Kiva.org is an online crowdfunding platform to extend financial services to poor and financially excluded people around the world. Kiva lenders have provided over $1 billion dollars in loans to over 2 million people. In order to set investment priorities, help inform lenders, and understand their target communities, knowing the level of poverty of each borrower is critical. However, this requires inference based on a limited set of information for each borrower.

In Kaggle Datasets' inaugural Data Science for Good challenge, Kiva is inviting the Kaggle community to help them build more localized models to estimate the poverty levels of residents in the regions where Kiva has active loans. Unlike traditional machine learning competitions with rigid evaluation criteria, participants will develop their own creative approaches to addressing the objective. Instead of making a prediction file as in a supervised machine learning problem, submissions in this challenge will take the form of Python and/or R data analyses using Kernels, Kaggle's hosted Jupyter Notebooks-based workbench.

Kiva has provided a dataset of loans issued over the last two years, and participants are invited to use this data as well as source external public datasets to help Kiva build models for assessing borrower welfare levels. Participants will write kernels on this dataset to submit as solutions to this objective and five winners will be selected by Kiva judges at the close of the event. In addition, awards will be made to encourage public code and data sharing. With a stronger understanding of their borrowers and their poverty levels, Kiva will be able to better assess and maximize the impact of their work.

The sections that follow describe in more detail how to participate, win, and use available resources to make a contribution towards helping Kiva better understand and help entrepreneurs around the world.

Problem Statement For the locations in which Kiva has active loans, your objective is to pair Kiva's data with additional data sources to estimate the welfare level of borrowers in specific regions, based on shared economic and demographic characteristics.

A good solution would connect the features of each loan or product to one of several poverty mapping datasets, which indicate the average level of welfare in a region on as granular a level as possible. Many datasets indicate the poverty rate in a given area, with varying levels of granularity. Kiva would like to be able to disaggregate these regional averages by gender, sector, or borrowing behavior in order to estimate a Kiva borrower’s level of welfare using all of the relevant information about them. Strong submissions will attempt to map vaguely described locations to more accurate geocodes.

Kernels submitted will be evaluated based on the following criteria:

1. Localization - How well does a submission account for highly localized borrower situations? Leveraging a variety of external datasets and successfully building them into a single submission will be crucial.

2. Execution - Submissions should be efficiently built and clearly explained so that Kiva's team can readily employ them in their impact calculations.

3. Ingenuity - While there are many best practices to learn from in the field, there is no one way of using data to assess welfare levels. It's a challenging, nuanced field and participants should experiment with new methods and diverse datasets.

Introduction

Kiva.org is an online crowdfunding platform dedicated to extend financial services to poor people around the World.

Kiva is inviting the Kaggle community to help them build more localized models to estimate the poverty levels of residents in the regions where Kiva has active loans.

How Kiva works? The lending process by Kiva is simple, following just few steps:

The borrower mets with a Field Partner of Kiva and requests a loan;

The Field Partner disburses a loan to the borrower;

The Field Partner uploads the loan request to Kiva. The request is reviewed by a team of volunteer editors and translators and then published on Kiva.org;

Kiva lenders found the loan request, and Kiva sends the loan to the Field Partner;

The borrower makes repayments and the Field Partner send funds owed to Kiva. Kiva repays lenders;

The lenders can make another loan, donate to Kiva or withdraw their money to their PayPal account. 70% of the lenders choose to fund another loan.

In this Kernel we will explore the data provided by Kiva, trying to understand the welfare condition of Kiva borrowers.

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objs as go
from plotly.offline import iplot, init_notebook_mode
from IPython.display import display, Markdown
```

In [ ]:
```
kiva_loans=pd.read_csv(r"C:\Users\Anurag Mohanta\OneDrive\Desktop\kiva\kiva_loan
kiva_location=pd.read_csv(r"C:\Users\Anurag Mohanta\OneDrive\Desktop\kiva\kiva_m
loan_theme=pd.read_csv(r"C:\Users\Anurag Mohanta\OneDrive\Desktop\kiva\loan_them
loan_themes_by_region=pd.read_csv(r"C:\Users\Anurag Mohanta\OneDrive\Desktop\kiv
country_profile_variables_df=pd.read_csv(r"C:\Users\Anurag Mohanta\OneDrive\Desk
kiva_country_profiles_variables_df =pd.read_csv(r"C:\Users\Anurag Mohanta\OneDri
```

BASIC DATA EXPLORATION

In [ ]:
```
data = [
    ("kiva_loans", kiva_loans),
    ("kiva_location", kiva_location),
    ("loan_theme", loan_theme),
    ("loan_themes_by_region", loan_themes_by_region)
]
```

In [ ]:
```
from IPython.display import display, Markdown

def basic_exploration(df, df_name):
    print(df_name)

    # Display DataFrame head with Markdown header
    display(Markdown(f"### {df_name}"))
    display(df.head())

    # Display DataFrame shape
    display(Markdown(f"### {df_name}:shape"))
    print("Number of rows:", df.shape[0])
    print("Number of columns:", df.shape[1])

    # Display DataFrame info
    display(Markdown(f"### {df_name}:information"))
    display(df.info())

    # Display DataFrame description
    display(Markdown(f"### {df_name}:Description"))
    display(df.describe())

    # Display number of null values per column
    display(Markdown(f"### {df_name}:Null Values per Column"))
    display(df.isnull().sum())

    # Display percentage of null values relative to total rows
    display(Markdown(f"### {df_name}: Percentage of Null Values"))
    display(df.isnull().sum() / df.shape[0]*100)
```

In [ ]:
```
for name, df in data:
    basic_exploration(df, name)
```

```
kiva_loans
```

kiva_loans

	id	funded_amount	loan_amount	activity	sector	use	count
0	653051	300.0	300.0	Fruits & Vegetables	Food	To buy seasonal, fresh fruits to sell.	
1	653053	575.0	575.0	Rickshaw	Transportation	to repair and maintain the auto rickshaw used ...	
2	653068	150.0	150.0	Transportation	Transportation	To repair their old cycle-van and buy another ...	
3	653063	200.0	200.0	Embroidery	Arts	to purchase an embroidery machine and a variet...	
4	653084	400.0	400.0	Milk Sales	Food	to purchase one buffalo.	

kiva_loans:shape

```
Number of rows: 671205
Number of columns: 20
```

kiva_loans:information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 671205 entries, 0 to 671204
Data columns (total 20 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   id                  671205 non-null  int64
 1   funded_amount       671205 non-null  float64
 2   loan_amount         671205 non-null  float64
 3   activity            671205 non-null  object
 4   sector              671205 non-null  object
 5   use                 666973 non-null  object
 6   country_code        671197 non-null  object
 7   country             671205 non-null  object
 8   region              614405 non-null  object
 9   currency            671205 non-null  object
 10  partner_id          657698 non-null  float64
 11  posted_time         671205 non-null  object
 12  disbursed_time      668809 non-null  object
 13  funded_time         622874 non-null  object
 14  term_in_months      671205 non-null  float64
 15  lender_count        671205 non-null  int64
 16  tags                499789 non-null  object
 17  borrower_genders    666984 non-null  object
 18  repayment_interval  671205 non-null  object
 19  date                671205 non-null  object
dtypes: float64(4), int64(2), object(14)
memory usage: 102.4+ MB
None
```

kiva_loans:Description

	id	funded_amount	loan_amount	partner_id	term_in_months	le
count	6.712050e+05	671205.000000	671205.000000	657698.000000	671205.000000	671
mean	9.932486e+05	785.995061	842.397107	178.199616	13.739022	
std	1.966113e+05	1130.398941	1198.660073	94.247581	8.598919	
min	6.530470e+05	0.000000	25.000000	9.000000	1.000000	
25%	8.230720e+05	250.000000	275.000000	126.000000	8.000000	
50%	9.927800e+05	450.000000	500.000000	145.000000	13.000000	
75%	1.163653e+06	900.000000	1000.000000	204.000000	14.000000	
max	1.340339e+06	100000.000000	100000.000000	536.000000	158.000000	2

kiva_loans:Null Values per Column

```
id                        0
funded_amount             0
loan_amount               0
activity                  0
sector                    0
use                    4232
country_code              8
country                   0
region                56800
currency                  0
partner_id            13507
posted_time               0
disbursed_time         2396
funded_time           48331
term_in_months            0
lender_count              0
tags                 171416
borrower_genders       4221
repayment_interval        0
date                      0
dtype: int64
```

kiva_loans: Percentage of Null Values

```
id                    0.000000
funded_amount         0.000000
loan_amount           0.000000
activity              0.000000
sector                0.000000
use                   0.630508
country_code          0.001192
country               0.000000
region                8.462392
currency              0.000000
partner_id            2.012351
posted_time           0.000000
disbursed_time        0.356970
funded_time           7.200632
term_in_months        0.000000
lender_count          0.000000
tags                 25.538546
borrower_genders      0.628869
repayment_interval    0.000000
date                  0.000000
dtype: float64
kiva_location
```

kiva_location

	LocationName	ISO	country	region	world_region	MPI	geo
0	Badakhshan, Afghanistan	AFG	Afghanistan	Badakhshan	South Asia	0.387	(36.7347725, 70.81199529999999)
1	Badghis, Afghanistan	AFG	Afghanistan	Badghis	South Asia	0.466	(35.1671339, 63.7695384)
2	Baghlan, Afghanistan	AFG	Afghanistan	Baghlan	South Asia	0.300	(35.8042947, 69.2877535)
3	Balkh, Afghanistan	AFG	Afghanistan	Balkh	South Asia	0.301	(36.7550603, 66.8975372)
4	Bamyan, Afghanistan	AFG	Afghanistan	Bamyan	South Asia	0.325	(34.8100067, 67.8212104)

kiva_location:shape

```
Number of rows: 2772
Number of columns: 9
```

kiva_location:information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   LocationName  984 non-null    object
 1   ISO           1008 non-null   object
 2   country       1008 non-null   object
 3   region        984 non-null    object
 4   world_region  1008 non-null   object
 5   MPI           984 non-null    float64
 6   geo           2772 non-null   object
 7   lat           892 non-null    float64
 8   lon           892 non-null    float64
dtypes: float64(3), object(6)
memory usage: 195.0+ KB
None
```

kiva_location:Description

	MPI	lat	lon
count	984.000000	892.000000	892.000000
mean	0.211330	9.169710	16.635888
std	0.183621	16.484531	61.234566
min	0.000000	-34.947896	-122.747131
25%	0.053000	-1.027901	-12.819854
50%	0.155000	11.211379	26.419389
75%	0.341500	18.084292	47.019436
max	0.744000	49.264748	138.581284

kiva_location:Null Values per Column

```
LocationName    1788
ISO             1764
country         1764
region          1788
world_region    1764
MPI             1788
geo                0
lat             1880
lon             1880
dtype: int64
```

kiva_location: Percentage of Null Values

```
LocationName    64.502165
ISO             63.636364
country         63.636364
region          64.502165
world_region    63.636364
MPI             64.502165
geo              0.000000
lat             67.821068
lon             67.821068
dtype: float64
loan_theme
```

loan_theme

	id	Loan Theme ID	Loan Theme Type	Partner ID
0	638631	a1050000000skGl	General	151.0
1	640322	a1050000000skGl	General	151.0
2	641006	a1050000002X1ij	Higher Education	160.0
3	641019	a1050000002X1ij	Higher Education	160.0
4	641594	a1050000002VbsW	Subsistence Agriculture	336.0

loan_theme:shape

```
Number of rows: 779092
Number of columns: 4
```

loan_theme:information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 779092 entries, 0 to 779091
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               779092 non-null  int64
 1   Loan Theme ID    764279 non-null  object
 2   Loan Theme Type  764279 non-null  object
 3   Partner ID       764279 non-null  float64
dtypes: float64(1), int64(1), object(2)
memory usage: 23.8+ MB
None
```

loan_theme:Description

	id	Partner ID
count	7.790920e+05	764279.000000
mean	1.047475e+06	180.825840
std	2.282538e+05	97.914029
min	6.386310e+05	9.000000
25%	8.499768e+05	126.000000
50%	1.046528e+06	145.000000
75%	1.244768e+06	204.000000
max	1.444243e+06	557.000000

loan_theme:Null Values per Column

```
id                     0
Loan Theme ID      14813
Loan Theme Type    14813
Partner ID         14813
dtype: int64
```

loan_theme: Percentage of Null Values

```
id                 0.000000
Loan Theme ID      1.901316
Loan Theme Type    1.901316
Partner ID         1.901316
dtype: float64
loan_themes_by_region
```

loan_themes_by_region

	Partner ID	Field Partner Name	sector	Loan Theme ID	Loan Theme Type	country	forkiva	
0	9	KREDIT Microfinance Institution	General Financial Inclusion	a1050000000slfi	Higher Education	Cambodia	No	M
1	9	KREDIT Microfinance Institution	General Financial Inclusion	a10500000068jPe	Vulnerable Populations	Cambodia	No	Batt I
2	9	KREDIT Microfinance Institution	General Financial Inclusion	a1050000000slfi	Higher Education	Cambodia	No	Batt I
3	9	KREDIT Microfinance Institution	General Financial Inclusion	a10500000068jPe	Vulnerable Populations	Cambodia	No	K I
4	9	KREDIT Microfinance Institution	General Financial Inclusion	a1050000002X1Uu	Sanitation	Cambodia	No	K I

5 rows × 21 columns

loan_themes_by_region:shape

```
Number of rows: 15736
Number of columns: 21
```

loan_themes_by_region:information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15736 entries, 0 to 15735
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Partner ID          15736 non-null  int64
 1   Field Partner Name  15736 non-null  object
 2   sector              15736 non-null  object
 3   Loan Theme ID       15736 non-null  object
 4   Loan Theme Type     15736 non-null  object
 5   country             15736 non-null  object
 6   forkiva             15736 non-null  object
 7   region              15736 non-null  object
 8   geocode_old         1200 non-null   object
 9   ISO                 15722 non-null  object
 10  number              15736 non-null  int64
 11  amount              15736 non-null  int64
 12  LocationName        15736 non-null  object
 13  geocode             13662 non-null  object
 14  names               13661 non-null  object
 15  geo                 15736 non-null  object
 16  lat                 13662 non-null  float64
 17  lon                 13662 non-null  float64
 18  mpi_region          15722 non-null  object
 19  mpi_geo             9671 non-null   object
 20  rural_pct           14344 non-null  float64
dtypes: float64(3), int64(3), object(15)
memory usage: 2.5+ MB
None
```

loan_themes_by_region:Description

	Partner ID	number	amount	lat	lon	rural_p
count	15736.000000	15736.000000	1.573600e+04	13662.000000	13662.000000	14344.0000
mean	191.376144	53.628432	2.003991e+04	14.328878	29.433569	68.0846
std	118.705003	403.079799	9.636941e+04	16.757689	83.255739	26.5500
min	9.000000	1.000000	2.500000e+01	-34.610548	-172.790661	0.0000
25%	123.000000	1.000000	9.500000e+02	6.266728	-71.967463	60.0000
50%	154.000000	4.000000	2.600000e+03	13.484101	41.899993	73.0000
75%	217.000000	15.000000	8.750000e+03	22.189940	106.677321	85.0000
max	545.000000	22538.000000	4.929900e+06	51.314017	159.972900	100.0000

loan_themes_by_region:Null Values per Column

```
Partner ID                0
Field Partner Name        0
sector                    0
Loan Theme ID             0
Loan Theme Type           0
country                   0
forkiva                   0
region                    0
geocode_old           14536
ISO                      14
number                    0
amount                    0
LocationName              0
geocode                2074
names                  2075
geo                       0
lat                    2074
lon                    2074
mpi_region               14
mpi_geo                6065
rural_pct              1392
dtype: int64
```

loan_themes_by_region: Percentage of Null Values

```
Partner ID             0.000000
Field Partner Name     0.000000
sector                 0.000000
Loan Theme ID          0.000000
Loan Theme Type        0.000000
country                0.000000
forkiva                0.000000
region                 0.000000
geocode_old           92.374174
ISO                    0.088968
number                 0.000000
amount                 0.000000
LocationName           0.000000
geocode               13.179969
names                 13.186324
geo                    0.000000
lat                   13.179969
lon                   13.179969
mpi_region             0.088968
mpi_geo               38.542196
rural_pct              8.845958
```

Loans in Number

In [ ]:
```
Markdown("""
Let’s explore now into more details the loans information.
There are a number of **{}** loans, grouped in **{}** sectors, for **{}** activi
The funded amounts are between **{}** and **{}** while the loan amount is betwee
The term in months for repayment of loan ranges from **{}** to **{}** months.
<font color="#64a444">

Let’s represent all sectors, and top 20 for activities, countries and regions by
""".format(kiva_loans.shape[0],
        kiva_loans.sector.nunique(),
```

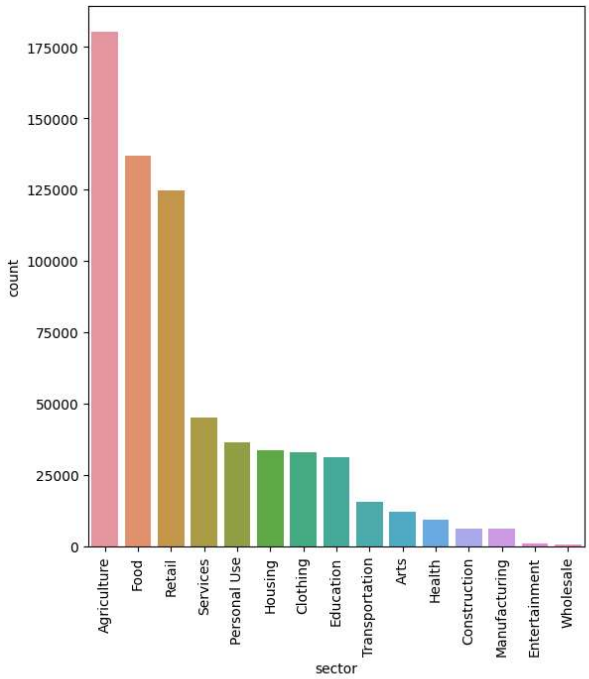
```
        kiva_loans.activity.nunique(),
        kiva_loans.country.nunique(),
        kiva_loans.region.nunique(),
        kiva_loans.funded_amount.min(),
        kiva_loans.funded_amount.max(),
        kiva_loans.loan_amount.min(),
        kiva_loans.loan_amount.max(),
        kiva_loans.currency.nunique(),
        kiva_loans.term_in_months.min(),
        kiva_loans.term_in_months.max())
)
```

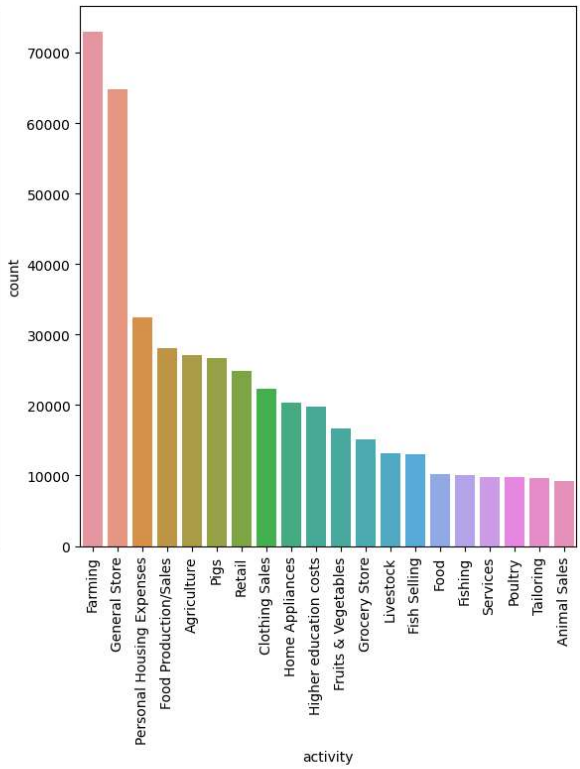
Out[ ]: Let's explore now into more details the loans information.
There are a number of **671205** loans, grouped in **15** sectors, for **163** activities, in **87** countries and in **12695** different regions.
The funded amounts are between **0.0** and **100000.0** while the loan amount is between **25.0** and **100000.0**, in **67** different currencies.
The term in months for repayment of loan ranges from **1.0** to **158.0** months.

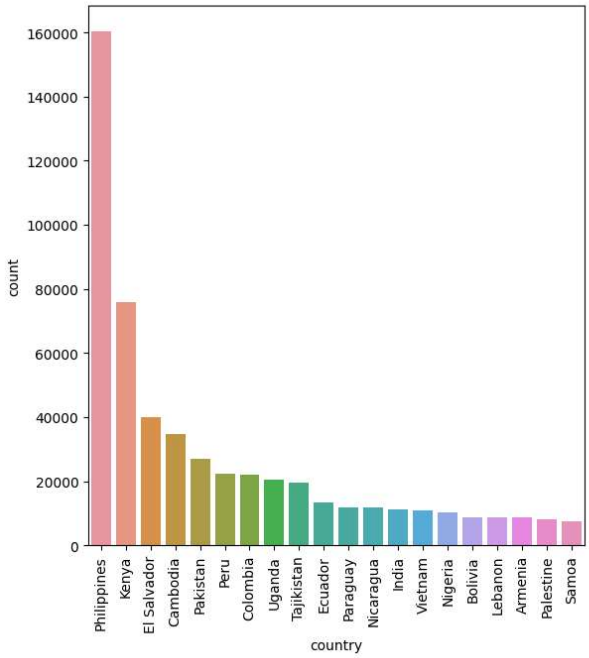
Let's represent all sectors, and top 20 for activities, countries and regions by number of loans.

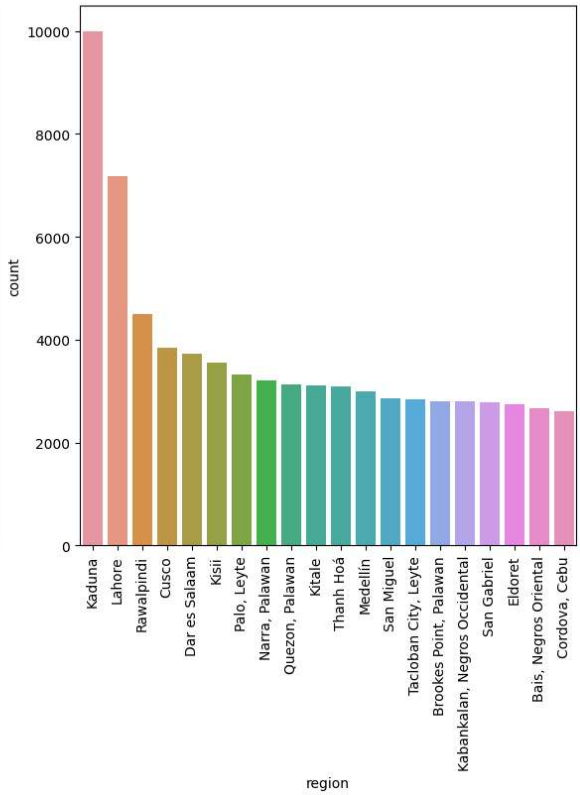
In [ ]:
```
plt.figure()

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 16))

df = kiva_loans.groupby(["sector"])["loan_amount"].agg(["count", "min", "max", "
df = df.sort_values(by="count", ascending=False)
sectors = sns.barplot(x=df['sector'], y=df['count'], ax=axs[0, 0])
sectors.set_xticklabels(sectors.get_xticklabels(), rotation=90)

df2 = kiva_loans.groupby(["activity"])["loan_amount"].agg(["count", "min", "max"
df2 = df2.sort_values(by="count", ascending=False)
df2 = df2[0:20]
activities = sns.barplot(x=df2['activity'], y=df2['count'], ax=axs[0, 1])
activities.set_xticklabels(activities.get_xticklabels(), rotation=90)

df3 = kiva_loans.groupby(["country"])["loan_amount"].agg(["count", "min", "max",
df3 = df3.sort_values(by="count", ascending=False)
df3 = df3[0:20]
countries = sns.barplot(x=df3['country'], y=df3['count'], ax=axs[1, 0])
countries.set_xticklabels(countries.get_xticklabels(), rotation=90)

df4 = kiva_loans.groupby(["region"])["loan_amount"].agg(["count", "min", "max",
df4 = df4.sort_values(by="count", ascending=False)
df4 = df4[0:20]
regions = sns.barplot(x=df4['region'], y=df4['count'], ax=axs[1, 1])
regions.set_xticklabels(regions.get_xticklabels(), rotation=90)

plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```









First three sectors in termns of number of loans are Agriculture, Food, Retail, all with over 100,000 loans.

The first two activities are Farming and General Store, both with over 60,000 loans. Philippines is dominating the countries top, with over 150,000 loans, followed, at large distance, by Kenya and El Salvador. The top 3 identified regions are Kaduna, Lahore and Rawalpindi.

Most of the loans are registered under unknown region (we can suppose that most of them will go to Philippines, but we will have to verify this).

Loans per country and region

Let's represent now the number of loans aggregated per country and region, as a treemap. We will filter only the first 100 entries/country. To show more than 100 entries/country is not necessary, since it will became very difficult to visualize (and is also using a lot of resources).

df = kiva_loans_df.groupby(["country", "region"]) ["loan_amount"].agg(["count"]).reset_index() df = df.sort_values(by=["country", "count"], ascending=False) df = df.groupby(["country"]).head(100) df.columns = ["Country", "Region", "Number of loans"] fig = px.treemap(df, path=['Country', 'Region'], values='Number of loans', color='Number of loans', hover_data=['Country'], color_continuous_scale="Greens", color_continuous_midpoint=np.average(df['Number of loans'], weights=df['Number of loans'])) fig.update_layout(margin = dict(t=50, l=25, r=25, b=25), height=1000, title="Loans per country and region (top 100 regions per country)") fig.show()

In [ ]:
```
kiva_loans.head()
```

Out[ ]:

	id	funded_amount	loan_amount	activity	sector	use	cou
0	653051	300.0	300.0	Fruits & Vegetables	Food	To buy seasonal, fresh fruits to sell.	
1	653053	575.0	575.0	Rickshaw	Transportation	to repair and maintain the auto rickshaw used ...	
2	653068	150.0	150.0	Transportation	Transportation	To repair their old cycle-van and buy another ...	
3	653063	200.0	200.0	Embroidery	Arts	to purchase an embroidery machine and a variet...	
4	653084	400.0	400.0	Milk Sales	Food	to purchase one buffalo.	

In [ ]:
```
import plotly.express as px
df = kiva_loans.groupby(["country", "region"])["loan_amount"].agg(["count"]).res
df = df.sort_values(by=["country", "count"], ascending=False)
df = df.groupby(["country"]).head(100)


df.columns = ["Country", "Region", "Number of loans"]


fig = px.treemap(df, path=['Country', 'Region'],
values='Number of loans', color='Number of loans',
hover_data=['Country'], color_continuous_scale="Greens",
color_continuous_midpoint=np.average(df['Number of loans'],
weights=df['Number of loans']))
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25), height=1000,
title="Loans per country and region (top 100 regions per country)")
fig.show()
```

In [ ]:
```
df = kiva_loans.groupby(["country", "country_code"])["loan_amount"].agg(["count"
```

In [ ]:
```
import country_converter as cc
```

In [ ]:
```
df = kiva_loans.groupby(["country", "country_code"])["loan_amount"].agg(["count"
df['iso_alpha'] = df.country_code.apply(lambda x: cc.convert(x, to='ISO3'))
fig = px.choropleth(df,locations='iso_alpha', color='count',
                            color_continuous_scale="Greens",
                            scope="world",
                            labels={'count':'Number of loans'}
                           )
fig.update_layout(title='Kiva : Loans per Country')
fig.show()
```

Loans per sector ans activity

Let’re represent now the treemap for the couple {sectors, activities}, showing the size and color of tiles proportional with the number of loans.

In [ ]:
```
df = kiva_loans.groupby(["sector", "activity"])["loan_amount"].agg(["count"]).re
df = df.sort_values(by=["sector", "activity"], ascending=False)
df.columns = ["Sector", "Activity", "Number of loans"]
fig = px.treemap(df, path=["Sector", "Activity"],
values='Number of loans', color='Number of loans', hover_data=['Sector'],
color_continuous_scale="Greens",
color_continuous_midpoint=np.average(df['Number of loans'],
weights=df['Number of loans']))
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25),
height=1000, title="Loans per sector and activity")
fig.show()
```

Loans currencies

Let’s see now what are the currencies in which these loans are given. First, let's look for the currency used in each country.

In [ ]:
```
df = kiva_loans.groupby(["country", "currency"])["loan_amount"].agg(["count"]).r
df = df.sort_values(by=["country", "currency"], ascending=False)
df.columns = ["Country", "Currency", "Number of loans"]
fig = px.treemap(df, path=["Country", "Currency"],
values='Number of loans', color='Number of loans',
hover_data=['Country'], color_continuous_scale="Greens",
color_continuous_midpoint=np.average(df['Number of loans'],
weights=df['Number of loans']))
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25),
height=1000, title="Loans per country and currency")
fig.show()
```

Most of the loans are in PHP (for Philippines - which is natural, since Phillipines has most of the loans) and in USD (El Salvador, Ecuador, Palestine, Lebanon, United States, Cambodia, Nicaragua).

Let's check as well the distribution of loan amount for the loans in USD, the term in months and the repayment interval.

We start with the distribution of loan amount for the loans in USD, grouped by sectors. We will also remove upper 99% quantile.

In [ ]:
```
q99 = np.quantile(kiva_loans.loan_amount, 0.99)
df = kiva_loans.loc[(kiva_loans.currency=="USD") & (kiva_loans.loan_amount < q99
```

In [ ]:
```
df = kiva_loans.loc[kiva_loans.currency=="USD"].groupby(["country"])["loan_amoun
```

In [ ]:
```
df['iso_alpha'] = df.country.apply(lambda x: cc.convert(x, to='ISO3'))
fig = px.choropleth(df,locations='iso_alpha', color='loan_amount',
                           color_continuous_scale="Greens",
                           scope="world",
                           labels={'loan_amount':'Total loans amount'}
                          )
fig.update_layout(title='Kiva:Loans per country ')
fig.show()
```

```
Virgin Islands not found in regex
```

Loan repayment term

We follow with the distribution of the loan repayment term (in months), grouped by sectors.

In [ ]:
```
def draw_trace(dataset, sector, feature):
    dfS = dataset[dataset['sector']==sector]
    trace = go.Box( x = dfS[feature], name=sector, marker=dict( line=dict( color
    return trace
```

In [ ]:
```
def draw_group(dataset, feature, title, feature_title, sectors, height=800):
    data = []
    for sector in sectors:
        data.append(draw_trace(dataset, sector, feature))
```

In [ ]:
```
draw_group(kiva_loans, "term_in_months", "Loans repayment terms per sector", 'Lo
```

In [ ]:
```
data
```

Out[ ]: [('kiva_loans',

```
                id  funded_amount  loan_amount             activity  \
0        653051          300.0        300.0  Fruits & Vegetables
1        653053          575.0        575.0             Rickshaw
2        653068          150.0        150.0       Transportation
3        653063          200.0        200.0           Embroidery
4        653084          400.0        400.0           Milk Sales
...         ...            ...          ...                  ...
671200  1340323            0.0         25.0            Livestock
671201  1340316           25.0         25.0            Livestock
671202  1340334            0.0         25.0                Games
671203  1340338            0.0         25.0            Livestock
671204  1340339            0.0         25.0            Livestock

                sector                                                use  \
0                 Food            To buy seasonal, fresh fruits to sell.
1       Transportation  to repair and maintain the auto rickshaw used ...
2       Transportation  To repair their old cycle-van and buy another ...
3                 Arts  to purchase an embroidery machine and a variet...
4                 Food                            to purchase one buffalo.
...                ...                                                ...
671200     Agriculture  [True, u'para compara: cemento, arenya y ladri...
671201     Agriculture  [True, u'to start a turducken farm.'] - this l...
671202   Entertainment                                                NaN
671203     Agriculture  [True, u'to start a turducken farm.'] - this l...
671204     Agriculture  [True, u'to start a turducken farm.'] - this l...

       country_code   country        region currency  partner_id  \
0                PK  Pakistan        Lahore      PKR       247.0
1                PK  Pakistan        Lahore      PKR       247.0
2                IN     India     Maynaguri      INR       334.0
3                PK  Pakistan        Lahore      PKR       247.0
4                PK  Pakistan  Abdul Hakeem      PKR       245.0
...             ...       ...           ...      ...         ...
671200           PY  Paraguay    Concepción      USD        58.0
671201           KE     Kenya           NaN      KES       138.0
671202           KE     Kenya           NaN      KES       138.0
671203           KE     Kenya           NaN      KES       138.0
671204           KE     Kenya           NaN      KES       138.0

                      posted_time             disbursed_time  \
0       2014-01-01 06:12:39+00:00  2013-12-17 08:00:00+00:00
1       2014-01-01 06:51:08+00:00  2013-12-17 08:00:00+00:00
2       2014-01-01 09:58:07+00:00  2013-12-17 08:00:00+00:00
3       2014-01-01 08:03:11+00:00  2013-12-24 08:00:00+00:00
4       2014-01-01 11:53:19+00:00  2013-12-17 08:00:00+00:00
...                           ...                        ...
671200  2017-07-25 16:55:34+00:00  2017-07-25 07:00:00+00:00
671201  2017-07-25 06:14:08+00:00  2017-07-24 07:00:00+00:00
671202  2017-07-26 00:02:07+00:00  2017-07-25 07:00:00+00:00
671203  2017-07-26 06:12:55+00:00  2017-07-25 07:00:00+00:00
671204  2017-07-26 06:31:46+00:00  2017-07-25 07:00:00+00:00

                      funded_time  term_in_months  lender_count  \
0       2014-01-02 10:06:32+00:00            12.0            12
1       2014-01-02 09:17:23+00:00            11.0            14
2       2014-01-01 16:01:36+00:00            43.0             6
3       2014-01-01 13:00:00+00:00            11.0             8
4       2014-01-01 19:18:51+00:00            14.0            16
...                           ...             ...           ...
```

```
671200                        NaN            13.0             0
671201  2017-07-26 02:09:43+00:00            13.0             1
671202                        NaN            13.0             0
671203                        NaN            13.0             0
671204                        NaN            13.0             0

                                tags borrower_genders repayment_interval  \
0                                NaN           female          irregular
1                                NaN   female, female          irregular
2       user_favorite, user_favorite           female             bullet
3                                NaN           female          irregular
4                                NaN           female            monthly
...                              ...              ...                ...
671200                           NaN           female            monthly
671201                           NaN           female            monthly
671202                           NaN              NaN            monthly
671203                           NaN           female            monthly
671204                           NaN           female            monthly

              date
0       2014-01-01
1       2014-01-01
2       2014-01-01
3       2014-01-01
4       2014-01-01
...            ...
671200  2017-07-25
671201  2017-07-25
671202  2017-07-26
671203  2017-07-26
671204  2017-07-26

[671205 rows x 20 columns]),
('kiva_location',
                 LocationName  ISO      country      region world_region  \
0     Badakhshan, Afghanistan  AFG  Afghanistan  Badakhshan   South Asia
1        Badghis, Afghanistan  AFG  Afghanistan     Badghis   South Asia
2        Baghlan, Afghanistan  AFG  Afghanistan     Baghlan   South Asia
3          Balkh, Afghanistan  AFG  Afghanistan       Balkh   South Asia
4         Bamyan, Afghanistan  AFG  Afghanistan      Bamyan   South Asia
...                       ...  ...          ...         ...          ...
2767                      NaN  NaN          NaN         NaN          NaN
2768                      NaN  NaN          NaN         NaN          NaN
2769                      NaN  NaN          NaN         NaN          NaN
2770                      NaN  NaN          NaN         NaN          NaN
2771                      NaN  NaN          NaN         NaN          NaN

        MPI                              geo        lat        lon
0     0.387  (36.7347725, 70.81199529999999)  36.734772  70.811995
1     0.466         (35.1671339, 63.7695384)  35.167134  63.769538
2     0.300         (35.8042947, 69.2877535)  35.804295  69.287754
3     0.301         (36.7550603, 66.8975372)  36.755060  66.897537
4     0.325         (34.8100067, 67.8212104)  34.810007  67.821210
...     ...                              ...        ...        ...
2767    NaN                 (1000.0, 1000.0)        NaN        NaN
2768    NaN                 (1000.0, 1000.0)        NaN        NaN
2769    NaN                 (1000.0, 1000.0)        NaN        NaN
2770    NaN                 (1000.0, 1000.0)        NaN        NaN
2771    NaN                 (1000.0, 1000.0)        NaN        NaN
```

```
 [2772 rows x 9 columns]),
('loan_theme',
             id    Loan Theme ID          Loan Theme Type  Partner ID
0        638631  a1050000000skGl                  General       151.0
1        640322  a1050000000skGl                  General       151.0
2        641006  a1050000002X1ij         Higher Education       160.0
3        641019  a1050000002X1ij         Higher Education       160.0
4        641594  a1050000002VbsW  Subsistence Agriculture       336.0
...         ...              ...                      ...         ...
779087  1444237  a1050000000wf0h                  General       136.0
779088  1444238  a1050000000wf0h                  General       136.0
779089  1444240  a1050000000wf0h                  General       136.0
779090  1444241  a1050000000wf22                  General       245.0
779091  1444243  a1050000000wf22                  General       245.0

 [779092 rows x 4 columns]),
('loan_themes_by_region',
       Partner ID               Field Partner Name  \
0               9  KREDIT Microfinance Institution
1               9  KREDIT Microfinance Institution
2               9  KREDIT Microfinance Institution
3               9  KREDIT Microfinance Institution
4               9  KREDIT Microfinance Institution
...           ...                              ...
15731         539           N/A, direct to Sevamob
15732         540           N/A, direct to Sanergy
15733         542       N/A direct to BioLite Inc.
15734         543      N/A direct to LegWorks Inc.
15735         545        N/A, direct to Solar Home

                            sector    Loan Theme ID         Loan Theme Type
\
0      General Financial Inclusion  a1050000000slfi        Higher Education
1      General Financial Inclusion  a10500000068jPe  Vulnerable Populations
2      General Financial Inclusion  a1050000000slfi        Higher Education
3      General Financial Inclusion  a10500000068jPe  Vulnerable Populations
4      General Financial Inclusion  a1050000002X1Uu              Sanitation
...                            ...              ...                     ...
15731                  DSE Direct  a10500000081IYl                  Health
15732                  DSE Direct  a10500000081HIw    Water and Sanitation
15733                  DSE Direct  a1050000008AQrj         Clean Cookstove
15734                  DSE Direct  a1050000008AQwt                  Health
15735                  DSE Direct  a1050000008ASxg      Solar Home Systems

               country forkiva                 region  \
0             Cambodia      No       Banteay Meanchey
1             Cambodia      No    Battambang Province
2             Cambodia      No    Battambang Province
3             Cambodia      No  Kampong Cham Province
4             Cambodia      No  Kampong Cham Province
...                ...     ...                     ...
15731     South Africa      No                Nkangala
15732            Kenya      No                 Nairobi
15733           Uganda      No                 Kampala
15734           Canada      No                 Toronto
15735  Myanmar (Burma)      No                  Yangon

                    geocode_old  ISO  ...  amount  \
0                (13.75, 103.0)  KHM  ...     450
1                           NaN  KHM  ...   20275
```

```
2                          NaN  KHM  ...    9150
3                (12.0, 105.5)  KHM  ...  604950
4                (12.0, 105.5)  KHM  ...     275
...                        ...  ...  ...     ...
15731                      NaN  ZAF  ...   50000
15732  (-1.2833333, 36.8166667)  KEN  ...   50000
15733   (0.3155556, 32.5655556)  UGA  ...   50000
15734                      NaN  CAN  ...   50000
15735                      NaN  MMR  ...   50000

                         LocationName                           geocode  \
0           Banteay Meanchey, Cambodia        [(13.6672596, 102.8975098)]
1        Battambang Province, Cambodia         [(13.0286971, 102.989615)]
2        Battambang Province, Cambodia         [(13.0286971, 102.989615)]
3      Kampong Cham Province, Cambodia        [(12.0982918, 105.3131185)]
4      Kampong Cham Province, Cambodia        [(12.0982918, 105.3131185)]
...                                ...                                ...
15731           Nkangala, South Africa        [(-25.9459898, 29.6035495)]
15732                   Nairobi, Kenya         [(-1.2920659, 36.8219462)]
15733                  Kampala, Uganda          [(0.3475964, 32.5825197)]
15734                  Toronto, Canada         [(43.653226, -79.3831843)]
15735          Yangon, Myanmar (Burma)  [(16.8660694, 96.19513200000002)]

                                          names  \
0             Banteay Meanchey Province; Cambodia
1                   Battambang Province; Cambodia
2                   Battambang Province; Cambodia
3                 Kampong Cham Province; Cambodia
4                 Kampong Cham Province; Cambodia
...                                           ...
15731          Nkangala; Mpumalanga; South Africa
15732              Nairobi; Nairobi County; Kenya
15733    Kampala; Kampala; Central Region; Uganda
15734  Toronto; Toronto Division; Ontario; Canada
15735      Yangon; Yangon Region; Myanmar (Burma)

                                geo        lat         lon  \
0            (13.6672596, 102.8975098)  13.667260  102.897510
1             (13.0286971, 102.989615)  13.028697  102.989615
2             (13.0286971, 102.989615)  13.028697  102.989615
3            (12.0982918, 105.3131185)  12.098292  105.313119
4            (12.0982918, 105.3131185)  12.098292  105.313119
...                                ...        ...         ...
15731        (-25.9459898, 29.6035495) -25.945990   29.603549
15732         (-1.2920659, 36.8219462)  -1.292066   36.821946
15733          (0.3475964, 32.5825197)   0.347596   32.582520
15734         (43.653226, -79.3831843)  43.653226  -79.383184
15735  (16.8660694, 96.19513200000002)  16.866069   96.195132

                       mpi_region                         mpi_geo rural_pct
0      Banteay Mean Chey, Cambodia        (13.6672596, 102.8975098)      90.0
1      Banteay Mean Chey, Cambodia        (13.6672596, 102.8975098)      90.0
2      Banteay Mean Chey, Cambodia        (13.6672596, 102.8975098)      90.0
3           Kampong Cham, Cambodia        (11.9924294, 105.4645408)      90.0
4           Kampong Cham, Cambodia        (11.9924294, 105.4645408)      90.0
...                            ...                              ...       ...
15731                         ZAF                              NaN       NaN
15732               Nairobi, Kenya         (-1.2920659, 36.8219462)       NaN
15733              Kampala, Uganda          (0.3475964, 32.5825197)       NaN
15734                         CAN                              NaN       NaN
```

```
15735              Yangon, Myanmar  (16.8660694, 96.19513200000002)       NaN

[15736 rows x 21 columns])]
```

Loan repayment interval

We show here the number of loans for each Sector, grouped by repayment interval (monthly, irregular, weekly, bullet).

In [ ]:
```
repayment_interval = kiva_loans.repayment_interval.unique()
repayment_interval
```

Out[ ]:
```
array(['irregular', 'bullet', 'monthly', 'weekly'], dtype=object)
```

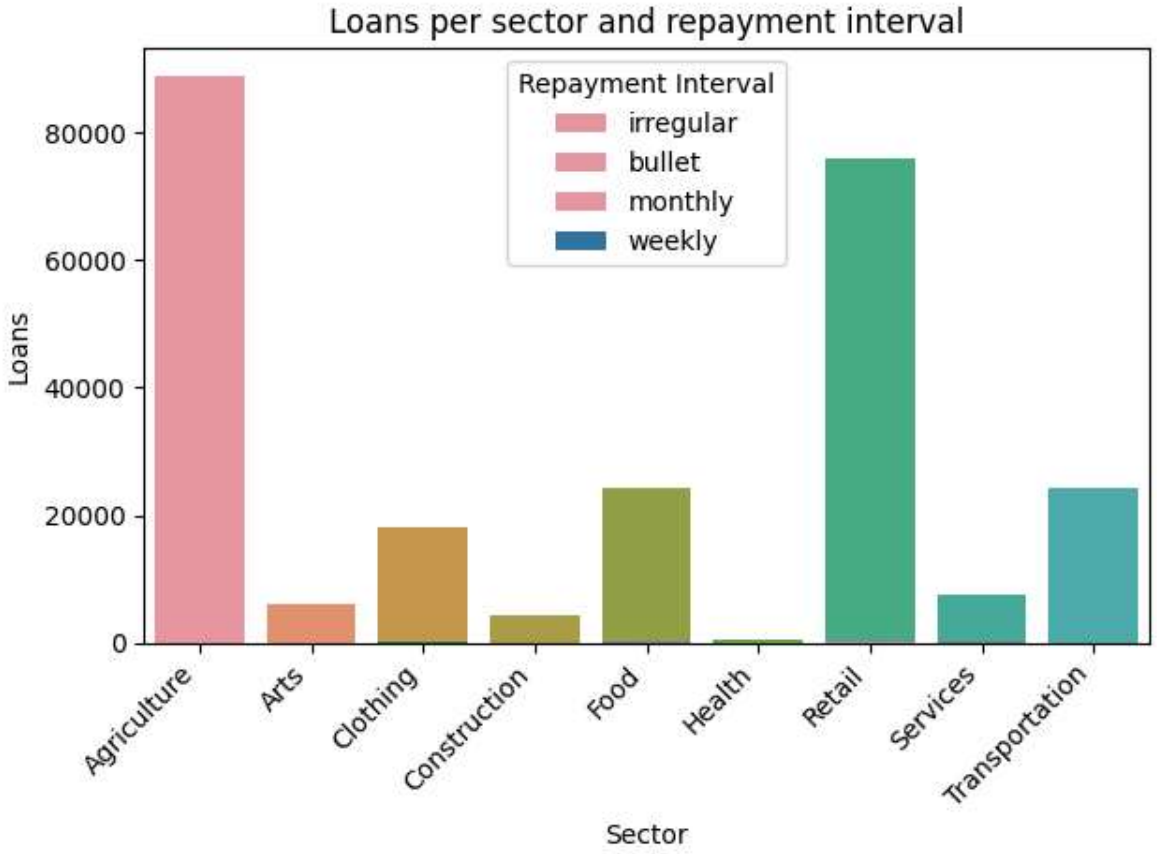
In [ ]:
```
kiva_loans.head()
```

Out[ ]:

	id	funded_amount	loan_amount	activity	sector	use	cou
0	653051	300.0	300.0	Fruits & Vegetables	Food	To buy seasonal, fresh fruits to sell.	
1	653053	575.0	575.0	Rickshaw	Transportation	to repair and maintain the auto rickshaw used ...	
2	653068	150.0	150.0	Transportation	Transportation	To repair their old cycle-van and buy another ...	
3	653063	200.0	200.0	Embroidery	Arts	to purchase an embroidery machine and a variet...	
4	653084	400.0	400.0	Milk Sales	Food	to purchase one buffalo.	

In [ ]:
```
# Initialize an empty list to store Seaborn bar plots
plots = []

# Iterate over each repayment interval
for i, ry in enumerate(repayment_interval):
    # Filter the DataFrame for the current repayment interval
    df = kiva_loans.loc[kiva_loans.repayment_interval == ry]
    # Group by sector and count the number of loans
```

```
    df = df.groupby("sector")["loan_amount"].count().reset_index()
    df.columns = ["sector", "loans"]

    # Create a bar plot for the current repayment interval and append it to the
    plot = sns.barplot(data=df, x="sector", y="loans", label=ry)
    plots.append(plot)

# Set title and axis labels
plt.title("Loans per sector and repayment interval")
plt.xlabel("Sector")
plt.ylabel("Loans")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()

# Show the legend
plt.legend(title="Repayment Interval")

# Show the plot
plt.show()
```



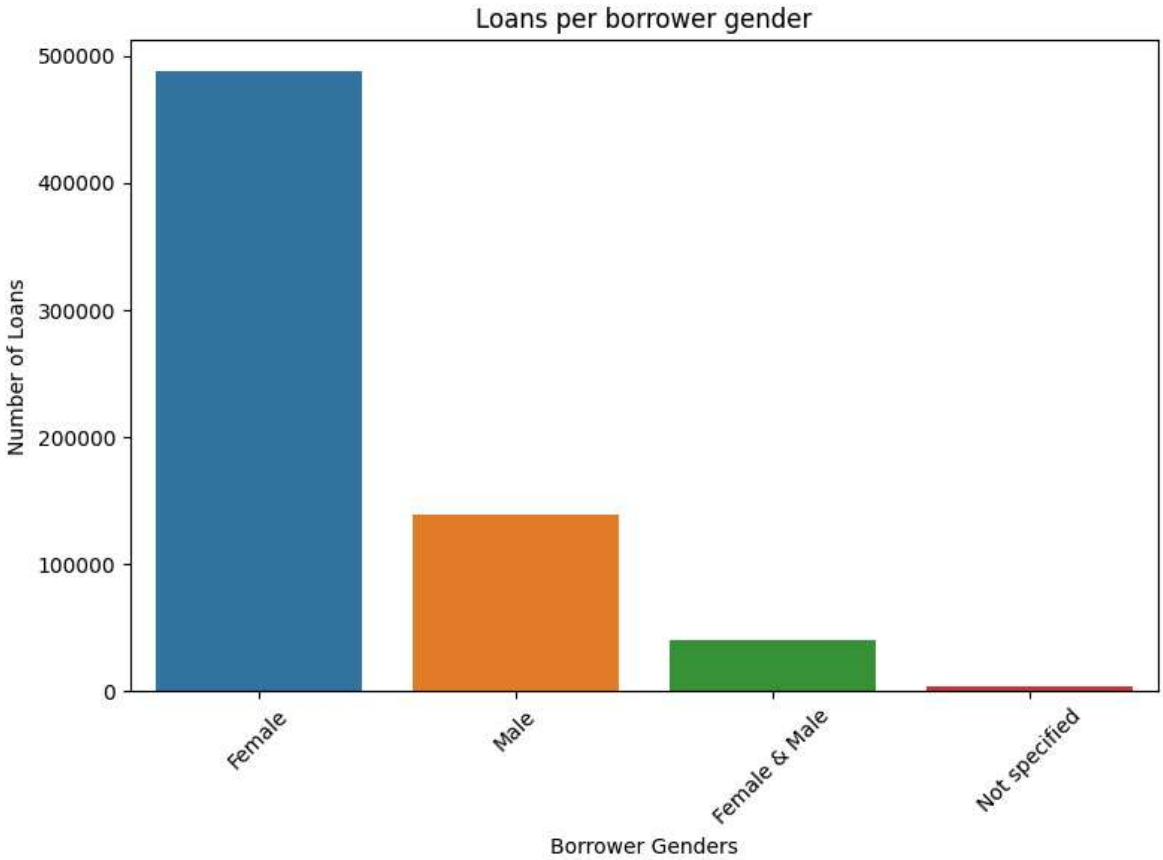
Loan borowers

In [ ]:
```
def get_gender_count(text,gen="female"):
    count = 0
    try:
        text = text.split(",")
        for t in text:
            if gen == t.rstrip().lstrip():
                count+=1
    except:
        return count
```

```
    return count
kiva_loans["n_male"] = kiva_loans["borrower_genders"].apply(lambda x: get_gender
kiva_loans["n_female"] = kiva_loans["borrower_genders"].apply(lambda x: get_gend
```

In [ ]:

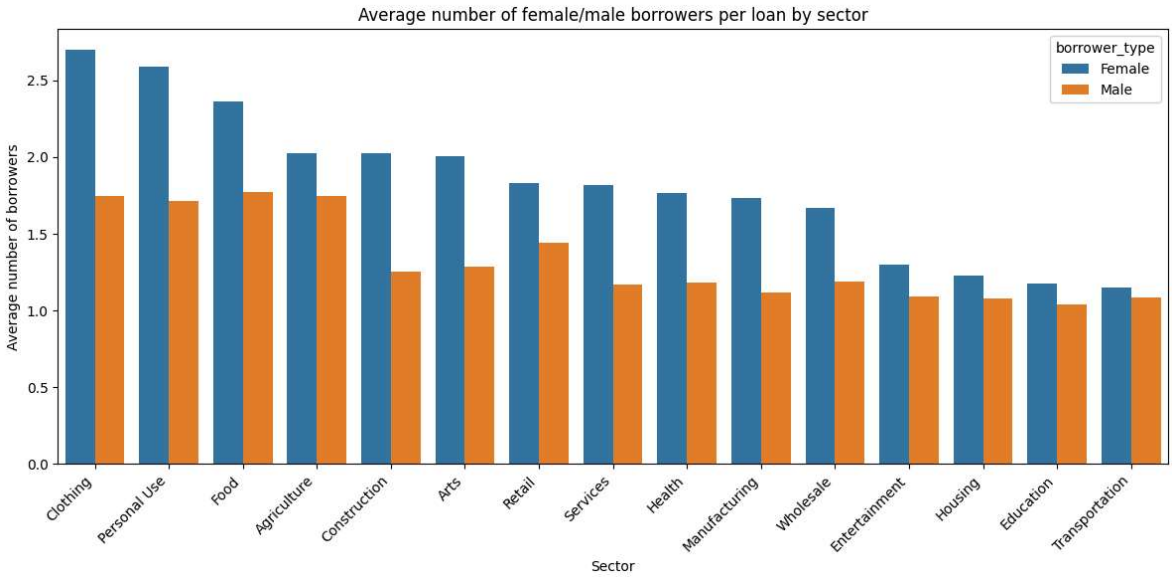
```
# Assign borrower gender based on number of male and female borrowers
kiva_loans["borrower_gen"] = "Not specified"
kiva_loans.loc[(kiva_loans.n_male != 0) & (kiva_loans.n_female == 0), "borrower_
kiva_loans.loc[(kiva_loans.n_male == 0) & (kiva_loans.n_female != 0), "borrower_
kiva_loans.loc[(kiva_loans.n_male != 0) & (kiva_loans.n_female != 0), "borrower_

# Group by borrower gender and count number of loans
df = kiva_loans.groupby("borrower_gen")["loan_amount"].count().reset_index()
df.columns = ["Borrower Gender", "Number of loans"]
df = df.sort_values(by="Number of loans", ascending=False)

# Create bar plot using Seaborn
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x="Borrower Gender", y="Number of loans")
plt.title("Loans per borrower gender")
plt.xlabel("Borrower Genders")
plt.ylabel("Number of Loans")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Filter loans with at least one female borrower
df_female = kiva_loans.loc[kiva_loans.n_female > 0]
# Group by sector and calculate count, mean, and max for female borrowers
df_female = df_female.groupby("sector")["n_female"].agg(["count", "mean", "max"]
# Sort values by mean number of female borrowers
```

```
df_female = df_female.sort_values(by="mean", ascending=False)

# Filter loans with at least one male borrower
df_male = kiva_loans.loc[kiva_loans.n_male > 0]
# Group by sector and calculate count, mean, and max for male borrowers
df_male = df_male.groupby("sector")["n_male"].agg(["count", "mean", "max"]).rese
# Sort values by mean number of male borrowers
df_male = df_male.sort_values(by="mean", ascending=False)

# Concatenate the two DataFrames
df_concat = pd.concat([df_female.assign(borrower_type="Female"), df_male.assign(

# Create the subplot with facet grids using Seaborn
plt.figure(figsize=(12, 6))
sns.barplot(data=df_concat, x="sector", y="mean", hue="borrower_type")
plt.title("Average number of female/male borrowers per loan by sector")
plt.xlabel("Sector")
plt.ylabel("Average number of borrowers")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```



In [ ]: