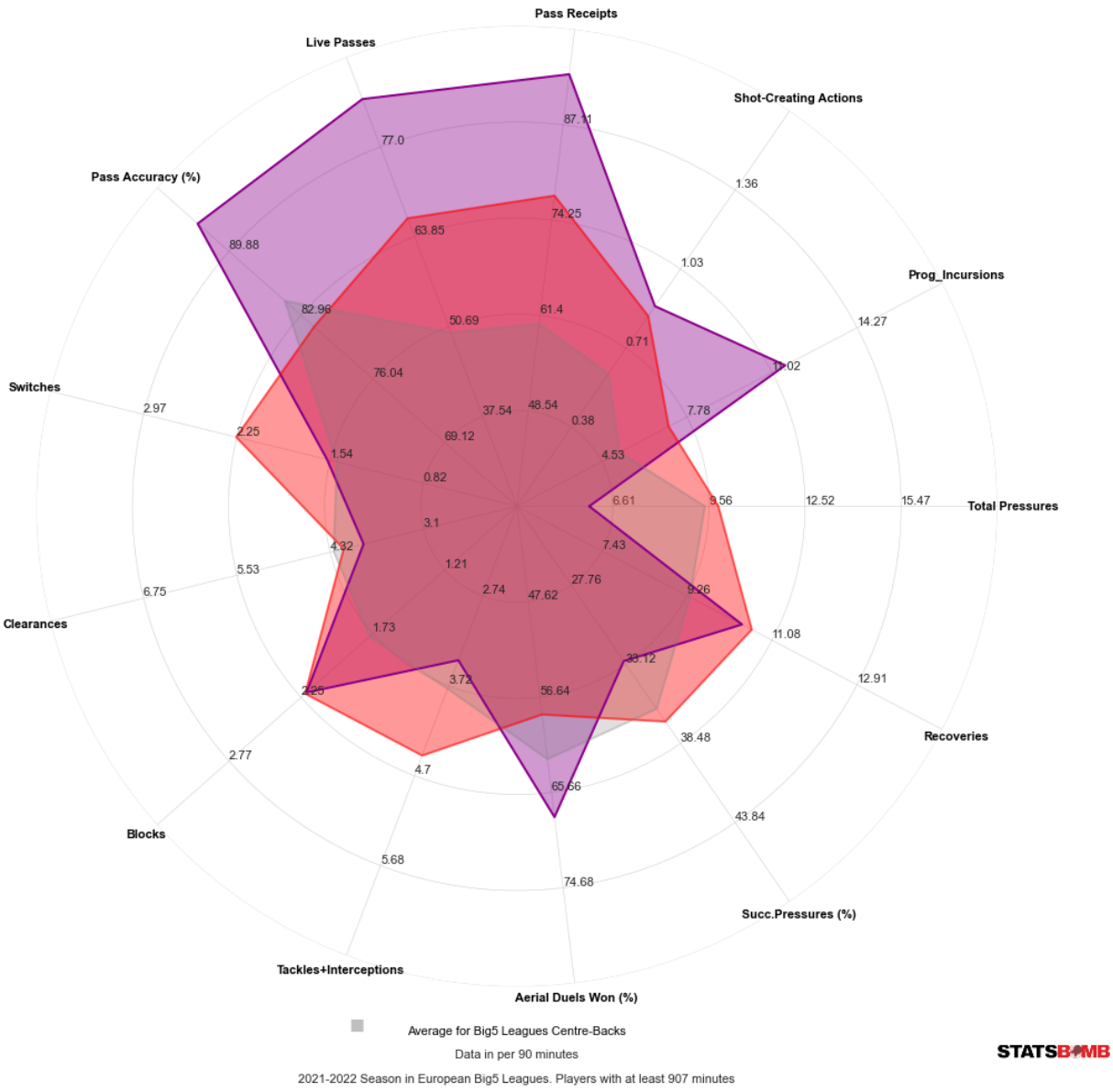
```
Could not load player photo 2
```



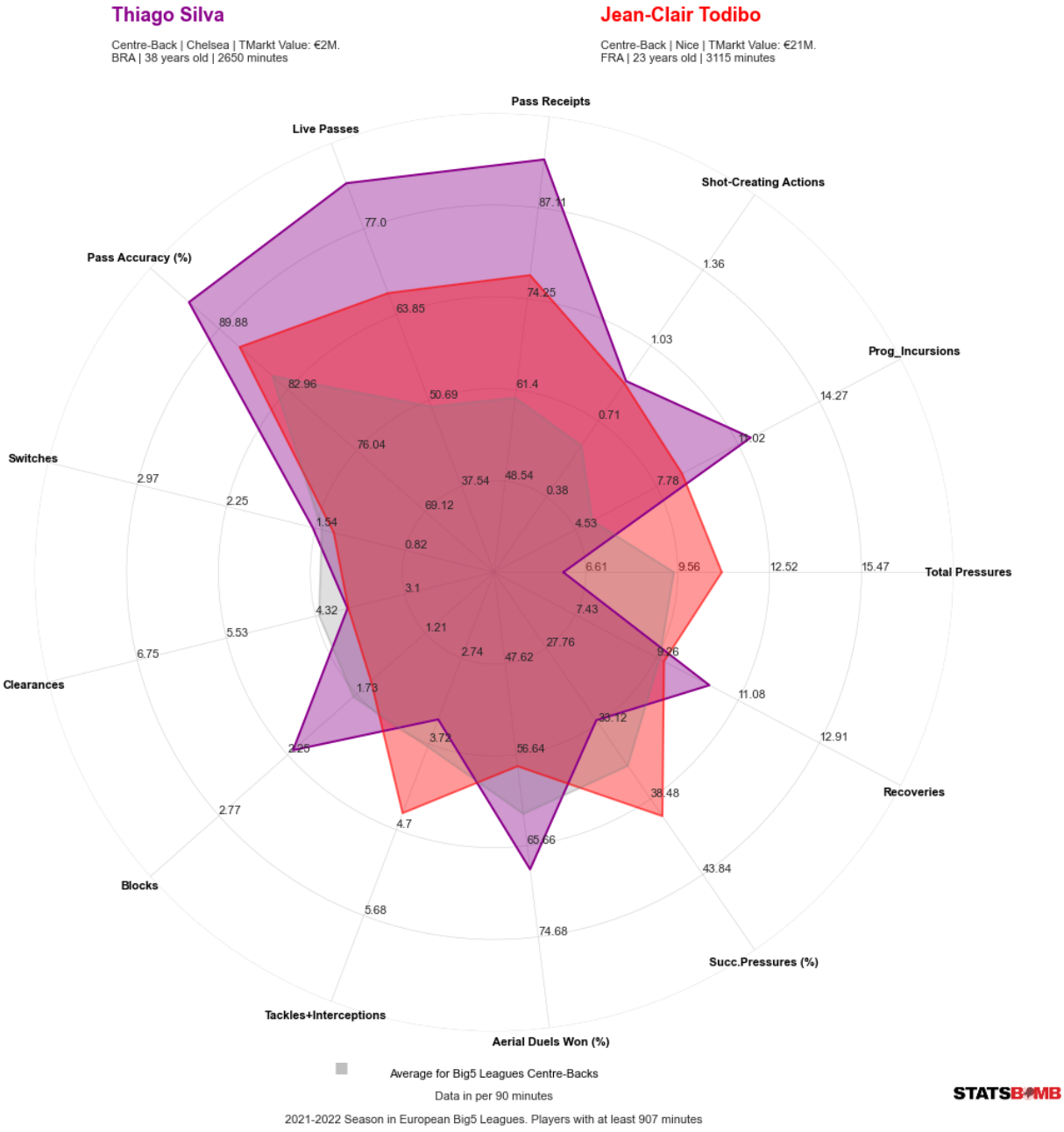
Could not load player photo 2



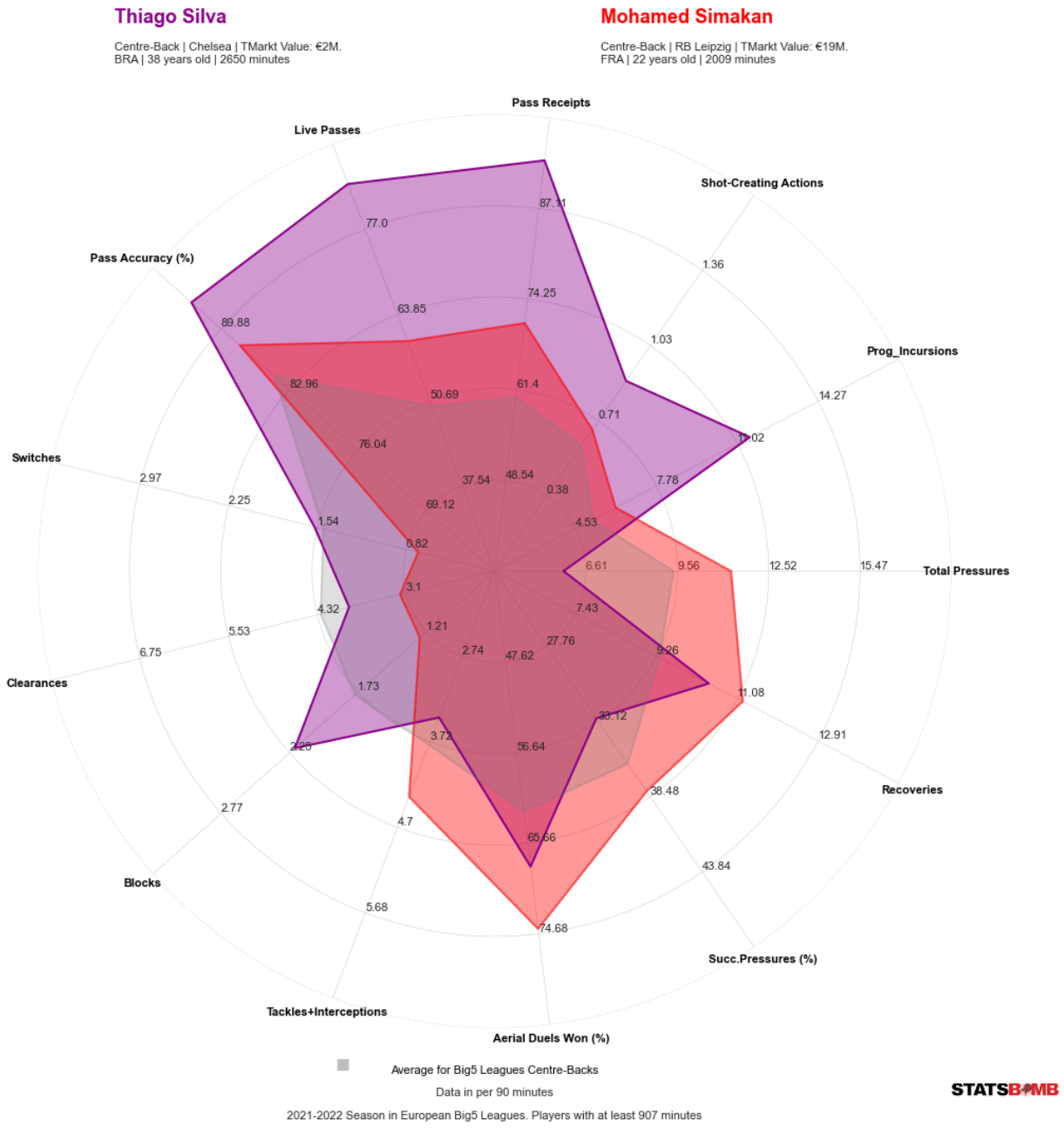
Could not load player photo 2



Could not load player photo 2



Could not load player photo 2



In [69]:

```
pp.scaplot_comp(players[players.Player.isin(chelsea)],
        n=players[players.Player.isin(chelsea)].shape[0],shower=1,x='Pressures_Press
        sizer = 'Tkl+Int',sizes=(4,2.5),
        sub='Pressing Actions - Volume vs Height',
        alpha=0.5,marker="o",index_x=False,index_y=False)
```

Pressing Actions - Volume vs Height
Top5 European Leagues | Area size determined by: Tkl+Int

STATSBOMB