data_exploration

March 6, 2021

[1]:
```python
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_profiling
np.random.seed(15)
```

[2]:
```python
# Loading data
data = pd.read_csv("forest_fires_dataset.csv")
attributes = pd.read_csv("attributes_forest_fires.csv")
```

[3]:
```python
attributes
```

[3]:

	name	type	description
0	X	integer	x-axis spatial coordinate within the Montesinh…
1	Y	integer	y-axis spatial coordinate within the Montesinh…
2	month	string	month of the year: 'jan' to 'dec'
3	day	string	day of the week: 'mon' to 'sun'
4	FFMC	float	FFMC index from the FWI system: 18.7 to 96.20
5	DMC	float	DMC index from the FWI system: 1.1 to 291.3
6	DC	float	DC index from the FWI system: 7.9 to 860.6
7	ISI	float	ISI index from the FWI system: 0.0 to 56.10
8	temp	float	temperature in Celsius degrees: 2.2 to 33.30
9	RH	float	relative humidity in %: 15.0 to 100
10	wind	float	wind speed in km/h: 0.40 to 9.40
11	rain	float	outside rain in mm/m2 : 0.0 to 6.4
12	area	float	the burned area of the forest (in ha): 0.00 to…

As we do not have many features in our dataset I decided to understand ambiguous ones before ongoing analysis.

FFMC, The Fine Fuel Moisture Code represents fuel moisture of forest litter fuels under the shade of a forest canopy. It is intended to represent moisture conditions for shaded litter fuels, the equivalent of 16-hour timelag. It ranges from 0-101.

DMC, The Duff Moisture Code represents fuel moisture of decomposed organic material underneath the litter. System designers suggest that it is represents moisture conditions for the equivalent of

15-day (or 360 hr) timelag fuels. It is unitless and open ended. It may provide insight to live fuel moisture stress.

DC, The Drought Code much like the Keetch-Byrum Drought Index, represents drying deep into the soil. It approximates moisture conditions for the equivalent of 53-day (1272 hour) timelag fuels. It is unitless, with a maximum value of 1000. Extreme drought conditions have produced DC values near 800.

ISI,The Initial Spread Index is analogous to the NFDRS Spread Component (SC). It integrates fuel moisture for fine dead fuels and surface windspeed to estimate a spread potential. ISI is a key input for fire behavior predictions in the FBP system. It is unitless and open ended.

Bigger values of indices means that forest is dryer

Let's take a first look what we have in our dataset

[5]:
```
data.shape
```

[5]:
```
(517, 13)
```

[6]:
```
data.info()
data.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X       517 non-null    int64
 1   Y       517 non-null    int64
 2   month   517 non-null    object
 3   day     517 non-null    object
 4   FFMC    517 non-null    float64
 5   DMC     517 non-null    float64
 6   DC      517 non-null    float64
 7   ISI     517 non-null    float64
 8   temp    517 non-null    float64
 9   RH      517 non-null    float64
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(9), int64(2), object(2)
memory usage: 52.6+ KB
```

[6]:
```
Index(['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain', 'area'],
      dtype='object')
```

[7]:
```
data.describe()
```

[7]:

```
               X           Y        FFMC         DMC          DC         ISI  \
count  517.000000  517.000000  517.000000  517.000000  517.000000  517.000000
mean     4.669246    4.299807   90.644681  110.872340  547.940039    9.021663
std      2.313778    1.229900    5.520111   64.046482  248.066192    4.559477
min      1.000000    2.000000   18.700000    1.100000    7.900000    0.000000
25%      3.000000    4.000000   90.200000   68.600000  437.700000    6.500000
50%      4.000000    4.000000   91.600000  108.300000  664.200000    8.400000
75%      7.000000    5.000000   92.900000  142.400000  713.900000   10.800000
max      9.000000    9.000000   96.200000  291.300000  860.600000   56.100000

            temp          RH        wind        rain         area
count  517.000000  517.000000  517.000000  517.000000   517.000000
mean    18.889168   44.288201    4.017602    0.021663    12.847292
std      5.806625   16.317469    1.791653    0.295959    63.655818
min      2.200000   15.000000    0.400000    0.000000     0.000000
25%     15.500000   33.000000    2.700000    0.000000     0.000000
50%     19.300000   42.000000    4.000000    0.000000     0.520000
75%     22.800000   53.000000    4.900000    0.000000     6.570000
max     33.300000  100.000000    9.400000    6.400000  1090.840000
```

[39]:

```python
data["area"].value_counts()
```

[39]:

```
0.00     247
1.94       3
28.66      2
0.52       2
9.96       2
        …
2.21       1
7.36       1
0.24       1
6.84       1
35.88      1
Name: area, Length: 251, dtype: int64
```

[9]:

```python
data["rain"].value_counts()
```

[9]:

```
0.0    509
0.8      2
0.2      2
0.4      1
1.4      1
6.4      1
1.0      1
Name: rain, dtype: int64
```

Observations:

We don't have any missing values, so we don't have to bother with any missing value treatment :)

Month and days columns are represented as strings. It shall be changed for numerical values.

99 percent of 'rain' values are 0 . it could be hard to imagine fire when it rains

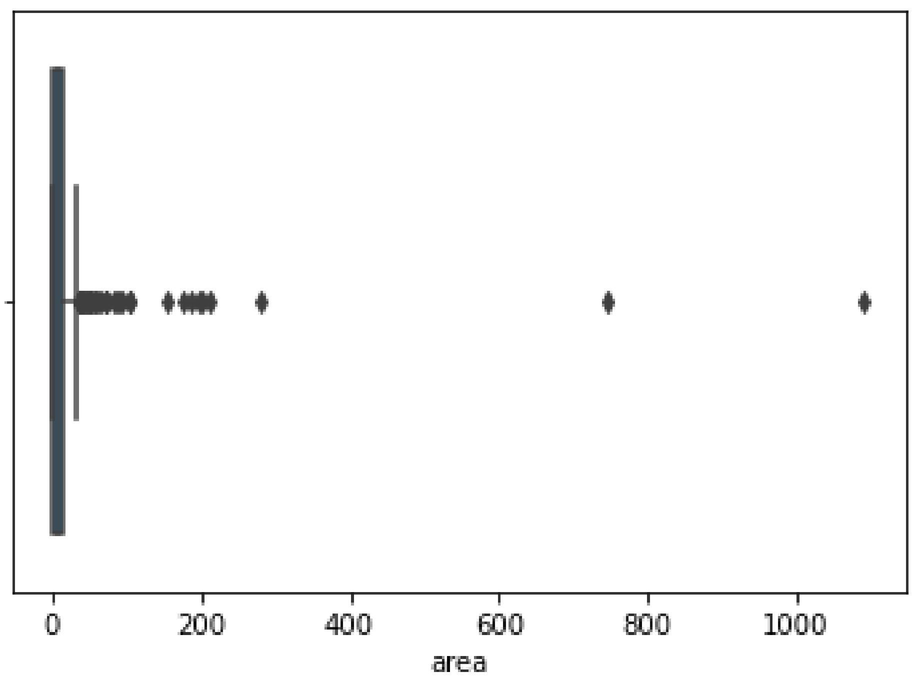
Half of observations depict situation when there was no fire in a forest. (I don't know if I should throw them away or what to do with them)

Let's analyse only these situations where there was a fire (area > 0)

[59]:
```
fires = data[data.area > 0]
```

[42]:
```
sns.boxplot(data = fires, x = "area")
```

[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b6dbab6888>
```



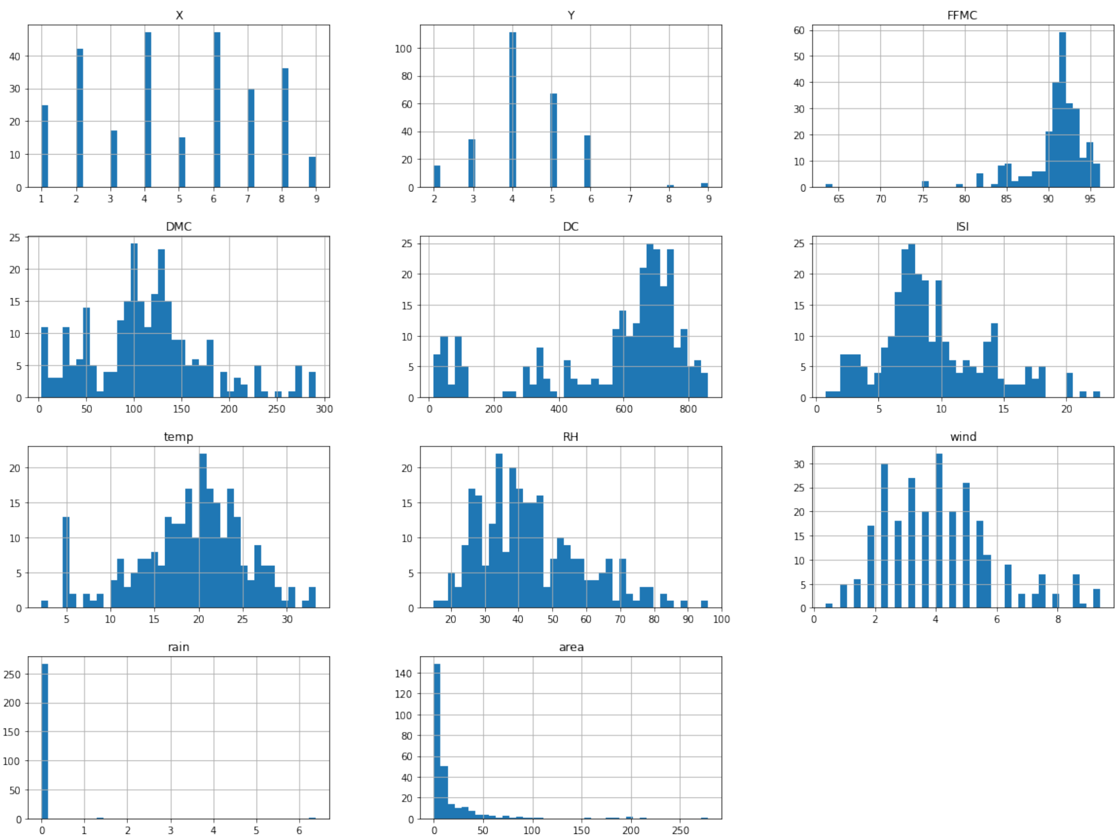
Let's get rid of outliers

[60]:
```
fires_without_outliers = fires[fires.area < 600]
```

[61]:
```
fires_without_outliers.hist(bins = 40,figsize=(20,15))
```

[61]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DD3122C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DD345F88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFCB51C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFCEAD08>,
```

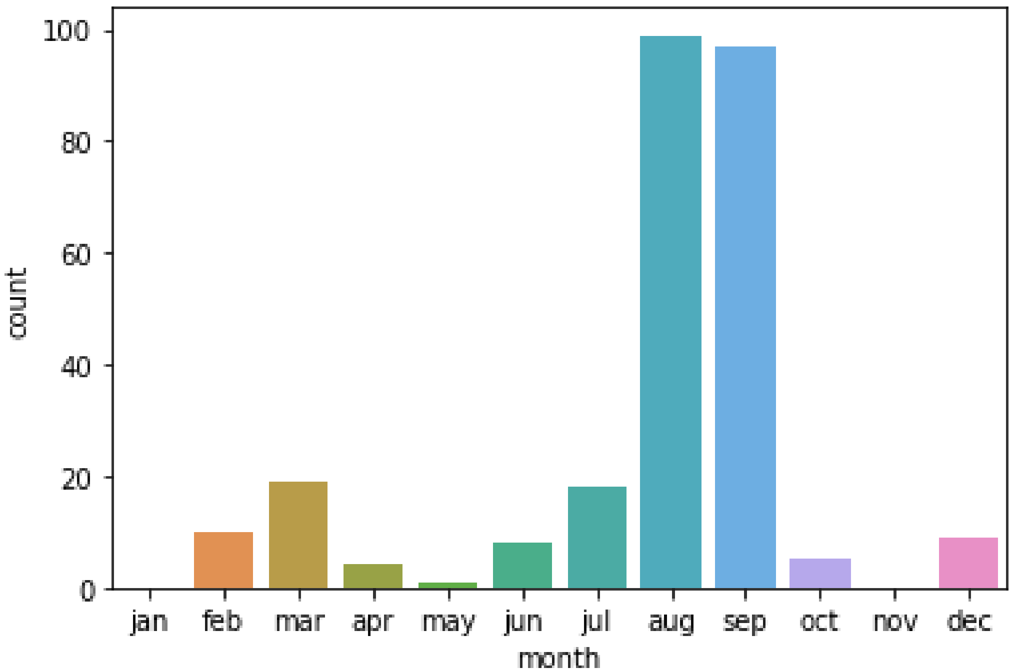
```
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFD23A48>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFD5D7C8>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFD97588>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFDD1208>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFDD1408>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFE0B248>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6DFE77B88>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001B6E0BF0888>]],
dtype=object)
```



When we get to the north part of Park, there are no fires. Higher indices implies higher amount of fires. We can see that that most of forest fires were really small.

[62]:
```python
months =␣
↪["jan","feb","mar","apr","may","jun","jul","aug","sep","oct","nov","dec"]
sns.countplot(data = fires, x = "month", order = months )
```

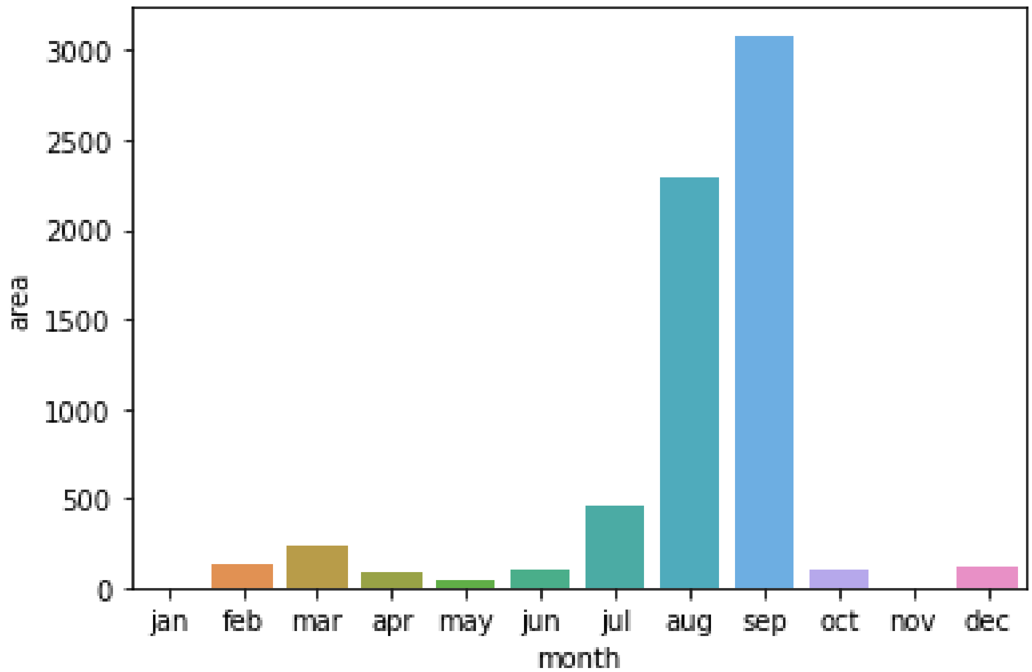
[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b6e05e2ec8>
```



We can see that almost all fires happen in August or September

[63]:
```
by_months = data.groupby("month").sum()
by_months = by_months.reset_index()
by_months = by_months[["month","area"]]
sns.barplot(data=by_months, x = "month", y = "area", order = months)
```

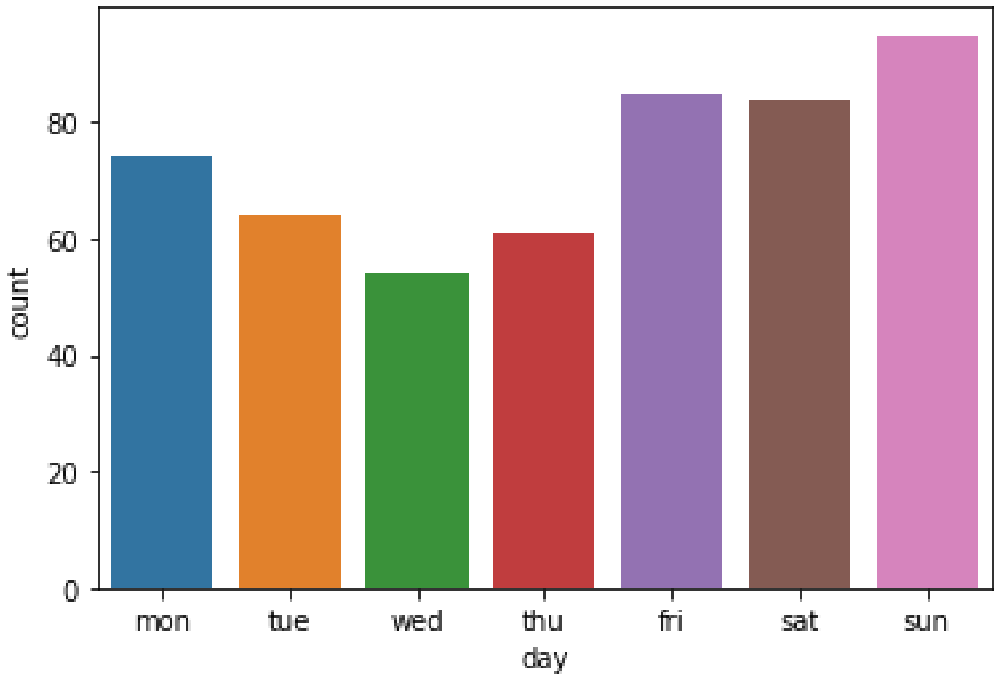
[63]: <matplotlib.axes._subplots.AxesSubplot at 0x1b6e06d0208>



Also the are burnt by fires is the biggest in those months.

[70]:
```
days = ['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
sns.countplot(data = data, x = "day", order = days )
```

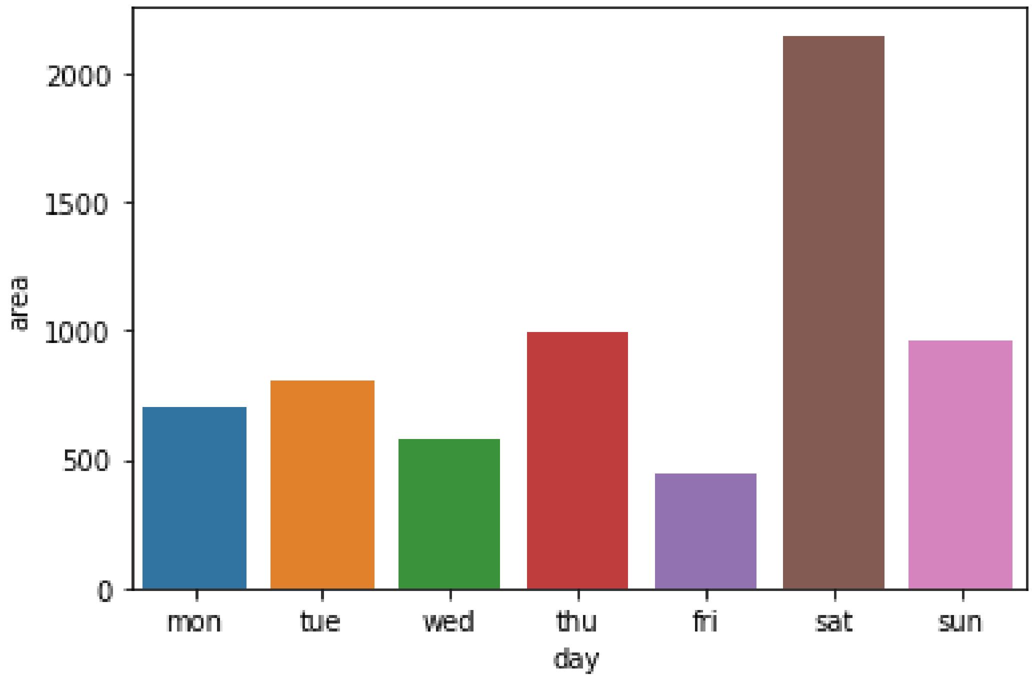
[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1b6e353e708>



Day of the week seems not to be relevant, although on Sunday there was the highest amount of forest fires.

[72]:
```
by_days = data.groupby("day").sum()
by_days = by_days.reset_index()
by_days = by_days[["day","area"]]
sns.barplot(data=by_days, x = "day", y = "area", order = days)
```

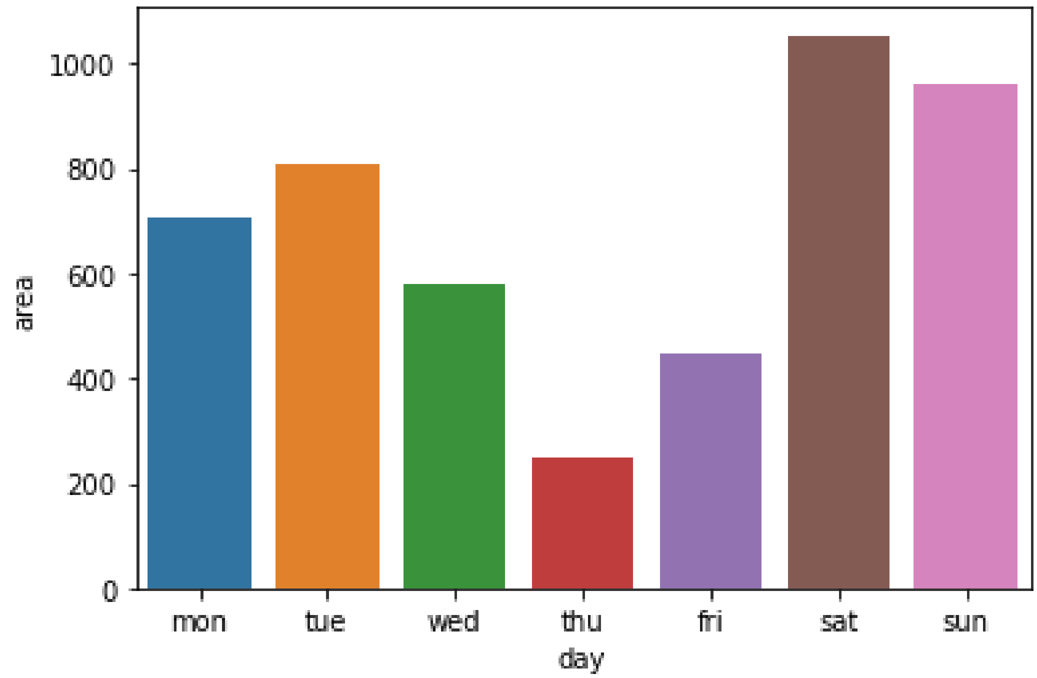
[72]: <matplotlib.axes._subplots.AxesSubplot at 0x1b6e05ea108>



But we can see that the biggest area was burnt on Saturday! but it might be due to our outliars. How does it look without them?

[73]:
```
by_days_without_outliers = fires_without_outliers.groupby("day").sum()
by_days_without_outliers = by_days_without_outliers.reset_index()
by_days_without_outliers = by_days_without_outliers[["day","area"]]
sns.barplot(data=by_days_without_outliers, x = "day", y = "area", order = days)
```

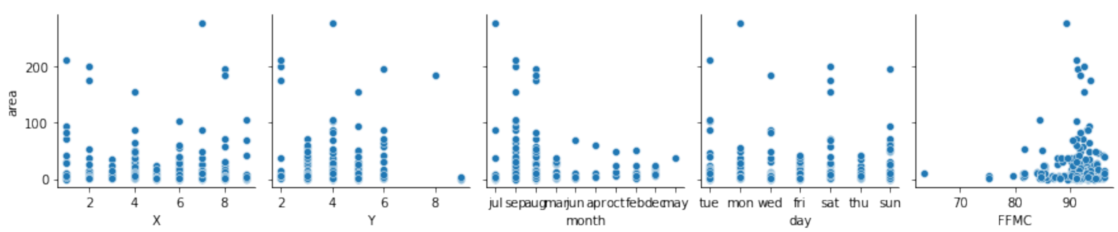
[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1b6dd31dcc8>

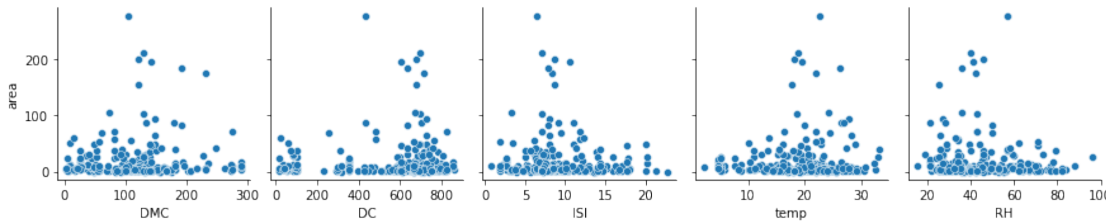


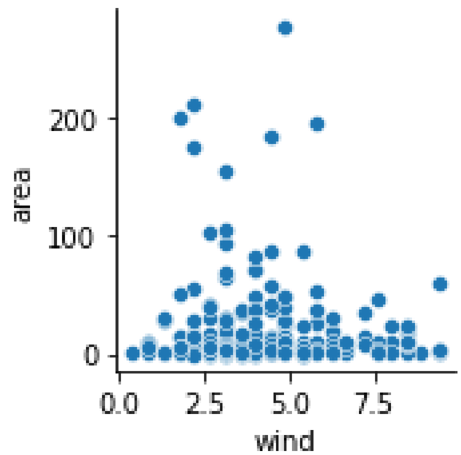
Now it looks a bit different. The biggest burnt area was still on Saturday, but the difference is not so massive.

[66]:
```
sns.pairplot(fires_without_outliers, y_vars = "area", x_vars = data.columns.
 ↪values[:5])
sns.pairplot(fires_without_outliers, y_vars = "area", x_vars = data.columns.
 ↪values[5:10])
sns.pairplot(fires_without_outliers, y_vars = "area", x_vars = data.columns.
 ↪values[10:11])
```

[66]:
```
<seaborn.axisgrid.PairGrid at 0x1b6e1ce8588>
```



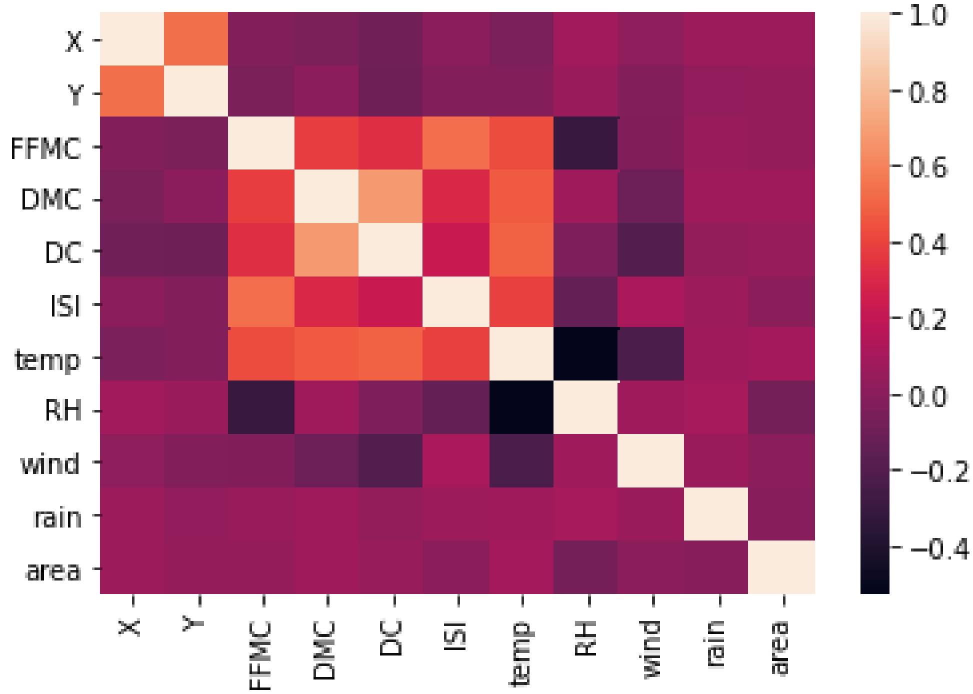





```
[ ]:
```

```
[18]: sns.heatmap(data.corr())
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1b6dcf61ec8>



It doesn't give us much information as we cannot see any correlation between area variable and other variables

[146]:
```
pandas_profiling.ProfileReport(data)
```

```
Summarize dataset:   0%|          | 0/25 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[146]:

It doesn't give us a possibility to somehow correlate categorical variables with continuous ones.