# ENIF24_credit

August 3, 2025

# 1 ENIF 2024: Usage of formal credit

## 1.1 Table of contents



## 1.2 Introduction

In this work we study the results from the National Survet on Financial Inclusion (ENIF), conducted by the National Banking Comission (CNBV) and the Statistics and Geography National Institute (INEGI), in 2024, which tries to "diagnose, design public policies, and establish goald regarding inclusion and financial education; similarly, it tries to suggest changes and updates to attend to new requirements and considerations on the National Policy of Financial Inclusion".

For this analysis, we focus on Section 4, Subsection 6, of said document, on financial attitudes, behaviour, vulnerability, and over-all well-being and, more especifically, on credit usage.

[1]:
```
### Basic packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

### ML models

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

### ML tools

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
 ↪root_mean_squared_error
```

## 1.3 Data loading

The tables are available at:

[ ]:
```
URL = 'https://www.inegi.org.mx/contenidos/programas/enif/2024/microdatos/
 ↪enif_2024_bd_csv.zip'
```

The relevant data can be found on the following document.

[2]:
```
data = pd.read_csv('./TMODULO.csv')
data.head(5)
```

[2]:
```
   LLAVEMOD  LLAVEVIV  LLAVEHOG  EDAD_V  NIV  GRA  P3_1A  P3_2  P3_3  P3_4  \
0    101101       101      1011      38    8    4      2     5     2     2
1    210101       210      2101      36    8    5      2     2     2     2
2    303102       303      3031      20    3    3      2     6     2     2
3    401101       401      4011      59    3    3      2     4     2     2
4    503101       503      5031      37    6    3      2     5     2     2

   …  FOLIO  VIV_SEL  HOGAR  N_REN  SEXO  TLOC  REGION  EST_DIS  UPM_DIS  \
0  …      1        1      1      1     1     1       1       17      196
1  …      2       10      1      1     1     3       6      179     2088
2  …      3        3      1      2     1     1       4       50      763
3  …      4        1      1      1     1     1       1       17      182
4  …      5        3      1      1     2     1       4       49      631

   FAC_PER
0     1233
1     1763
2      903
3      720
4     8114

[5 rows x 398 columns]
```

[3]:
```
### Subsection
```

```
numss = 6

### The relevant data for Subsectiob 4.6 are extracted

data1 = data[['LLAVEMOD', 'SEXO', 'EDAD_V', 'NIV', 'GRA', 'REGION', 'TLOC',␣
↪f'P4_{numss}_1',f'P4_{numss}_2',f'P4_{numss}_3',f'P4_{numss}_4',f'P4_{numss}_5',f'P4_{numss
data1.head()
```

[3]:
```
   LLAVEMOD  SEXO  EDAD_V  NIV  GRA  REGION  TLOC  P4_6_1  P4_6_2  P4_6_3  \
0    101101     1      38    8    4       1     1       1       1       2
1    210101     1      36    8    5       6     3       1       1       2
2    303102     1      20    3    3       4     1       2       2       2
3    401101     1      59    3    3       1     1       1       1       1
4    503101     2      37    6    3       4     1       1       1       3

   P4_6_4  P4_6_5  P4_6_6
0       2       3       2
1       1       1       1
2       2       3       3
3       3       1       2
4       1       3       3
```

We can see that most of the data are discreet, esentially categorical. We consider sex, age, education, ans region/locality as independent variables, and the answers to all of the questions as the dependent variables. Later, we will define a continuous metric as an aggregate of some of these variables, as to facilitate analysis and interpretation.

Back to the introduction

## 1.4 Cleaning and integrity

We check for duplicates and NAs, and we verify that all of the inputs are within the ranges specified by INEGI.

[4]:
```
### data types

data1.dtypes
```

[4]:
```
LLAVEMOD    int64
SEXO        int64
EDAD_V      int64
NIV         int64
GRA         int64
REGION      int64
TLOC        int64
P4_6_1      int64
P4_6_2      int64
P4_6_3      int64
```

```
P4_6_4      int64
P4_6_5      int64
P4_6_6      int64
dtype: object
```

We are dealing with numeric data only.

Now we look for duplicates and NAs.

[5]:
```
print(f"Raw data: {data1.shape[1]} columns and {data1.shape[0]} rows")
print(f"Withous NAs: {data1.dropna().shape[1]} columns and {data1.dropna().
 ↪shape[0]} rows")
print(f"Without duplicates: {data1.drop_duplicates().shape[1]} columns and␣
 ↪{data1.drop_duplicates().shape[0]} rows")
```

```
Raw data: 13 columns and 13502 rows
Withous NAs: 13 columns and 13502 rows
Without duplicates: 13 columns and 13502 rows
```

We conclude that the data was pretty clean to begin with. Now we check the ranges for each column.

Sex: Male (1), Female (2).

[6]:
```
pd.DataFrame(data1['SEXO'].value_counts()).sort_values('SEXO').head()
```

[6]:

| SEXO | count |
|---|---|
| 1 | 6082 |
| 2 | 7420 |

Region: Northwest (1), Northeast (2), Bajío (3), CDMX (4), South/East (5), South (6).

[7]:
```
pd.DataFrame(data1['REGION'].value_counts()).sort_values('REGION').head(6)
```

[7]:

| REGION | count |
|---|---|
| 1 | 2431 |
| 2 | 2499 |
| 3 | 2581 |
| 4 | 952 |
| 5 | 2528 |
| 6 | 2511 |

Education: ranging from 0 (No studies) to 11 (PhD), and 99 for anyone who answered “Doesn’t know”.

[8]:
```
pd.DataFrame(data1['NIV'].value_counts()).sort_values('NIV').head(13)
```

[8]:

```
      count
NIV
```

```
0       532
1        20
2      2697
3      3635
4        15
5       261
6      2808
7       324
8      2853
9        52
10      256
11       44
99        5
```

Each question from this Subsection can be aswered with a number: "Always" (1), "Sometimes" (2), "Never" (3), "Doesn't answer" (8) o "Doesn't know" (9).

[9]:
```
### Number of question: 1 to 6

numq = 1

pd.DataFrame(data1[f'P4_{numss}_{numq}'].value_counts()).
↪sort_values(f'P4_{numss}_{numq}').head(5)
```

[9]:
```
        count
P4_6_1
1        9698
2        2935
3         788
8          28
9          53
```

Back to the Intro

### 1.5 Data normalization and KPIs

Now we define auxiliary columns as functions of the original entries, either to aggregate data, to help with visualizations, of to define useful emtrics.

[11]:
```
### Sex, alphanumeric

def set_sexo(a):
    if a == 1:
        return 'M'
    elif a == 2:
        return 'F'
    else:
        return "N/A"
```

```python
### Region alphanumeric

def set_region(a):
    if a == 1:
        return 'Northwest'
    elif a == 2:
        return 'Northeast'
    elif a == 3:
        return 'Bajío'
    elif a == 4:
        return 'CDMX'
    elif a == 5:
        return 'South/East'
    elif a == 6:
        return 'South'

### Normalized answers in {1,...,5}

def set_norm(a):
    if int(a) == 8:
        return 4
    elif int(a) == 9:
        return 5
    else:
        return int(a)

### Answers, alphanumeric

def set_norm_t(a):
    if int(a) == 1:
        return "Always"
    elif int(a) == 2:
        return "Sometimes"
    elif int(a) == 3:
        return "Never"
    elif int(a) == 4:
        return "Doesn't answer"
    elif int(a) == 5:
        return "Doesn't know"

### Grading: 'Agrees' corresponds to +1, 'Never' corresponds to -1, and 'Doesn't agree nor disagree' corresponds to 0

def set_norm_calif(a):

    return 2-a
```

```python
### Total grade: -6 pts corresponds to 5, and 6 pts corresponds to 10

def set_tot_calif(a):

    return 5/12*(a-6)+10

### Education, alphanumeric, aggregated (for visualizations)

def set_niv(a):
    if int(a) == 0:
        return "No studies"
    elif int(a) < 6:
        return "Basic"
    elif int(a) < 8:
        return "high School"
    elif int(a) == 8:
        return "Undergraduate"
    elif int(a) < 99:
        return "Graduate"
    else:
        return "Doesn't know"
```

New columns are added.

[12]:
```python
### Alphanumeric sex

data1.loc[:,'SEXO_HM'] = data1['SEXO'].map(set_sexo)
```

```
C:\Users\patju\AppData\Local\Temp\ipykernel_26000\3748415747.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1.loc[:,'SEXO_HM'] = data1['SEXO'].map(set_sexo)
```

[13]:
```python
### Alphanumeric region

data1.loc[:,'REGION_T'] = data1['REGION'].map(set_region)
```

```
C:\Users\patju\AppData\Local\Temp\ipykernel_26000\821602288.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1.loc[:,'REGION_T'] = data1['REGION'].map(set_region)
```

```
[14]: ### Alphanumeric education

data1.loc[:,'NIVED_T'] = data1['NIV'].map(set_niv)
```

```
C:\Users\patju\AppData\Local\Temp\ipykernel_26000\3861912701.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1.loc[:,'NIVED_T'] = data1['NIV'].map(set_niv)
```

Normalized and alphanumeric answers.

```
[15]: for numq in range(1,7):
    data1.loc[:,f'p4{numss}{numq}'] = data1[f'P4_{numss}_{numq}'].map(set_norm)
    data1.loc[:,f'p4{numss}{numq}_t'] = data1[f'p4{numss}{numq}'].
  ↪map(set_norm_t)
    data1 = data1.drop(f'P4_{numss}_{numq}',axis=1)
```

```
C:\Users\patju\AppData\Local\Temp\ipykernel_26000\3780884663.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1.loc[:,f'p4{numss}{numq}'] = data1[f'P4_{numss}_{numq}'].map(set_norm)
C:\Users\patju\AppData\Local\Temp\ipykernel_26000\3780884663.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1.loc[:,f'p4{numss}{numq}_t'] = data1[f'p4{numss}{numq}'].map(set_norm_t)
```

We get our new, provisional table. Later, after filtering, we will add a KPI column.

```
[16]: data1 = data1[['LLAVEMOD', 'SEXO', 'SEXO_HM', 'EDAD_V', 'NIV','NIVED_T',␣
  ↪'REGION',␣
  ↪'REGION_T','TLOC',f'p4{numss}1',f'p4{numss}2',f'p4{numss}3',f'p4{numss}4',f'p4{numss}5',f'p
data1.head()
```

```
[16]:    LLAVEMOD  SEXO SEXO_HM  EDAD_V  NIV       NIVED_T  REGION    REGION_T  \
0     101101     1       M      38    8  Undergraduate       1  Northwest
1     210101     1       M      36    8  Undergraduate       6       South
2     303102     1       M      20    3          Basic       4        CDMX
```

```
3   401101     1        M      59     3          Basic        1  Northwest
4   503101     2        F      37     6    high School        4       CDMX

   TLOC  p461  …  p463  p464  p465  p466     p461_t     p462_t     p463_t  \
0     1     1  …     2     2     3     2     Always     Always  Sometimes
1     3     1  …     2     1     1     1     Always     Always  Sometimes
2     1     2  …     2     2     3     3  Sometimes  Sometimes  Sometimes
3     1     1  …     1     3     1     2     Always     Always     Always
4     1     1  …     3     1     3     3     Always     Always      Never

      p464_t  p465_t     p466_t
0  Sometimes   Never  Sometimes
1     Always  Always     Always
2  Sometimes   Never      Never
3      Never  Always  Sometimes
4     Always   Never      Never

[5 rows x 21 columns]
```

Back to the Intro

## 1.6 Distributions

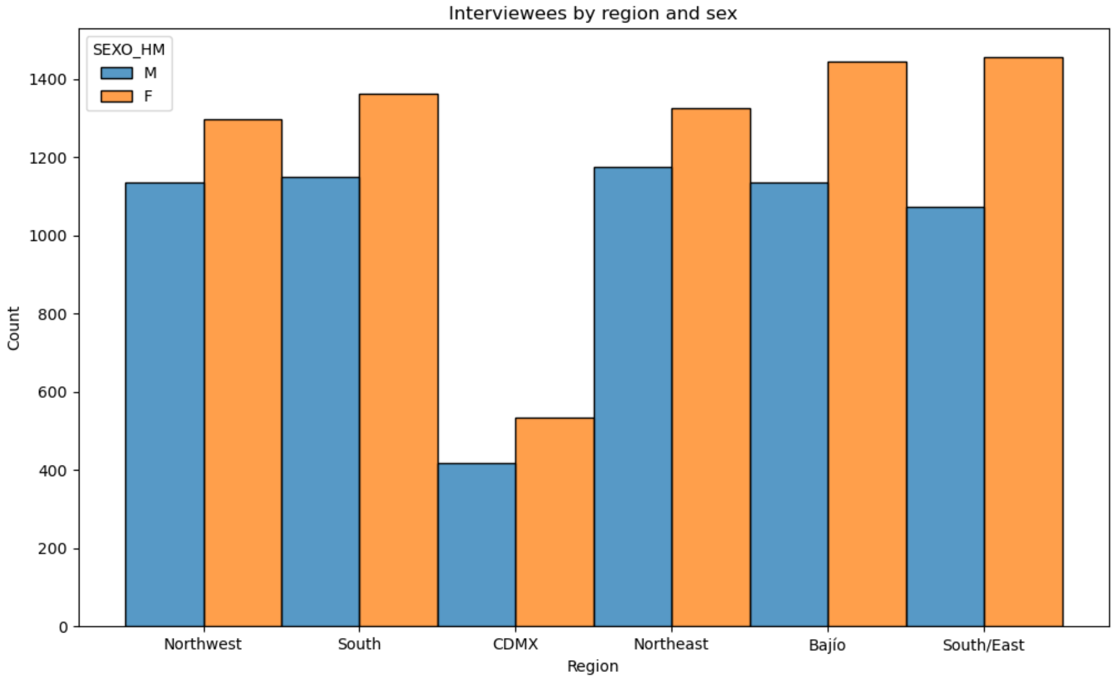
Before beginning with the predictive analysis, we will visualize our data to better understand it.

### 1.6.1 Overall distro

We plot sex, region, and education.

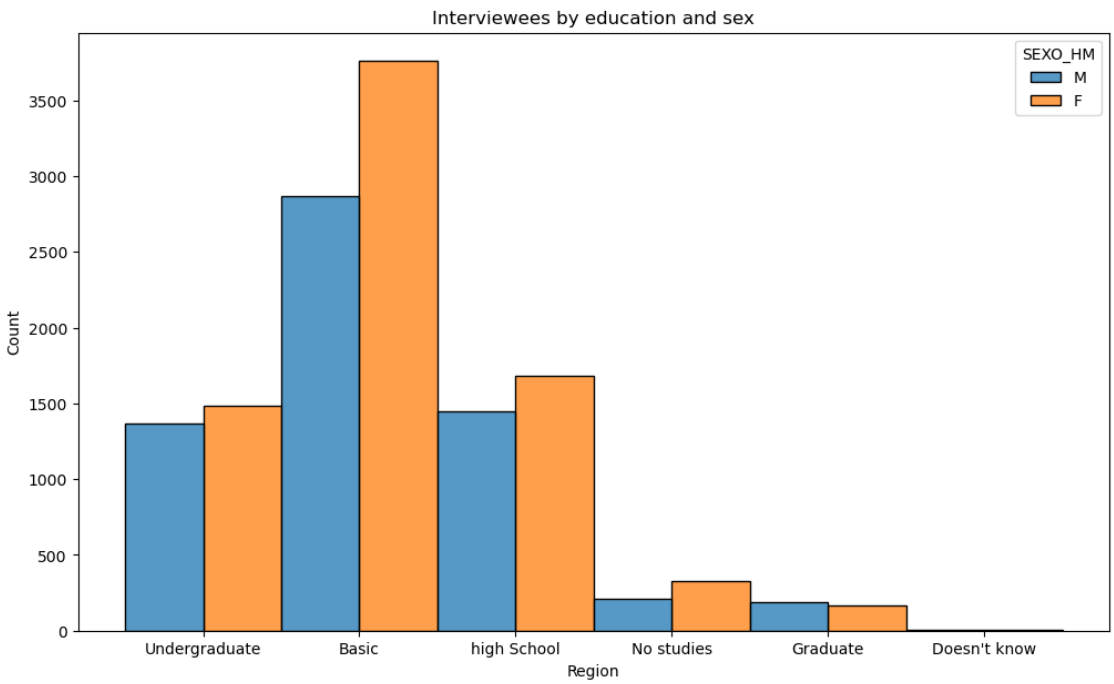
[17]:
```
fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data1, x = 'REGION_T', multiple="dodge", hue = 'SEXO_HM')

ax.set_xlabel('Region')
ax.set_ylabel('Count')
ax.set_title('Interviewees by region and sex')
```

[17]:
```
Text(0.5, 1.0, 'Interviewees by region and sex')
```




[18]:
```
fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data1, x = 'NIVED_T', multiple="dodge", hue = 'SEXO_HM')

ax.set_xlabel('Region')
ax.set_ylabel('Count')
ax.set_title('Interviewees by education and sex')
```

[18]:
```
Text(0.5, 1.0, 'Interviewees by education and sex')
```




### 1.6.2 Answers

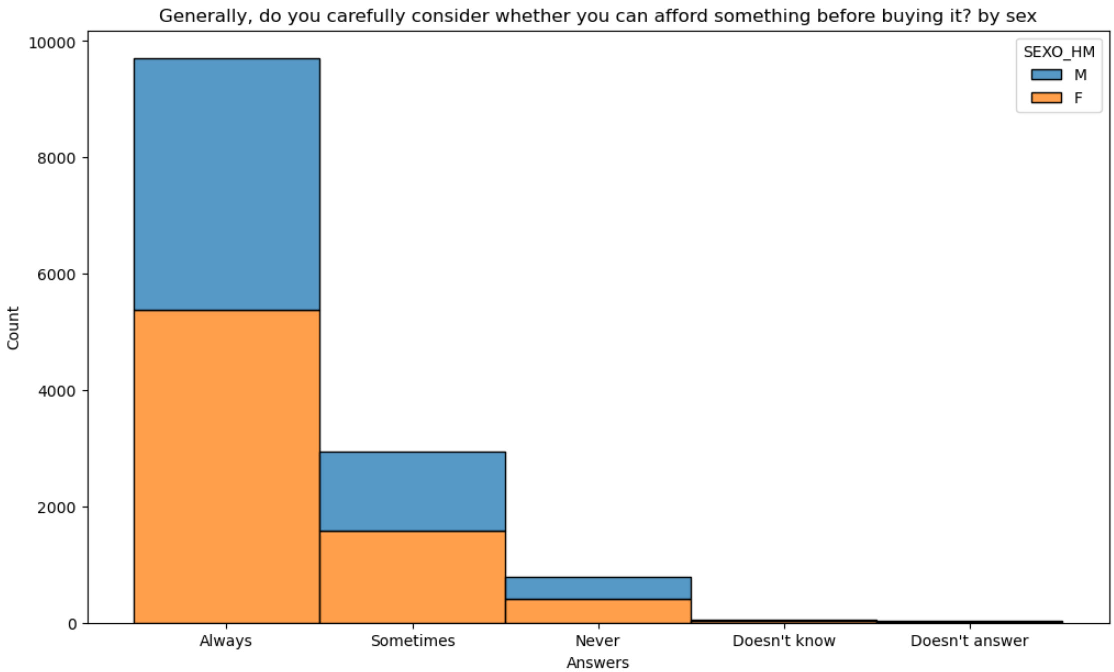
We plot the distributions of the answers with respect to sex, region, and education.

[21]:
```
### Questions dictionnary

questions = {'1' : 'do you carefully consider whether you can afford something␣
↪before buying it?',
             '2' : 'do you pay your bills on time?',
             '3' : 'do you prefer to spending money rather than saving it?',
             '4' : 'do you tend to define long-term economic goals and make and␣
↪effort to complete them?',
             '5' : 'does handling your incomes and expenditures take control␣
↪over your life?',
             '6' : 'do you have money left at the end of the month?'}
```

[22]:
```
### Question

numq = 1

fig, ax = plt.subplots(figsize=(12,7))

### General histogram by sex
```

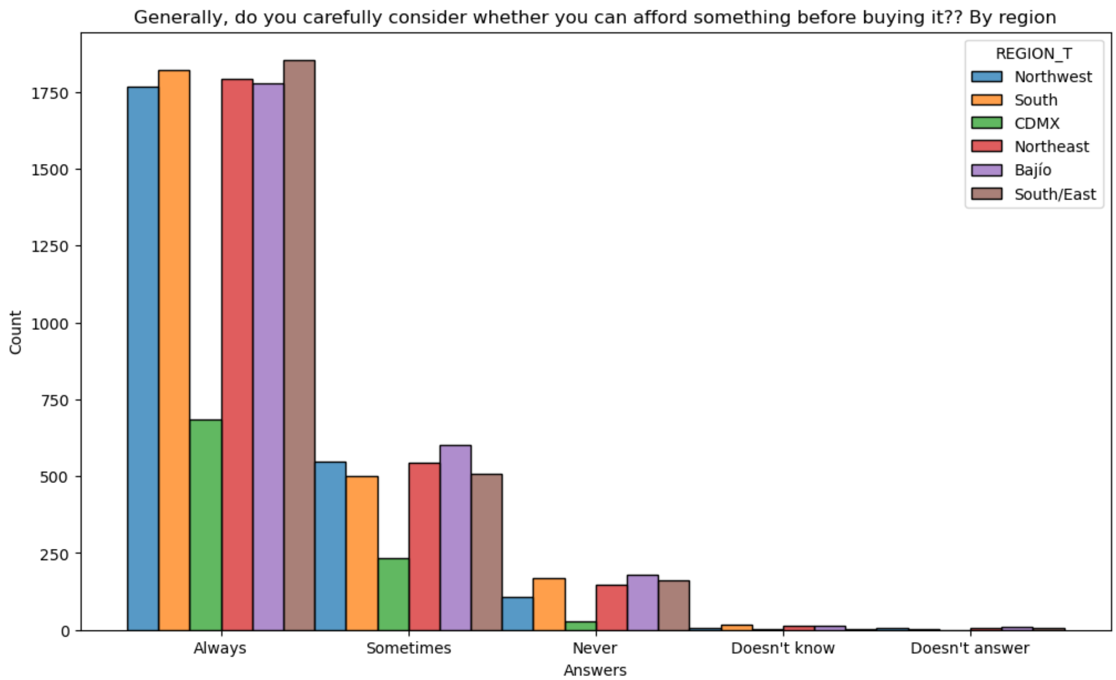
```
plot1 = sns.histplot(data1, x = f'p4{numss}{numq}_t', multiple="stack", hue =␣
↪"SEXO_HM")

ax.set_xlabel('Answers')
ax.set_ylabel('Count')
ax.set_title(f'Generally, {questions[str(numq)]} by sex')
```

[22]: Text(0.5, 1.0, 'Generally, do you carefully consider whether you can afford something before buying it? by sex')



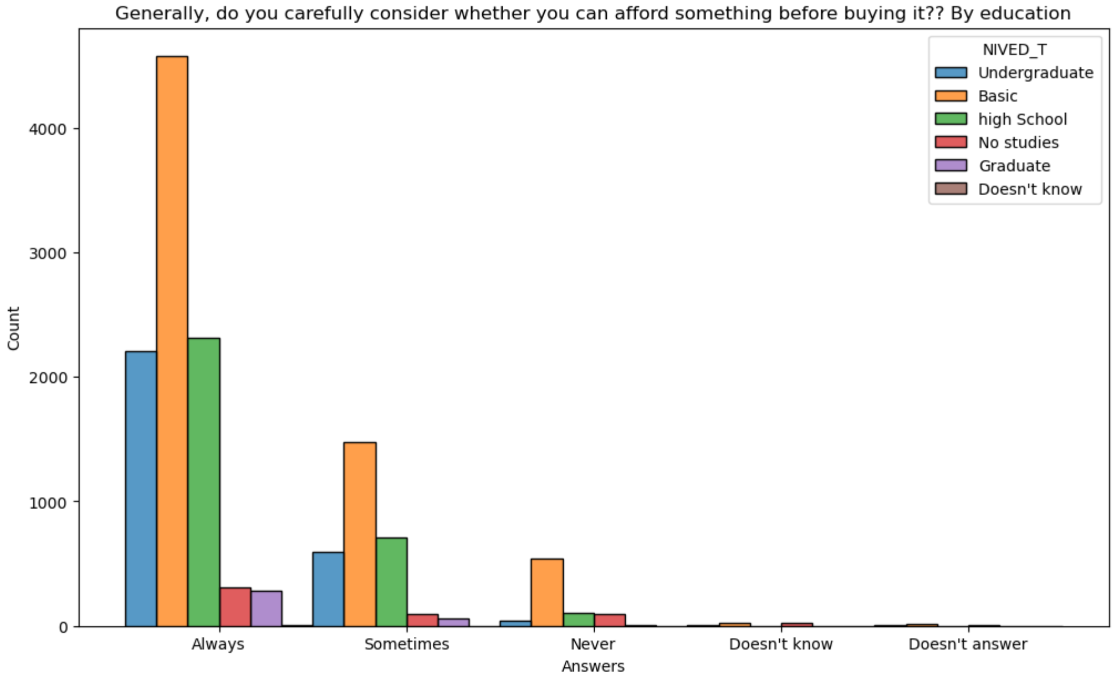

[23]:
```
### Question

numq = 1

fig, ax = plt.subplots(figsize=(12,7))

### General histogram by region

plot1 = sns.histplot(data1, x = f'p4{numss}{numq}_t', multiple="dodge", hue =␣
↪"REGION_T")

ax.set_xlabel('Answers')
ax.set_ylabel('Count')
ax.set_title(f'Generally, {questions[str(numq)]}? By region')
```

[23]: Text(0.5, 1.0, 'Generally, do you carefully consider whether you can afford something before buying it?? By region')




[24]:
```
### Question

numq = 1

fig, ax = plt.subplots(figsize=(12,7))

### General histogram by region

plot1 = sns.histplot(data1, x = f'p4{numss}{numq}_t', multiple="dodge", hue =
    "NIVED_T")

ax.set_xlabel('Answers')
ax.set_ylabel('Count')
ax.set_title(f'Generally, {questions[str(numq)]}? By education')
```

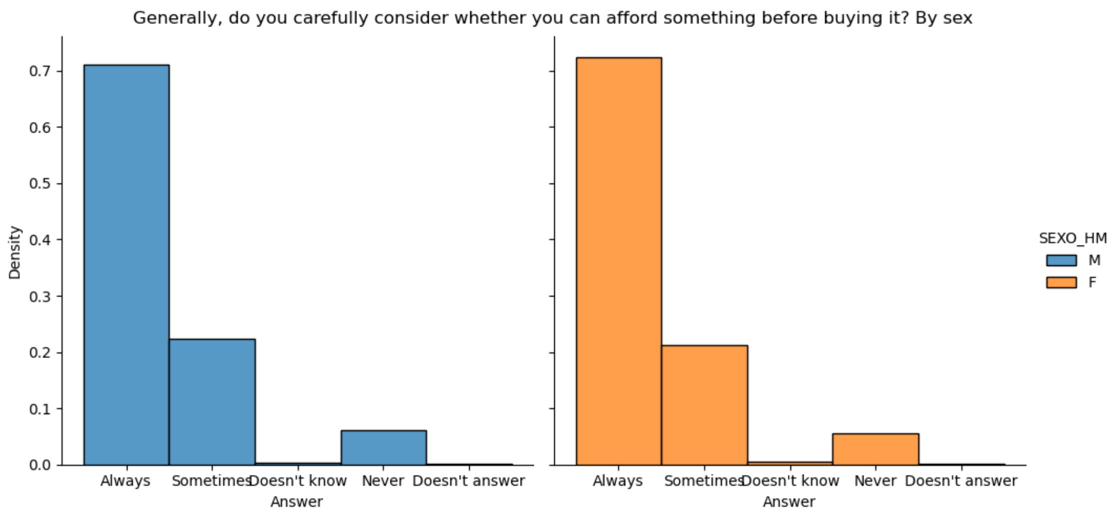
[24]: Text(0.5, 1.0, 'Generally, do you carefully consider whether you can afford something before buying it?? By education')




We see that some classes, such as females (F) for sex, CDMX for region, and Basic for education, have a significatively different number of answers from the rest of their respective category. To check whether all subpopulations behave similarly, we plot independent histograms.

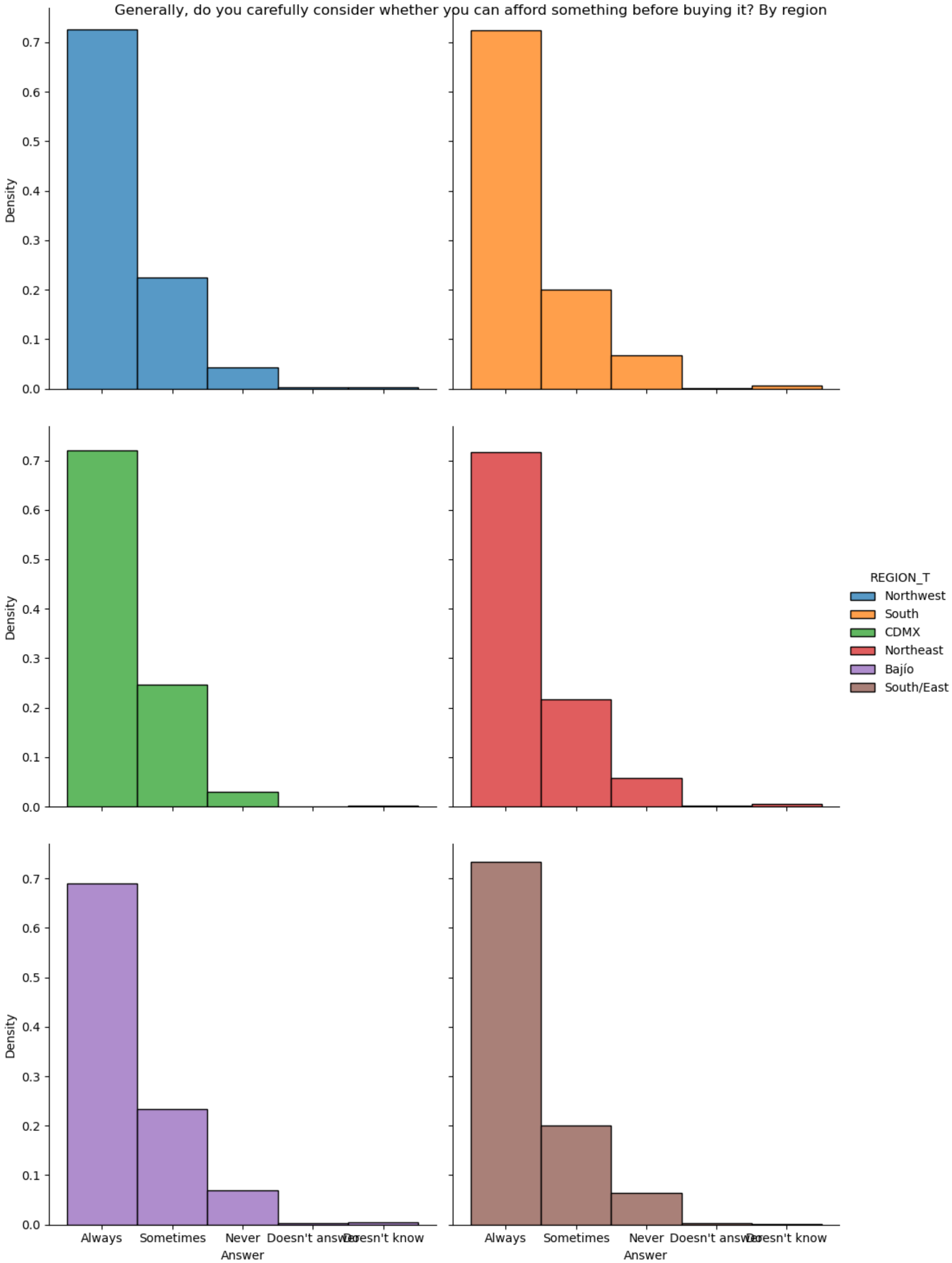
[25]:
```python
### Filtered histograms

### Question

numq = 1

### By sex

g = sns.FacetGrid(data1, col="SEXO_HM", hue="SEXO_HM", height = 5)
g.map(sns.histplot, f'p4{numss}{numq}_t', stat='density', bins=10)
g.set_axis_labels("Answer", "Density")
g.set_titles('')
g.fig.suptitle(f'Generally, {questions[str(numq)]} By sex')
g.add_legend()
```

[25]: `<seaborn.axisgrid.FacetGrid at 0x23b1c053d40>`




[26]:
```python
### Filtered histograms

### Question

numq = 1

### By region

g = sns.FacetGrid(data1, col="REGION_T", hue="REGION_T", height = 5, col_wrap =
    ↪2)
g.map(sns.histplot, f'p4{numss}{numq}_t', stat='density', bins=10)
g.set_axis_labels("Answer", "Density")
g.set_titles('')
g.fig.suptitle(f'Generally, {questions[str(numq)]} By region')
g.add_legend()
```

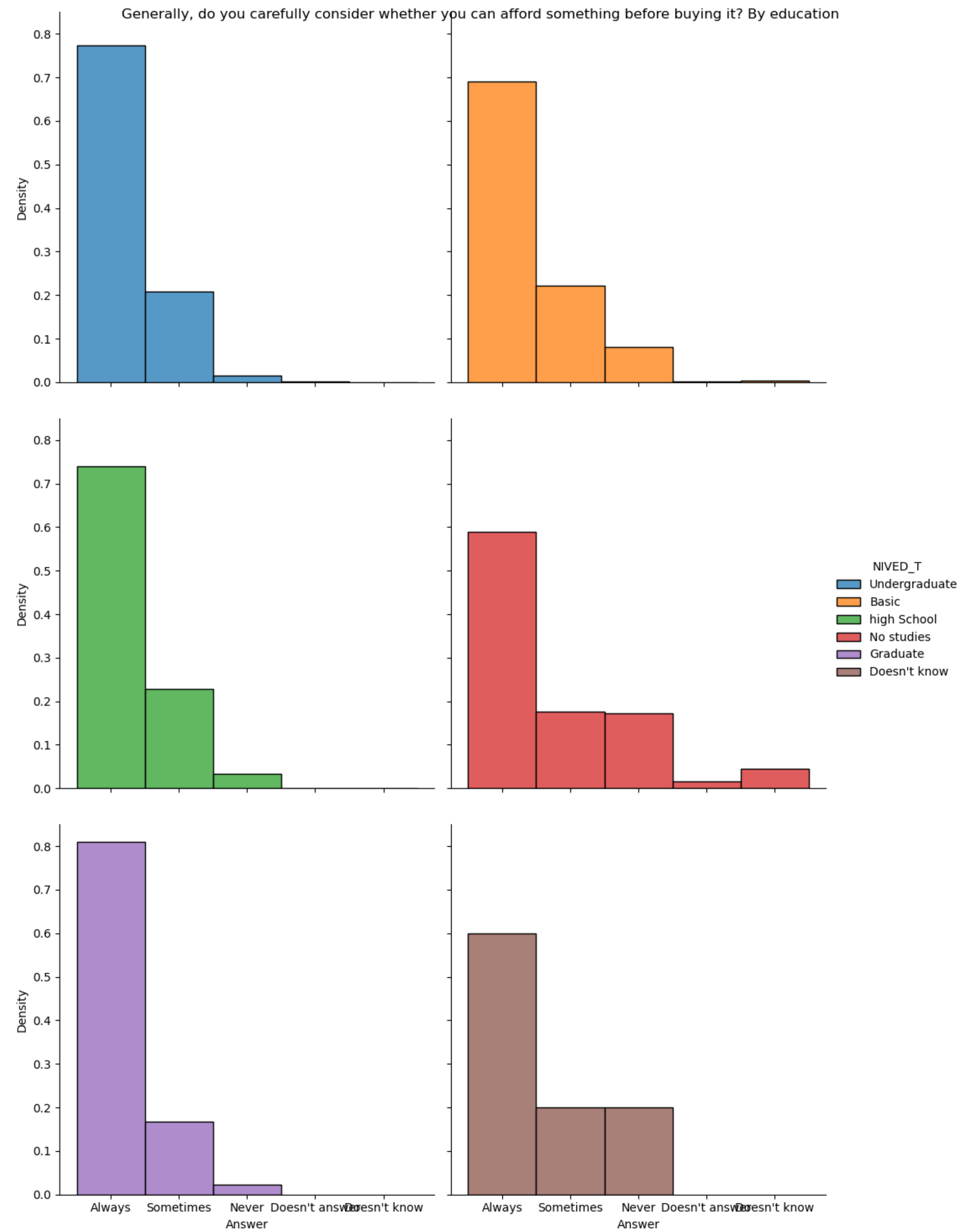
[26]: <seaborn.axisgrid.FacetGrid at 0x23b1c873a70>




[27]:
```
### Filtered histograms

### Question
```

```python
numq = 1

### By education

g = sns.FacetGrid(data1, col="NIVED_T", hue="NIVED_T", height = 5, col_wrap = 2)
g.map(sns.histplot, f'p4{numss}{numq}_t', stat='density', bins=10)
g.set_axis_labels("Answer", "Density")
g.set_titles('')
g.fig.suptitle(f'Generally, {questions[str(numq)]} By education')
g.add_legend()
```

[27]: <seaborn.axisgrid.FacetGrid at 0x23b1db3d5b0>




We see different behaviours amongst subpopulations, more concretely amongst regions and education. We expect this to be addressed by the correlation analysis.

Back to the intro

### 1.7 Data filtering

Given that some answers were given by very few interviewees, we filter the data to avoid skewing the results.

[28]:
```
data2 = data1
```

[29]:
```
### 'Doesnt know' or 'Doesnt answer' responses are filtered out

for numq in range(1,7):

    data2 = data2.loc[data1[f'p4{numss}{numq}']<4]

data2 = data2.loc[data1['NIVED_T']!="Doesn't know"]
```

[30]:
```
### Dimensiones

data2 = data2.dropna()
data2.shape
```

[30]:
```
(13196, 21)
```

[31]:
```
print(f'After filtering "Doesnt answer" and "Doesnt know", we get {round(data2.
↪shape[0]/data1.shape[0]*100,2)}% of the original sample')
print(f'{round((data1.shape[0]-data2.shape[0])/data1.shape[0]*100,2)}% of the␣
↪original sample was filtered out')
```

```
After filtering "Doesnt answer" and "Doesnt know", we get 97.73% of the original
sample
2.27% of the original sample was filtered out
```

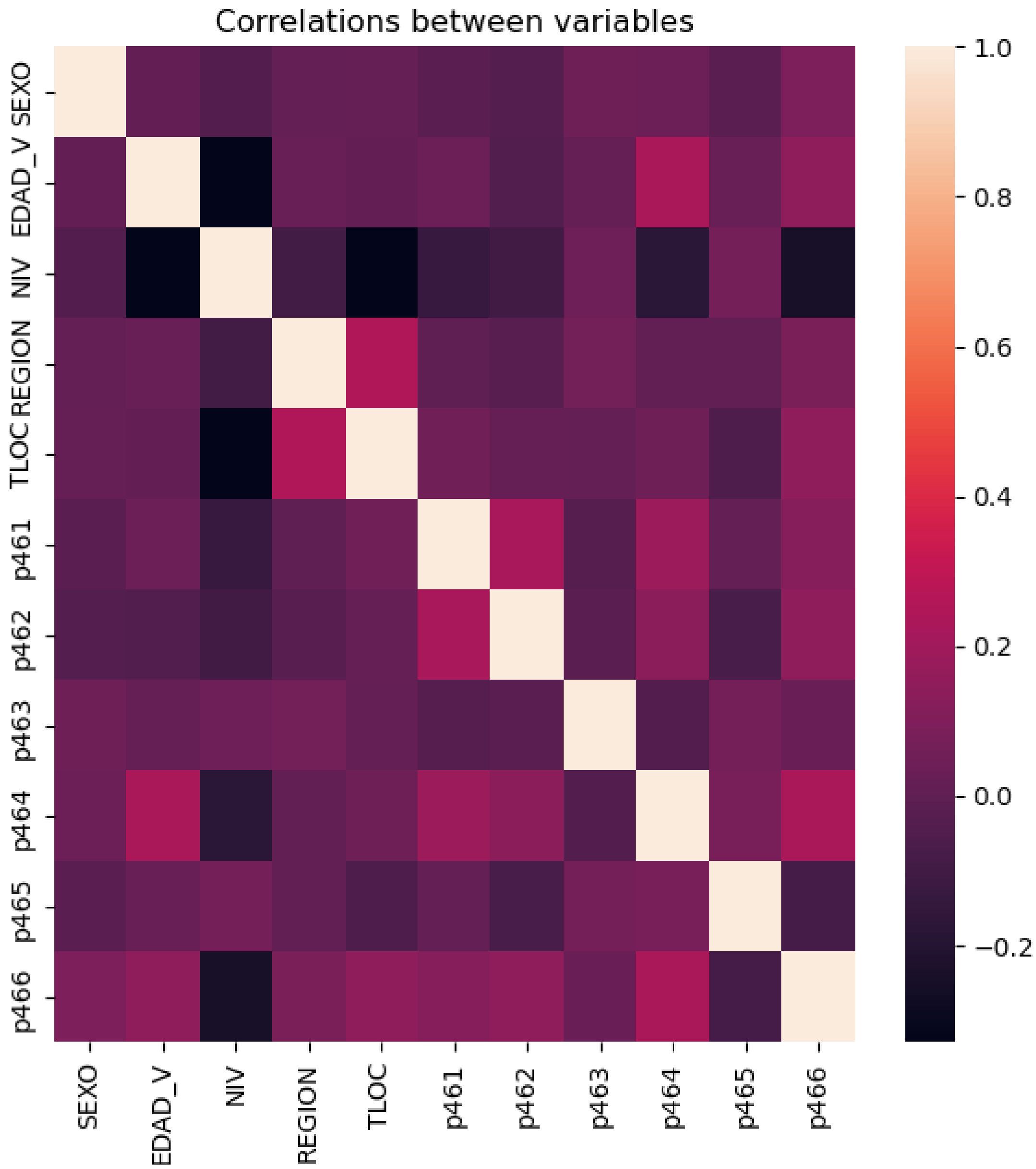
Back to the intro

### 1.8 Correlation analysis

We wish to establish some relationships amongst our data prior to conducting the predictive analysis.

[32]:
```
corr = data2.drop(['LLAVEMOD'],axis=1).select_dtypes('number').corr()

fig, ax = plt.subplots(figsize=(7,7))

# plot the heatmap
sns.heatmap(corr)

ax.set_title('Correlations between variables')
```

[32]:
```
Text(0.5, 1.0, 'Correlations between variables')
```



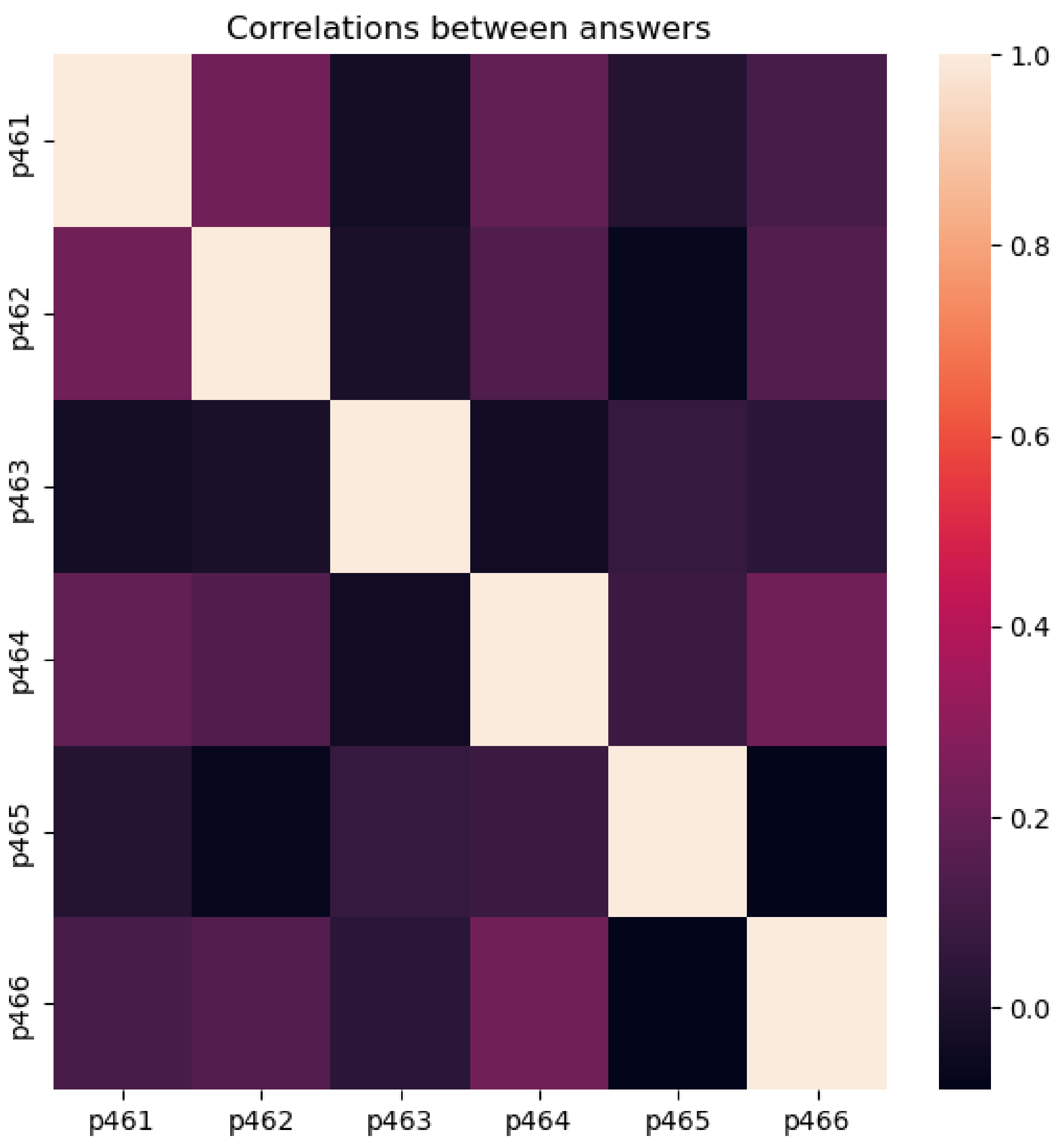

[33]:
```
corr[[f'p4{numss}1',f'p4{numss}2',f'p4{numss}3',f'p4{numss}4',f'p4{numss}5',f'p4{numss}6']].
↪head(5)
```

[33]:

| | p461 | p462 | p463 | p464 | p465 | p466 |
|---|---|---|---|---|---|---|
| SEXO | -0.014282 | -0.035750 | 0.044067 | 0.035876 | -0.015325 | 0.094473 |
| EDAD_V | 0.039528 | -0.045747 | 0.018877 | 0.236124 | 0.028242 | 0.152063 |
| NIV | -0.136090 | -0.099896 | 0.046668 | -0.179151 | 0.067336 | -0.241550 |
| REGION | 0.001882 | -0.017657 | 0.061323 | 0.006531 | 0.003909 | 0.089135 |
| TLOC | 0.052650 | 0.022706 | 0.018633 | 0.042497 | -0.057069 | 0.153184 |

[34]:
```
corr1 = data2.drop(['LLAVEMOD','REGION','SEXO','EDAD_V', 'NIV','TLOC'],axis=1).
 ↪select_dtypes('number').corr()

fig, ax = plt.subplots(figsize=(7,7))

# plot the heatmap
sns.heatmap(corr1)

ax.set_title('Correlations between answers')
```

[34]:
```
Text(0.5, 1.0, 'Correlations between answers')
```



[35]:
```
corr1.head(6)
```

[35]:
```
          p461      p462      p463      p464      p465      p466
p461  1.000000  0.223896 -0.032235  0.187797  0.017397  0.121445
p462  0.223896  1.000000 -0.013912  0.142069 -0.069505  0.149470
p463 -0.032235 -0.013912  1.000000 -0.038438  0.070094  0.033512
p464  0.187797  0.142069 -0.038438  1.000000  0.079489  0.229434
p465  0.017397 -0.069505  0.070094  0.079489  1.000000 -0.086507
p466  0.121445  0.149470  0.033512  0.229434 -0.086507  1.000000
```

We conclude that there are no significative correlations between answers. There is, however, a weak correlation between the age of the interviewee and their response to question 4 (economic goals). There seems to be no significative correlation with sex, age, education or locality variables, thus, there is no one evident factor to answering each of the questions.

Back to the Intro

## 1.9 Key indicator

Now we define a metric that measures the credit responsibility of the interviewees. We take "Always" as +1, "Never" as -1, and "Sometimes" as 0. Then we sum these independent scores to get a grade ranging from -6 to 6; finally, this grading is transformed to a final score ranging from 5 to 10, where higher scores indicate more credit responsibility. Note that the answers to question 3 must be reversed in order to be consistent with the previous interpretation.

[36]:
```
for numq in range(1,7):
    data2[f'c4{numss}{numq}'] = data1[f'p4{numss}{numq}'].map(set_norm_calif)

data2[f't4{numss}'] =␣
 ↪data2[f'c4{numss}1']+data2[f'c4{numss}2']-data2[f'c4{numss}3']+data2[f'c4{numss}4']+data2[f

data2[f't4{numss}_calif'] = data2[f't4{numss}'].map(set_tot_calif)
```

[37]:
```
data3 = data2

for numq in range(1,7):
    data3 = data3.drop(f'c4{numss}{numq}', axis = 1)

data3 = data3.drop(f't4{numss}', axis = 1)
data3 = data3.dropna()
data3.head()
```

[37]:
```
   LLAVEMOD  SEXO SEXO_HM  EDAD_V  NIV        NIVED_T  REGION   REGION_T  \
0    101101     1       M      38    8  Undergraduate       1  Northwest
1    210101     1       M      36    8  Undergraduate       6      South
2    303102     1       M      20    3          Basic       4       CDMX
3    401101     1       M      59    3          Basic       1  Northwest
4    503101     2       F      37    6    high School       4       CDMX
```

```
   TLOC  p461  …  p464  p465  p466      p461_t      p462_t      p463_t  \
0     1     1  …     2     3     2      Always      Always   Sometimes
1     3     1  …     1     1     1      Always      Always   Sometimes
2     1     2  …     2     3     3   Sometimes   Sometimes   Sometimes
3     1     1  …     3     1     2      Always      Always      Always
4     1     1  …     1     3     3      Always      Always       Never

      p464_t  p465_t     p466_t t46_calif
0  Sometimes   Never  Sometimes  7.916667
1     Always  Always     Always  9.583333
2  Sometimes   Never      Never  6.666667
3      Never  Always  Sometimes  7.916667
4     Always   Never      Never  8.333333

[5 rows x 22 columns]
```

In the sequel, we will work with table `data3` for visualizing the KPI and conducting the predictive analysis. We can export this data to other visualizing tools.

[38]:
```
data3.to_csv(f'./tabla_4{numss}.csv')
```

### 1.9.1 Visualizing the KPI

We analyze the distribution of the KPI with respect to the independent variables. As it is a continuous metric, we start with simple descriptive statistics.

[39]:
```
pd.DataFrame(data3[f't4{numss}_calif']).describe()
```

[39]:

| | t46_calif |
|---|---|
| count | 13196.000000 |
| mean | 8.118496 |
| std | 0.765402 |
| min | 5.000000 |
| 25% | 7.500000 |
| 50% | 8.333333 |
| 75% | 8.750000 |
| max | 10.000000 |

We get the mean $\bar{x} = 8.11$, median $\text{MED} = 8.\bar{3}$, and the standard deviation $\sigma = 0.76$. The mean is slightly less than the median, which indicates that the values in the first and seconds quartiles are sparser.

[43]:
```
data3_desc = pd.DataFrame(data3[f't4{numss}_calif']).describe()

mu = data3_desc.loc['mean','t46_calif']
sig = data3_desc.loc['std','t46_calif']

print(f'We claim that 95% of the sample is contained within the range␣
 ↪[{round(mu-2*sig,2)},{round(mu+2*sig,2)}].')
```

We claim that 95% of the sample is contained within the range [6.59,9.65].

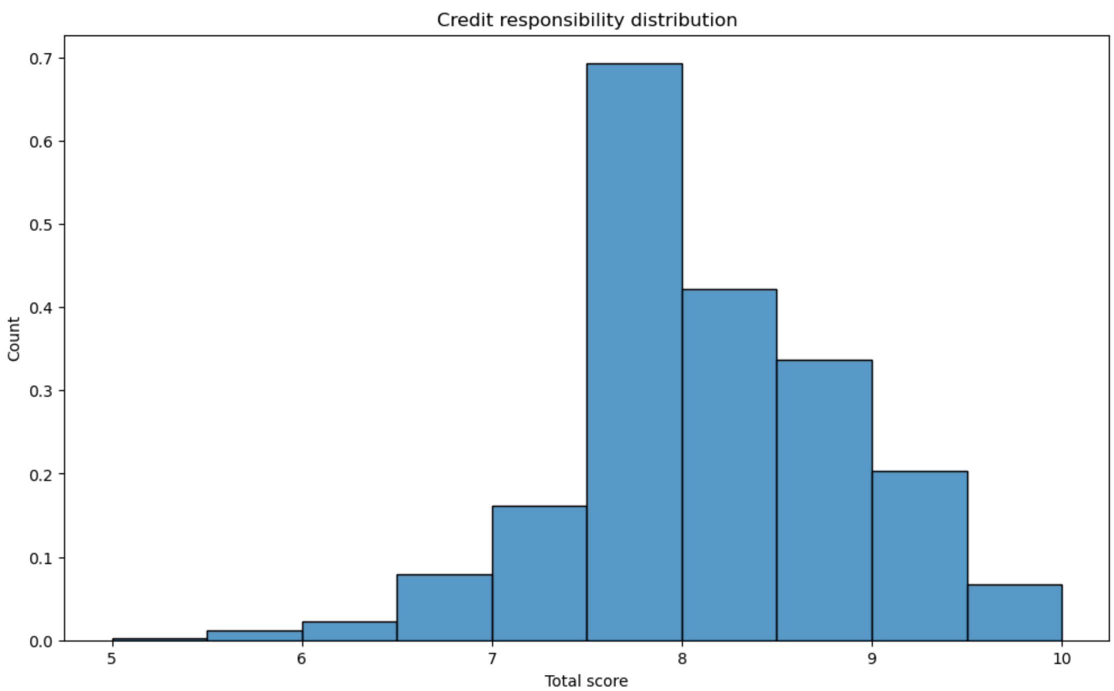
[45]:
```
q1 = data3_desc.loc['25%','t46_calif']
q3 = data3_desc.loc['75%','t46_calif']
k = 1.5

print(f'Moreover, any entry less than {round(q1-k*(q3-q1),2)} and greater than␣
↪{round(q3+k*(q3-q1),2)} is considered as outliers.')
```

```
Moreover, any entry less than 5.62 and greater than 10.62 is considered as
outliers.
```

Now we visualize the KPI.

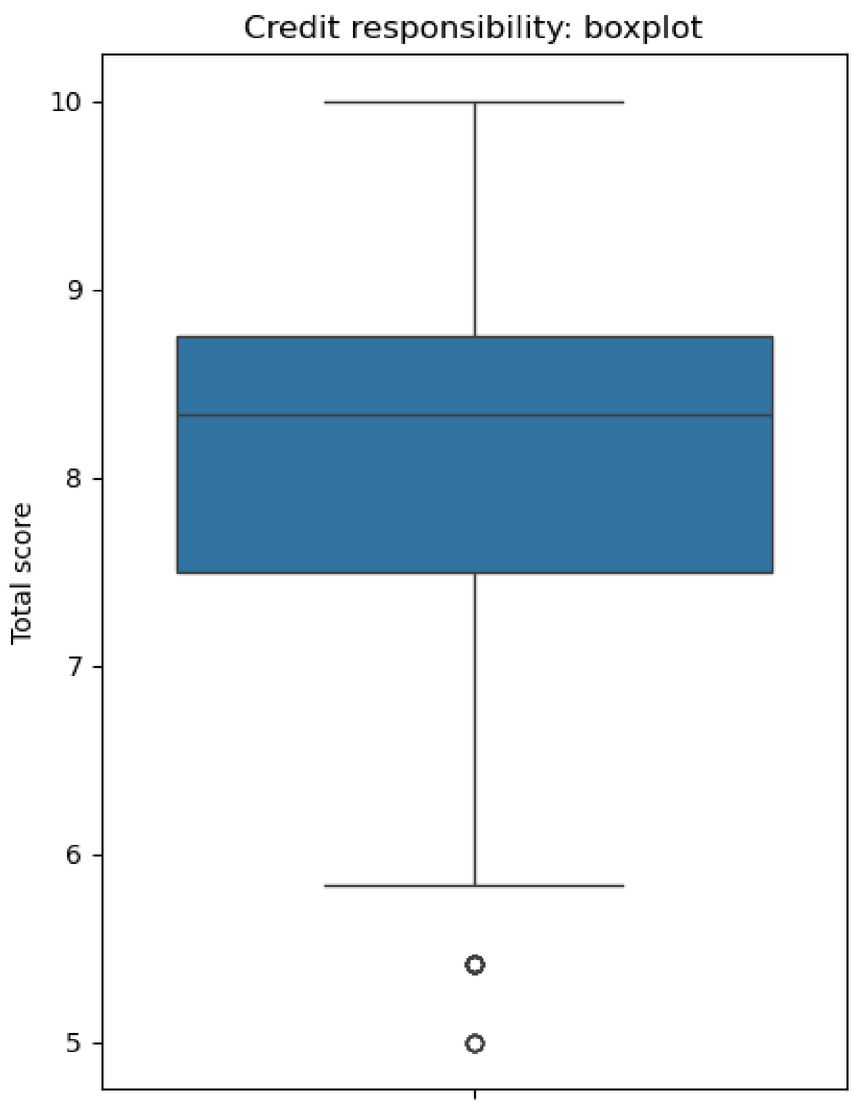
[46]:
```
### General histogram

fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data3, x = f't4{numss}_calif', stat = 'density', bins=10)

ax.set_xlabel('Total score')
ax.set_ylabel('Count')
ax.set_title('Credit responsibility distribution')
```

[46]:
```
Text(0.5, 1.0, 'Credit responsibility distribution')
```



[47]:
```
### General boxplot

fig, ax = plt.subplots(figsize=(5,7))

plot1 = sns.boxplot(data3, y = f't4{numss}_calif')

ax.set_ylabel('Total score')
ax.set_title('Credit responsibility: boxplot')
```

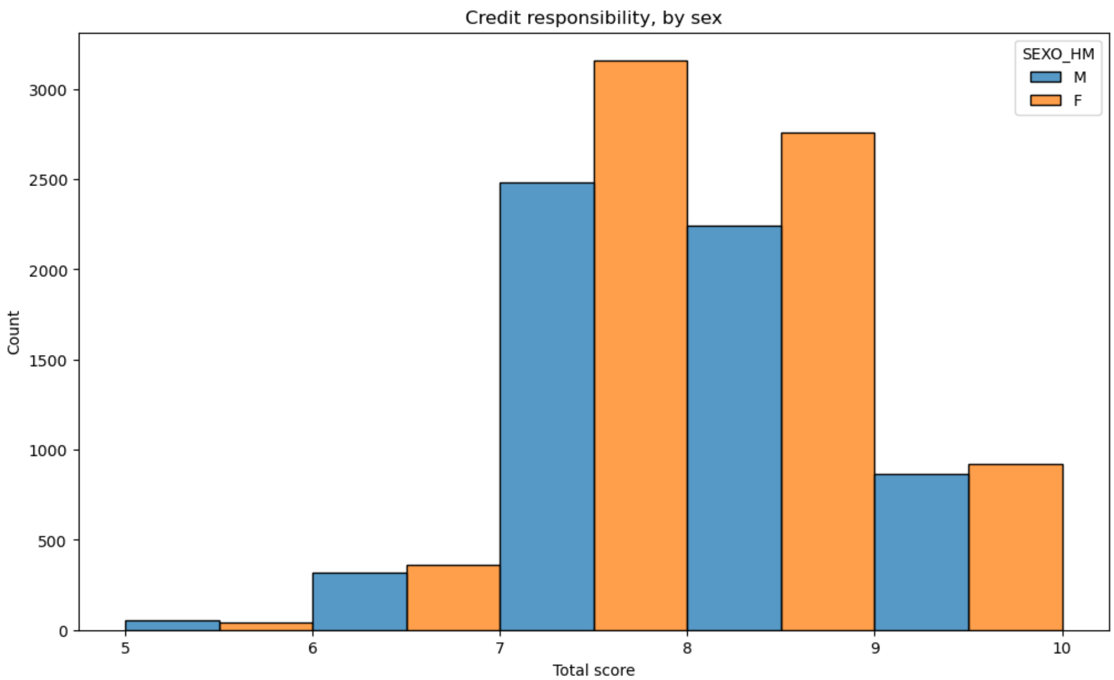
[47]: Text(0.5, 1.0, 'Credit responsibility: boxplot')




**KPI by sex**

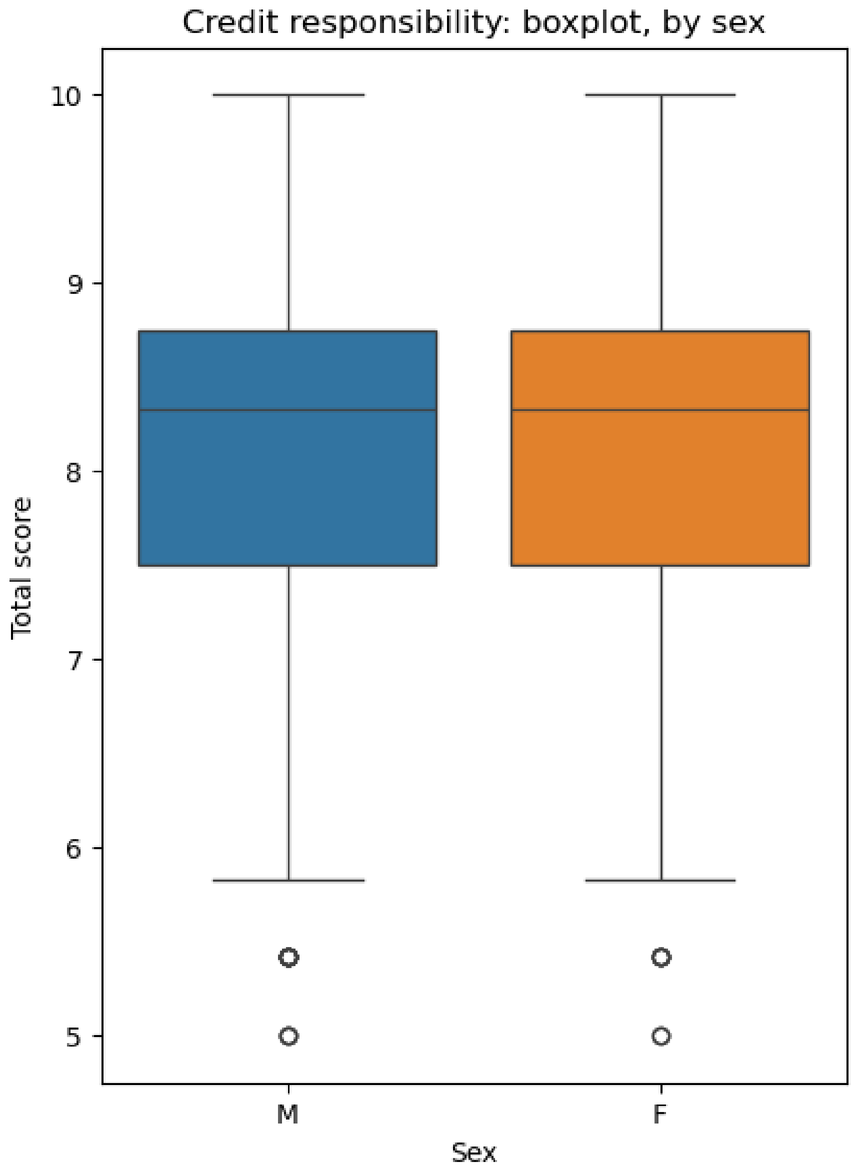
[48]:
```
### Histograms by variables

fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data3, x = f't4{numss}_calif', multiple="dodge", hue =␣
↪'SEXO_HM', bins=5)

ax.set_xlabel('Total score')
ax.set_ylabel('Count')
ax.set_title('Credit responsibility, by sex')
```

[48]:
```
Text(0.5, 1.0, 'Credit responsibility, by sex')
```




[49]:
```
### Boxplot by sex

fig, ax = plt.subplots(figsize=(5,7))

plot1 = sns.boxplot(data3, x = 'SEXO_HM', y = f't4{numss}_calif', hue =␣
↪'SEXO_HM')

ax.set_ylabel('Total score')
ax.set_xlabel('Sex')
ax.set_title('Credit responsibility: boxplot, by sex')
```

[49]:
```
Text(0.5, 1.0, 'Credit responsibility: boxplot, by sex')
```

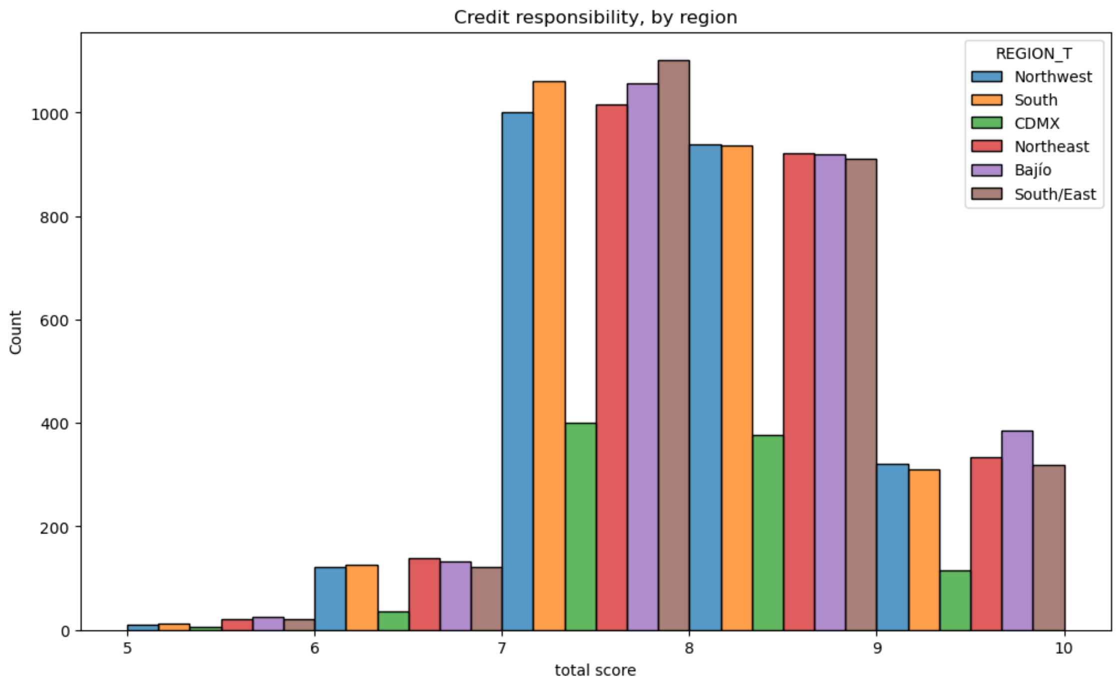



### 1.9.2 KPI by region

[50]:
```
### Histograms by variables

fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data3, x = f't4{numss}_calif', multiple="dodge", hue =␣
↪'REGION_T', bins=5)

ax.set_xlabel('total score')
ax.set_ylabel('Count')
```

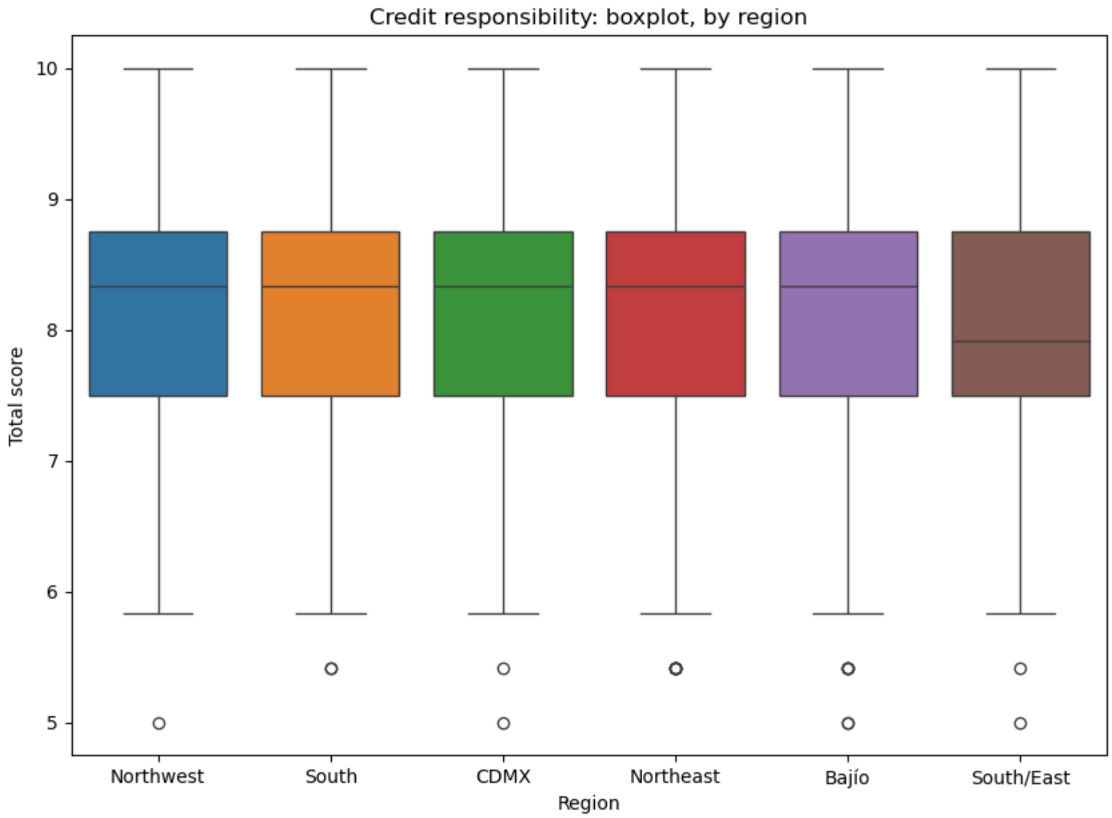
```python
ax.set_title('Credit responsibility, by region')
```

[50]: Text(0.5, 1.0, 'Credit responsibility, by region')




[51]:
```python
### Boxplot by region

fig, ax = plt.subplots(figsize=(10,7))

plot1 = sns.boxplot(data3, x = 'REGION_T', y = f't4{numss}_calif', hue =␣
↪'REGION_T')

ax.set_ylabel('Total score')
ax.set_xlabel('Region')
ax.set_title('Credit responsibility: boxplot, by region')
```

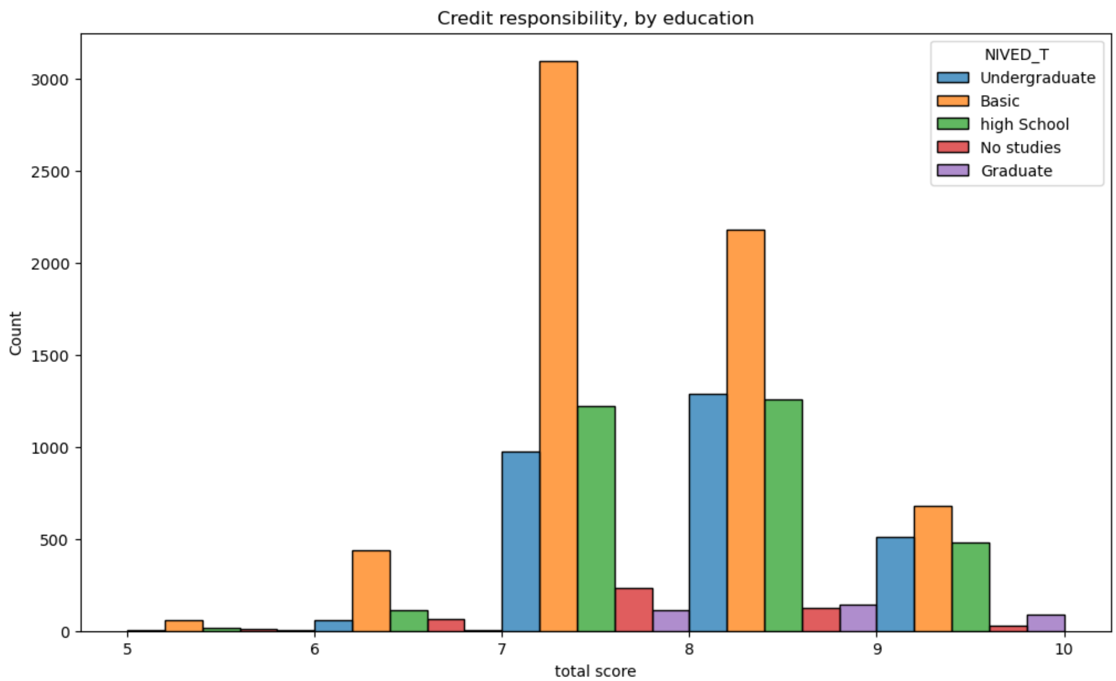
[51]: Text(0.5, 1.0, 'Credit responsibility: boxplot, by region')




### 1.9.3 KPI by education

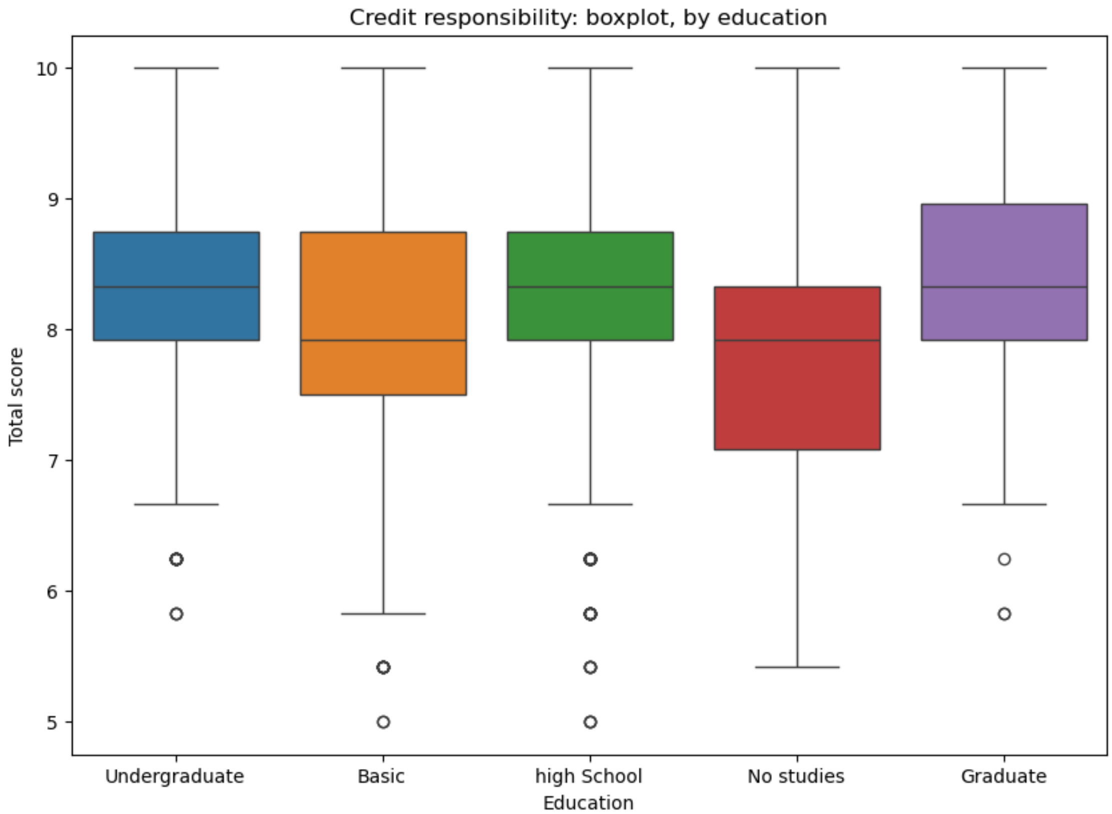
[52]:
```
### Histograms by variable

fig, ax = plt.subplots(figsize=(12,7))

plot1 = sns.histplot(data3, x = f't4{numss}_calif', multiple="dodge", hue =␣
↪'NIVED_T', bins=5)

ax.set_xlabel('total score')
ax.set_ylabel('Count')
ax.set_title('Credit responsibility, by education')
```

[52]:
```
Text(0.5, 1.0, 'Credit responsibility, by education')
```




[53]:
```
### Boxplot por nivel educativo

fig, ax = plt.subplots(figsize=(10,7))

plot1 = sns.boxplot(data3, x = 'NIVED_T', y = f't4{numss}_calif', hue =␣
 ↪'NIVED_T')

ax.set_ylabel('Total score')
ax.set_xlabel('Education')
ax.set_title('Credit responsibility: boxplot, by education')
```

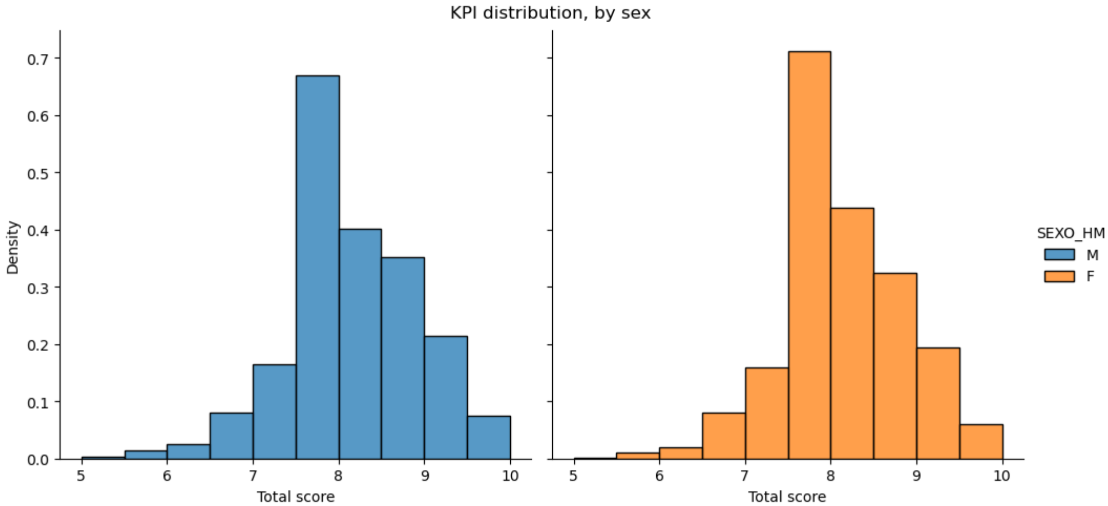
[53]:
```
Text(0.5, 1.0, 'Credit responsibility: boxplot, by education')
```




### 1.9.4 Filtered histograms

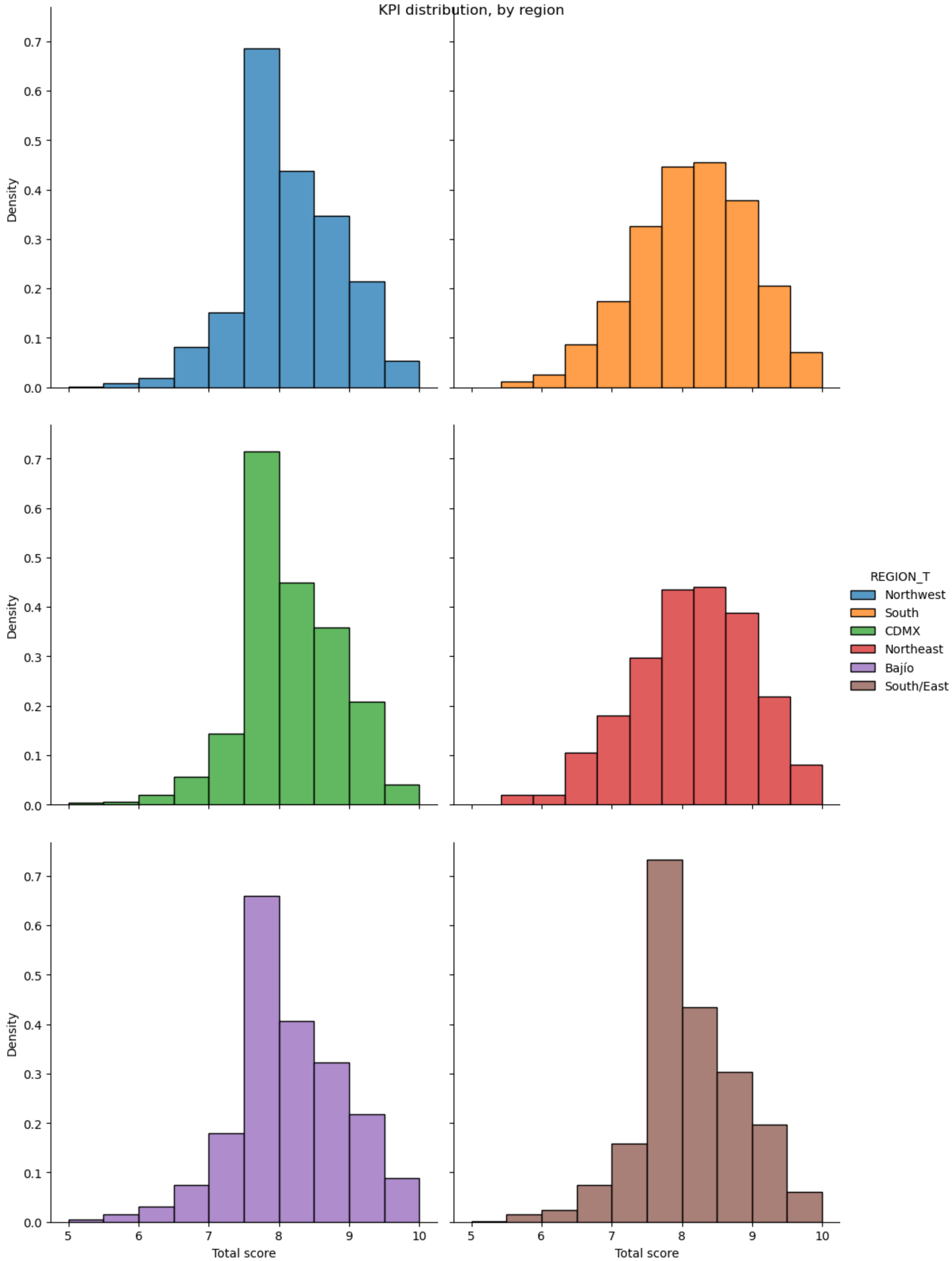
[54]:
```
### Filtered histograms

### By sex

g = sns.FacetGrid(data3, col="SEXO_HM", hue="SEXO_HM", height = 5)
g.map(sns.histplot, f't4{numss}_calif', stat='density', bins=10)
g.set_axis_labels("Total score", "Density")
g.set_titles('')
g.fig.suptitle('KPI distribution, by sex')
g.add_legend()
```

[54]: <seaborn.axisgrid.FacetGrid at 0x23b295cf530>




```
[55]: ### Filtered histograms

### By region

g = sns.FacetGrid(data3, col="REGION_T", hue="REGION_T", height = 5, col_wrap =␣
  ↪2)
g.map(sns.histplot, f't4{numss}_calif', stat='density', bins=10)
g.set_axis_labels("Total score", "Density")
g.set_titles('')
g.fig.suptitle('KPI distribution, by region')
g.add_legend()
```

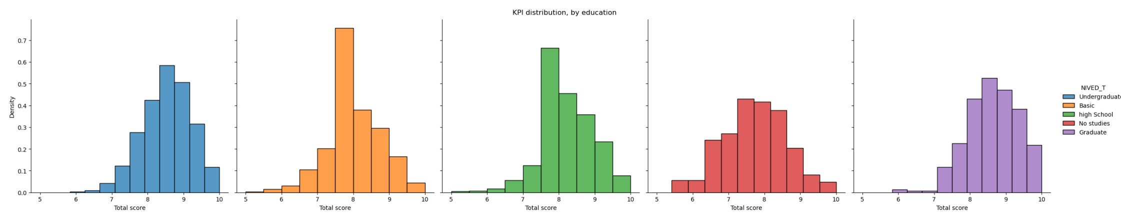
```
[55]: <seaborn.axisgrid.FacetGrid at 0x23b29013530>
```




[56]:
```
### Filtered histograms

### By region
```

```
g = sns.FacetGrid(data3, col="NIVED_T", hue="NIVED_T", height = 5)
g.map(sns.histplot, f't4{numss}_calif', stat='density', bins=10)
g.set_axis_labels("Total score", "Density")
g.set_titles('')
g.fig.suptitle('KPI distribution, by education')
g.add_legend()
```

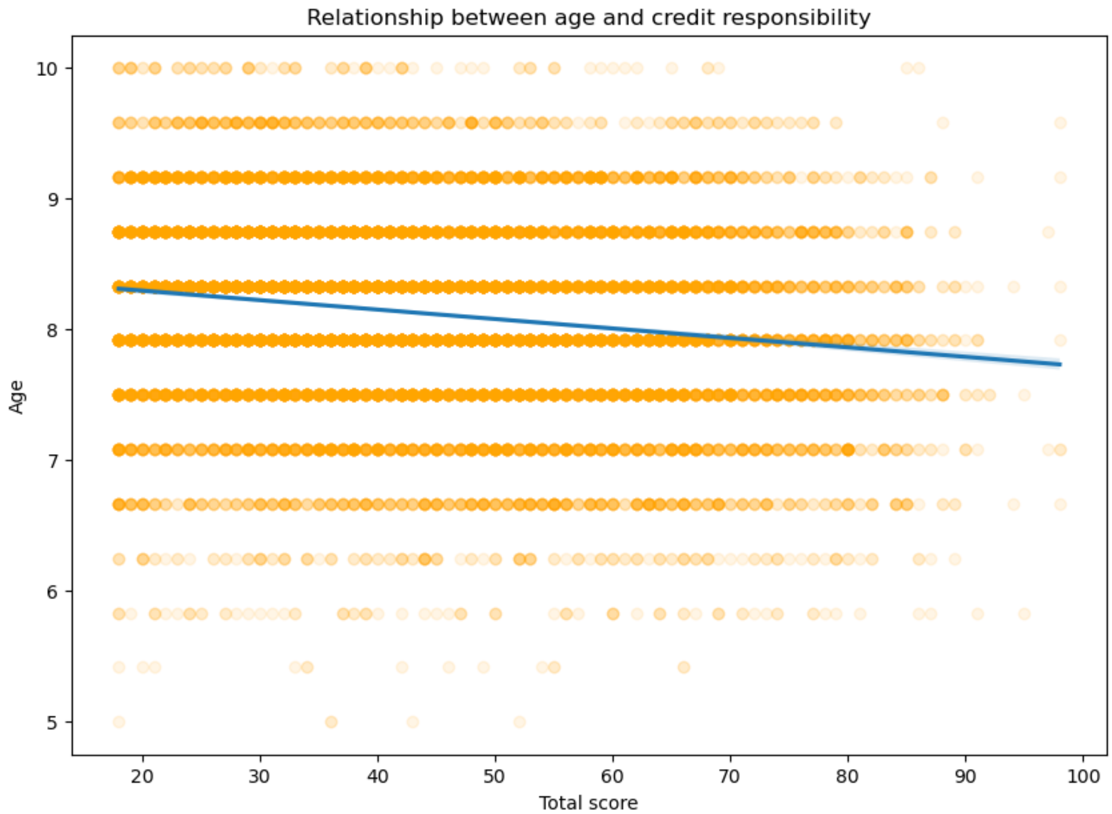
[56]: <seaborn.axisgrid.FacetGrid at 0x23b288978c0>




### 1.9.5 Age and credit responsibility

[57]:
```
fig, ax = plt.subplots(figsize=(10,7))

plot1 = sns.regplot(data3, x = 'EDAD_V', y = f't4{numss}_calif',␣
  ↪scatter_kws=dict(color="orange", alpha = 0.1))

ax.set_xlabel('Total score')
ax.set_ylabel('Age')
ax.set_title('Relationship between age and credit responsibility')
```

[57]: Text(0.5, 1.0, 'Relationship between age and credit responsibility')




We can draw some conclusions: - The Mexican population was positive expectations towards their credit status, with a KPI median of $8.3\bar{3}$, and only scores under 5.62 considered as outiers. - There aren't any significative differences between men and women, nor amongst regions, regarding the KPI. - There are some differences in KPI distributions amongst education levels: undergraduate and graduate subpopulations tend to be more responsible towards their credit status (MED = 8.33).

Back to the Intro

## 1.10 Predictive analysis

Now we try to predict the answers to the questions given the independent variables: sex, age, education, and locality. A posteriori, we may exclude one or more variables if they are not seen to be significative in predictive power.

### 1.10.1 Classification model

We will determine which answers, and with which certainty, can be predicted using the independent variables. To do this, we consider three different ML methods: logistic regression, decision trees, and K nearest neighbours.

#### Logistic regression

[58]:
```
### Useful for question 4.6.4

### Question

numq = 4

### Independent variables

X = data3[['SEXO','EDAD_V', 'NIV', 'TLOC']].to_numpy()

### Dependient variable

Y = data3[f'p4{numss}{numq}_t']

### Training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,␣
↪random_state=42)

### Method

pipeline = Pipeline(steps=[("scaler", StandardScaler()), ("lr",␣
↪LogisticRegression(max_iter=10000000))])
pipeline.fit(X_train,y_train)

### Confusion matrix

y_pred = pipeline.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[1373  104  270]
 [ 398  169  171]
 [1063  130  281]]
```

[59]:
```
### Reporte de la clasificación

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

      Always       0.48      0.79      0.60      1747
       Never       0.42      0.23      0.30       738
   Sometimes       0.39      0.19      0.26      1474

    accuracy                           0.46      3959
   macro avg       0.43      0.40      0.38      3959
weighted avg       0.44      0.46      0.42      3959
```

### Decision Tree

[60]:
```
### Useful for questions 4.6.1 and 4.6.2

### Question

numq = 1

### Independent variables

X = data3[['SEXO','EDAD_V', 'NIV', 'TLOC']].to_numpy()

### Dependient variable

Y = data3[f'p4{numss}{numq}_t']

### Training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,␣
↪random_state=42)

### Method

pipeline = Pipeline(steps=[("scaler", StandardScaler()), ("tree",␣
↪DecisionTreeClassifier())])
pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[2642   46  185]
 [ 177    4   23]
 [ 793   12   77]]
```

[61]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

      Always       0.73      0.92      0.81      2873
       Never       0.06      0.02      0.03       204
   Sometimes       0.27      0.09      0.13       882

    accuracy                           0.69      3959
   macro avg       0.36      0.34      0.33      3959
weighted avg       0.59      0.69      0.62      3959
```

### K Nearest Neighbours

```
[62]: ### Useful for question 4.6.1

### Question

numq = 1

### Independent variables

X = data3[['SEXO','EDAD_V', 'NIV', 'TLOC']].to_numpy()

### Dependient variable

Y = data3[f'p4{numss}{numq}_t']

### Training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,␣
↪random_state=42)

### Method

pipeline = Pipeline(steps=[("scaler", StandardScaler()), ("knn",␣
↪KNeighborsClassifier(n_neighbors=20))])
pipeline.fit(X_train,y_train)

y_pred = pipeline.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
[[2857    6   10]
 [ 204    0    0]
 [ 878    3    1]]
```

```
[63]: print(classification_report(y_test,y_pred,zero_division = 0.0))
```

```
              precision    recall  f1-score   support

      Always       0.73      0.99      0.84      2873
       Never       0.00      0.00      0.00       204
   Sometimes       0.09      0.00      0.00       882

    accuracy                           0.72      3959
   macro avg       0.27      0.33      0.28      3959
weighted avg       0.55      0.72      0.61      3959
```

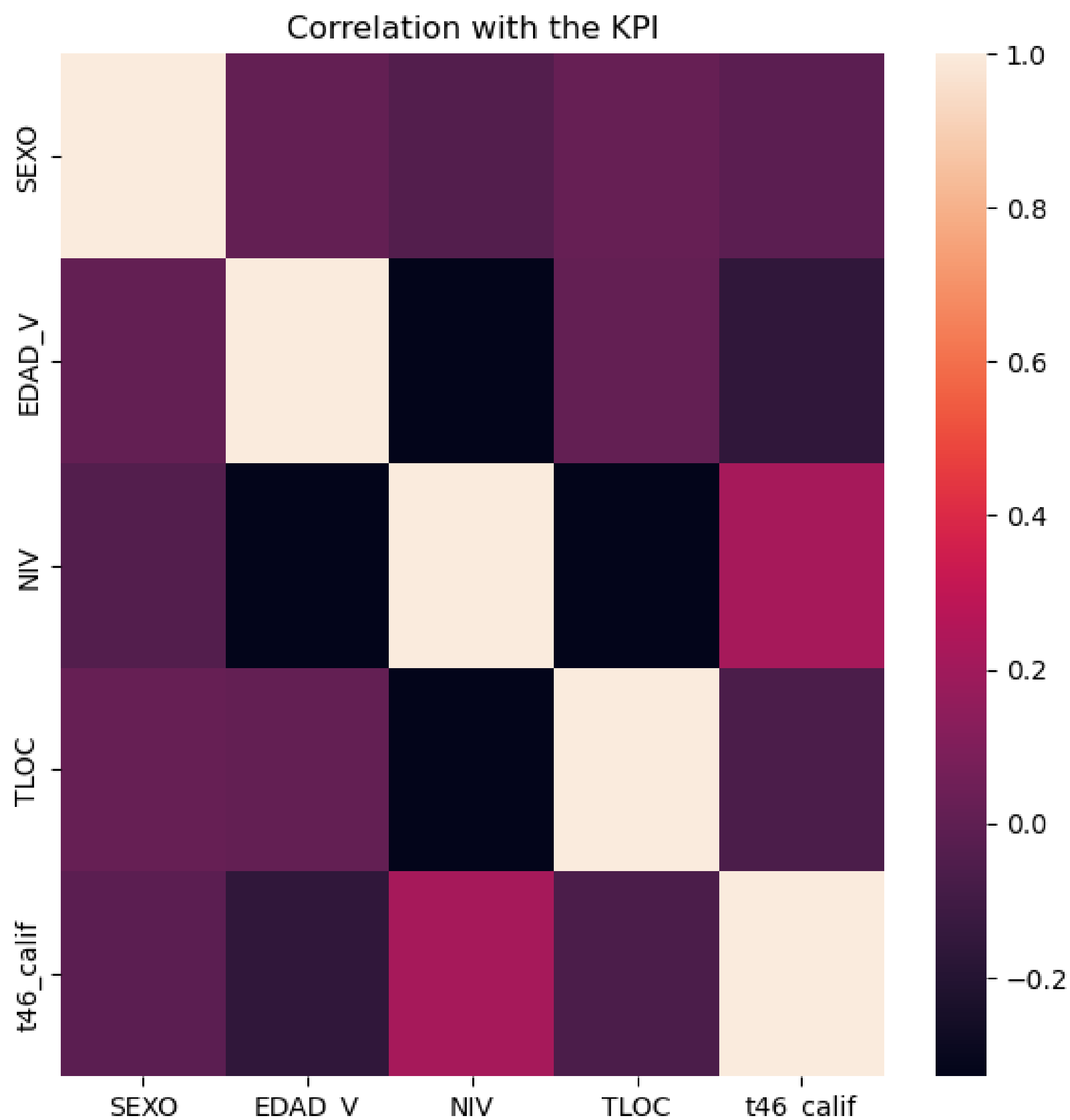
Our analysis is inconclusive for questions 4.6.3, 4.6.5, and 4.6.6, as the methods classify all answers under “Always”. This is possibly due to the semi-categorical nature of the independent variables and/or unbalanced data, i.e. overrepresented answers. More advanced techniques are required.

### 1.10.2 Regression model

Now we wish to predict credit responsibility using the independent variables. First, we wish to determine any obvious variables with significative contribution on the target variable.

[65]:
```
corr2 = data3[['SEXO', 'EDAD_V', 'NIV', 'TLOC', f't4{numss}_calif']].corr()

fig, ax = plt.subplots(figsize=(7,7))

# plot the heatmap
sns.heatmap(corr2)

ax.set_title('Correlation with the KPI')
```

[65]:
```
Text(0.5, 1.0, 'Correlation with the KPI')
```



[66]:
```
corr2.head(5)
```

[66]:

| | SEXO | EDAD_V | NIV | TLOC | t46_calif |
|---|---|---|---|---|---|
| SEXO | 1.000000 | 0.011764 | -0.043006 | 0.021964 | -0.013346 |
| EDAD_V | 0.011764 | 1.000000 | -0.329022 | 0.012414 | -0.158654 |
| NIV | -0.043006 | -0.329022 | 1.000000 | -0.327809 | 0.220502 |
| TLOC | 0.021964 | 0.012414 | -0.327809 | 1.000000 | -0.065835 |
| t46_calif | -0.013346 | -0.158654 | 0.220502 | -0.065835 | 1.000000 |

We find a weak positive correlation between education and the KPI, as previously suggested.

[68]:
```
X = data3[['SEXO', 'EDAD_V', 'NIV','TLOC']].to_numpy()
Y = data3[f't4{numss}_calif']
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,
 ↪random_state=42)

### Model

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)
```

[70]:
```
### Coefficients

print(lr.coef_)

### r2 score

print(lr.score(X_test, y_test))
```

```
[-0.00807646 -0.00415225  0.05767917  0.00261124]
0.05633137043838565
```

[71]:
```
### Obtaining the errors

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_pred)
rmse = root_mean_squared_error(y_true=y_test,y_pred=y_pred)

print(f'Mean absolute error: {mae}')
print(f'Mean squared error: {mse}')
print(f'Root mean squared error: {rmse}')
```

```
Mean absolute error: 0.5933030516406795
Mean squared error: 0.5517427551897718
```

```
Root mean squared error: 0.7427938847283085
```

Although we claimed that education was the most significative variable regarding credit responsibility, we conclude that the total score cannot be accurately predicted using the independent variables. This was expected as our classification task was also inconclusive, and because a previous visualization showed no observable trend between age and the KPI.

Back to the Intro

### 1.11 Summary

- Overall, the answers to the survey have similar distributions with respect to sex, region, and education.
- The answers have weak correlations with the independent variables.
- The credit responsibility score, defined as an aggregate of the answers to the survey, does show differences amongst education levels.
- The answers to questions 4.6.1 (considerations before buying), 4.6.2 (timely bill payments), and 4.6.4 (economic goals) can be predicted through the independent variables.
- Our ML analysis is not conclusive neither for the rest of the answers nor for the KPI, possibly due to overrepresentation of some answers.

### 1.12 Key takeaways

- The answers to the questions of Subsection 4.6 of the ENIF exhibit a high degree of homogeneity with respect to various socioeconomic descriptors, such as sex, age, region, and education.
- By defining and aggregate of the answers, we can observe a higher notion of economic stability from certain subpopulations, such as graduates and undergraduates when compared to other education levels.
- Credit responsibility seems to be similar amongst sexes and amongst regions.
- It is possible to predict, with some certainty, the behaviour of the population regarding responsible purchases, timely bill payments, and long-term economic goals, based on sex, age, locality, and education.

Back to the Intro