Advanced Data Science Capstone

Correlation of air pollution and Prevalence of Heart failures in Germany

Data sources

As data sources I use the data sets oficially published by Geschäfts- und Koordinierungsstelle GovData, the search engine is available at https://www.govdata.de/web/guest/suchen (https://www.govdata.de/web/guest/suchen). The list of relevant data sets is following:

Air quality:

Datastream **E1a** contains measured (Link to Datastream D) values of gas phase pollutants (e.g. Ozone, NO2, SO2, CO), particle pollutats (e.g. dust) and dust constituants (e.g. heavy metals, PAK in PM10, PM2.5, TSP) as well es total deposition (BULK), wet deposition and meteorologic data (e.g. temperature, wind, pressure)for every measurement location.

The data for years 2013 - 2018 is available. For the project start I will limit myself with 2017 data, however the method and the model shoud be easily extendable for accommodation of the previous data.

Compressed dataset is available at https://datahub.uba.de/server/rest/directories/arcgisforinspire/INSPIRE/aqd_MapServer/Daten/AQD_DE_E1a_2017.zip (https://datahub.uba.de/server/rest/directories/arcgisforinspire/INSPIRE/aqd_MapServer/Daten/AQD_DE_E1a_2017.zip) .

- Air quality data (Datastream E1a) - Validated measurements from 2018 (Dataset).
  - Metadata in RDF format: https://www.govdata.de/ckan/dataset/cdadb71f-5571-4c95-af2a-d926efb4e3a0.rdf (https://www.govdata.de/ckan/dataset/cdadb71f-5571-4c95-af2a-d926efb4e3a0.rdf) .
  - ATOM: Air quality data (INSPIRE Download/Atom Feed): https://datahub.uba.de/server/rest/directories/arcgisforinspire/INSPIRE/aqd_MapServer/Service_5e7bb800-c1db-4343-ac3e-a63204b0f6b7.atom.xml (https://datahub.uba.de/server/rest/directories/arcgisforinspire/INSPIRE/aqd_MapServer/Service_5e7bb800-c1db-4343-ac3e-a63204b0f6b7.atom.xml) (N/A)
  - XML-Metadaa: Air quality data (Datastream E1a) - Validated measurements from 2018 (Dataset): http://www.geoportal.de/gds/xml.php?uuid=cdadb71f-5571-4c95-af2a-d926efb4e3a0 (http://www.geoportal.de/gds/xml.php?uuid=cdadb71f-5571-4c95-af2a-d926efb4e3a0)
  - Positions of the air quality sensors: http://www.geoportal.de/gds/xml.php?uuid=DC9CAF92-8868-4E42-A718-C0EA9A99A5F0 (http://www.geoportal.de/gds/xml.php?uuid=DC9CAF92-8868-4E42-A718-C0EA9A99A5F0).

Health related information

Starting the search at the same site (https://www.govdata.de/web/guest/suchen (https://www.govdata.de/web/guest/suchen)) it was found, that the health indicator data with a fine (county) grid is not available at this site.

At the https://de.statista.com/statistik/ (https://de.statista.com/statistik/) the data of interest seems to be appearing, however a payed subscription is needed.

Finally, the https://www.versorgungsatlas.de/themen/alle-analysen-nach-datum-sortiert/ (https://www.versorgungsatlas.de/themen/alle-analysen-nach-datum-sortiert/) site contains county-averaged data on health indicators.

- Health indicators
  - Self-evaluation of the health status, distribution of "bad" values by counties: https://www.versorgungsatlas.de/fileadmin/excel/data_id_31_kreis_1_j_1262217600.xlsx (https://www.versorgungsatlas.de/fileadmin/excel/data_id_31_kreis_1_j_1262217600.xlsx)
  - Prevalence of Heart failures in the 2017 (used for the study): https://www.versorgungsatlas.de/fileadmin/excel/data_id_97_kreis11_2_j_1483228800.xlsx (https://www.versorgungsatlas.de/fileadmin/excel/data_id_97_kreis11_2_j_1483228800.xlsx)

Auxiliary information

In order to connect the datasets from different origin, one need to have a geotagging in the same format. The easiest way is to stick on county-based data, attribyte all air pollution sensors positions to corresponding counties.

Initial Data Exploration

- Load the data set and check it's structure, size and data quality (non-informative entries, data variability)
- If the data set is big, make a sample subset
- Make exploratory plots
- May be make some e.g. hierarchical/k-means (?) clustering, finding patterns

File naming conventions:

project_name.**data_exp**.*technology.version.extension*
project_name.**etl**.*technology.version.extension*
project_name.**feature_eng**.*technology.version.extension*
project_name.**model_def**.*technology.version.extension*
project_name.**model_train**.*technology.version.extension*
project_name.**model_evaluate**.*technology.version.extension*
project_name.**model_deployment**.*technology.version.extension*

Raw data files are placed to *./project_name*.**rawData/**

Loading all necessary libraries:

In [49]:
```
###import rdfpandas as pd
#!pip install rdflib
#!pip install networkx
#!pip install xlrd

import urllib.request
import xml.etree.ElementTree as ET
from lxml import etree
import pandas as pd
import numpy as np

import re, collections
from io import StringIO
import os, fnmatch

import matplotlib.pyplot as plt
```

Creating function for printing XML file structure

It will be useful, since almost all data at GovData.de is kept in *xml* format.

In [2]:
```
def PrintXML(XMLfileName):
    xml_root = (etree.parse(XMLfileName)).getroot()
    raw_tree = etree.ElementTree(xml_root)
    nice_tree = collections.OrderedDict()

    for tag in xml_root.iter():
        path = re.sub('\[[0-9]+\]', '', raw_tree.getpath(tag))
        if path not in nice_tree:
            nice_tree[path] = []
        if len(tag.keys()) > 0:
            nice_tree[path].extend(attrib for attrib in tag.keys() if attrib no
t in nice_tree[path])

    for path, attribs in nice_tree.items():
        indent = int(path.count('/') - 1)
        print('{0}{1}: {2} [{3}]'.format('    ' * indent, indent, path.split
('/')[-1], ', '.join(attribs) if len(attribs) > 0 else '-'))
```

Downloading the data

The code chunk is commented out in order to prevent multiple download of data.

In [3]:
```
## Download and decompress the dataset itself:
#!mkdir Capstone.rawData
#!mkdir Capstone.rawData/AQD_DE_E1a_2017
#!ls -l Capstone.rawData/
#urllib.request.urlretrieve("https://datahub.uba.de/server/rest/directories/arc
gisforinspire/INSPIRE/aqd_MapServer/Daten/AQD_DE_E1a_2017.zip", "Capstone.rawDa
ta/AQD_DE_E1a_2017.zip")
#!mv Capstone.rawData/AQD_DE_E1a_2017.zip Capstone.rawData/AQD_DE_E1a_2017/
#!unzip Capstone.rawData/AQD_DE_E1a_2017/AQD_DE_E1a_2017.zip -d Capstone.rawDat
a/
#!rm Capstone.rawData/AQD_DE_E1a_2017/AQD_DE_E1a_2017.zip
#!unzip Capstone.rawData/AQD_DE_E1a_2017/DISKO.zip -d Capstone.rawData/AQD_DE_E
1a_2017/
#!unzip Capstone.rawData/AQD_DE_E1a_2017/KONTI.zip -d Capstone.rawData/AQD_DE_E
1a_2017/
#!rm Capstone.rawData/AQD_DE_E1a_2017/DISKO.zip Capstone.rawData/AQD_DE_E1a_
2017/KONTI.zip
#!mv ./E1a Capstone.rawData/AQD_DE_E1a_2018/
#!mv Capstone.rawData/AQD_DE_E1a_2018/E1a/* Capstone.rawData/AQD_DE_E1a_2018/
#!rm -rf Capstone.rawData/AQD_DE_E1a_2018/E1a
#!ls -la /home/spark/shared/

#Download the dataset Metadata xml
#urllib.request.urlretrieve("http://www.geoportal.de/gds/xml.php?uuid=cdadb71f
-5571-4c95-af2a-d926efb4e3a0", "Capstone.rawData/AQD_DE_E1a_2018/E1a2018_meta.x
ml")
#urllib.request.urlretrieve("http://www.geoportal.de/gds/xml.php?uuid=c533b9a5-
e518-4bf8-9a0a-1b829acd561d", "Capstone.rawData/AQD_DE_E1a_2017/E1a2017_meta.xm
l")

#Download Sensor positions
#urllib.request.urlretrieve("http://www.geoportal.de/gds/xml.php?uuid=DC9CAF
92-8868-4E42-A718-C0EA9A99A5F0", "Capstone.rawData/AQD_DE_E1a_2018/Sensor_Posit
ions.xml")

#!ls -la Capstone.rawData/AQD_DE_E1a_2017/
#!pwd
```

Metadata XML file

Now let's take a look at the downloaded *E1a2018_meta.xml* file. First print it's structure

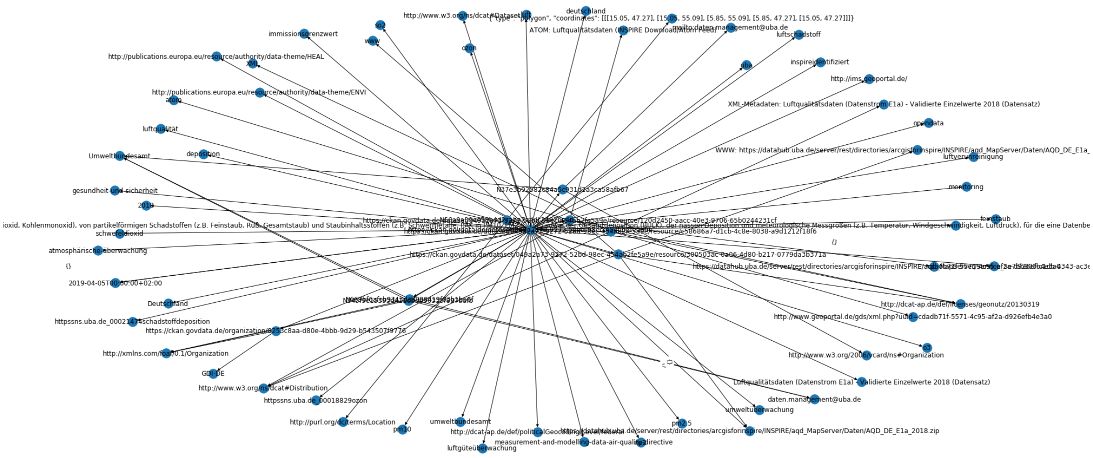
In [4]:

```
PrintXML("Capstone.rawData/AQD_DE_E1a_2018/E1a2018_meta.xml")
```

```
0: gmd:MD_Metadata [{http://www.w3.org/2001/XMLSchema-instance}schemaLocation]
    1: gmd:fileIdentifier [-]
        2: gco:CharacterString [-]
    1: gmd:language [-]
        2: * [codeList, codeListValue]
    1: gmd:characterSet [-]
        2: * [codeList, codeListValue]
    1: gmd:hierarchyLevel [-]
        2: * [codeList, codeListValue]
    1: gmd:hierarchyLevelName [-]
        2: gco:CharacterString [-]
    1: gmd:contact [-]
        2: gmd:CI_ResponsibleParty [-]
            3: gmd:individualName [-]
                4: gco:CharacterString [-]
            3: gmd:organisationName [-]
                4: gco:CharacterString [-]
            3: gmd:positionName [-]
                4: gco:CharacterString [-]
            3: gmd:contactInfo [-]
                4: gmd:CI_Contact [-]
                    5: gmd:phone [-]
                        6: gmd:CI_Telephone [-]
                            7: gmd:voice [-]
                                8: gco:CharacterString [-]
                            7: gmd:facsimile [-]
                                8: gco:CharacterString [-]
                    5: gmd:address [-]
                        6: gmd:CI_Address [-]
                            7: gmd:deliveryPoint [-]
                                8: gco:CharacterString [-]
                            7: gmd:city [-]
                                8: gco:CharacterString [-]
                            7: gmd:administrativeArea [-]
                                8: gco:CharacterString [-]
                            7: gmd:postalCode [-]
                                8: gco:CharacterString [-]
                            7: gmd:country [-]
                                8: gco:CharacterString [-]
                            7: gmd:electronicMailAddress [-]
                                8: gco:CharacterString [-]
                    5: gmd:onlineResource [-]
                        6: gmd:CI_OnlineResource [-]
                            7: gmd:linkage [-]
                                8: gmd:URL [-]
            3: gmd:role [-]
                4: gmd:CI_RoleCode [codeList, codeListValue]
    1: gmd:dateStamp [-]
        2: gco:Date [-]
    1: gmd:metadataStandardName [-]
        2: gco:CharacterString [-]
    1: gmd:metadataStandardVersion [-]
        2: gco:CharacterString [-]
    1: gmd:dataSetURI [-]
        2: gco:CharacterString [-]
    1: gmd:referenceSystemInfo [-]
        2: gmd:MD_ReferenceSystem [-]
            3: gmd:referenceSystemIdentifier [-]
                4: gmd:RS_Identifier [-]
                    5: gmd:code [-]
                        6: gco:CharacterString [-]
                    5: gmd:codeSpace [-]
                        6: gco:CharacterString [-]
                    5: gmd:version [-]
                        6: gco:CharacterString [-]
    1: gmd:identificationInfo [-]
        2: gmd:MD_DataIdentification [-]
            3: gmd:citation [-]
                4: gmd:CI_Citation [-]
                    5: gmd:title [-]
```

... that bring us not so much information. Let's try to visualize it's **rdf** version using *networkx* library:

In [5]:
```python
#### Metadata xml also contains info on the relations between datasets, models
and organizations; It is not very informative.
import rdflib
from rdflib import Graph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_multidigraph
import networkx as nx


### Metadata Graph shows relations between datasets, models and organizations;
It is not very informative.
E1a2018_meta_graph = Graph()
E1a2018_meta_graph.parse("https://www.govdata.de/ckan/dataset/cdadb71f-5571-4c
95-af2a-d926efb4e3a0.rdf")
E1a2018_meta_network = rdflib_to_networkx_multidigraph(E1a2018_meta_graph)
# Plot Networkx instance of RDF Graph
plt.figure(figsize=(36,18))
pos = nx.spring_layout(E1a2018_meta_network, scale=5)
edge_labels = nx.get_edge_attributes(E1a2018_meta_network, 'r')
nx.draw_networkx_edge_labels(E1a2018_meta_network, pos, labels=edge_labels)
nx.draw(E1a2018_meta_network, with_labels=True)
plt.show()
```



This also give us almost no information about the dataset. Finaly, printing out persed *xml* file (it is not so big, just 37 kB, for code see commented chunk below) I found, that it contains urls to data formats and standarts: http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/codelist/ML_gmxCodelists.xml#CI_DateTypeCode, https://registry.gdi-de.org/id/de.bund.uba.inspire.aqd/cdadb71f-5571-4c95-af2a-d926efb4e3a0, etc.

Also it contains description of the measurement details: *Datenstrom E1a umfasst gemessene (Link zu Datenstrom D) Einzelwerte von gasförmigen Schadstoffen (z. B. Ozon, Stickstoffdixoid, Schwefeldioxid, Kohlenmonoxid), von partikelförmigen Schadstoffen (z.B. Feinstaub, Ruß, Gesamtstaub) und Staubinhaltsstoffen (z.B. Schwermetalle, PAK in PM10, PM2.5, TSP) sowie der Gesamtdeposition (BULK), der nassen Deposition und meteorologische Messgrößen (z.B. Temperatur, Windgeschwindigkeit, Luftdruck), für die eine Datenbereitstellungspflicht besteht. Der Bericht umfasst zudem die Datenqualitätsziele (Messunsicherheit, Mindestzeiterfassung (time coverage) erfüllt ja/nein, Mindestdatenerfassung (data capture) erfüllt ja/nein) und Informationen zu Konzentrationswerten die natürlichen Quellen und der Ausbringung von Streusand und –salz zuzurechnen sind (Konzentrationswerte ohne etwaige Korrekturabzüge).*

Since I will normalize the data anyway, the only important thing is that all the data sets contains the same units. However the units of measurements are also mentioned in the data files themselves, as it will be shown later, so these metadata files are out of use, at least for the moment.

In [6]:
```
#import xml.dom.minidom
#with open('Capstone.rawData/AQD_DE_E1a_2018/E1a2018_meta.xml', encoding='utf
-8') as xmldata:
#    xml = xml.dom.minidom.parseString(xmldata.read())  # or xml.dom.minidom.pa
rseString(xml_string)
#    xml_pretty_str = xml.toprettyxml()
#print(xml_pretty_str)
```

Data set files analysis

Let's take a look at the downloaded data set:

In [7]:
```
#PrintXML("Capstone.rawData/AQD_DE_E1a_2018/Sensor_Positions.xml")
```

In [8]:

```
!ls -la Capstone.rawData/AQD_DE_E1a_2018/
```

```
total 2239472
drwxr-xr-x  156 gorelov  staff      4992 Jul  8 11:08 .
drwxr-xr-x    6 gorelov  staff       192 Jul  8 11:24 ..
-rw-r--r--@   1 gorelov  staff      6148 Jul  8 11:09 .DS_Store
-rw-r--r--    1 gorelov  staff   2737854 Feb  4 15:11 DE_BB_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff  13254333 Feb  4 15:11 DE_BB_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff  13169219 Feb  4 15:11 DE_BB_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff  13290213 Feb  4 15:11 DE_BB_2018_NOx_hour.xml
-rw-r--r--    1 gorelov  staff   8897254 Feb  4 15:11 DE_BB_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff  13292670 Feb  4 15:11 DE_BB_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff  13250141 Feb  4 15:12 DE_BB_2018_PM2_hour.xml
-rw-r--r--    1 gorelov  staff   2738411 Feb  4 15:12 DE_BB_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   1654796 Feb  4 15:12 DE_BE_2018_CHB_hour.xml
-rw-r--r--    1 gorelov  staff   1104344 Feb  4 15:12 DE_BE_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff   8917510 Feb  4 15:12 DE_BE_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff   8799792 Feb  4 15:12 DE_BE_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff   8886254 Feb  4 15:12 DE_BE_2018_NOx_hour.xml
-rw-r--r--    1 gorelov  staff   3915493 Feb  4 15:12 DE_BE_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff   6140177 Feb  4 15:12 DE_BE_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff   1104543 Feb  4 15:12 DE_BE_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   4379211 Feb  4 15:15 DE_BW_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff  22162549 Feb  4 15:15 DE_BW_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff  19930475 Feb  4 15:15 DE_BW_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff  14998614 Feb  4 15:15 DE_BW_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff   1204632 Feb  4 15:16 DE_BW_2018_PM1_day.xml
-rw-r--r--    1 gorelov  staff  16290298 Feb  4 15:15 DE_BW_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff    624138 Feb  4 15:16 DE_BW_2018_PM2_day.xml
-rw-r--r--    1 gorelov  staff   2193394 Feb  4 15:16 DE_BW_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   1195946 Feb  4 15:12 DE_BY_2018_CHB_hour.xml
-rw-r--r--    1 gorelov  staff   7779233 Feb  4 15:14 DE_BY_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff  27665884 Feb  4 15:13 DE_BY_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff  27484399 Feb  4 15:14 DE_BY_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff  21027058 Feb  4 15:13 DE_BY_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff  19237226 Feb  4 15:13 DE_BY_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff  18979929 Feb  4 15:14 DE_BY_2018_PM2_hour.xml
-rw-r--r--    1 gorelov  staff   1288659 Feb  4 15:14 DE_BY_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   2205414 Feb  4 15:19 DE_HB_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff   4462580 Feb  4 15:19 DE_HB_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff   4397937 Feb  4 15:19 DE_HB_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff   4445978 Feb  4 15:19 DE_HB_2018_NOx_hour.xml
-rw-r--r--    1 gorelov  staff   2796735 Feb  4 15:19 DE_HB_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff   4457989 Feb  4 15:19 DE_HB_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff   1666645 Feb  4 15:19 DE_HB_2018_PM2_hour.xml
-rw-r--r--    1 gorelov  staff   3306616 Feb  4 15:19 DE_HB_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   2989575 Feb  4 15:17 DE_HE_2018_CHB_hour.xml
-rw-r--r--    1 gorelov  staff   7781105 Feb  4 15:18 DE_HE_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff  21154955 Feb  4 15:17 DE_HE_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff  20962107 Feb  4 15:18 DE_HE_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff  21207768 Feb  4 15:18 DE_HE_2018_NOx_hour.xml
-rw-r--r--    1 gorelov  staff  14566004 Feb  4 15:18 DE_HE_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff     32181 Feb  4 15:18 DE_HE_2018_PM1_day.xml
-rw-r--r--    1 gorelov  staff  19325435 Feb  4 15:17 DE_HE_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff   5727673 Feb  4 15:18 DE_HE_2018_PM2_hour.xml
-rw-r--r--    1 gorelov  staff   6592473 Feb  4 15:18 DE_HE_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   1112896 Feb  4 15:16 DE_HH_2018_CHB_hour.xml
-rw-r--r--    1 gorelov  staff   1114374 Feb  4 15:16 DE_HH_2018_CHT_hour.xml
-rw-r--r--    1 gorelov  staff   1703076 Feb  4 15:16 DE_HH_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff   8430234 Feb  4 15:16 DE_HH_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff   8310360 Feb  4 15:16 DE_HH_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff   8449651 Feb  4 15:17 DE_HH_2018_NOx_hour.xml
-rw-r--r--    1 gorelov  staff   2254282 Feb  4 15:16 DE_HH_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff   6179603 Feb  4 15:16 DE_HH_2018_PM1_hour.xml
-rw-r--r--    1 gorelov  staff   3168949 Feb  4 15:17 DE_HH_2018_PM2_hour.xml
-rw-r--r--    1 gorelov  staff   3296671 Feb  4 15:17 DE_HH_2018_SO2_hour.xml
-rw-r--r--    1 gorelov  staff   1644332 Feb  4 15:19 DE_MV_2018_CO_hour.xml
-rw-r--r--    1 gorelov  staff   7719841 Feb  4 15:19 DE_MV_2018_NO2_hour.xml
-rw-r--r--    1 gorelov  staff   7681727 Feb  4 15:20 DE_MV_2018_NO_hour.xml
-rw-r--r--    1 gorelov  staff   5004752 Feb  4 15:19 DE_MV_2018_O3_hour.xml
-rw-r--r--    1 gorelov  staff     30853 Feb  4 15:20 DE_MV_2018_PM1_day.xml
-rw-r--r--    1 gorelov  staff   8382426 Feb  4 15:19 DE_MV_2018_PM1_hour.xml
```

It is clear, that filenames contain information on the file contents: e.g. **DE_BB_2018_NOx_hour.xml**: **DE** for Germany, **BB** for Brandenburg region, **2018** for year of measurements, **NOx** for kind of pollutant (nitrogen oxides), **hour** for type of measurements.
Now let's take a look at the *xml* file structure:

In [9]:
```
PrintXML("Capstone.rawData/AQD_DE_E1a_2017/DE_BB_2017_NOx_hour.xml")
```

```
0: gml:FeatureCollection [{http://www.w3.org/2001/XMLSchema-instance}schemaLoc
ation, {http://www.opengis.net/gml/3.2}id]
    1: gml:featureMember [-]
        2: om:OM_Observation [{http://www.opengis.net/gml/3.2}id]
            3: om:phenomenonTime [-]
                4: gml:TimePeriod [{http://www.opengis.net/gml/3.2}id]
                    5: gml:beginPosition [-]
                    5: gml:endPosition [-]
            3: om:resultTime [-]
                4: gml:TimeInstant [{http://www.opengis.net/gml/3.2}id]
                    5: gml:timePosition [-]
            3: om:procedure [{http://www.w3.org/1999/xlink}href]
            3: om:parameter [-]
                4: om:NamedValue [-]
                    5: om:name [{http://www.w3.org/1999/xlink}href]
                    5: om:value [{http://www.w3.org/1999/xlink}href, {http://w
ww.w3.org/2001/XMLSchema-instance}type]
            3: om:observedProperty [{http://www.w3.org/1999/xlink}href]
            3: om:featureOfInterest [{http://www.w3.org/1999/xlink}href]
            3: om:result [{http://www.w3.org/2001/XMLSchema-instance}type]
                4: ns:elementCount [-]
                    5: ns:Count [-]
                        6: ns:value [-]
                4: ns:elementType [name]
                    5: ns:DataRecord [-]
                        6: ns:field [name]
                            7: ns:Time [definition]
                                8: ns:uom [{http://www.w3.org/1999/xlink}href]
                            7: ns:Category [definition]
                            7: ns:Quantity [definition]
                                8: ns:uom [{http://www.w3.org/1999/xlink}href]
                4: ns:encoding [-]
                    5: ns:TextEncoding [decimalSeparator, tokenSeparator, bloc
kSeparator]
                4: ns:values [-]
        2: aqd:AQD_ReportingHeader [{http://www.opengis.net/gml/3.2}id]
            3: aqd:change [-]
            3: aqd:changeDescription [-]
            3: aqd:inspireId [-]
                4: base:Identifier [-]
                    5: base:localId [-]
                    5: base:namespace [-]
                    5: base:versionId [-]
            3: aqd:reportingAuthority [-]
                4: base2:RelatedParty [-]
                    5: base2:individualName [-]
                        6: gmd:PT_FreeText [-]
                            7: gmd:textGroup [-]
                                8: gmd:LocalisedCharacterString [-]
                    5: base2:organisationName [-]
                        6: gmd:PT_FreeText [-]
                            7: gmd:textGroup [-]
                                8: gmd:LocalisedCharacterString [-]
                    5: base2:contact [-]
                        6: base2:Contact [-]
                            7: base2:address [-]
                                8: ad:AddressRepresentation [-]
                                    9: ad:adminUnit [-]
                                        10: gn:GeographicalName [-]
                                            11: gn:language [-]
                                            11: gn:nativeness [-]
                                            11: gn:nameStatus [nilReason, {htt
p://www.w3.org/2001/XMLSchema-instance}nil]
                                            11: gn:sourceOfName [nilReason, {h
ttp://www.w3.org/2001/XMLSchema-instance}nil]
                                            11: gn:pronunciation [nilReason,
{http://www.w3.org/2001/XMLSchema-instance}nil]
                                            11: gn:spelling [-]
                                                12: gn:SpellingOfName [-]
                                                    13: gn:text [-]
```

Agan, a lot of urls describing standards, however the structure of file becomes clearer. Let's check the first rank entries from the root:

In [10]:
```
xml_tree = etree.parse("Capstone.rawData/AQD_DE_E1a_2017/DE_BB_2017_NOx_hour.xm
l")
xml_root = xml_tree.getroot()
for child in xml_root:
    print(child.tag, child.attrib)
```

```
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
{http://www.opengis.net/gml/3.2}featureMember {}
```

It seems to be, that the file can contain measurements from 25 sensors at different locations (as it will be shown later, actual file contains only 24 sensors data, the first entry is used for auxiliary information). An *xml* file entry consist of three possible elements: **tag**, like key or variable name; **attrib**, like value of variable, and **text**, that accommodates everything between corresponding tags (up to video in *base64* encoding). For the further use I will collect all the **tag**s from the dataset file and store it to the *AllTags* list.

In [11]:
```
# pick all tags from the XML file
Etree = ET.parse("Capstone.rawData/AQD_DE_E1a_2017/DE_BB_2017_NOx_hour.xml")
Eroot = Etree.getroot()
Eroot.tag
Eroot.attrib
AllTags = [elem.tag for elem in Eroot.iter()]
print(AllTags[23:35])
#varName = 'observedProperty'
#print("\n".join([s for s in AllTags if varName in s]))
```

```
['{http://www.opengis.net/swe/2.0}Count', '{http://www.opengis.net/swe/2.0}val
ue', '{http://www.opengis.net/swe/2.0}elementType', '{http://www.opengis.net/s
we/2.0}DataRecord', '{http://www.opengis.net/swe/2.0}field', '{http://www.open
gis.net/swe/2.0}Time', '{http://www.opengis.net/swe/2.0}uom', '{http://www.ope
ngis.net/swe/2.0}field', '{http://www.opengis.net/swe/2.0}Time', '{http://www.
opengis.net/swe/2.0}uom', '{http://www.opengis.net/swe/2.0}field', '{http://ww
w.opengis.net/swe/2.0}Category']
```

Exploring Dataset XML structure

First of all, let's write functions, that will fetch *xml* tags by contained keyword. The ideas about keywords needed can be got from looking at *AllTags* list. The function *FetchAllXMLsensor* parses the sensor ID from the corresponding *xml* **attr** at the **value** tag.

In [12]:
```
def FetchXMLentryByWord(varName, NumToPrint):
    varFull = [s for s in AllTags if varName in s][NumToPrint]
    print(varFull)
    print([(varr.attrib, varr.text) for varr in Eroot.iter(varFull)][NumToPrin
t])
    print('\n')
def FetchAllXMLentriesByWord(varName):
    varFull = [s for s in AllTags if varName in s][0]
    print([(varr.attrib) for varr in Eroot.iter(varFull)])
    print('\n')
def FetchAllXMLsensorID():
    varFull = [s for s in AllTags if 'value' in s][0]
    print([re.sub(r'[^a-zA-Z0-9:]*\'{http(.*)$', r'', re.sub(r'^.*AQD\/SPO.DE_
', r'', str(varr.attrib))) for varr in Eroot.iter(varFull) if 'AQD' in str(var
r.attrib)])
    print('\n')
def SelectAllXMLsensorID():
    varFull = [s for s in AllTags if 'value' in s][0]
    return([re.sub(r'[^a-zA-Z0-9:]*\'{http(.*)$', r'', re.sub(r'^.*AQD\/SPO.DE_
', r'', str(varr.attrib))) for varr in Eroot.iter(varFull) if 'AQD' in str(var
r.attrib)])
```

Using the Fetch functions, one can show, that units of measurements in this *xml* file are **microgramm pro cubic meter**, the pollutant is of type 9 (according to the given url it is **NOX as NO2**) and observation time os an **hour**.

One can parse sensor IDs in form of **DEBB007_NOx_dataGroup1**, where **DE** stands for Germany, **BB** for Brandenburg region, **007** is the sensor number (this information will be used later for geotagging), **NOx** for kind of pollutant (nitrogen oxides), and the **dataGroup1** is some common index.

The pollutant concentration data is stored in **text** fields of the entries with **value** tag; the data is stored in the *csv* format, end line symbol is "@@":

In [13]:
```
FetchXMLentryByWord('Quantity', 0)

FetchXMLentryByWord('uom', 2)
FetchXMLentryByWord('observedProperty', 0)

FetchAllXMLsensorID()

FetchXMLentryByWord('TextEncoding', 0)
ColNamesExp=SelectAllXMLsensorID()
```

```
{http://www.opengis.net/swe/2.0}Quantity
({'definition': 'http://dd.eionet.europa.eu/vocabulary/aq/primaryObservation/h
our'}, '\n                 ')


{http://www.opengis.net/swe/2.0}uom
({'{http://www.w3.org/1999/xlink}href': 'http://dd.eionet.europa.eu/vocabulary
/uom/concentration/ug.m-3'}, None)


{http://www.opengis.net/om/2.0}observedProperty
({'{http://www.w3.org/1999/xlink}href': 'http://dd.eionet.europa.eu/vocabulary
/aq/pollutant/9'}, None)


['DEBB007_NOx_dataGroup1', 'DEBB021_NOx_dataGroup1', 'DEBB029_NOx_dataGroup1',
'DEBB032_NOx_dataGroup1', 'DEBB044_NOx_dataGroup1', 'DEBB045_NOx_dataGroup1',
'DEBB048_NOx_dataGroup1', 'DEBB049_NOx_dataGroup1', 'DEBB053_NOx_dataGroup1',
'DEBB054_NOx_dataGroup1', 'DEBB055_NOx_dataGroup1', 'DEBB060_NOx_dataGroup1',
'DEBB063_NOx_dataGroup1', 'DEBB064_NOx_dataGroup1', 'DEBB065_NOx_dataGroup1',
'DEBB066_NOx_dataGroup1', 'DEBB067_NOx_dataGroup1', 'DEBB068_NOx_dataGroup1',
'DEBB073_NOx_dataGroup1', 'DEBB075_NOx_dataGroup1', 'DEBB083_NOx_dataGroup1',
'DEBB086_NOx_dataGroup1', 'DEBB092_NOx_dataGroup1', 'DEBB099_NOx_dataGroup1']


{http://www.opengis.net/swe/2.0}TextEncoding
({'decimalSeparator': '.', 'tokenSeparator': ',', 'blockSeparator': '@@'}, Non
e)
```

Reading Pollutant concentration Measurements to DataFrame

For exploratory analysis the data should be loaded to some toolbox, in this case *Pandas* in form of *Pandas Dataframe*. As it was shown before, one have to parse **text** fields of the entries with **value** tag as a normal *csv* file:

In [14]:
```
varFull = [s for s in AllTags if 'values' in s][0]

dff=[]
for varr in Eroot.iter(varFull):
    dff.append(pd.read_csv(StringIO((varr.text).replace("@@","\n")), sep=",", h
eader=None))
```

In [15]:
```
[dff[s].shape for s in range(0,len(dff))]
```

Out[15]:
```
[(8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5),
 (8760, 5)]
```

In [16]:
```
dff[22].tail(5)
```

Out[16]:

	0	1	2	3	4
8755	2017-12-31T19:00:00+01:00	2017-12-31T20:00:00+01:00	1	1	3.82
8756	2017-12-31T20:00:00+01:00	2017-12-31T21:00:00+01:00	1	1	3.82
8757	2017-12-31T21:00:00+01:00	2017-12-31T22:00:00+01:00	1	1	5.07
8758	2017-12-31T22:00:00+01:00	2017-12-31T23:00:00+01:00	1	1	3.82
8759	2017-12-31T23:00:00+01:00	2017-12-31T24:00:00+01:00	1	1	3.82

From the chunks above one can see that in the current **DE_BB_2018_NOx_hour.xml** file there is measurement data for 24 sensors, each containing **8760** values - one measurement for each hour of the year. At least in the current file there are no NA values:

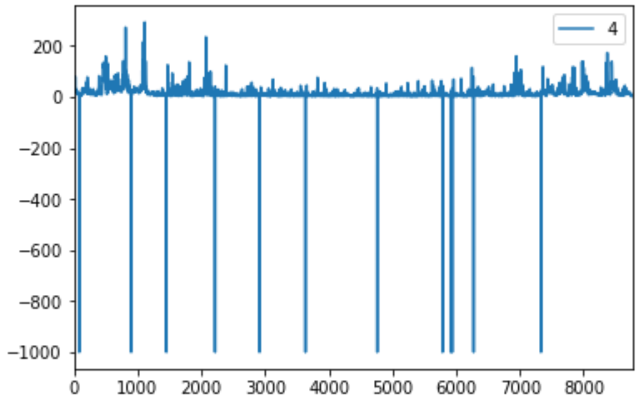
In [17]:
```
[dff[s].isnull().sum() for s in range(0,len(dff))]
```

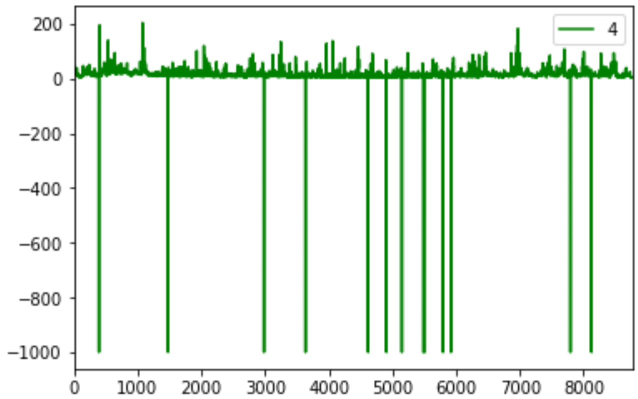
Out[17]:
```
[0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
dtype: int64, 0     0
1     0
2     0
3     0
4     0
```

In [18]:
```
plt.figure()
((dff[0])[[4]]).plot()
((dff[22])[[4]]).plot(style='g')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x11e777cf8>

<Figure size 432x288 with 0 Axes>





From the figures above it is clear that the data contains also failed measurements, encoded as negative values (pollutant concentration cannot be negative). It meams that *imputting* strategy should be developed in order to treat the problem. At the moment these negative values will be replaced with zeroes, it will cause no problem for initial exploratory analysis:

In [19]:
```
dffAll=pd.concat([dff[s][4] for s in range(0,len(dff))], axis=1)
dffAll.columns=ColNamesExp
dffAll=dffAll.clip(lower=0)
dffAll.head(5)
```

Out[19]:

	DEBB007_NOx_dataGroup1	DEBB021_NOx_dataGroup1	DEBB029_NOx_dataGroup1	DEBB032_NOx_dataGr
0	72.39	25.92	27.44	
1	45.29	15.77	14.03	
2	34.04	13.66	15.87	
3	30.59	12.83	15.87	
4	30.31	15.04	21.78	

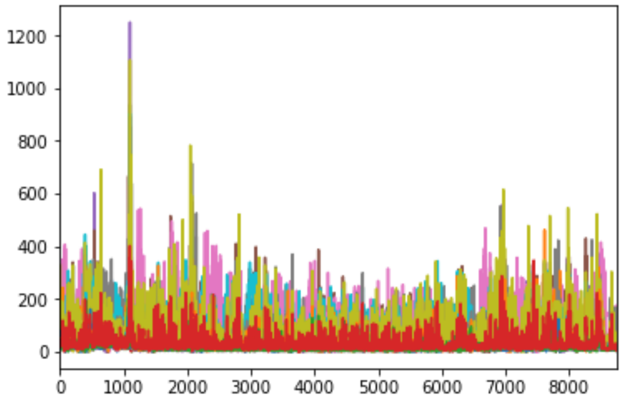
5 rows × 24 columns

Now we can plot selected **DE_BB_2018_NOx_hour.xml** file as a timeseries:

In [20]:
```
plt.figure(figsize=(36,18))
dffAll.plot(legend=None)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x11ea36ba8>

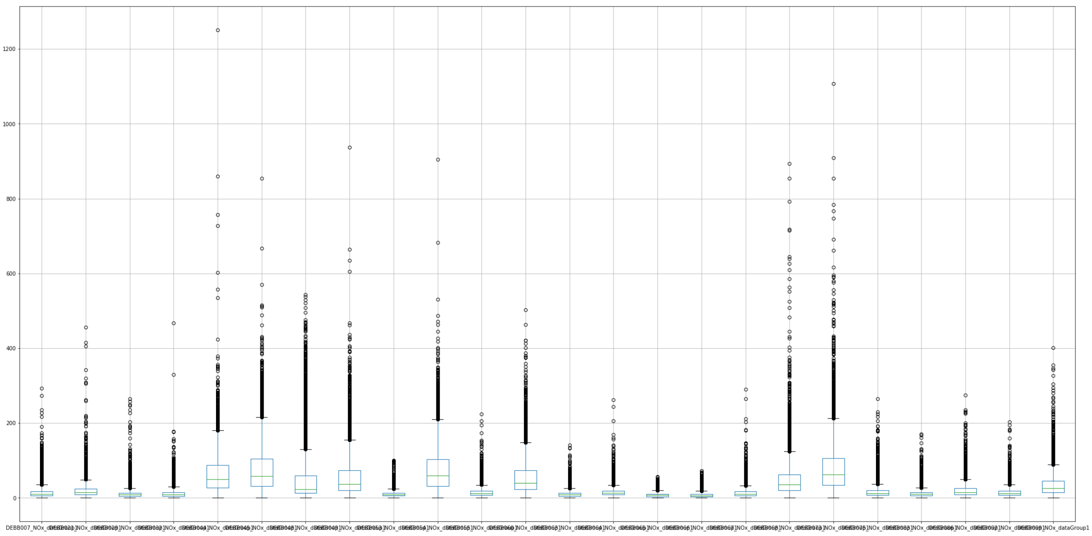
<Figure size 2592x1296 with 0 Axes>



However for such a long-term effects as a public health impact, some detived quantities, like *number of days in year with too high pollutant concentration*, or some other integral quantities will be used. At the current stage a summary over the year w.r.t. the sensors can be done:

In [21]:
```
plt.figure(figsize=(36,18))
dffAll.boxplot()
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x11e7bc978>

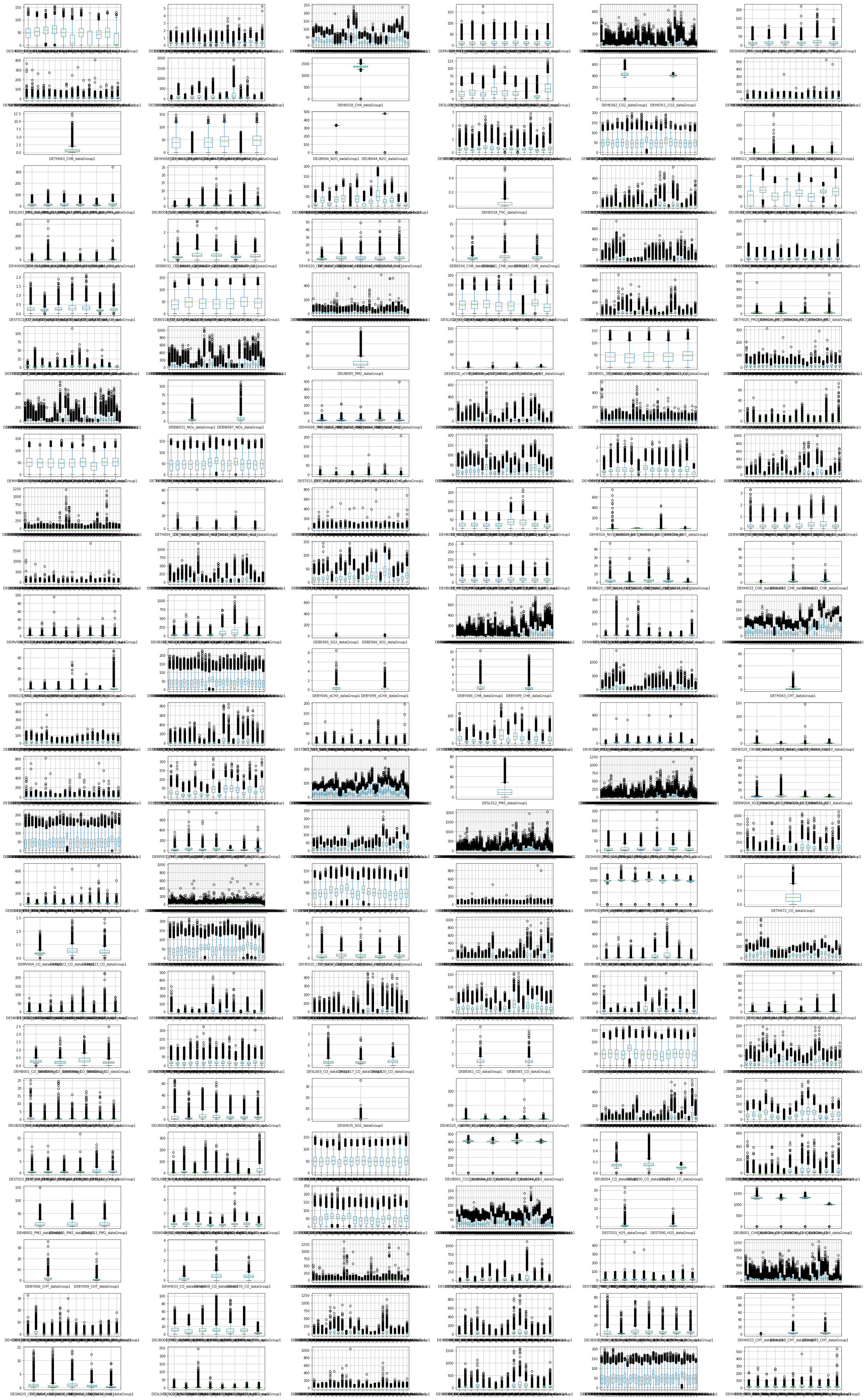


On the figure above one can see that different sensors demonstrate significantly different statistics, so the data variativity should be enough to be used in the project. Also a quick check of the sanity of the whole pollutant dataset could be done by plotting all the files in the manner showed above:

In [161]:

```
#!ls Capstone.rawData/AQD_DE_E1a_2017/*hour*
FilesHour=[]

for file in os.listdir('Capstone.rawData/AQD_DE_E1a_2017/'):
    if fnmatch.fnmatch(file, '*hour*'):
        FilesHour.append(file)
print(len(FilesHour))

#FilesHour=FilesHour[0:12]

fig = plt.figure(figsize=(36,58))
NfigRows=26
NfigCols=6


for file in FilesHour:
    Etree = ET.parse('Capstone.rawData/AQD_DE_E1a_2017/'+file)
    Eroot = Etree.getroot()
    Eroot.tag
    Eroot.attrib
    AllTags = [elem.tag for elem in Eroot.iter()]

    ColNamesExp=SelectAllXMLsensorID()

    varFull = [s for s in AllTags if 'values' in s][0]
    dff=[]
    for varr in Eroot.iter(varFull):
        dff.append(pd.read_csv(StringIO((varr.text).replace("@@","\n")), se
p=",", header=None))

    dffAll=pd.concat([dff[s][4] for s in range(0,len(dff))], axis=1)
    dffAll.columns=ColNamesExp
    dffAll=dffAll.clip(lower=0)

    ax = fig.add_subplot(NfigRows,NfigCols,(FilesHour.index(file)+1))
    ax = (dffAll.boxplot(return_type='both'))

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

156



Prevalence of Heart failures

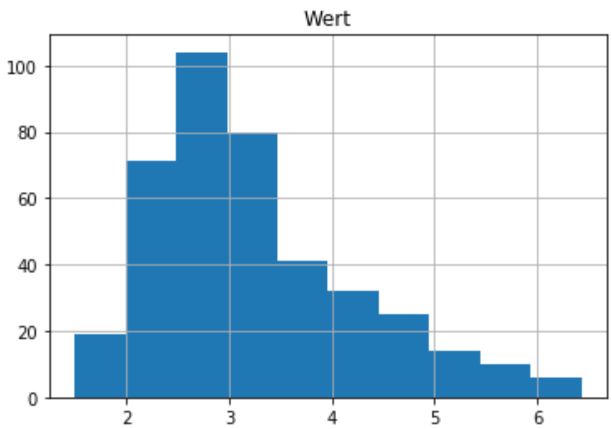
The dataset can be downloaded at [http://www.geoportal.de/gds/xml.php?uuid=DC9CAF92-8868-4E42-A718-C0EA9A99A5F0 (http://www.geoportal.de/gds/xml.php?uuid=DC9CAF92-8868-4E42-A718-C0EA9A99A5F0)](http://www.geoportal.de/gds/xml.php?uuid=DC9CAF92-8868-4E42-A718-C0EA9A99A5F0) in xlsx format:

In [44]:
```
#!mkdir Capstone.rawData/Heart_2017
#urllib.request.urlretrieve("https://www.versorgungsatlas.de/fileadmin/excel/data_id_97_kreis11_2_j_1483228800.xlsx", "Capstone.rawData/Heart_2017/data_id_97_kreis11_2_j_1483228800.xlsx")

xlsx_file = pd.ExcelFile("Capstone.rawData/Heart_2017/data_id_97_kreis11_2_j_1483228800.xlsx")
print("xls sheet names: ",xlsx_file.sheet_names)
dfHeart = xlsx_file.parse('Daten', header=3, decimal=",")
print(dfHeart.head(5))
print("Length of the dataset: ",len(dfHeart))
plt.figure()
dfHeart.hist(column='Wert')
```

```
xls sheet names:  ['Hintergrundinformationen', 'Daten']
             Region  Regions-ID  KV          Kreistyp  Wert  Bundeswert
0            Lk.Hof        9475  BY  Ländliches Umland  6.43        3.11
1  Mansfeld-Südharz       15087  ST    Ländlicher Raum  6.37        3.11
2               Hof        9464  BY  Ländliches Umland  6.36        3.11
3              Harz       15085  ST  Ländliches Umland  6.16        3.11
4             Börde       15083  ST    Ländlicher Raum  6.14        3.11
Length of the dataset:  402
```

Out[44]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1187883c8>]],
      dtype=object)
```

```
<Figure size 432x288 with 0 Axes>
```



One can see, that the dataset is of length 402, and it contains name and ID of county (**Region**,**Regions-ID**), state ID (**KV**), type of county (**Kreistyp**), normalized value of hearth failures prevalence (**Wert**), and the prevalence average (**Bundeswert**). The histogram of the normalized value of hearth failures prevalence is shown above. The data seems to have enough variability to be used in the study.

In [ ]: