Pràctica 3: Chess FIDE Ratings (2015 - 2021)

github kaggle

Aquest dataset conte el rating en diferents disciplines competitives dels escacs, com ara l'estandard, que és la més jugada, on el temps inicial per cada jugador és de 90 minuts; les ràpides, on el temps al inici és de 15 minuts; i les blitz, que són les parides ràpides, on es comença amb 3 minuts.

Així, partim de les dades de les persones registrades en la FIDE (Federació Internacional de Escacs) des de el seu inici fins l'any 2021, on tenim l'elo cada mes.

In [1]:
```
import pandas as pd
import numpy as np

rating_2015 = pd.read_csv(f"data/ratings_2015.csv", header=0)
rating_2016 = pd.read_csv(f"data/ratings_2016.csv", header=0)
rating_2017 = pd.read_csv(f"data/ratings_2017.csv", header=0)
rating_2018 = pd.read_csv(f"data/ratings_2018.csv", header=0)
rating_2019 = pd.read_csv(f"data/ratings_2019.csv", header=0)
rating_2020 = pd.read_csv(f"data/ratings_2020.csv", header=0)
rating_2021 = pd.read_csv(f"data/ratings_2021.csv", header=0)

ratings = pd.concat([rating_2015, rating_2016, rating_2017, rating_2018, rating_2019, rating_202

players = pd.read_csv("data/players.csv")
players = players.drop(["name"], axis=1)
```

1 Anàlisi de les dades

Podem veure que dins dels ratings, tenim els atributs de:

- fide_id: on cada persona té un identificador únic que el relaciona amb la taula de players.
- year and month: mes i any dels ratings.
- rating_standard: rating de la modalitat estandard.
- rating_rapid: rating de la modalitat ràpida.
- rating_blitz: rating de la modalitat blitz.

In [2]:
```
pd.set_option('display.float_format', lambda x: '%.2f' % x)
ratings.describe()
```

Out[2]:

	fide_id	year	month	rating_standard	rating_rapid	rating_blitz
count	25428233.00	25428233.00	25428233.00	22440674.00	9511956.00	7076020.00
mean	15698833.99	2018.08	6.33	1718.53	1652.11	1720.69
std	33940467.89	1.79	3.50	341.14	344.36	339.56
min	100013.00	2015.00	1.00	1001.00	1000.00	1000.00
25%	2117894.00	2017.00	3.00	1463.00	1380.00	1467.00
50%	8604282.00	2018.00	6.00	1740.00	1650.00	1733.00
75%	24189464.00	2020.00	9.00	1983.00	1904.00	1968.00
max	651081830.00	2021.00	12.00	2882.00	2919.00	2986.00

Com podem veure, el dataset de ratings és força gran, ja que compta amb 25.5M de mostres. També és notabla la popularitat entre les diferents modalitats: estandard té 22.5M, ràpida 9.5M i blitz 7M. Això implica què de les mostres que tenim, en les modalitats de ràpides i blitzs molt més de la meitat no en tenen cap valor.

Per altre banda, comptem amb els atributs de les persones registrades:

- fide_id: on cada persona té un identificador únic que el relaciona amb la taula de ratings.
- federation: el país de procedencia.
- gender: gènere de la persona (dona o home)
- title: títol dins dels escacs, on hi ha 13 diferents tot i que la seva majoria són NaN ja que has de estar en una posició alta dins dels elos per tenir-ne un.
- yob: any de naixement.

In [3]:
```
players.describe(include='all')
```

Out[3]:

	fide_id	federation	gender	title	yob
count	433388.00	433388	433388	19498	433388.00
unique	NaN	193	2	13	NaN
top	NaN	RUS	M	FM	NaN
freq	NaN	49603	388983	8245	NaN
mean	19793306.04	NaN	NaN	NaN	1940.03
std	51381915.93	NaN	NaN	NaN	290.26
min	100013.00	NaN	NaN	NaN	0.00
25%	2510166.00	NaN	NaN	NaN	1966.00
50%	12327081.00	NaN	NaN	NaN	1986.00
75%	25009710.75	NaN	NaN	NaN	2001.00
max	651081830.00	NaN	NaN	NaN	2015.00

Podem observar ja de primeres que hi ha unes 433k persones registrades. També veiem què hi ha un gran desvalanceig entre les dones i els homes dins de la FIDE, amb un total de 388k d'homes versus 44k què són números molt allunyats l'un del altre. Dels anys de naixement podem veure que hi ha gent que no té l'any de neixement registrat ja que és mostra el mínim en 0 i que la persona més jove és del 2015, sent així que

tenia 6 anys a finals del 2021. També veiem dominancia en Russia, que és el país amb més representació dins dels 193 que hi ha.

Països

Com ja hem mencionat, Russia presenta una majoria del 7.16% de representació dins la FIDE. Això és una senyal de la popularitat que al llarg de la història a tingut dins del país. Durant el segle passat, és va tornar un passatemps impresindible per als russos i aquests van fer les primeres olimpiades dels escacs l'any 1920, gunyant encara més interes per la població.

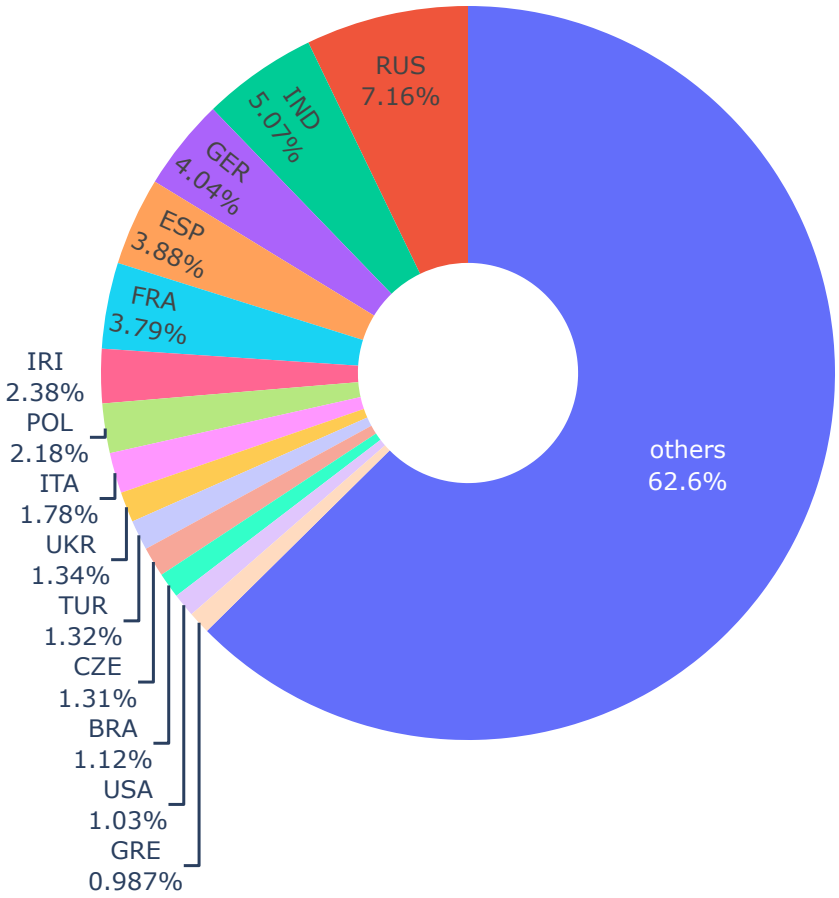
Durant els anys és va mantenir una gran dominancia del país respecte els altres, on els grans mestres dels escacs van tenir noms de russos. Tot hi que actualment ja no encapçalen el top mundial, segueix havent-hi una gran popularitat dins del país.

Seguint els persentages, segeuix la India, Alemanya, Espanya i França.

In [4]:
```
import plotly.graph_objs as go
from plotly.offline import iplot

labels_federation, count_federation = np.unique(players['federation'], return_counts=True)

print(f"Hi ha un total de {len(labels_federation)} paisos")

count_federation, labels_federation = zip(*sorted(zip(count_federation, labels_federation)))
count_federation = np.flip(count_federation)
labels_federation = np.flip(labels_federation)

count_federation_other = count_federation[:14]
count_federation_other = np.append(count_federation_other, np.sum(count_federation, initial=14))

labels_federation_other = labels_federation[:14]
labels_federation_other = np.append(labels_federation_other, "others")

# plot
trace = go.Pie(labels=labels_federation_other,
               values=count_federation_other,
               textinfo='label+percent',
               hole=0.3)

data_fed = [trace]

figure = go.Figure(data_fed)
figure.update_layout(title='distribució dels països', width=800, height=550)

iplot(figure)
```

```
Hi ha un total de 193 paisos
```

distribució dels països

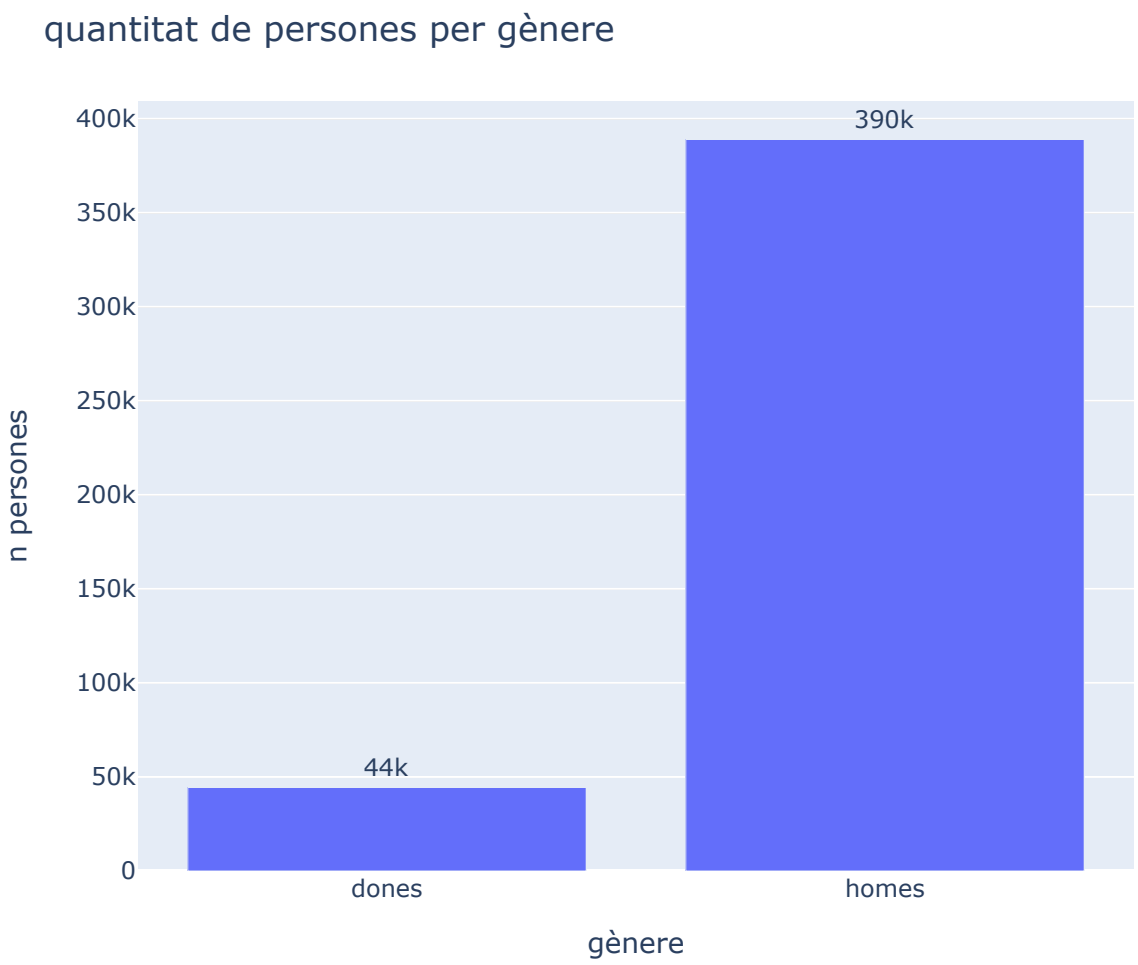


Gèneres

Als escacs és coneguda la falta de dones dins d'aquest. Per les dades que tenim, estem parlant que 1 de cada 10 persones registrades a la fide són dones. Això deu estar causat per motius socials (que no pas intel·lectuals com creu molta gent). La falta de referents, la hostilitat al ser un món d'homes o el fet de creure no ser suficients són possibles explicacions de la falta d'interés dins el món femení per adentrar-se en els escacs.

In [5]:
```
import plotly.express as px

labels_gender, count_gender = np.unique(players['gender'], return_counts=True)

print (f"El percentage de dones és de {count_gender[0] / np.sum(count_gender) * 100}% envers del

# plot
fig = px.bar(y=count_gender, x=['dones', 'homes'], text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_layout(title="quantitat de persones per gènere",
                  xaxis=dict(title="gènere"),
                  yaxis=dict(title="n persones"))
fig.show()
```

```
El percentage de dones és de 10.246015118092794% envers del de homes que és de 89.7539848819072%
```



Primers registres a la FIDE

Com la nostra base de dades comença al 2015, però ja hi havia gent que jugava (obviament), aquest any és el que més registres té. Els següents podem considerar que seria la distribució normal del que s'hauria d'esperar cada any.

Des del 2016 fins el 2019 veiem una tendència a que és registrin unes 43k-45k, però curiosament els anys 2020 i 2021 no segeixen aquest ritme.

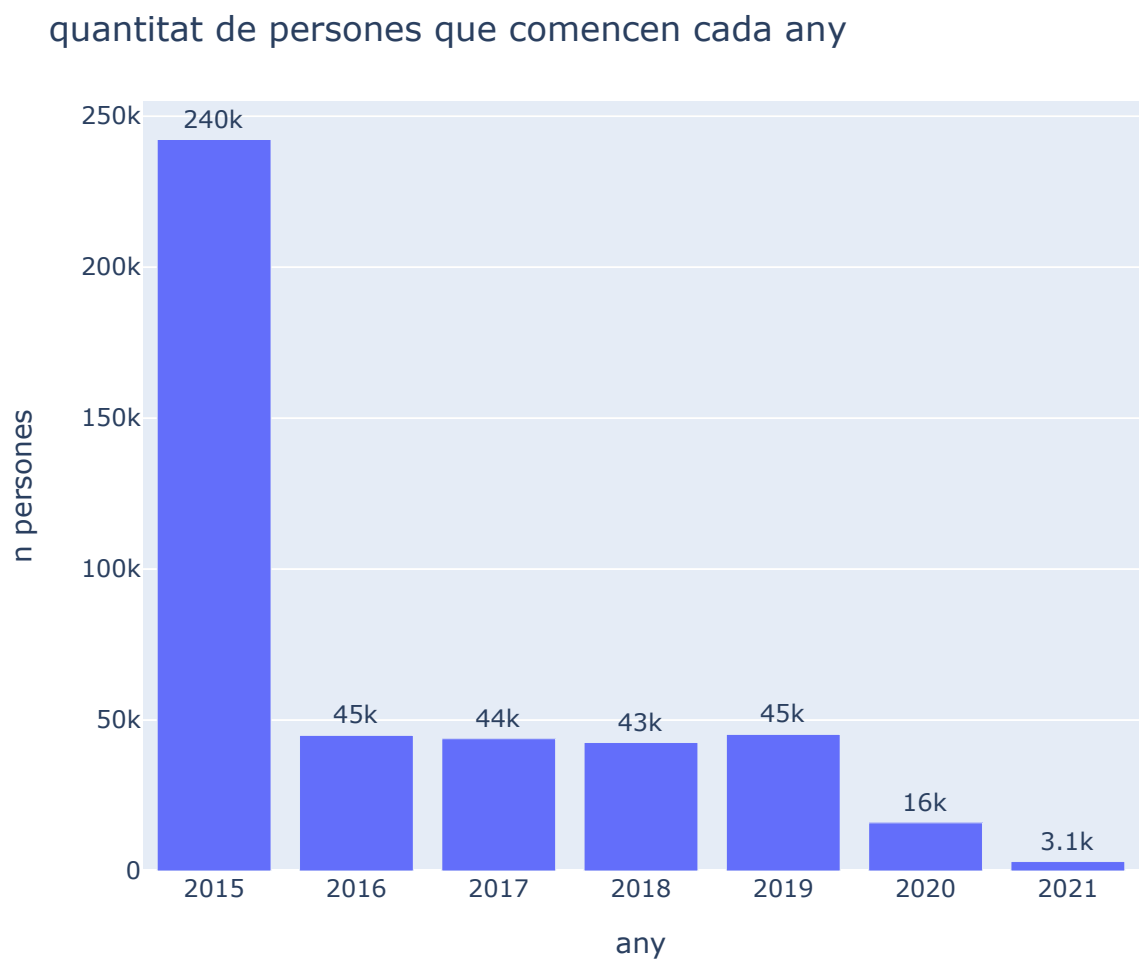
Durant l'any 2020, al ser l'any de confinament per la pandemia, és van haver de paralitzar els tornejos d'escacs o les activitats dins de clubs oficials, que és a través d'on et pots registrar a la FIDE. Per tant, pot ser aquest el motiu d'aquesta parada.

L'any 2021, la nostra base de dades només té registat els 4 primers mesos del any. Per tant aquest és el motiu d'aquest números tan baixos, tot hi així, el ritme és més lent que la dels anys passats.

In [6]:
```
# primer rating en standard per any
unique_id, start_index = np.unique(ratings['fide_id'], return_index=True)
oldest_ratings = ratings.iloc[start_index]

labels_year, count_year = np.unique(oldest_ratings['year'], return_counts=True)

print (f"El percentage de persones que van començar a jugar")
for x, y in enumerate(count_year):
    print(f" l'any {labels_year[x]} és del {y / np.sum(count_year) * 100}")
```

```
# plot
fig = px.bar(y=count_year, x=labels_year, text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_layout(title="quantitat de persones que comencen cada any",
                  xaxis=dict(title="any"),
                  yaxis=dict(title="n persones"))
fig.show()
```

```
El percentage de persones que van començar a jugar
 l'any 2015 és del 55.32829009579411
 l'any 2016 és del 10.25595344141576
 l'any 2017 és del 10.006304046924036
 l'any 2018 és del 9.7182821928947
 l'any 2019 és del 10.33018587802126
 l'any 2020 és del 3.6456120635703746
 l'any 2021 és del 0.7153722813797639
```

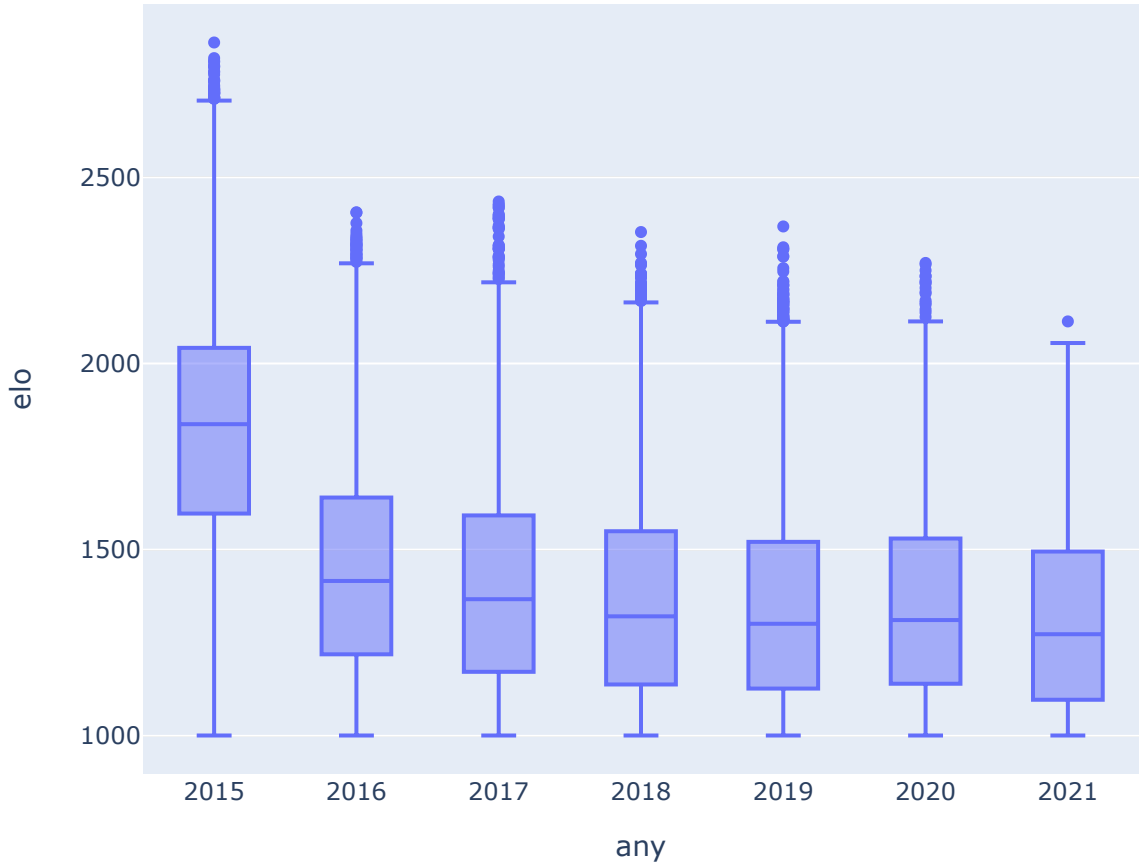


Per altre banda, també podem veure els elos inicials de les persones en la modalitat estandard. Com el 2015 no va ser un any real en quant a primer registres, podem veure que la concentració de les dades és superior que la resta d'anys.

Per la resta d'anys podem veure que la majoria de persones entra amb un valor d'elo de 1500-1100 punts.

In [7]:
```
# plot
fig = px.box(x=oldest_ratings['year'], y=oldest_ratings['rating_standard'])
fig.update_layout(title="primer elo registrat",
                  xaxis=dict(title="any"),
                  yaxis=dict(title="elo"))
fig.show()
```

primer elo registrat



Edats

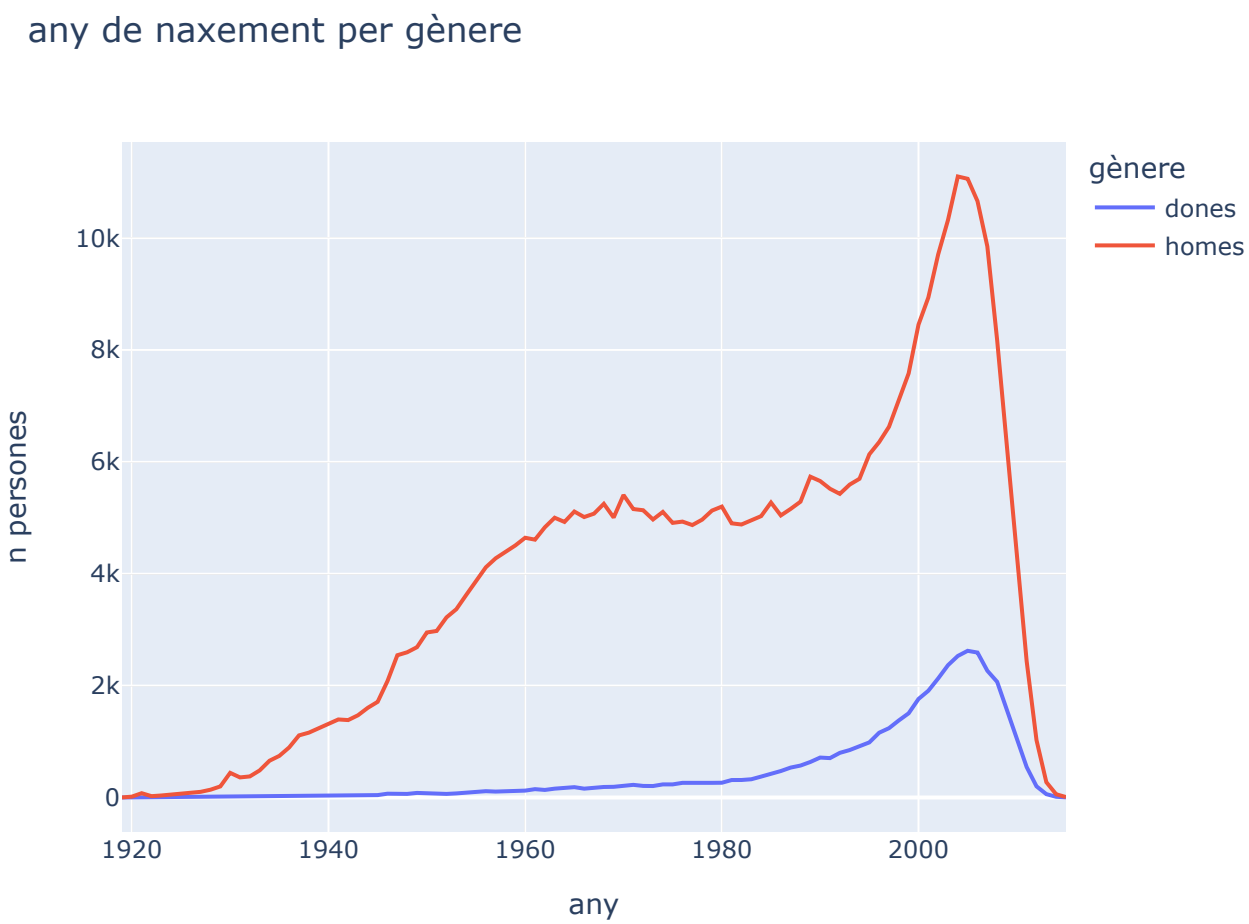
Per fer una comparativa d'edats, les diferenciarem pel gènere també.

Pels homes, nascuts entre els anys 20 i 50 podem veure que ha anat augmentant l'interés, fins els nascuts entre els 60 i els 80 que sembla que hi ha una constancia. Però a partir dels nascuts als 90 hi ha molta més participació, fins el 2005 que és l'any de naixement amb més persones registrades. Potser és degut a que la cultura pop ha anat popularitzant més i més els escacs i potser per això té tant d'exit entre els joves. També aquest pic pot estar donat per els anys de registre (si van començar fa deu anys, potser hi ha més posibilitat que apuntats hi hagi més jovent que si el registre és des de els anys 20, però tampoc podem saber-ho).

Per les dones, podem veure que hi ha dones de totes les edat jugant, al igual que els homes, però en comparativa, sempre hi ha molts més homes com ja hem vist abans. Tot hi així, el pic de gent nascuda l'any 2005 també està present de forma notoria.

In [8]:
```
born_year = np.unique(players['yob'])
born_year = np.delete(born_year, 0)
male_born = np.array([])
female_born = np.array([])

for yob in born_year:
    all_in_year = players.loc[players['yob'] == yob]
    _, count_in_year = np.unique(all_in_year['gender'], return_counts=True)
    if count_in_year.size == 1:
        female_born = np.append(female_born, 0)
        male_born = np.append(male_born, count_in_year[0])
```

```
    else:
        female_born = np.append(female_born, count_in_year[0])
        male_born = np.append(male_born, count_in_year[1])

# plot
fig = go.Figure()
fig.add_trace(go.Scatter(x=born_year,
                         y=female_born,
                         name='dones'))
fig.add_trace(go.Scatter(x=born_year,
                         y=male_born,
                         name='homes'))
fig.update_layout(title="any de naxement per gènere",
                  xaxis=dict(title="any"),
                  yaxis=dict(title="n persones"),
                  legend_title_text='gènere')
fig.show()
```

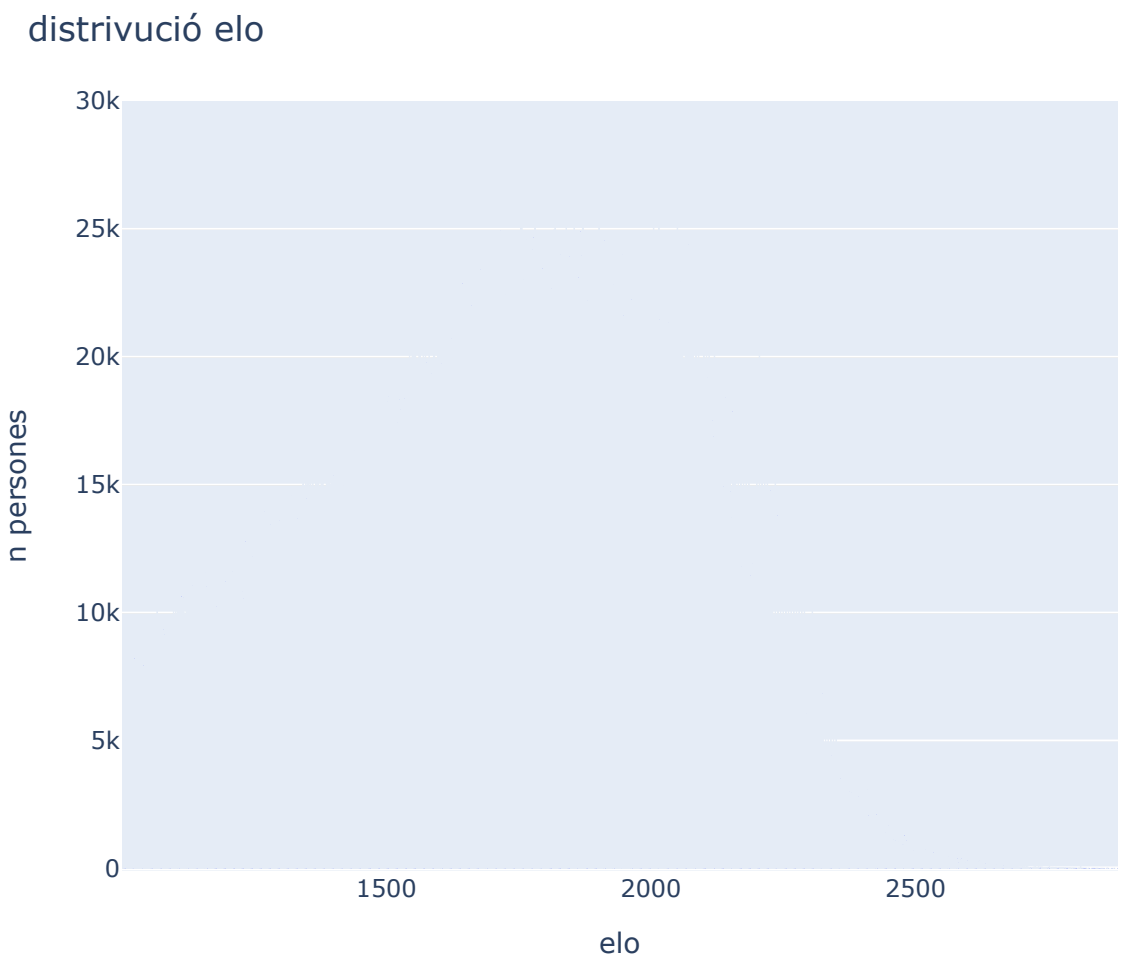


Comparativa elos

Com podem observar, el total de elos que tenim, podem veure que intenta tenir forma de campana de Gauss, però al estar el límit inferior a 1000 punts, treu part d'aquesta distrivució. La majoria de la gent esta acumulada en per sota dels 2100, sobre tot entre els 1600 i els 2100 punts. Arribar per sobre del 2400 punts ja és bastant extrany i hi ha menys gent a messura que abancen els punts.

In [9]:
```
descending_ratings = ratings.sort_values(by=['year', 'month'], ascending=False)

_, unique_id = np.unique(descending_ratings['fide_id'], return_index=True)
```

```
last_ratings = descending_ratings.iloc[unique_id]


label_elo, count_elo = np.unique(descending_ratings['rating_standard'], return_counts=True)

fig = px.bar(y=count_elo, x=label_elo, text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_layout(title="distrivució elo",
                    xaxis=dict(title="elo"),
                    yaxis=dict(title="n persones"))
fig.update_yaxes(range=[500,3000])
fig.update_yaxes(range=[-100,30000])
fig.show()
```



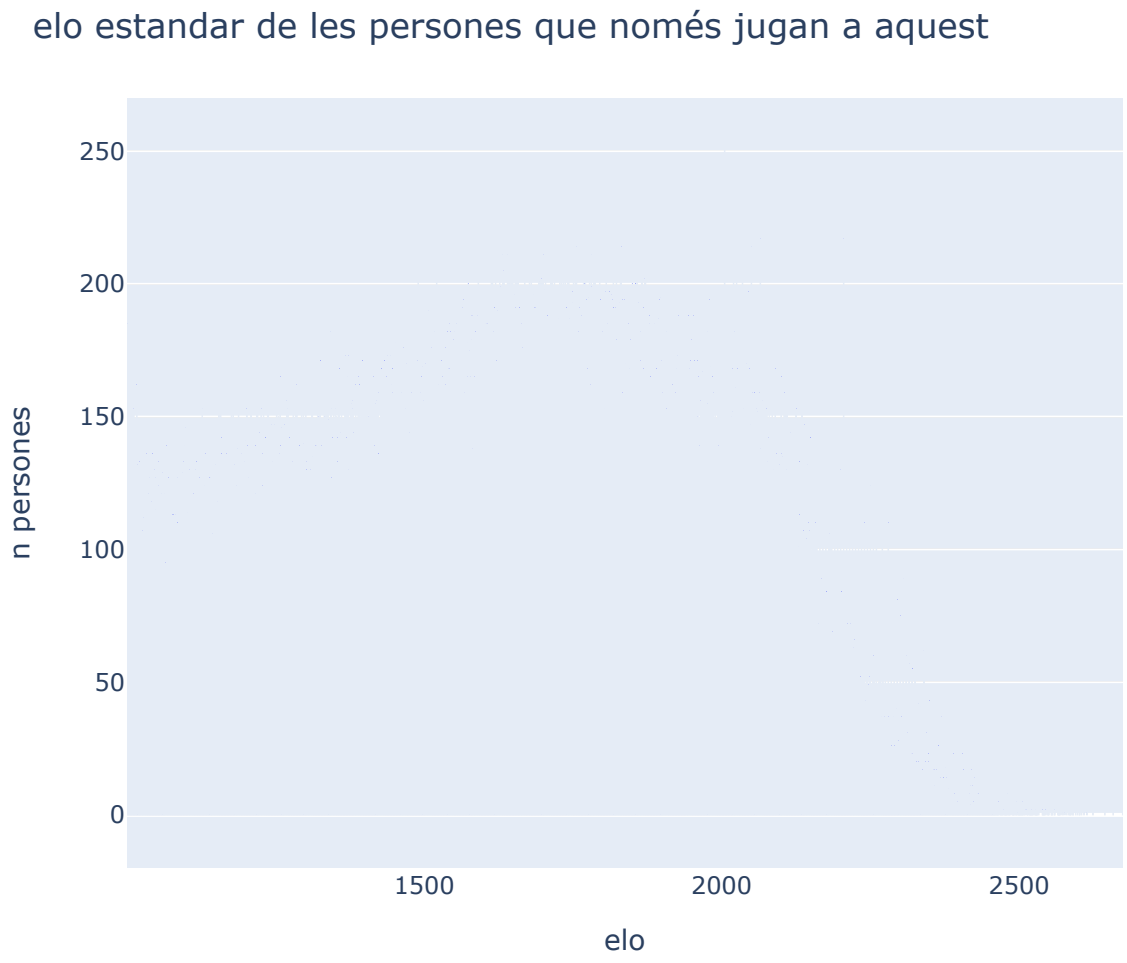
Ara volem comparar la distribució dels elos en funció de les modalitats que juguen.

Per començar, mirarem la gent que juga només a estandard, que és gairabé la meitat, un 45%. Podem veure que la gran majoria esta per sota dels 2100 i la majoria és concentra en els 1600 i 1800. La persona amb més elo és de 2674 punts.

In [10]:
```
# elo solo de las personas que solo juegan a standard

standard = last_ratings.loc[np.isnan(last_ratings.loc[:, 'rating_rapid'])]
standard = standard.loc[np.isnan(standard.loc[:, 'rating_blitz'])]

label_elo, count_elo = np.unique(standard['rating_standard'], return_counts=True)

print(f"Són {len(standard)} persones les que jugen només estandard, un {len(standard) / 433388 *
```

```
# plot
fig = px.bar(x=label_elo, y=count_elo, text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_yaxes(range=[500,3000])
fig.update_yaxes(range=[-20,270])
fig.update_layout(title="elo estandar de les persones que només jugan a aquest",
                  xaxis=dict(title="elo"),
                  yaxis=dict(title="n persones"))
fig.show()
```

```
Són 197414 persones les que jugen només estandard, un 45.55133044754354%
```

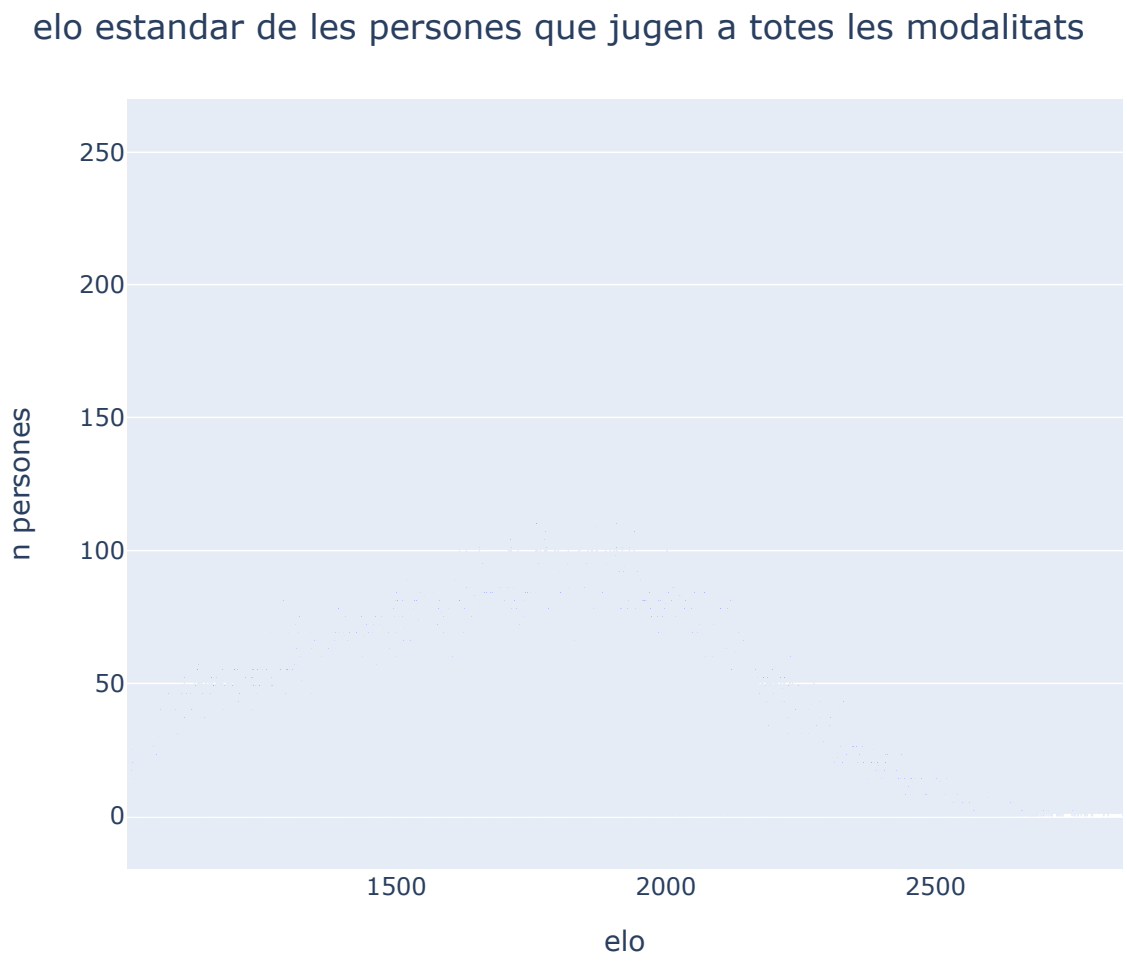


En el cas de l'elo de la gent que juga totes les modalitats, podem veure una certa diferencia, ja que la quantitat de gent és molt més homogenia per cada elo. La concentració de la majoria de la gent està entre els 1700 i els 2000 punts. I la persona amb més elo és de 2847 punts. Això ens deixa veure que la gent que juga a les tres categories acostuma a ser més bona que la gent que només en juga una.

In [11]:
```
# elo nomes de les persones que tenen les tres disciplines
all_elo = last_ratings.loc[~np.isnan(last_ratings.loc[:, 'rating_rapid'])]
all_elo = all_elo.loc[~np.isnan(all_elo.loc[:, 'rating_blitz'])]
label_elo, count_elo = np.unique(all_elo['rating_standard'], return_counts=True)
print(f"Són {len(all_elo)} persones les que jugen a totes les modalitats, un {len(all_elo) / 433
# plot
fig = px.bar(x=label_elo, y=count_elo, text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_yaxes(range=[500,3000])
fig.update_yaxes(range=[-20,270])
fig.update_layout(title="elo estandar de les persones que jugen a totes les modalitats",
                  xaxis=dict(title="elo"),
```

```
                  yaxis=dict(title="n persones"))

fig.show()
```

```
Són 101226 persones les que jugen a totes les modalitats, un 23.356899591128506%
```



Així mateix també és interessant veure quina correlació hi ha entre els diferents elos. Aquesta estan molt correlacionades. Així pot ser fàcil de saber l'elo d'una modalitat sabent la d'un altre.

In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt

correlacio = ratings.iloc[:,3:].corr()

ax = sns.heatmap(correlacio, annot=True)
```

Títols

Dins dels escacs, ha partir de certa quantitat d'elo, trobem que és posan títols per designar el nivell. Només un 4.5% de les persones tenen un títol assignat.

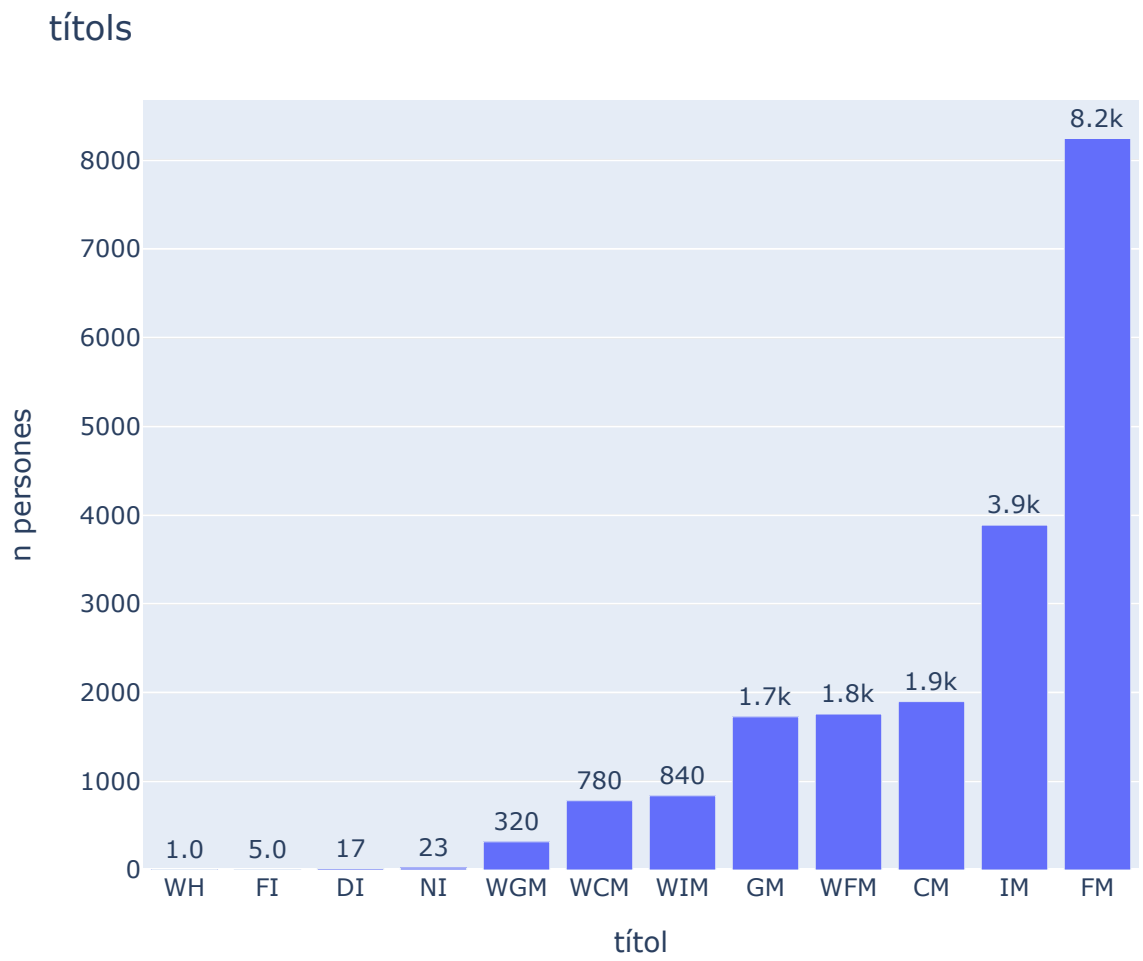
Els diferents títiols que hi ha són:

	elo	títol
	1400-1700	Arena FIDE Master (FM)
	1700-2000	Arena International Master (IM)
dones	2000-2099	candidata a mestra (WCM)
homes		expert nacional
dones	2100-2199	mestra FIDE (WFM)
homes		expert nacional
dones	2200-2299	mestra internacional (WIM)
homes		candidat a mestre (CM)
dones	2300-2399	gran mestra (WGM)
homes		mestre FIDE (MF)
	2400-2499	mestre internacional (MI)
	2500-2599	gran mestre (GM)
	2600-2699	super gran mestre (NI)
	2700-2799	candidat a campeó mundial (DI)
	≥2800	campeó mundial (WH)

In [13]:
```
label_titles, count_titles = np.unique(players['title'].astype('str'), return_counts=True)
count_all = players.shape[0]

label_titles, count_titles = np.delete(label_titles, 0), np.delete(count_titles, 0)
mask_sort = np.argsort(count_titles)


label_titles, count_titles = label_titles[mask_sort], count_titles[mask_sort]

print(f"Hi ha {count_titles[-1]/ 433388 * 100}% sense cap títol, per tant, amb és un {(1 - (coun
```

```
# plot
fig = px.bar(y=count_titles[:-1], x=label_titles[:-1], text_auto='.2s')
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.update_layout(title="títols",
                  xaxis=dict(title="títol"),
                  yaxis=dict(title="n persones"))
fig.show()
```

```
Hi ha 95.5010291009442% sense cap títol, per tant, amb és un 4.498970899055809%
```



2 Tractament de les dades

Creem la atribut *time_played* on estan contabilitzats els mesos que ha jugat una persona a el mode estandard. Per tant, treurem totes les files que el rating estandard és NaN i contarem cada més jugat. Així ens desdfem de les columnes *month* i *year* que ja no tenen utilitat.

L'objectiu d'això és fer que tots els ratings comencin en el mateix punt, cosa que facilitara a l'hora de fer les prediccions, les comparatives de resultats i la visualització serà més clara.

In [14]:
```
data = ratings
# data = data[(data['month'] == 1) | (data['month'] == 7)]
data = data[data['rating_standard'].notna()]

df = pd.DataFrame()
df['time_playing'] = data.groupby('fide_id')['fide_id'].cumcount() + 1
data = pd.concat([data, df], axis=1)
data = data.drop(['year', 'month'], axis=1)
```

```
data.head(5)
```

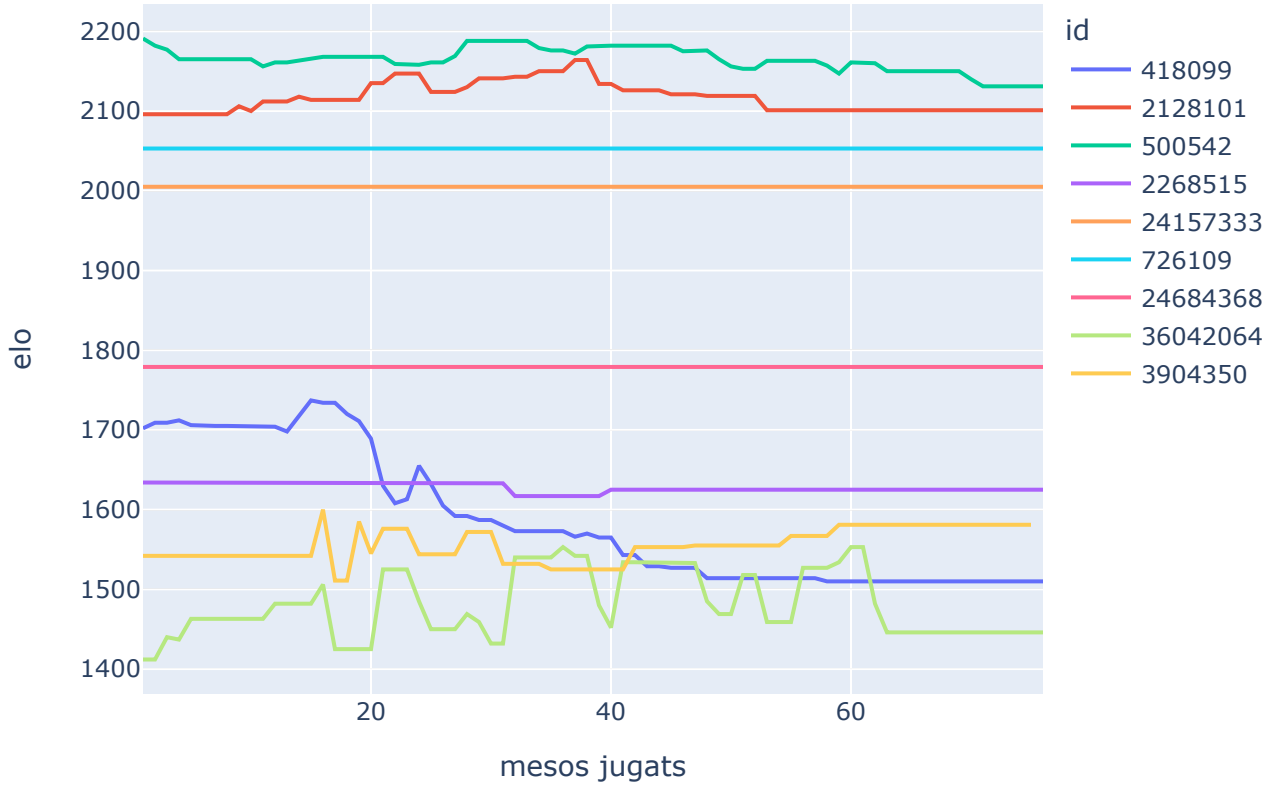
Out[14]:

	fide_id	rating_standard	rating_rapid	rating_blitz	time_playing
0	100013	2456.00	2462.00	NaN	1
1	100021	2422.00	NaN	NaN	1
2	100048	1607.00	NaN	NaN	1
3	100064	2116.00	NaN	NaN	1
4	100072	2469.00	NaN	NaN	1

Ara, visualitzarem una gràfica per de 10 persones (aproximadament, perquè no tots tenen valors en el rating estandard).

Podem veure com totes parteixen del mateix punt temporal i acaben en funció del temps que hagin jugat en total. Les funcions rectes, indiquen que no hi ha hagut una actualització del elo durant el temps. Podem observar que no hi ha una tendencia clara dins de les funcions.

In [15]:
```
# seleccionem 10 persones aleatoriament per veure la seva evolució en el rating estandard al llar
random_players = players.sample(n = 10)

fig = go.Figure()

for id in random_players['fide_id']:
    pl = data[id == data['fide_id']]
    # plot
    fig.add_trace(go.Scatter(x=pl['time_playing'],
                             y=pl['rating_standard'],
                             name=id))

fig.update_layout(title="evolució elo de 10 persones aprox",
                      xaxis=dict(title="mesos jugats"),
                      yaxis=dict(title="elo"),
                      legend_title_text='id')
fig.show()
```

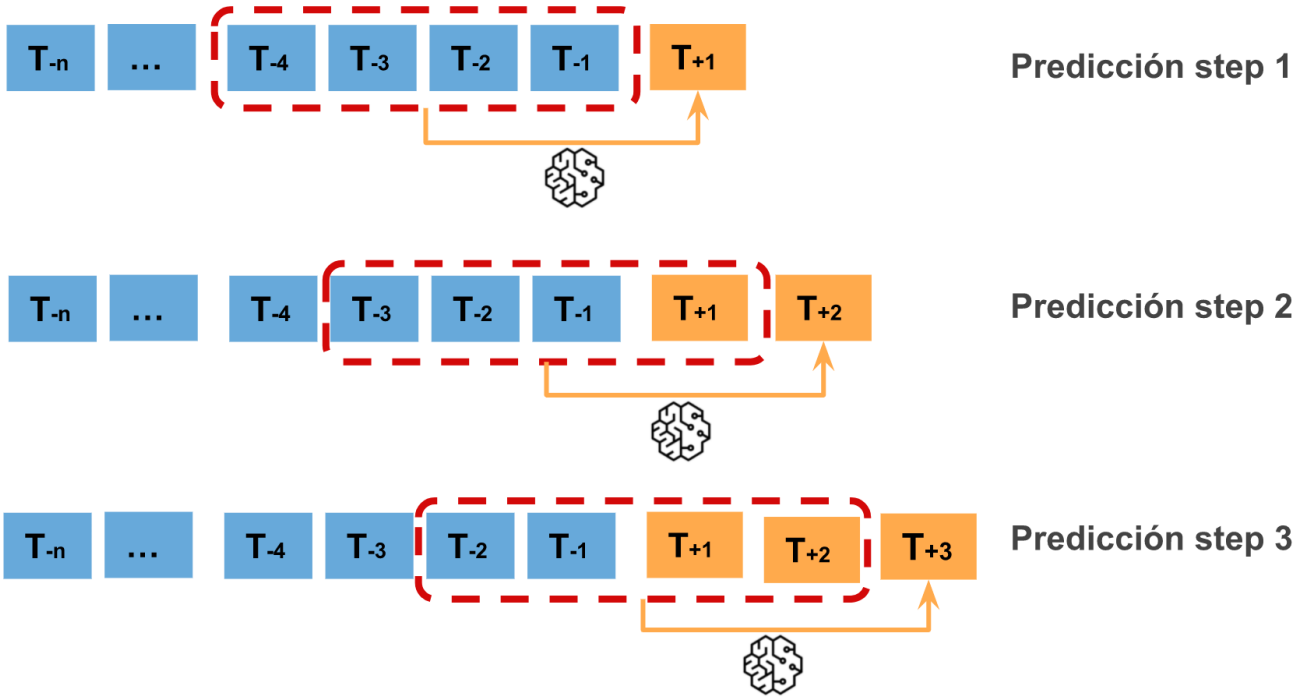


3 Sèries temporals

L'objectiu és fer una predicció del *rating_standard* basant-nos en els retings anteriors. Això és denomina **forecasting**, on el que és preten predir el futur dins d'una sèrie temporal (com és aquest cas).

Per fer-ho, aplicarem el mètode de **Recursive multi-step forecasting** que és el model més senzill. Aquest consisteix en un procés recursiu on cada nova predicció fa ús de les dades anteriors i les prediccions

anteiors.



Primer de tot, he volgut inicialitzar un seguit de funcions perquè a les cel·les següents sigui més clar de veure que s'esta fent a cada pas: `split_train_test(data, id)` on és va la divisió de train i test (70%-30%) del rating en funció de l'id que se li passa per paràmetres i `hyperparameter(train, steps)` on és busquen els hiperparametres que donden un *mean squared error* més baix.

In [21]:
```
from skforecast.model_selection import grid_search_forecaster
from skforecast.ForecasterAutoreg import ForecasterAutoreg
from sklearn.ensemble import RandomForestRegressor

# separar dades train i test
def split_train_test(data, id):
    # pereparació de les dades del id actual
    act_player = data[data['fide_id'] == id]
    act_player.reset_index(inplace=True)
    act_player.index = act_player.index + 1

    # separació de train i test en 70-30
    seed = int(act_player.shape[0] * 0.7 + 1)
    train = act_player[:seed + 1]
    test = act_player[seed:]

    return train, test

def hyperparameter(train, test, steps):
    # Lags utilizats com predictors
    lags_grid = [10, test.shape[0]]

    # Hiperparametres del regresor
    param_grid = {'n_estimators': [100, 250, 500],
                  'max_depth': [3, 5, 7, 10]}

    hiperparametres = grid_search_forecaster(
                            forecaster         = forecaster,
                            y                  = train['rating_standard'],
                            param_grid         = param_grid,
                            lags_grid          = lags_grid,
```

```
                            steps              = steps,
                            refit              = True,
                            metric             = 'mean_squared_error',
                            initial_train_size = int(len(train)*0.7),
                            fixed_train_size   = False,
                            return_best        = True,
                            verbose            = False
                        )
    return hiperparametres
```

He seleccionat un id de forma aleatoria per explicar com va pas a pas aquests model.

In [22]:
```
# seleccionem 1 persona aleatoriament per veure la seva evolució en el rating estandard al llarg
random_player = players.loc[2802724 == players['fide_id']]
random_player
```

Out[22]:

	fide_id	federation	gender	title	yob
110824	2802724	ISR	M	NaN	1965

Per aquest fem la partició de les dades en train i test, on test serà el resultat correcte de la nostra predicció temporal.

Com podem veure, els resultats de la partició fan que el test i el train, al ser força diferents en quant a tendències, és possible que al intentar fer una predicció no surti tan clara.

In [23]:
```
random_id = random_player.values[0, 0]
train, test = split_train_test(data, random_id)

print(f"Quantitat de train n={len(train)}")
print(f"Quantitat de test n={len(test)}")

# plot
fig = go.Figure()

fig.add_trace(go.Scatter(x=train['time_playing'],
                         y=train['rating_standard'],
                         name='train'))
fig.add_trace(go.Scatter(x=test['time_playing'],
                         y=test['rating_standard'],
                         name='test'))
fig.update_layout(title=f"evolució elo {random_id}",
                  xaxis=dict(title="mesos jugats"),
                  yaxis=dict(title="elo"),
                  legend_title_text='id')
```

```
Quantitat de train n=55
Quantitat de test n=22
```

El model es basa en **forecaster** com ja he explicat al principi del apartat. Aquest necessita un regressor, en el meu cas **randomForest**, per tal de que per cada mes aquest fa una predicció basant-se en els *lags* (mesos) anteriors.

Una vegada generat el model, és farà un entrenament amb les dades.

In [24]:
```
# creació model forecaster amb randomForest
forecaster = ForecasterAutoreg(
                regressor = RandomForestRegressor(random_state=123),
                lags = 12 # ultims mesos utilitzats del entrenament
             )
# entrenament
forecaster.fit(y=train['rating_standard'][:-1])
```

Farem doncs, una predicció amb el model indicant-li els mesos que volem predir (per ser equivalent, tants mesos com tingui el test).

Per veure si fa una bona predicció, utilitzaré el *mean squared error* i la representació gràfica del rating i la predicció.

Així doncs, com podem veure en la representació gràfica, les tendències de la prediccó no corresponen amb la forma dels valors reals. Pot estar degut al que he mencionat de la diferència en les tendècies i perquè no estigui possant uns valors bons en els paràmetres del model.

In [25]:
```
from sklearn.metrics import mean_squared_error
```

```
# predicció
steps = len(test) # mesos a predir
pred = forecaster.predict(steps=steps)

#metrics
print(f"mean-squared error {mean_squared_error(y_true=test['rating_standard'], y_pred=pred)}")

# plot
fig.add_trace(go.Scatter(x=pred.index,
                         y=pred,
                         name='pred'))
```

```
mean-squared error 44.376072743376724
```

Per intentar millorar doncs la predicció, mirarem quins són els millors hiperparametres: pel **randomForest** la profunditat màxima del arbre sigui 3, la quantitat d'estimadors sigui 100 i pel **forecaster** els lags siguin 10.

In [26]:
```
hiper = hyperparameter(train, test, steps)
```

```
Number of models compared: 24.
```

```
loop lags_grid:   0%|                                                  | 0/2 [00:00<?, ?it/s]
loop param_grid:   0%|                                                | 0/12 [00:00<?, ?it/s]
loop param_grid:   8%|███                                    | 1/12 [00:00<00:01,  6.21it/s]
loop param_grid:  17%|██████                                 | 2/12 [00:00<00:03,  3.31it/s]
loop param_grid:  25%|█████████                              | 3/12 [00:01<00:04,  1.86it/s]
loop param_grid:  33%|████████████                           | 4/12 [00:01<00:03,  2.55it/s]
loop param_grid:  42%|███████████████                        | 5/12 [00:01<00:02,  2.53it/s]
loop param_grid:  50%|██████████████████                     | 6/12 [00:02<00:03,  1.79it/s]
loop param_grid:  58%|█████████████████████                  | 7/12 [00:02<00:02,  2.32it/s]
loop param_grid:  67%|████████████████████████               | 8/12 [00:03<00:01,  2.34it/s]
loop param_grid:  75%|███████████████████████████            | 9/12 [00:04<00:01,  1.84it/s]
loop param_grid:  83%|██████████████████████████████        | 10/12 [00:04<00:00,  2.33it/s]
loop param_grid:  92%|█████████████████████████████████     | 11/12 [00:04<00:00,  2.39it/s]
loop param_grid: 100%|████████████████████████████████████| 12/12 [00:05<00:00,  1.86it/s]
loop lags_grid:  50%|██████████████████                      | 1/2 [00:05<00:05,  5.59s/it]
loop param_grid:   0%|                                                | 0/12 [00:00<?, ?it/s]
loop param_grid:   8%|███                                    | 1/12 [00:00<00:01,  6.32it/s]
loop param_grid:  17%|██████                                 | 2/12 [00:00<00:03,  3.32it/s]
loop param_grid:  25%|█████████                              | 3/12 [00:01<00:04,  1.92it/s]
loop param_grid:  33%|████████████                           | 4/12 [00:01<00:03,  2.59it/s]
loop param_grid:  42%|███████████████                        | 5/12 [00:01<00:02,  2.54it/s]
loop param_grid:  50%|██████████████████                     | 6/12 [00:02<00:03,  1.87it/s]
loop param_grid:  58%|█████████████████████                  | 7/12 [00:02<00:02,  2.39it/s]
loop param_grid:  67%|████████████████████████               | 8/12 [00:03<00:01,  2.42it/s]
loop param_grid:  75%|███████████████████████████            | 9/12 [00:04<00:01,  1.90it/s]
loop param_grid:  83%|██████████████████████████████        | 10/12 [00:04<00:00,  2.38it/s]
loop param_grid:  92%|█████████████████████████████████     | 11/12 [00:04<00:00,  2.42it/s]
loop param_grid: 100%|████████████████████████████████████| 12/12 [00:05<00:00,  1.89it/s]
loop lags_grid: 100%|████████████████████████████████████████| 2/2 [00:11<00:00,  5.53s/it]
`Forecaster` refitted using the best-found lags and parameters, and the whole data set:
  Lags: [ 1  2  3  4  5  6  7  8  9 10]
  Parameters: {'max_depth': 3, 'n_estimators': 100}
  Backtesting metric: 100.47058823529412
```

In [27]:
```
min_lags = int(hiper.iloc[0, 0][-1])
min_lags
```

Out[27]:
```
10
```

Com podem veure, la modificació dels hiperparametres ha sigut una millora en el resultat, tot i no ser exacte, dona una tendencia molt més similar al que és en relaitat. A més, hi ha una millora considerable dins del que és el mean-square error.

In [28]:
```
# creació model forecaster amb randomForest amb els hiperparametres
forecaster = ForecasterAutoreg(

    regressor = RandomForestRegressor(random_state=123, n_estimators=100, max_depth=3),
                lags = min_lags # ultims mesos utilitzats del entrenament
            )
# entrenament
forecaster.fit(y=train['rating_standard'][:-1])

# predicció
pred = forecaster.predict(steps=steps)

#metrics
print(f"mean-squared error {mean_squared_error(y_true=test['rating_standard'], y_pred=pred)}")

# plot
```

```
fig.add_trace(go.Scatter(x=pred.index,
                         y=pred,
                         name='pred_hyperparam'))
```

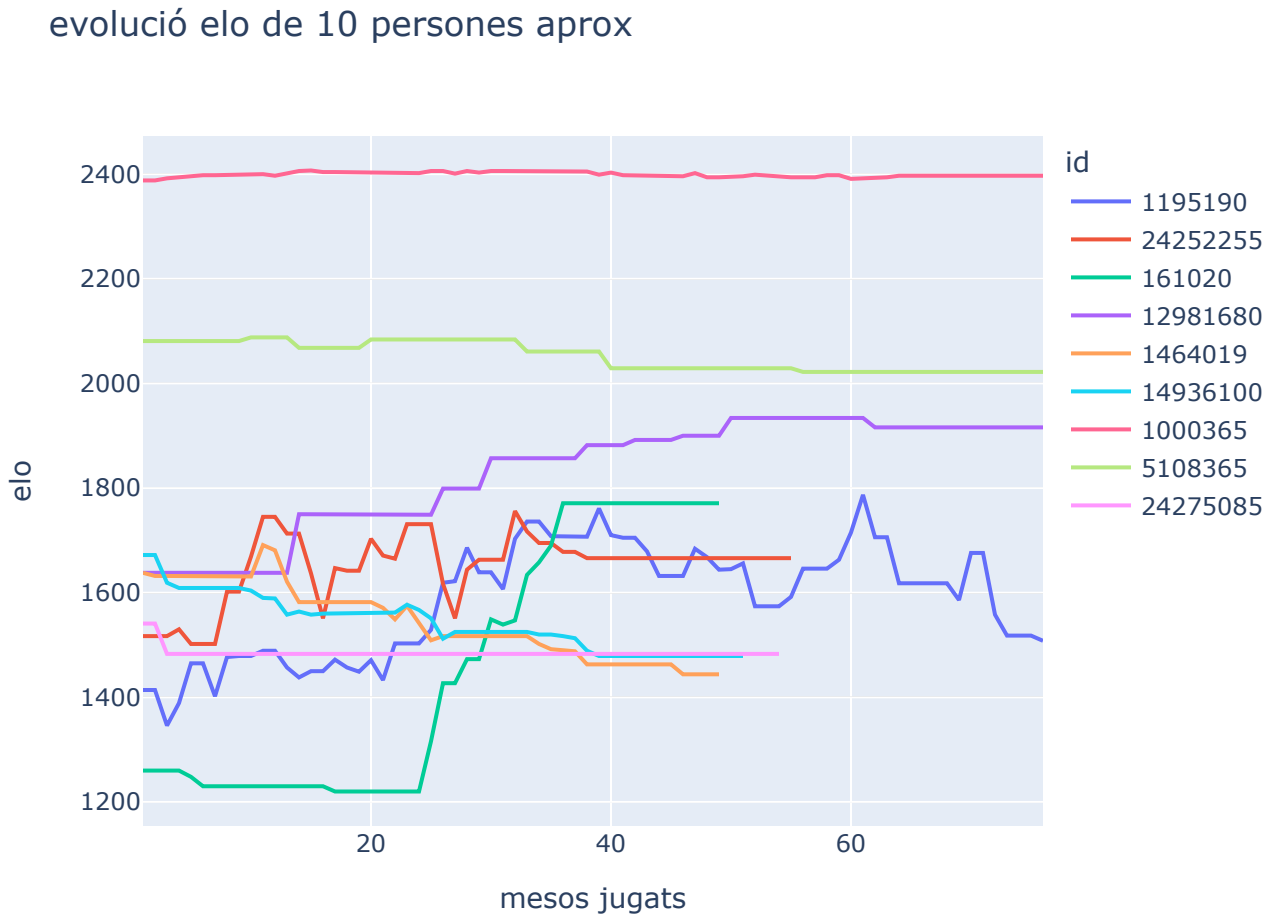
```
mean-squared error 4.191414596538592
```

4 Anàlisi de resultats

Per fer l'anàlisi de resultats hem escollit un seguit de persones que ens han semblat tenen uns valors de ratings interessants. Així bé, per poder treure unes conclusons més organitzades, els hem dividit en quatre grups: tendència a créixer, tendència a decréixer, tendència lineal i molta variació.

In [29]:
```
ids_train = np.array([1195190, 24252255, 161020, 12981680, 1464019, 14936100, 1000365, 5108365, 1

fig = go.Figure()

for id in ids_train:

    pl = data[id == data['fide_id']]
    # plot
    fig.add_trace(go.Scatter(x=pl['time_playing'],
                             y=pl['rating_standard'],
                             name=str(id)))

fig.update_layout(title="evolució elo de 10 persones aprox",
                  xaxis=dict(title="mesos jugats"),
                  yaxis=dict(title="elo"),
```

```
                    legend_title_text='id')
fig.show()
```



Abans, crearem un model forecaster per a entrenar a tots els models amb el matix classificador, amb les dades de hiperparametres que han sortit més vegades en una prova de 100 ids diferents que hem provat a un document a part.

In [30]:
```
# creació model forecaster amb randomForest amb els hiperparametres
forecaster = ForecasterAutoreg(
                regressor = RandomForestRegressor(random_state=123, n_estimators=250, max_depth=
                lags = 30 # ultims mesos utilitzats del entrenament
             )
```

In [31]:
```
def generador_reultat_predicció_id(array_id):

    for id in array_id:
        fig = go.Figure()
        fig.update_layout(title=f"evolució elo",
                          xaxis=dict(title="mesos jugats"),
                          yaxis=dict(title="elo"),
                          legend_title_text='id')
        print(id)
        # test i train
        train, test = split_train_test(data, id)
        # plot
        fig.add_trace(go.Scatter(x=train['time_playing'],
                                 y=train['rating_standard'],
                                 name=f'train {id}'))
        fig.add_trace(go.Scatter(x=test['time_playing'],
```

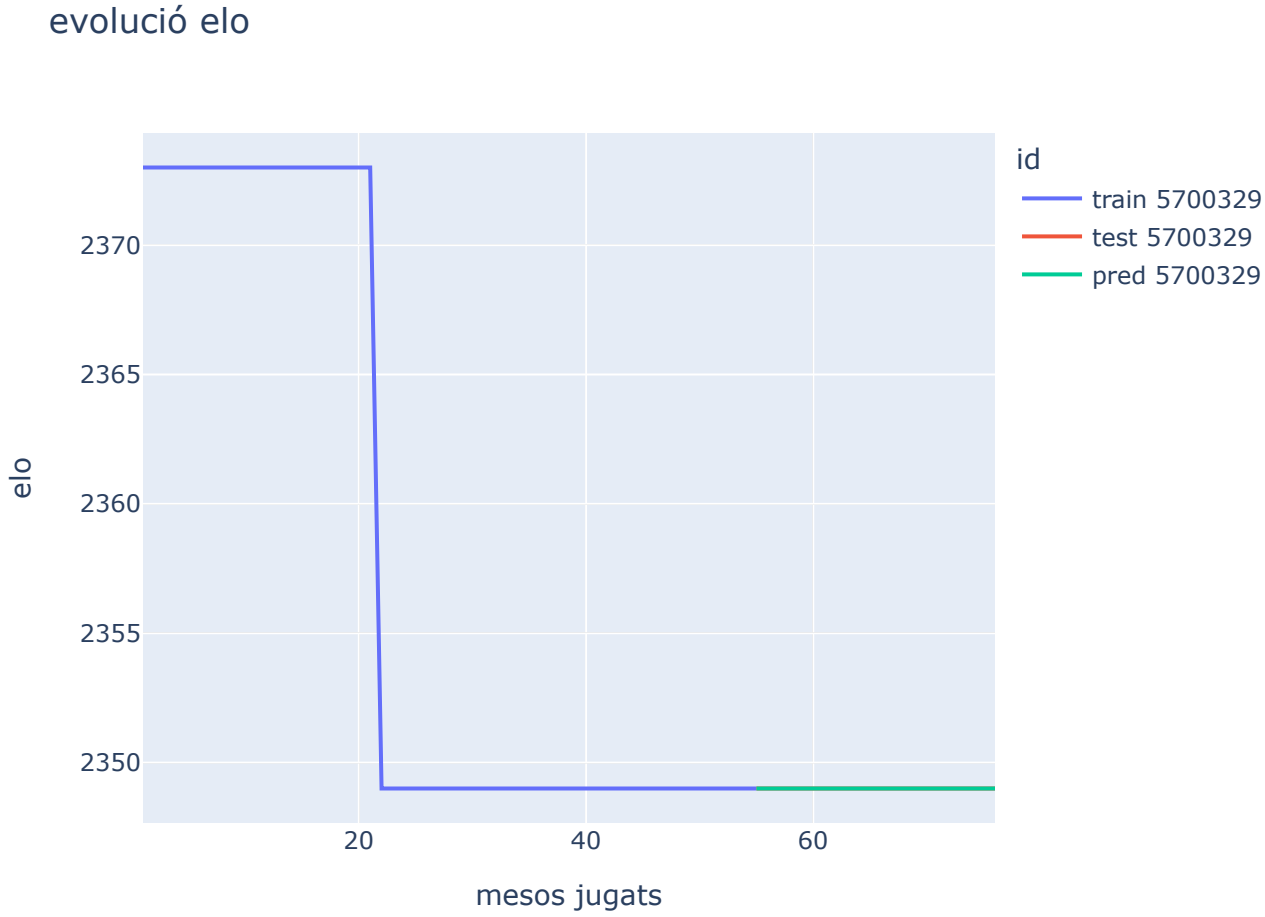
```
                                y=test['rating_standard'],
                                name=f'test {id}'))

        # entrenament
        forecaster.fit(y=train['rating_standard'][:-1])

        # predicció
        steps = len(test) # mesos a predir
        pred = forecaster.predict(steps=steps)

        #metrics
        print(f"mean-squared error {mean_squared_error(y_true=test['rating_standard'], y_pred=pr

        # plot
        fig.add_trace(go.Scatter(x=pred.index,
                                y=pred,
                                name=f'pred {id}'))
        fig.update_traces(hoverinfo='text+name', mode='lines')

        fig.show()
```

Tendència linial

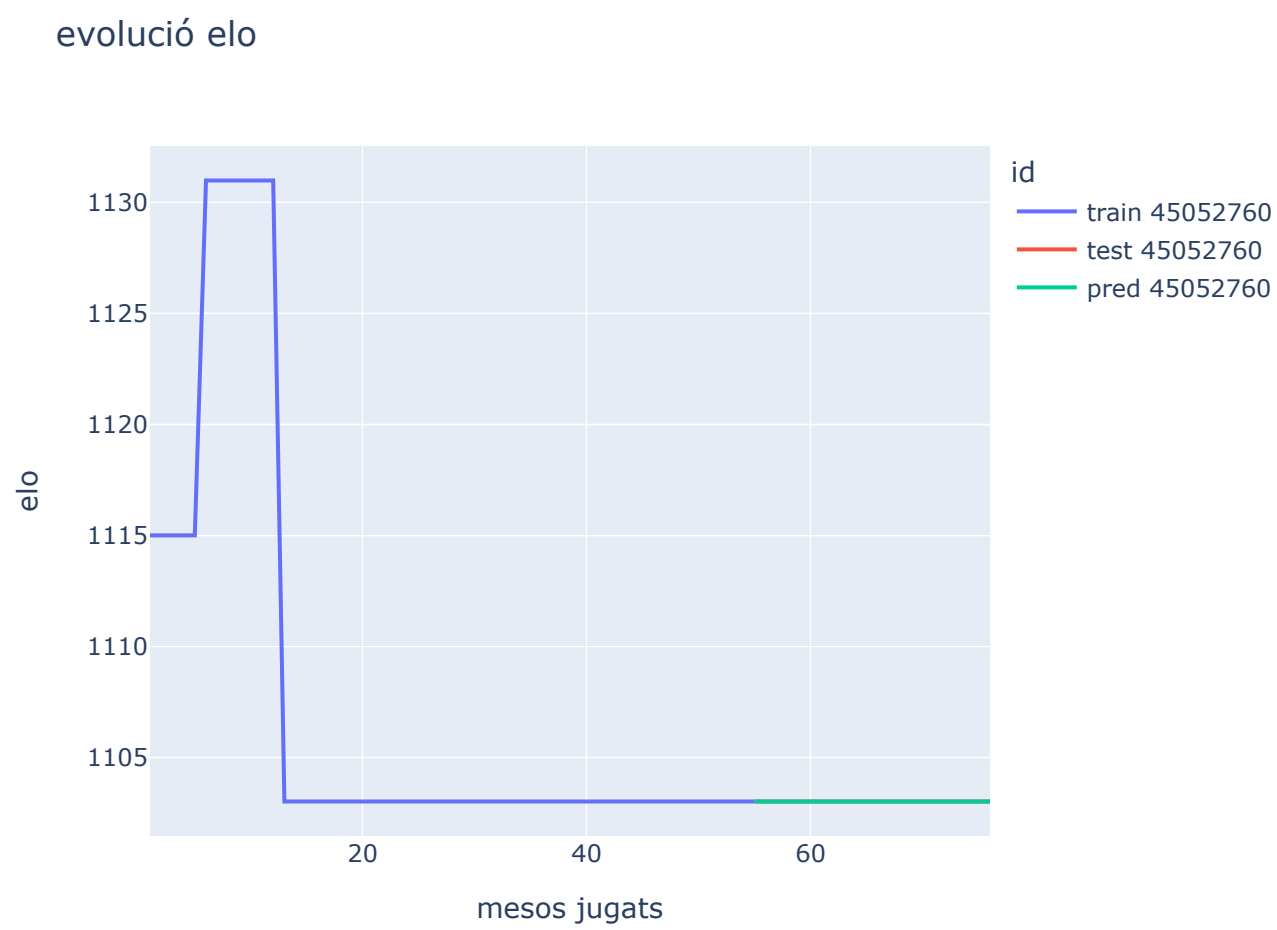
Com podem veure a la representació gràfica, per les prediccions que parteixen de un test força constant en el temps, és molt acertat. Totes tenen una predicció exacta del test, ja que enten que tots els mesos que ha mirat són constants i, per tant, el més probable és que aquest segueixi igual. Aquest és en part el més fàcil d'acertar sempre que no hi hagi un canvi abrupte en el test.

In [32]:
```
lineal = np.array([5700329, 45052760, 24275085])

generador_reultat_predicció_id(lineal)
```

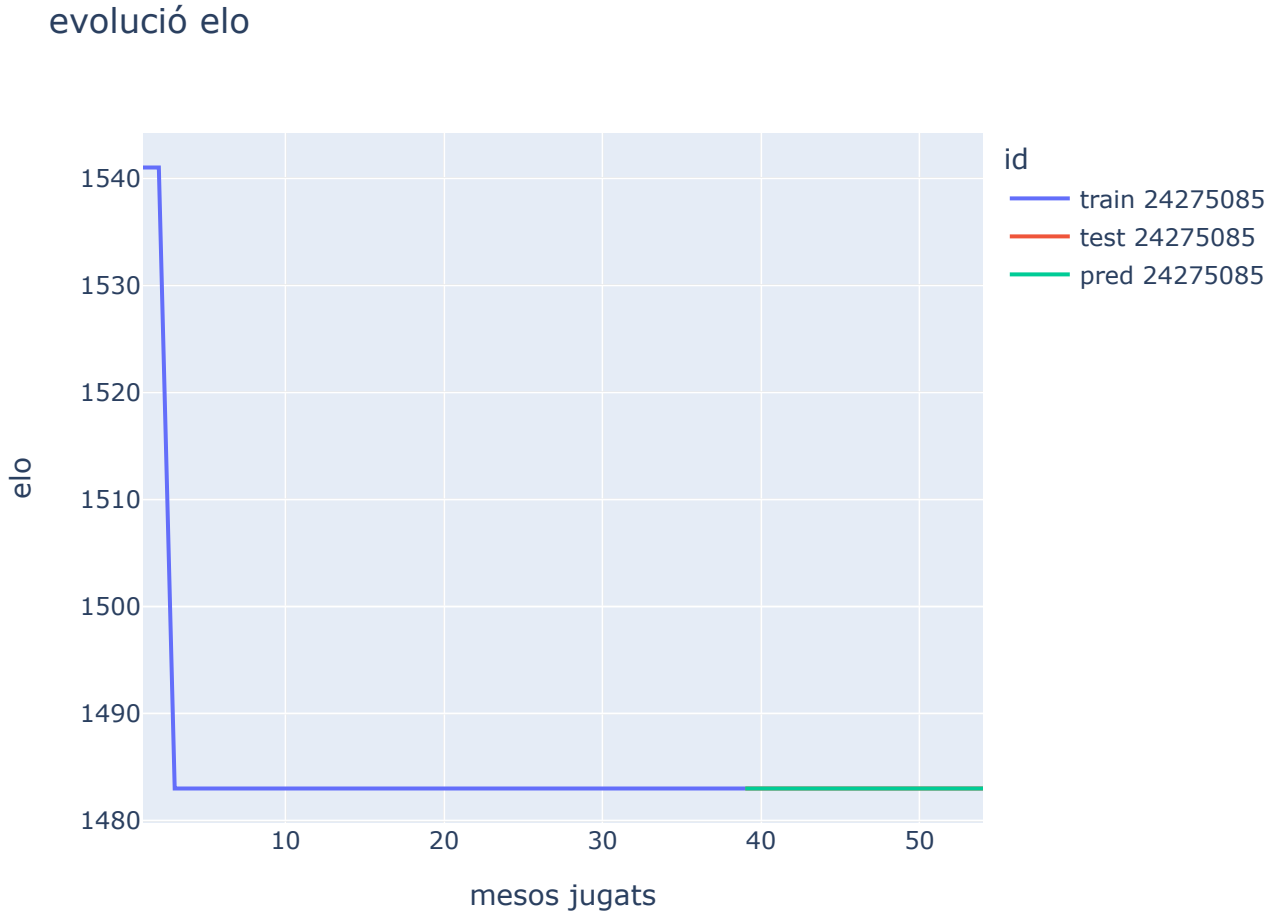
```
5700329
mean-squared error 0.0
```




```
45052760
mean-squared error 0.0
```




```
24275085
mean-squared error 0.0
```



Tendència a créixer

Els resultats per aquest tipus de elo no són tan homgenis com a l'apartat anterior, i ara bé, a partir d'aquest apartat la cosa es complica per la predicció de sèries temporals.

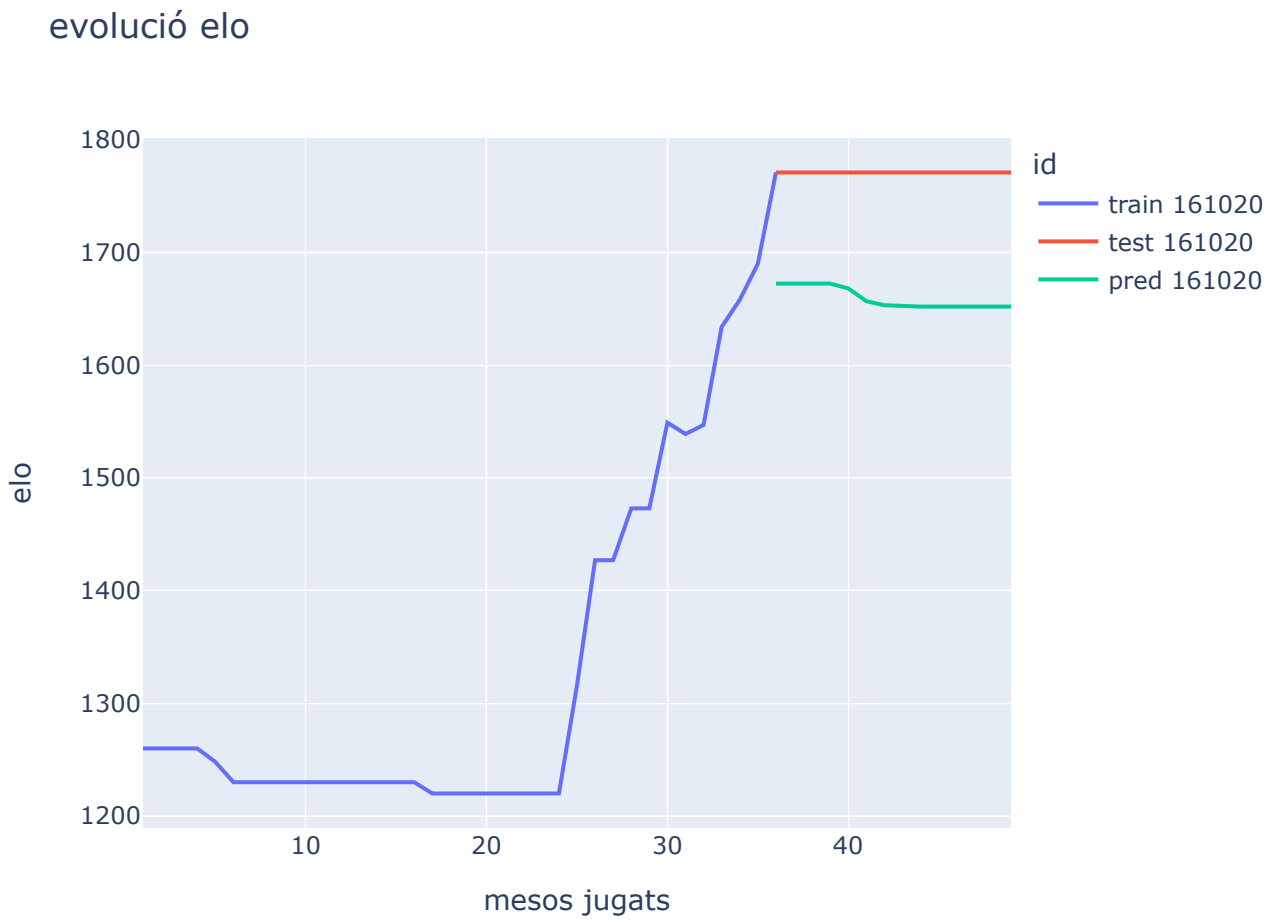
En el primer cas, 161020, si que és cert que el test varia molt del que és el train, ja que un és una constant i l'altre és creixent, però de totes formes el model no ha sigut capaç ni de predir una cosa ni l'altre. El primer punt de predicció és erroni a més de que ho fa a la baixa tot i que el que tenia per fer el fit era tot el contrari. A més, la forma que té no coincideix en absolut, fins i tot decreix una mica. Sembla que les pendents tan abruptes com aquesta, que és en 10 mesos pujar més de 500 punts, no ho acaba de processar del tot bé i per això fa aquesta caiguda i aquest descens tot i que és contrari al que ha vist.

En el següent, 12981680, sembla que a precebut que l'últim tram era constant i s'ha aferrat a ell. No era tant d'esperar que pugues perveure la caiguda que és mostra en el test, però ha sabut veure que una possibilitat era que és mantingués així de constant al menys un tram.

Per últim, 45165912, tot i que la tendència és a pujar, com és el que ha fet el test també, ha acabat fent una petita caiguda, cosa que és veu en alguns punts del train (com el mes 29-30) però després és manté constant, cosa que no és veu per tant temps en cap lloc del train. No queda allunyada del que ha passat en realitat, però no és prou precissa.

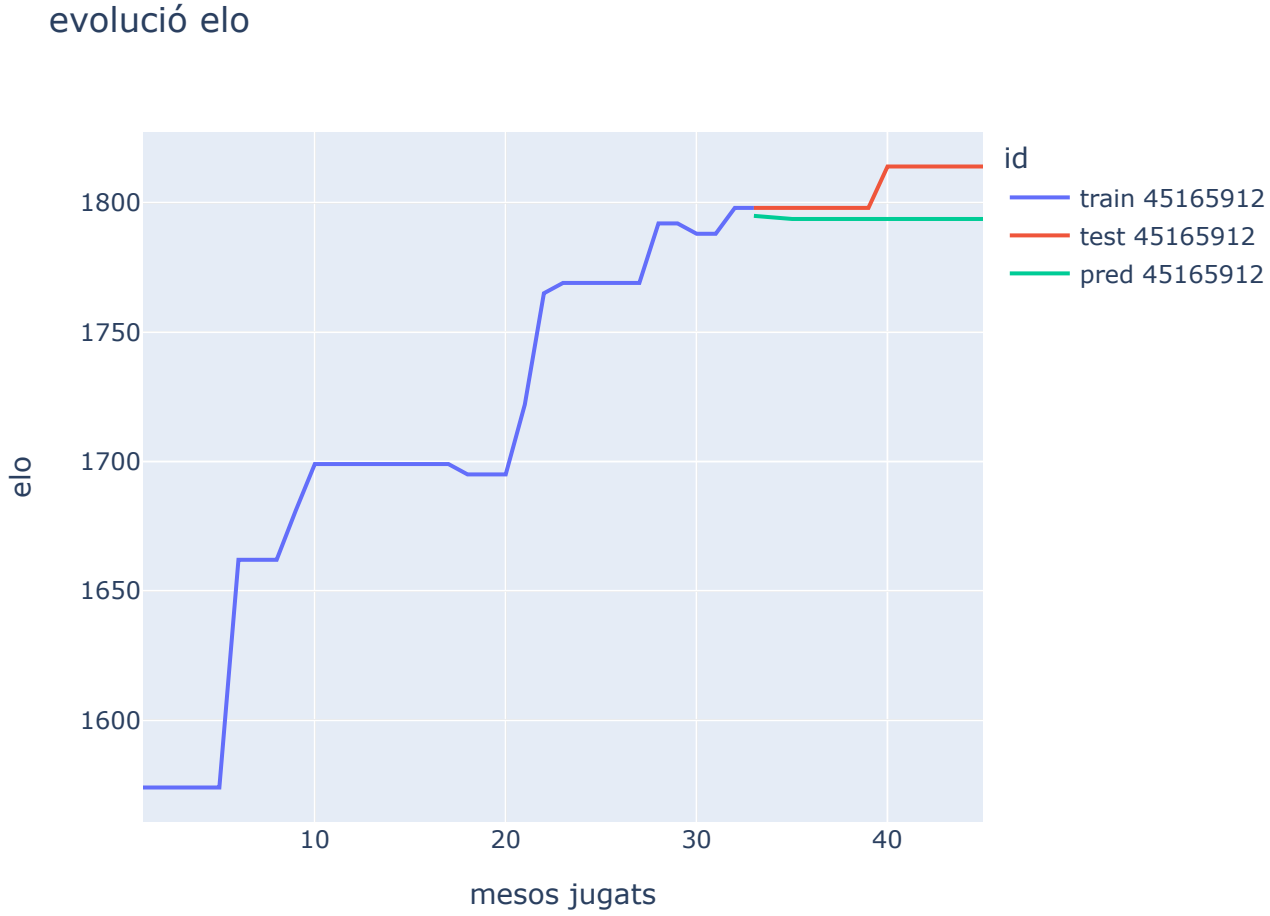
In [33]:
```
creixer = np.array([161020, 12981680, 45165912])

generador_reultat_predicció_id(creixer)
```

```
161020
mean-squared error 12532.584215999994
```




```
12981680
mean-squared error 204.67301527272576
```


```
45165912
mean-squared error 198.7068307692301
```

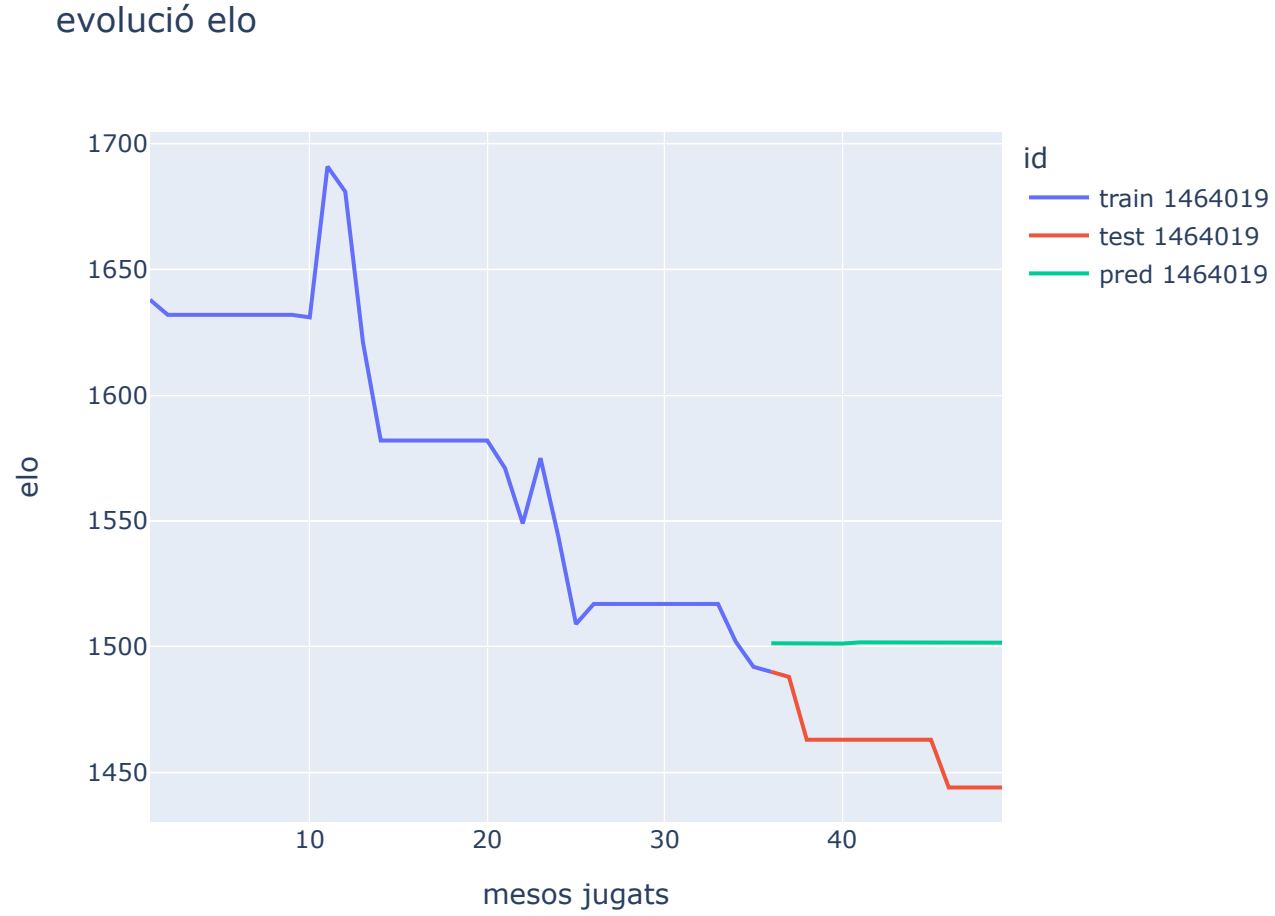


Tendència a decréixer

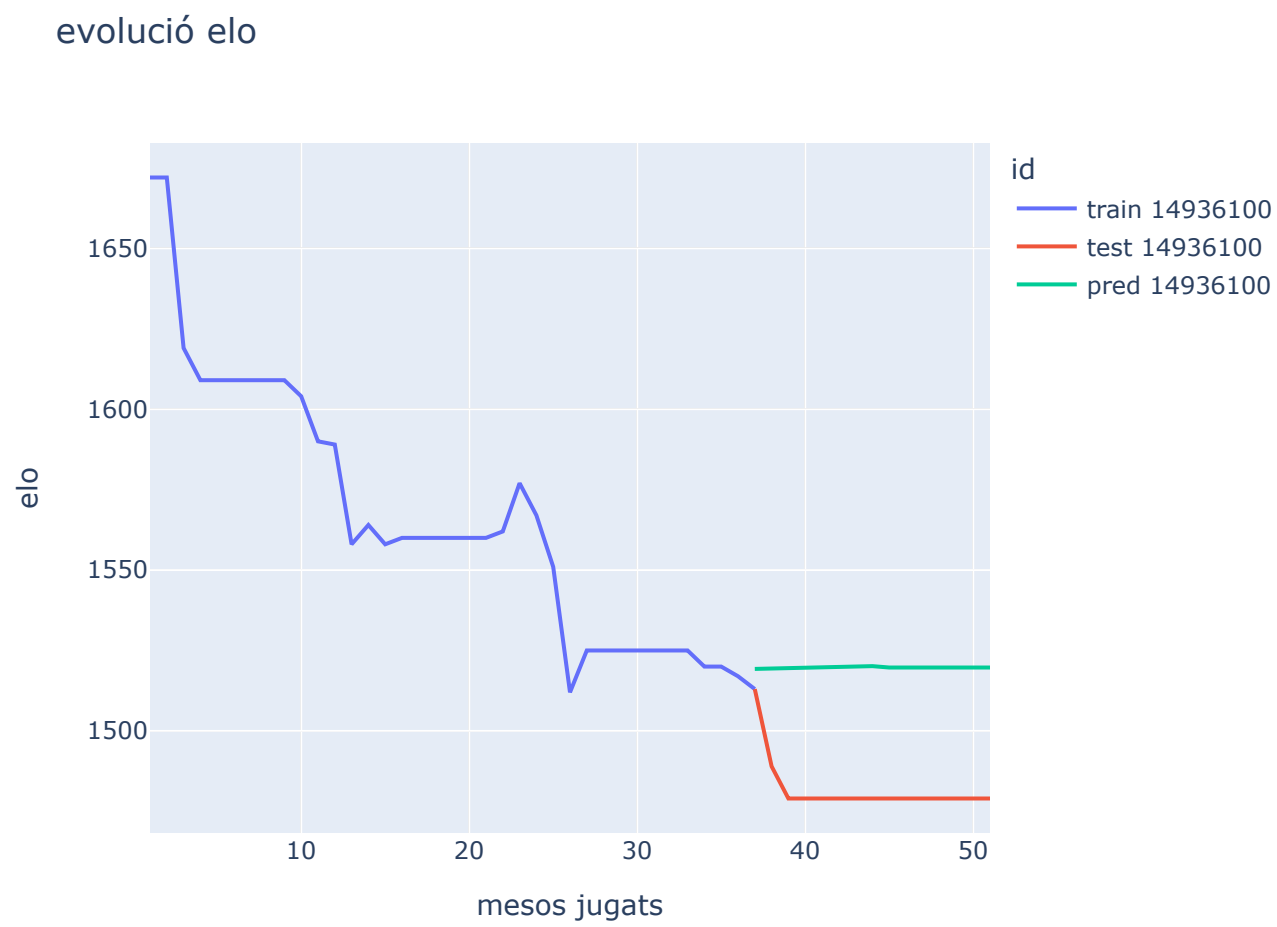
Els ratings que tenen una tendència general a descendre, els ha fallat en tots els casos. No acabo d'entendre perquè, en tots els casos marca una constant que no s'apropa gens ni al que ha tingut per entrenar, ni el que s'esperava que predis. Potser el pitjor cas pel model que hem creat, sense dubte, són els casos que mostren caiguda dels elos.

In [34]:
```
decreixer = np.array([1464019, 14936100, 953199])

generador_reultat_predicció_id(decreixer)
```

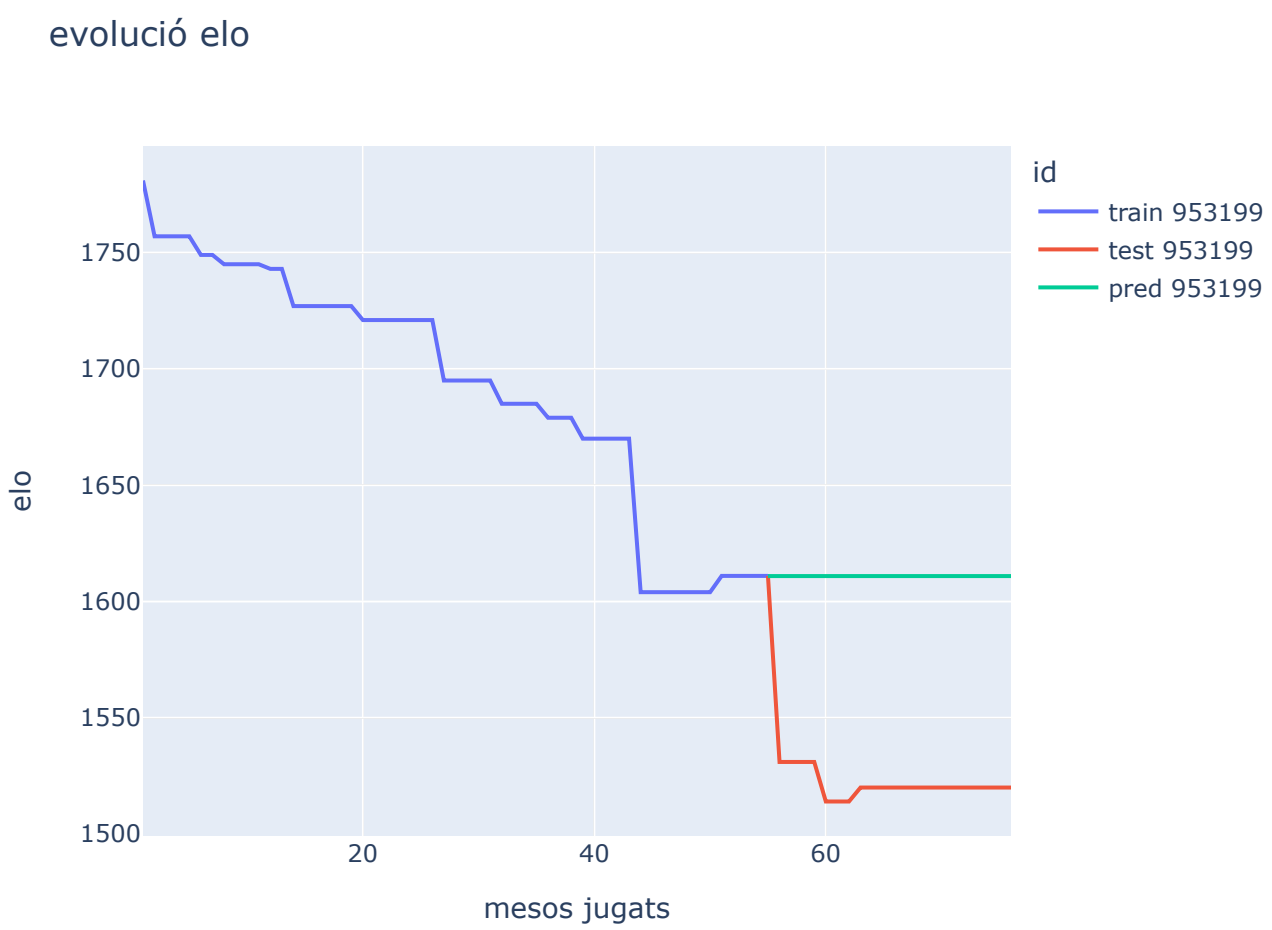
```
1464019
mean-squared error 1815.9327714285678
```




```
14936100
mean-squared error 1499.6892992000016
```



953199
mean-squared error 7702.021601454538



Amb variació

Aquests són els elos que presenten irregularitats tant de pujada com de baixada en la seva forma al llarg del temps.

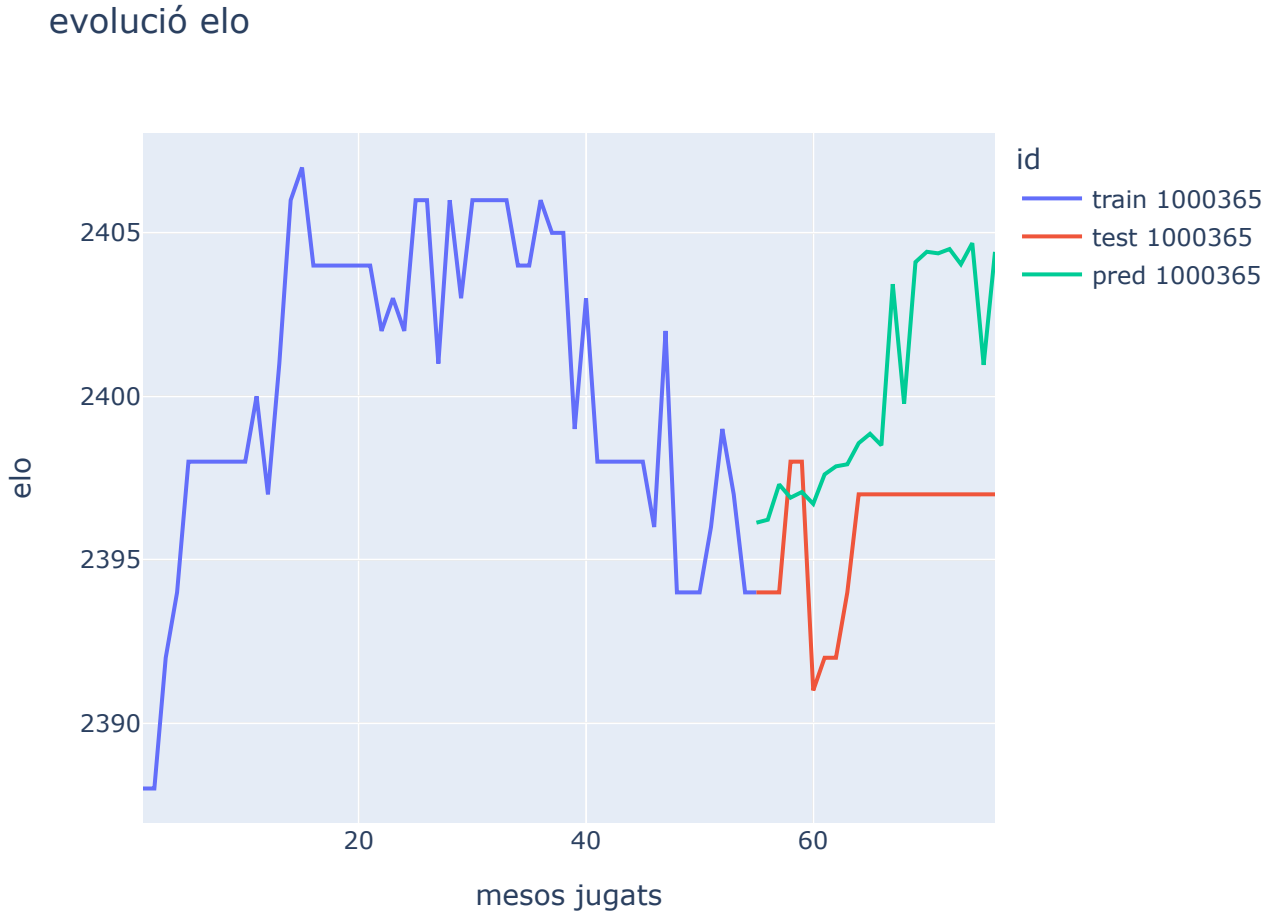
La primera, 1000365, el primer punt no l'acaba de predir bé, ja que crec que al ser més alta la part del train en general a considerat que havia de ser més. De totes formes, la part final de la predicció (65-75) em fa la sensació que a intentat representar una forma similar al que és veu als mesos 23-47, així que és veu una mica en que s'ha basat en aquest cas.

La segona, 1195190, no ha captat cap de les formes, a més el test sí que sembla que segueix minimament un patró. Tot i així, la predicció a sortit una forma gens esperada que no sé atribuir a cap raó. Al menys, el punt de partida és queda aproximat, i la tendència que segueix és minimament coherent.

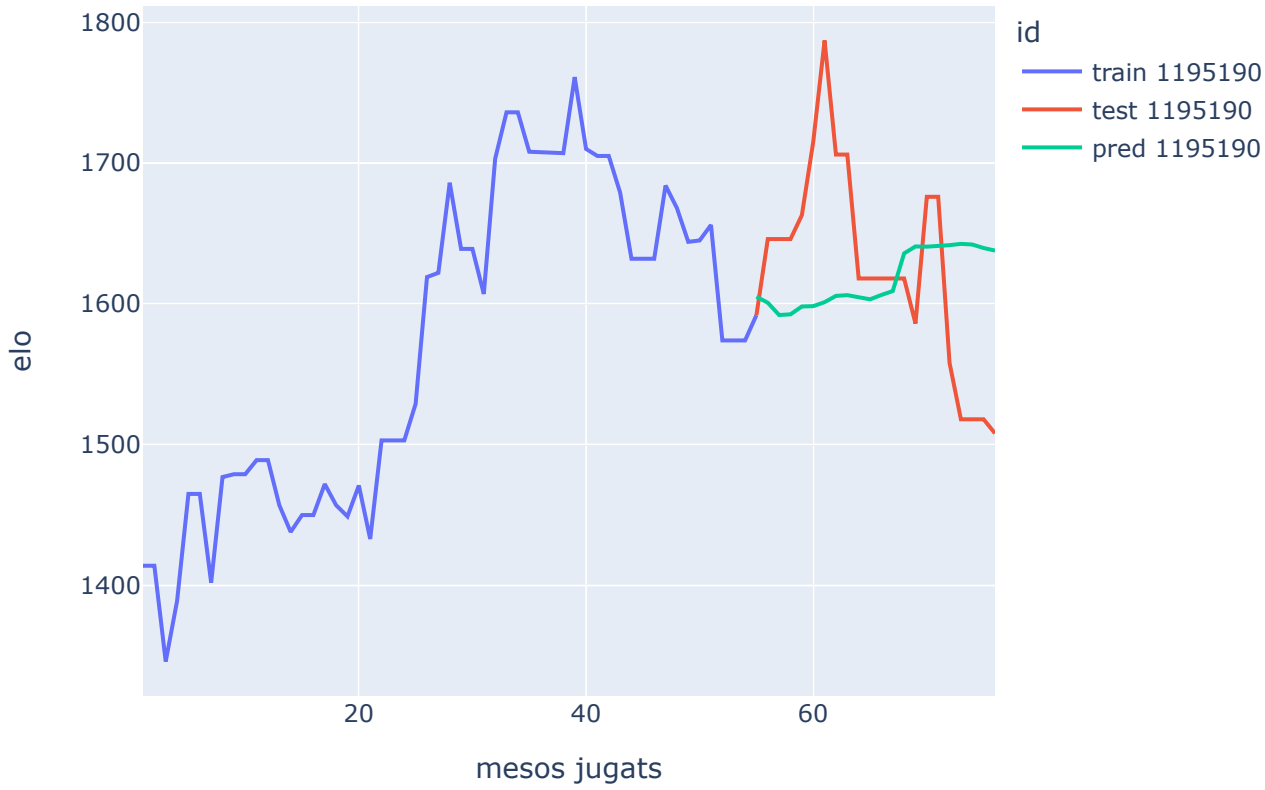
El següent, 24252255, torna a ser un test molt diferent al train, però, de totes formes, la tendència de la predicció sembla que és queda al voltant de una forma similar a la part menys variant del train, com els mesos 33 al 40 i no pren com a referència cap part més variant.

In [35]:
```
var = np.array([1000365, 1195190, 24252255])

generador_reultat_predicció_id(var)
```

```
1000365
mean-squared error 26.773483425259748
```




```
1195190
mean-squared error 7107.573555236912
```




```
24252255
mean-squared error 297.5390638225018
```

5 Conculsions

Per fer prediccions de sèries temporals en una base de dades com aquesta, que no hi ha una seqüència que és vagi repetint amb el temps és complicat, al menys amb el mètode recursiu forecasting no veiem una molt bona predicció. Li costa veure tendències i per les que acostumen a ser decréxents, ni tan sols fa més d'una linia recta. Així, no sembla que el model hagi sigut un grna encert.

Per seguir treballant en aquest, propondria utilitzar altres models de sèries temporals i/o altres regressiors, a vera sí és més factible amb altres tipus de sistemes intentar fer una predicció. Tot i això, no esperaria grans resultats, ja que tot depen de les persones, que són dificils de predir, tot i que si un juga molt als escacs és normal que seguiexi una petita tendència que si es podria visualitzar en els models de predicció.