Pandas gym

Prepared as part of MOD510 Computational Engineering and Modeling

Aug 19, 2022

Learning objectives:

1. number one
2. number two

1 Exercise 1 Create a data frame

In the folder `data` there is a file named `field_production_monthly.csv`.

1. Read this file into a Pandas DataFrame
2. Make a function that take as input a name of a file, and returns an error message if it fails to open it

SOLUTION: If the csv file is located in the same folder as your python program

```
import pandas as pd
df=pd.read_csv('field_production_monthly.csv', sep=',')
```

Next, we make a function

```
def read_data_frame(file_name,sep=','):
    try:
        df=pd.read_csv(file_name,sep=sep)
        return df
    except:
        print('Could not open file: ', file_name)
        return pd.DataFrame()

df=read_data_frame('field_production_monthly.csv')
```

There are a couple of things to note here, first it is the try except statement. This basically tells Python to try to execute the `pd.read_csv()` command, if this fail it jumps to the except statement and execute that. This is an extremely elegant way to make your code able to run without stopping and also write out your own error messages instead of the usual error messages generated from Python. Next, it is a good practice to return an object of the same type, thus here we returns an empty DataFrame. Note that in the function we have also used `sep=','` as the default argument.

2 Exercise 2 Extract data for Snorre

The file `field_production_gross_monthly.csv` contains production data from the Norwegian Petroleum Directorate factpages[1].

1. Extract a DataFrame for the Snorre field
2. Plot the production of oil equivalents as a function of time
   - Use Matplotlib, and
   - the built in plotting functionality of Pandas
3. Compare your plot with the one on the factpages[2]. Use the `pd.groupby()`[3] function to make a new data frame that contains the production per year and see if you can reproduce figure 1 from the factpages.

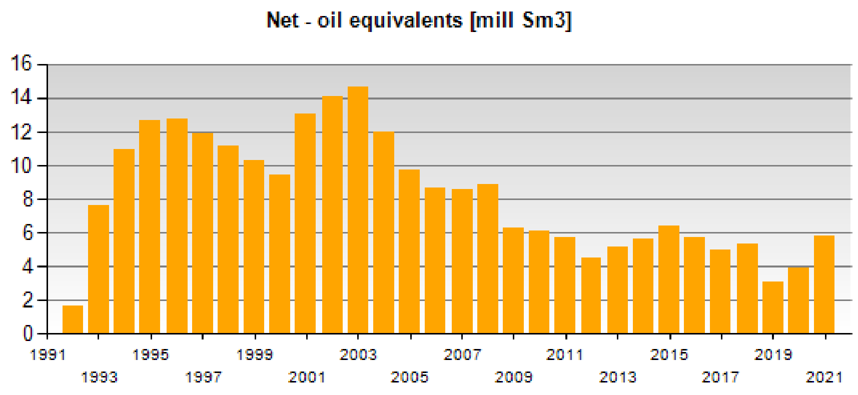

Figure 1: Yearly production from the Snorre field.

[1]`https://factpages.npd.no/`
[2]`https://factpages.npd.no/en/field/PageView/All/43718`
[3]`https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html`

```
import matplotlib.pyplot as plt

df_snorre=df[df['prfInformationCarrier']=='SNORRE']
x=df_snorre['prfYear']+df_snorre['prfMonth']/12
y=df_snorre['prfPrdOeNetMillSm3']

plt.plot(x,y)
```

To use the built in Pandas plotting, we have to make a new column that contains the x-values

```
df_snorre['time']=df_snorre['prfYear']+df_snorre['prfMonth']/12 # make new column
df_snorre.plot(x='time',y='prfPrdOeNetMillSm3')
```

Next, we want to make a plot similar as the one in the factpages (see figure 1). What we want to do is to sum the production of all the columns that has the same year, this can be done quite elegantly by the `pd.groupby.sum()` function

```
df_snorre_year=df_snorre.groupby('prfYear').sum()
print(df_snorre)
```

Note that the `prfYear` column is now missing, it has automatically been set by pandas as the index column, to plot it we need to add it

```
df_snorre_year['prfYear']=df_snorre_year.index
df_snorre_year.plot(x='prfYear',y='prfPrdOeNetMillSm3')
# or a barplot
df_snorre_year.plot.bar(x='prfYear',y='prfPrdOeNetMillSm3')
```

3 Exercise 3 Extract data for any field

Now we want to write some functions that are more general, which can extract information from any field.

1. Make a general function that can extract a DataFrame given any name of a field in the database. If you want to be fancy you could also make the function case insensitive. The function should write a reasonable error message if something went wrong
2. Write a function that takes as argument a DataFrame for a given field and makes a plot of the monthly production of oil equivalents. Give the plot a reasonable title and axes labels

Solutions:

```
df_full=pd.read_csv('field_production_monthly.csv',sep=',')
```

```
def get_field_data_frame(field_name,df=df_full):
    '''
    Returns a dataframe given a field name,
    returns empty dataframe if field does not exist
    '''
    FIELD_NAME=field_name.upper()
    all_fields=df['prfInformationCarrier'].unique()
    if FIELD_NAME in all_fields:
        return df[df['prfInformationCarrier']==FIELD_NAME]
    else:
        print('Field name ', field_name, ' does not exists')
        print('Available field names are: ')
        print(all_fields)
        return pd.DataFrame()
df=get_field_data_frame('snore') # Wrong name
df=get_field_data_frame('snorRe')

def plot_field_prod_data(df,y='prfPrdOeNetMillSm3'):
    name=df['prfInformationCarrier'].iloc[0]
    df['time']=df['prfYear']+df['prfMonth']/12
    df.plot(x='time',y=y,title=name,grid=True,xlabel='Years',ylabel=r'Production per month [mil

plot_field_prod_data(df)
```

If you are having problem understanding the code above, you should set `df=df_full`, and `field_name='snorre`, and then execute each line in a jupyter notebook. Note that we send in the full DataFrame to the function, this is important if the database is large, you do not want read the excel or csv file over and over each time you call the function - this significantly slows down your code!

1. Boolean indexing - Using a single column's values to select data

```
df[df["A"] > 0]
```

1. Selecting values from a DataFrame where a boolean condition is met

```
df[df > 0]
```

1. Using the `isin()` method for filtering:

`df2["E"] = ["one", "one", "two", "three", "four", "three"] df2[df2["E"].isin(["two", "four"])]`

1. Grouping

By “group by” we are referring to a process involving one or more of the following steps:

1. Splitting the data into groups based on some criteria
2. Applying a function to each group independently
3. Combining the results into a data structure