Analysis of Dataset

We chose REFIT, a dataset of - 20 households (refit.buildings), recorded between - Oct 2013 - Jun 2015, with a sampling rate of - 8 seconds interval and - 9 possible appliances in each household (refit.elecs(): some of them are missing in house 12, 13 and 20).

The included raw electrical consumption data in Watt were collected during a project regarding research in the field of energy conservation and advanced energy services. More information can be found in:

https://pureportal.strath.ac.uk/en/datasets/refit-electrical-load-measurements

https://pure.strath.ac.uk/ws/portalfiles/portal/45410335/REFITREADME.txt

https://www.nature.com/articles/sdata2016122

import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
from nilmtk import DataSet

DATA\_PATH = '.\data\REFIT.h5'  
refit = DataSet(DATA\_PATH)

type(refit)

nilmtk.dataset.DataSet

How does our dataset look like?

### Number of Available Buildings

# easy way to find out the number of households  
refit.buildings

OrderedDict([(1, Building(instance=1, dataset='REFIT')),  
 (10, Building(instance=10, dataset='REFIT')),  
 (11, Building(instance=11, dataset='REFIT')),  
 (12, Building(instance=12, dataset='REFIT')),  
 (13, Building(instance=13, dataset='REFIT')),  
 (14, Building(instance=14, dataset='REFIT')),  
 (15, Building(instance=15, dataset='REFIT')),  
 (16, Building(instance=16, dataset='REFIT')),  
 (17, Building(instance=17, dataset='REFIT')),  
 (18, Building(instance=18, dataset='REFIT')),  
 (19, Building(instance=19, dataset='REFIT')),  
 (2, Building(instance=2, dataset='REFIT')),  
 (20, Building(instance=20, dataset='REFIT')),  
 (3, Building(instance=3, dataset='REFIT')),  
 (4, Building(instance=4, dataset='REFIT')),  
 (5, Building(instance=5, dataset='REFIT')),  
 (6, Building(instance=6, dataset='REFIT')),  
 (7, Building(instance=7, dataset='REFIT')),  
 (8, Building(instance=8, dataset='REFIT')),  
 (9, Building(instance=9, dataset='REFIT'))])

There are 20 buildings in the refit-Dataset.

### Available Appliances

# electric meters and the appliances for each household (-> 9?!) were checked with  
refit.elecs()  
#...

There seems to be some missing appliances in building 12, 13 and 20…

### Characteristics of the Power Consumption

Now let’s go more in detail: For analysing the dataset we choose two different time windows, both 4 months long - one is set during spring/summer 2014, the other one during autumn/winter 2014/15. The function describe() results in a first overview of all households:

refit.set\_window(start='2014-04-01', end='2014-07-31')  
refit.describe()

(Table output omitted due to very wide format)

refit.set\_window(start='2014-10-01', end='2015-01-31')  
refit.describe()

(Table output omitted due to very wide format)

First impression: not all households have the same quality. Our focus is on duration and uptime, but also on dropout rates and correlation. There are differences during summertime and wintertime, too.

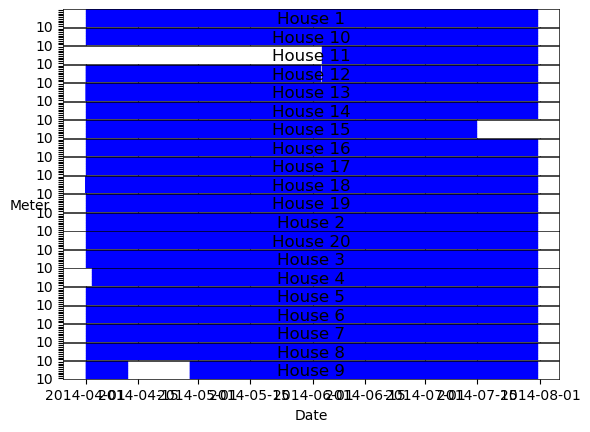
‘Proportion of energy submetered’ is quite low in all houses, therefore the amount of noise is quite high for all of them (but looking on the measured appliances itself, it seems that there are some important ones unmeasured). For the tasks of this Case Study - looking on appliances separately - this kind of noise shouldn’t affect the results…

Although we have an impression, which households could fit for our project, we do some more investigation to learn about our data.

# back to the first time window  
refit.set\_window(start='2014-04-01', end='2014-07-31')

refit.plot\_good\_sections()

c:\Users\Chris\.conda\envs\case-study\lib\site-packages\pandas\plotting\\_matplotlib\converter.py:103: FutureWarning: Using an implicitly registered datetime converter for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will require you to explicitly register matplotlib converters.  
  
To register the converters:  
 >>> from pandas.plotting import register\_matplotlib\_converters  
 >>> register\_matplotlib\_converters()  
 warnings.warn(msg, FutureWarning)  
c:\Users\Chris\.conda\envs\case-study\lib\site-packages\nilmtk\dataset.py:133: UserWarning: Tight layout not applied. tight\_layout cannot make axes height small enough to accommodate all axes decorations  
 plt.tight\_layout()

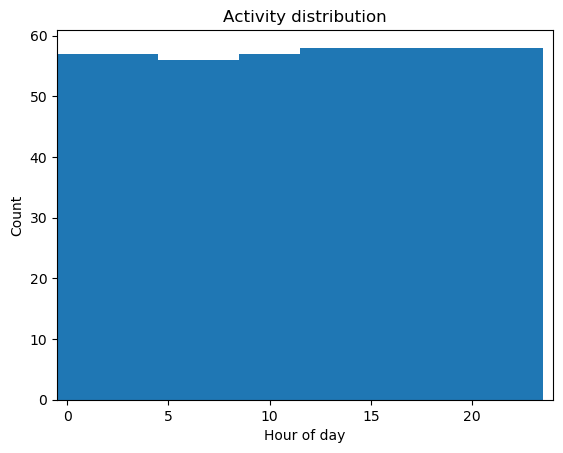


This plot strengthens the considerations about the problems of some households during the first time frame, which we saw also in the coresponding description table before. For example: 64 days are missing during the measurement of house 11. Let’s check the activity histogram:

refit.buildings[11].elec.plot\_activity\_histogram()

Loading data for meter ElecMeterID(instance=10, building=11, dataset='REFIT')   
Done loading data all meters for this chunk.

<matplotlib.axes.\_subplots.AxesSubplot at 0x170c8099040>

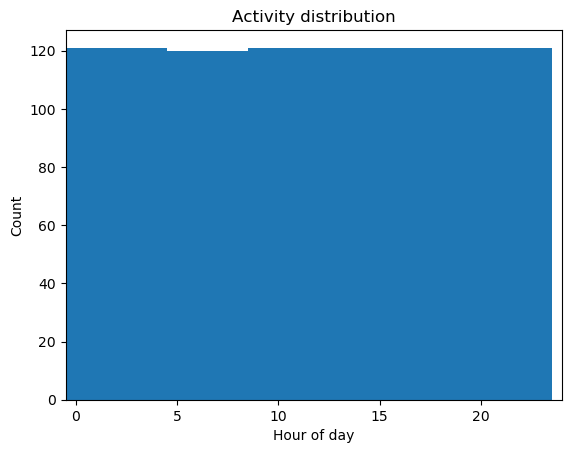


As we can see, the measurement amounted to only 58 days. Comparing to another building:

refit.buildings[5].elec.plot\_activity\_histogram()

Loading data for meter ElecMeterID(instance=10, building=5, dataset='REFIT')   
Done loading data all meters for this chunk.

<matplotlib.axes.\_subplots.AxesSubplot at 0x170c84dee50>



The gap percentage is near zero for some of the buildings (e.g. house 5, 7, 10, 14), but higher for houses like 15, 9 and special for house 11 (good to see - for example in histograms above for house 11 and 5 - by values for ‘Count’).

Let’s check out the dropout-rate for some buildings too. First we will ignore the gaps, and in the second step we won’t ignore them. This will help us to decide on the right buildings where the measurements are good.

refit.buildings[1].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=1, dataset='REFIT') ...

0.0555345075877723

refit.buildings[1].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=1, dataset='REFIT') ...

0.040622436783764204

refit.buildings[5].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=5, dataset='REFIT') ...

0.0005233707743938547

refit.buildings[5].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=5, dataset='REFIT') ...

0.0

refit.buildings[7].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=7, dataset='REFIT') ...

0.010221714327823239

refit.buildings[7].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=7, dataset='REFIT') ...

0.0

refit.buildings[9].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=9, dataset='REFIT') ...

0.0020554369924269894

refit.buildings[9].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=9, dataset='REFIT') ...

0.04029833557194373

refit.buildings[12].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=12, dataset='REFIT') ...

6.508721234459649e-06

refit.buildings[12].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=12, dataset='REFIT') ...

0.0

refit.buildings[14].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=14, dataset='REFIT') ...

0.003631467858543161

refit.buildings[14].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=14, dataset='REFIT') ...

0.0

refit.buildings[20].elec.dropout\_rate()

Calculating dropout\_rate for ElecMeterID(instance=10, building=20, dataset='REFIT') ...

0.08217909797525261

refit.buildings[20].elec.dropout\_rate(ignore\_gaps=False)

Calculating dropout\_rate for ElecMeterID(instance=10, building=20, dataset='REFIT') ...

0.07861880099230989

Our previous research gave us a first impression of the households. Now we will focus on four households, which seem to be appropriate: 5, 7 and 14.

# The buildings 5, 7 and 14

First we will look at the submeters, then we will calculate the total energy and finally look at the plots for each building.

refit.buildings[5].elec.submeters()

MeterGroup(meters=  
 ElecMeter(instance=2, building=5, dataset='REFIT', appliances=[Appliance(type='fridge freezer', instance=1)])  
 ElecMeter(instance=3, building=5, dataset='REFIT', appliances=[Appliance(type='tumble dryer', instance=1)])  
 ElecMeter(instance=4, building=5, dataset='REFIT', appliances=[Appliance(type='washing machine', instance=1)])  
 ElecMeter(instance=5, building=5, dataset='REFIT', appliances=[Appliance(type='dish washer', instance=1)])  
 ElecMeter(instance=6, building=5, dataset='REFIT', appliances=[Appliance(type='computer', instance=1)])  
 ElecMeter(instance=7, building=5, dataset='REFIT', appliances=[Appliance(type='television', instance=1)])  
 ElecMeter(instance=8, building=5, dataset='REFIT', appliances=[Appliance(type='microwave', instance=1)])  
 ElecMeter(instance=9, building=5, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)])  
 ElecMeter(instance=10, building=5, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)])  
)

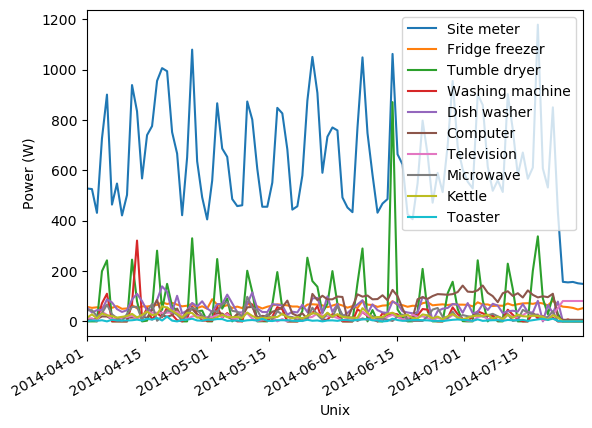
refit.buildings[5].elec.total\_energy()

Calculating total\_energy for ElecMeterID(instance=10, building=5, dataset='REFIT') ...

active 2819.161941  
dtype: float64

refit.buildings[5].elec.plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170c90c9bb0>



refit.buildings[7].elec.submeters()

MeterGroup(meters=  
 ElecMeter(instance=2, building=7, dataset='REFIT', appliances=[Appliance(type='fridge', instance=1)])  
 ElecMeter(instance=3, building=7, dataset='REFIT', appliances=[Appliance(type='freezer', instance=1)])  
 ElecMeter(instance=4, building=7, dataset='REFIT', appliances=[Appliance(type='freezer', instance=2)])  
 ElecMeter(instance=5, building=7, dataset='REFIT', appliances=[Appliance(type='tumble dryer', instance=1)])  
 ElecMeter(instance=6, building=7, dataset='REFIT', appliances=[Appliance(type='washing machine', instance=1)])  
 ElecMeter(instance=7, building=7, dataset='REFIT', appliances=[Appliance(type='dish washer', instance=1)])  
 ElecMeter(instance=8, building=7, dataset='REFIT', appliances=[Appliance(type='television', instance=1)])  
 ElecMeter(instance=9, building=7, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)])  
 ElecMeter(instance=10, building=7, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)])  
)

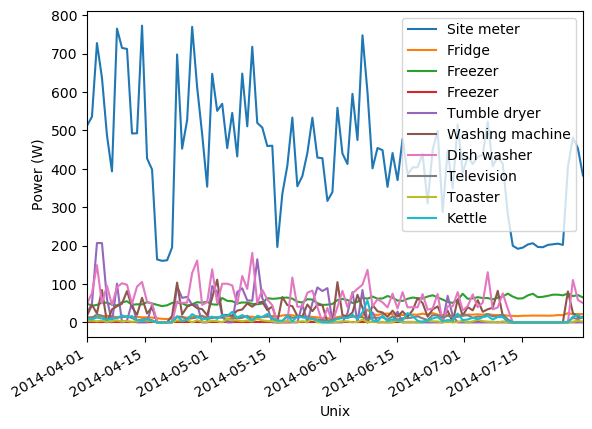
refit.buildings[7].elec.total\_energy()

Calculating total\_energy for ElecMeterID(instance=10, building=7, dataset='REFIT') ...

active 1836.118926  
dtype: float64

refit.buildings[7].elec.plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170c8b49850>



refit.buildings[14].elec.submeters()

MeterGroup(meters=  
 ElecMeter(instance=2, building=14, dataset='REFIT', appliances=[Appliance(type='fridge freezer', instance=1)])  
 ElecMeter(instance=3, building=14, dataset='REFIT', appliances=[Appliance(type='tumble dryer', instance=1)])  
 ElecMeter(instance=4, building=14, dataset='REFIT', appliances=[Appliance(type='washing machine', instance=1)])  
 ElecMeter(instance=5, building=14, dataset='REFIT', appliances=[Appliance(type='dish washer', instance=1)])  
 ElecMeter(instance=6, building=14, dataset='REFIT', appliances=[Appliance(type='computer', instance=1)])  
 ElecMeter(instance=7, building=14, dataset='REFIT', appliances=[Appliance(type='television', instance=1)])  
 ElecMeter(instance=8, building=14, dataset='REFIT', appliances=[Appliance(type='microwave', instance=1)])  
 ElecMeter(instance=9, building=14, dataset='REFIT', appliances=[Appliance(type='audio system', instance=1)])  
 ElecMeter(instance=10, building=14, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)])  
)

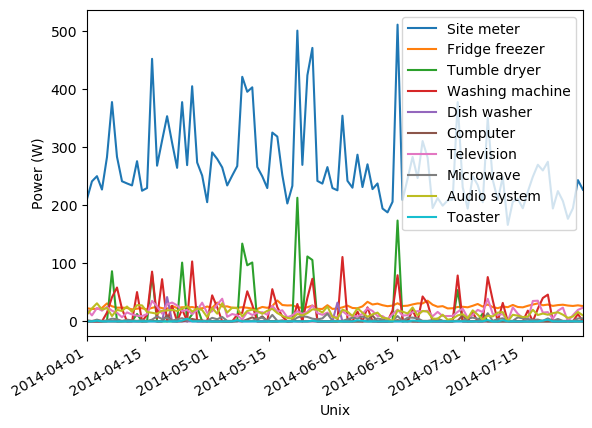
refit.buildings[14].elec.total\_energy()

Calculating total\_energy for ElecMeterID(instance=10, building=14, dataset='REFIT') ...

active 4038.971498  
dtype: float64

refit.buildings[14].elec.plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170c9223b50>



For each of these buildings it would be possible to go more in detail - for example household 5:

refit.buildings[5].elec.appliances

[Appliance(type='television', instance=1),  
 Appliance(type='microwave', instance=1),  
 Appliance(type='kettle', instance=1),  
 Appliance(type='dish washer', instance=1),  
 Appliance(type='washing machine', instance=1),  
 Appliance(type='toaster', instance=1),  
 Appliance(type='tumble dryer', instance=1),  
 Appliance(type='computer', instance=1),  
 Appliance(type='fridge freezer', instance=1)]

refit.buildings[5].elec.submeters().energy\_per\_meter()

9/9 ElecMeter(instance=10, building=5, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)]))))1)])

|  | (2, 5, REFIT) | (3, 5, REFIT) | (4, 5, REFIT) | (5, 5, REFIT) | (6, 5, REFIT) | (7, 5, REFIT) | (8, 5, REFIT) | (9, 5, REFIT) | (10, 5, REFIT) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| active | 182.598021 | 213.850364 | 66.053814 | 155.131404 | 173.101041 | 59.511101 | 72.94899 | 59.675328 | 12.943093 |
| apparent | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| reactive | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

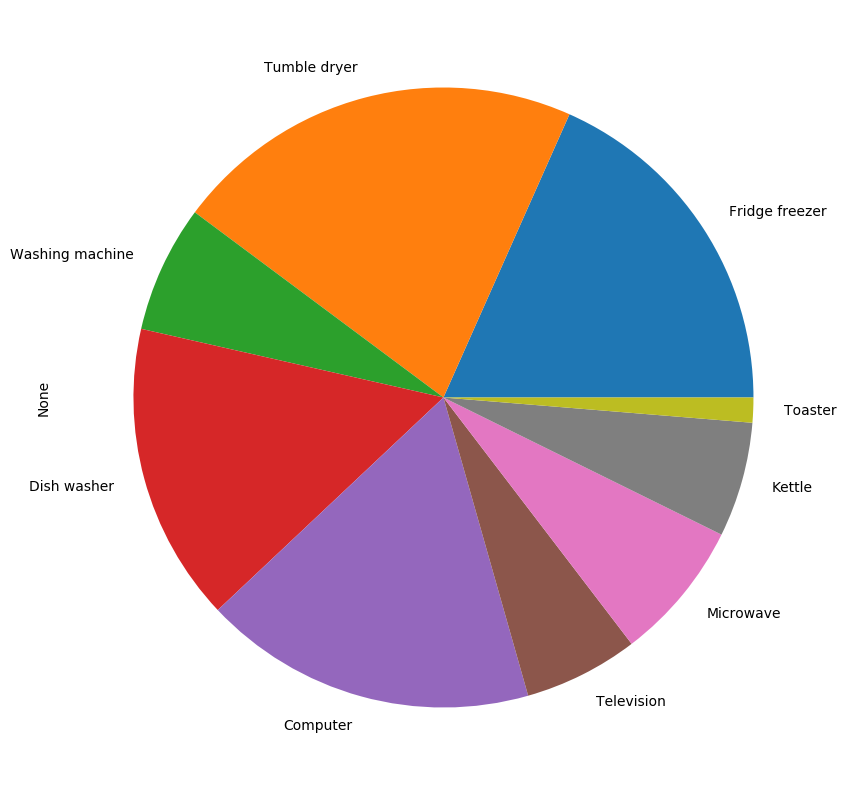
But more interesting are the proportions of the single appliances. Appliances with high power values shadow the smaller ones…

Let us look on our three houses:

fraction\_5 = refit.buildings[5].elec.submeters().fraction\_per\_meter().dropna()  
labels\_5 = refit.buildings[5].elec.get\_labels(fraction\_5.index)  
plt.figure(figsize=(10,30))  
fraction\_5.plot(kind='pie', labels=labels\_5)

9/9 ElecMeter(instance=10, building=5, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)]))))1)])

<matplotlib.axes.\_subplots.AxesSubplot at 0x2819ab8ec70>



refit.buildings[5].elec.submeters().fraction\_per\_meter()

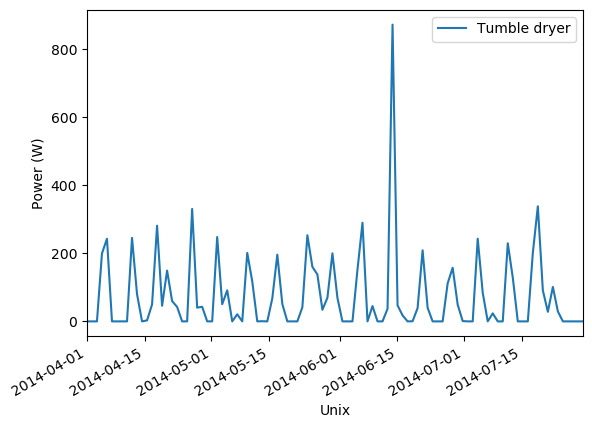
9/9 ElecMeter(instance=10, building=5, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)]))))1)])

(2, 5, REFIT) 0.183366  
(3, 5, REFIT) 0.214749  
(4, 5, REFIT) 0.066332  
(5, 5, REFIT) 0.155784  
(6, 5, REFIT) 0.173829  
(7, 5, REFIT) 0.059761  
(8, 5, REFIT) 0.073256  
(9, 5, REFIT) 0.059926  
(10, 5, REFIT) 0.012998  
dtype: float64

Interesting appliances of building 5: tumble dryer (instance 3) and computer (instance 6).

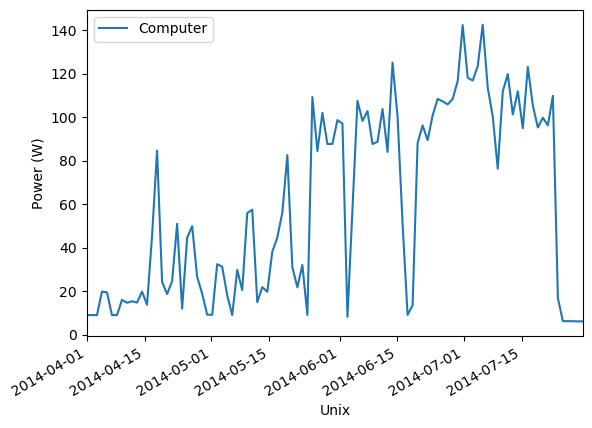
refit.buildings[5].elec[3].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170de7e0bb0>



refit.buildings[5].elec[6].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

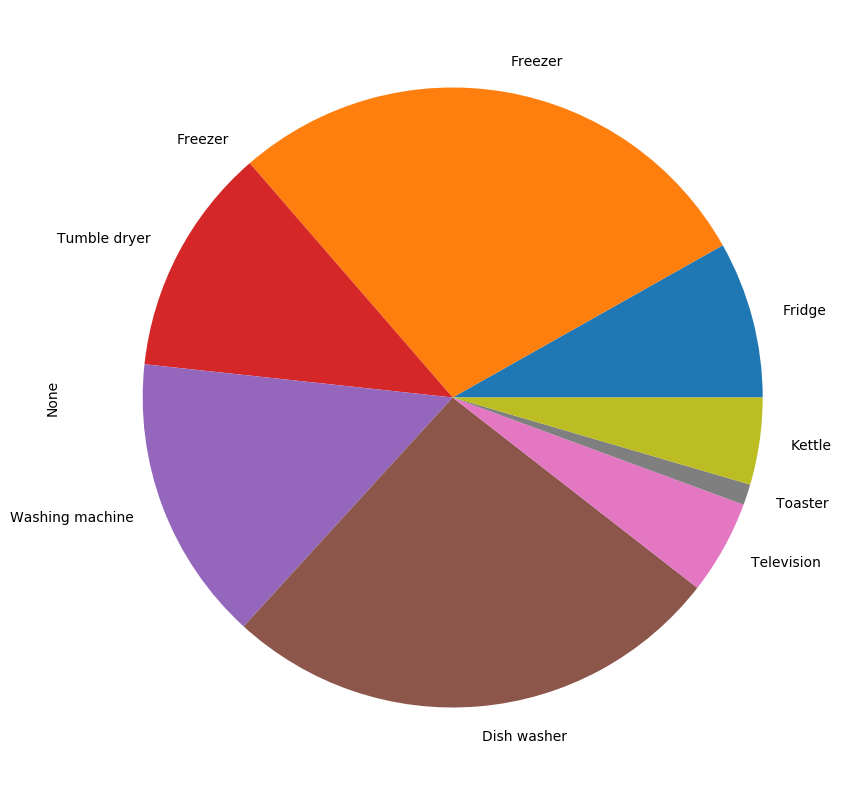
<matplotlib.axes.\_subplots.AxesSubplot at 0x170ca9d6e50>



fraction\_7 = refit.buildings[7].elec.submeters().fraction\_per\_meter().dropna()  
labels\_7 = refit.buildings[7].elec.get\_labels(fraction\_7.index)  
plt.figure(figsize=(10,30))  
fraction\_7.plot(kind='pie', labels=labels\_7)

9/9 ElecMeter(instance=10, building=7, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)]))]))1)])

<matplotlib.axes.\_subplots.AxesSubplot at 0x2819ac0ec40>



refit.buildings[7].elec.submeters().fraction\_per\_meter()

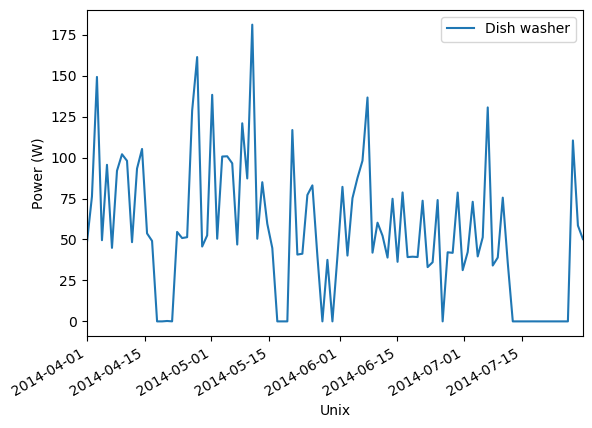
9/9 ElecMeter(instance=10, building=7, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)]))]))1)])

(2, 7, REFIT) 8.169317e-02  
(3, 7, REFIT) 2.819017e-01  
(4, 7, REFIT) 7.059968e-09  
(5, 7, REFIT) 1.192980e-01  
(6, 7, REFIT) 1.493387e-01  
(7, 7, REFIT) 2.625398e-01  
(8, 7, REFIT) 4.903015e-02  
(9, 7, REFIT) 1.086364e-02  
(10, 7, REFIT) 4.533493e-02  
dtype: float64

For us choosing are dataset interessting appliances of household 7 are: the dish washer (instance 7) and the kettle (instance 10).

refit.buildings[7].elec[7].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

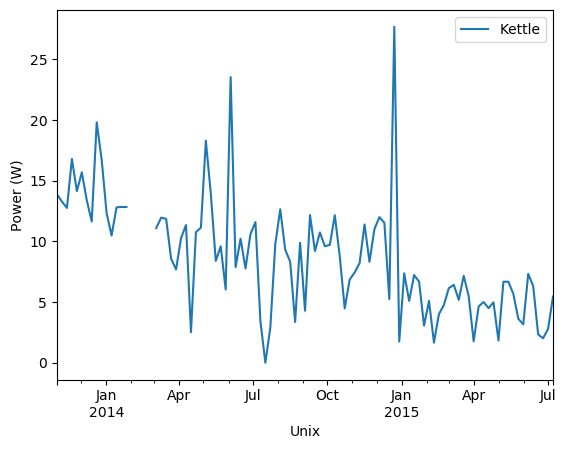
<matplotlib.axes.\_subplots.AxesSubplot at 0x170cabf1160>



refit.buildings[7].elec[10].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

c:\Users\Chris\.conda\envs\case-study\lib\site-packages\pandas\core\arrays\datetimes.py:1266: UserWarning: Converting to PeriodArray/Index representation will drop timezone information.  
 warnings.warn(

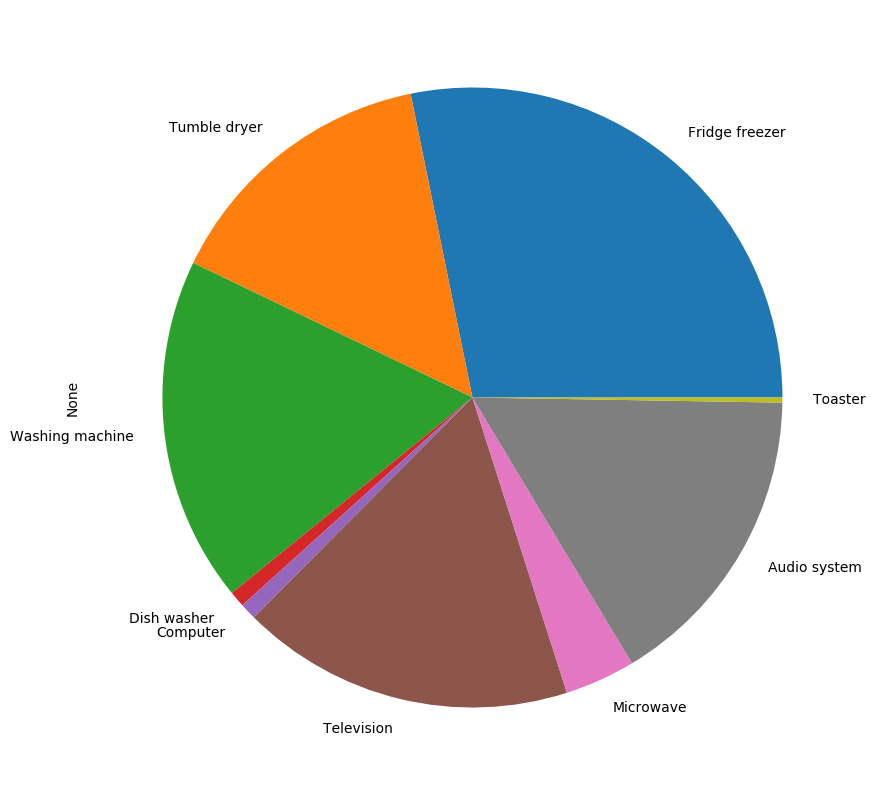
<matplotlib.axes.\_subplots.AxesSubplot at 0x15117c027c0>



fraction\_14 = refit.buildings[14].elec.submeters().fraction\_per\_meter().dropna()  
labels\_14 = refit.buildings[14].elec.get\_labels(fraction\_14.index)  
plt.figure(figsize=(10,30))  
fraction\_14.plot(kind='pie', labels=labels\_14)

9/9 ElecMeter(instance=10, building=14, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)])1)]))])

<matplotlib.axes.\_subplots.AxesSubplot at 0x2819d224250>



refit.buildings[14].elec.submeters().fraction\_per\_meter()

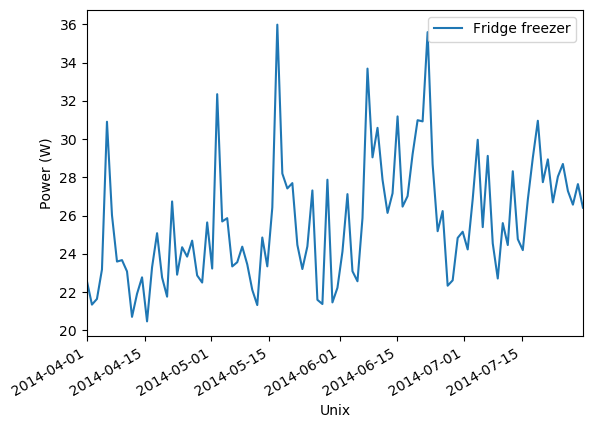
9/9 ElecMeter(instance=10, building=14, dataset='REFIT', appliances=[Appliance(type='toaster', instance=1)])1)]))])

(2, 14, REFIT) 0.281964  
(3, 14, REFIT) 0.146541  
(4, 14, REFIT) 0.180122  
(5, 14, REFIT) 0.008278  
(6, 14, REFIT) 0.008606  
(7, 14, REFIT) 0.173769  
(8, 14, REFIT) 0.036765  
(9, 14, REFIT) 0.161324  
(10, 14, REFIT) 0.002631  
dtype: float64

Building 14 shows a different plot because of 3 very small parts. Appliances with high proportion are instance 2 - the fridge freezer, instance 4 - the washing machine, instance 7 - the television and instance 9 - the audio system.

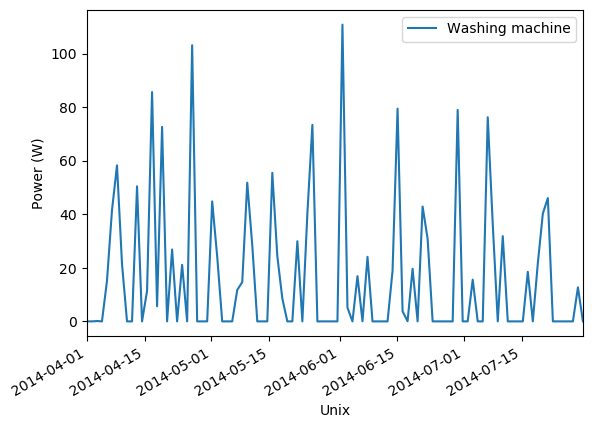
refit.buildings[14].elec[2].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170cab62d00>



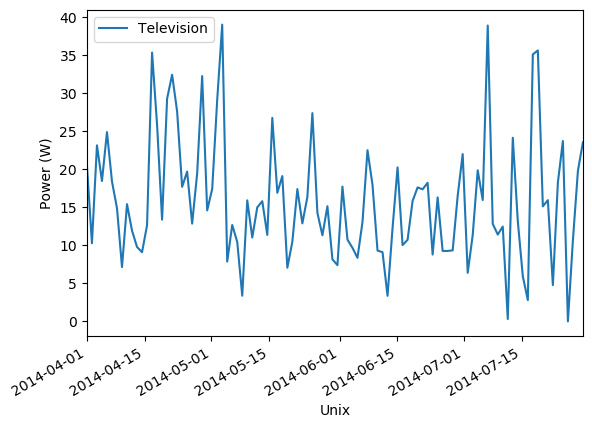
refit.buildings[14].elec[4].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170cadbe970>



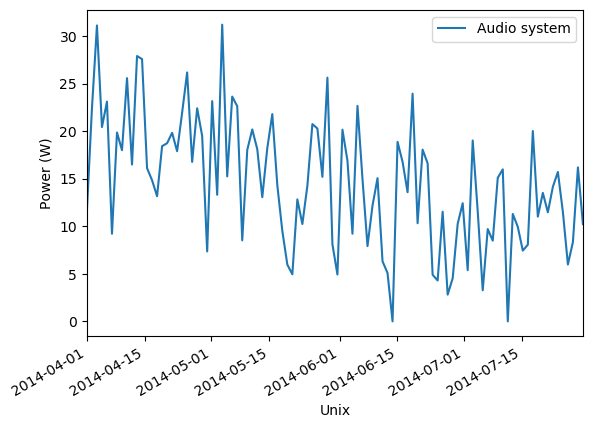
refit.buildings[14].elec[7].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170cae3ad00>



refit.buildings[14].elec[9].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170cae8d0d0>



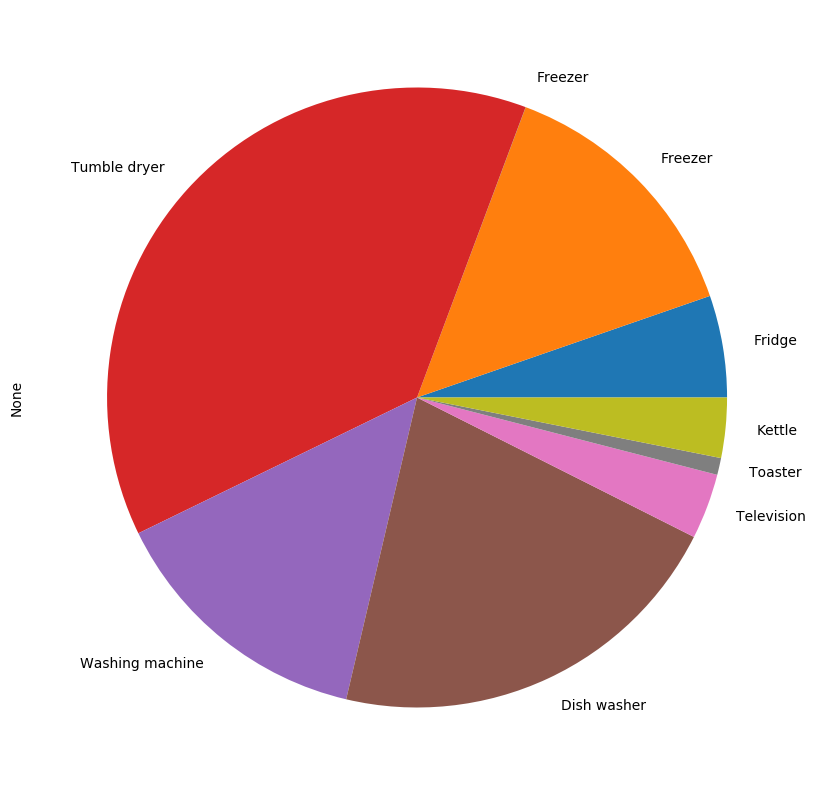
We also had a look on all the appliances during wintertime, but just household 7 shows an interesting change in the proportion of its items:

refit.set\_window(start='2014-10-01', end='2015-01-31')

fraction\_7 = refit.buildings[7].elec.submeters().fraction\_per\_meter().dropna()  
labels\_7 = refit.buildings[7].elec.get\_labels(fraction\_7.index)  
plt.figure(figsize=(10,30))  
fraction\_7.plot(kind='pie', labels=labels\_7)

9/9 ElecMeter(instance=10, building=7, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)]))]))1)])

<matplotlib.axes.\_subplots.AxesSubplot at 0x2819f545430>



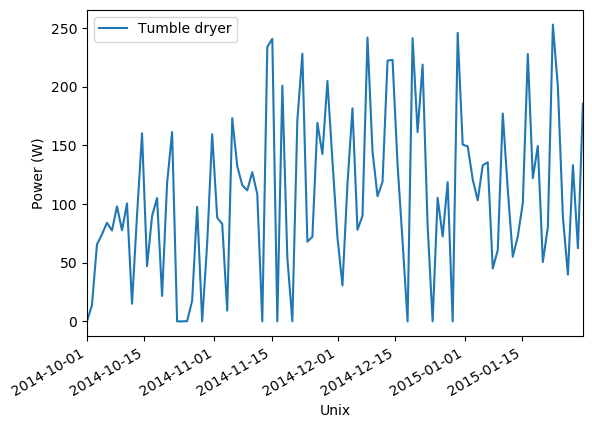
refit.buildings[7].elec.submeters().fraction\_per\_meter()

9/9 ElecMeter(instance=10, building=7, dataset='REFIT', appliances=[Appliance(type='kettle', instance=1)]))]))1)])

(2, 7, REFIT) 0.053146  
(3, 7, REFIT) 0.139833  
(4, 7, REFIT) 0.000077  
(5, 7, REFIT) 0.379108  
(6, 7, REFIT) 0.141110  
(7, 7, REFIT) 0.212471  
(8, 7, REFIT) 0.034119  
(9, 7, REFIT) 0.008816  
(10, 7, REFIT) 0.031320  
dtype: float64

refit.buildings[7].elec[5].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170caf3cdf0>



The tumble dryer (instance 5) was used very intensive in building 7 during October until end of January.

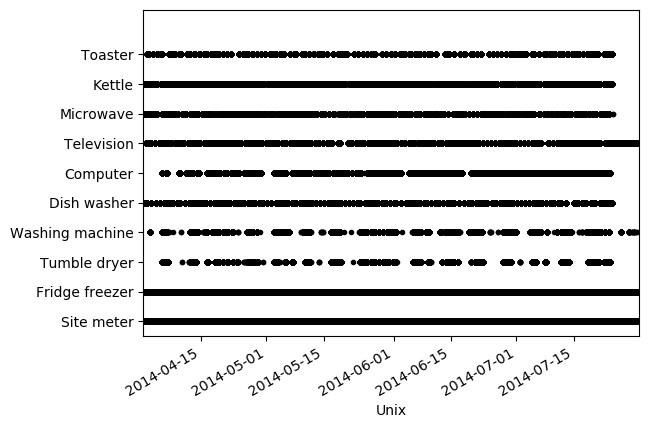
But lets go back to summertime:

refit.set\_window(start='2014-04-01', end='2014-07-31')

We will focus on building 5 again…

refit.buildings[5].elec.plot\_when\_on(on\_power\_threshold=40)

<matplotlib.axes.\_subplots.AxesSubplot at 0x2819a64fd90>

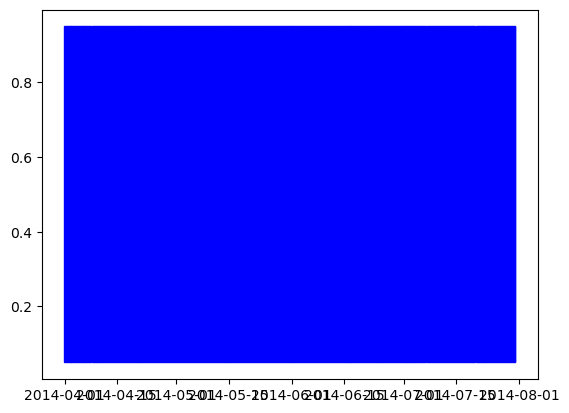


…and in detail on the computer…

refit.buildings[5].elec['computer'].good\_sections(full\_results=True).plot()

WARNING: search terms match 1 appliances. Instance 0 was selected

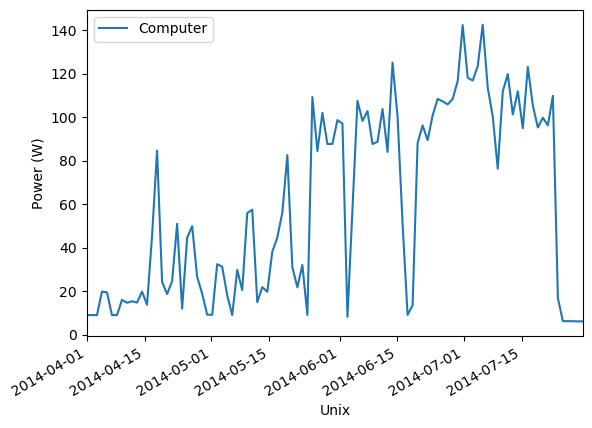
<matplotlib.axes.\_subplots.AxesSubplot at 0x2819ce74e50>



This looks fine.

refit.buildings[5].elec[6].plot(ax=None, timeframe=None, plot\_legend=True, unit='W', width=100)

<matplotlib.axes.\_subplots.AxesSubplot at 0x170caa74730>



Checking the other appliances in a similiar way we are now ready to start with modelling…