Annexos

Annex A. Codi de la fase Preparació de les dades

```
import pandas as pd

#NETEJA I RENAME DE LES DADES INICIALS
est=pd.read_csv("dpersnomespreins19esc.csv", sep=";")
est.drop(["TIPUS_ACCES"], axis='columns', inplace=True)
est=est.rename(columns={'CODI_EXPEDIENT':'EXPED', "SEXE":"SEX",
"CP_FAMILIAR":"CP_FAM", "ANY_ACCES":"ANY_A", "NOTA_ACCES":"SELE",
"CENTRE_SECUNDARIA":"COLE", "CP_CENTRE_SEC":"CP_COLE"})
est.to_pickle("est.pkl")

fi=pd.read_csv("qfaseini19.csv", sep=";")
fi=fi.drop(fi[fi.CODI_PROGRAMA!=752].index)
fi=fi.drop(fi[fi.QUAD==0].index)
fi.drop(["CODI_PROGRAMA", "NOTA_PROF", "NOTA_NUM_AVAL", "GRUP_CLASSE"],
axis='columns', inplace=True)
fi=fi.rename(columns = {'CODI_EXPEDIENT':'EXPED','CODI_UPC_UD':'ASSIG',
"NOTA_NUM_DEF":"NOTA"})
fi.to_pickle("fi.pkl")

fni=pd.read_csv("qfasenoini19.csv", sep=";")
fni=fni.drop(fni[fni.CODI_PROGRAMA!=752].index)
fni=fni.drop(fni[fni.QUAD==0].index)
fni.drop(["CODI_PROGRAMA", "NOTA_PROF", "NOTA_NUM_AVAL", "GRUP_CLASSE"],
axis='columns', inplace=True)
fni=fni.rename(columns = {'CODI_EXPEDIENT':'EXPED','CODI_UPC_UD':'ASSIG',
"NOTA_NUM_DEF":"NOTA"})
for ass in fni["ASSIG"].unique():
   if ass[0:3]!='240':
      fni=fni.drop(fni[fni.ASSIG==ass].index)
fni.to_pickle("fni.pkl")

#CREA DATAFRAME AMB COLS: EXPED, SEX, CODI ASSIGNATURES.(PER CURSOS)
notes=fi.append(fni, ignore_index=True)
estud=est.iloc[:,[0,1,3]]
notes=estud.merge(notes, on="EXPED")
notes.sort_values(["CURS","QUAD"], inplace=True)
notes.to_pickle("notes.pkl")

mitjanes=notes.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitjanes=estud.merge(mitjanes, on="EXPED")
mitjanes=mitjanes.round(decimals=1)
mitjanes.to_pickle("mitjanes.pkl")
```

```
primconv=notes.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
primconv=estud.merge(primconv, on="EXPED")
primconv=primconv.round(decimals=1)
primconv.to_pickle("primconv.pkl")

notes10=notes[notes['CURS']==2010]
notes10.reset_index()
mitj10=notes10.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj10=mitj10.round(decimals=1)
mitj10=estud.merge(mitj10, on="EXPED")
prim10=notes10.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj10.to_pickle("m10.pkl")
prim10.to_pickle("p10.pkl")

notes11=notes[notes['CURS']==2011]
notes11.reset_index()
mitj11=notes11.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj11=mitj11.round(decimals=1)
mitj11=estud.merge(mitj11, on="EXPED")
prim11=notes11.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj11.to_pickle("m11.pkl")
prim11.to_pickle("p11.pkl")

notes12=notes[notes['CURS']==2012]
notes12.reset_index()
mitj12=notes12.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj12=mitj12.round(decimals=1)
mitj12=estud.merge(mitj12, on="EXPED")
prim12=notes12.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj12.to_pickle("m12.pkl")
prim12.to_pickle("p12.pkl")

notes13=notes[notes['CURS']==2013]
notes13.reset_index()
mitj13=notes13.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj13=mitj13.round(decimals=1)
mitj13=estud.merge(mitj13, on="EXPED")
prim13=notes13.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
```

```
mitj13.to_pickle("m13.pkl")
prim13.to_pickle("p13.pkl")

notes14=notes[notes['CURS']==2014]
notes14.reset_index()
mitj14=notes14.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj14=mitj14.round(decimals=1)
mitj14=estud.merge(mitj14, on="EXPED")
prim14=notes14.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj14.to_pickle("m14.pkl")
prim14.to_pickle("p14.pkl")

notes15=notes[notes['CURS']==2015]
notes15.reset_index()
mitj15=notes15.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj15=mitj15.round(decimals=1)
mitj15=estud.merge(mitj15, on="EXPED")
prim15=notes15.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj15.to_pickle("m15.pkl")
prim15.to_pickle("p15.pkl")

notes16=notes[notes['CURS']==2016]
notes16.reset_index()
mitj16=notes16.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj16=mitj16.round(decimals=1)
mitj16=estud.merge(mitj16, on="EXPED")
prim16=notes16.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj16.to_pickle("m16.pkl")
prim16.to_pickle("p16.pkl")

notes17=notes[notes['CURS']==2017]
notes17.reset_index()
mitj17=notes17.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='mean')
mitj17=mitj17.round(decimals=1)
mitj17=estud.merge(mitj17, on="EXPED")
prim17=notes17.pivot_table(index="EXPED", columns="ASSIG", values="NOTA",
aggfunc='first')
mitj17.to_pickle("m17.pkl")
prim17.to_pickle("p17.pkl")

#CREA DF AMB UNA COLUMNA EXPED SEX ASSIG CONVOC
```

```
convoc=notes.sort_values(["EXPED","ASSIG","CURS","QUAD"])
convoc=convoc.groupby(["EXPED","ASSIG"]).size().reset_index(name="Convoc")
convoc=estud.merge(convoc, on="EXPED")
convoc.to_pickle("convoc.pkl")
```

Annex B. Codi de l'anàlisi exploratòria de dades

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 14 09:22:52 2020

@author: laura
"""

import pandas as pd
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import normaltest
from scipy.stats import ttest_ind as tstudent
from scipy.stats import mannwhitneyu as mwtest

#LLEGIR CSV i DF
est=pd.read_pickle("est.pkl")
fi=pd.read_pickle("fi.pkl")
fni=pd.read_pickle("fni.pkl")
notes=pd.read_pickle("notes.pkl")
mitj=pd.read_pickle("mitjanes.pkl")
pconv=pd.read_pickle("primconv.pkl")
convoc=pd.read_pickle("convoc.pkl")

#LLEGIR DF
m10=pd.read_pickle("m10.pkl")
m11=pd.read_pickle("m11.pkl")
m12=pd.read_pickle("m12.pkl")
m13=pd.read_pickle("m13.pkl")
m14=pd.read_pickle("m14.pkl")
m15=pd.read_pickle("m15.pkl")
m16=pd.read_pickle("m16.pkl")
m17=pd.read_pickle("m17.pkl")
dfmitjanes=[m10,m11,m12,m13,m14,m15,m16,m17]

p10=pd.read_pickle("p10.pkl")
p11=pd.read_pickle("p11.pkl")
p12=pd.read_pickle("p12.pkl")
p13=pd.read_pickle("p13.pkl")
p14=pd.read_pickle("p14.pkl")
p15=pd.read_pickle("p15.pkl")
p16=pd.read_pickle("p16.pkl")
p17=pd.read_pickle("p17.pkl")
dfprimeres=[p10,p11,p12,p13,p14,p15,p16,p17]


#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
```

```
        '240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
años=[2010,2011,2012,2013,2014,2015,2016,2017,2018]
años_ab=[2010,2011,2012,2013,2014,2015,2016]

#FUNCIONS AUXILIARS
def normalitat(df):
    s, pvalue=normaltest(df)
    df.hist()
    qqplot(df, line="s")
    if pvalue<0.05:
        return (pvalue, 'No és normal')
    else:
        return (pvalue, "Si és normal")


#QUADRE AMB TOTS ELS TEST DE SIGNIFICACIÓ
def tt():
    test=[]
    test.append(test_nota_acces_total())
    test.append(test_mitjana_total())
    test.append(test_mitjana_fi())
    test.append(test_tot_fi())
    test.append(test_mitjana_fi_1conv())
    test.append(test_convoc_tot())
    test.append(test_fi_grup1())
    test.append(test_fi_grup2())
    test.append(test_fi_grup3())
    test.append(test_fi_grup4())
    test.append(test_fi_grup5())
    test.append(test_fni_grup1())
    test.append(test_fni_grup2())
    test.append(test_fni_grup3())
    test.append(test_fni_grup4())
    test.append(test_fni_grup5())
    test.append(test_fni_grup6())
    test.append(test_fni_grup7())
    test.append(test_fni_grup8())
    test.append(test_fni_grup9())
    test.append(test_fni_grup10())
    test.append(test_fni_grup11())
    test.append(test_fni_grup12())
    testdf=pd.DataFrame(test, columns=["Funció","Normalitat", "p-valor", "Dif. significativa?"])
    testdf=testdf.round(decimals=4)
    return testdf
```

```
#PERCENTATGE DE NOIES MATRICULADES ANUALMENT
def matricula ():
    #crea serie contant expedients per any per sexe
    m=est.groupby(['ANY_A','SEX'])['EXPED'].count()
    #crea dataframe amb index anys i columnes sexe
    idx=m.index.levels
    c=len(idx[1])
    mdf=pd.DataFrame(m.values.reshape(-1,c), index=idx[0].values, columns=idx[1].values)
    #afegeix columnes Tot i Percentatge Dones (arrodonit a 2)
    mdf.eval('Tot=D+H', inplace=True)
    mdf.eval('PercD=D*100/Tot', inplace=True)
    mdf=mdf.reset_index()
    mdf=mdf.rename(columns={"index": "ANY_A"})
    mdf.eval('PercH=100-PercD', inplace=True)
    mdf=mdf.round(decimals=2)
    mdf=mdf.rename(columns={"PercD":"Dones", "PercH":"Homes"})
    return mdf


#DIFERENCIA ENTRE LA MITJANA DE NOTA D'ACCES
def nota_acces():
    na=est.groupby(['ANY_A','SEX'])['SELE'].mean()
    idx=na.index.levels
    c=len(idx[1])
    nadf=pd.DataFrame(na.values.reshape(-1,c), index=idx[0].values, columns=idx[1].values)
    #afegeix la fila de les mitjanes totals, canvia el nom i arrodoneix valors
    nadf.loc[len(nadf.index)]=nadf.mean()
    nadf=nadf.rename(index={len(nadf.index)-1:"Total"})
    nadf.eval('Dif=D-H', inplace=True)
    nadf=nadf.round(decimals=3)
    return nadf

def nota_acces_signif():
    cols=['EXPED','SEX','ANY_A', 'SELE']
    na=est.loc[:,cols]
    na=na.dropna()
    sele=nota_acces()
    sele['Normal']=0
    sele['p-valor']=0
    sele['Dif. signif.?']=0
    for a in años:
        s=na[na.ANY_A==a]
        d=s[s.SEX=="D"]
        dones=d['SELE']
        h=s[s.SEX=="H"]
        homes=h['SELE']
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
```

```
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        sele['Normal'].loc[a]=norma
        sele['p-valor'].loc[a]=pval
        sele['Dif. signif.?'].loc[a]=sig
    sele["p-valor"]=sele["p-valor"].round(decimals=4)
    return sele


#DIFERENCIA ENTRE % D'ABANDONAMENT ABANS D'ACABAR LA FI
def abandfi_llista():
    est=pd.read_pickle("est.pkl")
    fi=pd.read_pickle("fi.pkl")
    fni=pd.read_pickle("fni.pkl")
    est=est.iloc[:,[0,1,3]] #cols: exped, sex, any_a
    #treu ANY_A=2017,2018
    est=est.drop(est[est.ANY_A==2017].index)
    est=est.drop(est[est.ANY_A==2018].index)
    fi=est.merge(fi, on="EXPED")
    fni=est.merge(fni, on="EXPED")
    fi=fi.iloc[:,[0,1,2]] #columnes exped sex, any_a
    fni=fni.iloc[:,[0,1,2,]] #columnes exped, sex, any_a
    #junta fi i fni i crea columna "_merge" que indica si index esta a fi(left), fni(right) o both
    ab=fi.merge(fni, how='outer', indicator=True)
    #tria nomes les files que estan a fi i no a fni
    aband=ab[ab._merge=="left_only"]
    aband=aband.drop_duplicates()
    abandonaments=[]
    for a in años_ab:
        año=aband[aband.ANY_A==a]
        d=len(año[año.SEX=="D"].index)
        h=len(año[año.SEX=="H"].index)
        tot=len(año.EXPED.unique())
        l=[a,tot,h,d]
        abandonaments.append(l)
    return abandonaments

def abandfi():
    abandonaments=abandfi_llista()
    mat=matricula()
    mat=mat.iloc[:,0:4]
```

```
    aban=pd.DataFrame(abandonaments, columns=["ANY_A", "aband_tot",
"aband_H","aband_D"])
    ab=pd.merge(mat,aban, on=["ANY_A"])
    ab.eval("pTot=aband_tot*100/Tot", inplace=True)
    ab.eval("pH=aband_H*100/H", inplace=True)
    ab.eval("pD=aband_D*100/D", inplace=True)
    ab.eval("Dif=pD-pH", inplace=True)
    ab.loc[len(ab.index)]=ab.mean()
    ab=ab.rename(index={len(ab.index)-1:"Mitjana"})
    ab=ab.round(decimals=2)
    return ab

#MITJANA DE TOTES LES ASSIGNATURES OBLIGATÒRIES DE LA CARRERA
def mitjana_total():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assoblig)
    mitja=mitj.loc[:,cols]
    superat=mitja.dropna()
    mtot=superat.loc[:,assoblig].mean(axis=1)
    superat.insert(len(superat.columns), "mitjana", mtot)
    mtot=superat.pivot_table(index="ANY_A", columns="SEX", values="mitjana")
    mtot.eval("Dif=D-H", inplace=True)
    mtot['Normal']=0
    mtot['p-valor']=0
    mtot['Dif. signif.?']=0
    for a in mtot.index:
        s=superat[superat.ANY_A==a]
        d=s[s.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=s[s.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mtot['Normal'].loc[a]=norma
        mtot['p-valor'].loc[a]=pval
        mtot['Dif. signif.?'].loc[a]=sig
    mtot.loc[len(mtot.index)]=mtot.mean()
    mtot=mtot.rename(index={len(mtot.index)-1:"Total"})
```

```
    mtot["p-valor"]=mtot["p-valor"].round(decimals=4)
    mtot.iloc[:,[0,1,2]]=mtot.iloc[:,[0,1,2]].round(decimals=2)
    return mtot

#MITJANA FASE INICIAL (TOTAL) amb la mitjana de les convocatòries
def mitjana_fi():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assfi)
    mitjfi=mitj.loc[:,cols]
    superatfi=mitjfi.dropna()
    #elimina matriculats 2017 i 2018 (encara poden seguir fent la fi)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2017].index)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2018].index)
    #fa la mitjana de les columnes (a cada fila)
    mfi=superatfi.loc[:,assfi].mean(axis=1)
    superatfi.insert(len(superatfi.columns), "mitjana", mfi)
    mfi=superatfi.pivot_table(index="ANY_A", columns="SEX", values="mitjana")
    mfi.eval("Dif=D-H", inplace=True) #afegeix col diferencia
    mfi['Normal']=0
    mfi['p-valor']=0
    mfi['Dif. signif.?']=0
    años=[2010,2011,2012,2013,2014,2015,2016]
    for a in años:
        s=superatfi[superatfi.ANY_A==a]
        d=s[s.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=s[s.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mfi['Normal'].loc[a]=norma
        mfi['p-valor'].loc[a]=pval
        mfi['Dif. signif.?'].loc[a]=sig
    mfi.loc[len(mfi.index)]=mfi.mean() #afegeix mitjana tot a la ultima fila
    mfi=mfi.rename(index={len(mfi.index)-1:"Total"}) #anomena ultima fila
    mfi["p-valor"]=mfi["p-valor"].round(decimals=4)
    mfi.iloc[:,[0,1,2]]=mfi.iloc[:,[0,1,2]].round(decimals=2)
    return mfi
```

```
#MITJANA FASE INICIAL (TOTAL) amb la nota primera convocatoria
def mitjana_fi_1conv():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assfi)
    primfi=pconv.loc[:,cols]
    superatfi=primfi.dropna()
    #elimina matriculats 2017 i 2018 (encara poden seguir fent la fi)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2017].index)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2018].index)
    #fa la mitjana de les columnes (a cada fila)
    pfi=superatfi.loc[:,assfi].mean(axis=1)
    superatfi.insert(len(superatfi.columns), "mitjana", pfi)
    pfi=superatfi.pivot_table(index="ANY_A", columns="SEX", values="mitjana")
    pfi.eval("Dif=D-H", inplace=True) #afegeix col diferencia
    pfi.eval("Dif=D-H", inplace=True) #afegeix col diferencia
    pfi['Normal']=0
    pfi['p-valor']=0
    pfi['Dif. signif.?']=0
    años=[2010,2011,2012,2013,2014,2015,2016]
    for a in años:
        s=superatfi[superatfi.ANY_A==a]
        d=s[s.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=s[s.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        pfi['Normal'].loc[a]=norma
        pfi['p-valor'].loc[a]=pval
        pfi['Dif. signif.?'].loc[a]=sig
    pfi.loc[len(pfi.index)]=pfi.mean() #afegeix mitjana tot a la ultima fila
    pfi=pfi.rename(index={len(pfi.index)-1:"Total"}) #anomena ultima fila
    pfi["p-valor"]=pfi["p-valor"].round(decimals=4)
    pfi.iloc[:,[0,1,2]]=pfi.iloc[:,[0,1,2]].round(decimals=2)
    return pfi
```

```
#ANALISI CONVOCATORIES TOTALS
def convoc_tot():
    cols=["EXPED","SEX","ANY_A"]
    estud=est.loc[:,cols]
    conv=convoc.pivot_table(index="EXPED",columns="ASSIG", values="Convoc")
    conv=estud.merge(conv, on="EXPED")
    cols.extend(assoblig)
    conv=conv.loc[:,cols]
    superat=conv.dropna()
    #fa la mitjana de les columnes (a cada fila)
    m=superat.loc[:,assoblig].mean(axis=1)
    superat.insert(len(superat.columns), "mitjana", m)
    m=superat.pivot_table(index="ANY_A", columns="SEX", values="mitjana")
    m.loc[len(m.index)]=m.mean() #afegeix mitjana tot a la ultima fila
    m=m.rename(index={len(m.index)-1:"Total"}) #anomena ultima fila
    m.eval("Dif=D-H", inplace=True) #afegeix col diferencia
    m=m.round(decimals=3)
    return m

def convoc_piechart():
    cH=[]
    convH=convoc[convoc.SEX=="H"]
    for num in range (1,4):
        cH.append([num,convH[convH.Convoc==num].count().EXPED])
    cH.append(['>3',convH[convH.Convoc>3].count().EXPED])
    convdf=pd.DataFrame(cH, columns=["Convocs", "H"])
    cD=[]
    convD=convoc[convoc.SEX=="D"]
    for num in range (1,4):
        cD.append(convD[convD.Convoc==num].count().EXPED)
    cD.append(convD[convD.Convoc>3].count().EXPED)
    convdf.insert(len(convdf.columns),"D", cD)
    convdf.index=convdf.Convocs
    convdf.plot.pie(y="H", autopct="%1.1f%%", explode=(0,0,0.2,0.2))
    convdf.plot.pie(y="D", autopct="%1.1f%%", explode=(0,0,0.2,0.2))
    return convdf

#ANALISI CONVOCATORIES FASE INICIAL
def convoc_fi():
    cols=["EXPED","SEX","ANY_A"]
    estud=est.loc[:,cols]
    conv=convoc.pivot_table(index="EXPED",columns="ASSIG", values="Convoc")
    conv=estud.merge(conv, on="EXPED")
    cols.extend(assfi)
    conv=conv.loc[:,cols]
    superatfi=conv.dropna()
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2017].index)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2018].index)
```

```
    #fa la mitjana de les columnes (a cada fila)
    mfi=superatfi.loc[:,assfi].mean(axis=1)
    superatfi.insert(len(superatfi.columns), "mitjana", mfi)
    mfi=superatfi.pivot_table(index="ANY_A", columns="SEX", values="mitjana")
    mfi.loc[len(mfi.index)]=mfi.mean() #afegeix mitjana tot a la ultima fila
    mfi=mfi.rename(index={len(mfi.index)-1:"Total"}) #anomena ultima fila
    mfi.eval("Dif=D-H", inplace=True) #afegeix col diferencia
    mfi=mfi.round(decimals=2)
    return mfi

def convoc_fi_piechart():
    cH=[]
    convH=convoc[convoc.SEX=="H"]
    for num in range (1,4):
        cH.append([num,convH[convH.Convoc==num].count().EXPED])
    cH.append(['>3',convH[convH.Convoc>3].count().EXPED])
    convdf=pd.DataFrame(cH, columns=["Convocs", "H"])
    cD=[]
    convD=convoc[convoc.SEX=="D"]
    for num in range (1,4):
        cD.append(convD[convD.Convoc==num].count().EXPED)
    cD.append(convD[convD.Convoc>3].count().EXPED)
    convdf.insert(len(convdf.columns),"D", cD)
    convdf.index=convdf.Convocs
    convdf.plot.pie(y="H", autopct="%1.1f%%", explode=(0,0,0.2,0.2))
    convdf.plot.pie(y="D", autopct="%1.1f%%", explode=(0,0,0.2,0.2))
    return convdf

#ANALISI PER GRUPS D'ASSIGNATURES FASE INICIAL
#GRUP 1: Matemàtiques (Àlgebra 11, Càlcul I 12, Geometria 21, Càlcul II 22)
def fi_grup1():
    cols=['EXPED','SEX']
    ass=[240011,240012,240021,240022]
    cols.extend(ass)
    mgrup=pd.DataFrame(columns=['D','H','Normal','p-valor','Dif. signif.?'])
    for df in dfmitjanes:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,ass].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
```

```
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    años=[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
    añosdf=pd.DataFrame(años)
    grup=añosdf.rename(columns={0:'CURS'})
    grup['D']=mgrup['D']
    grup['H']=mgrup['H']
    grup.eval("Dif=D-H", inplace=True)
    grup['Normal']=mgrup['Normal']
    grup['p-valor']=mgrup['p-valor']
    grup['Dif. signif.?']=mgrup['Dif. signif.?']
    grup['D']=grup['D'].round(decimals=2)
    grup['H']=grup['H'].round(decimals=2)
    grup['Dif']=grup['Dif'].round(decimals=3)
    grup['p-valor']=grup['p-valor'].round(decimals=4)
    return grup

#Grup 2.Física: Mec.Fonam. 13, Termo. Fonam. 23
def fi_grup2():
    cols=['EXPED','SEX']
    ass=[240013,240023]
    cols.extend(ass)
    mgrup=pd.DataFrame(columns=['D','H','Normal','p-valor','Dif. signif.?'])
    for df in dfmitjanes:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,ass].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
```

```
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    años=[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
    añosdf=pd.DataFrame(años)
    grup=añosdf.rename(columns={0:'CURS'})
    grup['D']=mgrup['D']
    grup['H']=mgrup['H']
    grup.eval("Dif=D-H", inplace=True)
    grup['Normal']=mgrup['Normal']
    grup['p-valor']=mgrup['p-valor']
    grup['Dif. signif.?']=mgrup['Dif. signif.?']
    grup['D']=grup['D'].round(decimals=2)
    grup['H']=grup['H'].round(decimals=2)
    grup['Dif']=grup['Dif'].round(decimals=3)
    grup['p-valor']=grup['p-valor'].round(decimals=4)
    return grup

#Grup 3. Fonaments d'informàtica 15
def fi_grup3():
    cols=['EXPED','SEX']
    ass=[240015]
    cols.extend(ass)
    mgrup=pd.DataFrame(columns=['D','H','Normal','p-valor','Dif. signif.?'])
    for df in dfmitjanes:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,ass].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
```

```
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    años=[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
    añosdf=pd.DataFrame(años)
    grup=añosdf.rename(columns={0:'CURS'})
    grup['D']=mgrup['D']
    grup['H']=mgrup['H']
    grup.eval("Dif=D-H", inplace=True)
    grup['Normal']=mgrup['Normal']
    grup['p-valor']=mgrup['p-valor']
    grup['Dif. signif.?']=mgrup['Dif. signif.?']
    grup['D']=grup['D'].round(decimals=2)
    grup['H']=grup['H'].round(decimals=2)
    grup['Dif']=grup['Dif'].round(decimals=3)
    grup['p-valor']=grup['p-valor'].round(decimals=4)
    return grup

#Grup 4. Expressió gràfica 25
def fi_grup4():
    cols=['EXPED','SEX']
    ass=[240025]
    cols.extend(ass)
    mgrup=pd.DataFrame(columns=['D','H','Normal','p-valor','Dif. signif.?'])
    for df in dfmitjanes:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,ass].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
```

```
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    años=[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
    añosdf=pd.DataFrame(años)
    grup=añosdf.rename(columns={0:'CURS'})
    grup['D']=mgrup['D']
    grup['H']=mgrup['H']
    grup.eval("Dif=D-H", inplace=True)
    grup['Normal']=mgrup['Normal']
    grup['p-valor']=mgrup['p-valor']
    grup['Dif. signif.?']=mgrup['Dif. signif.?']
    grup['D']=grup['D'].round(decimals=2)
    grup['H']=grup['H'].round(decimals=2)
    grup['Dif']=grup['Dif'].round(decimals=3)
    grup['p-valor']=grup['p-valor'].round(decimals=4)
    return grup

#Grup 5. Quimica I 14, Quimica II 24
def fi_grup5():
    cols=['EXPED','SEX']
    ass=[240014,240024]
    cols.extend(ass)
    mgrup=pd.DataFrame(columns=['D','H','Normal','p-valor','Dif. signif.?'])
    for df in dfmitjanes:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,ass].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
```

```
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    años=[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017]
    añosdf=pd.DataFrame(años)
    grup=añosdf.rename(columns={0:'CURS'})
    grup['D']=mgrup['D']
    grup['H']=mgrup['H']
    grup.eval("Dif=D-H", inplace=True)
    grup['Normal']=mgrup['Normal']
    grup['p-valor']=mgrup['p-valor']
    grup['Dif. signif.?']=mgrup['Dif. signif.?']
    grup['D']=grup['D'].round(decimals=2)
    grup['H']=grup['H'].round(decimals=2)
    grup['Dif']=grup['Dif'].round(decimals=3)
    grup['p-valor']=grup['p-valor'].round(decimals=4)
    return grup


#FASE NO INICIAL

def fni_grup1():
    cols=['EXPED','SEX']
    ass=['240132']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
```

```
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
               pval=tstudent(homes,dones)[1]
               norma='Sí'
            else:
               pval=mwtest(homes, dones)[1]
               norma='No'
            if pval<0.05:
               sig='Sí'
            else:
               sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
         else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
      mgrup.reset_index(inplace=True)
      del mgrup['index']
      mgrup.eval('Dif=D-H', inplace=True)
      mgrup['D']=mgrup['D'].round(decimals=2)
      mgrup['H']=mgrup['H'].round(decimals=2)
      mgrup['Dif']=mgrup['Dif'].round(decimals=3)
      mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
      return mgrup

   def fni_grup2():
      cols=['EXPED','SEX']
      ass=['240133','240141','240054','240063','240073']
      assigs=[]
      mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
      año=2010
```

```
for df in dfmitjanes:
    for assig in ass:
        if assig in df.columns:
            cols.append(assig)
            assigs.append(assig)
    if len(cols)>2:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,assigs].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['CURS']=año
        año=año+1
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
        cols=['EXPED','SEX']
    else:
        m=pd.DataFrame()
        m['CURS']=[año]
        año=año+1
        m['D']=[0.0]
        m['H']=[0.0]
        m['Normal']=['No data']
        m['p-valor']=[0.0]
        m['Dif. signif.?']=['No data']
        mgrup=mgrup.append(m)
mgrup.reset_index(inplace=True)
del mgrup['index']
mgrup.eval('Dif=D-H', inplace=True)
mgrup['D']=mgrup['D'].round(decimals=2)
mgrup['H']=mgrup['H'].round(decimals=2)
```

```
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup3():
    cols=['EXPED','SEX']
    ass=['240031','240053','240161','240072']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
                pval=tstudent(homes,dones)[1]
                norma='Sí'
            else:
                pval=mwtest(homes, dones)[1]
                norma='No'
            if pval<0.05:
                sig='Sí'
            else:
                sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
        else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
```

```
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup4():
    cols=['EXPED','SEX']
    ass=['240044','240064','240071']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
                pval=tstudent(homes,dones)[1]
                norma='Sí'
            else:
                pval=mwtest(homes, dones)[1]
                norma='No'
            if pval<0.05:
                sig='Sí'
            else:
                sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
```

```
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
        else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup5():
    cols=['EXPED','SEX']
    ass=['240032','240131']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
                pval=tstudent(homes,dones)[1]
                norma='Sí'
```

```
            else:
               pval=mwtest(homes, dones)[1]
               norma='No'
            if pval<0.05:
               sig='Sí'
            else:
               sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
         else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
   mgrup.reset_index(inplace=True)
   del mgrup['index']
   mgrup.eval('Dif=D-H', inplace=True)
   mgrup['D']=mgrup['D'].round(decimals=2)
   mgrup['H']=mgrup['H'].round(decimals=2)
   mgrup['Dif']=mgrup['Dif'].round(decimals=3)
   mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
   return mgrup

def fni_grup6():
   cols=['EXPED','SEX']
   ass=['240043','240172']
   assigs=[]
   mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
   año=2010
   for df in dfmitjanes:
      for assig in ass:
         if assig in df.columns:
            cols.append(assig)
            assigs.append(assig)
      if len(cols)>2:
         mitjg=df.loc[:,cols]
         mg=mitjg.loc[:,assigs].mean(axis=1)
```

```
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
               pval=tstudent(homes,dones)[1]
               norma='Sí'
            else:
               pval=mwtest(homes, dones)[1]
               norma='No'
            if pval<0.05:
               sig='Sí'
            else:
               sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
         else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
      mgrup.reset_index(inplace=True)
      del mgrup['index']
      mgrup.eval('Dif=D-H', inplace=True)
      mgrup['D']=mgrup['D'].round(decimals=2)
      mgrup['H']=mgrup['H'].round(decimals=2)
      mgrup['Dif']=mgrup['Dif'].round(decimals=3)
      mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
      return mgrup

   def fni_grup7():
      cols=['EXPED','SEX']
      ass=['240042','240055']
      assigs=[]
```

```
mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
año=2010
for df in dfmitjanes:
    for assig in ass:
        if assig in df.columns:
            cols.append(assig)
            assigs.append(assig)
    if len(cols)>2:
        mitjg=df.loc[:,cols]
        mg=mitjg.loc[:,assigs].mean(axis=1)
        mitjg.insert(len(mitjg.columns), "mitjana", mg)
        d=mitjg[mitjg.SEX=="D"]
        do=d['mitjana']
        dones=do.dropna()
        h=mitjg[mitjg.SEX=="H"]
        ho=h['mitjana']
        homes=ho.dropna()
        if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
            pval=tstudent(homes,dones)[1]
            norma='Sí'
        else:
            pval=mwtest(homes, dones)[1]
            norma='No'
        if pval<0.05:
            sig='Sí'
        else:
            sig='No'
        mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
        mgany['CURS']=año
        año=año+1
        mgany['Normal']=norma
        mgany['p-valor']=pval
        mgany['Dif. signif.?']=sig
        mgrup=mgrup.append(mgany)
        cols=['EXPED','SEX']
    else:
        m=pd.DataFrame()
        m['CURS']=[año]
        año=año+1
        m['D']=[0.0]
        m['H']=[0.0]
        m['Normal']=['No data']
        m['p-valor']=[0.0]
        m['Dif. signif.?']=['No data']
        mgrup=mgrup.append(m)
mgrup.reset_index(inplace=True)
del mgrup['index']
mgrup.eval('Dif=D-H', inplace=True)
```

```
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup8():
    cols=['EXPED','SEX']
    ass=['240041','240062','240162']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
                pval=tstudent(homes,dones)[1]
                norma='Sí'
            else:
                pval=mwtest(homes, dones)[1]
                norma='No'
            if pval<0.05:
                sig='Sí'
            else:
                sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
        else:
            m=pd.DataFrame()
            m['CURS']=[año]
```

```
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup9():
    cols=['EXPED','SEX']
    ass=['240033','240151']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
                pval=tstudent(homes,dones)[1]
                norma='Sí'
            else:
                pval=mwtest(homes, dones)[1]
                norma='No'
            if pval<0.05:
                sig='Sí'
            else:
                sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
```

```
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
        else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup10():
    cols=['EXPED','SEX']
    ass=['240052','240171']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
        for assig in ass:
            if assig in df.columns:
                cols.append(assig)
                assigs.append(assig)
        if len(cols)>2:
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
```

```
            pval=tstudent(homes,dones)[1]
            norma='Sí'
          else:
            pval=mwtest(homes, dones)[1]
            norma='No'
          if pval<0.05:
            sig='Sí'
          else:
            sig='No'
          mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
          mgany['CURS']=año
          año=año+1
          mgany['Normal']=norma
          mgany['p-valor']=pval
          mgany['Dif. signif.?']=sig
          mgrup=mgrup.append(mgany)
          cols=['EXPED','SEX']
        else:
          m=pd.DataFrame()
          m['CURS']=[año]
          año=año+1
          m['D']=[0.0]
          m['H']=[0.0]
          m['Normal']=['No data']
          m['p-valor']=[0.0]
          m['Dif. signif.?']=['No data']
          mgrup=mgrup.append(m)
    mgrup.reset_index(inplace=True)
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

def fni_grup11():
    cols=['EXPED','SEX']
    ass=['240061']
    assigs=[]
    mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
    año=2010
    for df in dfmitjanes:
      for assig in ass:
        if assig in df.columns:
          cols.append(assig)
          assigs.append(assig)
      if len(cols)>2:
```

```
            mitjg=df.loc[:,cols]
            mg=mitjg.loc[:,assigs].mean(axis=1)
            mitjg.insert(len(mitjg.columns), "mitjana", mg)
            d=mitjg[mitjg.SEX=="D"]
            do=d['mitjana']
            dones=do.dropna()
            h=mitjg[mitjg.SEX=="H"]
            ho=h['mitjana']
            homes=ho.dropna()
            if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
               pval=tstudent(homes,dones)[1]
               norma='Sí'
            else:
               pval=mwtest(homes, dones)[1]
               norma='No'
            if pval<0.05:
               sig='Sí'
            else:
               sig='No'
            mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
            mgany['CURS']=año
            año=año+1
            mgany['Normal']=norma
            mgany['p-valor']=pval
            mgany['Dif. signif.?']=sig
            mgrup=mgrup.append(mgany)
            cols=['EXPED','SEX']
         else:
            m=pd.DataFrame()
            m['CURS']=[año]
            año=año+1
            m['D']=[0.0]
            m['H']=[0.0]
            m['Normal']=['No data']
            m['p-valor']=[0.0]
            m['Dif. signif.?']=['No data']
            mgrup=mgrup.append(m)
      mgrup.reset_index(inplace=True)
      del mgrup['index']
      mgrup.eval('Dif=D-H', inplace=True)
      mgrup['D']=mgrup['D'].round(decimals=2)
      mgrup['H']=mgrup['H'].round(decimals=2)
      mgrup['Dif']=mgrup['Dif'].round(decimals=3)
      mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
      return mgrup

   def fni_grup12():
      cols=['EXPED','SEX']
```

```
ass=['240051']
assigs=[]
mgrup=pd.DataFrame(columns=['CURS','D','H','Normal','p-valor','Dif. signif.?'])
año=2010
for df in dfmitjanes:
   for assig in ass:
      if assig in df.columns:
         cols.append(assig)
         assigs.append(assig)
   if len(cols)>2:
      mitjg=df.loc[:,cols]
      mg=mitjg.loc[:,assigs].mean(axis=1)
      mitjg.insert(len(mitjg.columns), "mitjana", mg)
      d=mitjg[mitjg.SEX=="D"]
      do=d['mitjana']
      dones=do.dropna()
      h=mitjg[mitjg.SEX=="H"]
      ho=h['mitjana']
      homes=ho.dropna()
      if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
         pval=tstudent(homes,dones)[1]
         norma='Sí'
      else:
         pval=mwtest(homes, dones)[1]
         norma='No'
      if pval<0.05:
         sig='Sí'
      else:
         sig='No'
      mgany=mitjg.pivot_table(columns="SEX", values="mitjana")
      mgany['CURS']=año
      año=año+1
      mgany['Normal']=norma
      mgany['p-valor']=pval
      mgany['Dif. signif.?']=sig
      mgrup=mgrup.append(mgany)
      cols=['EXPED','SEX']
   else:
      m=pd.DataFrame()
      m['CURS']=[año]
      año=año+1
      m['D']=[0.0]
      m['H']=[0.0]
      m['Normal']=['No data']
      m['p-valor']=[0.0]
      m['Dif. signif.?']=['No data']
      mgrup=mgrup.append(m)
mgrup.reset_index(inplace=True)
```

```
    del mgrup['index']
    mgrup.eval('Dif=D-H', inplace=True)
    mgrup['D']=mgrup['D'].round(decimals=2)
    mgrup['H']=mgrup['H'].round(decimals=2)
    mgrup['Dif']=mgrup['Dif'].round(decimals=3)
    mgrup['p-valor']=mgrup['p-valor'].round(decimals=4)
    return mgrup

#FUNCIONS TEST (totals)

def test_nota_acces_total():
    sele=est.loc[:,["SEX","ANY_A", "SELE"]]
    sele=sele.round(decimals=2)
    homes=sele[sele.SEX=="H"]
    dones=sele[sele.SEX=="D"]
    if normaltest(homes.SELE)[1]>0.05 and normaltest(dones.SELE)[1]>0.05:
        pval=tstudent(homes["SELE"],dones["SELE"])[1]
        norma='Sí'
    else:
        pval=mwtest(homes["SELE"], dones["SELE"])[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['Nota Acces',norma, pval, sig]

def test_mitjana_total():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assoblig)
    mitja=mitj.loc[:,cols]
    superat=mitja.dropna()
    mtot=superat.loc[:,assoblig].mean(axis=1)
    superat.insert(len(superat.columns), "mitjana", mtot)
    homes=superat[superat.SEX=="H"]
    dones=superat[superat.SEX=="D"]
    if normaltest(homes.mitjana)[1]>0.05 and normaltest(dones.mitjana)[1]>0.05:
        pval=tstudent(homes["mitjana"],dones["mitjana"])[1]
        norma='Sí'
    else:
        pval=mwtest(homes["mitjana"], dones["mitjana"])[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['Mitjana total',norma, pval, sig]
```

```
def test_mitjana_fi():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assfi)
    mitjfi=mitj.loc[:,cols]
    superatfi=mitjfi.dropna()
    #elimina matriculats 2017 i 2018 (encara poden seguir fent la fi)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2017].index)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2018].index)
    #fa la mitjana de les columnes (a cada fila)
    mfi=superatfi.loc[:,assfi].mean(axis=1)
    superatfi.insert(len(superatfi.columns), "mitjana", mfi)
    homes=superatfi[superatfi.SEX=="H"]
    dones=superatfi[superatfi.SEX=="D"]
    if normaltest(homes.mitjana)[1]>0.05 and normaltest(dones.mitjana)[1]>0.05:
        pval=tstudent(homes["mitjana"],dones["mitjana"])[1]
        norma='Sí'
    else:
        pval=mwtest(homes["mitjana"], dones["mitjana"])[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['Mitjana FI', norma, pval, sig]

def test_tot_fi():
    cols=['EXPED','SEX','ANY_A']
    ass=assfi
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    grup=grup.dropna()
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
```

```
        sig='No'
    return ['Tot FI', norma, pval, sig]

def test_mitjana_fi_1conv():
    cols=['EXPED','SEX','ANY_A']
    cols.extend(assfi)
    primfi=pconv.loc[:,cols]
    superatfi=primfi.dropna()
    #elimina matriculats 2017 i 2018 (encara poden seguir fent la fi)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2017].index)
    superatfi=superatfi.drop(superatfi[superatfi.ANY_A==2018].index)
    #fa la mitjana de les columnes (a cada fila)
    pfi=superatfi.loc[:,assfi].mean(axis=1)
    superatfi.insert(len(superatfi.columns), "mitjana", pfi)
    homes=superatfi[superatfi.SEX=="H"]
    dones=superatfi[superatfi.SEX=="D"]
    if normaltest(homes.mitjana)[1]>0.05 and normaltest(dones.mitjana)[1]>0.05:
        pval=tstudent(homes["mitjana"],dones["mitjana"])[1]
        norma='Sí'
    else:
        pval=mwtest(homes["mitjana"], dones["mitjana"])[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['Mitjana FI 1a Conv', norma, pval, sig]

def test_convoc_tot():
    cols=["EXPED","SEX","ANY_A"]
    estud=est.loc[:,cols]
    conv=convoc.pivot_table(index="EXPED",columns="ASSIG", values="Convoc")
    conv=estud.merge(conv, on="EXPED")
    cols.extend(assoblig)
    conv=conv.loc[:,cols]
    superat=conv.dropna()
    #fa la mitjana de les columnes (a cada fila)
    m=superat.loc[:,assoblig].mean(axis=1)
    superat.insert(len(superat.columns), "mitjana", m)
    homes=superat[superat.SEX=="H"]
    dones=superat[superat.SEX=="D"]
    if normaltest(homes.mitjana)[1]>0.05 and normaltest(dones.mitjana)[1]>0.05:
        pval=tstudent(homes["mitjana"],dones["mitjana"])[1]
        norma='Sí'
    else:
        pval=mwtest(homes["mitjana"], dones["mitjana"])[1]
        norma='No'
    if pval<0.05:
```

```
        sig='Sí'
    else:
        sig='No'
    return ['Mitjana Convocatories', norma, pval, sig]

def test_fi_grup1():
    cols=['EXPED','SEX']
    ass=[240011,240012,240021,240022]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FI g1: Matemàtiques', norma, pval, sig]

def test_fi_grup2():
    cols=['EXPED','SEX']
    ass=[240013,240023]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
```

```
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FI g2: Física', norma, pval, sig]

def test_fi_grup3():
    cols=['EXPED','SEX']
    ass=[240015]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FI g3: Info', norma, pval, sig]

def test_fi_grup4():
    cols=['EXPED','SEX']
    ass=[240025]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
```

```
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FI g4: Expre', norma, pval, sig]

def test_fi_grup5():
    cols=['EXPED','SEX']
    ass=[240014,240024]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FI g5: Quimica', norma, pval, sig]

#FASE NO INICIAL
def test_fni_grup1():
    cols=['EXPED','SEX']
    ass=["240132"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
```

```
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g1: CS', norma, pval, sig]

def test_fni_grup2():
    cols=['EXPED','SEX']
    ass=["240133","240141","240054","240063","240073"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g2: EM i RMEE', norma, pval, sig]

def test_fni_grup3():
    cols=['EXPED','SEX']
    ass=["240031","240053","240161","240072"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
```

```
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g3: FIS, DEE, EEL', norma, pval, sig]

def test_fni_grup4():
    cols=['EXPED','SEX']
    ass=["240044","240064","240071"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g4: ETSEIB, EPC', norma, pval, sig]

def test_fni_grup5():
    cols=['EXPED','SEX']
    ass=["240031","240131"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
```

```
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g5: MAT', norma, pval, sig]

def test_fni_grup6():
    cols=['EXPED','SEX']
    ass=["240043","240172"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g6: ESAII', norma, pval, sig]

def test_fni_grup7():
    cols=['EXPED','SEX']
    ass=["240042","240055"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
```

```
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g7: EIO', norma, pval, sig]

def test_fni_grup8():
    cols=['EXPED','SEX']
    ass=["240041","240062","240162"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g8: OE', norma, pval, sig]

def test_fni_grup9():
    cols=['EXPED','SEX']
    ass=["240033","240151"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
```

```
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g9: CEM', norma, pval, sig]

def test_fni_grup10():
    cols=['EXPED','SEX']
    ass=["240052","240171"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g10: MMT', norma, pval, sig]

def test_fni_grup11():
    cols=['EXPED','SEX']
    ass=["240061"]
```

```
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g11: MF', norma, pval, sig]

def test_fni_grup12():
    cols=['EXPED','SEX']
    ass=["240051"]
    cols.extend(ass)
    grup=mitj.loc[:,cols]
    mg=grup.loc[:,ass].mean(axis=1)
    grup.insert(len(grup.columns),'mitjana',mg)
    d=grup[grup.SEX=="D"]
    do=d["mitjana"]
    dones=do.dropna()
    h=grup[grup.SEX=="H"]
    ho=h["mitjana"]
    homes=ho.dropna()
    if normaltest(homes)[1]>0.05 and normaltest(dones)[1]>0.05:
        pval=tstudent(homes,dones)[1]
        norma='Sí'
    else:
        pval=mwtest(homes, dones)[1]
        norma='No'
    if pval<0.05:
        sig='Sí'
    else:
        sig='No'
    return ['FNI g12: EQ', norma, pval, sig]

#CREA CSV RESULTATS
```

```
import pandas as pd
import Analisi_Inicial as AI

#test significacio
ttdf=AI.tt()
ttdf.to_csv("resultats_testsignif.csv", encoding="utf-8-sig", sep=";")

#matriculació
mat_df=AI.matricula()
mat_df.to_csv("resultats_matriculacio.csv", encoding="utf-8-sig", sep=";")
#nota d'acces
na_df=AI.nota_acces_signif()
na_df.to_csv("resultats_nota_acces.csv", encoding="utf-8-sig", sep=";")

#abandonament abans d'acabar fase inicial
aband_fi=AI.abandfi()
aband_fi.to_csv("resultats_abandonament_fi.csv", encoding="utf-8-sig", sep=";")

#mitjana total
m_tot=AI.mitjana_total()
m_tot.to_csv("resultats_mitjana_total.csv", encoding="utf-8-sig", sep=";")


#mitjana fi amb la mitjana de les convocatòries
m_fi=AI.mitjana_fi()
m_fi.to_csv("resultats_mitjana_fi.csv", encoding="utf-8-sig", sep=";")

#mitjana fi amb la 1a convocatòria
m_fi1c=AI.mitjana_fi_1conv()
m_fi1c.to_csv("resultats_mitjana_fi_1conv.csv", encoding="utf-8-sig", sep=";")

#fase inicial per grups d'assignatures
g1=AI.fi_grup1()
g1.to_csv("resultats_fi_g1.csv", encoding="utf-8-sig", sep=";")
g2=AI.fi_grup2()
g2.to_csv("resultats_fi_g2.csv", encoding="utf-8-sig", sep=";")
g3=AI.fi_grup3()
g3.to_csv("resultats_fi_g3.csv", encoding="utf-8-sig", sep=";")
g4=AI.fi_grup4()
g4.to_csv("resultats_fi_g4.csv", encoding="utf-8-sig", sep=";")
g5=AI.fi_grup5()
g5.to_csv("resultats_fi_g5.csv", encoding="utf-8-sig", sep=";")

#fase no inicial per grups d'assignatures
ng1=AI.fni_grup1()
ng1.to_csv("resultats_fni_g1.csv", encoding="utf-8-sig", sep=";")
ng2=AI.fni_grup2()
```

```
ng2.to_csv("resultats_fni_g2.csv", encoding="utf-8-sig", sep=";")
ng3=AI.fni_grup3()
ng3.to_csv("resultats_fni_g3.csv", encoding="utf-8-sig", sep=";")
ng4=AI.fni_grup4()
ng4.to_csv("resultats_fni_g4.csv", encoding="utf-8-sig", sep=";")
ng5=AI.fni_grup5()
ng5.to_csv("resultats_fni_g5.csv", encoding="utf-8-sig", sep=";")
ng6=AI.fni_grup6()
ng6.to_csv("resultats_fni_g6.csv", encoding="utf-8-sig", sep=";")
ng7=AI.fni_grup7()
ng7.to_csv("resultats_fni_g7.csv", encoding="utf-8-sig", sep=";")
ng8=AI.fni_grup8()
ng8.to_csv("resultats_fni_g8.csv", encoding="utf-8-sig", sep=";")
ng9=AI.fni_grup9()
ng9.to_csv("resultats_fni_g9.csv", encoding="utf-8-sig", sep=";")
ng10=AI.fni_grup10()
ng10.to_csv("resultats_fni_g10.csv", encoding="utf-8-sig", sep=";")
ng11=AI.fni_grup11()
ng11.to_csv("resultats_fni_g11.csv", encoding="utf-8-sig", sep=";")
ng12=AI.fni_grup12()
ng12.to_csv("resultats_fni_g12.csv", encoding="utf-8-sig", sep=";")
```

Annex C. Codi del modelatge i avaluació.

```
#ELBOW METHOD

# -*- coding: utf-8 -*-
"""
Created on Sat Jan  2 10:39:40 2021

@author: laura
"""
#import

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
        '240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
assfni=['240031','240032','240033','240131','240132','240133','240041','240042',
      '240043','240141','240044','240051','240052','240053','240054','240055',
      '240151','240061','240062','240063','240064','240161','240162','240071',
      '240072','240073','240171','240172']

#LLEGIR DF i TREURE ASSIGNATURES FORA DE assoblig
notes=pd.read_pickle("notes.pkl")
for a in notes.ASSIG.unique():
   if a not in assoblig:
      notes=notes.drop(notes[notes.ASSIG==a].index)
mitj=pd.read_pickle("mitjanes.pkl")
for a in mitj.columns[3:]:
   if a not in assoblig:
      del mitj[a]

#clust_fi
colsfi=["SEX"]
colsfi.extend(assfi)
fi=mitj.loc[:,colsfi]
fi=fi.dropna()
#elbow method per saber nombre de k adequat
ssefi = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
```

```
    model.fit(fi.loc[:,assfi])
    ssefi.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssefi)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_fi")
plt.show()

#clust_fni
colsfni=["SEX"]
colsfni.extend(assfni)
fni=mitj.loc[:,colsfni]
fni=fni.dropna()
#elbow method per saber nombre de k adequat
ssefni = []
for K in range(1, 11):
    model = KMeans(n_clusters=K)
    model.fit(fni.loc[:,assfni])
    ssefni.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssefni)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_fni")
plt.show()

#clust_tot
colstot=["SEX"]
colstot.extend(assoblig)
tot=mitj.loc[:,colstot]
tot=tot.dropna()
#elbow method per saber nombre de k adequat
ssetot = []
for K in range(1, 11):
    model = KMeans(n_clusters=K)
    model.fit(tot.loc[:,assoblig])
    ssetot.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssetot)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_tot")
plt.show()

#clust_fi_h
colsfi=["SEX"]
colsfi.extend(assfi)
```

```
fi=mitj.loc[:,colsfi]
fi=fi.dropna()
fih=fi[fi.SEX=="H"]
#elbow method per saber nombre de k adequat
ssefi = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
   model.fit(fih.loc[:,assfi])
   ssefi.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssefi)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_fi_H")
plt.show()

#clust_tot_h
colstot=["SEX"]
colstot.extend(assoblig)
tot=mitj.loc[:,colstot]
tot=tot.dropna()
toth=tot[tot.SEX=="H"]
#elbow method per saber nombre de k adequat
ssetot = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
   model.fit(toth.loc[:,assoblig])
   ssetot.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssetot)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_tot_H")
plt.show()

#clust_fi_d
colsfi=["SEX"]
colsfi.extend(assfi)
fi=mitj.loc[:,colsfi]
fi=fi.dropna()
fid=fi[fi.SEX=="D"]
#elbow method per saber nombre de k adequat
ssefi = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
   model.fit(fid.loc[:,assfi])
   ssefi.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
```

```
plt.plot(x,ssefi)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_fi_D")
plt.show()


#clust_tot_d
colstot=["SEX"]
colstot.extend(assoblig)
tot=mitj.loc[:,colstot]
tot=tot.dropna()
totd=tot[tot.SEX=="D"]
#elbow method per saber nombre de k adequat
ssetot = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
   model.fit(totd.loc[:,assoblig])
   ssetot.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssetot)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("clust_tot_D")
plt.show()

#CLUSTERING

# -*- coding: utf-8 -*-
"""
Created on Tue Dec 15 09:02:14 2020

@author: laura
"""
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
        '240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
assfni=['240031','240032','240033','240131','240132','240133','240041','240042',
      '240043','240141','240044','240051','240052','240053','240054','240055',
```

```
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']

#LLEGIR DF i TREURE ASSIGNATURES FORA DE assoblig
notes=pd.read_pickle("notes.pkl")
for a in notes.ASSIG.unique():
    if a not in assoblig:
        notes=notes.drop(notes[notes.ASSIG==a].index)
mitj=pd.read_pickle("mitjanes.pkl")
for a in mitj.columns[3:]:
    if a not in assoblig:
        del mitj[a]


def clust_fi(k):
    cols=["SEX"]
    cols.extend(assfi)
    fi=mitj.loc[:,cols]
    fi=fi.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(fi.loc[:,assfi])
    label=km.labels_
    rfi=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=fi[label==i]
        rfi[i]=l.groupby("SEX")["SEX"].count()
    rfi=rfi.T
    rfi.eval("Total=D+H", inplace=True)
    rfi.eval("PercD=D*100/Total", inplace=True)
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    rfi["MitjCentres"]=mitjcent
    rfi=rfi.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom", "HTop", "HBot", "DTop", "DBot"])
    for i in u_labels:
        g=fi[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        gd=g[g.SEX=="D"]
        gh=g[g.SEX=="H"]
        mgd=gd.mean()
        mgh=gh.mean()
        topd=mgd[mgd==mgd.max()].index
        botd=mgd[mgd==mgd.min()].index
```

```
        toph=mgh[mgh==mgh.max()].index
        both=mgh[mgh==mgh.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0], toph[0],both[0],topd[0],botd[0]]
    return rfi, a

def clust_fni(k):
    cols=["SEX"]
    cols.extend(assfni)
    fni=mitj.loc[:,cols]
    fni=fni.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(fni.loc[:,assfni])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=fni[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    r.eval("Total=D+H", inplace=True)
    r.eval("PercD=D*100/Total", inplace=True)
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom", "HTop", "HBot", "DTop", "DBot"])
    for i in u_labels:
        g=fni[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        gd=g[g.SEX=="D"]
        gh=g[g.SEX=="H"]
        mgd=gd.mean()
        mgh=gh.mean()
        topd=mgd[mgd==mgd.max()].index
        botd=mgd[mgd==mgd.min()].index
        toph=mgh[mgh==mgh.max()].index
        both=mgh[mgh==mgh.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0], toph[0],both[0],topd[0],botd[0]]
    return r, a

def clust_tot(k):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
```

```
    tot=tot.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=tot[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    r.eval("Total=D+H", inplace=True)
    r.eval("PercD=D*100/Total", inplace=True)
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom", "HTop", "HBot", "DTop", "DBot"])
    for i in u_labels:
        g=tot[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        gd=g[g.SEX=="D"]
        gh=g[g.SEX=="H"]
        mgd=gd.mean()
        mgh=gh.mean()
        topd=mgd[mgd==mgd.max()].index
        botd=mgd[mgd==mgd.min()].index
        toph=mgh[mgh==mgh.max()].index
        both=mgh[mgh==mgh.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0], toph[0],both[0],topd[0],botd[0]]
    return r, a

def clust_fi_h(k):
    cols=["SEX"]
    cols.extend(assfi)
    fi=mitj.loc[:,cols]
    fi=fi.dropna()
    fi=fi[fi.SEX=="H"]
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(fi.loc[:,assfi])
    label=km.labels_
    rfi=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=fi[label==i]
```

```
        rfi[i]=l.groupby("SEX")["SEX"].count()
    rfi=rfi.T
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    rfi["MitjCentres"]=mitjcent
    rfi=rfi.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom"])
    for i in u_labels:
        g=fi[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0]]
    return rfi, a

def clust_tot_h(k):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    tot=tot[tot.SEX=="H"]
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=tot[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "H_Top", "H_Bottom"])
    for i in u_labels:
        g=tot[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0]]
    return r, a
```

```
def clust_fi_d(k):
    cols=["SEX"]
    cols.extend(assfi)
    fi=mitj.loc[:,cols]
    fi=fi.dropna()
    fi=fi[fi.SEX=="D"]
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(fi.loc[:,assfi])
    label=km.labels_
    rfi=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=fi[label==i]
        rfi[i]=l.groupby("SEX")["SEX"].count()
    rfi=rfi.T
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    rfi["MitjCentres"]=mitjcent
    rfi=rfi.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom"])
    for i in u_labels:
        g=fi[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0]]
    return rfi, a

def clust_tot_d(k):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    tot=tot[tot.SEX=="D"]
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=tot[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
```

```
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "D_Top", "D_Bottom"])
    for i in u_labels:
        g=tot[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0]]
    return r, a

#CLUSTERING DISTANCIES

# -*- coding: utf-8 -*-
"""
Created on Wed Dec 16 19:27:47 2020

@author: laura
"""

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
from statistics import mode

#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
          '240031','240032','240033','240131','240132','240133','240041','240042',
          '240043','240141','240044','240051','240052','240053','240054','240055',
          '240151','240061','240062','240063','240064','240161','240162','240071',
          '240072','240073','240171','240172']
assfni=['240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
#LLEGIR DF
notes=pd.read_pickle("notes.pkl")
for a in notes.ASSIG.unique():
    if a not in assoblig:
        notes=notes.drop(notes[notes.ASSIG==a].index)
mitj=pd.read_pickle("mitjanes.pkl")
for a in mitj.columns[3:]:
    if a not in assoblig:
        del mitj[a]
```

```
def clust_tot(k):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=tot[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    r.eval("Total=D+H", inplace=True)
    r.eval("PercD=D*100/Total", inplace=True)
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    a=pd.DataFrame(columns=["Grup", "Top", "Bottom", "HTop", "HBot", "DTop", "DBot"])
    for i in u_labels:
        g=tot[label==i]
        m=g.mean()
        top=m[m==m.max()].index
        bot=m[m==m.min()].index
        gd=g[g.SEX=="D"]
        gh=g[g.SEX=="H"]
        mgd=gd.mean()
        mgh=gh.mean()
        topd=mgd[mgd==mgd.max()].index
        botd=mgd[mgd==mgd.min()].index
        toph=mgh[mgh==mgh.max()].index
        both=mgh[mgh==mgh.min()].index
        a.loc[len(a.index)]=[i, top[0], bot[0], toph[0],both[0],topd[0],botd[0]]
    return r, a

def ass_max_dist():
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
```

```
    u_labels=np.unique(label)
    a_max_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mxd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            maxdist=dist[pers][dist[pers]==dist[pers].max()].index
            mxd.append(maxdist[0])
        a_max_dist.append((i,mode(maxdist)))
    return a_max_dist

def ass_min_dist():
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels=np.unique(label)
    a_min_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mnd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            mindist=dist[pers][dist[pers]==dist[pers].min()].index
            mnd.append(mindist[0])
        a_min_dist.append((i,mode(mindist)))
    return a_min_dist

def ass_max_dist_H():
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot[tot.SEX=="H"]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
```

```
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels=np.unique(label)
    a_max_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mxd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            maxdist=dist[pers][dist[pers]==dist[pers].max()].index
            mxd.append(maxdist[0])
        a_max_dist.append((i,mode(maxdist)))
    return a_max_dist

def ass_min_dist_H():
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot[tot.SEX=="H"]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels=np.unique(label)
    a_min_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mnd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            mindist=dist[pers][dist[pers]==dist[pers].min()].index
            mnd.append(mindist[0])
        a_min_dist.append((i,mode(mindist)))
    return a_min_dist

def ass_max_dist_D():
    cols=["SEX"]
    cols.extend(assoblig)
```

```
    tot=mitj.loc[:,cols]
    tot=tot[tot.SEX=="D"]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels=np.unique(label)
    a_max_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mxd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            maxdist=dist[pers][dist[pers]==dist[pers].max()].index
            mxd.append(maxdist[0])
        a_max_dist.append((i,mode(maxdist)))
    return a_max_dist

def ass_min_dist_D():
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot[tot.SEX=="D"]
    tot=tot.dropna()
    model=KMeans(n_clusters=3, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels=np.unique(label)
    a_min_dist=[]
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        dist=[]
        mnd=[]
        for pers in range (len(lab)):
            d=lab.iloc[pers,1:]-c
            dist.append(d)
        for pers in range (len(dist)):
            mindist=dist[pers][dist[pers]==dist[pers].min()].index
            mnd.append(mindist[0])
        a_min_dist.append((i,mode(mindist)))
    return a_min_dist
```

```
#CLUSTERING ASSIGS DISCRIMINANTS

# -*- coding: utf-8 -*-
"""
Created on Thu Nov 12 18:33:49 2020

@author: laura
"""

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt
import heapq

#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
        '240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
assfni=['240031','240032','240033','240131','240132','240133','240041','240042',
       '240043','240141','240044','240051','240052','240053','240054','240055',
       '240151','240061','240062','240063','240064','240161','240162','240071',
       '240072','240073','240171','240172']

#LLEGIR DF i TREURE ASSIGNATURES FORA DE assoblig
notes=pd.read_pickle("notes.pkl")
for a in notes.ASSIG.unique():
    if a not in assoblig:
        notes=notes.drop(notes[notes.ASSIG==a].index)
mitj=pd.read_pickle("mitjanes.pkl")
for a in mitj.columns[3:]:
    if a not in assoblig:
        del mitj[a]




def dist_assig(k,i):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
```

```
    tot=tot.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    dist_assig=pd.DataFrame(columns=cols)
    lab=tot[label==i]
    c=centroid[i]
    for p, pers in lab.iterrows():
        d=0
        dist_assig.loc[p,"SEX"]=pers.SEX
        for a in range(1,len(pers)):
            d=abs(pers.iloc[a]-c[a-1])
            ass=dist_assig.columns[a]
            dist_assig.loc[p,ass]=d
    md=dist_assig.mean()
    top5=heapq.nlargest(5, dist_assig.mean())
    bot5=heapq.nsmallest(5, dist_assig.mean())
    asstop5=[]
    assbot5=[]
    for e in top5:
        for a in range(len(md.index)):
            if md.iloc[a]==e:
                ass=md[md==md.iloc[a]].index
                asstop5.append(ass[0])
    for e in bot5:
        for a in range(len(md.index)):
            if md.iloc[a]==e:
                ass=md[md==md.iloc[a]].index
                assbot5.append(ass[0])
    return asstop5, assbot5

def labels_tot(k):
    cols=["SEX"]
    cols.extend(assoblig)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,assoblig])
    centroid=km.cluster_centers_
    label=km.labels_
    u_labels = np.unique(label)
    mitj_dist=pd.DataFrame(columns=["Grup", "DistTot", "DistH", "DistD"])
    for i in u_labels:
        lab=tot[label==i]
        c=centroid[i]
        ld=[]
        for p, pers in lab.iterrows():
```

```
            d=0
            for a in range(1,len(pers)):
                d=d+(pers.iloc[a]-c[a-1])**2
            dist=sqrt(d)
            ld.append(dist)
        lab["Dist"]=ld #afegeix col amb la distancia de cada punt al centre
        dones=lab[lab.SEX=="D"]
        md=dones.Dist.mean()
        homes=lab[lab.SEX=="H"]
        mh=homes.Dist.mean()
        dades_label=[i, lab.Dist.mean(), mh, md]
        mitj_dist.loc[len(mitj_dist.index)]=dades_label #mitjana de dist al centre de cada cluster
    return mitj_dist



#CLUSTERING PER GRUPS

# -*- coding: utf-8 -*-
"""
Created on Wed Dec 16 19:27:47 2020

@author: laura
"""

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

#LLISTES AUXILIARS
assfi=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025]
assoblig=[240011,240012,240013,240014,240015,240021,240022,240023,240024,240025,
          '240031','240032','240033','240131','240132','240133','240041','240042',
          '240043','240141','240044','240051','240052','240053','240054','240055',
          '240151','240061','240062','240063','240064','240161','240162','240071',
          '240072','240073','240171','240172']
assfni=['240031','240032','240033','240131','240132','240133','240041','240042',
        '240043','240141','240044','240051','240052','240053','240054','240055',
        '240151','240061','240062','240063','240064','240161','240162','240071',
        '240072','240073','240171','240172']
#LLEGIR DF
notes=pd.read_pickle("notes.pkl")
for a in notes.ASSIG.unique():
    if a not in assoblig:
        notes=notes.drop(notes[notes.ASSIG==a].index)
mitj=pd.read_pickle("mitjanes.pkl")
for a in mitj.columns[3:]:
```

```
    if a not in assoblig:
        del mitj[a]


cols=["SEX"]
grup1=[240011,240012,240021,240022,240013,240023,240014,240024]
cols.extend(grup1)
tot=mitj.loc[:,cols]
tot=tot.dropna()
#elbow method per saber nombre de k adequat
ssetot = []
for K in range(1, 11):
    model = KMeans(n_clusters=K)
    model.fit(tot.loc[:,grup1])
    ssetot.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssetot)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("Fase Inicial. Grups 1,2 i 5")
plt.show()

def clust_fi_g1(k):
    cols=["SEX"]
    cols.extend(grup1)
    tot=mitj.loc[:,cols]
    tot=tot.dropna()
    model=KMeans(n_clusters=k, random_state=0)
    km=model.fit(tot.loc[:,grup1])
    label=km.labels_
    r=pd.DataFrame()
    u_labels = np.unique(label)
    for i in u_labels:
        l=tot[label==i]
        r[i]=l.groupby("SEX")["SEX"].count()
    r=r.T
    r.eval("Total=D+H", inplace=True)
    r.eval("PercD=D*100/Total", inplace=True)
    centroids=km.cluster_centers_
    mitjcent=[]
    for c in centroids:
        mitjcent.append(c.mean())
    r["MitjCentres"]=mitjcent
    r=r.round(decimals=2)
    return r

cols=["SEX"]
grup2=[240015,240025]
```

```
cols.extend(grup2)
tot=mitj.loc[:,cols]
tot=tot.dropna()
#elbow method per saber nombre de k adequat
ssetot = []
for K in range(1, 11):
   model = KMeans(n_clusters=K)
   model.fit(tot.loc[:,grup2])
   ssetot.append(model.inertia_)
x=[1,2,3,4,5,6,7,8,9,10]
plt.plot(x,ssetot)
plt.xlabel("k")
plt.ylabel("inèrcia")
plt.title("Fase Inicial. Grups 3 i 4")
plt.show()

def clust_fi_g2(k):
   cols=["SEX"]
   cols.extend(grup2)
   tot=mitj.loc[:,cols]
   tot=tot.dropna()
   model=KMeans(n_clusters=k, random_state=0)
   km=model.fit(tot.loc[:,grup2])
   label=km.labels_
   r=pd.DataFrame()
   u_labels = np.unique(label)
   for i in u_labels:
      l=tot[label==i]
      r[i]=l.groupby("SEX")["SEX"].count()
   r=r.T
   r.eval("Total=D+H", inplace=True)
   r.eval("PercD=D*100/Total", inplace=True)
   centroids=km.cluster_centers_
   mitjcent=[]
   for c in centroids:
      mitjcent.append(c.mean())
   r["MitjCentres"]=mitjcent
   r=r.round(decimals=2)
   return r

#CLUSTER amb les assignatures InfoFonamental i Expre
calc=mitj.loc[:,["SEX",240015,240025]]
calc=calc.dropna()

#trobem que la k adequada es k=3
model=KMeans(n_clusters=3, random_state=0)
kmcalc=model.fit(calc.loc[:,[240015,240025]])
label=kmcalc.labels_
```

```
centroids=model.cluster_centers_
fig, ax=plt.subplots()
color0={"H":"#f5df4d", "D":"#ca8d22"}
color1={"H":"#b68aed", "D":"#7a48b0"}
color2={"H":"#b2d51f", "D":"#064416"}
ax.scatter(calc[label==0].loc[:,240015],calc[label==0].loc[:,240025], label=0,
c=calc[label==0].SEX.map(color0))
plt.scatter(calc[label==1].loc[:,240015],calc[label==1].loc[:,240025], label=1,
c=calc[label==1].loc[:,"SEX"].map(color1))
plt.scatter(calc[label==2].loc[:,240015],calc[label==2].loc[:,240025], label=2,
c=calc[label==2].loc[:,"SEX"].map(color2))
plt.scatter(centroids[:,0] , centroids[:,1] , s = 80, color = 'red')
plt.legend()
plt.xlabel("Nota Fonaments d'Informàtica")
plt.ylabel("Nota Expressió Gràfica")
plt.show()
```