Pràctica 2. Neteja i anàlisi de les dades

Tipologia i cicle de vida de les dades

Alejandro Santamarta

1. Descripció del dataset. Perquè és important i quina pregunta/problema pretén respondre?

Per a aquesta pràctica utilitzarem el dataset generat a la pràctica anterior d'scraping, que conté estadístics i dades diverses per a cada jugador i cada partit de la temporada 2019-2020 de la NBA. Donat que l'scraping va agafar una quantitat considerable de dades al respecte de moltes variables de joc intentarem resoldre preguntes amb la menor quantitat necessària de dades.

Les preguntes que volem respondre tenen a veure, per una part, en validar coses que es donen per fet al món de l'esport sense que s'aportin moltes dades al respecte i per una altra part a intentar treure un profit predictiu a les dades de les que disposam. Per tant s'intentarà respondre a les següents preguntes:

- **Efecte de joc local (apartat 4.1):** Quasi tothom dóna per suposat que el fet de jugar a casa és un avantatge per l'equip local, i que normalment es juga millor a casa que a pabellons dels equips rivals, ja sigui pel públic o per altres factors, com a desplaçaments. A més se suposa que el públic pot afectar les decisions dels àrbitres, que es poden mostrar més tendents a decisions favorables a l'equip local. Així, plantejam la següent pregunta: Realment té un efecte sobre el joc individual dels jugadors locals i sobre les decisions de l'àrbitre sobre aquests jugadors el fet de jugar al seu pabelló? Això ho traduirem a dues preguntes concretes:
    - Són els jugadors més propensos a encertar els tir a cistella (de dos o de tres punts) quan juguen a casa?
    - Són els jugadors locals menys propensos a que els pitin faltes personals?

- **Minuts i rendiment (apartat 4.2):** Els jugadors que juguen més són realment els millors? Això podria ser en termes genèrics la pregunta que voldríem resoldre. Aquesta pregunta té moltes ramificacions que poden sortir inclús del mesurable, així que intentarem restringir la pregunta als efectes estadístics dels jugadors, i per tant a si realment els millors jugadors estadísticament són els que més juguen. A més podem ampliar la pregunta per demanar-nos també si una millor producció estadística d'un determinat tipus pot estar relacionada amb una altra, és a dir: quant millor és un jugador agafant rebots ofensius ho és també agafant rebots defensius? els millors jugadors en assistències són també els millors robant pilotes?. Concretant més les preguntes finals serien:
    - Estan correlacionats els minuts de joc d'un jugador amb el rendiment estadístic? És a dir, els que juguen més minuts són realment els que tenen millors estadístiques? (No entrarem al debat de què és causa i què és conseqüència en aquesta possible relació).
    - Estan correlacionats els diferents estadístics de joc?

- **Prediccions (apartat 4.3):** Arribats a aquest punt intentarem veure si podem obtenir prediccions sobre estadístics futurs. Així intentarem resoldre aquestes dues preguntes:
    - Podem predir la quantitat de minuts que jugarà un jugador al proper partit en condicions normals?
    - Podem predir, donada aquesta quantitat de minuts, quins seran els seus nombres a nivell d'estadístics de joc?

2. Integració i selecció de les dades d'interès a analitzar.

2.1. Selecció de variables i dades

Recordem de la pràctica anterior que partim d'un joc de dades amb les següents variables:

- date: data de la realització del partit a on s'han recopilat aquestes estadístiques en format *YYYYMMDD*
- team: equip per a qui juga aquest jugador en aquest partit
- against: equip rival a aquest partit
- local: si l'equip del jugador juga com a local (True) o visitant (False)
- team_score: puntuació final de l'equip del jugador
- rival_score: puntuació final de l'equip rival
- result: resultat final, victòria per l'equip del jugador (True) o derrota (False)
- player: Nom del jugador
- mp: minuts disputats del jugador al partit en format *minuts:segons*
- fg (Field Goals): cistelles anotades al partit
- fga (Field Goals Attempts): cistelles intentades al partit
- fg_pct (Field Goals Percentage): la divisió de fga/fg

- fg3 (Field Goals 3pt): triples encertats pel jugador al partit
- fg3a (Field Goals 3pt Attempted): triples intentats pel jugador
- fg3_pct (Field Goals 3pt Percentage): la divisió de fg3a/fg3
- ft (Free Throws): tirs lliures encertats pel jugador al partit
- fta (Free Throws Attempted): tirs lliures intentats pel jugador al partit
- ft_pct (Free Throws Percentage): la divisió de fta/f
- orb (Offensive Rebounds): rebots ofensius (davall la cistella rival) aconseguits pel jugador al partit
- drb (Defensive Rebounds): rebots defensius (davall la pròpia cistella) aconseguits pel jugador al partit
- trb (Total Rebounds): la suma de orb + drb
- ast (Assistències): passades del jugador que han acabat amb cistella per part d'un company immediatament després aconseguides pel jugador al partit
- stl (Steals): pilotes robades pel jugador a un jugador de l'equip rival durant el partit.
- blk (Blocks): "taps" (https://ca.wikipedia.org/wiki/Tap_(b%C3%A0squet)) aconseguits del jugador al partit
- tov (Turnovers): pèrdues de possessió, pilotes que el jugador ha "perdut" i ha significat que la possessió ha passat al rival durant el partit
- pf (Personal Fouls): faltes personals del jugador durant el partit
- pts (Points): punts aconseguits pel jugador durant el partit
- plus_minus (plus-minus): diferència de punts aconseguits per l'equip del jugador vs l'equip rival el temps que el jugador ha estat jugant (https://en.wikipedia.org/wiki/Plus%E2%80%93minus)
- ts_pct (True Shooting Percentage): % d'efectivitat combinada de tirs de 2, triples i tirs lliures.
- efg_pct (Effective Field Goald Percentage): % reajustat tenint en compte els punts que val un tir de 2 i un tir de 3 (https://en.wikipedia.org/wiki/Effective_field_goal_percentage)
- fg3a_per_fga_pct (3PAr o Three Point Attempt Rate): percentatge de triples intentats sobre el total de tirs del jugador al partit.
- fta_per_fga_pct (FTr o Free Throw Attempt Rate): tirs lliures intentats pel jugador respecte de tirs de camp.
- orb_pct (Offensive Rebound Percentage): percentatge (estimat) de rebots ofensius que un jugador ha aconseguit del total de rebots ofensius "disponibles" pel jugador.
- drb_pct (Defensive Rebound Percentage): percentatge (estimat) de rebots defensius que un jugador ha aconseguit del total de rebots ofensius "disponibles" pel jugador.
- trb_pct (Total Rebound Percentage): percentatge (estimat) total de rebots que un jugador ha aconseguit del total de rebots "disponibles" pel jugador.
- ast_pct (Assist Percentage): percentatge de tirs aconseguits pels compnyas assistits pel jugador mentre ha estat a pista
- stl_pct (Steal Percentage): percentatge de pilotes robades pel jugador sobre el total de possessions de pilota de l'equip rival mentre el jugador ha estat a pista.
- blk_pct (Block Percentage): percentatge de jugades taponades pel judador del total de tirs de camp intentats per l'equip rival mentre el jugador ha estat a pista.

- tov_pct (Turnover Percentage): percentatge (estimat) de pilotes perdudes per un jugador per cada 100 jugades on ha participat.
- usg_pct (Usage Percentage): percentatge de jugades de l'equip on el jugador ha participat.
- off_rtg (Offensive Rating): punts produïts (punts del jugador + punts aconseguits per companys després d'assistències del jugador) per cada 100 possessions de l'equip
- def_rtg (Deffensive Rating): punts encaixats per l'equip per cada 100 possessions.
- bpm (Box Plus-Minus): comparació del rati plus-minus amb la mitja de la temporada de la resta de jugadors (https://www.basketball-reference.com/about/bpm2.html)

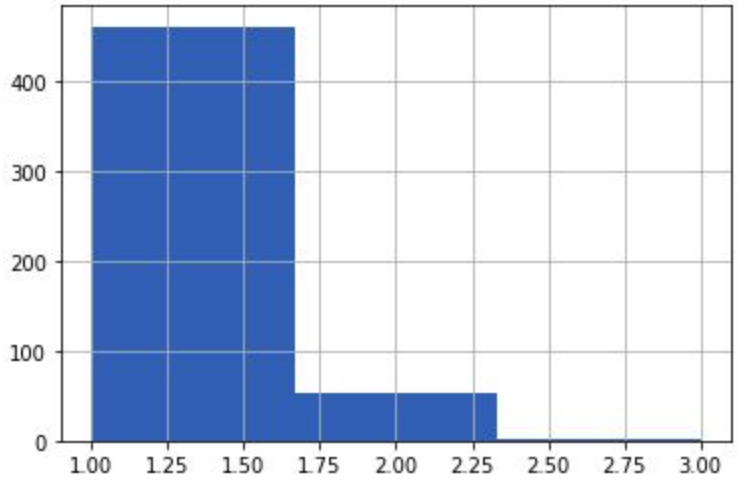
Revisant tota la llista prenem les següents decisions sobre les variables:

- **Intentem simplificar inicialment la problemàtica**, ja que intentar extreure un rendiment futur de variables complexes és complicat, així que primer intentem tenir una primera versió amb els estadístics bàsics, per tant eliminem plus_minus, ts_pct, efg_pct, fg3a_per_fga_pct, fta_per_fga_pct, orb_pct, drb_pct, trb_pct, ast_pct, stl_pct, blk_pct, tov_pct, usg_pct, off_rtg i def_rtg. Moltes d'aquestes estadístiques estan indirectament relacionades amb algunes de les bàsiques (orb_pct amb orb, per exemple), però també depenen d'altres variables que ara mateix no disposem (al cas de orb_pct el percentatge de rebots "disponibles" per al jugador a un partit). Idealment orb_pct tindria pes sobre una estimació futura de rebots ofensius a un partit del jugador. D'aquests estadístics avançats mantindrem bpm, que es una estimació bastant adequada de si el rendiment d'un jugador global sobre els indicadors.
- També eliminem una sèrie d'**estadístics que en realitat són calculables** a partir d'altres. Això inclou fg (calculable des de fga i fg_pct), fg3 (ídem amb fg3a i fg3_pct), ft (amb fta i ft_pct), trb (és la suma de orb i drb) i pts (és la suma dels tirs pel valor de cadascun).
- No seleccionem també aquells **estadístics que són referits al conjunt de l'equip**, ja que cercam el rendiment individual, no col·lectiu, per tant eliminem les variables team_score, rival_score i result.

La variable team ens presenta dubtes, ja que la primera suposició és que no molts de jugadors han canviat d'equip. Comprovem.

```
# Grafiquem la distribució d'equips per jugador a la temporada
num_teams_df['player'].value_counts().hist(bins=3)
```

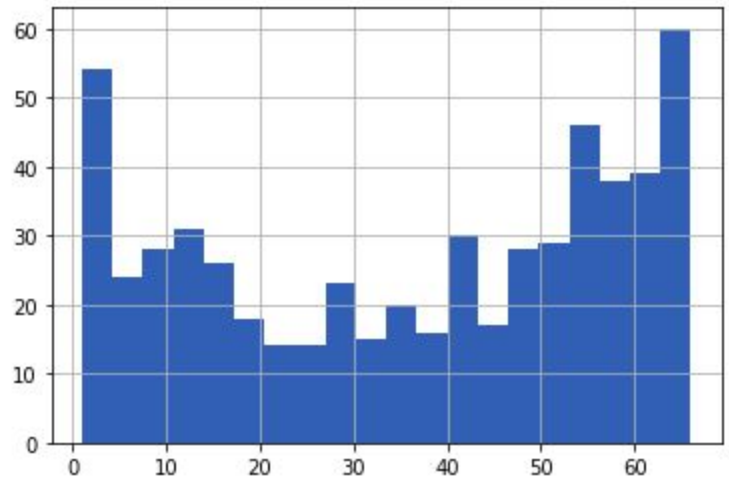
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a0837a860>
```



Davant això podem eliminar els jugadors que han tingut més d'un equip a la temporada, però trob que és més interessant fer l'avaluació de rendiment amb la dupla equip-jugador. Tot i així sí que podem avaluar la distribució de nombre de partits per jugador-equip, perquè jugadors que hagin jugat un nombre molt limitat de partits a una temporada poden no tenir bagatge suficient per ser avaluats. Comprovem.

```
# Grafiquem la distribució de partits per equip i jugador a la temporada
num_teams_df[0].hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6a07a16c50>
```



Existeixen una quantitat rellevant de jugadors amb molts pocs partits. Eliminem els registres de les combinacions amb menys de 10 partits.

```
df = df.groupby(['player','team']).filter(lambda x: len(x) >= 10)
```

Així, el dataset després del primer filtrar és el següent:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20055 entries, 0 to 20500
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     20055 non-null  int64
 1   team     20055 non-null  object
 2   against  20055 non-null  object
 3   local    20055 non-null  bool
 4   player   20055 non-null  object
 5   mp       20055 non-null  object
 6   fga      20055 non-null  int64
 7   fg_pct   19328 non-null  float64
 8   fg3a     20055 non-null  int64
 9   fg3_pct  15916 non-null  float64
 10  fta      20055 non-null  int64
 11  ft_pct   11562 non-null  float64
 12  orb      20055 non-null  int64
 13  drb      20055 non-null  int64
 14  ast      20055 non-null  int64
 15  stl      20055 non-null  int64
 16  blk      20055 non-null  int64
 17  tov      20055 non-null  int64
 18  pf       20055 non-null  int64
 19  bpm      20055 non-null  float64
dtypes: bool(1), float64(4), int64(11), object(4)
memory usage: 3.1+ MB
```

2.2. Revisió dels tipus de les variables

Comencem pel més evident, que és que el camp date no té un format de data adequat, passam a datetime64.

A més, convertim les variables team i against en categòriques.

Els minuts jugats (mp) estan en format “mm:ss”, amb el que ho convertirem a un format més llegible. Podriem senzillament passar-ho a minuts i fraccions de minut.

```
df['mp_time'] = pd.to_datetime(df['mp'], format='%M:%S')
df['minutes'] = df.apply(lambda x : x['mp_time'].minute + (x['mp_time'].second / 60),axis=1)
df = df.drop(columns=['mp', 'mp_time'])
```

3. Neteja de les dades.

3.1. Les dades contenen zeros o elements buits? Com gestionaries aquests casos?

Comencem avaluant si existeixen valors ja identificats com a nuls al joc de dades.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20055 entries, 0 to 20500
Data columns (total 20 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     20055 non-null  datetime64[ns]
 1   team     20055 non-null  category
 2   against  20055 non-null  category
 3   local    20055 non-null  bool
 4   player   20055 non-null  object
 5   fga      20055 non-null  int64
 6   fg_pct   19328 non-null  float64
 7   fg3a     20055 non-null  int64
 8   fg3_pct  15916 non-null  float64
 9   fta      20055 non-null  int64
 10  ft_pct   11562 non-null  float64
 11  orb      20055 non-null  int64
 12  drb      20055 non-null  int64
 13  ast      20055 non-null  int64
 14  stl      20055 non-null  int64
 15  blk      20055 non-null  int64
 16  tov      20055 non-null  int64
 17  pf       20055 non-null  int64
 18  bpm      20055 non-null  float64
 19  minutes  20055 non-null  float64
dtypes: bool(1), category(2), datetime64[ns](1), float64(5), int64(10), object(1)
memory usage: 2.8+ MB
```

ura gráfica

Amb una inspecció visual ja veiem que els camps de percentatges d'encert als diferents tirs tenen valors nuls. La primera hipòtesi pot ser que sigui quan un jugador no ha realitzat un tir del tipus demanat. Comprovem primer visualment si té sentit la teoria.

```
df[df['fg_pct'].isnull()].head()
```

	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
57	2019-10-23	Charlotte Hornets	Chicago Bulls	True	Nicolas Batum	0	NaN	0	NaN	0	NaN	1	3	2	0	0	2	2	-8.2	11.200000
67	2019-10-23	Detroit Pistons	Indiana Pacers	False	Christian Wood	0	NaN	0	NaN	0	NaN	0	0	0	0	0	1	0	-18.6	3.833333
117	2019-10-23	Brooklyn Nets	Minnesota Timberwolves	True	David Nwaba	0	NaN	0	NaN	2	0.5	0	3	0	0	1	1	2	-6.9	13.083333
130	2019-10-23	Memphis Grizzlies	Miami Heat	False	De'Anthony Melton	0	NaN	0	NaN	2	0.0	0	0	2	1	0	0	0	25.1	2.750000
165	2019-10-23	Philadelphia 76ers	Boston Celtics	True	Raul Neto	0	NaN	0	NaN	0	NaN	0	0	0	0	0	0	0	-5.6	1.100000

Sembla que sí. Per tant intentem constatar numèricament.

```
# Constatació numèrica sobre fg
df[df['fg_pct'].isnull()]['fga'].value_counts()
```

```
0    727
Name: fga, dtype: int64
```

```
# Constatació numèrica sobre fg3a
df[df['fg3_pct'].isnull()]['fg3a'].value_counts()
```

```
0    4139
Name: fg3a, dtype: int64
```

```
# Constatació numèrica sobre fta
df[df['ft_pct'].isnull()]['fta'].value_counts()
```

```
0    8493
Name: fta, dtype: int64
```

Efectivament. Tenim diferents opcions amb aquest valors. Podem omplir els valors amb un valor constant, posem 0, que pot desvirtuar creant “negativitat” sobre la valoració general de l’eficiència dels jugadors, però suposa una transformació senzilla i té la virtut de tenir poca afectació sobre l’estadística general de jugadors “habituals” (juguen molt i per tant abans o després han intentat mínim un tir al partit), i màxima afectació sobre jugadors que juguen esporàdicament i tenen tan poca afectació sobre el joc que ni tan sevols tiren. Aquest ja són jugadors sobre els que probablement qualsevol decisió que prenem sigui incorrecta.

Un altra opció seria trobar la mitja/mitjana/moda dels valors d’aquest camp, però probablement això difumini molts els resultats, empenyent els resultats d’alguns jugadors cap adalt i d’altres cap abaix. Probablement d’aquesta manera afegiríem un biaix sobre els jugadors, però no seriem tan conscients com amb el valor constant de 0 de la direcció d’aquest biaix.

Per tant, optam per substituir per 0 aquests valors.

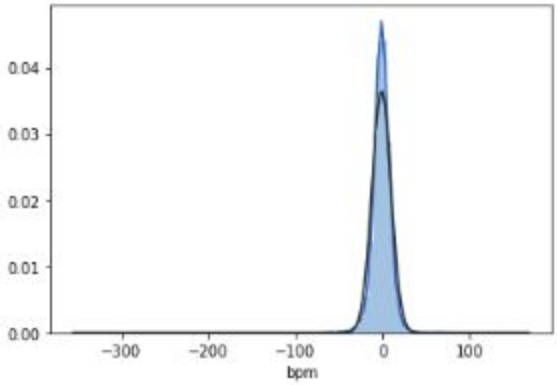
```
# Decidim per 0
df['fg_pct'] = df['fg_pct'].fillna(0)
df['fg3_pct'] = df['fg3_pct'].fillna(0)
df['ft_pct'] = df['ft_pct'].fillna(0)
df.info()
```

3.2. Identificació i tractament de valors extrems.

3.2.1. Identificació i tractament de valors extrems sobre els valors absoluts dels estadístics

Revisam un per un els estadístics i mirem quina és la distribució de les dades (comparant amb una normal) i quins són els valors extrems (superiors e inferiors) de cadascun d'aquests estadístics.

BPM



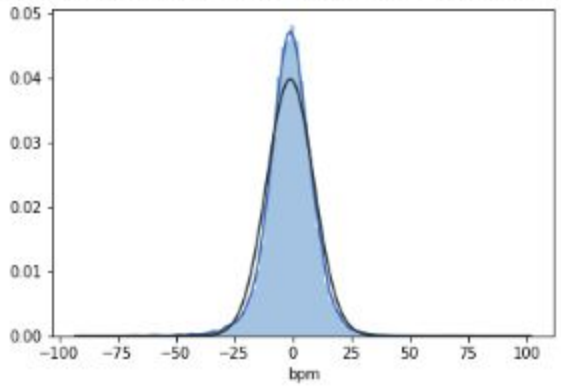
	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
971	2019-10-28	Oklahoma City Thunder	Houston Rockets	False	Deonte Burton	0	0.0	0	0.0	0	0.000	0	0	0	0	0	0	1	-350.9	0.050000
757	2019-10-26	Los Angeles Clippers	Phoenix Suns	False	Johnathan Motley	2	0.0	0	0.0	0	0.000	1	0	0	0	0	0	0	-229.0	0.233333
17302	2020-02-13	Los Angeles Clippers	Boston Celtics	False	Terance Mann	1	0.0	0	0.0	0	0.000	0	0	0	0	0	0	0	-183.9	0.200000
15252	2020-01-31	Toronto Raptors	Detroit Pistons	False	Oshae Brissett	1	0.0	1	0.0	0	0.000	0	0	0	0	0	1	2	-89.0	1.550000
10171	2019-12-28	Dallas Mavericks	Golden State Warriors	False	Boban Marjanović	3	0.0	2	0.0	0	0.000	0	0	0	0	0	0	1	-86.4	2.066667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3351	2019-11-13	Los Angeles Clippers	Houston Rockets	False	Terance Mann	1	1.0	0	0.0	0	0.000	0	0	0	0	0	0	0	125.8	0.400000
8772	2019-12-18	Dallas Mavericks	Boston Celtics	True	Ryan Broekhoff	1	1.0	1	1.0	3	0.667	0	0	0	0	0	0	2	133.0	0.783333
11145	2020-01-04	Orlando Magic	Utah Jazz	True	Melvin Frazier	1	1.0	1	1.0	0	0.000	0	0	0	0	0	0	0	147.7	0.733333
17301	2020-02-13	Los Angeles Clippers	Boston Celtics	False	Johnathan Motley	0	0.0	0	0.0	2	0.500	1	0	0	0	0	0	0	150.9	0.200000
7238	2019-12-08	Oklahoma City Thunder	Portland Trail Blazers	False	Mike Muscala	1	1.0	1	1.0	0	0.000	0	0	0	0	0	0	0	163.8	0.666667

No acaba d'encaixar amb una normal, tot i que té una forma pseudonormal. Té molta coa la distribució i podem observar que els valors extrems són molt puntuals, es podrien considerar anomalies. (Un jugador que juga uns pocs segons i només li dóna temps a cometre una personal en joc té un bpm de -350) i estan relacionats amb jugadors que no han jugat temps suficient al partit com per a disposar d'informació rellevant sobre el seu rendiment.

Intentem extreure quin és el mínim de minuts rellevants per a deixar de tenir aquest tipus de comportament.
Provam 1 minut mínim de joc:

```
sns.distplot(df[df['minutes'] > 1]['bpm'], bins=100, fit=norm)
```

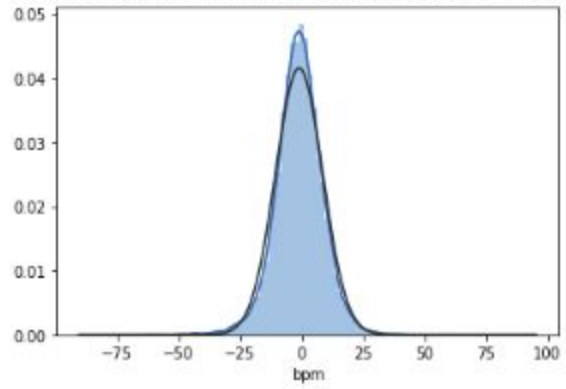
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5375e630>
```



2 minuts:

```
sns.distplot(df[df['minutes'] > 2]['bpm'], bins=100, fit=norm)
```

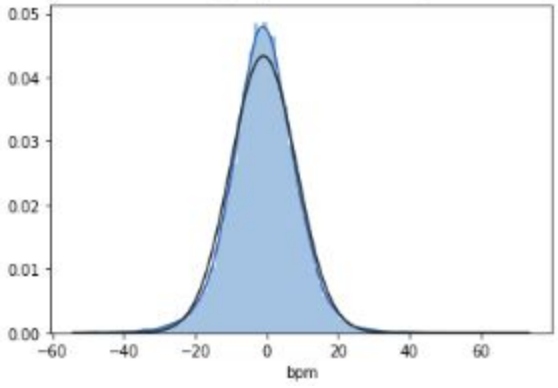
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb536900f0>
```



3 minuts:

```
sns.distplot(df[df['minutes'] > 3]['bpm'], bins=100, fit=norm)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5330e9e8>
```



Sembla que a partir dels 3 minuts s'estabilitza un poc i ja tenim informació més rellevant per a ubicar bé el rendiment dels jugadors al partit.

Ens quedem, per tant, amb aquelles intervencions de més de 3 minuts a un partit.

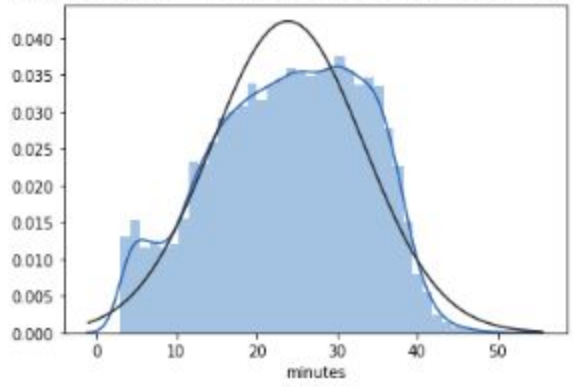
```
# Ens quedem amb els registres de joc de al menys 3 minuts
df = df[df['minutes'] > 3]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19437 entries, 0 to 20500
```

Això deixa un total de 19437 mostres, de les 20500 que disposàvem inicialment. No hem perdut un volum rellevant de dades.

Minuts jugats

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb53284198>
```



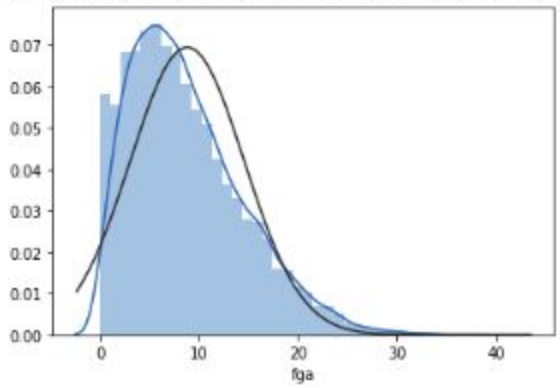
	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
8822	2019-12-19	Los Angeles Lakers	Milwaukee Bucks	False	Troy Daniels	1	0.000	1	0.000	0	0.000	0	0	0	0	0	0	0	-17.3	3.016667
4988	2019-11-24	Houston Rockets	Dallas Mavericks	True	Tyson Chandler	0	0.000	0	0.000	0	0.000	0	0	0	0	0	0	1	-11.7	3.016667
7625	2019-12-11	Los Angeles Clippers	Toronto Raptors	False	Mfiondu Kabengele	1	1.000	1	1.000	0	0.000	0	1	0	0	0	0	1	27.3	3.016667
15908	2020-02-04	Milwaukee Bucks	New Orleans Pelicans	False	Sterling Brown	3	0.000	2	0.000	0	0.000	1	0	1	0	0	0	0	-23.0	3.016667
7626	2019-12-11	Los Angeles Clippers	Toronto Raptors	False	Johnathan Motley	0	0.000	0	0.000	0	0.000	0	0	0	1	0	1	0	-2.3	3.016667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6418	2019-12-03	Houston Rockets	San Antonio Spurs	False	Russell Westbrook	30	0.233	6	0.167	4	1.000	2	8	10	2	1	5	6	-8.6	48.416667
6417	2019-12-03	Houston Rockets	San Antonio Spurs	False	James Harden	37	0.297	20	0.200	24	1.000	3	6	6	4	0	5	4	8.5	48.516667
11942	2020-01-09	Detroit Pistons	Cleveland Cavaliers	True	Andre Drummond	24	0.542	0	0.000	5	0.400	10	13	1	1	1	3	2	-1.3	49.000000
16660	2020-02-09	Atlanta Hawks	New York Knicks	True	John Collins	19	0.632	3	0.333	9	0.778	5	11	3	0	1	0	2	5.5	49.883333
6416	2019-12-03	Houston Rockets	San Antonio Spurs	False	P.J. Tucker	8	0.500	5	0.400	0	0.000	2	7	2	2	2	1	5	0.1	51.550000

Als valors extrems observem el tall de 3 minuts que ens hem auto-imposat al punt anterior. Els valors més extrems per la part superior tenen sentit, jugadors essencials als equips que juguen partits amb múltiples pròrrogues, ampliant els seus minuts de joc. Una opció seria eliminar els registres que superessin els minuts màxims de joc d'un partit sense pròrrogues, però això deixaria registres de partits amb pròrroga per a jugadors que no han jugat aquesta quantitat de minuts. També podem eliminar aquests registres, però les estadístiques d'un

partit amb pròrroga semblen igual de rellevants que les d'un partit sense pròrroga, per tant deixam aquests valors sense modificar i amb tota la informació ja veurem si podem relativitzar l'impacte dels minuts de joc sobre els estadístics.

Tirs intentats

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb52916e48>
```

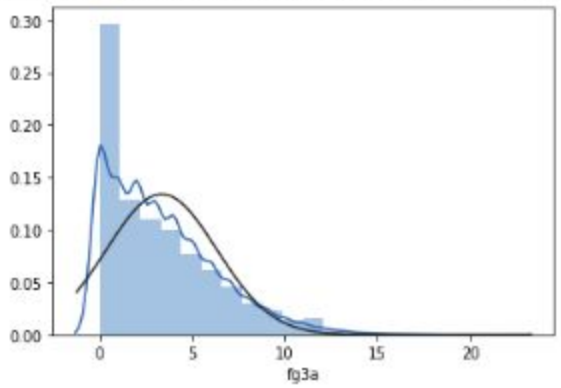


	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
14059	2020-01-22	Houston Rockets	Denver Nuggets	True	Thabo Sefolosha	0	0.000	0	0.000	0	0.000	0	3	1	0	0	1	0	-1.1	3.083333
2677	2019-11-08	Portland Trail Blazers	Brooklyn Nets	True	Rodney Hood	0	0.000	0	0.000	0	0.000	0	1	0	0	0	0	2	-11.6	6.016667
1775	2019-11-02	Charlotte Hornets	Golden State Warriors	False	Cody Martin	0	0.000	0	0.000	2	0.500	1	1	0	0	1	0	0	14.3	4.466667
5273	2019-11-25	Los Angeles Lakers	San Antonio Spurs	False	Dwight Howard	0	0.000	0	0.000	0	0.000	0	1	0	0	0	0	5	-13.2	17.983333
10829	2020-01-02	Miami Heat	Toronto Raptors	True	Kelly Olynyk	0	0.000	0	0.000	0	0.000	0	1	0	0	0	0	0	-4.9	3.483333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6417	2019-12-03	Houston Rockets	San Antonio Spurs	False	James Harden	37	0.297	20	0.200	24	1.000	3	6	6	4	0	5	4	8.5	48.516667
2557	2019-11-08	Golden State Warriors	Minnesota Timberwolves	False	D'Angelo Russell	37	0.514	17	0.412	8	0.875	0	9	5	3	2	3	0	22.3	40.083333
12727	2020-01-14	Houston Rockets	Memphis Grizzlies	False	James Harden	37	0.351	19	0.263	11	0.909	3	3	6	0	1	1	3	6.1	39.133333
9457	2019-12-23	Washington Wizards	New York Knicks	False	Bradley Beal	38	0.342	11	0.182	4	0.500	3	1	3	2	2	3	2	-8.9	36.533333
3793	2019-11-16	Houston Rockets	Minnesota Timberwolves	False	James Harden	41	0.390	22	0.364	11	0.818	1	5	6	0	0	5	4	2.9	38.683333

No hi ha valors extrems estranys. Hi ha jugadors que no han intentat cap tir (ja ho sabíem d'abans), que es un valor amb tot el sentit del món, i de forma esporàdica algun jugador disposa de molts de minuts i donada la seva rellevància al joc de l'equip té molts de tirs a cistella. No s'observen motius per a tractar els valors extrems.

Triples intentats

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb52809320>
```

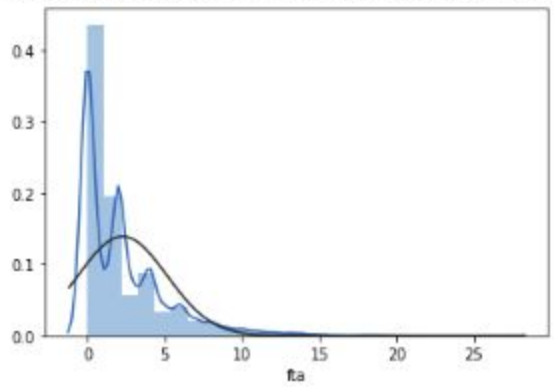


	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
20500	2020-03-11	Dallas Mavericks	Denver Nuggets	True	Willie Cauley-Stein	3	1.000	0	0.000	0	0.000	1	6	0	0	0	1	1	6.5	10.500000
3358	2019-11-13	Houston Rockets	Los Angeles Clippers	True	Tyson Chandler	1	1.000	0	0.000	4	0.750	2	7	0	0	2	0	0	11.2	16.150000
9268	2019-12-21	Minnesota Timberwolves	Portland Trail Blazers	False	Josh Okogie	5	0.400	0	0.000	0	0.000	1	0	0	1	1	1	1	-6.3	23.216667
3361	2019-11-13	San Antonio Spurs	Minnesota Timberwolves	False	DeMar DeRozan	18	0.667	0	0.000	4	0.750	1	4	4	1	2	2	3	7.7	33.433333
16824	2020-02-10	Orlando Magic	Atlanta Hawks	True	Michael Carter-Williams	3	0.000	0	0.000	6	0.833	1	3	5	1	0	1	2	-4.6	19.650000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13836	2020-01-20	Portland Trail Blazers	Golden State Warriors	True	Damian Lillard	37	0.459	20	0.550	16	1.000	0	10	7	1	0	2	1	20.3	45.133333
11826	2020-01-08	Houston Rockets	Atlanta Hawks	False	James Harden	34	0.265	20	0.200	23	0.826	2	8	10	2	3	8	2	4.4	39.733333
5164	2019-11-25	Sacramento Kings	Boston Celtics	False	Buddy Hield	26	0.577	21	0.524	1	0.000	0	5	2	0	1	4	0	11.9	41.916667
3793	2019-11-16	Houston Rockets	Minnesota Timberwolves	False	James Harden	41	0.390	22	0.364	11	0.818	1	5	6	0	0	5	4	2.9	38.683333
13333	2020-01-18	Boston Celtics	Phoenix Suns	True	Marcus Smart	25	0.520	22	0.500	0	0.000	0	5	8	4	0	1	4	23.8	33.350000

Mateixa situació que amb tirs intentats.

Tirs lliures intentats

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb527776a0>
```

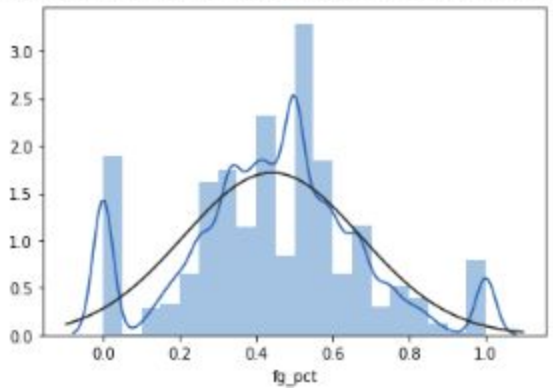


	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
20500	2020-03-11	Dallas Mavericks	Denver Nuggets	True	Willie Cauley-Stein	3	1.000	0	0.000	0	0.000	1	6	0	0	0	1	1	6.5	10.500000
14823	2020-01-28	New York Knicks	Charlotte Hornets	False	Mitchell Robinson	2	1.000	0	0.000	0	0.000	2	8	0	1	5	0	4	5.8	25.900000
7004	2019-12-07	Dallas Mavericks	New Orleans Pelicans	True	Dorian Finney-Smith	5	0.600	4	0.500	0	0.000	1	3	2	0	1	1	1	8.4	20.483333
14822	2020-01-28	New York Knicks	Charlotte Hornets	False	Taj Gibson	2	0.500	1	1.000	0	0.000	0	3	1	0	1	0	2	2.8	15.700000
7007	2019-12-07	Dallas Mavericks	New Orleans Pelicans	True	Dwight Powell	2	1.000	0	0.000	0	0.000	1	5	0	3	0	1	1	8.4	18.766667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
973	2019-10-28	Houston Rockets	Oklahoma City Thunder	True	James Harden	21	0.381	14	0.214	22	0.955	0	3	7	1	1	4	5	12.8	36.866667
5976	2019-11-30	Houston Rockets	Atlanta Hawks	True	James Harden	24	0.667	14	0.571	23	0.870	0	3	8	3	1	5	3	29.1	30.683333
11826	2020-01-08	Houston Rockets	Atlanta Hawks	False	James Harden	34	0.265	20	0.200	23	0.826	2	8	10	2	3	8	2	4.4	39.733333
6417	2019-12-03	Houston Rockets	San Antonio Spurs	False	James Harden	37	0.297	20	0.200	24	1.000	3	6	6	4	0	5	4	8.5	48.516667
1162	2019-10-29	Los Angeles Lakers	Memphis Grizzlies	True	Anthony Davis	17	0.412	2	0.000	27	0.963	8	12	2	0	2	0	2	22.2	30.566667

Mateixa situació que amb tirs intentats i triples intentats.

Percentatge de tirs

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb52670978>
```

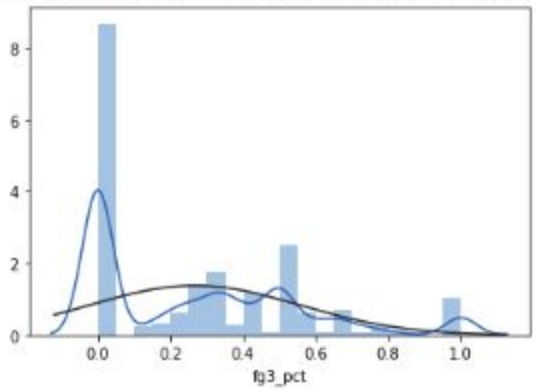


	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
18516	2020-02-27	Los Angeles Lakers	Golden State Warriors	False	Alex Caruso	5	0.0	1	0.0	2	0.0	0	4	3	1	1	1	0	-9.4	15.450000
13749	2020-01-20	Los Angeles Lakers	Boston Celtics	False	Jared Dudley	0	0.0	0	0.0	0	0.0	0	0	0	0	0	1	0	-17.5	4.566667
1679	2019-11-02	Detroit Pistons	Brooklyn Nets	True	Christian Wood	0	0.0	0	0.0	0	0.0	0	0	0	0	0	1	0	-16.4	6.750000
4477	2019-11-20	Houston Rockets	Denver Nuggets	False	Thabo Sefolosha	2	0.0	1	0.0	0	0.0	1	2	1	0	0	0	1	-6.5	15.000000
11219	2020-01-04	Denver Nuggets	Washington Wizards	False	Juan Hernangómez	0	0.0	0	0.0	0	0.0	0	1	1	0	0	0	0	3.4	5.383333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3713	2019-11-16	Brooklyn Nets	Chicago Bulls	False	Jarrett Allen	5	1.0	0	0.0	1	1.0	1	6	2	2	2	0	2	13.4	21.716667
3705	2019-11-15	Los Angeles Lakers	Sacramento Kings	True	Dwight Howard	1	1.0	0	0.0	2	1.0	1	6	0	0	1	1	3	-3.9	21.833333
14994	2020-01-29	Chicago Bulls	Indiana Pacers	False	Kris Dunn	3	1.0	0	0.0	0	0.0	0	4	3	1	0	1	1	8.2	16.350000
3777	2019-11-16	New Orleans Pelicans	Miami Heat	False	Derrick Favors	1	1.0	0	0.0	0	0.0	0	4	2	0	0	0	1	10.9	8.233333
20500	2020-03-11	Dallas Mavericks	Denver Nuggets	True	Willie Cauley-Stein	3	1.0	0	0.0	0	0.0	1	6	0	0	0	1	1	6.5	10.500000

Tots els valors són factibles (estan entre 0 i 1). Tot i que la distribució s'aferra molt als dos extrems (més al 0 per la decisió que vàrem prendre sobre els NaN), aquests valors són representatius. Si els tirs són pocs és normal que els valors s'acumulin a determinats valors (0, 1, 0.5, 0.25, 0.75, …), corresponents a les combinacions de possibilitats d'encert amb pocs tirs llençats. Donada la distribució de tirs ja sabem que en general la major part dels jugadors intenten relativament pocs tirs per partit, amb el que aquest valors són "normals". No modificam res, ido.

Percentatge de triples

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5263a080>
```

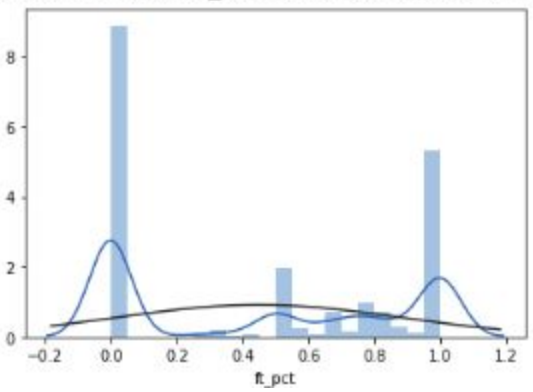


	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
20500	2020-03-11	Dallas Mavericks	Denver Nuggets	True	Willie Cauley-Stein	3	1.000	0	0.0	0	0.00	1	6	0	0	0	1	1	6.5	10.500000
12888	2020-01-15	Washington Wizards	Chicago Bulls	False	Jordan McRae	13	0.462	5	0.0	4	1.00	0	4	3	1	0	1	3	-3.1	25.483333
12889	2020-01-15	Washington Wizards	Chicago Bulls	False	Ish Smith	3	0.667	0	0.0	2	0.50	0	2	5	0	1	2	2	-4.0	22.583333
6110	2019-12-01	Detroit Pistons	San Antonio Spurs	True	Blake Griffin	12	0.333	3	0.0	2	1.00	0	2	6	0	0	2	1	-9.2	21.933333
6109	2019-12-01	Detroit Pistons	San Antonio Spurs	True	Andre Drummond	8	0.500	0	0.0	4	0.25	6	10	2	1	3	4	0	-2.1	25.983333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7904	2019-12-13	Orlando Magic	Houston Rockets	True	Mohamed Bamba	6	0.500	1	1.0	2	1.00	3	3	1	0	0	0	1	3.4	15.000000
17047	2020-02-11	Boston Celtics	Houston Rockets	False	Grant Williams	2	0.500	1	1.0	4	1.00	1	3	0	0	0	0	1	14.3	8.750000
10847	2020-01-02	Chicago Bulls	Utah Jazz	True	Shaquille Harrison	2	0.500	1	1.0	0	0.00	0	0	0	1	0	0	0	3.1	10.600000
10804	2020-01-02	Indiana Pacers	Denver Nuggets	True	Jeremy Lamb	13	0.692	5	1.0	7	1.00	1	5	2	1	0	0	2	13.2	36.816667
7730	2019-12-11	New Orleans Pelicans	Milwaukee Bucks	False	Nickeil Alexander-Walker	4	0.500	2	1.0	0	0.00	0	1	0	0	0	0	0	3.6	10.116667

Situació semblant a la del percentatge de tirs (més aferrat al 0 que a l'1 per ser un tipus de tir més complicat i per tenir més incidència la nostra transformació de valors NaN). El raonament és el mateix, ho deixam estar.

Percentatge de tirs lliures

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5253c908>
```



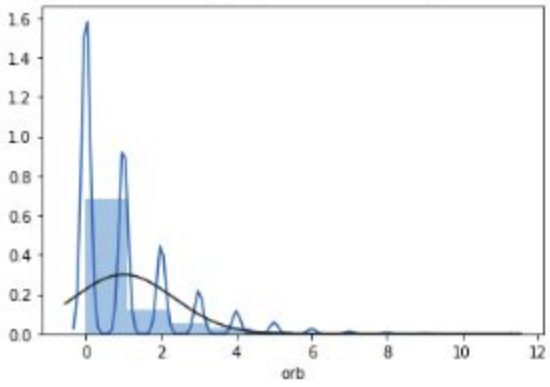
	date	team	against	local	player	fga	fg_pct	fg3a	fg3_pct	fta	ft_pct	orb	drb	ast	stl	blk	tov	pf	bpm	minutes
0	2019-10-22	New Orleans Pelicans	Toronto Raptors	False	Jrue Holiday	15	0.400	6	0.167	2	0.0	2	2	6	0	2	5	2	-6.8	41.083333
10790	2020-01-02	Cleveland Cavaliers	Charlotte Hornets	True	John Henson	6	0.833	1	0.000	0	0.0	1	5	1	0	0	1	3	2.6	14.816667
10792	2020-01-02	Cleveland Cavaliers	Charlotte Hornets	True	Matthew Dellavedova	3	0.333	2	0.000	0	0.0	0	0	4	0	0	0	2	-6.9	10.700000
10798	2020-01-02	Denver Nuggets	Indiana Pacers	False	Mason Plumlee	6	0.667	0	0.000	1	0.0	2	5	4	2	2	1	2	9.8	23.816667
10801	2020-01-02	Denver Nuggets	Indiana Pacers	False	Monte Morris	5	0.600	1	1.000	0	0.0	0	0	3	0	0	0	0	5.4	13.983333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13065	2020-01-16	Milwaukee Bucks	Boston Celtics	True	George Hill	6	0.667	3	1.000	2	1.0	0	2	2	0	0	2	4	0.8	25.466667
13064	2020-01-16	Milwaukee Bucks	Boston Celtics	True	Donte DiVincenzo	11	0.545	6	0.667	3	1.0	1	2	1	2	0	0	1	11.4	27.766667
13063	2020-01-16	Milwaukee Bucks	Boston Celtics	True	Eric Bledsoe	9	0.222	3	0.333	4	1.0	0	2	4	1	0	2	4	-7.8	20.100000
13090	2020-01-16	Denver Nuggets	Golden State Warriors	False	Will Barton	20	0.550	10	0.700	2	1.0	1	4	7	1	1	3	2	9.1	43.050000
10141	2019-12-28	New York Knicks	Washington Wizards	False	Marcus Morris	9	0.222	5	0.400	4	1.0	0	4	3	0	0	1	5	-4.1	23.366667

Mateixa situació que als dos estadístics anteriors.

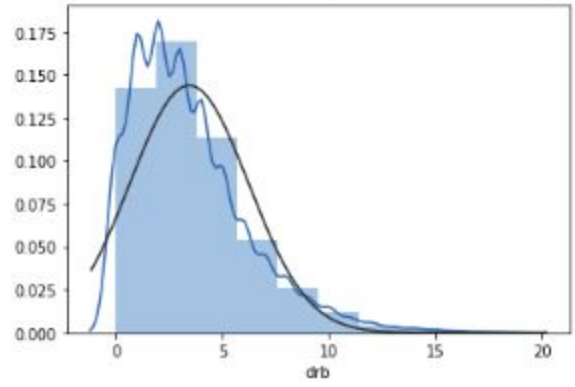
Rebots ofensius, rebots defensius, assistències, robatoris, taps, pèrdues i personals

Tots aquests estadístics tenen el mateix comportament que el de tirs, al Colab se troben tots els valors extrems. Aquí a la documentació, per simplicitat només posarem les distribucions.

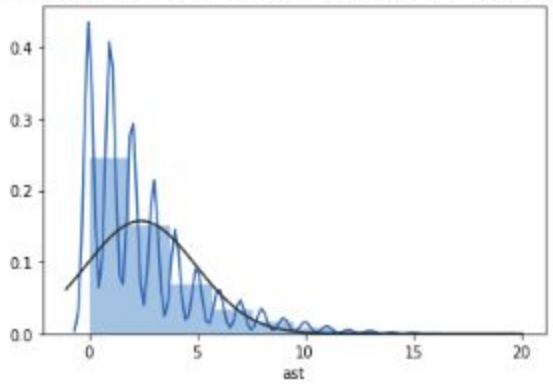
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb524c50f0>
```



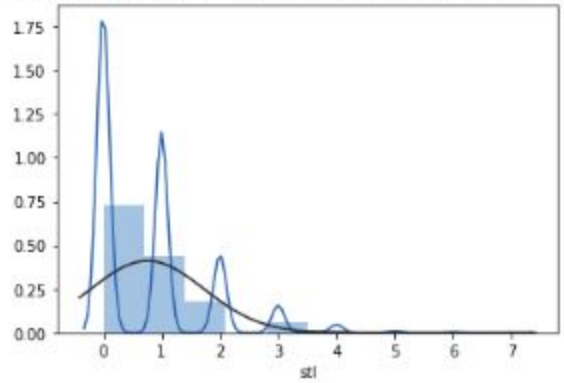

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb523cc128>
```



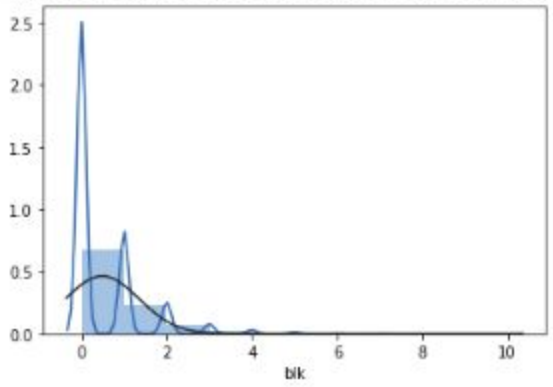
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb523666a0>
```



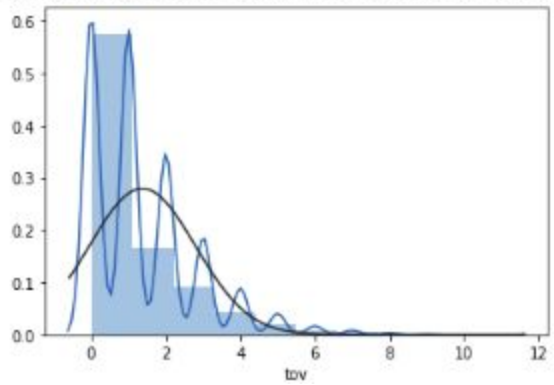

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb52324588>
```



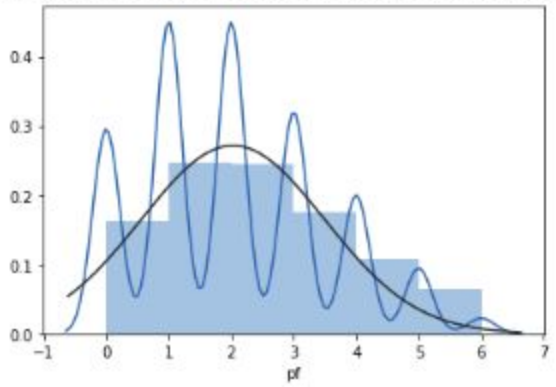

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb522aa6d8>
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb522124e0>
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5215d6d8>
```



3.2.2. Identificació i tractament de valors extrems sobre estadístics per minut de joc

Aquestes distribucions no són normals, presenten valors extrems i no tenen un barem adequat per determinar si més és millor o no. Per ser correcta, l'estadístic de 5 rebots defensius d'un jugador a un partit hauria de venir acompanyada del minuts que ha necessitat aquest jugador per a aconseguir aquests rebots. No és el mateix 5 rebots en 40 minuts que en 20.

Així ido sabem que volem matisar els estadístics amb els minuts de joc. Tenim dues formes de fer-ho: o bé miram el nombre d'estadístics per minut o miram els minuts de joc necessaris per aquest jugador per a aconseguir un estadístic. En tots dos casos se genera un problema de matisos sobre els valors de 0 sobre n minuts. És a dir: 0 assistències sobre 5 minuts o 0 assistències sobre 20 minuts en ambdós casos repercuteix en la mateixa xifra (0 o infinit).

Optam per calcular el nombre de minuts necessaris per a aconseguir un estadístic.

```
df['min_per_fga'] = df['minutes']/df['fga']
df['min_per_fg3a'] = df['minutes']/df['fg3a']
df['min_per_fta'] = df['minutes']/df['fta']
df['min_per_orb'] = df['minutes']/df['orb']
df['min_per_drb'] = df['minutes']/df['drb']
df['min_per_ast'] = df['minutes']/df['ast']
df['min_per_stl'] = df['minutes']/df['stl']
df['min_per_blk'] = df['minutes']/df['blk']
df['min_per_tov'] = df['minutes']/df['tov']
df['min_per_pf'] = df['minutes']/df['pf']
```

Això genera valors infinits, que substituim per NaN per simplificar el seu tractament.

```
# Canviem els infinits per NaN
df = df.replace([np.inf, -np.inf], np.nan)
```

En primera instància decidim que per a aquells partits a on no s'ha generat cap estadístic substituirem el valor per la mitja d'aquest valor a la temporada per a aquell jugador en concret.

```
# Primer tractament de NaN -> substituir per la mitja del jugador durant la temporada
df["min_per_fga"] = df[['player','min_per_fga']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_fg3a"] = df[['player','min_per_fg3a']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_fta"] = df[['player','min_per_fta']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_orb"] = df[['player','min_per_orb']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_drb"] = df[['player','min_per_drb']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_ast"] = df[['player','min_per_ast']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_stl"] = df[['player','min_per_stl']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_blk"] = df[['player','min_per_blk']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_tov"] = df[['player','min_per_tov']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df["min_per_pf"] = df[['player','min_per_pf']].groupby("player").transform(lambda x: x.fillna(x.mean()))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19437 entries, 0 to 20500
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          19437 non-null  datetime64[ns]
 1   team          19437 non-null  category
 2   against       19437 non-null  category
 3   local         19437 non-null  bool
 4   player        19437 non-null  object
 5   fga           19437 non-null  int64
 6   fg_pct        19437 non-null  float64
 7   fg3a          19437 non-null  int64
 8   fg3_pct       19437 non-null  float64
 9   fta           19437 non-null  int64
 10  ft_pct        19437 non-null  float64
 11  orb           19437 non-null  int64
 12  drb           19437 non-null  int64
 13  ast           19437 non-null  int64
 14  stl           19437 non-null  int64
 15  blk           19437 non-null  int64
 16  tov           19437 non-null  int64
 17  pf            19437 non-null  int64
 18  bpm           19437 non-null  float64
 19  minutes       19437 non-null  float64
 20  min_per_fga   19437 non-null  float64
 21  min_per_fg3a  18872 non-null  float64
 22  min_per_fta   19422 non-null  float64
 23  min_per_orb   19416 non-null  float64
 24  min_per_drb   19437 non-null  float64
 25  min_per_ast   19431 non-null  float64
 26  min_per_stl   19395 non-null  float64
 27  min_per_blk   19259 non-null  float64
 28  min_per_tov   19432 non-null  float64
 29  min_per_pf    19437 non-null  float64
dtypes: bool(1), category(2), datetime64[ns](1), float64(15), int64(10), object(1)
memory usage: 4.2+ MB
```

Aquesta tècnica funciona completament per a minuts per tir intentat, minuts per personal i minuts per rebot defensiu, però no funciona completament per a la

resta de valors. Això implica que hi ha jugadors que no han aconseguit un estadístic en tota la temporada (un rebot ofensiu, una assistència, un triple, …). Donada l'especialització que tenen els jugadors a diferents posicions és factible que hagi jugadors jugant de pivot que no generin cap assistència i bases que no generin cap rebot ofensiu, per exemple.

Eliminar aquests registres no sembla òptim: molt probablement els registres sense un estadístic i els demés no coincideixin, amb el que estem eliminant una quantitat rellevant de registres. Una altra opció seria que aquests valors tinguessin el valor mig que aconsegueixen companys a la mateixa posició o un rol semblant. Com que això és bastant eteri amb les dades que disposem (tot i que es podria intentar una clusterització amb la resta de valors per obtenir la mitjana dels jugadors del mateix cluster), en aquest punt optam per assignar el valor mig de la temporada a aquests NaN.

```
df = df.fillna(df.mean())
df.info()
```

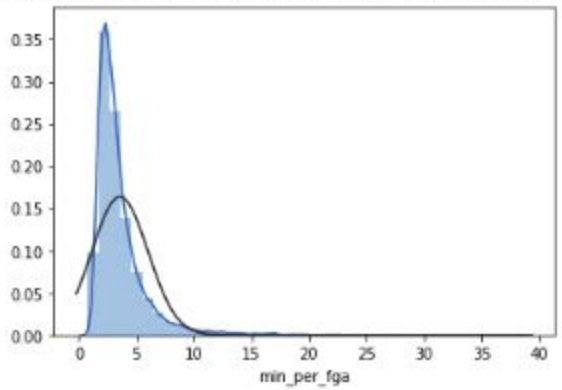
Finalment, eliminem els estadístics no ponderats per minut.

```
df = df.drop(columns=['fga', 'fg3a', 'fta', 'orb', 'drb', 'ast', 'stl', 'blk', 'tov', 'pf'])
```

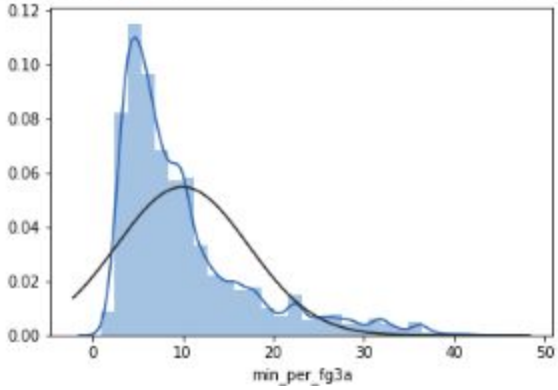
Tornam analitzar les distribucions i els valors extrems per a cadascun dels estadístics, ara ponderats per minut. Novament deixam els valors extrems al colab i adjuntam només les distribucions.

Estadístics ponderats per minuts de joc

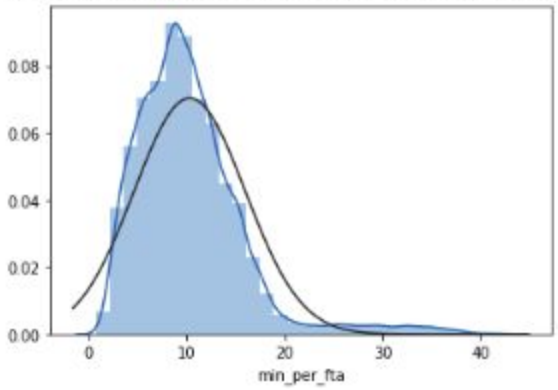
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb52befe10>
```



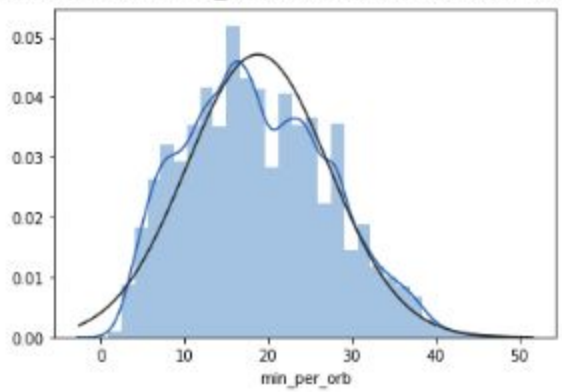
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51fe70f0>
```



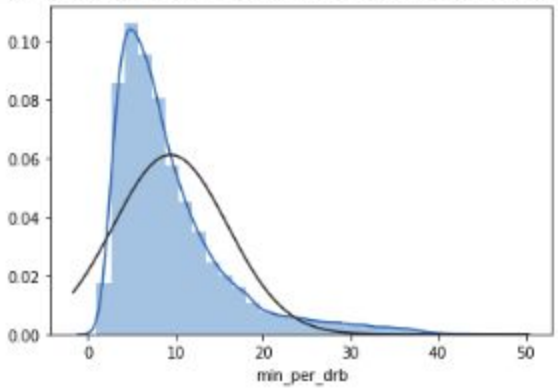

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51ef3320>
```



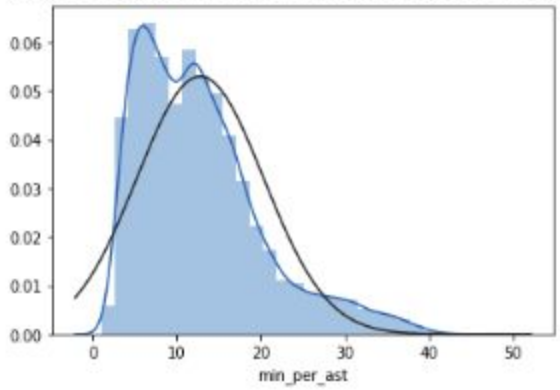

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51f15748>
```



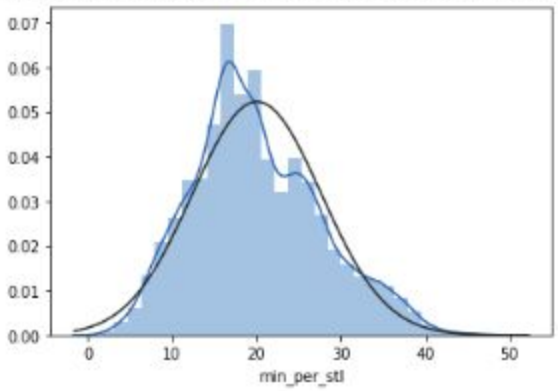
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51d8d438>
```



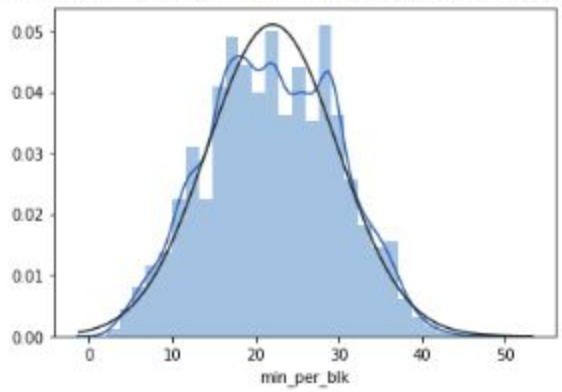

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51d33470>
```



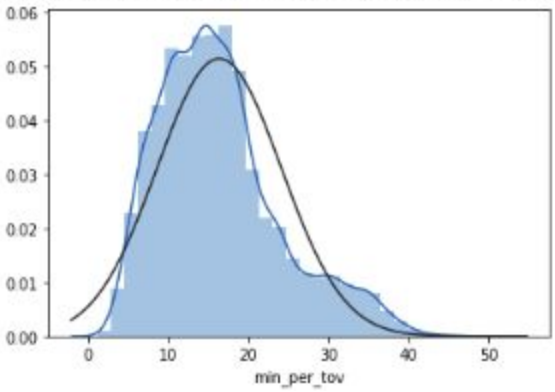

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51fa7eb8>
```



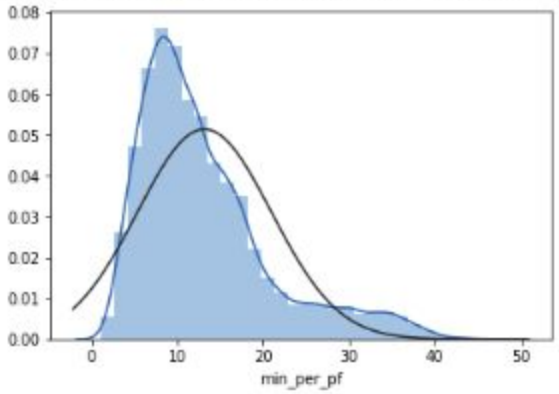
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51c1b2b0>
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51b2a4a8>
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb51a42e80>
```

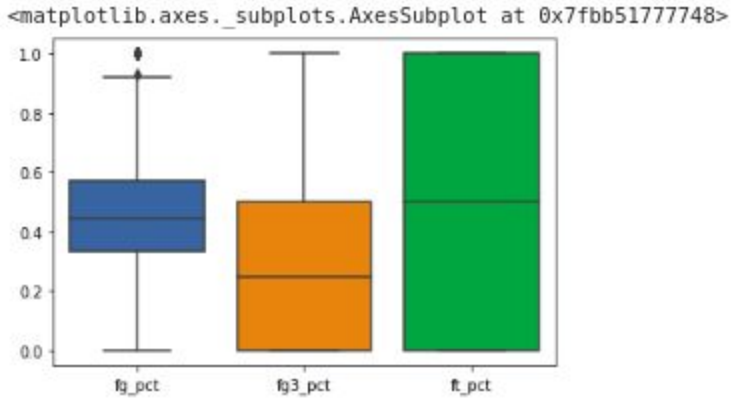


Com podem veure, els robatoris i els taps segueixen una distribució més tendent cap a una normal, mentre que les demés tiren més cap a una “long tail”.
Els valors extrems propers al 0 són actuacions molt òptimes per part d’un jugador a un partit, mentre que les més allunyades del 0 corresponen a actuaciones molt dolentes, però que es pot veure com a “normals” (La interpretació de min_per_pf hauria de ser la contrària, més òptima quant més allunyat de 0).

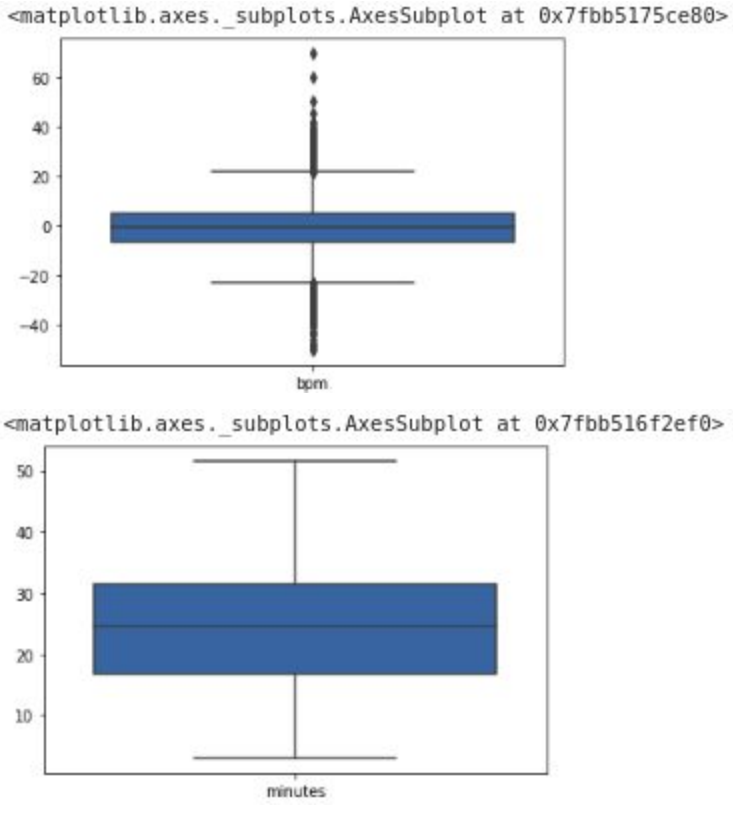
Considerem que aquests valors són valors extrems d'actuacions destacades, però no suposen una anomalia dins el que cercam, així que no modifiquem cap valor més.

Finalment, visualitzem uns boxplot per veure la distribució dels valors extrems.

Comparam els estadístics d'eficiència:

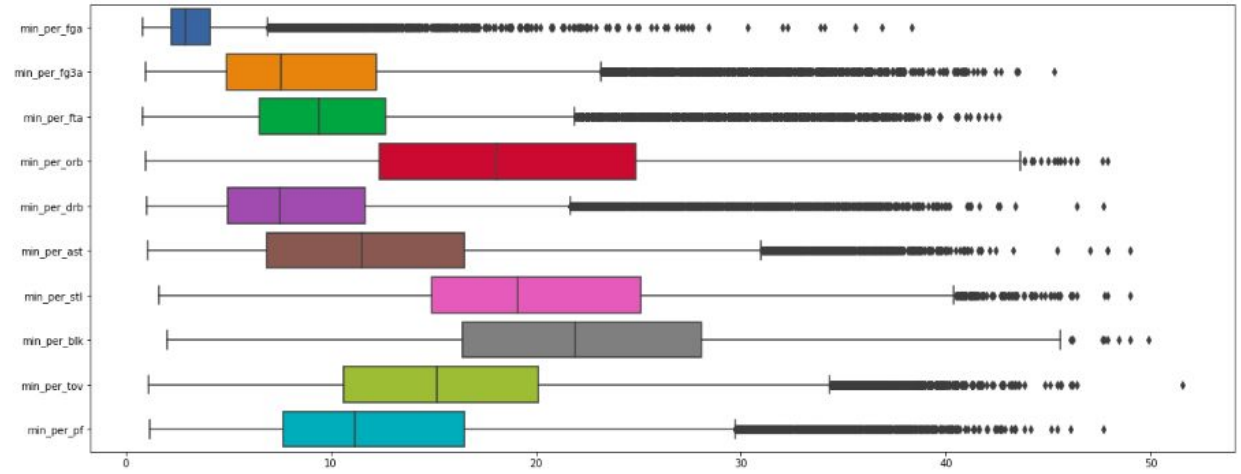


Els estadístics no poderats per minuts de joc (els propis minuts i el bpm):



I els estadístics ponderats per minut de joc:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb516a92b0>
```



4. Anàlisi de les dades.

Dividirem la secció en funció de la pregunta a la que estiguem responent a cada moment.

4.1. Distinció anotació entre local i no local? Distinció personals entre local i no local?

4.1.1. Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar)

En aquest cas ens interessen només la característica de si el jugador estava jugant de local o no (local), els estadístics de percentatge de tir (fg_pct i fg3_pct) i l'estadístic per minut de faltes personals (min_per_pf). Ens quedem amb els identificadors de jugador i equip per si surten coses interessants poder ampliar l'estudi.

```
df_local = df[['player', 'team', 'local', 'fg_pct', 'fg3_pct', 'min_per_pf']]
df_local.head()
```

	player	team	local	fg_pct	fg3_pct	min_per_pf
0	Jrue Holiday	New Orleans Pelicans	False	0.400	0.167	20.541667
1	Brandon Ingram	New Orleans Pelicans	False	0.421	0.400	8.775000
2	J.J. Redick	New Orleans Pelicans	False	0.667	0.667	9.016667
3	Lonzo Ball	New Orleans Pelicans	False	0.286	0.667	12.416667
4	Derrick Favors	New Orleans Pelicans	False	0.500	0.000	4.153333

4.1.2. Comprovació de la normalitat i homogeneïtat de la variància.

Per a comprovar la normalitat del conjunt de dades seleccionat utilitzarem Kolmogorov-Smirnov. Tot i que Saphiro-Wilk és probablement més fiable (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3693611/, *"the Shapiro-Wilk normality tests and recommends these tests only for a sample size of less than 50"*), la quantitat de mostres és massa gran i l'invalida.

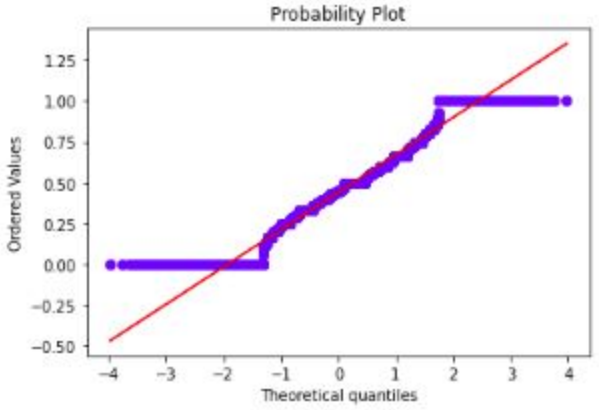
```
#Test de normalitat de Kolmogorov-Smirnov (Saphiro-Wilk no recomanat per grans volums de mostres)
loc, scale = stats.norm.fit(df_local['fg_pct'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per fg_pct')
print(stats.kstest(df_local['fg_pct'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_local['fg3_pct'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per fg3_pct')
print(stats.kstest(df_local['fg3_pct'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_local['min_per_pf'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_pf')
print(stats.kstest(df_local['min_per_pf'], n.cdf))
print('****************************************')
```

```
Resultat Kolmogorov-Smirnov per fg_pct
KstestResult(statistic=0.07558565959271557, pvalue=7.023613845316533e-97)
****************************************
Resultat Kolmogorov-Smirnov per fg3_pct
KstestResult(statistic=0.25586986698879816, pvalue=0.0)
****************************************
Resultat Kolmogorov-Smirnov per min_per_pf
KstestResult(statistic=0.11572010528189414, pvalue=1.6654482295393037e-226)
****************************************
```

Inicialment sembla que no podem acceptar la hipòtesi de normalitat per a cap de les 3 variables. Contrastam amb el Q-Q Plot.

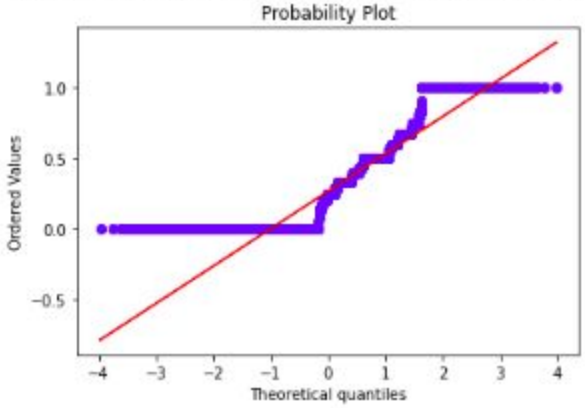
```
stats.probplot(df_local['fg_pct'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
         3.75531996,  3.9718348 ]), array([0., 0., 0., ..., 1., 1., 1.])),
 (0.22882000839305752, 0.4417292792097547, 0.9840191633271155))
```



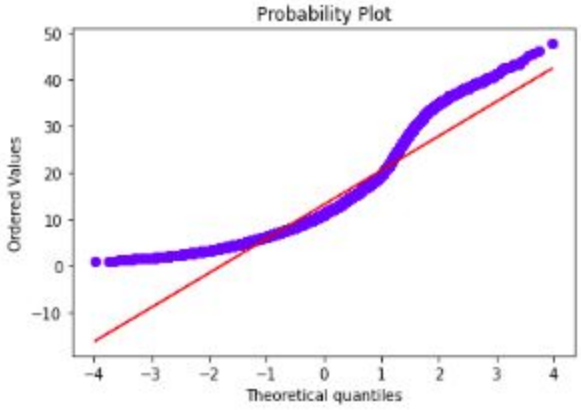
```
stats.probplot(df_local['fg3_pct'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
         3.75531996,  3.9718348 ]), array([0., 0., 0., ..., 1., 1., 1.])),
 (0.2644262433215677, 0.26646555538406147, 0.9148163884508301))
```




```
stats.probplot(df_local['min_per_pf'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
         3.75531996,  3.9718348 ]),
  array([ 1.14444444,  1.20833333,  1.225     , ..., 45.43333333,
         46.1       , 47.66666667])),
 (7.329145349672709, 13.168333500275715, 0.9429855133801653))
```



Clarament cap de les 3 variables compleix. Als percentatges de tir fg_pct podria ser que fos una normal en el fons, però es veu altament penalitzada pel biaix dels extrems (0 i 1). Com ja hem comentat, això pot afegir renou a la variable però aporta informació valuosa per a aquells jugadors que normalment juguen uns pocs minuts i no tenen un gran volum d'oportunitats de tirar a cistella cada partit.

Finalment, per revisar la homoscedasticitat aplicarem un test de Levene. Ja que les distribucions no són normals l'aplicarem sobre la mediana i per generar els dos grupos revisarem la diferència entre local i no (anticipant ja l'aplicació de proves estadístiques per a respondre la pregunta)

```
print('Resultat Levene per fg_pct en funció de si es local o no')
print(stats.levene(df_local[df_local['local'] == True]['fg_pct'],df_local[df_local['local'] == False]['fg_pct'] , center='median'))
print('****************************************')
print('Resultat Levene per fg3_pct en funció de si es local o no')
print(stats.levene(df_local[df_local['local'] == True]['fg3_pct'],df_local[df_local['local'] == False]['fg3_pct'] , center='median'))
print('****************************************')
print('Resultat Levene per min_per_pf en funció de si es local o no')
print(stats.levene(df_local[df_local['local'] == True]['min_per_pf'],df_local[df_local['local'] == False]['min_per_pf'] , center='median'))
print('****************************************')
```

```
Resultat Levene per fg_pct en funció de si es local o no
LeveneResult(statistic=0.7495997817247593, pvalue=0.3866136591858028)
****************************************
Resultat Levene per fg3_pct en funció de si es local o no
LeveneResult(statistic=6.4965142902829625, pvalue=0.010816237686177772)
****************************************
Resultat Levene per min_per_pf en funció de si es local o no
LeveneResult(statistic=2.172262708198749, pvalue=0.14053553982303876)
****************************************
```

Amb fg_pct i min_per_pf sembla que es supera el p-value de 0,05 fàcilment, amb el que podem concloure que es tracta de variances significativament diferents, no així amb el p-value mostrat per fg3_pct. Al següent apartat revisarem visualment

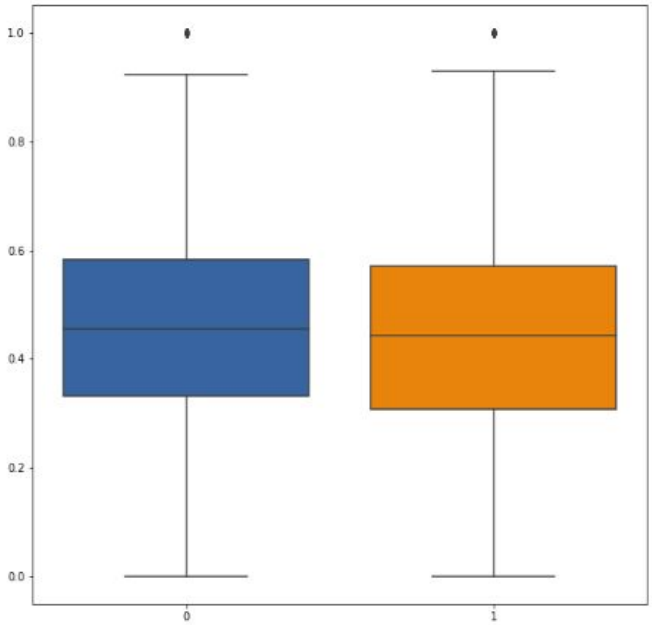
4.1.3. Aplicació de proves estadístiques per comparar els grups de dades.

Revisem visualment per respondre si els equips es comporten de manera diferent

FG_PCT

```
fig_dims = (10, 10)
fig, ax = plt.pyplot.subplots(figsize=fig_dims)
sns.boxplot(data=[df_local[df_local['local'] == True]['fg_pct'], df_local[df_local['local'] == False]['fg_pct']])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb665ea53c8>
```

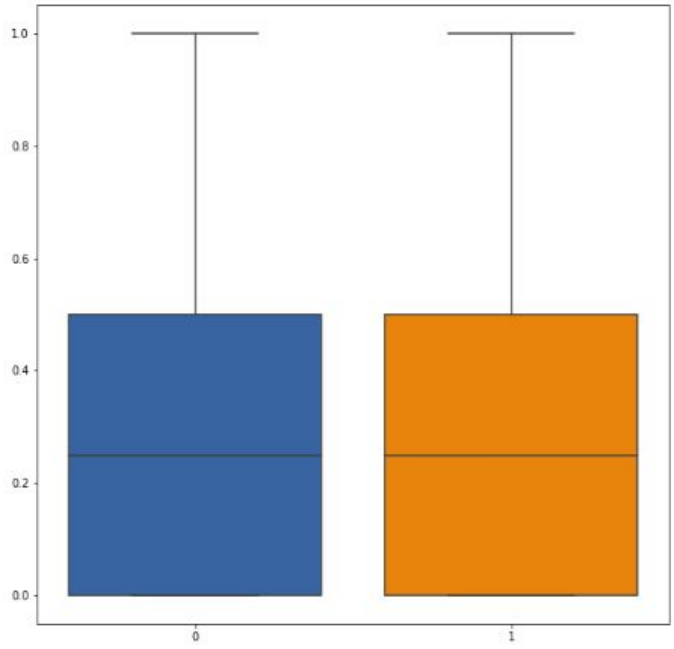


Tot i que visualment se pot observar que la distribució està lleugerament desplaçada a nivell visual, el que podria dur a explicar que els jugadors tendeixen molt lleugerament a encertar menys fora de casa que a casa, no hi ha una diferència estadísticament significativa com per a extreure una conclusió ferma.

FG3_PCT

```
fig_dims = (10, 10)
fig, ax = plt.pyplot.subplots(figsize=fig_dims)
sns.boxplot(data=[df_local[df_local['local'] == True]['fg3_pct'], df_local[df_local['local'] == False]['fg3_pct']])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb665f039e8>
```

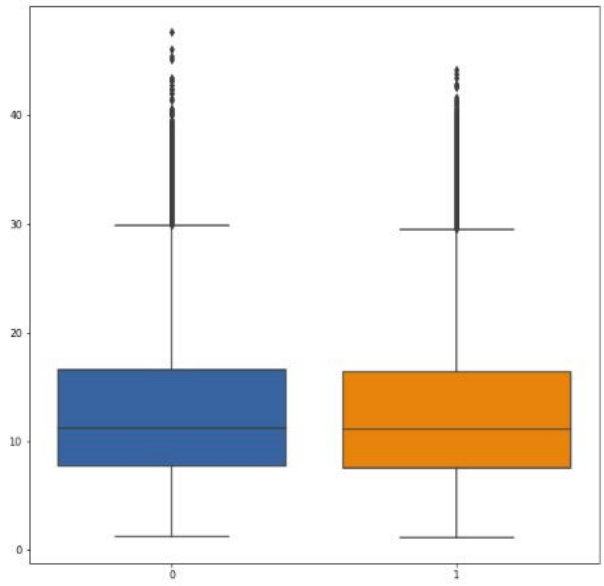


Tot i els resultats dels tests de Levene l'observació visual sembla descartar cap tipus de diferència estadísticament rellevant entre l'eficiència dels jugadors al triple als partits locals o fora del seu pabelló.

MIN_PER_PF

```
fig_dims = (10, 10)
fig, ax = plt.pyplot.subplots(figsize=fig_dims)
sns.boxplot(data=[df_local[df_local['local'] == True]['min_per_pf'], df_local[df_local['local'] == False]['min_per_pf']])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb665d98c88>
```



De manera molt semblant al fg_pct visualment s'observa una lleugera diferència entre els valors extrems (tendents a que se necessitin més minuts per a que te pitin una falta persona quan es juga com a local) i una encara més lleugera diferència entre els valors dins els marges de les variances, novament no és significativa com per a extreure cap tipus de conclusió ferma.

CONCLUSIÓ

Donat l'anàlisi podem concloure que no hi ha diferències estadísticament significatives al respecte de l'eficiència de tir (general i de triple) ni de minuts "necessaris" per a que els pitin una falta personal per als jugadors en funció de si juguen al seu pabelló o al del rival.

4.2. Correlació entre minuts de joc i estadístics bàsics per minut i entre els estadístics entre sí? Correlació entre percentatge de tir i tirs intentats (fg i fg3a)?

4.2.1. Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar)

Bàsicament en aquest cas agafarem els estadístics sobre els que volem establir si existeix alguna correlació (minuts i tots els basats en producció per minut)

```
df_corr = df[['minutes','min_per_fga','min_per_fg3a','min_per_fta','min_per_orb','min_per_drb','min_per_ast','min_per_stl','min_per_blk','min_per_tov','min_per_pf']]
df_corr.head()
```

	minutes	min_per_fga	min_per_fg3a	min_per_fta	min_per_orb	min_per_drb	min_per_ast	min_per_stl	min_per_blk	min_per_tov	min_per_pf
0	41.083333	2.738889	6.847222	20.541667	20.541667	20.541667	6.847222	19.967721	20.541667	8.216667	20.541667
1	35.100000	1.847368	7.020000	8.775000	27.951515	7.020000	7.020000	35.100000	17.550000	17.550000	8.775000
2	27.050000	3.005556	4.508333	8.906410	28.100000	13.525000	27.050000	26.100000	23.189583	9.016667	9.016667
3	24.833333	3.547619	8.277778	12.416667	24.345417	4.966667	4.966667	20.579108	30.735764	24.833333	12.416667
4	20.766667	3.461111	28.552381	15.045238	20.766667	3.461111	10.383333	23.236905	20.766667	20.766667	4.153333

4.2.2. Comprovació de la normalitat i homogeneïtat de la variància.

Novament utilitzarem Kolmogorov-Smirnov per a comprovar la normalitat del conjunt de dades seleccionat, ja que la grandària del mostreig segueix sent la mateixa.

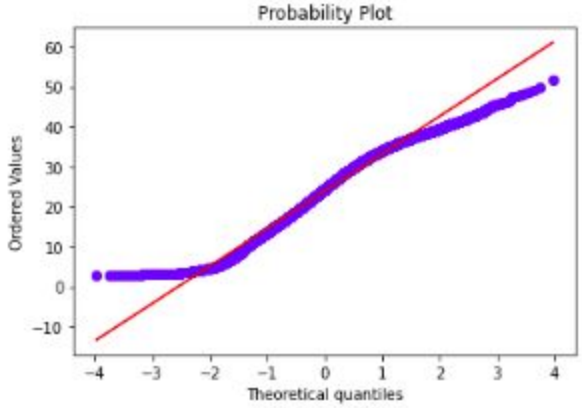
```
#Test de normalitat de Kolmogorov-Smirnov (Saphiro-Wilk no recomanat per grans volums de mostres)
loc, scale = stats.norm.fit(df_corr['minutes'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per minutes')
print(stats.kstest(df_corr['minutes'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_fga'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_fga')
print(stats.kstest(df_corr['min_per_fga'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_fg3a'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_fg3a')
print(stats.kstest(df_corr['min_per_fg3a'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_fta'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_fta')
print(stats.kstest(df_corr['min_per_fta'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_orb'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_orb')
print(stats.kstest(df_corr['min_per_orb'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_drb'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_drb')
print(stats.kstest(df_corr['min_per_drb'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_ast'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_ast')
print(stats.kstest(df_corr['min_per_ast'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_stl'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_stl')
print(stats.kstest(df_corr['min_per_stl'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_blk'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_blk')
print(stats.kstest(df_corr['min_per_blk'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_tov'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_tov')
print(stats.kstest(df_corr['min_per_tov'], n.cdf))
print('****************************************')
loc, scale = stats.norm.fit(df_corr['min_per_pf'])
n = stats.norm(loc=loc, scale=scale)
print('Resultat Kolmogorov-Smirnov per min_per_pf')
print(stats.kstest(df_corr['min_per_pf'], n.cdf))
print('****************************************')
```

```
Resultat Kolmogorov-Smirnov per minutes
KstestResult(statistic=0.047596509659171726, pvalue=1.1333105851690347e-38)
****************************************
Resultat Kolmogorov-Smirnov per min_per_fga
KstestResult(statistic=0.17357296584672577, pvalue=0.0)
****************************************
Resultat Kolmogorov-Smirnov per min_per_fg3a
KstestResult(statistic=0.16882749395483, pvalue=0.0)
****************************************
Resultat Kolmogorov-Smirnov per min_per_fta
KstestResult(statistic=0.09053959677222723, pvalue=8.052842098644129e-139)
****************************************
Resultat Kolmogorov-Smirnov per min_per_orb
KstestResult(statistic=0.04028287389031804, pvalue=8.038994472520251e-28)
****************************************
Resultat Kolmogorov-Smirnov per min_per_drb
KstestResult(statistic=0.13871615061142362, pvalue=0.0)
****************************************
Resultat Kolmogorov-Smirnov per min_per_ast
KstestResult(statistic=0.08021126655465355, pvalue=4.785272943276658e-109)
****************************************
Resultat Kolmogorov-Smirnov per min_per_stl
KstestResult(statistic=0.060543533943580696, pvalue=2.6117429901561197e-62)
****************************************
Resultat Kolmogorov-Smirnov per min_per_blk
KstestResult(statistic=0.037191140189459904, pvalue=8.894336908617832e-24)
****************************************
Resultat Kolmogorov-Smirnov per min_per_tov
KstestResult(statistic=0.07375495556837075, pvalue=2.8992275763841314e-92)
****************************************
Resultat Kolmogorov-Smirnov per min_per_pf
KstestResult(statistic=0.11572010528189414, pvalue=1.6654482295393037e-226)
****************************************
```

Igual que amb les 3 variables anteriors sembla que no podem acceptar la hipòtesi de normalitat per a cap d'aquestes variables. Novament contrastam amb el Q-Q Plot per a cada variable.

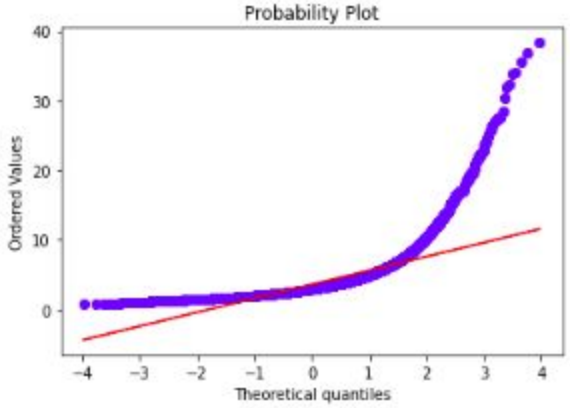
```
stats.probplot(df_corr['minutes'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 3.01666667,  3.01666667,  3.01666667, ..., 49.        ,
         49.88333333, 51.55      ])),
 (9.346834534654633, 23.880564559002593, 0.9904348460551818))
```



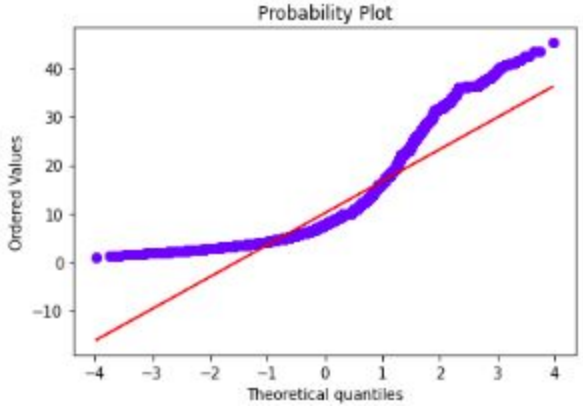
```
stats.probplot(df_corr['min_per_fga'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 0.825     ,  0.84166667,  0.85      , ..., 35.6       ,
         36.88333333, 38.35      ])),
 (2.0020117244121733, 3.586995787083429, 0.8209730702922537))
```



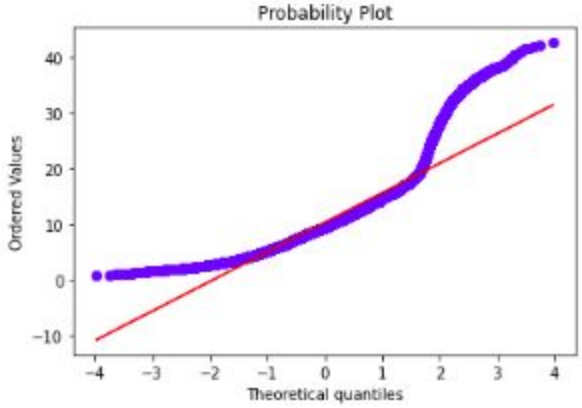

```
stats.probplot(df_corr['min_per_fg3a'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 0.96      ,  1.10555556,  1.16666667, ..., 43.45      ,
         43.51666667, 45.3       ])),
 (6.596086559680648, 9.977976210845487, 0.9056956967841939))
```



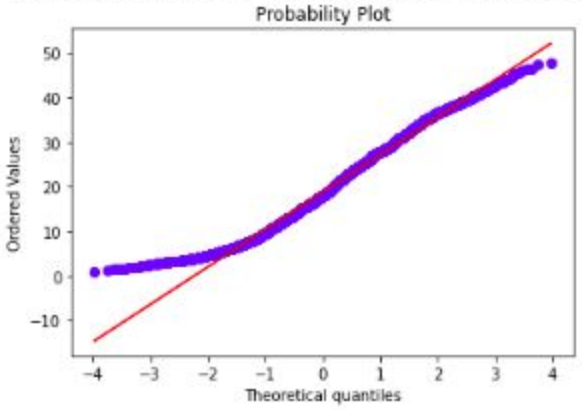
```
stats.probplot(df_corr['min_per_fta'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 0.8       ,  0.8625    ,  1.05      , ..., 41.91666667,
         42.21666667, 42.6       ])),
 (5.298828791965229, 10.326412581480824, 0.9339154253231954))
```



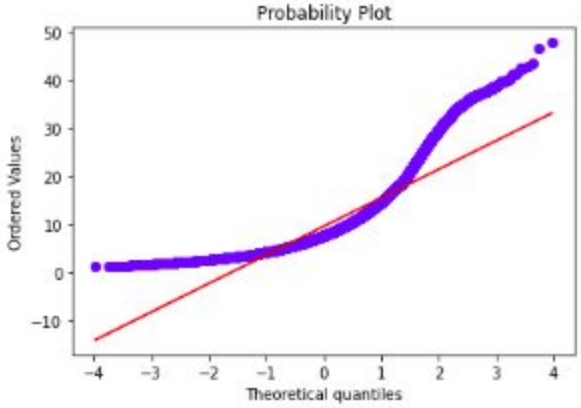

```
stats.probplot(df_corr['min_per_orb'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 0.97083333,  1.10833333,  1.47380952, ..., 46.4       ,
         47.63333333, 47.88333333])),
 (8.403227257139292, 18.787187665486154, 0.991832363018946))
```



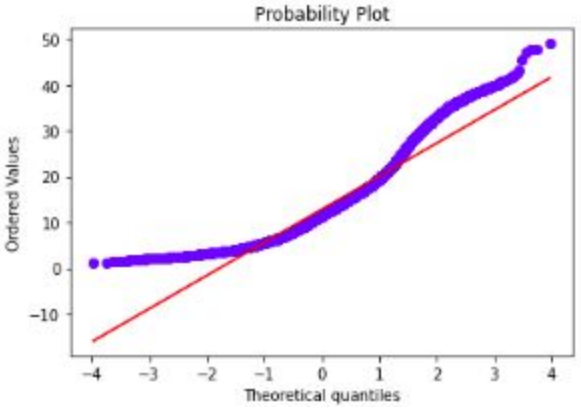
```
stats.probplot(df_corr['min_per_drb'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 1.02777778,  1.05      ,  1.06666667, ..., 43.4       ,
         46.38333333, 47.66666667])),
 (5.931047196701027, 9.427897948368821, 0.9090847309031939))
```



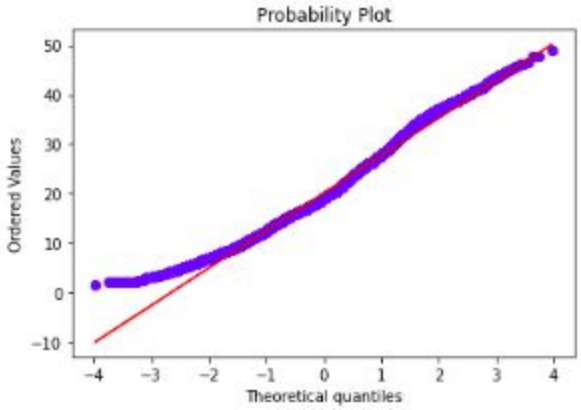

```
stats.probplot(df_corr['min_per_ast'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 1.06111111,  1.31111111,  1.42666667, ..., 47.88333333,
         47.88333333, 49.        ])),
 (7.2189473314758414, 12.785298467158652, 0.9581754754299023))
```



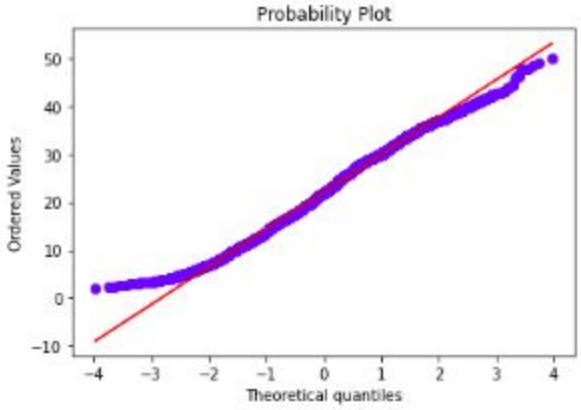
```
stats.probplot(df_corr['min_per_stl'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 1.59166667,  2.07777778,  2.125     , ..., 47.73333333,
         47.88333333, 49.        ])),
 (7.569903927853319, 20.05381920017928, 0.9919489060909458))
```



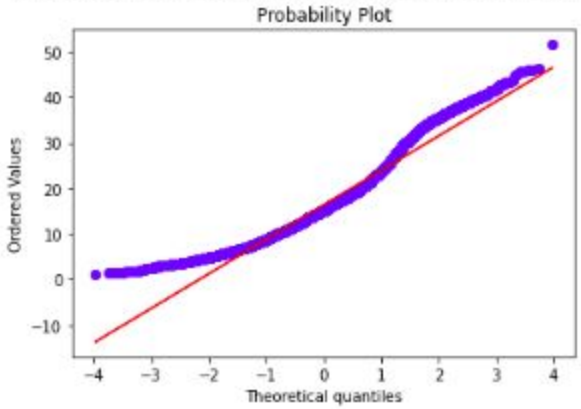

```
stats.probplot(df_corr['min_per_blk'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
          3.75531996,  3.9718348 ]),
  array([ 2.03333333,  2.28333333,  2.33333333, ..., 48.41666667,
         49.        , 49.88333333])),
 (7.795440920457514, 22.045080517539592, 0.9970109932733806))
```



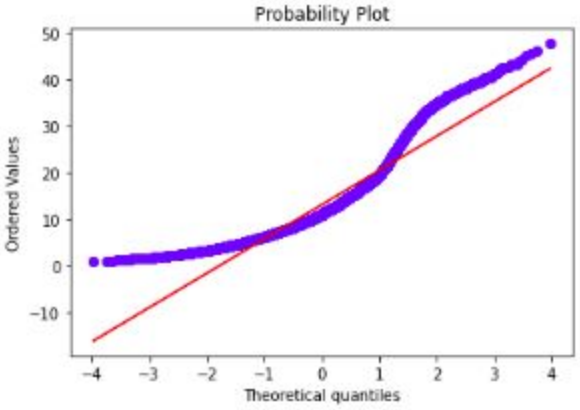
```
stats.probplot(df_corr['min_per_tov'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
         3.75531996,  3.9718348 ]),
  array([ 1.11111111,  1.57222222,  1.57916667, ..., 46.16666667,
        46.38333333, 51.55      ])),
 (7.5606258415535725, 16.35925809141769, 0.9748656880858666))
```




```
stats.probplot(df_corr['min_per_pf'], plot= plt.pyplot)
```

```
((array([-3.9718348 , -3.75531996, -3.63682924, ...,  3.63682924,
         3.75531996,  3.9718348 ]),
  array([ 1.14444444,  1.20833333,  1.225     , ..., 45.43333333,
        46.1       , 47.66666667])),
 (7.329145349672709, 13.168333500275715, 0.9429855133801653))
```



Tot i que alguna variable visualment s'acosta més a l'esperat d'una normal (min_per_orb, min_per_stl, min_per_blk) en general cap d'elles és una normal.

En quant a la homoscedasticitat no aplicarem cap test en aquest cas ja que no estem comparant distribucions iguals (ja s'aprecia visualment) ni pretenem dividir per a cada distribució les dades en sub-grups com si hem fet a l'apartat anterior sobre l'efecte local.

4.2.3. Aplicació de proves estadístiques per comparar els grups de dades.

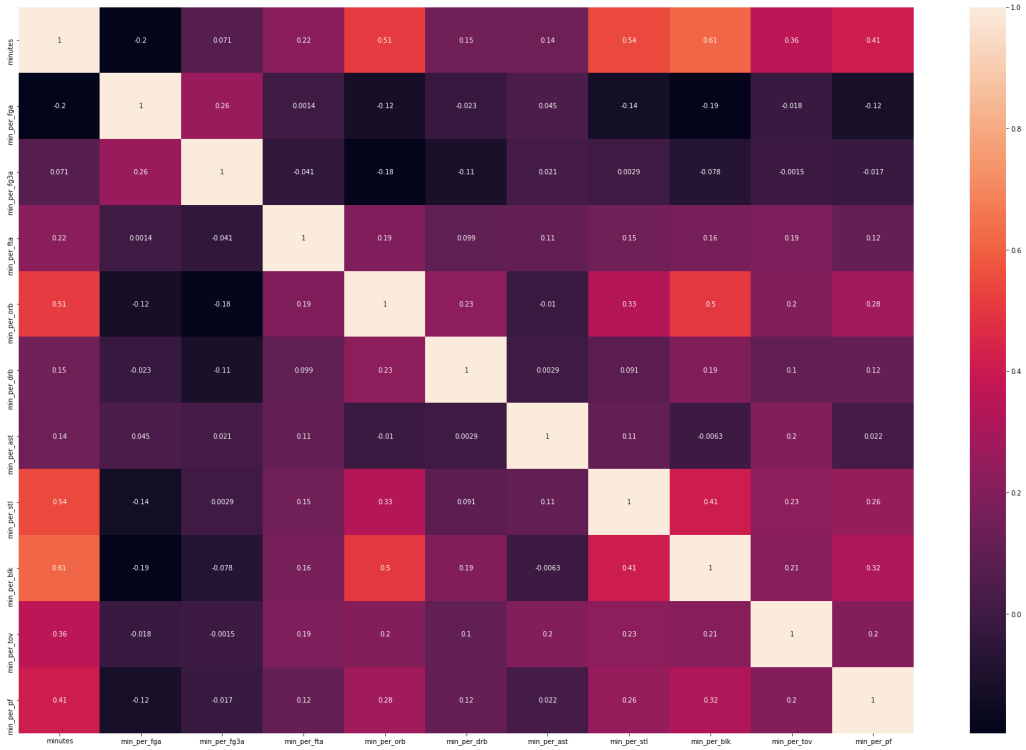
Per a revisar la possible correlació utilitzarem la matriu de correlació entre totes les variables que estem analitzant.

Calculam i visualitzam.

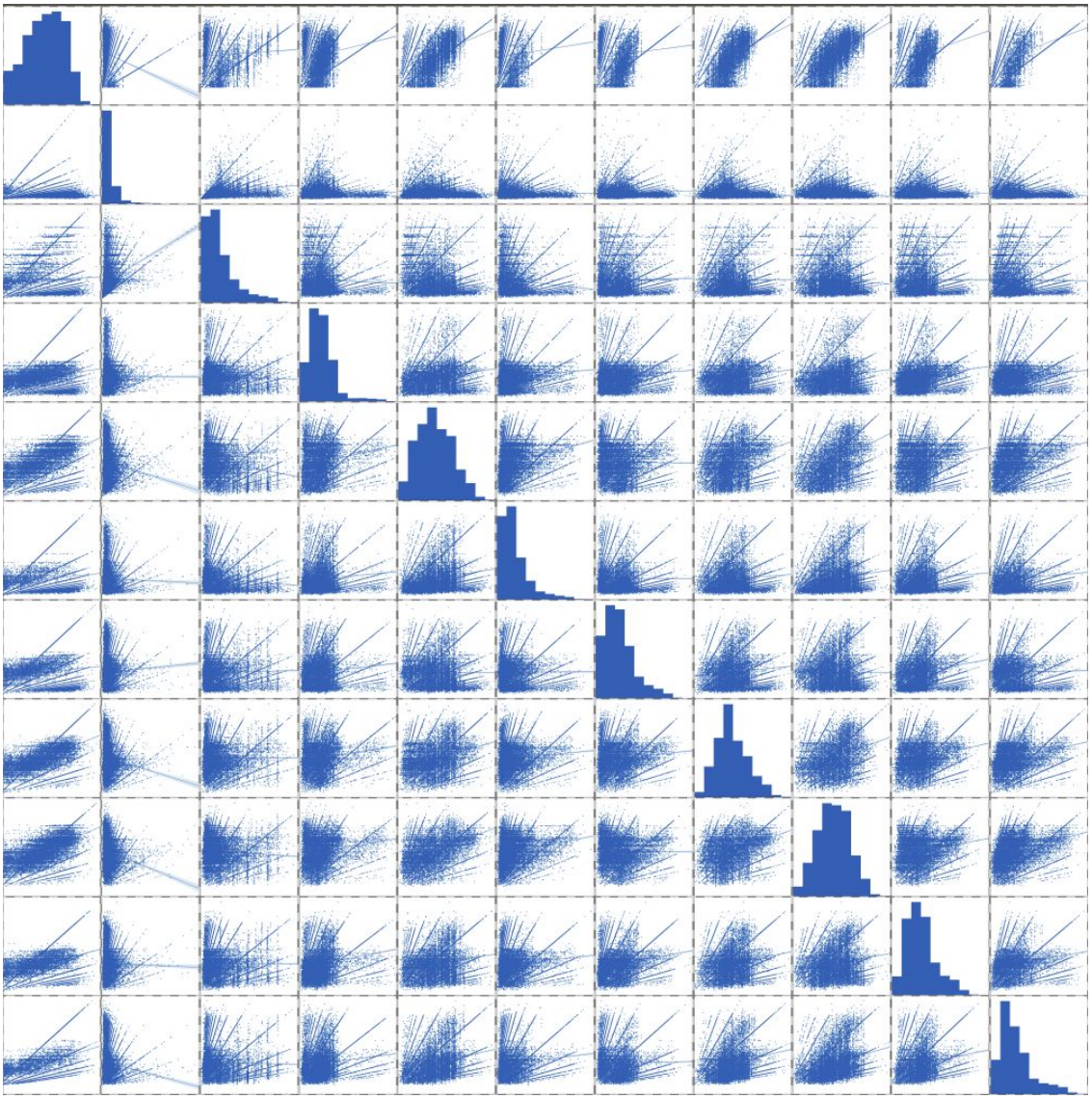
```
# Obtenir matriu de correlació
corrMatrix = df_corr.corr()
print (corrMatrix)
```

```
                minutes  min_per_fga  ...  min_per_tov  min_per_pf
minutes        1.000000    -0.196729  ...     0.355416    0.410804
min_per_fga   -0.196729     1.000000  ...    -0.017746   -0.118214
min_per_fg3a   0.071279     0.258721  ...    -0.001515   -0.017378
min_per_fta    0.218428     0.001421  ...     0.186397    0.118353
min_per_orb    0.512180    -0.123402  ...     0.203019    0.284338
min_per_drb    0.148436    -0.023424  ...     0.104972    0.117187
min_per_ast    0.135577     0.044522  ...     0.199086    0.022444
min_per_stl    0.544143    -0.135603  ...     0.229704    0.256368
min_per_blk    0.612668    -0.190538  ...     0.209463    0.322375
min_per_tov    0.355416    -0.017746  ...     1.000000    0.204067
min_per_pf     0.410804    -0.118214  ...     0.204067    1.000000

[11 rows x 11 columns]
```



Addicionalment podem fer un pairplot per veure visualment la distribució obtinguda de creuar cada variable amb l'altra. (*Nota: per problemes a l'editor de text s'ha hagut d'adjuntar una imatge no molt significativa del pairplot, al Colab se troba el codi i l'imatge sencera i amb millor resolució*).



CONCLUSIÓ

Respecte de la correlació entre minuts i els estadístics per minut observem tres grups clarament diferenciats:

- Estadístics de tir de camp (min_per_fga (-0.2), min_per_fg3a (0.071)): clarament no correlats, això indica que els jugadors tiren amb més o menys freqüència sense relació amb els minuts de joc.

- Estadístics lleument correlats amb els minuts (min_per_ast (0.14), min_per_drb (0.15), min_per_fta (0.22)): En aquests estadístics hi ha un nivell mínim de correlació. Al pairplot s'observen "línies" molt marcades dins la dispersió de punts, molt probablement relacionades amb les mitjes extretes durant la transformació a estadístics ponderats per minut de joc. Aquestes línies poden introduir un poc de renou sobre la correlació, per tant la conclusió per aquests estadístics és que no sembla que tinguem suficient informació com per a concloure que están correlats amb la quantitat de minuts de joc.

- Estadístics moderada o fortament correlats amb els minuts (min_per_tov (0.36) min_per_pf (0.41), min_per_orb (0.51), min_per_stl (0.54), min_per_blk (0.61)): En aquests estadístics la correlació és suficientment forta com per a extreure que sí que hi ha una correlació directa, és a dir, que independentment d'altres variables (rol del jugador a l'equip, nivell d'habilitat del jugador, …) la productivitat en general s'incrementa quan més temps estan a pista (a més minuts a pista, més pilotes robades per minut, més rebots ofensius per minut, …). Una possible línea d'investigació que es podria continuar a partir d'aquí és si això indica que aquests estadístics són menys depenents de la qualitat del jugador que altres, és a dir, si dos jugadors amb diferents nivells d'habilitat a pista tenen la mateixa progressió en quant a increment de produccions d'aquests estadístics per minut respecte de l'increment de temps de joc.

Així doncs sembla que no sempre els jugadors que millor tiren a cistella són els que més minuts juguen, però sí sembla que hi ha una tendència a que els jugadors que millor rendiment tenen als altres estadístics són els que més minuts juguen.

En relació a la correlació entre els estadístics en sí s'observen 4 grans tendències de correlació:

- Els estadístics de tir no semblen estar gens correlats a cap altres estadístic. És a dir, no podem inferir que els jugadors més tiradors (que no necessàriament els més eficients) siguin els millors o els pitjors a cap altre aspecte del joc. Entre les dues estadístiques de tir (min_per_fga i min_per_fg3a) sí que s'observa una lleugera correlació, el que podria donar la interpretació de que els jugadors que més tiren de dos punts en general tiren més sovint de triple (i viceversa).

- Les correlacions més fortes entre estadístics són les que relacionen justament els estadístics que tenien una correlació amb els minuts (taps, rebots ofensius i pilotes robades essencialment: min_per_blk -

min_per_orb (0.5), min_per_blk - min_per_stl (0.41), min_per_orb - min_per_stl (0.33); més lleugerament hi ha una certa correlació amb faltes personals, que també estava moderament correlada amb minuts: min_per_blk - min_per_pf (0.32)). Així doncs, abans d'extreure la conclusió que realment aquests estadístics estan relacionats i que els millors jugadors posant taps són també els millors robant pilotes, sembla que se li podria donar la interpretació de que en estar aquests estadístics fortament relacionats amb els minuts de joc, els mateixos jugadors que juguen més minuts tendran millors resultats en tots aquests estadístics simultàneament, el que donaria aquesta falsa sensació de correlació directa. S'hauria d'estudiar amb molta cura aquest aspecte abans de treure una conclusió.

- Una correlació no molt forta i que prèviament a la resolució podria semblar que hauria de ser major és entre rebots ofensius i rebots defensius (min_per_drb - min_per_orb (0.23)). Sembla de que de manera lleugera els millors jugadors agafant rebots a un costat de la pista seran també els millors jugdors per agafar-los a l'altre costat. Això podria semblar intuitivament més correlat del que diuen finalment les xifres.

- Finalment també hi ha una sèrie de correlacions bastant moderades de pèrdues de pilota amb altres estadístics: min_per_tov - min_per_stl (0.23), min_per_tov - min_per_orb (0.21), min_per_tov - min_per_blk (0.21), min_per_tov - min_per_ast (0.20), min_per_tov - min_per_pf (0.20). És relativament lleu per treure unes conclusions fermes

4.3. Predicció de minuts de joc al pròxim partit d'un jugador? Predicció d'estadístics de joc en funció dels minuts?

4.3.1. Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar)

Per contestar les dues preguntes se requereixen dos jocs de dades diferents. Per un costat intentarem respondre a la predicció de minuts de joc mitjançant una anàlisi de sèries temporals, amb el que necessitam bàsicament és la sèrie temporal (a partir de la data de partit), l'identificador de jugador + equip i els valors a analitzar, és a dir, els minuts.

```
#Predicció minuts
df_minutes = df[['date', 'team', 'player', 'minutes']]
df_minutes.head()
```

	date	team	player	minutes
0	2019-10-22	New Orleans Pelicans	Jrue Holiday	41.083333
1	2019-10-22	New Orleans Pelicans	Brandon Ingram	35.100000
2	2019-10-22	New Orleans Pelicans	J.J. Redick	27.050000
3	2019-10-22	New Orleans Pelicans	Lonzo Ball	24.833333
4	2019-10-22	New Orleans Pelicans	Derrick Favors	20.766667

Per altra banda, per a la predicció d'estadístics farem la prova amb els min_per_fga, amb el que necessitarem inicialment les variables que poden tenir afectació sobre min_per_fga (jugador, equip, rival, local, minuts de joc) i la classe a predir: els min_per_fga.

```
# Predicció estadistics
df_min_per_fga = df [['player','team','against','local','minutes','min_per_fga']]
df_min_per_fga.head()
```

	player	team	against	local	minutes	min_per_fga
0	Jrue Holiday	New Orleans Pelicans	Toronto Raptors	False	41.083333	2.738889
1	Brandon Ingram	New Orleans Pelicans	Toronto Raptors	False	35.100000	1.847368
2	J.J. Redick	New Orleans Pelicans	Toronto Raptors	False	27.050000	3.005556
3	Lonzo Ball	New Orleans Pelicans	Toronto Raptors	False	24.833333	3.547619
4	Derrick Favors	New Orleans Pelicans	Toronto Raptors	False	20.766667	3.461111

En aquest cas haurem de fer un tractament addicional sobre minutes, ja que no ens interessa mantenir un rang continuu sobre aquesta variable, així que cream 20 categories de minuts en funció dels percentils.

```
df_min_per_fga['minutes'] = pd.qcut(df_min_per_fga['minutes'], q=20)
df_min_per_fga.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

	player	team	against	local	minutes	min_per_fga
0	Jrue Holiday	New Orleans Pelicans	Toronto Raptors	False	(37.783, 51.55]	2.738889
1	Brandon Ingram	New Orleans Pelicans	Toronto Raptors	False	(34.35, 35.833]	1.847368
2	J.J. Redick	New Orleans Pelicans	Toronto Raptors	False	(25.9, 27.317]	3.005556
3	Lonzo Ball	New Orleans Pelicans	Toronto Raptors	False	(24.5, 25.9]	3.547619
4	Derrick Favors	New Orleans Pelicans	Toronto Raptors	False	(20.05, 21.583]	3.461111

Revisam visualment els grups que han sortit per comprovar que tenen un mínim de sentit

```
# Valors únics per minutes
df_min_per_fga['minutes'].unique()
```

```
[(37.783, 51.55], (34.35, 35.833], (25.9, 27.317], (24.5, 25.9], (20.05, 21.583], ..., (6.75, 10.767], (28.717, 30.1], (32.883, 34.35], (21.583, 23.033], (3.016, 6.75]]
Length: 20
Categories (20, interval[float64]): [(3.016, 6.75] < (6.75, 10.767] < (10.767, 13.217] <
                                     (13.217, 15.183] ... (32.883, 34.35] < (34.35, 35.833] <
                                     (35.833, 37.783] < (37.783, 51.55]]
```

Finalment, dividim el dataset en les “features” i la classe.

```
X = df_min_per_fga.iloc[:,0:-1]
X.head()
```

	player	team	against	local	minutes
0	Jrue Holiday	New Orleans Pelicans	Toronto Raptors	False	(37.783, 51.55]
1	Brandon Ingram	New Orleans Pelicans	Toronto Raptors	False	(34.35, 35.833]
2	J.J. Redick	New Orleans Pelicans	Toronto Raptors	False	(25.9, 27.317]
3	Lonzo Ball	New Orleans Pelicans	Toronto Raptors	False	(24.5, 25.9]
4	Derrick Favors	New Orleans Pelicans	Toronto Raptors	False	(20.05, 21.583]

```
y = df_min_per_fga.iloc[:,-1:]
y.head()
```

	min_per_fga
0	2.738889
1	1.847368
2	3.005556
3	3.547619
4	3.461111

4.3.2. Comprovació de la normalitat i homogeneïtat de la variància.

En aquest apartat no farem cap comprovació, a que a les preguntes anteriors (4.1.2 i 4.2.2) ja varem comprovar l’absència de normalitat a les distribucions de les variables involucrades.

4.3.3. Aplicació de proves estadístiques per comparar els grups de dades.

En primer lloc montarem el model de predicció de l'estadístic. Per a l'efecte utilitzarem un RandomForestRegressor. Així la primera passa seria adaptar les variables d'entrada per a que el RandomForestRegressor les pugui emprar.

Prrimerament utilitzarem un Label Encoder per a transformar les variables de local i minutes a valors sencers.

```
#Label Encoder
le_local = LabelEncoder()
le_minutes = LabelEncoder()

X['local'] = le_local.fit_transform(X['local'])
X['minutes'] = le_minutes.fit_transform(X['minutes'])

X.head()
```

	player	team	against	local	minutes
0	Jrue Holiday	New Orleans Pelicans	Toronto Raptors	0	19
1	Brandon Ingram	New Orleans Pelicans	Toronto Raptors	0	17
2	J.J. Redick	New Orleans Pelicans	Toronto Raptors	0	11
3	Lonzo Ball	New Orleans Pelicans	Toronto Raptors	0	10
4	Derrick Favors	New Orleans Pelicans	Toronto Raptors	0	7

Per a les variables categòriques farem un One Hot Enconding, és a dir, crearem una variable nova per a cada valor de la variable indicant amb valors binaris de 0 o 1 si aquesta variable existeix amb aquest valor.

```
#OneHotEncoding
X = pd.get_dummies(X, columns=['player','team','against'])
X.head()
```

	local	minutes	player_Aaron Gordon	player_Aaron Holiday	player_Abdel Nader	player_Adam Mokoka	player_Admiral Schofield	player_Al Horford	player_Al-Farouq Aminu	player_Alec Burks	player_Alen Smailagic	player_Alex Caruso	player_Alex Len	player_Alfonzo McKinnie	player_Alize Johnson	player_Allen Crabbe	player_Allonzo Trier	player_Amile Jefferson	player_Amir Coffey	player_Andre Drummond	player_Andre Iguodala	p
0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Ara separam els jocs de dades entre les que reservarem per a l'entrenament i les que reservarem per a provar. Idealment aquesta separació hauria de tenir en compte la distribució de les dades, amb tal que hagués presència de totes les possibles "features" tant a l'entrenament com a les proves, en una proporció semblant al conjunt sencer de dades. En aquest cas no és possible per les poca quantitat de mostres que tenim, amb el que ens haurem de conformar amb el

que l'atzar disposi (un altra solució hagués estat generar mostres sintètiques per ampliar aquest volum de mostres).

```
# Separam sets d'entrenament i prova (poques mostres per estratificar)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Ara entrenam el model de RandomForestRegressor.

```
# Entrenam el model
RandomForestRegressorModel = RandomForestRegressor()
RandomForestRegressorModel.fit(X_train, y_train)
```

Comprovam la fiabilitat del model llençant el joc de proves i comprovant l'arrel de l'error quadràtic mig.

```
# Comprovam la fiabilitat del model
y_pred = RandomForestRegressorModel.predict(X_test)
```

```
# Comprovació de l'error quadràtic mig
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
rmse
```

```
2.3894303768818865
```

Sense ser un error menyspreable, tampoc sembla una xifra escandalosa per a ser una primera aproximació. Evidentment enriquint aquest model amb més característiques d'entrada rellevants i optimizant la parametrització probablement podem millorar aquest resultat.

Donat aquest resultat intentarem seleccionar una predicció de minuts i estadístic. Seleccionam el que hagués estat un partit que es jugués a continuació dels darrers partits que tenim al joc de dades (que arriba fins abans que s'aturés la NBA pel covid-19), un Houston Rockets - Los Angeles Lakers del 12 de Març. De tots els jugadors d'aquests equips agafam un d'ells, Russell Westbrook i intentarem obtenir una predicció dels minuts de joc d'aquest jugador a aquest partit i, en funció d'aquests minuts, quants de minuts per tir intentat hagués tingut aquest jugador a aquest partit.

Per a obtenir la predicció de minuts jugats emprarem Prophet, que es l'eina de Facebook per a predicció de sèries temporals.

```
df_prophet = df_minutes[df_minutes['player'] == 'Russell Westbrook'][['minutes', 'date']].copy()
df_prophet = df_prophet.rename(columns={"date": "ds", "minutes": "y"})
minutes_model = Prophet(interval_width=0.95)
minutes_model.fit(df_prophet)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
<fbprophet.forecaster.Prophet at 0x7fb661851fd0>
```

```
minutes_forecast = minutes_model.make_future_dataframe(periods=30, freq='d')
minutes_forecast = minutes_model.predict(minutes_forecast)
```

Amb això tindrem una predicció del que serien els minuts que jugaria Russel Westbrook en cas de jugar els propers dies.

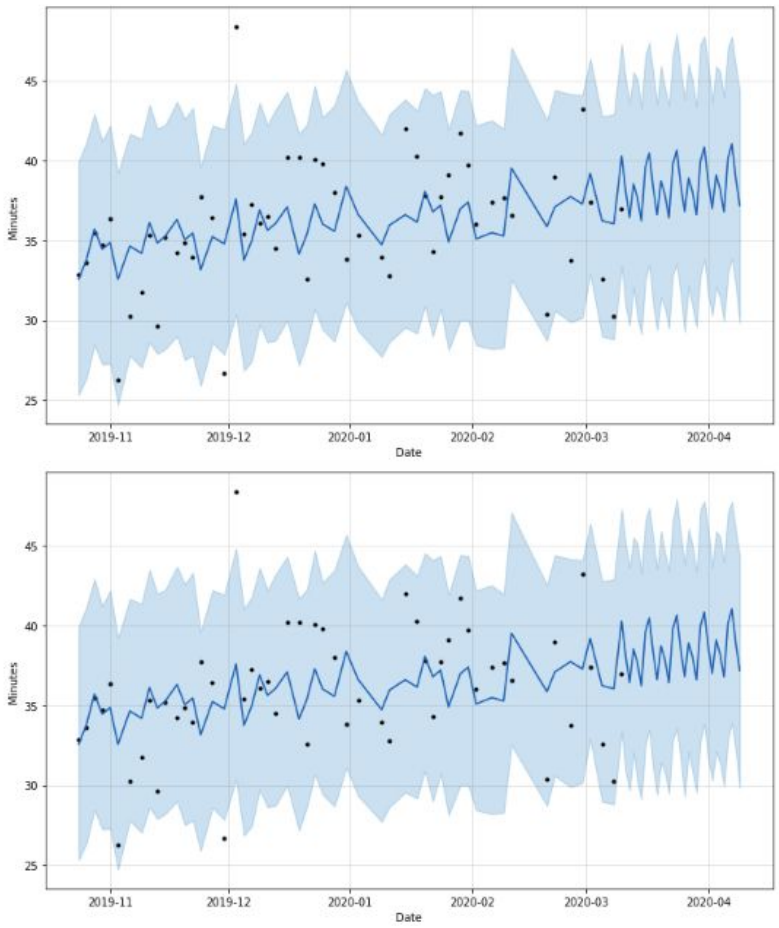
```
print(minutes_forecast)
```

```
           ds      trend  ...  multiplicative_terms_upper       yhat
0  2019-10-24  34.226729  ...                         0.0  32.566088
1  2019-10-26  34.283282  ...                         0.0  33.797444
2  2019-10-28  34.339834  ...                         0.0  35.717805
3  2019-10-30  34.396387  ...                         0.0  34.451996
4  2019-11-01  34.452939  ...                         0.0  34.867872
..        ...        ...  ...                         ...        ...
78 2020-04-05  38.745105  ...                         0.0  36.799036
79 2020-04-06  38.772147  ...                         0.0  40.150118
80 2020-04-07  38.799188  ...                         0.0  41.043224
81 2020-04-08  38.826230  ...                         0.0  38.881839
82 2020-04-09  38.853272  ...                         0.0  37.192631

[83 rows x 16 columns]
```

Validam visualment si té sentit aquesta predicció.

```
minutes_model.plot(minutes_forecast, xlabel = 'Date', ylabel = 'Minutes')
```



Sembla que la tendència positiva de minuts d'aquest jugador ha estat ben capturada per part del model, tot i que genera una estacionalitat setmanal que probablement és un poc falsa. Podem mirar d'optimitzar aquest resultat, però les prediccions a llarg plaç no són tan rellevants, donat que no hi ha tanta diferència de dies entre partits.

Així dons, revisam quina és la predicció per a dia 12 de Març.

```
minutes_forecast[minutes_forecast['ds'] == '2020-03-12']
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_lower	weekly_upper	multiplicative_terms	multiplicative_terms_lower	multiplicative_terms_upper	yhat
54	2020-03-12	38.096109	29.680584	43.652903	38.096011	38.096215	-1.660641	-1.660641	-1.660641	-1.660641	-1.660641	-1.660641	0.0	0.0	0.0	36.435468

Obtenim 36.43 minuts, que es correspondria amb el grup 19 generat pel LabelEncoder.

Així preparam la mostra que volem predir per a Russel Westbrook, jugant una quantitat de minuts englobada al quantil 19 de 20, a Houston Rockets contra Los Angeles Lakers com a equip visitant. La resta de valors queda a 0.

```
prediction_russell = pd.DataFrame(data=None, columns=X_train.columns)
prediction_russell = prediction_russell.append(pd.Series(0, index=prediction_russell.columns), ignore_index=True)
prediction_russell['minutes'] = 19
prediction_russell['player_Russell Westbrook'] = 1
prediction_russell['team_Houston Rockets'] = 1
prediction_russell['against_Los Angeles Lakers'] = 1
prediction_russell.head()
```

	local	minutes	player_Aaron Gordon	player_Aaron Holiday	player_Abdel Nader	player_Adam Mokoka	player_Admiral Schofield	player_Al Horford	player_Al-Farouq Aminu	player_Alec Burks	player_Alen Smailagic	player_Alex Caruso	player_Alex Len	player_Alfonzo McKinnie	player_Alize Johnson	player_Allen Crabbe	player_Allonzo Trier	player_Amile Jefferson	player_Amir Coffey	player_Andre Drummond	player_Andre Iguodala	pl
0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

1 rows × 506 columns

Fem la predicció.

```
print(RandomForestRegressorModel.predict(prediction_russell))
```

```
[1.45265446]
```

Abans de donar per vàlida la predicció anem a veure si té sentit.

```
df_min_per_fga[df_min_per_fga['player'] == 'Russell Westbrook']
```

7558	Russell Westbrook	Houston Rockets	Cleveland Cavaliers	False	(35.833, 37.783]	1.825833
7887	Russell Westbrook	Houston Rockets	Orlando Magic	False	(34.35, 35.833]	1.918519
8401	Russell Westbrook	Houston Rockets	San Antonio Spurs	True	(37.783, 51.55]	1.609333
8858	Russell Westbrook	Houston Rockets	Los Angeles Clippers	False	(37.783, 51.55]	1.297849
9245	Russell Westbrook	Houston Rockets	Phoenix Suns	False	(31.483, 32.883]	1.551587
9583	Russell Westbrook	Houston Rockets	Sacramento Kings	False	(37.783, 51.55]	1.822727
9667	Russell Westbrook	Houston Rockets	Golden State Warriors	False	(37.783, 51.55]	1.244271
10094	Russell Westbrook	Houston Rockets	Brooklyn Nets	True	(37.783, 51.55]	1.357738
10612	Russell Westbrook	Houston Rockets	Denver Nuggets	True	(32.883, 34.35]	1.537121
11031	Russell Westbrook	Houston Rockets	Philadelphia 76ers	True	(34.35, 35.833]	1.606061
11993	Russell Westbrook	Houston Rockets	Oklahoma City Thunder	False	(32.883, 34.35]	1.305769
12248	Russell Westbrook	Houston Rockets	Minnesota Timberwolves	True	(31.483, 32.883]	1.424638
12979	Russell Westbrook	Houston Rockets	Portland Trail Blazers	True	(37.783, 51.55]	1.908333
13458	Russell Westbrook	Houston Rockets	Los Angeles Lakers	True	(37.783, 51.55]	1.752899
13667	Russell Westbrook	Houston Rockets	Oklahoma City Thunder	True	(37.783, 51.55]	1.575000
14048	Russell Westbrook	Houston Rockets	Denver Nuggets	True	(32.883, 34.35]	1.373333
14317	Russell Westbrook	Houston Rockets	Minnesota Timberwolves	False	(35.833, 37.783]	1.397531
14521	Russell Westbrook	Houston Rockets	Denver Nuggets	False	(37.783, 51.55]	1.348851
15076	Russell Westbrook	Houston Rockets	Portland Trail Blazers	False	(37.783, 51.55]	1.439080
15298	Russell Westbrook	Houston Rockets	Dallas Mavericks	True	(37.783, 51.55]	1.417857
15641	Russell Westbrook	Houston Rockets	New Orleans Pelicans	True	(35.833, 37.783]	1.501389
16252	Russell Westbrook	Houston Rockets	Los Angeles Lakers	False	(35.833, 37.783]	1.335119
16721	Russell Westbrook	Houston Rockets	Utah Jazz	True	(35.833, 37.783]	1.142424
17055	Russell Westbrook	Houston Rockets	Boston Celtics	True	(35.833, 37.783]	1.591304
17435	Russell Westbrook	Houston Rockets	Golden State Warriors	False	(30.1, 31.483]	1.598246
17779	Russell Westbrook	Houston Rockets	Utah Jazz	False	(37.783, 51.55]	1.498718
18379	Russell Westbrook	Houston Rockets	Memphis Grizzlies	True	(32.883, 34.35]	1.406250
18846	Russell Westbrook	Houston Rockets	Boston Celtics	False	(37.783, 51.55]	1.601852
19067	Russell Westbrook	Houston Rockets	New York Knicks	False	(35.833, 37.783]	1.966667
19580	Russell Westbrook	Houston Rockets	Los Angeles Clippers	True	(31.483, 32.883]	1.207407
20115	Russell Westbrook	Houston Rockets	Orlando Magic	True	(30.1, 31.483]	1.592105
20297	Russell Westbrook	Houston Rockets	Minnesota Timberwolves	True	(35.833, 37.783]	2.175490

És un valor que té sentit. dins els valors habituals de Russel Westbrook.

CONCLUSIÓ

Tot i que el model generat dista molt de ser un model que pugui ser tractat com a definitiu per utilitzar per realitzar prediccions totalment fiables o que, per exemple, es pugui utilitzar per fer apostes esportives, sí que aparentment valida que, amb una quantitat inicial suficient de dades (és a dir, no el primer dia de la temporada), és factible predir amb uns marges d'error no molt escandalosos els minuts de joc que ha de tenir un jugador a un partit donat i els estadístics per minut d'aquest jugador al partit.

5. Representació dels resultats a partir de taules i gràfiques.

La representació dels resultats visualment s'ha anat adjuntant als diferents apartats 4.1.3, 4.2.3 i 4.3.3.

6. Resolució del problema. A partir dels resultats obtinguts, quines són les conclusions? Els resultats permeten respondre al problema?

La resolució del problema i les conclusions s'ha anat adjuntant als diferents apartats 4.1.3, 4.2.3 i 4.3.3.

7. Codi: Cal adjuntar el codi, preferiblement en R, amb el que s'ha realitzat la neteja, anàlisi i representació de les dades. Si ho preferiu, també podeu treballar en Python.

El codi se troba a un document de Google Colab: https://colab.research.google.com/drive/1I2QkLI3q7jJ224y7YBGuwNQ34suyur2z?usp=sharing