Pandas gym

Prepared as part of Python for Subsurface Applications and Management

Aksel Hiorth University of Stavanger

Sep 5, 2022

Learning objectives:

1. how to create a DataFrame
2. plot data in DataFrame
3. write code that is robust, i.e. gives error messages when it fails
4. do simple statistics on data in a DataFrame
5. group and filter data
6. work with files and folders, split data into different files

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 reads a file into a DataFrame, and returns an error message if it fails to open the file

```
df = pd.read_csv(..)

def read_data_frame(file_name,sep=','):
 ... write function
```

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

```
df=pd.read_csv('../data/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('../data/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.

```
# fill inn code
```

[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`

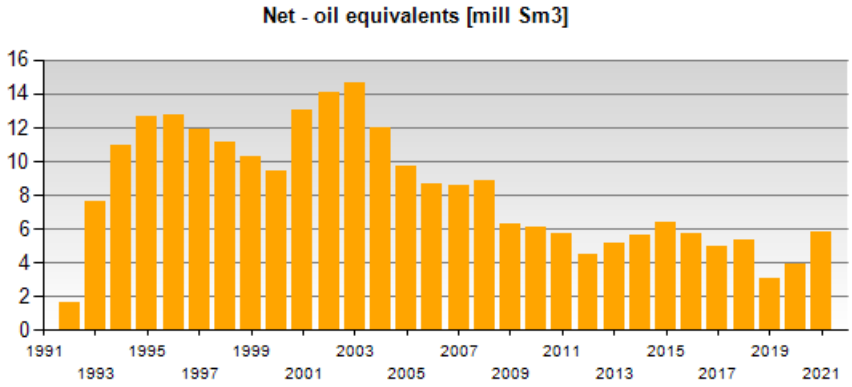

Figure 1: Yearly production from the Snorre field.

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

```
df_full=pd.read_csv('../data/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
    '''
    #... fill in code
```

SOLUTION:

```
df_full=pd.read_csv('../data/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!

4 Exercise 4 Plot the total production data for NCS

1. Use the `pd.groupby()`[4] functionality in Pandas to create a DataFrame containing production data for NCS as a whole
2. Make a plot of the production

SOLUTION:

```
# create data frame
df_ncs=df_full.groupby('prfYear').sum()

# plot the data
df_ncs.plot(y=df_full.columns[3:9],grid=True)
# alternatively
df_ncs.plot(y=df_full.columns[3:9],grid=True,subplots=True)
```

[4] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

5 Exercise 5 Split data into folders and files

Create a new data directory in which you create one folder for each field that contains an excel sheet with production data for that field. Use the `Pathlib` library to create directories.

Special characters in names.

Here you will encounter names with special characters, it is usually a good idea to replace those characters with a suitable replacement, before creating names or directories. To help you, you can use the function `replace_chars` below.

```
# fill in code
from pathlib import Path
def replace_chars(name, chars=["/"," ", "Å", "Ø", "Æ"], new_chars=["_","_","AA","O","AE"]):
    ''' replace Norwegian characters, space and slash in names'''
    new_name = name
    for ch,nch in zip(chars,new_chars):
        new_name = new_name.replace(ch, nch)
    return new_name
```

SOLUTION:

```
from pathlib import Path
col='prfInformationCarrier'
fields = df_full[col].unique() #skip duplicates
data_folder=Path('../production-data')
data_folder.mkdir(exist_ok=True)
for field in fields:
    new_name=replace_chars(field)
    new_path=data_folder / new_name
    new_path.mkdir(exist_ok=True)
    excel_file=new_name+'.xlsx'
    df2=df_full[df_full[col]==field]
    df2.to_excel(new_path/excel_file,index=False)
```

6 Exercise 6 Combine DataFrames

Write a code to collect all the excel files you stored in different folder into a single DataFrame (Hint: use `concat()`[5] to combine data)

```
df_new=pd.DataFrame()
```

[5] https://pandas.pydata.org/docs/reference/api/pandas.concat.html

SOLUTION:

```
df_new=pd.DataFrame()
data_folder=Path('../production-data')
for dir in data_folder.iterdir():
    if dir.is_dir():
        file=dir.name+'.xlsx'
        try:
            df=pd.read_excel(dir/file)
            print('Reading file ', file)
            df_new=pd.concat([df_new,df],ignore_index=True)
        except:
            print('No data in folder ', dir.name)
```

7 Exercise 7 Scrap data from the web

In this exercise we will collect data from the web, note that this will require some additional checking if the data you have read is of the correct type. Note that you can always do `df.dtypes` to list the types in the DataFrame.

1. Use the function `pandas.read_html()`[6] to scrap production data from the Snorre field directly from the NPD factpages[7]. (Hint: `pandas.read_html()` returns a list containing all tables in a website as DataFrames)
2. Make a plot of the production data and compare with the production data in figure 1. (Hint: you might need to sort values in the DataFrame and to convert some values to the correct type)

```
# enter code here
```

SOLUTION:

```
df_lists=pd.read_html('https://factpages.npd.no/en/field/PageView/All/43718')
df_prod=df_lists[14] # production data are in table no 14
```

Next, we need to drop the first column

```
df_prod2=df_prod.drop([0])
print(df_prod2)
print(df_prod2.dtypes)
```

There are two challenges (at least when I did this) 1) One column has been named `Unnamed: 0`, this is the column with the year data 2) the year data are read in as string (or `dtype: object`) and we need to convert to integer or float

[6] `https://pandas.pydata.org/docs/reference/api/pandas.read_html.html`
[7] `https://factpages.npd.no/en/field/PageView/All/43718`

```
df_prod2=df_prod2.rename(columns={df_prod2.columns[0]:"Year"})
df_prod2['Year']=pd.to_numeric(df_prod2['Year'])
```

Next, we need to sort the data if we want to plot them in historical correct order

```
df_prod2=df_prod2.set_index("Year")
df_prod2=df_prod2.sort_index()
df_prod2
```

Finally, we can do the plotting

```
df_prod2.plot(y=df_prod2.columns[1:],grid=True)
df_prod2.plot.bar(y=df_prod2.columns[-1])
```