Preference Learning - Part IV (warm)

1. Principle Component Analysis (PCA)

2. K-mean clustering with/without normalization

3. Multiple regression

4. Groups' Price Responsiveness

In [2]:
```
%matplotlib inline
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import json
import seaborn as sns
import math
from pandas.io.json import json_normalize #package for flattening json in pandas df
import matplotlib.pyplot as plt
from datetime import date, timedelta, datetime
```

1. Principle Component Analysis

We first focus on the non-event data. For each hour,

(1) the first approach is we do principle component analysis and get several explanable top PCs and then use the top PCs to do K-mean clustering to find out groups;

(2) the second approach is forget about PCA, and start by drawing histogram of distribution on a temp, consumption, occurence 3D space, and using GMM to fit the data and find out the classes.

In [ ]:
```
# Import all preprocessed data necessary for the analysis
df_tou1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_tou2013_1h.csv")
df_Ntou1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\Consumption_Ntou2013_1h.csv")
df_wea1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\weather\\LondonWeather2013_interpolated.csv")
df_tariff_1h = pd.read_csv("C:\\Users\\Rockwell\\Desktop\\Paper4\\data_collection\\data_tables\\df_tariff_1h.csv")
```

In [3]:
```
import os
os.getcwd()
```

Out[3]:
```
'/Users/Rockwell/Documents/GitHub/Demand-Response'
```

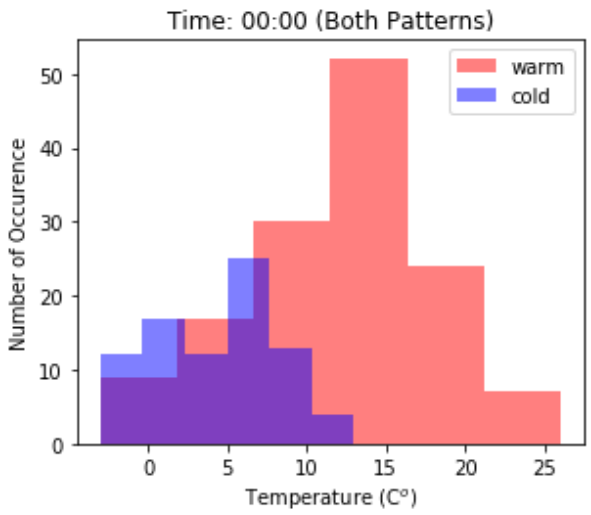
In [4]:
```
# for ios system, import all data necessary for the analysis
df_tou1h = pd.read_csv('/Users/Rockwell/Desktop/PhD/Paper4PreferenceLearning/data/Consumption_tou2013_1h.csv')
df_Ntou1h = pd.read_csv('/Users/Rockwell/Desktop/PhD/Paper4PreferenceLearning/data/Consumption_Ntou2013_1h.csv')
df_wea1h = pd.read_csv('/Users/Rockwell/Desktop/PhD/Paper4PreferenceLearning/data/LondonWeather2013_interpolated.csv')
df_tariff_1h = pd.read_csv('/Users/Rockwell/Desktop/PhD/Paper4PreferenceLearning/data/df_tariff_1h.csv')
```

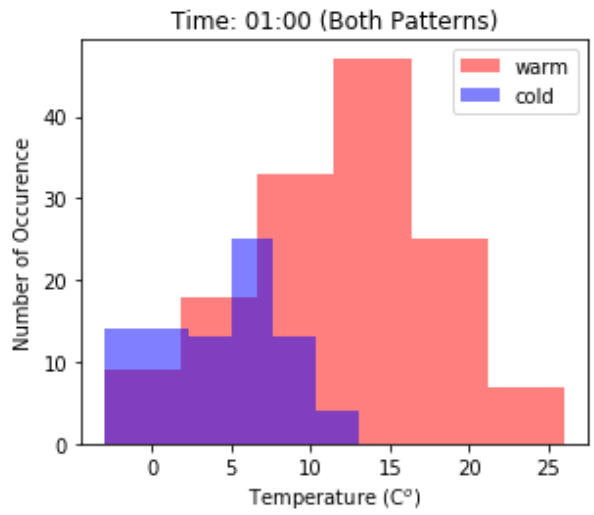
In [5]:
```
# first, create a list of days that belongs to event days
event_days = set()
event_series = df_tariff_1h[df_tariff_1h.Event_tags.notnull()].GMT
for i in event_series:
    event_days.add(datetime.strptime(i[:10], "%Y-%m-%d" ).date()) # add all event dates to the set
df_help = pd.DataFrame(pd.to_datetime(df_tariff_1h.GMT).dt.date) #str to datatime and extract date then make it to dataframe
# df_help[df_help['GMT'].isin(event_days)] #this shows the event days
# we can use ~df_help['GMT'].isin(event_days) to generate any non-flexible period items

# create TOU and non-TOU demand data in non-flexible hours
df_wea1h_nf = df_wea1h[~df_help['GMT'].isin(event_days)]
df_Ntou1h_nf = df_Ntou1h[~df_help['GMT'].isin(event_days)]
df_tou1h_nf = df_tou1h[~df_help['GMT'].isin(event_days)]

# create event data
df_tariff_1h_event = df_tariff_1h[df_tariff_1h.GMT.isin(event_series)]
df_wea1h_event = df_wea1h[df_wea1h.GMT.isin(event_series)]
df_tou1h_event = df_tou1h[df_tou1h.GMT.isin(event_series)]
```

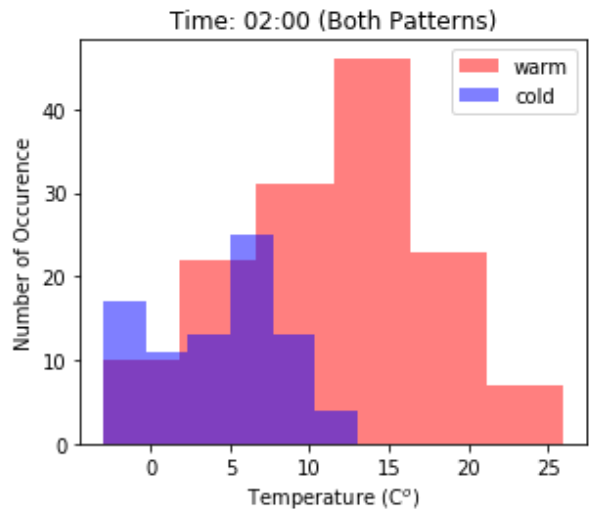
In [6]:
```
# seperate the above data set based on the seasonal effect
# i.e., months of 11, 12, 1, 2, 3 are in a group - cold season
# months of 4, 5, 6, 7, 8, 9, 10 are in another group - warm season
cold_season = [11, 12, 1, 2, 3]
warm_season = [4, 5, 6, 7, 8, 9, 10]
df_help_season = pd.DataFrame(pd.to_datetime(df_wea1h_nf.GMT).dt.month)
df_wea1h_nf_cold = df_wea1h_nf[df_help_season['GMT'].isin(cold_season)]
df_wea1h_nf_warm = df_wea1h_nf[df_help_season['GMT'].isin(warm_season)]
df_Ntou1h_nf_cold = df_Ntou1h_nf[df_help_season['GMT'].isin(cold_season)]
df_Ntou1h_nf_warm = df_Ntou1h_nf[df_help_season['GMT'].isin(warm_season)]
df_tou1h_nf_cold = df_tou1h_nf[df_help_season['GMT'].isin(cold_season)]
df_tou1h_nf_warm = df_tou1h_nf[df_help_season['GMT'].isin(warm_season)]

df_help_season_event = pd.DataFrame(pd.to_datetime(df_wea1h_event.GMT).dt.month)
df_wea1h_event_cold = df_wea1h_event[df_help_season_event['GMT'].isin(cold_season)]
df_wea1h_event_warm = df_wea1h_event[df_help_season_event['GMT'].isin(warm_season)]
df_tou1h_event_cold = df_tou1h_event[df_help_season_event['GMT'].isin(cold_season)]
df_tou1h_event_warm = df_tou1h_event[df_help_season_event['GMT'].isin(warm_season)]
df_tariff_1h_event_cold = df_tariff_1h_event[df_help_season_event['GMT'].isin(cold_season)]
df_tariff_1h_event_warm = df_tariff_1h_event[df_help_season_event['GMT'].isin(warm_season)]
```

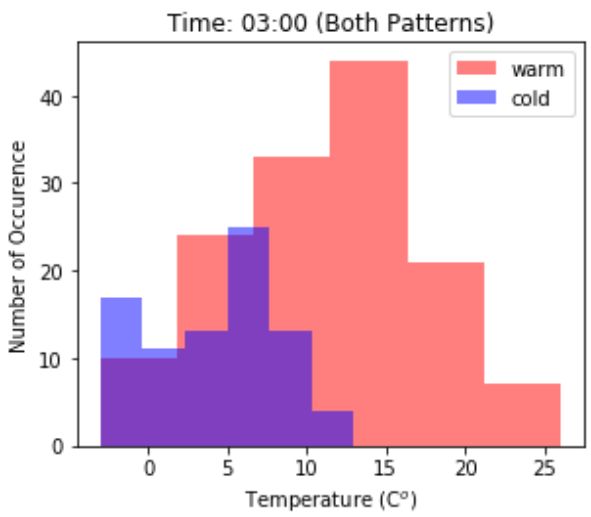
(0) Some basic temperature distribution of the two groups are given below

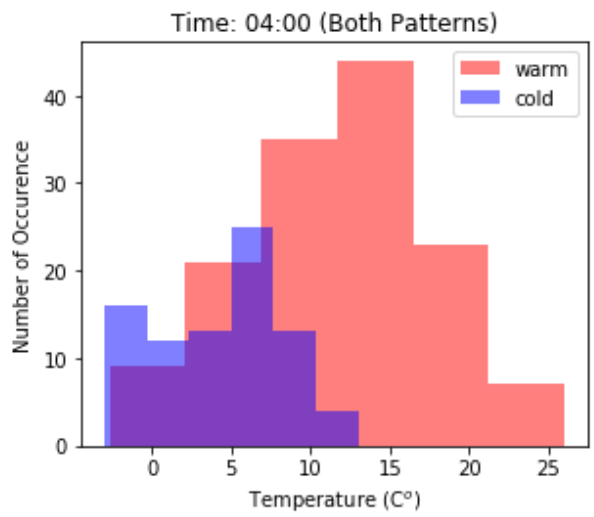
In [6]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.hist(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]['TempC'], bins=6
, color = 'red', alpha = 0.5, label = 'warm')
        ax.hist(df_wea1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]['TempC'], bins=6
, color = 'blue', alpha = 0.5, label = 'cold')
        ax.set_title('Time: 0' + str(i) + ':00' + ' (Both Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
    else:
        ax.hist(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=6, colo
r = 'red', alpha = 0.5, label = 'warm')
        ax.hist(df_wea1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=6, colo
r = 'blue', alpha = 0.5, label = 'cold')
        ax.set_title('Time: ' + str(i) + ':00' + ' (Both Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
plt.tight_layout()
```

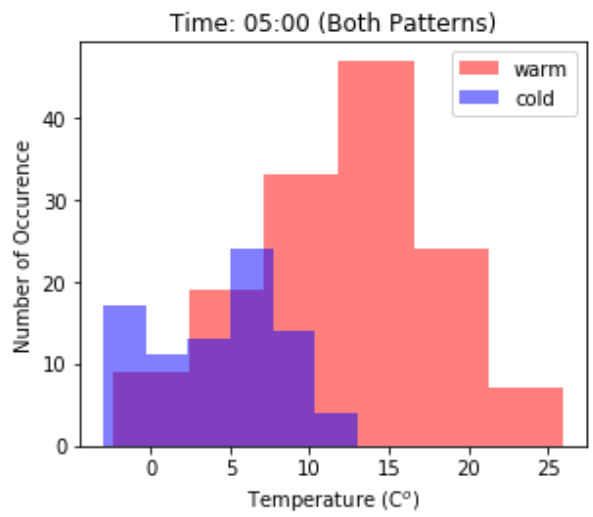

Time: 00:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

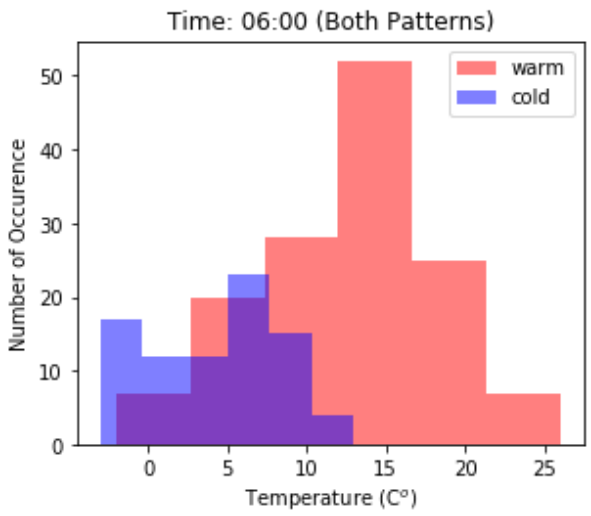

Time: 01:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

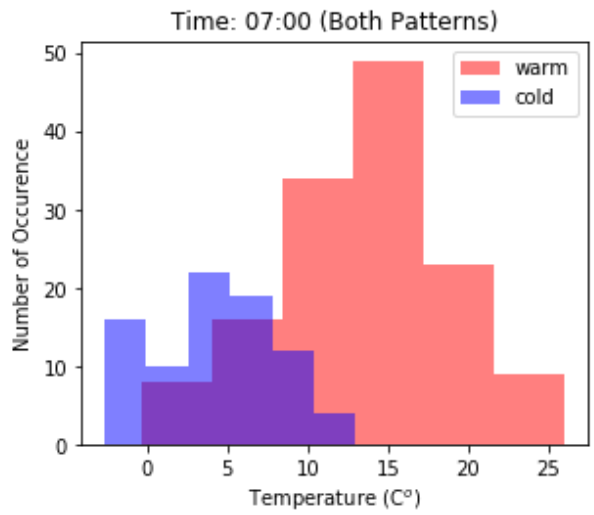

Time: 02:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

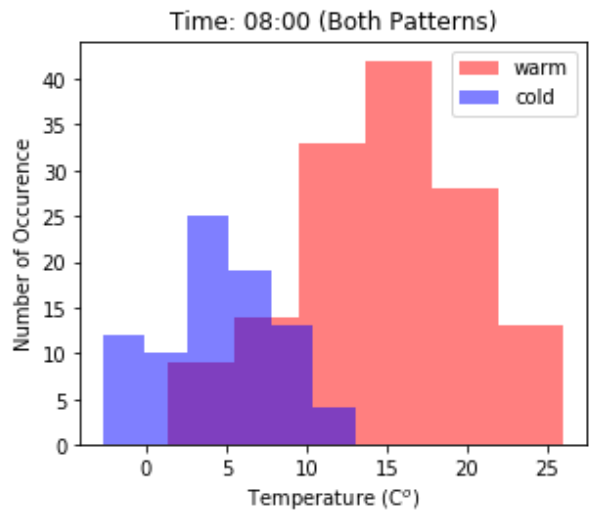

Time: 03:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

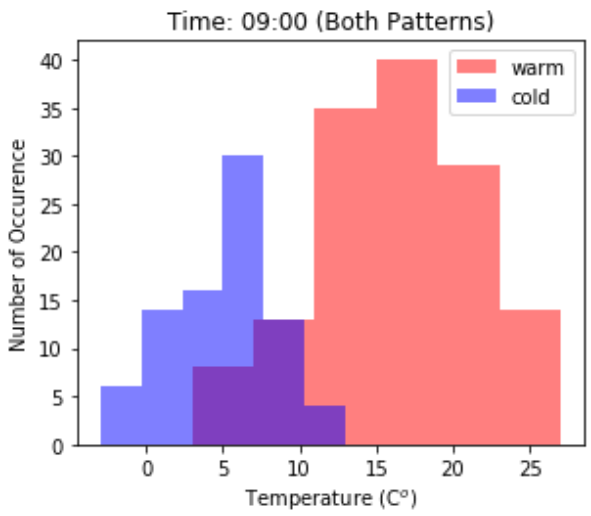

Time: 04:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

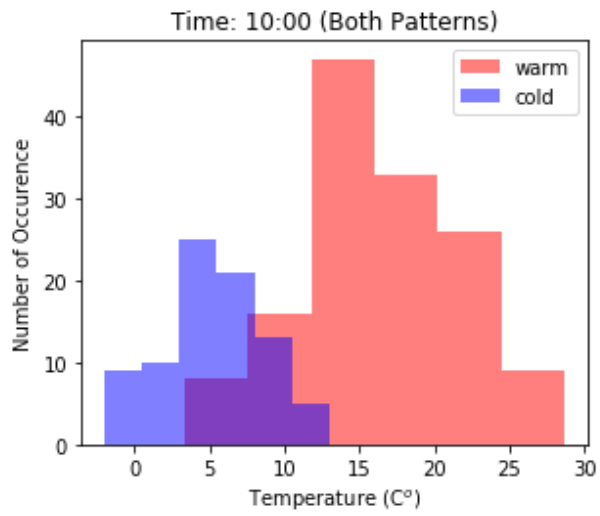

Time: 05:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

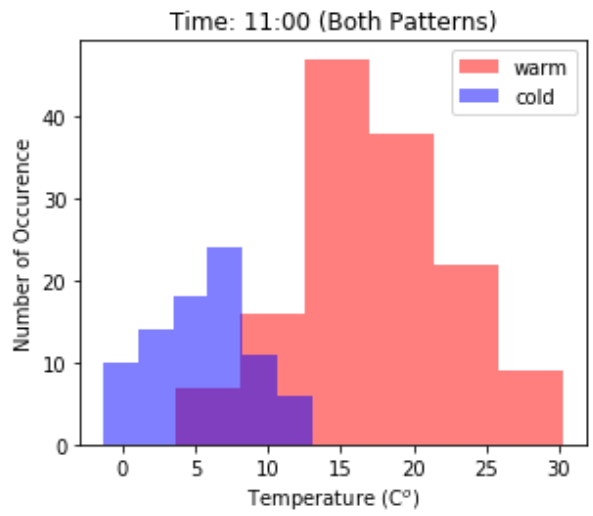

Time: 06:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

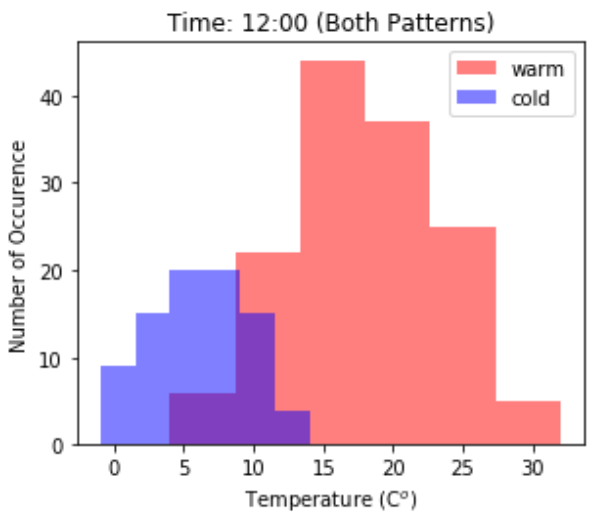

Time: 07:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

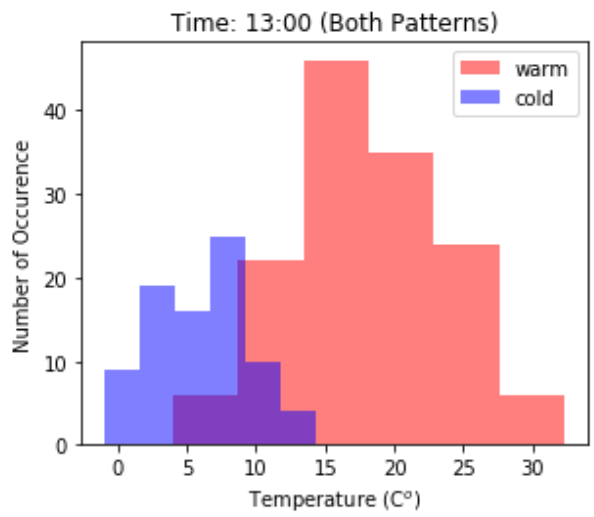

Time: 08:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

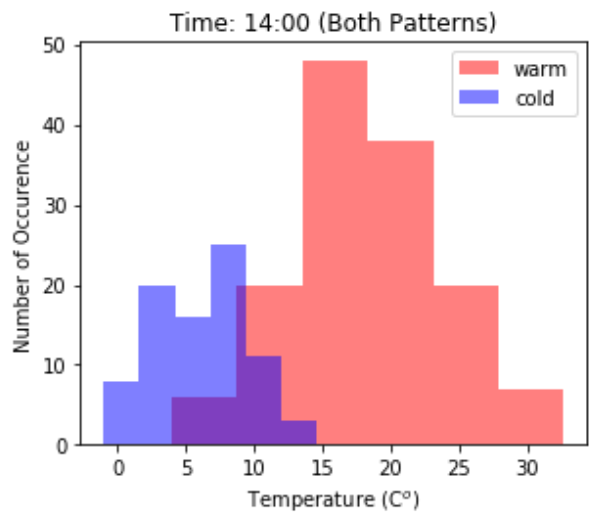

Time: 09:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

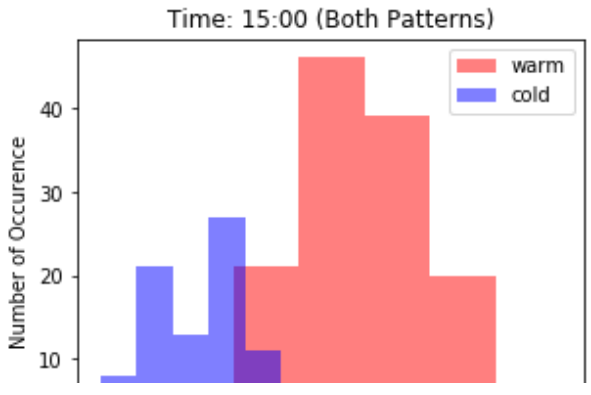

Time: 10:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

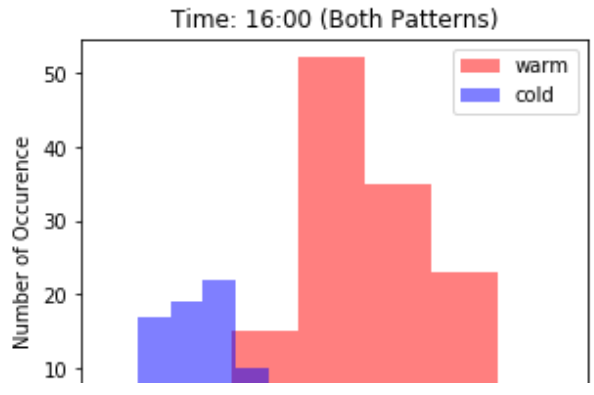

Time: 11:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

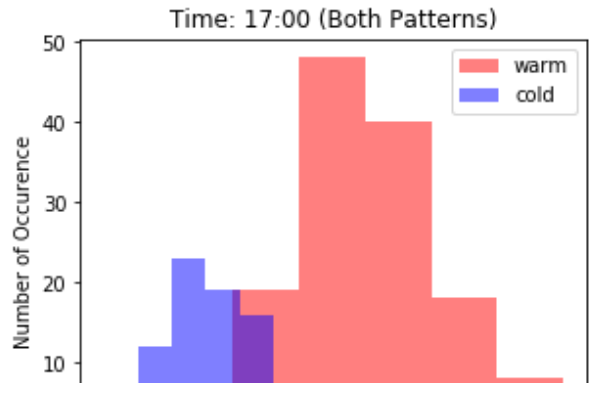

Time: 12:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

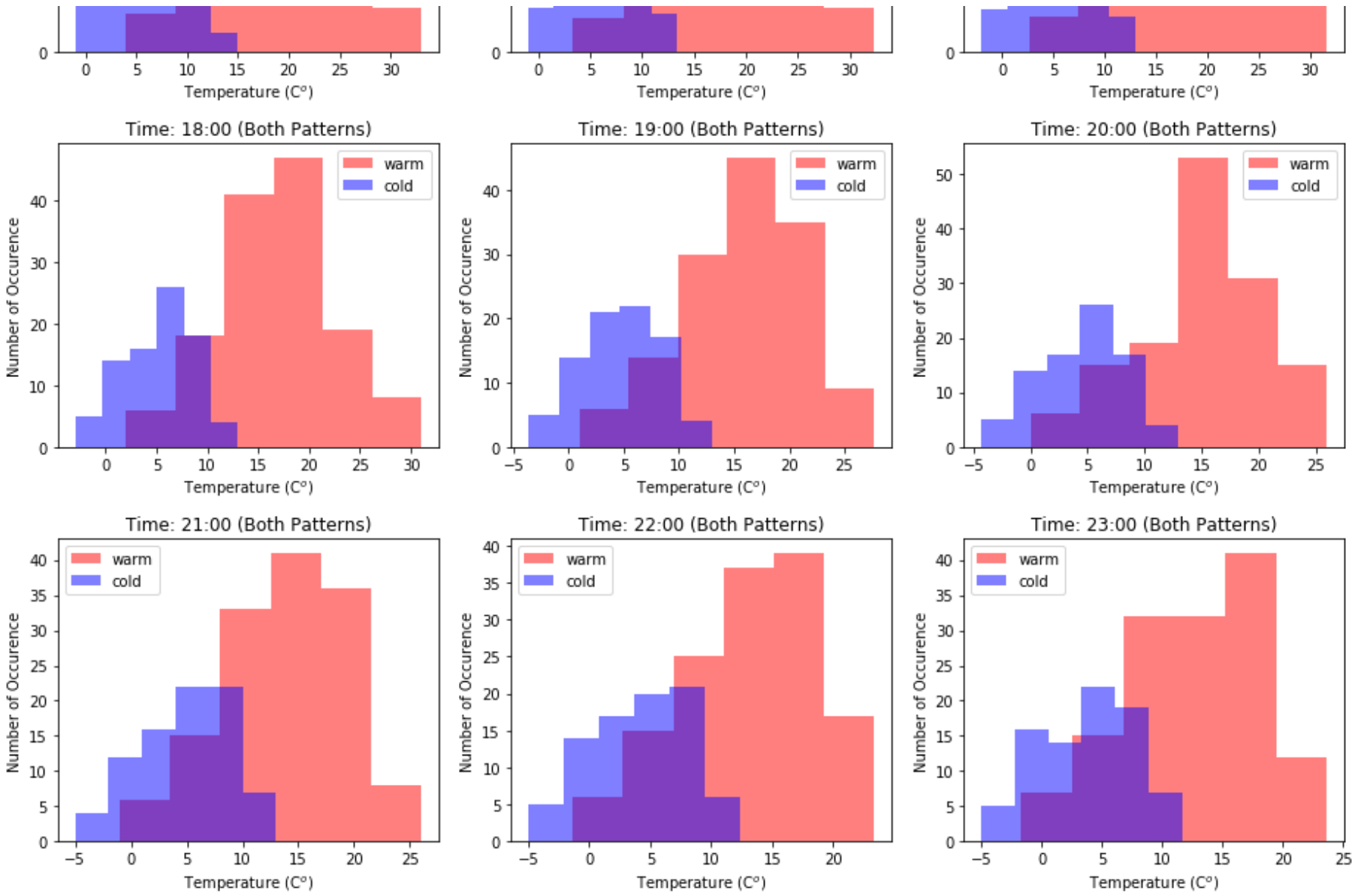

Time: 13:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)


Time: 14:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)


Time: 15:00 (Both Patterns)
warm
cold
Number of Occurence


Time: 16:00 (Both Patterns)
warm
cold
Number of Occurence


Time: 17:00 (Both Patterns)
warm
cold
Number of Occurence



(1) In PCA, we don't consider normalization since all the consumptions have the same unit, we care about the absolute change rather than relative change since DR targets the customer who has largest potential shiftable load instead of largest price-responsiveness rate. Also we have seperate data by hours, so the temp-consumption relationship at different hours has been treated seprately and equally, so the time effect won't be a concern to let us normalize the data.

In [7]:
```
df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('09:00:00')].transpose()
```

Out[7]:

	2169	2193	2217	2241	2289	2313	2361	2385	2433	2481	...	6825	7065	7089	
GMT	2013-04-01 09:00:00	2013-04-02 09:00:00	2013-04-03 09:00:00	2013-04-04 09:00:00	2013-04-06 09:00:00	2013-04-07 09:00:00	2013-04-09 09:00:00	2013-04-10 09:00:00	2013-04-12 09:00:00	2013-04-14 09:00:00	...	2013-10-12 09:00:00	2013-10-22 09:00:00	2013-10-23 09:00:00	2 1 0
D0000	0.11	0.063	0.178	0.528	0.089	0.351	0.218	0.101	0.425	0.079	...	0.072	0.317	0.104	0
D0001	0.197	0.223	0.06	0.293	0.148	0.138	0.152	0.158	0.248	0.162	...	0.125	0.274	0.132	0
D0002	2.787	0.979	1.644	0.402	0.697	1.273	1.715	2.663	1.392	1.196	...	1.309	0.171	0.374	0
D0003	0.148	0.144	0.324	0.143	0.148	1.059	0.147	0.152	0.156	1.187	...	0.16	0.168	0.179	0
D0004	0.307	0.461	0.23	0.429	0.404	0.488	0.455	0.238	0.242	0.912	...	0.279	0.142	0.34	0
D0005	0.343	0.287	0.223	0.422	0.37	0.373	0.347	0.29	0.343	0.244	...	0.476	0.181	0.269	0
D0006	0.84	0.181	0.149	0.702	0.096	0.131	0.25	0.767	0.285	0.098	...	0.194	0.286	0.376	0
D0007	0.342	0.209	0.293	0.787	0.288	0.506	0.322	0.273	0.288	0.559	...	1.012	0.187	0.873	0
D0008	0.104	0.126	0.124	0.288	0.098	0.576	0.171	0.257	0.121	1.002	...	0.258	0.129	0.129	0
D0009	0.169	0.158	0.181	0.164	0.165	0.117	0.055	0.17	0.481	0.49	...	0.237	0.187	0.252	0
D0010	0.001	0	0	0	0	0	0	0.1	0	0.02	...	0.022	0.031	0.023	0
D0011	1.363	0.483	0.465	0.705	0.131	0.415	0.192	0.255	0.235	0.248	...	0.546	0.107	0.101	0
D0012	0.225	0.208	0.236	0.284	0.17	0.175	0.223	0.115	0.282	0.154	...	0.122	0.454	1.081	0
D0013	0.236	0.137	0.124	0.175	0.224	1.559	0.111	0.12	0.131	1.181	...	0.146	1.661	0.2	0
D0014	0.863	1.694	1.654	0.703	0.799	0.477	1.488	1.542	0.511	0.701	...	1.338	0.843	0.635	1
D0015	0.124	0.124	0.12	0.109	1.681	0.22	0.213	0.186	0.127	0.123	...	0.28	0.272	0.179	0
D0016	0.368	0.072	0.495	0.212	0.238	0.291	0.295	0.075	0.068	0.165	...	0.128	0.161	0.28	0
D0017	0.201	0.423	0.339	0.219	1.239	2.217	0.39	0.864	0.24	2.729	...	0.215	0.38	0.378	0
D0018	0.294	0.087	0.208	0.211	0.164	0.132	0.157	0.533	0.155	0.087	...	0.08	0.904	0.137	0
D0019	0.257	0.47	0.308	1.003	0.524	0.398	2.511	0.304	0.322	0.668	...	0.079	0.278	0.181	0
D0020	0.321	0.142	0.582	0.203	0.581	0.2	0.172	0.497	0.183	2.166	...	0.085	0.023	0.665	0
D0021	0	0	0	0	0	0	0	0.031	0.012	0	...	0.047	0	0	0
D0022	0.205	0.222	0.285	0.421	0.38	0.22	0.449	0.408	0.238	0.196	...	0.4	0.407	0.225	0
D0023	0.671	1.012	0.437	0.487	0.423	0.762	0.361	0.304	0.689	0.332	...	0.618	1.714	0.381	0
D0024	0.153	0.071	0.055	0.156	0.262	0.136	0.078	0.057	0.153	0.094	...	0.087	0.062	0.078	0
D0025	1.111	1.102	1.145	1.005	0.551	1.139	0.489	0.544	0.739	0.779	...	0.806	0.332	0.509	1
D0026	0.096	0.093	0.09	0.105	0.078	0.241	0.086	0.11	0.106	0.266	...	0.188	0.102	0.103	0
D0027	0.249	0.159	0.655	0.175	0.158	0.164	0.177	0.178	0.341	0.171	...	0.11	0.098	0.092	0
D0028	0.495	0.499	0.51	0.716	0.344	0.337	0.474	0.761	0.325	0.237	...	0.402	0.367	0.473	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
D0995	0.585	0.29	0.172	0.108	0.253	0.241	0.308	0.252	0.158	0.123	...	0.195	0.223	0.484	0
D0996	2.289	0.497	0.634	0.752	0.594	0.328	2.345	4.032	2.106	0.737	...	0.988	2.393	0.965	0
D0997	1.056	1.302	0.498	0.526	0.243	1.296	0.544	1.28	0.787	1.292	...	0.704	0.391	0.922	0
D0998	0.556	0.332	0.584	1.094	0.66	0.672	0.267	1.098	0.357	0.529	...	0.456	0.519	0.499	0
D0999	0.19	0.204	0.348	0.586	0.343	0.343	0.237	0.354	0.257	0.438	...	0.313	0.981	0.333	0
D1000	0.279	0.279	0.328	0.189	0.306	0.247	0.355	0.753	0.374	0.493	...	0.23	0.562	0.179	0
D1001	0.325	0.704	0.719	0.093	0.077	0.091	0.134	0.231	0.117	0.116	...	0.085	0.15	0.121	0
D1002	1.668	0.825	0.456	0.322	3.52	0.816	1.111	0.132	0.729	0.097	...	0.55	0.26	0.827	0
D1003	0.067	0.12	0.811	0.921	0.16	0.528	0.688	0.251	0.237	0.167	...	0.385	0.164	0.213	0
D1004	0.125	0.047	0.048	0.342	0.21	0.374	0.176	0.079	0.341	0.271	...	0.266	0.17	0.073	0
D1005	0.177	0.204	0.224	0.242	2.302	2.166	0.92	0.703	1.276	0.431	...	0.365	0.55	0.501	0
D1006	0.253	0.215	0.09	0.1	0.09	0.106	0.141	0.369	0.185	0.325	...	0.086	0.13	0.062	0
D1007	0.462	0.143	0.213	0.277	0.144	0.191	0.732	0.201	0.264	0.3	...	0.117	1.289	0.135	0
D1008	0.133	0.566	0.138	0.256	0.483	0.412	0.292	0.342	0.437	0.288	...	0.288	0.361	0.18	0
D1009	0.416	0.437	0.719	0.607	0.452	0.92	0.484	0.696	0.542	1.226	...	0.881	0.517	0.61	0
D1010	1.975	0.266	0.405	0.516	3.512	1.045	0.391	0.821	0.733	1.039	...	0.537	0.57	0.178	0
D1011	0.444	0.344	0.353	0.373	0.395	0.33	0.306	0.36	0.313	0.38	...	0.297	0.338	0.233	0
D1012	1.847	0.184	1.899	0.208	0.145	0.27	0.082	0.728	0.258	0.138	...	0.101	0.157	0.454	0

	2169	2193	2217	2241	2289	2313	2361	2385	2433	2481	...	6825	7065	7089	
D1013	3.323	3.443	1.273	2.057	3.202	2.467	1.946	2.186	1.75	0.717	...	0.856	1.528	1.514	1
D1014	1.542	1.457	1.383	0.469	0.602	0.601	0.382	2.121	1.626	1.161	...	0.848	2.51	3.014	0
D1015	0.031	0.074	0.053	0.038	0.049	0.053	0.071	0.047	0.056	0.035	...	0.057	0.06	0.151	0
D1016	0.264	0.272	0.035	0.239	0.267	0.033	0.244	0.094	0.608	0.048	...	0.067	0.066	0	0
D1017	0.442	0.297	0.333	0.263	0.4	0.209	0.383	0.412	0.421	0.363	...	0.369	0.381	0.391	0
D1018	0.093	0.095	0.093	0.098	0.097	0.368	0.11	0.104	0.109	0.149	...	0.129	0.105	0.145	0
D1019	0.032	0.163	0.158	0.172	0.286	0.056	0.043	0.183	0.094	0.223	...	0.046	0.514	0.379	0
D1020	0.148	0.159	0.158	0.181	0.742	0.272	0.215	0.167	0.252	0.353	...	0.202	0.207	0.283	0
D1021	0.097	0.084	0.105	0.095	0.17	1.425	0.25	0.078	0.157	0.097	...	0.108	0.698	0.127	0
D1022	0.155	0.106	0.154	0.123	0.124	0.12	0.125	0.125	0.063	0.085	...	0.064	0.078	0.084	0
D1023	0.016	0.017	0.039	0.035	0.046	0.038	0.074	0.114	0.09	0.055	...	0.049	0.039	0.042	0
D1024	0.197	0.312	0.175	0.172	0.144	0.146	0.119	0.837	0.258	0.071	...	0.144	0.074	0.164	0

1026 rows × 139 columns

In [8]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Test with 9:00 am TOU data
x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('09:00:00')]
x.set_index('GMT', inplace = True)
x = x.transpose().dropna() # data for the specific hour, and filtering out null value
x = x.values
x = StandardScaler().fit_transform(x)
pca = PCA()# keep all columns to cover 100% variance
principleComponents = pca.fit_transform(x)
principleDf = pd.DataFrame(data = principleComponents)
principleDf
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	...	129	
0	-4.512222	1.015687	-0.355179	-0.436117	-0.189375	1.374784	-1.288974	1.202487	0.761967	0.469236	...	1.071673	-0.334
1	-3.046918	-0.430773	-0.309117	0.136009	0.278526	-0.077276	-0.847145	-0.863169	-0.471936	0.135650	...	-0.303065	-0.451
2	11.478668	5.324820	1.575769	0.353479	1.342200	1.226339	2.637426	0.257290	-4.632804	-0.125207	...	-0.404059	0.9215
3	-1.340653	4.430653	-0.857014	0.946635	-0.013841	-3.105843	-1.606426	-0.334299	-2.430122	-1.553990	...	0.060032	0.1973
4	-6.190426	1.171094	1.754095	0.380849	0.191071	0.198235	-0.048337	0.178417	-0.554229	-0.123905	...	-0.082153	-0.205
5	-1.231413	-0.350828	-0.673411	-0.177436	-0.480530	-0.172093	-0.079046	0.023071	-0.015000	-0.037657	...	0.071770	0.0794
6	-0.295792	-0.968340	-0.555583	0.130429	-0.483096	2.060626	-0.761686	0.080714	0.291958	-0.351315	...	0.646844	-0.403
7	-1.355168	-0.249267	0.361870	-0.080553	-0.341466	0.402250	-0.288570	-1.010213	1.124667	0.435958	...	0.143460	-0.840
8	-5.339566	-0.349525	-0.238231	0.127173	-0.617952	-0.300913	0.085161	0.034700	0.093144	-0.602490	...	-0.063281	0.0369
9	1.594617	3.759965	-1.263836	2.478668	-1.564113	0.212554	-2.305600	-2.726842	-1.465381	-0.982981	...	1.073802	-0.251
10	-9.639351	-0.647861	0.464085	0.323556	0.352476	0.011932	0.059784	0.022269	-0.164712	0.183052	...	0.005130	-0.018
11	-1.442067	0.245890	0.193734	0.719246	-0.561145	0.217221	0.035779	0.337547	-0.425412	-0.328396	...	0.779053	0.3830
12	-1.577523	-1.244185	-0.337935	-0.298160	0.077438	0.112237	-1.098724	-0.773379	-0.708718	0.016898	...	0.112017	0.0834
13	-2.125715	2.711509	-0.622894	1.176684	0.141209	-1.424607	-0.331539	0.561681	-0.959121	-0.728359	...	0.470543	0.3909
14	13.121437	-3.035858	1.608437	1.587707	1.089343	1.856240	1.288642	-1.231034	-0.180316	-1.399257	...	-0.134723	-0.686
15	-3.052158	-0.757713	0.527327	-0.425208	-0.079381	0.047366	0.238736	-1.715172	0.017014	0.048020	...	-0.264421	0.2288
16	-6.083458	0.891250	-0.214220	0.222147	-0.141191	-0.593200	-0.103183	-0.076725	0.530901	-0.053085	...	-0.103454	0.3197
17	1.434359	6.324802	1.306695	-0.698214	0.645091	-1.475785	-0.056580	0.441419	-1.494332	-0.565527	...	0.313796	-1.012
18	-6.367134	-0.278755	0.419944	1.082329	0.306659	-0.402911	0.134913	0.124970	-0.130992	-0.274807	...	0.082774	-0.241
19	4.011328	1.827811	0.194275	-1.628212	-0.908681	1.278036	-2.302290	1.028443	-2.331441	-0.188827	...	0.733136	0.1177
20	-1.364409	2.626805	0.559167	0.737993	1.531242	0.411756	-0.828412	2.600642	-0.770085	-1.207028	...	-0.020067	-0.058
21	-9.926693	-0.610546	0.810988	0.333324	0.580164	0.012799	0.030092	-0.022357	-0.149321	0.161743	...	-0.007635	0.1071
22	-0.372468	-0.678076	-1.132175	-0.434996	-0.422039	-1.125904	-0.635836	-0.695251	0.582243	-0.475490	...	0.096300	0.0894
23	6.095033	0.262529	-1.120467	1.540511	-0.025735	0.955728	-0.505688	-1.477974	-0.450742	-0.591099	...	-0.072564	0.0050
24	-7.250615	-0.462963	0.082721	0.551320	0.187333	-0.144509	-0.189599	-0.019172	-0.131173	-0.004547	...	0.123123	0.1576
25	-6.007972	0.090382	-0.690833	-0.110593	-0.123870	0.139676	-0.559622	0.443976	0.256887	0.230213	...	-0.298425	-0.005
26	-5.408896	-0.214922	0.399155	-0.210698	0.015532	0.431078	-0.307346	0.387932	0.240799	-0.220881	...	0.367881	0.4954
27	0.797186	0.911725	0.022422	0.314950	-0.038957	0.854224	0.780326	0.181902	-0.003022	0.088759	...	-0.401795	0.0132
28	17.067722	0.815304	16.496756	-8.980625	-1.564113	-2.732175	-4.227812	-0.909515	0.551132	2.050351	...	1.041187	-0.426
29	2.101676	0.306077	-0.136663	2.128976	-1.733014	4.739782	-1.439399	-0.868281	1.091229	1.838123	...	0.036046	-0.043
...	...	...	...	...	...	...	...	...	...	...	...	...	...
915	18.163801	-0.555929	0.786531	-0.753284	2.406616	1.617423	-1.071443	0.090457	-0.169234	-1.044034	...	-0.516890	-0.041
916	-4.127316	-0.779572	0.164774	0.439736	-0.135854	1.010505	-0.254017	0.297940	-0.591987	0.286293	...	0.021114	-0.260
917	23.309735	-2.630396	0.313506	1.737093	1.723385	4.338141	2.672543	1.185883	-5.068112	0.138585	...	-0.053723	0.1549
918	8.598908	-0.576539	-1.717446	0.114511	0.199700	-0.857896	-0.076789	0.288845	1.112946	-0.585049	...	0.116888	-0.351
919	4.682335	1.095920	-1.549097	0.169857	-0.388231	-1.577948	-0.533172	-0.624004	-0.145549	-0.823452	...	0.051852	0.0837
920	-1.547062	-0.992232	-0.952826	0.055519	0.329949	0.325661	0.482589	-0.077398	0.366215	-0.267865	...	0.034470	-0.492
921	-0.520844	-0.047002	-0.211098	-0.091880	0.241422	-0.501056	-0.422044	0.319560	-0.083435	0.449160	...	-0.186803	-0.235
922	-3.669560	-1.759440	0.491971	0.729492	-0.484798	-0.270062	0.521191	0.293846	0.809176	0.854257	...	-0.188825	-0.254
923	8.404591	2.310620	-0.434291	-0.561177	-6.329485	-0.210240	2.387605	3.681032	-3.133732	-0.030098	...	-0.473601	0.4511
924	-2.191053	1.585421	-0.861352	-1.036410	-0.480673	1.241087	0.114374	0.768028	0.794892	0.440266	...	-0.175909	-0.451
925	-6.909990	-0.319302	0.245488	-0.025647	-0.107755	0.030365	0.029710	0.009843	-0.122989	0.178591	...	-0.004873	0.2236
926	2.549964	0.425430	2.922496	-1.973045	-0.029095	1.602659	0.139031	0.792535	-0.826451	0.351082	...	-0.140897	-0.584
927	-4.708458	0.261125	-0.449676	-0.295832	0.559766	0.025081	-0.005531	-0.126721	-0.305765	-0.117961	...	0.243585	0.1168
928	-3.860062	-0.314753	0.287854	1.090181	-0.106893	-0.966021	-1.316240	0.139158	-0.376550	0.319396	...	-0.679465	0.6656
929	0.646095	0.235627	-1.925833	-0.054970	0.443104	0.364404	1.081161	0.088929	0.800264	-0.332837	...	-0.324342	-0.300
930	8.635751	1.530393	-2.788555	-0.938038	0.256930	-0.745242	0.018643	-1.048247	-0.006015	-0.769122	...	0.097407	0.1249
931	6.162222	1.372751	3.432804	-0.545159	-0.958292	-0.387687	-0.028456	0.037723	-0.028058	-0.145303	...	-0.095991	0.4222
932	0.839431	-0.133373	-0.681193	0.143031	0.604995	-0.071887	-0.919669	0.007984	0.753483	0.624755	...	-0.614985	0.1646
933	-3.619951	-1.003801	1.052126	0.573675	-0.380424	-0.456452	0.546088	-0.338131	0.367981	0.265455	...	0.497582	0.7909

	0	1	2	3	4	5	6	7	8	9	...	129	
934	23.725092	-7.204944	3.305524	-2.085993	14.206224	-0.459352	-1.290797	15.095077	-7.791476	6.716550	...	0.046170	0.255(
935	-7.273826	-1.101513	0.335385	0.133547	0.938196	-0.061717	0.148742	0.556689	0.284985	0.261463	...	-0.010970	0.147
936	-7.285627	-0.838435	0.907847	-0.315298	0.621928	-0.264597	-0.582919	0.682987	0.012967	-0.195573	...	-0.080571	-0.510
937	1.071899	-0.290715	-0.785939	0.005810	0.241315	-0.070963	-0.525343	-0.253821	-0.154625	-0.034037	...	0.035545	-0.256
938	-6.022776	-0.378201	-0.450970	-0.271958	-0.101511	0.461885	-0.150962	0.181595	-0.257883	-0.470833	...	0.282474	0.305
939	-5.423947	-0.011220	-0.140830	0.209782	0.293905	0.265561	-0.003789	0.353700	0.395939	0.226448	...	-0.209090	0.098
940	-2.656478	0.534735	-1.124158	-0.293869	-0.533387	0.118191	-0.760425	0.064379	0.778966	0.034424	...	0.228936	0.677
941	-5.086264	-0.271159	-0.012567	0.658203	0.018962	-0.585131	0.825126	0.287956	-0.016560	0.210624	...	0.228533	-0.097
942	-7.428534	-0.907531	0.297246	0.185032	0.950158	0.107094	-0.062366	0.317623	-0.154845	0.228435	...	0.505413	-0.198
943	-8.752728	-0.760504	0.415760	0.139956	0.244867	0.043667	0.041391	0.014013	-0.035069	0.267356	...	-0.030241	0.090
944	-6.563086	-1.022306	0.089565	0.046444	-0.137239	-0.321302	0.297489	0.399314	-0.137933	0.218216	...	-0.034066	-0.074

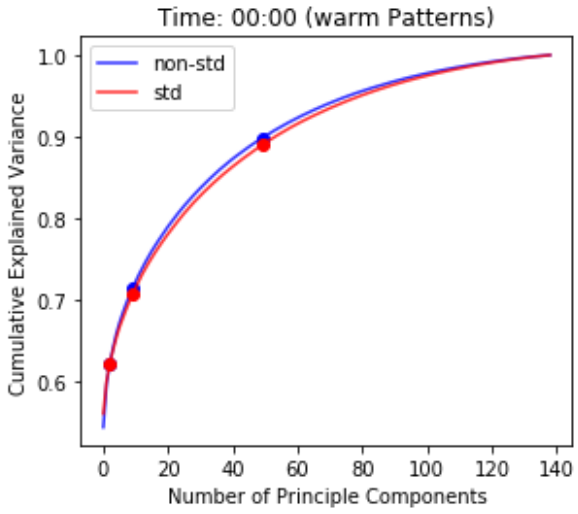
945 rows × 139 columns

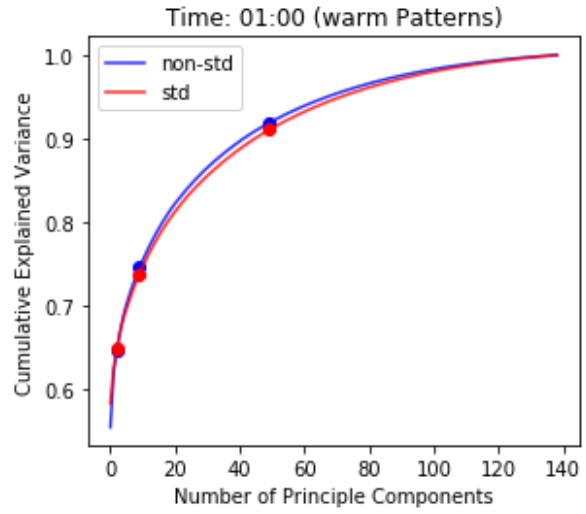
Note: in cold seasons non-null value users are 958, which is more than these in the warm seasons.

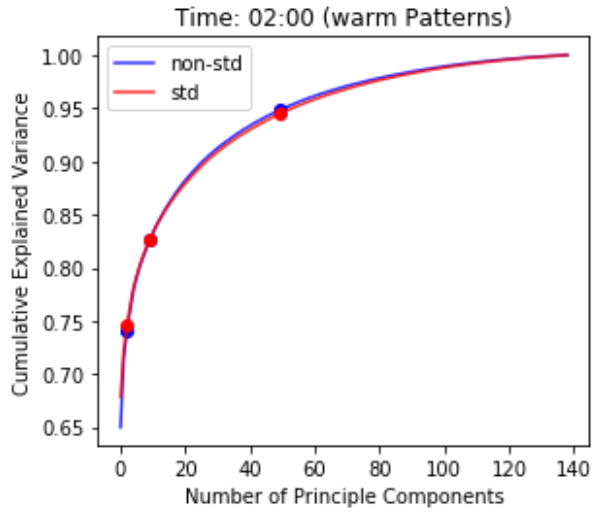
```
In [15]: sum(x[:,0])
```

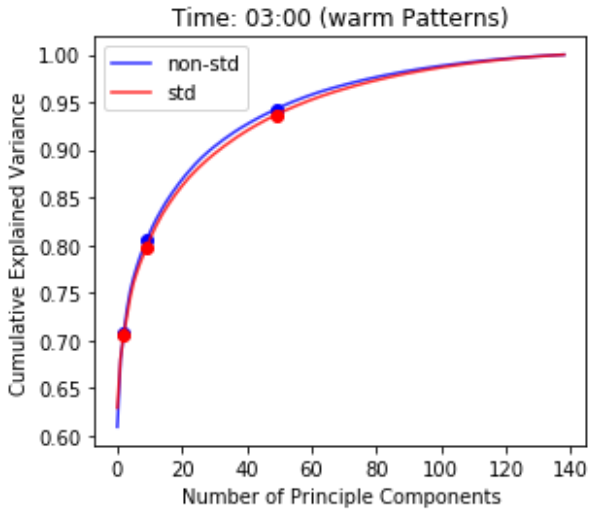
```
Out[15]: 4.796163466380676e-14
```

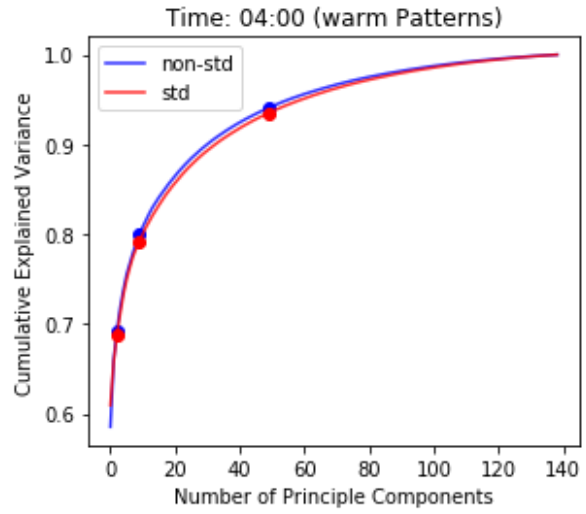
In [9]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.plot(np.cumsum(pca_nstd.explained_variance_ratio_), c = 'blue', label = 'non-std', alpha = 0.8)
        ax.plot([2, 9, 49], [np.cumsum(pca_nstd.explained_variance_ratio_)[2], np.cumsum(pca_nstd.explained_v
ariance_ratio_)[9], np.cumsum(pca_nstd.explained_variance_ratio_)[49]], 'bo')
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.plot(np.cumsum(pca.explained_variance_ratio_), c = 'red', label = 'std', alpha = 0.8)
        ax.plot([2, 9, 49], [np.cumsum(pca.explained_variance_ratio_)[2], np.cumsum(pca.explained_variance_ra
tio_)[9], np.cumsum(pca.explained_variance_ratio_)[49]], 'ro')
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Number of Principle Components')
        ax.set_ylabel('Cumulative Explained Variance')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.plot(np.cumsum(pca_nstd.explained_variance_ratio_), c = 'blue', label = 'non-std', alpha = 0.8)
        ax.plot([2, 9, 49], [np.cumsum(pca_nstd.explained_variance_ratio_)[2], np.cumsum(pca_nstd.explained_v
ariance_ratio_)[9], np.cumsum(pca_nstd.explained_variance_ratio_)[49]], 'bo')
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.plot(np.cumsum(pca.explained_variance_ratio_), c = 'red', label = 'std', alpha = 0.8)
        ax.plot([2, 9, 49], [np.cumsum(pca.explained_variance_ratio_)[2], np.cumsum(pca.explained_variance_ra
tio_)[9], np.cumsum(pca.explained_variance_ratio_)[49]], 'ro')
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Number of Principle Components')
        ax.set_ylabel('Cumulative Explained Variance')
plt.tight_layout()
```

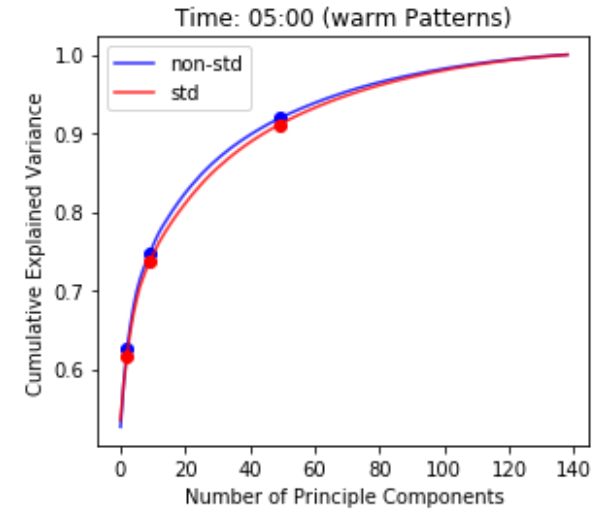

Time: 00:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

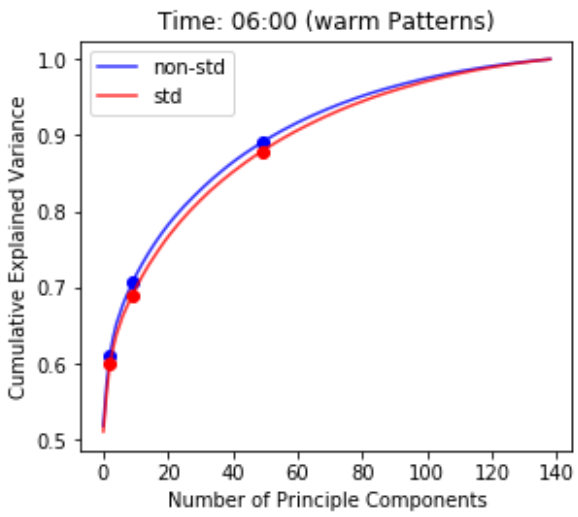

Time: 01:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

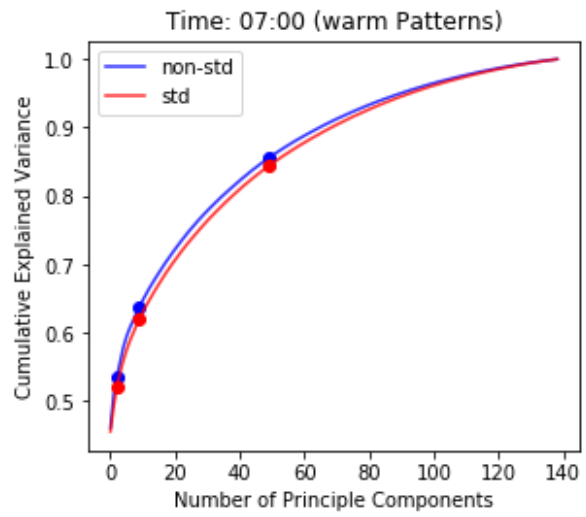

Time: 02:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

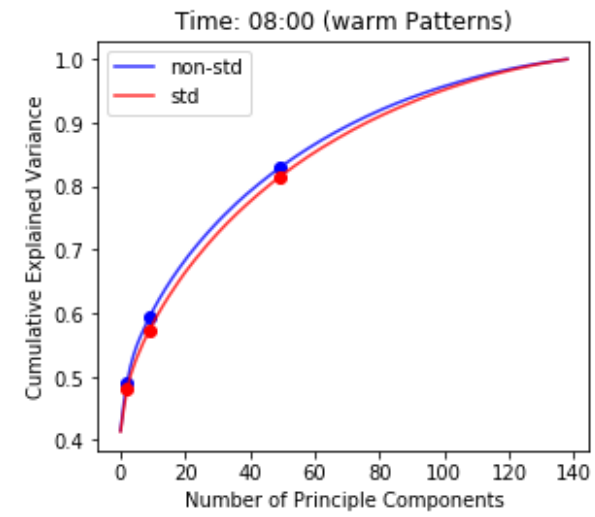

Time: 03:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

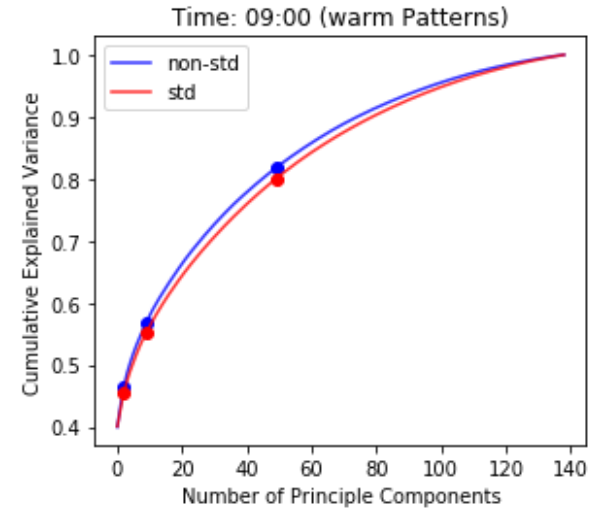

Time: 04:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

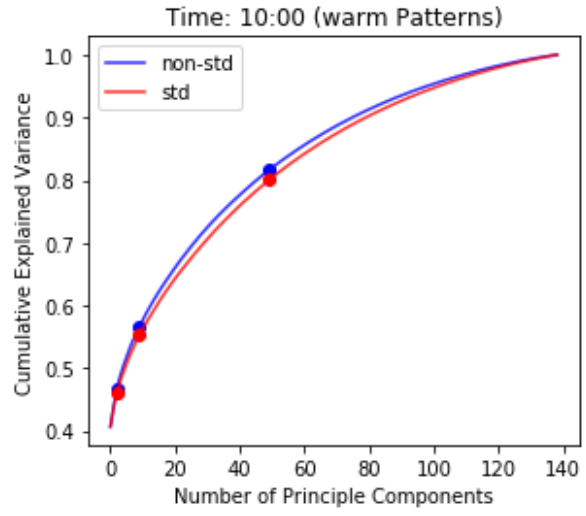

Time: 05:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

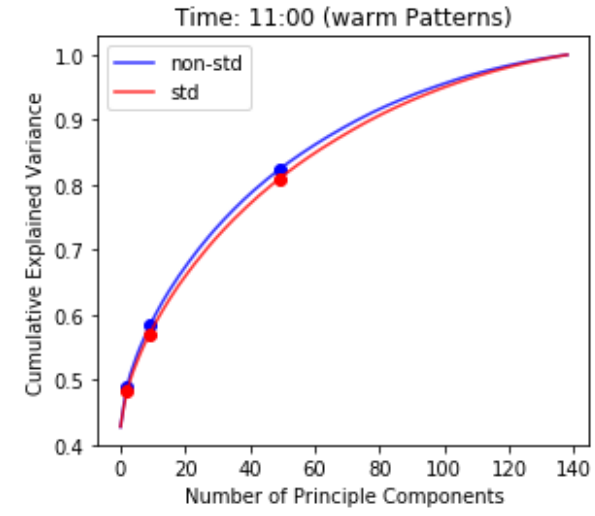

Time: 06:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

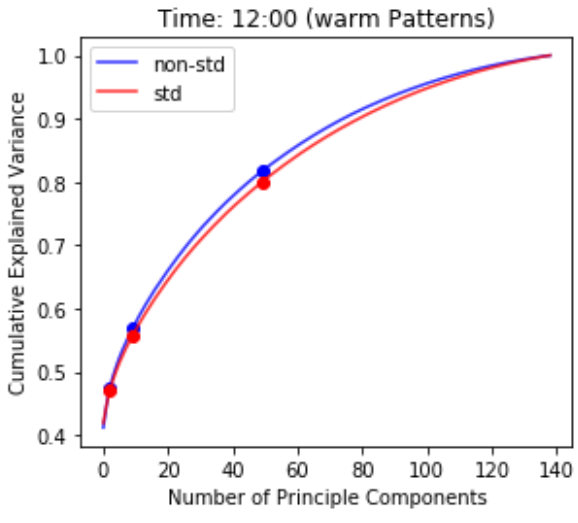

Time: 07:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

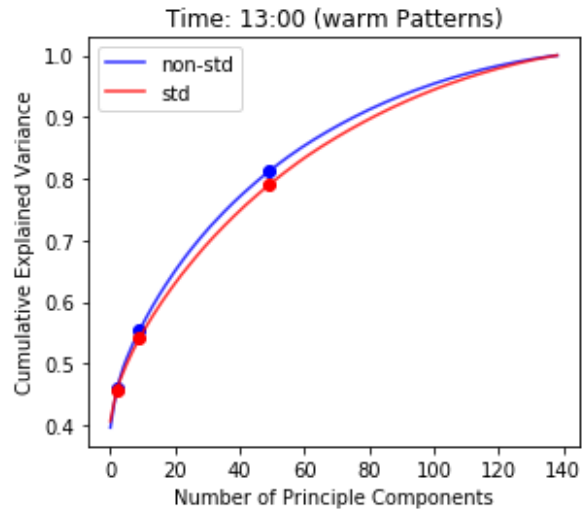

Time: 08:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

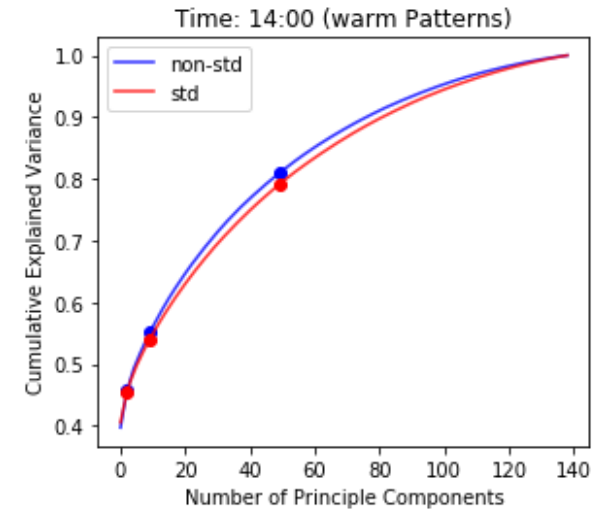

Time: 09:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components


Time: 10:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components


Time: 11:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components


Time: 12:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

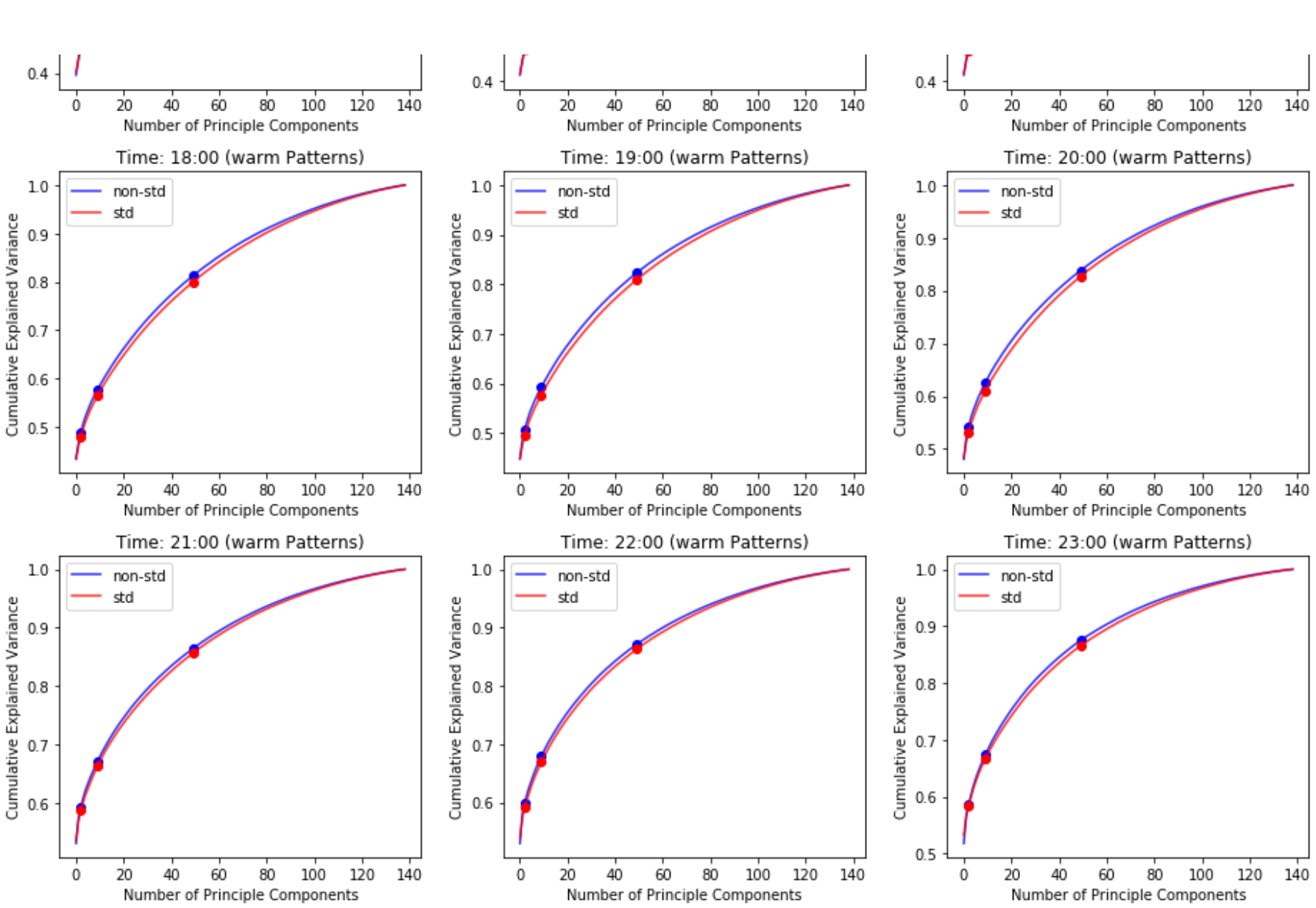

Time: 13:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components


Time: 14:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

Time: 15:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

Time: 16:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

Time: 17:00 (warm Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components



The above pictures show that using the first 3 PCs can always explain from 45% to 74% amount of variance.

Next we will give the meaning of the first 3 PCs:

PC1 & PC2: temperature related (no matter of standarization, PC1 and PC2 are somehow correlated)

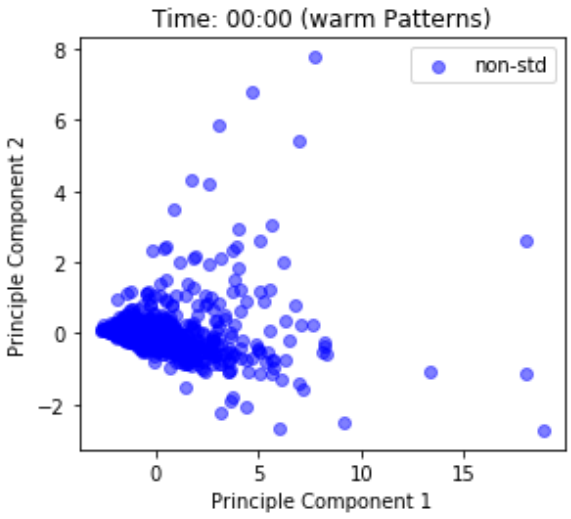
PC3: related to days of week, since the coeffients of Friday and Saturday are skewied to positive area, if there's not correlation, it should always be equally distributed around 0

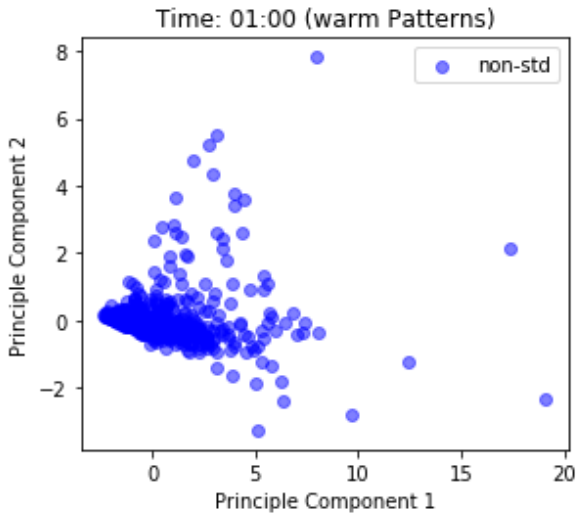
Using the 3 components to do k-mean clustering and we can get several groups

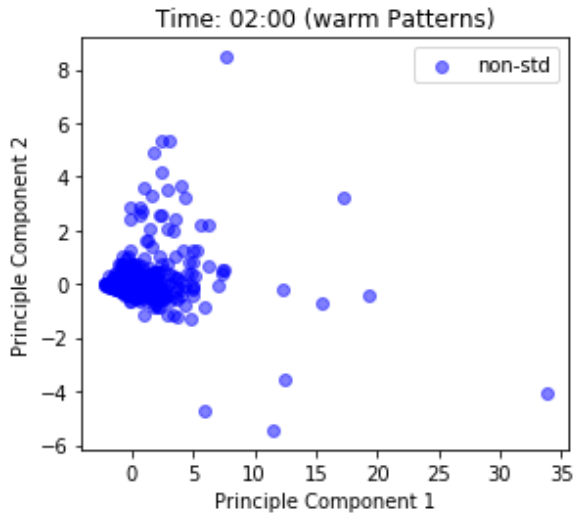
Coloring them in a 3D space (or if the first two PCs leads to some confustion, maybe do 2D with 3 differnt projections)

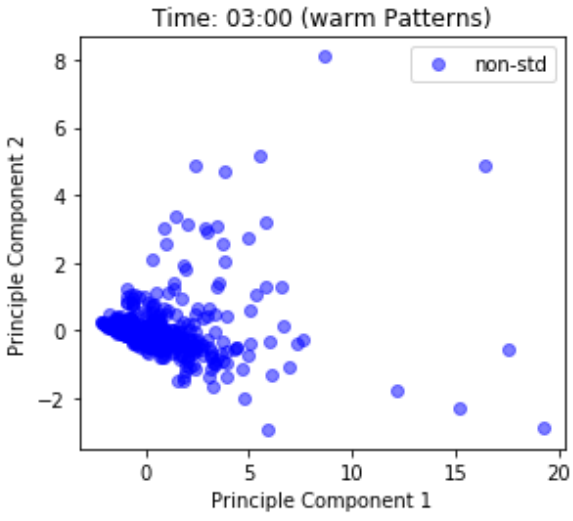
Then we could try to color them in a 2D slice surphase (temp vs day of week, temp)

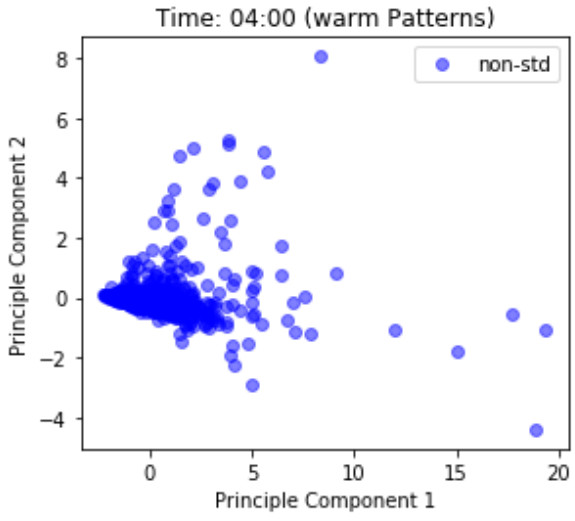
In [10]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[0], principleDf_nstd[1], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[0], principleDf[1], c = 'red', label = 'std', alpha = 0.1)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[0], principleDf_nstd[1], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[0], principleDf[1], c = 'red', label = 'std', alpha = 0.1)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

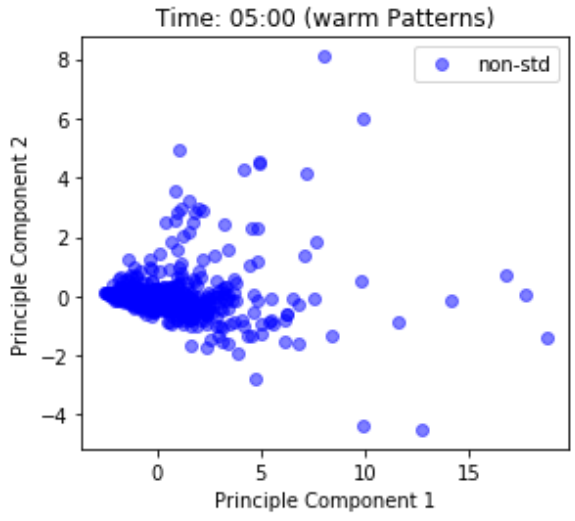

Time: 00:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

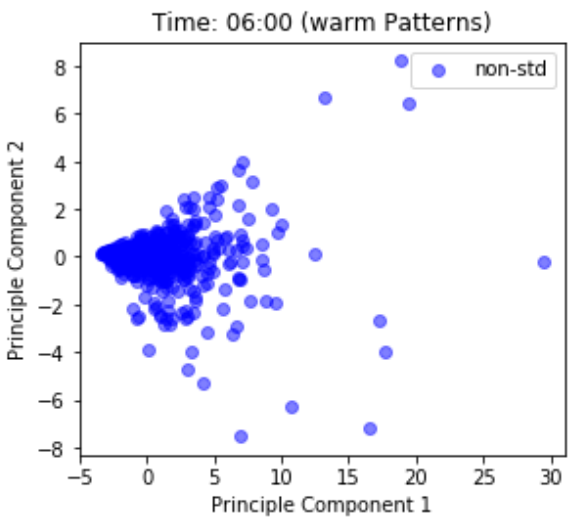

Time: 01:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

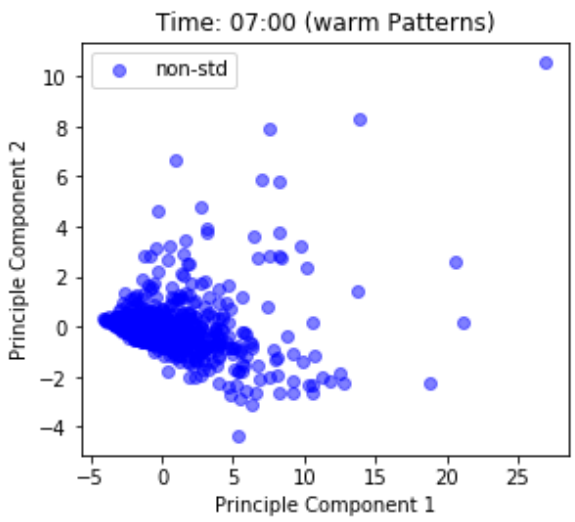

Time: 02:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

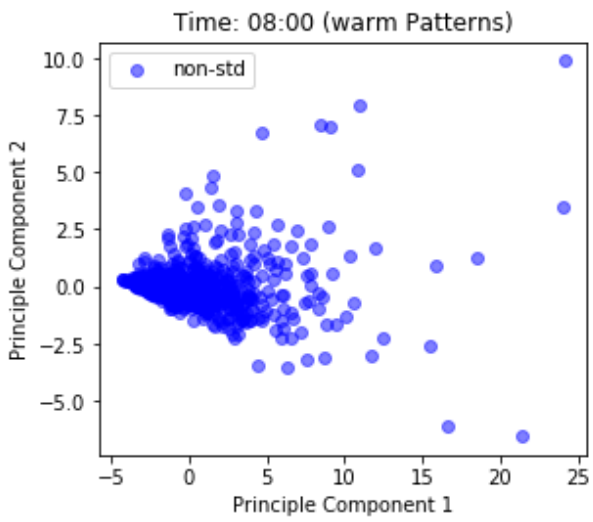

Time: 03:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

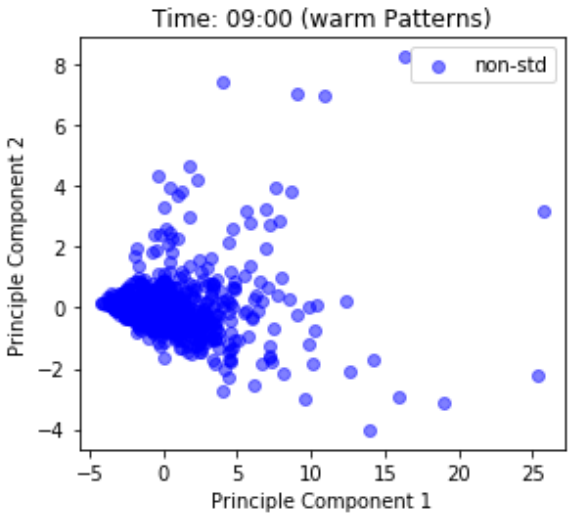

Time: 04:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

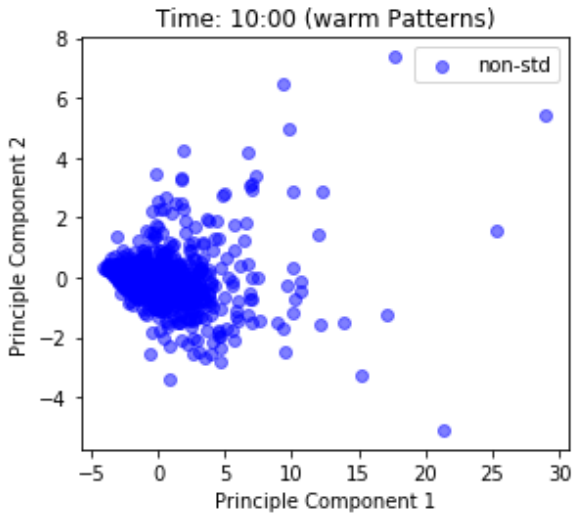

Time: 05:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

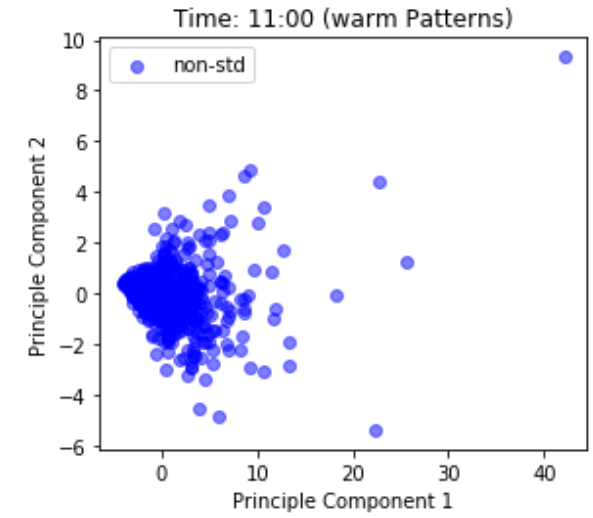

Time: 06:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

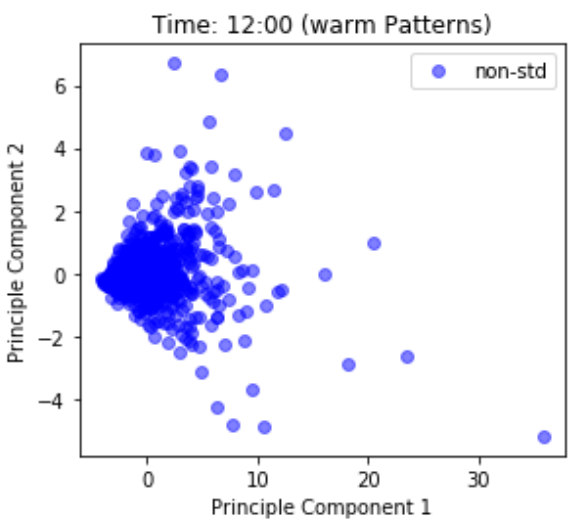

Time: 07:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

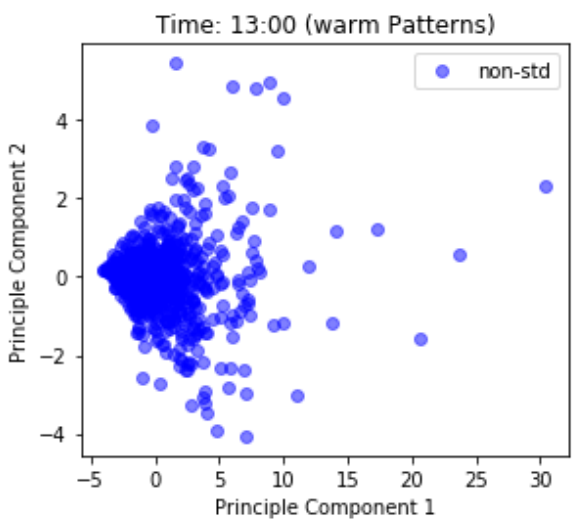

Time: 08:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

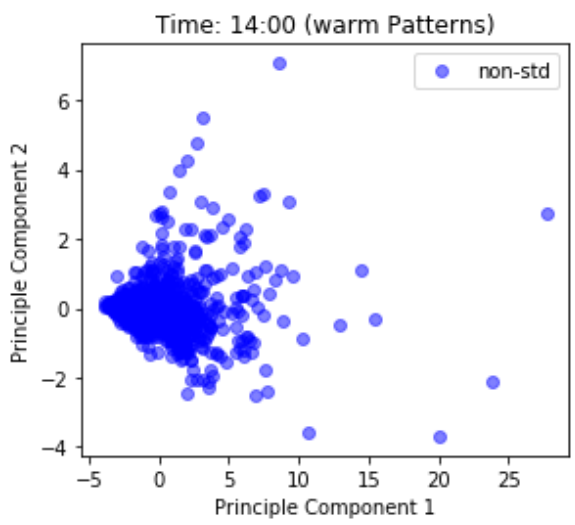

Time: 09:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

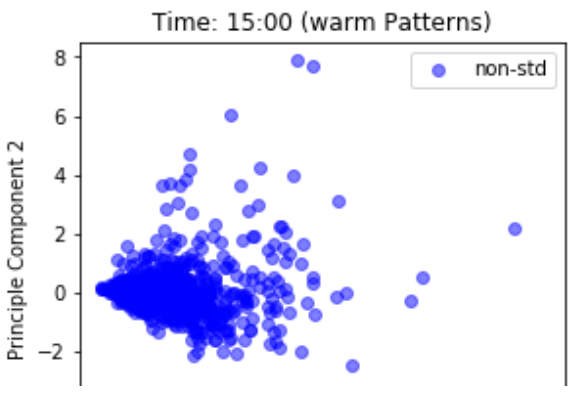

Time: 10:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

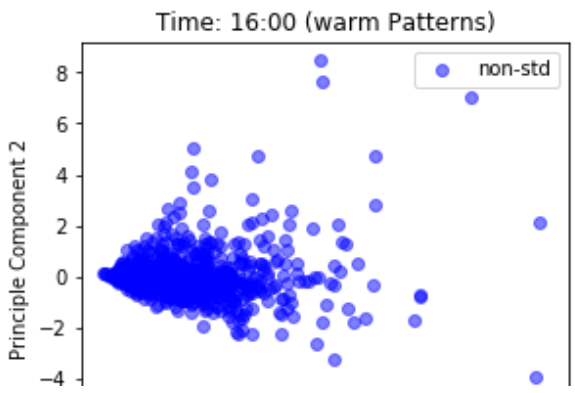

Time: 11:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

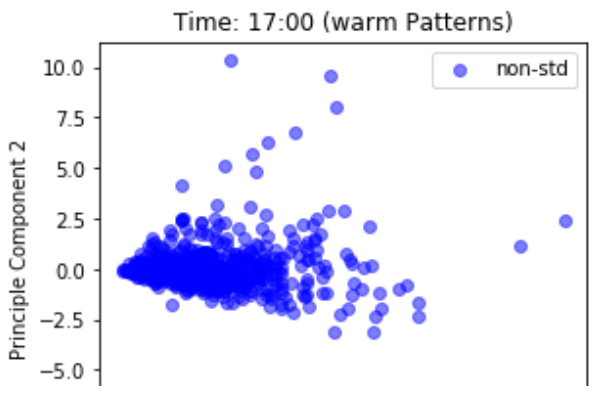

Time: 12:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

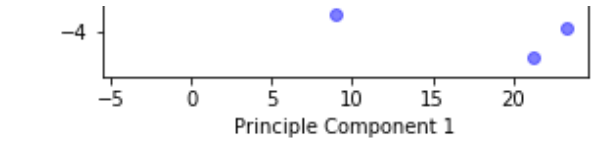

Time: 13:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

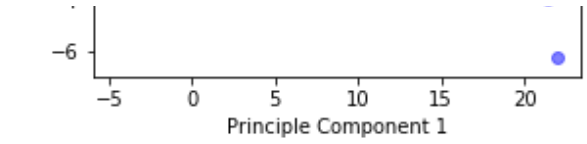

Time: 14:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

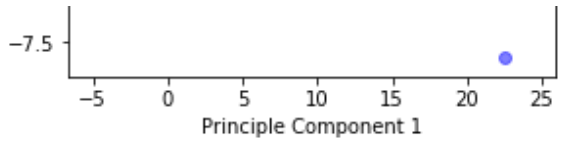

Time: 15:00 (warm Patterns)
non-std
Principle Component 2

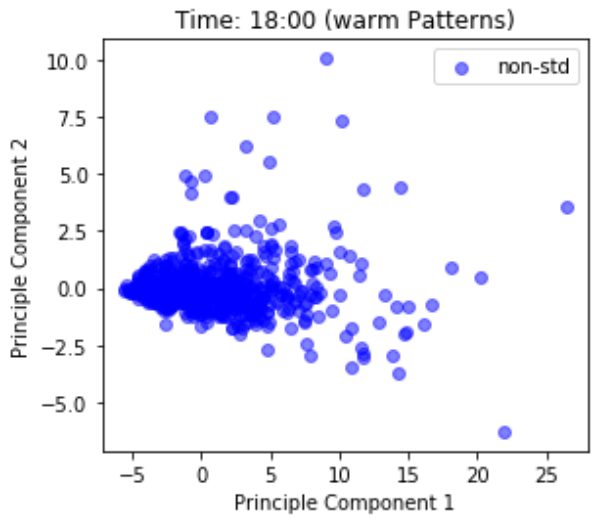

Time: 16:00 (warm Patterns)
non-std
Principle Component 2

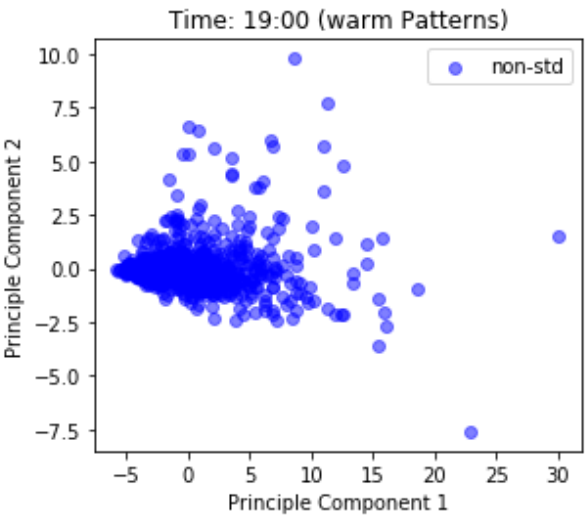

Time: 17:00 (warm Patterns)
non-std
Principle Component 2

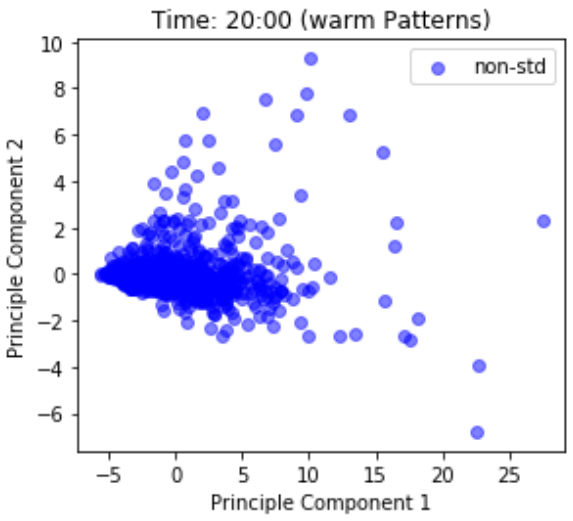

Principle Component 1

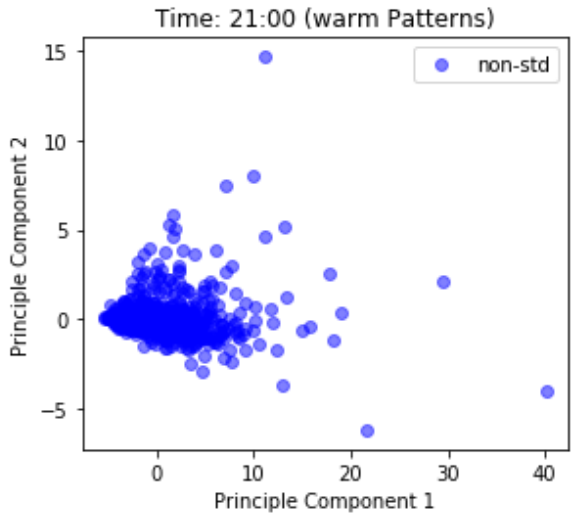

Principle Component 1

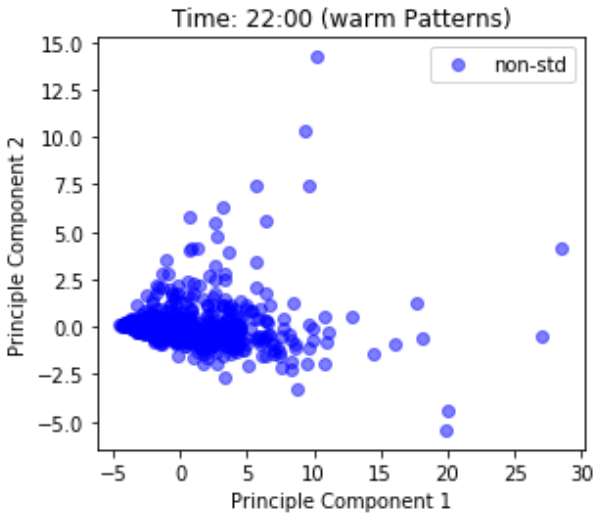

Principle Component 1

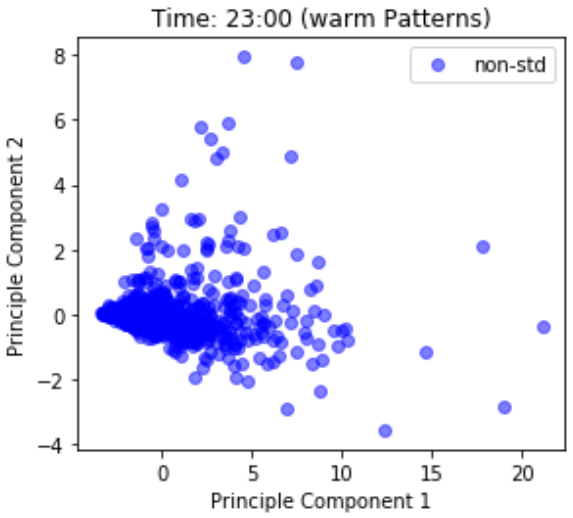

Time: 18:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1


Time: 19:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

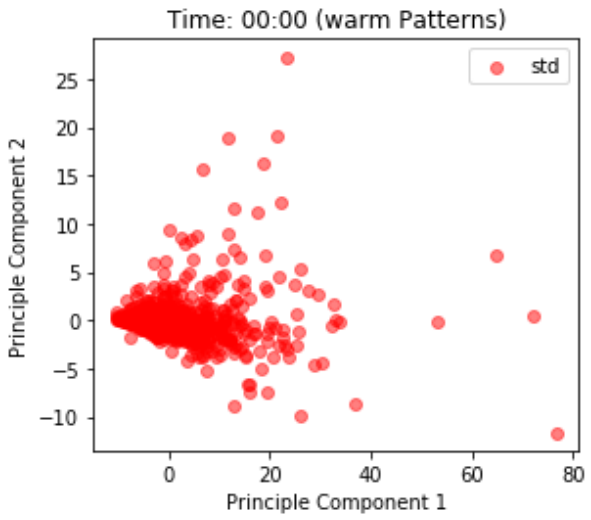

Time: 20:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

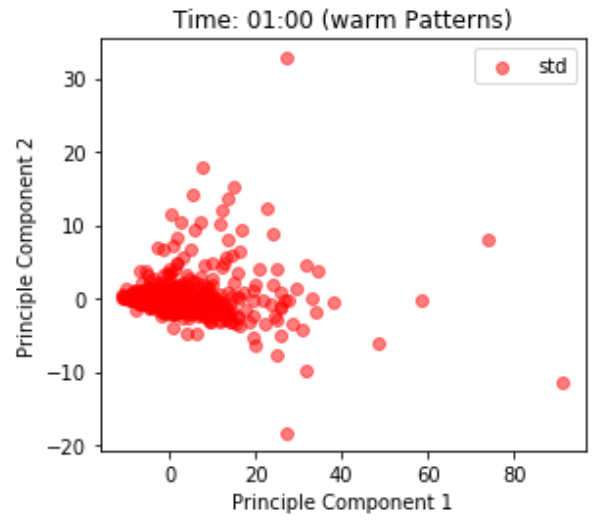

Time: 21:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

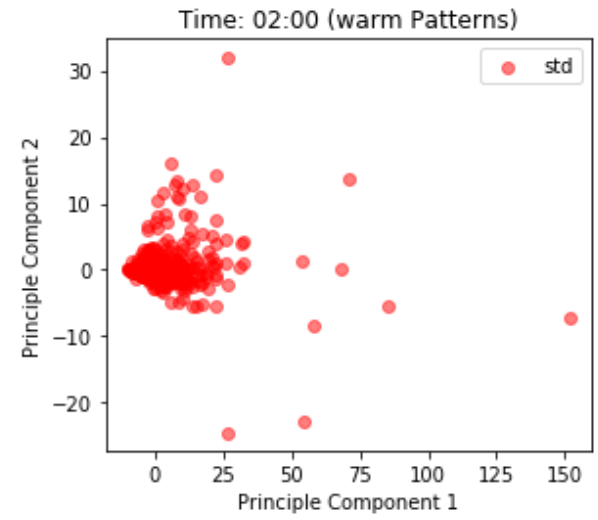

Time: 22:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

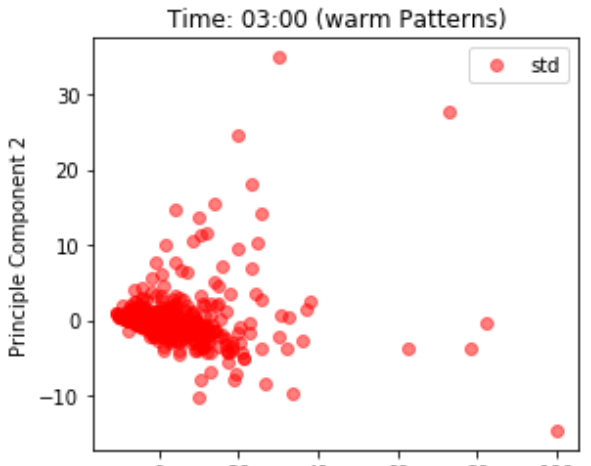

Time: 23:00 (warm Patterns)
non-std
Principle Component 2
Principle Component 1

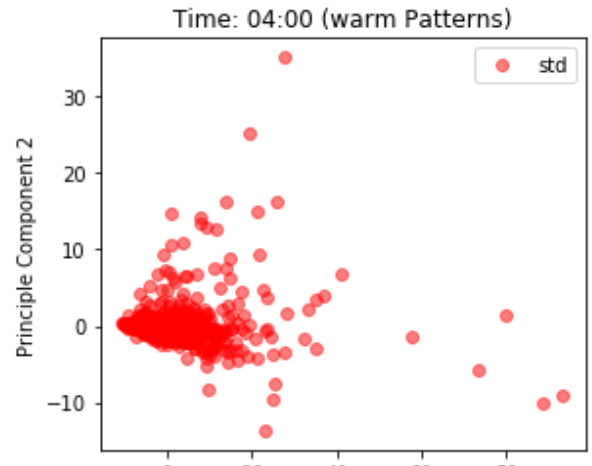
In [11]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[0], principleDf_nstd[1], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[0], principleDf[1], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[0], principleDf_nstd[1], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[0], principleDf[1], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

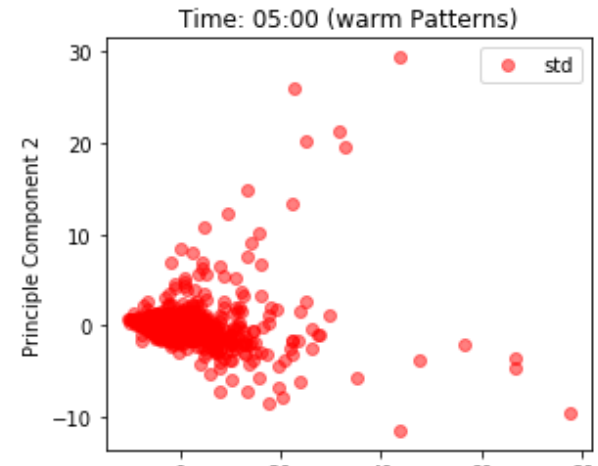

Time: 00:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

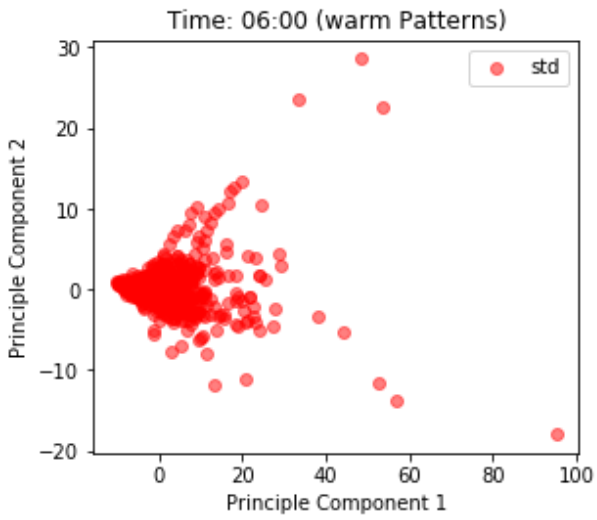

Time: 01:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

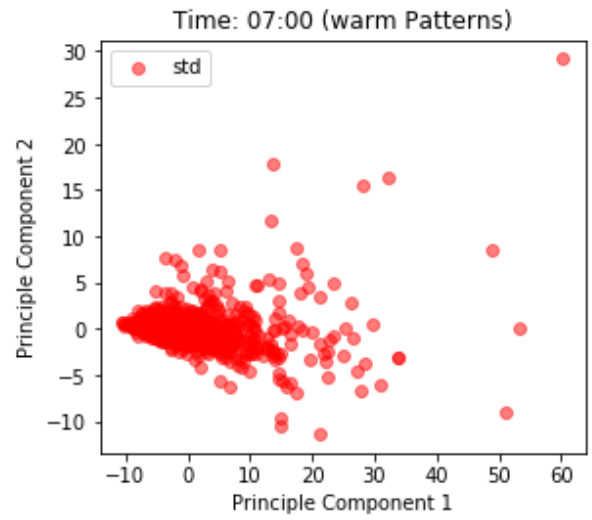

Time: 02:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

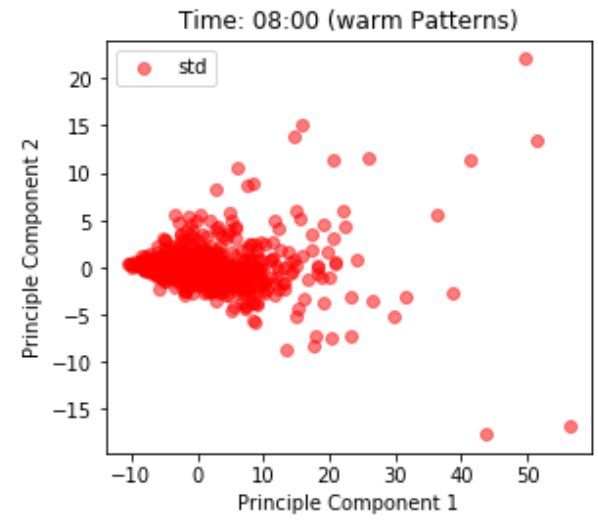

Time: 03:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

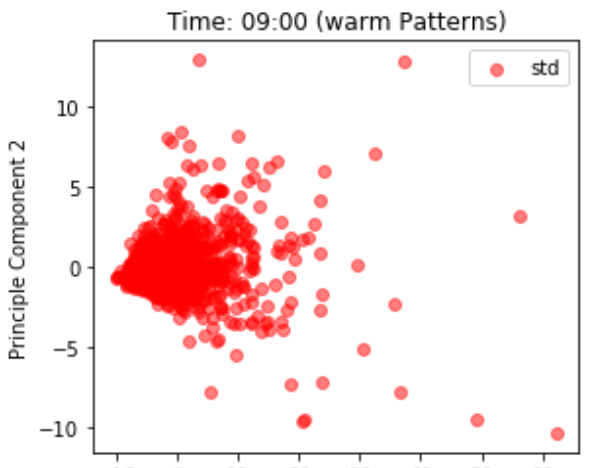

Time: 04:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

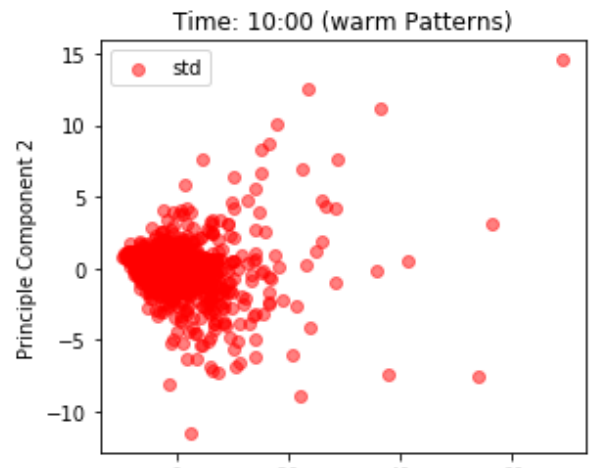

Time: 05:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

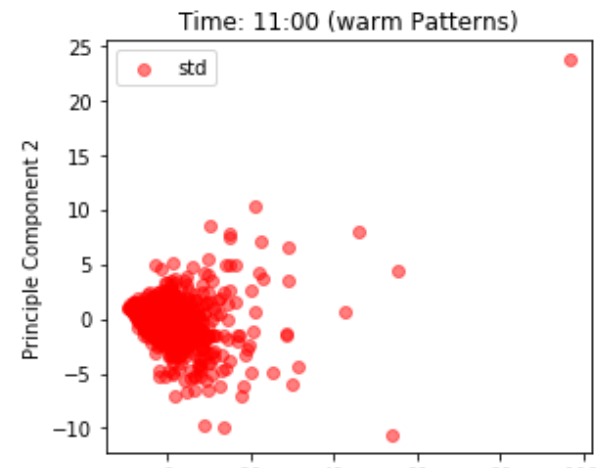

Time: 06:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

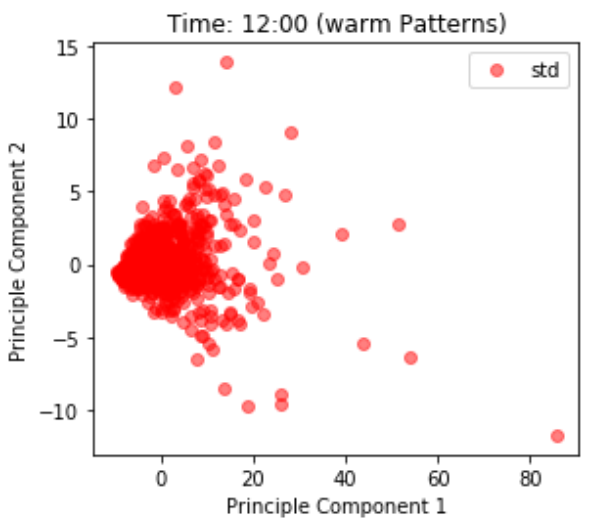

Time: 07:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

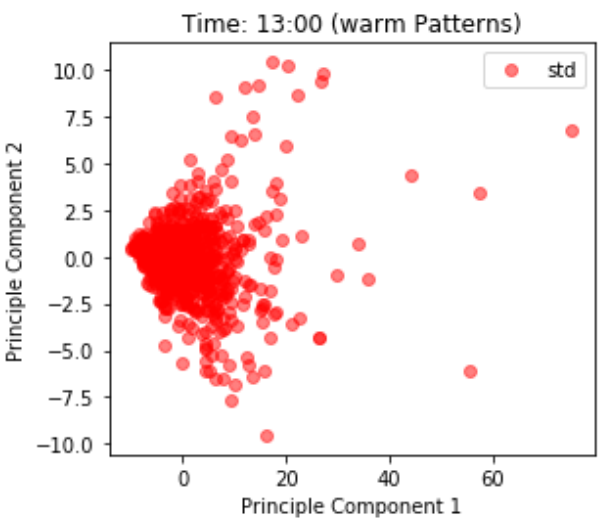

Time: 08:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

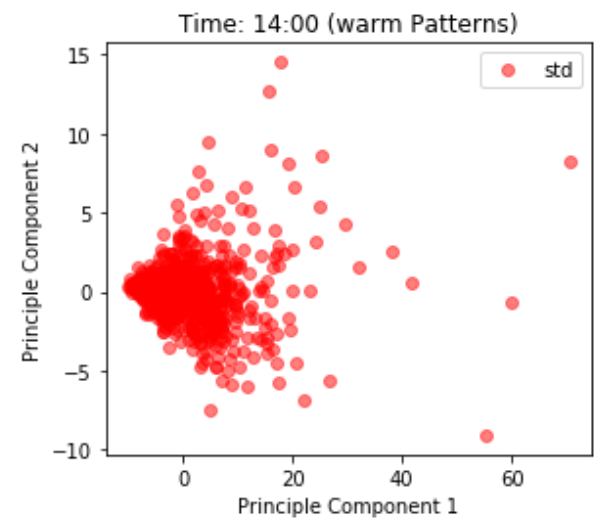

Time: 09:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

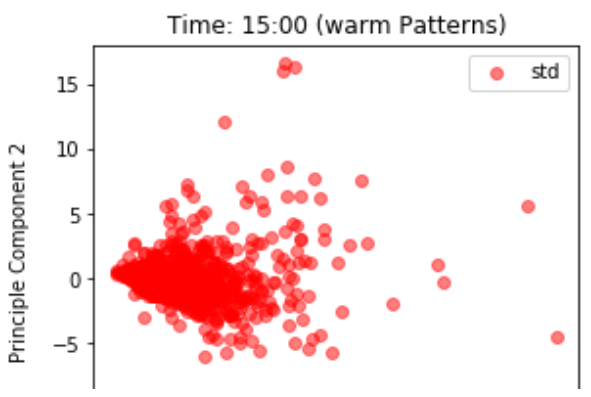

Time: 10:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

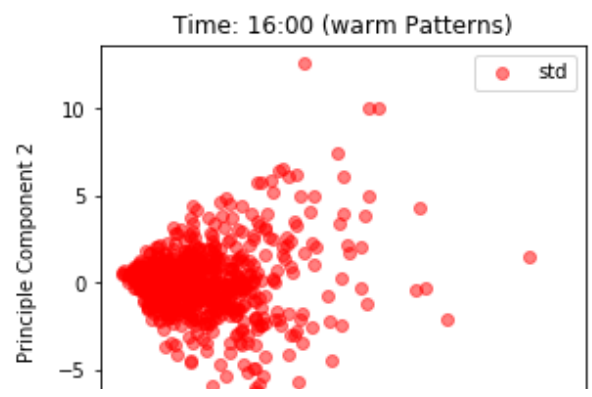

Time: 11:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

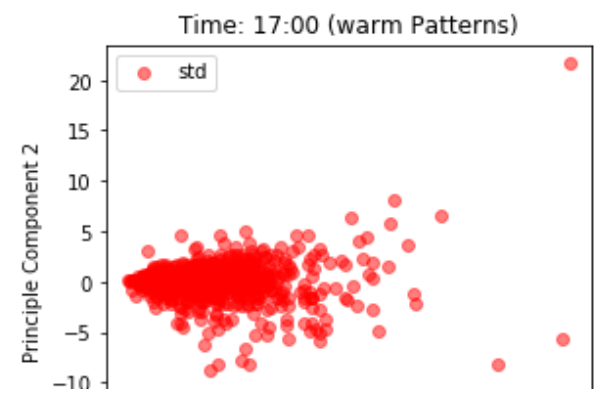

Time: 12:00 (warm Patterns)
std
Principle Component 2
Principle Component 1

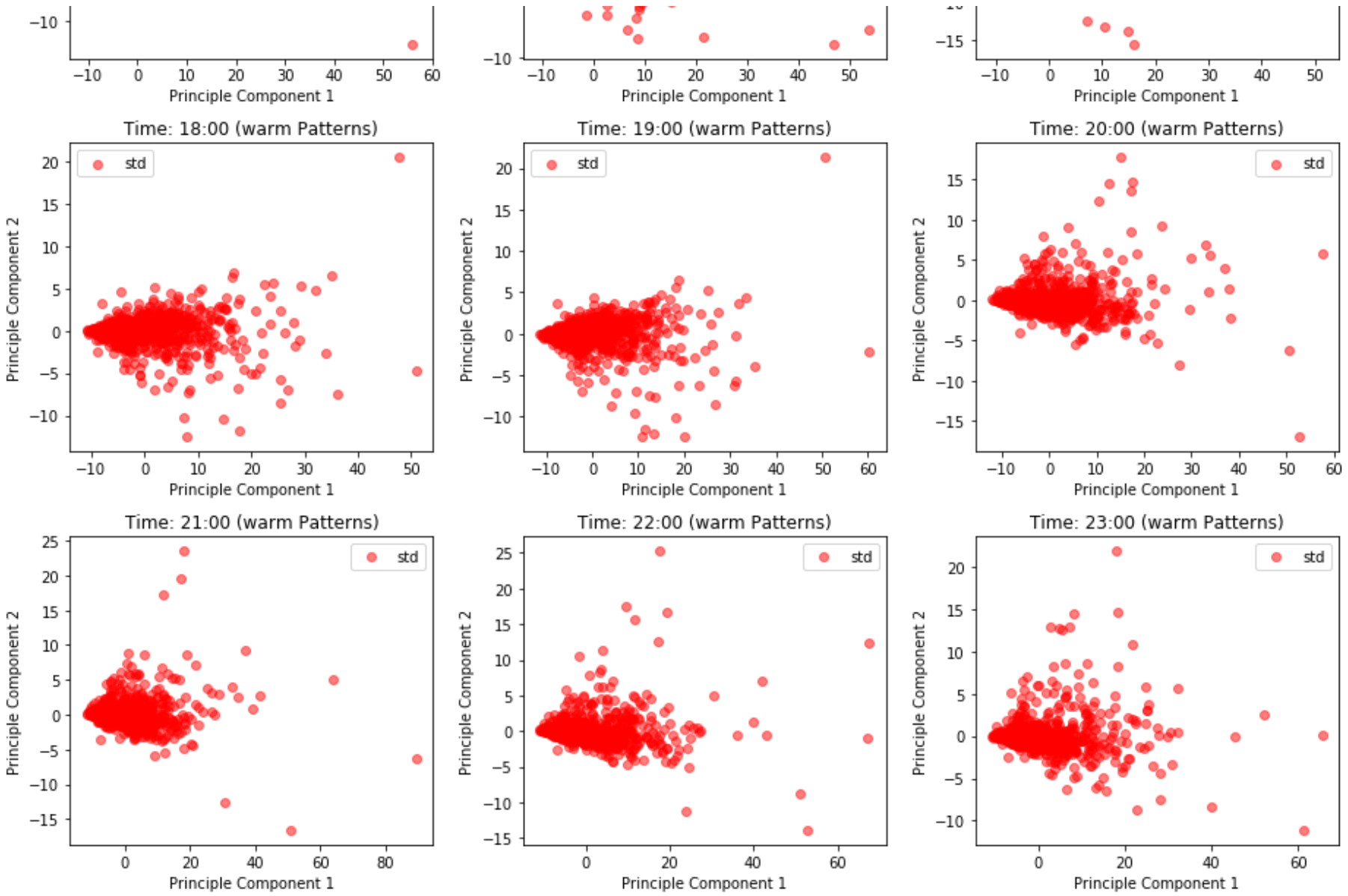

Time: 13:00 (warm Patterns)
std
Principle Component 2
Principle Component 1


Time: 14:00 (warm Patterns)
std
Principle Component 2
Principle Component 1


Time: 15:00 (warm Patterns)
std
Principle Component 2


Time: 16:00 (warm Patterns)
std
Principle Component 2

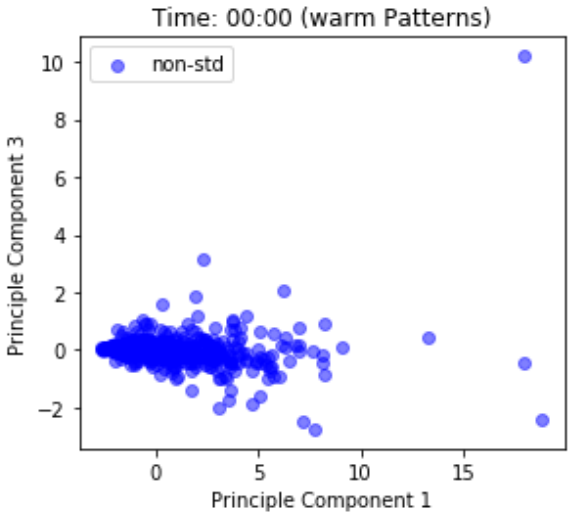

Time: 17:00 (warm Patterns)
std
Principle Component 2

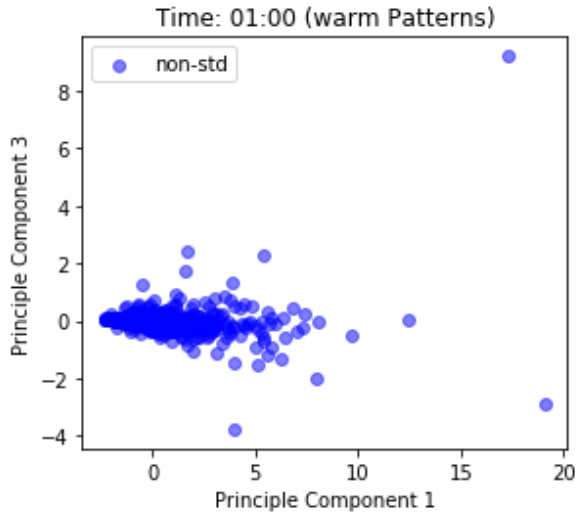


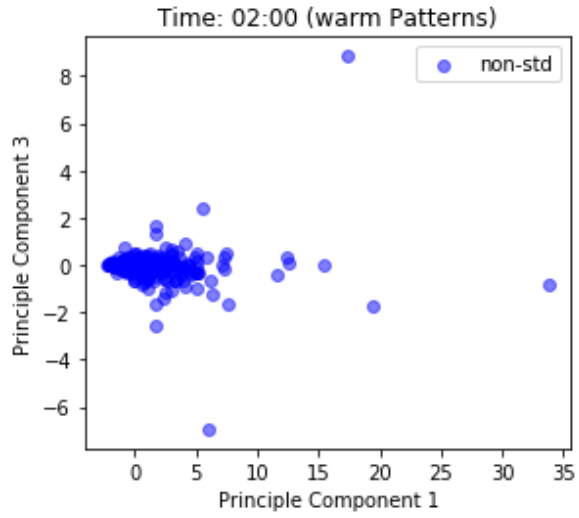
Different from the size-and-variable both standardization in paper 2, we care about the size effect. So we do not perform size standardization. Also in paper2, the author worried about the correlated problems, which wouldn't appear if they didn't use the no-size-scaled PCA to perform K-mean clustering. So the reason that problem seems to exist is they used the different PCA as the input of K-mean clustering and get the different labels, then put these labels to the previous PCA method, it seems that in Fig2 of their paper, each "category" seems has PC1 and PC2 linearly correlated, but this problem shouldn't have.

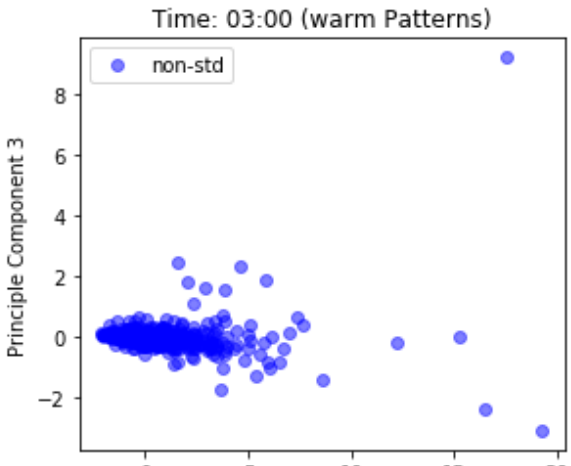
Component 1 and Component 3 relationship is shown below

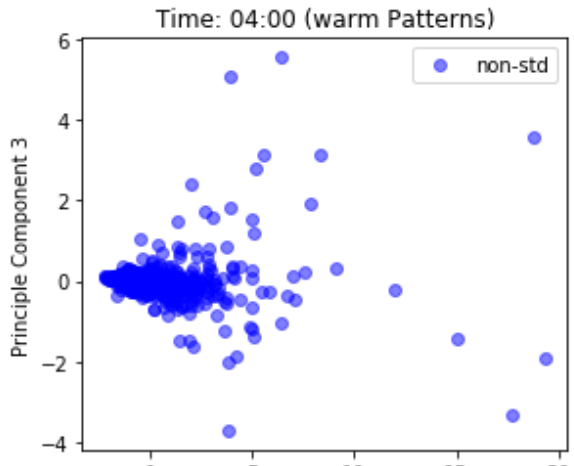
In [12]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[0], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[0], principleDf[2], c = 'red', label = 'std', alpha = 0.1)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[0], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[0], principleDf[2], c = 'red', label = 'std', alpha = 0.1)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

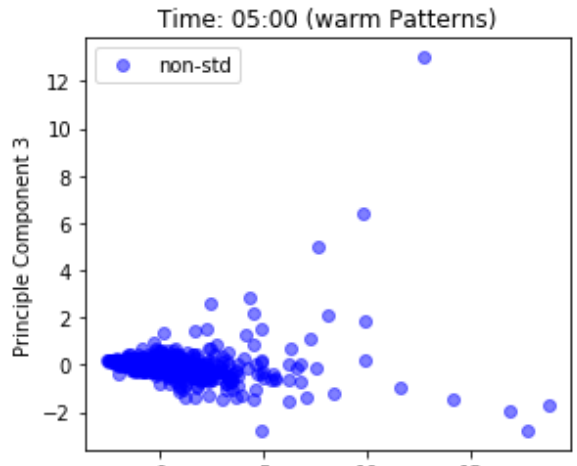

Time: 00:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

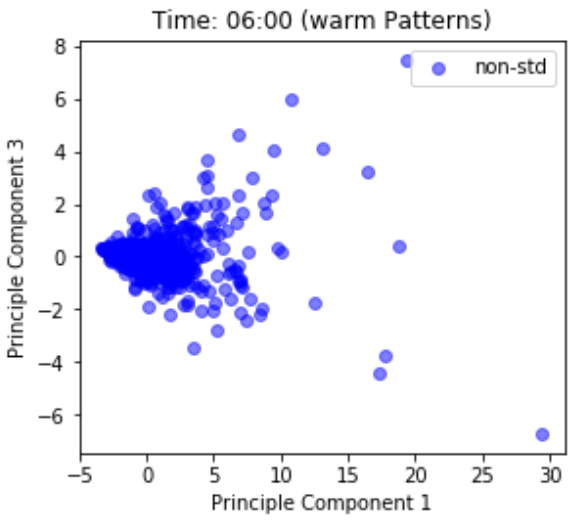

Time: 01:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

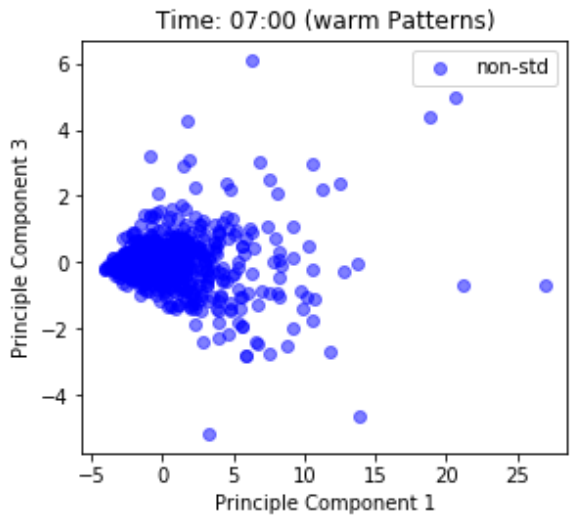

Time: 02:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

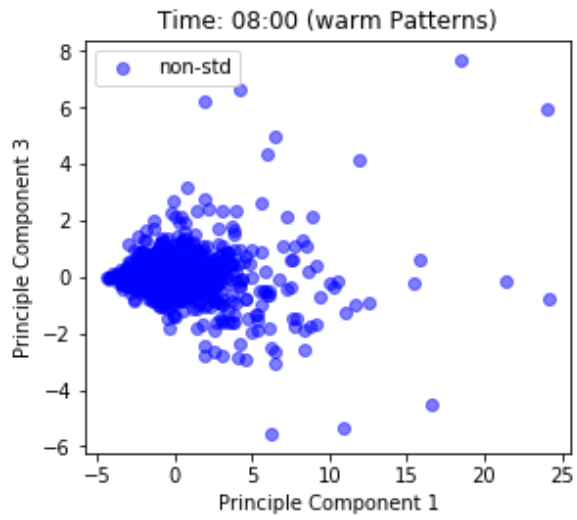

Time: 03:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

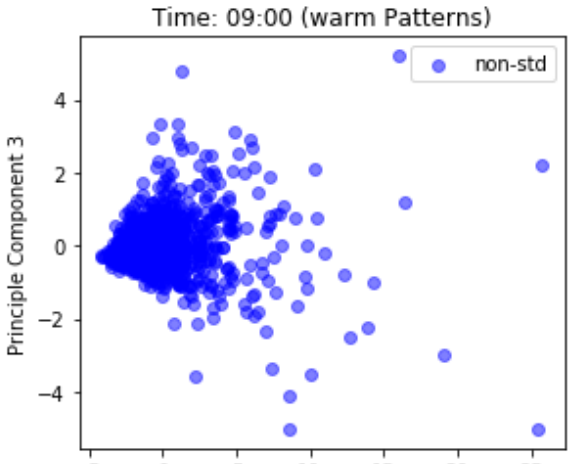

Time: 04:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

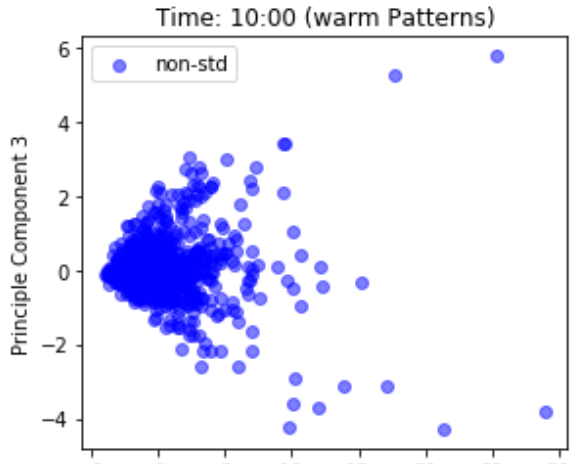

Time: 05:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

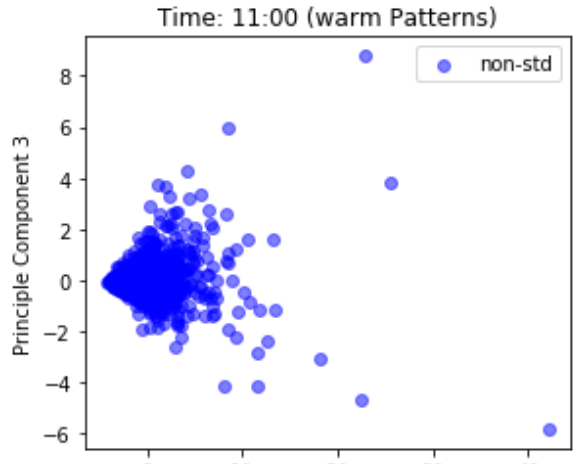

Time: 06:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

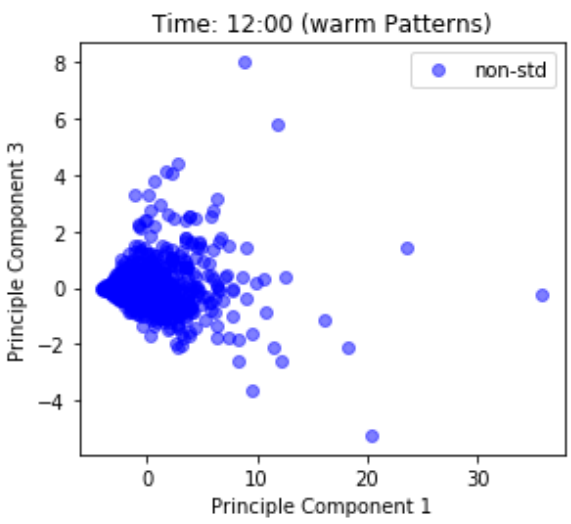

Time: 07:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

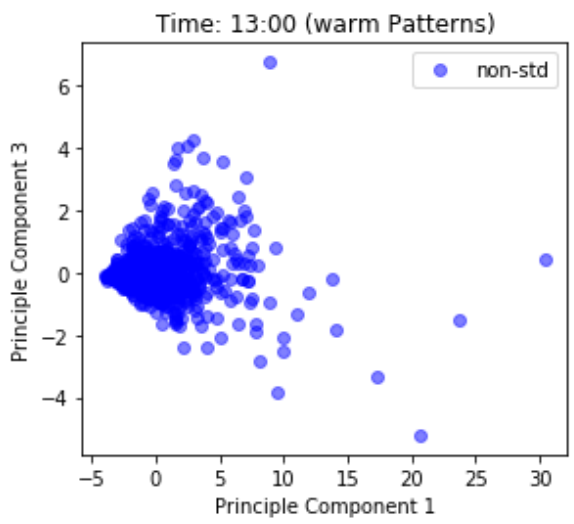

Time: 08:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

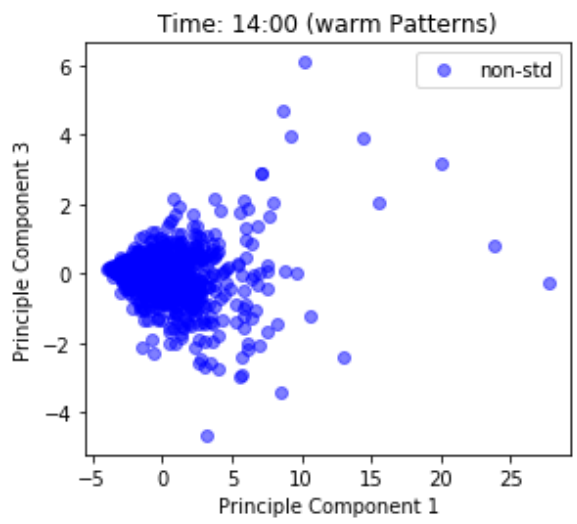

Time: 09:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

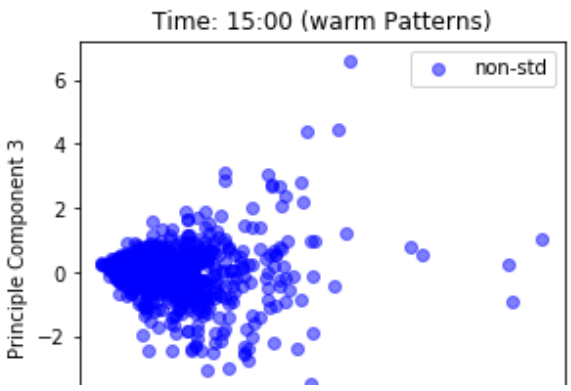

Time: 10:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

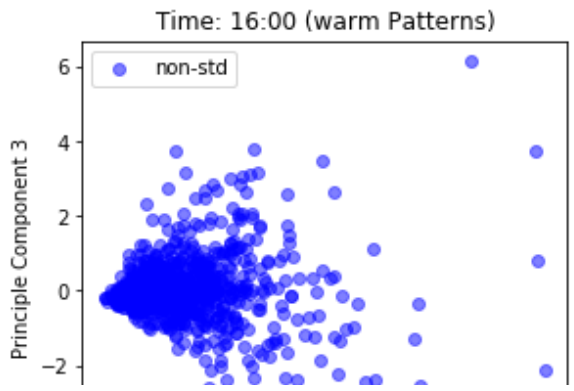

Time: 11:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

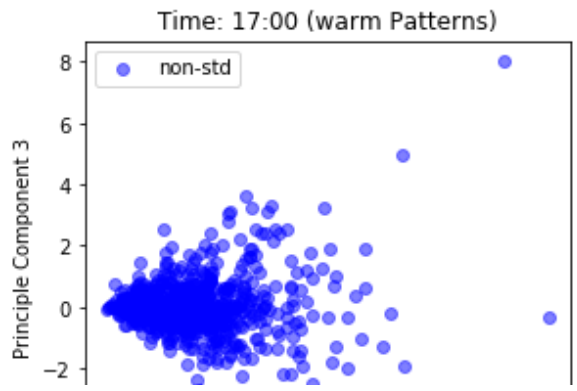

Time: 12:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

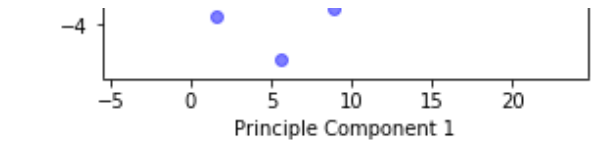

Time: 13:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

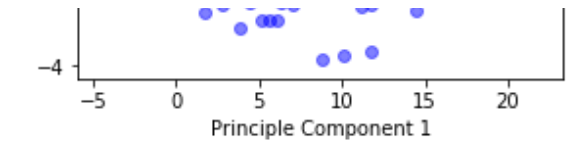

Time: 14:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

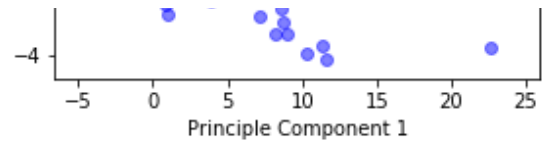

Time: 15:00 (warm Patterns)
non-std
Principle Component 3

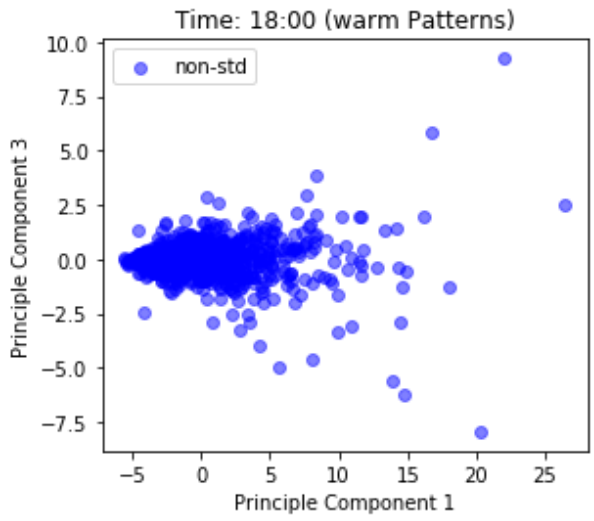

Time: 16:00 (warm Patterns)
non-std
Principle Component 3

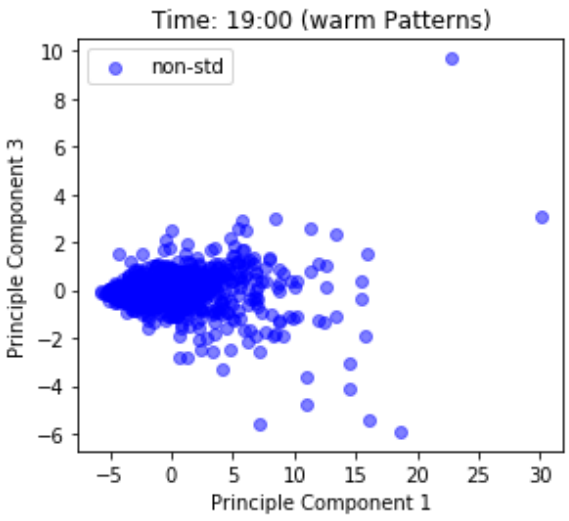

Time: 17:00 (warm Patterns)
non-std
Principle Component 3

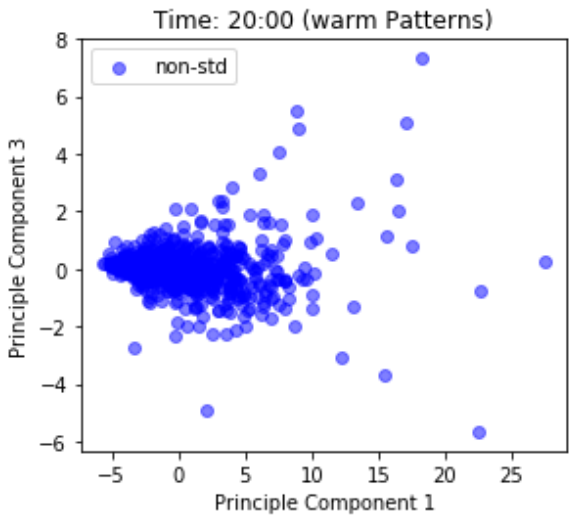

Principle Component 1

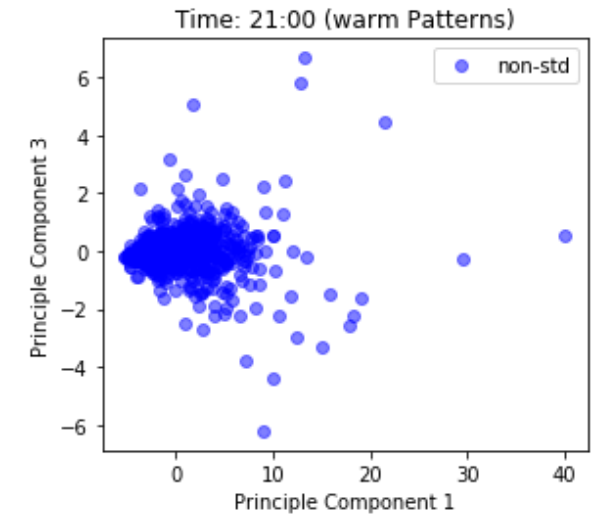

Principle Component 1

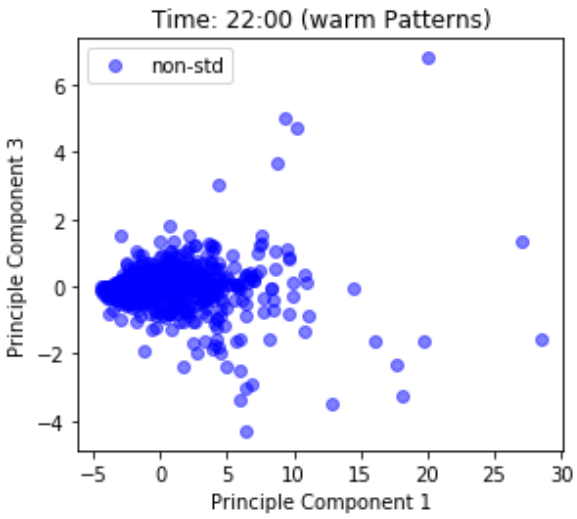

Principle Component 1

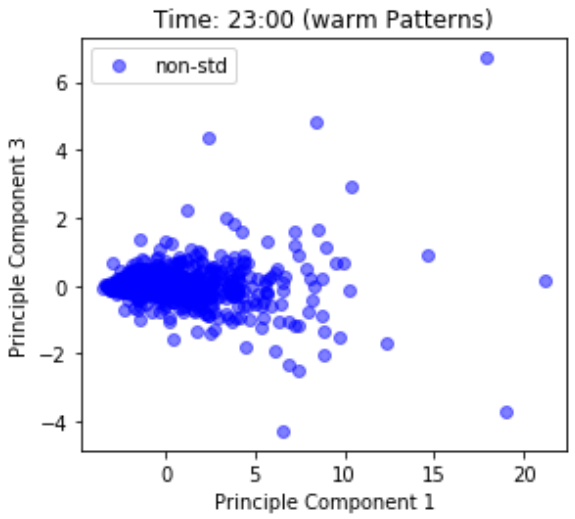

Time: 18:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1


Time: 19:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

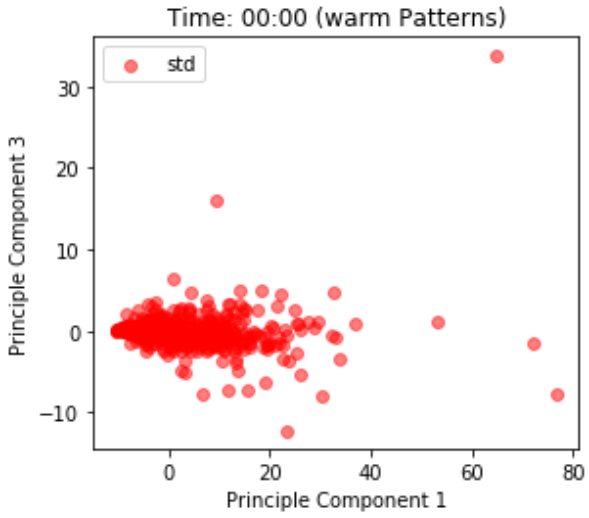

Time: 20:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

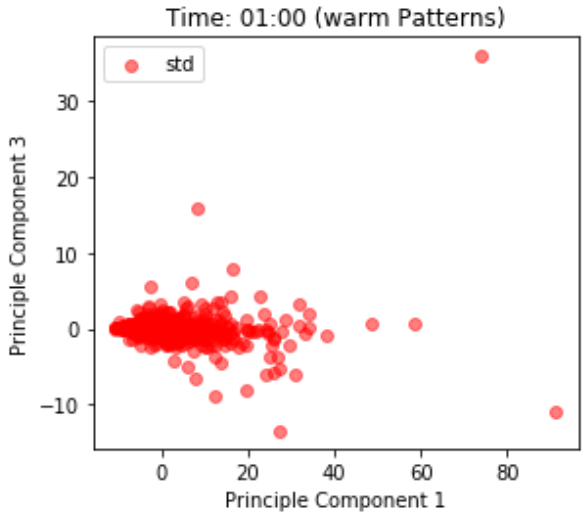

Time: 21:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

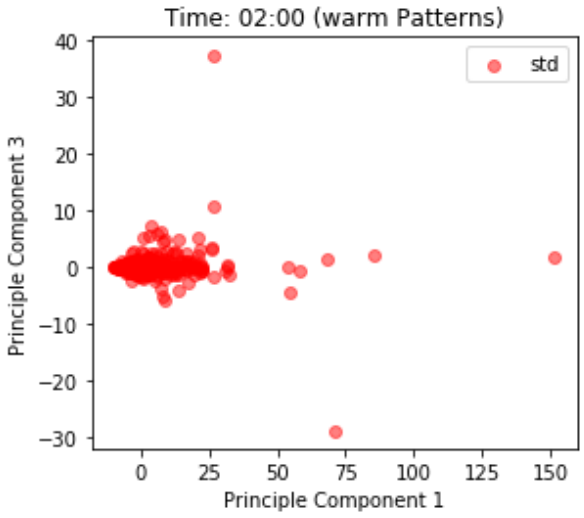

Time: 22:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

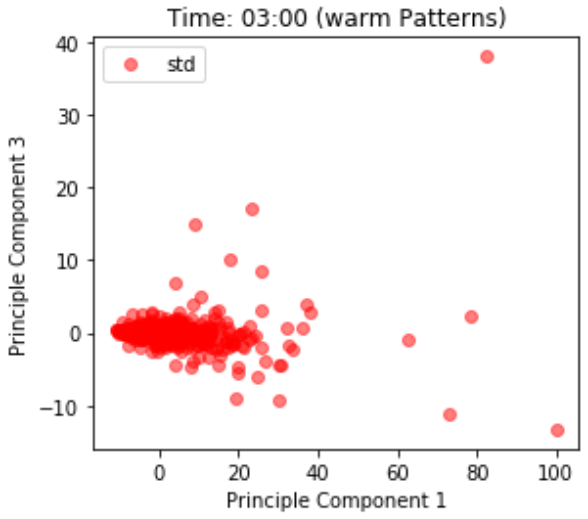

Time: 23:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 1

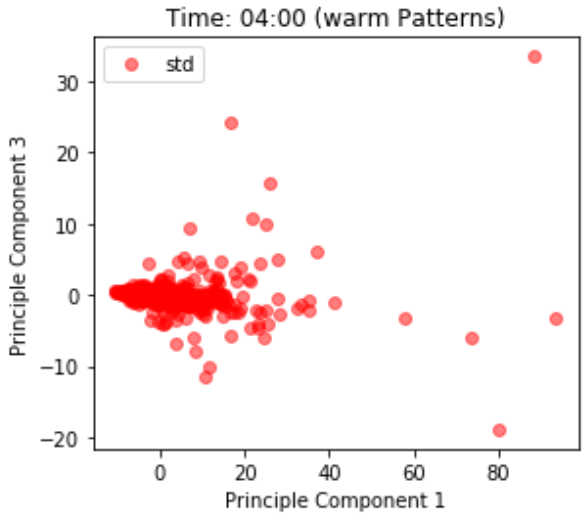
In [13]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[0], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[0], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[0], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[0], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

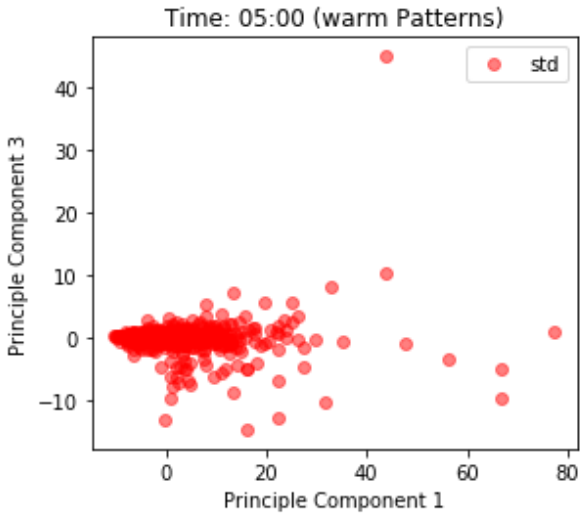

Time: 00:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

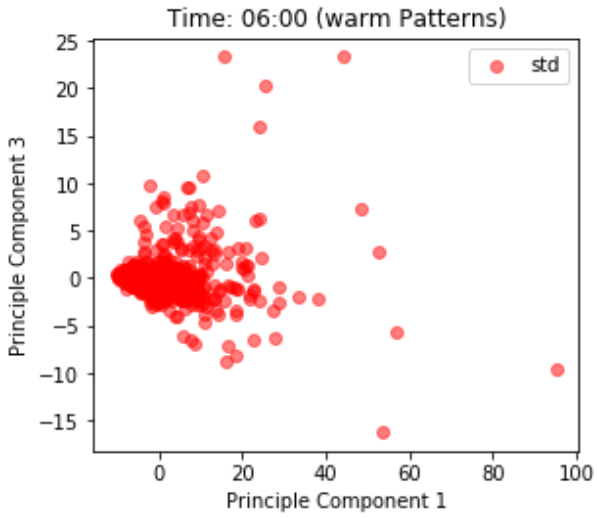

Time: 01:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

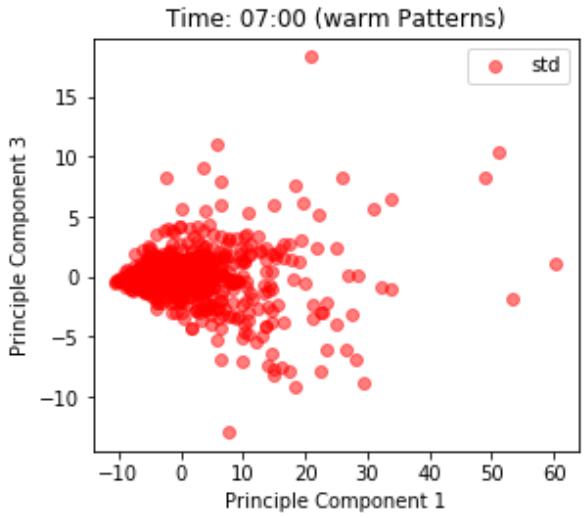

Time: 02:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

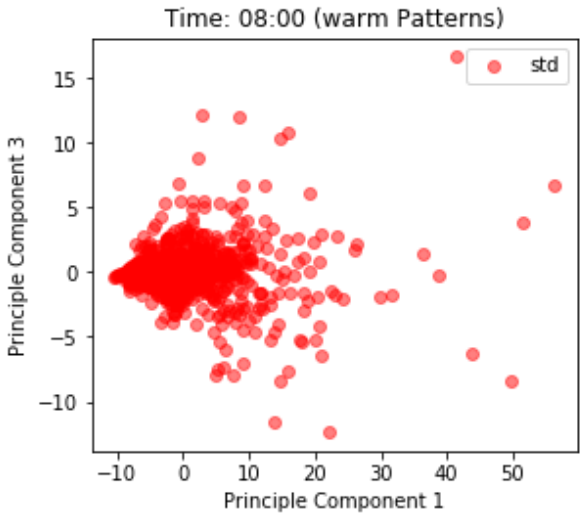

Time: 03:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

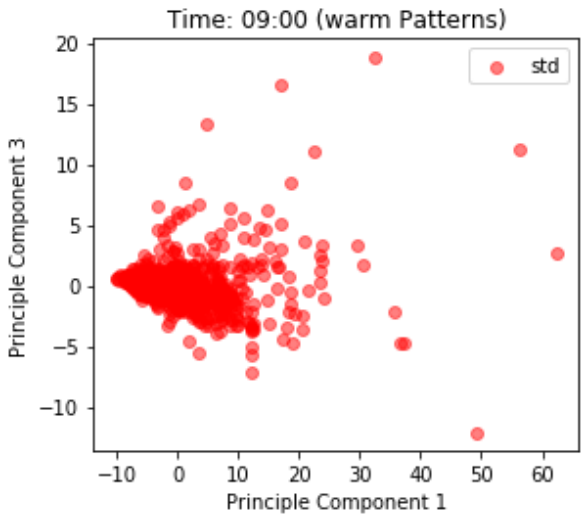

Time: 04:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

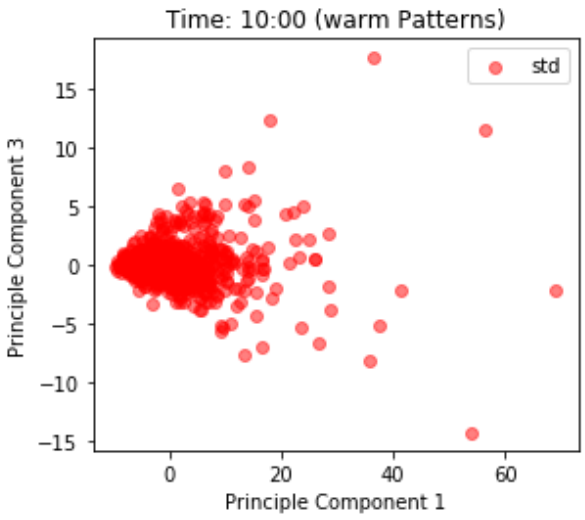

Time: 05:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

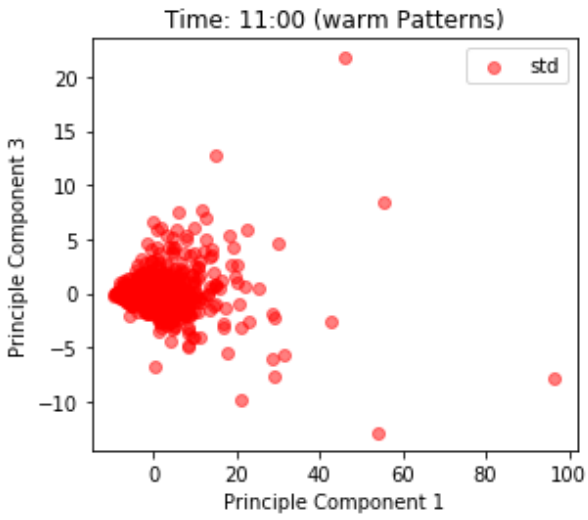

Time: 06:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

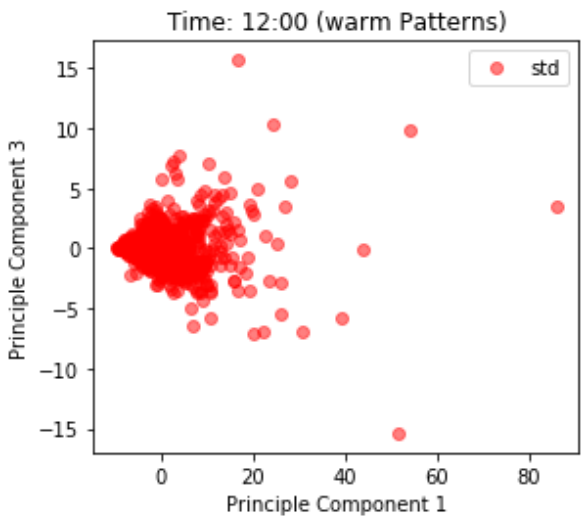

Time: 07:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

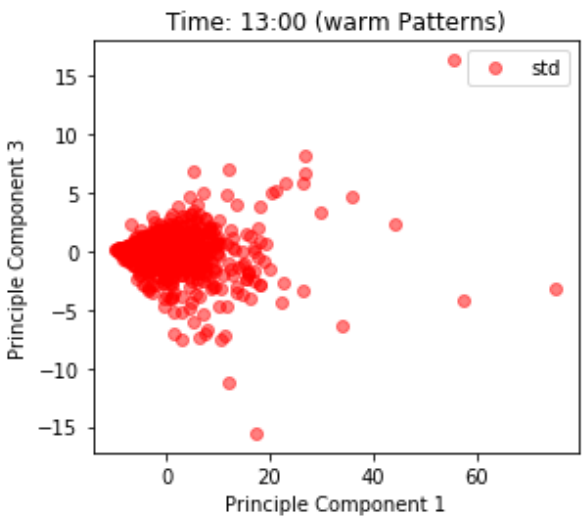

Time: 08:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

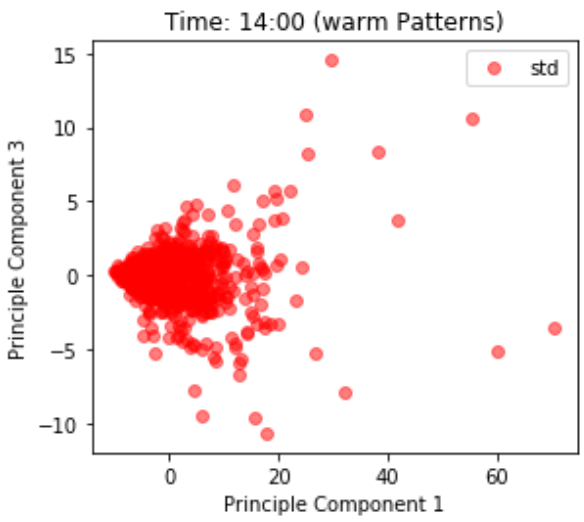

Time: 09:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

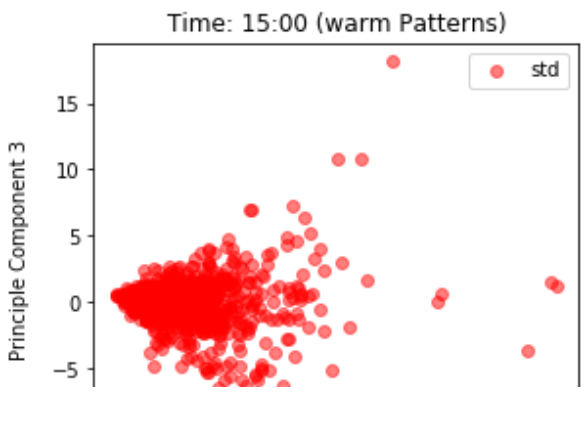

Time: 10:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

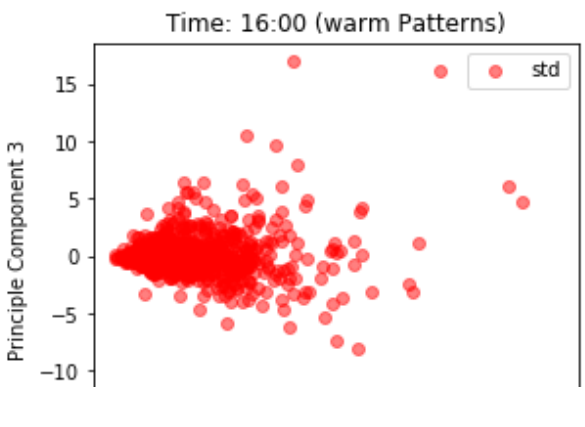

Time: 11:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

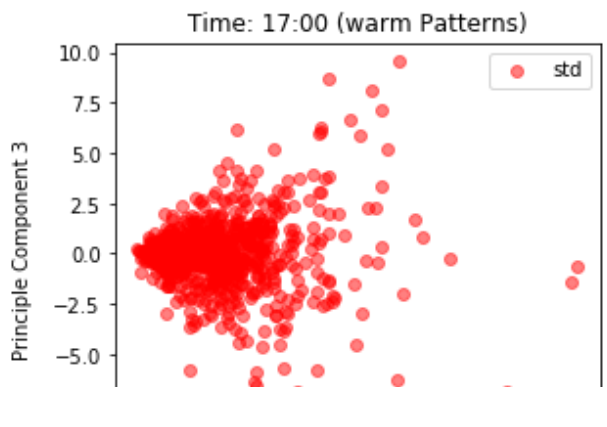

Time: 12:00 (warm Patterns)
std
Principle Component 3
Principle Component 1

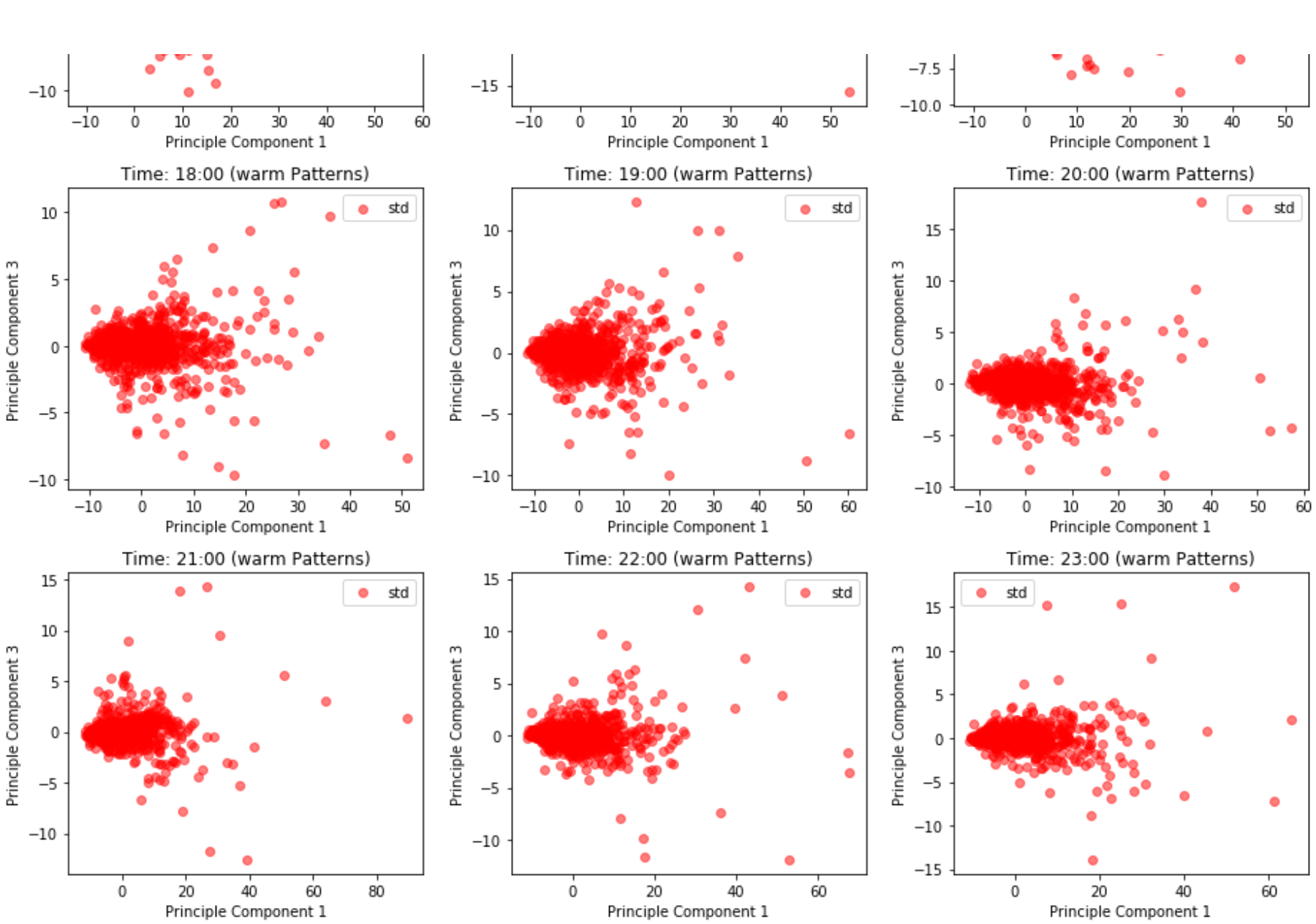

Time: 13:00 (warm Patterns)
std
Principle Component 3
Principle Component 1


Time: 14:00 (warm Patterns)
std
Principle Component 3
Principle Component 1


Time: 15:00 (warm Patterns)
std
Principle Component 3

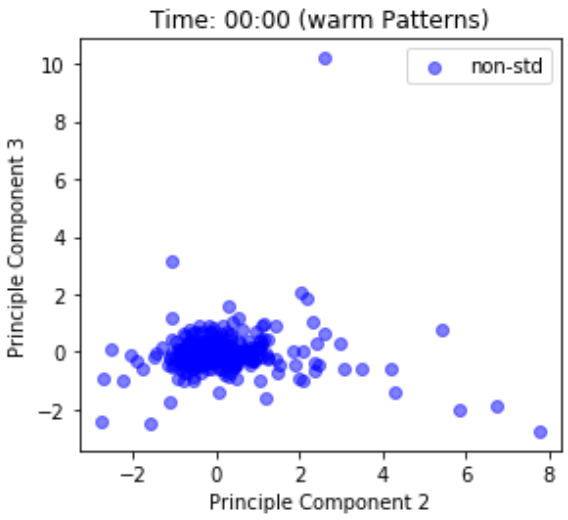

Time: 16:00 (warm Patterns)
std
Principle Component 3

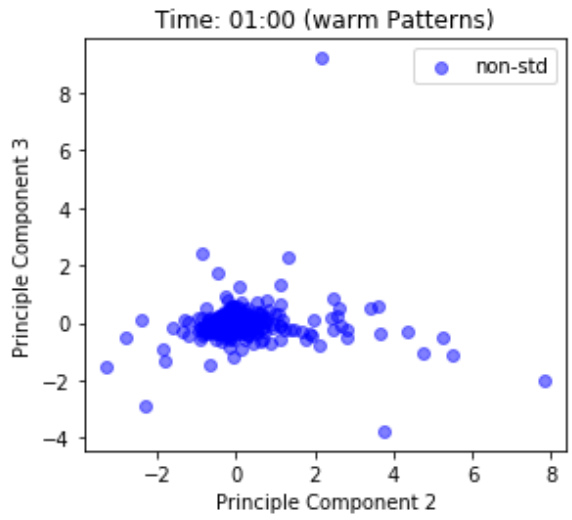

Time: 17:00 (warm Patterns)
std
Principle Component 3

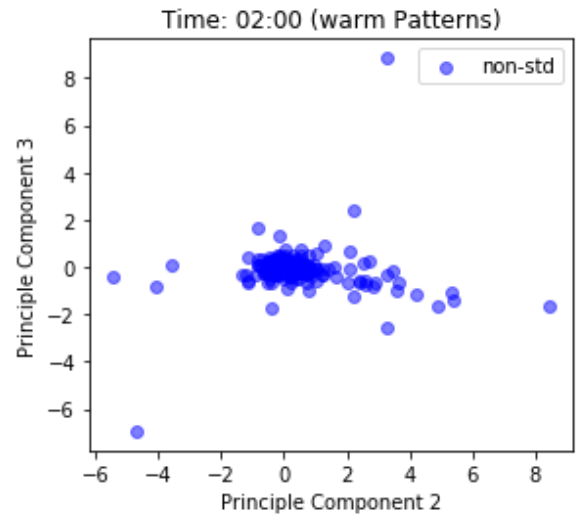


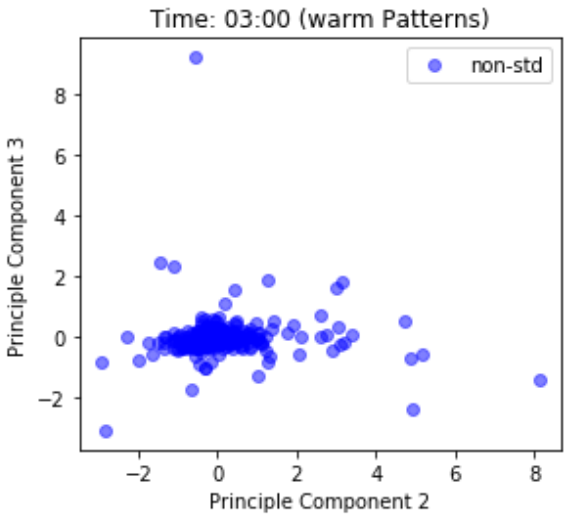
Component 2 and Component 3 relationship is shown below

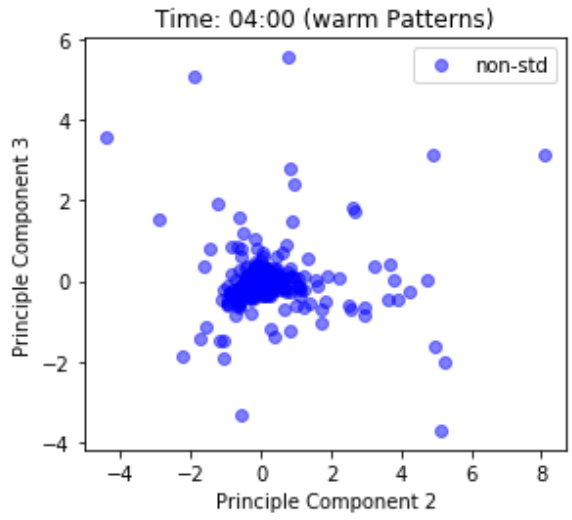
In [14]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[1], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[1], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(principleDf_nstd[1], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
#         ax.scatter(principleDf[1], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

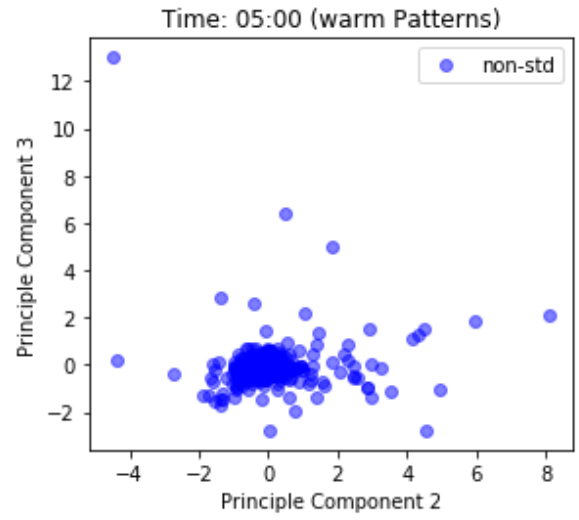

Time: 00:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

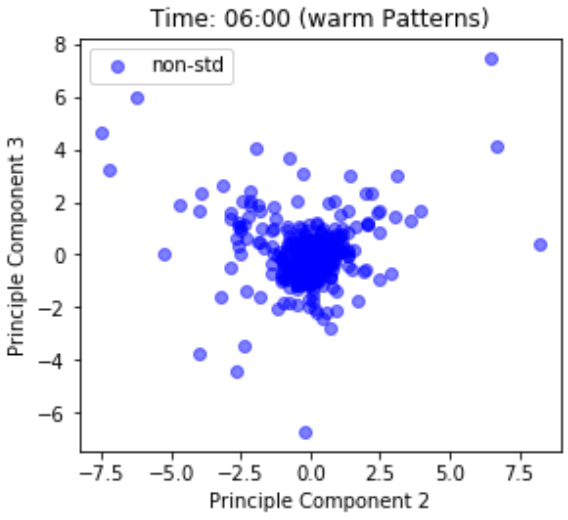

Time: 01:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

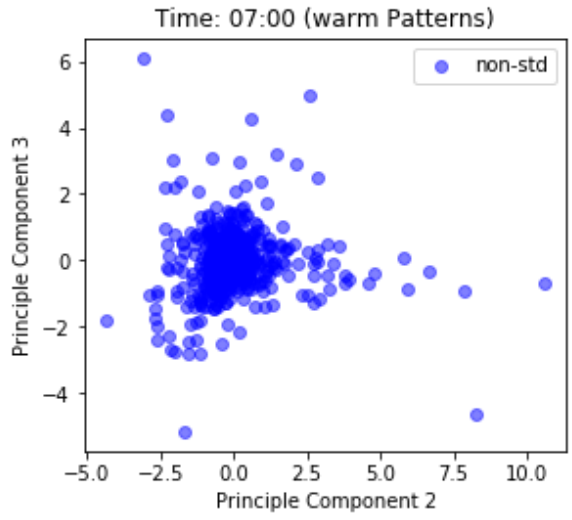

Time: 02:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

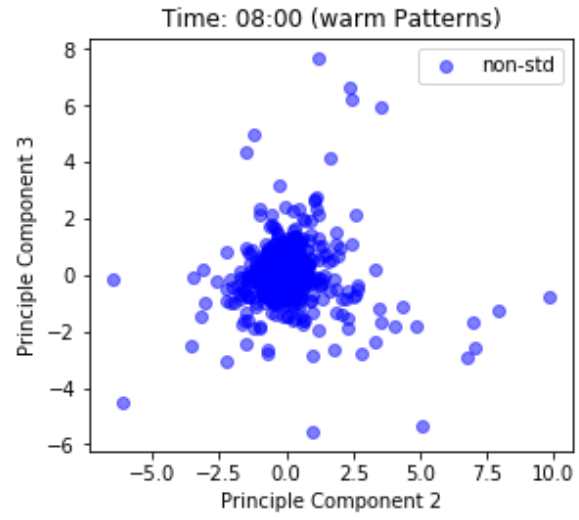

Time: 03:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

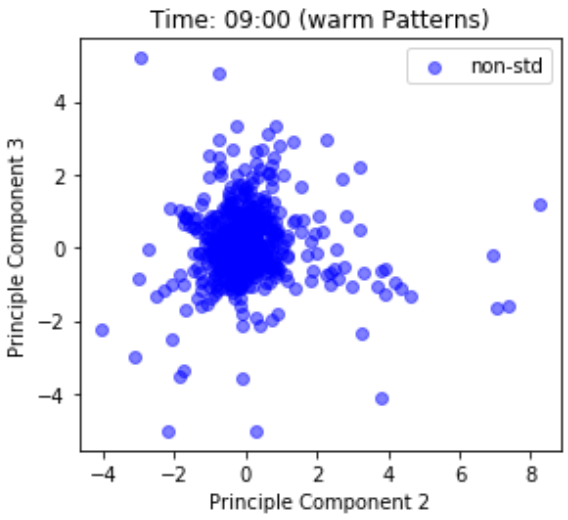

Time: 04:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

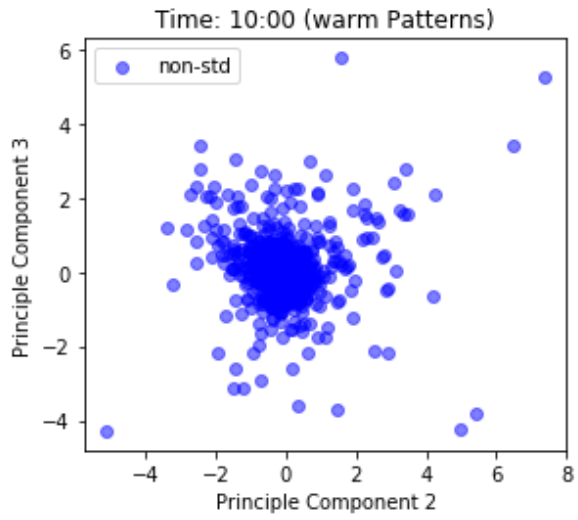

Time: 05:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

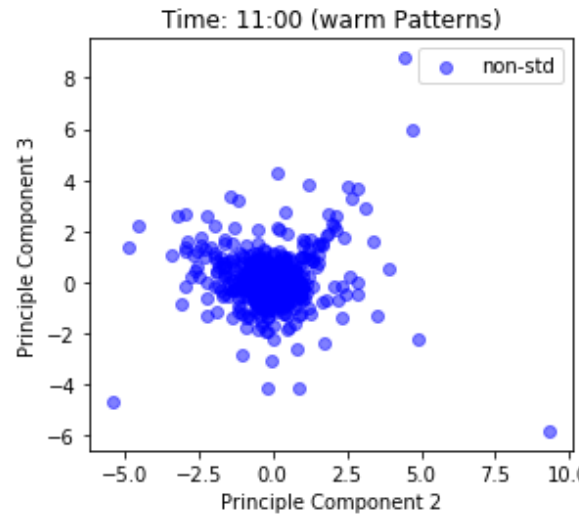

Time: 06:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

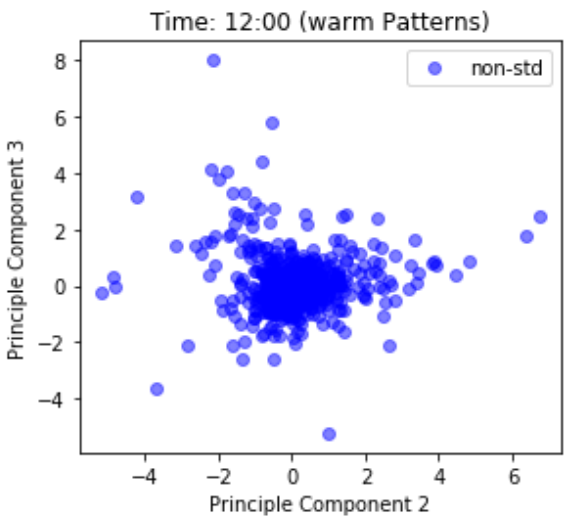

Time: 07:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

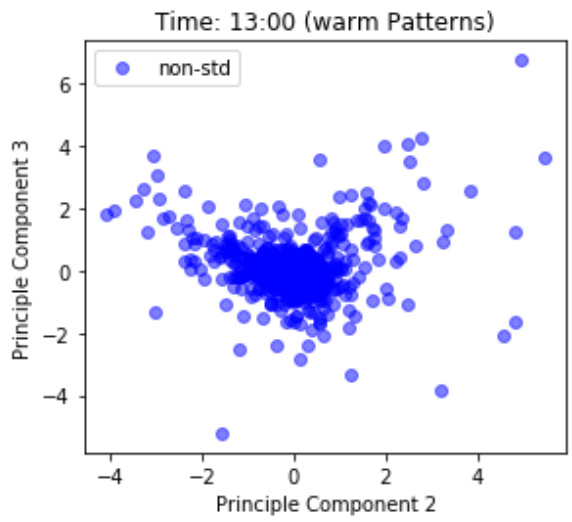

Time: 08:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

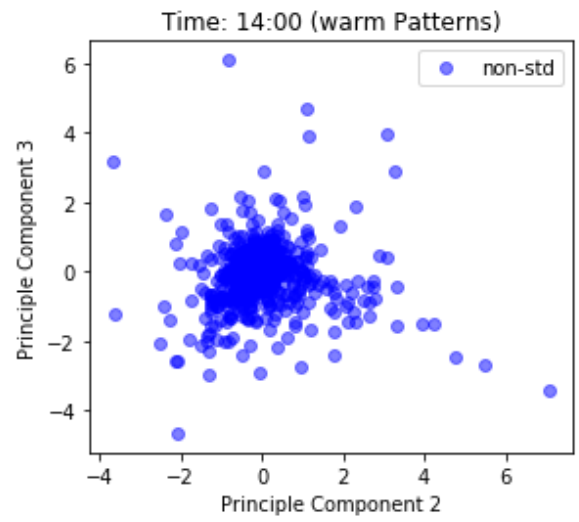

Time: 09:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

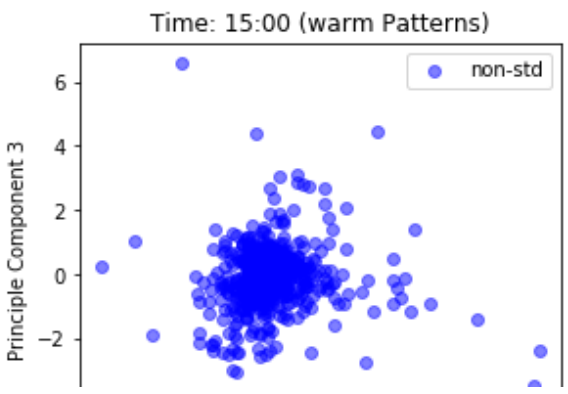

Time: 10:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

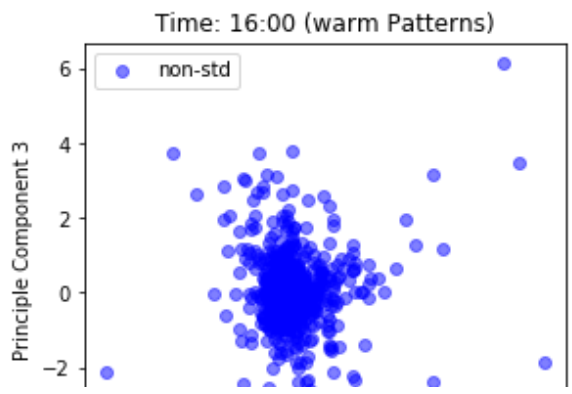

Time: 11:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

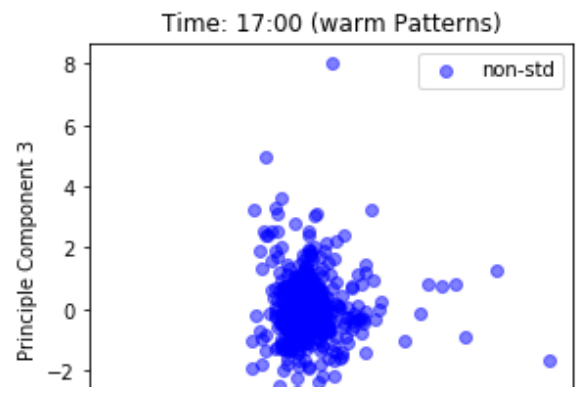

Time: 12:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

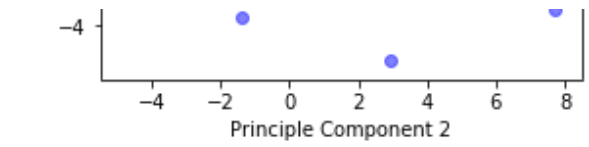

Time: 13:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

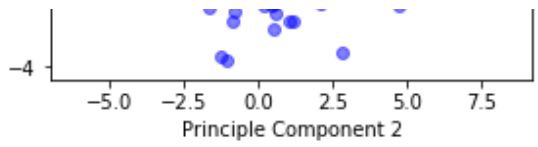

Time: 14:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

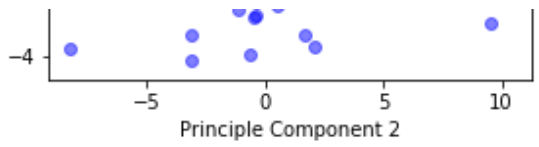

Time: 15:00 (warm Patterns)
non-std
Principle Component 3

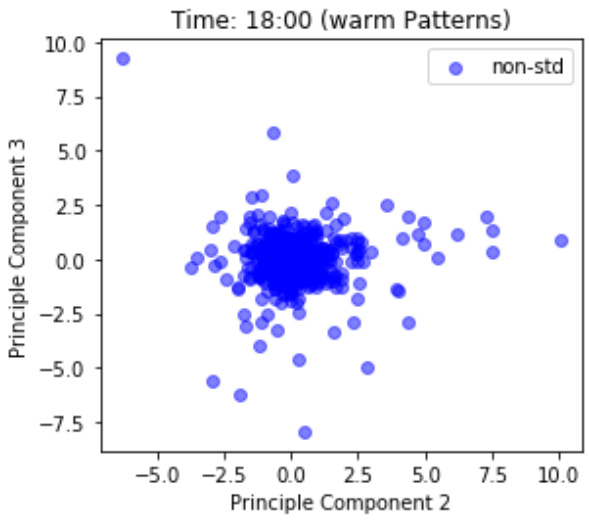

Time: 16:00 (warm Patterns)
non-std
Principle Component 3

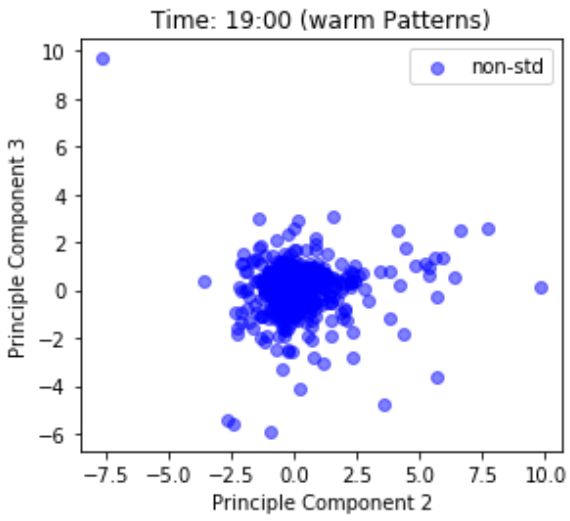

Time: 17:00 (warm Patterns)
non-std
Principle Component 3

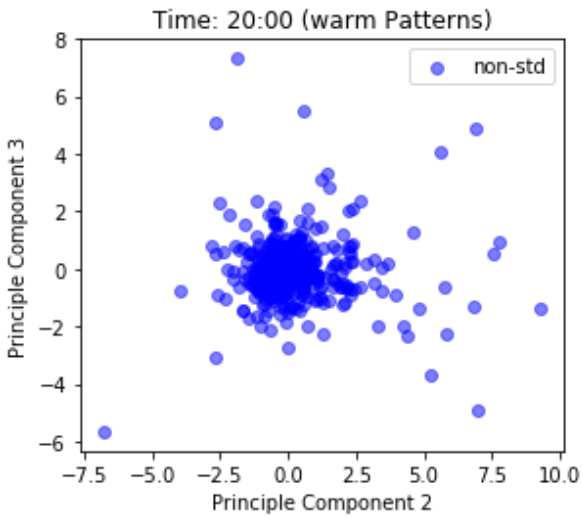

Principle Component 2

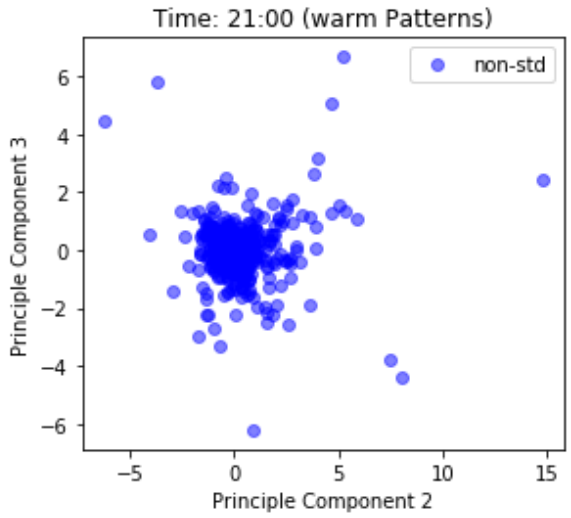

Principle Component 2

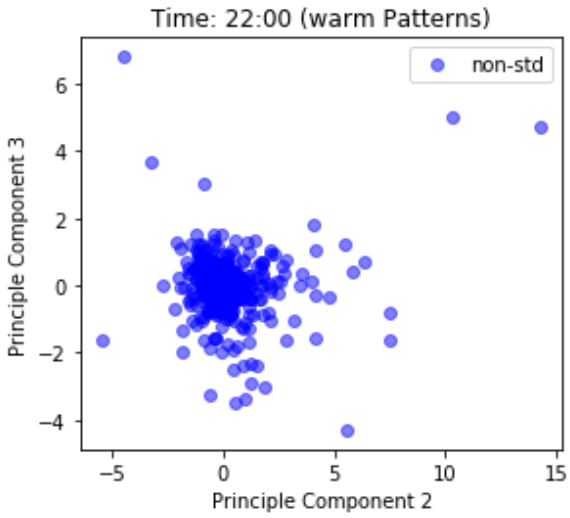

Principle Component 2

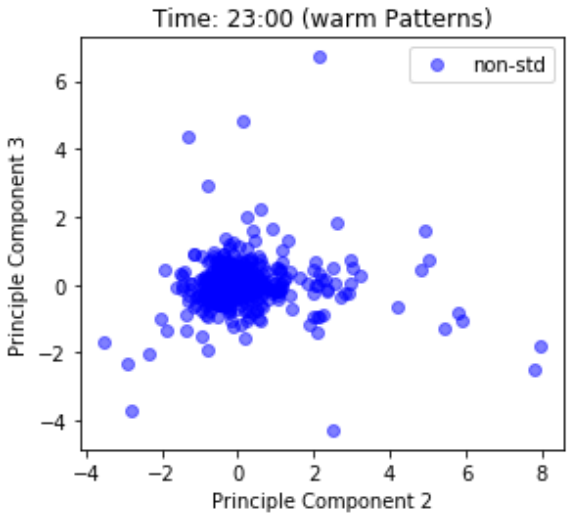

Time: 18:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2


Time: 19:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

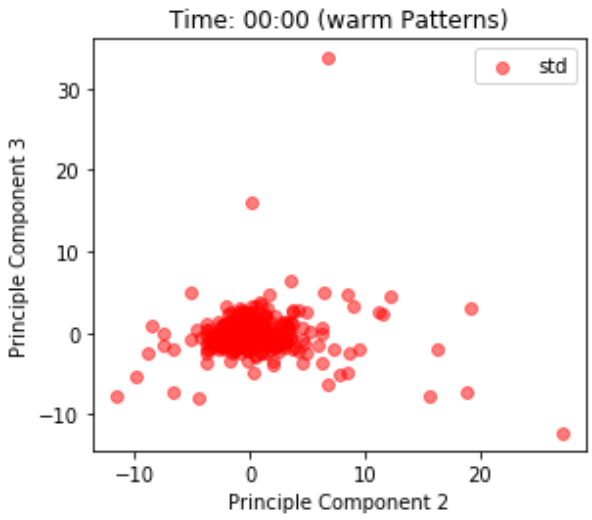

Time: 20:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

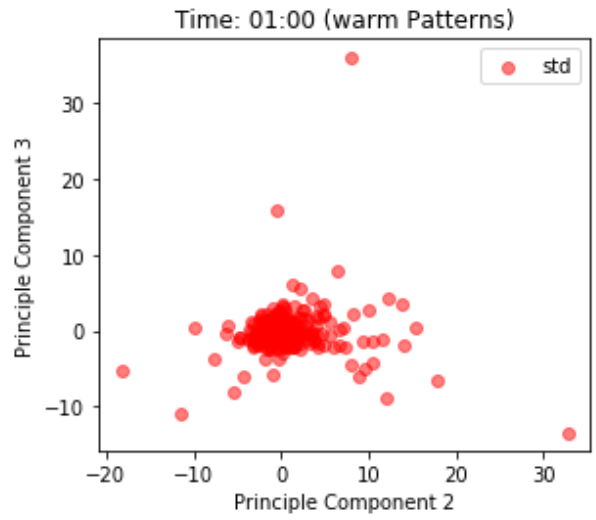

Time: 21:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

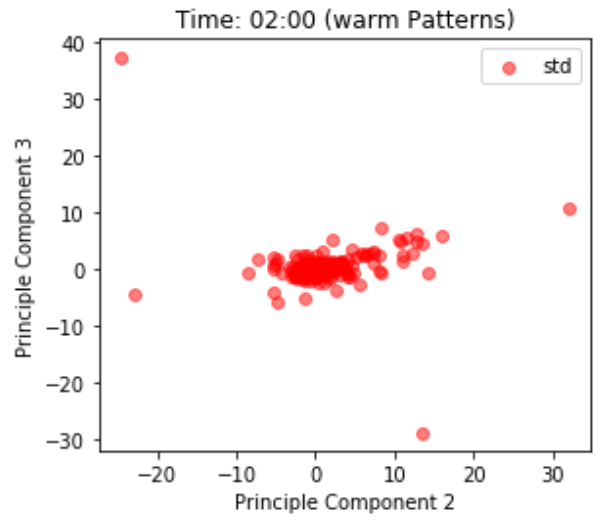

Time: 22:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

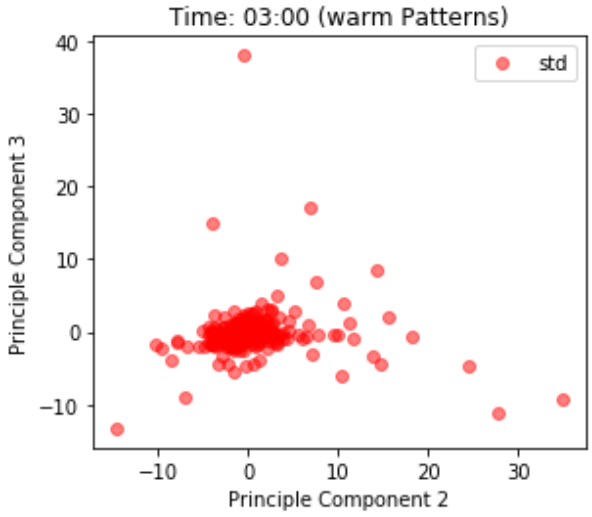

Time: 23:00 (warm Patterns)
non-std
Principle Component 3
Principle Component 2

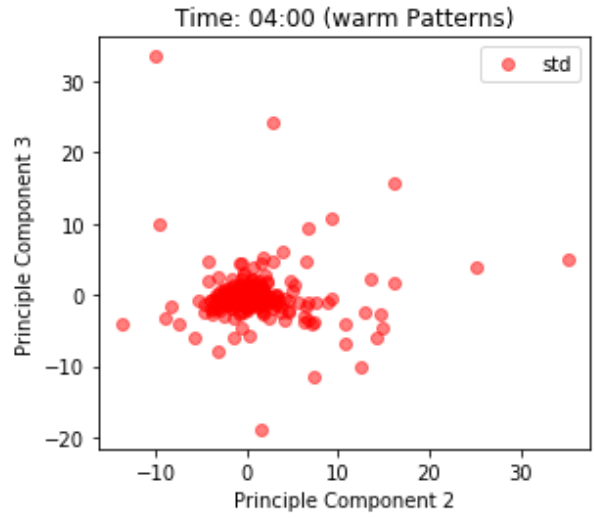
In [15]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[1], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[1], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         ax.scatter(principleDf_nstd[1], principleDf_nstd[2], c = 'blue', label = 'non-std', alpha = 0.5)
        # PCA with standardization
        x = x.values
        x = StandardScaler().fit_transform(x)
        pca = PCA()# keep all columns to cover 100% variance
        principleComponents = pca.fit_transform(x)
        principleDf = pd.DataFrame(data = principleComponents)
        ax.scatter(principleDf[1], principleDf[2], c = 'red', label = 'std', alpha = 0.5)
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

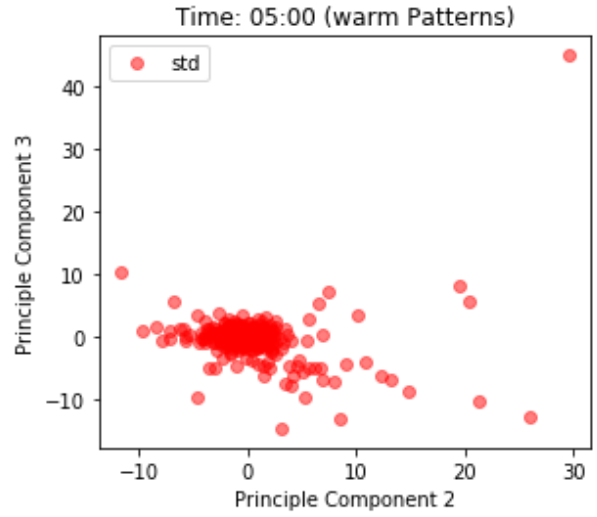

Time: 00:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

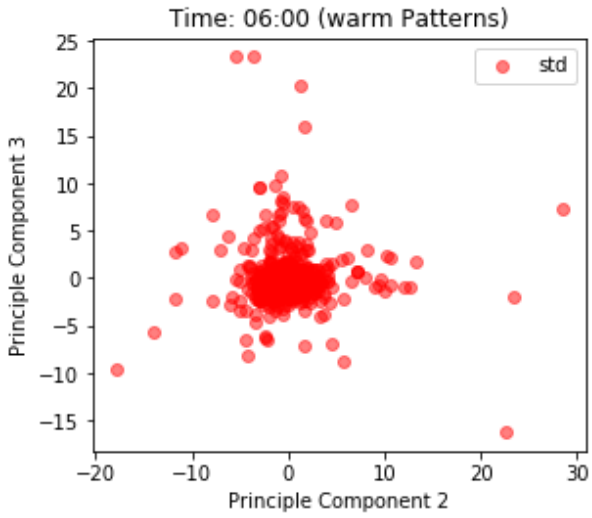

Time: 01:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

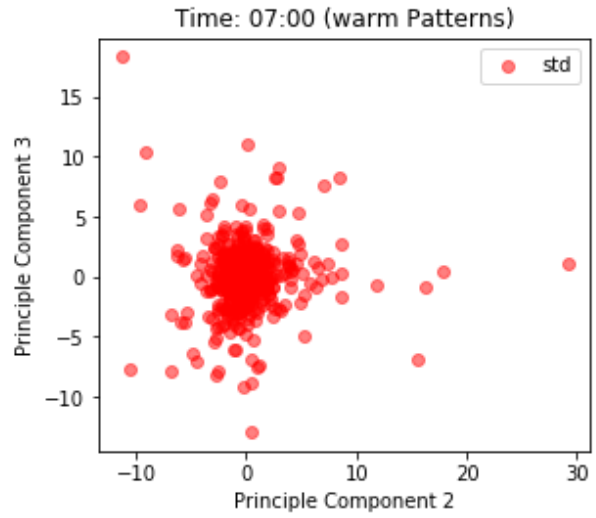

Time: 02:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

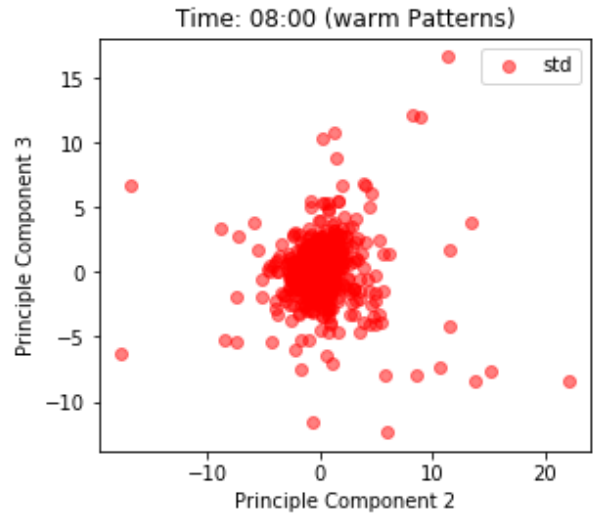

Time: 03:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

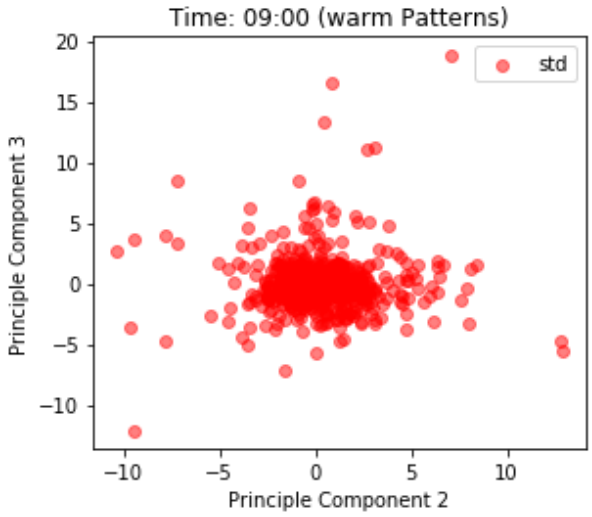

Time: 04:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

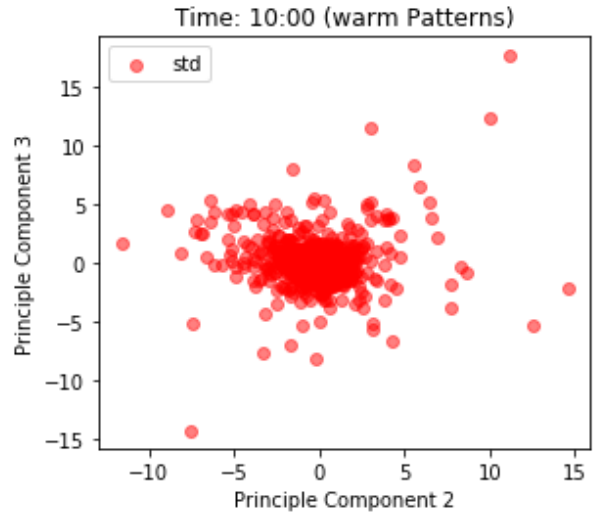

Time: 05:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

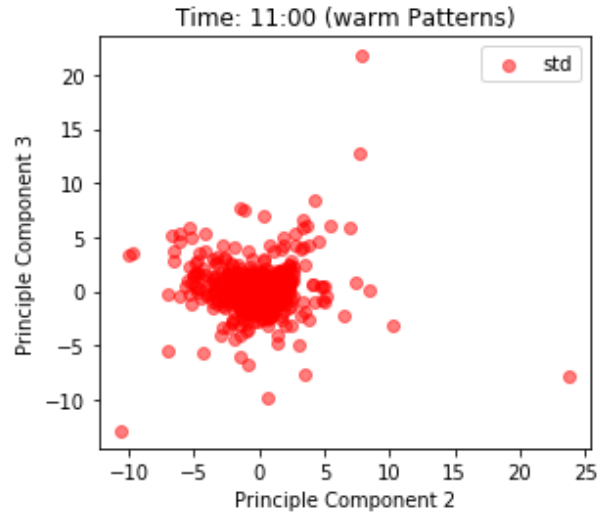

Time: 06:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

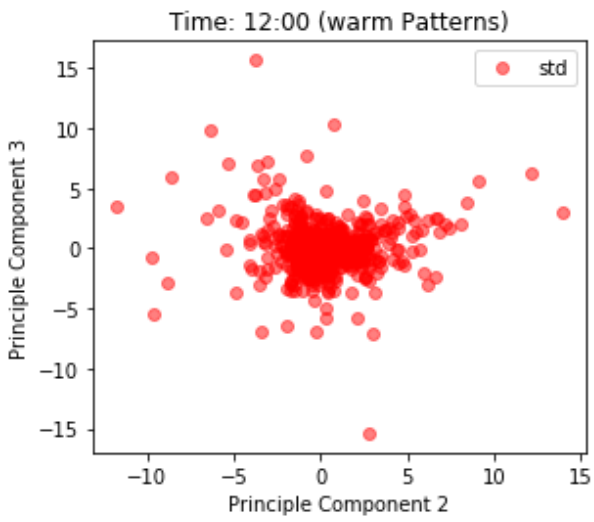

Time: 07:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

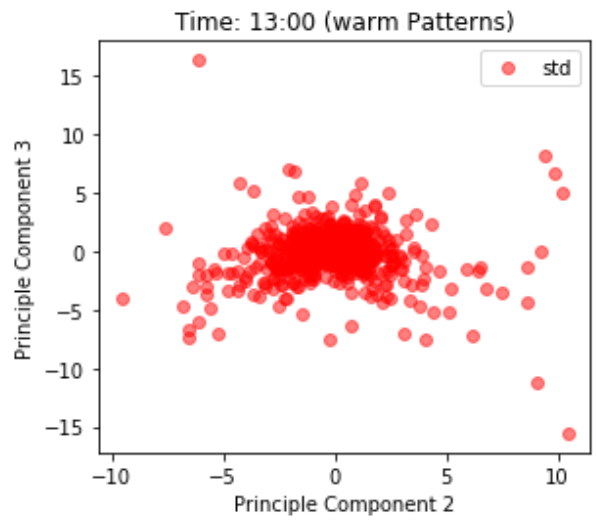

Time: 08:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

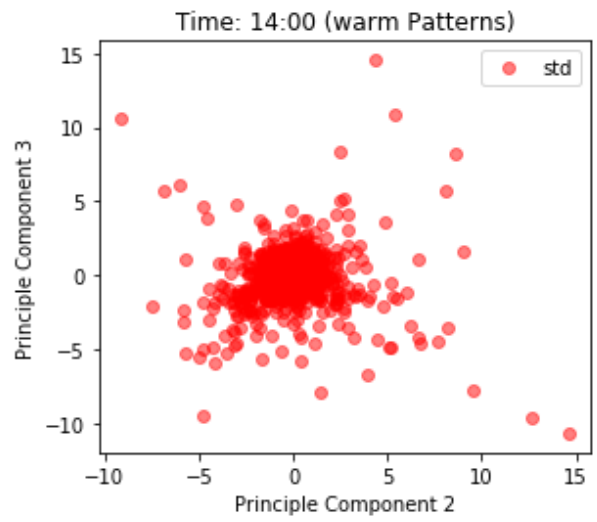

Time: 09:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

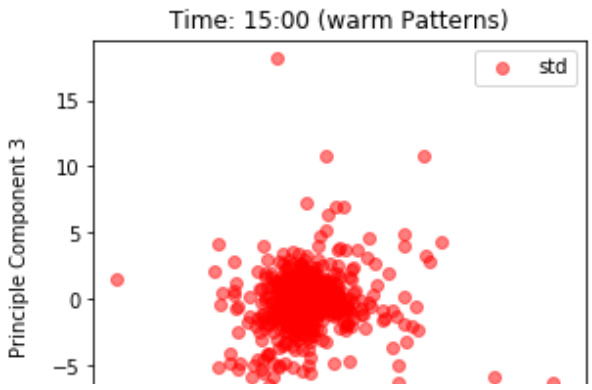

Time: 10:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

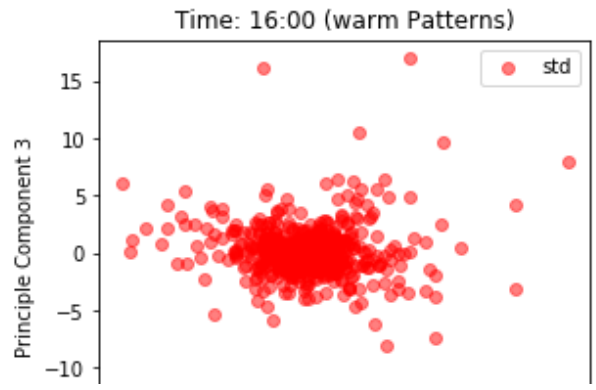

Time: 11:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

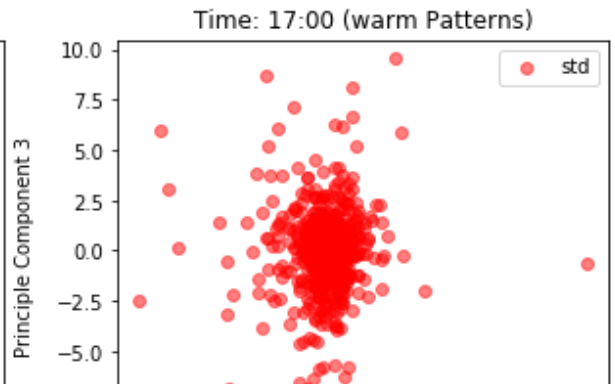

Time: 12:00 (warm Patterns)
std
Principle Component 3
Principle Component 2

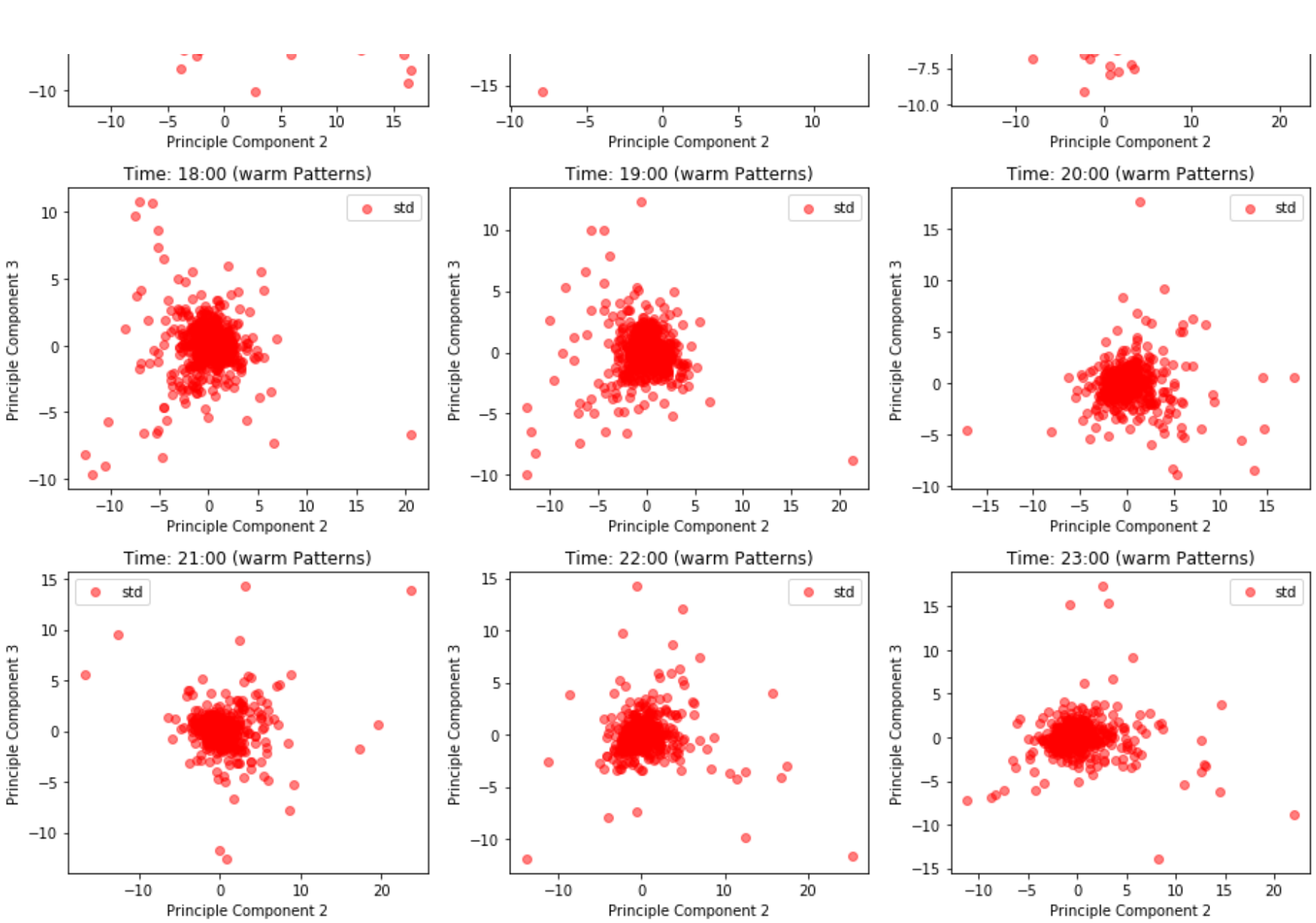

Time: 13:00 (warm Patterns)
std
Principle Component 3
Principle Component 2


Time: 14:00 (warm Patterns)
std
Principle Component 3
Principle Component 2


Time: 15:00 (warm Patterns)
std
Principle Component 3


Time: 16:00 (warm Patterns)
std
Principle Component 3


Time: 17:00 (warm Patterns)
std
Principle Component 3

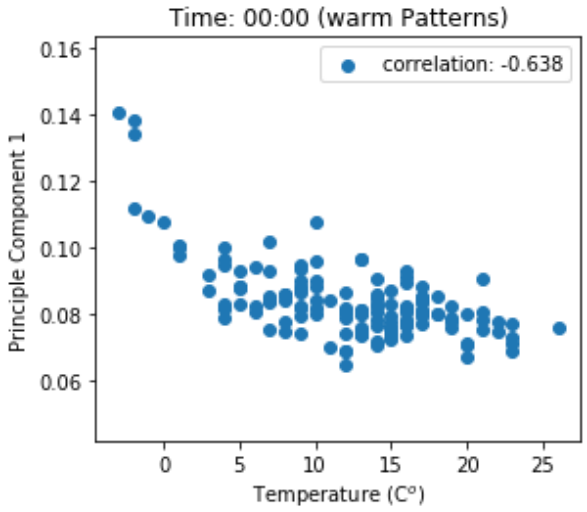


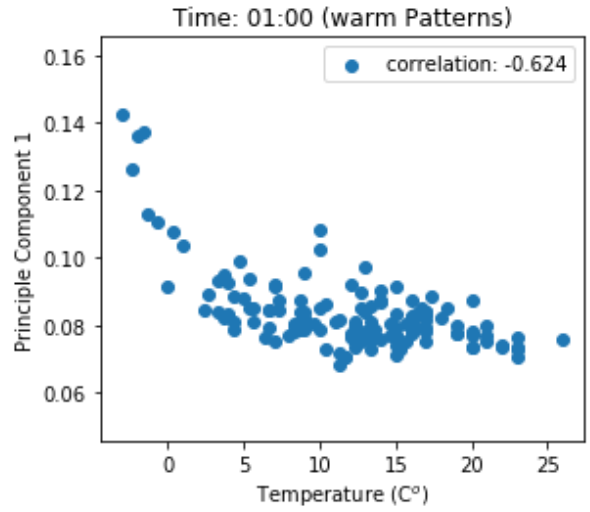
The above three types of plots show that std don't have significant advantage in this case, so we will use non-std in our later analysis.

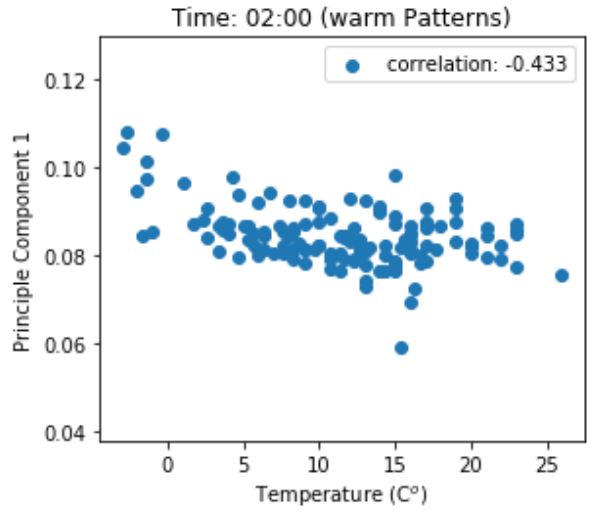
Let's try to interprete PCs' meaning:

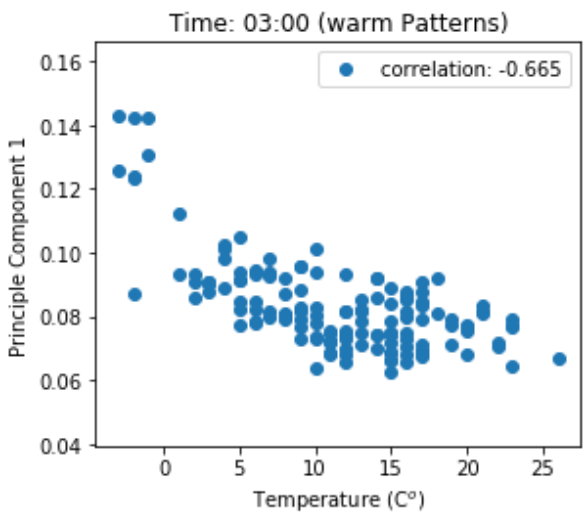
(1) PC1's coefficients vs Temperature:

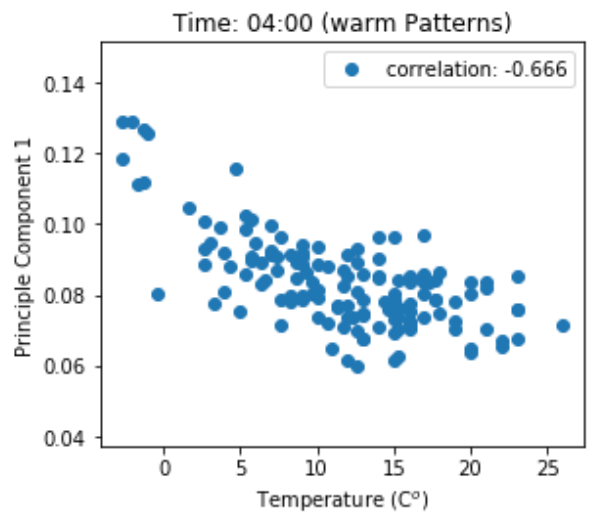
In [16]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(df_wea1h_nf_warm.TempC[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')], pca_ns
td.components_[0], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[0], df_wea1h_nf_warm.
TempC[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 1')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(df_wea1h_nf_warm.TempC[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')], pca_nstd.com
ponents_[0], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[0], df_wea1h_nf_warm.TempC[
df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 1')
plt.tight_layout()
```

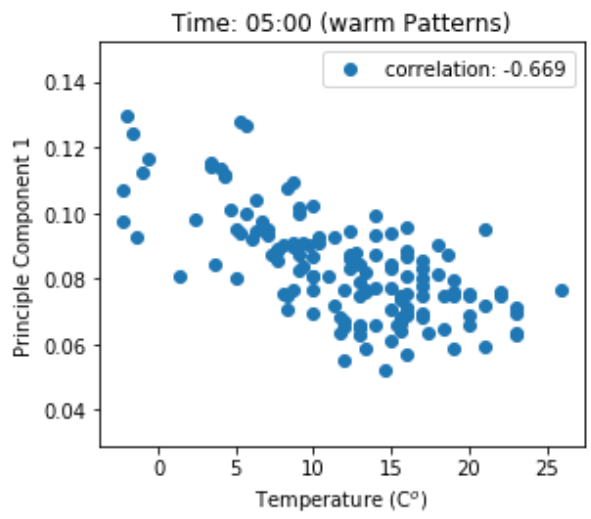

Time: 00:00 (warm Patterns)
correlation: -0.638
Principle Component 1
Temperature (C°)

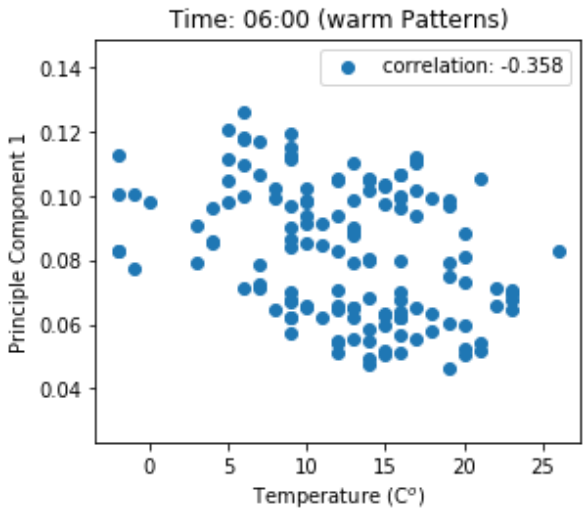

Time: 01:00 (warm Patterns)
correlation: -0.624
Principle Component 1
Temperature (C°)

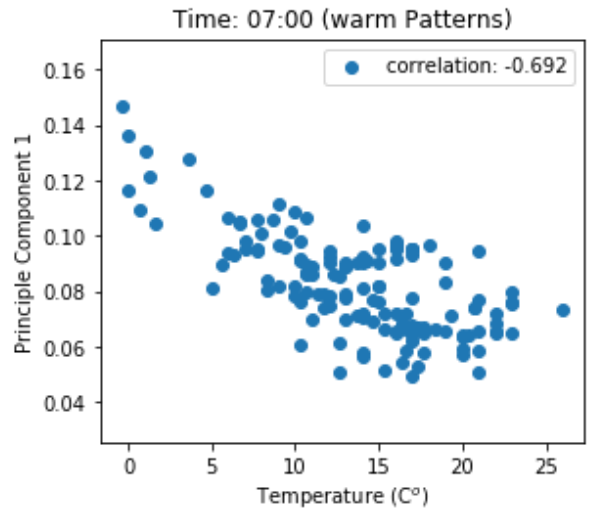

Time: 02:00 (warm Patterns)
correlation: -0.433
Principle Component 1
Temperature (C°)

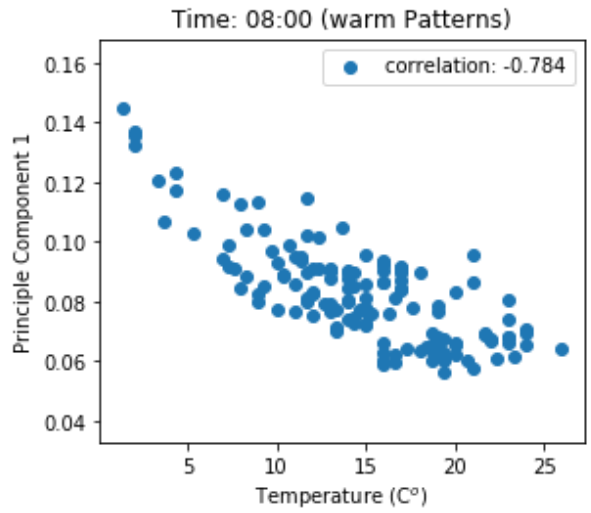

Time: 03:00 (warm Patterns)
correlation: -0.665
Principle Component 1
Temperature (C°)

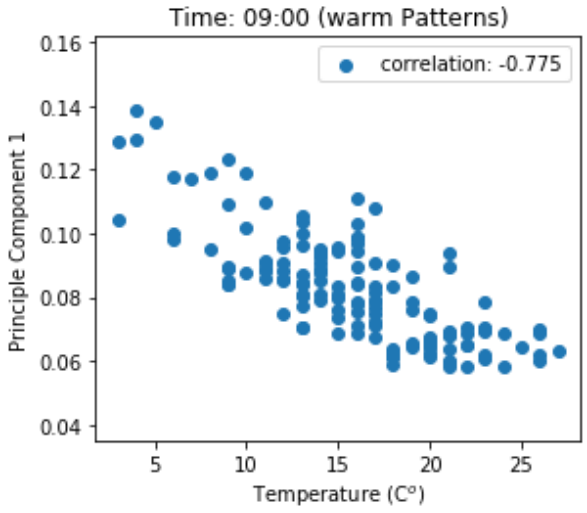

Time: 04:00 (warm Patterns)
correlation: -0.666
Principle Component 1
Temperature (C°)

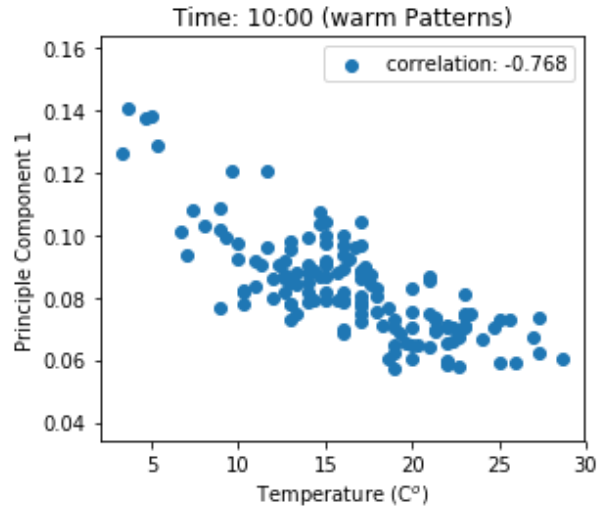

Time: 05:00 (warm Patterns)
correlation: -0.669
Principle Component 1
Temperature (C°)

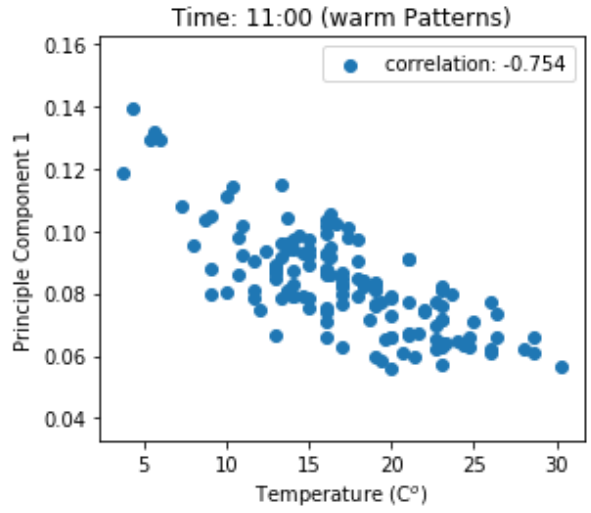

Time: 06:00 (warm Patterns)
correlation: -0.358
Principle Component 1
Temperature (C°)

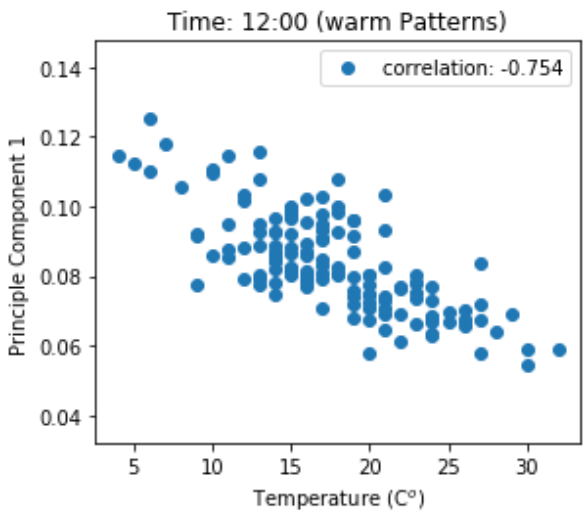

Time: 07:00 (warm Patterns)
correlation: -0.692
Principle Component 1
Temperature (C°)

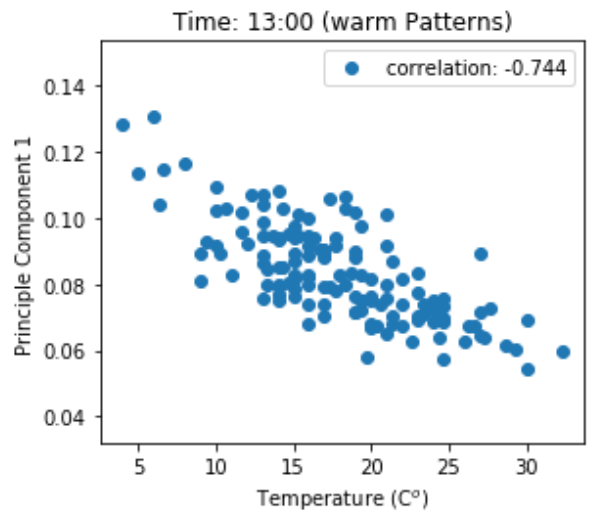

Time: 08:00 (warm Patterns)
correlation: -0.784
Principle Component 1
Temperature (C°)

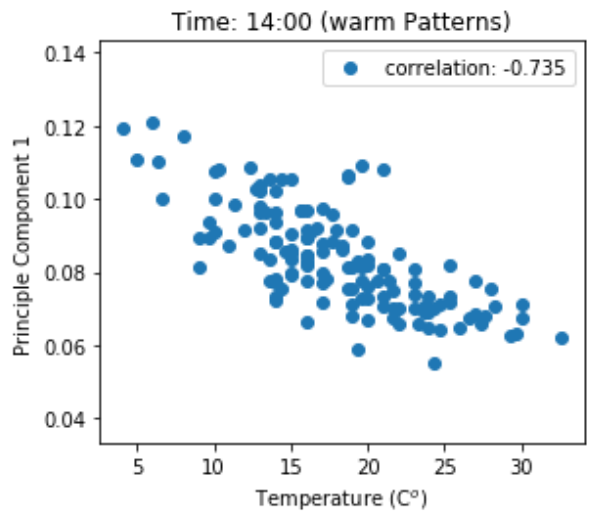

Time: 09:00 (warm Patterns)
correlation: -0.775
Principle Component 1
Temperature (C°)

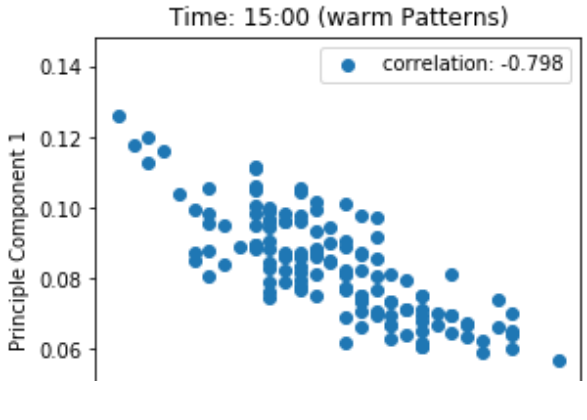

Time: 10:00 (warm Patterns)
correlation: -0.768
Principle Component 1
Temperature (C°)

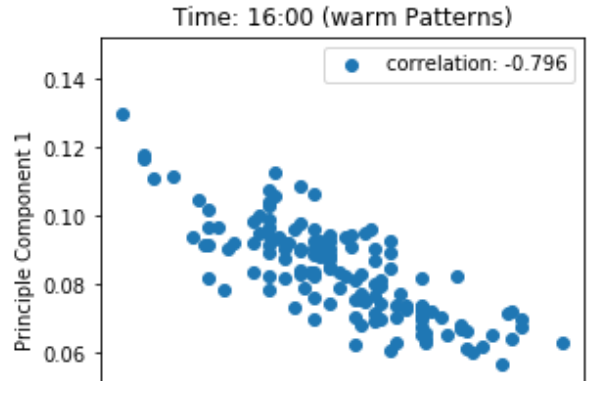

Time: 11:00 (warm Patterns)
correlation: -0.754
Principle Component 1
Temperature (C°)

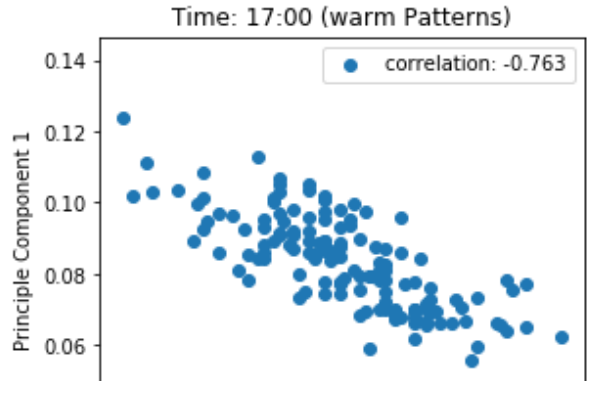

Time: 12:00 (warm Patterns)
correlation: -0.754
Principle Component 1
Temperature (C°)

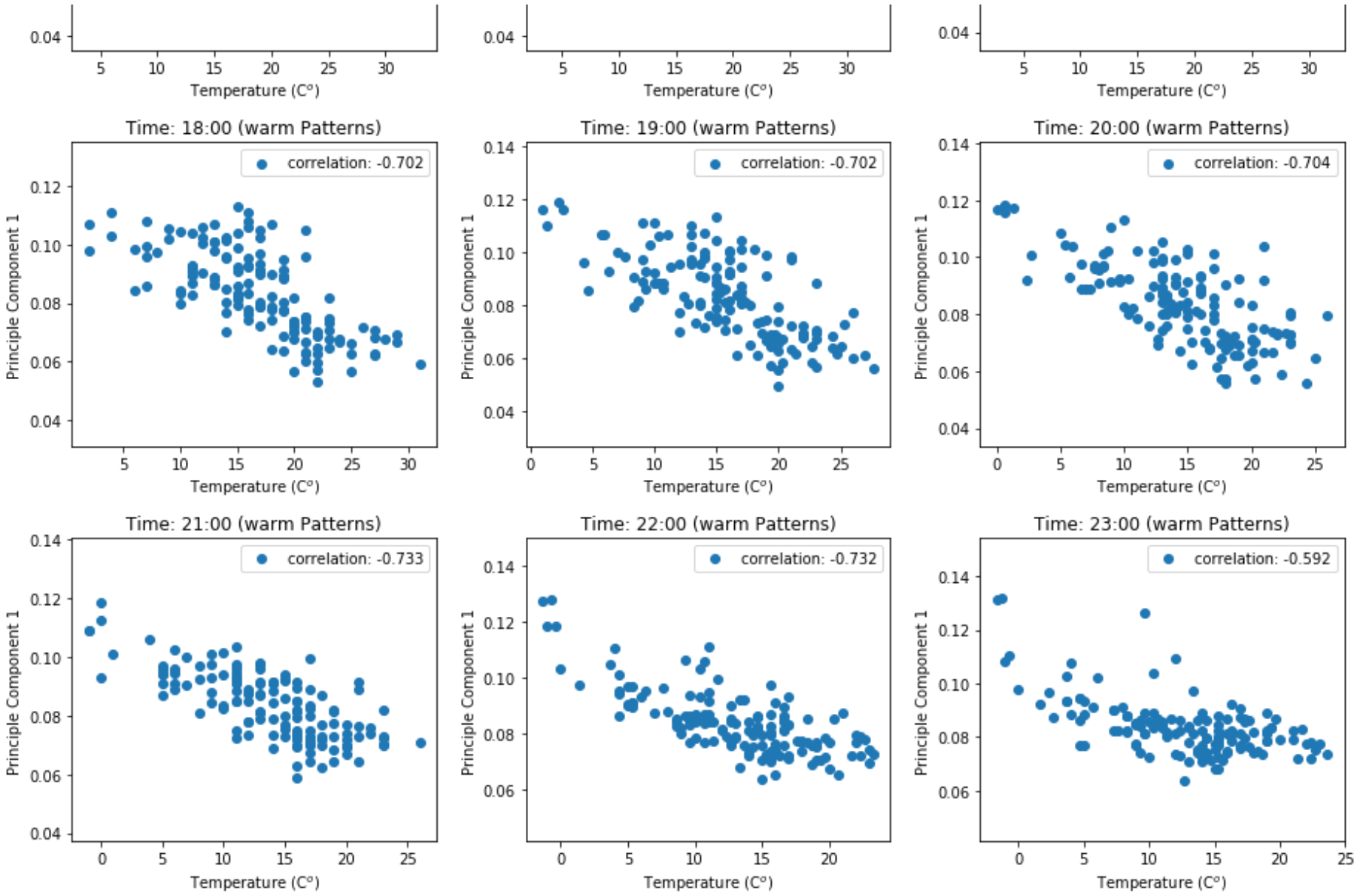

Time: 13:00 (warm Patterns)
correlation: -0.744
Principle Component 1
Temperature (C°)


Time: 14:00 (warm Patterns)
correlation: -0.735
Principle Component 1
Temperature (C°)


Time: 15:00 (warm Patterns)
correlation: -0.798
Principle Component 1

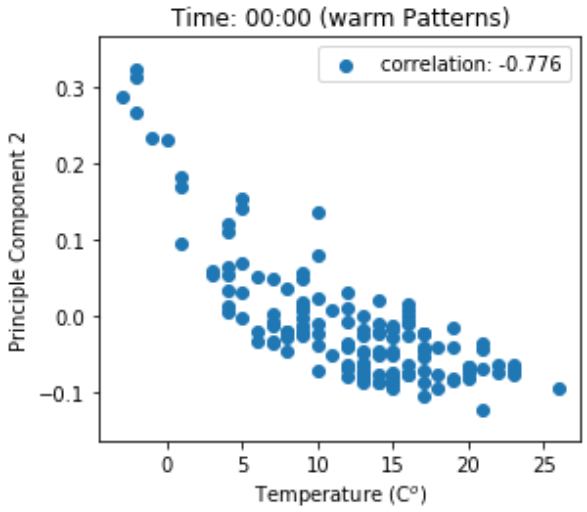

Time: 16:00 (warm Patterns)
correlation: -0.796
Principle Component 1

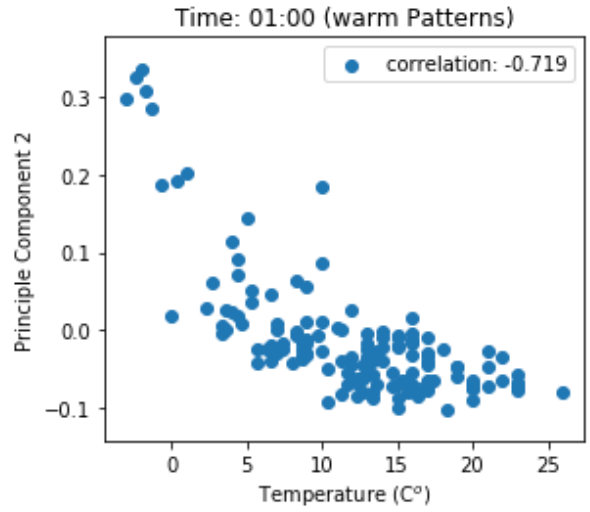

Time: 17:00 (warm Patterns)
correlation: -0.763
Principle Component 1

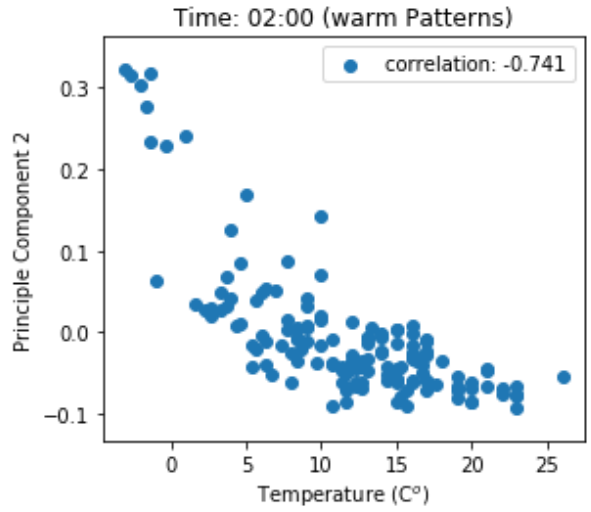


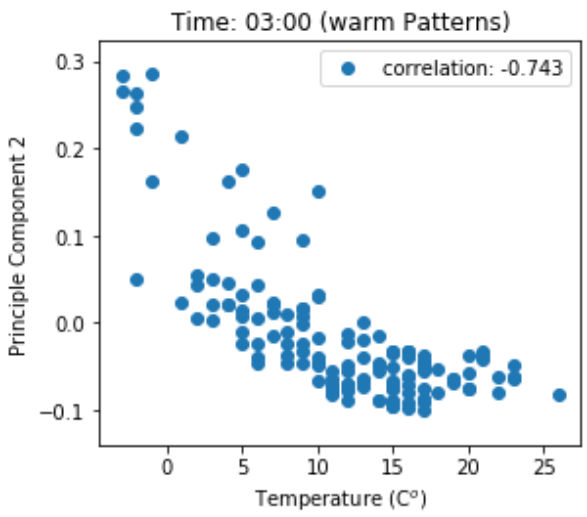
2) PC2's coefficients vs Temperature:

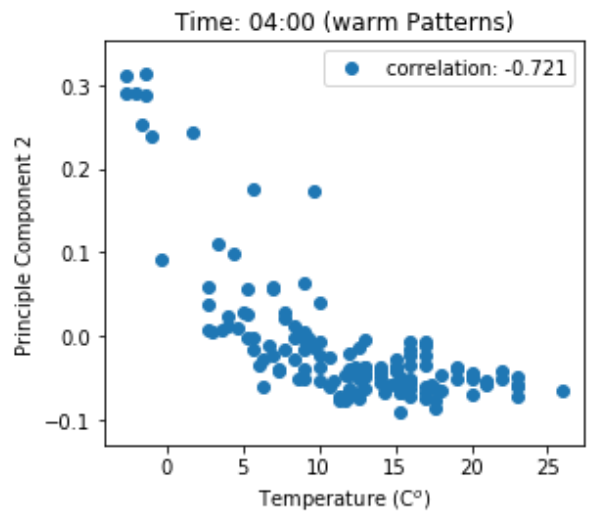
In [17]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(df_wea1h_nf_warm.TempC[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')], pca_ns
td.components_[1], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[1], df_wea1h_nf_warm.
TempC[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(df_wea1h_nf_warm.TempC[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')], pca_nstd.com
ponents_[1], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[1], df_wea1h_nf_warm.TempC[
df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

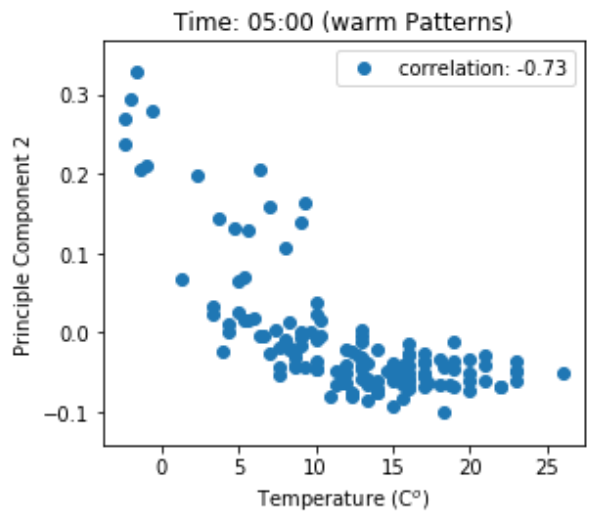

Time: 00:00 (warm Patterns)
correlation: -0.776
Principle Component 2
Temperature (C°)

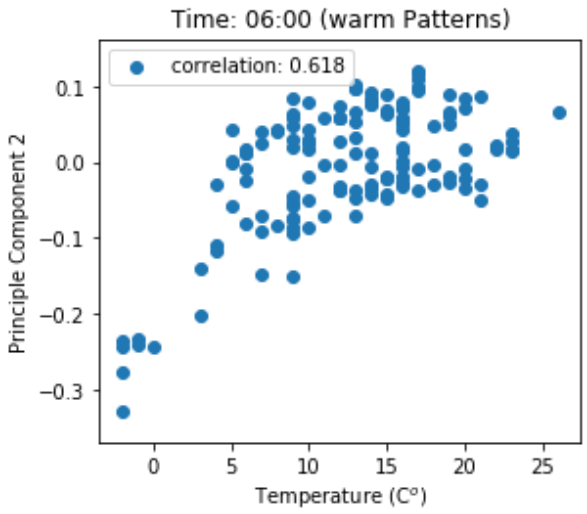

Time: 01:00 (warm Patterns)
correlation: -0.719
Principle Component 2
Temperature (C°)

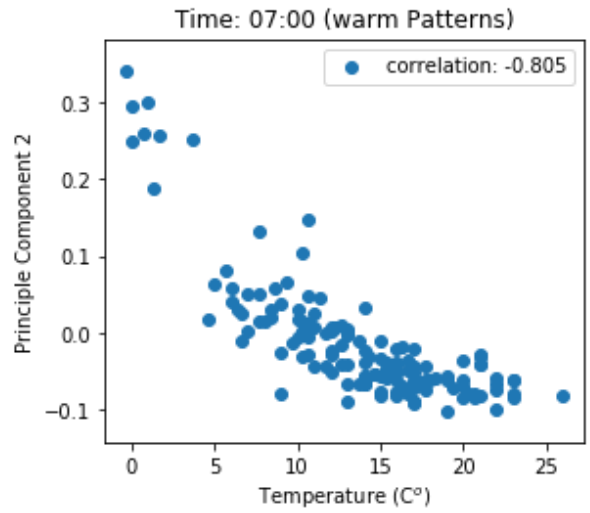

Time: 02:00 (warm Patterns)
correlation: -0.741
Principle Component 2
Temperature (C°)

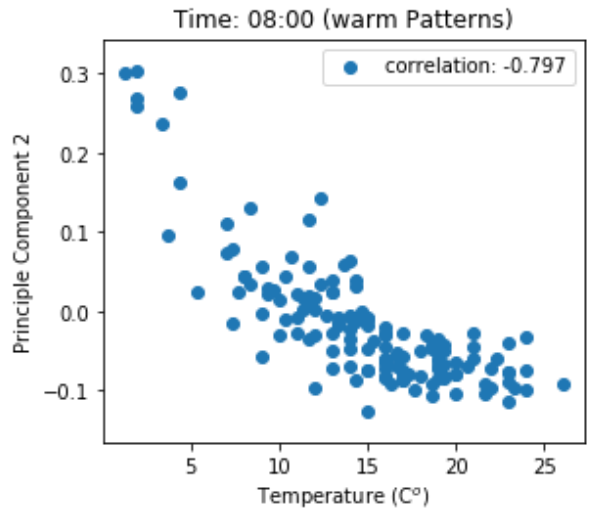

Time: 03:00 (warm Patterns)
correlation: -0.743
Principle Component 2
Temperature (C°)

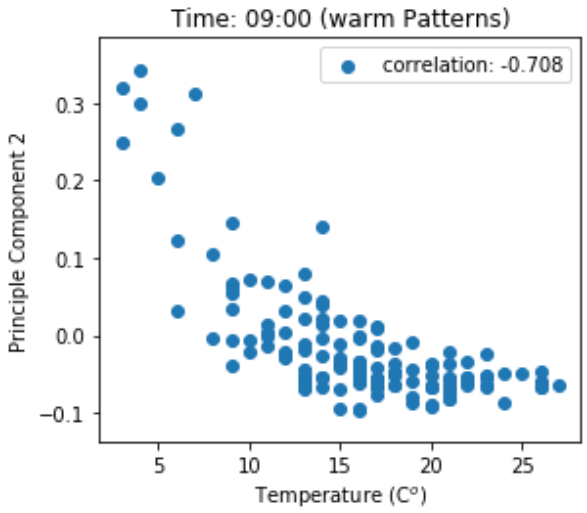

Time: 04:00 (warm Patterns)
correlation: -0.721
Principle Component 2
Temperature (C°)

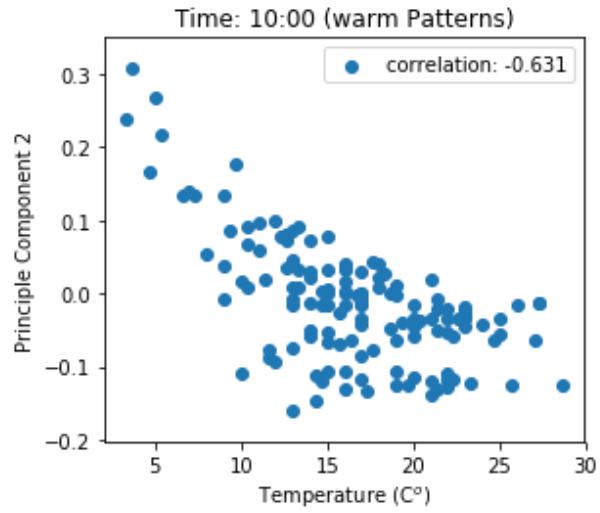

Time: 05:00 (warm Patterns)
correlation: -0.73
Principle Component 2
Temperature (C°)

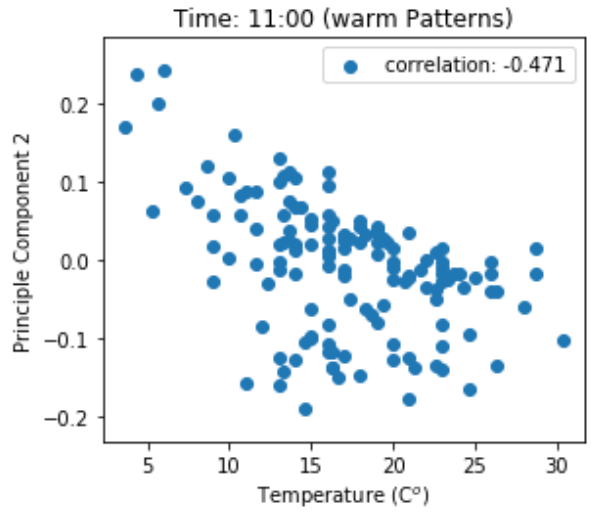

Time: 06:00 (warm Patterns)
correlation: 0.618
Principle Component 2
Temperature (C°)

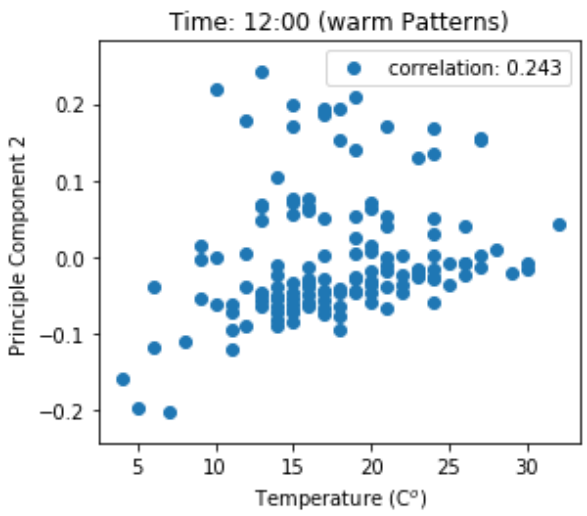

Time: 07:00 (warm Patterns)
correlation: -0.805
Principle Component 2
Temperature (C°)

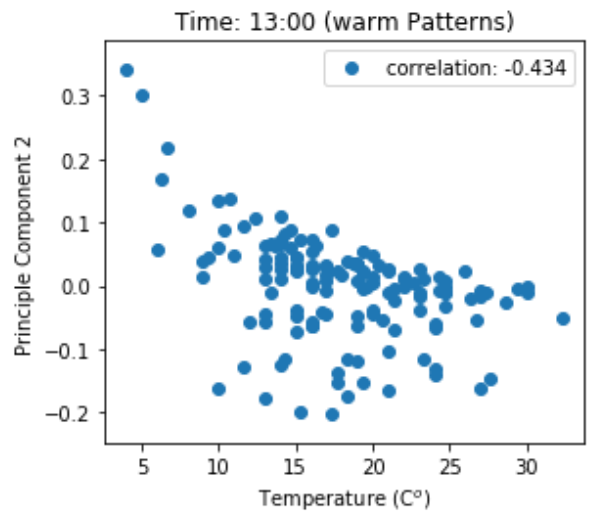

Time: 08:00 (warm Patterns)
correlation: -0.797
Principle Component 2
Temperature (C°)

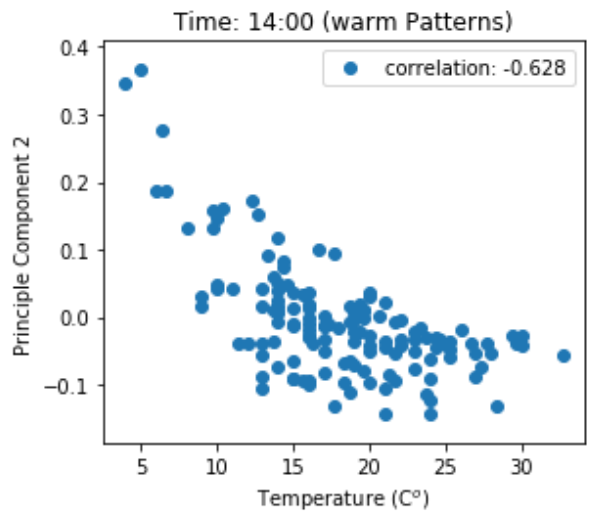

Time: 09:00 (warm Patterns)
correlation: -0.708
Principle Component 2
Temperature (C°)


Time: 10:00 (warm Patterns)
correlation: -0.631
Principle Component 2
Temperature (C°)


Time: 11:00 (warm Patterns)
correlation: -0.471
Principle Component 2
Temperature (C°)


Time: 12:00 (warm Patterns)
correlation: 0.243
Principle Component 2
Temperature (C°)

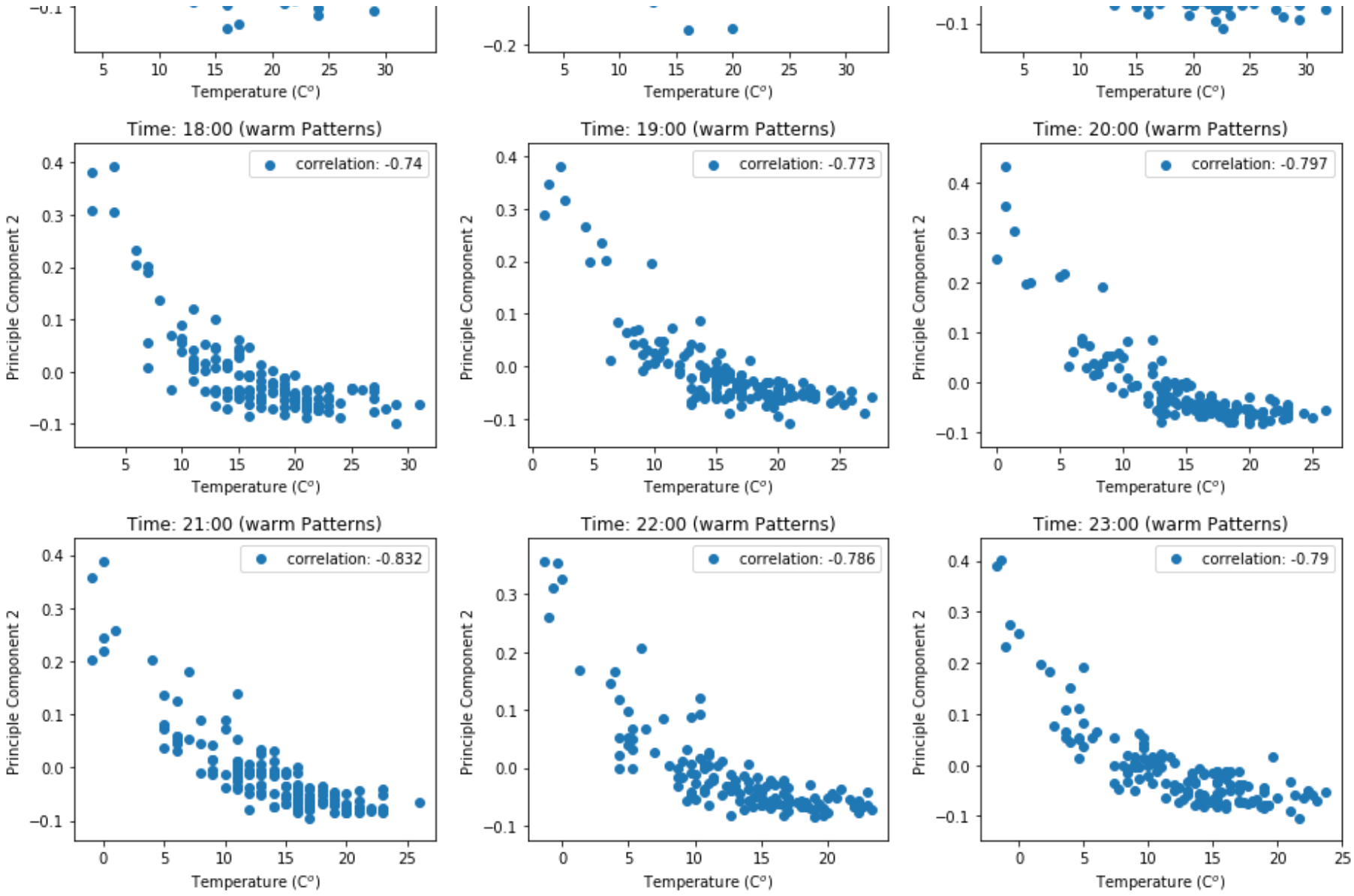

Time: 13:00 (warm Patterns)
correlation: -0.434
Principle Component 2
Temperature (C°)


Time: 14:00 (warm Patterns)
correlation: -0.628
Principle Component 2
Temperature (C°)

Time: 15:00 (warm Patterns)
correlation: -0.664
Principle Component 2

Time: 16:00 (warm Patterns)
correlation: -0.633
Principle Component 2

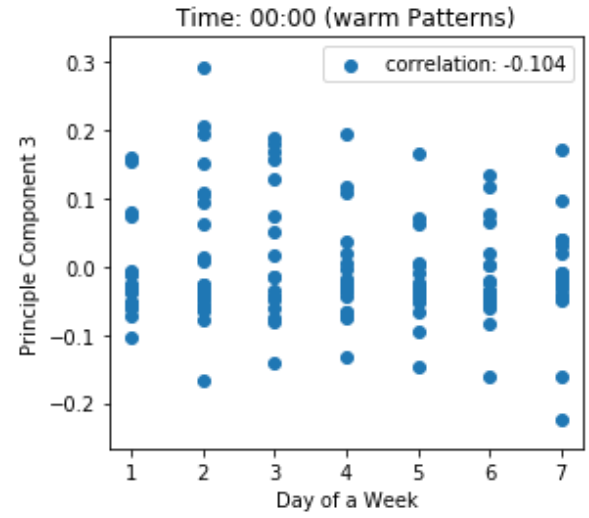
Time: 17:00 (warm Patterns)
correlation: -0.743
Principle Component 2

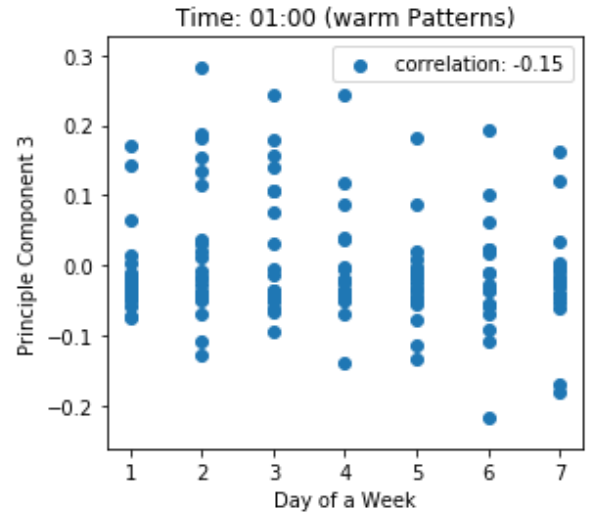


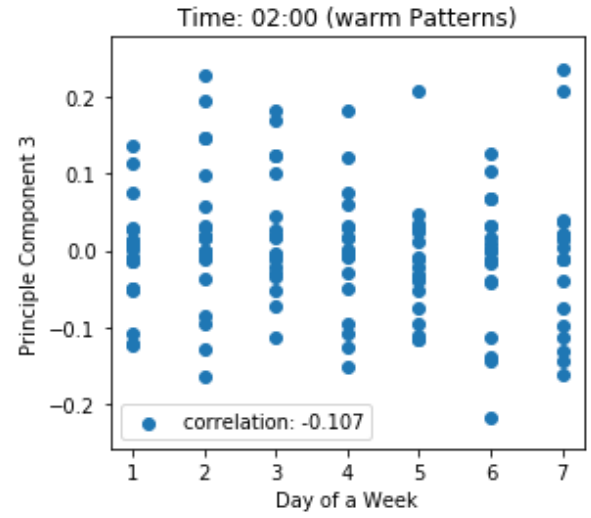
The above figures show that PC2 still has certain amount of correlation with temperature but not as significant as PC1 and temperature have.

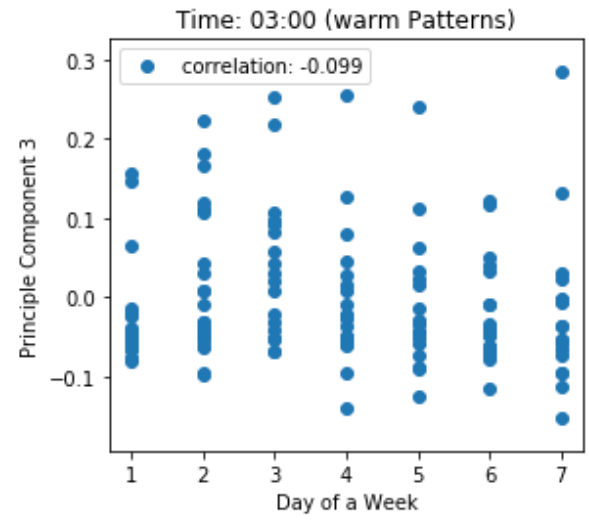
3) PC3's coefficients vs Day of a Week:

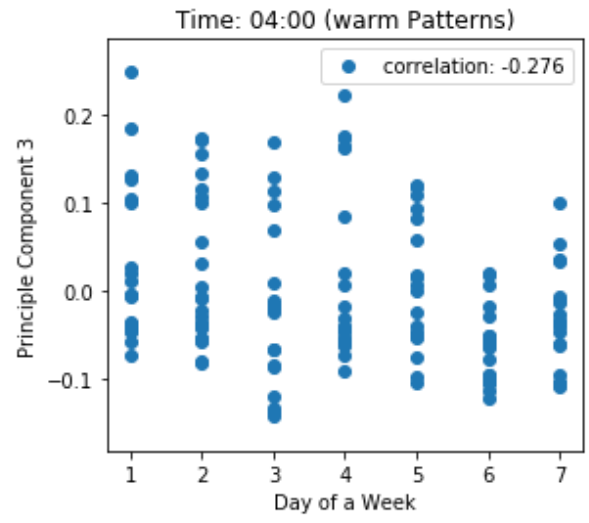
In [18]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(dayofWeek + 1, pca_nstd.components_[2], label = 'correlation: ' + str(round(np.corrcoef(pc
a_nstd.components_[2], dayofWeek.GMT)[0][1], 3)))
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Day of a Week')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        ax.scatter(dayofWeek + 1, pca_nstd.components_[2], label = 'correlation: ' + str(round(np.corrcoef(pc
a_nstd.components_[2], dayofWeek.GMT)[0][1], 3)))
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel('Day of a Week')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

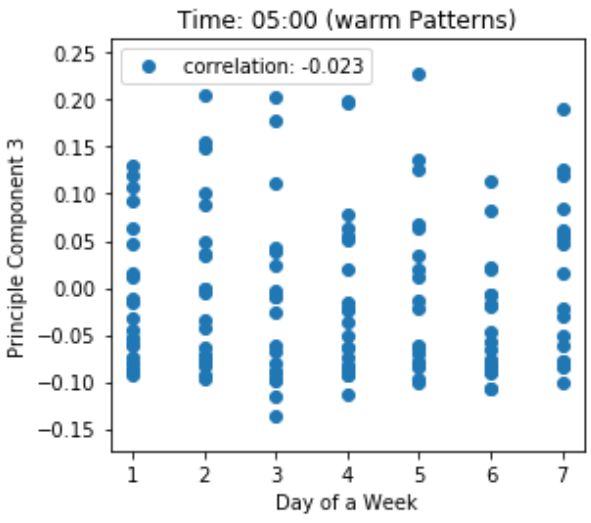

Time: 00:00 (warm Patterns)
correlation: -0.104
Principle Component 3
Day of a Week

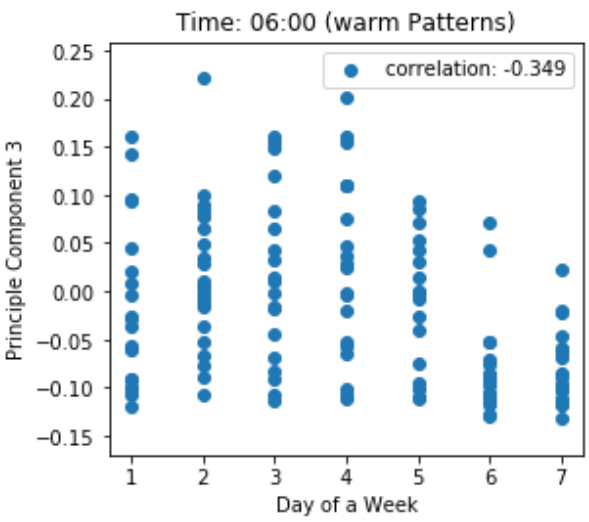

Time: 01:00 (warm Patterns)
correlation: -0.15
Principle Component 3
Day of a Week

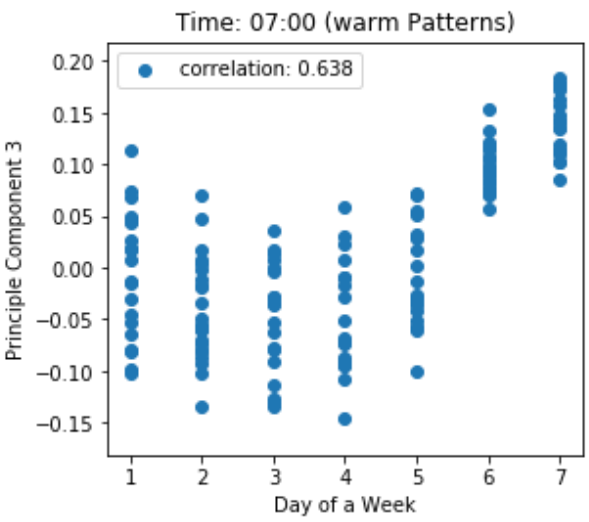

Time: 02:00 (warm Patterns)
correlation: -0.107
Principle Component 3
Day of a Week

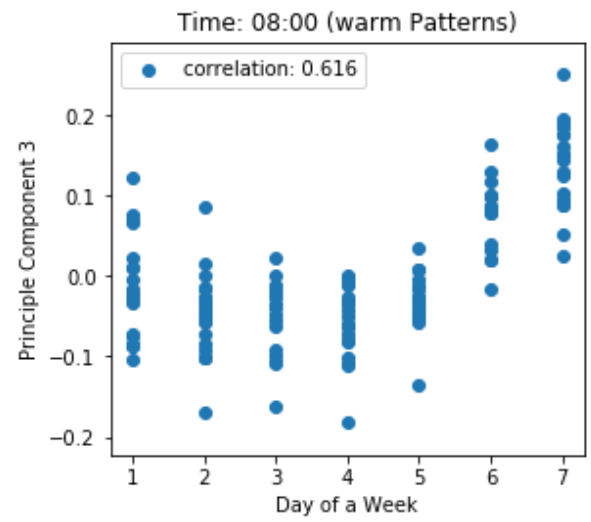

Time: 03:00 (warm Patterns)
correlation: -0.099
Principle Component 3
Day of a Week

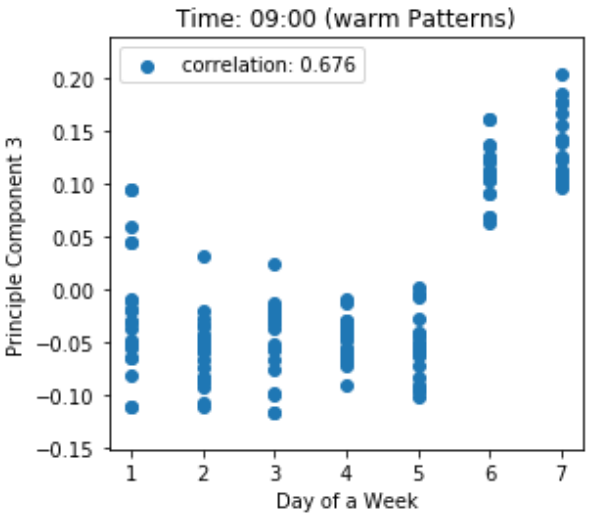

Time: 04:00 (warm Patterns)
correlation: -0.276
Principle Component 3
Day of a Week

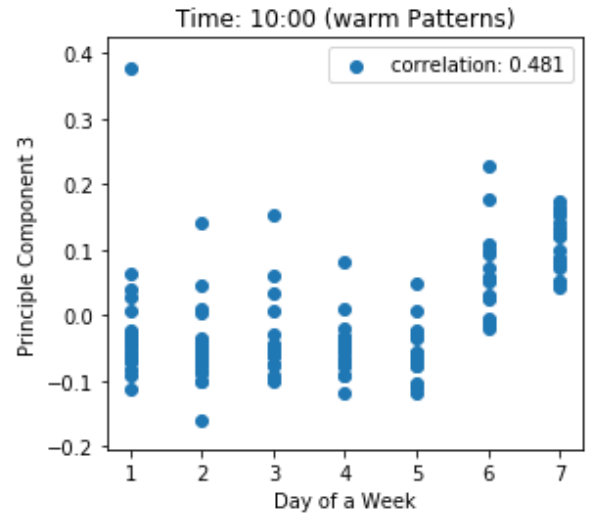

Time: 05:00 (warm Patterns)
correlation: -0.023
Principle Component 3
Day of a Week

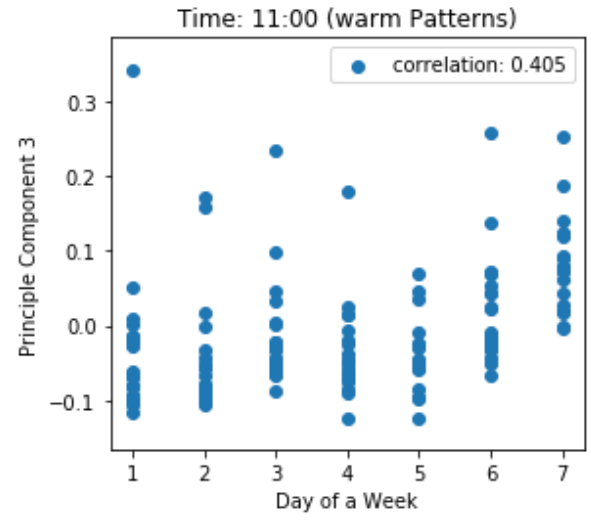

Time: 06:00 (warm Patterns)
correlation: -0.349
Principle Component 3
Day of a Week

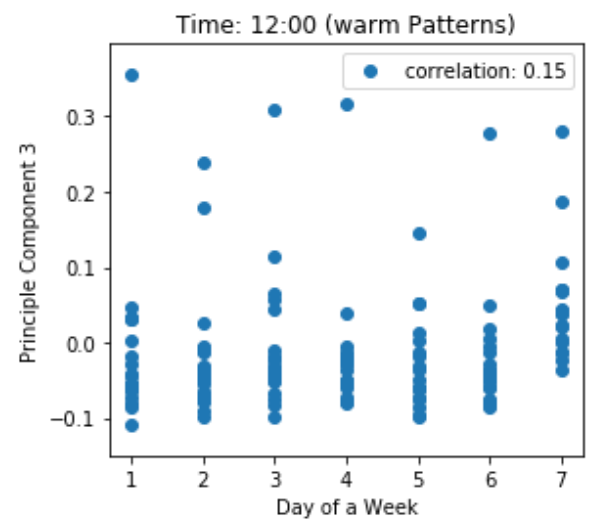

Time: 07:00 (warm Patterns)
correlation: 0.638
Principle Component 3
Day of a Week

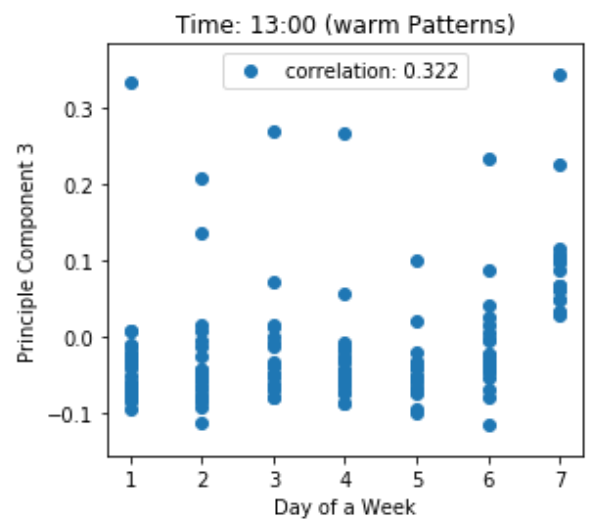

Time: 08:00 (warm Patterns)
correlation: 0.616
Principle Component 3
Day of a Week

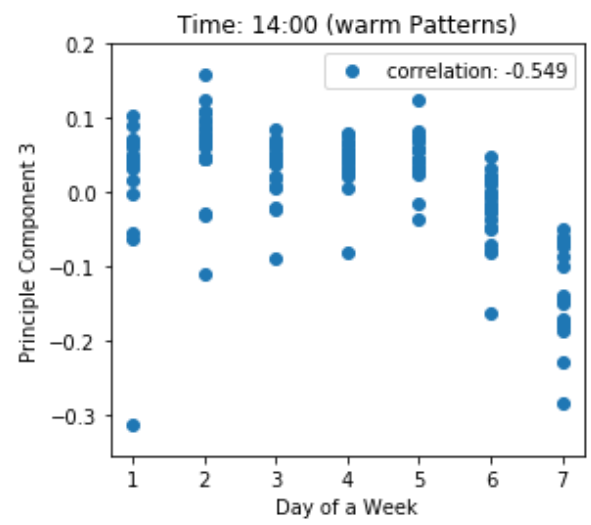

Time: 09:00 (warm Patterns)
correlation: 0.676
Principle Component 3
Day of a Week

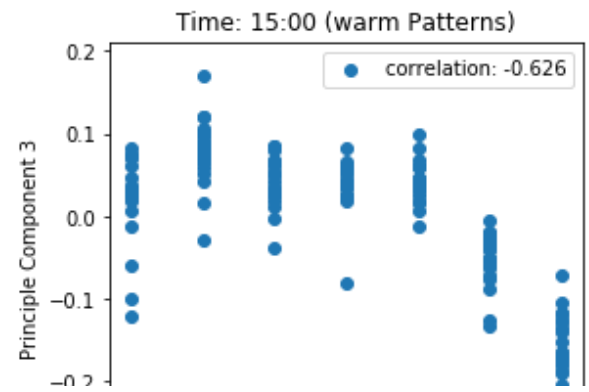

Time: 10:00 (warm Patterns)
correlation: 0.481
Principle Component 3
Day of a Week

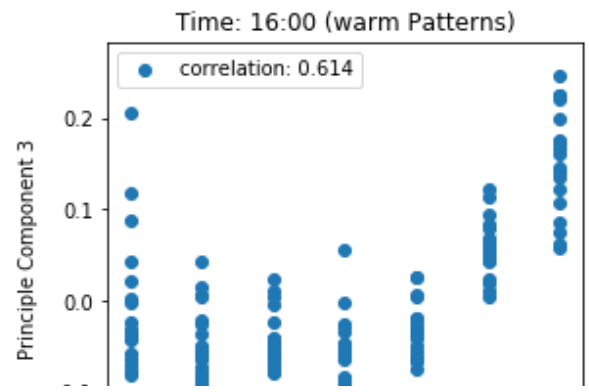

Time: 11:00 (warm Patterns)
correlation: 0.405
Principle Component 3
Day of a Week

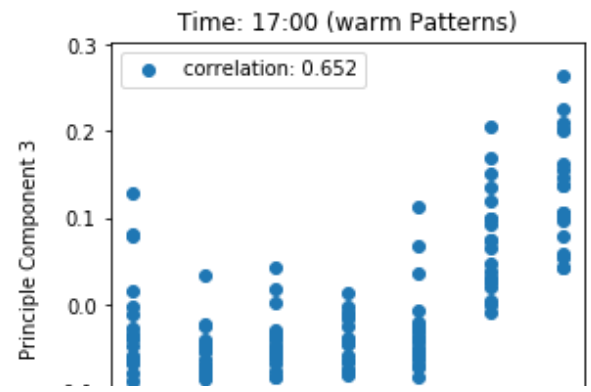

Time: 12:00 (warm Patterns)
correlation: 0.15
Principle Component 3
Day of a Week

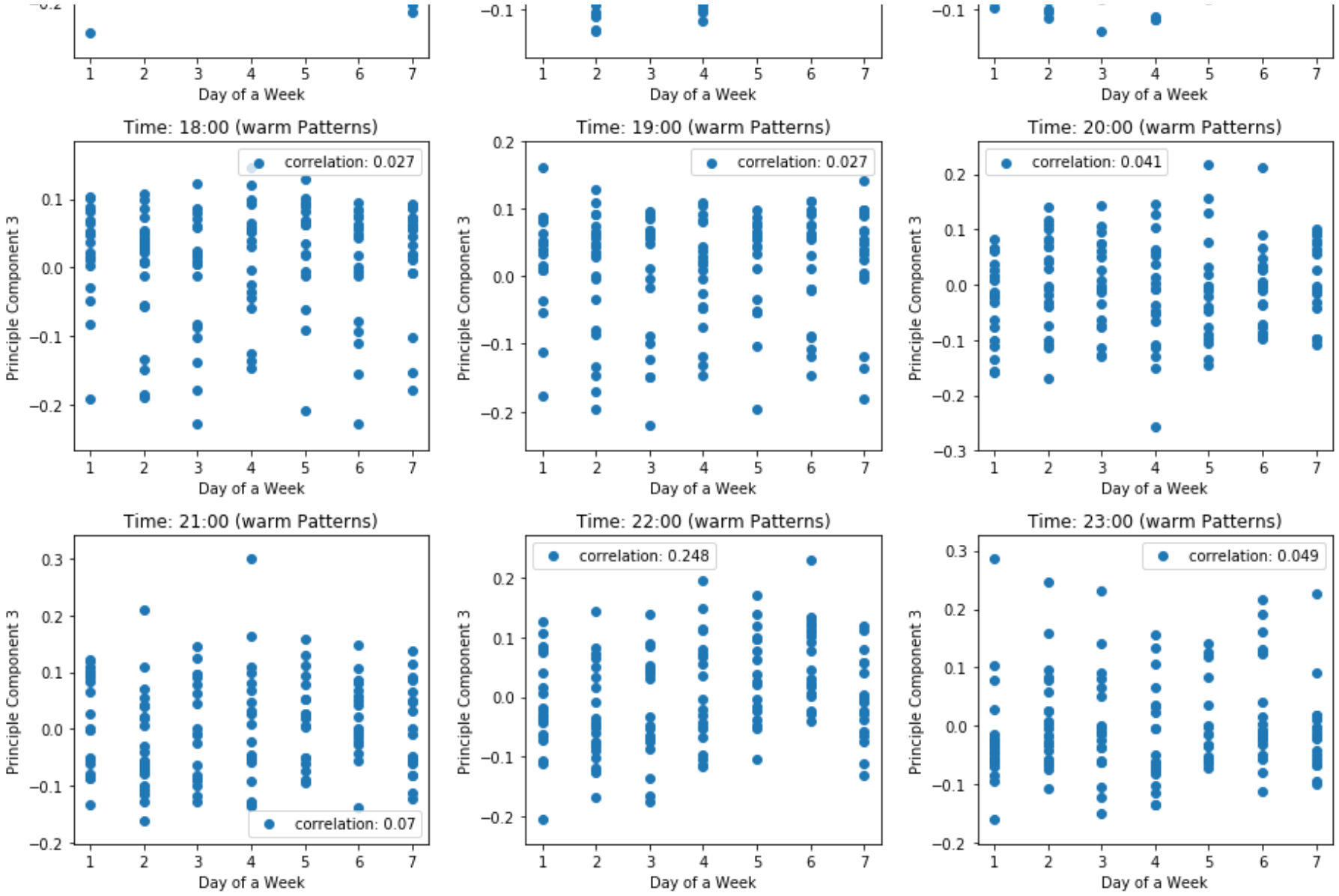

Time: 13:00 (warm Patterns)
correlation: 0.322
Principle Component 3
Day of a Week


Time: 14:00 (warm Patterns)
correlation: -0.549
Principle Component 3
Day of a Week


Time: 15:00 (warm Patterns)
correlation: -0.626
Principle Component 3


Time: 16:00 (warm Patterns)
correlation: 0.614
Principle Component 3


Time: 17:00 (warm Patterns)
correlation: 0.652
Principle Component 3

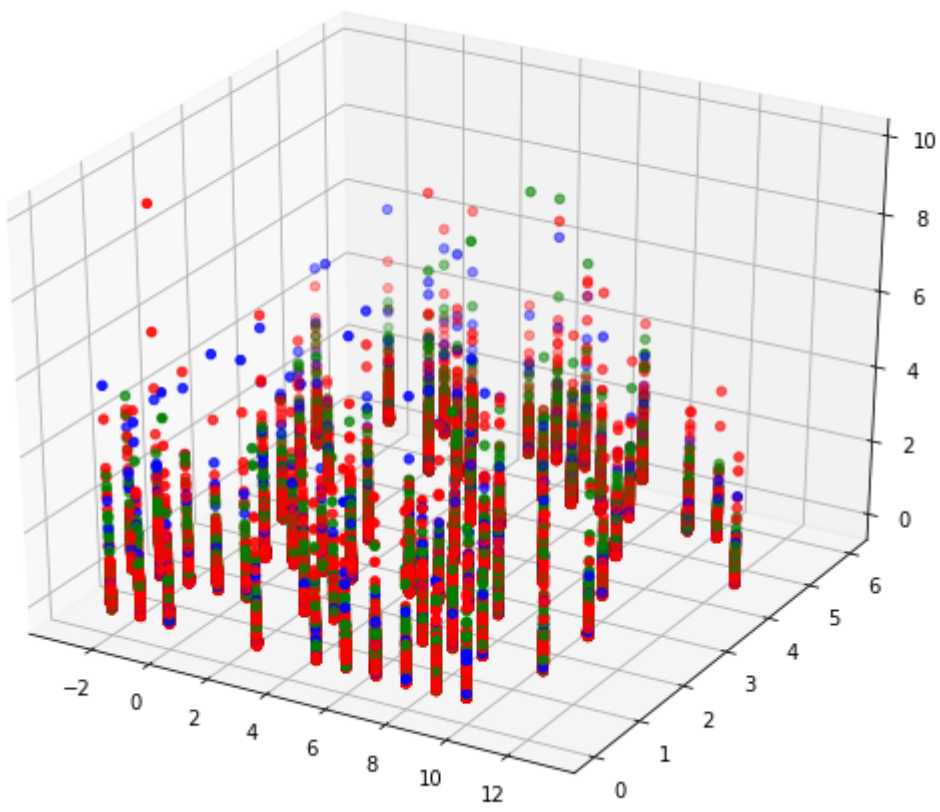


2. K-mean clustering

Milestone 4: K-mean clustering

The first three PCs are selected as inputs for K-mean clustering, due to their significance and interpretability

In [431]:
```
x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]
t = x.dropna(axis = 'columns')
t2 = t[t.columns[np.insert(kmeans.labels_ == 0, 0, True)]].copy() # select all data from the n-th cluster
t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['TempC'].values) # add
temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['GMT']).dt.d
ayofweek # add day of a week column to dataframe
t2
fig_all = plt.figure(figsize = (10,8))
ax = fig_all.add_subplot(1, 1, 1, projection = '3d')
for i in range(1, 1 + 528):
    ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[i]], color = 'red')

t2 = t[t.columns[np.insert(kmeans.labels_ == 1, 0, True)]].copy() # select all data from the n-th cluster
t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['TempC'].values) # add
temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['GMT']).dt.d
ayofweek # add day of a week column to dataframe
for i in range(1, 1 + 107):
    ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[i]], color = 'green')

t2 = t[t.columns[np.insert(kmeans.labels_ == 2, 0, True)]].copy() # select all data from the n-th cluster
t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['TempC'].values) # add
temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['GMT']).dt.d
ayofweek # add day of a week column to dataframe
for i in range(1, 1 + 25):
    ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[i]], color = 'blue')
```

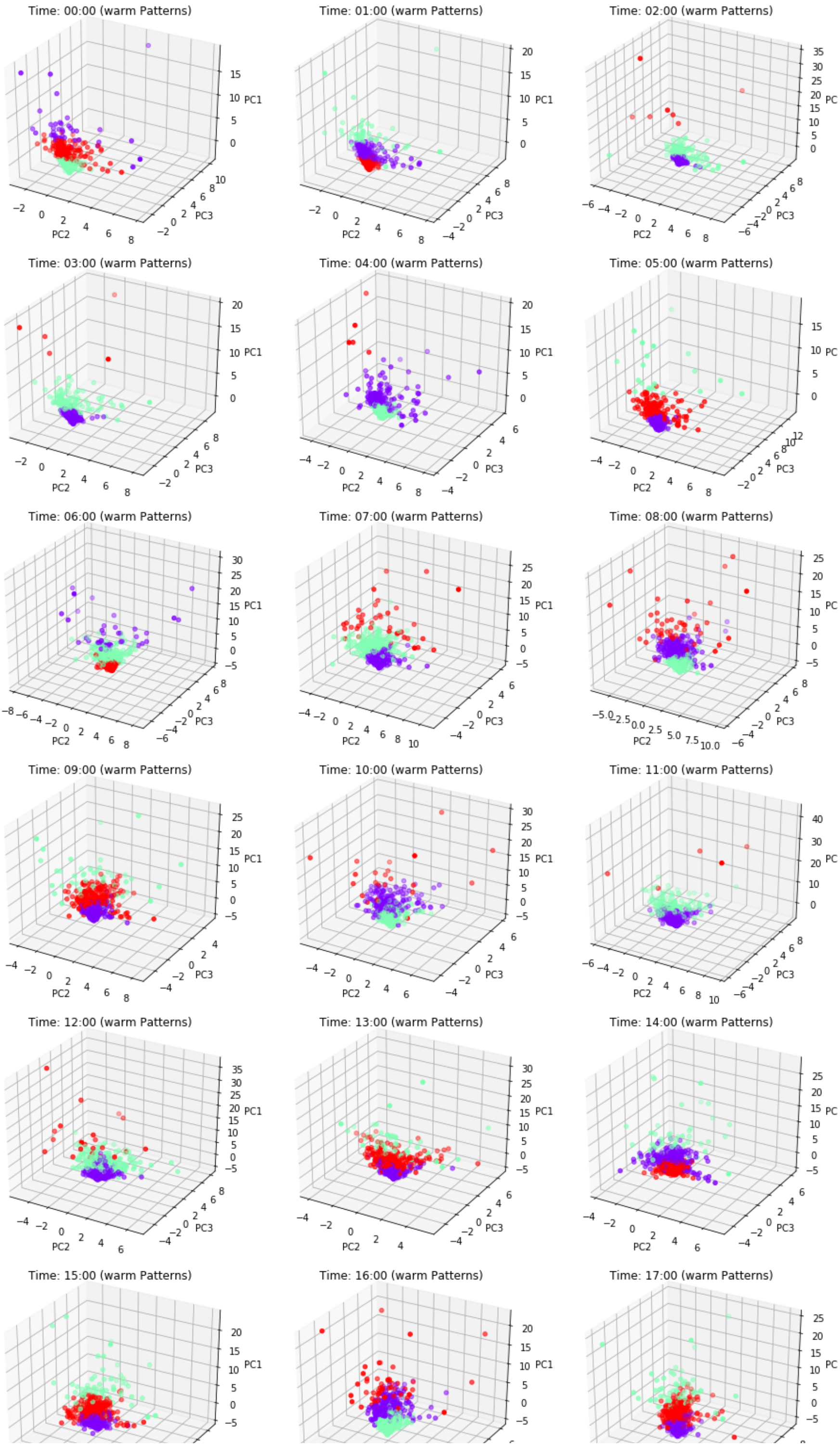


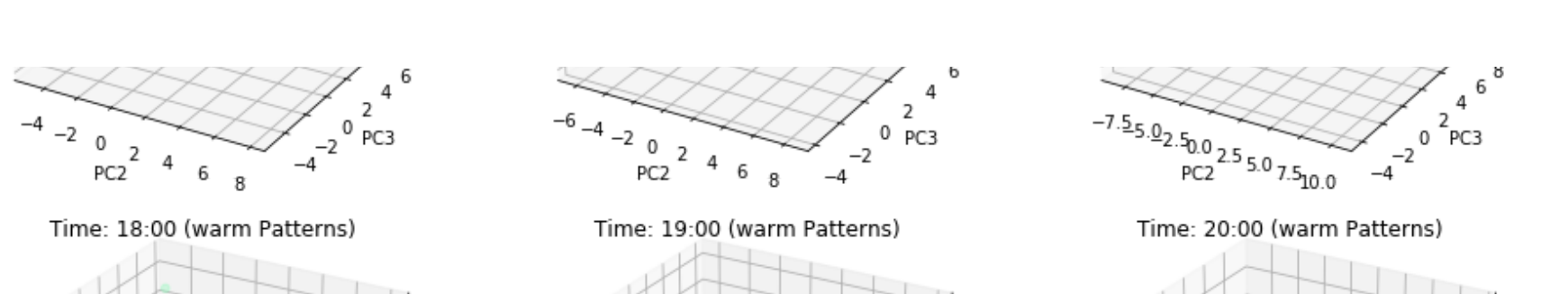
In [424]:
```
# the first groups consumption dataframe
(kmeans.labels_ == 2).sum()
```

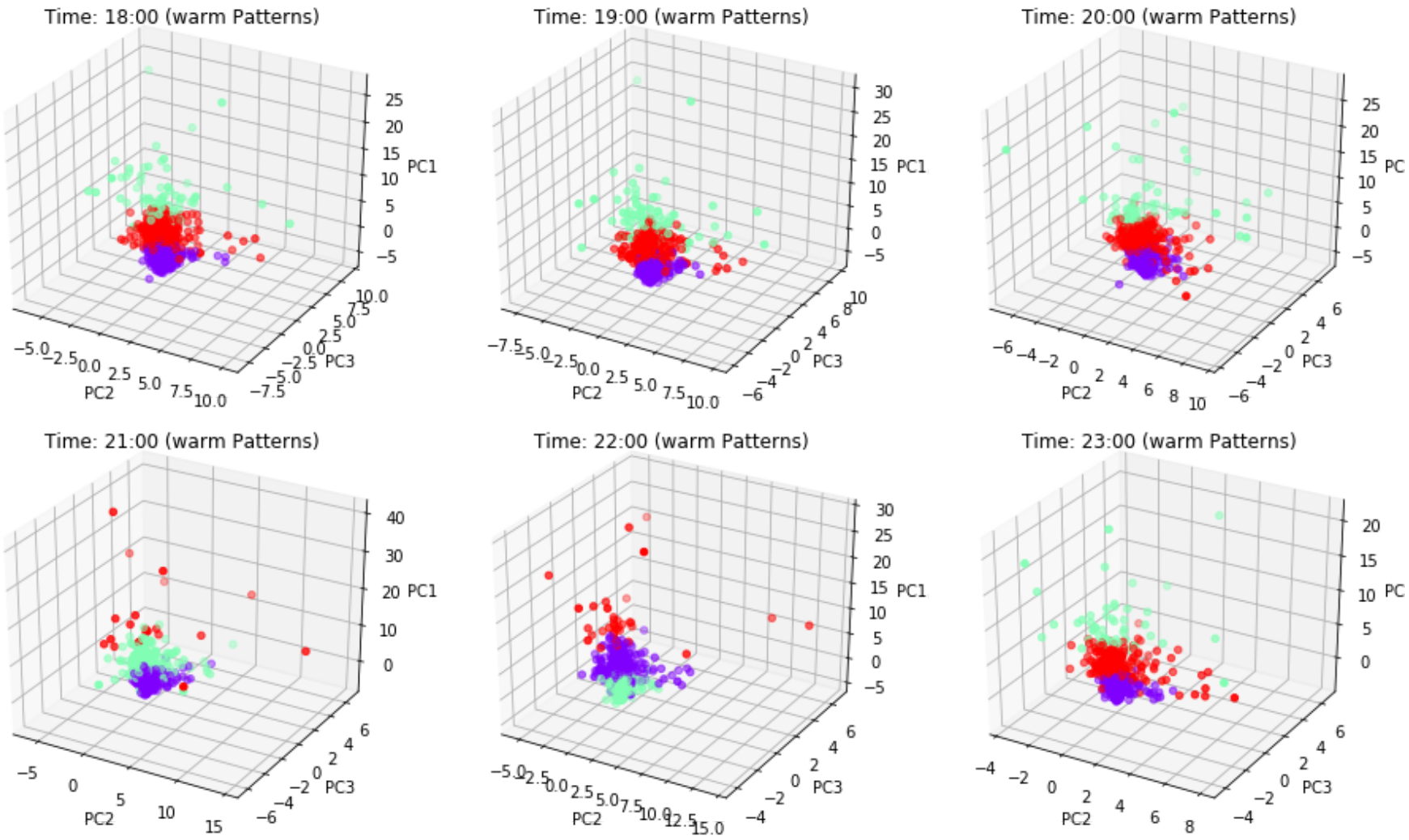
Out[424]: 25

In [ ]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1, projection = '3d')
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = 3).fit(principleDf_nstd.iloc[:,:3])
        ax.scatter(principleDf_nstd[1], principleDf_nstd[2], principleDf_nstd[0], c = (kmeans.labels_*10), cm
ap = 'rainbow')
        # what is the center of the cluster
        kmeans.cluster_centers_
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
#     else:
#         x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
#         dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
#         x.set_index('GMT', inplace = True)
#         x = x.transpose().dropna() # data for the specific hour, and filtering out null value
#         # PCA without standardization
#         pca_nstd = PCA()
#         principleComponents_nstd = pca_nstd.fit_transform(x)
#         principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
#         kmeans = KMeans(n_clusters = 3).fit(principleDf_nstd.iloc[:,:3])
#         ax.scatter(principleDf_nstd[1], principleDf_nstd[2], principleDf_nstd[0], c = (kmeans.labels_*10),
 cmap = 'rainbow')
#         ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
#         ax.set_xlabel('PC2')
#         ax.set_ylabel('PC3')
#         ax.set_zlabel('PC1')
plt.tight_layout()
```

In [20]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1, projection = '3d')
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = 3).fit(principleDf_nstd.iloc[:,:3])
        ax.scatter(principleDf_nstd[1], principleDf_nstd[2], principleDf_nstd[0], c = (kmeans.labels_*10), cm
ap = 'rainbow')
        # what is the center of the cluster
        kmeans.cluster_centers_
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = 3).fit(principleDf_nstd.iloc[:,:3])
        ax.scatter(principleDf_nstd[1], principleDf_nstd[2], principleDf_nstd[0], c = (kmeans.labels_*10), cm
ap = 'rainbow')
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
plt.tight_layout()
```


Time: 00:00 (warm Patterns)
Time: 01:00 (warm Patterns)
Time: 02:00 (warm Patterns)
Time: 03:00 (warm Patterns)
Time: 04:00 (warm Patterns)
Time: 05:00 (warm Patterns)
Time: 06:00 (warm Patterns)
Time: 07:00 (warm Patterns)
Time: 08:00 (warm Patterns)
Time: 09:00 (warm Patterns)
Time: 10:00 (warm Patterns)
Time: 11:00 (warm Patterns)
Time: 12:00 (warm Patterns)
Time: 13:00 (warm Patterns)
Time: 14:00 (warm Patterns)
Time: 15:00 (warm Patterns)
Time: 16:00 (warm Patterns)
Time: 17:00 (warm Patterns)
PC1
PC2
PC3





The ways of checking the clustering theta threshold condition

In [323]:
```
kmeans.cluster_centers_[1]
```

Out[323]:
```
array([13.69917824,  0.46278999, -0.14054859])
```

In [348]:
```
np.square(kmeans.cluster_centers_[1]).sum() * 0.2
```

Out[348]:
```
37.580282564512224
```

In [324]:
```
principleDf_nstd.iloc[kmeans.labels_ == 1][2].mean()
```

Out[324]:
```
-0.14054858878175358
```

In [366]:
```
(7+189+441)/976
```

Out[366]:
```
0.6526639344262295
```

In [379]:
```
test = principleDf_nstd.iloc[kmeans.labels_ == 0][[0,1,2]] - kmeans.cluster_centers_[0]
sum(np.square(test).sum(axis = 1) > np.square(kmeans.cluster_centers_[0]).sum() * 0.2)
```

Out[379]:
```
0
```

In [385]:
```
test = principleDf_nstd.iloc[kmeans.labels_ == 1][[0,1,2]] - kmeans.cluster_centers_[1]
sum(np.square(test).sum(axis = 1) > np.square(kmeans.cluster_centers_[1]).sum() * 0.2)
```

Out[385]:
```
63
```

In [384]:
```
test = principleDf_nstd.iloc[kmeans.labels_ == 2][[0,1,2]] - kmeans.cluster_centers_[2]
sum(np.square(test).sum(axis = 1) > np.square(kmeans.cluster_centers_[2]).sum() * 0.2)
```

Out[384]:
```
118
```

In [383]:
```
test = principleDf_nstd.iloc[kmeans.labels_ == 3][[0,1,2]] - kmeans.cluster_centers_[3]
sum(np.square(test).sum(axis = 1) > np.square(kmeans.cluster_centers_[3]).sum() * 0.2)
```

Out[383]:
```
1
```

In [395]:
```
tot = 0
for i in range(50):
    test = principleDf_nstd.iloc[kmeans.labels_ == i][[0,1,2]] - kmeans.cluster_centers_[i]
    tot = tot + sum(np.square(test).sum(axis = 1) > np.square(kmeans.cluster_centers_[i]).sum() * 0.2)
tot
```

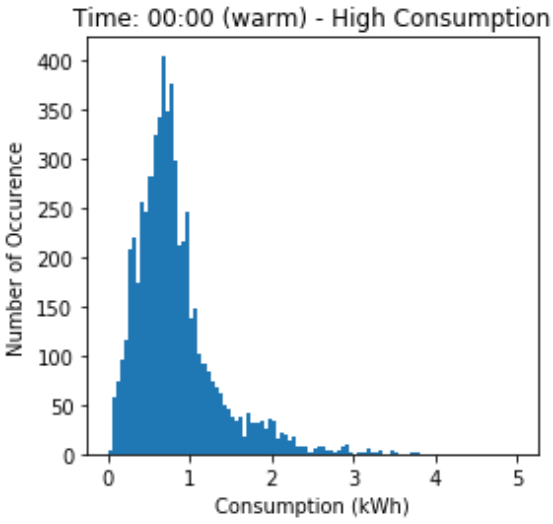
Out[395]:
```
184
```

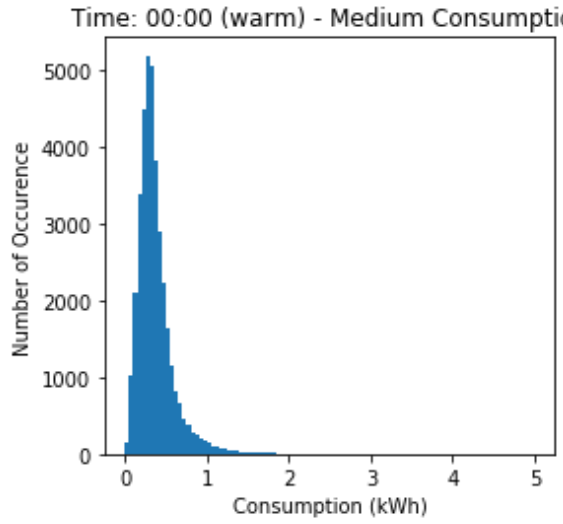
In [359]:
```
kmeans.cluster_centers_[2]
```

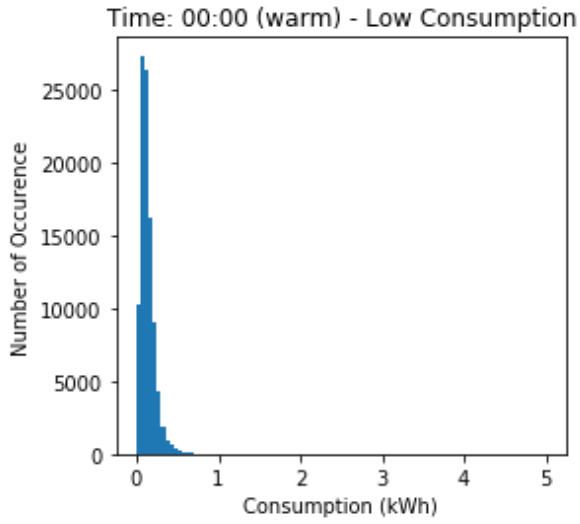
Out[359]:
```
array([-1.85445773,  0.06602678, -0.00851691])
```

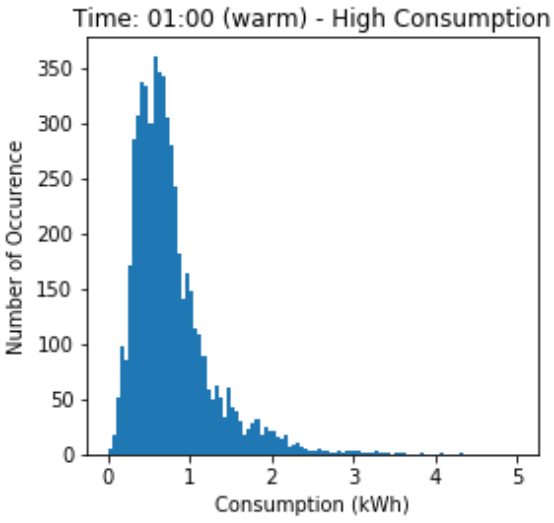
General consumptions distribution property of each cluster at each hour

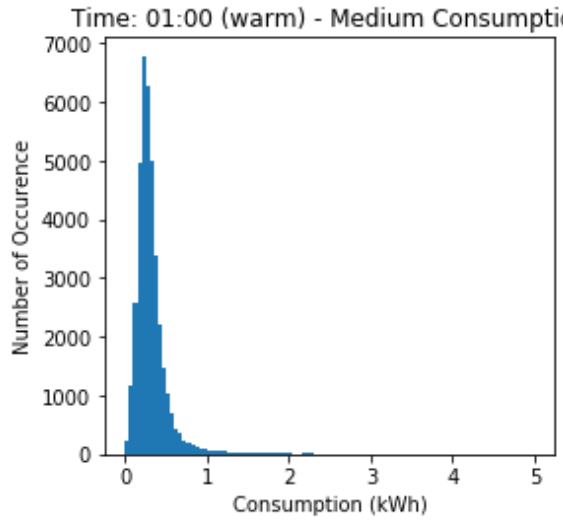
In [21]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1)
            t2 = t[t.columns[kmeans.labels_ == sorted_label[k][0]]].copy() # select all data from the n-th cl
uster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00'
)]['TempC'].values) # add temperature column to dataframe
            ax.hist(t2[t2.columns[1:-1]].values.flatten(), bins=100, range=(0,5))
            ax.set_title('Time: 0' + str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Number of Occurence')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1)
            t2 = t[t.columns[kmeans.labels_ == sorted_label[k][0]]].copy() # select all data from the n-th cl
uster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['Te
mpC'].values) # add temperature column to dataframe
            ax.hist(t2[t2.columns[1:-1]].values.flatten(), bins=100, range=(0,5))
            ax.set_title('Time: ' + str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Number of Occurence')
plt.tight_layout()
```

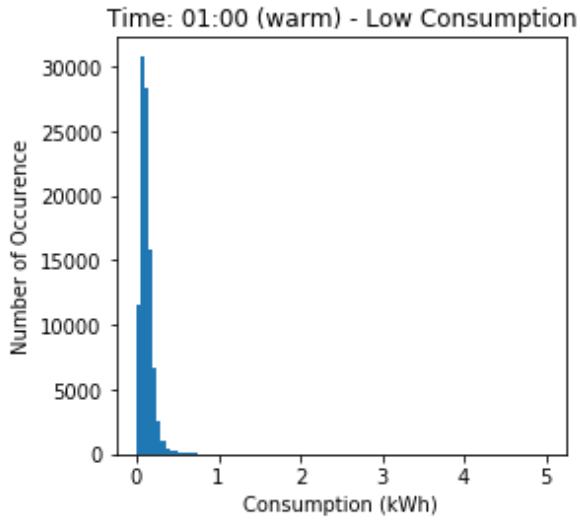

Time: 00:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

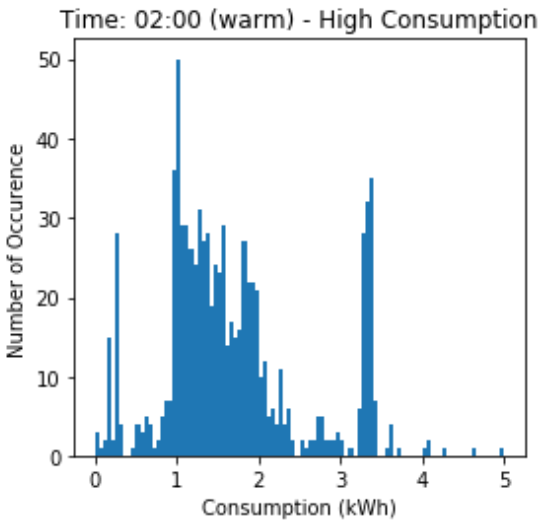

Time: 00:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

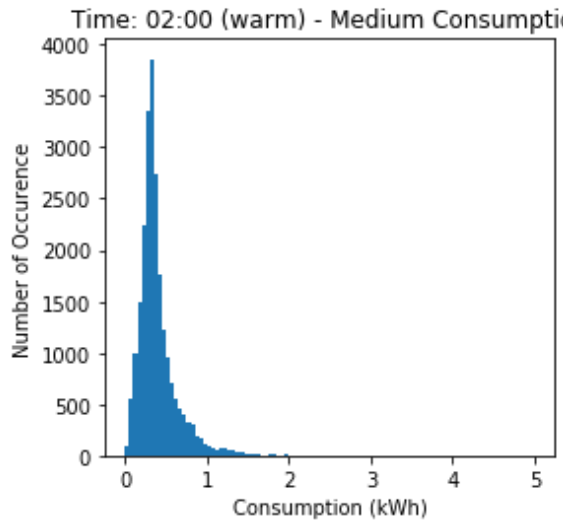

Time: 00:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

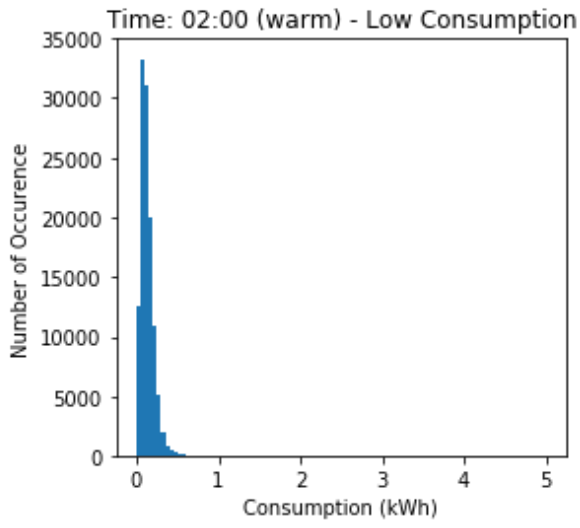

Time: 01:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

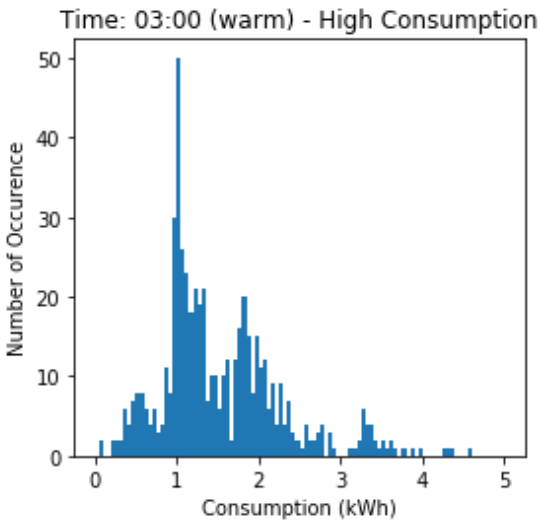

Time: 01:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

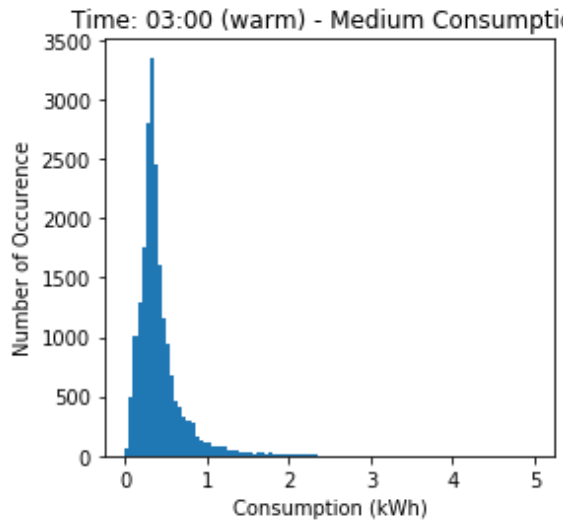

Time: 01:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

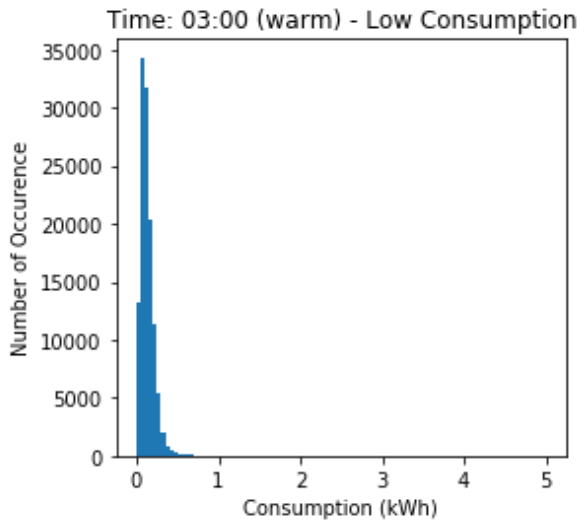

Time: 02:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

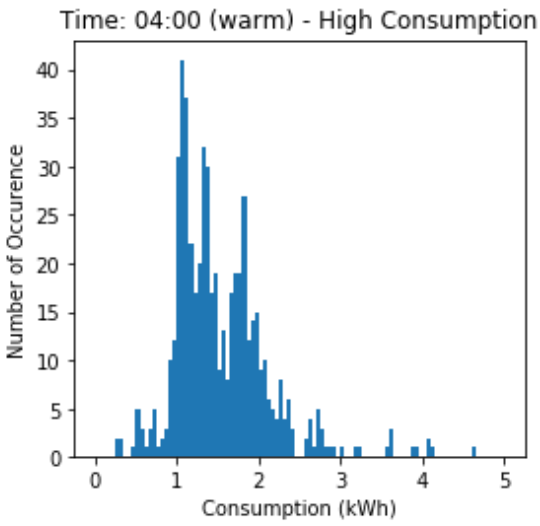

Time: 02:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

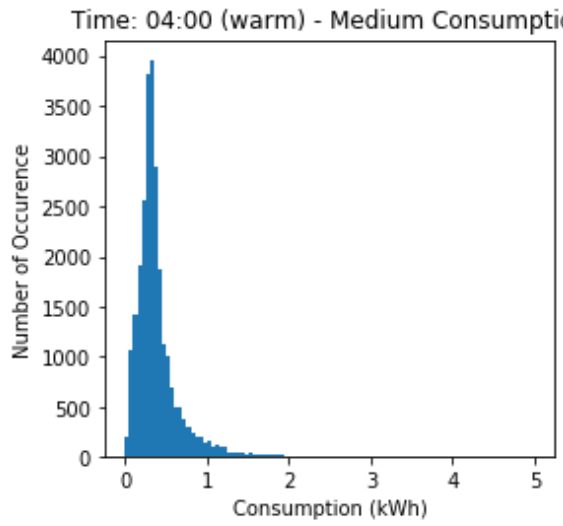

Time: 02:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

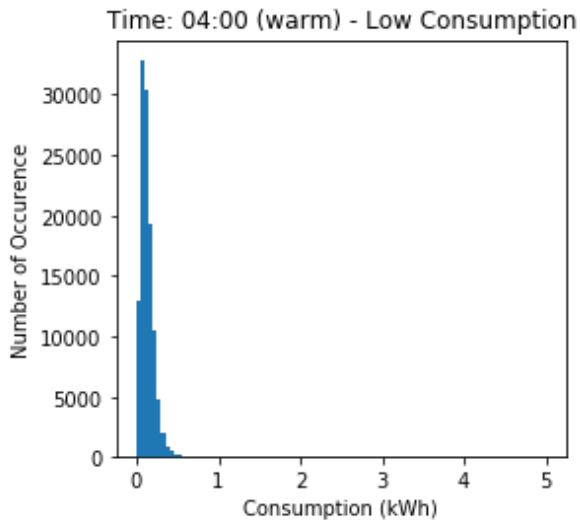

Time: 03:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

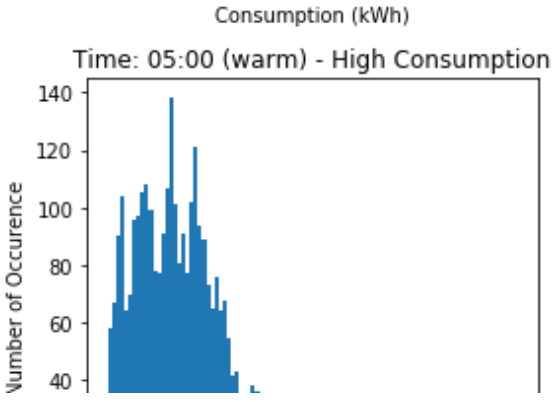

Time: 03:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

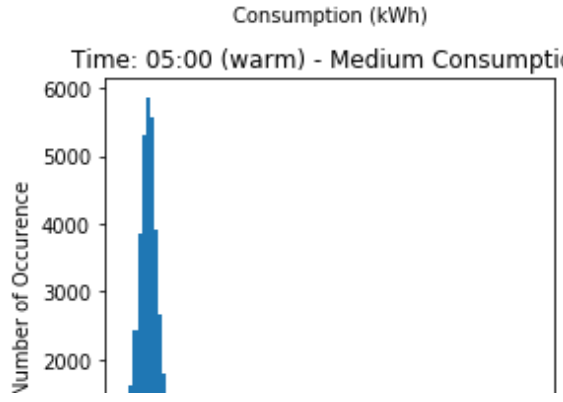

Time: 03:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

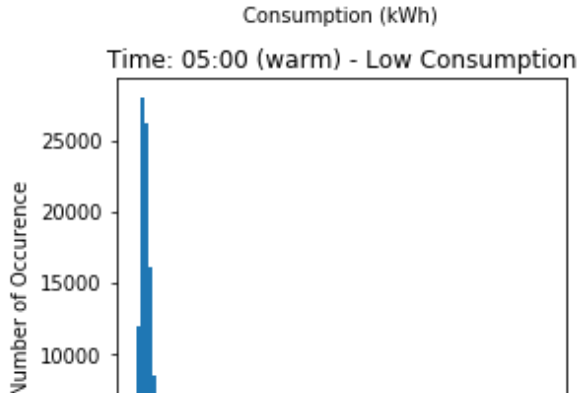

Time: 04:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

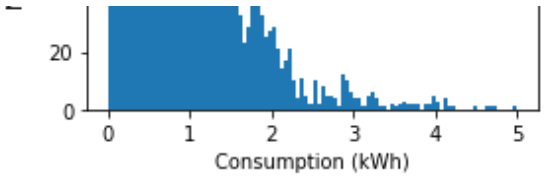

Time: 04:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

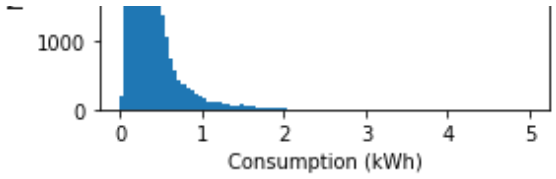

Time: 04:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

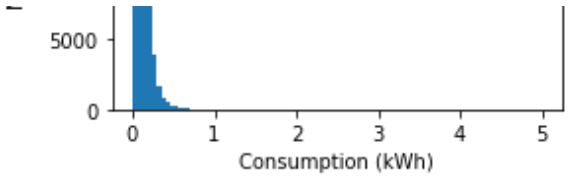

Time: 05:00 (warm) - High Consumption
Number of Occurence

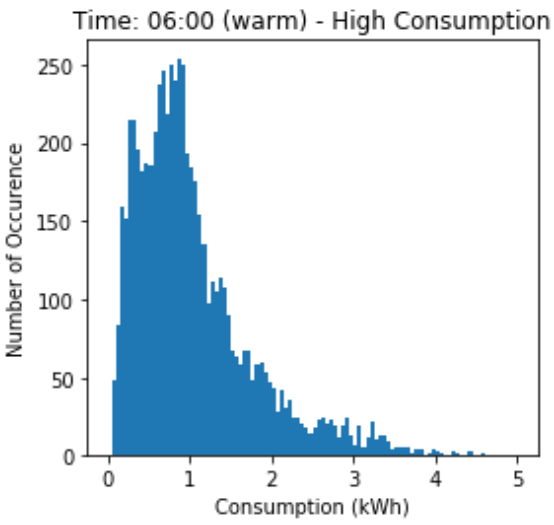

Time: 05:00 (warm) - Medium Consumption
Number of Occurence

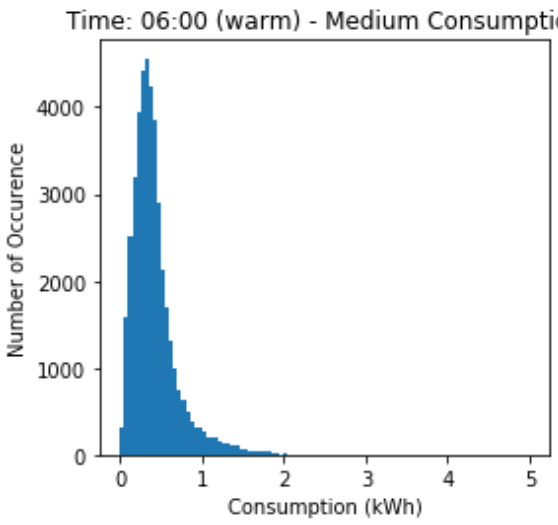

Time: 05:00 (warm) - Low Consumption
Number of Occurence

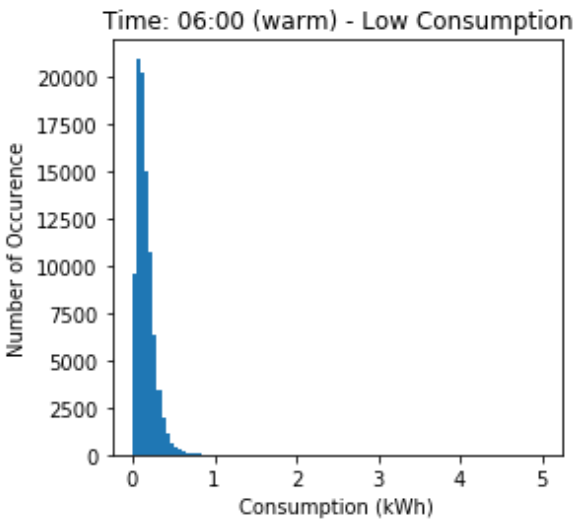

Consumption (kWh)

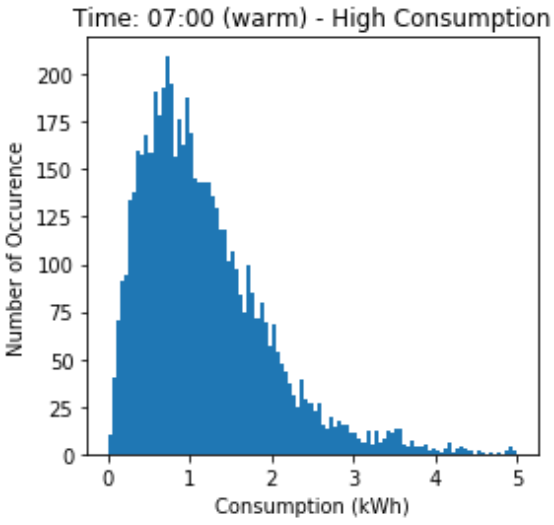

Consumption (kWh)

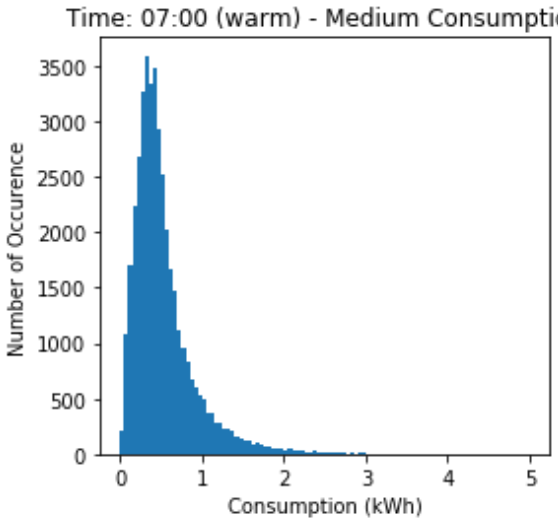

Consumption (kWh)

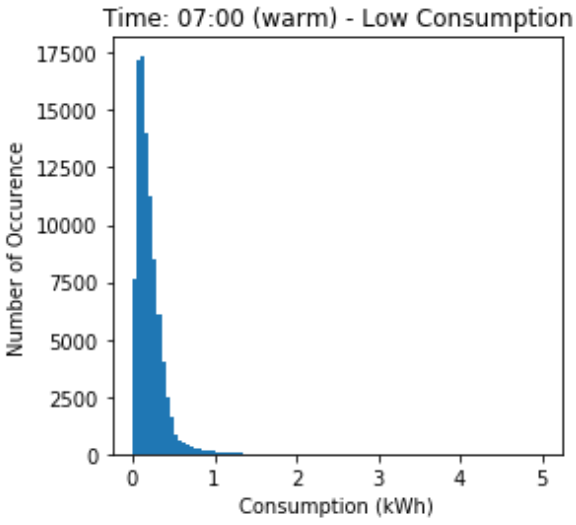

Time: 06:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

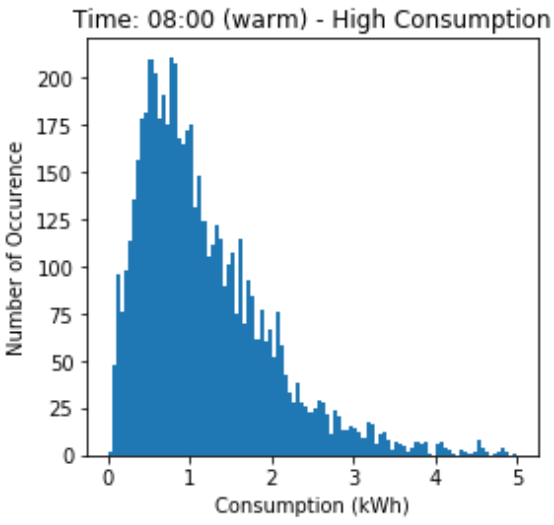

Time: 06:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

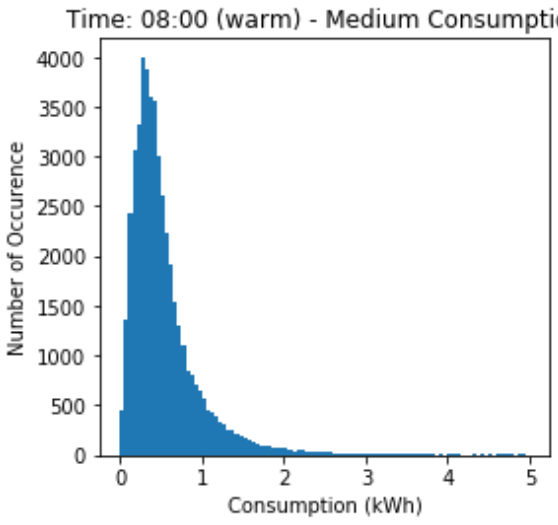

Time: 06:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

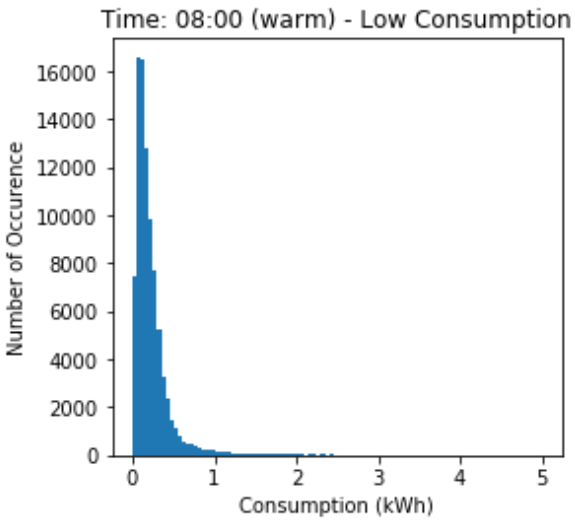

Time: 07:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

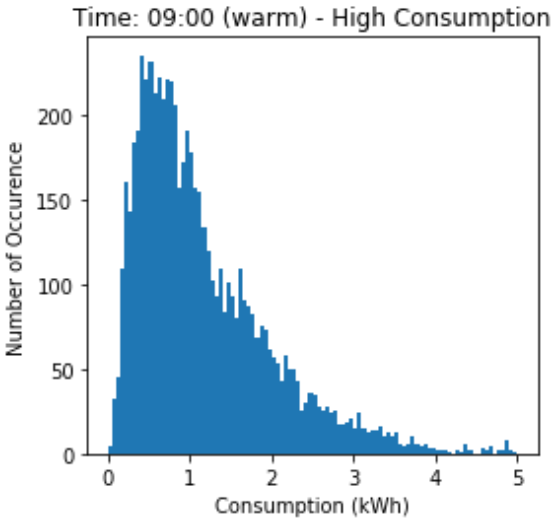

Time: 07:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

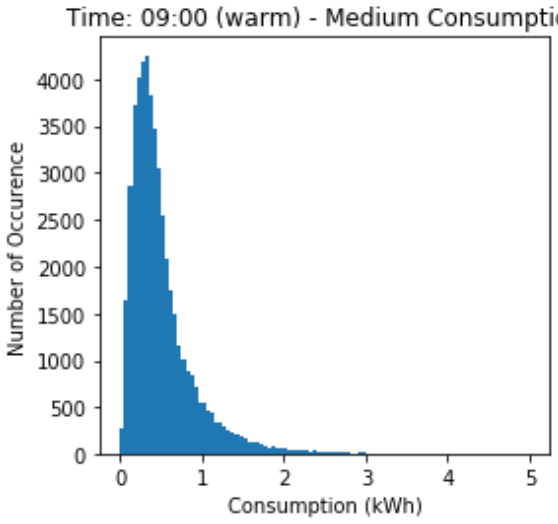

Time: 07:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

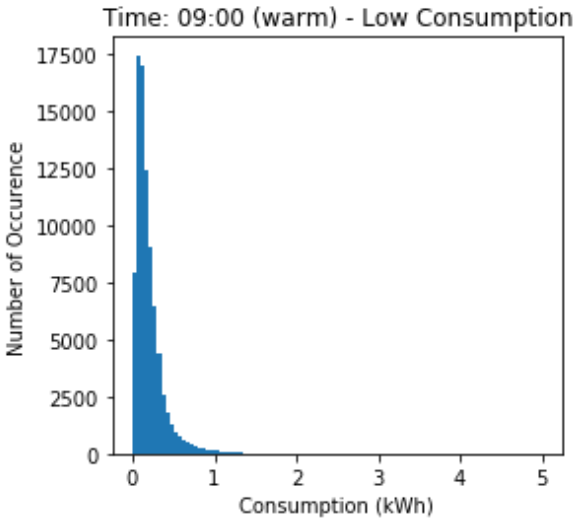

Time: 08:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

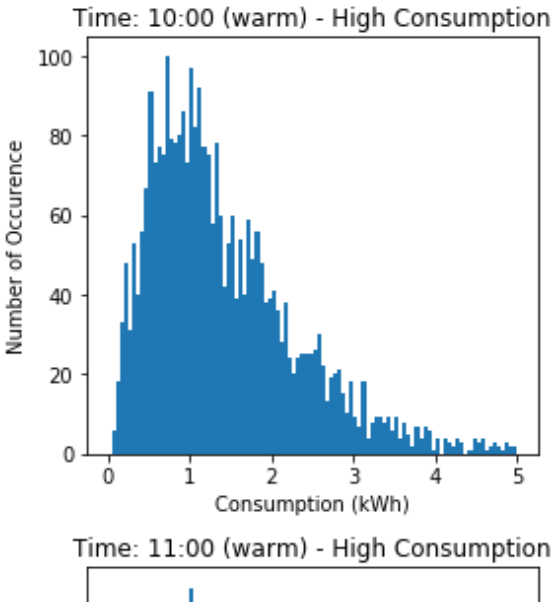

Time: 08:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

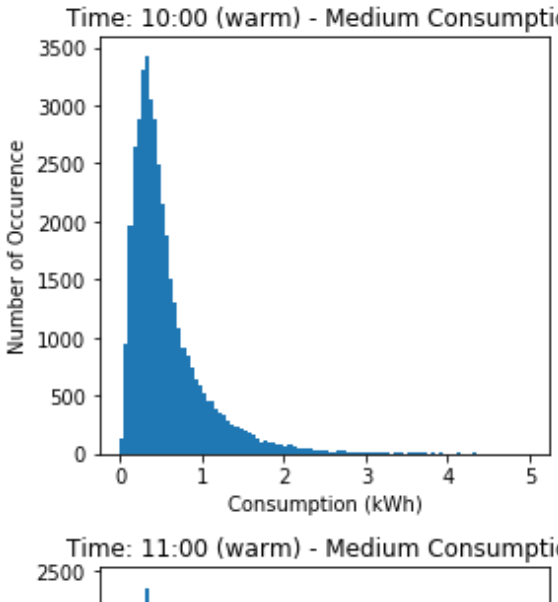

Time: 08:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

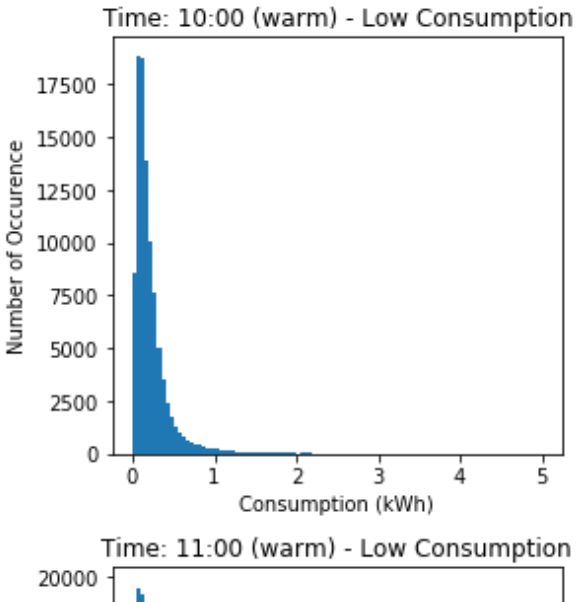

Time: 09:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)


Time: 09:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)


Time: 09:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)


Time: 10:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

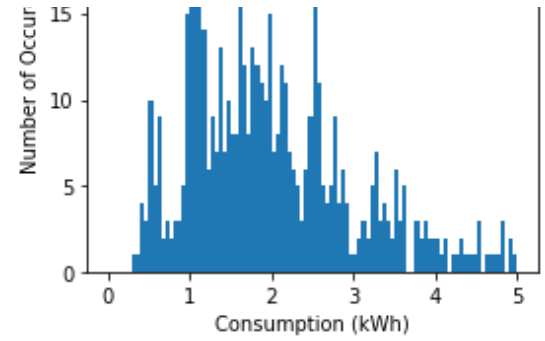

Time: 10:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)


Time: 10:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

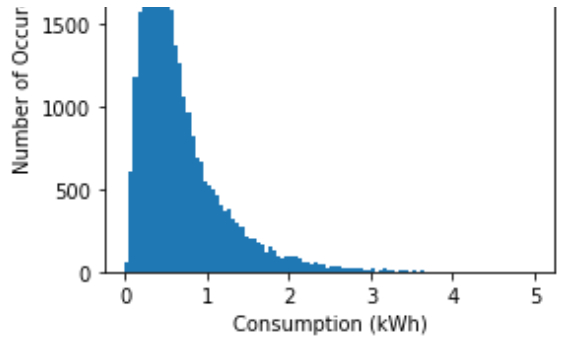
Time: 11:00 (warm) - High Consumption

Time: 11:00 (warm) - Medium Consumption

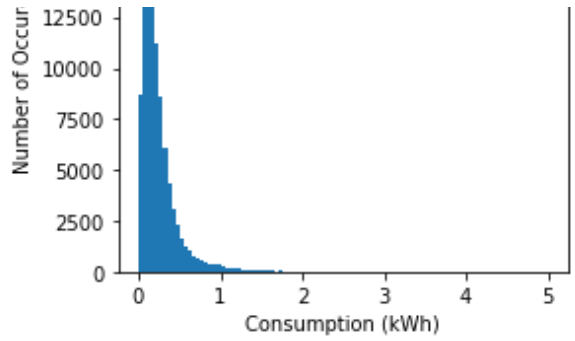
Time: 11:00 (warm) - Low Consumption



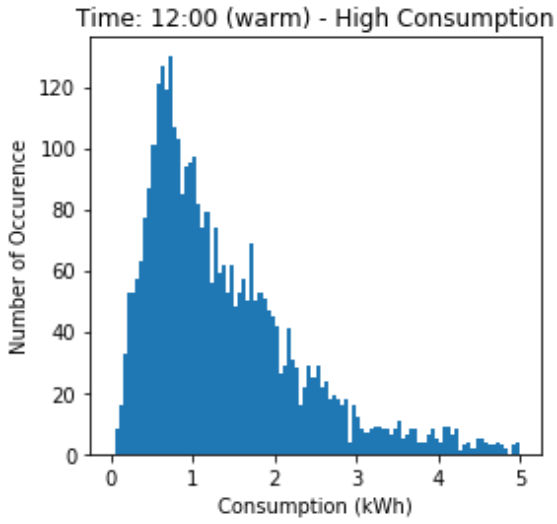
Time: 12:00 (warm) - High Consumption



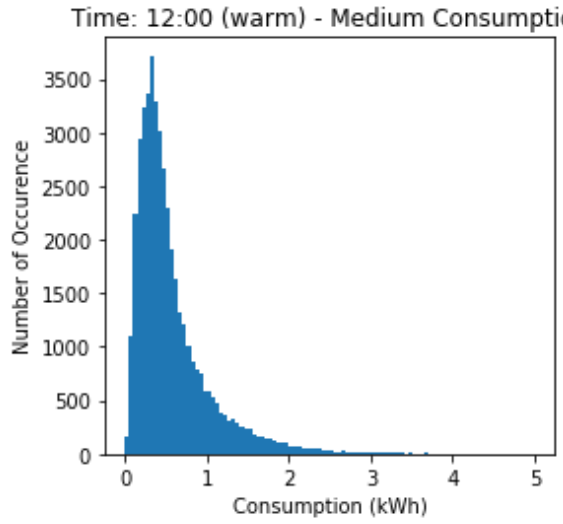
Time: 12:00 (warm) - Medium Consumption



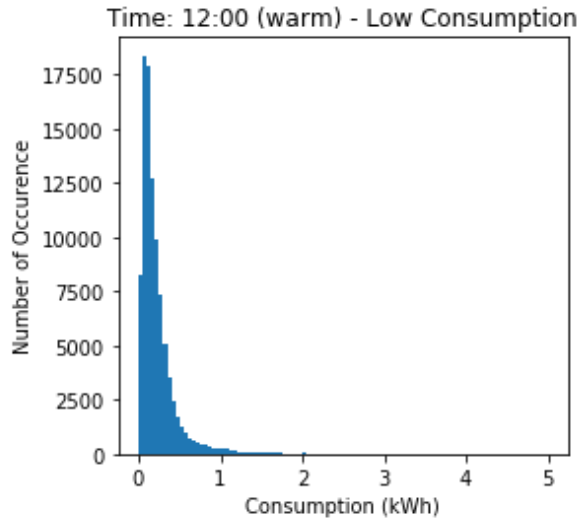
Time: 12:00 (warm) - Low Consumption



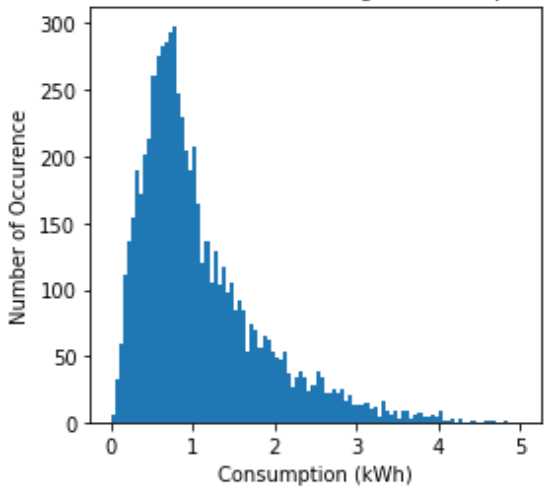
Time: 13:00 (warm) - High Consumption



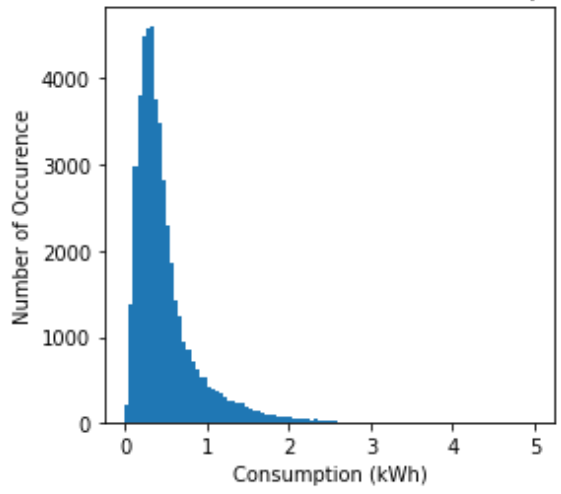
Time: 13:00 (warm) - Medium Consumption



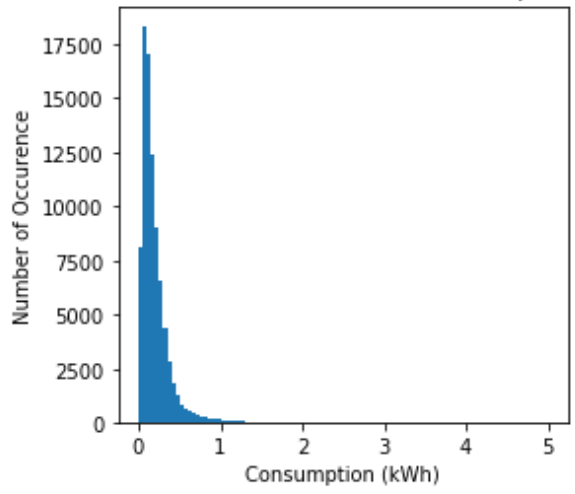
Time: 13:00 (warm) - Low Consumption



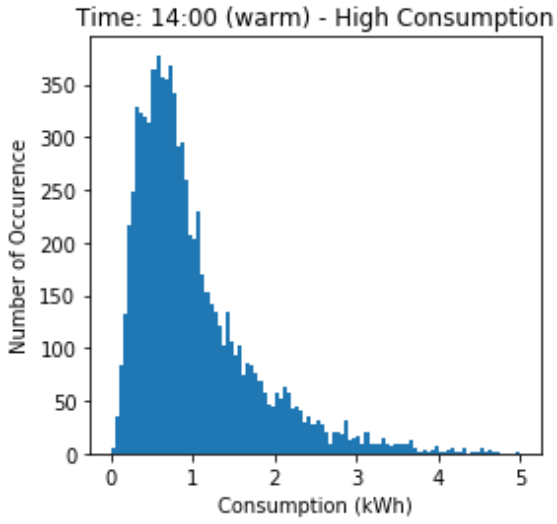
Time: 14:00 (warm) - High Consumption



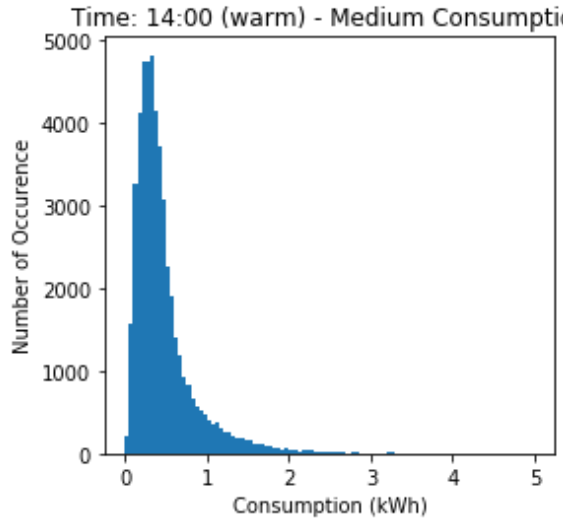
Time: 14:00 (warm) - Medium Consumption



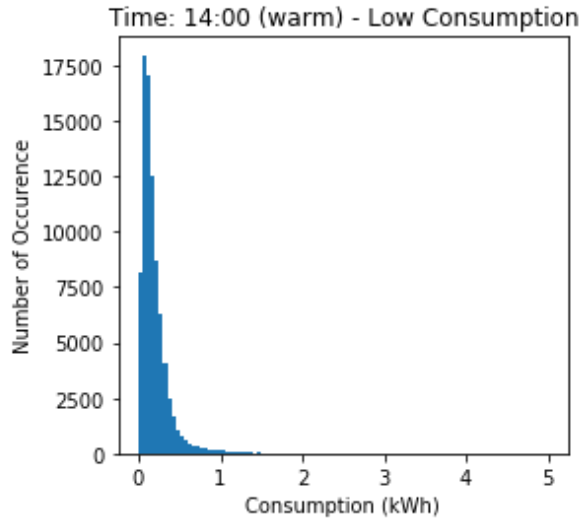
Time: 14:00 (warm) - Low Consumption



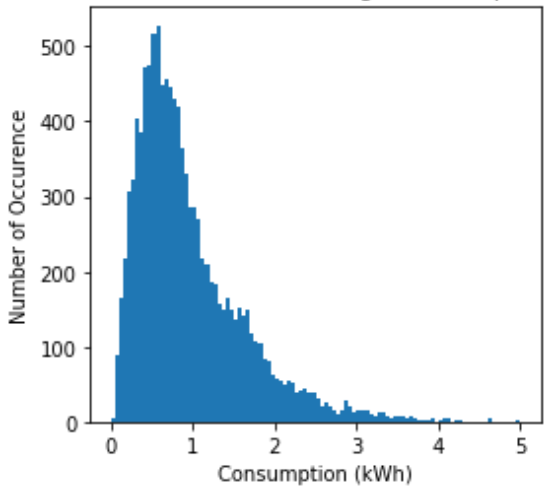
Time: 15:00 (warm) - High Consumption



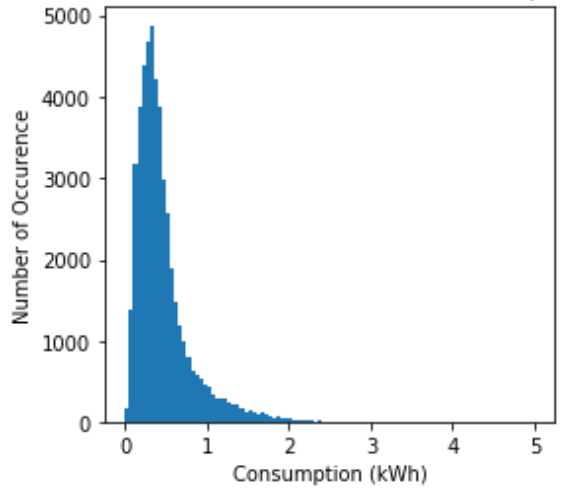
Time: 15:00 (warm) - Medium Consumption



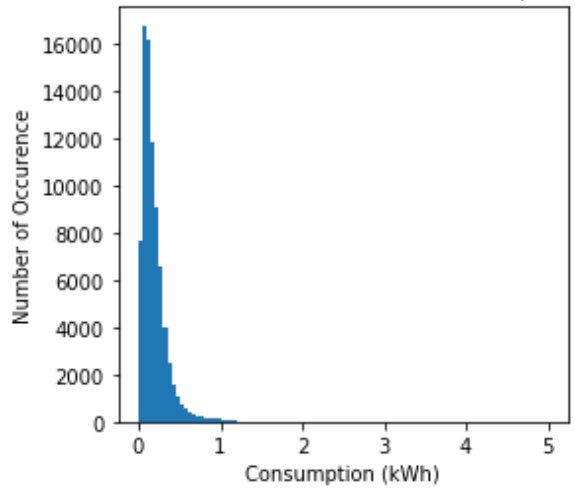
Time: 15:00 (warm) - Low Consumption



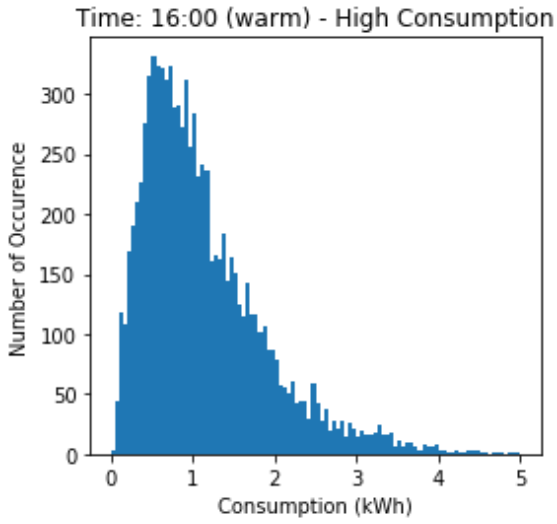
Time: 16:00 (warm) - High Consumption



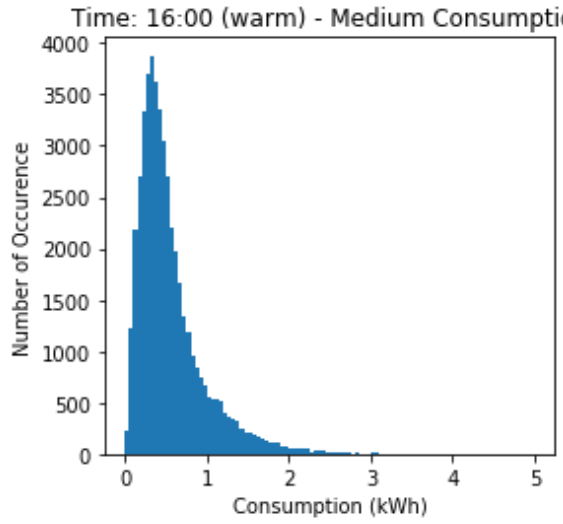
Time: 16:00 (warm) - Medium Consumption



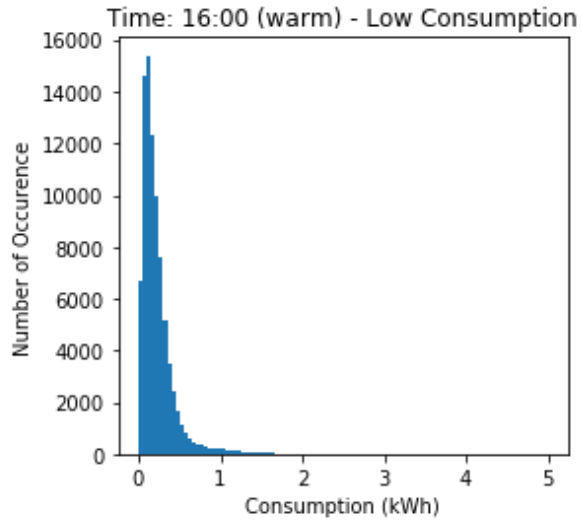
Time: 16:00 (warm) - Low Consumption



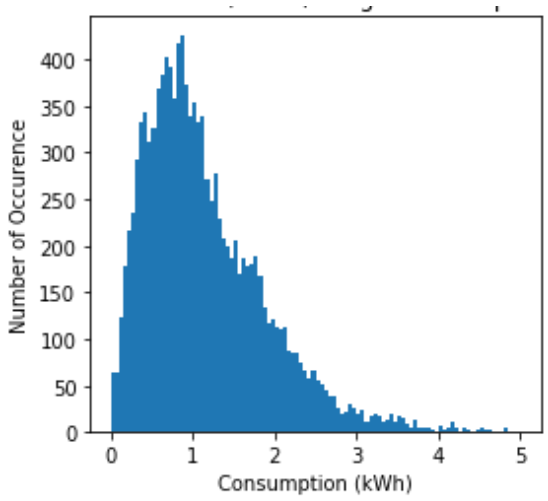
Time: 17:00 (warm) - High Consumption

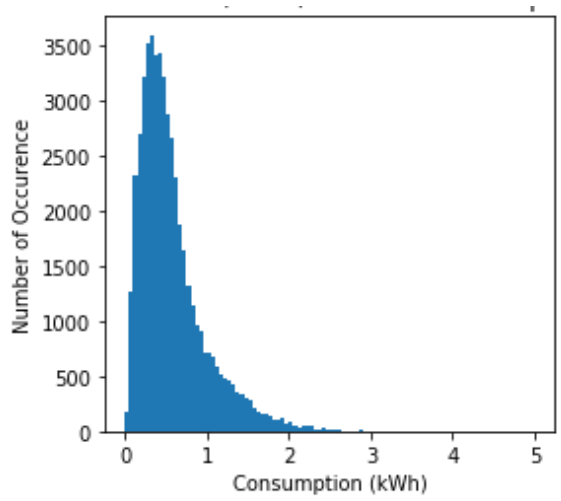


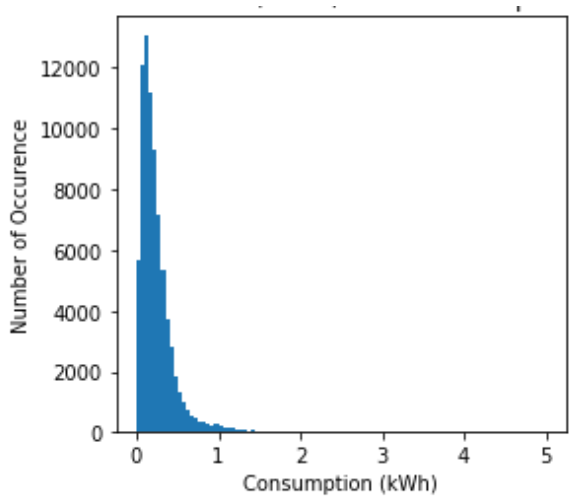
Time: 17:00 (warm) - Medium Consumption

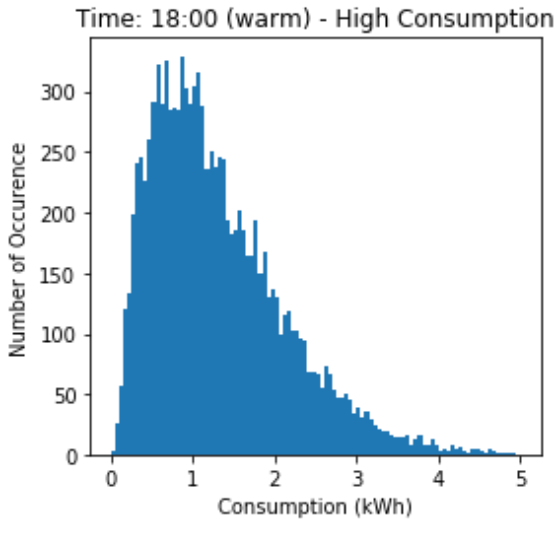


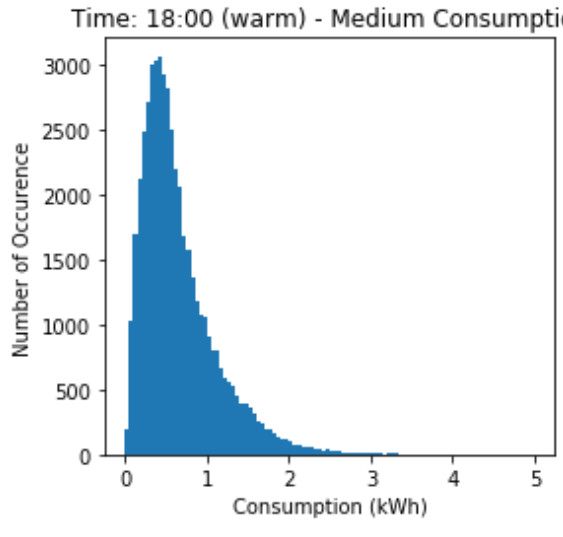
Time: 17:00 (warm) - Low Consumption

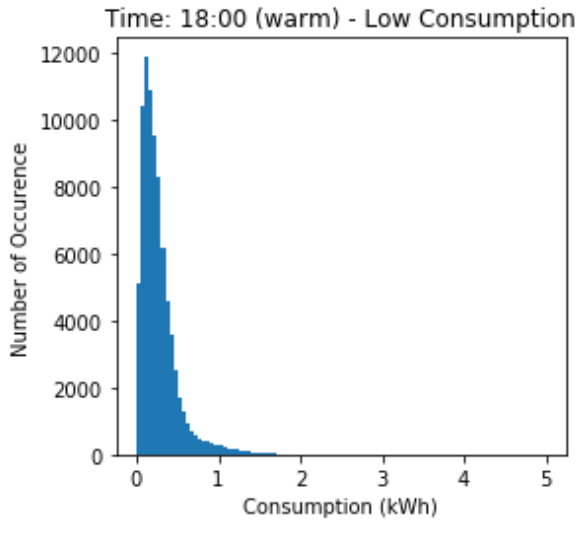

Number of Occurence
Consumption (kWh)

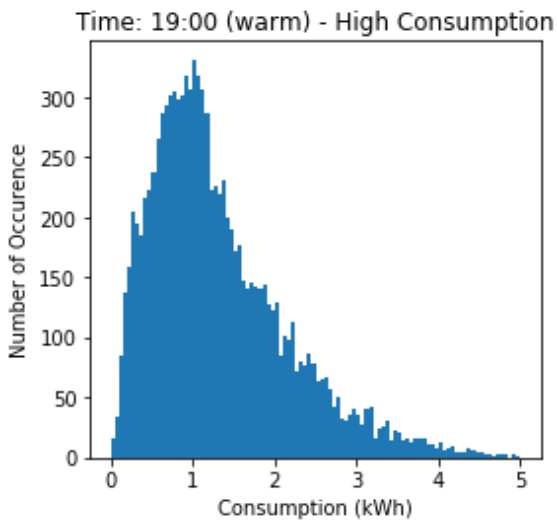

Number of Occurence
Consumption (kWh)

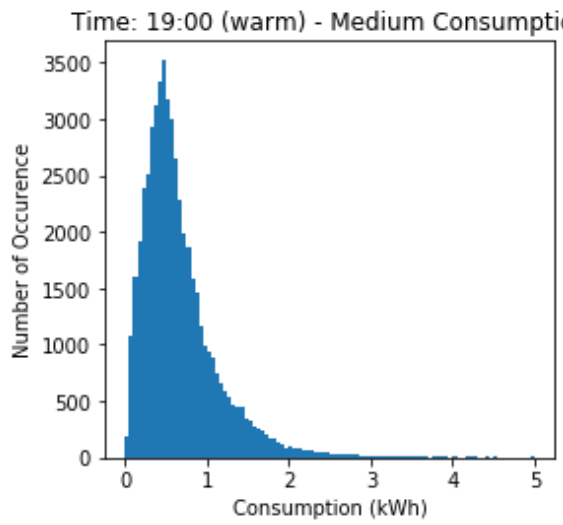

Number of Occurence
Consumption (kWh)

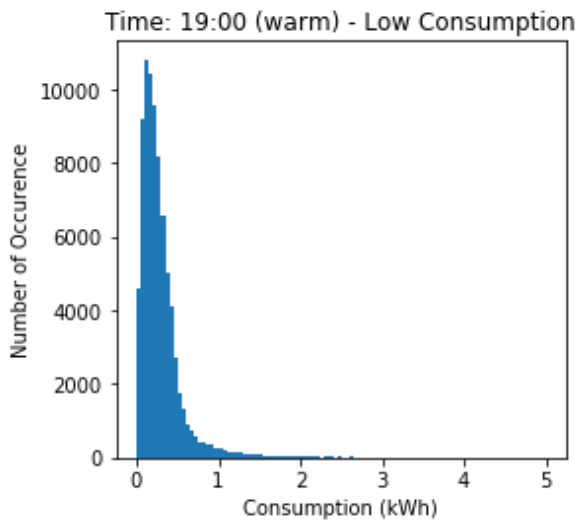

Time: 18:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

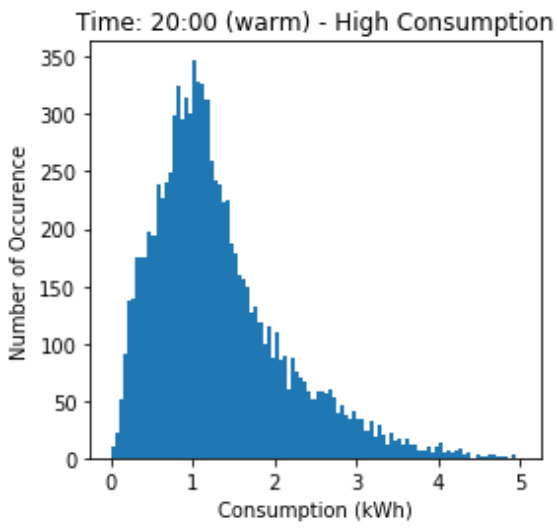

Time: 18:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

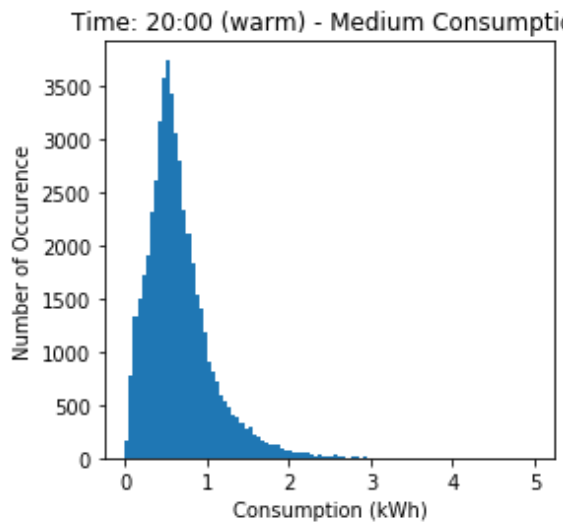

Time: 18:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

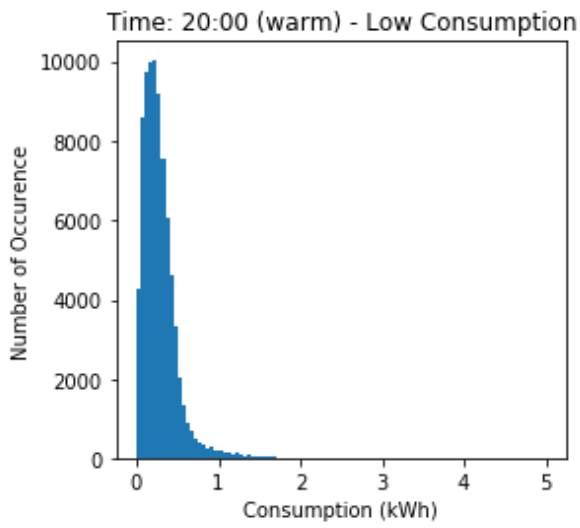

Time: 19:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

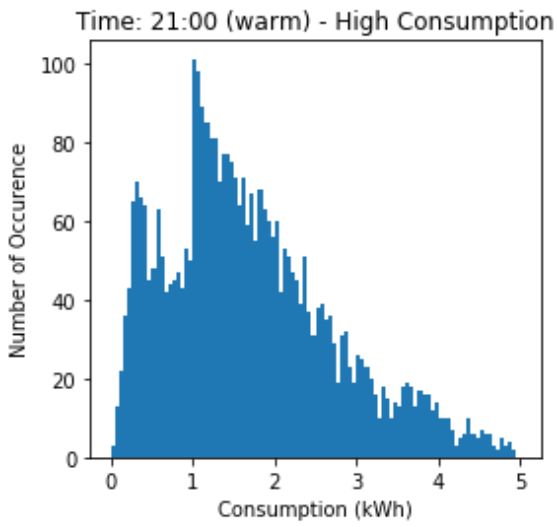

Time: 19:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

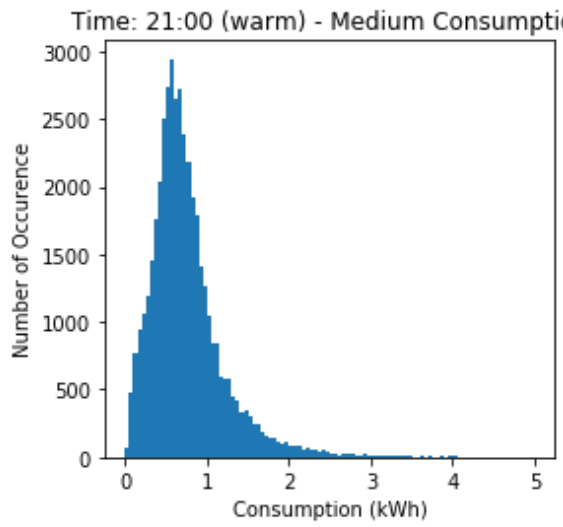

Time: 19:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

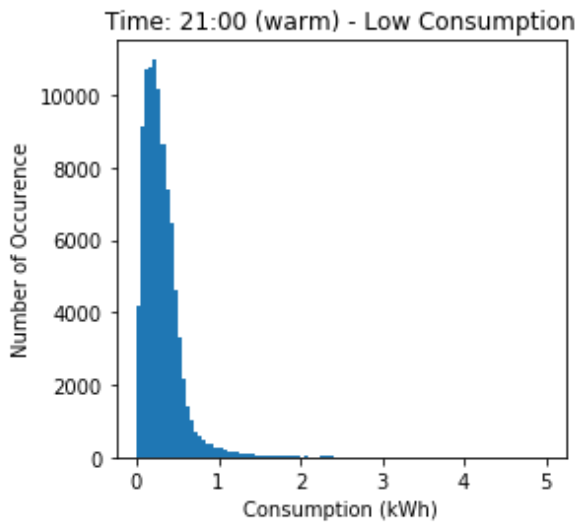

Time: 20:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

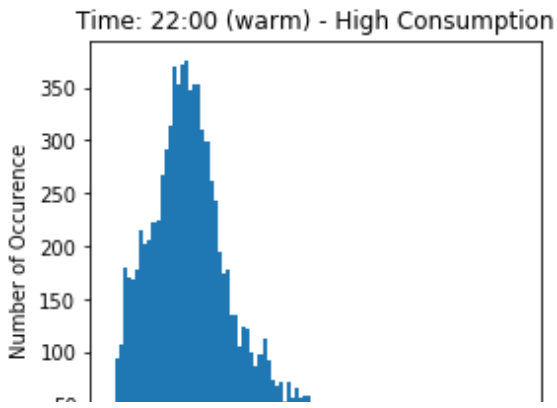

Time: 20:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)

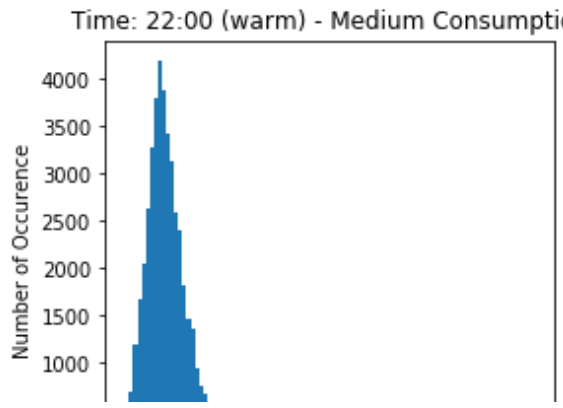

Time: 20:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)

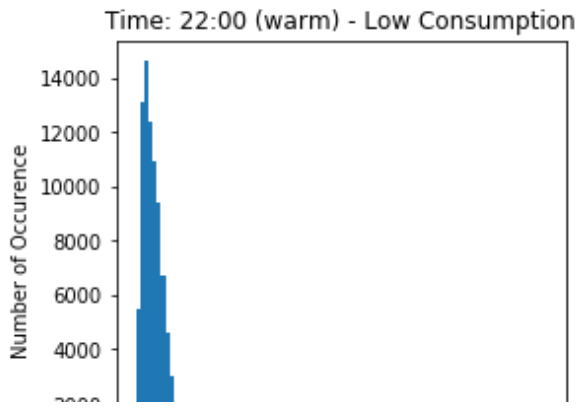

Time: 21:00 (warm) - High Consumption
Number of Occurence
Consumption (kWh)

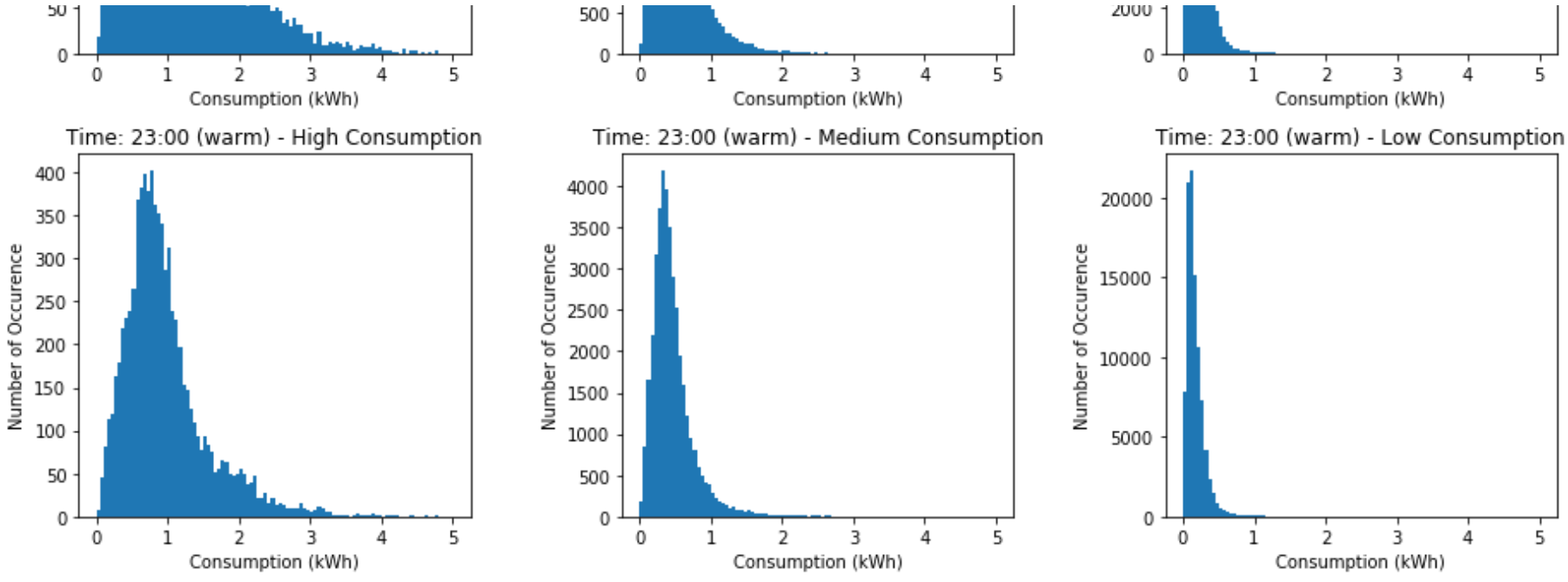

Time: 21:00 (warm) - Medium Consumption
Number of Occurence
Consumption (kWh)


Time: 21:00 (warm) - Low Consumption
Number of Occurence
Consumption (kWh)


Time: 22:00 (warm) - High Consumption
Number of Occurence

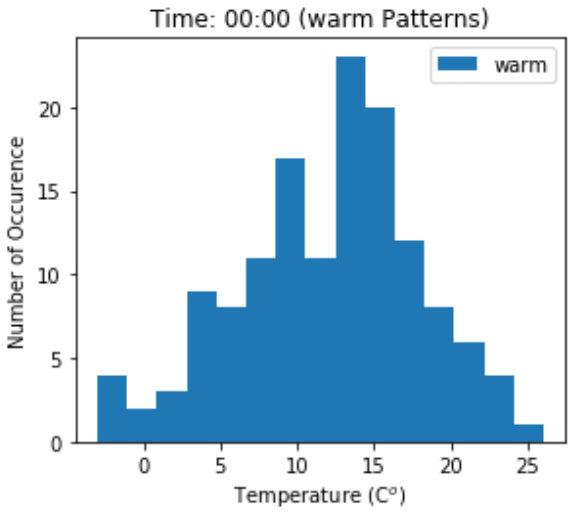

Time: 22:00 (warm) - Medium Consumption
Number of Occurence

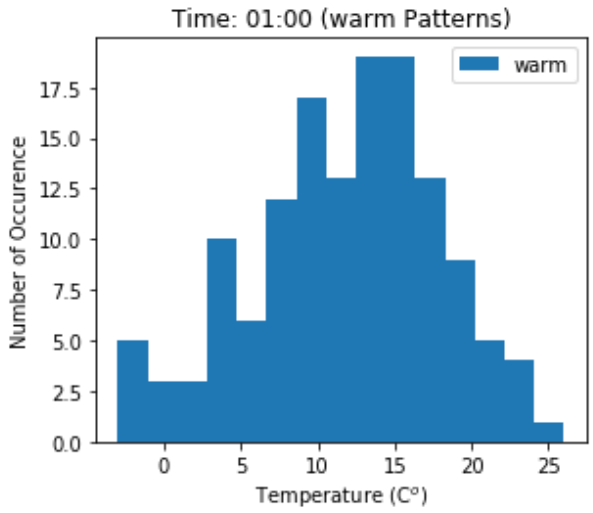

Time: 22:00 (warm) - Low Consumption
Number of Occurence

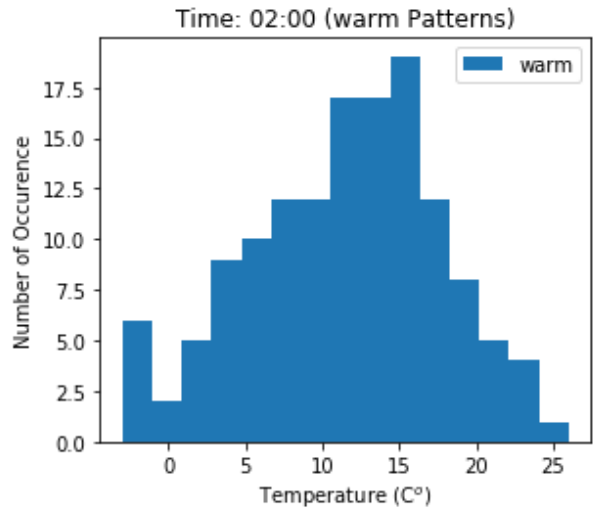


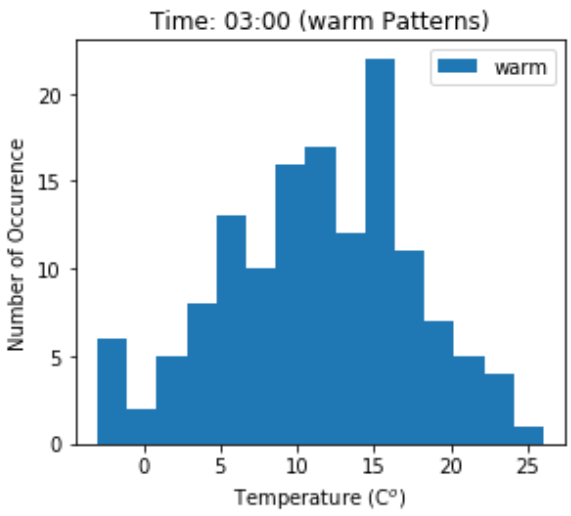
General days of a week distribution property of each cluster at each hour

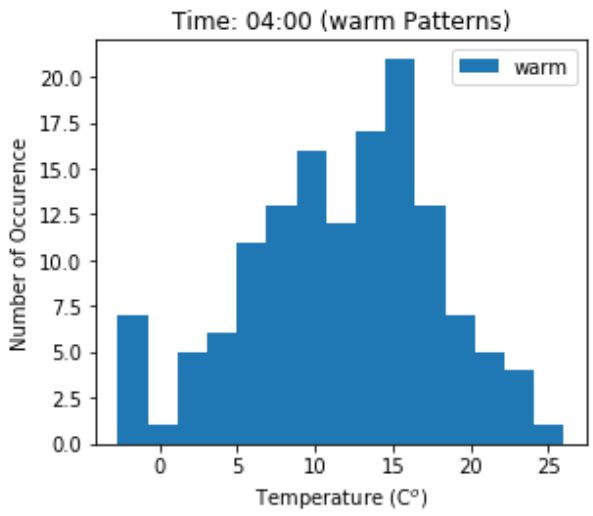
In [22]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.hist(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]['TempC'], bins=1
5, label = 'warm')
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
    else:
        ax.hist(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=15, lab
el = 'warm')
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
plt.tight_layout()
```

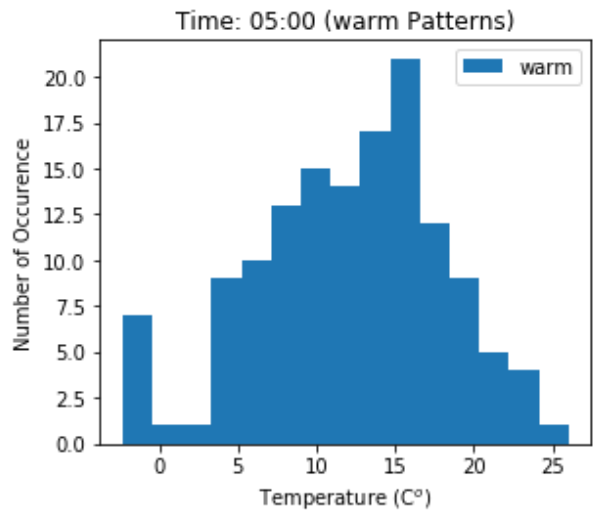

Time: 00:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

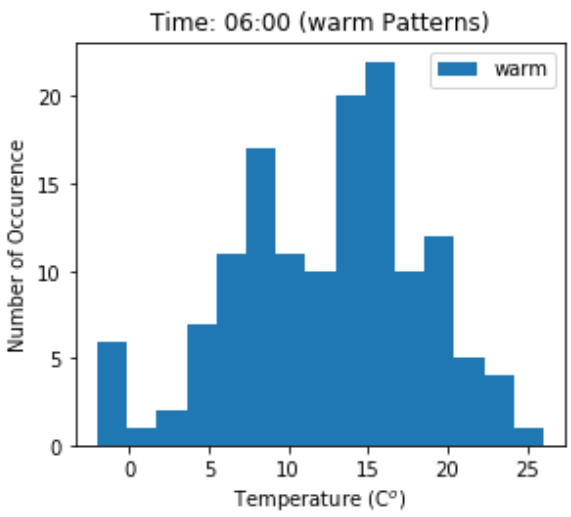

Time: 01:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

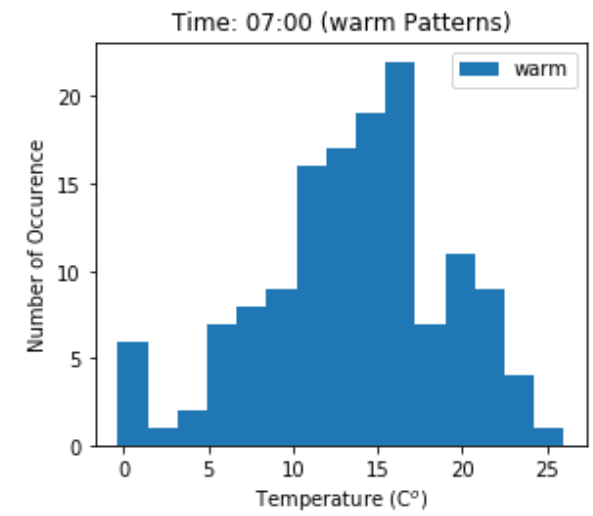

Time: 02:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

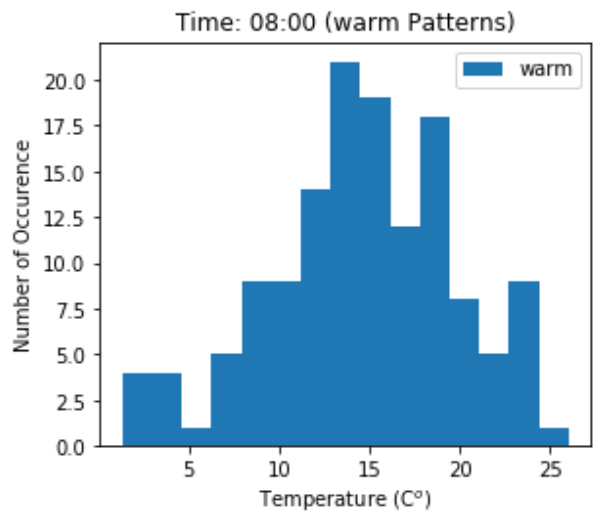

Time: 03:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

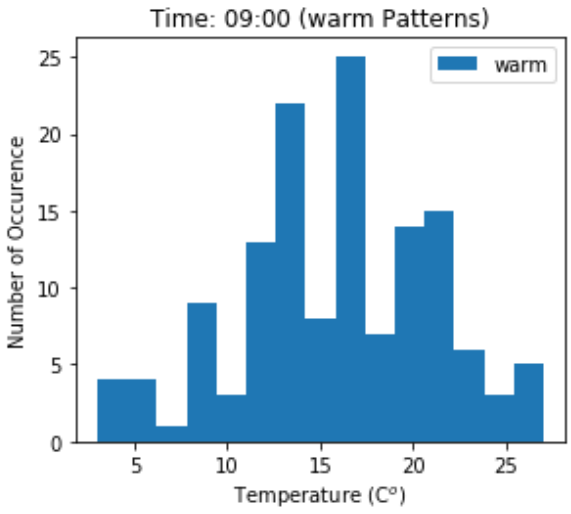

Time: 04:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

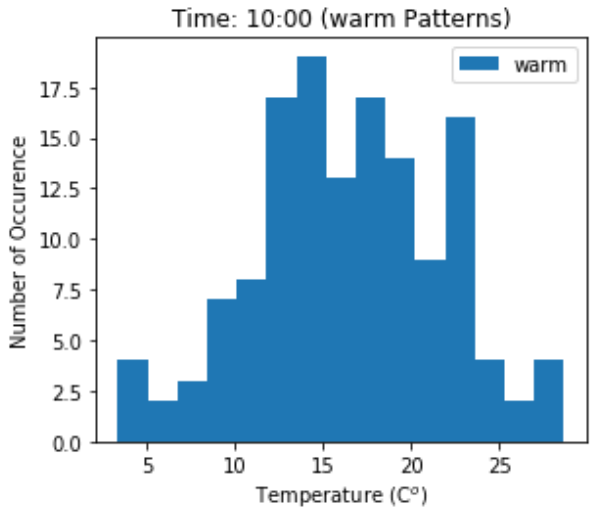

Time: 05:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

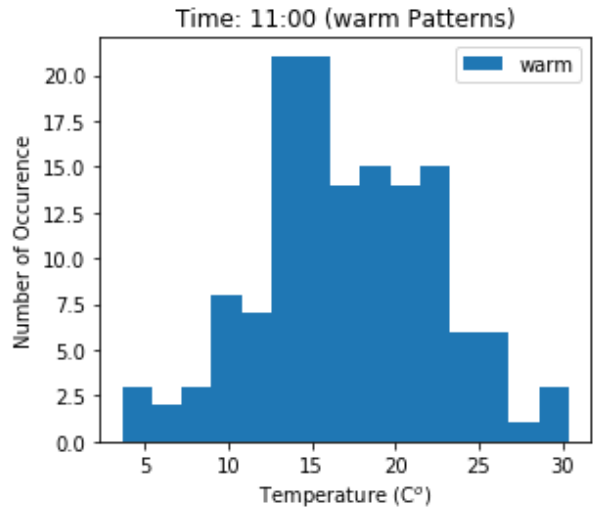

Time: 06:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

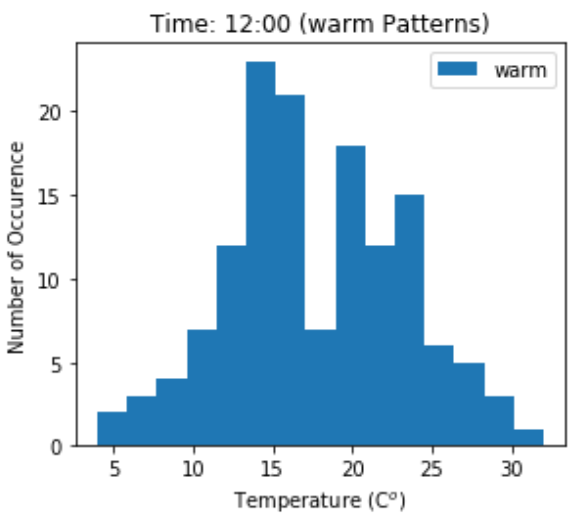

Time: 07:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

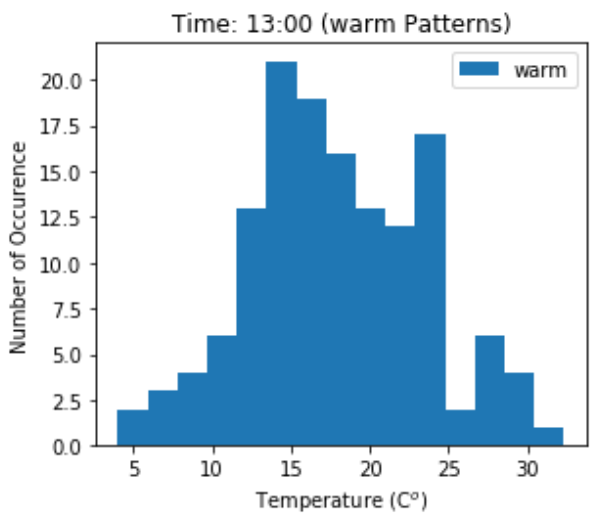

Time: 08:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

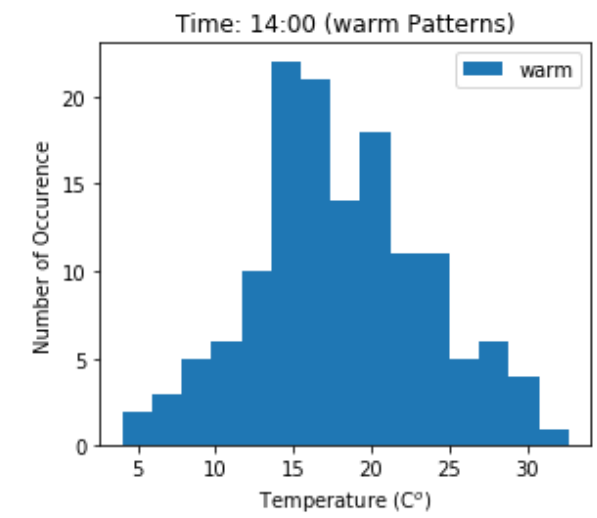

Time: 09:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

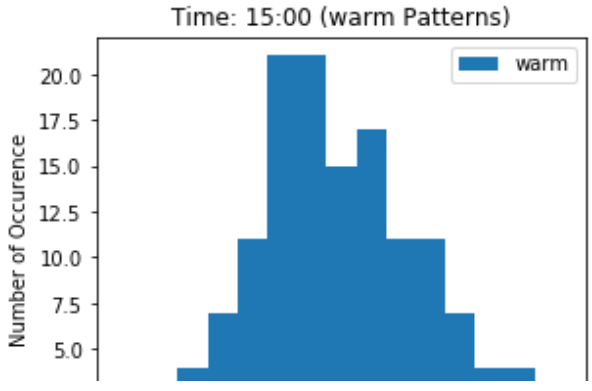

Time: 10:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

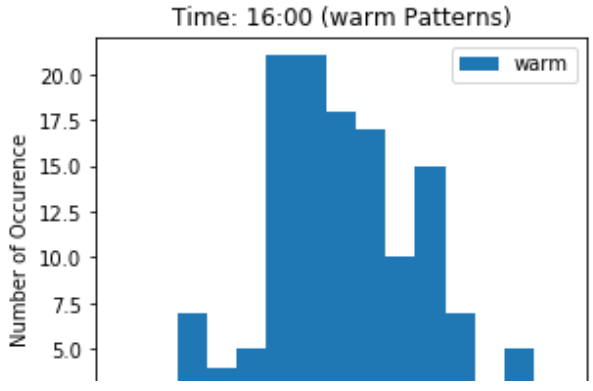

Time: 11:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

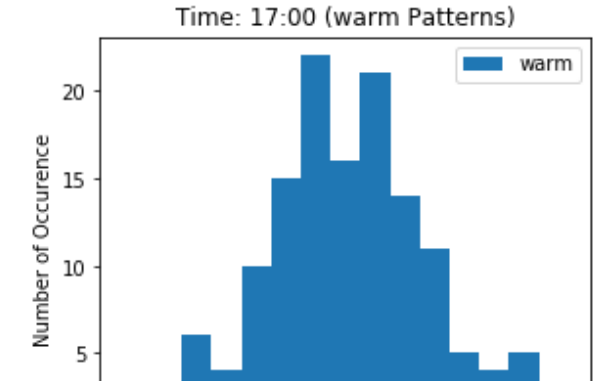

Time: 12:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

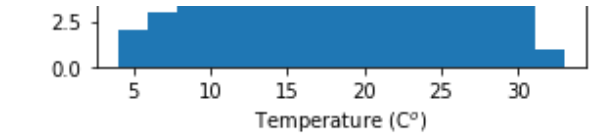

Time: 13:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

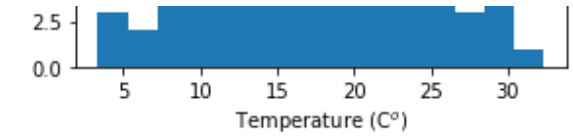

Time: 14:00 (warm Patterns)
warm
Number of Occurence
Temperature (C°)

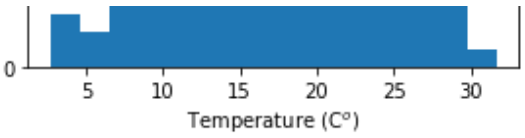

Time: 15:00 (warm Patterns)
warm
Number of Occurence

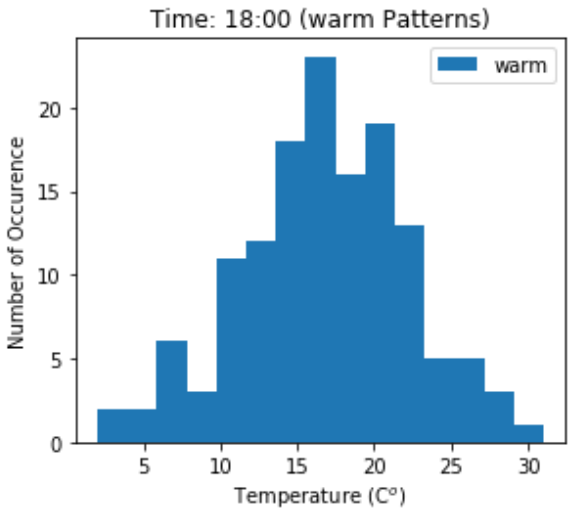

Time: 16:00 (warm Patterns)
warm
Number of Occurence

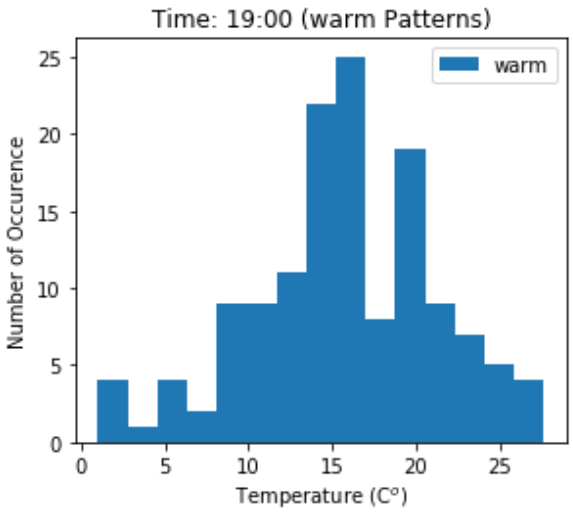

Time: 17:00 (warm Patterns)
warm
Number of Occurence

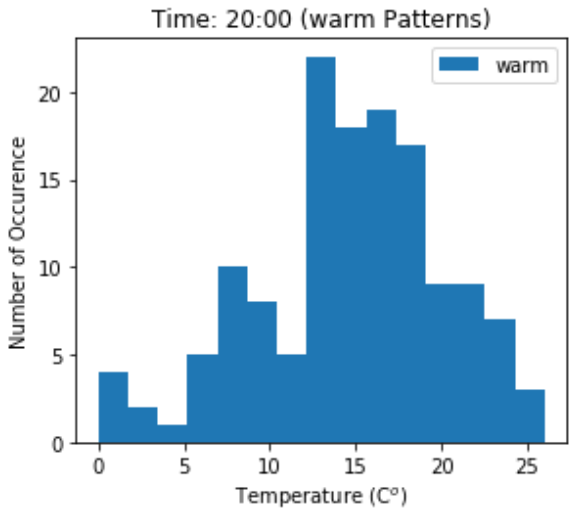


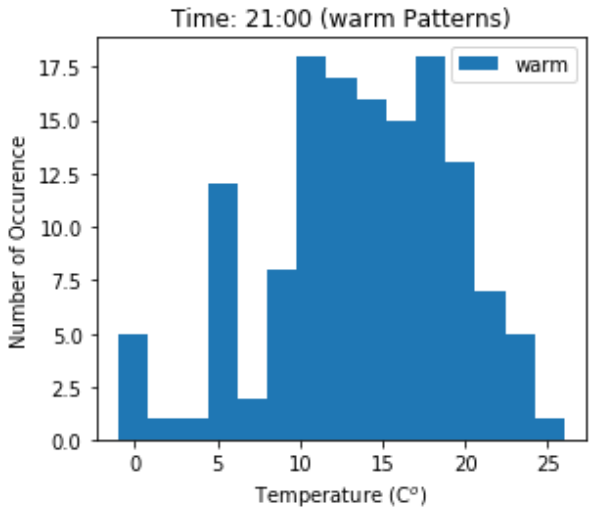


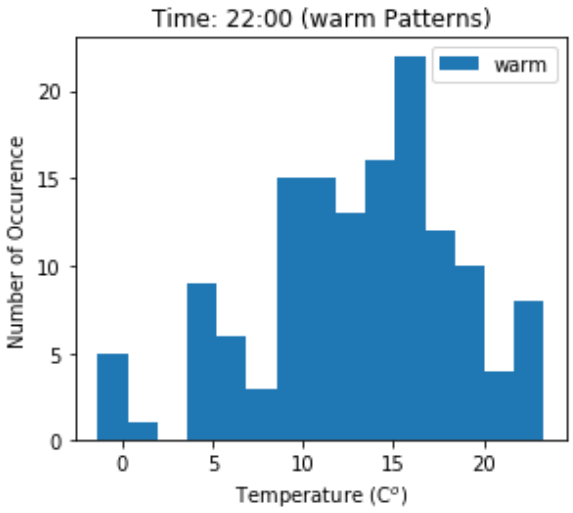


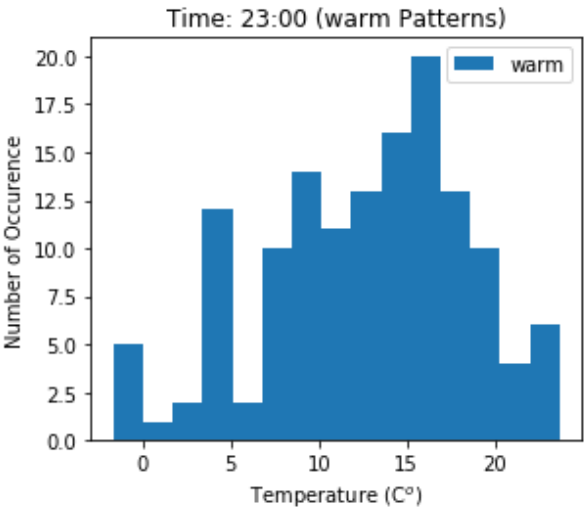






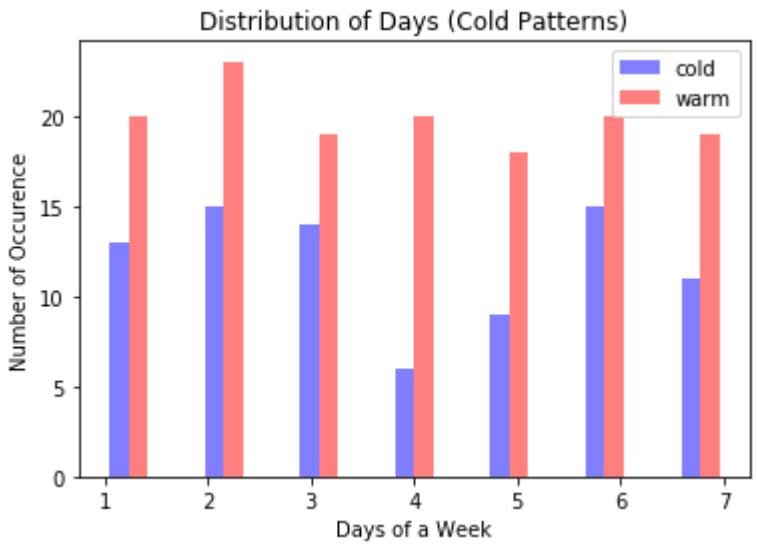






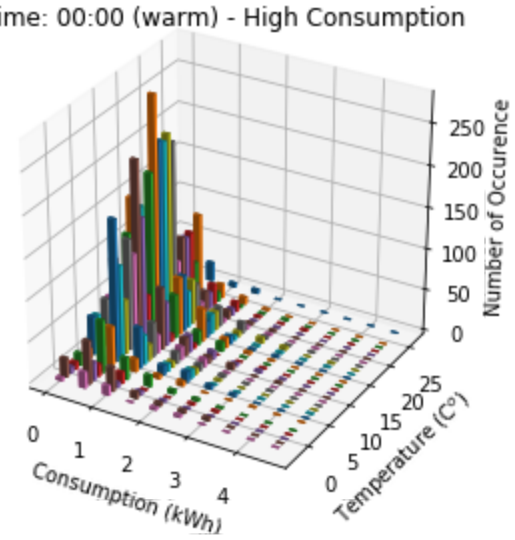
The distribution of days of a week (since it's non-event days, so it's worth checking this property)

In [24]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (6,4))
ax = fig_all.add_subplot(1, 1, 1)
ax.hist([pd.to_datetime(df_wea1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains('00:00:00')]['GMT']).dt.dayofweek
+ 1,
         pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('00:00:00')]['GMT']).dt.dayofweek
+ 1], bins=13, alpha = 0.5, color = ['blue', 'red'], label = ['cold', 'warm'])
ax.set_title('Distribution of Days' + ' (Cold Patterns)')
ax.legend()
ax.set_xlabel('Days of a Week')
ax.set_ylabel('Number of Occurence')
plt.show()
```

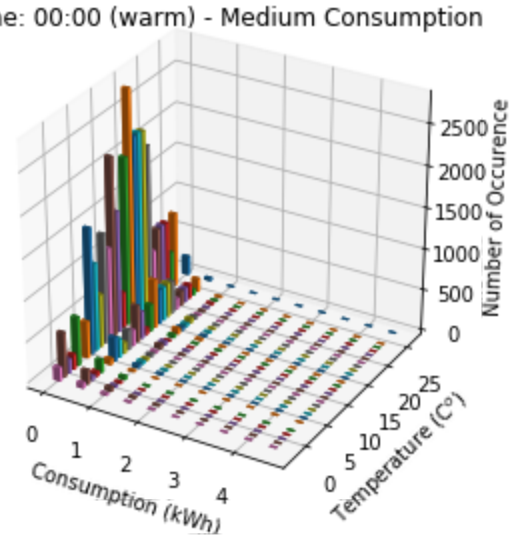


In [25]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1, projection = '3d')
            t2 = t[t.columns[kmeans.labels_ == sorted_label[k][0]]].copy() # select all data from the n-th cl
uster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00'
)]['TempC'].values) # add temperature column to dataframe
            temp_list = list(set(t2['TempC']))
            temp_list.sort(reverse = True)
            for temp in temp_list:
                t3 = t2[t2['TempC'] == temp] # select the data with a specific temperature
                t3 = np.histogram(t3[t3.columns[1:-1]].values.flatten(), bins=10, range=(0,5))
                xpos = t3[1][:-1]
                ypos = np.ones(10) * temp
                num_elements = len(xpos)
                zpos = np.zeros(num_elements)
                dx = np.ones(10) * 0.15
                dy = np.ones(10) * 0.5
                dz = t3[0]
                pl = ax.bar3d(xpos, ypos, zpos, dx, dy, dz, alpha = 0.5, zsort='max')
                pl._sort_zpos = i * 3 + k + 1
            ax.set_title('Time: 0' + str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel(r'Temperature (C$^o$)')
            ax.set_zlabel('Number of Occurence')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1, projection = '3d')
            t2 = t[t.columns[kmeans.labels_ == sorted_label[k][0]]].copy() # select all data from the n-th cl
uster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['Te
mpC'].values) # add temperature column to dataframe
            temp_list = list(set(t2['TempC']))
            temp_list.sort(reverse = True)
            for temp in temp_list:
                t3 = t2[t2['TempC'] == temp] # select the data with a specific temperature
                t3 = np.histogram(t3[t3.columns[1:-1]].values.flatten(), bins=10, range=(0,5))
                xpos = t3[1][:-1]
                ypos = np.ones(10) * temp
                num_elements = len(xpos)
                zpos = np.zeros(num_elements)
                dx = np.ones(10) * 0.15
                dy = np.ones(10) * 0.5
                dz = t3[0]
                pl = ax.bar3d(xpos, ypos, zpos, dx, dy, dz, alpha = 0.5, zsort='max')
                pl._sort_zpos = i * 3 + k + 1
            ax.set_title('Time: ' +str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel(r'Temperature (C$^o$)')
            ax.set_zlabel('Number of Occurence')
plt.tight_layout()
```

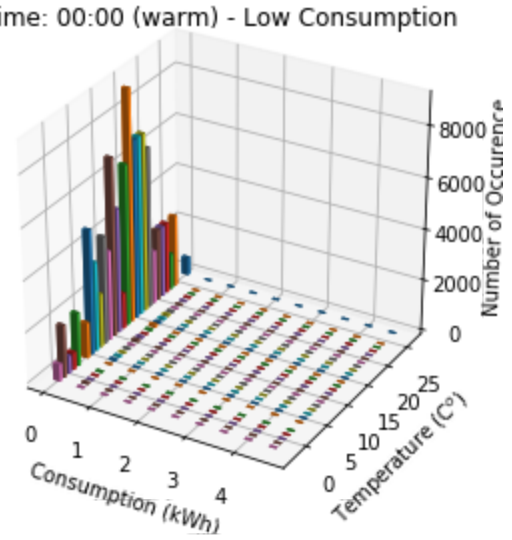
Time: 00:00 (warm) - High Consumption



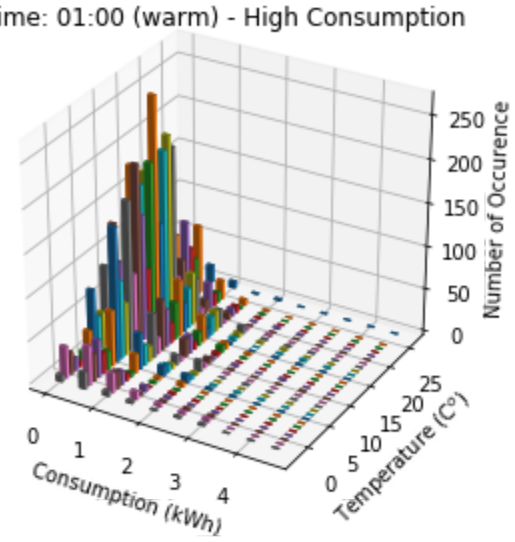
Time: 00:00 (warm) - Medium Consumption



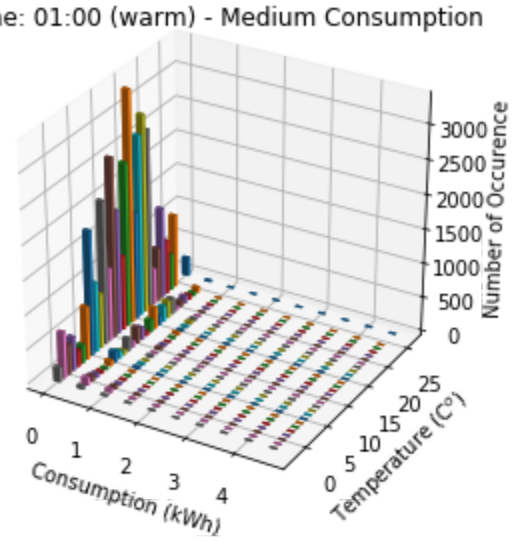
Time: 00:00 (warm) - Low Consumption



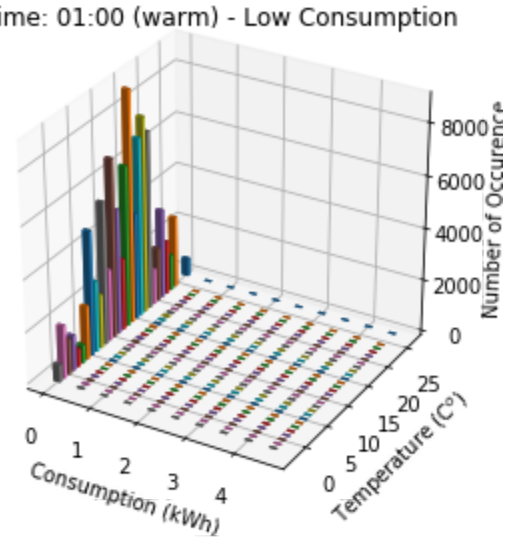
Time: 01:00 (warm) - High Consumption



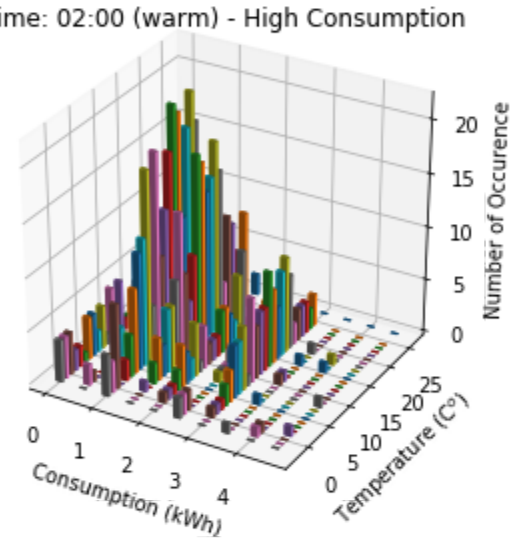
Time: 01:00 (warm) - Medium Consumption



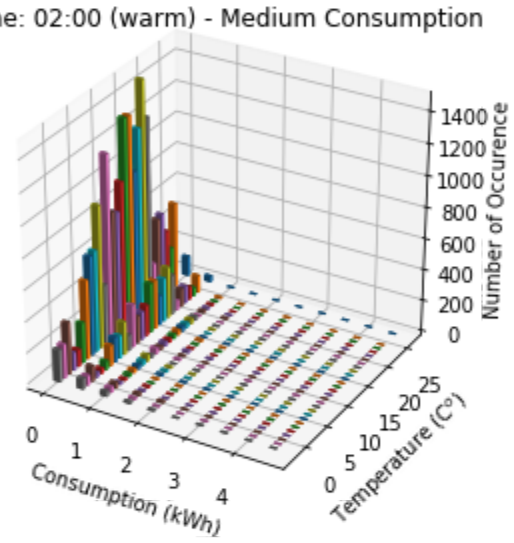
Time: 01:00 (warm) - Low Consumption



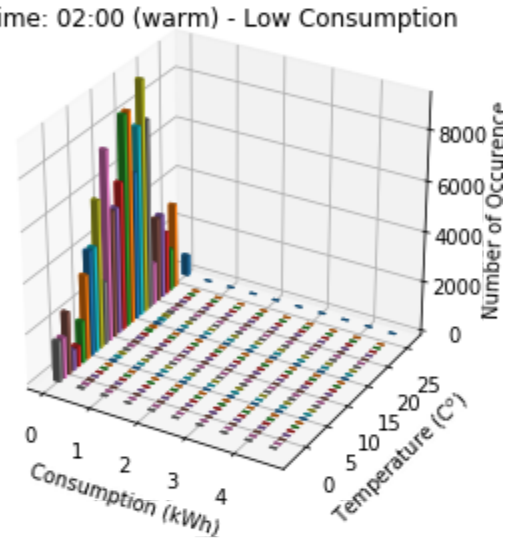
Time: 02:00 (warm) - High Consumption



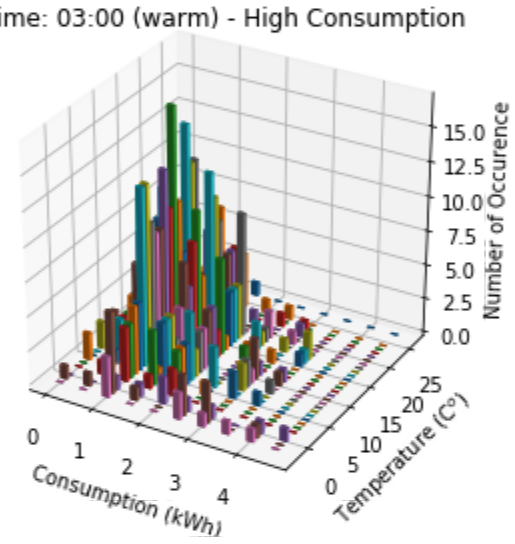
Time: 02:00 (warm) - Medium Consumption



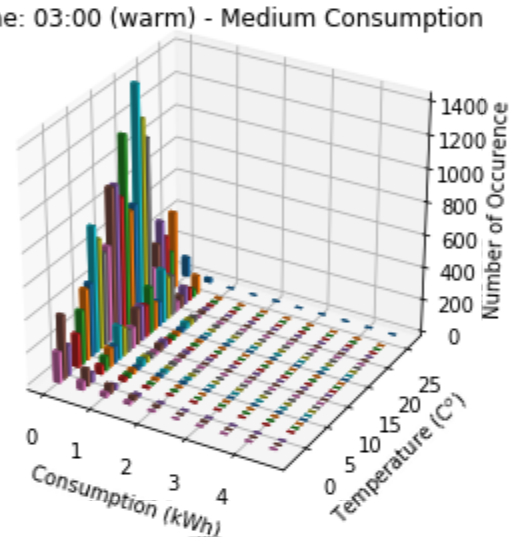
Time: 02:00 (warm) - Low Consumption



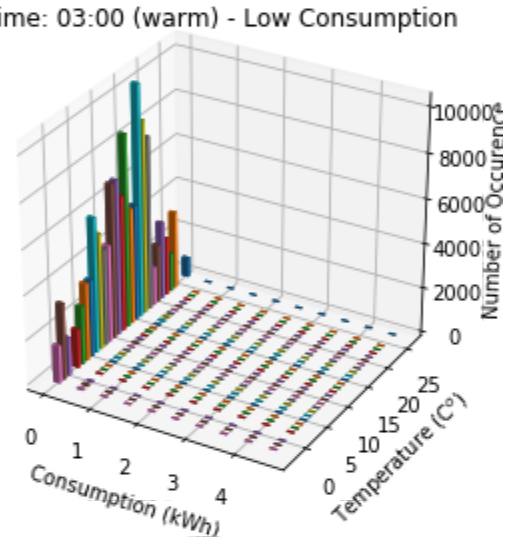
Time: 03:00 (warm) - High Consumption



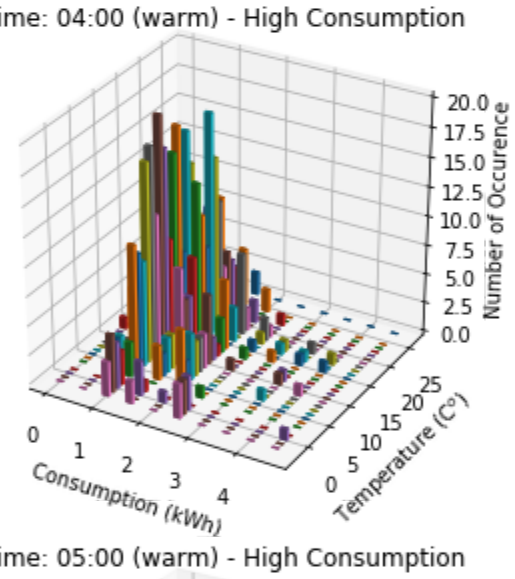
Time: 03:00 (warm) - Medium Consumption



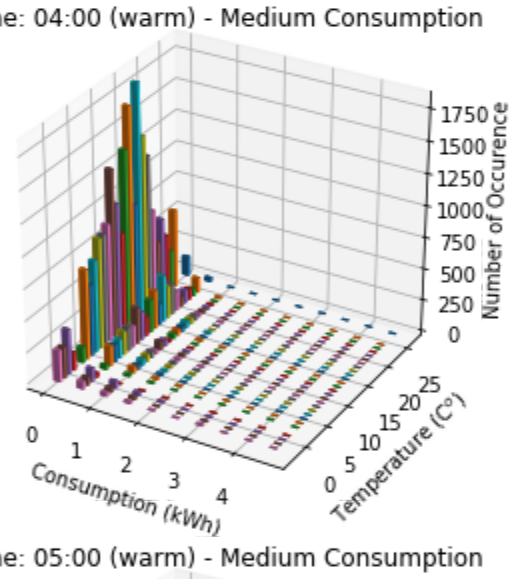
Time: 03:00 (warm) - Low Consumption



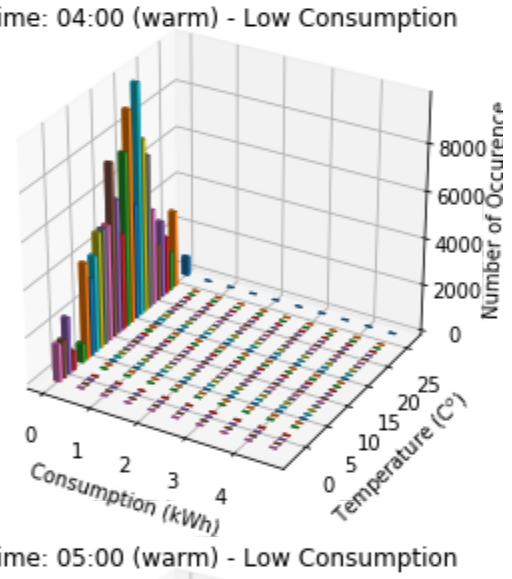
Time: 04:00 (warm) - High Consumption



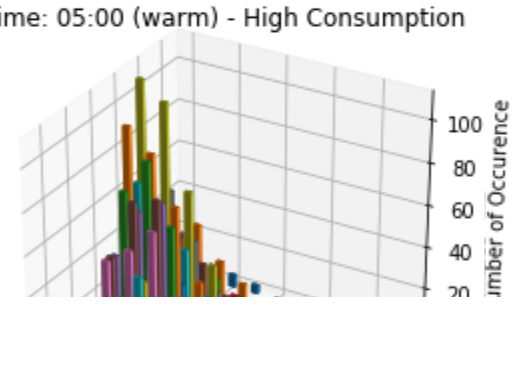
Time: 04:00 (warm) - Medium Consumption



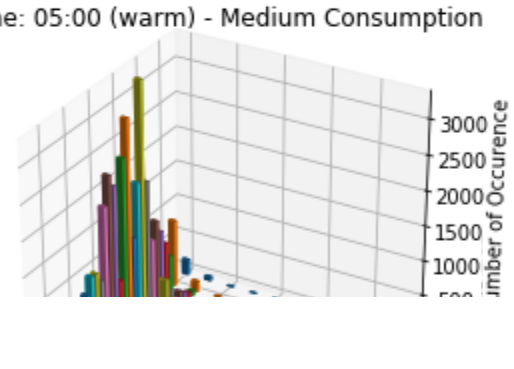
Time: 04:00 (warm) - Low Consumption



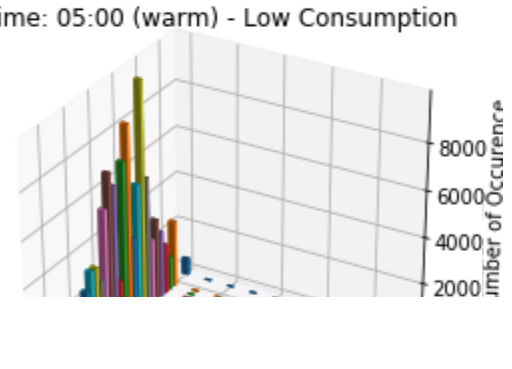
Time: 05:00 (warm) - High Consumption

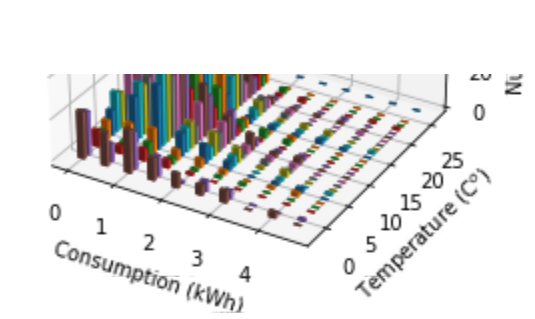


Time: 05:00 (warm) - Medium Consumption

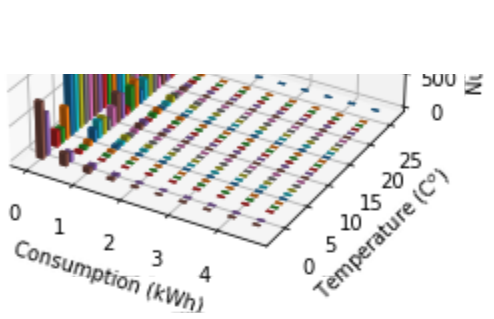


Time: 05:00 (warm) - Low Consumption

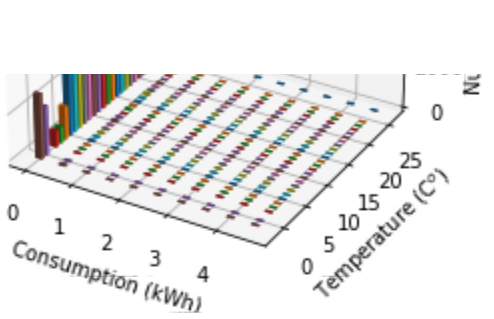




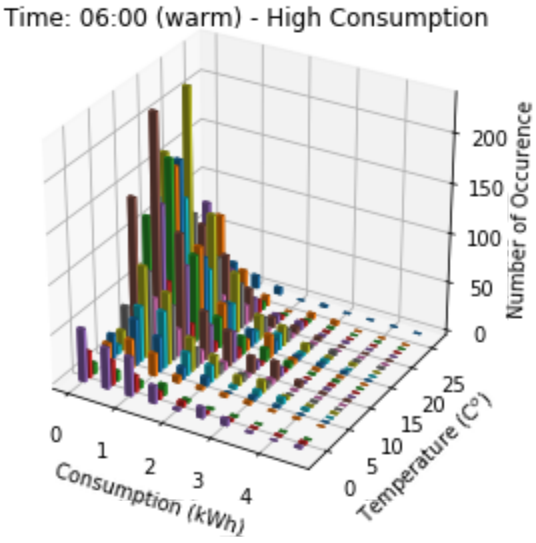
Time: 06:00 (warm) - High Consumption



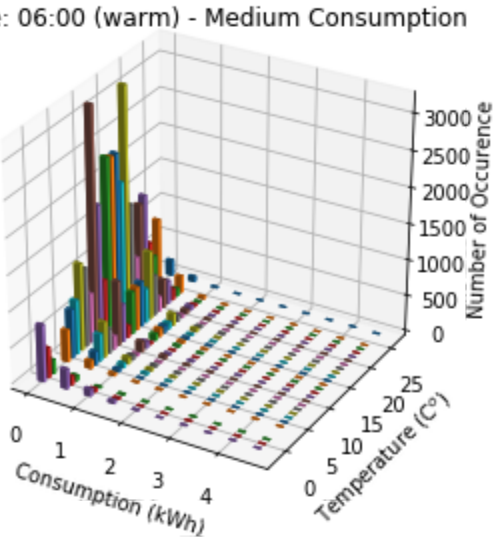
Time: 06:00 (warm) - Medium Consumption



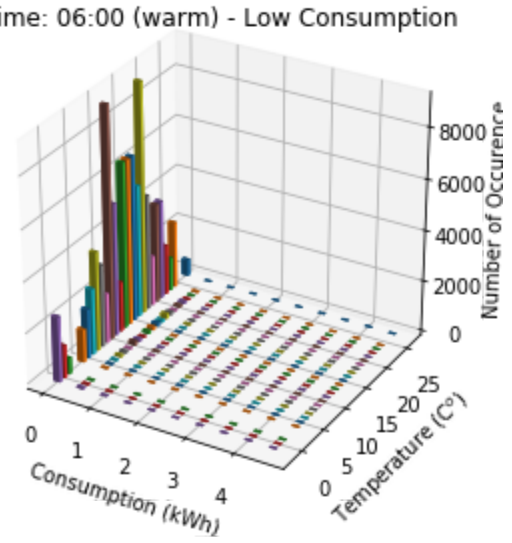
Time: 06:00 (warm) - Low Consumption



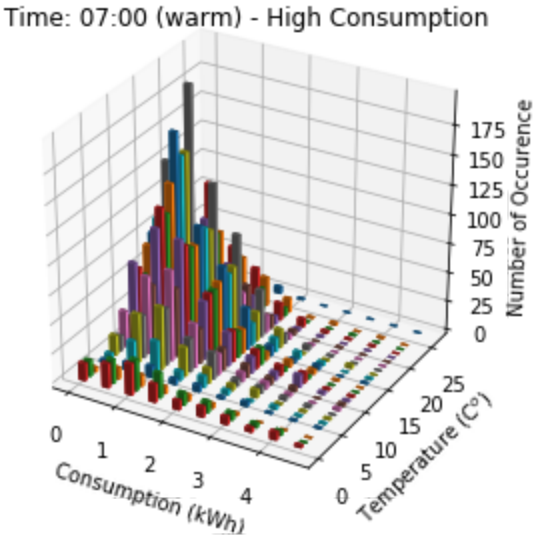
Time: 07:00 (warm) - High Consumption



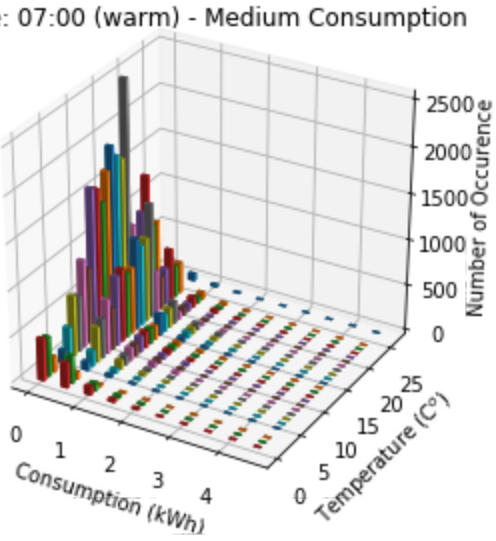
Time: 07:00 (warm) - Medium Consumption



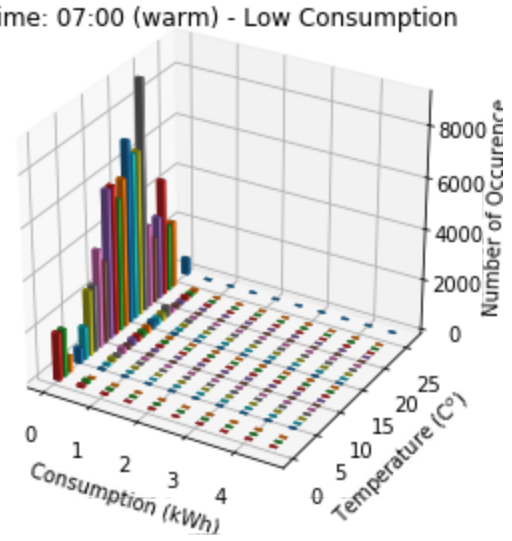
Time: 07:00 (warm) - Low Consumption



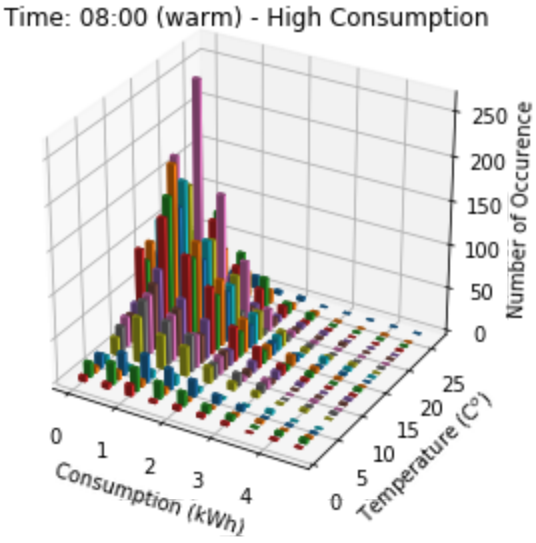
Time: 08:00 (warm) - High Consumption



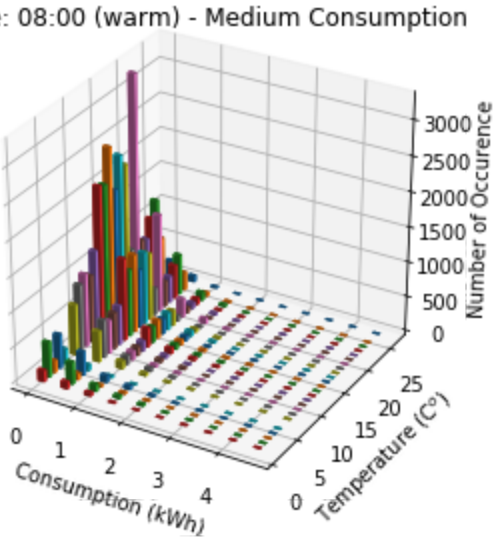
Time: 08:00 (warm) - Medium Consumption



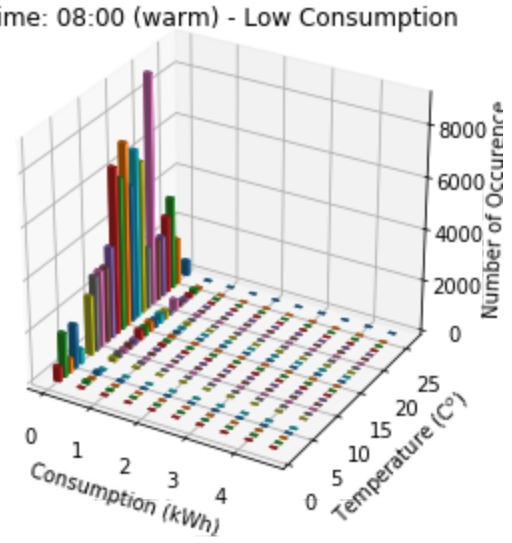
Time: 08:00 (warm) - Low Consumption



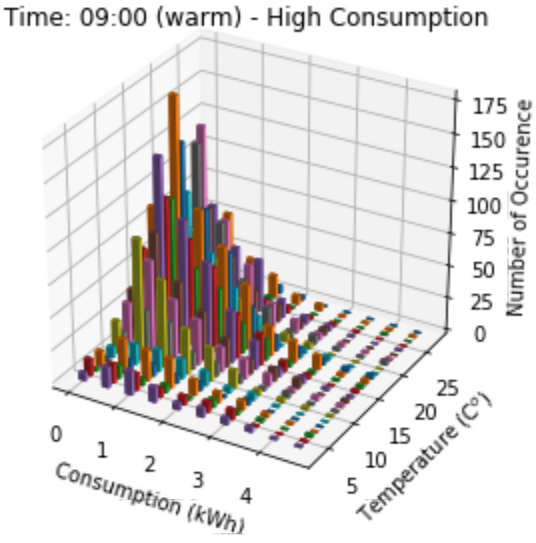
Time: 09:00 (warm) - High Consumption



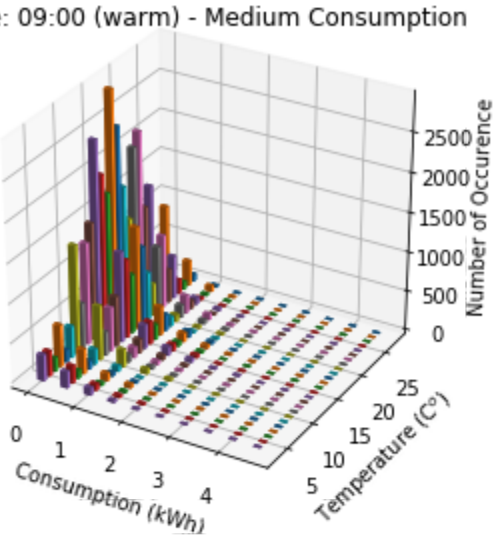
Time: 09:00 (warm) - Medium Consumption



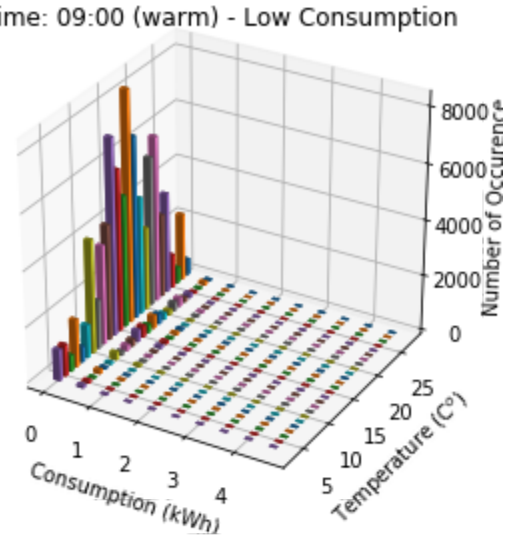
Time: 09:00 (warm) - Low Consumption



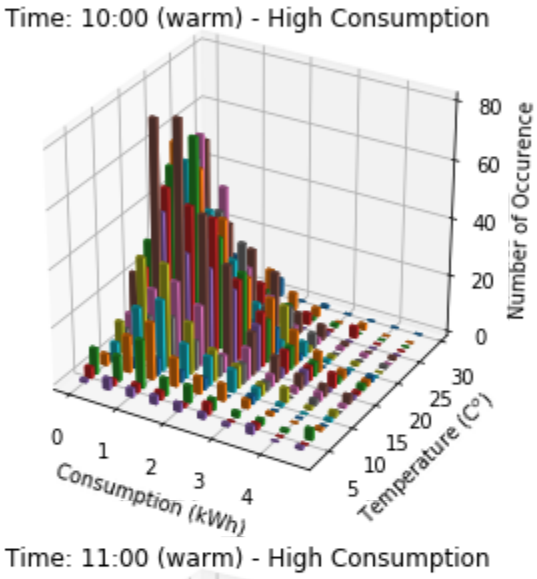
Time: 10:00 (warm) - High Consumption



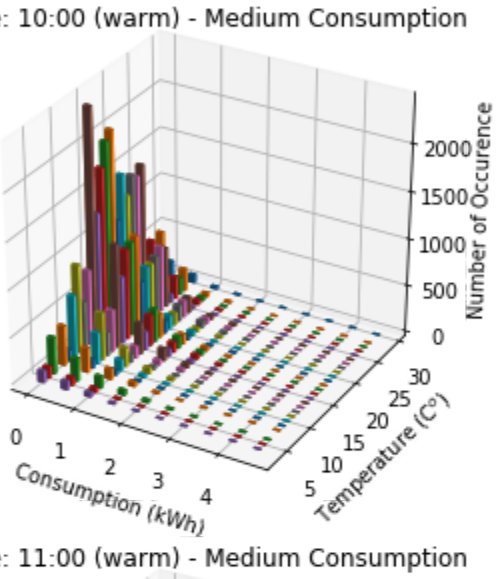
Time: 10:00 (warm) - Medium Consumption



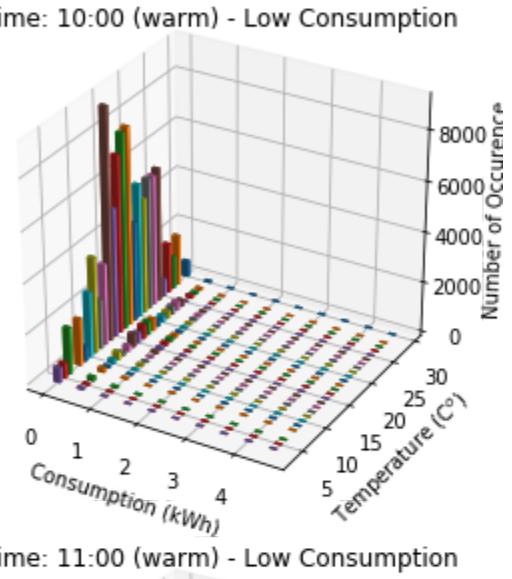
Time: 10:00 (warm) - Low Consumption



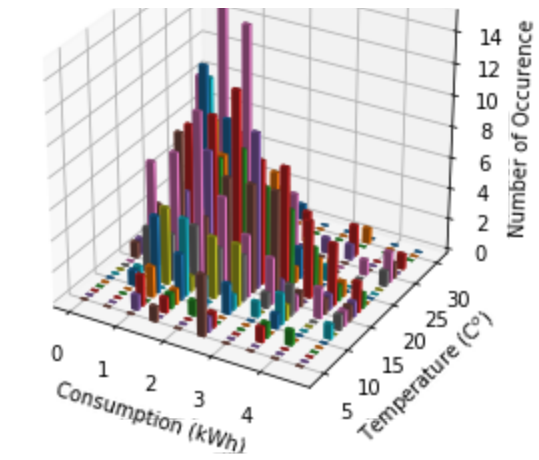
Time: 11:00 (warm) - High Consumption



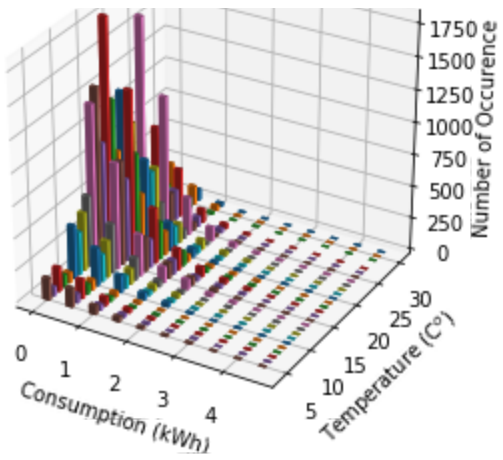
Time: 11:00 (warm) - Medium Consumption



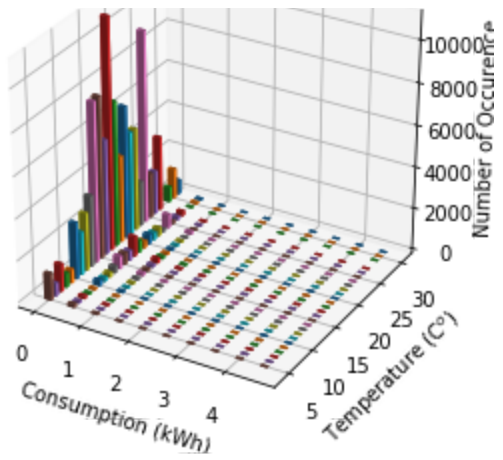
Time: 11:00 (warm) - Low Consumption



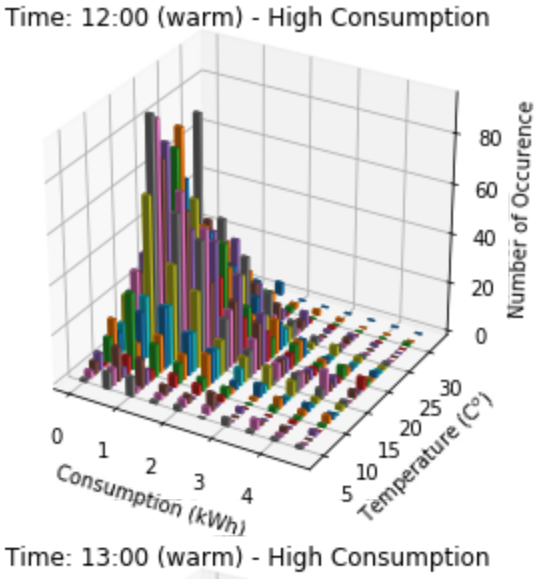
Time: 12:00 (warm) - High Consumption



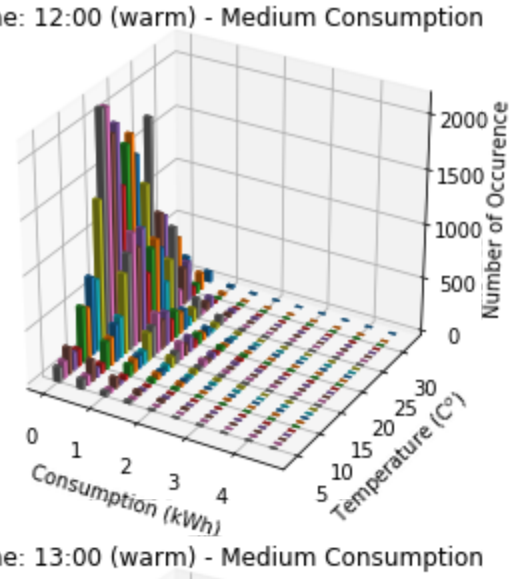
Time: 12:00 (warm) - Medium Consumption



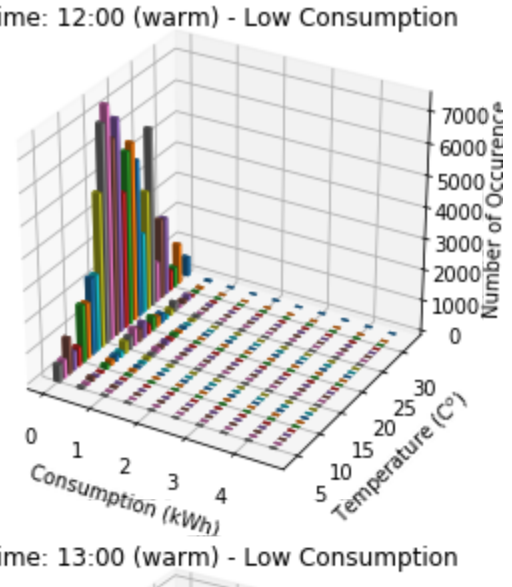
Time: 12:00 (warm) - Low Consumption



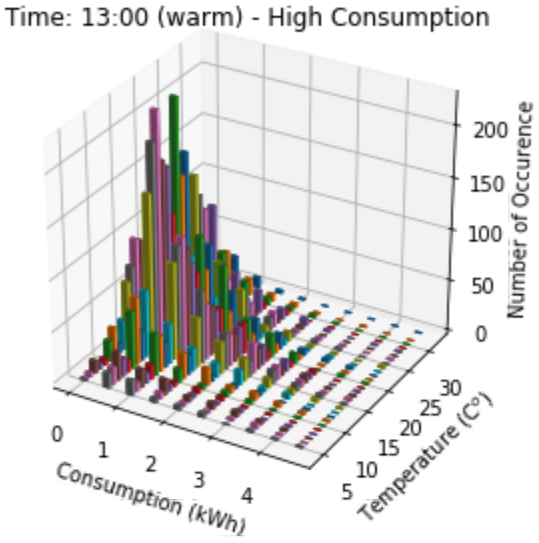
Time: 13:00 (warm) - High Consumption



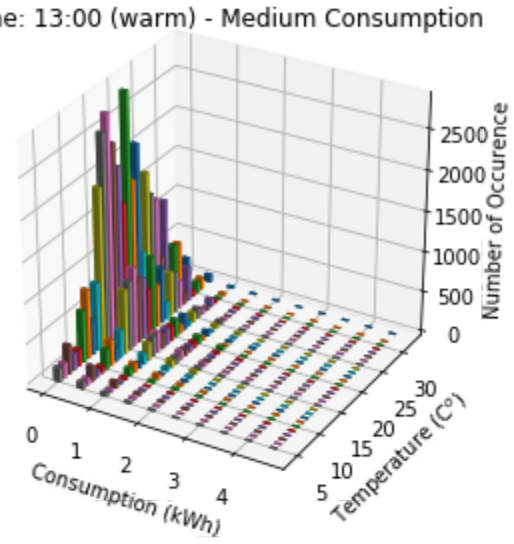
Time: 13:00 (warm) - Medium Consumption



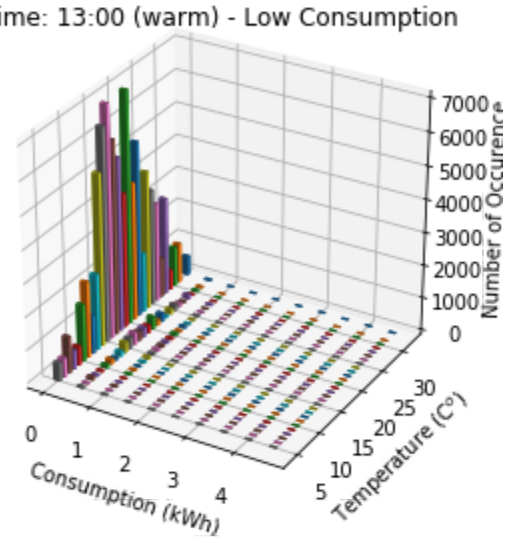
Time: 13:00 (warm) - Low Consumption



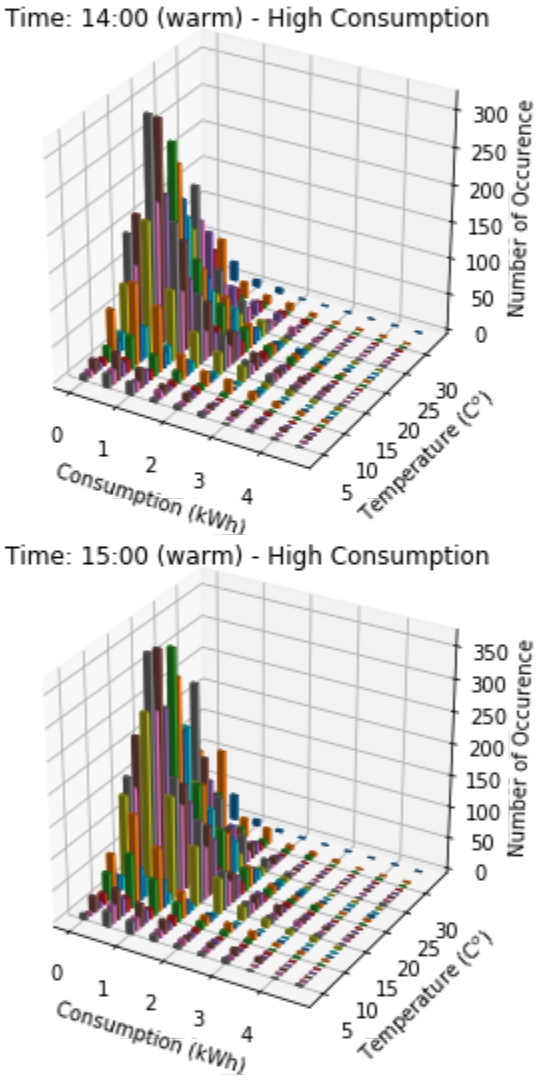
Time: 14:00 (warm) - High Consumption



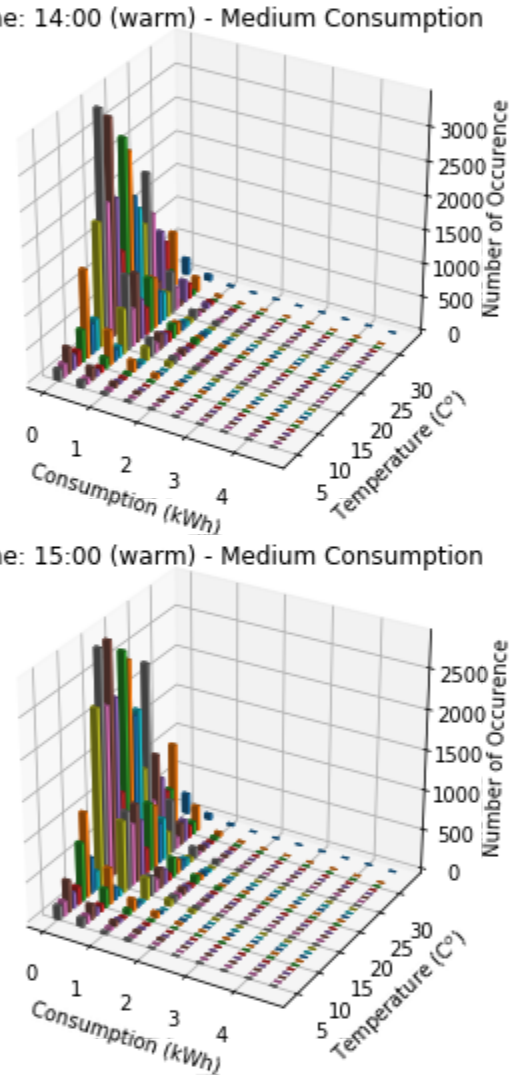
Time: 14:00 (warm) - Medium Consumption



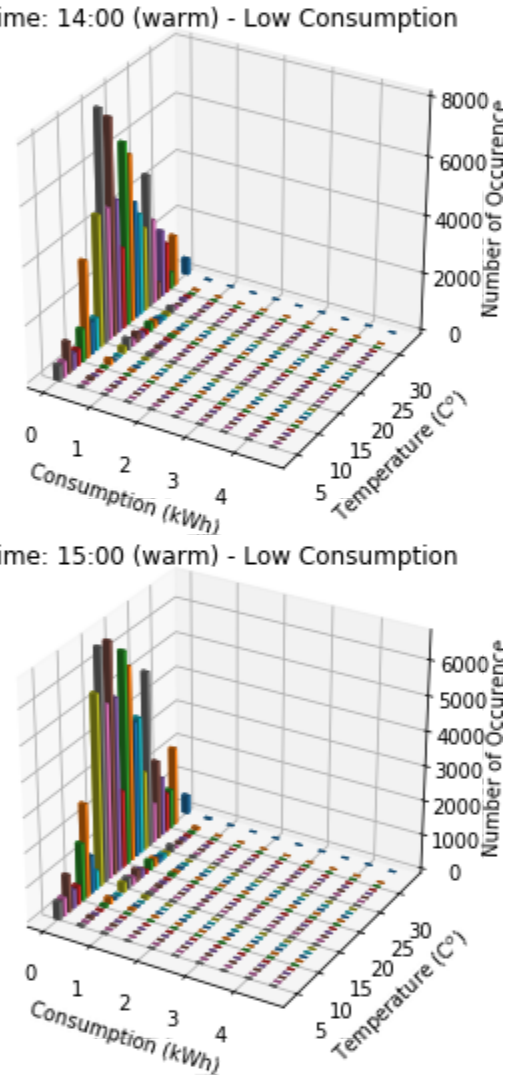
Time: 14:00 (warm) - Low Consumption



Time: 15:00 (warm) - High Consumption



Time: 15:00 (warm) - Medium Consumption

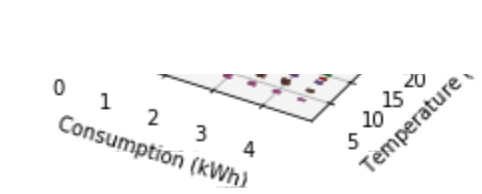


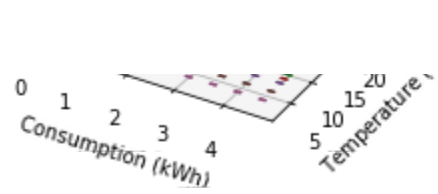
Time: 15:00 (warm) - Low Consumption

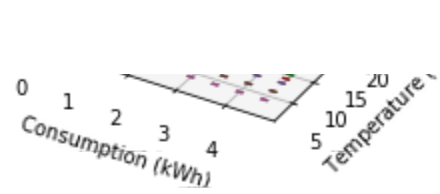
Time: 16:00 (warm) - High Consumption

Time: 16:00 (warm) - Medium Consumption

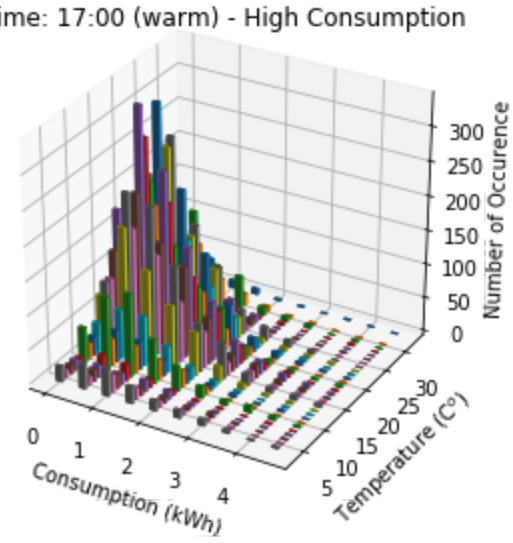
Time: 16:00 (warm) - Low Consumption



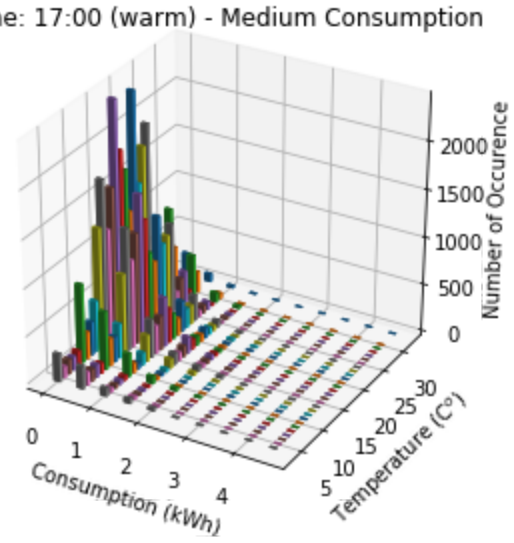




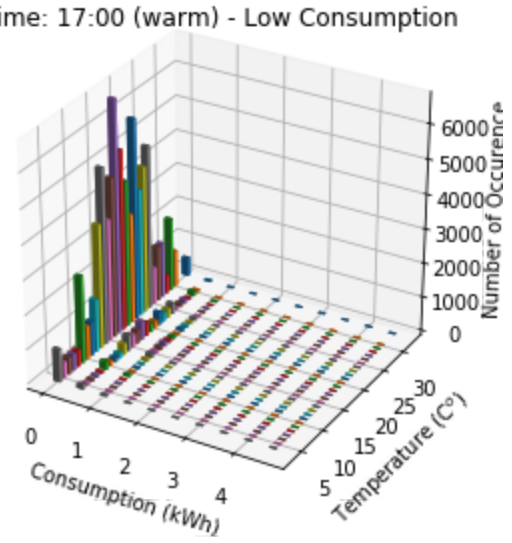
Time: 17:00 (warm) - High Consumption



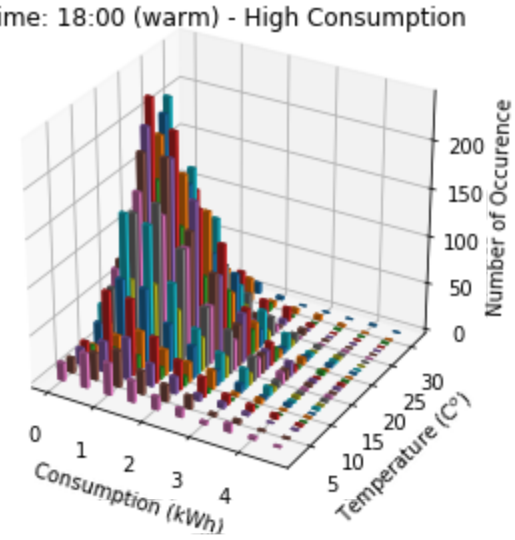
Time: 17:00 (warm) - Medium Consumption



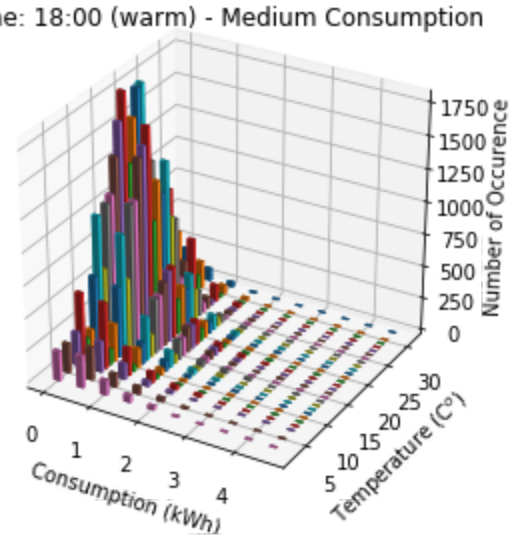
Time: 17:00 (warm) - Low Consumption



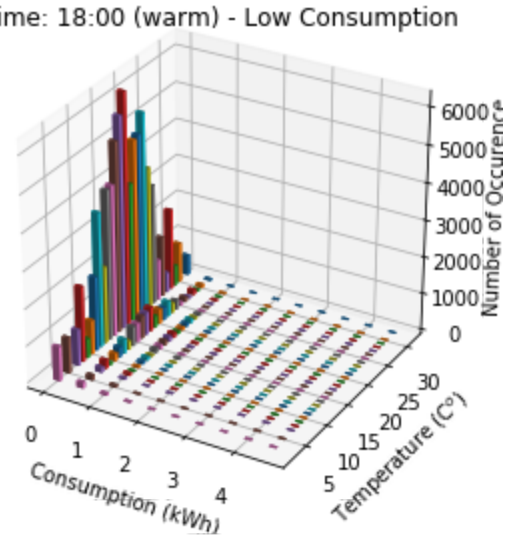
Time: 18:00 (warm) - High Consumption



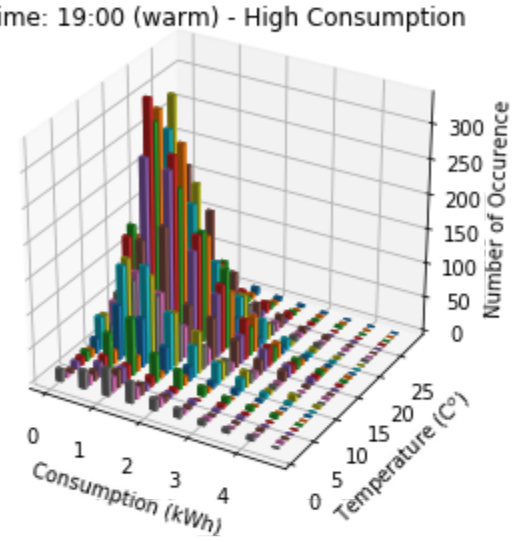
Time: 18:00 (warm) - Medium Consumption



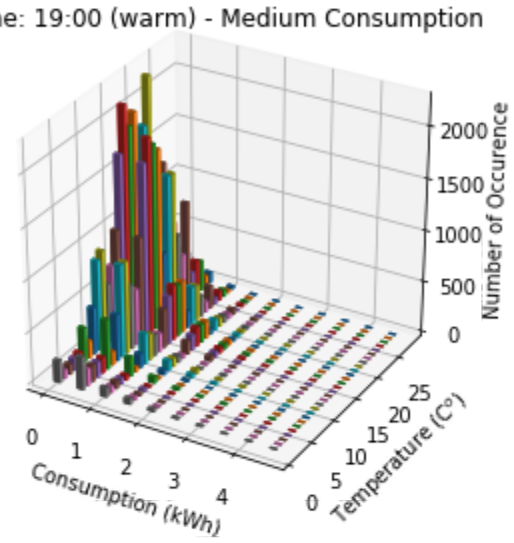
Time: 18:00 (warm) - Low Consumption



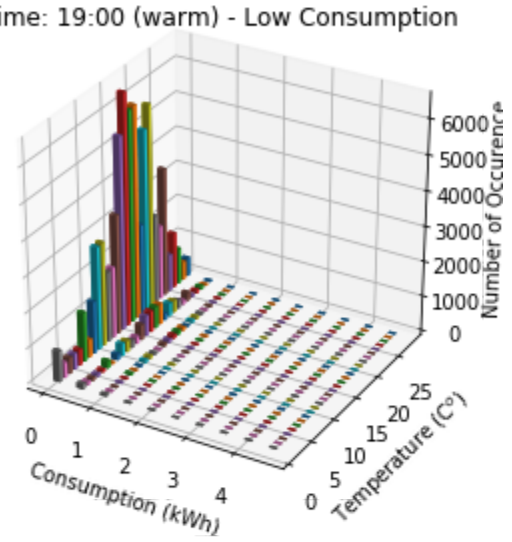
Time: 19:00 (warm) - High Consumption



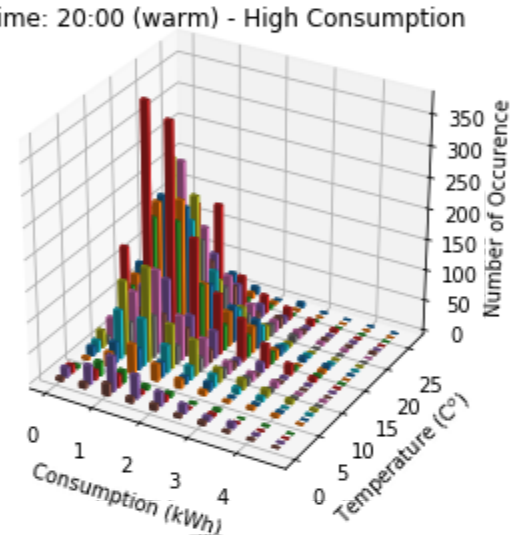
Time: 19:00 (warm) - Medium Consumption



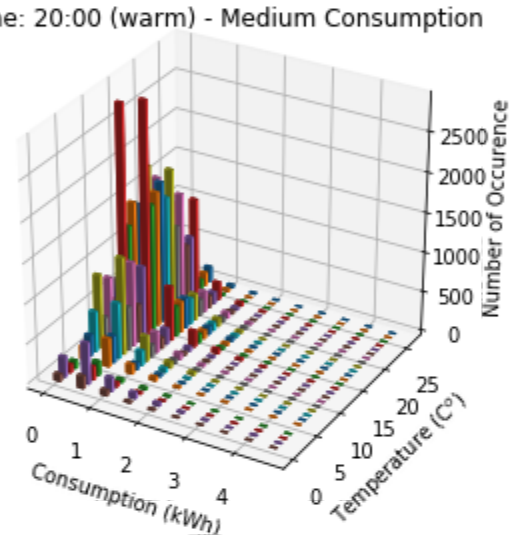
Time: 19:00 (warm) - Low Consumption



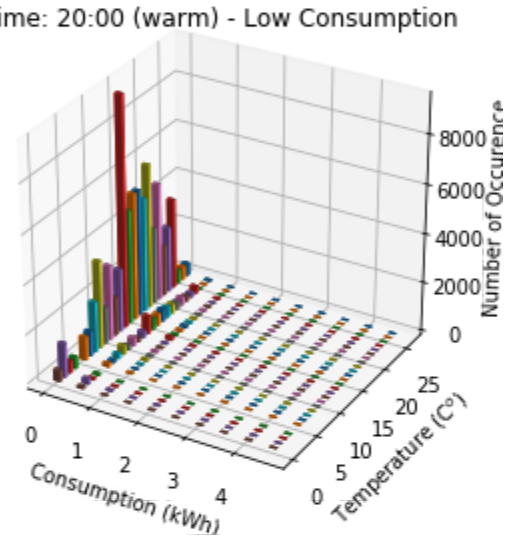
Time: 20:00 (warm) - High Consumption



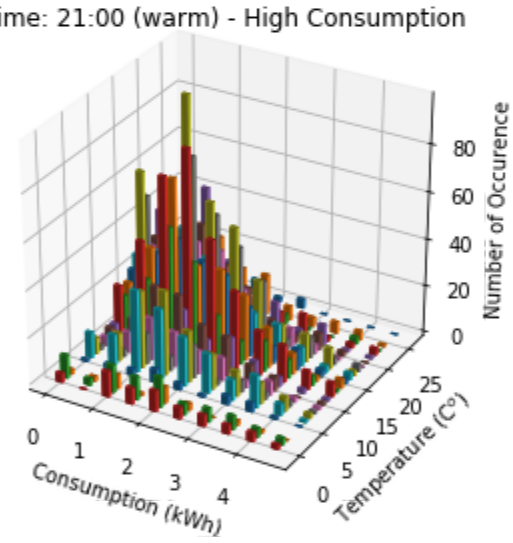
Time: 20:00 (warm) - Medium Consumption



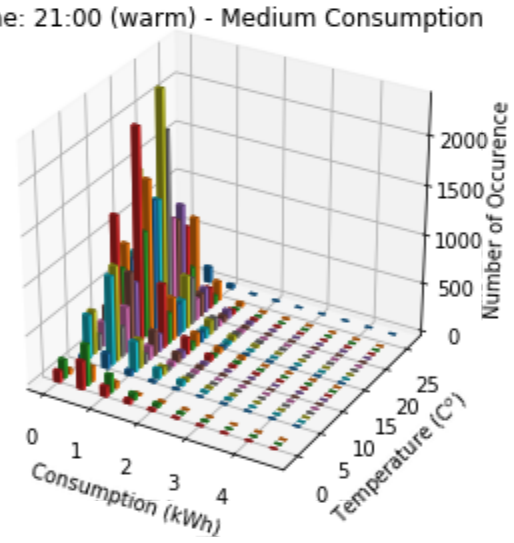
Time: 20:00 (warm) - Low Consumption



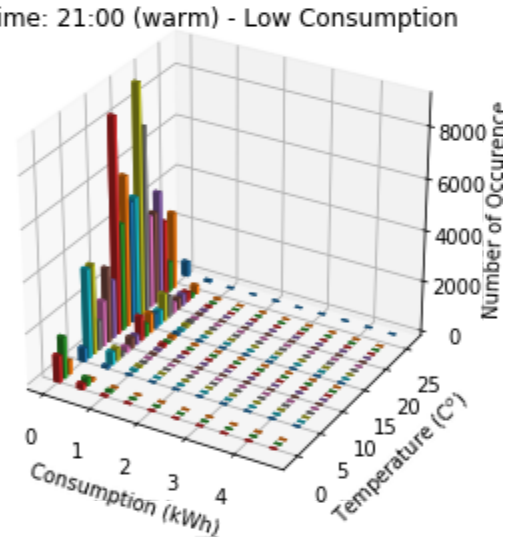
Time: 21:00 (warm) - High Consumption



Time: 21:00 (warm) - Medium Consumption



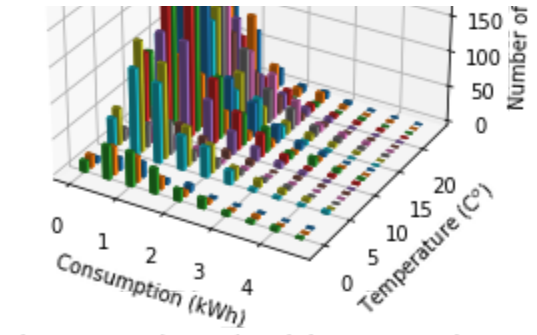
Time: 21:00 (warm) - Low Consumption



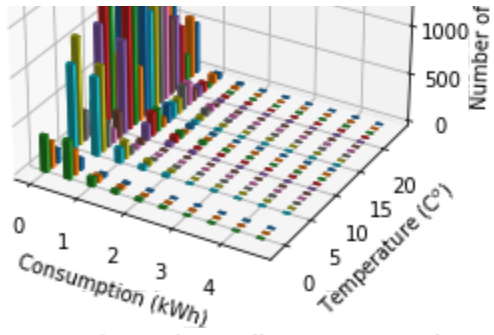
Time: 22:00 (warm) - High Consumption

Time: 22:00 (warm) - Medium Consumption

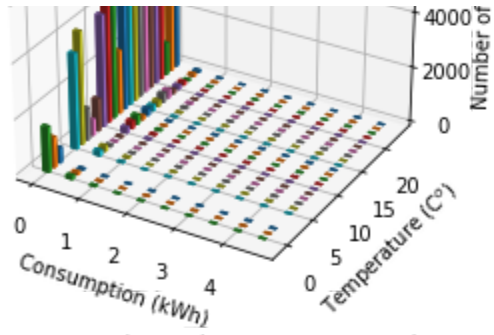
Time: 22:00 (warm) - Low Consumption



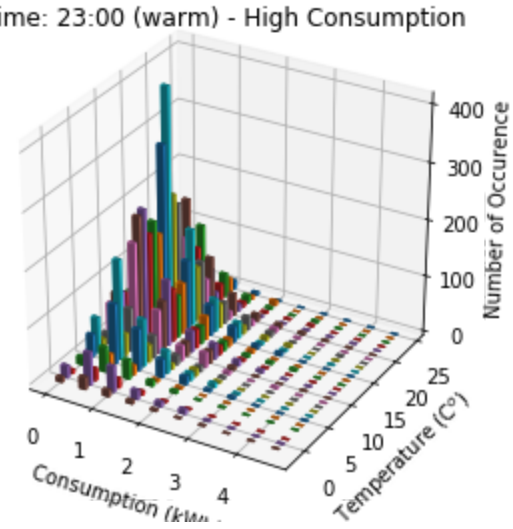
Time: 23:00 (warm) - High Consumption

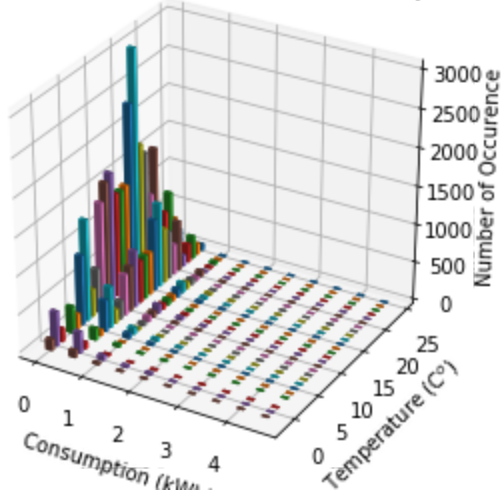


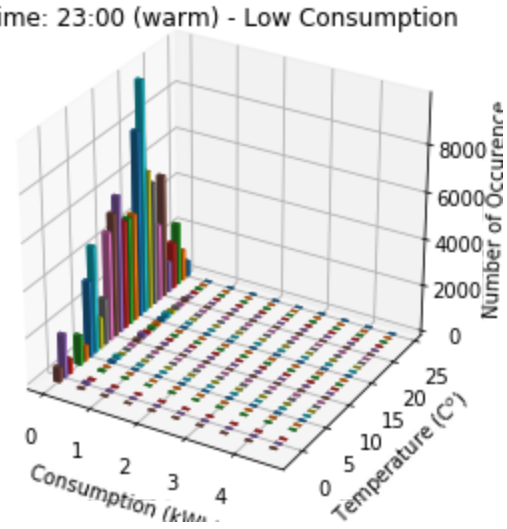
Time: 23:00 (warm) - Medium Consumption



Time: 23:00 (warm) - Low Consumption







Most of the above plots show that the highest demand of each group happens around 21-22 celcius degree

In [40]:
```
t
```

Out[40]:

	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	D0009	...	D1015	D1016	D1017	D1018	D1019	D1020
GMT																	
2013-01-01 00:00:00	1.447	0.429000	0.451	0.155	0.397	0.106	0.307	0.268	0.390	0.123	...	0.048	0.033	0.502	0.117	0.378	1.143
2013-01-02 00:00:00	0.358	0.153000	0.463	0.169	0.356	0.255	0.436	0.364	0.167	0.104	...	0.061	0.027	0.605	0.673	0.348	0.574
2013-01-03 00:00:00	0.142	0.385000	0.463	0.154	0.476	0.425	0.829	0.349	0.111	0.435	...	0.080	0.088	0.272	0.093	0.551	0.335
2013-01-05 00:00:00	0.230	0.137000	0.421	0.156	0.062	0.323	0.498	0.234	0.501	0.202	...	0.045	0.010	0.317	0.096	0.232	0.525
2013-01-06 00:00:00	0.303	0.229000	0.468	0.158	0.075	0.275	0.467	0.313	0.522	0.403	...	0.185	0.010	0.268	0.105	0.425	0.321
2013-01-09 00:00:00	0.086	0.330000	0.250	0.156	0.064	0.397	0.418	0.295	0.512	0.221	...	0.067	0.010	0.454	0.091	0.209	0.481
2013-01-12 00:00:00	0.218	0.317000	0.410	0.149	0.055	0.451	0.410	0.234	0.428	0.285	...	0.083	0.010	0.369	0.089	0.569	0.519
2013-01-14 00:00:00	0.220	0.120000	0.288	0.147	0.055	0.378	0.540	0.185	0.545	0.297	...	0.066	0.611	0.459	0.079	1.062	0.483
2013-01-15 00:00:00	0.218	0.222000	0.292	0.149	0.055	0.385	0.336	0.205	0.573	0.239	...	0.030	0.074	0.394	0.102	1.051	0.356
2013-01-18 00:00:00	0.095	0.193000	0.504	0.144	0.175	0.352	0.447	0.183	0.674	0.185	...	0.042	0.412	0.584	0.100	0.940	0.252
2013-01-22 00:00:00	0.099	0.155000	0.404	0.142	0.057	0.463	0.068	0.201	0.479	0.270	...	0.067	0.327	0.486	0.113	0.880	0.206
2013-01-23 00:00:00	0.100	0.106000	0.380	0.141	0.058	0.394	0.257	0.304	0.439	0.256	...	0.071	0.024	0.383	0.109	1.011	0.239
2013-01-24 00:00:00	0.103	0.215000	0.407	0.144	0.122	0.386	0.155	0.302	0.376	0.378	...	0.040	0.704	0.430	0.150	0.690	0.229
2013-01-26 00:00:00	0.152	0.268000	1.102	0.148	0.059	0.323	0.403	0.169	0.841	0.317	...	0.058	0.019	0.433	0.144	0.956	0.699
2013-01-27 00:00:00	0.076	0.449000	0.291	0.151	0.208	0.307	0.253	0.256	0.538	0.466	...	0.054	0.031	0.269	0.119	0.660	0.403
2013-01-31 00:00:00	0.098	0.163000	0.266	0.152	0.237	0.434	0.125	0.192	0.124	0.224	...	0.029	0.032	0.461	0.179	0.970	0.383
2013-02-01 00:00:00	0.088	0.115000	0.264	0.154	0.291	0.267	0.295	0.150	0.167	0.266	...	0.045	0.537	0.431	0.873	0.611	0.230
2013-02-02 00:00:00	0.227	0.371000	0.327	0.160	0.057	0.369	0.263	0.134	0.420	0.404	...	0.049	0.055	0.240	0.114	0.910	0.361
2013-02-04 00:00:00	0.105	0.099000	0.327	0.160	0.052	0.182	0.816	0.234	0.584	0.247	...	0.044	0.054	0.429	0.107	0.734	0.507
2013-02-06 00:00:00	0.100	0.088000	0.285	0.155	0.062	0.406	0.472	0.263	0.428	0.172	...	0.029	0.034	0.208	0.106	0.491	0.255

	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	D0009	...	D1015	D1016	D1017	D1018	D1019	D1020
GMT																	
2013-02-12 00:00:00	0.107	0.109000	0.354	0.149	0.056	0.337	0.743	0.238	0.368	0.423	...	0.058	0.021	0.269	0.127	0.804	0.165
2013-02-13 00:00:00	0.226	0.167000	0.406	0.151	0.070	0.400	0.682	0.291	0.501	0.246	...	0.036	0.043	0.646	0.327	0.827	0.249
2013-02-14 00:00:00	0.102	0.246000	0.350	0.155	0.056	0.375	0.410	0.252	0.106	0.256	...	0.061	0.515	0.488	0.103	0.638	0.198
2013-02-16 00:00:00	0.117	0.168000	0.411	0.157	0.249	0.350	0.155	0.278	0.407	0.306	...	0.082	0.370	0.245	0.115	0.680	0.279
2013-02-19 00:00:00	0.159	0.255000	0.481	0.154	0.057	0.453	0.410	0.245	0.428	0.267	...	0.065	0.065	0.376	0.114	0.802	0.158
2013-02-23 00:00:00	0.135	0.113000	0.402	0.145	0.346	0.456	0.667	0.294	0.419	0.338	...	0.037	0.018	0.473	0.139	0.772	0.497
2013-02-24 00:00:00	0.193	0.157000	0.404	0.196	0.100	0.478	0.595	0.273	0.396	0.475	...	0.063	0.059	0.450	0.104	0.663	0.500
2013-02-25 00:00:00	0.111	0.192000	1.046	0.149	0.073	0.161	0.885	0.258	0.536	0.221	...	0.060	0.046	0.246	0.112	1.015	0.211
2013-03-03 00:00:00	0.145	0.293000	0.277	0.201	0.076	0.432	1.003	0.179	0.389	0.223	...	0.129	0.027	0.223	1.074	0.142	0.363
2013-03-04 00:00:00	0.165	0.141000	0.283	0.153	0.058	0.328	0.524	0.209	0.383	0.214	...	0.077	0.014	0.228	0.679	0.540	0.285
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2013-11-12 00:00:00	0.204	0.109000	1.029	0.099	0.028	0.322	0.161	0.241	0.584	0.326	...	0.088	0.006	0.190	0.042	0.423	0.304
2013-11-13 00:00:00	0.121	0.136000	0.243	0.100	0.038	0.394	0.164	0.196	0.603	0.340	...	0.068	0.031	0.196	0.120	0.742	0.462
2013-11-15 00:00:00	0.091	0.206000	0.300	0.097	0.040	0.363	0.155	0.232	0.108	0.294	...	0.054	0.028	0.182	0.746	0.518	0.371
2013-11-16 00:00:00	0.183	0.194000	0.258	0.098	0.119	0.314	0.138	0.161	0.110	0.295	...	0.084	0.050	0.205	0.206	0.598	0.138
2013-11-17 00:00:00	0.089	0.323000	0.174	0.102	0.070	0.396	0.189	0.285	0.664	0.364	...	0.074	0.035	0.265	0.117	0.745	0.534
2013-11-18 00:00:00	0.091	0.223000	0.213	0.101	0.063	0.424	0.380	0.255	0.111	0.451	...	0.058	0.005	0.251	0.118	0.081	0.439
2013-11-19 00:00:00	0.120	0.090000	0.214	0.102	0.079	0.250	0.108	0.292	0.316	0.356	...	0.058	0.049	0.189	0.117	0.139	0.252
2013-11-20 00:00:00	0.181	0.134000	0.286	0.098	0.040	0.445	0.097	0.134	0.825	0.256	...	0.068	0.035	0.241	0.124	0.361	0.156
2013-11-22 00:00:00	0.190	0.245000	0.283	0.098	0.103	0.503	0.871	0.181	0.255	0.261	...	0.078	0.033	0.207	0.109	0.873	0.603
2013-11-23 00:00:00	0.510	0.286000	0.582	0.099	0.067	0.332	0.227	0.138	0.403	0.343	...	0.035	0.034	0.205	0.113	0.483	0.119

	D0000	D0001	D0002	D0003	D0004	D0005	D0006	D0007	D0008	D0009	...	D1015	D1016	D1017	D1018	D1019	D1020
GMT																	
2013-11-24 00:00:00	0.469	0.294000	0.243	0.100	0.025	0.434	0.086	0.039	0.600	0.806	...	0.059	0.046	0.211	0.116	0.640	1.369
2013-11-25 00:00:00	0.083	0.168000	0.538	0.100	0.013	0.185	0.110	0.043	0.392	0.454	...	0.065	0.001	0.196	0.125	0.721	0.546
2013-12-02 00:00:00	0.106	0.358000	0.427	0.100	0.085	0.228	0.384	0.245	0.312	0.293	...	0.048	0.045	0.212	0.203	0.324	0.350
2013-12-03 00:00:00	0.095	0.152000	0.726	0.100	0.017	0.371	0.331	0.217	0.368	0.268	...	0.039	0.019	0.199	0.154	0.417	0.546
2013-12-05 00:00:00	0.103	0.160000	1.086	0.099	0.024	0.398	0.595	0.305	0.491	0.268	...	0.063	0.062	0.285	0.306	0.392	0.602
2013-12-06 00:00:00	0.094	0.163000	0.324	0.098	0.082	0.125	0.470	0.082	0.518	0.267	...	0.053	0.011	0.227	0.137	0.687	0.617
2013-12-09 00:00:00	0.092	0.193829	0.692	0.101	0.056	0.402	0.446	0.199	0.372	0.245	...	0.055	0.000	0.206	0.111	0.266	0.439
2013-12-10 00:00:00	0.465	0.148000	0.240	0.101	0.013	0.349	0.485	0.272	0.469	0.324	...	0.061	0.016	0.290	0.115	0.607	0.502
2013-12-11 00:00:00	0.116	0.212000	0.324	0.097	0.016	0.480	0.429	0.199	0.651	0.271	...	0.082	0.033	0.203	0.150	0.972	0.255
2013-12-14 00:00:00	0.477	0.177000	0.216	0.099	0.035	0.557	0.265	0.312	1.049	0.255	...	0.080	0.035	0.203	0.108	0.242	0.237
2013-12-17 00:00:00	0.172	0.215000	0.206	0.104	0.022	0.424	0.625	0.258	0.845	0.274	...	0.055	0.028	0.228	0.125	0.509	0.300
2013-12-18 00:00:00	0.091	0.292000	0.252	0.167	0.022	0.147	0.208	0.213	0.900	0.264	...	0.069	0.000	0.259	0.125	0.462	0.495
2013-12-21 00:00:00	0.292	0.302000	0.169	0.100	0.035	0.464	0.428	0.241	0.742	0.254	...	0.073	0.015	0.230	0.270	0.330	0.400
2013-12-22 00:00:00	0.289	0.212000	0.455	0.099	0.124	0.285	0.270	0.259	0.461	0.360	...	0.070	0.001	0.319	0.107	0.300	0.614
2013-12-23 00:00:00	0.473	0.080000	0.782	0.100	0.103	0.453	0.173	0.137	0.242	0.647	...	0.055	0.006	0.179	0.787	0.449	0.480
2013-12-24 00:00:00	0.326	0.076000	0.302	0.099	0.017	0.507	0.257	0.315	0.857	0.260	...	0.054	0.000	0.244	0.114	0.769	0.591
2013-12-25 00:00:00	0.632	0.371000	0.636	0.099	0.283	0.265	0.434	0.216	0.472	0.678	...	0.034	0.020	0.259	0.113	0.601	0.535
2013-12-27 00:00:00	0.318	0.397000	0.279	0.099	0.050	0.342	0.550	0.244	0.473	0.103	...	0.050	0.035	0.247	0.122	0.441	0.647
2013-12-28 00:00:00	0.298	0.110000	0.284	0.099	0.150	0.359	0.793	0.242	0.647	1.235	...	0.053	0.026	0.183	0.235	0.451	2.069
2013-12-31 00:00:00	0.335	0.107000	0.415	0.101	0.295	0.338	0.624	0.139	0.410	0.550	...	0.078	0.033	0.194	0.120	0.742	0.341

83 rows × 1025 columns

In [44]:
```
i
```

Out[44]:
```
23
```

In [45]:
```
pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('23:00:00')]['GMT']).dt.dayofweek
```

Out[45]:
```
23      1
47      2
71      3
119     5
143     6
215     2
287     5
335     0
359     1
431     4
527     1
551     2
575     3
623     5
647     6
743     3
767     4
791     5
839     0
887     2
1031    1
1055    2
1079    3
1127    5
1199    1
1295    5
1319    6
1343    0
1487    6
1511    0
       ..
7583    1
7607    2
7655    4
7679    5
7703    6
7727    0
7751    1
7775    2
7823    4
7847    5
7871    6
7895    0
8063    0
8087    1
8135    3
8159    4
8231    0
8255    1
8279    2
8351    5
8423    1
8447    2
8519    5
8543    6
8567    0
8591    1
8615    2
8663    4
8687    5
8759    1
Name: GMT, Length: 83, dtype: int64
```

In [47]:
```
t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('23:00:00')]['GM
T']).dt.dayofweek)
t2
```

Out[47]:

	GMT	D0000	D0001	D0003	D0004	D0005	D0007	D0008	D0009	D0010	...	D1015	D1016	D1017	D1018	D1020	D1021	D102
0	2013-01-01 23:00:00	0.323	0.269	0.241	0.349	0.293	0.495	0.533	0.110	0.019	...	0.031	0.121	0.656	0.294	0.715	0.102	0.337
1	2013-01-02 23:00:00	0.232	0.452	0.153	0.500	0.458	0.357	0.268	0.327	0.028	...	0.036	0.166	0.394	0.087	0.400	0.142	0.348
2	2013-01-03 23:00:00	0.315	0.375	0.156	0.502	0.352	0.402	0.457	0.655	0.028	...	0.047	0.072	0.420	0.085	1.110	0.128	0.373
3	2013-01-05 23:00:00	0.064	0.403	0.206	0.263	0.288	0.457	0.463	0.624	0.024	...	0.080	0.136	0.368	0.083	0.455	0.502	0.357
4	2013-01-06 23:00:00	0.297	0.333	0.157	0.233	0.495	0.194	0.376	0.592	0.024	...	0.072	0.204	0.463	0.083	0.352	0.152	0.365
5	2013-01-09 23:00:00	0.164	0.245	0.153	0.057	0.326	0.220	0.437	0.230	0.020	...	0.033	0.048	0.439	0.091	0.365	0.110	0.351
6	2013-01-12 23:00:00	0.262	0.440	0.168	0.255	0.157	0.259	0.424	0.405	0.027	...	0.045	0.521	0.482	0.081	0.685	0.127	0.363
7	2013-01-14 23:00:00	1.416	0.479	0.147	0.064	0.384	0.239	0.501	0.291	0.018	...	0.107	0.271	0.418	0.086	0.355	0.136	0.350
8	2013-01-15 23:00:00	0.098	0.305	0.145	0.056	0.407	0.229	0.396	0.218	0.025	...	0.082	0.074	0.598	0.080	0.554	0.289	0.357
9	2013-01-18 23:00:00	0.299	0.367	0.141	0.265	0.459	0.219	0.447	0.250	0.022	...	0.057	0.780	0.460	0.100	0.496	0.186	0.341
10	2013-01-22 23:00:00	0.103	0.398	0.141	0.229	0.491	0.381	0.617	0.265	0.026	...	0.056	0.742	0.419	0.171	0.381	0.135	0.418
11	2013-01-23 23:00:00	0.126	0.418	0.143	0.224	0.380	0.694	0.425	0.535	0.030	...	0.067	0.054	0.451	0.364	0.253	0.113	0.348
12	2013-01-24 23:00:00	0.103	0.458	0.152	0.114	0.399	0.249	0.536	0.330	0.015	...	0.046	0.702	0.514	0.116	0.320	0.163	0.423
13	2013-01-26 23:00:00	0.215	0.303	0.208	0.303	0.389	0.257	0.361	0.499	0.021	...	0.055	0.635	0.297	0.118	0.559	0.119	0.432
14	2013-01-27 23:00:00	0.156	0.257	0.144	0.128	0.168	0.145	0.447	0.508	0.023	...	0.031	0.053	0.453	1.516	0.573	0.124	0.355
15	2013-01-31 23:00:00	0.089	0.302	0.154	0.506	0.362	0.143	0.568	0.449	0.023	...	0.072	0.247	0.478	0.572	0.201	0.213	0.366
16	2013-02-01 23:00:00	0.268	0.397	0.160	0.186	0.347	0.213	0.434	0.395	0.031	...	0.063	0.660	0.414	0.120	0.380	0.166	0.375
17	2013-02-02 23:00:00	0.285	0.483	0.158	0.230	0.579	0.284	0.539	1.603	0.030	...	0.062	0.130	0.604	1.494	0.377	0.217	0.340
18	2013-02-04 23:00:00	0.166	0.242	0.156	0.252	0.392	0.290	0.472	0.291	0.020	...	0.050	0.058	0.568	0.116	0.437	0.169	0.366
19	2013-02-06 23:00:00	0.158	0.384	0.154	0.100	0.365	0.137	0.518	0.234	0.018	...	0.078	0.435	0.445	0.119	0.842	0.163	0.368
20	2013-02-12 23:00:00	0.206	0.320	0.150	0.070	0.424	0.258	0.387	0.943	0.018	...	0.091	0.666	0.636	0.833	0.575	0.139	0.365

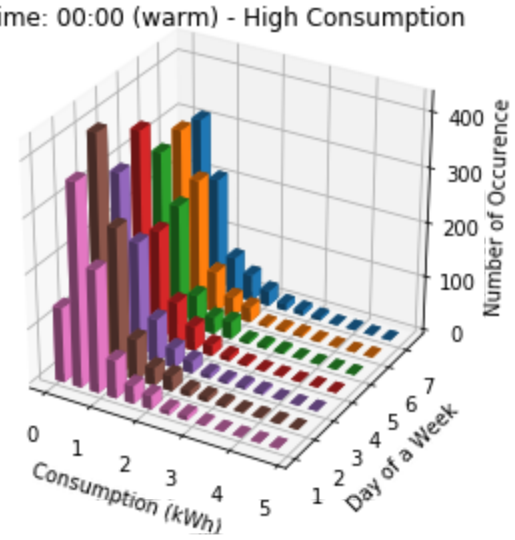
	GMT	D0000	D0001	D0003	D0004	D0005	D0007	D0008	D0009	D0010	...	D1015	D1016	D1017	D1018	D1020	D1021	D102
21	2013-02-13 23:00:00	0.119	0.196	0.154	0.115	0.354	0.292	0.238	0.284	0.025	...	0.030	0.728	0.479	0.117	0.183	0.214	0.355
22	2013-02-14 23:00:00	0.586	0.248	0.154	0.061	0.147	0.263	0.395	0.375	0.026	...	0.075	0.475	0.537	0.109	0.464	0.584	0.368
23	2013-02-16 23:00:00	0.261	0.077	0.245	0.218	0.144	0.326	0.368	0.421	0.022	...	0.068	0.529	0.413	0.110	0.633	0.425	0.353
24	2013-02-19 23:00:00	0.224	0.241	0.158	0.276	0.368	0.397	0.401	0.513	0.022	...	0.056	0.059	0.526	0.422	0.661	0.152	0.350
25	2013-02-23 23:00:00	0.330	0.298	0.236	0.355	0.470	0.305	0.408	0.499	0.019	...	0.060	0.060	0.579	0.088	0.698	0.122	0.329
26	2013-02-24 23:00:00	0.099	0.292	0.150	0.223	0.184	0.250	0.443	0.237	0.017	...	0.036	0.058	0.512	0.113	0.173	0.131	0.338
27	2013-02-25 23:00:00	0.292	0.293	0.148	0.082	0.448	0.270	0.458	0.349	0.016	...	0.056	0.058	0.401	0.114	0.389	0.123	0.444
28	2013-03-03 23:00:00	0.299	0.283	0.153	0.061	0.162	0.238	0.386	0.205	0.000	...	0.042	0.054	0.237	1.215	0.210	0.081	0.386
29	2013-03-04 23:00:00	0.149	0.245	0.154	0.151	0.392	0.239	0.387	0.188	0.002	...	0.042	0.059	0.210	1.040	0.265	0.110	0.293
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
53	2013-11-12 23:00:00	0.189	0.231	0.098	0.135	0.428	0.261	0.249	0.840	0.020	...	0.060	0.048	0.249	0.116	0.500	0.163	0.321
54	2013-11-13 23:00:00	0.416	0.156	0.095	0.189	0.347	0.239	0.129	0.284	0.018	...	0.055	0.048	0.199	0.143	0.368	0.228	0.324
55	2013-11-15 23:00:00	0.215	0.281	0.099	0.039	0.361	0.158	0.337	0.287	0.022	...	0.055	0.145	0.289	1.459	0.161	0.800	0.357
56	2013-11-16 23:00:00	0.050	0.541	0.149	0.045	0.463	0.324	0.344	0.543	0.020	...	0.044	0.193	0.459	0.113	0.483	0.179	0.319
57	2013-11-17 23:00:00	0.255	0.422	0.100	0.168	0.450	0.245	0.160	0.230	0.022	...	0.095	0.220	0.244	0.118	0.489	0.488	0.340
58	2013-11-18 23:00:00	0.299	0.348	0.102	0.096	0.302	0.292	0.573	0.365	0.020	...	0.055	0.202	0.256	0.246	0.417	0.134	0.346
59	2013-11-19 23:00:00	0.405	0.131	0.150	0.137	0.441	0.212	0.908	0.254	0.014	...	0.035	0.047	0.258	0.127	0.150	0.105	0.326
60	2013-11-20 23:00:00	0.354	0.352	0.098	0.126	0.398	0.234	0.402	0.281	0.025	...	0.072	0.074	0.302	0.179	0.255	0.306	0.344
61	2013-11-22 23:00:00	1.767	0.256	0.100	0.095	0.354	0.161	0.438	0.690	0.014	...	0.088	0.149	0.170	0.111	0.217	0.240	0.344
62	2013-11-23 23:00:00	0.392	0.510	0.147	0.142	0.502	0.176	0.370	1.092	0.055	...	0.076	0.152	0.187	0.111	0.742	0.159	0.351
63	2013-11-24 23:00:00	0.121	0.380	0.101	0.119	0.267	0.069	0.451	0.482	0.022	...	0.061	0.079	0.248	0.114	0.452	0.161	0.352
64	2013-11-25 23:00:00	0.091	0.307	0.100	0.108	0.420	0.292	0.525	0.253	0.023	...	0.054	0.047	0.288	0.118	0.383	0.126	0.347

	GMT	D0000	D0001	D0003	D0004	D0005	D0007	D0008	D0009	D0010	...	D1015	D1016	D1017	D1018	D1020	D1021	D102
65	2013-12-02 23:00:00	0.152	0.629	0.099	0.163	0.411	0.233	0.687	0.256	0.020	...	0.082	0.048	0.272	0.189	0.519	0.073	0.329
66	2013-12-03 23:00:00	0.149	0.270	0.098	0.217	0.188	0.199	0.368	0.259	0.022	...	0.092	0.076	0.259	0.136	0.558	0.197	0.356
67	2013-12-05 23:00:00	0.177	0.304	0.099	0.147	0.206	0.155	0.586	0.291	0.021	...	0.088	0.076	0.298	0.263	0.831	0.153	0.384
68	2013-12-06 23:00:00	0.423	0.523	0.124	0.397	0.463	0.330	0.529	0.342	0.021	...	0.094	0.047	0.173	0.112	1.073	0.175	0.386
69	2013-12-09 23:00:00	0.294	0.369	0.100	0.041	0.439	0.314	0.410	0.337	0.020	...	0.069	0.077	0.238	0.392	0.489	0.174	0.377
70	2013-12-10 23:00:00	0.287	0.232	0.152	0.232	0.470	0.339	0.505	0.263	0.016	...	0.071	0.124	0.228	0.344	0.445	0.317	0.367
71	2013-12-11 23:00:00	0.289	0.255	0.103	0.079	0.531	0.396	0.841	0.274	0.015	...	0.038	0.076	0.207	0.827	0.632	0.244	0.346
72	2013-12-14 23:00:00	1.094	0.354	0.146	0.269	0.398	0.355	0.457	0.226	0.029	...	0.042	0.075	0.314	0.107	0.985	0.124	0.348
73	2013-12-17 23:00:00	0.272	0.182	0.186	0.040	0.174	0.290	0.489	0.234	0.026	...	0.062	0.170	0.239	0.126	0.498	0.108	0.358
74	2013-12-18 23:00:00	0.165	0.448	0.100	0.173	0.333	0.357	0.860	0.332	0.029	...	0.086	0.049	0.302	0.129	0.468	0.190	0.337
75	2013-12-21 23:00:00	0.294	0.378	0.149	0.219	0.457	0.262	0.430	0.454	0.020	...	0.059	0.051	0.362	0.107	0.458	0.114	0.354
76	2013-12-22 23:00:00	0.544	0.123	0.150	0.388	0.430	0.200	0.937	0.941	0.026	...	0.116	0.073	0.199	1.462	0.721	0.189	0.415
77	2013-12-23 23:00:00	0.359	0.154	0.127	0.225	0.550	0.379	0.451	0.307	0.026	...	0.086	0.073	0.277	0.118	0.658	0.249	0.140
78	2013-12-24 23:00:00	0.775	0.274	0.099	0.386	0.435	0.285	0.450	1.049	0.016	...	0.065	0.144	0.304	0.116	0.554	0.101	0.140
79	2013-12-25 23:00:00	0.813	0.313	0.100	0.313	0.166	0.329	0.447	0.124	0.027	...	0.065	0.047	0.319	0.111	0.405	0.133	0.141
80	2013-12-27 23:00:00	0.313	0.264	0.099	0.251	0.481	0.294	0.451	0.235	0.032	...	0.066	0.057	0.237	0.333	1.297	0.121	0.342
81	2013-12-28 23:00:00	0.295	0.248	0.151	0.211	0.348	0.189	0.435	0.189	0.022	...	0.039	0.078	0.511	0.109	0.266	0.151	0.373
82	2013-12-31 23:00:00	0.319	0.103	0.104	0.037	0.453	0.382	0.415	0.161	0.017	...	0.073	0.225	0.495	0.110	0.935	0.103	0.354

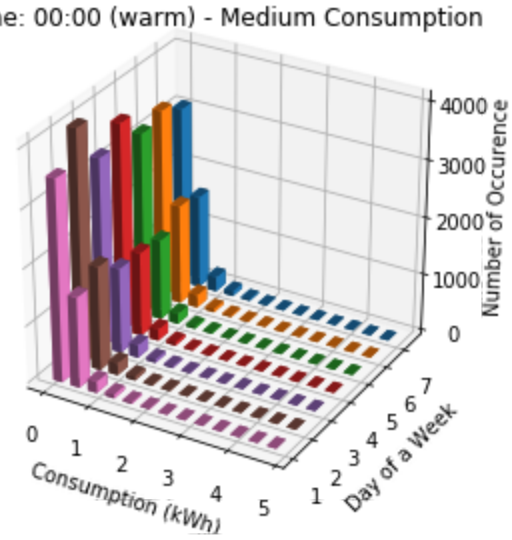
83 rows × 720 columns

In [26]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        t.reset_index(inplace = True)
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1, projection = '3d')
            t2 = t[t.columns[np.insert(kmeans.labels_ == sorted_label[k][0], 0, True)]].copy() # select all d
ata from the n-th cluster
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0'
+ str(i) + ':00:00')]['GMT']).dt.dayofweek) # add day of a week column to dataframe
            for day in reversed(range(7)):
                t3 = t2[t2['DayofWeek'] == day] # select the data with a specific day of a week
                t3 = np.histogram(t3[t3.columns[1:-1]].values.flatten(), bins=13, range=(0,5))
                xpos = t3[1][:-1]
                ypos = np.ones(13) * (day+1)
                num_elements = len(xpos)
                zpos = np.zeros(num_elements)
                dx = np.ones(13) * 0.2
                dy = np.ones(13) * 0.5
                # dz = t3[0] / ((kmeans.labels_ == sorted_label[k][0]).sum() * (t2['DayofWeek'] == day).sum
())
                dz = t3[0]
                pl = ax.bar3d(xpos, ypos, zpos, dx, dy, dz, alpha = 0.5, zsort='max')
                pl._sort_zpos = i * 3 + k + 1
            ax.set_title('Time: 0' + str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Day of a Week')
            ax.set_zlabel('Number of Occurence')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        t.reset_index(inplace = True)
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        for k in range(n):
            ax = fig_all.add_subplot(24, 3, i * 3 + k + 1, projection = '3d')
            t2 = t[t.columns[np.insert(kmeans.labels_ == sorted_label[k][0], 0, True)]].copy() # select all d
ata from the n-th cluster
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(
i) + ':00:00')]['GMT']).dt.dayofweek) # add day of weeks column to dataframe
            for day in reversed(range(7)):
                t3 = t2[t2['DayofWeek'] == day] # select the data with a specific day of a week
                t3 = np.histogram(t3[t3.columns[1:-1]].values.flatten(), bins=13, range=(0,5))
                xpos = t3[1][:-1]
                ypos = np.ones(13) * (day+1)
                num_elements = len(xpos)
                zpos = np.zeros(num_elements)
                dx = np.ones(13) * 0.2
                dy = np.ones(13) * 0.5
                # dz = t3[0] / ((kmeans.labels_ == sorted_label[k][0]).sum() * (t2['DayofWeek'] == day).sum
())
                dz = t3[0]
                pl = ax.bar3d(xpos, ypos, zpos, dx, dy, dz, alpha = 0.5, zsort='max')
                pl._sort_zpos = i * 3 + k + 1
            ax.set_title('Time: ' + str(i) + ':00' + ' (warm) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Day of a Week')
            ax.set_zlabel('Number of Occurence')
plt.tight_layout()
```

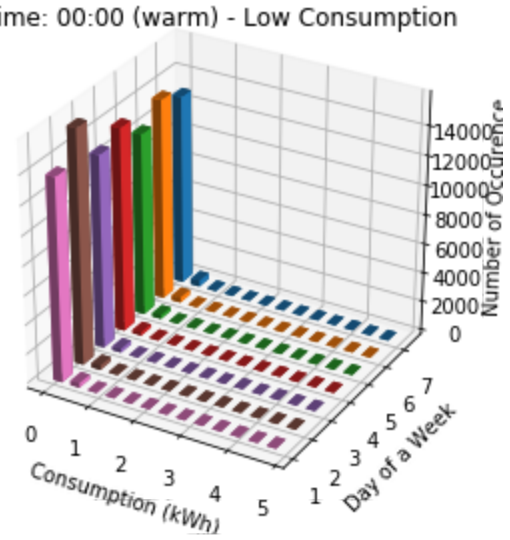
Time: 00:00 (warm) - High Consumption



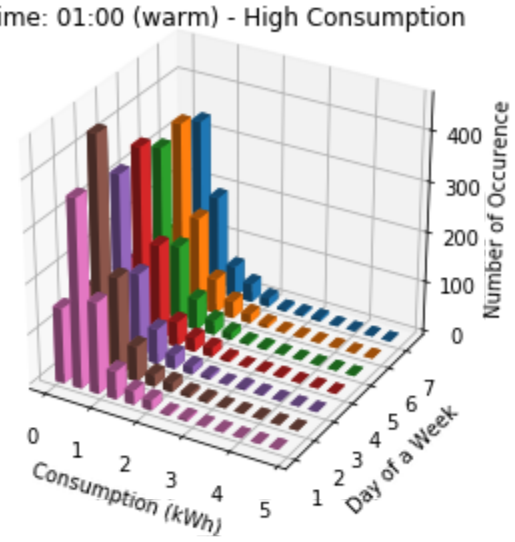
Time: 00:00 (warm) - Medium Consumption



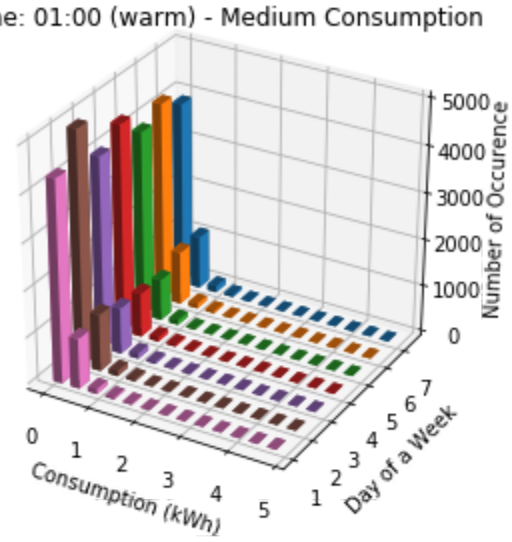
Time: 00:00 (warm) - Low Consumption



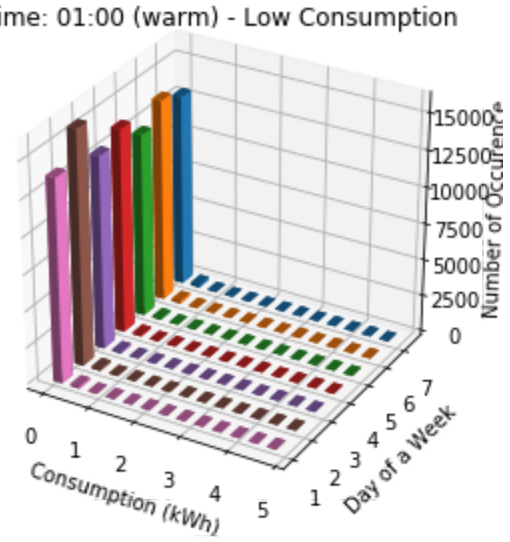
Time: 01:00 (warm) - High Consumption



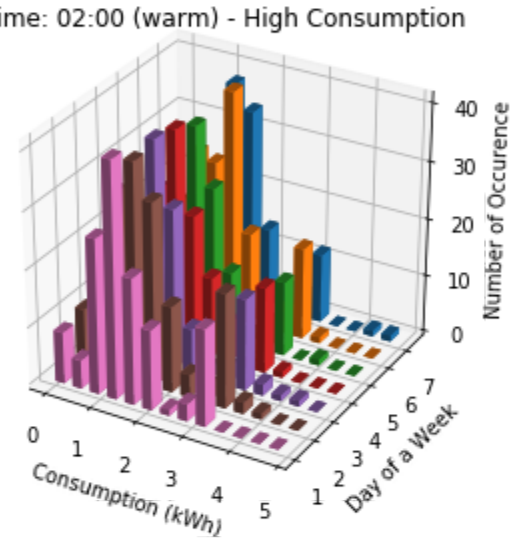
Time: 01:00 (warm) - Medium Consumption



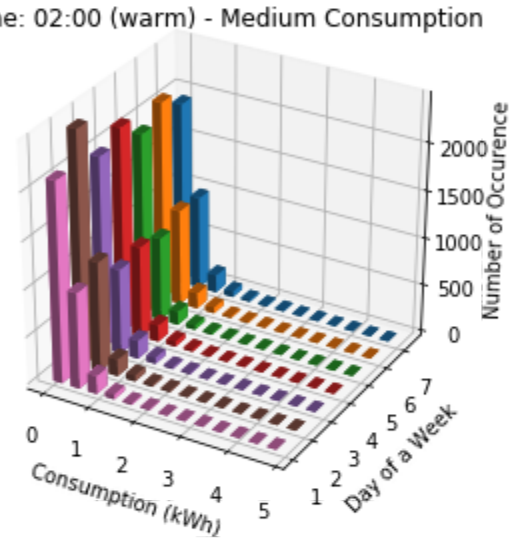
Time: 01:00 (warm) - Low Consumption



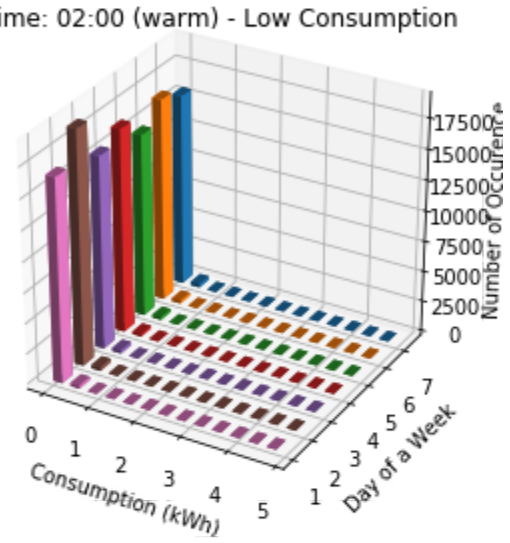
Time: 02:00 (warm) - High Consumption



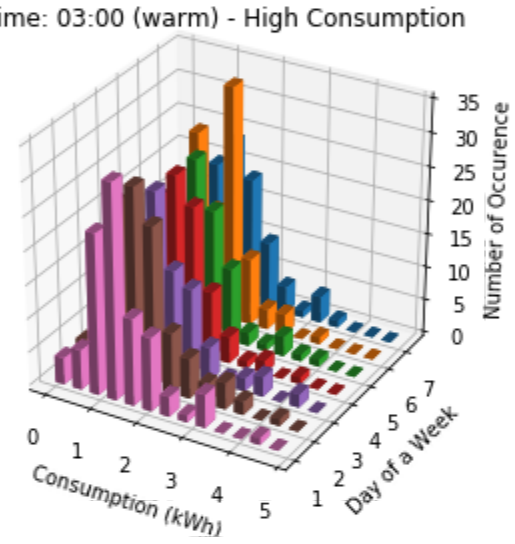
Time: 02:00 (warm) - Medium Consumption



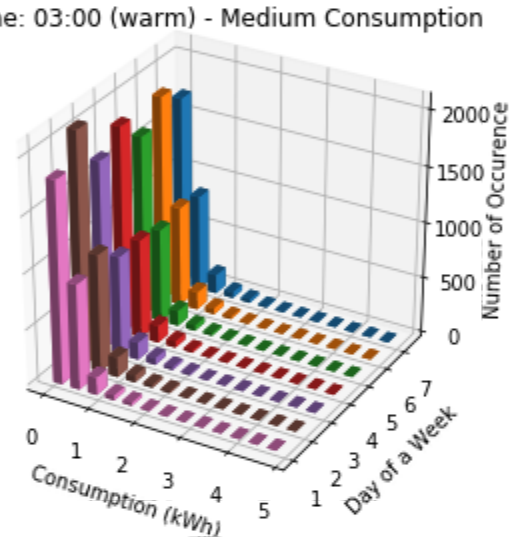
Time: 02:00 (warm) - Low Consumption



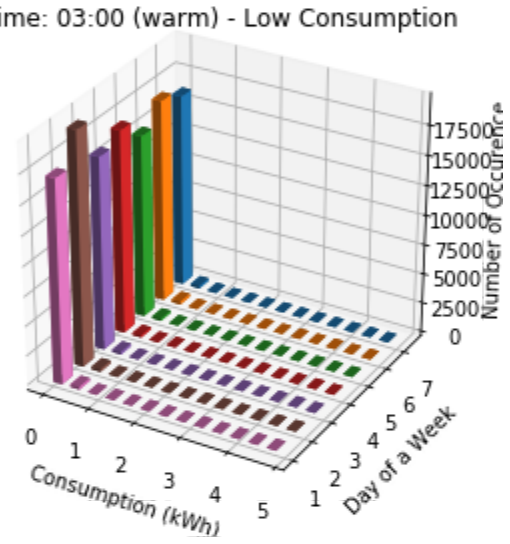
Time: 03:00 (warm) - High Consumption



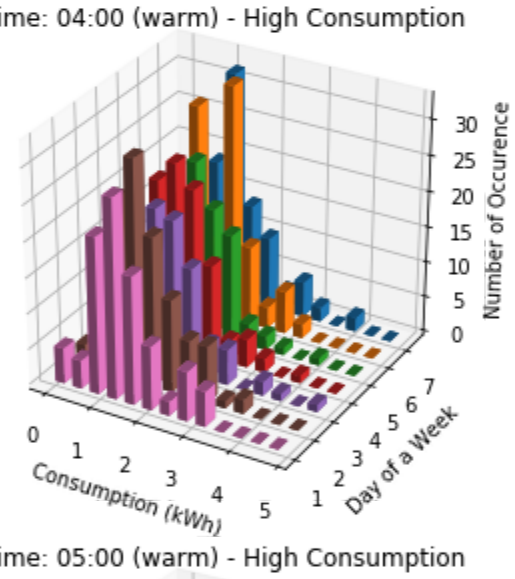
Time: 03:00 (warm) - Medium Consumption



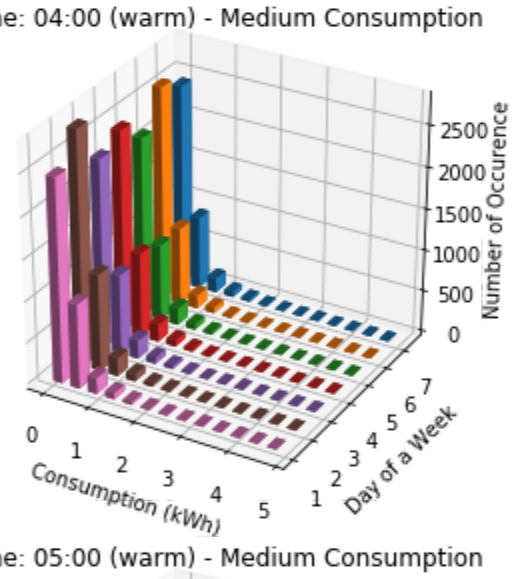
Time: 03:00 (warm) - Low Consumption



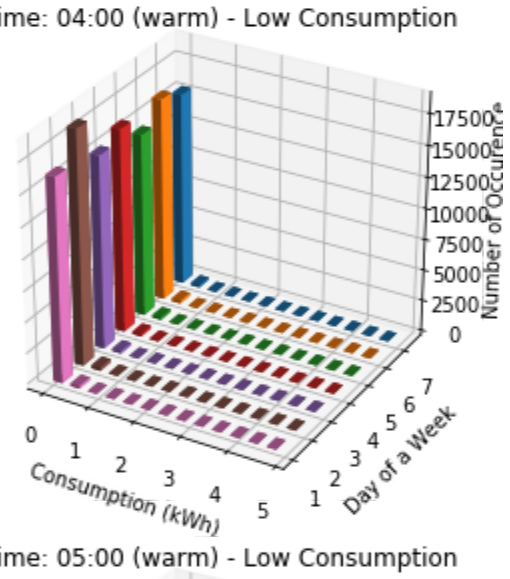
Time: 04:00 (warm) - High Consumption



Time: 04:00 (warm) - Medium Consumption



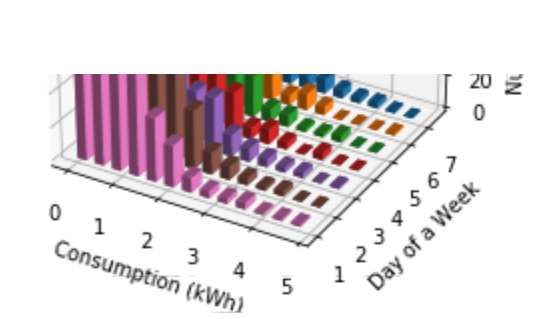
Time: 04:00 (warm) - Low Consumption



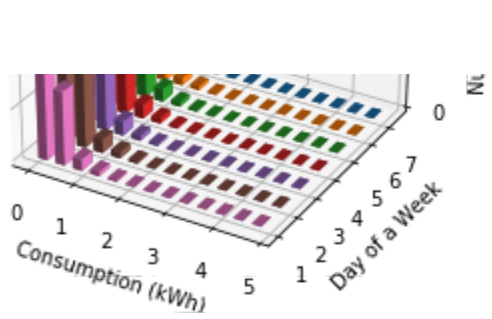
Time: 05:00 (warm) - High Consumption

Time: 05:00 (warm) - Medium Consumption

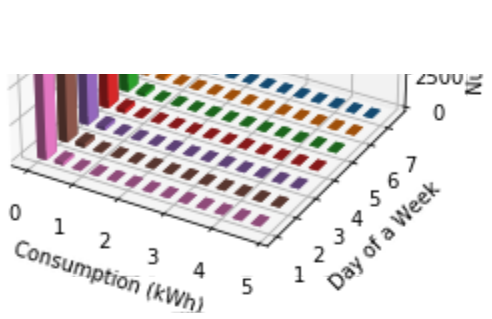
Time: 05:00 (warm) - Low Consumption



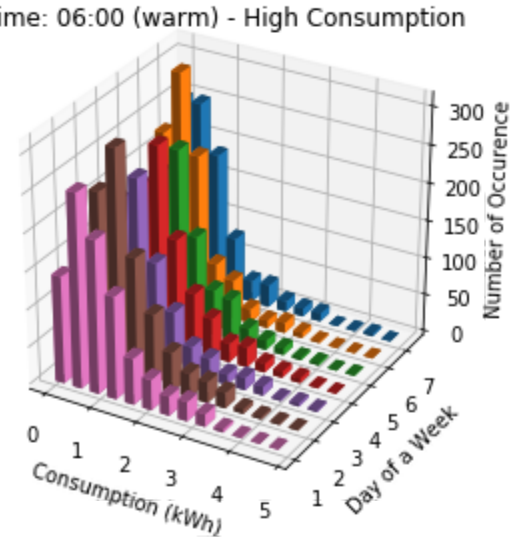
Time: 06:00 (warm) - High Consumption



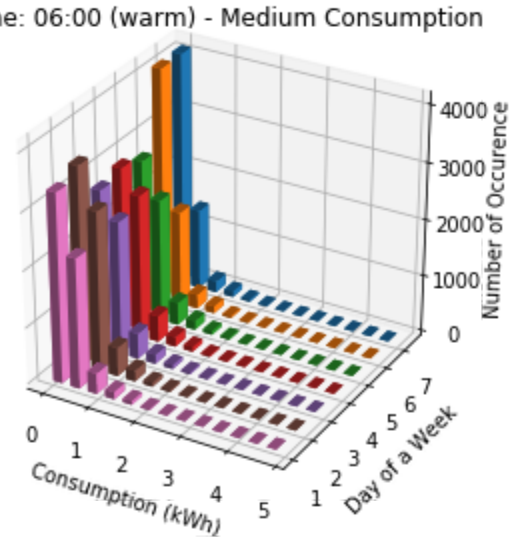
Time: 06:00 (warm) - Medium Consumption



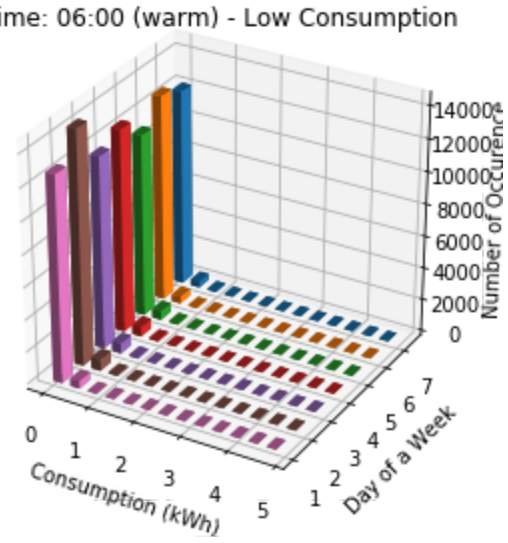
Time: 06:00 (warm) - Low Consumption



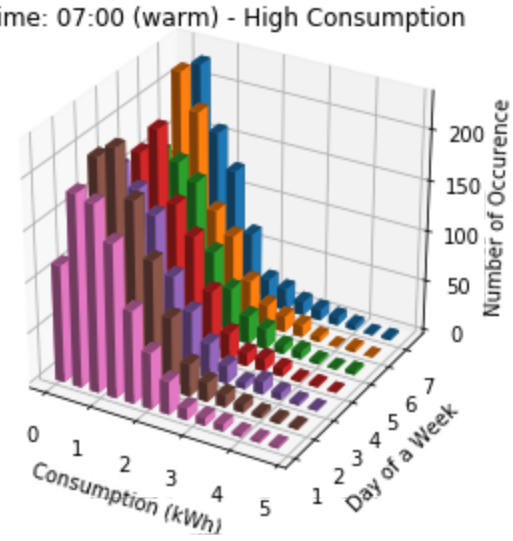
Time: 07:00 (warm) - High Consumption



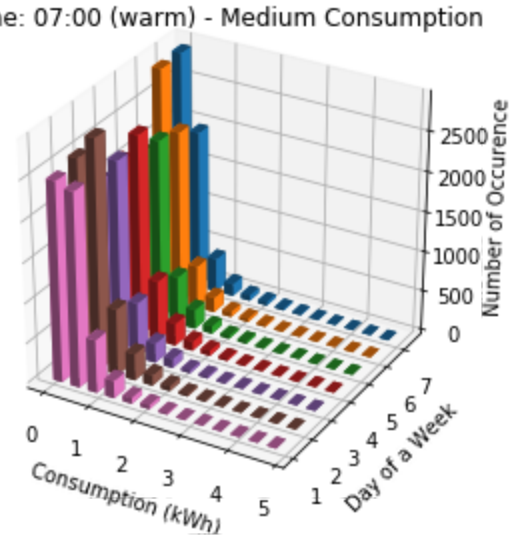
Time: 07:00 (warm) - Medium Consumption



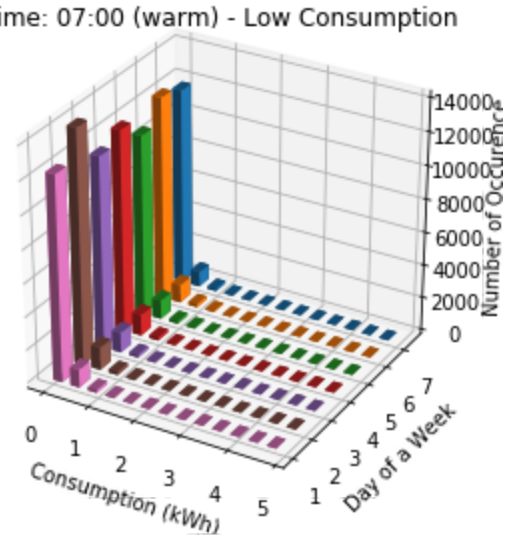
Time: 07:00 (warm) - Low Consumption



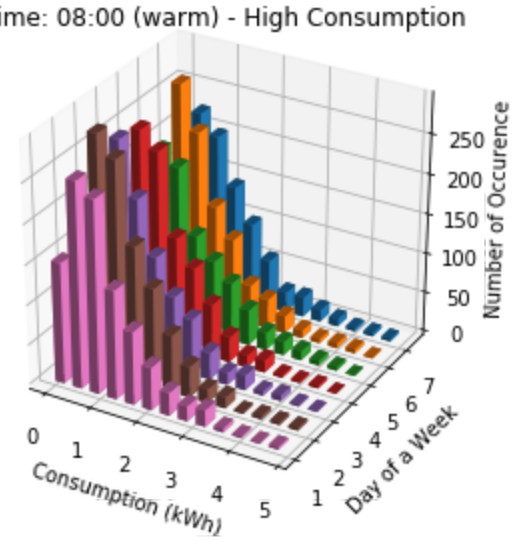
Time: 08:00 (warm) - High Consumption



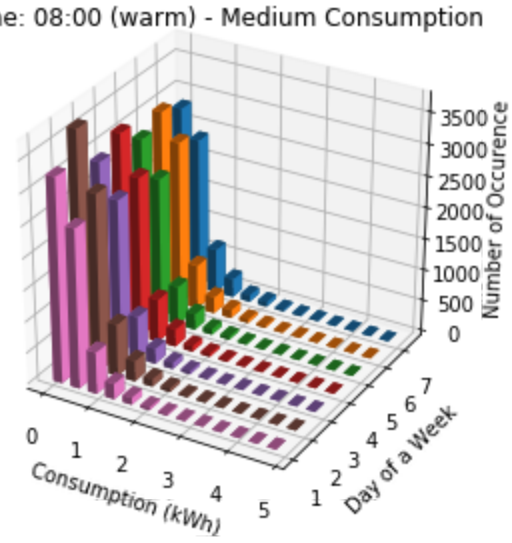
Time: 08:00 (warm) - Medium Consumption



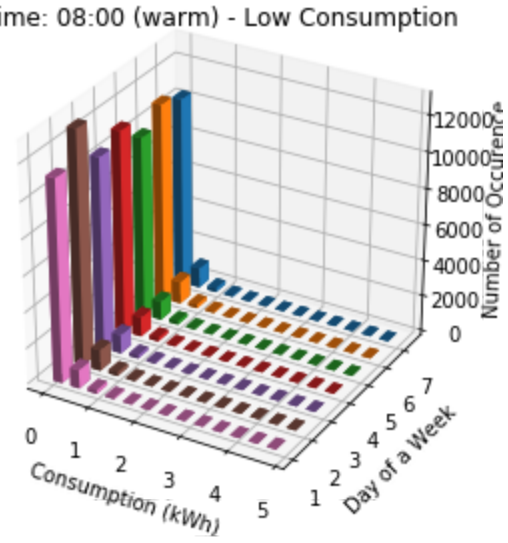
Time: 08:00 (warm) - Low Consumption



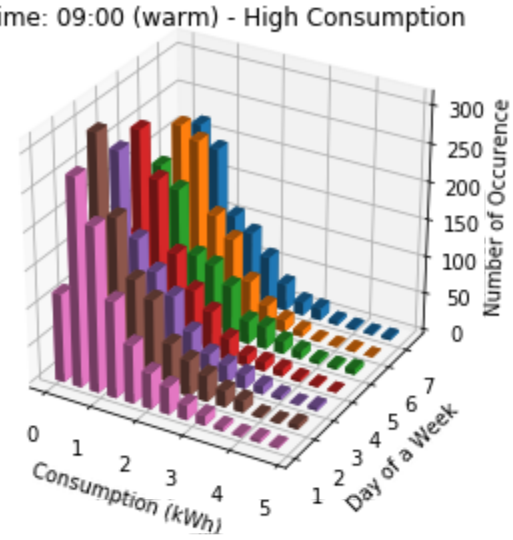
Time: 09:00 (warm) - High Consumption



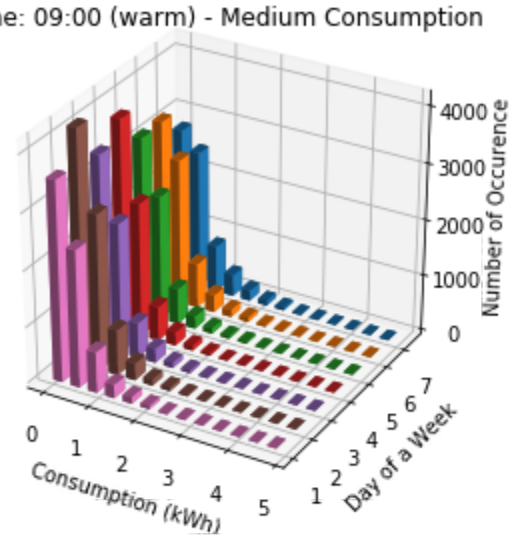
Time: 09:00 (warm) - Medium Consumption



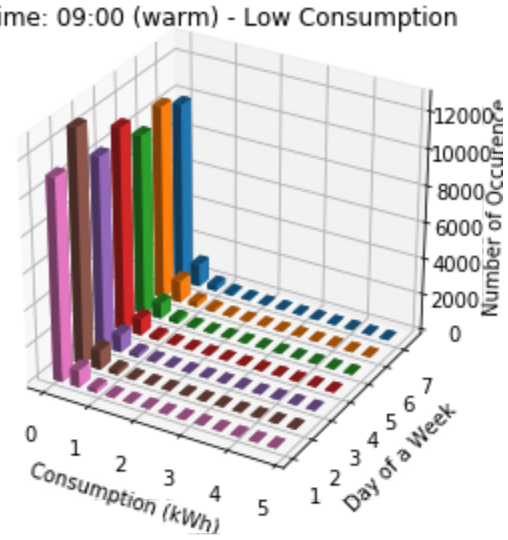
Time: 09:00 (warm) - Low Consumption



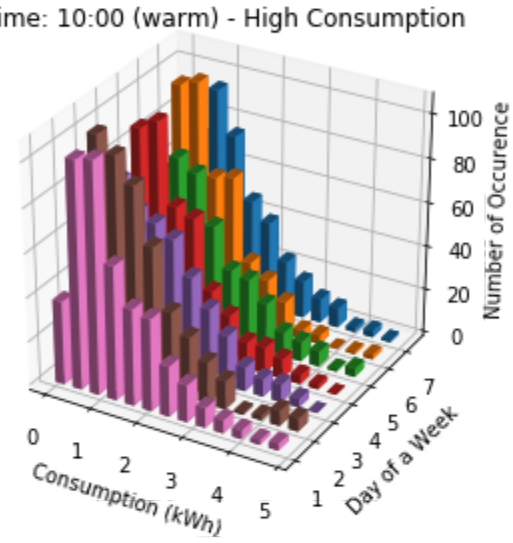
Time: 10:00 (warm) - High Consumption



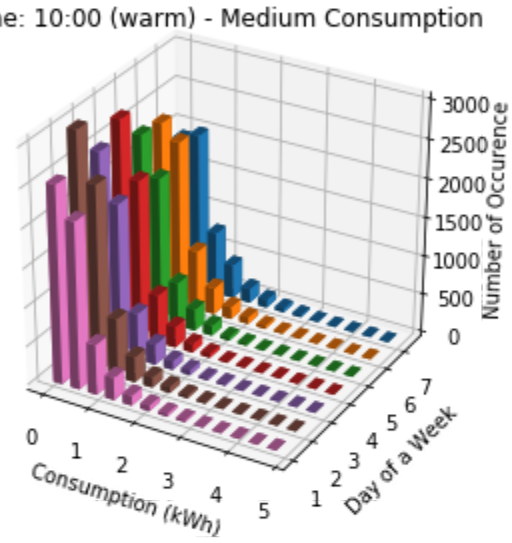
Time: 10:00 (warm) - Medium Consumption



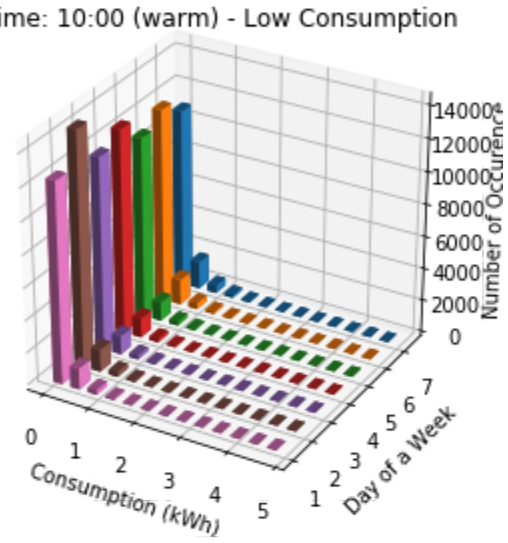
Time: 10:00 (warm) - Low Consumption



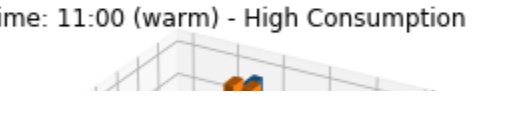
Time: 11:00 (warm) - High Consumption

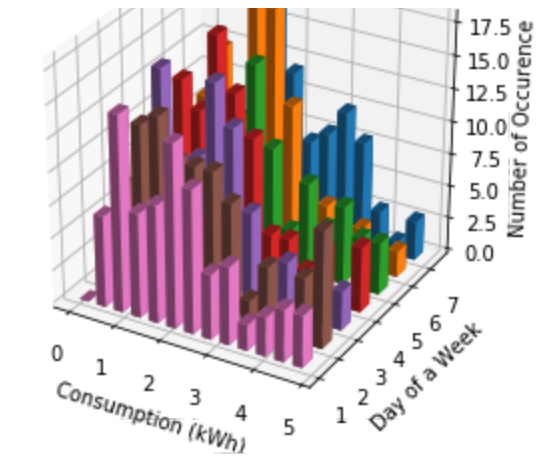


Time: 11:00 (warm) - Medium Consumption

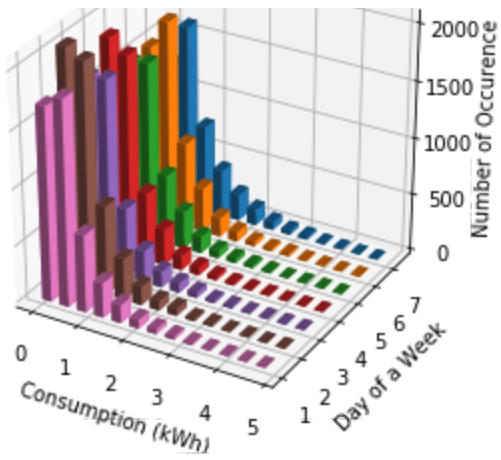


Time: 11:00 (warm) - Low Consumption

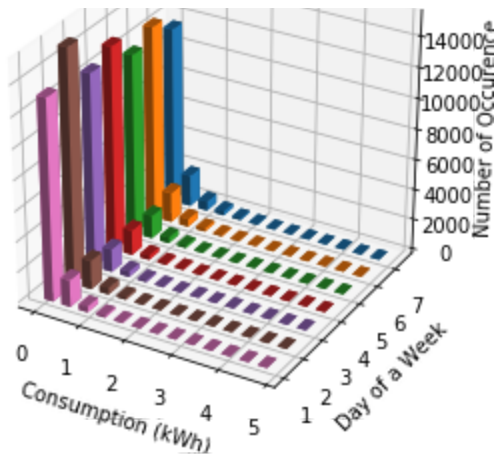




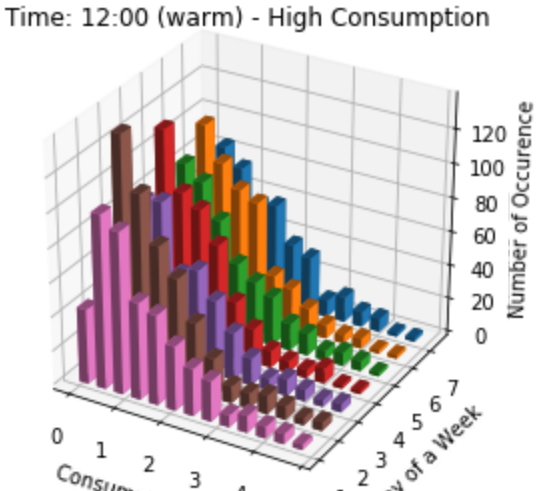
Time: 12:00 (warm) - High Consumption



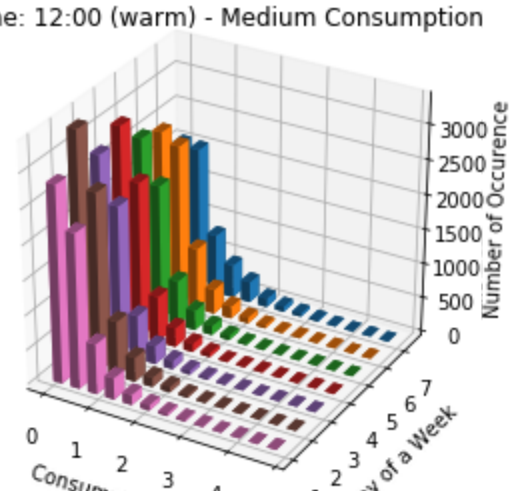
Time: 12:00 (warm) - Medium Consumption



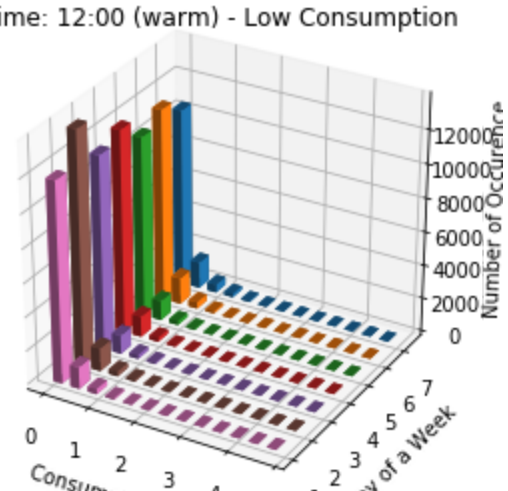
Time: 12:00 (warm) - Low Consumption



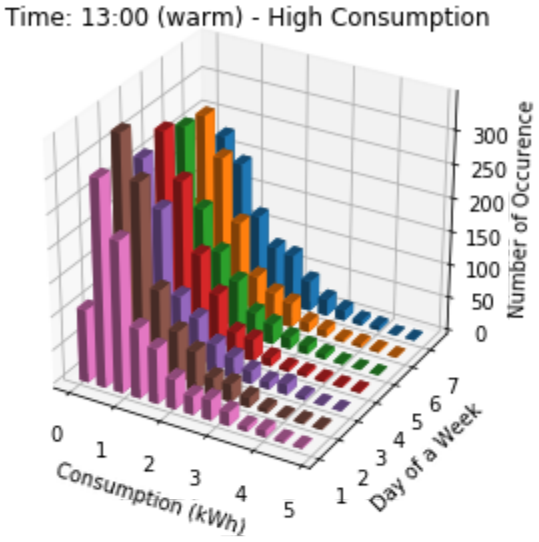
Time: 13:00 (warm) - High Consumption



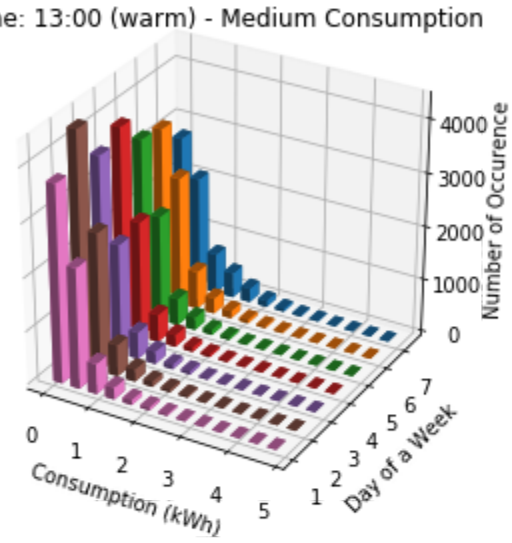
Time: 13:00 (warm) - Medium Consumption



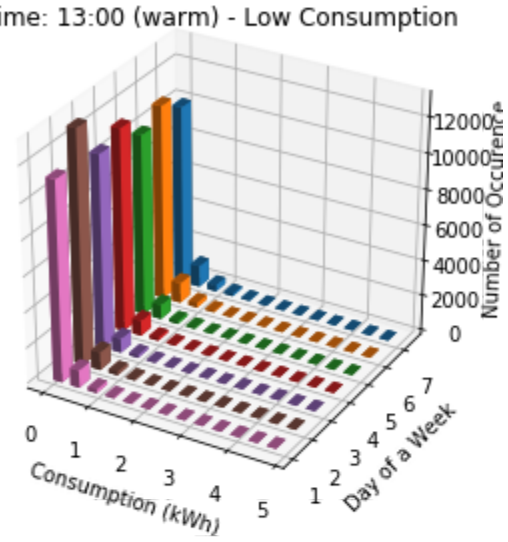
Time: 13:00 (warm) - Low Consumption



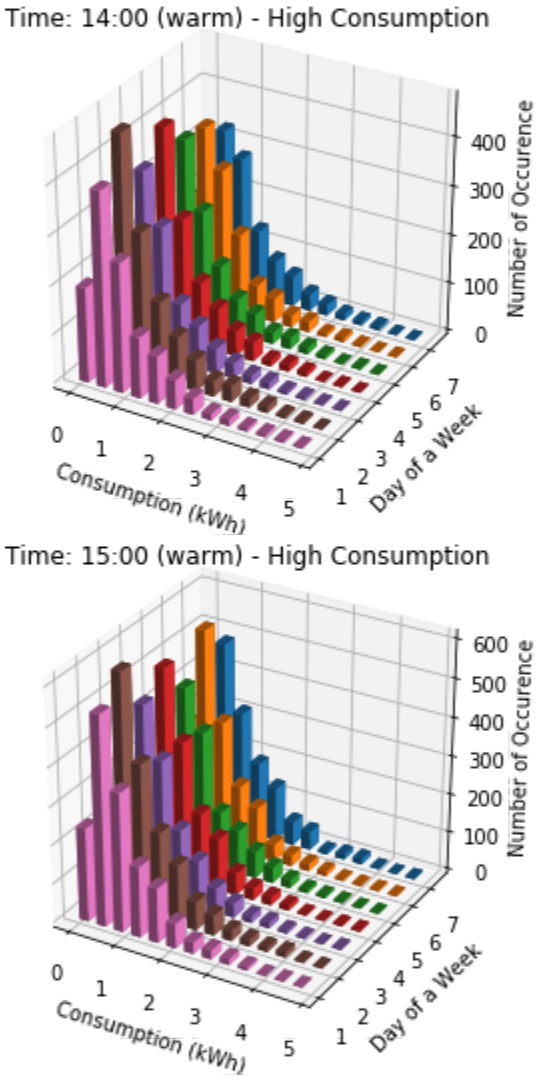
Time: 14:00 (warm) - High Consumption



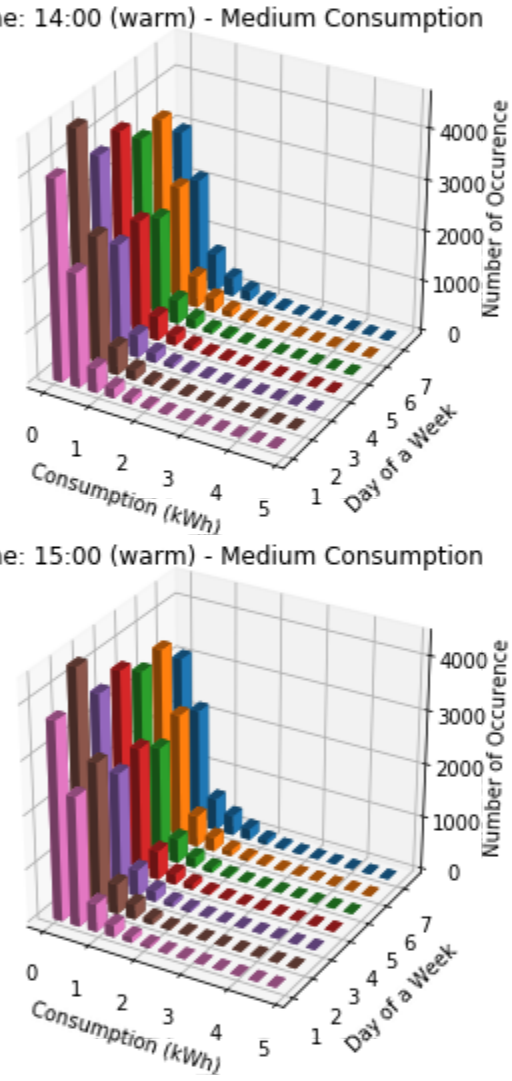
Time: 14:00 (warm) - Medium Consumption



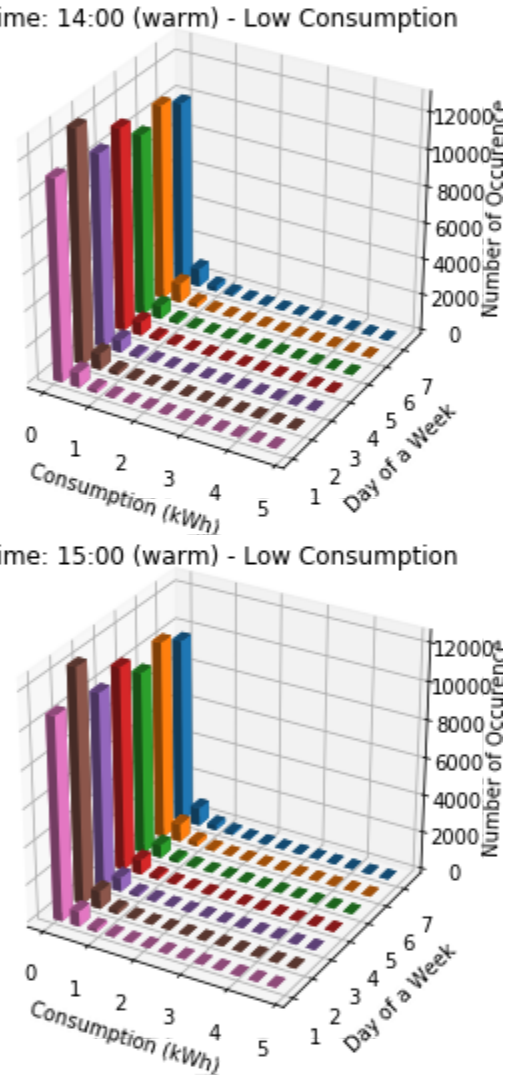
Time: 14:00 (warm) - Low Consumption



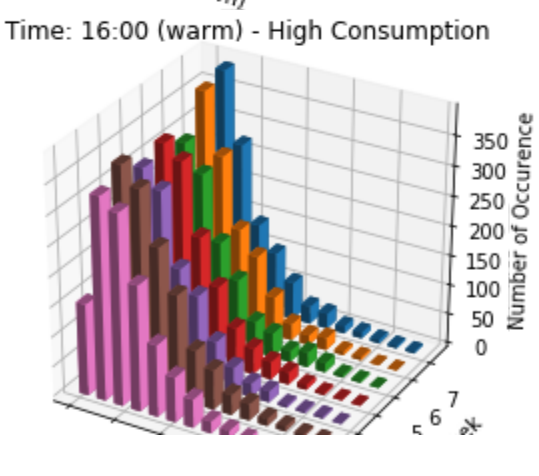
Time: 15:00 (warm) - High Consumption



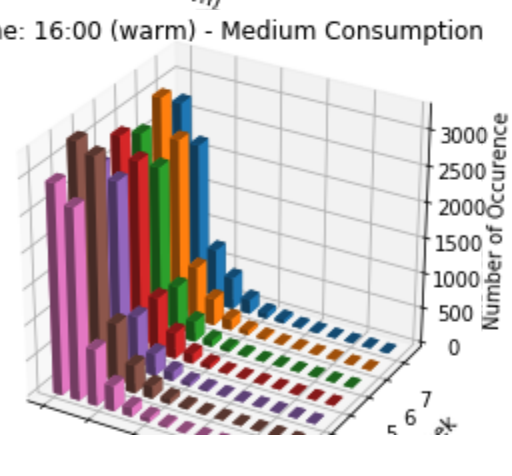
Time: 15:00 (warm) - Medium Consumption



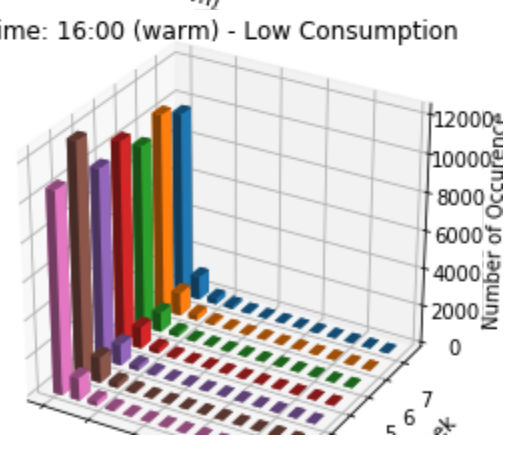
Time: 15:00 (warm) - Low Consumption



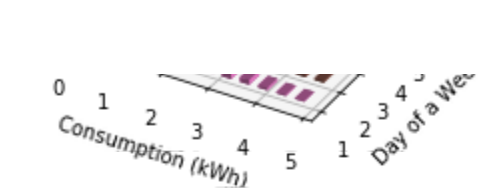
Time: 16:00 (warm) - High Consumption

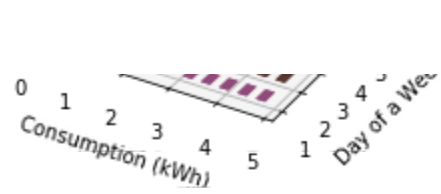


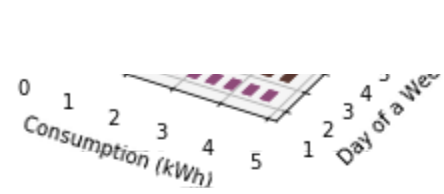
Time: 16:00 (warm) - Medium Consumption



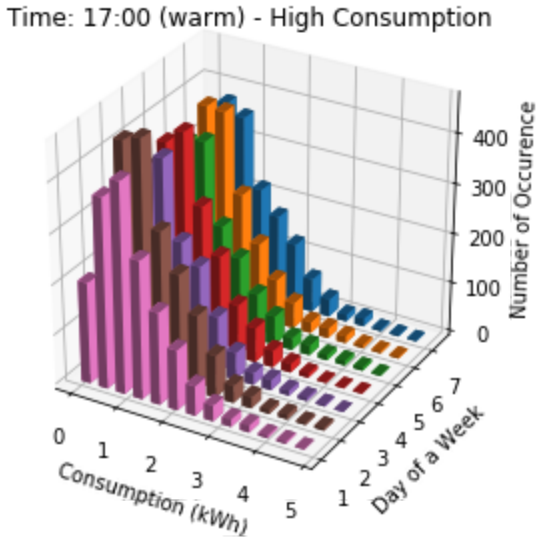
Time: 16:00 (warm) - Low Consumption



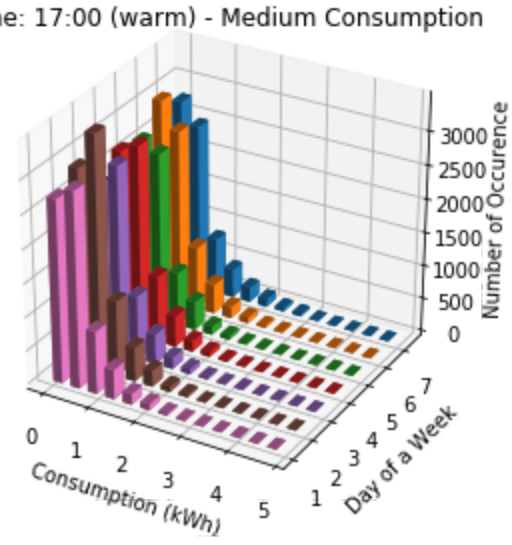




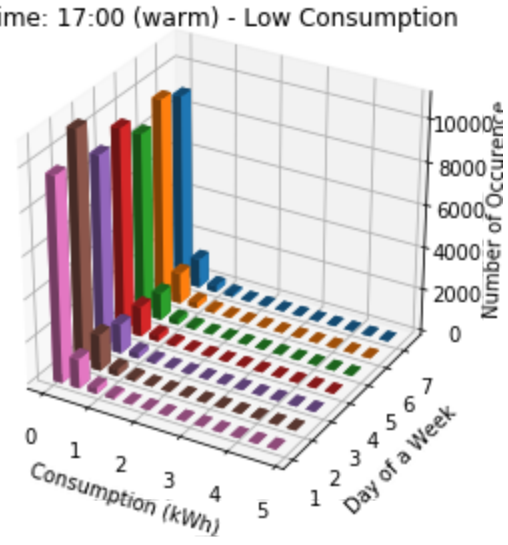
Time: 17:00 (warm) - High Consumption



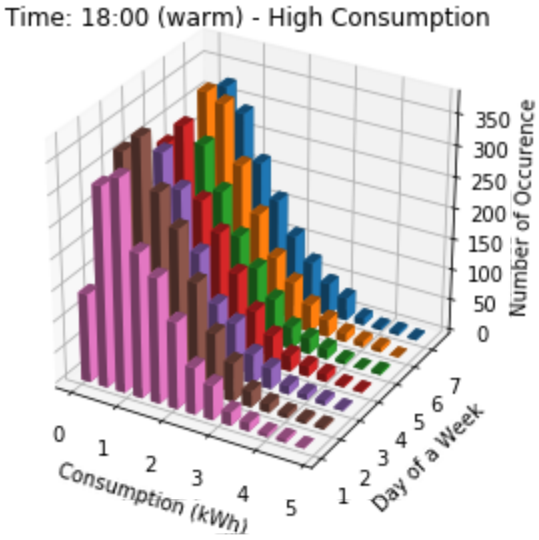
Time: 17:00 (warm) - Medium Consumption



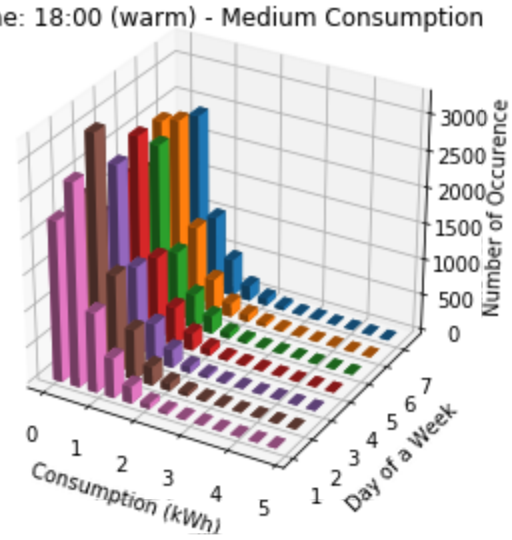
Time: 17:00 (warm) - Low Consumption



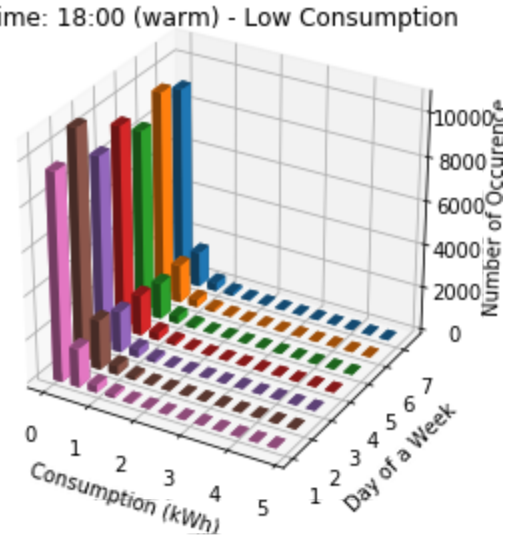
Time: 18:00 (warm) - High Consumption



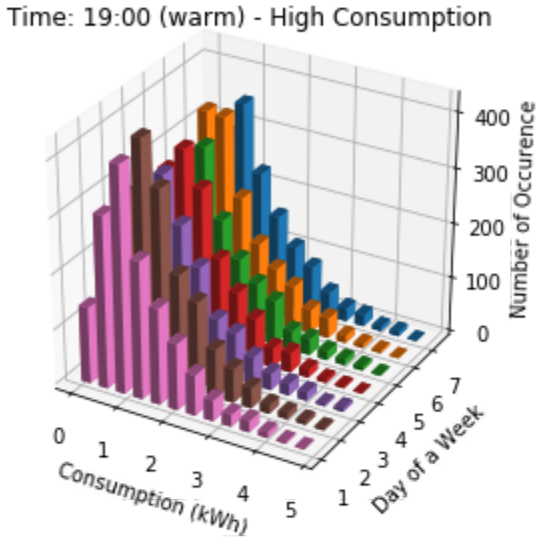
Time: 18:00 (warm) - Medium Consumption



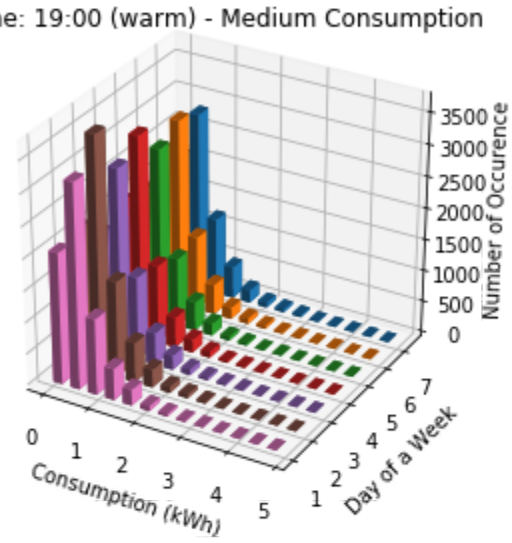
Time: 18:00 (warm) - Low Consumption



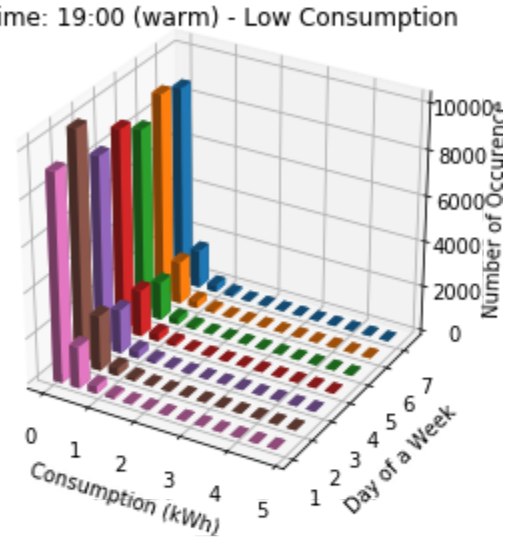
Time: 19:00 (warm) - High Consumption



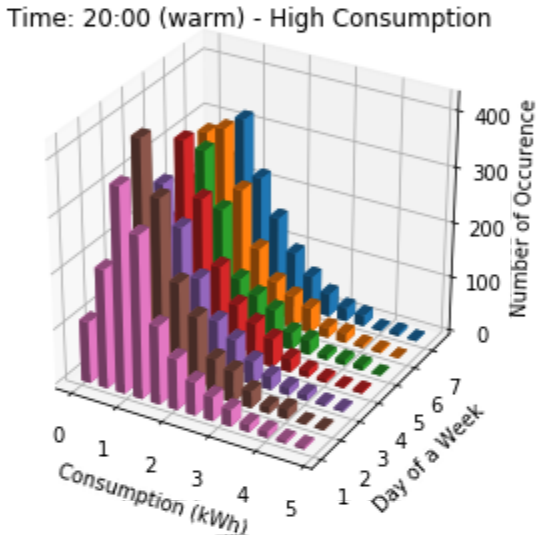
Time: 19:00 (warm) - Medium Consumption



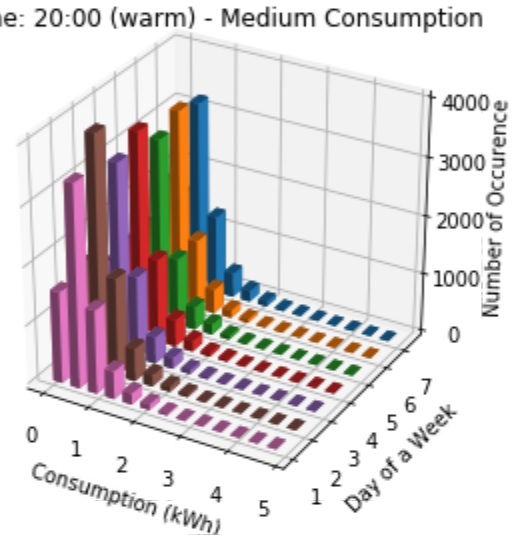
Time: 19:00 (warm) - Low Consumption



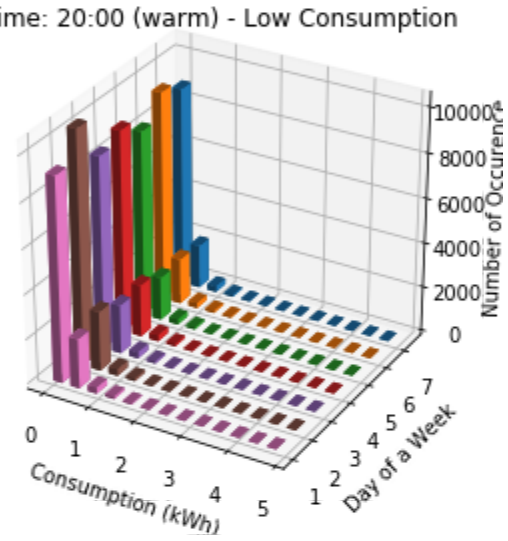
Time: 20:00 (warm) - High Consumption



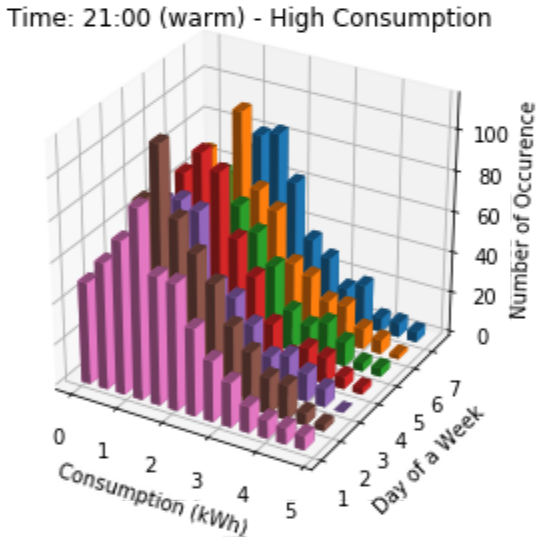
Time: 20:00 (warm) - Medium Consumption



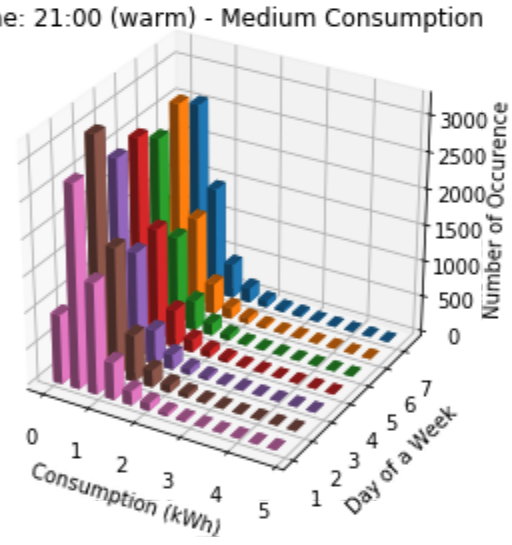
Time: 20:00 (warm) - Low Consumption



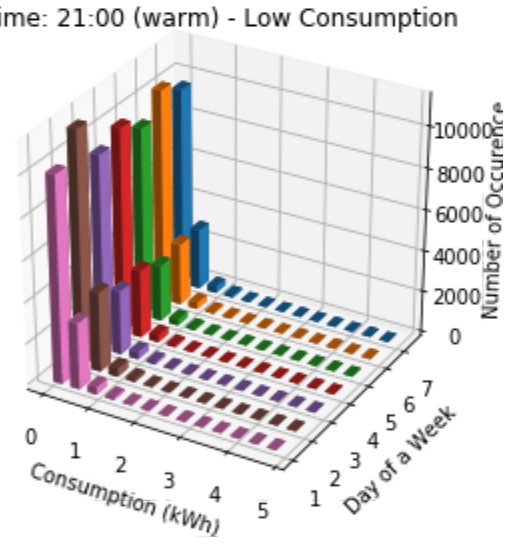
Time: 21:00 (warm) - High Consumption



Time: 21:00 (warm) - Medium Consumption



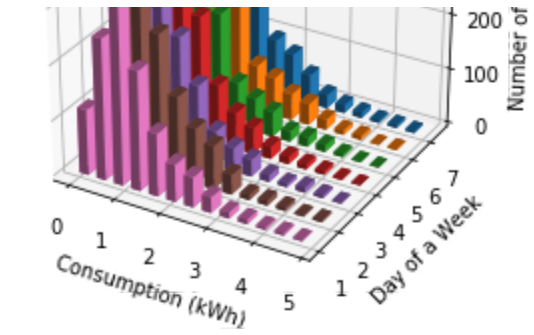
Time: 21:00 (warm) - Low Consumption



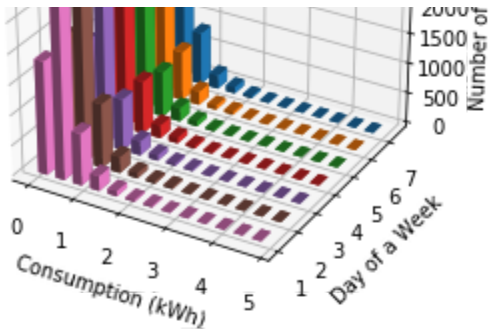
Time: 22:00 (warm) - High Consumption

Time: 22:00 (warm) - Medium Consumption

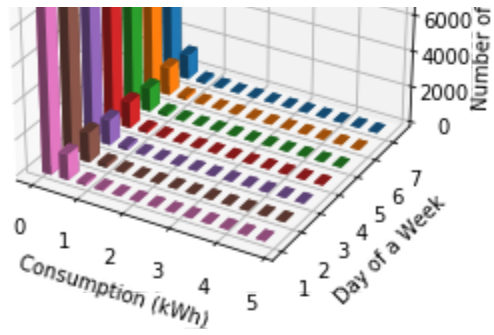
Time: 22:00 (warm) - Low Consumption



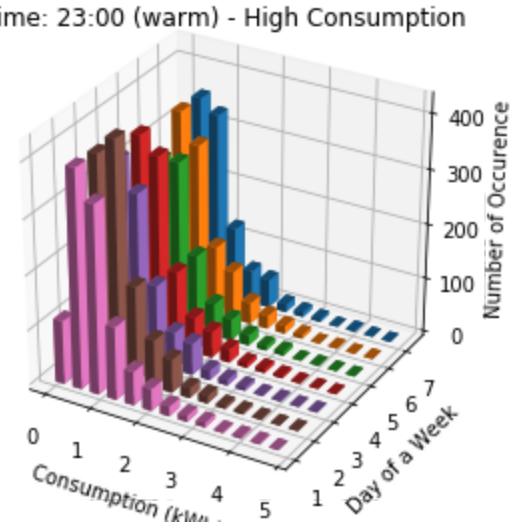
Time: 23:00 (warm) - High Consumption

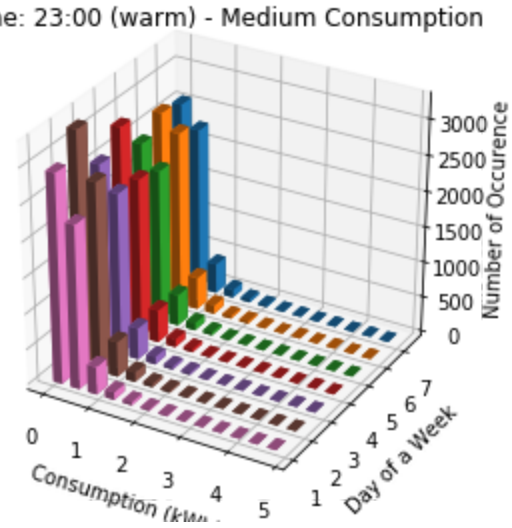


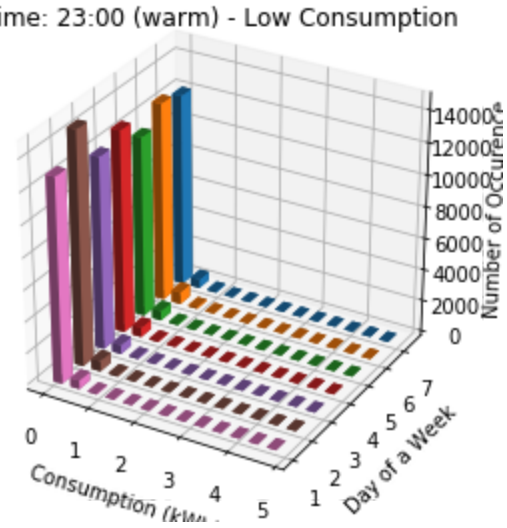
Time: 23:00 (warm) - Medium Consumption



Time: 23:00 (warm) - Low Consumption

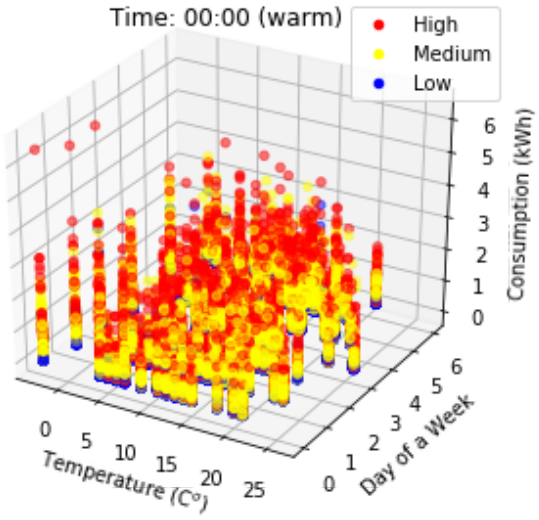


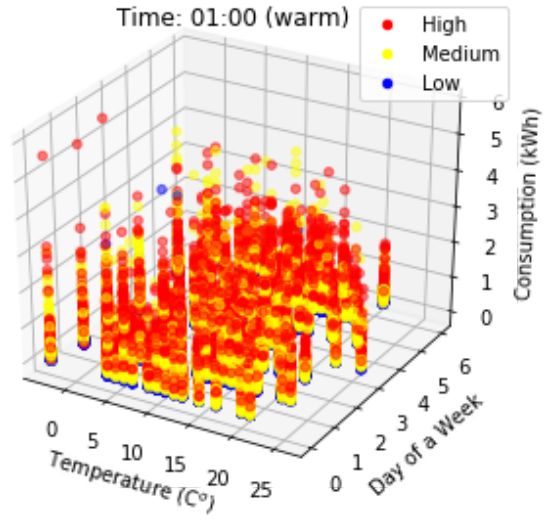


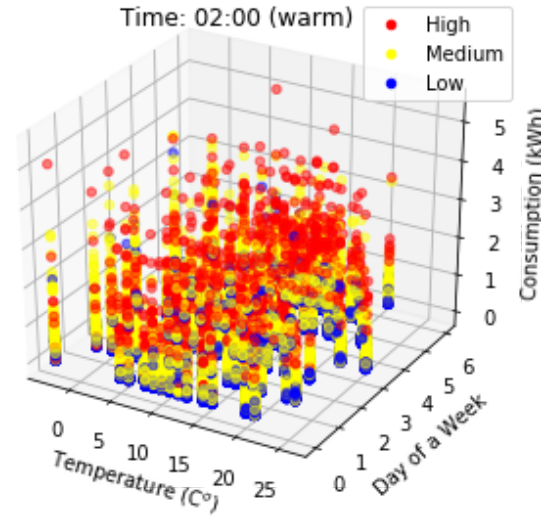


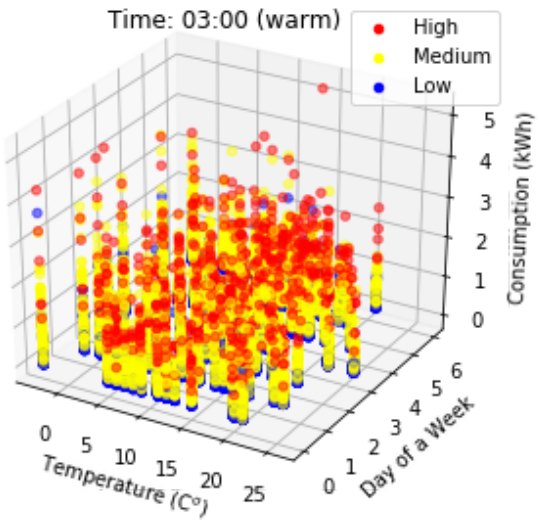
Plot the three groups generated by PCA and k-mean clustering, in a space with 3 dimensions, temperature, day of week and consumption, and see if there is any obvious pattern.

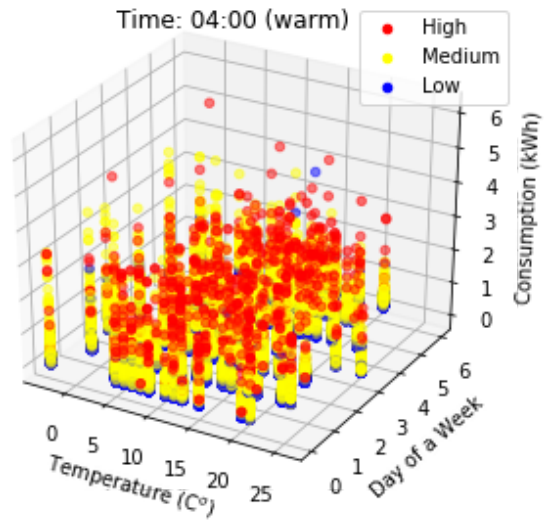
In [27]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
cons_type = ['High', 'Medium', 'Low']
color_type = ['red', 'yellow', 'blue']
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        t.reset_index(inplace = True)
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        ax = fig_all.add_subplot(8, 3, i + 1, projection = '3d')
        for k in range(n):
            t2 = t[t.columns[np.insert(kmeans.labels_ == sorted_label[k][0], 0, True)]].copy() # select all d
ata from the n-th cluster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00'
)]['TempC'].values) # add temperature column to dataframe
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0'
+ str(i) + ':00:00')]['GMT']).dt.dayofweek) # add day of a week column to dataframe
            ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[1]], color = color_type[k], labe
l = cons_type[k])
            for j in range(2, 1 + sorted_label[k][1]):
                ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[j]], color = color_type[k],
alpha = 0.5)
                ax.legend()
        ax.set_title('Time: 0' + str(i) + ':00' + ' (warm)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Day of a Week')
        ax.set_zlabel('Consumption (kWh)')
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.fillna(x.mean()) # data used for analyzing consumption distribution
        x = t.transpose() # data for the specific hour, and filtering out null value
        t.reset_index(inplace = True)
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n).fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        for k in range(n):
            cluster_dict[k] = (kmeans.labels_ == k).sum() # store the labels and the number of data in a dict
        sorted_label = sorted(cluster_dict.items(), key=lambda kv: kv[1])
        ax = fig_all.add_subplot(8, 3, i + 1, projection = '3d')
        for k in range(n):
            t2 = t[t.columns[np.insert(kmeans.labels_ == sorted_label[k][0], 0, True)]].copy() # select all d
ata from the n-th cluster
            t2['TempC'] = np.round(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['Te
mpC'].values) # add temperature column to dataframe
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(
i) + ':00:00')]['GMT']).dt.dayofweek) # add day of a week column to dataframe
            ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[1]], color = color_type[k], labe
l = cons_type[k])
            for j in range(2, 1 + sorted_label[k][1]):
                ax.scatter(t2[t2.columns[-2]], t2[t2.columns[-1]], t2[t2.columns[j]], color = color_type[k],
alpha = 0.5)
                ax.legend()
        ax.set_title('Time: ' + str(i) + ':00' + ' (warm)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Day of a Week')
        ax.set_zlabel('Consumption (kWh)')
plt.tight_layout()
```

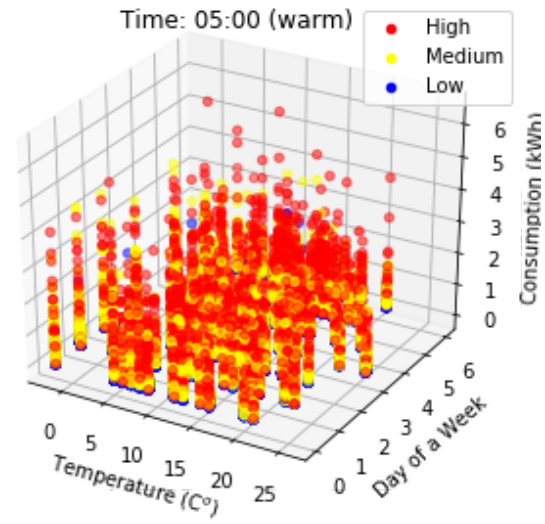

Time: 00:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

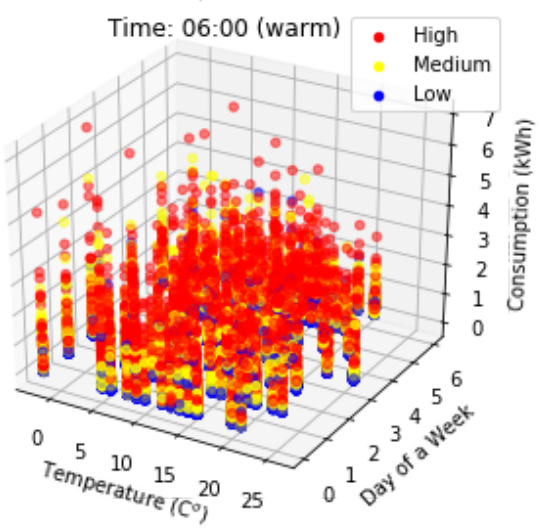

Time: 01:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

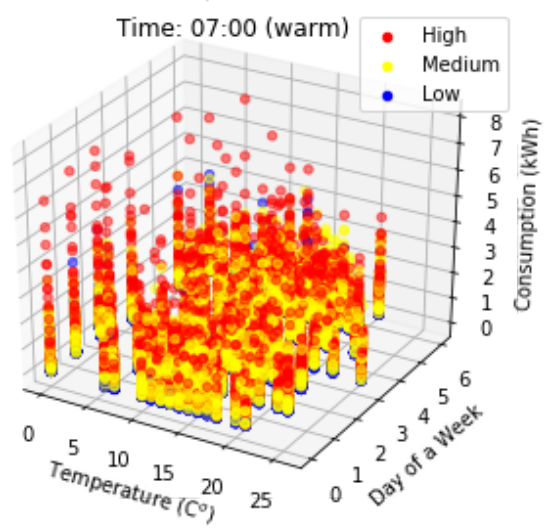

Time: 02:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

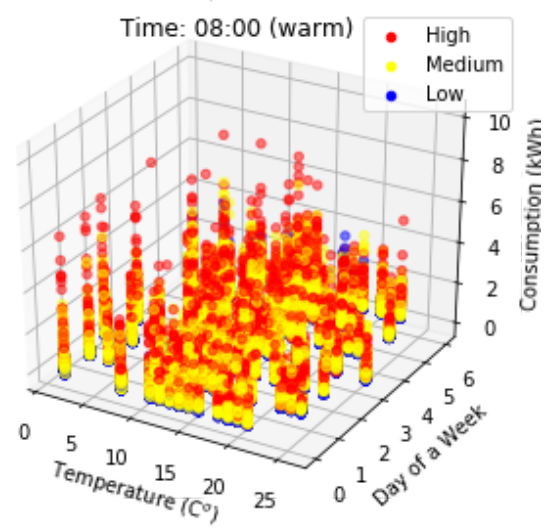

Time: 03:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

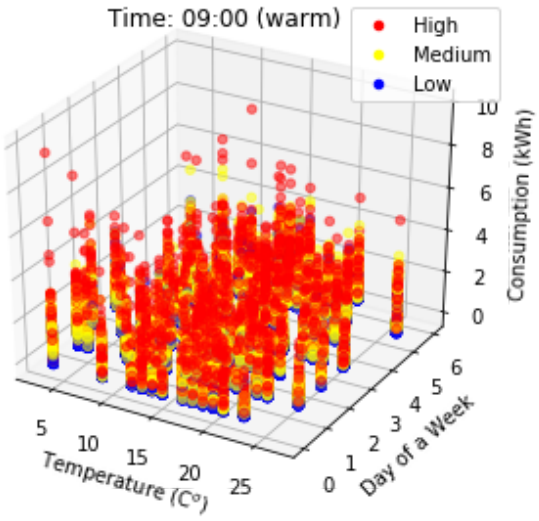

Time: 04:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

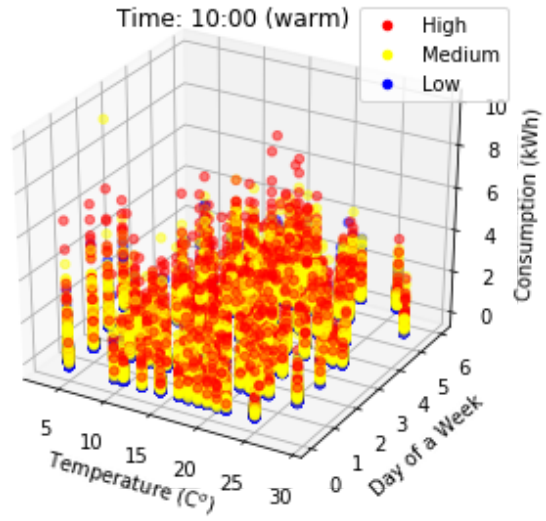

Time: 05:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

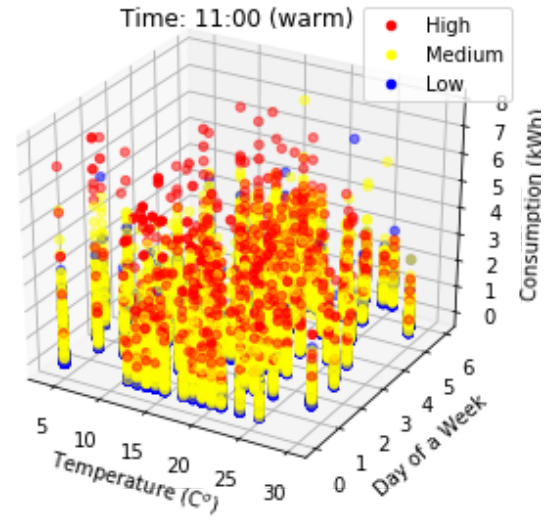

Time: 06:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

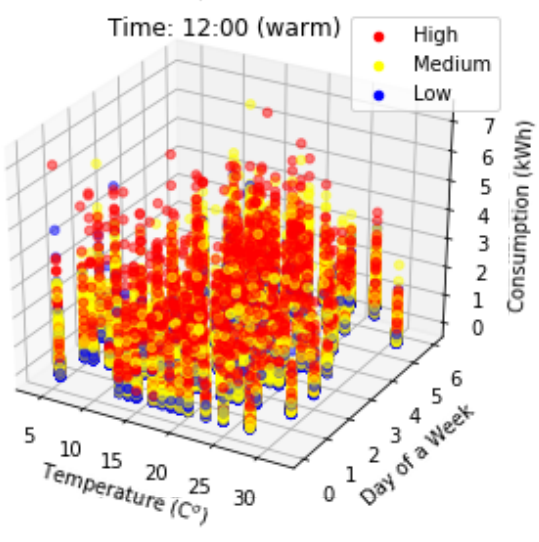

Time: 07:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

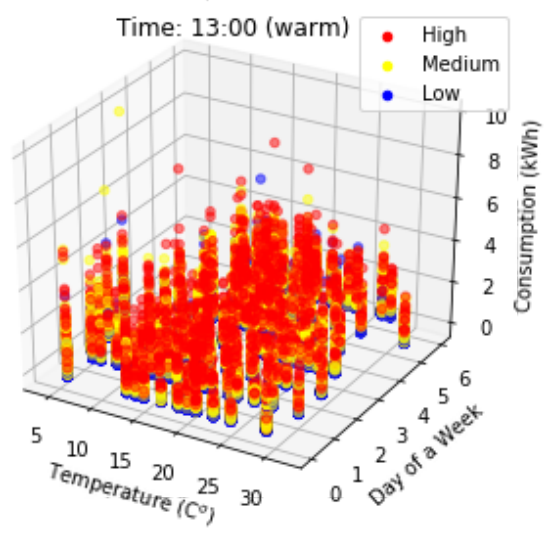

Time: 08:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

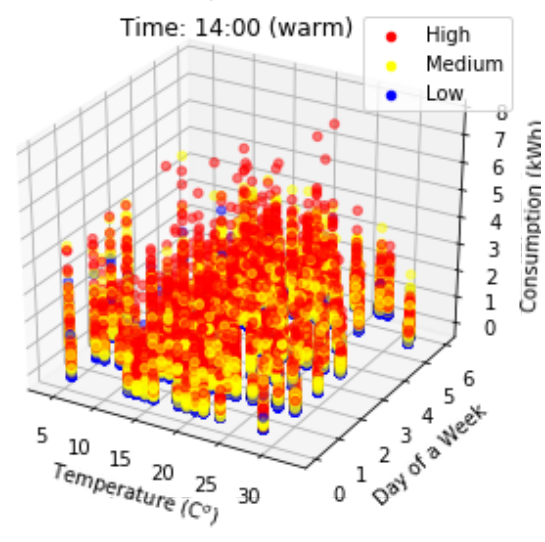

Time: 09:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

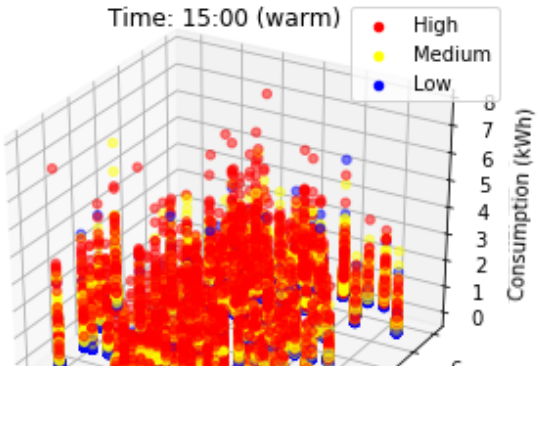

Time: 10:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

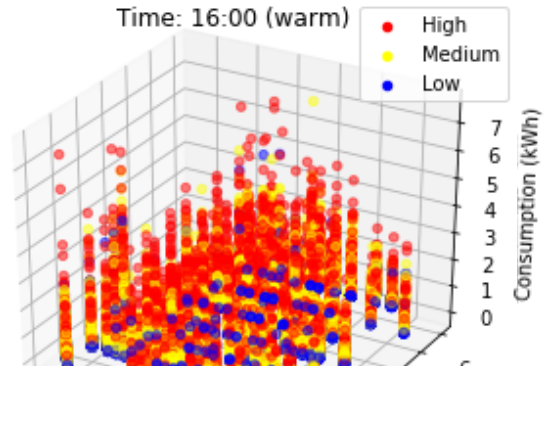

Time: 11:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

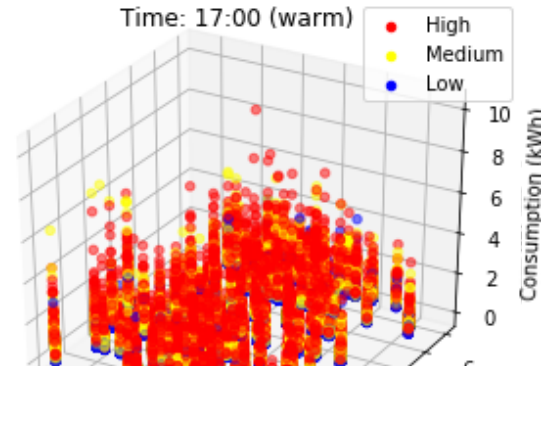

Time: 12:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)

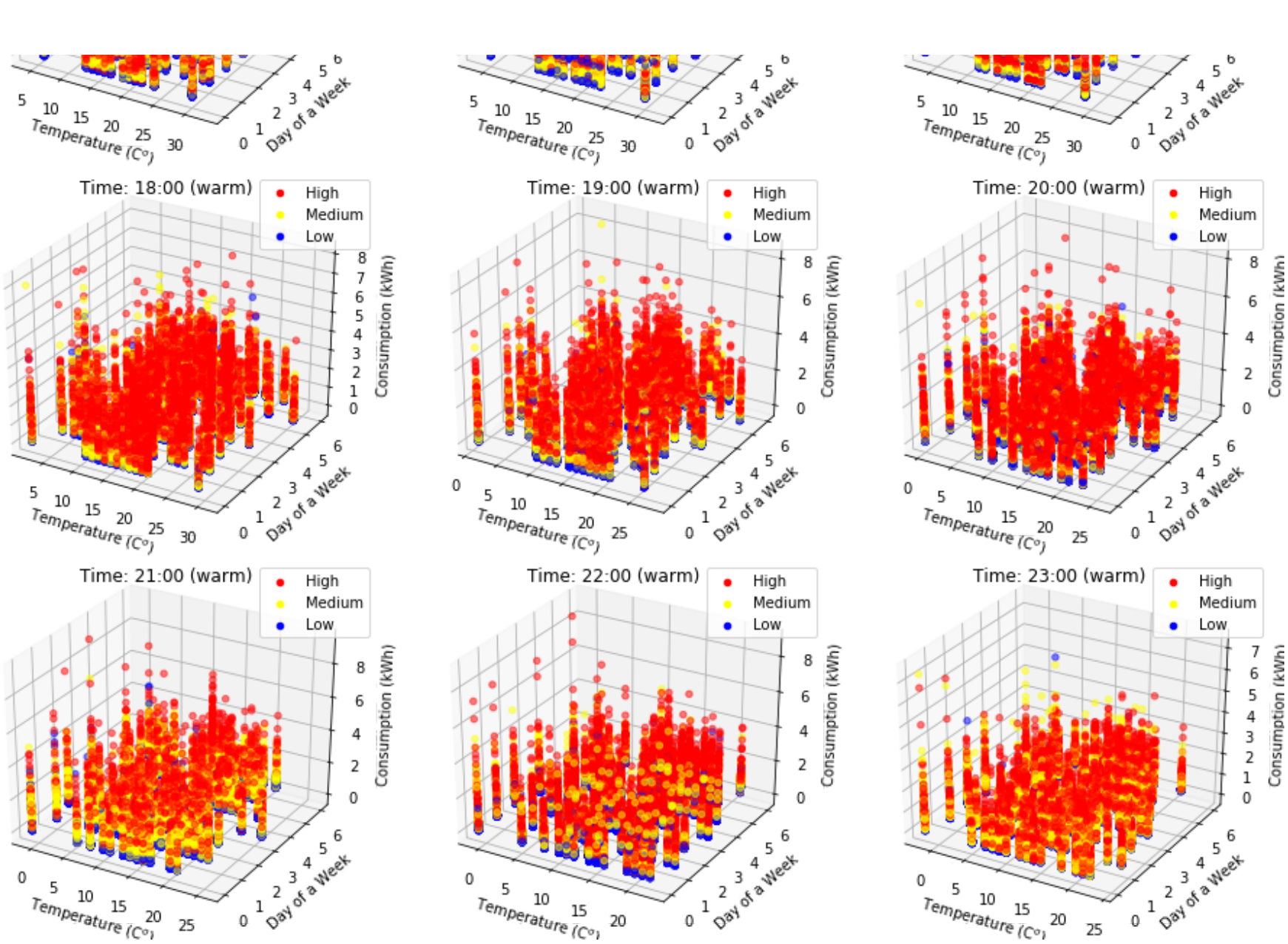

Time: 13:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)


Time: 14:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)


Time: 15:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)


Time: 16:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)


Time: 17:00 (warm)
High
Medium
Low
Temperature (C°)
Day of a Week
Consumption (kWh)



Let us deal with the classes of households. Each household should have a label vector of 24 hours.

In [31]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
n_house = 1025 # total household numbers
houseLabel = [] # store the 24hour group labels for each hourshold
for i in range(1025):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and fill in null value with mean
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
# We obtain the house Label shown below
# print(houseLabel)

# Group the house based on their 24 hour labels
houseLableDf = pd.DataFrame()
houseLableDf['House'] = df_tou1h_nf_warm.columns[1:]
houseLableDf['Label'] = houseLabel
houseGroupDf = houseLableDf.groupby('Label').size().reset_index(name='counts')
houseGroupDf = houseGroupDf.sort_values(by=['counts'], ascending=False)
houseGroupDf
```

Out[31]:

	Label	counts
153	011101211001000000001111	314
381	120010002110221121222000	18
162	011101211001000001001111	9
43	011100211001000000001111	8
498	121101211001000000001110	6
156	011101211001000000021111	6
127	011101202001000000001111	5
173	011101211001000021001111	5
170	011101211001000020001111	5
23	011100002110221121222111	4
460	121100002110221121222000	4
157	011101211001000000022111	4
485	121101002110221121222000	4
158	011101211001000000201111	4
500	121101211001000000022000	4
247	011101212101000000001111	4
151	011101211001000000001101	4
155	011101211001000000002111	4
575	202222120222112212110222	4
237	011101212001000000001111	3
202	011101211001200000001111	3
403	120010012110221121222000	3
178	011101211001000121001111	3
29	011100011001000000001111	3
161	011101211001000000222111	3
503	121101211001000001201100	2
182	011101211001001000001111	2
184	011101211001001100001111	2
185	011101211001001120001111	2
188	011101211001001121222111	2
...	...	...
235	011101211111221120001111	1
234	011101211111020020022000	1
233	011101211111000000022000	1
232	011101211111000000001111	1
231	011101211110221121222111	1
230	011101211110221121222000	1
229	011101211110221121220020	1
228	011101211110221121022001	1
227	011101211110221111222000	1
226	011101211110221000022001	1
225	011101211110201120001111	1
224	011101211101221121222000	1
223	011101211101021121201111	1
221	011101211101020001202011	1
203	011101211001200001201111	1
220	011101211101001120022111	1
219	011101211101000121222111	1
218	011101211101000021001111	1
217	011101211101000020001111	1

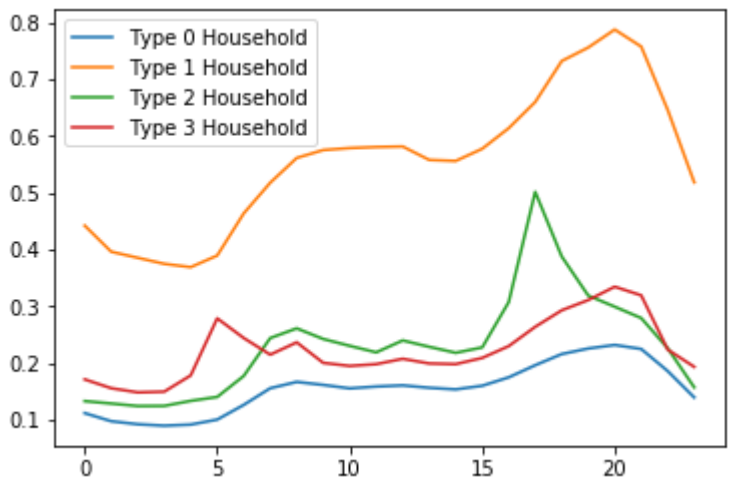
	Label	counts
216	011101211101000001001111	1
214	011101211101000000022000	1
212	011101211011200000001111	1
211	011101211010221122112001	1
210	011101211010221100222000	1
209	011101211010221001022111	1
208	011101211001221121222111	1
207	011101211001221121222011	1
205	011101211001221120001111	1
204	011101211001200001222111	1
583	221101211110221121222002	1

584 rows × 2 columns

```
<Figure size 936x6480 with 0 Axes>
```

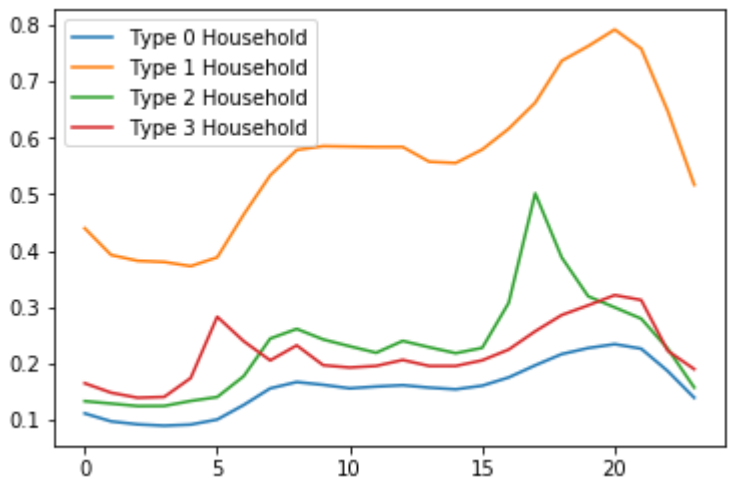
Note that the generated labels for the first 3 groups may change a bit each time, this is because some group of points which are very close to each other also have equal distance two both cluster centers, which leads to different label names. However, this won't affect the group property, we will show we run it for 100 times, and average 24 hour load curve of every time will be ploted. And you will see, the average load curve almost doesn't change. So label names wouldn't be an important concern here, we more care about the property of groups themselves. In addition, from the above result we can see the first 4 groups each has at least around 10 household in it. Next what we want to do is to plot the load property of each group, and see how does there average 24 hour load look like.

In [32]:
```
for i in range(4):
    load = []
    for j in range(24):
        if j <= 9:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
        else:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
    plt.plot(load, label = 'Type ' + str(i) + ' Household')
plt.legend()
plt.show()
```



See if the shape is sensitive to initialization, run it again:

In [30]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
n_house = 1025 # total household numbers
houseLabel = [] # store the 24hour group labels for each hoursehold
for i in range(1025):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and fill in null value with mean
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
# We obtain the house Label shown below
# print(houseLabel)

# Group the house based on their 24 hour labels
houseLableDf = pd.DataFrame()
houseLableDf['House'] = df_tou1h_nf_warm.columns[1:]
houseLableDf['Label'] = houseLabel
houseGroupDf = houseLableDf.groupby('Label').size().reset_index(name='counts')
houseGroupDf = houseGroupDf.sort_values(by=['counts'], ascending=False)

for i in range(4):
    load = []
    for j in range(24):
        if j <= 9:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
        else:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
    plt.plot(load, label = 'Type ' + str(i) + ' Household')
plt.legend()
plt.show()
```



Some observations and thoughts of the above daily load curves:

1) the type 0 household with the most amount of households has a relatively smooth and low consumption during a day, they consume the most amount of energy around 20:00, and consume least at 02:00-05:00.

2) type 1 household is high energy consumption households, besides of all the consumption is around 4 times the consumption of type 0 group, they have a relatively similar trend of energy usage pattern peak at 20:00 and valley at 02:00-05:00

3) type 2 household has a special usage pattern where their peak happens at 17:00 (maybe they come back home earlier), other parts about the same.

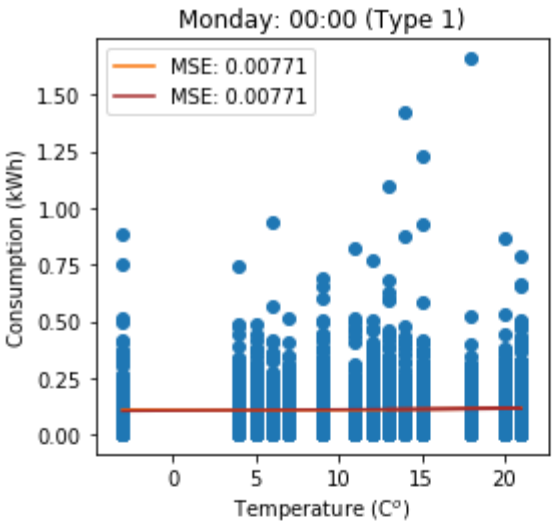
4) type 3 household also has a special usage pattern, where their peak happens at 05:00 (maybe they work in the morning, at home with some works that consumes a lot of energy, maybe breakfast, maybe heating at every morning?) and 20:00

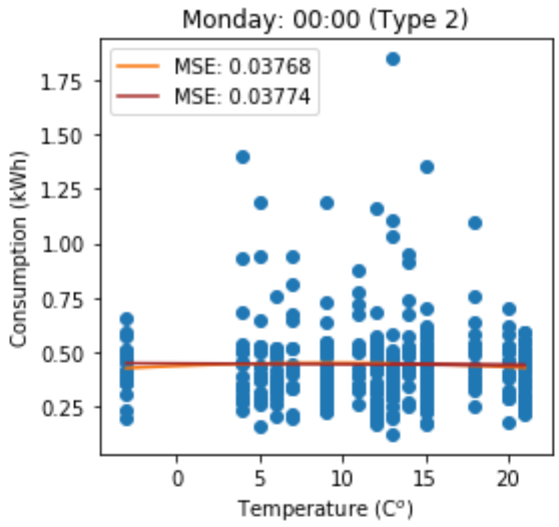
3. Multiple Regression

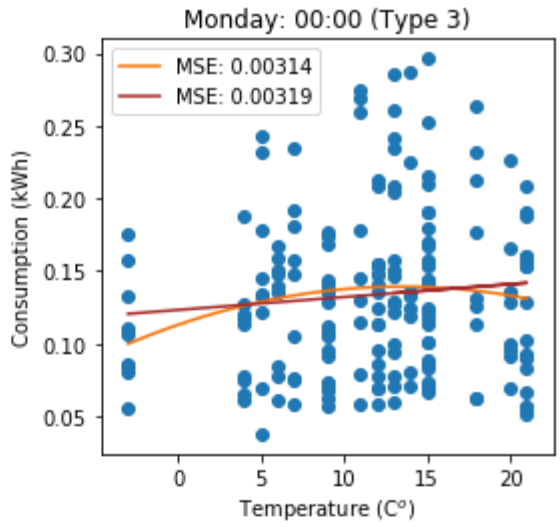
Multiple Regression is to predict the consumption of the group in event (pricing change) time periods, so that we can analyze the price responsivness later.

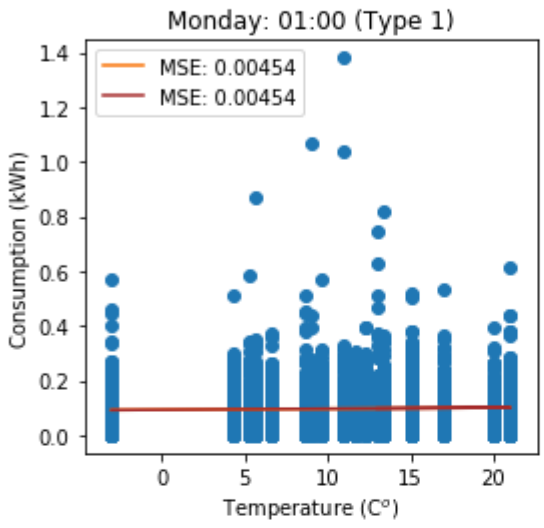
It is clear that the temperature and days of a weeks are kinda correlated with the the first 3 PCA coefficients, we are using them as the features to do multiple regression (for each hour, temperature and days of week do regression).

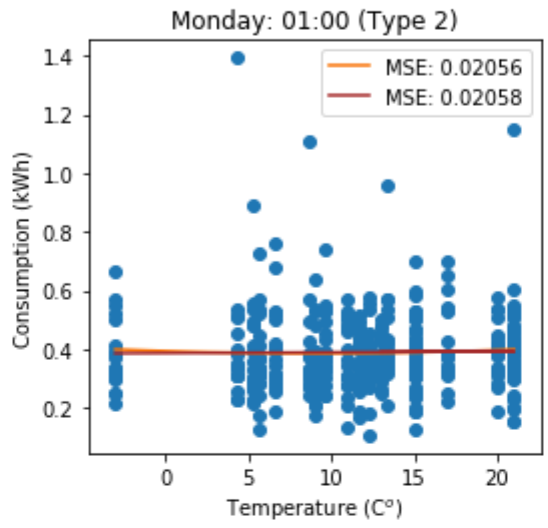
In [33]:
```
# Monday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 0 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Monday
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Monday
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

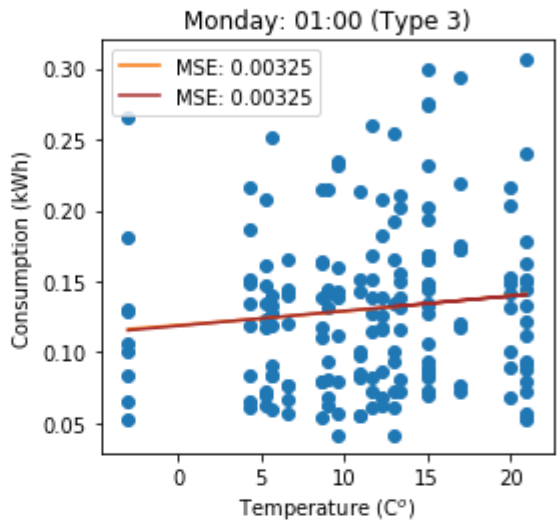

Monday: 00:00 (Type 1)
MSE: 0.00771
MSE: 0.00771
Consumption (kWh)
Temperature (C°)

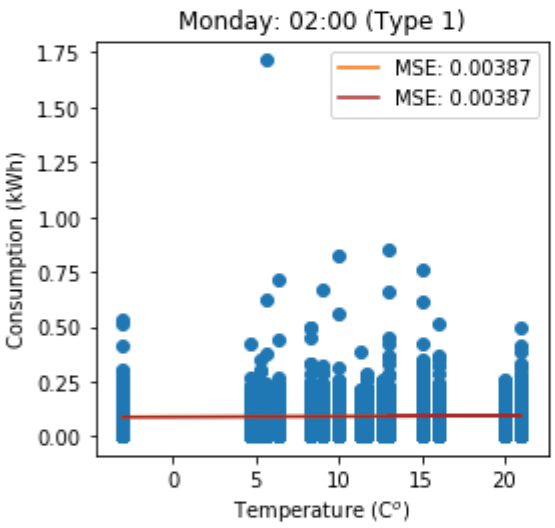

Monday: 00:00 (Type 2)
MSE: 0.03768
MSE: 0.03774
Consumption (kWh)
Temperature (C°)

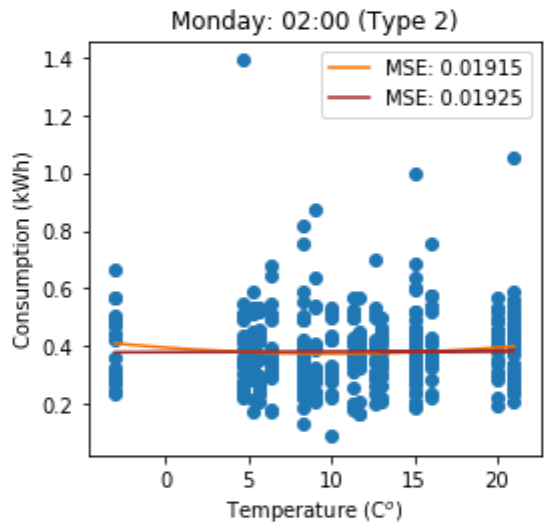

Monday: 00:00 (Type 3)
MSE: 0.00314
MSE: 0.00319
Consumption (kWh)
Temperature (C°)

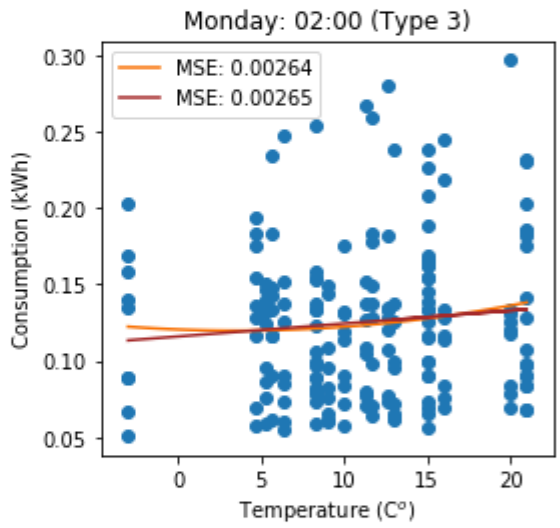

Monday: 01:00 (Type 1)
MSE: 0.00454
MSE: 0.00454
Consumption (kWh)
Temperature (C°)

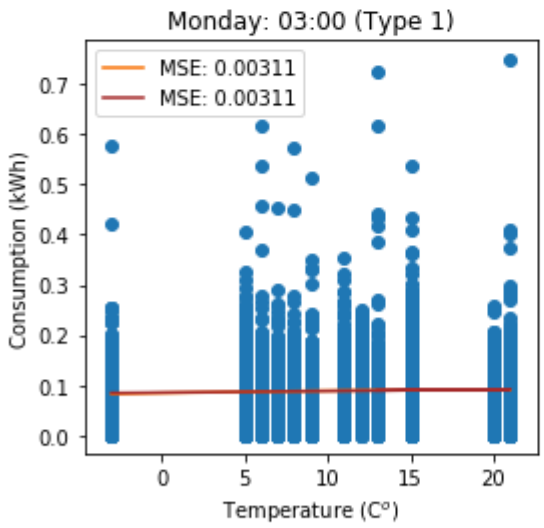

Monday: 01:00 (Type 2)
MSE: 0.02056
MSE: 0.02058
Consumption (kWh)
Temperature (C°)

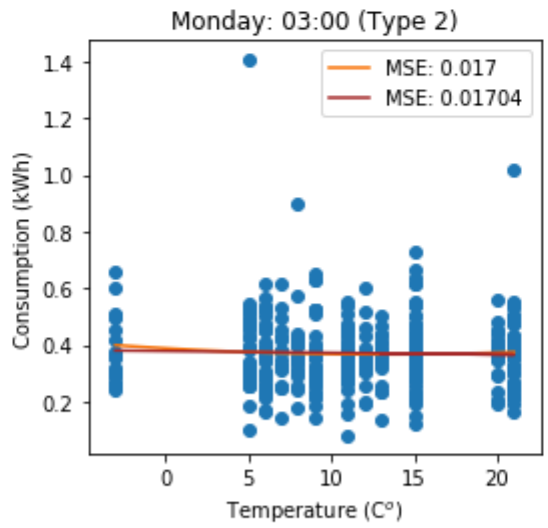

Monday: 01:00 (Type 3)
MSE: 0.00325
MSE: 0.00325
Consumption (kWh)
Temperature (C°)

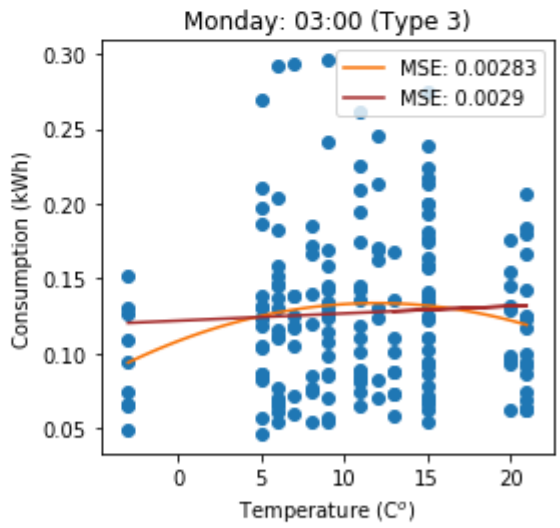

Monday: 02:00 (Type 1)
MSE: 0.00387
MSE: 0.00387
Consumption (kWh)
Temperature (C°)

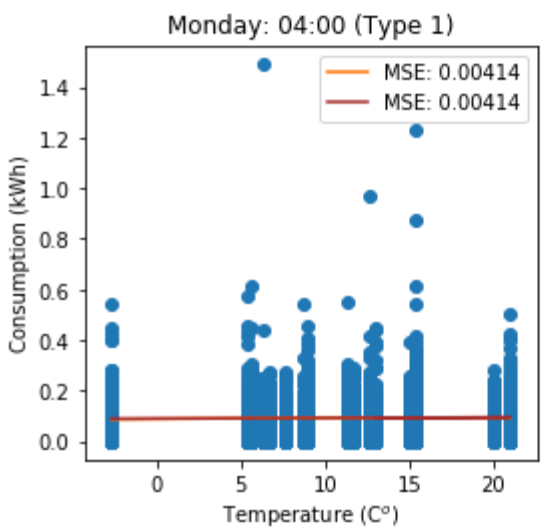

Monday: 02:00 (Type 2)
MSE: 0.01915
MSE: 0.01925
Consumption (kWh)
Temperature (C°)

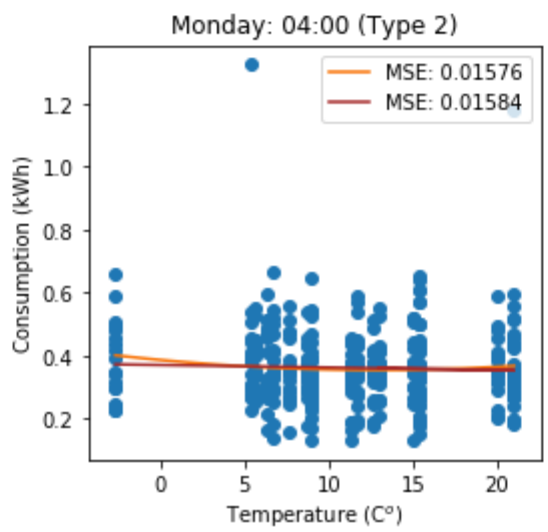

Monday: 02:00 (Type 3)
MSE: 0.00264
MSE: 0.00265
Consumption (kWh)
Temperature (C°)

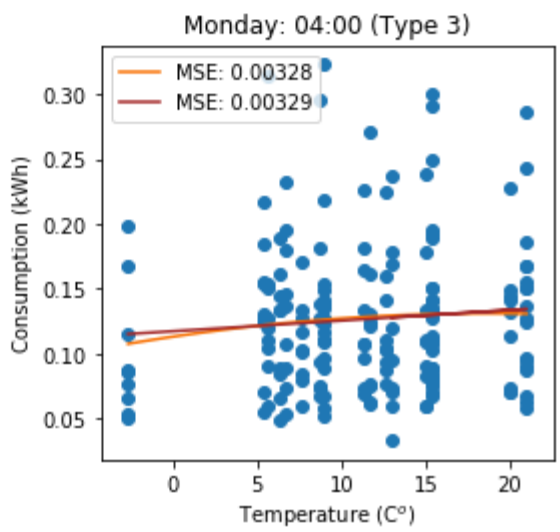

Monday: 03:00 (Type 1)
MSE: 0.00311
MSE: 0.00311
Consumption (kWh)
Temperature (C°)

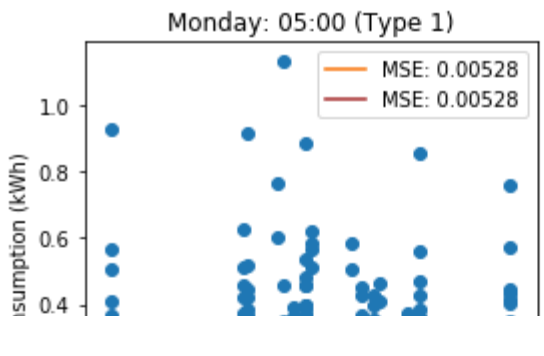

Monday: 03:00 (Type 2)
MSE: 0.017
MSE: 0.01704
Consumption (kWh)
Temperature (C°)

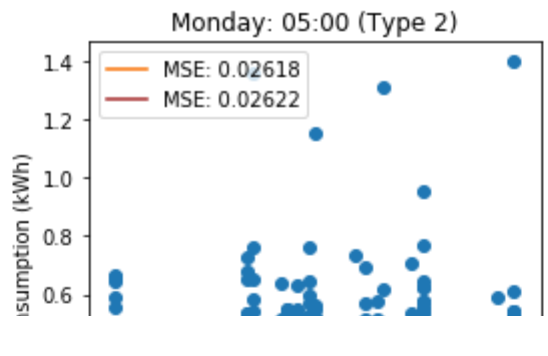

Monday: 03:00 (Type 3)
MSE: 0.00283
MSE: 0.0029
Consumption (kWh)
Temperature (C°)

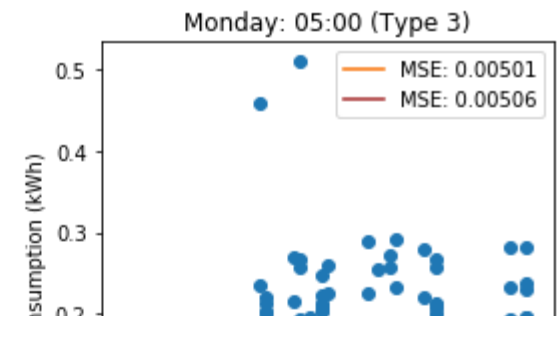

Monday: 04:00 (Type 1)
MSE: 0.00414
MSE: 0.00414
Consumption (kWh)
Temperature (C°)

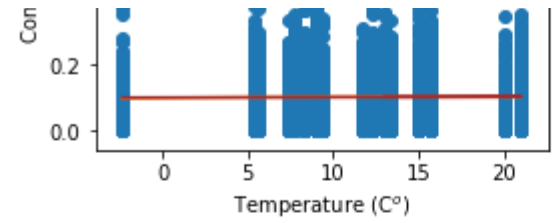

Monday: 04:00 (Type 2)
MSE: 0.01576
MSE: 0.01584
Consumption (kWh)
Temperature (C°)

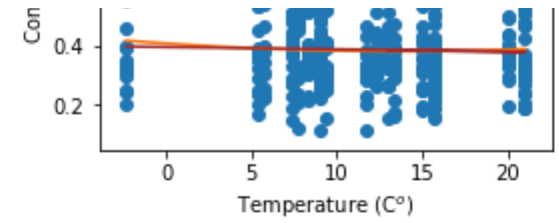

Monday: 04:00 (Type 3)
MSE: 0.00328
MSE: 0.00329
Consumption (kWh)
Temperature (C°)

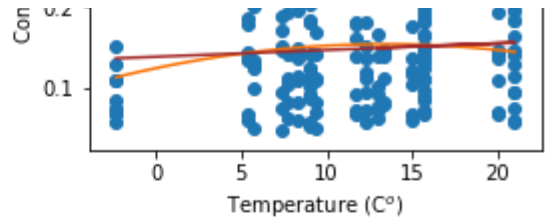

Monday: 05:00 (Type 1)
MSE: 0.00528
MSE: 0.00528

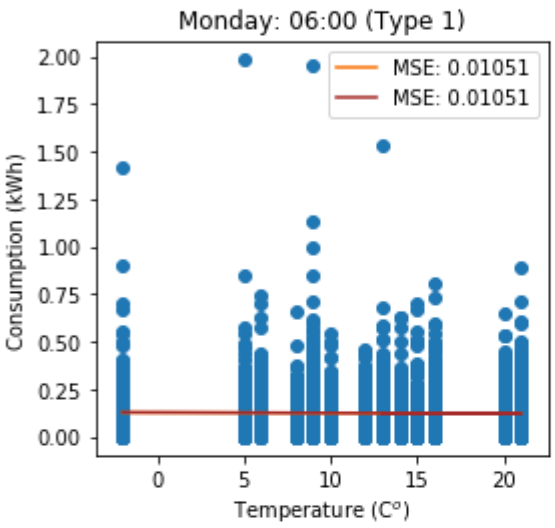

Monday: 05:00 (Type 2)
MSE: 0.02618
MSE: 0.02622

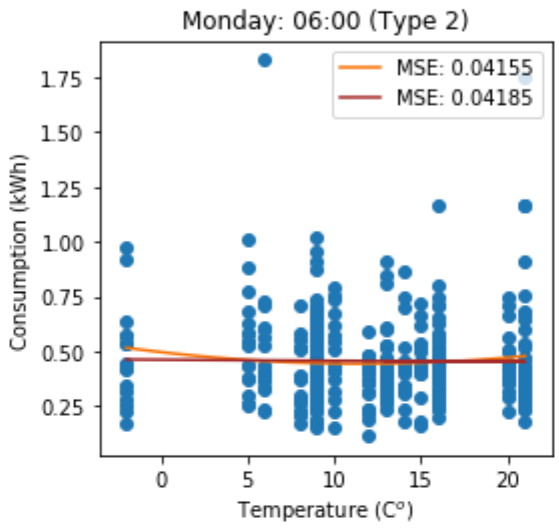

Monday: 05:00 (Type 3)
MSE: 0.00501
MSE: 0.00506

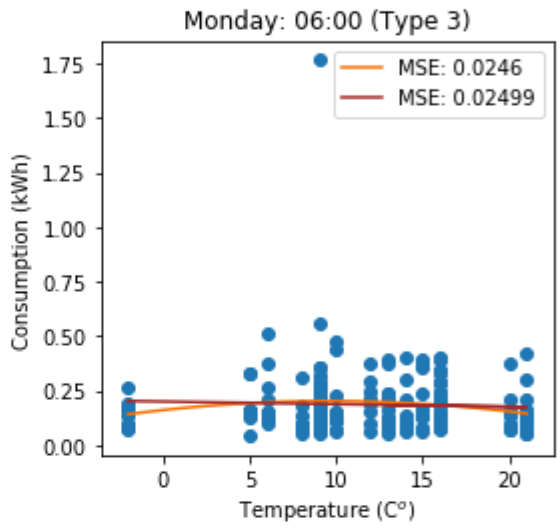

Con
0.2
0.0
0
5
10
15
20
Temperature (C°)

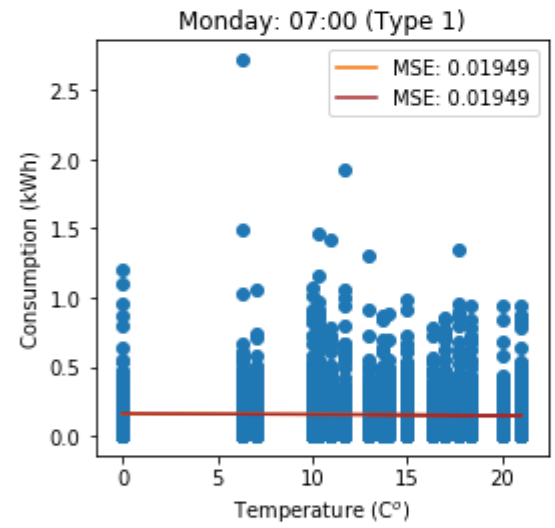

Con
0.4
0.2
0
5
10
15
20
Temperature (C°)

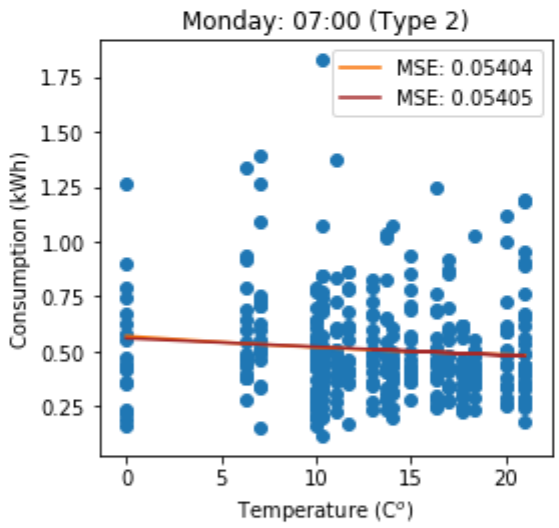

Con
0.1
0
5
10
15
20
Temperature (C°)

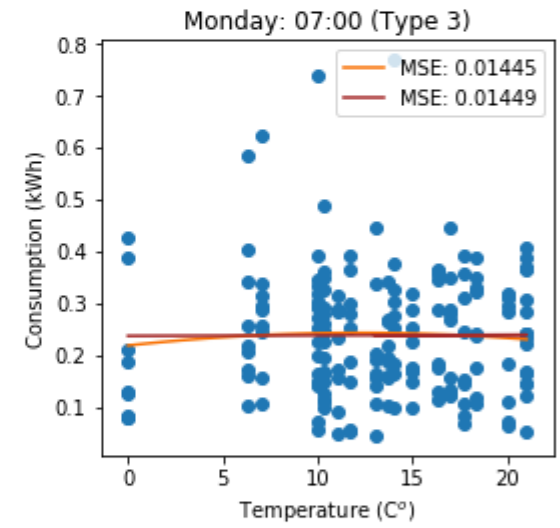

Monday: 06:00 (Type 1)
MSE: 0.01051
MSE: 0.01051
Consumption (kWh)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
Temperature (C°)

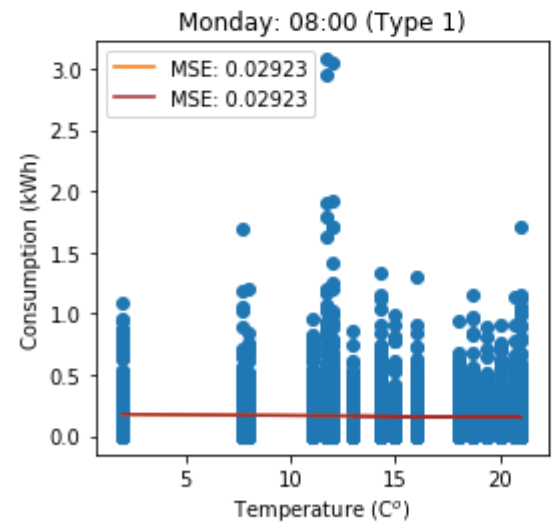

Monday: 06:00 (Type 2)
MSE: 0.04155
MSE: 0.04185
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
5
10
15
20
Temperature (C°)

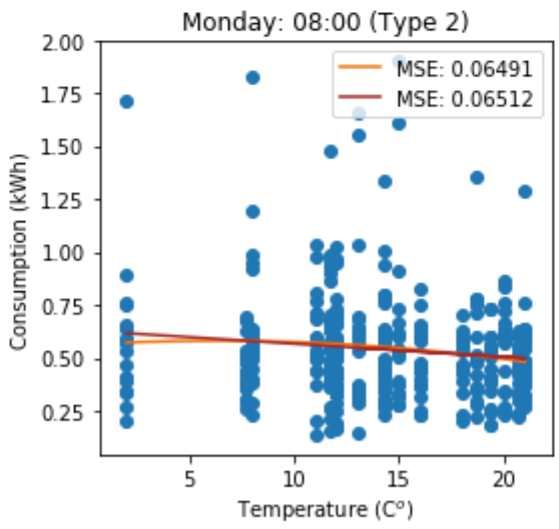

Monday: 06:00 (Type 3)
MSE: 0.0246
MSE: 0.02499
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
Temperature (C°)

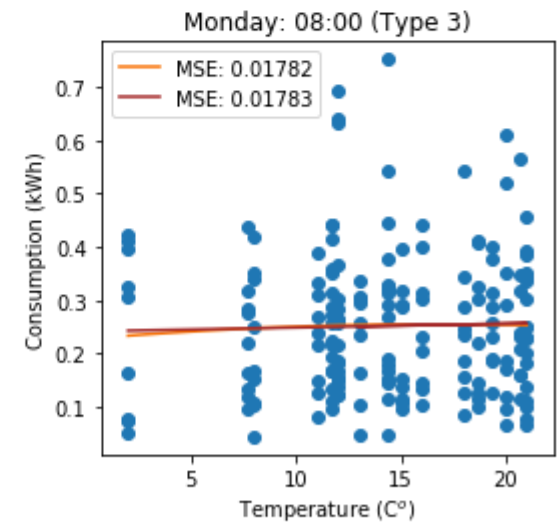

Monday: 07:00 (Type 1)
MSE: 0.01949
MSE: 0.01949
Consumption (kWh)
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
Temperature (C°)

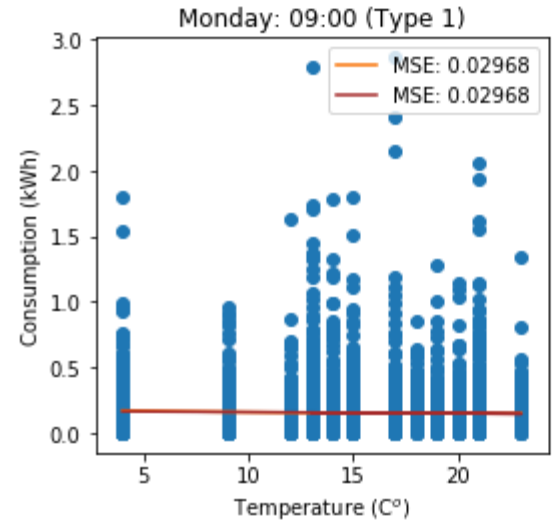

Monday: 07:00 (Type 2)
MSE: 0.05404
MSE: 0.05405
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
5
10
15
20
Temperature (C°)

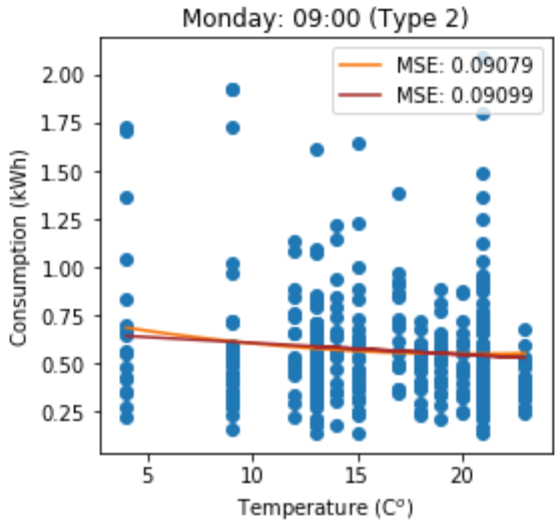

Monday: 07:00 (Type 3)
MSE: 0.01445
MSE: 0.01449
Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
Temperature (C°)

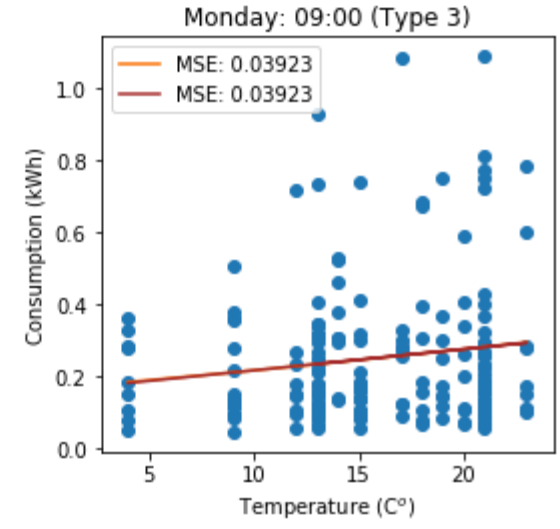

Monday: 08:00 (Type 1)
MSE: 0.02923
MSE: 0.02923
Consumption (kWh)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
Temperature (C°)

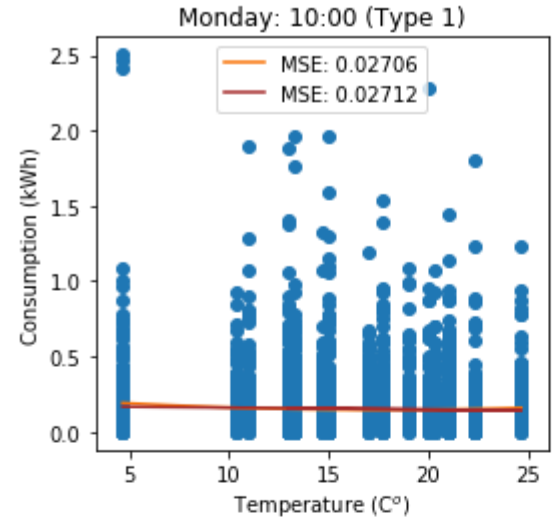

Monday: 08:00 (Type 2)
MSE: 0.06491
MSE: 0.06512
Consumption (kWh)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
5
10
15
20
Temperature (C°)

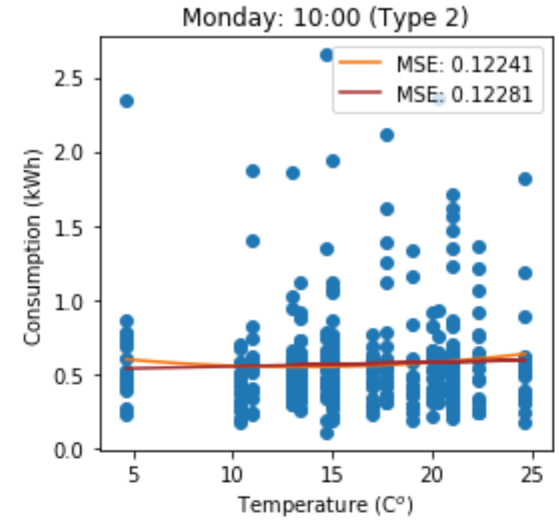

Monday: 08:00 (Type 3)
MSE: 0.01782
MSE: 0.01783
Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
5
10
15
20
Temperature (C°)

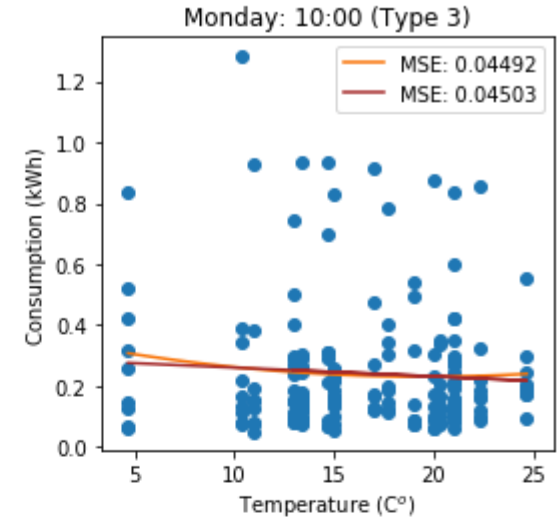

Monday: 09:00 (Type 1)
MSE: 0.02968
MSE: 0.02968
Consumption (kWh)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
Temperature (C°)


Monday: 09:00 (Type 2)
MSE: 0.09079
MSE: 0.09099
Consumption (kWh)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
5
10
15
20
Temperature (C°)


Monday: 09:00 (Type 3)
MSE: 0.03923
MSE: 0.03923
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
Temperature (C°)


Monday: 10:00 (Type 1)
MSE: 0.02706
MSE: 0.02712
Consumption (kWh)
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
25
Temperature (C°)

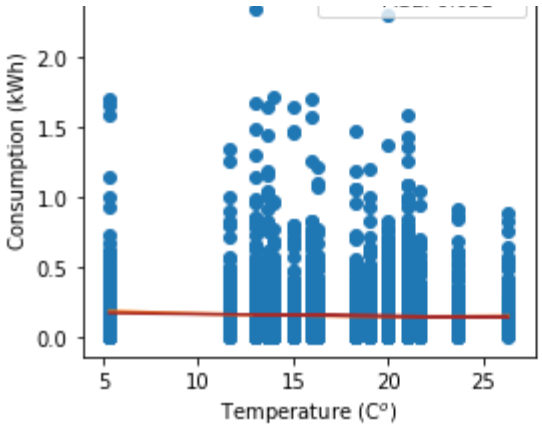

Monday: 10:00 (Type 2)
MSE: 0.12241
MSE: 0.12281
Consumption (kWh)
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
25
Temperature (C°)

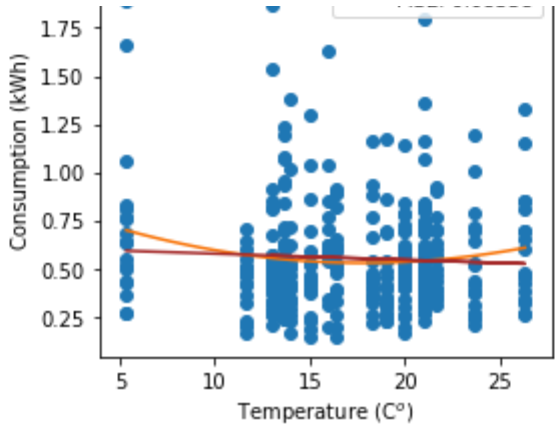

Monday: 10:00 (Type 3)
MSE: 0.04492
MSE: 0.04503
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
Temperature (C°)

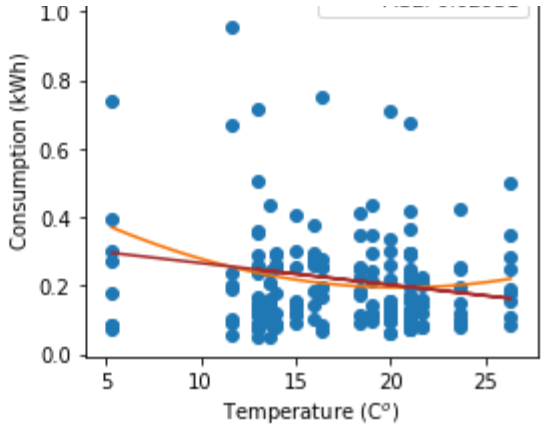

Monday: 11:00 (Type 1)
MSE: 0.03098
MSE: 0.031
2.5

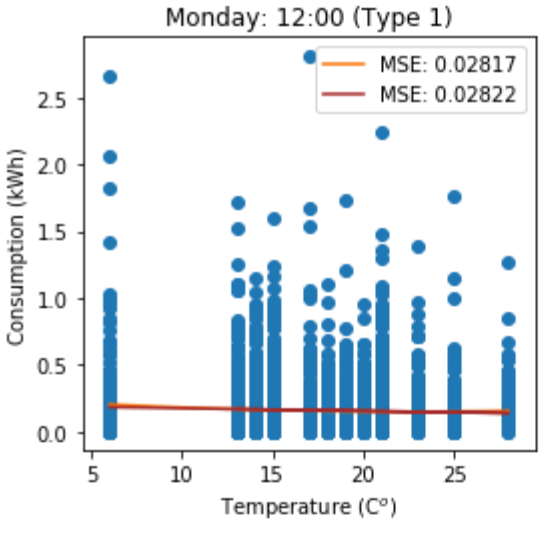

Monday: 11:00 (Type 2)
MSE: 0.0844
MSE: 0.08558
2.00

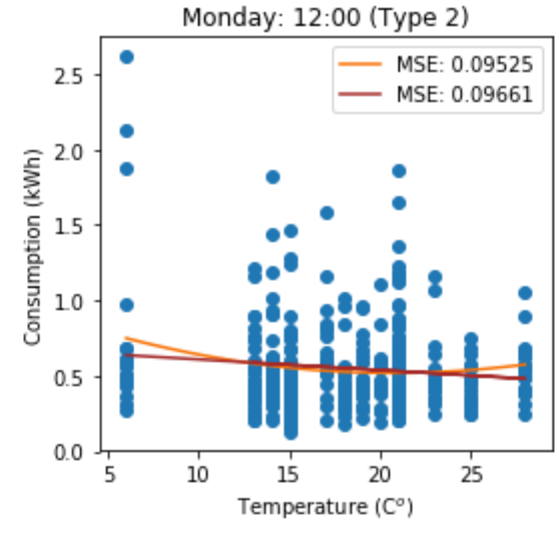

Monday: 11:00 (Type 3)
MSE: 0.02874
MSE: 0.02931
1.2

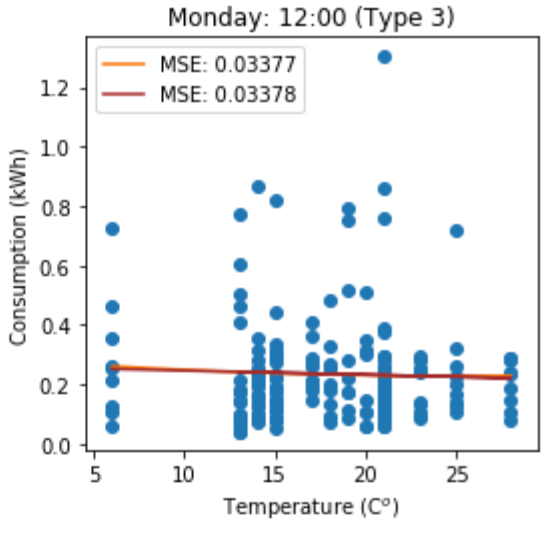

Consumption (kWh)
Temperature (C°)

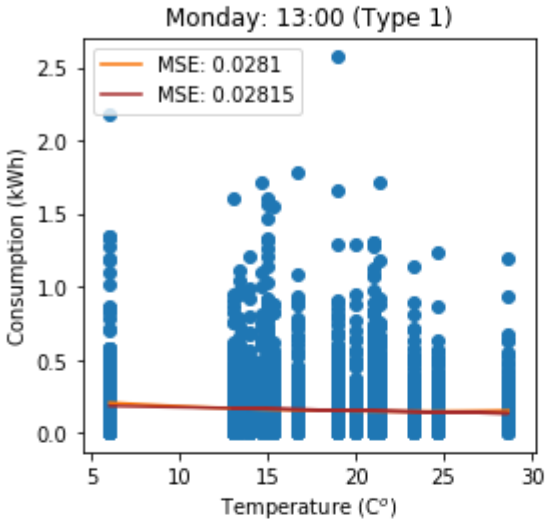

Consumption (kWh)
Temperature (C°)

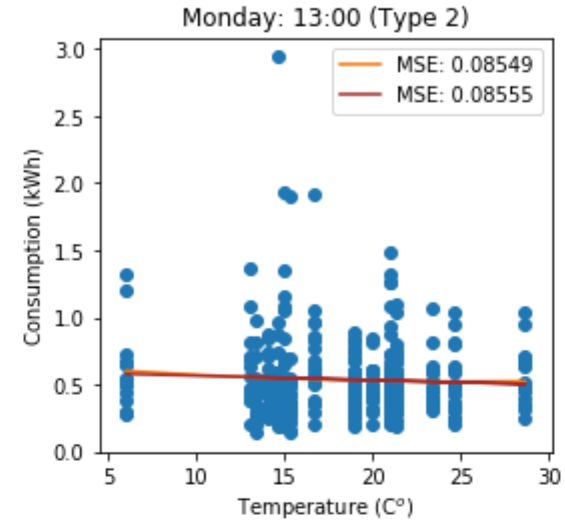

Consumption (kWh)
Temperature (C°)

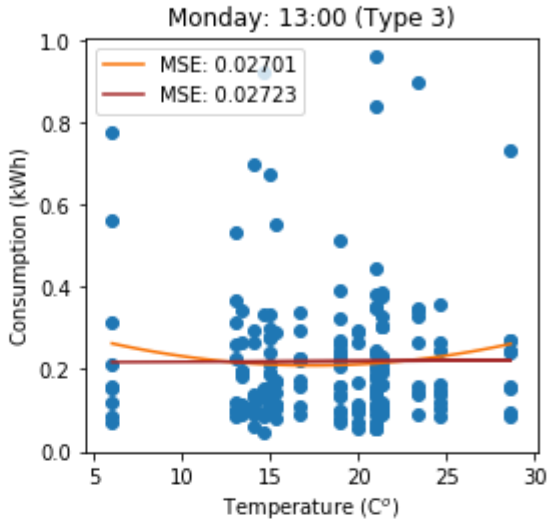

Monday: 12:00 (Type 1)
MSE: 0.02817
MSE: 0.02822
Consumption (kWh)
Temperature (C°)

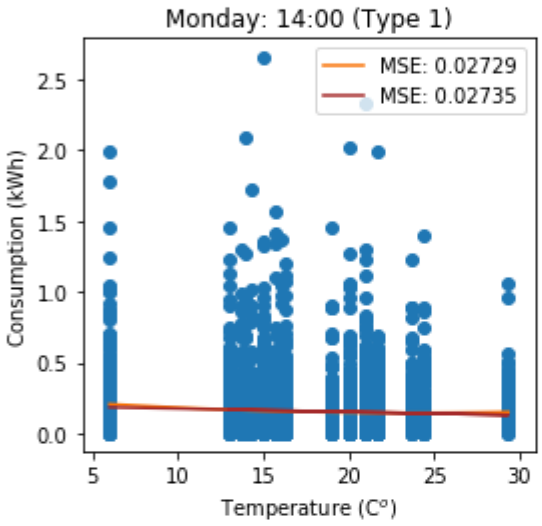

Monday: 12:00 (Type 2)
MSE: 0.09525
MSE: 0.09661
Consumption (kWh)
Temperature (C°)

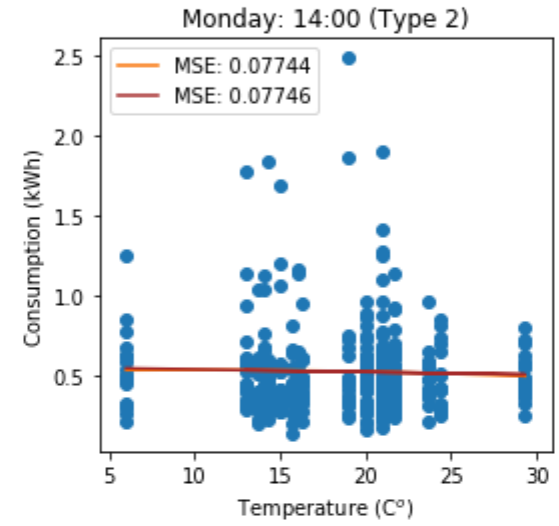

Monday: 12:00 (Type 3)
MSE: 0.03377
MSE: 0.03378
Consumption (kWh)
Temperature (C°)

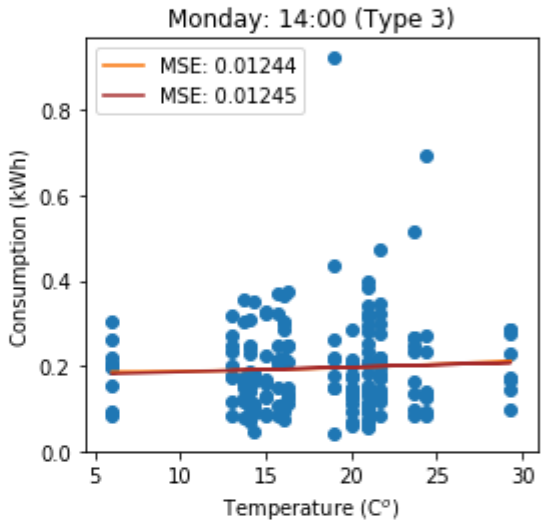

Monday: 13:00 (Type 1)
MSE: 0.0281
MSE: 0.02815
Consumption (kWh)
Temperature (C°)

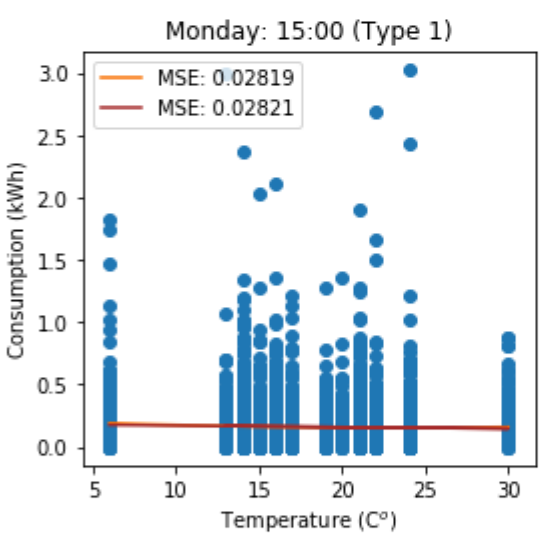

Monday: 13:00 (Type 2)
MSE: 0.08549
MSE: 0.08555
Consumption (kWh)
Temperature (C°)

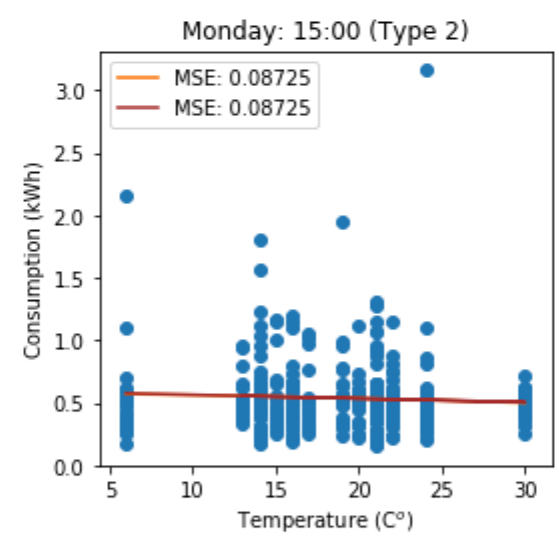

Monday: 13:00 (Type 3)
MSE: 0.02701
MSE: 0.02723
Consumption (kWh)
Temperature (C°)

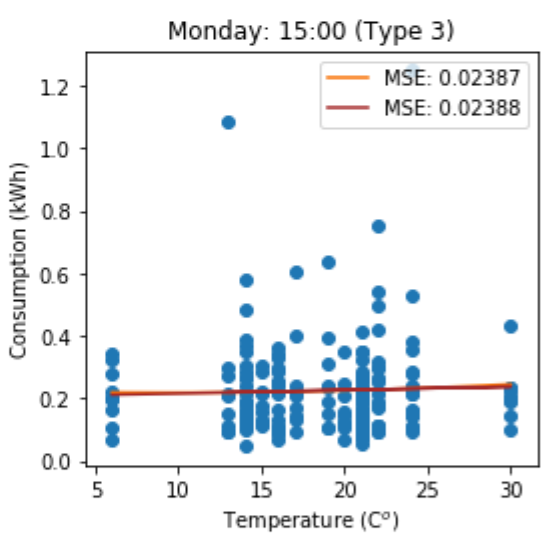

Monday: 14:00 (Type 1)
MSE: 0.02729
MSE: 0.02735
Consumption (kWh)
Temperature (C°)

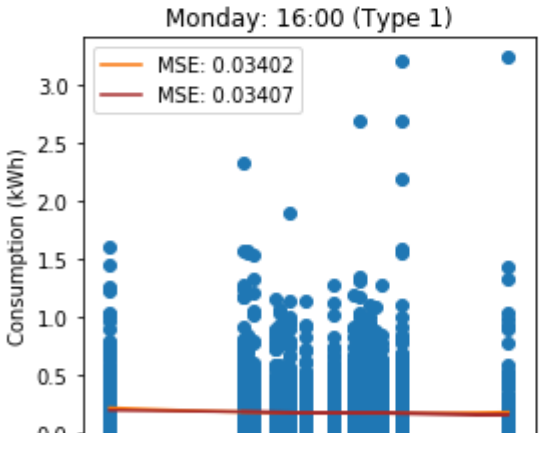

Monday: 14:00 (Type 2)
MSE: 0.07744
MSE: 0.07746
Consumption (kWh)
Temperature (C°)

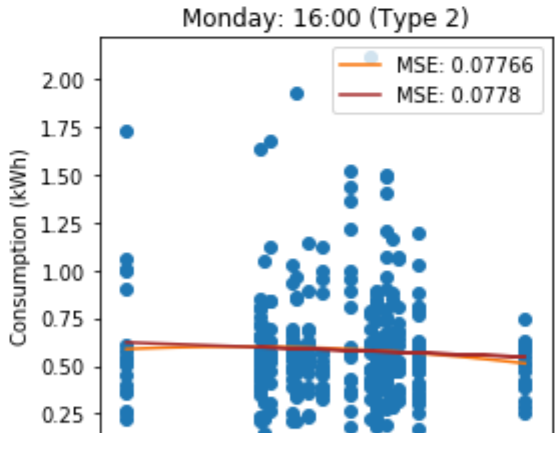

Monday: 14:00 (Type 3)
MSE: 0.01244
MSE: 0.01245
Consumption (kWh)
Temperature (C°)

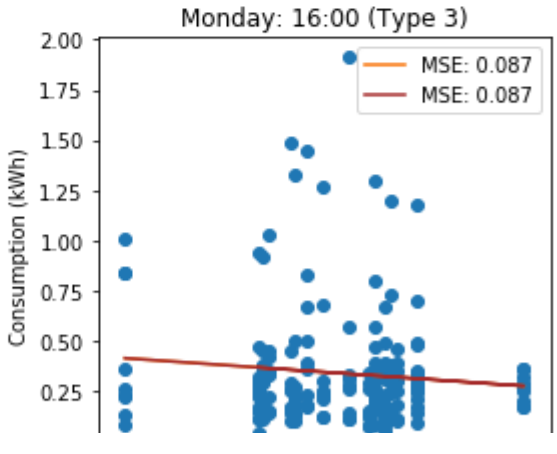

Monday: 15:00 (Type 1)
MSE: 0.02819
MSE: 0.02821
Consumption (kWh)
Temperature (C°)

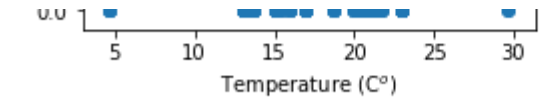

Monday: 15:00 (Type 2)
MSE: 0.08725
MSE: 0.08725
Consumption (kWh)
Temperature (C°)

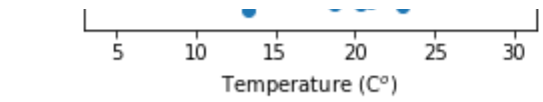

Monday: 15:00 (Type 3)
MSE: 0.02387
MSE: 0.02388
Consumption (kWh)
Temperature (C°)

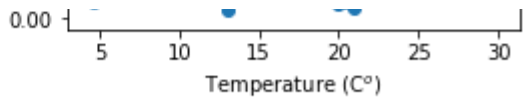

Monday: 16:00 (Type 1)
MSE: 0.03402
MSE: 0.03407
Consumption (kWh)

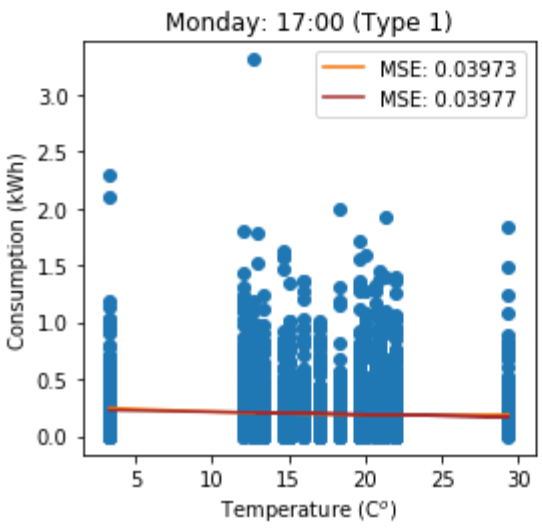

Monday: 16:00 (Type 2)
MSE: 0.07766
MSE: 0.0778
Consumption (kWh)

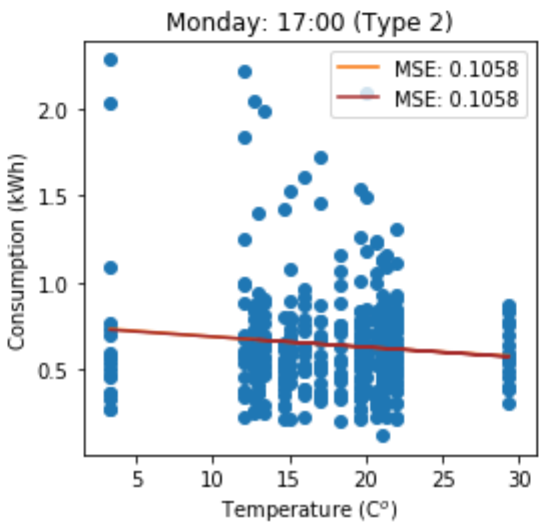

Monday: 16:00 (Type 3)
MSE: 0.087
MSE: 0.087
Consumption (kWh)

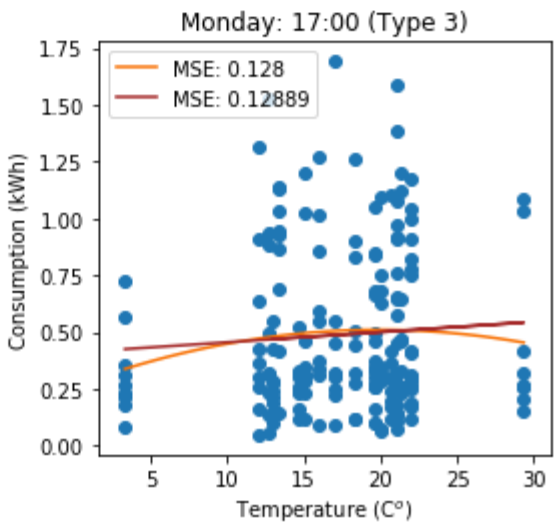

Temperature (C°)

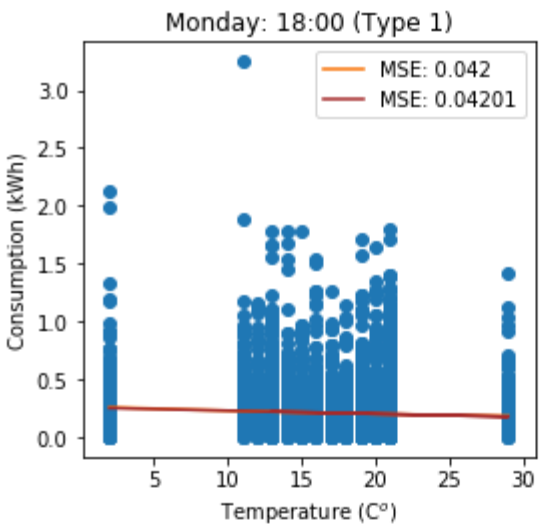

Temperature (C°)

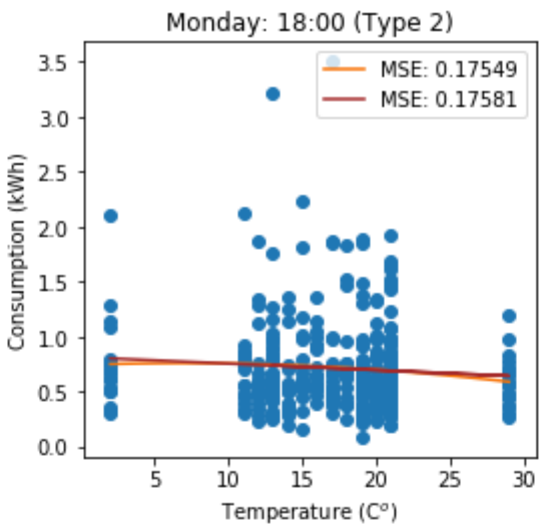

Temperature (C°)

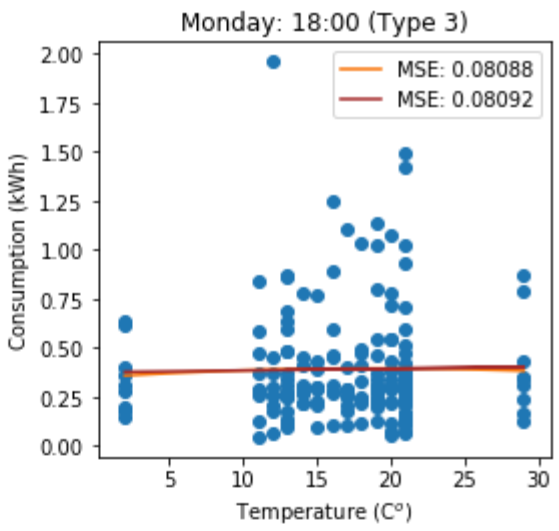

Monday: 17:00 (Type 1)
MSE: 0.03973
MSE: 0.03977
Consumption (kWh)
Temperature (C°)

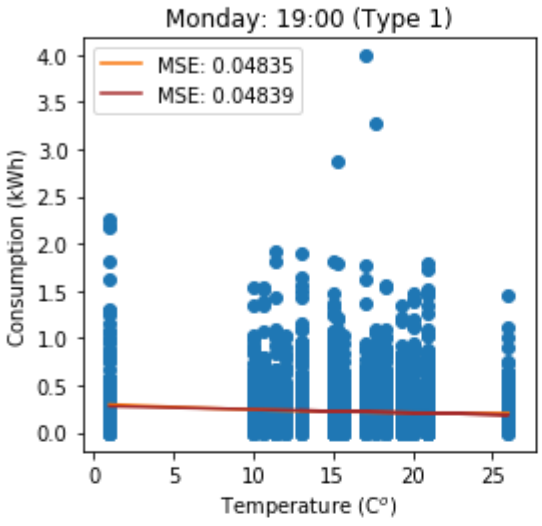

Monday: 17:00 (Type 2)
MSE: 0.1058
MSE: 0.1058
Consumption (kWh)
Temperature (C°)

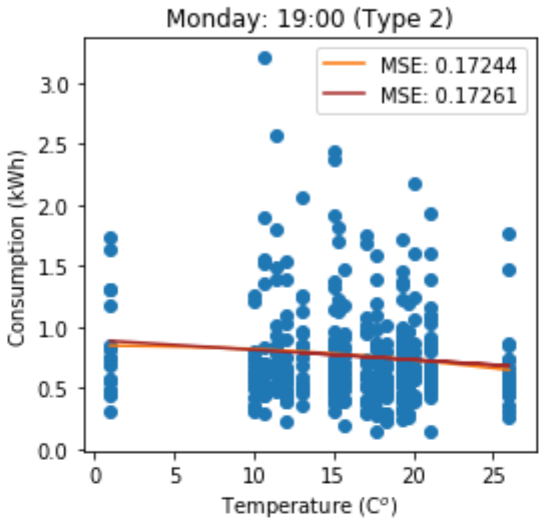

Monday: 17:00 (Type 3)
MSE: 0.128
MSE: 0.12889
Consumption (kWh)
Temperature (C°)

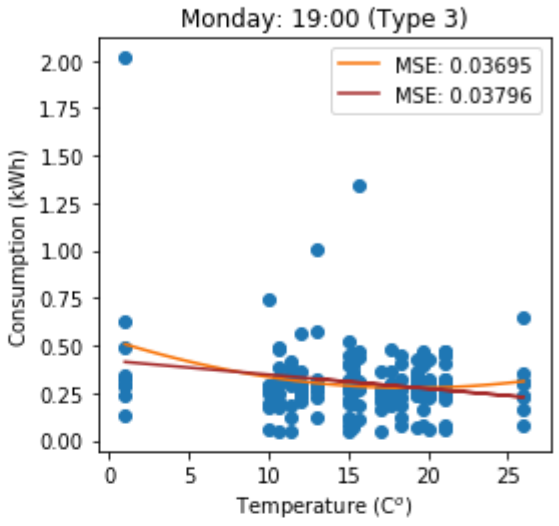

Monday: 18:00 (Type 1)
MSE: 0.042
MSE: 0.04201
Consumption (kWh)
Temperature (C°)

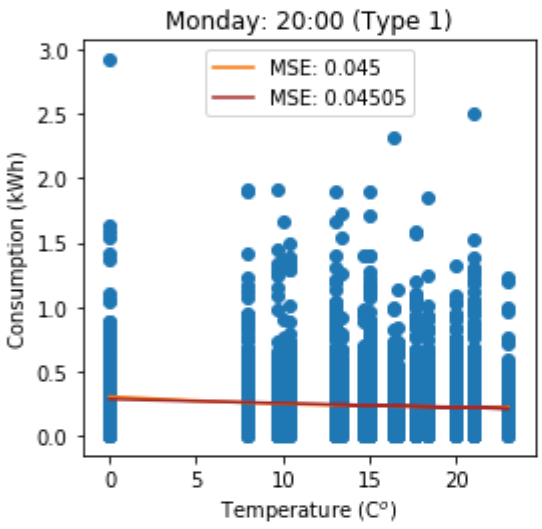

Monday: 18:00 (Type 2)
MSE: 0.17549
MSE: 0.17581
Consumption (kWh)
Temperature (C°)

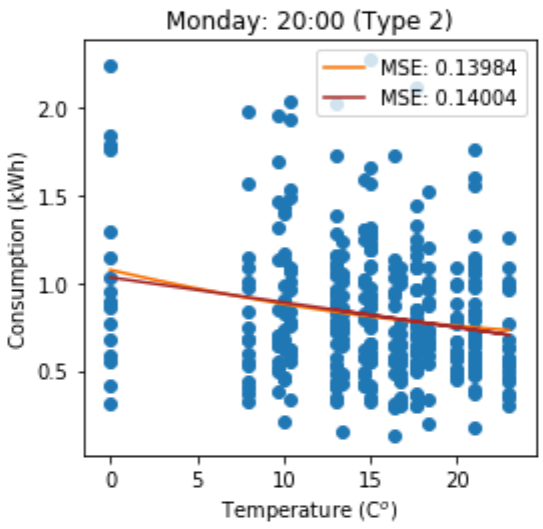

Monday: 18:00 (Type 3)
MSE: 0.08088
MSE: 0.08092
Consumption (kWh)
Temperature (C°)

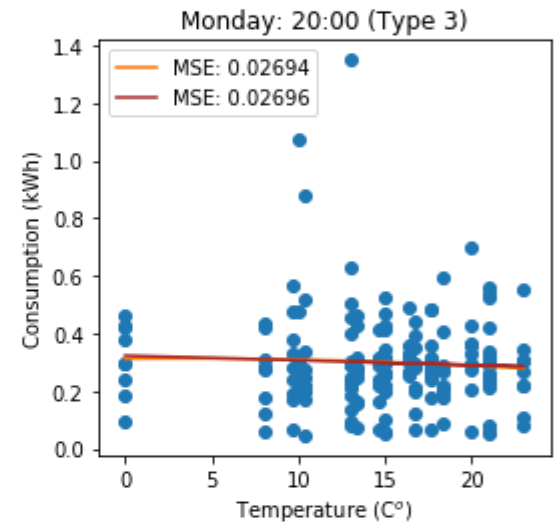

Monday: 19:00 (Type 1)
MSE: 0.04835
MSE: 0.04839
Consumption (kWh)
Temperature (C°)

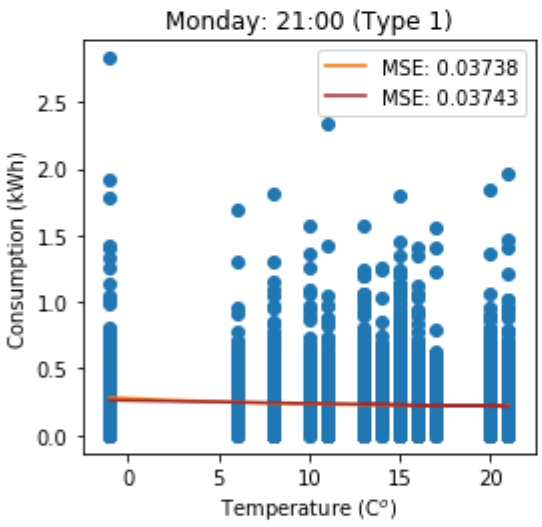

Monday: 19:00 (Type 2)
MSE: 0.17244
MSE: 0.17261
Consumption (kWh)
Temperature (C°)

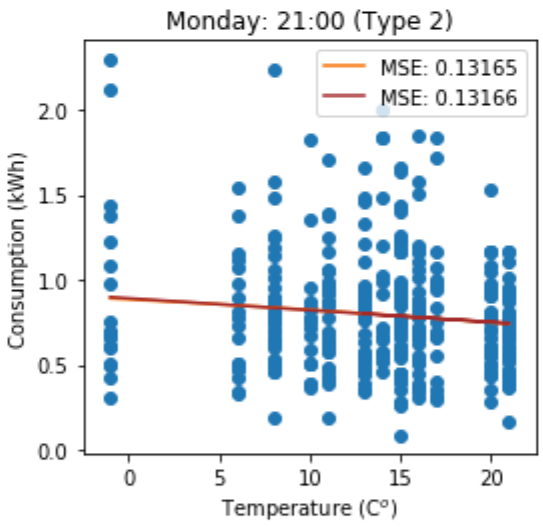

Monday: 19:00 (Type 3)
MSE: 0.03695
MSE: 0.03796
Consumption (kWh)
Temperature (C°)

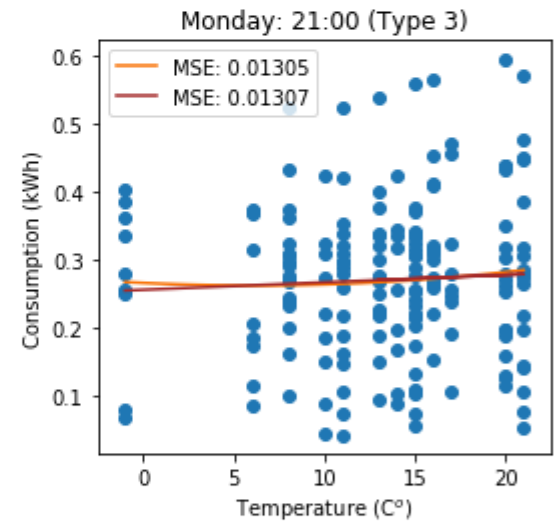

Monday: 20:00 (Type 1)
MSE: 0.045
MSE: 0.04505
Consumption (kWh)
Temperature (C°)

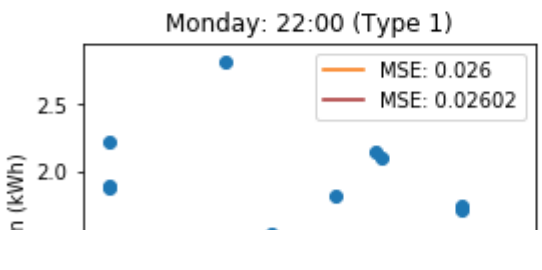

Monday: 20:00 (Type 2)
MSE: 0.13984
MSE: 0.14004
Consumption (kWh)
Temperature (C°)

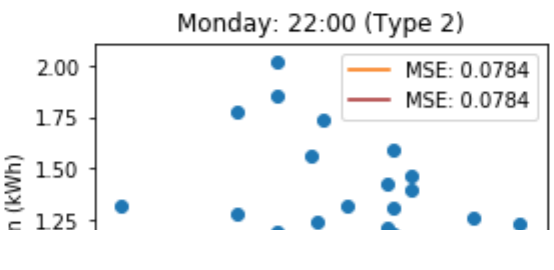

Monday: 20:00 (Type 3)
MSE: 0.02694
MSE: 0.02696
Consumption (kWh)
Temperature (C°)

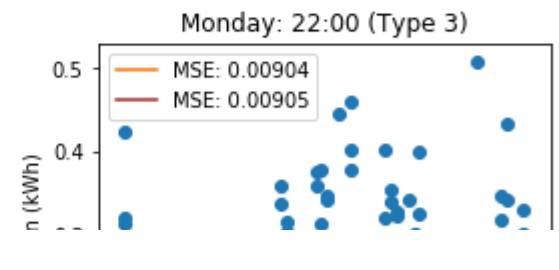

Monday: 21:00 (Type 1)
MSE: 0.03738
MSE: 0.03743
Consumption (kWh)
Temperature (C°)

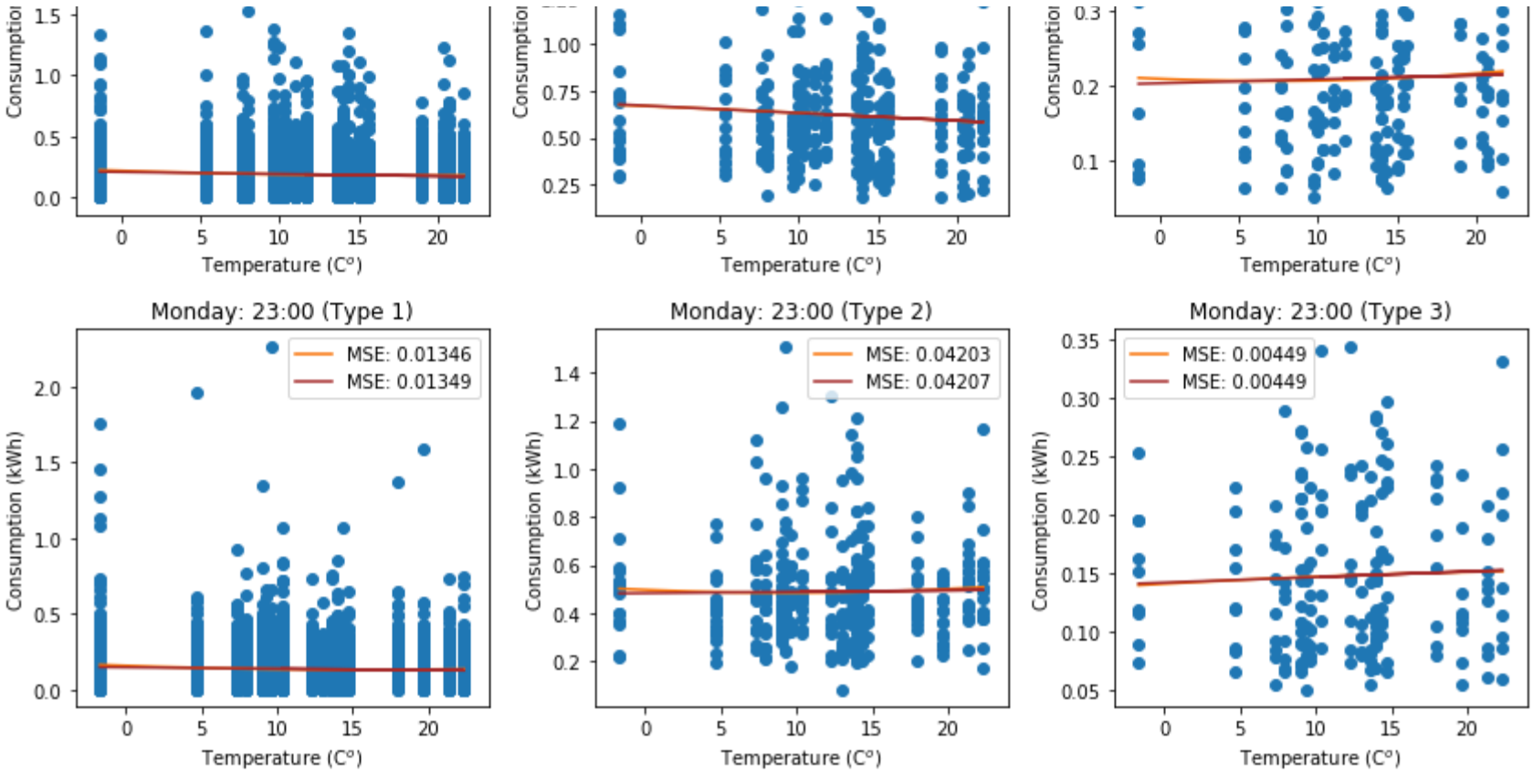

Monday: 21:00 (Type 2)
MSE: 0.13165
MSE: 0.13166
Consumption (kWh)
Temperature (C°)


Monday: 21:00 (Type 3)
MSE: 0.01305
MSE: 0.01307
Consumption (kWh)
Temperature (C°)

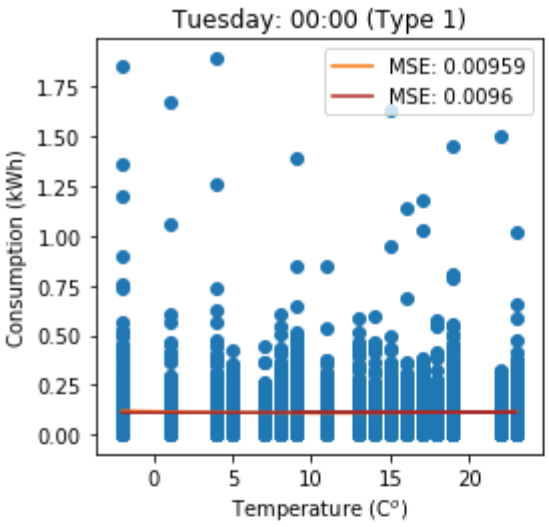

Monday: 22:00 (Type 1)
MSE: 0.026
MSE: 0.02602

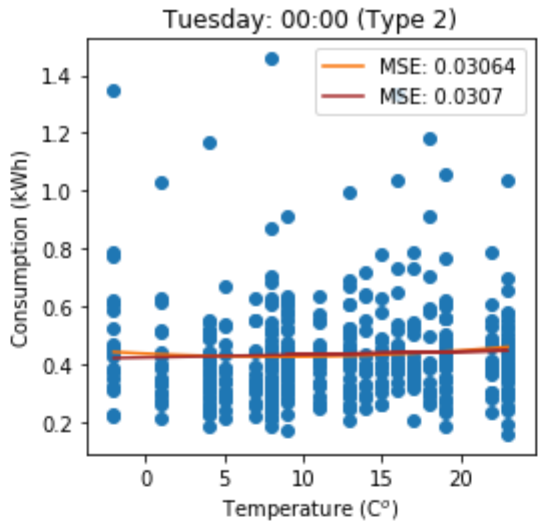

Monday: 22:00 (Type 2)
MSE: 0.0784
MSE: 0.0784

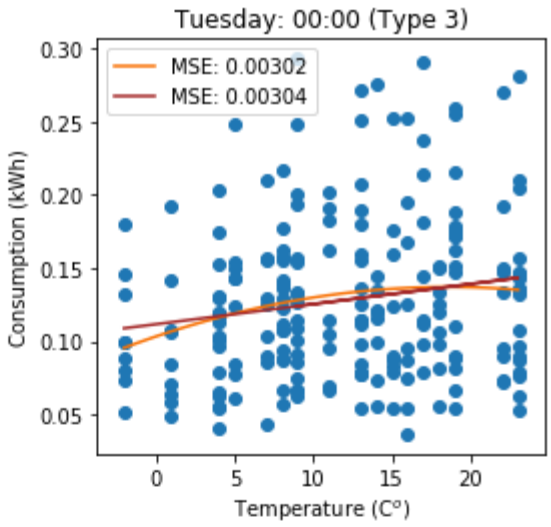

Monday: 22:00 (Type 3)
MSE: 0.00904
MSE: 0.00905

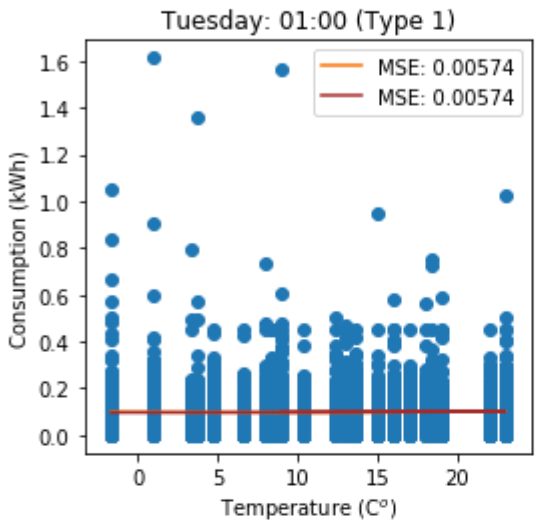

Monday: 23:00 (Type 1)
Monday: 23:00 (Type 2)
Monday: 23:00 (Type 3)
MSE: 0.01346
MSE: 0.01349
MSE: 0.04203
MSE: 0.04207
MSE: 0.00449
MSE: 0.00449
Consumption (kWh)
Temperature (C°)

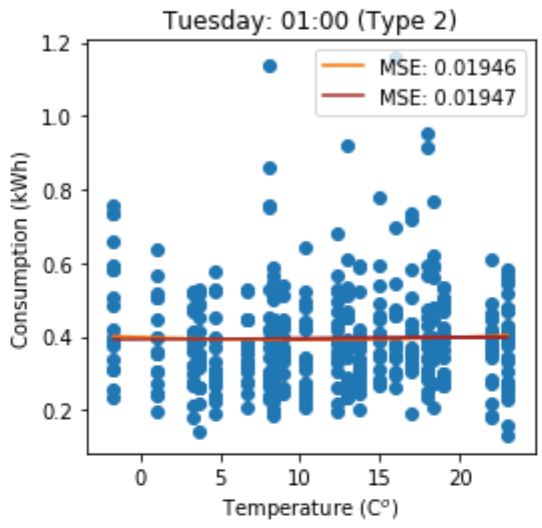
In [34]:
```
# Tuesday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 1 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Tuesday
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Tuesday
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

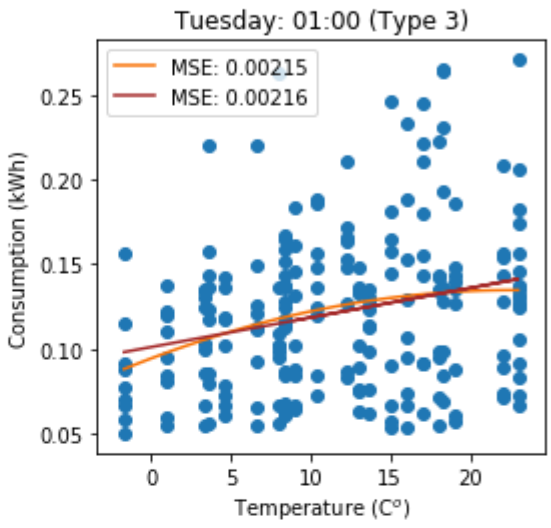

Tuesday: 00:00 (Type 1)
MSE: 0.00959
MSE: 0.0096
Consumption (kWh)
Temperature (C°)

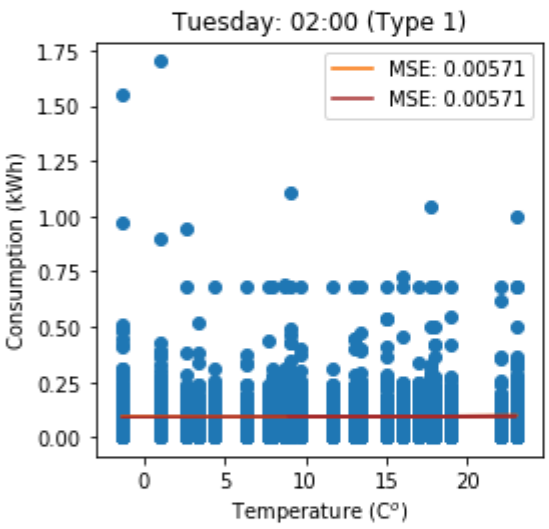

Tuesday: 00:00 (Type 2)
MSE: 0.03064
MSE: 0.0307
Consumption (kWh)
Temperature (C°)

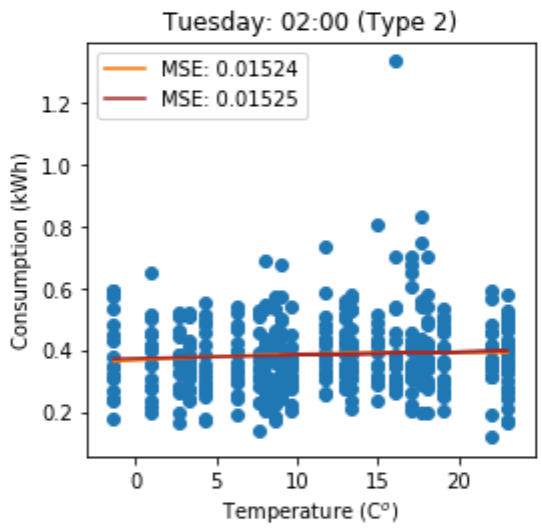

Tuesday: 00:00 (Type 3)
MSE: 0.00302
MSE: 0.00304
Consumption (kWh)
Temperature (C°)

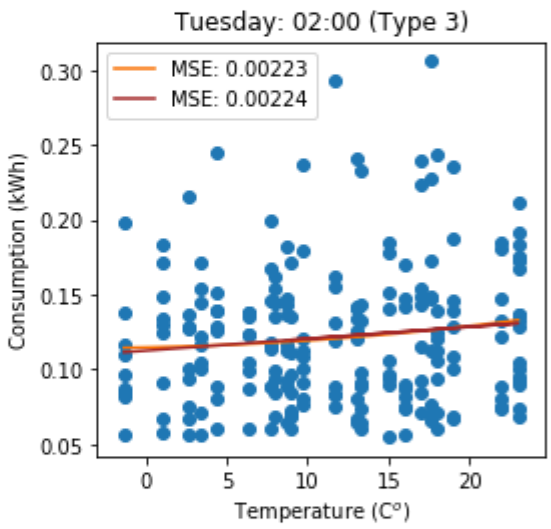

Tuesday: 01:00 (Type 1)
MSE: 0.00574
MSE: 0.00574
Consumption (kWh)
Temperature (C°)

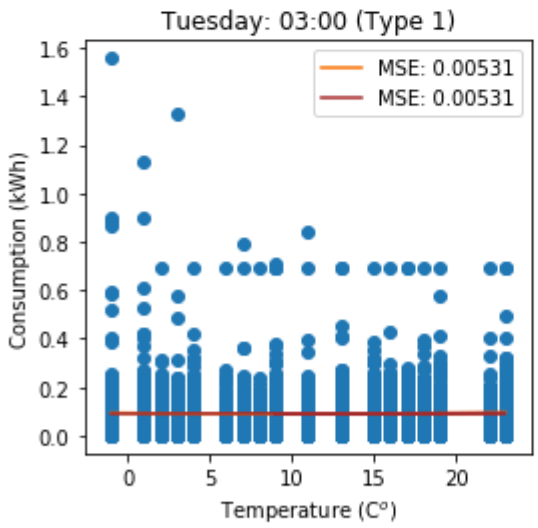

Tuesday: 01:00 (Type 2)
MSE: 0.01946
MSE: 0.01947
Consumption (kWh)
Temperature (C°)

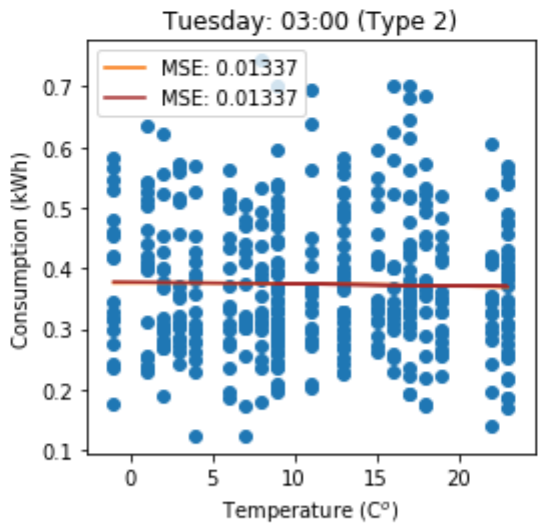

Tuesday: 01:00 (Type 3)
MSE: 0.00215
MSE: 0.00216
Consumption (kWh)
Temperature (C°)

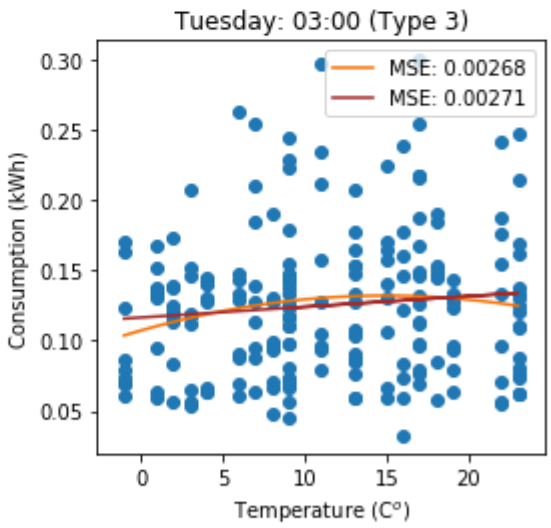

Tuesday: 02:00 (Type 1)
MSE: 0.00571
MSE: 0.00571
Consumption (kWh)
Temperature (C°)

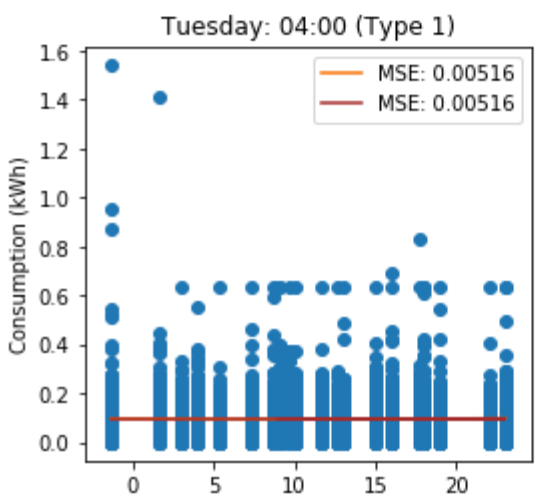

Tuesday: 02:00 (Type 2)
MSE: 0.01524
MSE: 0.01525
Consumption (kWh)
Temperature (C°)

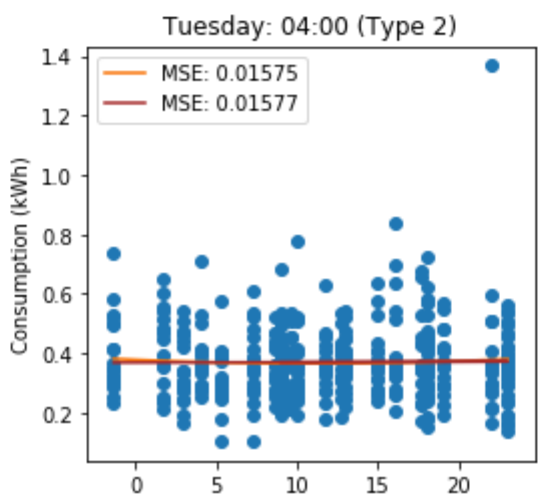

Tuesday: 02:00 (Type 3)
MSE: 0.00223
MSE: 0.00224
Consumption (kWh)
Temperature (C°)

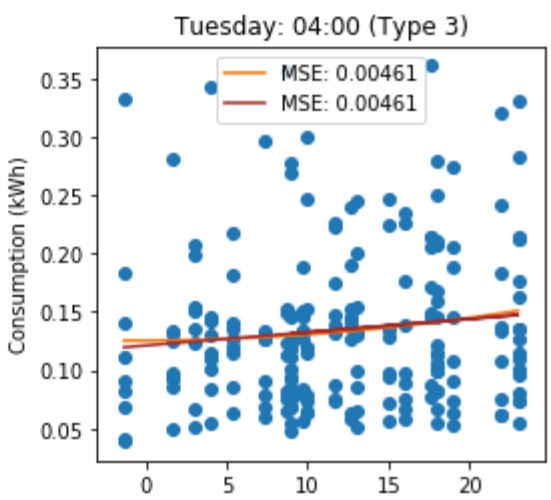

Tuesday: 03:00 (Type 1)
MSE: 0.00531
MSE: 0.00531
Consumption (kWh)
Temperature (C°)


Tuesday: 03:00 (Type 2)
MSE: 0.01337
MSE: 0.01337
Consumption (kWh)
Temperature (C°)


Tuesday: 03:00 (Type 3)
MSE: 0.00268
MSE: 0.00271
Consumption (kWh)
Temperature (C°)


Tuesday: 04:00 (Type 1)
MSE: 0.00516
MSE: 0.00516
Consumption (kWh)
Temperature (C°)

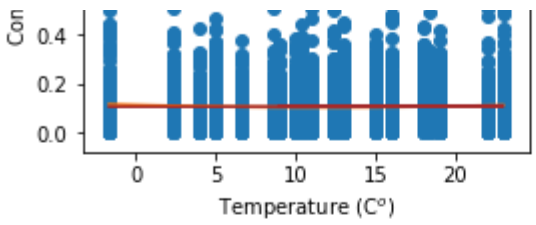

Tuesday: 04:00 (Type 2)
MSE: 0.01575
MSE: 0.01577
Consumption (kWh)
Temperature (C°)

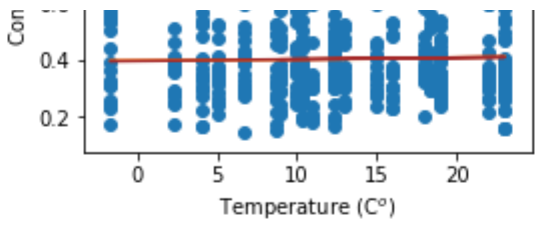

Tuesday: 04:00 (Type 3)
MSE: 0.00461
MSE: 0.00461
Consumption (kWh)
Temperature (C°)

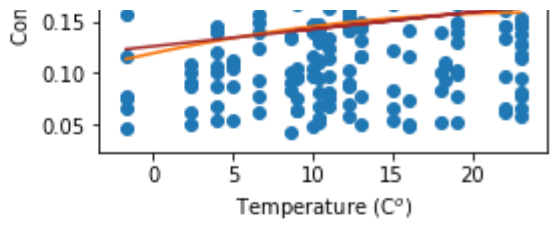
Tuesday: 05:00 (Type 1)
MSE: 0.00731
MSE: 0.00732
Temperature (C°)

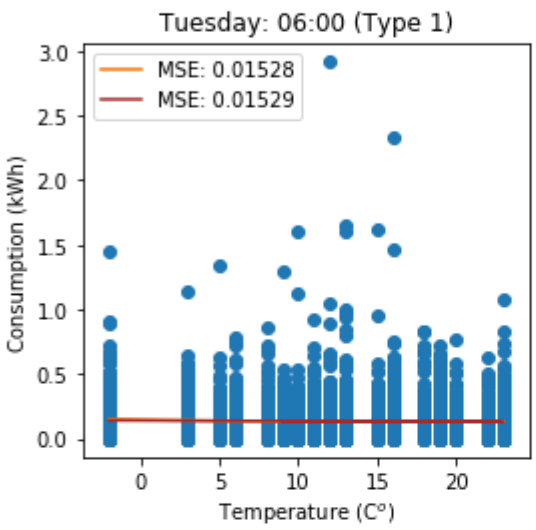
Tuesday: 05:00 (Type 2)
MSE: 0.02702
MSE: 0.02702
Temperature (C°)

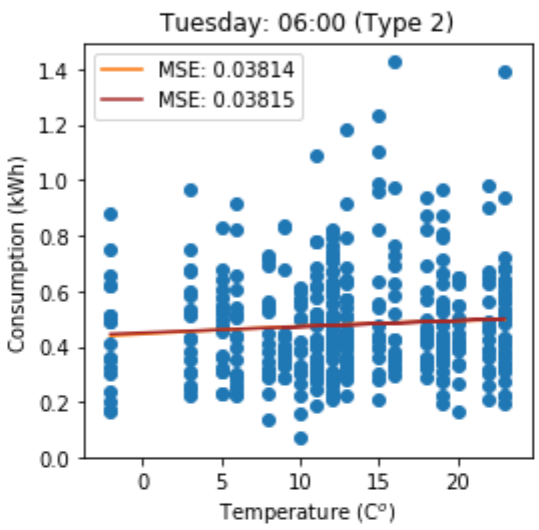
Tuesday: 05:00 (Type 3)
MSE: 0.00527
MSE: 0.00528
Temperature (C°)

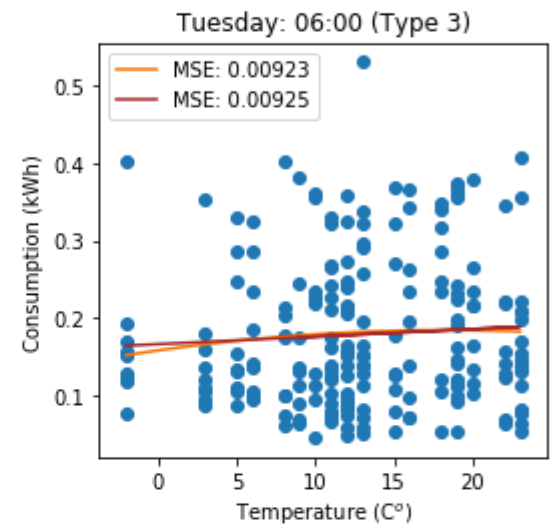

Temperature (C°)

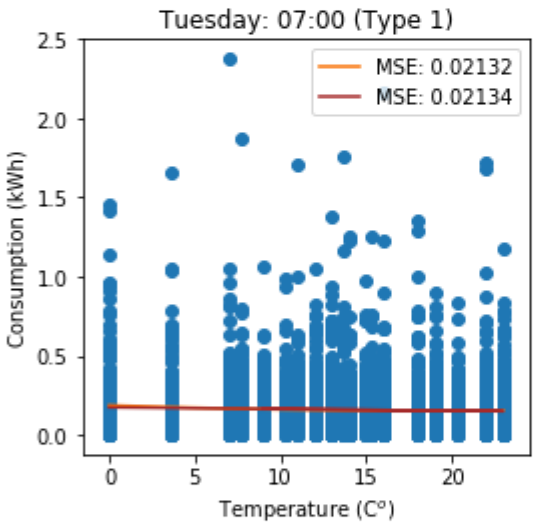

Temperature (C°)

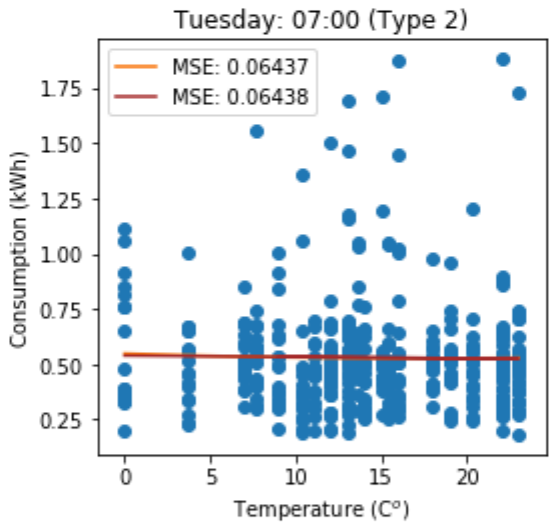

Temperature (C°)

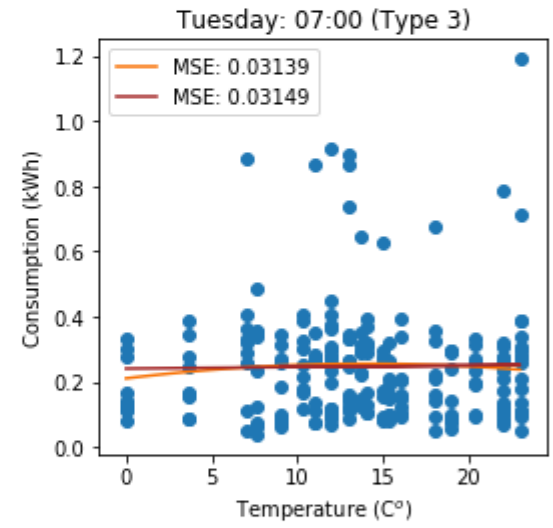

Tuesday: 06:00 (Type 1)
MSE: 0.01528
MSE: 0.01529
Consumption (kWh)
Temperature (C°)

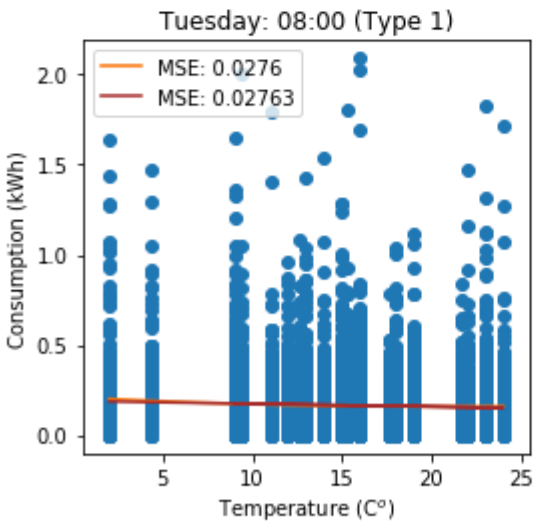

Tuesday: 06:00 (Type 2)
MSE: 0.03814
MSE: 0.03815
Consumption (kWh)
Temperature (C°)

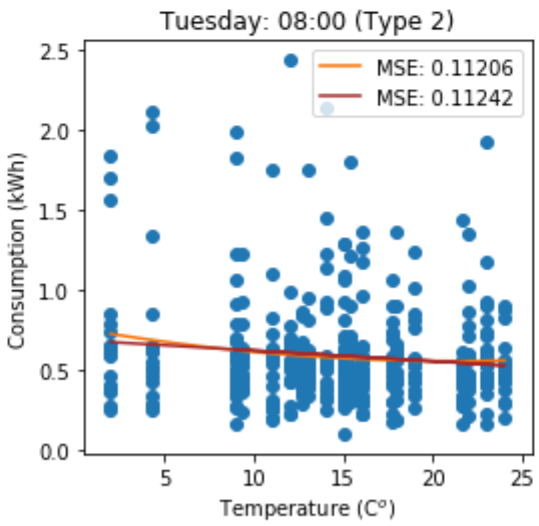

Tuesday: 06:00 (Type 3)
MSE: 0.00923
MSE: 0.00925
Consumption (kWh)
Temperature (C°)

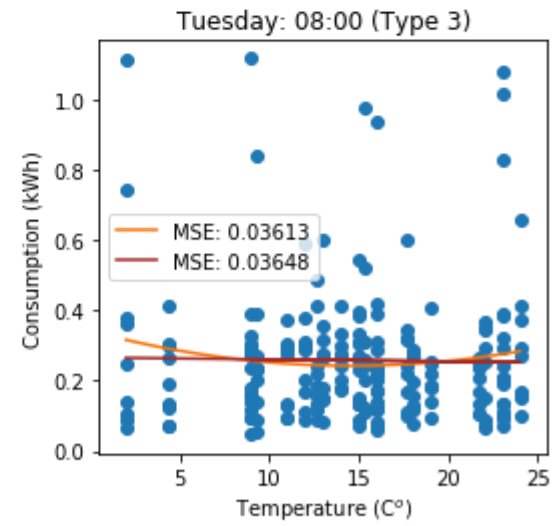

Tuesday: 07:00 (Type 1)
MSE: 0.02132
MSE: 0.02134
Consumption (kWh)
Temperature (C°)

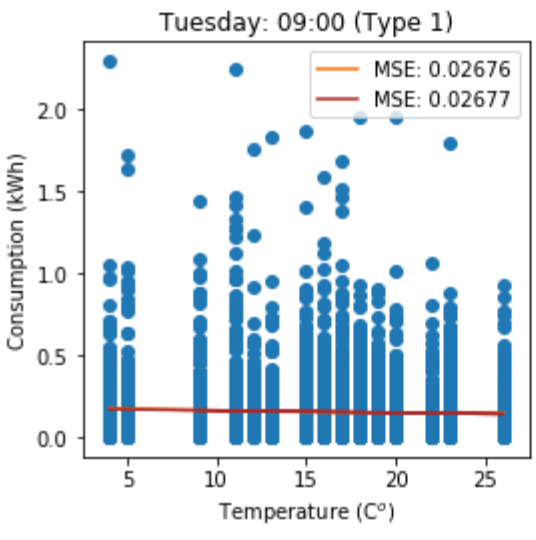

Tuesday: 07:00 (Type 2)
MSE: 0.06437
MSE: 0.06438
Consumption (kWh)
Temperature (C°)

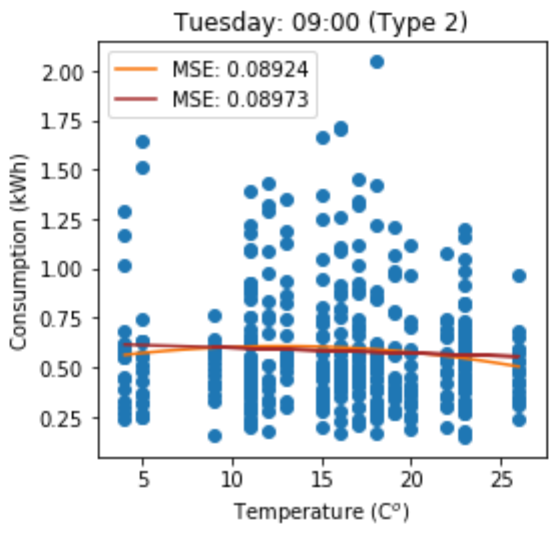

Tuesday: 07:00 (Type 3)
MSE: 0.03139
MSE: 0.03149
Consumption (kWh)
Temperature (C°)

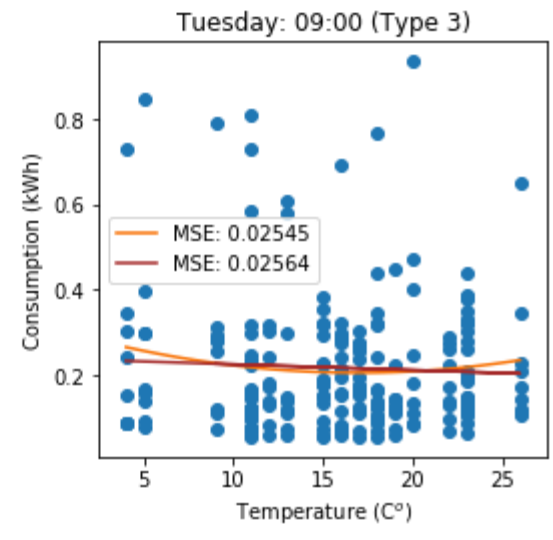

Tuesday: 08:00 (Type 1)
MSE: 0.0276
MSE: 0.02763
Consumption (kWh)
Temperature (C°)

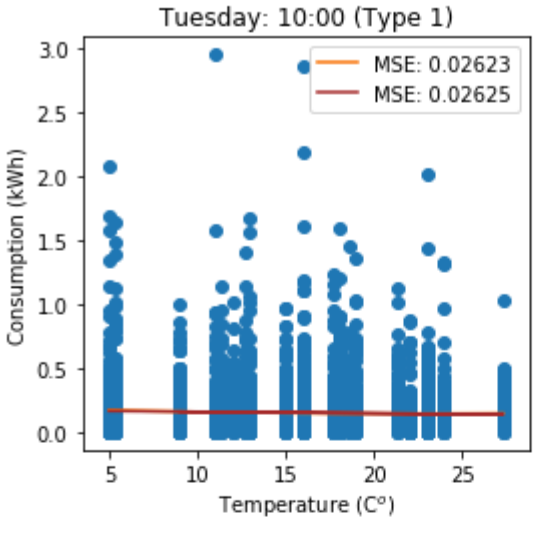

Tuesday: 08:00 (Type 2)
MSE: 0.11206
MSE: 0.11242
Consumption (kWh)
Temperature (C°)

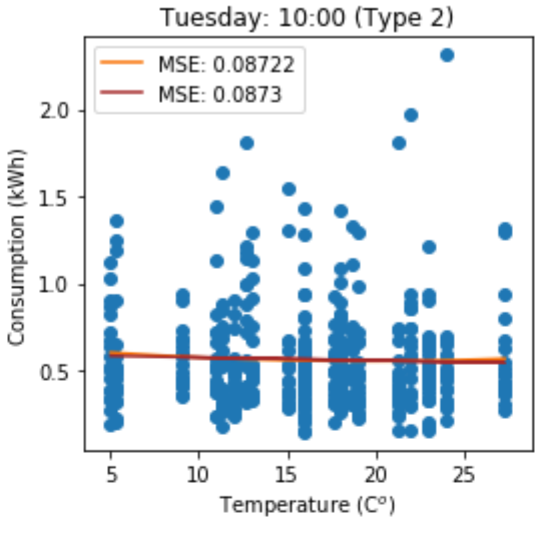

Tuesday: 08:00 (Type 3)
MSE: 0.03613
MSE: 0.03648
Consumption (kWh)
Temperature (C°)

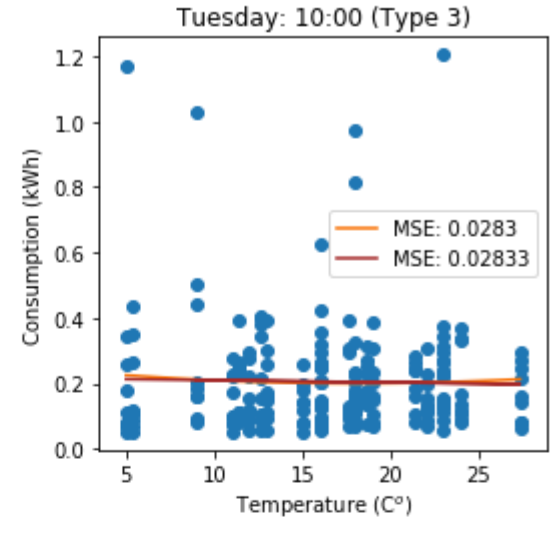

Tuesday: 09:00 (Type 1)
MSE: 0.02676
MSE: 0.02677
Consumption (kWh)
Temperature (C°)


Tuesday: 09:00 (Type 2)
MSE: 0.08924
MSE: 0.08973
Consumption (kWh)
Temperature (C°)

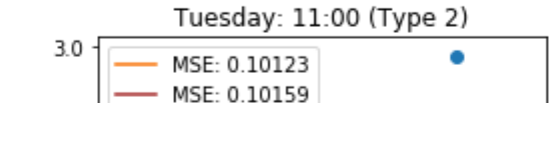

Tuesday: 09:00 (Type 3)
MSE: 0.02545
MSE: 0.02564
Consumption (kWh)
Temperature (C°)

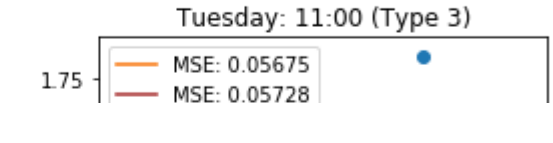

Tuesday: 10:00 (Type 1)
MSE: 0.02623
MSE: 0.02625
Consumption (kWh)
Temperature (C°)

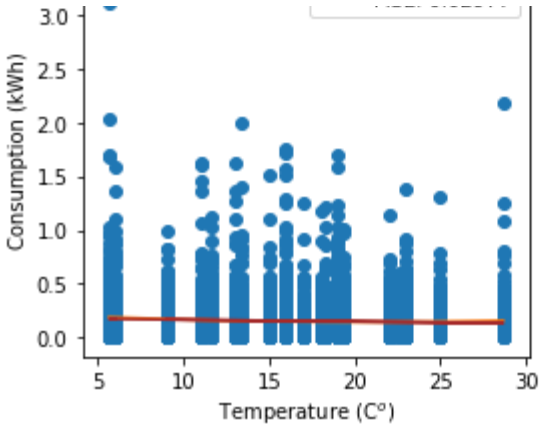

Tuesday: 10:00 (Type 2)
MSE: 0.08722
MSE: 0.0873
Consumption (kWh)
Temperature (C°)

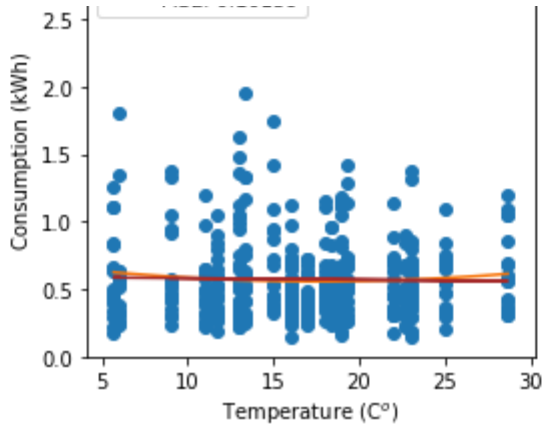

Tuesday: 10:00 (Type 3)
MSE: 0.0283
MSE: 0.02833
Consumption (kWh)
Temperature (C°)

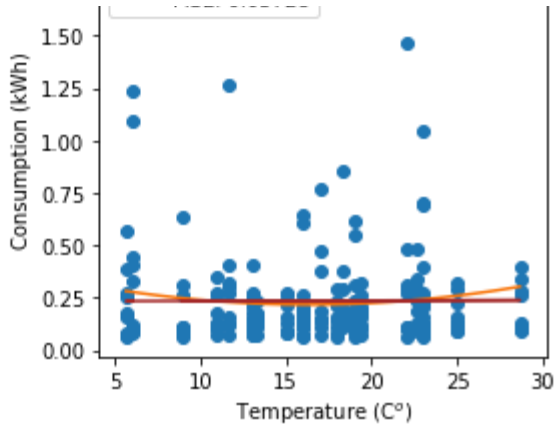

Tuesday: 11:00 (Type 1)
MSE: 0.02971
MSE: 0.02977

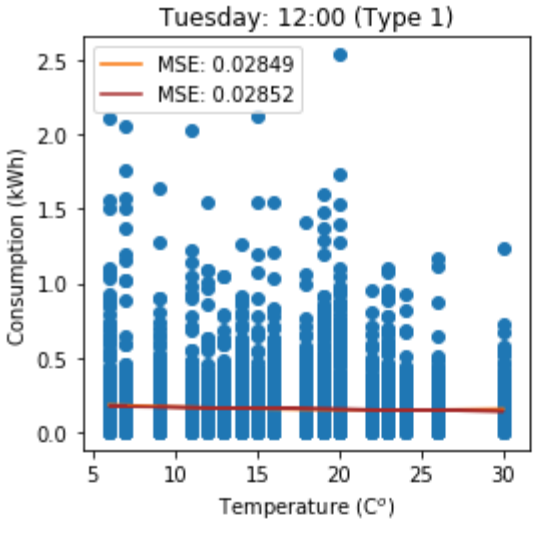

Tuesday: 11:00 (Type 2)
MSE: 0.10123
MSE: 0.10159

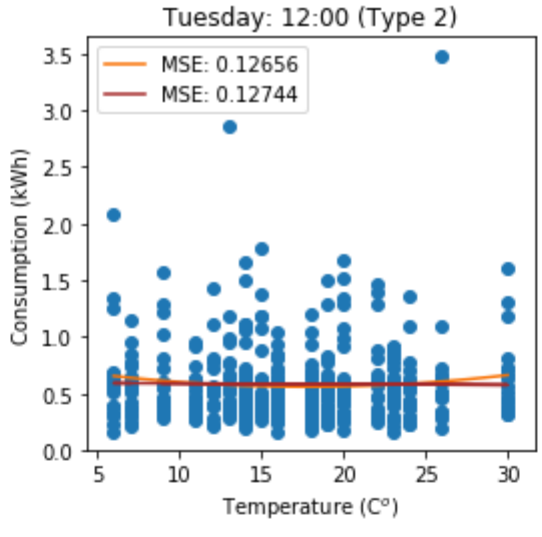

Tuesday: 11:00 (Type 3)
MSE: 0.05675
MSE: 0.05728

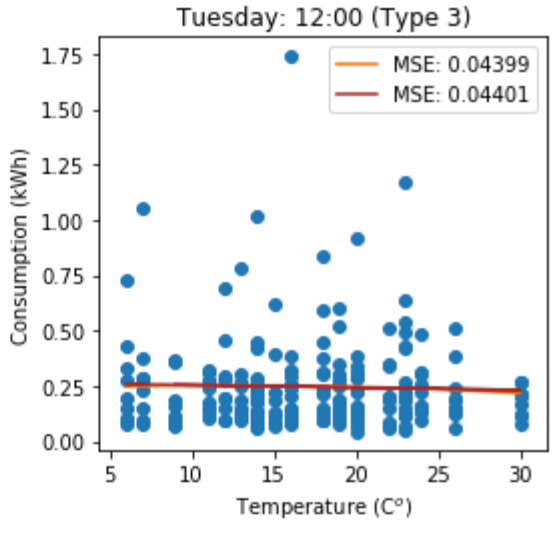

Consumption (kWh)
Temperature (C°)

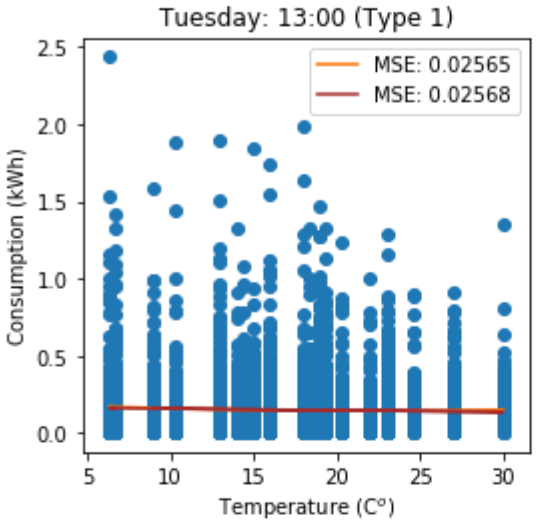

Consumption (kWh)
Temperature (C°)

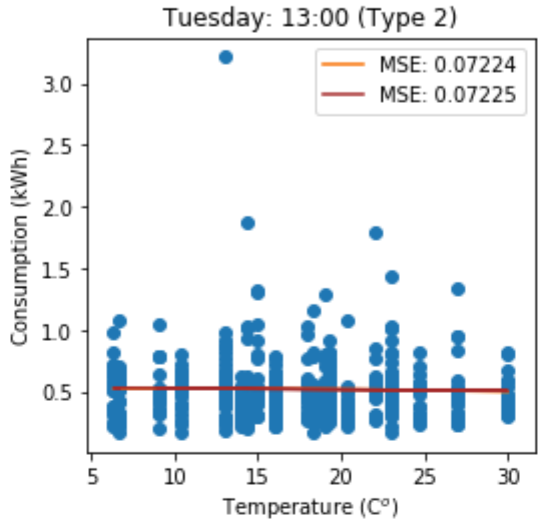

Consumption (kWh)
Temperature (C°)

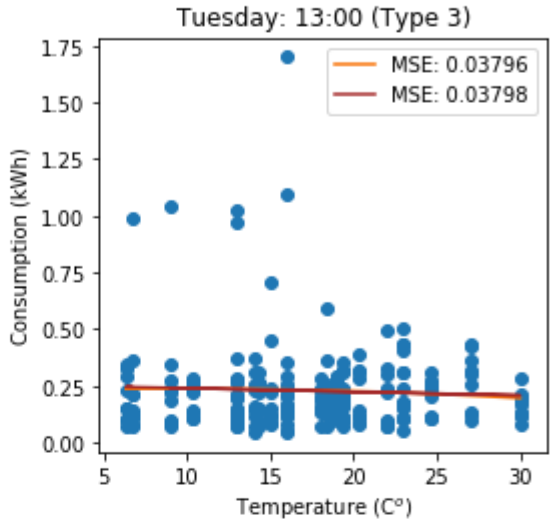

Tuesday: 12:00 (Type 1)
MSE: 0.02849
MSE: 0.02852
Consumption (kWh)
Temperature (C°)

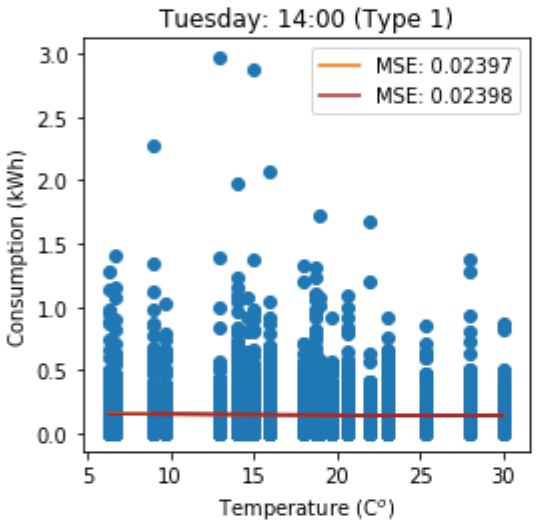

Tuesday: 12:00 (Type 2)
MSE: 0.12656
MSE: 0.12744
Consumption (kWh)
Temperature (C°)

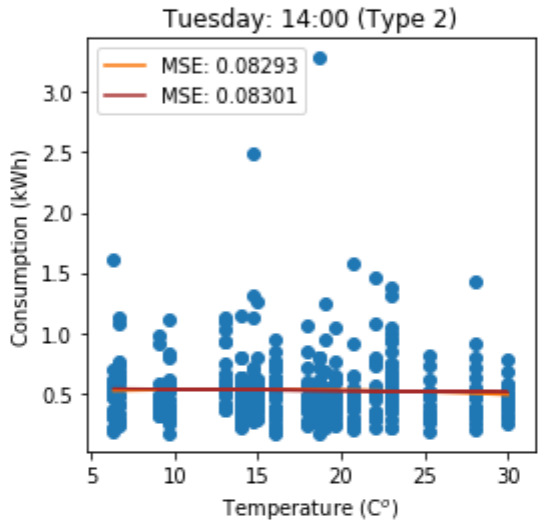

Tuesday: 12:00 (Type 3)
MSE: 0.04399
MSE: 0.04401
Consumption (kWh)
Temperature (C°)

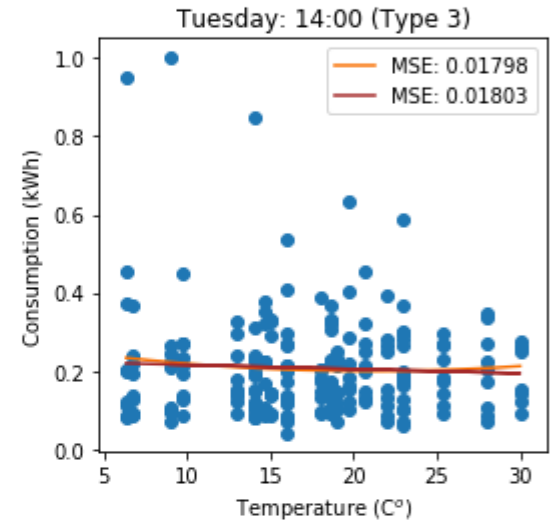

Tuesday: 13:00 (Type 1)
MSE: 0.02565
MSE: 0.02568
Consumption (kWh)
Temperature (C°)

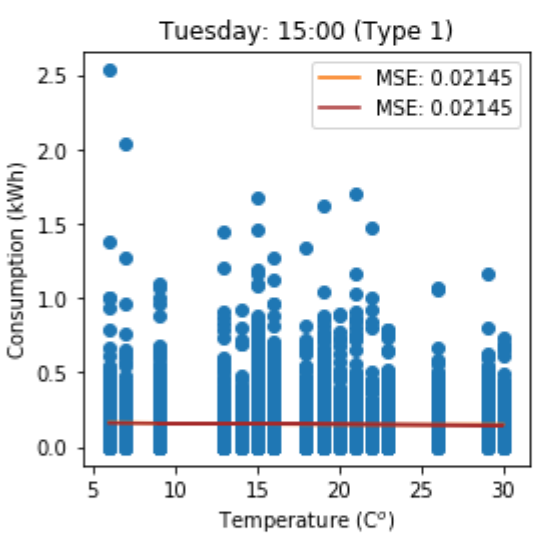

Tuesday: 13:00 (Type 2)
MSE: 0.07224
MSE: 0.07225
Consumption (kWh)
Temperature (C°)

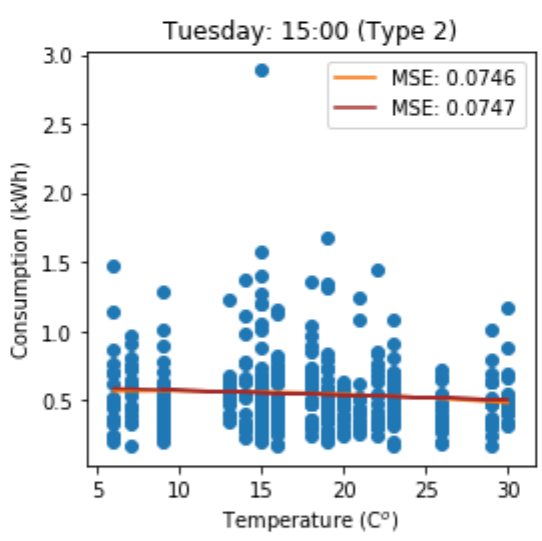

Tuesday: 13:00 (Type 3)
MSE: 0.03796
MSE: 0.03798
Consumption (kWh)
Temperature (C°)

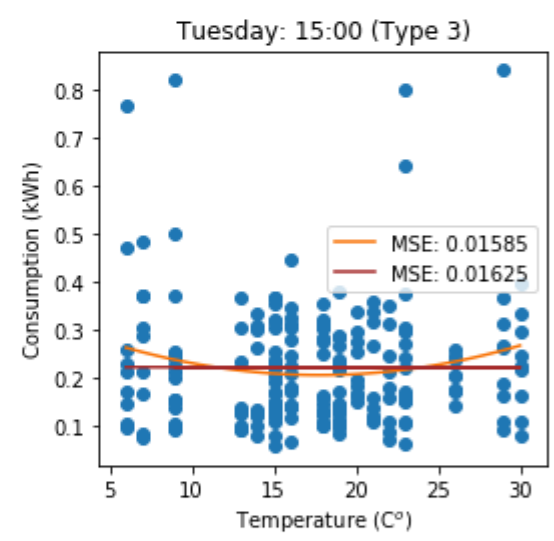

Tuesday: 14:00 (Type 1)
MSE: 0.02397
MSE: 0.02398
Consumption (kWh)
Temperature (C°)

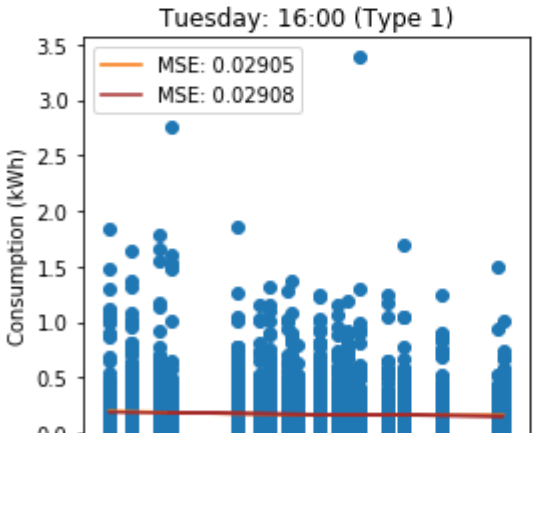

Tuesday: 14:00 (Type 2)
MSE: 0.08293
MSE: 0.08301
Consumption (kWh)
Temperature (C°)

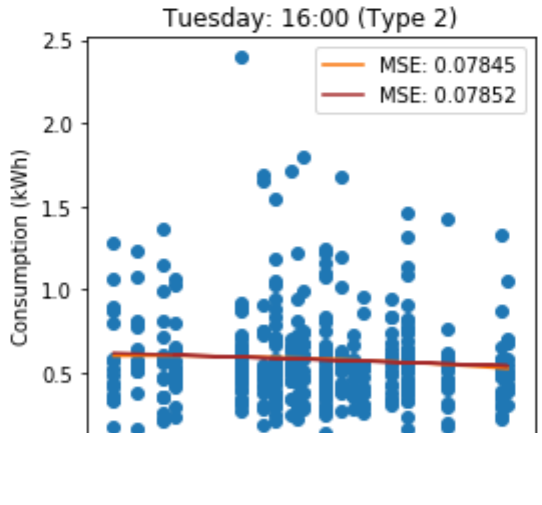

Tuesday: 14:00 (Type 3)
MSE: 0.01798
MSE: 0.01803
Consumption (kWh)
Temperature (C°)

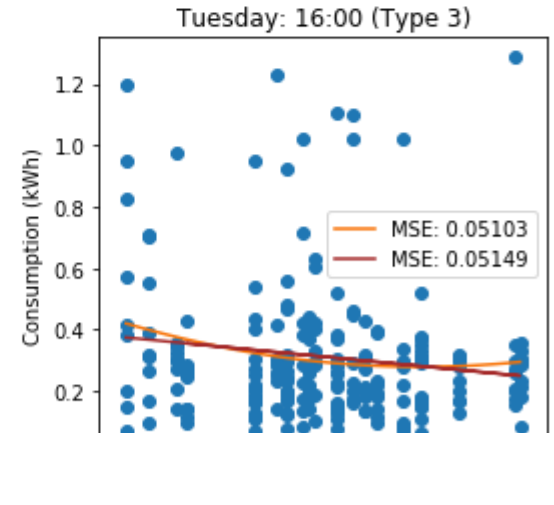

Tuesday: 15:00 (Type 1)
MSE: 0.02145
MSE: 0.02145
Consumption (kWh)
Temperature (C°)

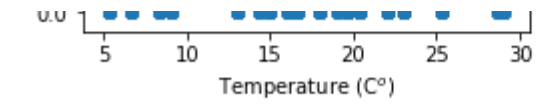

Tuesday: 15:00 (Type 2)
MSE: 0.0746
MSE: 0.0747
Consumption (kWh)
Temperature (C°)

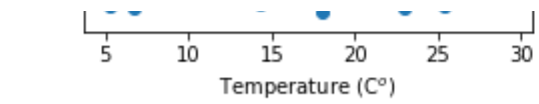

Tuesday: 15:00 (Type 3)
MSE: 0.01585
MSE: 0.01625
Consumption (kWh)
Temperature (C°)

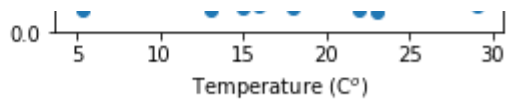

Tuesday: 16:00 (Type 1)
MSE: 0.02905
MSE: 0.02908
Consumption (kWh)

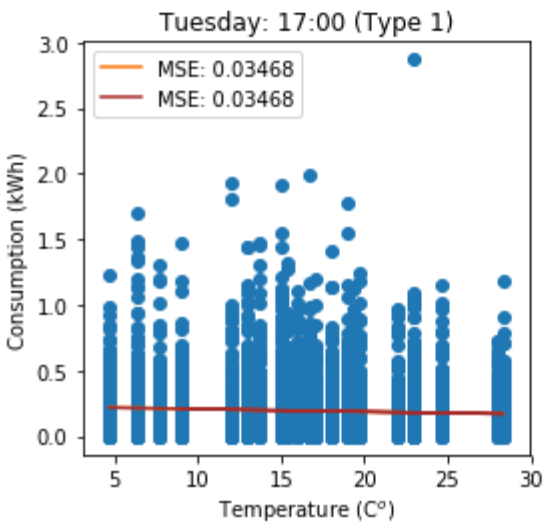

Tuesday: 16:00 (Type 2)
MSE: 0.07845
MSE: 0.07852
Consumption (kWh)

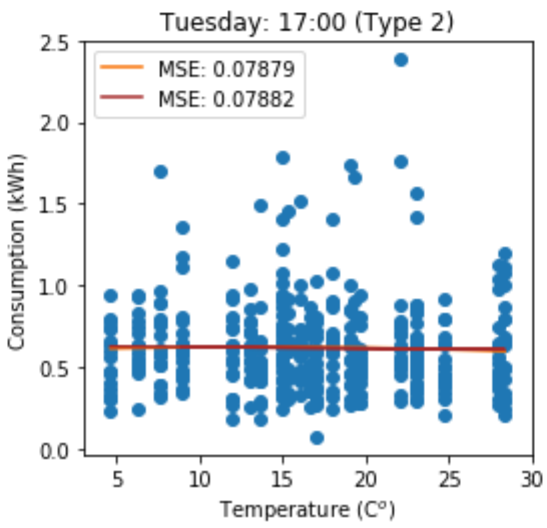

Tuesday: 16:00 (Type 3)
MSE: 0.05103
MSE: 0.05149
Consumption (kWh)

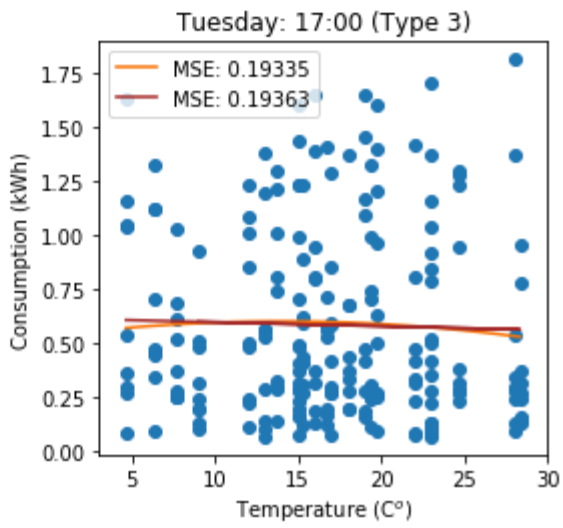

Temperature (C°)

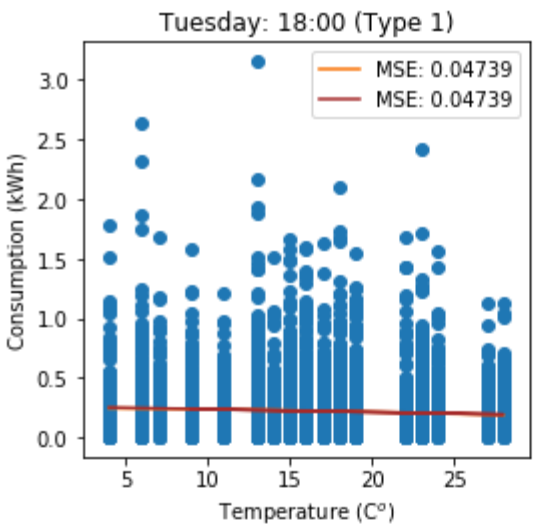

Temperature (C°)

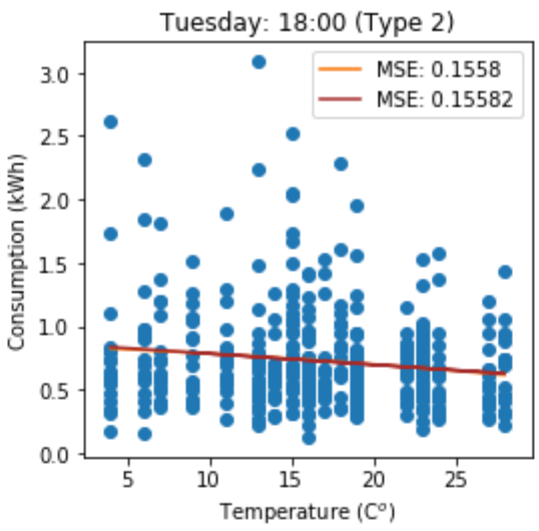

Temperature (C°)

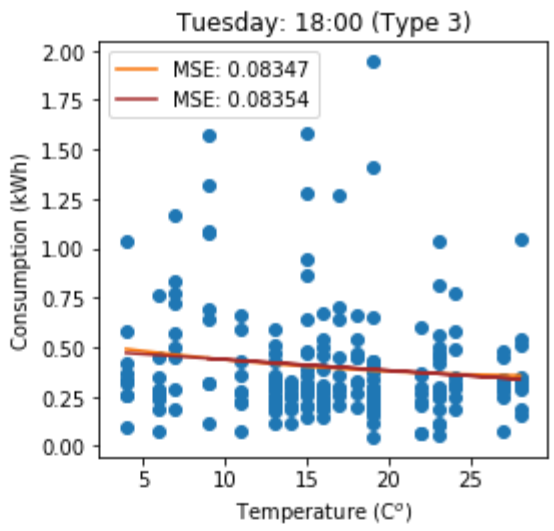

Tuesday: 17:00 (Type 1)
MSE: 0.03468
MSE: 0.03468
Consumption (kWh)
Temperature (C°)

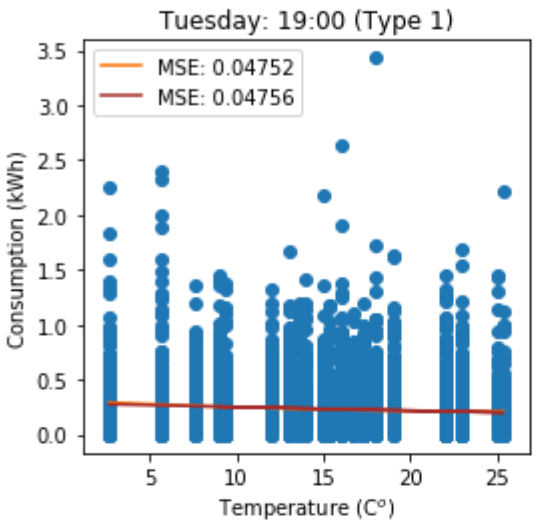

Tuesday: 17:00 (Type 2)
MSE: 0.07879
MSE: 0.07882
Consumption (kWh)
Temperature (C°)

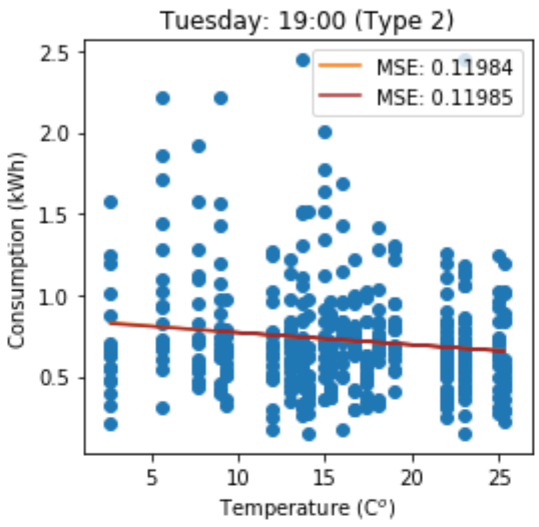

Tuesday: 17:00 (Type 3)
MSE: 0.19335
MSE: 0.19363
Consumption (kWh)
Temperature (C°)

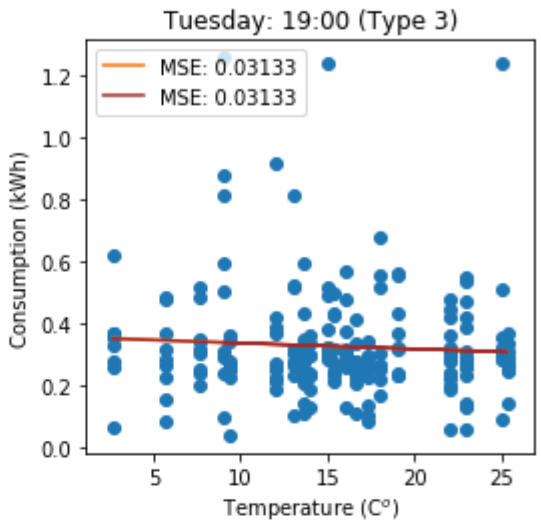

Tuesday: 18:00 (Type 1)
MSE: 0.04739
MSE: 0.04739
Consumption (kWh)
Temperature (C°)

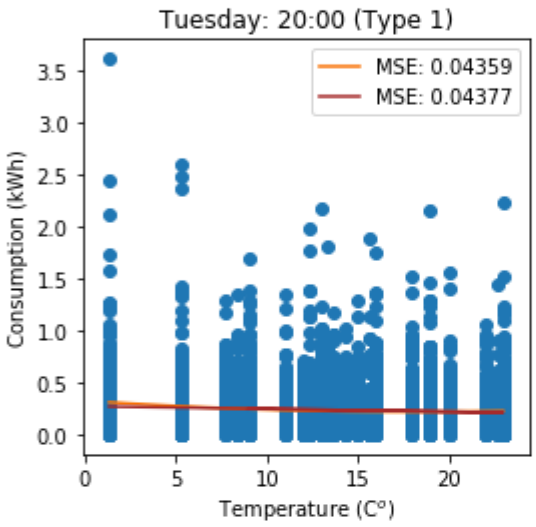

Tuesday: 18:00 (Type 2)
MSE: 0.1558
MSE: 0.15582
Consumption (kWh)
Temperature (C°)

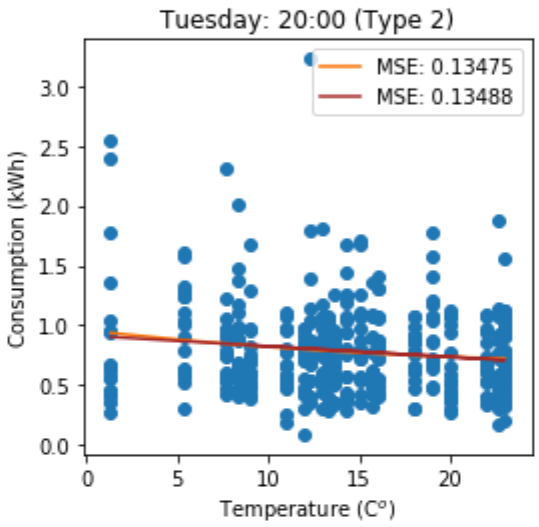

Tuesday: 18:00 (Type 3)
MSE: 0.08347
MSE: 0.08354
Consumption (kWh)
Temperature (C°)

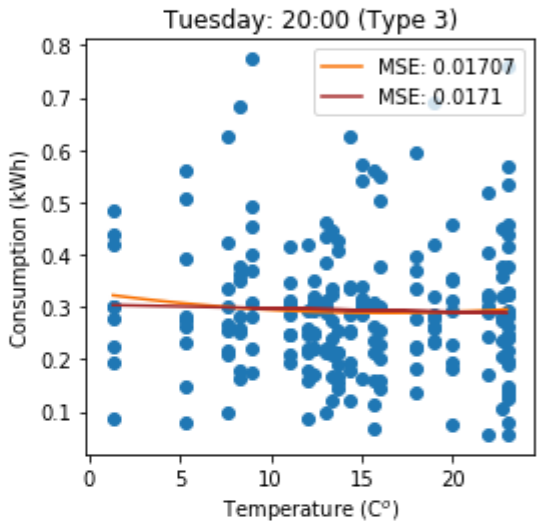

Tuesday: 19:00 (Type 1)
MSE: 0.04752
MSE: 0.04756
Consumption (kWh)
Temperature (C°)

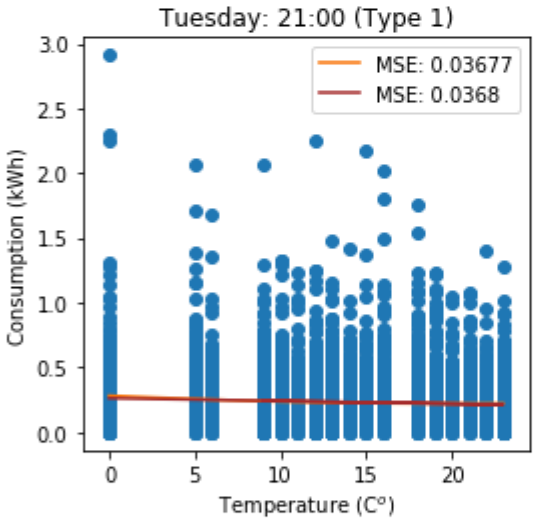

Tuesday: 19:00 (Type 2)
MSE: 0.11984
MSE: 0.11985
Consumption (kWh)
Temperature (C°)

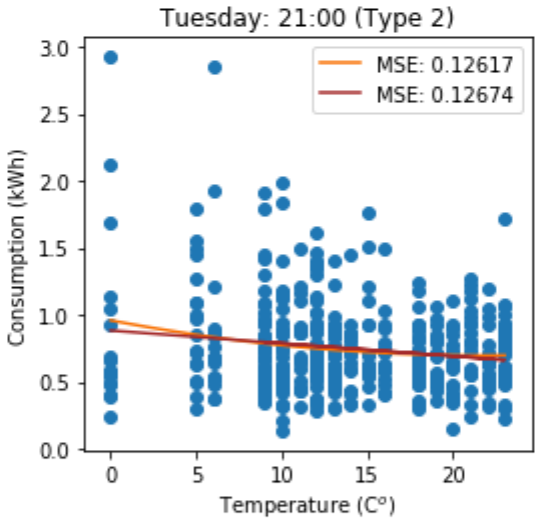

Tuesday: 19:00 (Type 3)
MSE: 0.03133
MSE: 0.03133
Consumption (kWh)
Temperature (C°)

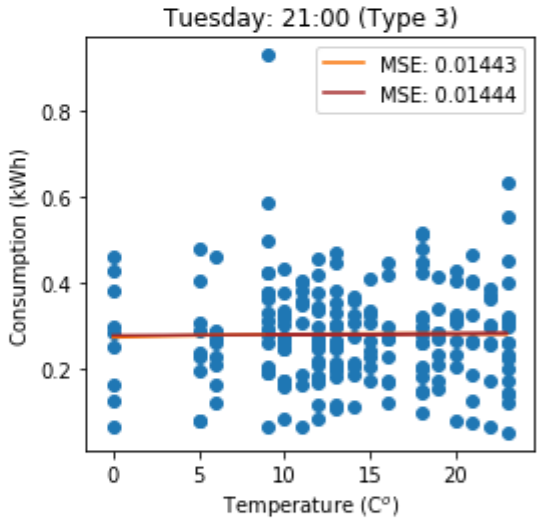

Tuesday: 20:00 (Type 1)
MSE: 0.04359
MSE: 0.04377
Consumption (kWh)
Temperature (C°)

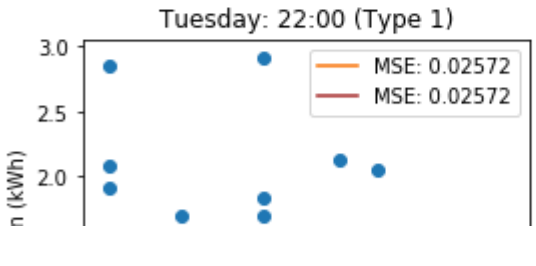

Tuesday: 20:00 (Type 2)
MSE: 0.13475
MSE: 0.13488
Consumption (kWh)
Temperature (C°)

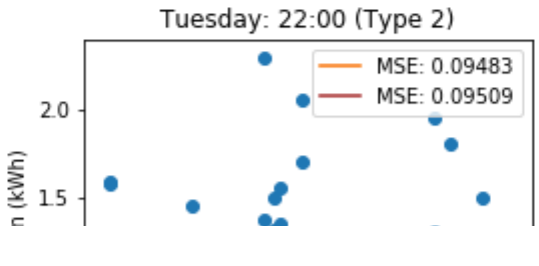

Tuesday: 20:00 (Type 3)
MSE: 0.01707
MSE: 0.0171
Consumption (kWh)
Temperature (C°)

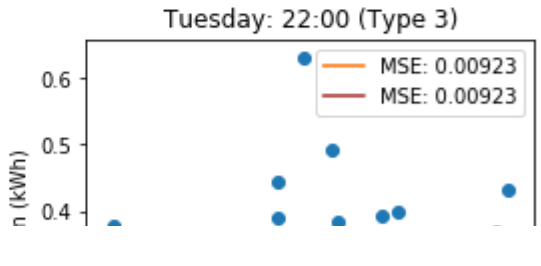

Tuesday: 21:00 (Type 1)
MSE: 0.03677
MSE: 0.0368
Consumption (kWh)
Temperature (C°)

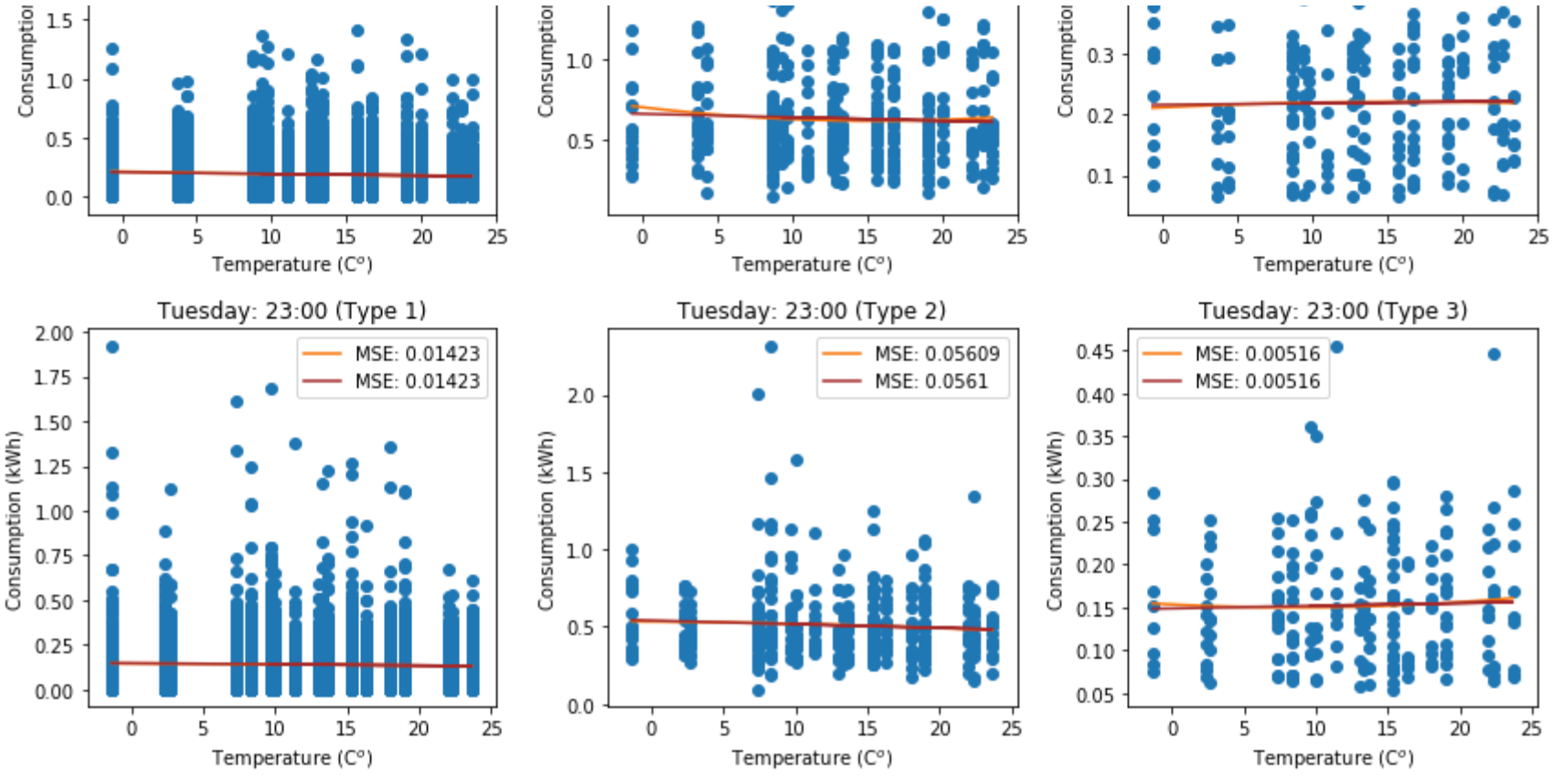

Tuesday: 21:00 (Type 2)
MSE: 0.12617
MSE: 0.12674
Consumption (kWh)
Temperature (C°)


Tuesday: 21:00 (Type 3)
MSE: 0.01443
MSE: 0.01444
Consumption (kWh)
Temperature (C°)

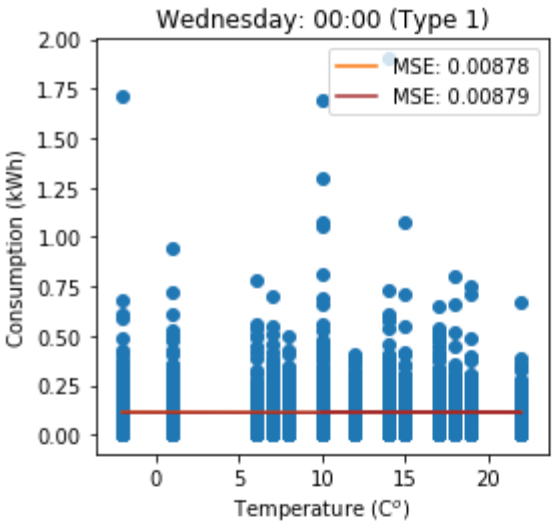

Tuesday: 22:00 (Type 1)
MSE: 0.02572
MSE: 0.02572
Temperature (C°)

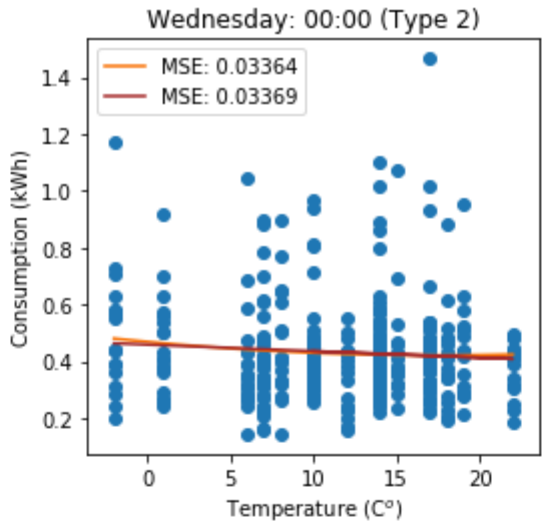

Tuesday: 22:00 (Type 2)
MSE: 0.09483
MSE: 0.09509
Temperature (C°)

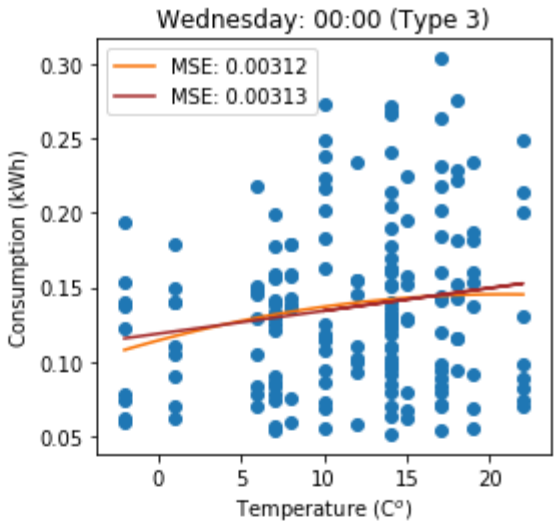

Tuesday: 22:00 (Type 3)
MSE: 0.00923
MSE: 0.00923
Temperature (C°)

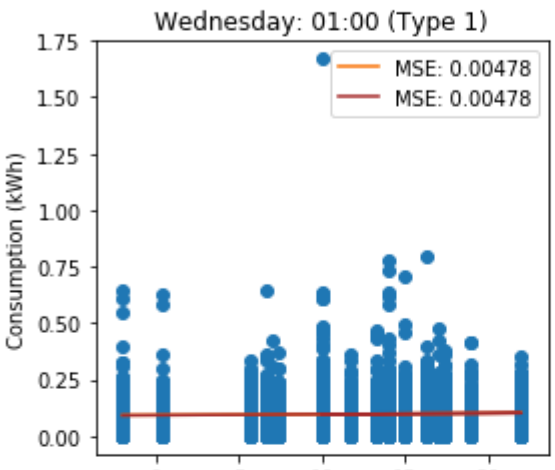

Tuesday: 23:00 (Type 1)
MSE: 0.01423
MSE: 0.01423
Tuesday: 23:00 (Type 2)
MSE: 0.05609
MSE: 0.0561
Tuesday: 23:00 (Type 3)
MSE: 0.00516
MSE: 0.00516
Consumption (kWh)
Temperature (C°)

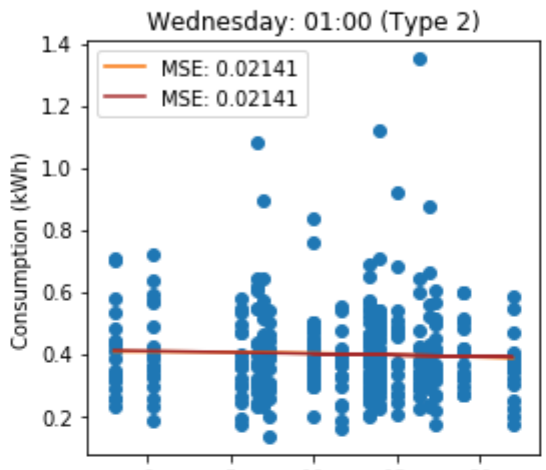
In [35]:
```
# Wednesday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 2 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

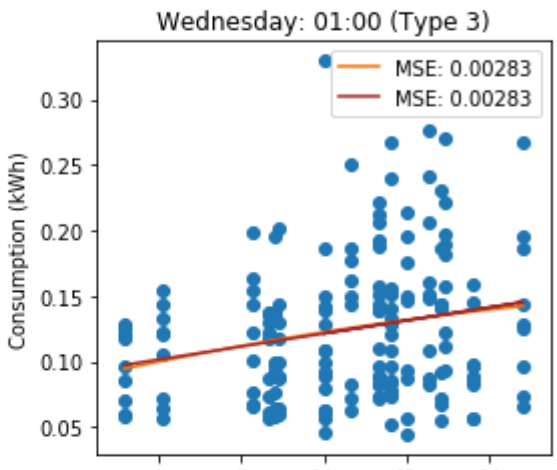

Wednesday: 00:00 (Type 1)
MSE: 0.00878
MSE: 0.00879
Consumption (kWh)
Temperature (C°)

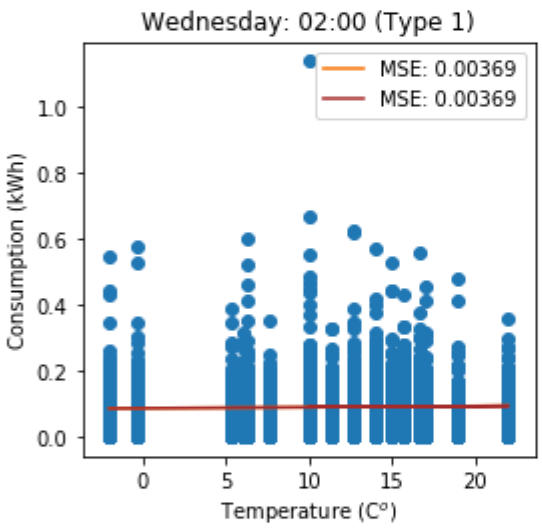

Wednesday: 00:00 (Type 2)
MSE: 0.03364
MSE: 0.03369
Consumption (kWh)
Temperature (C°)

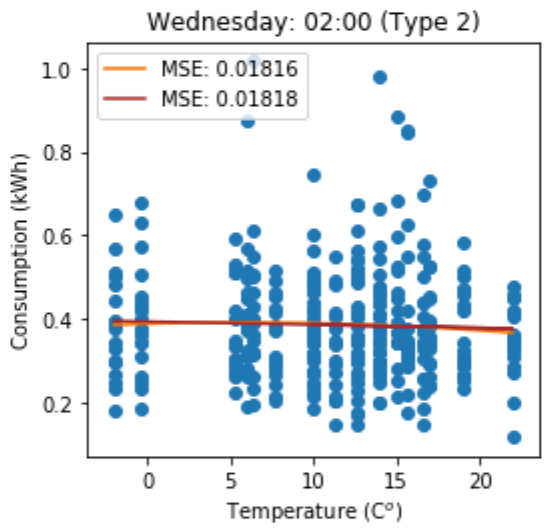

Wednesday: 00:00 (Type 3)
MSE: 0.00312
MSE: 0.00313
Consumption (kWh)
Temperature (C°)

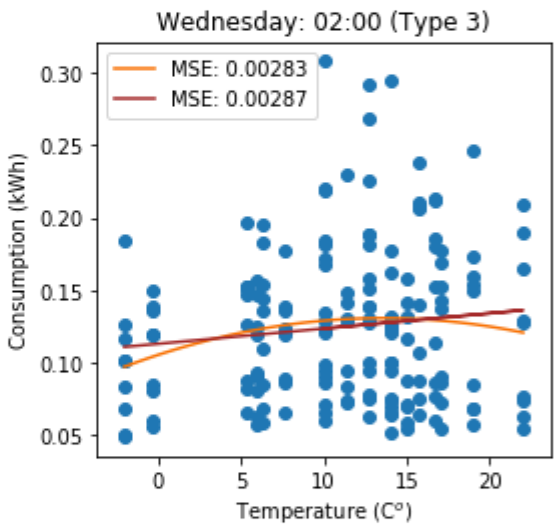

Wednesday: 01:00 (Type 1)
MSE: 0.00478
MSE: 0.00478
Consumption (kWh)
Temperature (C°)

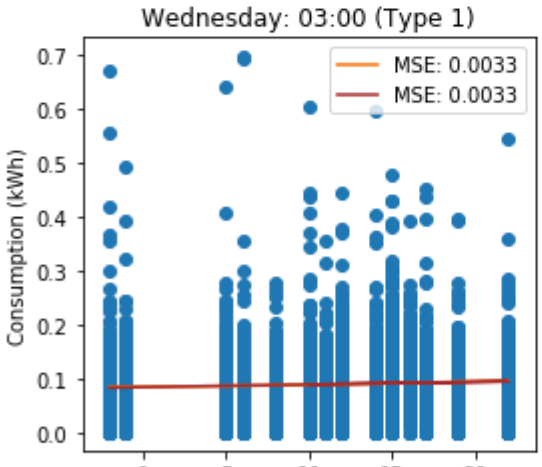

Wednesday: 01:00 (Type 2)
MSE: 0.02141
MSE: 0.02141
Consumption (kWh)
Temperature (C°)

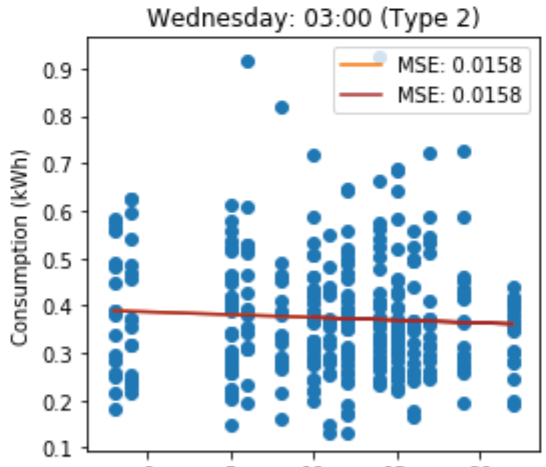

Wednesday: 01:00 (Type 3)
MSE: 0.00283
MSE: 0.00283
Consumption (kWh)
Temperature (C°)

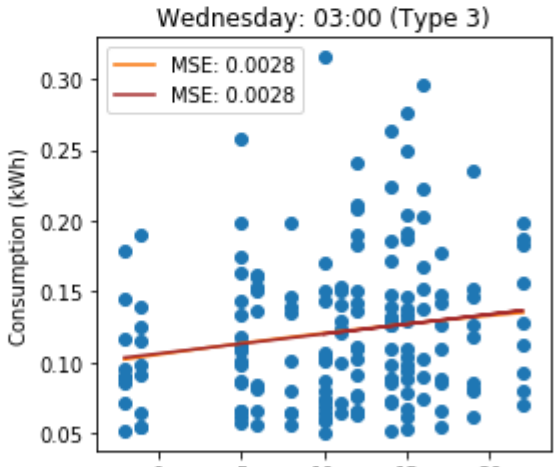

Wednesday: 02:00 (Type 1)
MSE: 0.00369
MSE: 0.00369
Consumption (kWh)
Temperature (C°)

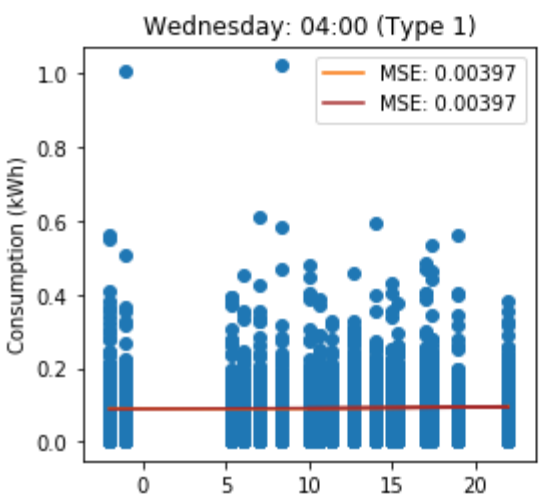

Wednesday: 02:00 (Type 2)
MSE: 0.01816
MSE: 0.01818
Consumption (kWh)
Temperature (C°)

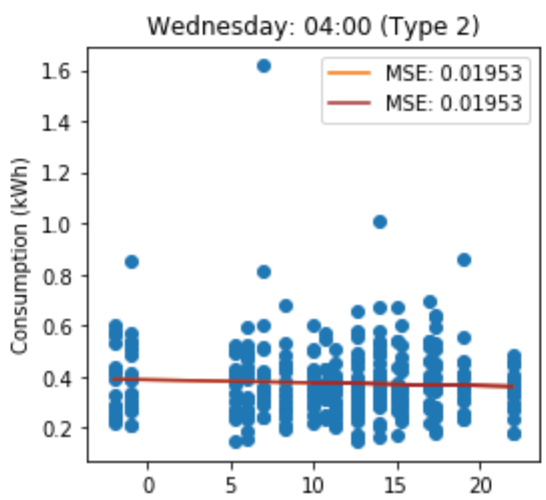

Wednesday: 02:00 (Type 3)
MSE: 0.00283
MSE: 0.00287
Consumption (kWh)
Temperature (C°)

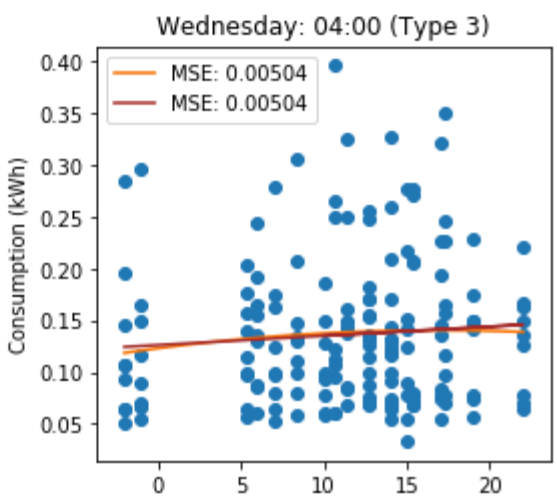

Wednesday: 03:00 (Type 1)
MSE: 0.0033
MSE: 0.0033
Consumption (kWh)
Temperature (C°)

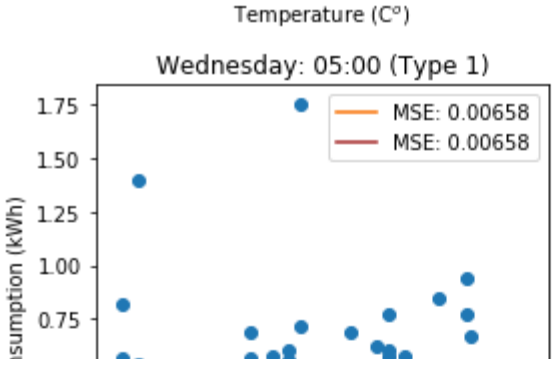

Wednesday: 03:00 (Type 2)
MSE: 0.0158
MSE: 0.0158
Consumption (kWh)
Temperature (C°)

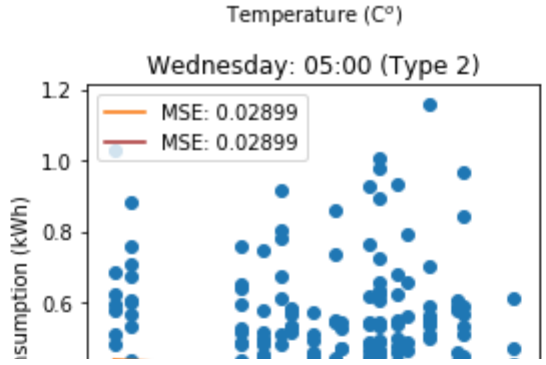

Wednesday: 03:00 (Type 3)
MSE: 0.0028
MSE: 0.0028
Consumption (kWh)
Temperature (C°)

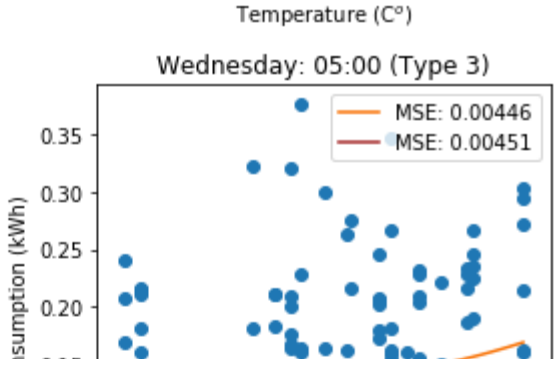

Wednesday: 04:00 (Type 1)
MSE: 0.00397
MSE: 0.00397
Consumption (kWh)
Temperature (C°)

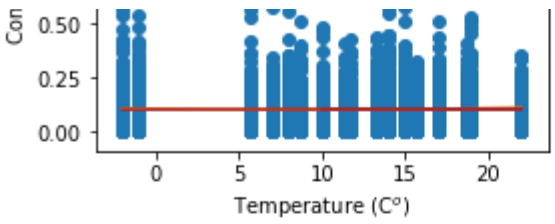

Wednesday: 04:00 (Type 2)
MSE: 0.01953
MSE: 0.01953
Consumption (kWh)
Temperature (C°)

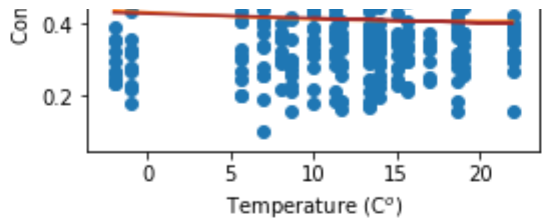

Wednesday: 04:00 (Type 3)
MSE: 0.00504
MSE: 0.00504
Consumption (kWh)
Temperature (C°)

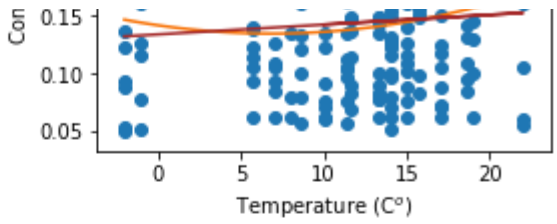

Wednesday: 05:00 (Type 1)
MSE: 0.00658
MSE: 0.00658

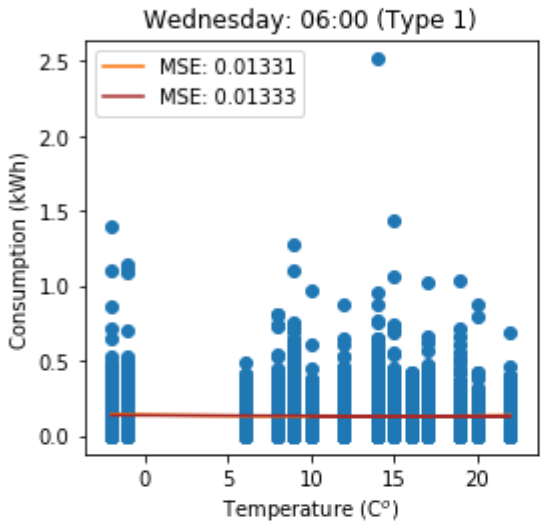

Wednesday: 05:00 (Type 2)
MSE: 0.02899
MSE: 0.02899

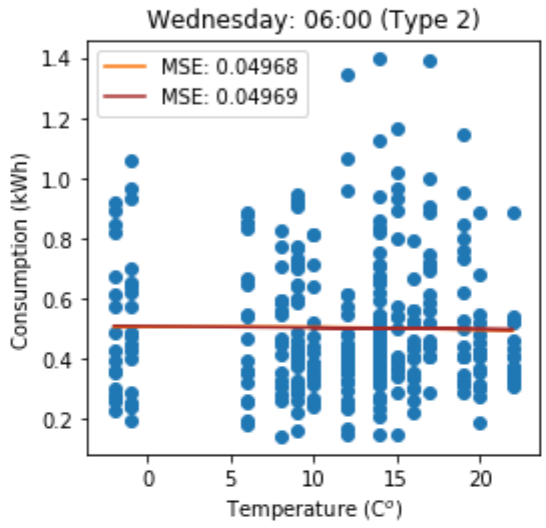

Wednesday: 05:00 (Type 3)
MSE: 0.00446
MSE: 0.00451

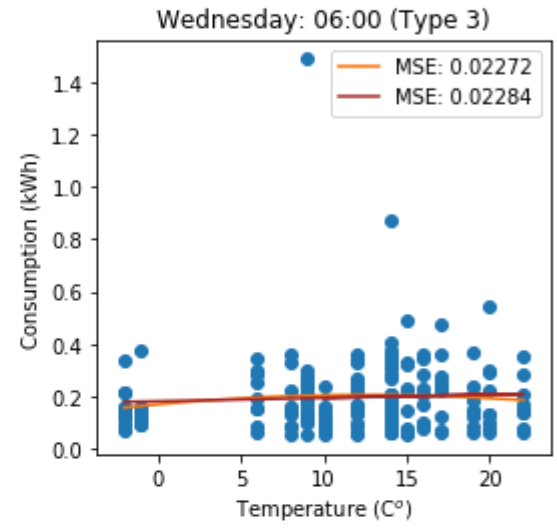

Con
0.50
0.25
0.00
0
5
10
15
20
Temperature (C°)

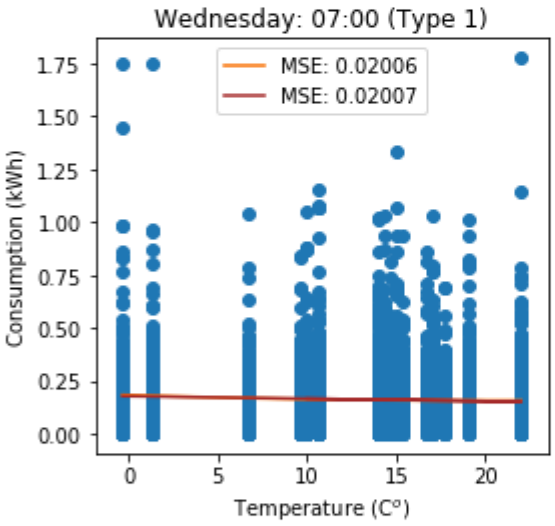

Con
0.4
0.2
0
5
10
15
20
Temperature (C°)

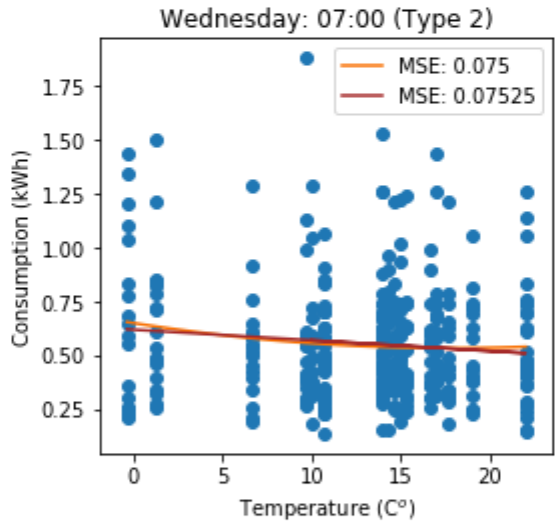

Con
0.15
0.10
0.05
0
5
10
15
20
Temperature (C°)

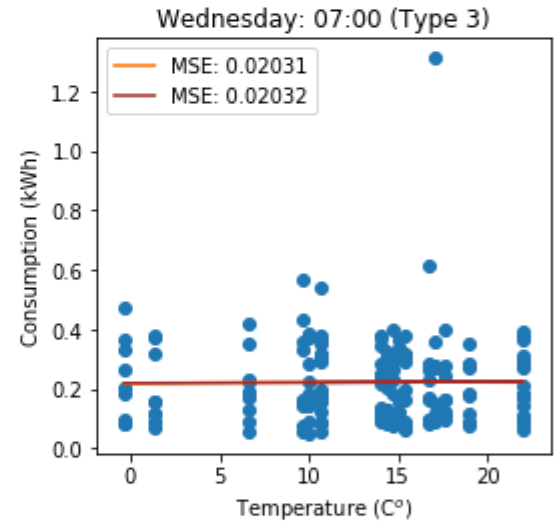

Wednesday: 06:00 (Type 1)
MSE: 0.01331
MSE: 0.01333
Consumption (kWh)
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
Temperature (C°)

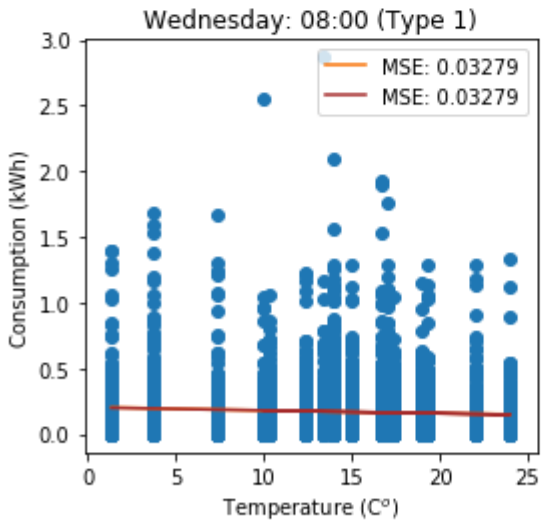

Wednesday: 06:00 (Type 2)
MSE: 0.04968
MSE: 0.04969
Consumption (kWh)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
Temperature (C°)

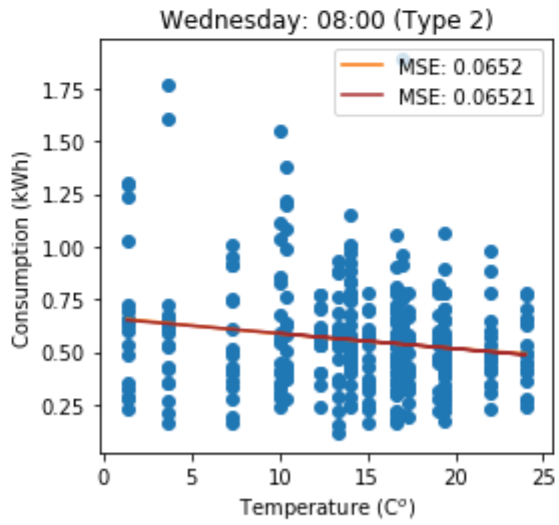

Wednesday: 06:00 (Type 3)
MSE: 0.02272
MSE: 0.02284
Consumption (kWh)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Temperature (C°)

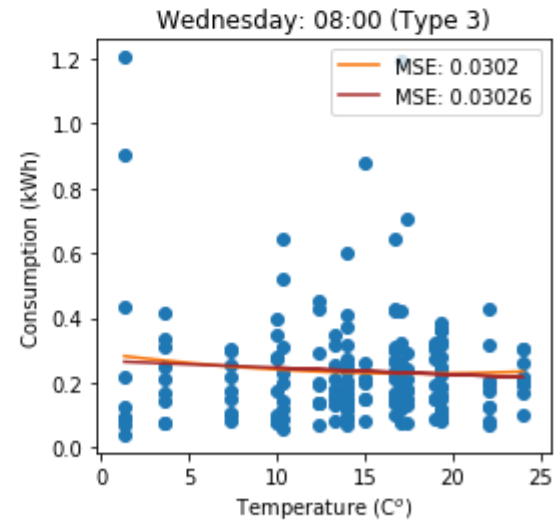

Wednesday: 07:00 (Type 1)
MSE: 0.02006
MSE: 0.02007
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
Temperature (C°)

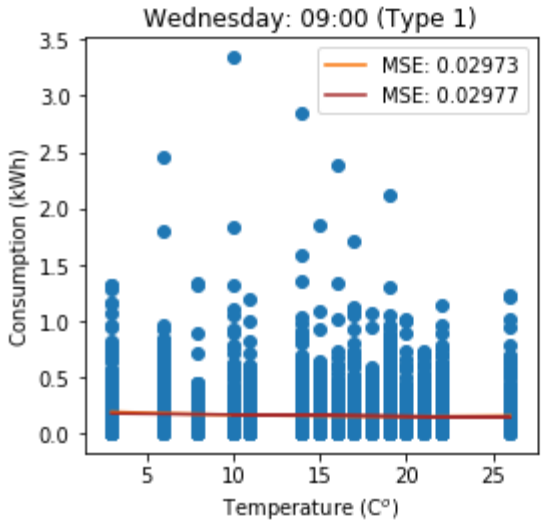

Wednesday: 07:00 (Type 2)
MSE: 0.075
MSE: 0.07525
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
5
10
15
20
Temperature (C°)

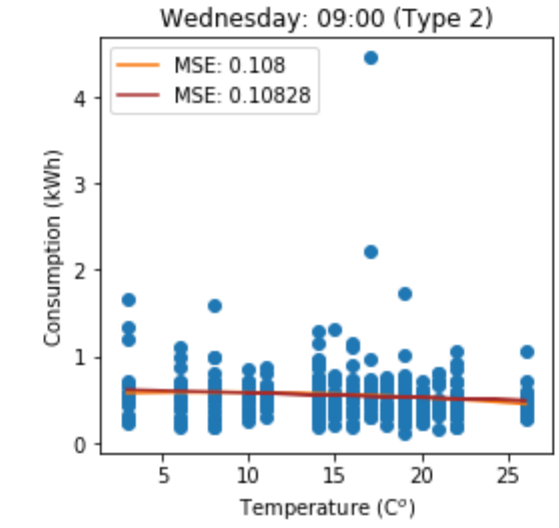

Wednesday: 07:00 (Type 3)
MSE: 0.02031
MSE: 0.02032
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Temperature (C°)

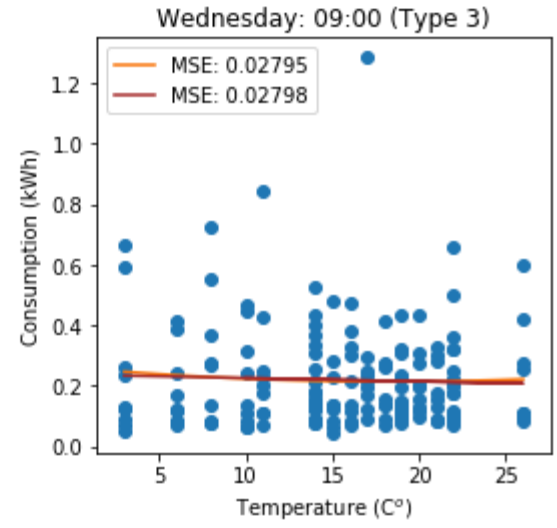

Wednesday: 08:00 (Type 1)
MSE: 0.03279
MSE: 0.03279
Consumption (kWh)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
25
Temperature (C°)

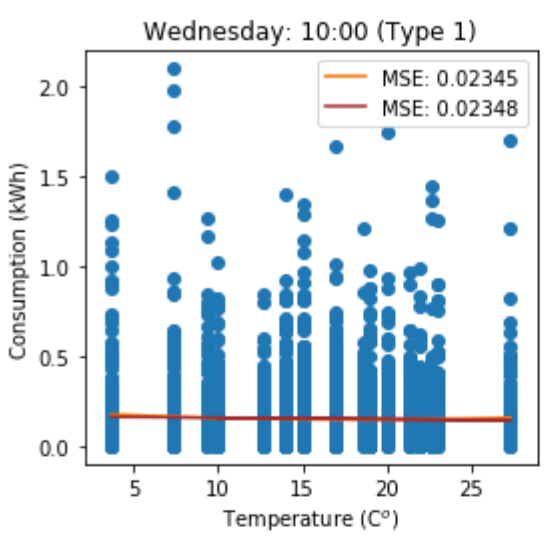

Wednesday: 08:00 (Type 2)
MSE: 0.0652
MSE: 0.06521
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
5
10
15
20
25
Temperature (C°)

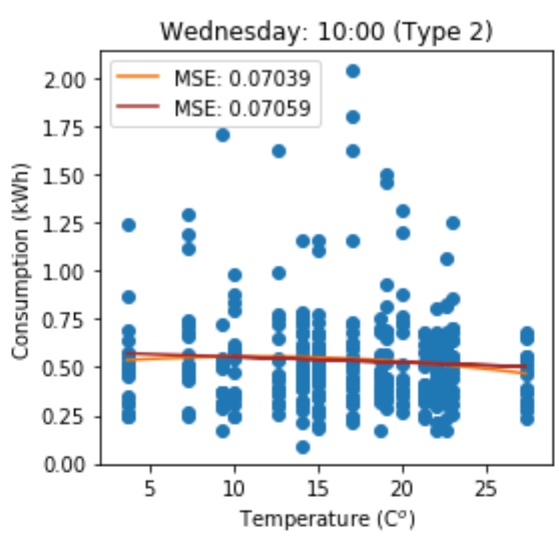

Wednesday: 08:00 (Type 3)
MSE: 0.0302
MSE: 0.03026
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
Temperature (C°)

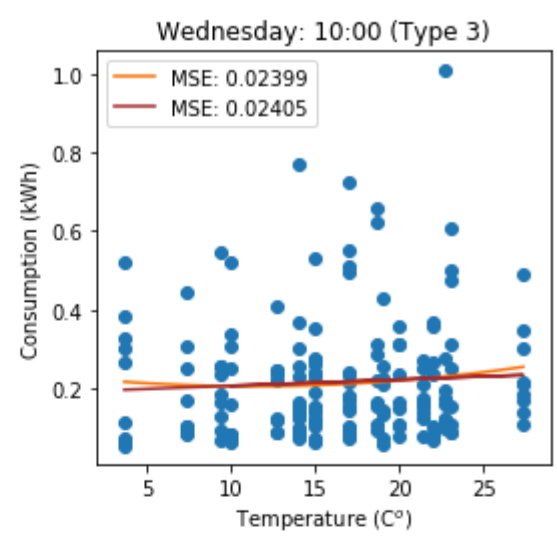

Wednesday: 09:00 (Type 1)
MSE: 0.02973
MSE: 0.02977
Consumption (kWh)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
25
Temperature (C°)

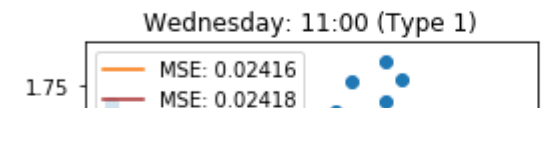

Wednesday: 09:00 (Type 2)
MSE: 0.108
MSE: 0.10828
Consumption (kWh)
4
3
2
1
0
5
10
15
20
25
Temperature (C°)

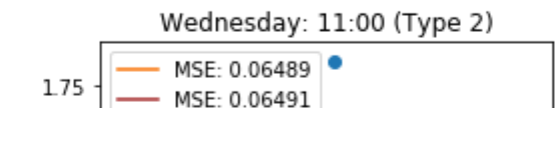

Wednesday: 09:00 (Type 3)
MSE: 0.02795
MSE: 0.02798
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
Temperature (C°)

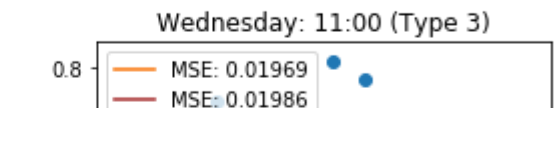

Wednesday: 10:00 (Type 1)
MSE: 0.02345
MSE: 0.02348
Consumption (kWh)
2.0
1.5
1.0
0.5
0.0
5
10
15
20
25
Temperature (C°)

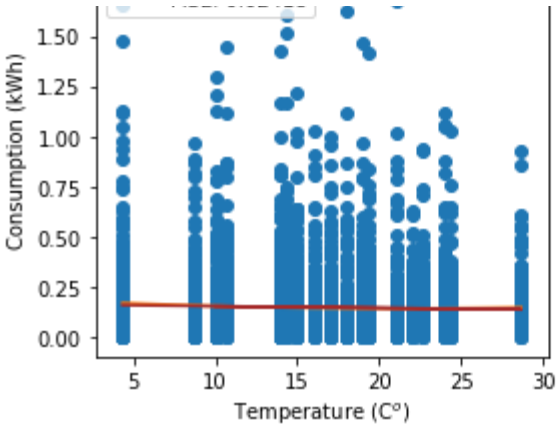

Wednesday: 10:00 (Type 2)
MSE: 0.07039
MSE: 0.07059
Consumption (kWh)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
5
10
15
20
25
Temperature (C°)

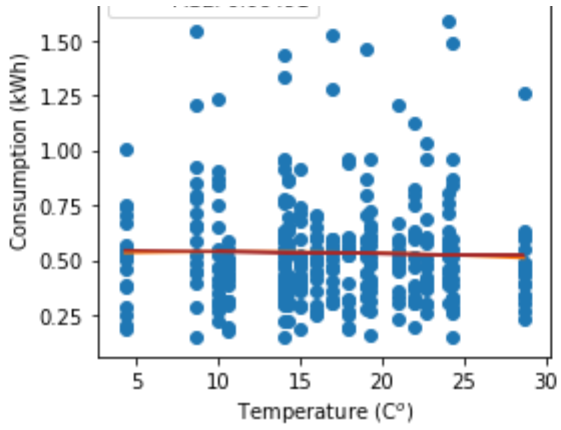

Wednesday: 10:00 (Type 3)
MSE: 0.02399
MSE: 0.02405
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
5
10
15
20
25
Temperature (C°)

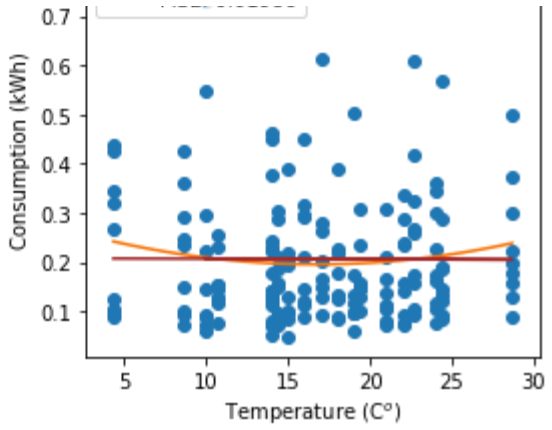

Wednesday: 11:00 (Type 1)
MSE: 0.02416
MSE: 0.02418
1.75

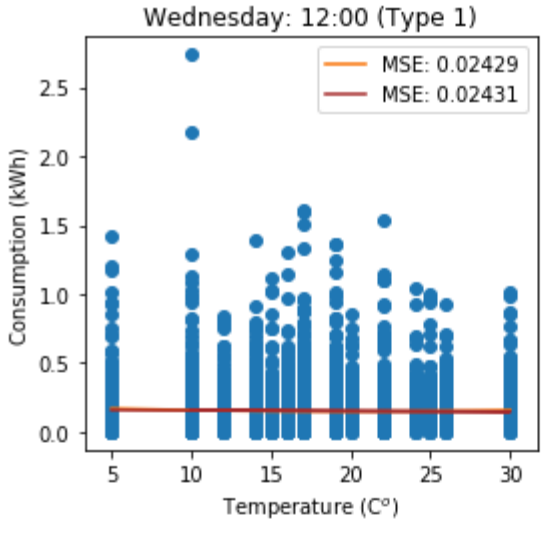

Wednesday: 11:00 (Type 2)
MSE: 0.06489
MSE: 0.06491
1.75

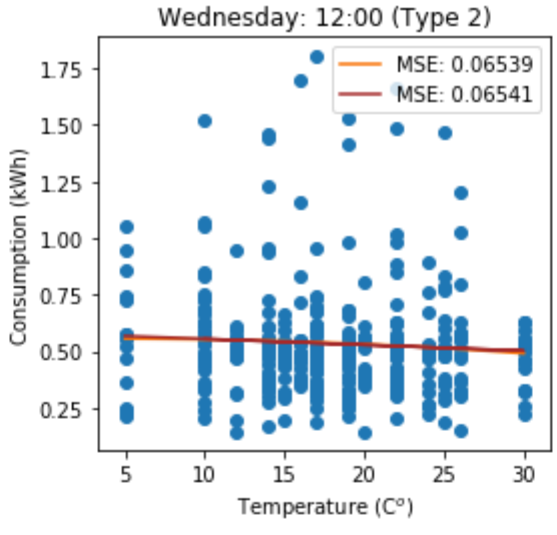

Wednesday: 11:00 (Type 3)
MSE: 0.01969
MSE: 0.01986
0.8

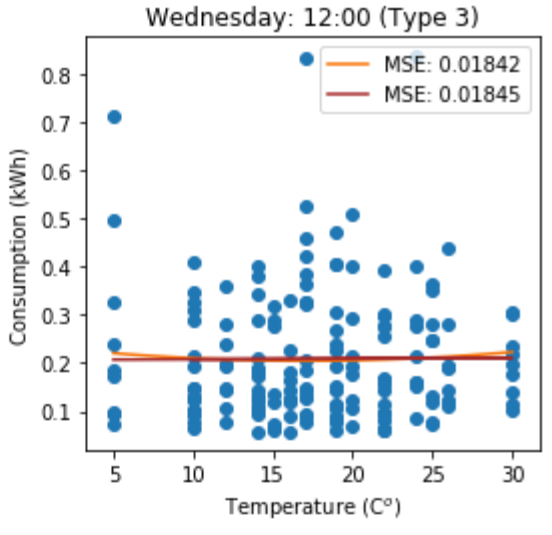

Consumption (kWh)
Temperature (C°)

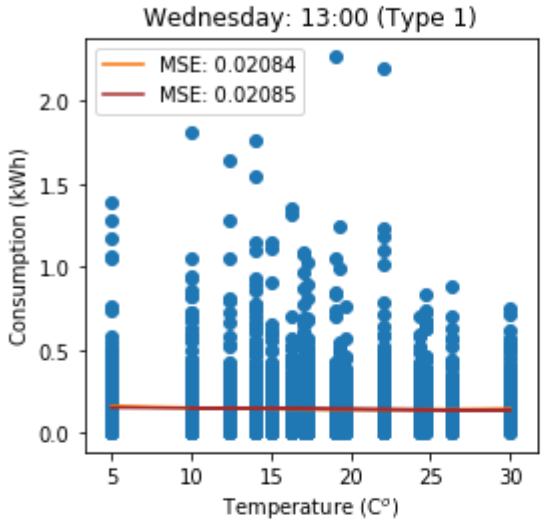

Consumption (kWh)
Temperature (C°)

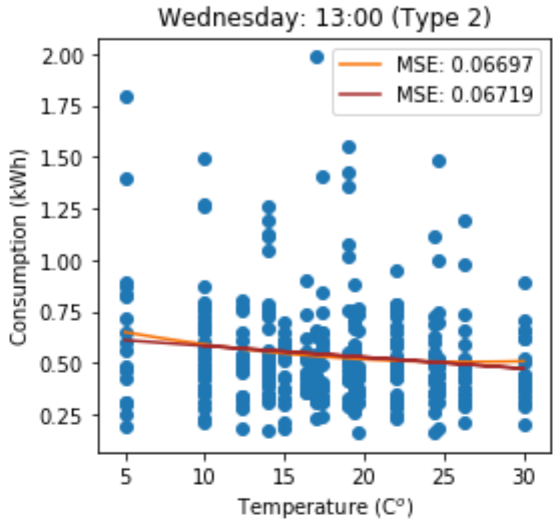

Consumption (kWh)
Temperature (C°)

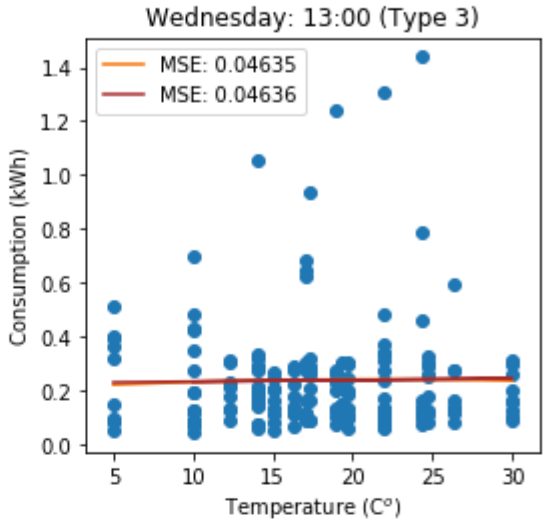

Wednesday: 12:00 (Type 1)
MSE: 0.02429
MSE: 0.02431
Consumption (kWh)
Temperature (C°)

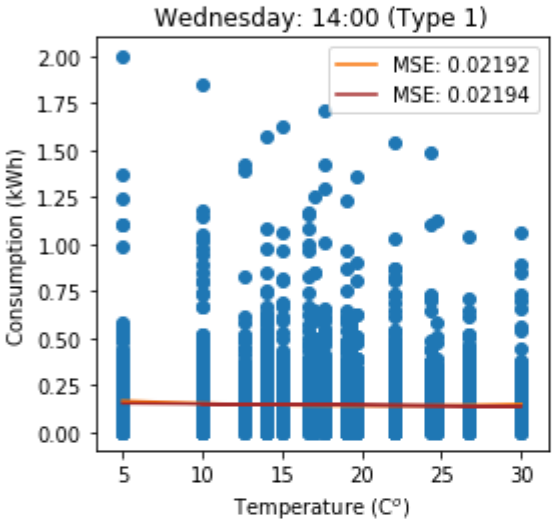

Wednesday: 12:00 (Type 2)
MSE: 0.06539
MSE: 0.06541
Consumption (kWh)
Temperature (C°)

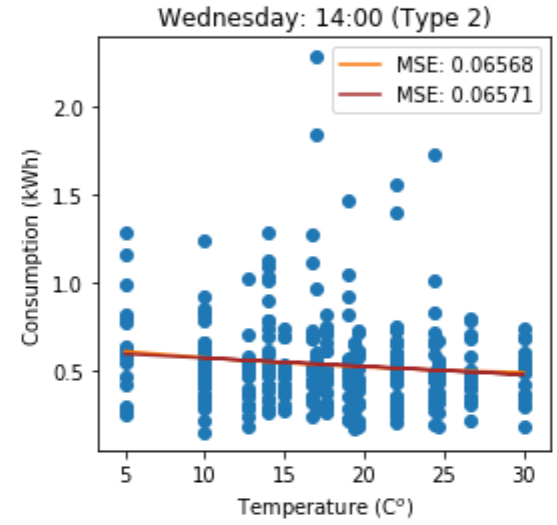

Wednesday: 12:00 (Type 3)
MSE: 0.01842
MSE: 0.01845
Consumption (kWh)
Temperature (C°)

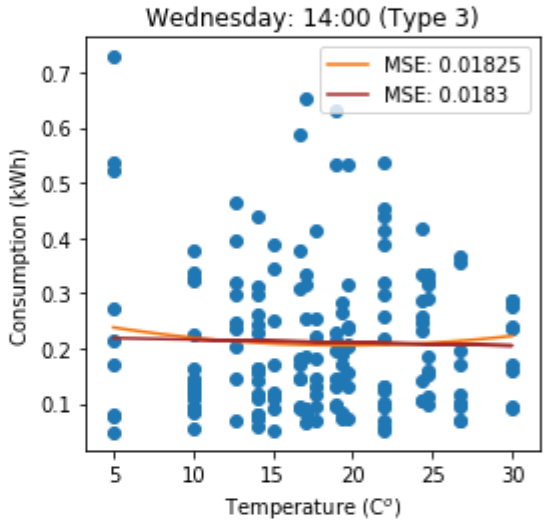

Wednesday: 13:00 (Type 1)
MSE: 0.02084
MSE: 0.02085
Consumption (kWh)
Temperature (C°)

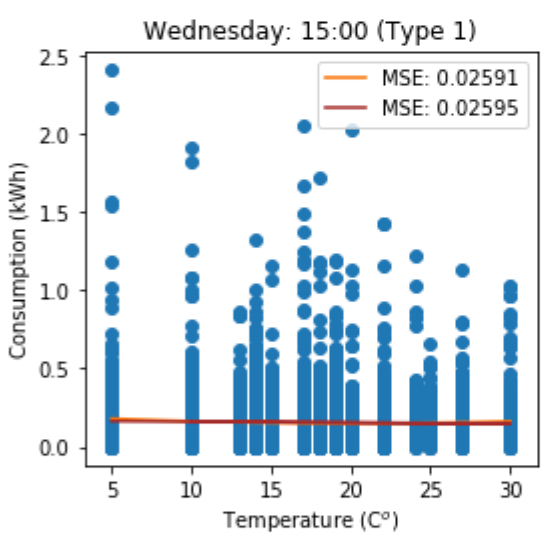

Wednesday: 13:00 (Type 2)
MSE: 0.06697
MSE: 0.06719
Consumption (kWh)
Temperature (C°)

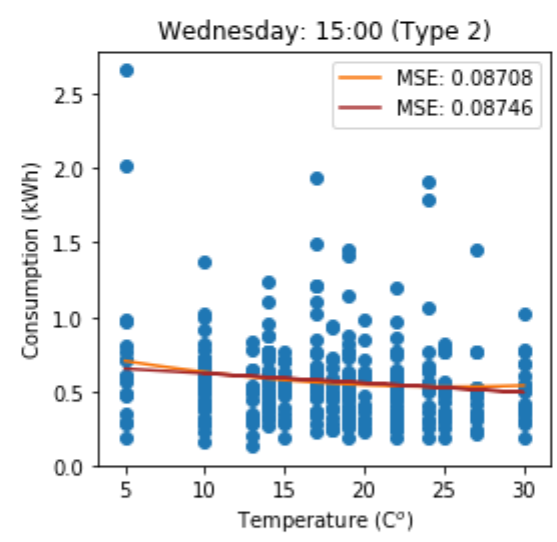

Wednesday: 13:00 (Type 3)
MSE: 0.04635
MSE: 0.04636
Consumption (kWh)
Temperature (C°)

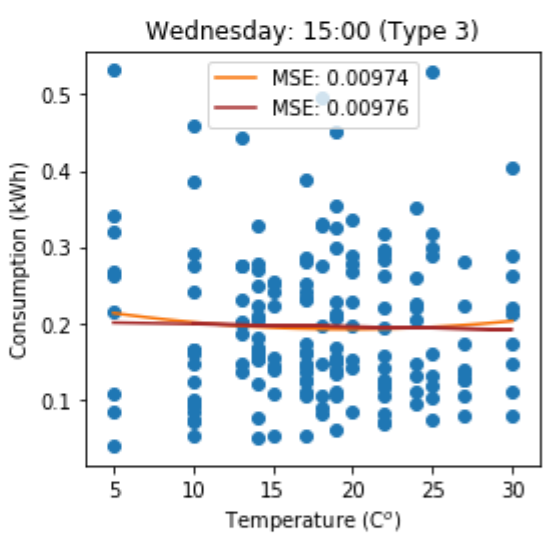

Wednesday: 14:00 (Type 1)
MSE: 0.02192
MSE: 0.02194
Consumption (kWh)
Temperature (C°)

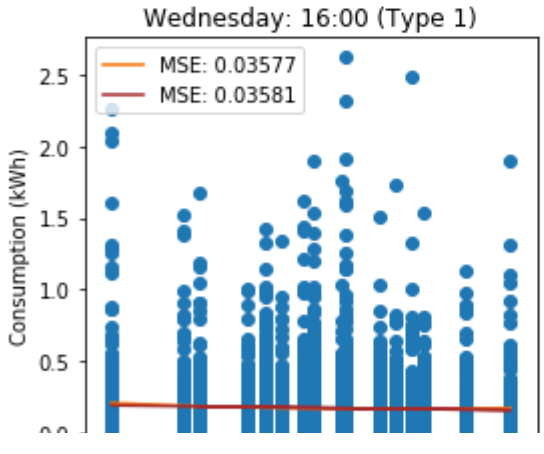

Wednesday: 14:00 (Type 2)
MSE: 0.06568
MSE: 0.06571
Consumption (kWh)
Temperature (C°)

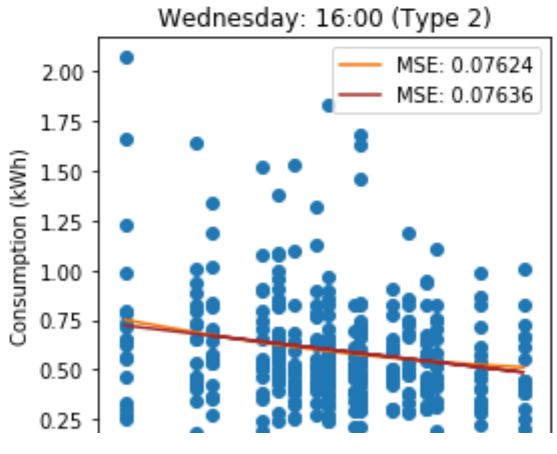

Wednesday: 14:00 (Type 3)
MSE: 0.01825
MSE: 0.0183
Consumption (kWh)
Temperature (C°)

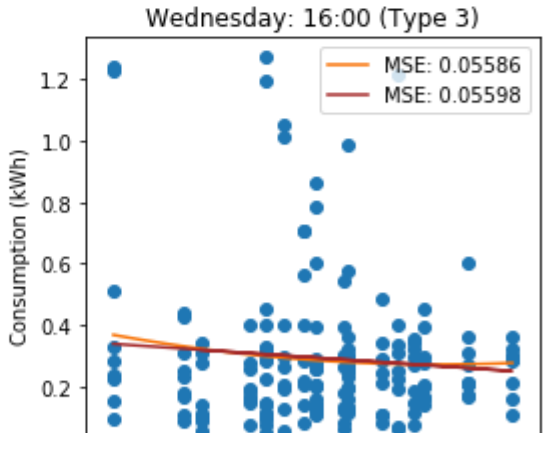

Wednesday: 15:00 (Type 1)
MSE: 0.02591
MSE: 0.02595
Consumption (kWh)
Temperature (C°)

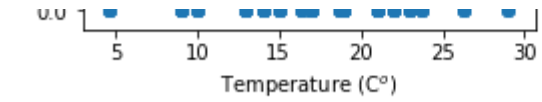

Wednesday: 15:00 (Type 2)
MSE: 0.08708
MSE: 0.08746
Consumption (kWh)
Temperature (C°)

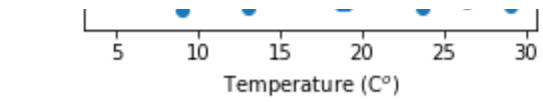

Wednesday: 15:00 (Type 3)
MSE: 0.00974
MSE: 0.00976
Consumption (kWh)
Temperature (C°)

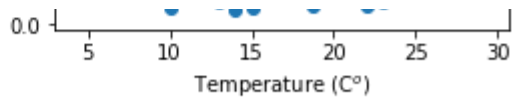

Wednesday: 16:00 (Type 1)
MSE: 0.03577
MSE: 0.03581
Consumption (kWh)

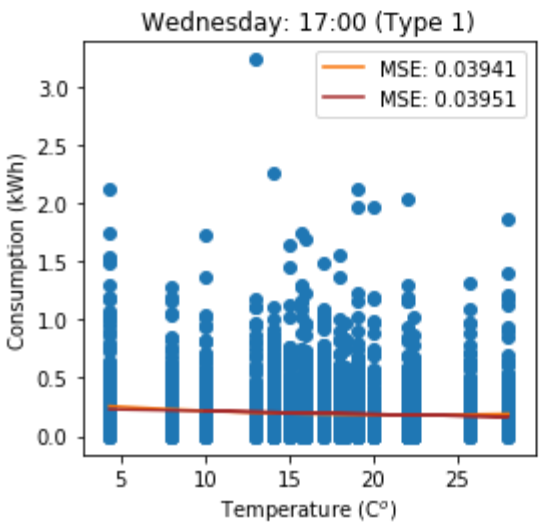

Wednesday: 16:00 (Type 2)
MSE: 0.07624
MSE: 0.07636
Consumption (kWh)

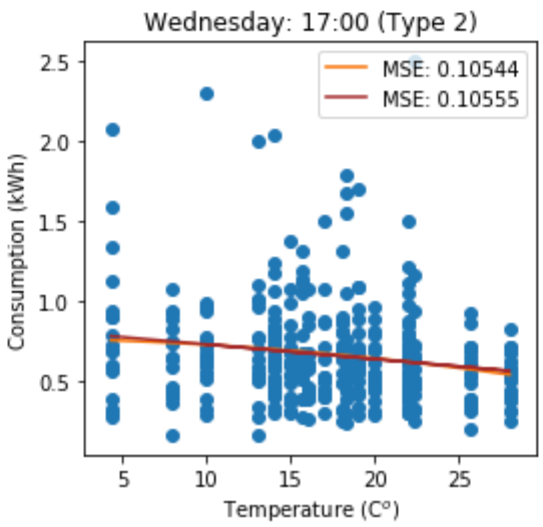

Wednesday: 16:00 (Type 3)
MSE: 0.05586
MSE: 0.05598
Consumption (kWh)

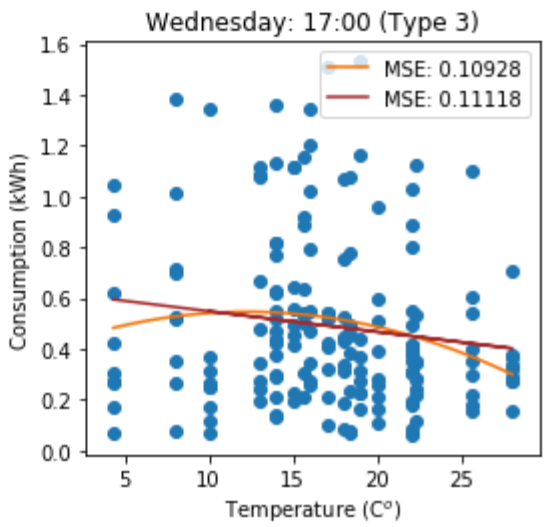

Temperature (C°)

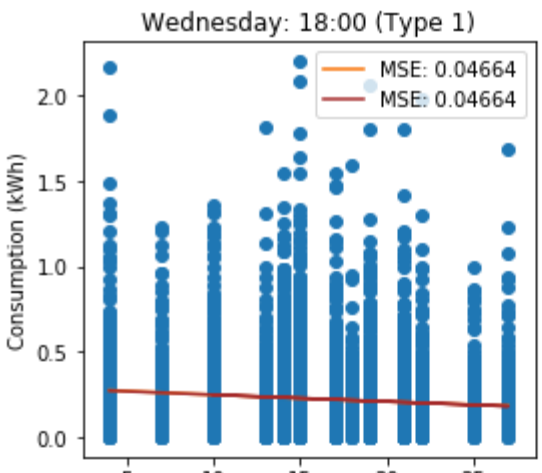

Temperature (C°)

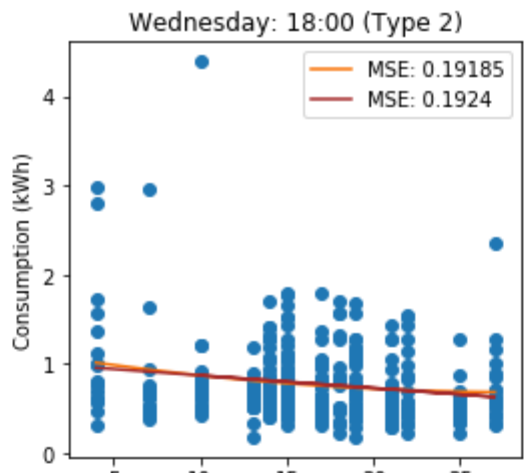

Temperature (C°)

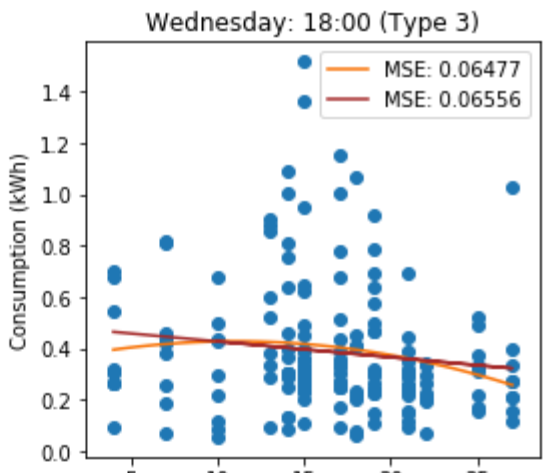

Wednesday: 17:00 (Type 1)
MSE: 0.03941
MSE: 0.03951
Consumption (kWh)
Temperature (C°)

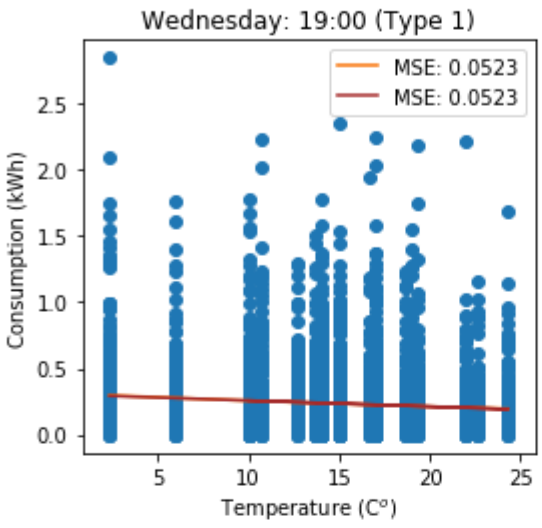

Wednesday: 17:00 (Type 2)
MSE: 0.10544
MSE: 0.10555
Consumption (kWh)
Temperature (C°)

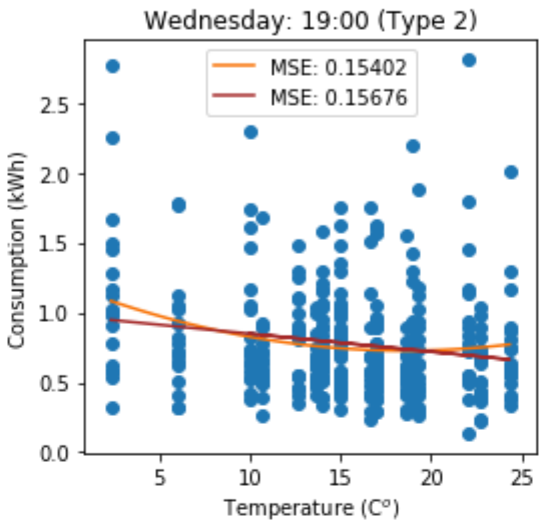

Wednesday: 17:00 (Type 3)
MSE: 0.10928
MSE: 0.11118
Consumption (kWh)
Temperature (C°)

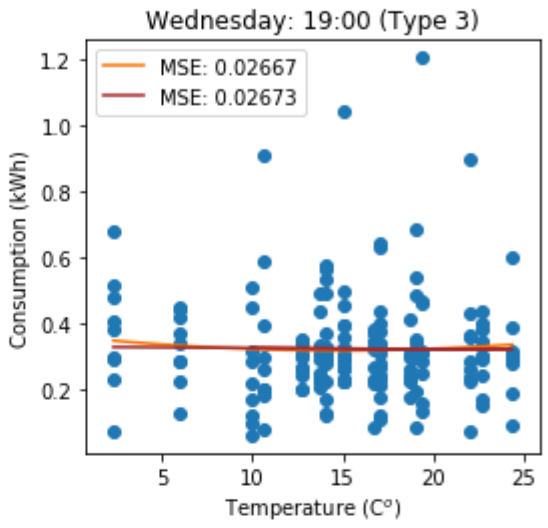

Wednesday: 18:00 (Type 1)
MSE: 0.04664
MSE: 0.04664
Consumption (kWh)
Temperature (C°)

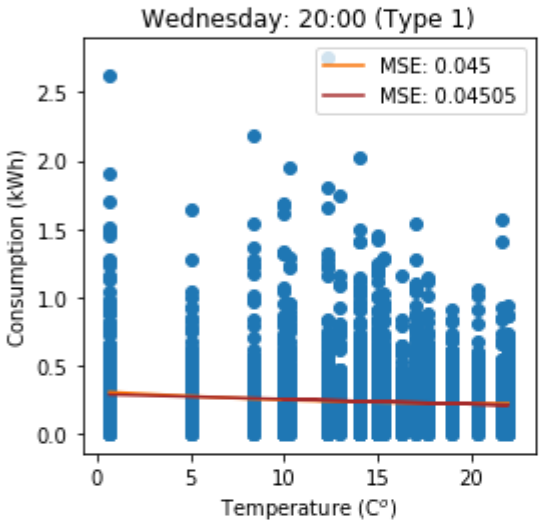

Wednesday: 18:00 (Type 2)
MSE: 0.19185
MSE: 0.1924
Consumption (kWh)
Temperature (C°)

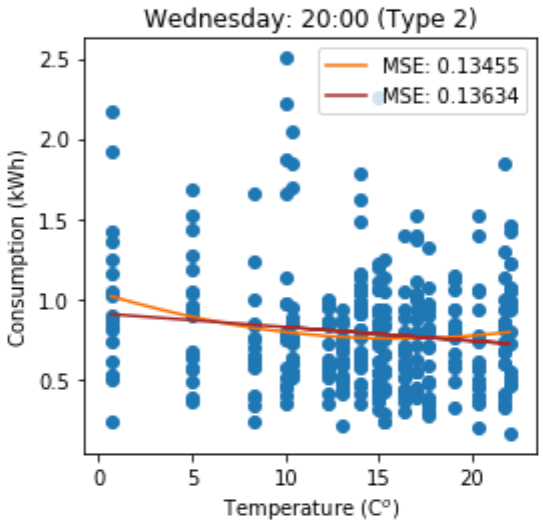

Wednesday: 18:00 (Type 3)
MSE: 0.06477
MSE: 0.06556
Consumption (kWh)
Temperature (C°)

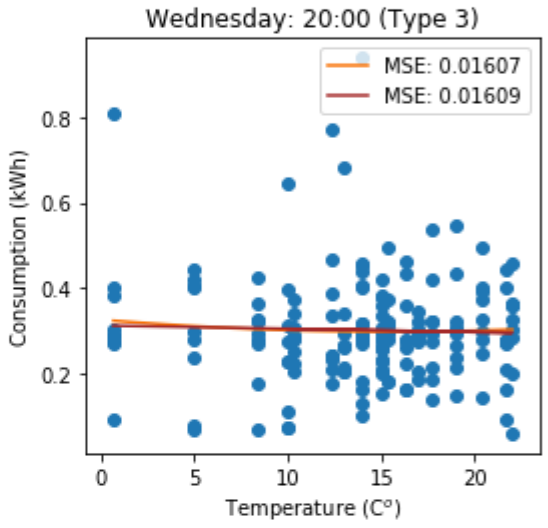

Wednesday: 19:00 (Type 1)
MSE: 0.0523
MSE: 0.0523
Consumption (kWh)
Temperature (C°)

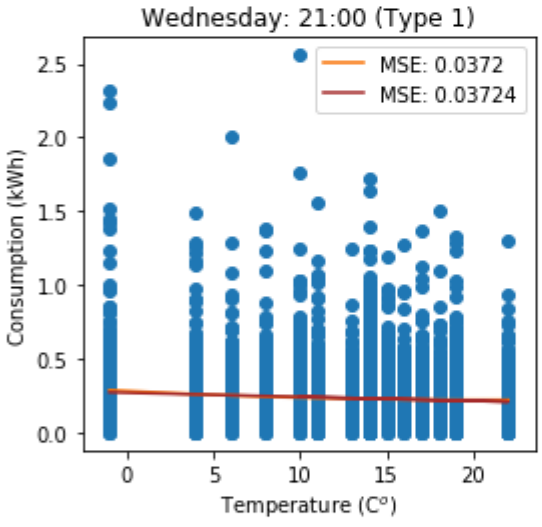

Wednesday: 19:00 (Type 2)
MSE: 0.15402
MSE: 0.15676
Consumption (kWh)
Temperature (C°)

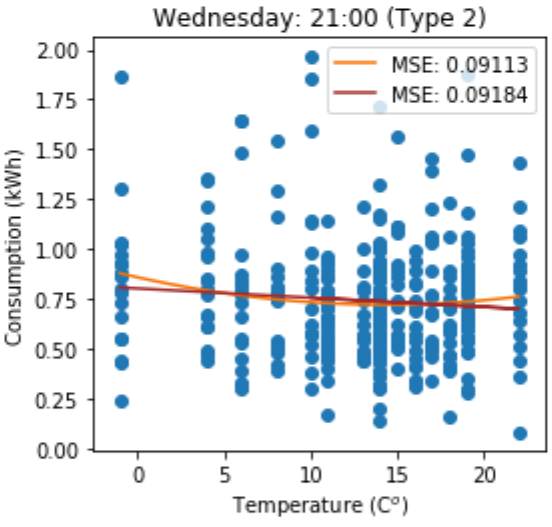

Wednesday: 19:00 (Type 3)
MSE: 0.02667
MSE: 0.02673
Consumption (kWh)
Temperature (C°)

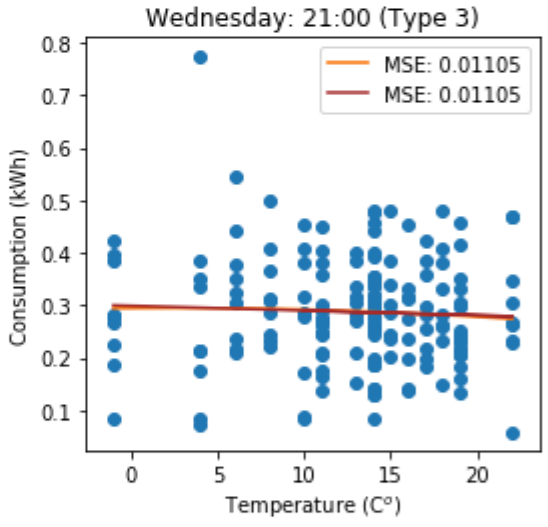

Wednesday: 20:00 (Type 1)
MSE: 0.045
MSE: 0.04505
Consumption (kWh)
Temperature (C°)

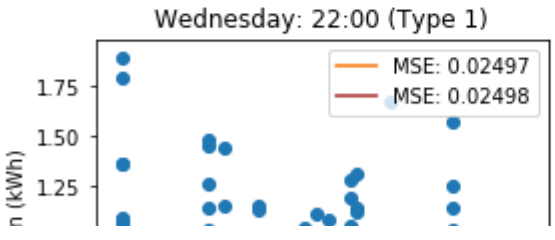

Wednesday: 20:00 (Type 2)
MSE: 0.13455
MSE: 0.13634
Consumption (kWh)
Temperature (C°)

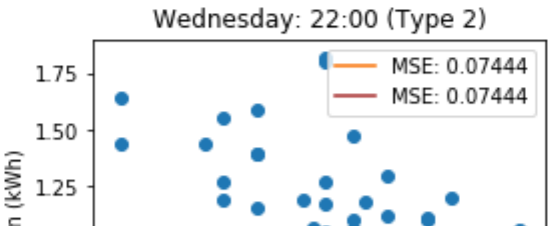

Wednesday: 20:00 (Type 3)
MSE: 0.01607
MSE: 0.01609
Consumption (kWh)
Temperature (C°)

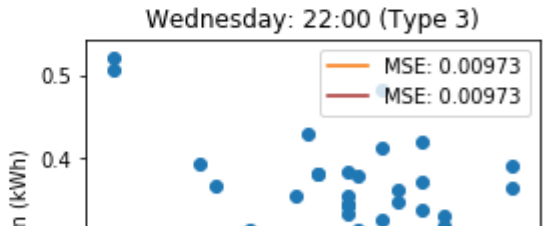

Wednesday: 21:00 (Type 1)
MSE: 0.0372
MSE: 0.03724
Consumption (kWh)
Temperature (C°)

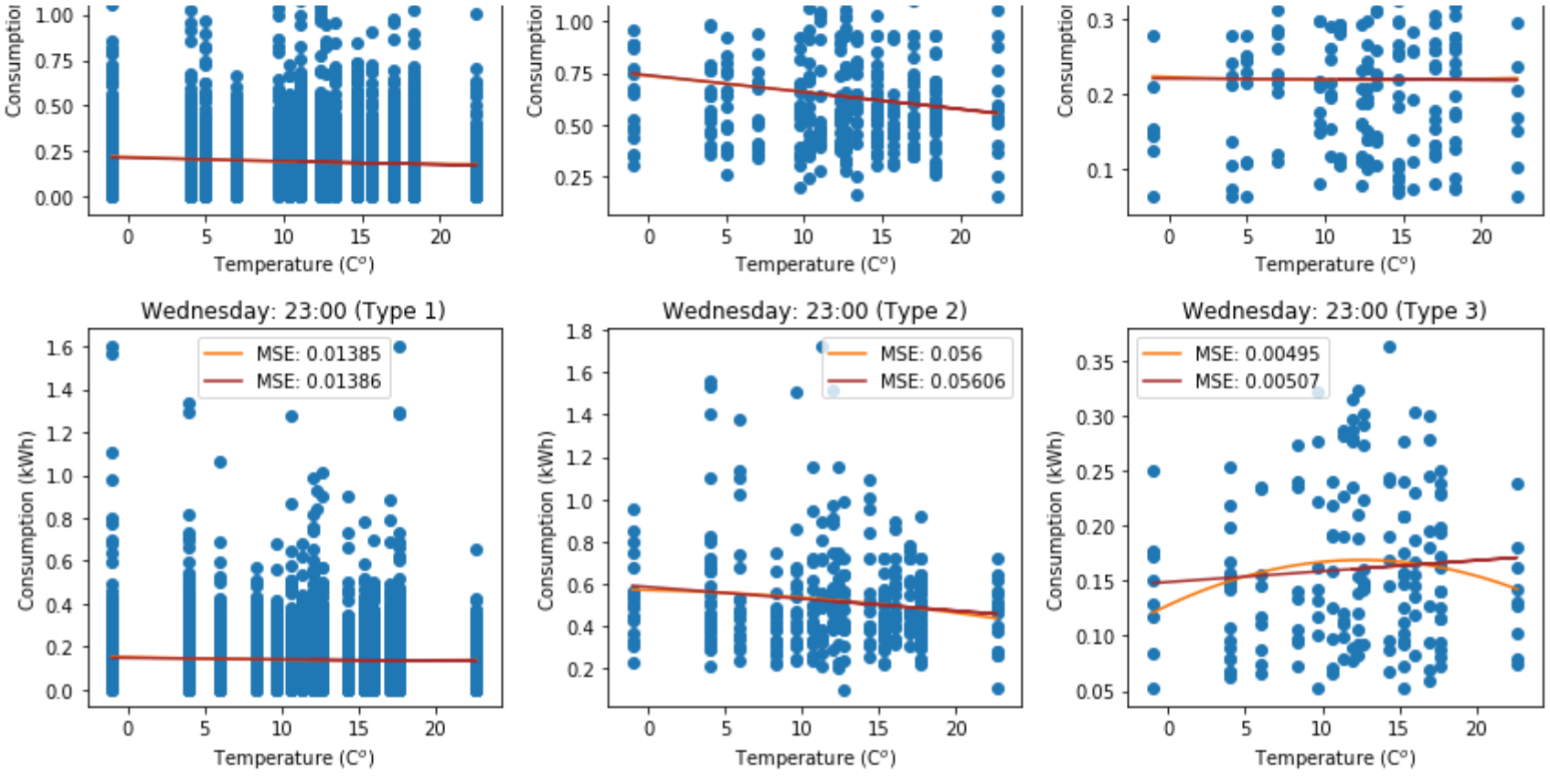

Wednesday: 21:00 (Type 2)
MSE: 0.09113
MSE: 0.09184
Consumption (kWh)
Temperature (C°)


Wednesday: 21:00 (Type 3)
MSE: 0.01105
MSE: 0.01105
Consumption (kWh)
Temperature (C°)

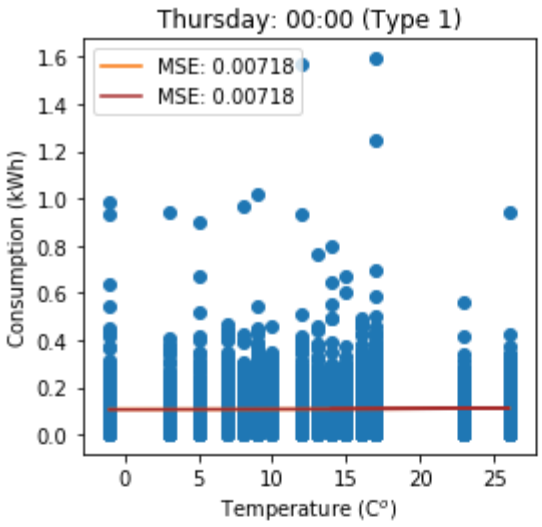

Wednesday: 22:00 (Type 1)
MSE: 0.02497
MSE: 0.02498

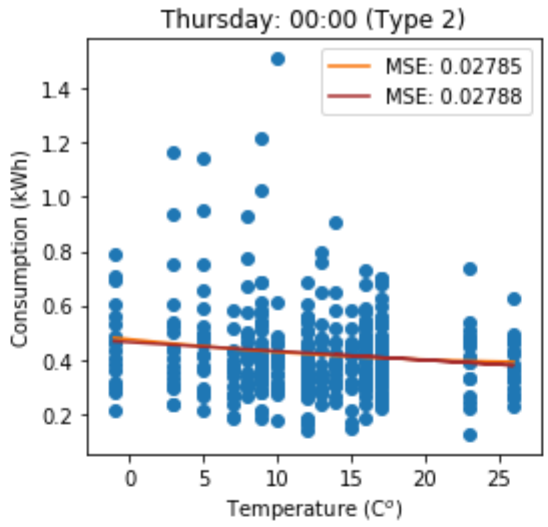

Wednesday: 22:00 (Type 2)
MSE: 0.07444
MSE: 0.07444

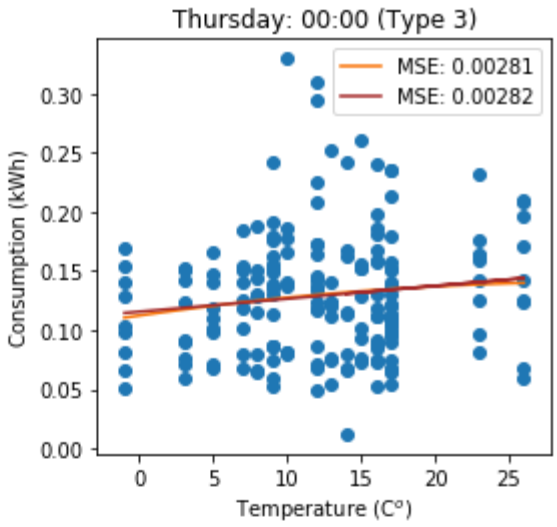

Wednesday: 22:00 (Type 3)
MSE: 0.00973
MSE: 0.00973

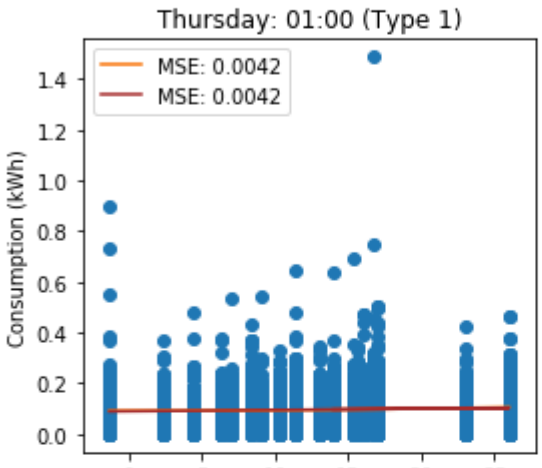

Wednesday: 23:00 (Type 1)
MSE: 0.01385
MSE: 0.01386
Wednesday: 23:00 (Type 2)
MSE: 0.056
MSE: 0.05606
Wednesday: 23:00 (Type 3)
MSE: 0.00495
MSE: 0.00507
Consumption (kWh)
Temperature (C°)

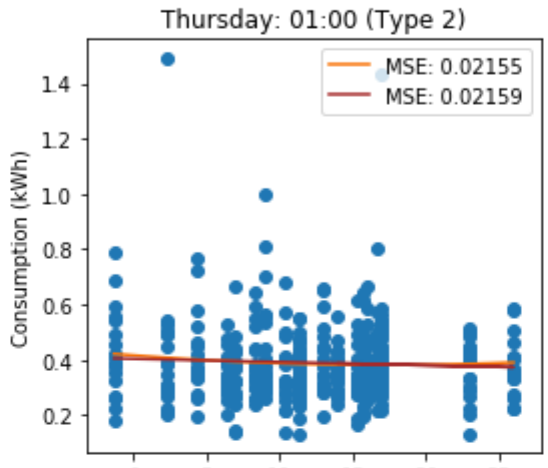
In [36]:
```
# Thursday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 3 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

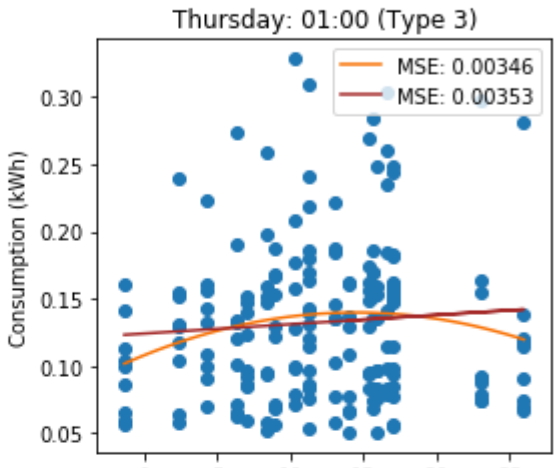

Thursday: 00:00 (Type 1)
MSE: 0.00718
MSE: 0.00718
Consumption (kWh)
Temperature (C°)

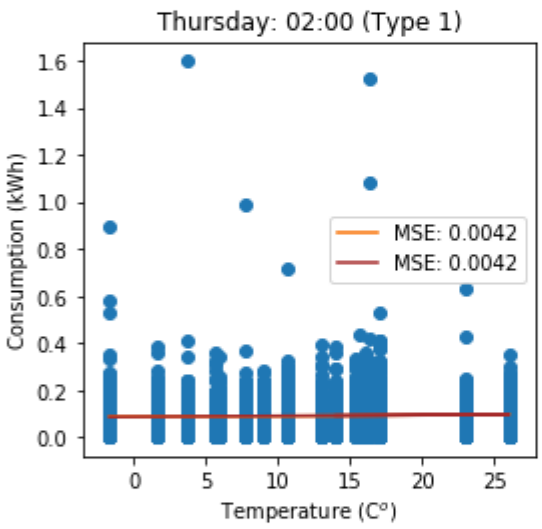

Thursday: 00:00 (Type 2)
MSE: 0.02785
MSE: 0.02788
Consumption (kWh)
Temperature (C°)

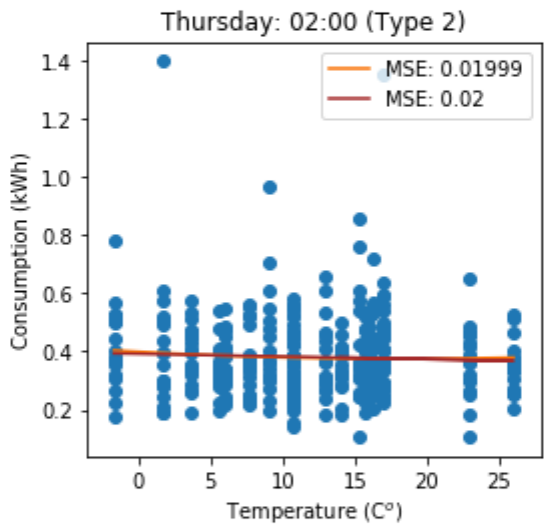

Thursday: 00:00 (Type 3)
MSE: 0.00281
MSE: 0.00282
Consumption (kWh)
Temperature (C°)

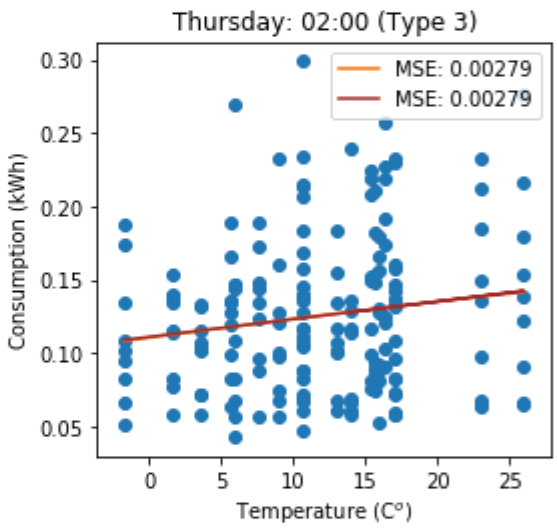

Thursday: 01:00 (Type 1)
MSE: 0.0042
MSE: 0.0042
Consumption (kWh)
Temperature (C°)

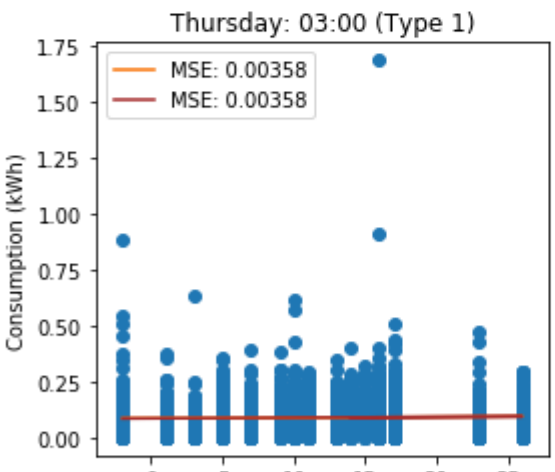

Thursday: 01:00 (Type 2)
MSE: 0.02155
MSE: 0.02159
Consumption (kWh)
Temperature (C°)

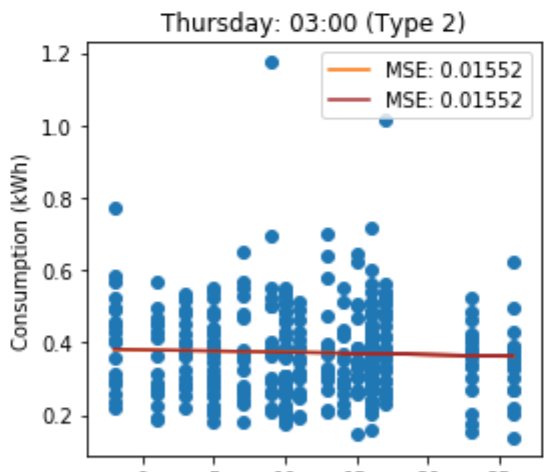

Thursday: 01:00 (Type 3)
MSE: 0.00346
MSE: 0.00353
Consumption (kWh)
Temperature (C°)

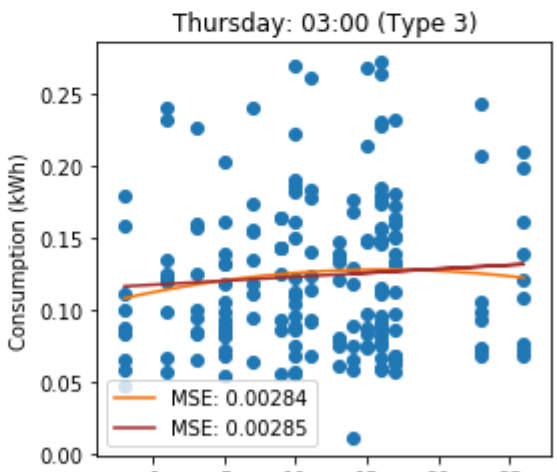

Thursday: 02:00 (Type 1)
MSE: 0.0042
MSE: 0.0042
Consumption (kWh)
Temperature (C°)

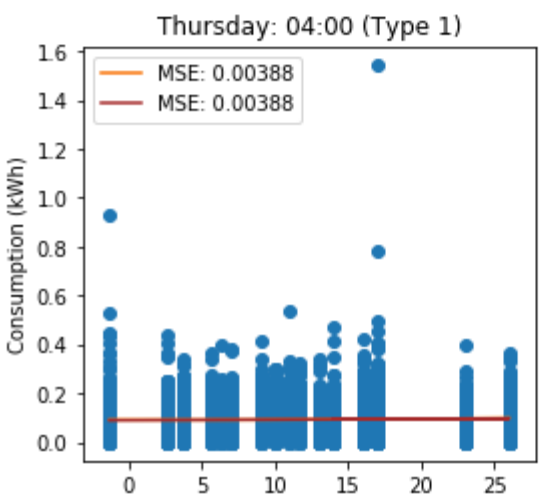

Thursday: 02:00 (Type 2)
MSE: 0.01999
MSE: 0.02
Consumption (kWh)
Temperature (C°)

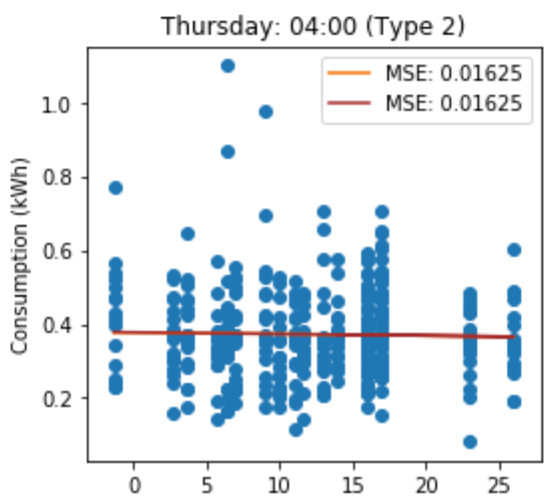

Thursday: 02:00 (Type 3)
MSE: 0.00279
MSE: 0.00279
Consumption (kWh)
Temperature (C°)

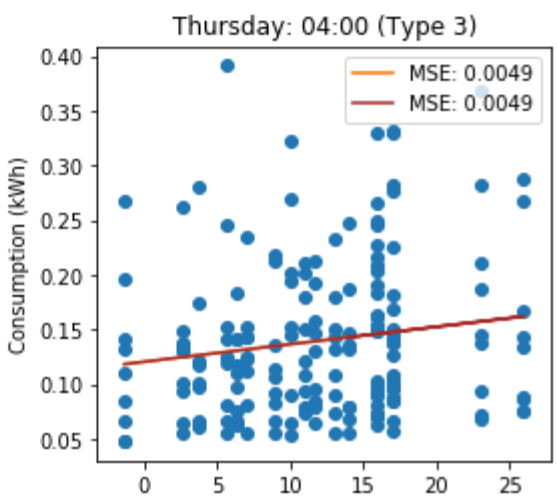

Thursday: 03:00 (Type 1)
MSE: 0.00358
MSE: 0.00358
Consumption (kWh)
Temperature (C°)

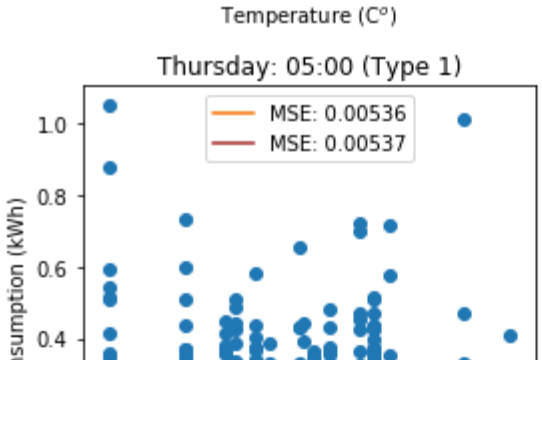

Thursday: 03:00 (Type 2)
MSE: 0.01552
MSE: 0.01552
Consumption (kWh)
Temperature (C°)

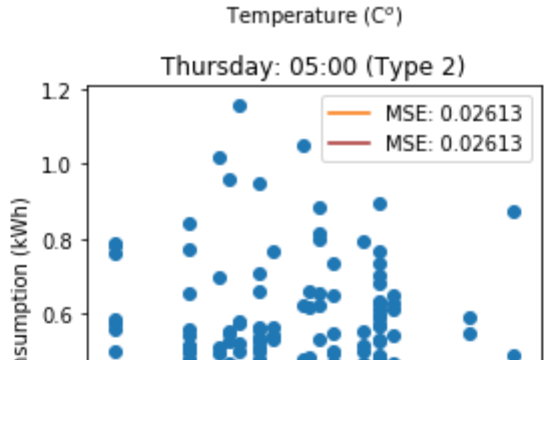

Thursday: 03:00 (Type 3)
MSE: 0.00284
MSE: 0.00285
Consumption (kWh)
Temperature (C°)

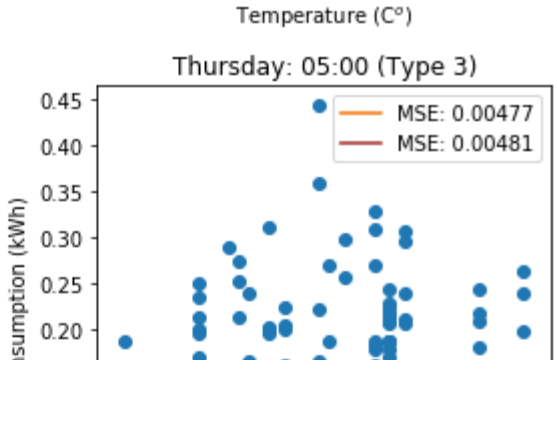

Thursday: 04:00 (Type 1)
MSE: 0.00388
MSE: 0.00388
Consumption (kWh)
Temperature (C°)

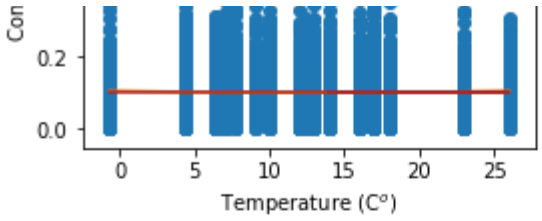

Thursday: 04:00 (Type 2)
MSE: 0.01625
MSE: 0.01625
Consumption (kWh)
Temperature (C°)

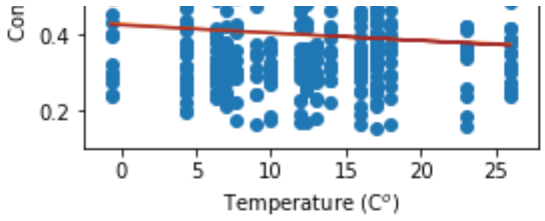

Thursday: 04:00 (Type 3)
MSE: 0.0049
MSE: 0.0049
Consumption (kWh)
Temperature (C°)

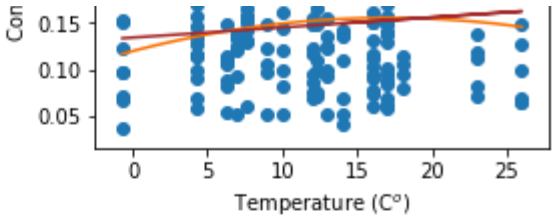

Thursday: 05:00 (Type 1)
MSE: 0.00536
MSE: 0.00537
Consumption (kWh)

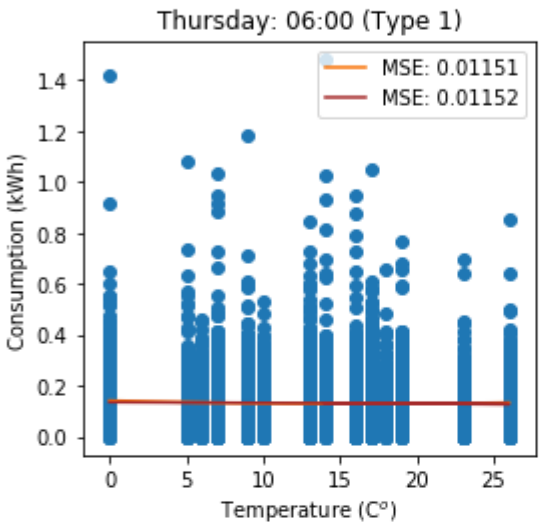

Thursday: 05:00 (Type 2)
MSE: 0.02613
MSE: 0.02613

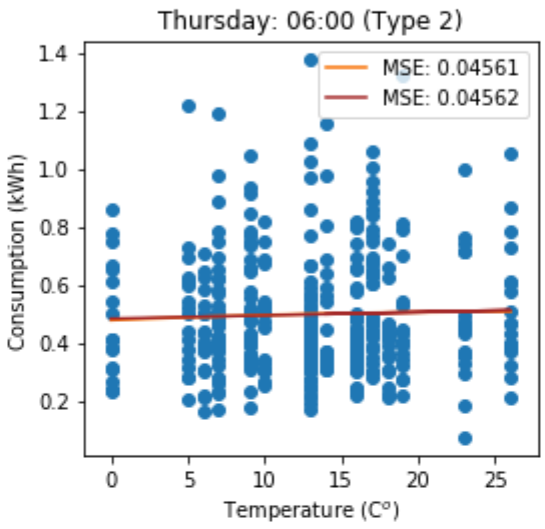

Thursday: 05:00 (Type 3)
MSE: 0.00477
MSE: 0.00481
Consumption (kWh)

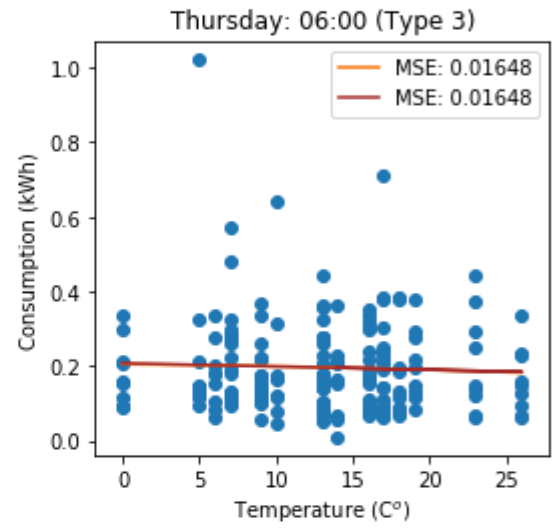

Temperature (C°)

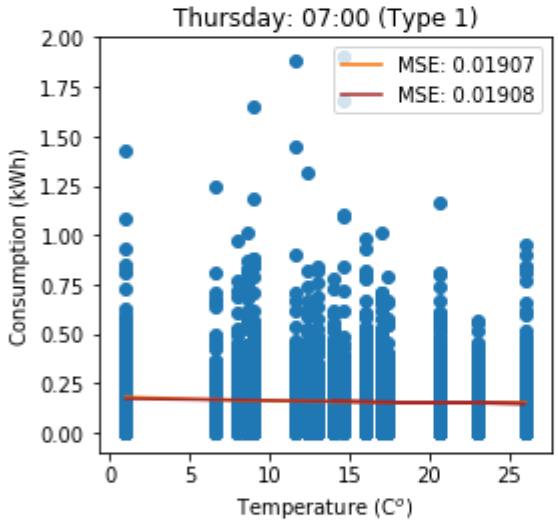

Temperature (C°)

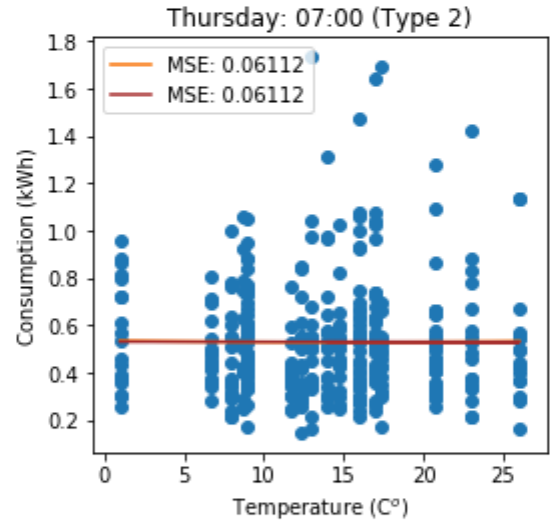

Temperature (C°)

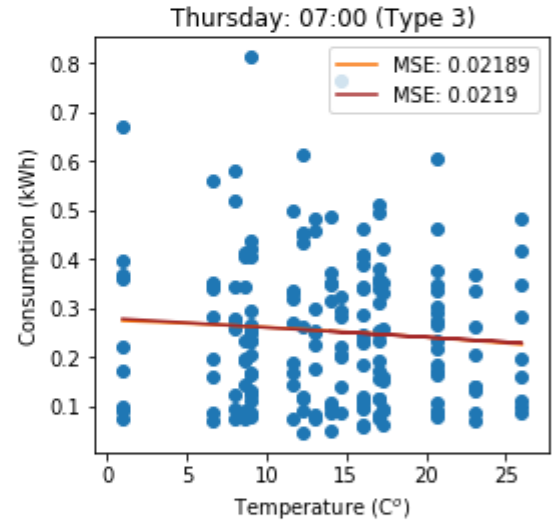

Thursday: 06:00 (Type 1)
MSE: 0.01151
MSE: 0.01152
Consumption (kWh)
Temperature (C°)

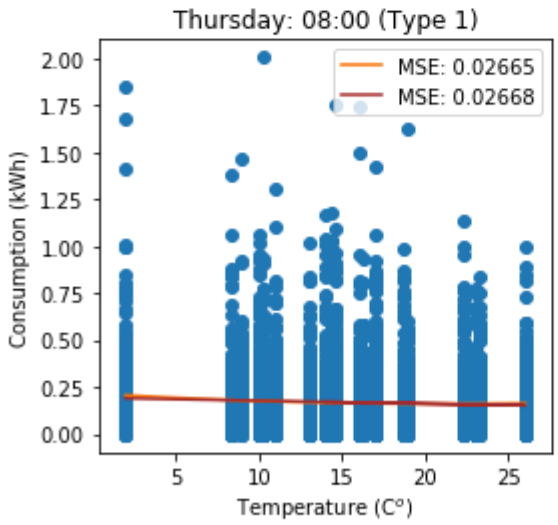

Thursday: 06:00 (Type 2)
MSE: 0.04561
MSE: 0.04562
Consumption (kWh)
Temperature (C°)

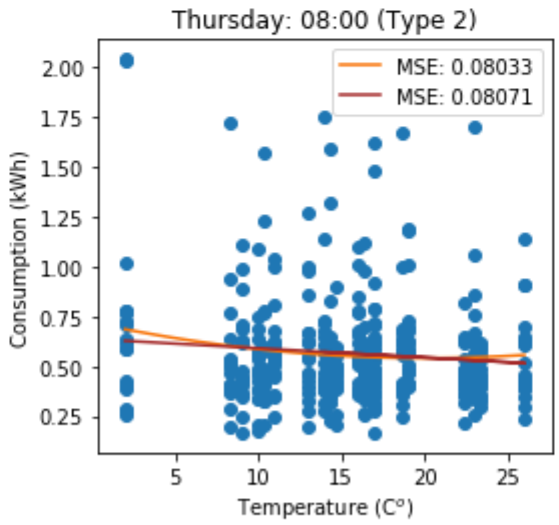

Thursday: 06:00 (Type 3)
MSE: 0.01648
MSE: 0.01648
Consumption (kWh)
Temperature (C°)

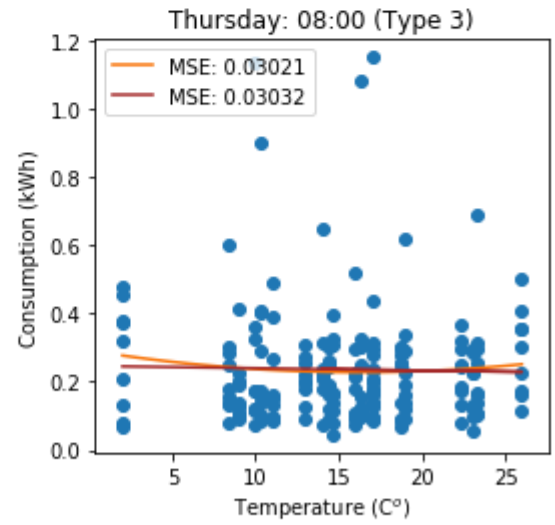

Thursday: 07:00 (Type 1)
MSE: 0.01907
MSE: 0.01908
Consumption (kWh)
Temperature (C°)

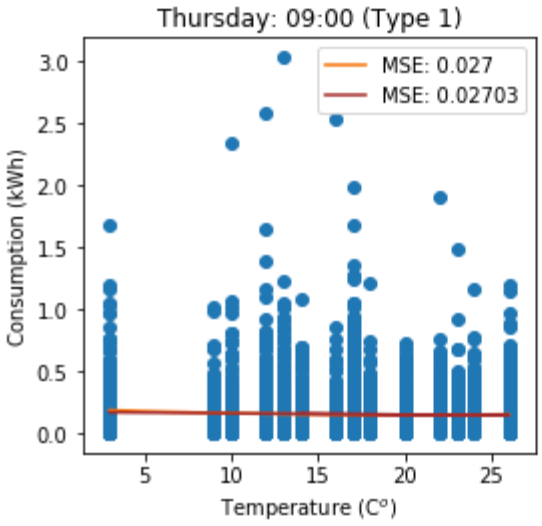

Thursday: 07:00 (Type 2)
MSE: 0.06112
MSE: 0.06112
Consumption (kWh)
Temperature (C°)

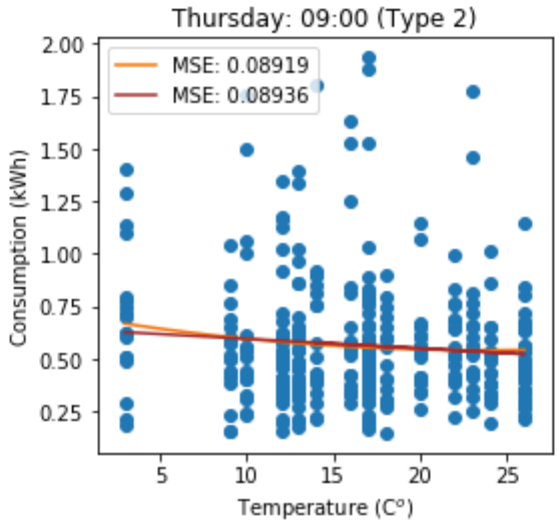

Thursday: 07:00 (Type 3)
MSE: 0.02189
MSE: 0.0219
Consumption (kWh)
Temperature (C°)

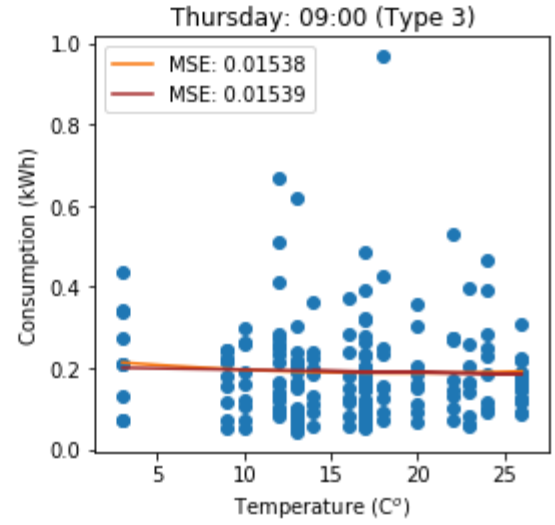

Thursday: 08:00 (Type 1)
MSE: 0.02665
MSE: 0.02668
Consumption (kWh)
Temperature (C°)

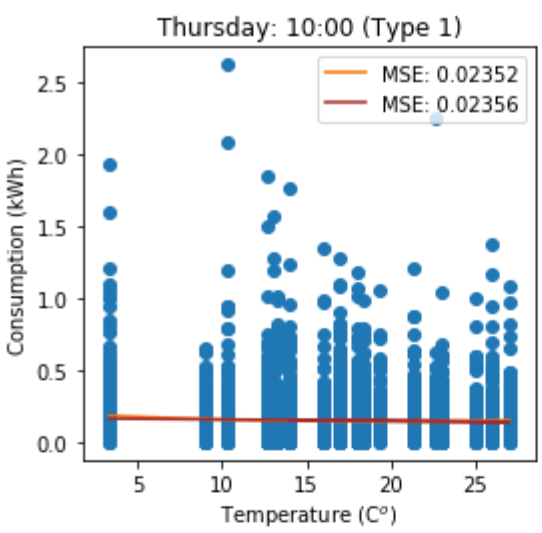

Thursday: 08:00 (Type 2)
MSE: 0.08033
MSE: 0.08071
Consumption (kWh)
Temperature (C°)

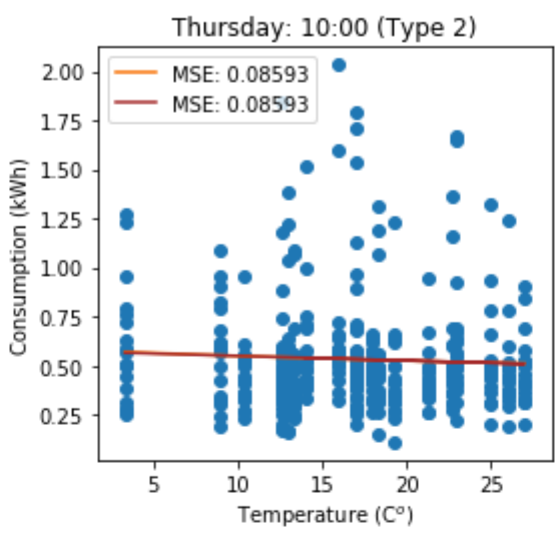

Thursday: 08:00 (Type 3)
MSE: 0.03021
MSE: 0.03032
Consumption (kWh)
Temperature (C°)

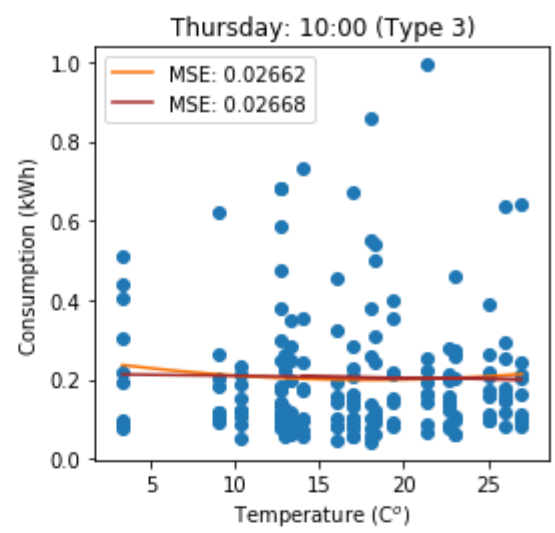

Thursday: 09:00 (Type 1)
MSE: 0.027
MSE: 0.02703
Consumption (kWh)
Temperature (C°)

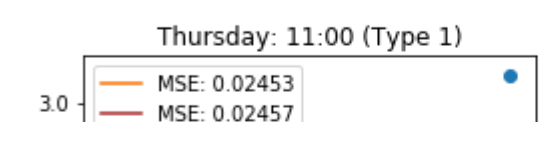

Thursday: 09:00 (Type 2)
MSE: 0.08919
MSE: 0.08936
Consumption (kWh)
Temperature (C°)


Thursday: 09:00 (Type 3)
MSE: 0.01538
MSE: 0.01539
Consumption (kWh)
Temperature (C°)


Thursday: 10:00 (Type 1)
MSE: 0.02352
MSE: 0.02356
Consumption (kWh)
Temperature (C°)

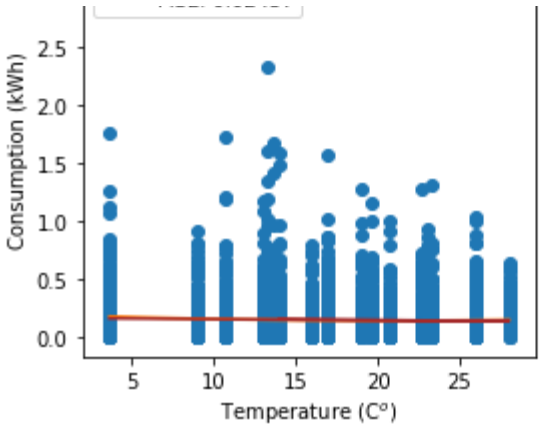

Thursday: 10:00 (Type 2)
MSE: 0.08593
MSE: 0.08593
Consumption (kWh)
Temperature (C°)

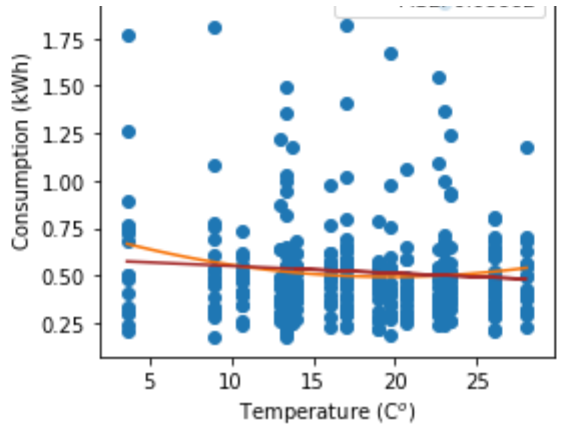

Thursday: 10:00 (Type 3)
MSE: 0.02662
MSE: 0.02668
Consumption (kWh)
Temperature (C°)

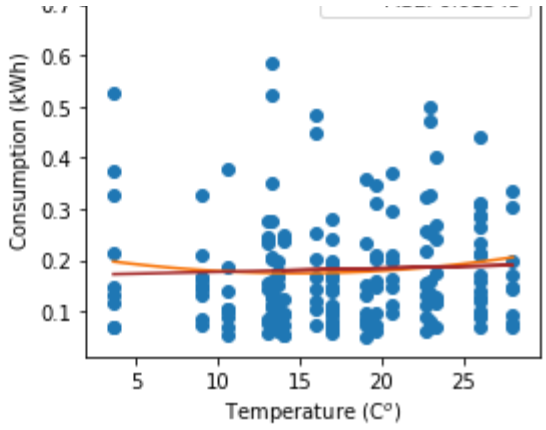

Thursday: 11:00 (Type 1)
MSE: 0.02453
MSE: 0.02457

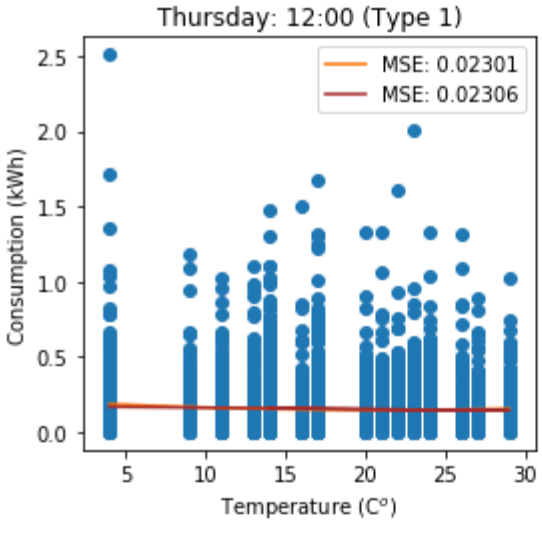

Thursday: 11:00 (Type 2)
MSE: 0.08567
MSE: 0.08662

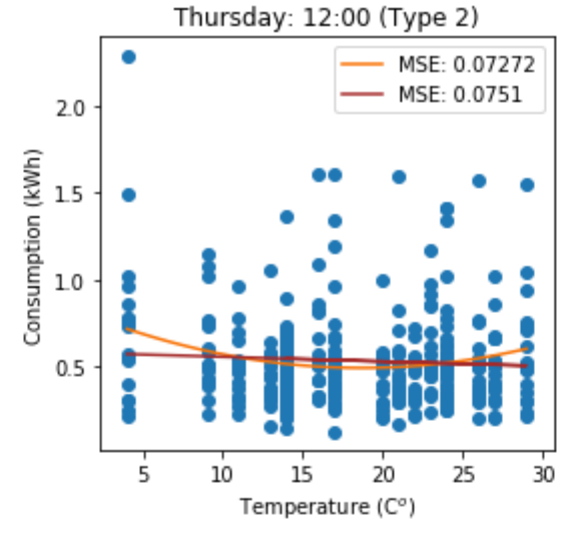

Thursday: 11:00 (Type 3)
MSE: 0.01538
MSE: 0.01545

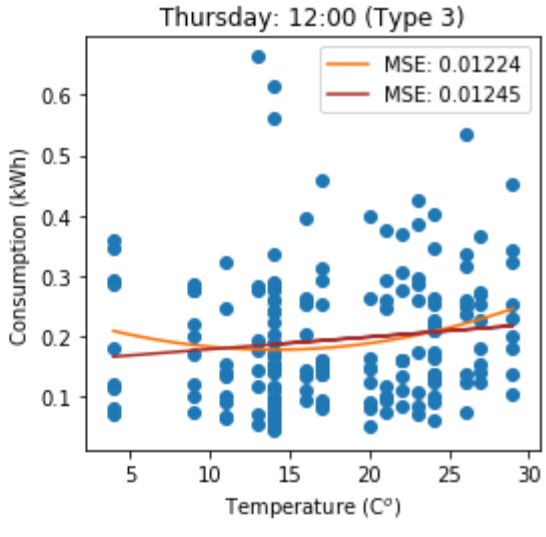

Consumption (kWh)
Temperature (C°)

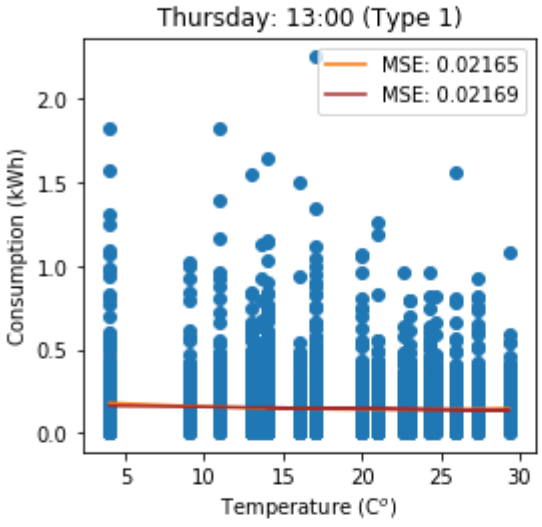

Consumption (kWh)
Temperature (C°)

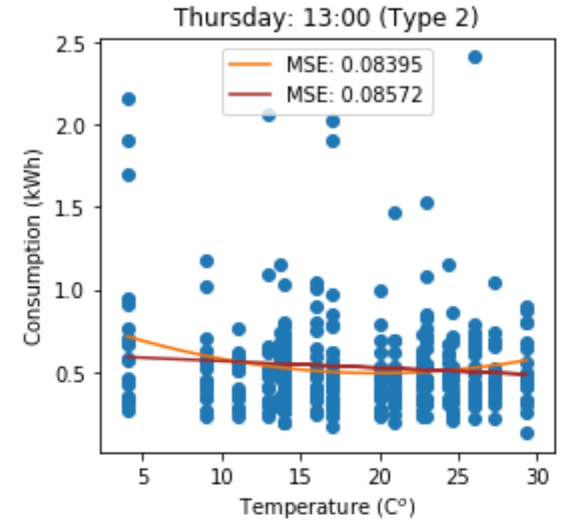

Consumption (kWh)
Temperature (C°)

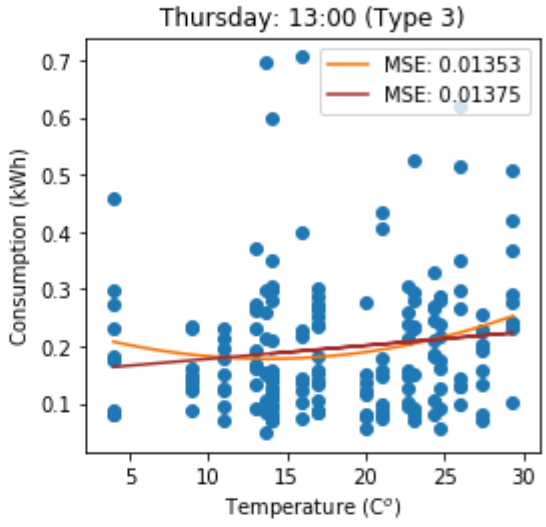

Thursday: 12:00 (Type 1)
MSE: 0.02301
MSE: 0.02306
Consumption (kWh)
Temperature (C°)

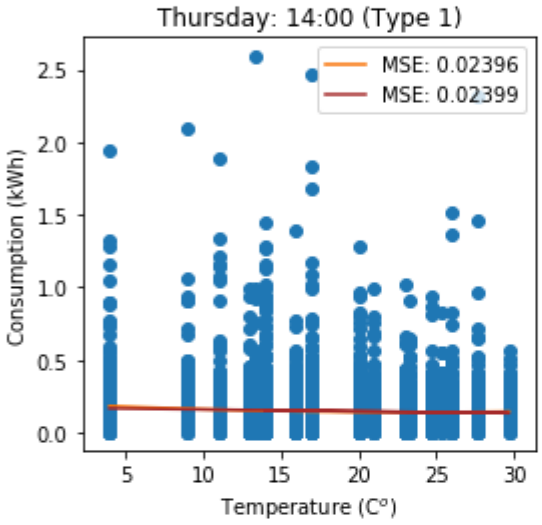

Thursday: 12:00 (Type 2)
MSE: 0.07272
MSE: 0.0751
Consumption (kWh)
Temperature (C°)

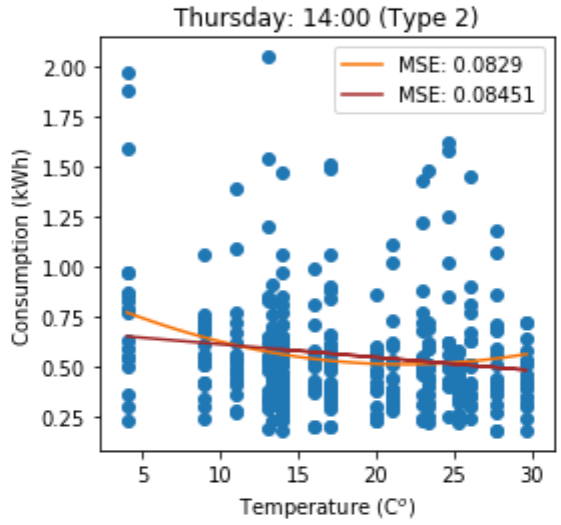

Thursday: 12:00 (Type 3)
MSE: 0.01224
MSE: 0.01245
Consumption (kWh)
Temperature (C°)

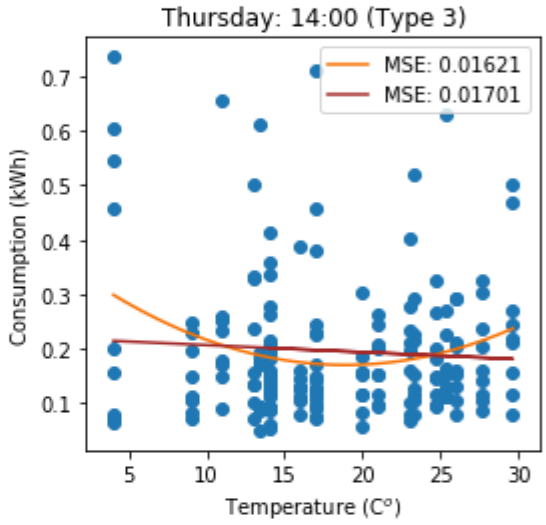

Thursday: 13:00 (Type 1)
MSE: 0.02165
MSE: 0.02169
Consumption (kWh)
Temperature (C°)

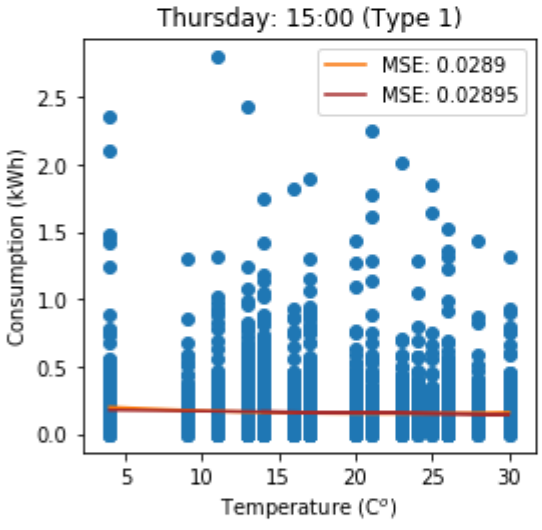

Thursday: 13:00 (Type 2)
MSE: 0.08395
MSE: 0.08572
Consumption (kWh)
Temperature (C°)

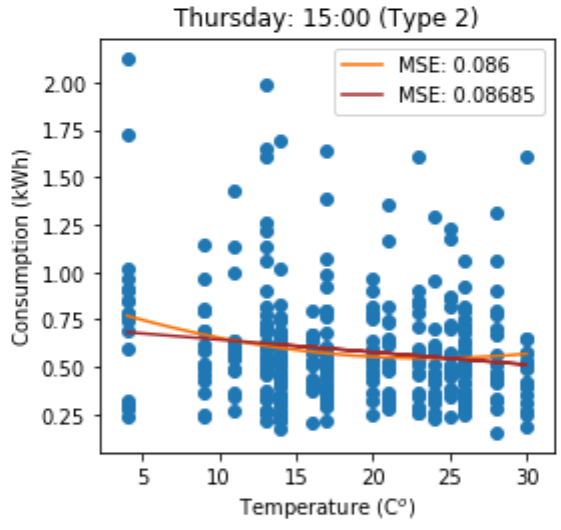

Thursday: 13:00 (Type 3)
MSE: 0.01353
MSE: 0.01375
Consumption (kWh)
Temperature (C°)

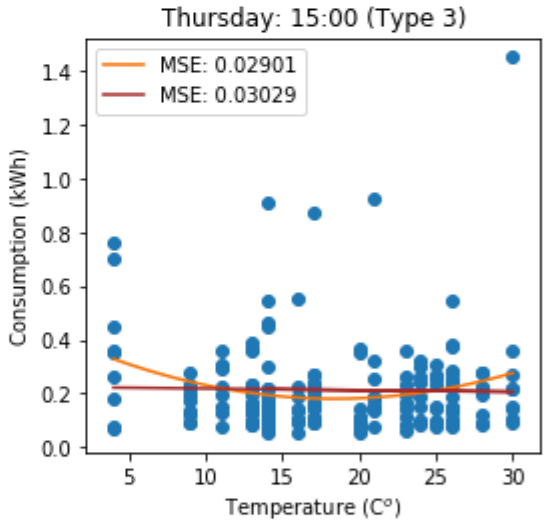

Thursday: 14:00 (Type 1)
MSE: 0.02396
MSE: 0.02399
Consumption (kWh)
Temperature (C°)

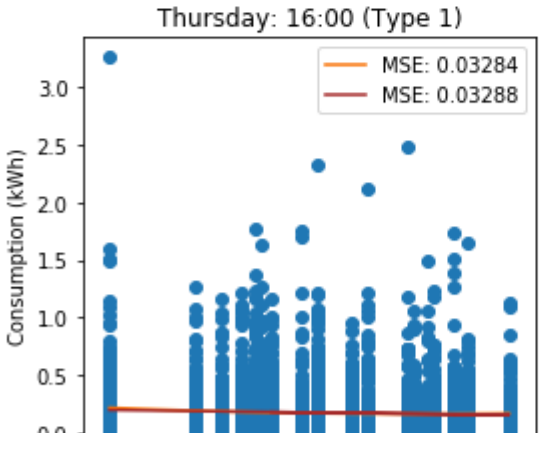

Thursday: 14:00 (Type 2)
MSE: 0.0829
MSE: 0.08451
Consumption (kWh)
Temperature (C°)

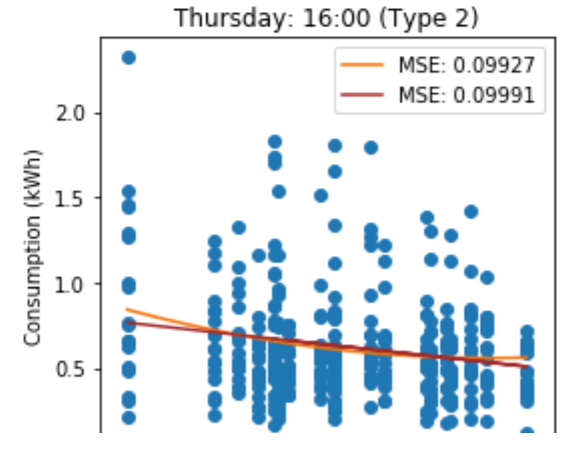

Thursday: 14:00 (Type 3)
MSE: 0.01621
MSE: 0.01701
Consumption (kWh)
Temperature (C°)

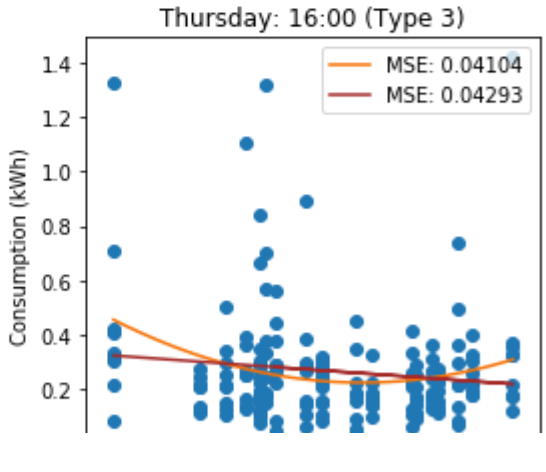

Thursday: 15:00 (Type 1)
MSE: 0.0289
MSE: 0.02895
Consumption (kWh)
Temperature (C°)

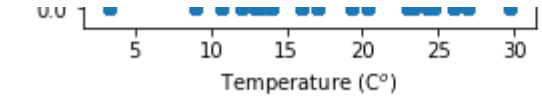

Thursday: 15:00 (Type 2)
MSE: 0.086
MSE: 0.08685
Consumption (kWh)
Temperature (C°)

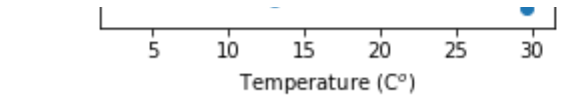

Thursday: 15:00 (Type 3)
MSE: 0.02901
MSE: 0.03029
Consumption (kWh)
Temperature (C°)

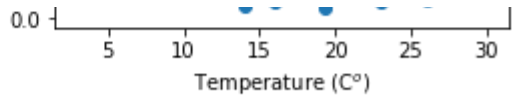

Thursday: 16:00 (Type 1)
MSE: 0.03284
MSE: 0.03288
Consumption (kWh)

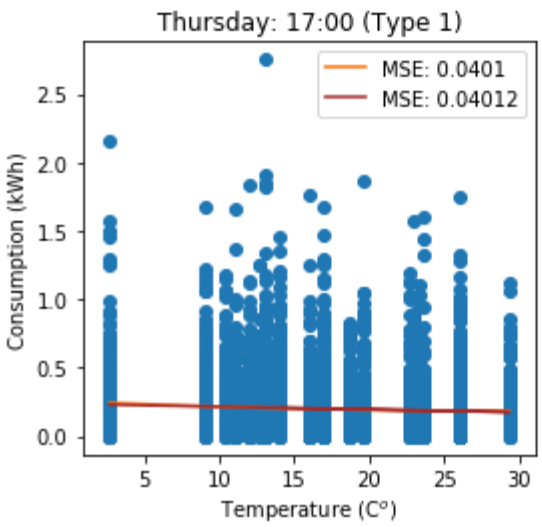

Thursday: 16:00 (Type 2)
MSE: 0.09927
MSE: 0.09991
Consumption (kWh)

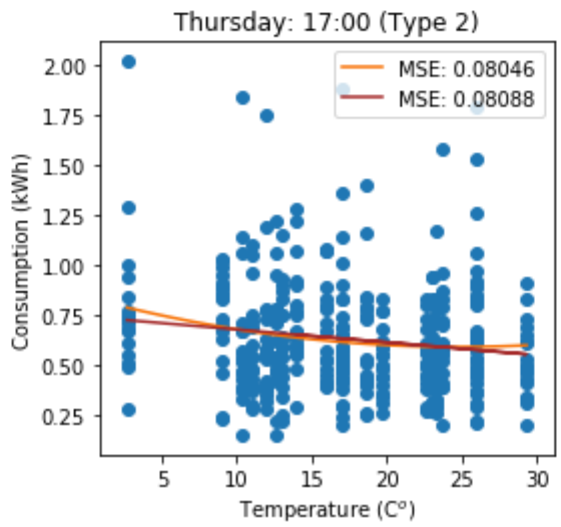

Thursday: 16:00 (Type 3)
MSE: 0.04104
MSE: 0.04293
Consumption (kWh)

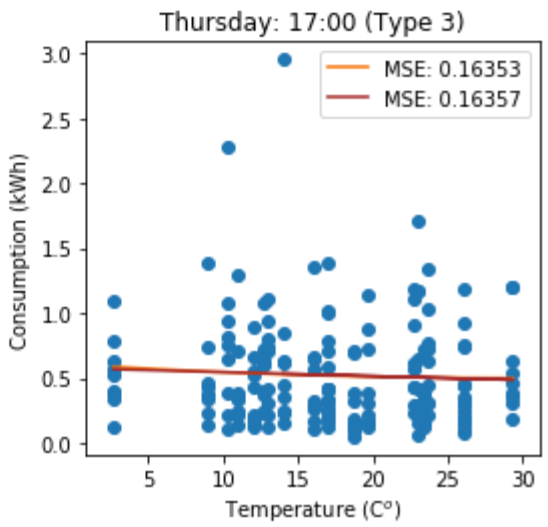

Temperature (C°)

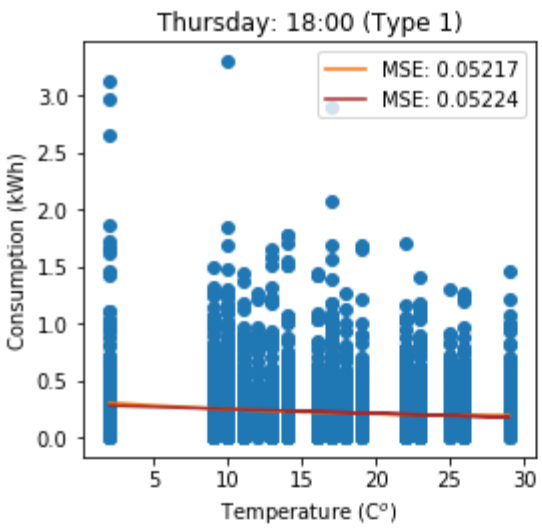

Temperature (C°)

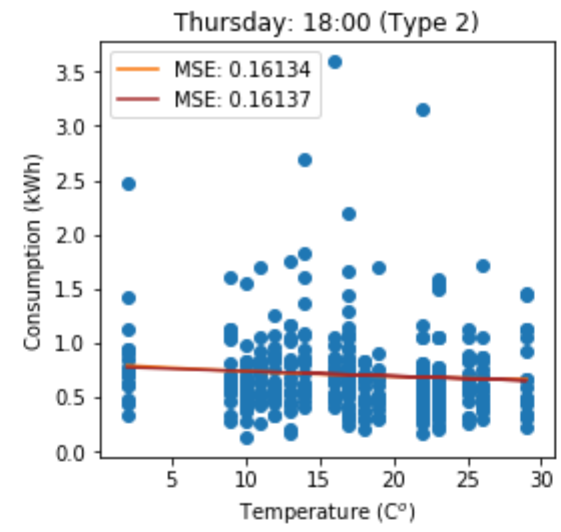

Temperature (C°)

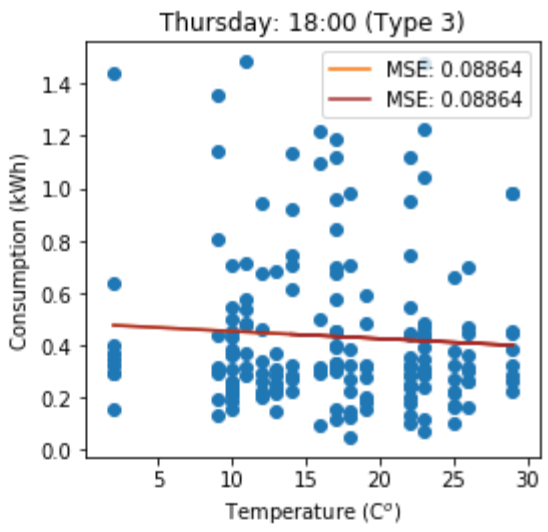

Thursday: 17:00 (Type 1)
MSE: 0.0401
MSE: 0.04012
Consumption (kWh)
Temperature (C°)

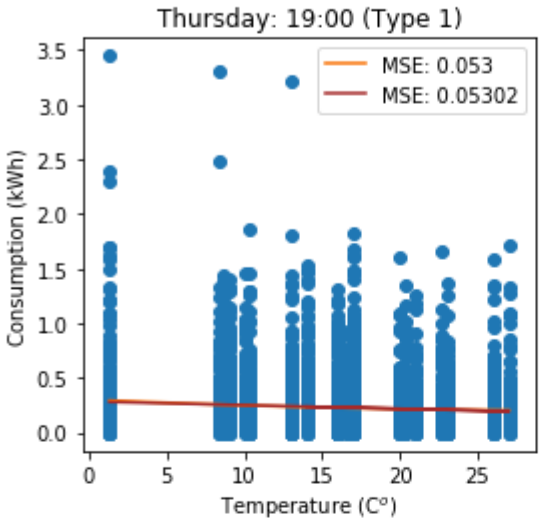

Thursday: 17:00 (Type 2)
MSE: 0.08046
MSE: 0.08088
Consumption (kWh)
Temperature (C°)

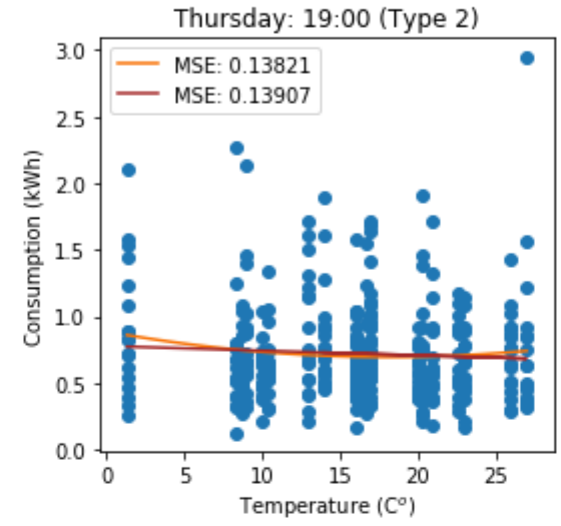

Thursday: 17:00 (Type 3)
MSE: 0.16353
MSE: 0.16357
Consumption (kWh)
Temperature (C°)

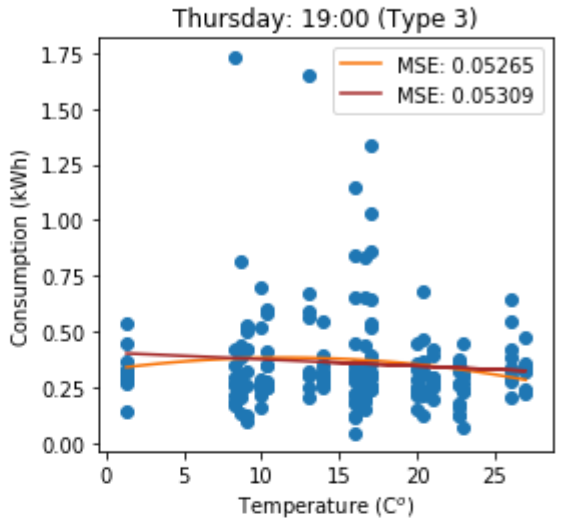

Thursday: 18:00 (Type 1)
MSE: 0.05217
MSE: 0.05224
Consumption (kWh)
Temperature (C°)

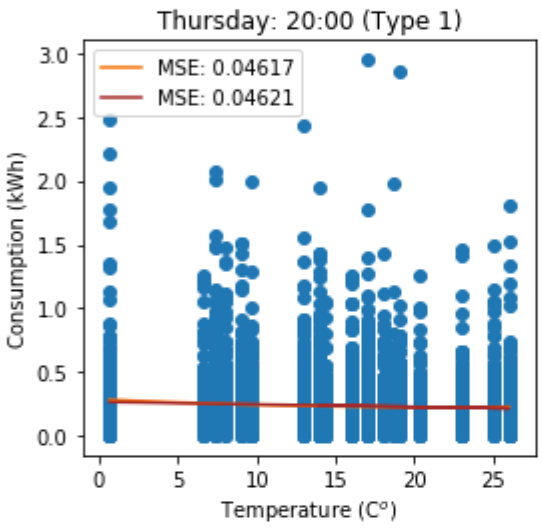

Thursday: 18:00 (Type 2)
MSE: 0.16134
MSE: 0.16137
Consumption (kWh)
Temperature (C°)

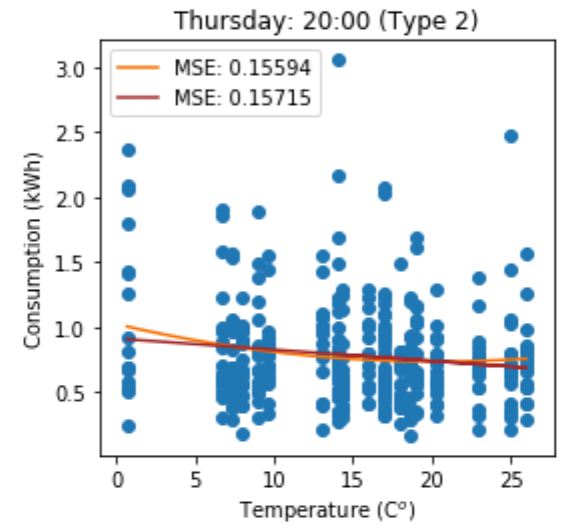

Thursday: 18:00 (Type 3)
MSE: 0.08864
MSE: 0.08864
Consumption (kWh)
Temperature (C°)

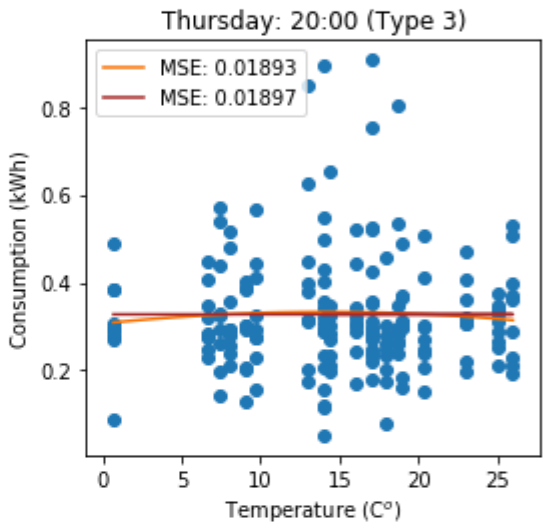

Thursday: 19:00 (Type 1)
MSE: 0.053
MSE: 0.05302
Consumption (kWh)
Temperature (C°)

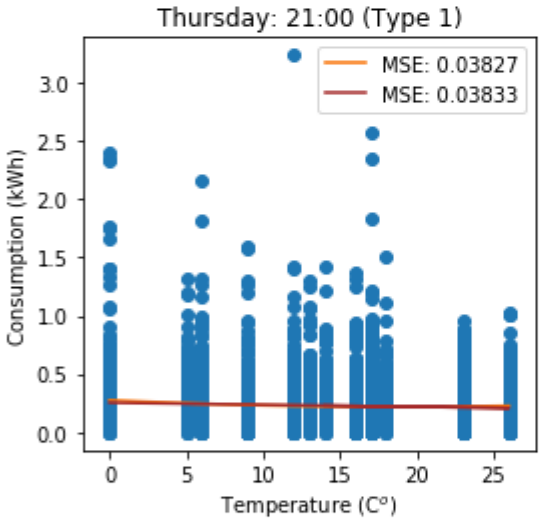

Thursday: 19:00 (Type 2)
MSE: 0.13821
MSE: 0.13907
Consumption (kWh)
Temperature (C°)

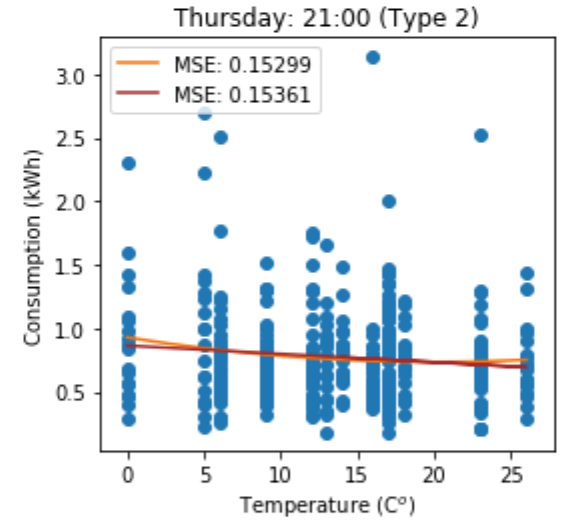

Thursday: 19:00 (Type 3)
MSE: 0.05265
MSE: 0.05309
Consumption (kWh)
Temperature (C°)

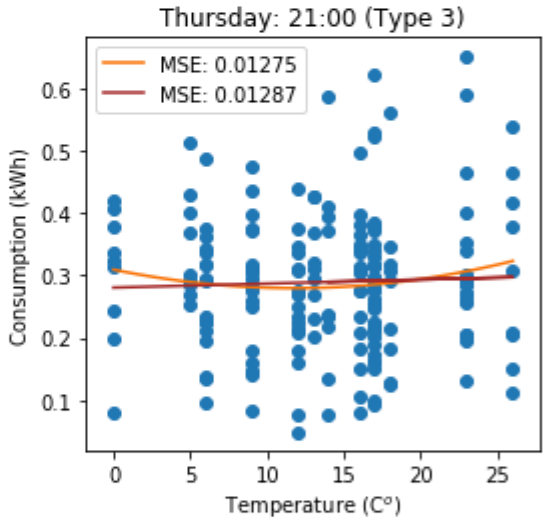

Thursday: 20:00 (Type 1)
MSE: 0.04617
MSE: 0.04621
Consumption (kWh)
Temperature (C°)

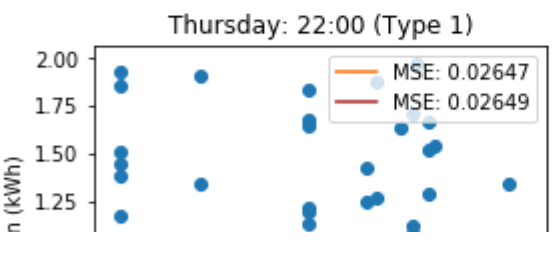

Thursday: 20:00 (Type 2)
MSE: 0.15594
MSE: 0.15715
Consumption (kWh)
Temperature (C°)

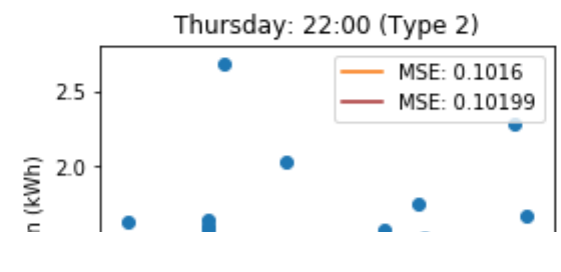

Thursday: 20:00 (Type 3)
MSE: 0.01893
MSE: 0.01897
Consumption (kWh)
Temperature (C°)

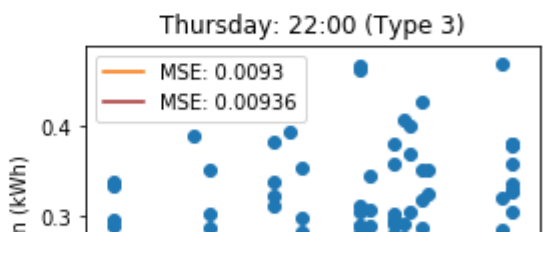

Thursday: 21:00 (Type 1)
MSE: 0.03827
MSE: 0.03833
Consumption (kWh)
Temperature (C°)

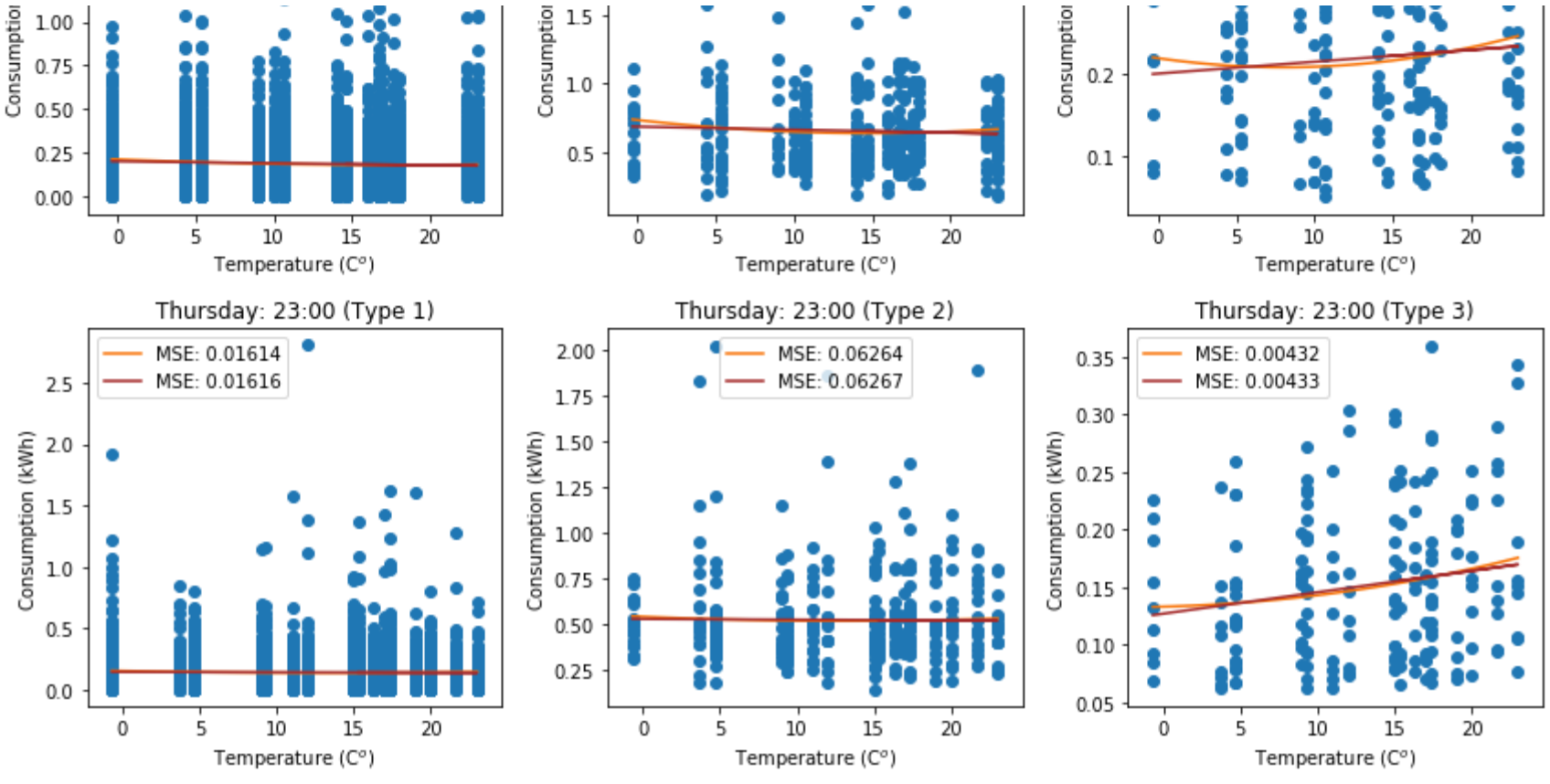

Thursday: 21:00 (Type 2)
MSE: 0.15299
MSE: 0.15361
Consumption (kWh)
Temperature (C°)


Thursday: 21:00 (Type 3)
MSE: 0.01275
MSE: 0.01287
Consumption (kWh)
Temperature (C°)

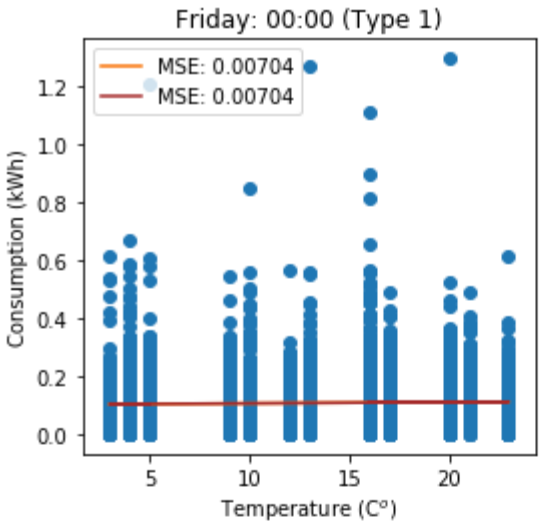

Thursday: 22:00 (Type 1)
MSE: 0.02647
MSE: 0.02649

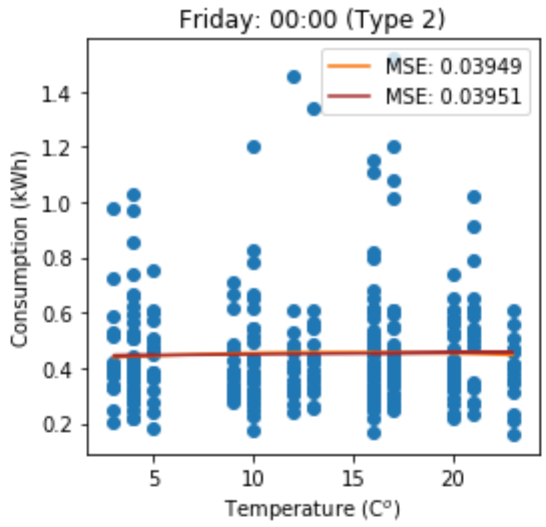

Thursday: 22:00 (Type 2)
MSE: 0.1016
MSE: 0.10199

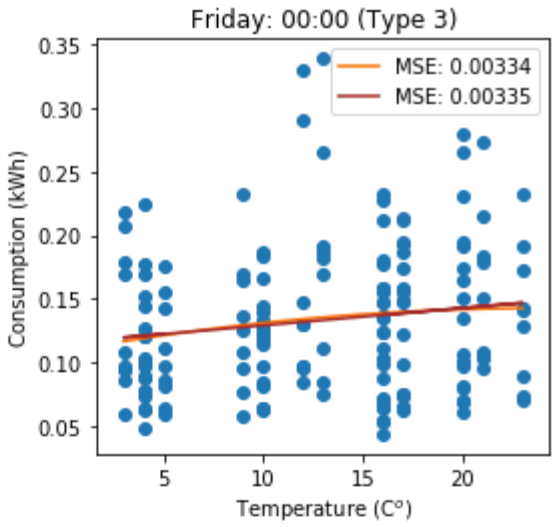

Thursday: 22:00 (Type 3)
MSE: 0.0093
MSE: 0.00936

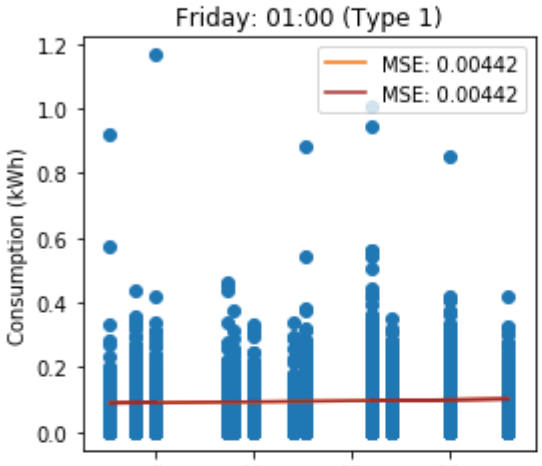

Thursday: 23:00 (Type 1)
MSE: 0.01614
MSE: 0.01616
Thursday: 23:00 (Type 2)
MSE: 0.06264
MSE: 0.06267
Thursday: 23:00 (Type 3)
MSE: 0.00432
MSE: 0.00433
Consumption (kWh)
Temperature (C°)

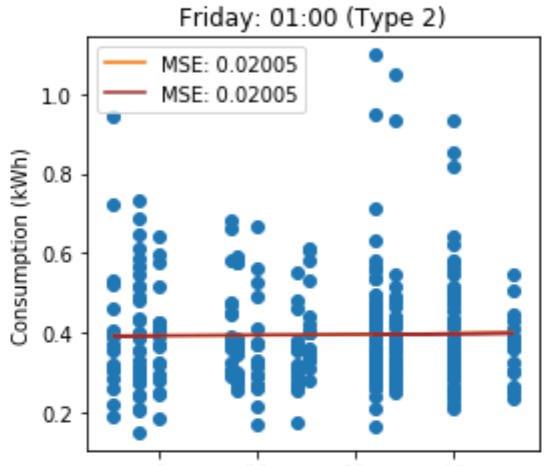
In [37]:
```
# Friday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 4 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

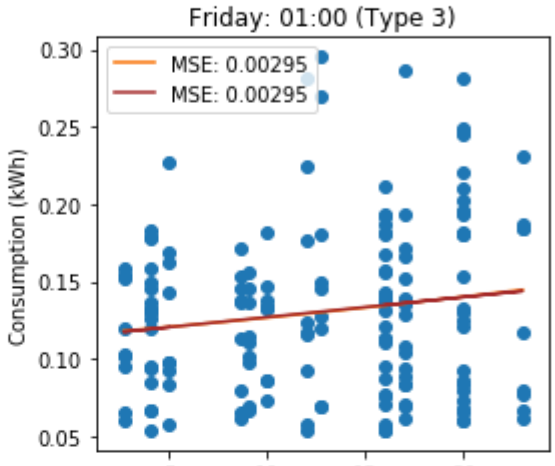

Friday: 00:00 (Type 1)
MSE: 0.00704
MSE: 0.00704
Consumption (kWh)
Temperature (C°)

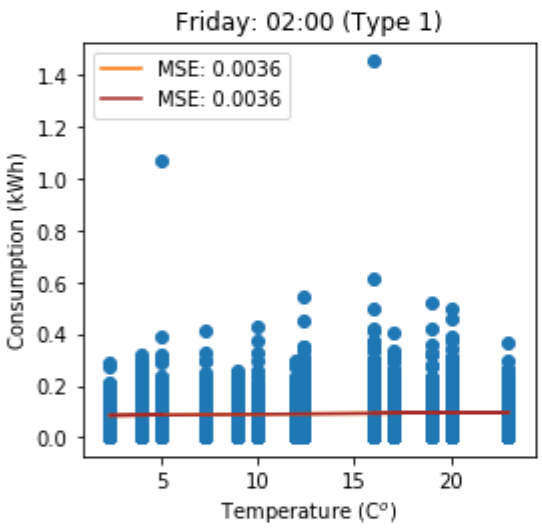

Friday: 00:00 (Type 2)
MSE: 0.03949
MSE: 0.03951
Consumption (kWh)
Temperature (C°)

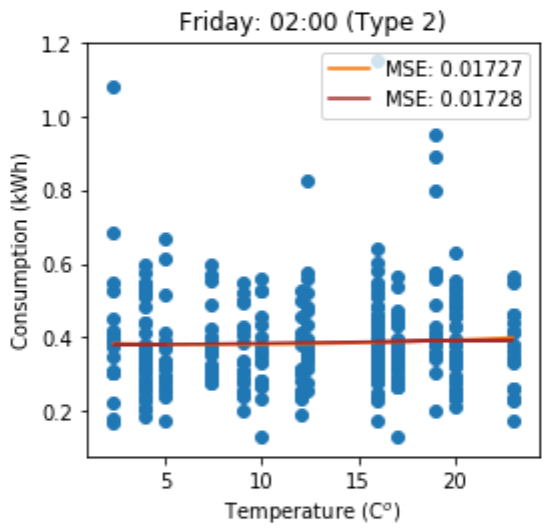

Friday: 00:00 (Type 3)
MSE: 0.00334
MSE: 0.00335
Consumption (kWh)
Temperature (C°)

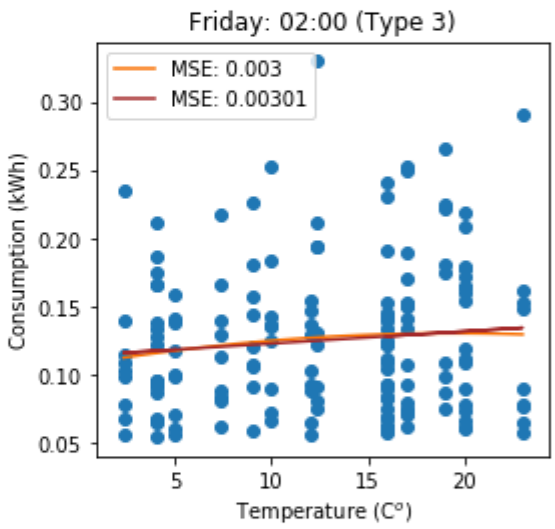

Friday: 01:00 (Type 1)
MSE: 0.00442
MSE: 0.00442
Consumption (kWh)
Temperature (C°)

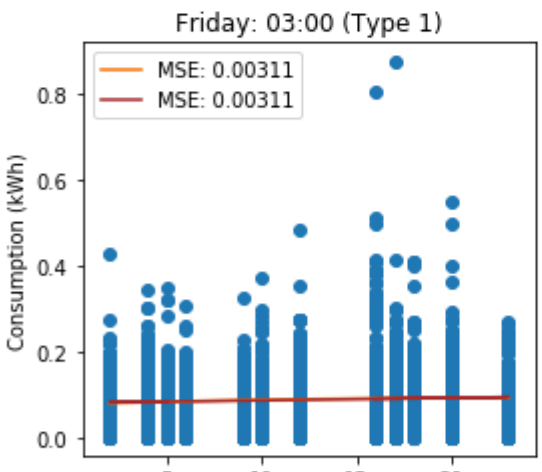

Friday: 01:00 (Type 2)
MSE: 0.02005
MSE: 0.02005
Consumption (kWh)
Temperature (C°)

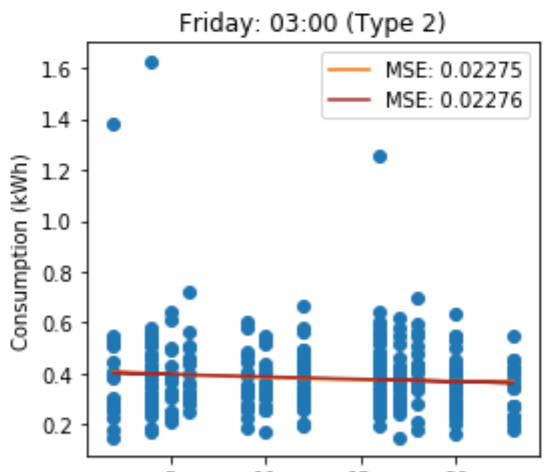

Friday: 01:00 (Type 3)
MSE: 0.00295
MSE: 0.00295
Consumption (kWh)
Temperature (C°)

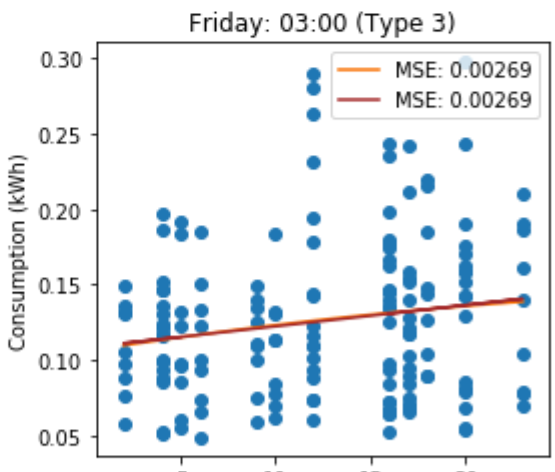

Friday: 02:00 (Type 1)
MSE: 0.0036
MSE: 0.0036
Consumption (kWh)
Temperature (C°)

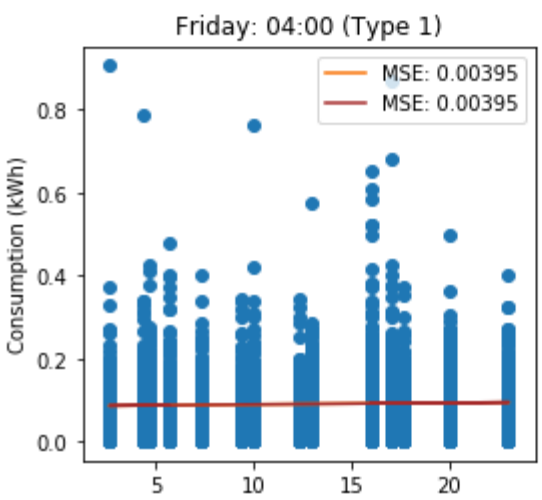

Friday: 02:00 (Type 2)
MSE: 0.01727
MSE: 0.01728
Consumption (kWh)
Temperature (C°)

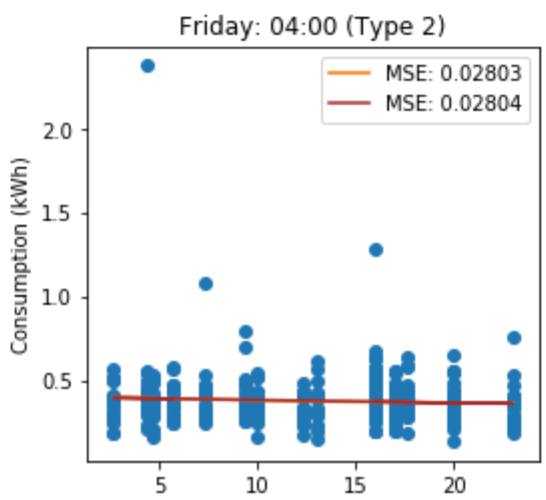

Friday: 02:00 (Type 3)
MSE: 0.003
MSE: 0.00301
Consumption (kWh)
Temperature (C°)

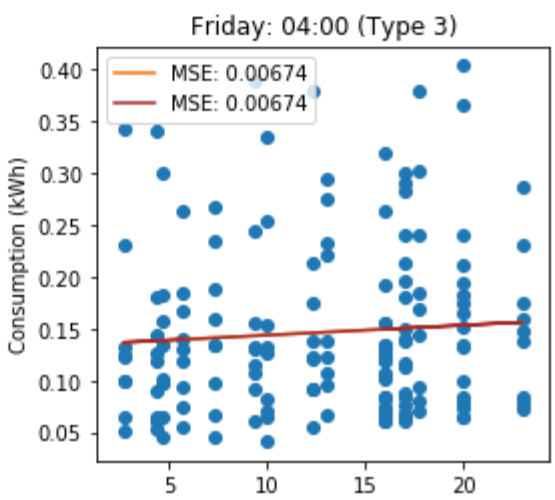

Friday: 03:00 (Type 1)
MSE: 0.00311
MSE: 0.00311
Consumption (kWh)
Temperature (C°)

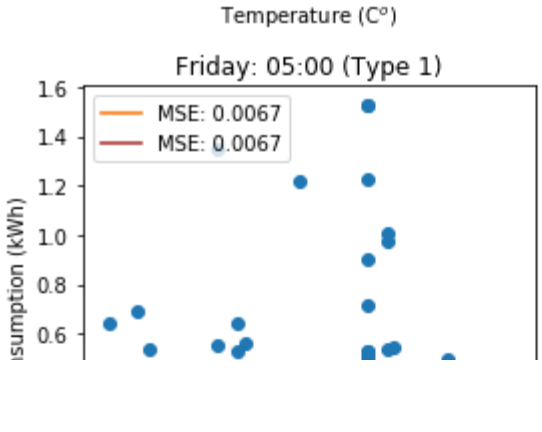

Friday: 03:00 (Type 2)
MSE: 0.02275
MSE: 0.02276
Consumption (kWh)
Temperature (C°)

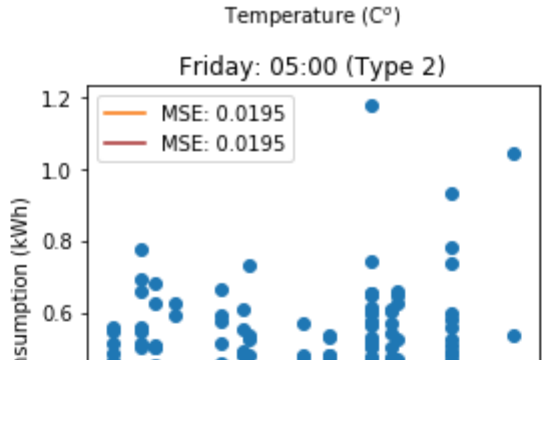

Friday: 03:00 (Type 3)
MSE: 0.00269
MSE: 0.00269
Consumption (kWh)
Temperature (C°)

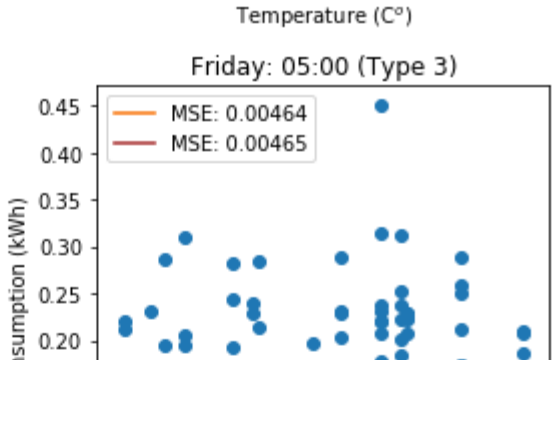

Friday: 04:00 (Type 1)
MSE: 0.00395
MSE: 0.00395
Consumption (kWh)
Temperature (C°)

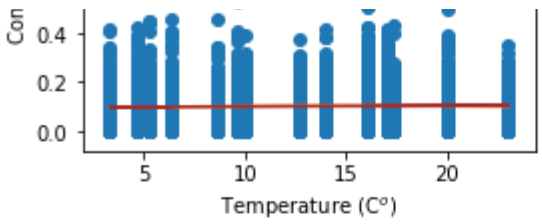

Friday: 04:00 (Type 2)
MSE: 0.02803
MSE: 0.02804
Consumption (kWh)
Temperature (C°)

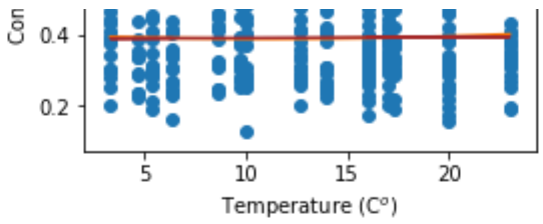

Friday: 04:00 (Type 3)
MSE: 0.00674
MSE: 0.00674
Consumption (kWh)
Temperature (C°)

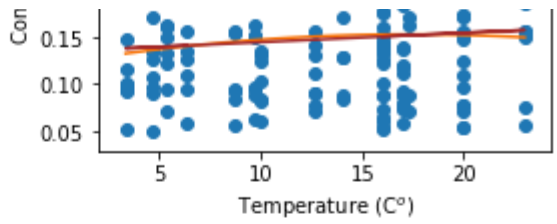

Friday: 05:00 (Type 1)
MSE: 0.0067
MSE: 0.0067

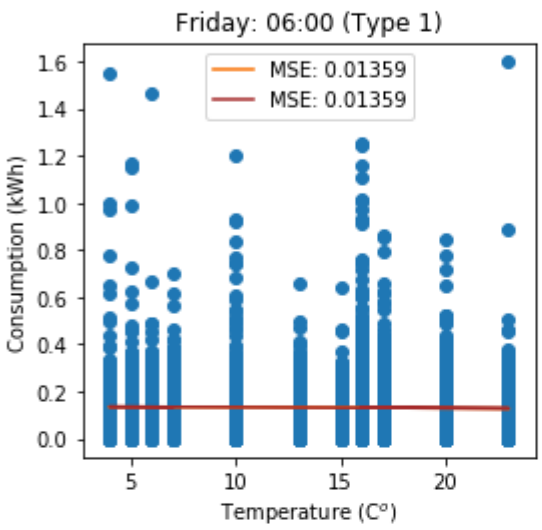

Friday: 05:00 (Type 2)
MSE: 0.0195
MSE: 0.0195

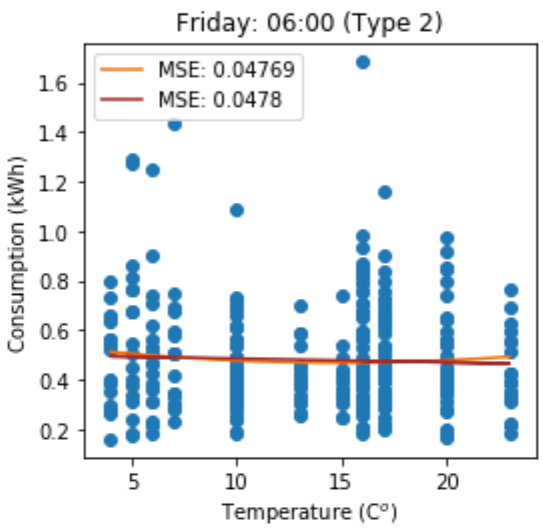

Friday: 05:00 (Type 3)
MSE: 0.00464
MSE: 0.00465

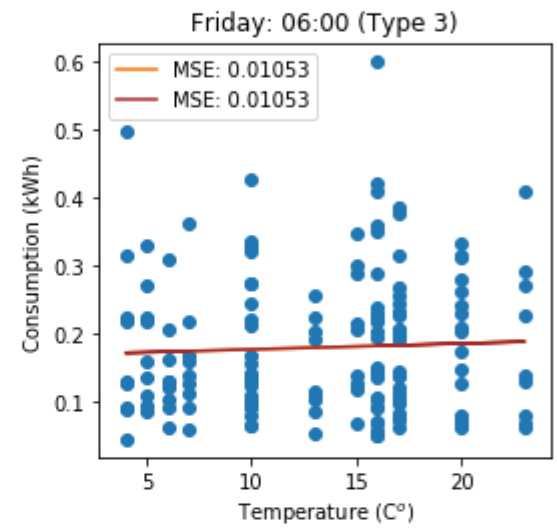

Con
0.4
0.2
0.0
5
10
15
20
Temperature (C°)

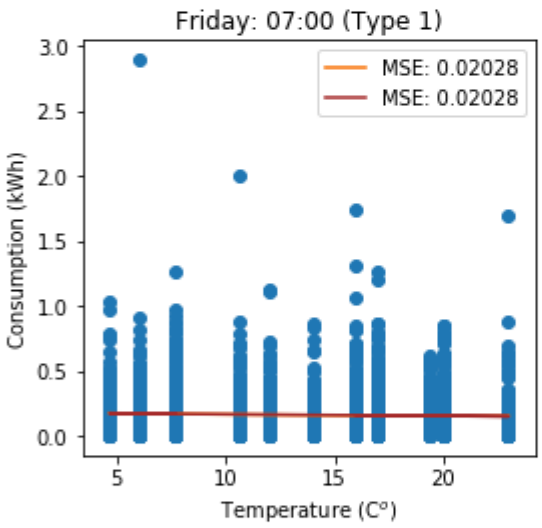

Con
0.4
0.2
5
10
15
20
Temperature (C°)

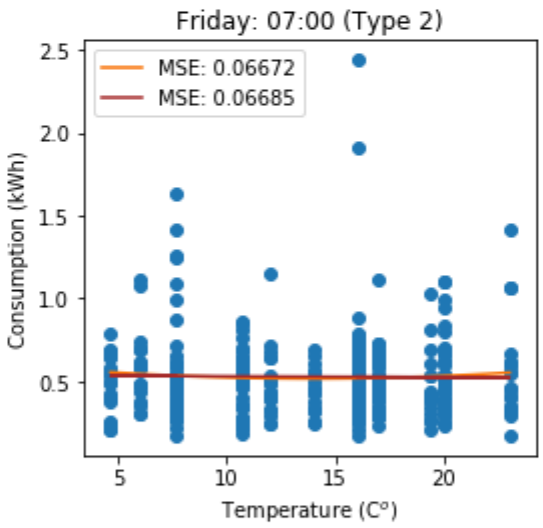

Con
0.15
0.10
0.05
5
10
15
20
Temperature (C°)

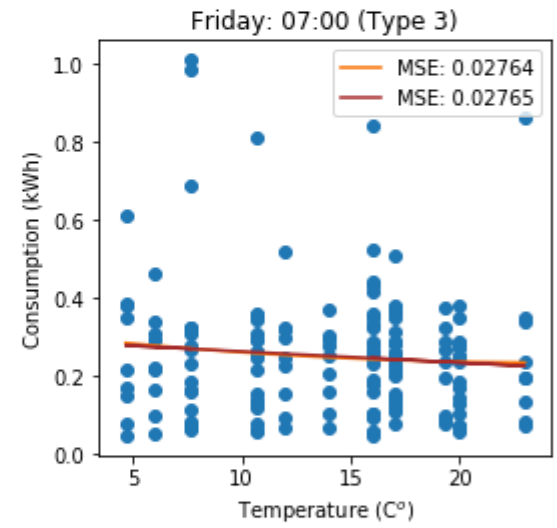

Friday: 06:00 (Type 1)
MSE: 0.01359
MSE: 0.01359
Consumption (kWh)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
Temperature (C°)

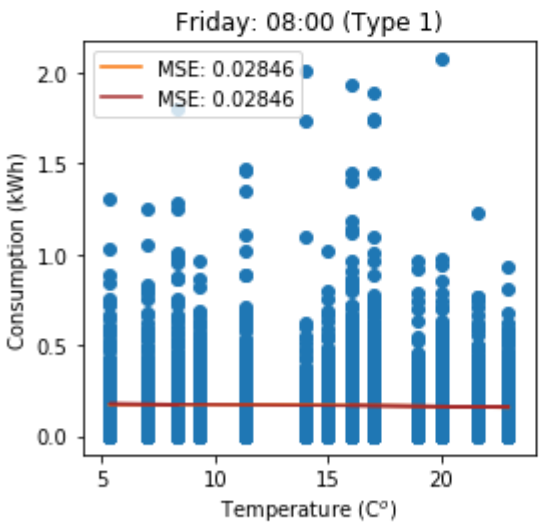

Friday: 06:00 (Type 2)
MSE: 0.04769
MSE: 0.0478
Consumption (kWh)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
5
10
15
20
Temperature (C°)

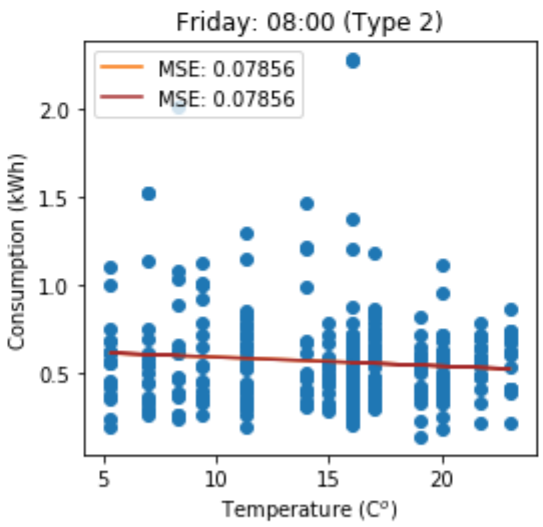

Friday: 06:00 (Type 3)
MSE: 0.01053
MSE: 0.01053
Consumption (kWh)
0.6
0.5
0.4
0.3
0.2
0.1
5
10
15
20
Temperature (C°)

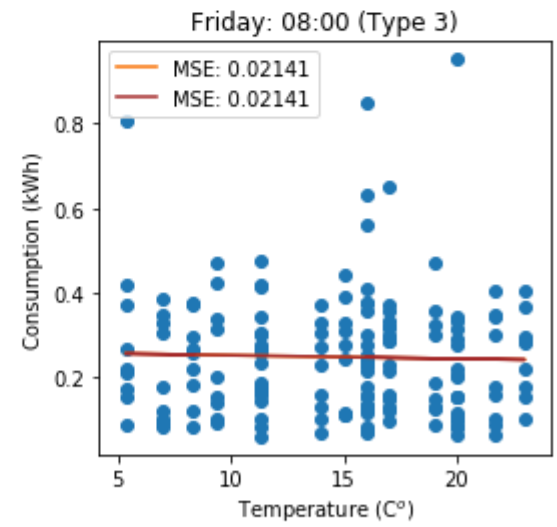

Friday: 07:00 (Type 1)
MSE: 0.02028
MSE: 0.02028
Consumption (kWh)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
Temperature (C°)

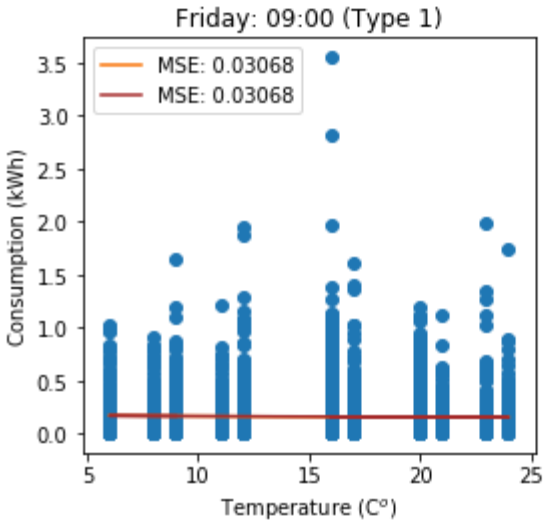

Friday: 07:00 (Type 2)
MSE: 0.06672
MSE: 0.06685
Consumption (kWh)
2.5
2.0
1.5
1.0
0.5
5
10
15
20
Temperature (C°)

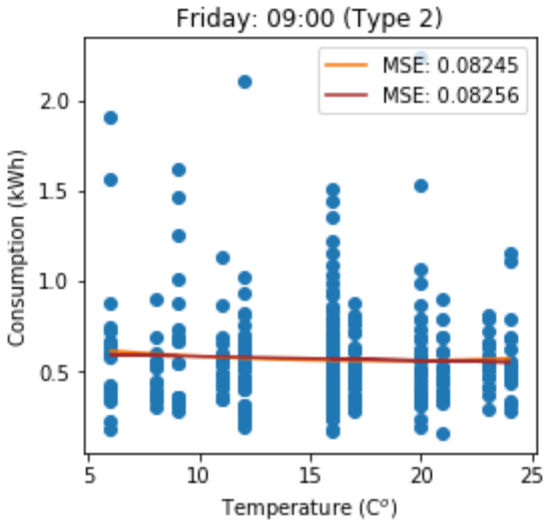

Friday: 07:00 (Type 3)
MSE: 0.02764
MSE: 0.02765
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
Temperature (C°)

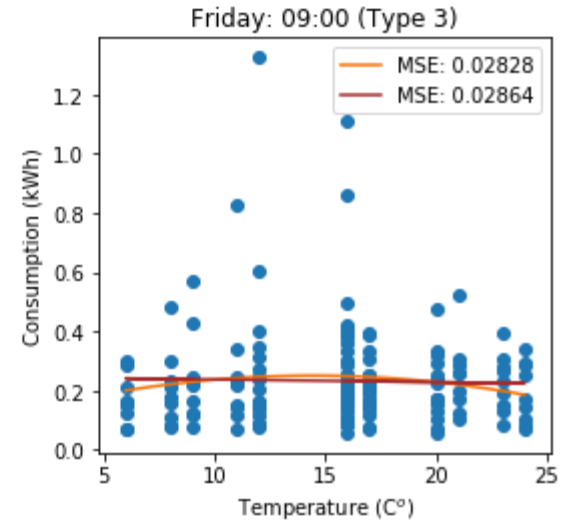

Friday: 08:00 (Type 1)
MSE: 0.02846
MSE: 0.02846
Consumption (kWh)
2.0
1.5
1.0
0.5
0.0
5
10
15
20
Temperature (C°)

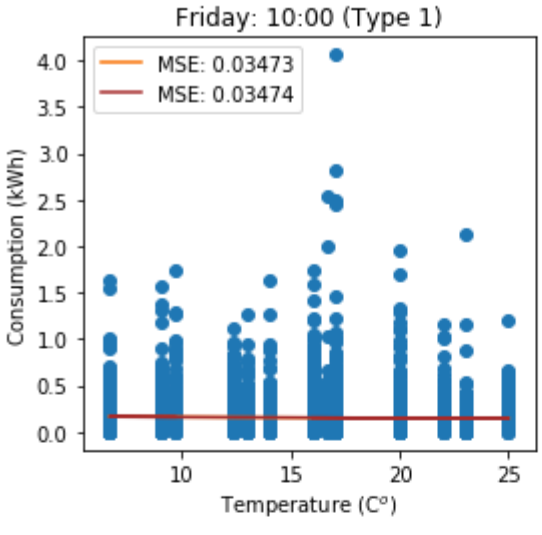

Friday: 08:00 (Type 2)
MSE: 0.07856
MSE: 0.07856
Consumption (kWh)
2.0
1.5
1.0
0.5
5
10
15
20
Temperature (C°)

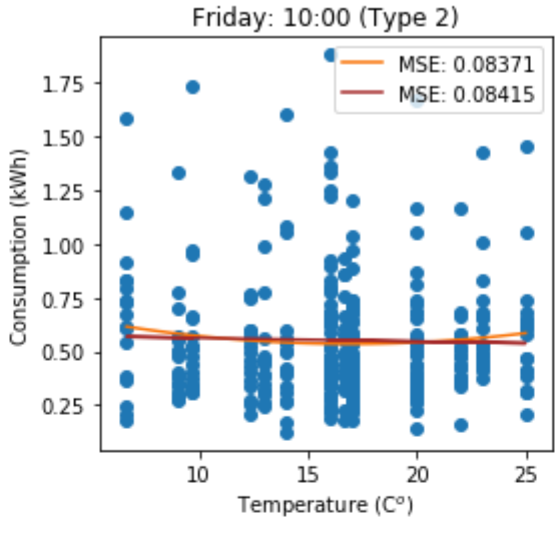

Friday: 08:00 (Type 3)
MSE: 0.02141
MSE: 0.02141
Consumption (kWh)
0.8
0.6
0.4
0.2
5
10
15
20
Temperature (C°)

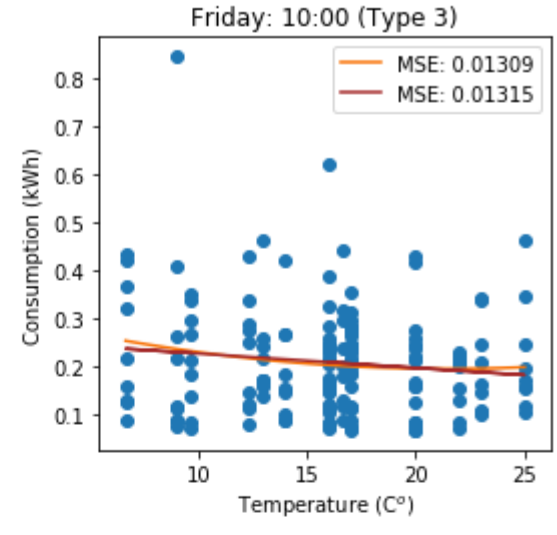

Friday: 09:00 (Type 1)
MSE: 0.03068
MSE: 0.03068
Consumption (kWh)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
5
10
15
20
25
Temperature (C°)


Friday: 09:00 (Type 2)
MSE: 0.08245
MSE: 0.08256
Consumption (kWh)
2.0
1.5
1.0
0.5
5
10
15
20
25
Temperature (C°)


Friday: 09:00 (Type 3)
MSE: 0.02828
MSE: 0.02864
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
5
10
15
20
25
Temperature (C°)


Friday: 10:00 (Type 1)
MSE: 0.03473
MSE: 0.03474
Consumption (kWh)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10
15
20
25
Temperature (C°)

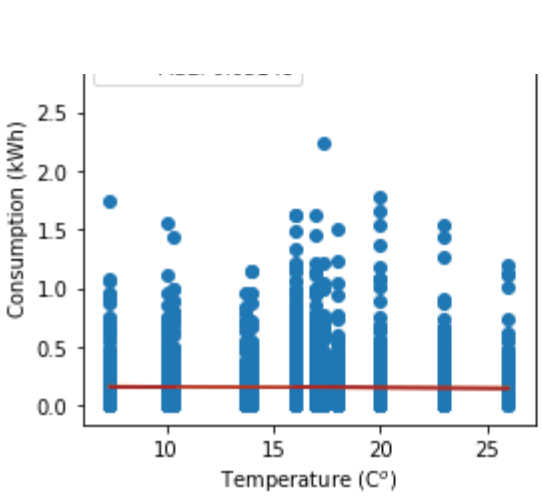

Friday: 10:00 (Type 2)
MSE: 0.08371
MSE: 0.08415
Consumption (kWh)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
10
15
20
25
Temperature (C°)

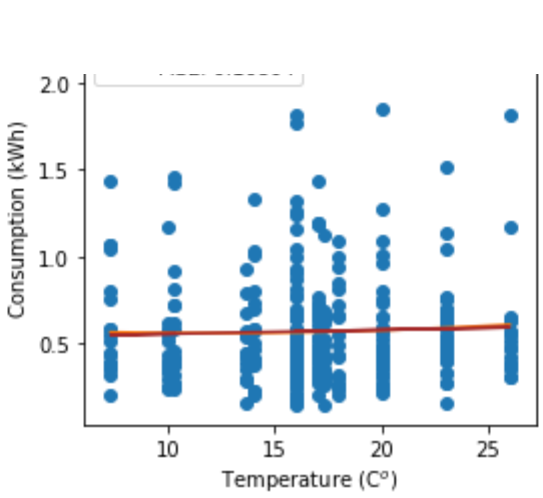

Friday: 10:00 (Type 3)
MSE: 0.01309
MSE: 0.01315
Consumption (kWh)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
10
15
20
25
Temperature (C°)

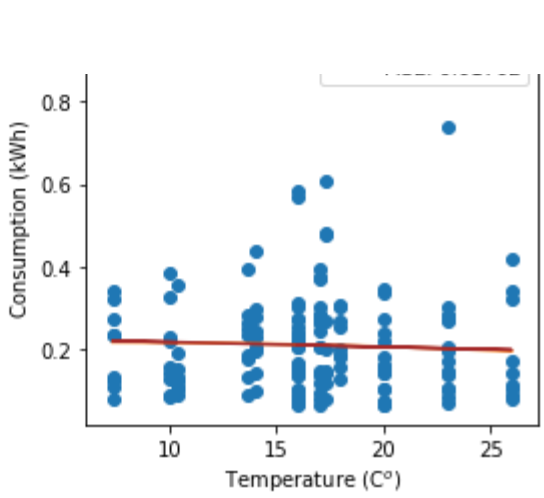
Friday: 11:00 (Type 1)
MSE: 0.03147
MSE: 0.03148
3.0

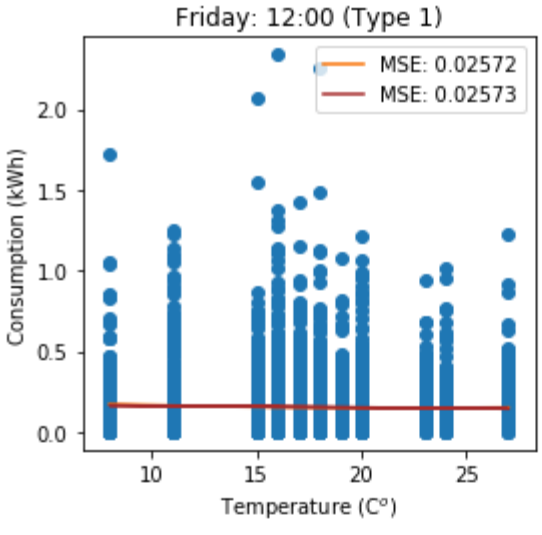
Friday: 11:00 (Type 2)
MSE: 0.108
MSE: 0.10804

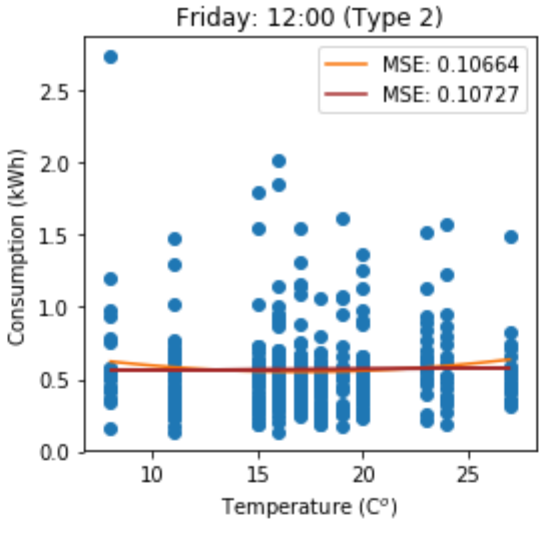
Friday: 11:00 (Type 3)
MSE: 0.01701
MSE: 0.01702
1.0

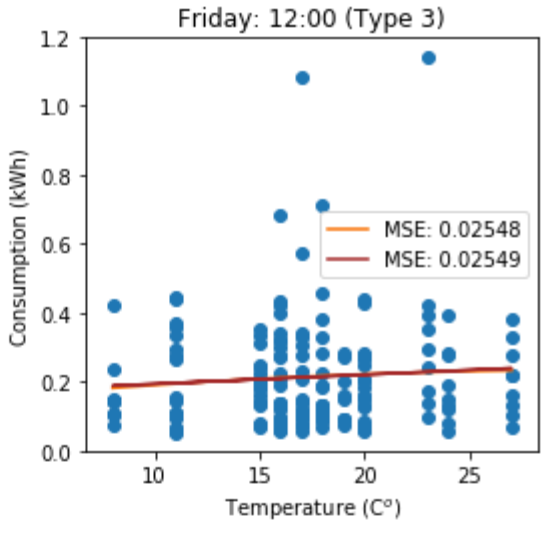

Consumption (kWh)
Temperature (C°)

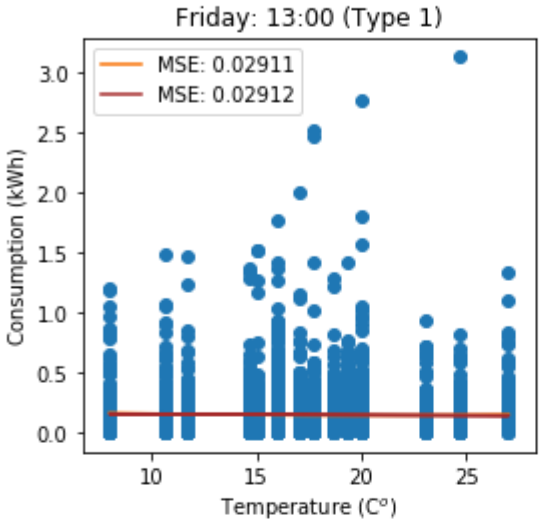

Consumption (kWh)
Temperature (C°)

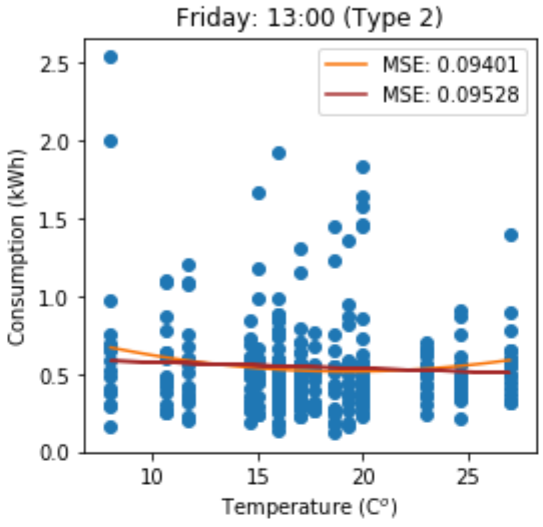

Consumption (kWh)
Temperature (C°)

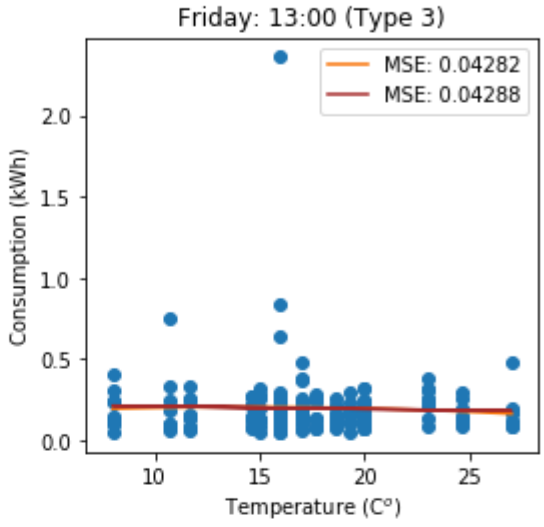

Friday: 12:00 (Type 1)
MSE: 0.02572
MSE: 0.02573
Consumption (kWh)
Temperature (C°)

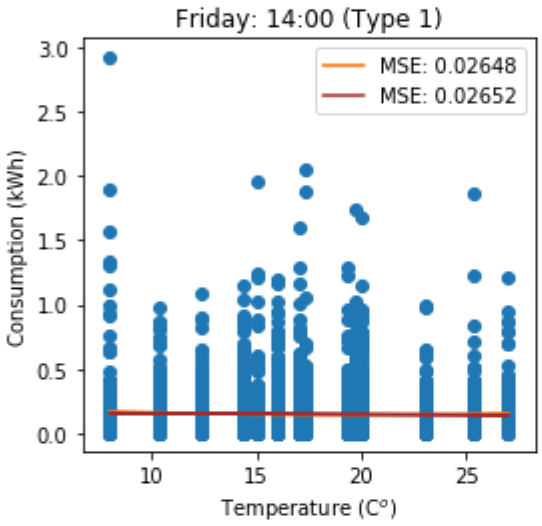

Friday: 12:00 (Type 2)
MSE: 0.10664
MSE: 0.10727
Consumption (kWh)
Temperature (C°)

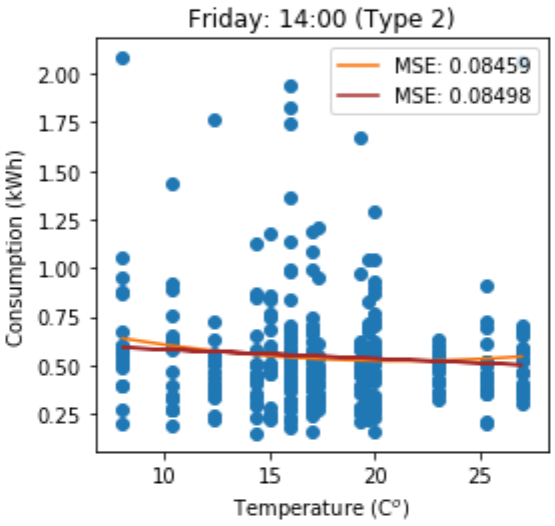

Friday: 12:00 (Type 3)
MSE: 0.02548
MSE: 0.02549
Consumption (kWh)
Temperature (C°)

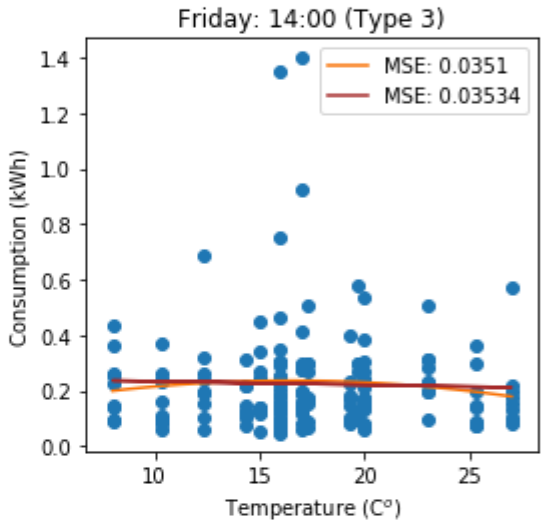

Friday: 13:00 (Type 1)
MSE: 0.02911
MSE: 0.02912
Consumption (kWh)
Temperature (C°)

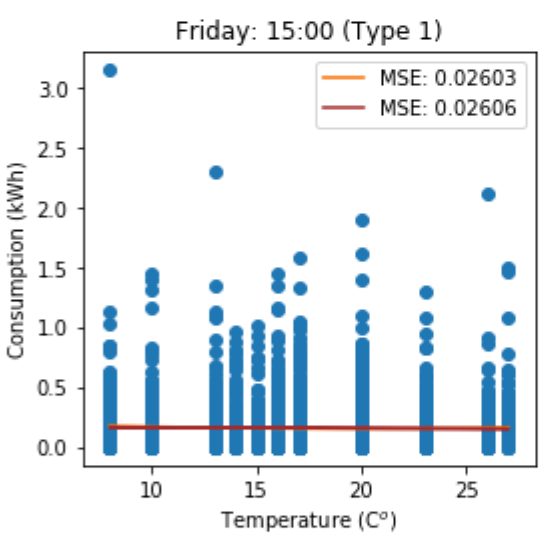

Friday: 13:00 (Type 2)
MSE: 0.09401
MSE: 0.09528
Consumption (kWh)
Temperature (C°)

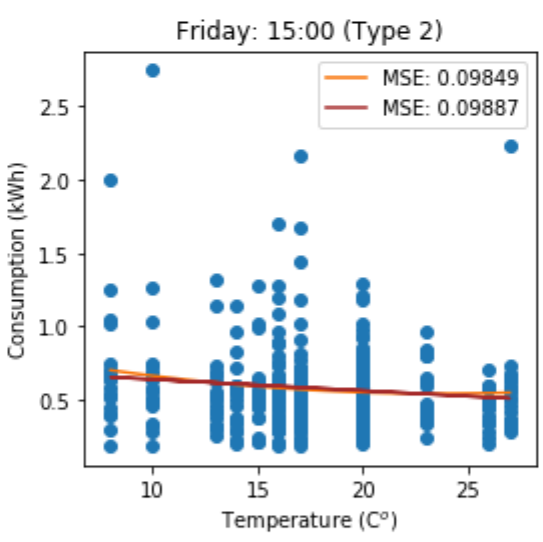

Friday: 13:00 (Type 3)
MSE: 0.04282
MSE: 0.04288
Consumption (kWh)
Temperature (C°)

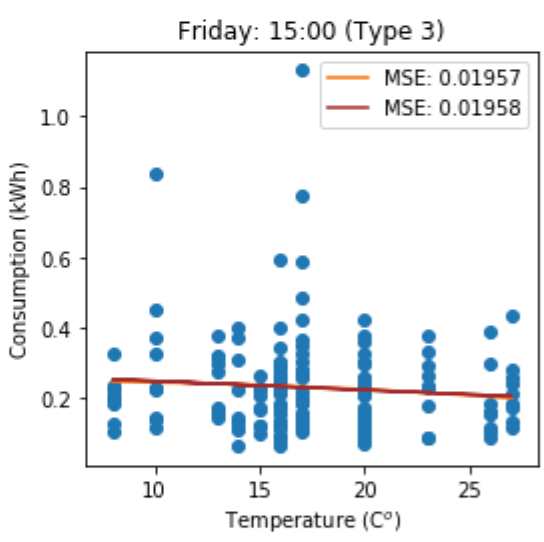

Friday: 14:00 (Type 1)
MSE: 0.02648
MSE: 0.02652
Consumption (kWh)
Temperature (C°)

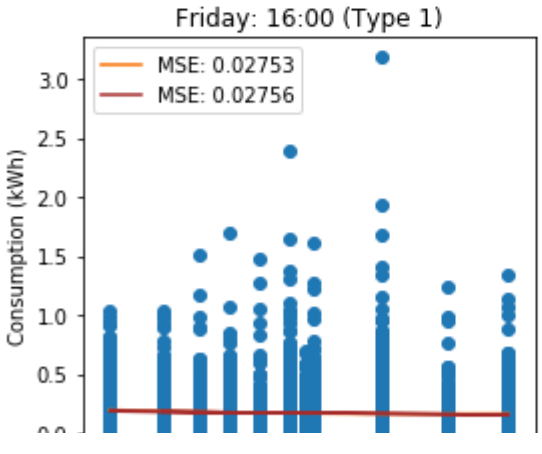

Friday: 14:00 (Type 2)
MSE: 0.08459
MSE: 0.08498
Consumption (kWh)
Temperature (C°)

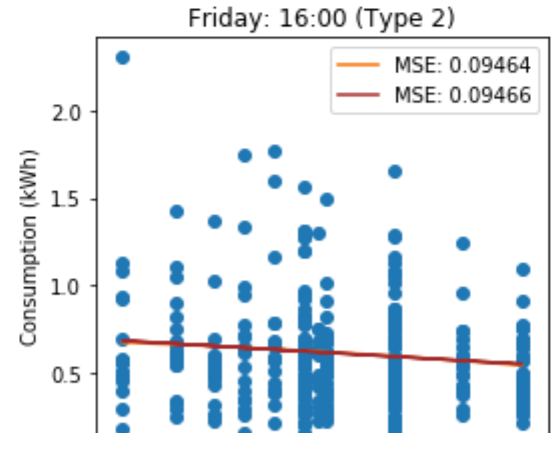

Friday: 14:00 (Type 3)
MSE: 0.0351
MSE: 0.03534
Consumption (kWh)
Temperature (C°)

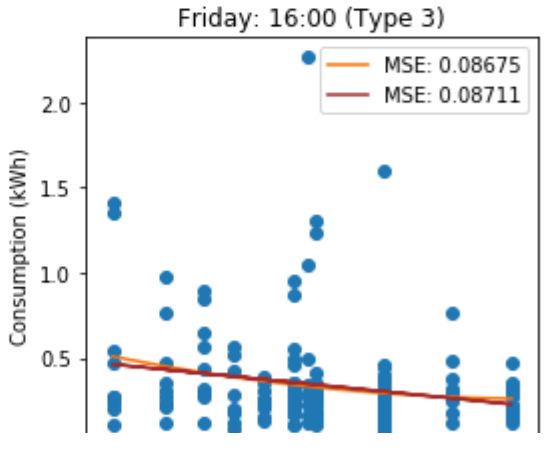

Friday: 15:00 (Type 1)
MSE: 0.02603
MSE: 0.02606
Consumption (kWh)
Temperature (C°)

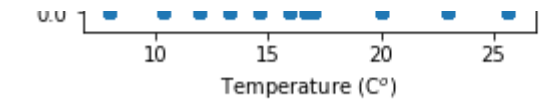

Friday: 15:00 (Type 2)
MSE: 0.09849
MSE: 0.09887
Consumption (kWh)
Temperature (C°)

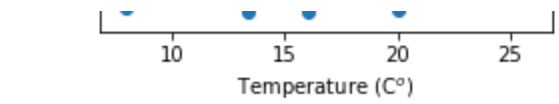

Friday: 15:00 (Type 3)
MSE: 0.01957
MSE: 0.01958
Consumption (kWh)
Temperature (C°)

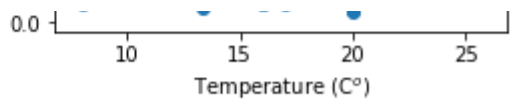

Friday: 16:00 (Type 1)
MSE: 0.02753
MSE: 0.02756
Consumption (kWh)

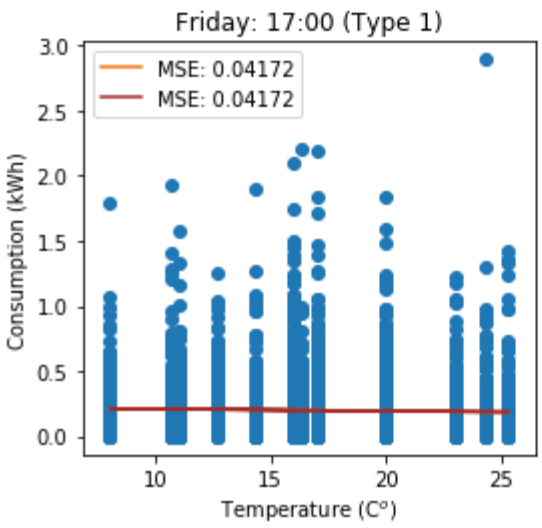

Friday: 16:00 (Type 2)
MSE: 0.09464
MSE: 0.09466
Consumption (kWh)

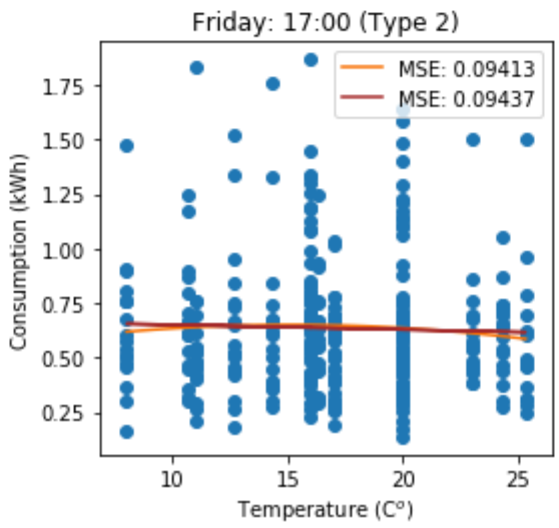

Friday: 16:00 (Type 3)
MSE: 0.08675
MSE: 0.08711
Consumption (kWh)

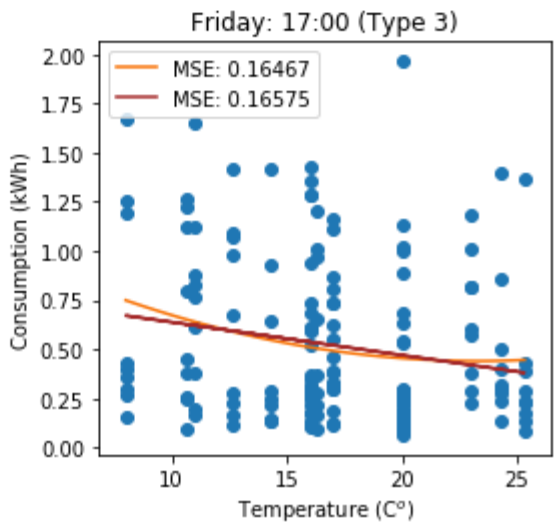

Temperature (C°)

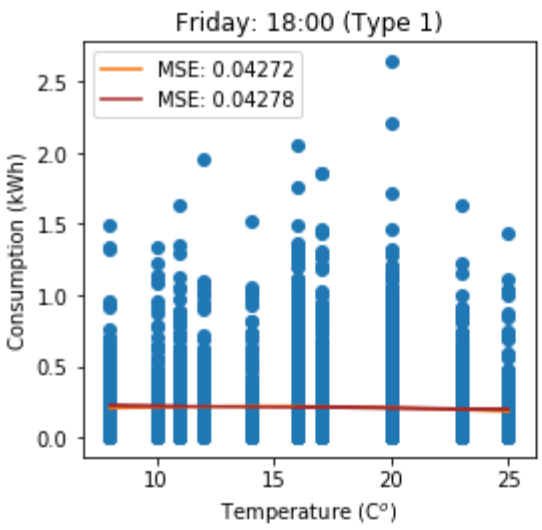

Temperature (C°)

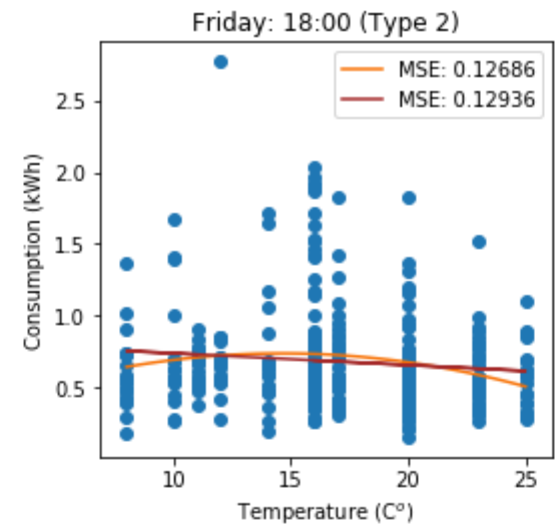

Temperature (C°)

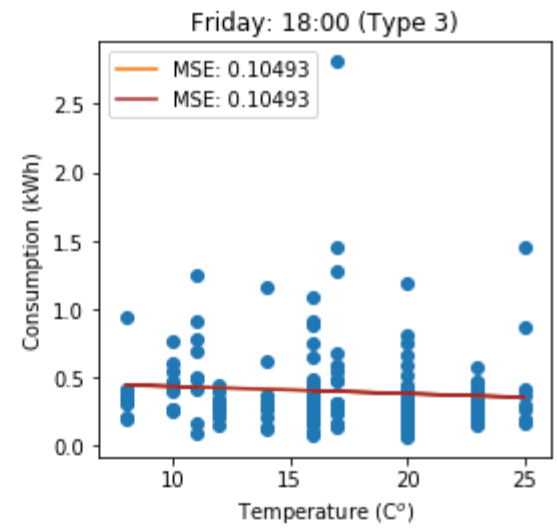

Friday: 17:00 (Type 1)
MSE: 0.04172
MSE: 0.04172
Consumption (kWh)
Temperature (C°)

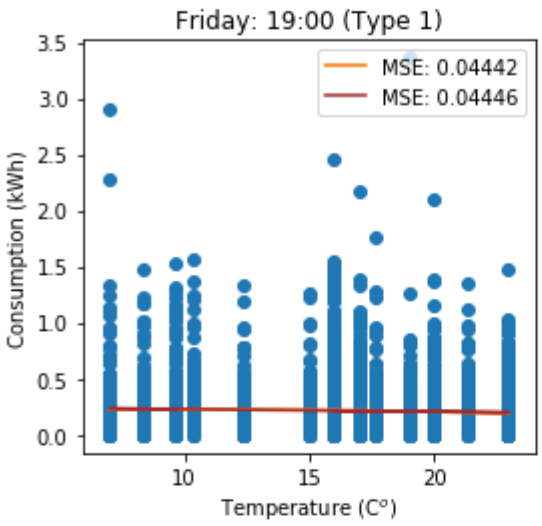

Friday: 17:00 (Type 2)
MSE: 0.09413
MSE: 0.09437
Consumption (kWh)
Temperature (C°)

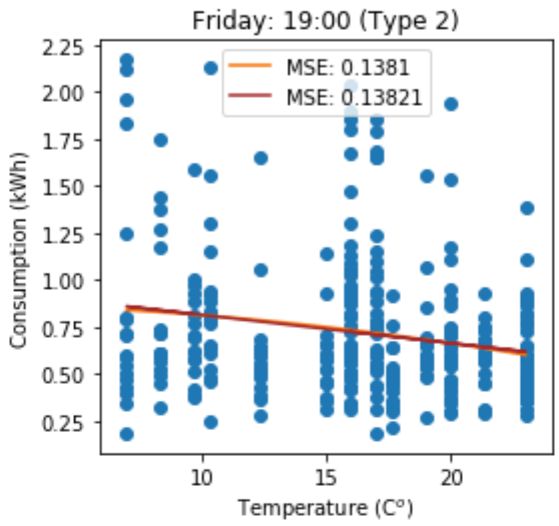

Friday: 17:00 (Type 3)
MSE: 0.16467
MSE: 0.16575
Consumption (kWh)
Temperature (C°)

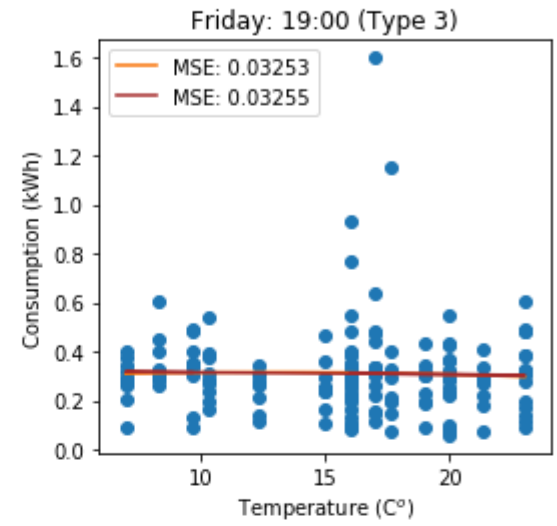

Friday: 18:00 (Type 1)
MSE: 0.04272
MSE: 0.04278
Consumption (kWh)
Temperature (C°)

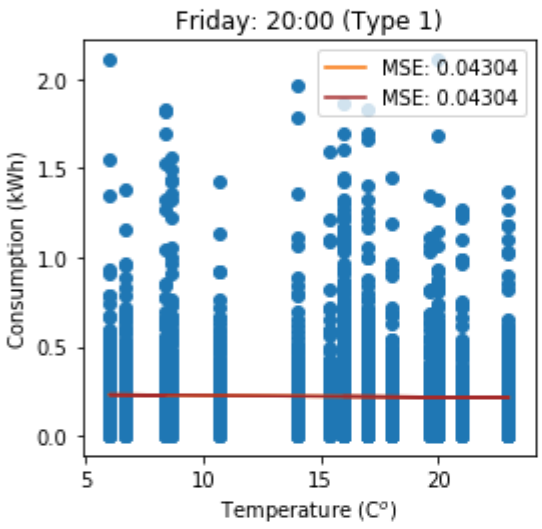

Friday: 18:00 (Type 2)
MSE: 0.12686
MSE: 0.12936
Consumption (kWh)
Temperature (C°)

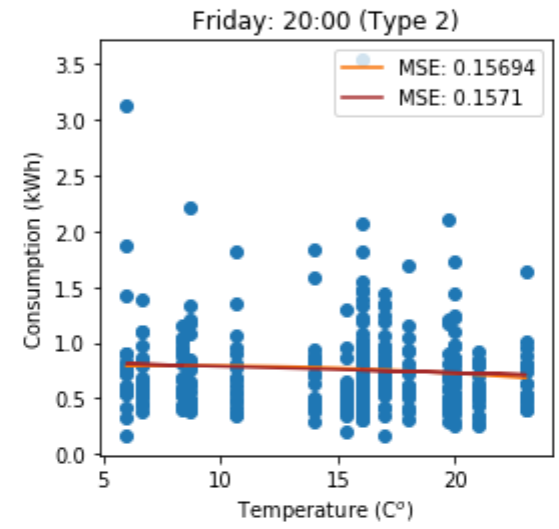

Friday: 18:00 (Type 3)
MSE: 0.10493
MSE: 0.10493
Consumption (kWh)
Temperature (C°)

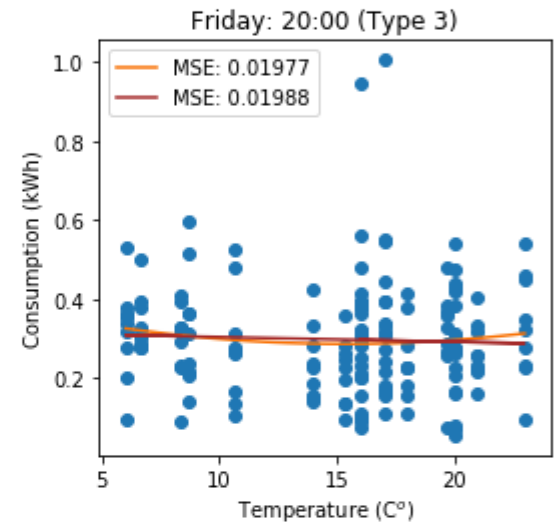

Friday: 19:00 (Type 1)
MSE: 0.04442
MSE: 0.04446
Consumption (kWh)
Temperature (C°)

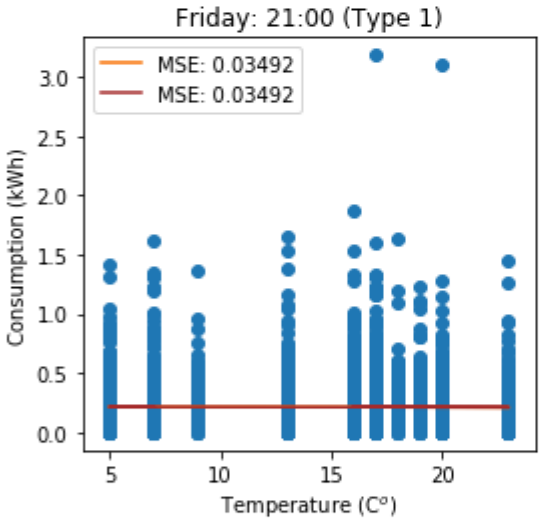

Friday: 19:00 (Type 2)
MSE: 0.1381
MSE: 0.13821
Consumption (kWh)
Temperature (C°)

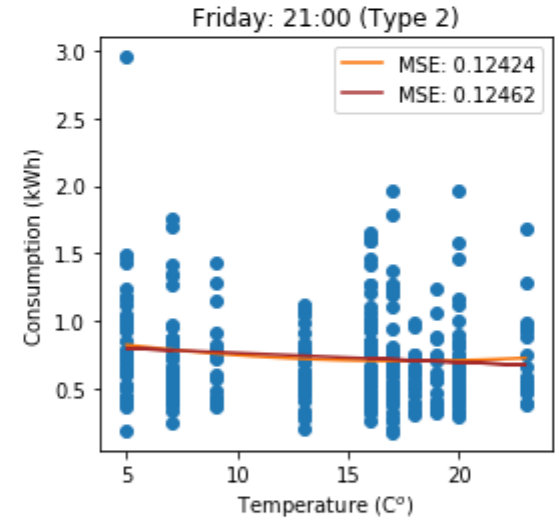

Friday: 19:00 (Type 3)
MSE: 0.03253
MSE: 0.03255
Consumption (kWh)
Temperature (C°)

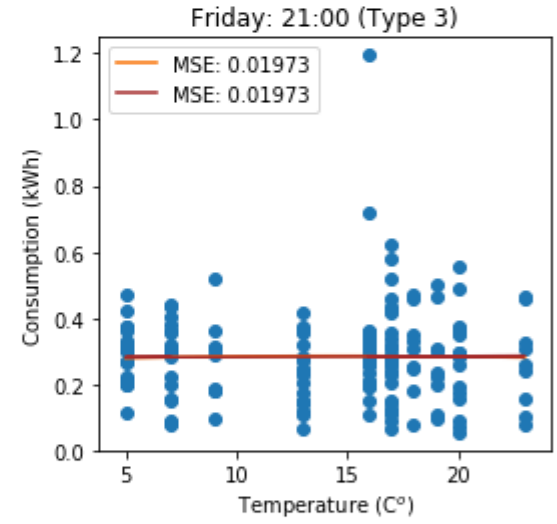

Friday: 20:00 (Type 1)
MSE: 0.04304
MSE: 0.04304
Consumption (kWh)
Temperature (C°)

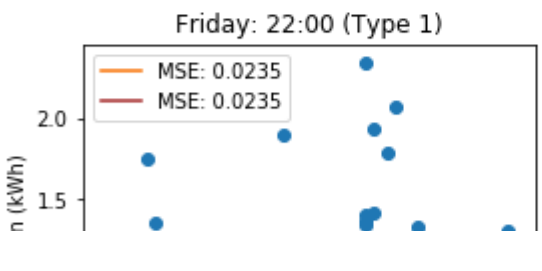

Friday: 20:00 (Type 2)
MSE: 0.15694
MSE: 0.1571
Consumption (kWh)
Temperature (C°)

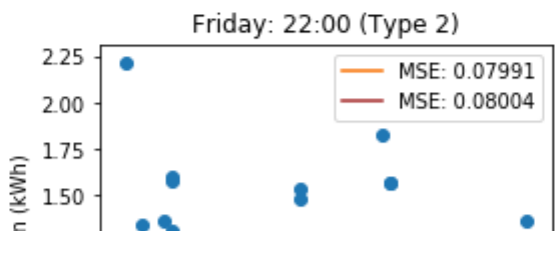

Friday: 20:00 (Type 3)
MSE: 0.01977
MSE: 0.01988
Consumption (kWh)
Temperature (C°)

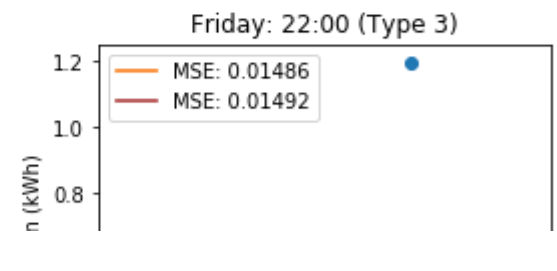

Friday: 21:00 (Type 1)
MSE: 0.03492
MSE: 0.03492
Consumption (kWh)
Temperature (C°)

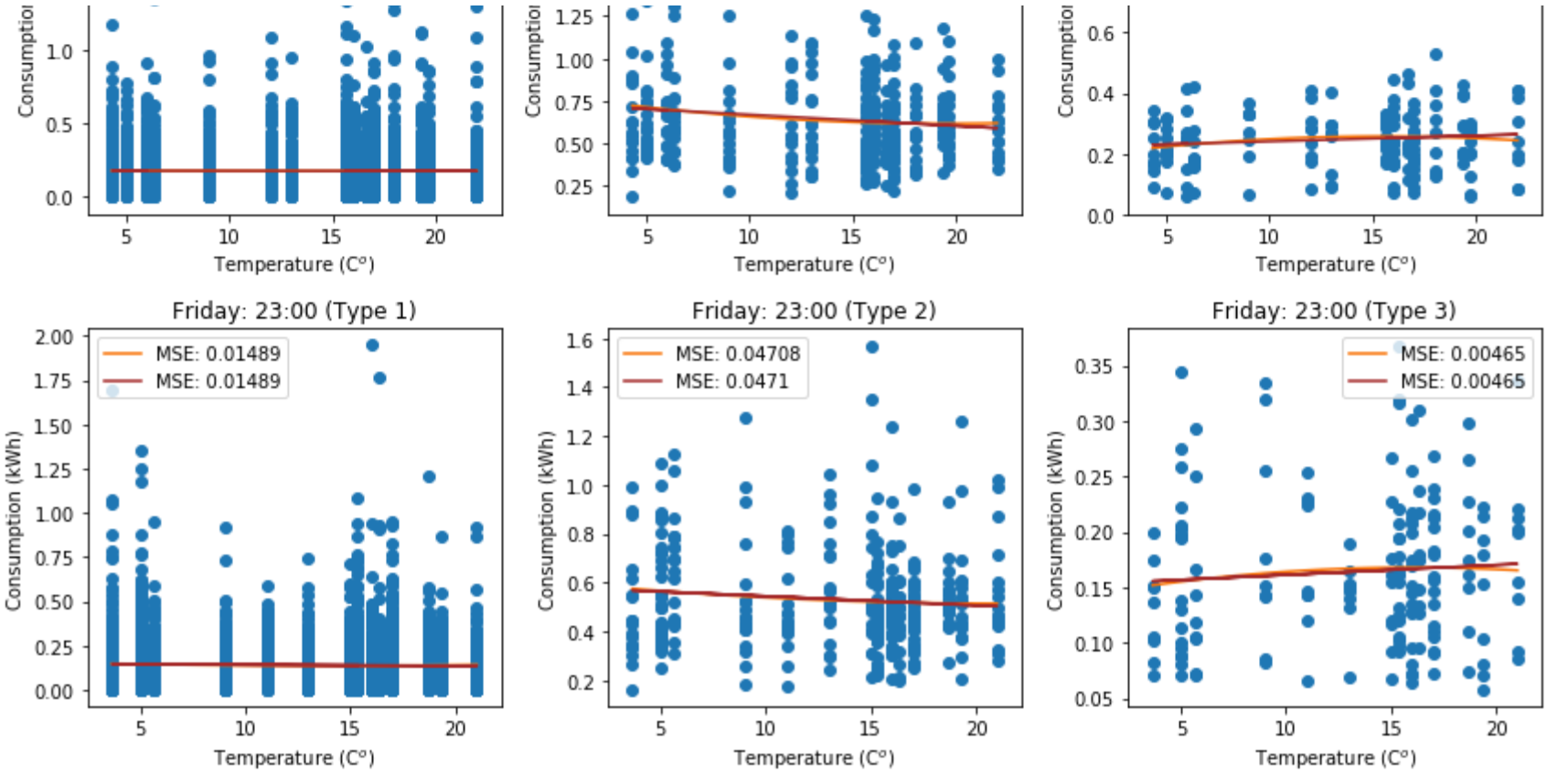

Friday: 21:00 (Type 2)
MSE: 0.12424
MSE: 0.12462
Consumption (kWh)
Temperature (C°)


Friday: 21:00 (Type 3)
MSE: 0.01973
MSE: 0.01973
Consumption (kWh)
Temperature (C°)

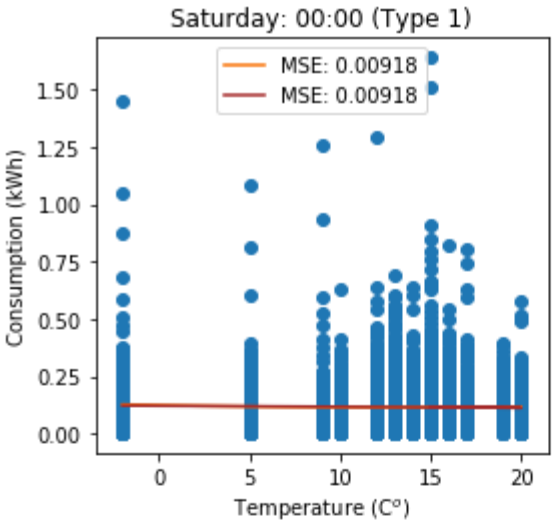

Friday: 22:00 (Type 1)
MSE: 0.0235
MSE: 0.0235

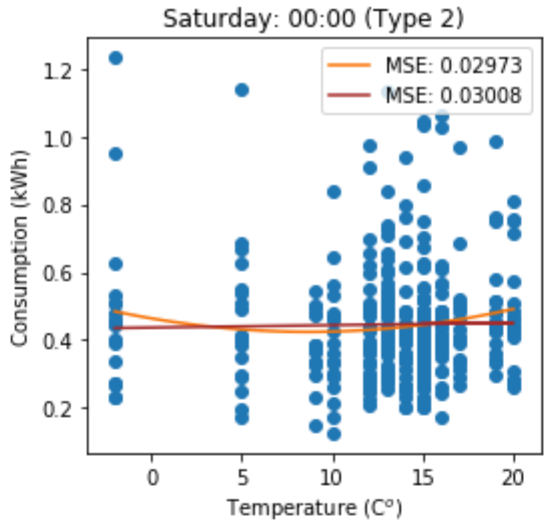

Friday: 22:00 (Type 2)
MSE: 0.07991
MSE: 0.08004

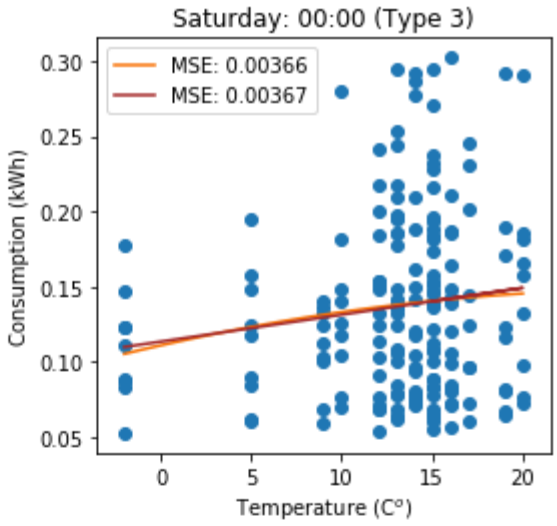

Friday: 22:00 (Type 3)
MSE: 0.01486
MSE: 0.01492

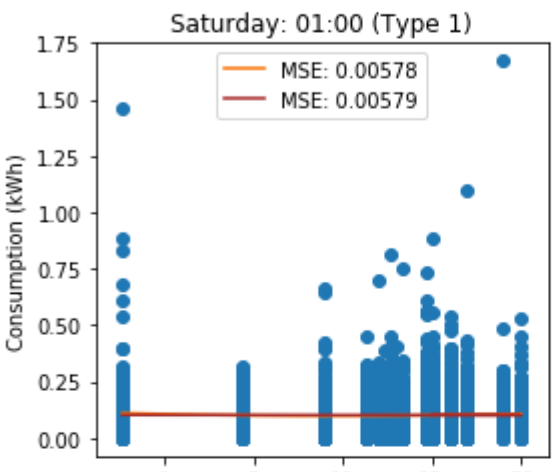

Friday: 23:00 (Type 1)
MSE: 0.01489
MSE: 0.01489
Friday: 23:00 (Type 2)
MSE: 0.04708
MSE: 0.0471
Friday: 23:00 (Type 3)
MSE: 0.00465
MSE: 0.00465
Consumption (kWh)
Temperature (C°)

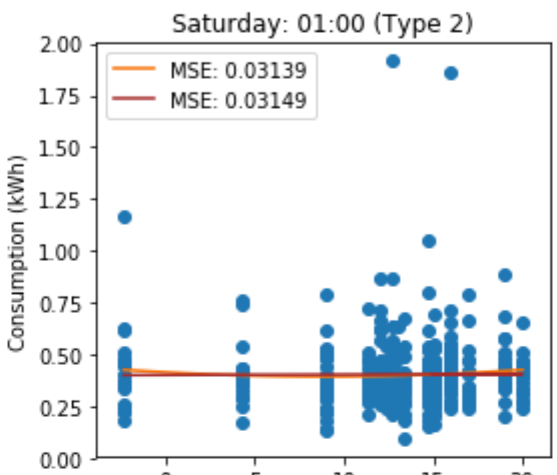
In [38]:
```
# Saturday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 5 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

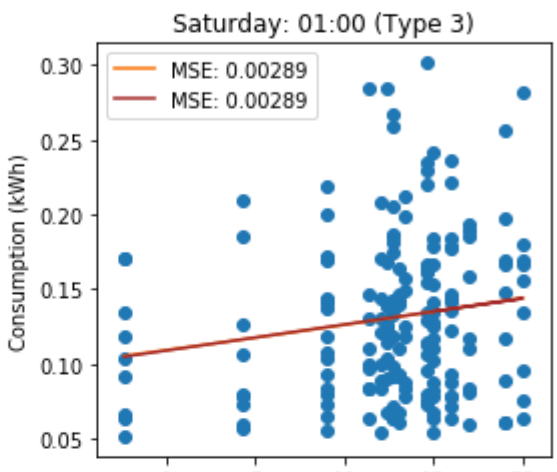

Saturday: 00:00 (Type 1)
MSE: 0.00918
MSE: 0.00918
Consumption (kWh)
Temperature (C°)

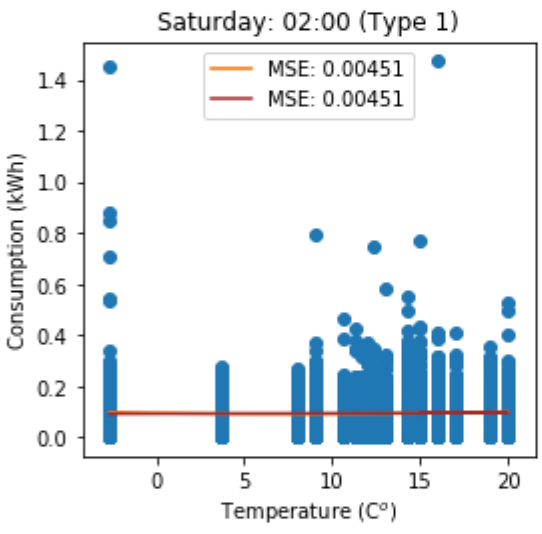

Saturday: 00:00 (Type 2)
MSE: 0.02973
MSE: 0.03008
Consumption (kWh)
Temperature (C°)

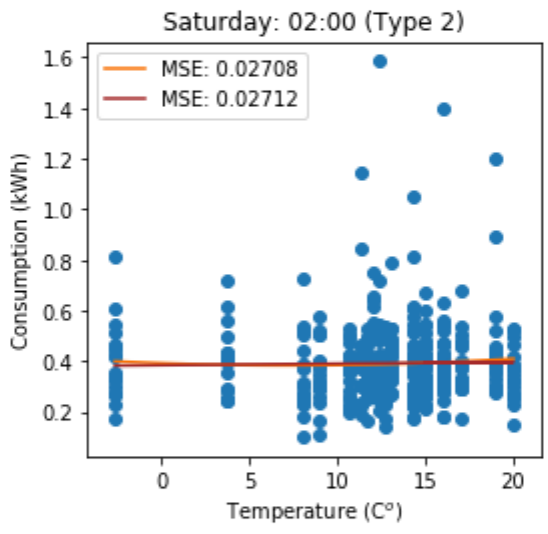

Saturday: 00:00 (Type 3)
MSE: 0.00366
MSE: 0.00367
Consumption (kWh)
Temperature (C°)

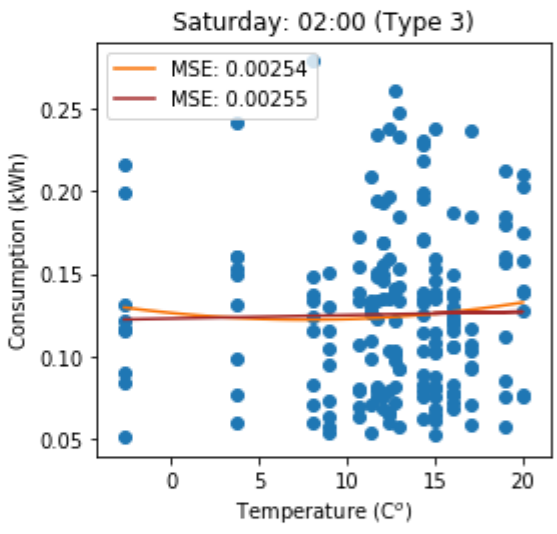

Saturday: 01:00 (Type 1)
MSE: 0.00578
MSE: 0.00579
Consumption (kWh)
Temperature (C°)

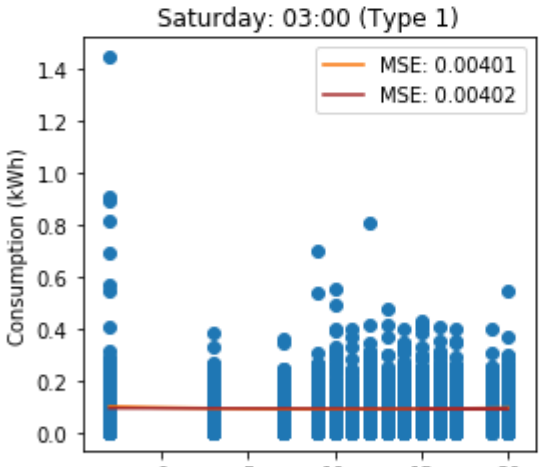

Saturday: 01:00 (Type 2)
MSE: 0.03139
MSE: 0.03149
Consumption (kWh)
Temperature (C°)

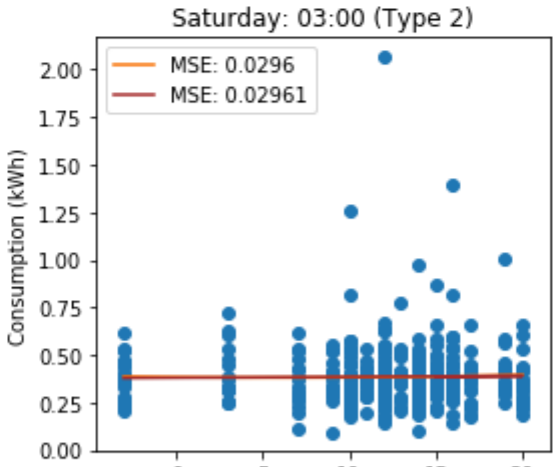

Saturday: 01:00 (Type 3)
MSE: 0.00289
MSE: 0.00289
Consumption (kWh)
Temperature (C°)

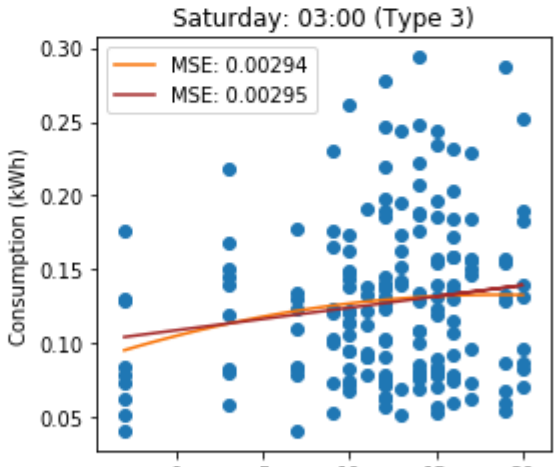

Saturday: 02:00 (Type 1)
MSE: 0.00451
MSE: 0.00451
Consumption (kWh)
Temperature (C°)

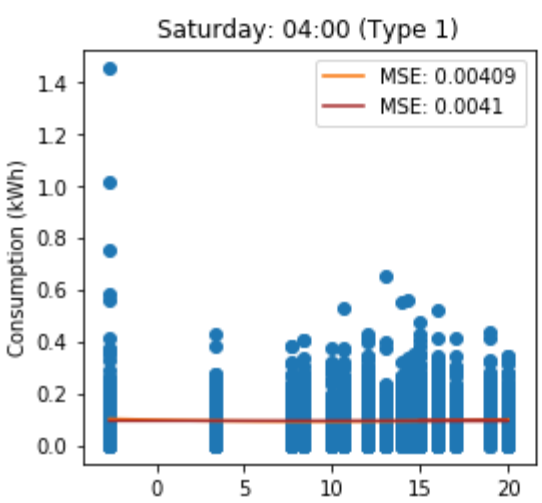

Saturday: 02:00 (Type 2)
MSE: 0.02708
MSE: 0.02712
Consumption (kWh)
Temperature (C°)

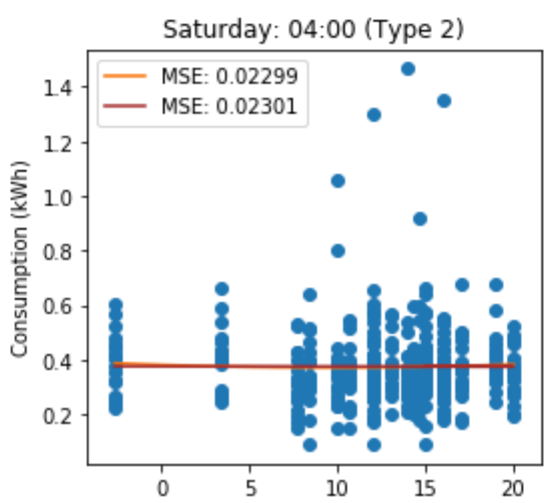

Saturday: 02:00 (Type 3)
MSE: 0.00254
MSE: 0.00255
Consumption (kWh)
Temperature (C°)

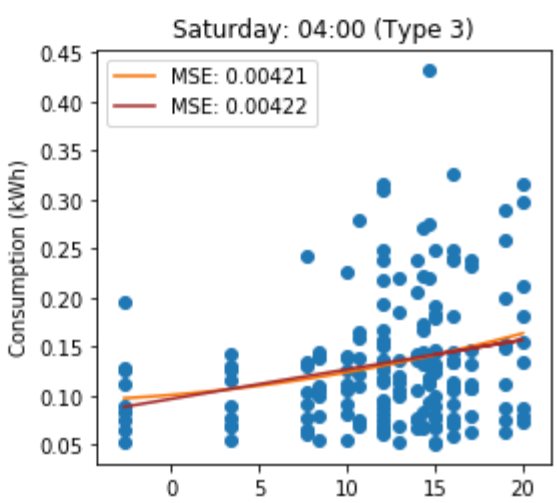

Saturday: 03:00 (Type 1)
MSE: 0.00401
MSE: 0.00402
Consumption (kWh)
Temperature (C°)

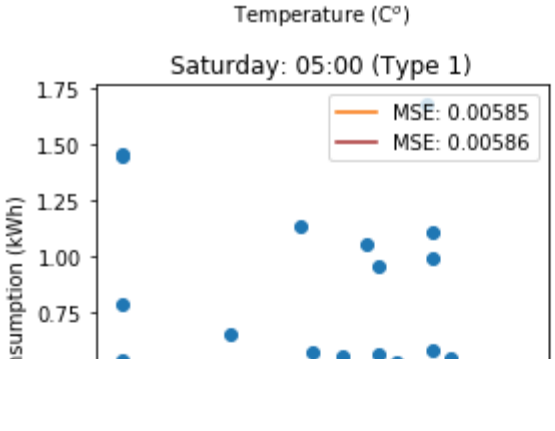

Saturday: 03:00 (Type 2)
MSE: 0.0296
MSE: 0.02961
Consumption (kWh)
Temperature (C°)

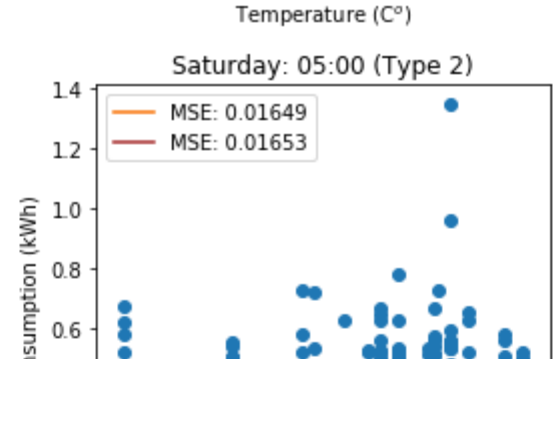

Saturday: 03:00 (Type 3)
MSE: 0.00294
MSE: 0.00295
Consumption (kWh)
Temperature (C°)

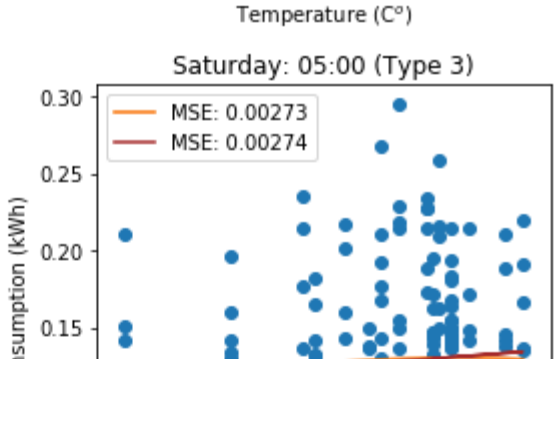

Saturday: 04:00 (Type 1)
MSE: 0.00409
MSE: 0.0041
Consumption (kWh)
Temperature (C°)

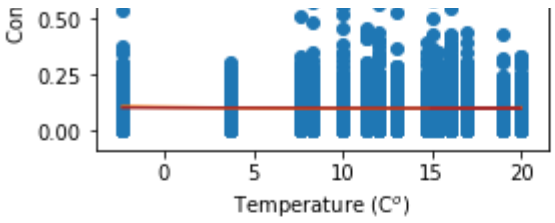

Saturday: 04:00 (Type 2)
MSE: 0.02299
MSE: 0.02301
Consumption (kWh)
Temperature (C°)

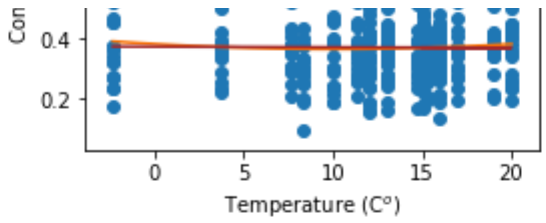

Saturday: 04:00 (Type 3)
MSE: 0.00421
MSE: 0.00422
Consumption (kWh)
Temperature (C°)

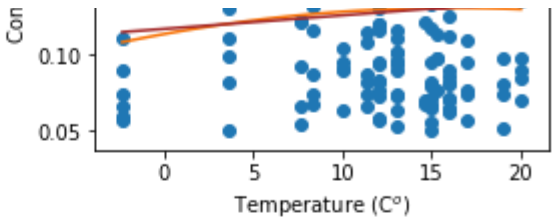

Saturday: 05:00 (Type 1)
MSE: 0.00585
MSE: 0.00586
Temperature (C°)

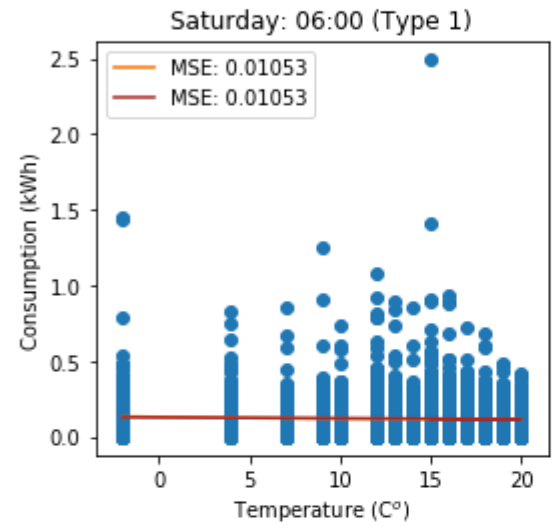

Saturday: 05:00 (Type 2)
MSE: 0.01649
MSE: 0.01653

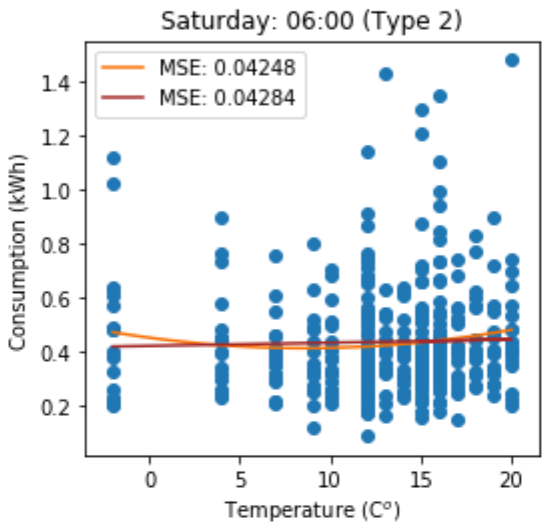

Saturday: 05:00 (Type 3)
MSE: 0.00273
MSE: 0.00274
Consumption (kWh)

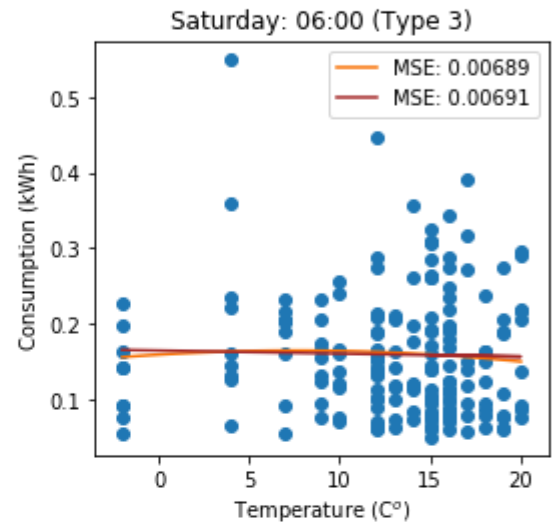

Temperature (C°)

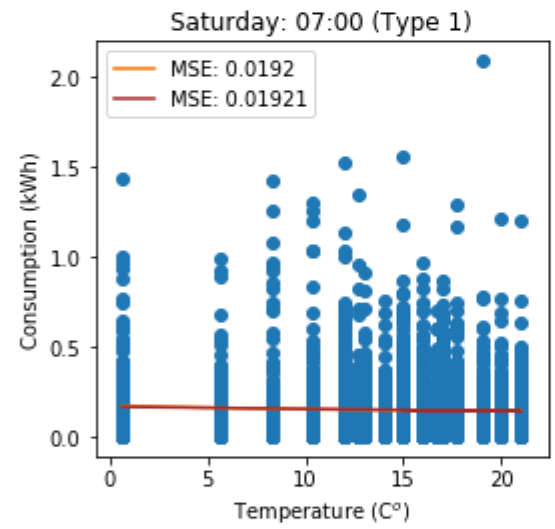

Temperature (C°)

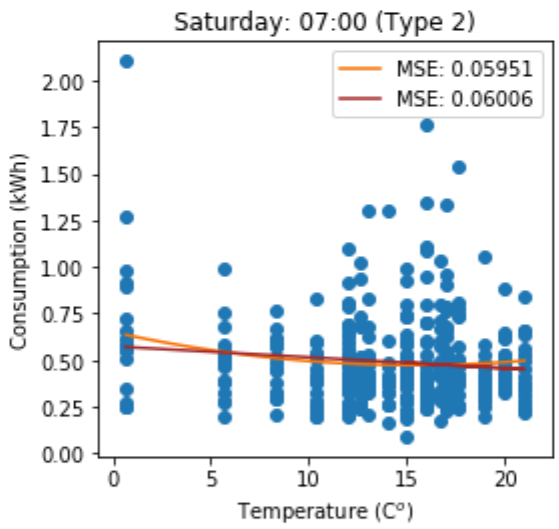

Temperature (C°)

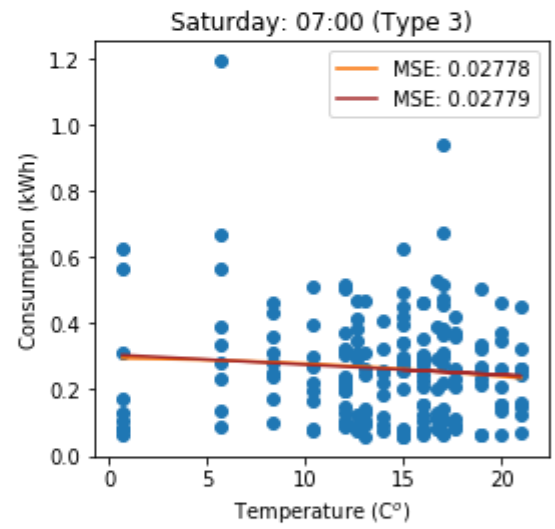

Saturday: 06:00 (Type 1)
MSE: 0.01053
MSE: 0.01053
Consumption (kWh)
Temperature (C°)

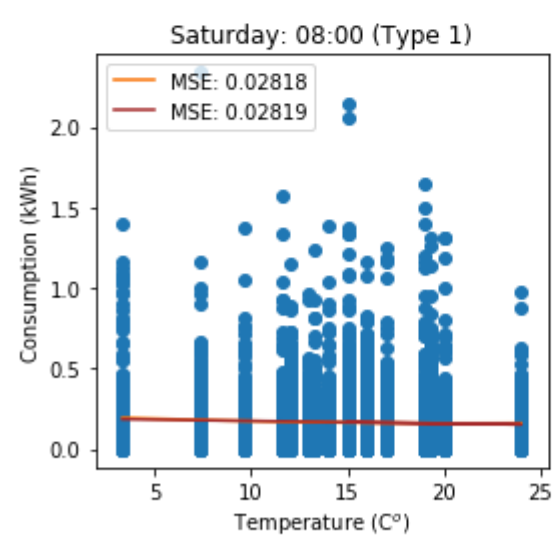

Saturday: 06:00 (Type 2)
MSE: 0.04248
MSE: 0.04284
Consumption (kWh)
Temperature (C°)

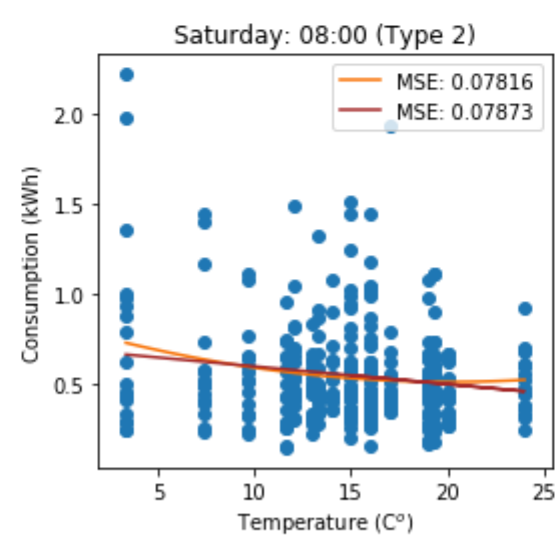

Saturday: 06:00 (Type 3)
MSE: 0.00689
MSE: 0.00691
Consumption (kWh)
Temperature (C°)

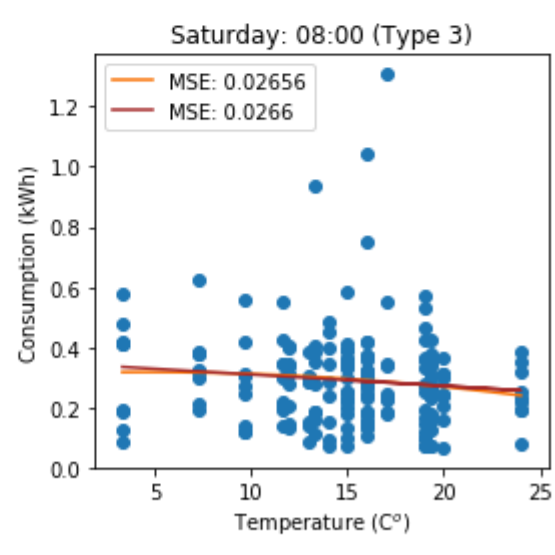

Saturday: 07:00 (Type 1)
MSE: 0.0192
MSE: 0.01921
Consumption (kWh)
Temperature (C°)

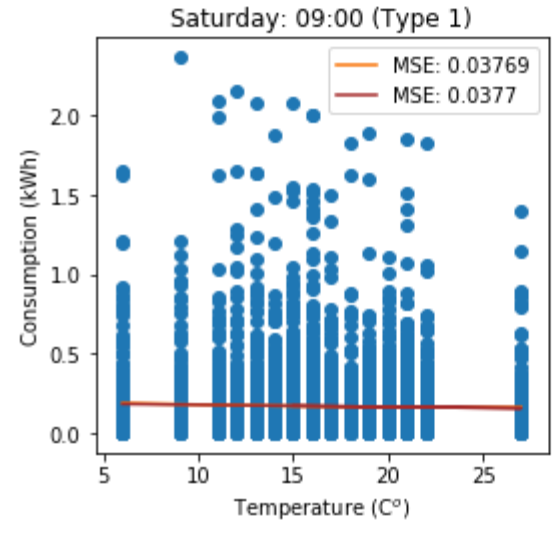

Saturday: 07:00 (Type 2)
MSE: 0.05951
MSE: 0.06006
Consumption (kWh)
Temperature (C°)

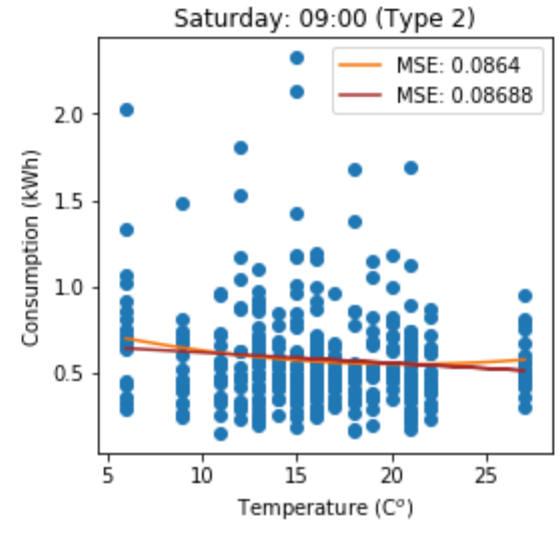

Saturday: 07:00 (Type 3)
MSE: 0.02778
MSE: 0.02779
Consumption (kWh)
Temperature (C°)

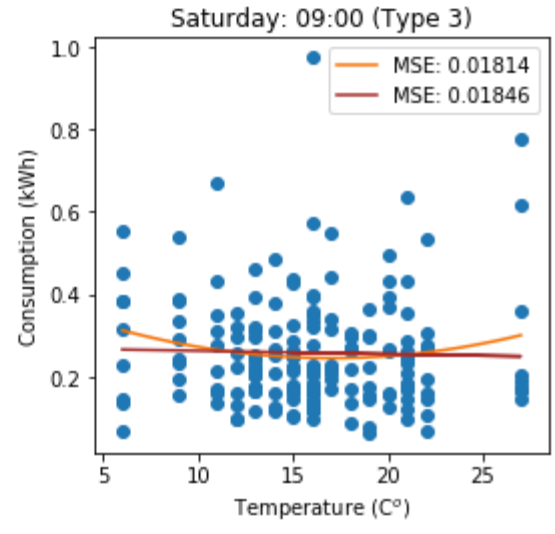

Saturday: 08:00 (Type 1)
MSE: 0.02818
MSE: 0.02819
Consumption (kWh)
Temperature (C°)

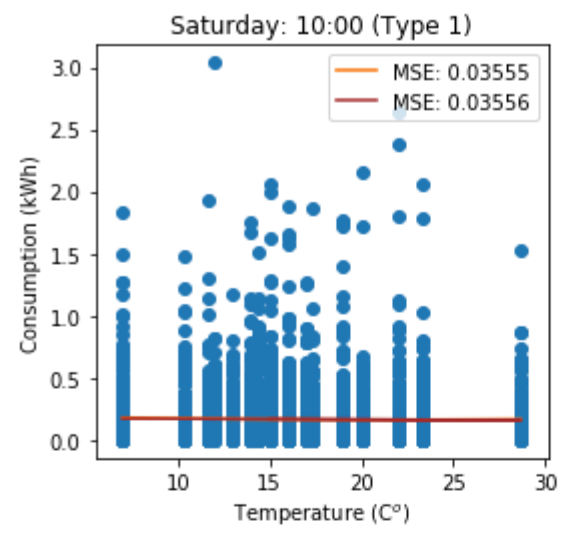

Saturday: 08:00 (Type 2)
MSE: 0.07816
MSE: 0.07873
Consumption (kWh)
Temperature (C°)

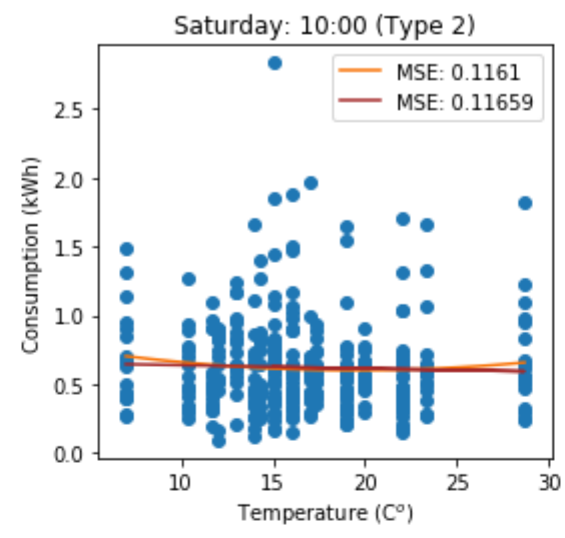

Saturday: 08:00 (Type 3)
MSE: 0.02656
MSE: 0.0266
Consumption (kWh)
Temperature (C°)

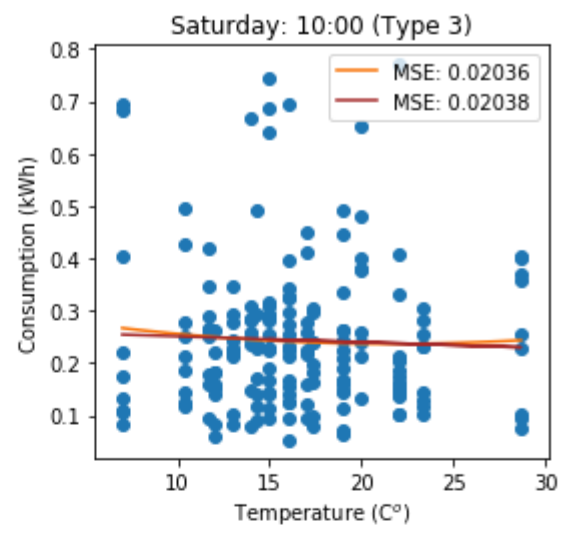

Saturday: 09:00 (Type 1)
MSE: 0.03769
MSE: 0.0377
Consumption (kWh)
Temperature (C°)


Saturday: 09:00 (Type 2)
MSE: 0.0864
MSE: 0.08688
Consumption (kWh)
Temperature (C°)


Saturday: 09:00 (Type 3)
MSE: 0.01814
MSE: 0.01846
Consumption (kWh)
Temperature (C°)


Saturday: 10:00 (Type 1)
MSE: 0.03555
MSE: 0.03556
Consumption (kWh)
Temperature (C°)

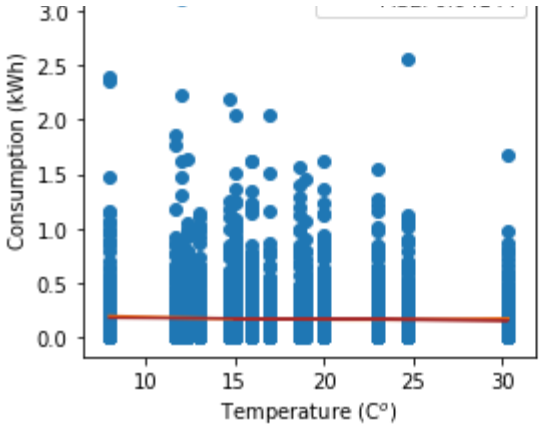

Saturday: 10:00 (Type 2)
MSE: 0.1161
MSE: 0.11659
Consumption (kWh)
Temperature (C°)

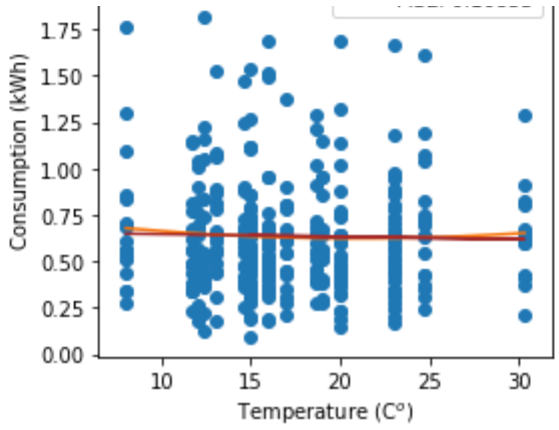

Saturday: 10:00 (Type 3)
MSE: 0.02036
MSE: 0.02038
Consumption (kWh)
Temperature (C°)

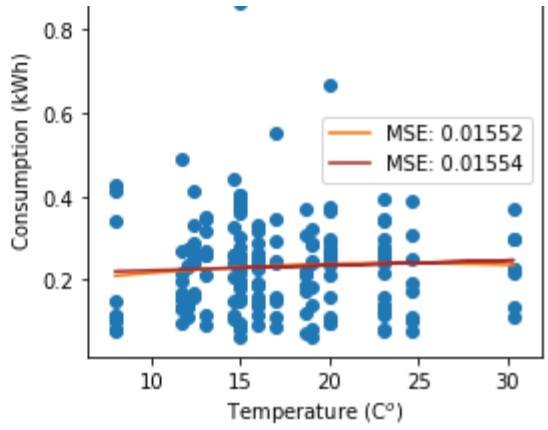
Saturday: 11:00 (Type 1)
MSE: 0.0414
MSE: 0.04144

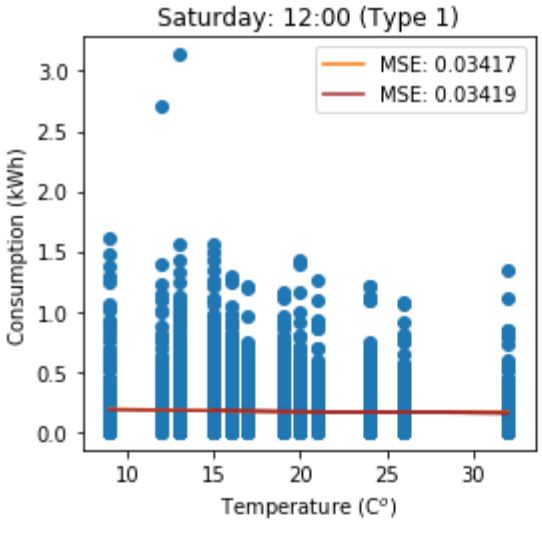
Saturday: 11:00 (Type 2)
MSE: 0.10839
MSE: 0.10853

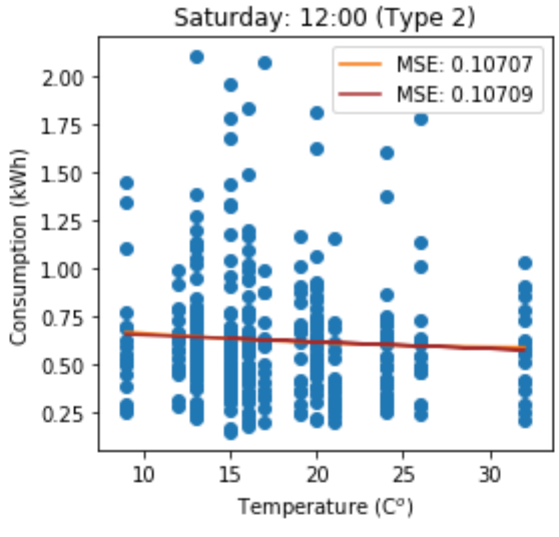
Saturday: 11:00 (Type 3)

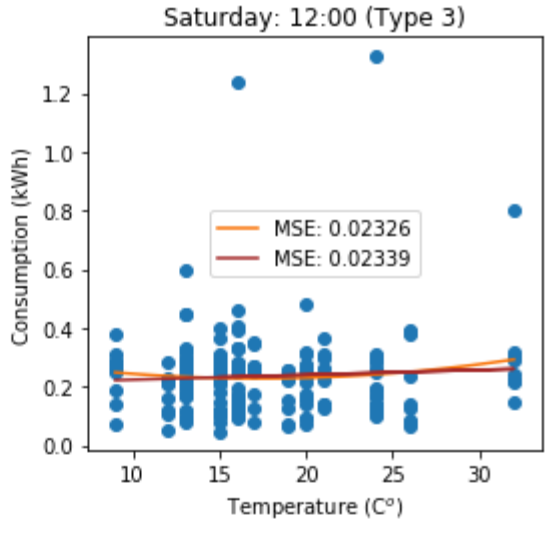

Consumption (kWh)
Temperature (C°)

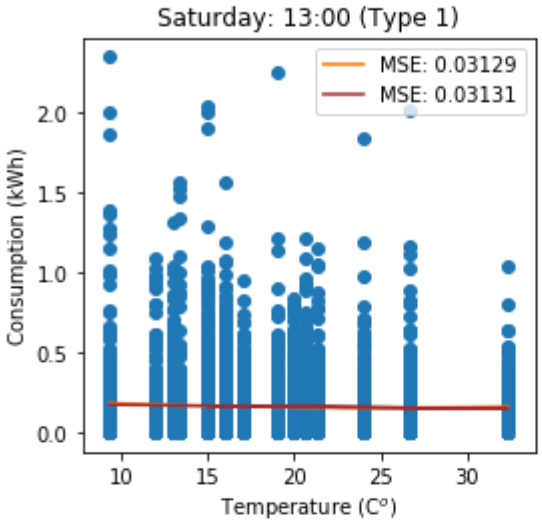

Consumption (kWh)
Temperature (C°)

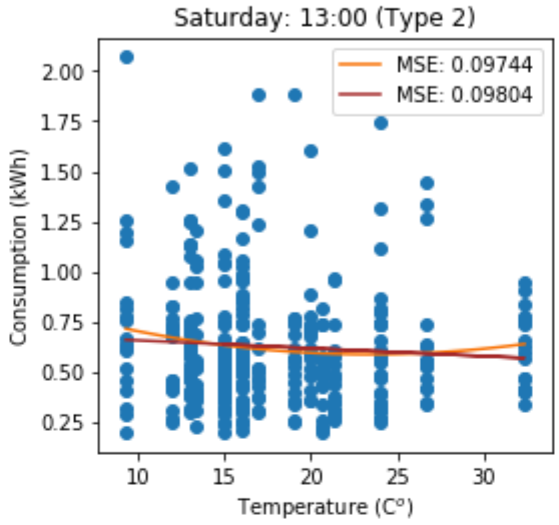

MSE: 0.01552
MSE: 0.01554
Consumption (kWh)
Temperature (C°)

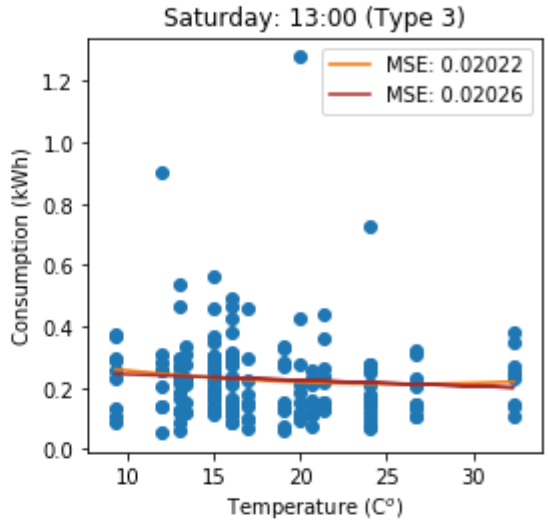

Saturday: 12:00 (Type 1)
MSE: 0.03417
MSE: 0.03419
Consumption (kWh)
Temperature (C°)

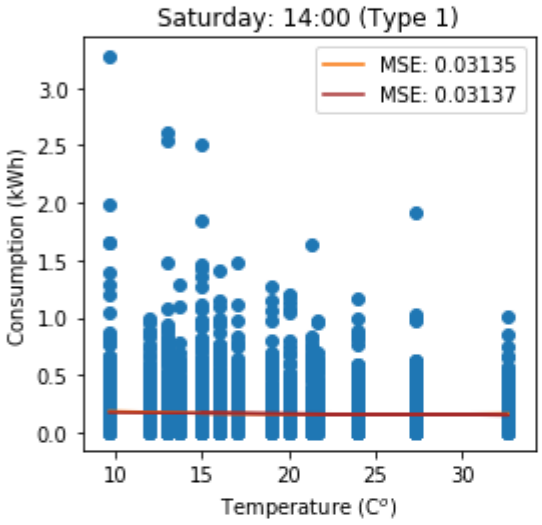

Saturday: 12:00 (Type 2)
MSE: 0.10707
MSE: 0.10709
Consumption (kWh)
Temperature (C°)

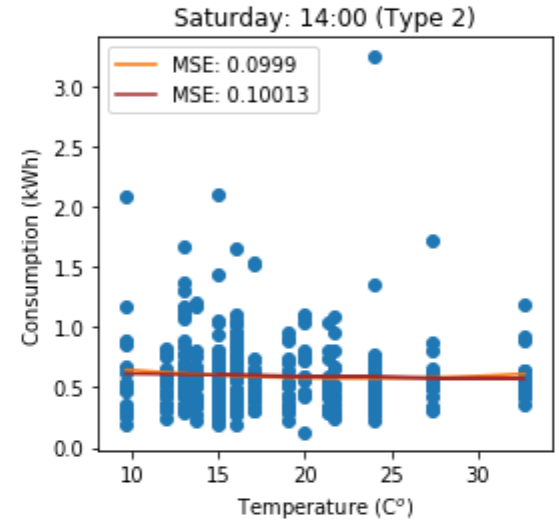

Saturday: 12:00 (Type 3)
MSE: 0.02326
MSE: 0.02339
Consumption (kWh)
Temperature (C°)

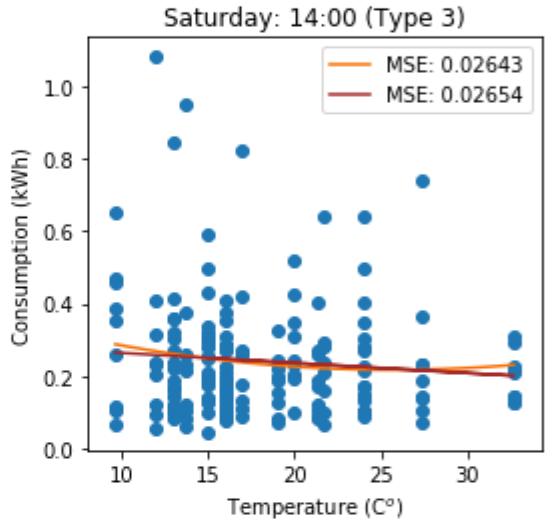

Saturday: 13:00 (Type 1)
MSE: 0.03129
MSE: 0.03131
Consumption (kWh)
Temperature (C°)

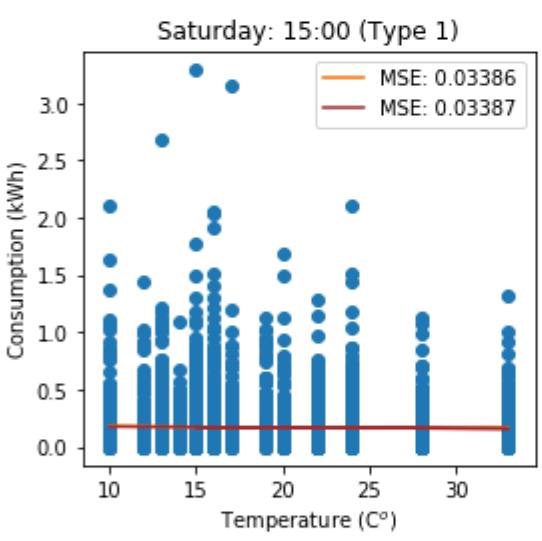

Saturday: 13:00 (Type 2)
MSE: 0.09744
MSE: 0.09804
Consumption (kWh)
Temperature (C°)

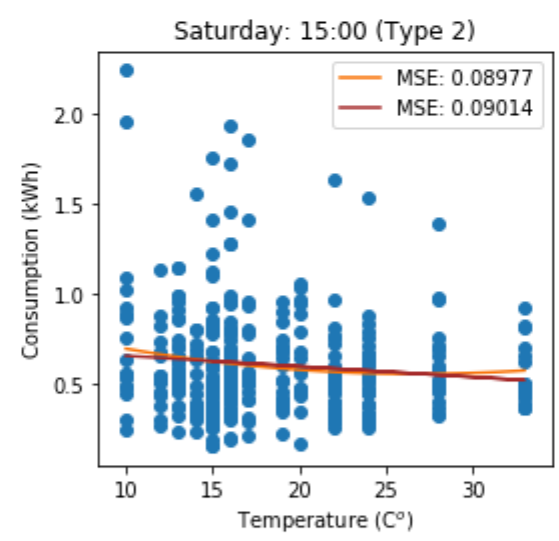

Saturday: 13:00 (Type 3)
MSE: 0.02022
MSE: 0.02026
Consumption (kWh)
Temperature (C°)

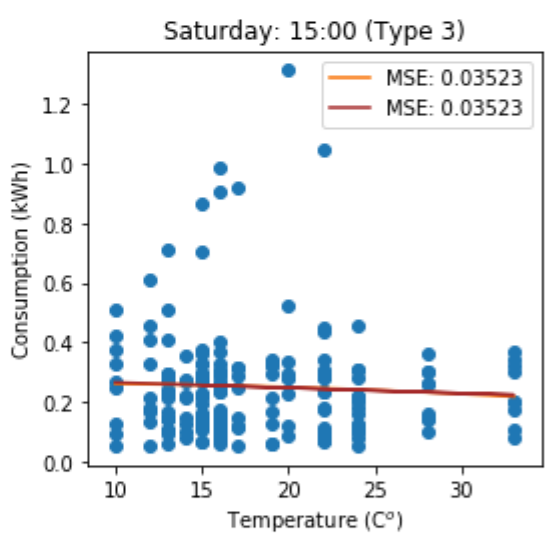

Saturday: 14:00 (Type 1)
MSE: 0.03135
MSE: 0.03137
Consumption (kWh)
Temperature (C°)

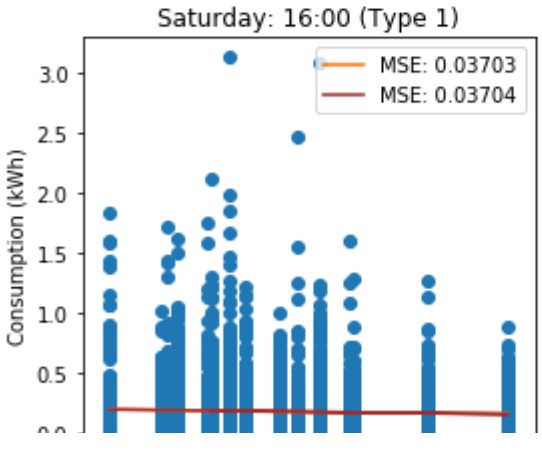

Saturday: 14:00 (Type 2)
MSE: 0.0999
MSE: 0.10013
Consumption (kWh)
Temperature (C°)

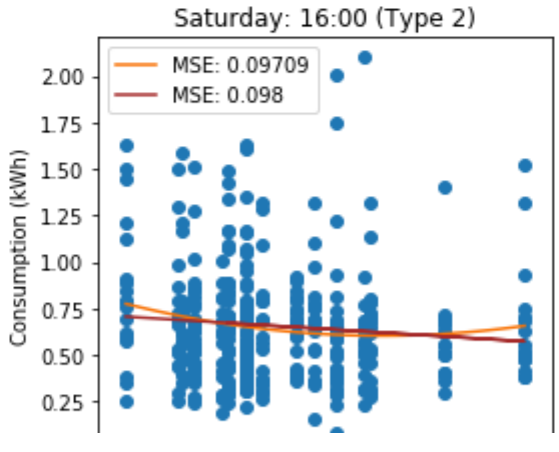

Saturday: 14:00 (Type 3)
MSE: 0.02643
MSE: 0.02654
Consumption (kWh)
Temperature (C°)

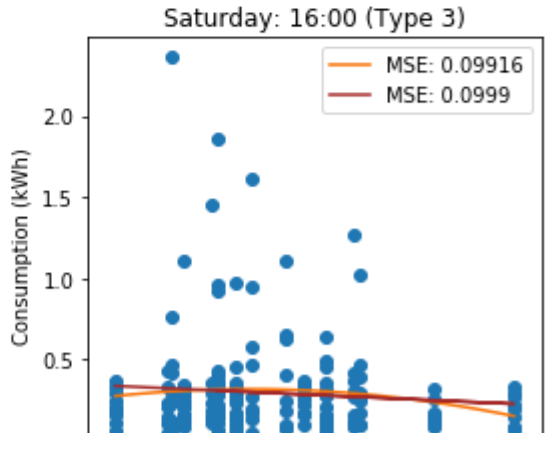

Saturday: 15:00 (Type 1)
MSE: 0.03386
MSE: 0.03387
Consumption (kWh)
Temperature (C°)

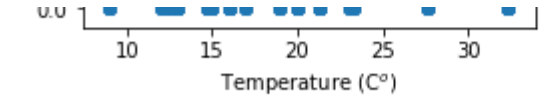

Saturday: 15:00 (Type 2)
MSE: 0.08977
MSE: 0.09014
Consumption (kWh)
Temperature (C°)

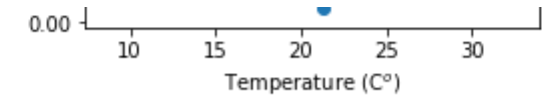

Saturday: 15:00 (Type 3)
MSE: 0.03523
MSE: 0.03523
Consumption (kWh)
Temperature (C°)

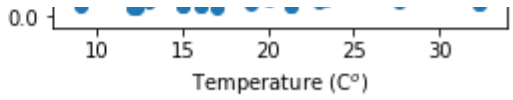

Saturday: 16:00 (Type 1)
MSE: 0.03703
MSE: 0.03704
Consumption (kWh)

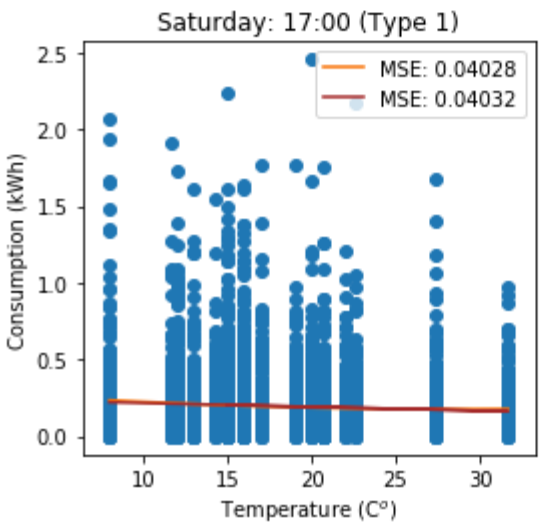

Saturday: 16:00 (Type 2)
MSE: 0.09709
MSE: 0.098
Consumption (kWh)

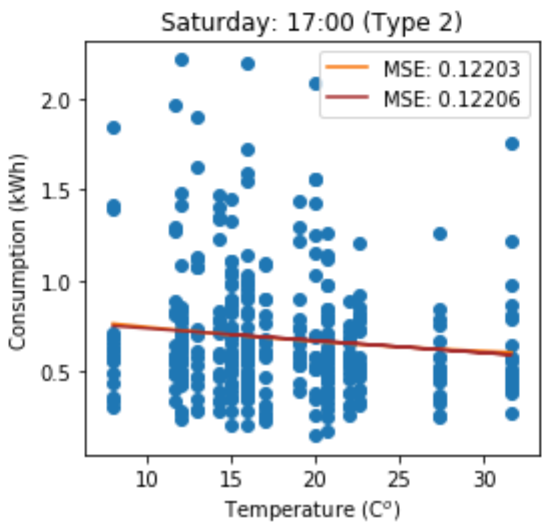

Saturday: 16:00 (Type 3)
MSE: 0.09916
MSE: 0.0999
Consumption (kWh)

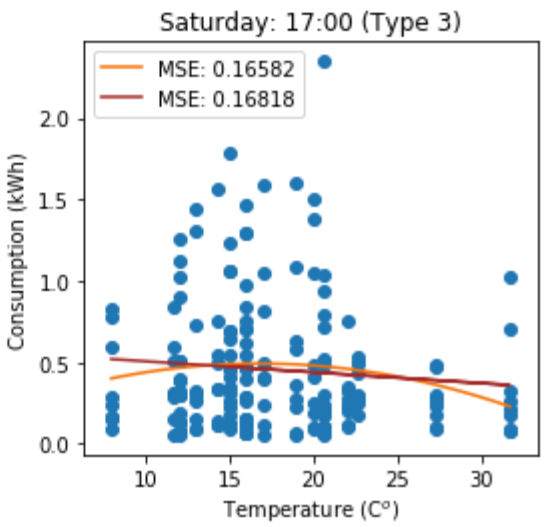

Temperature (C°)

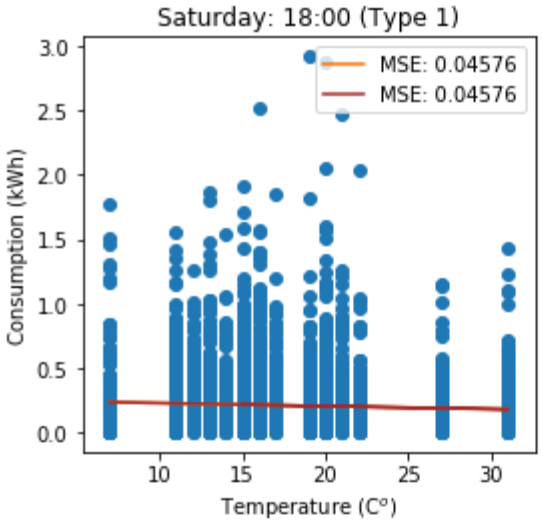

Temperature (C°)

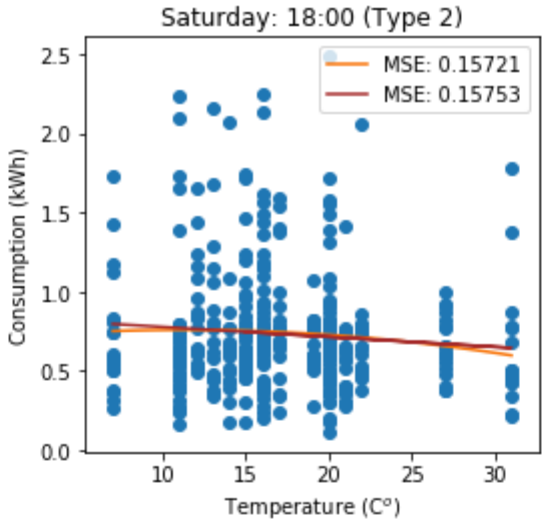

Temperature (C°)

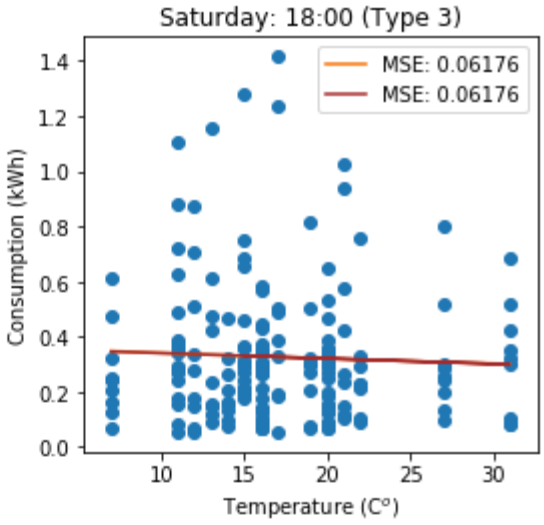

Saturday: 17:00 (Type 1)
MSE: 0.04028
MSE: 0.04032
Consumption (kWh)
Temperature (C°)

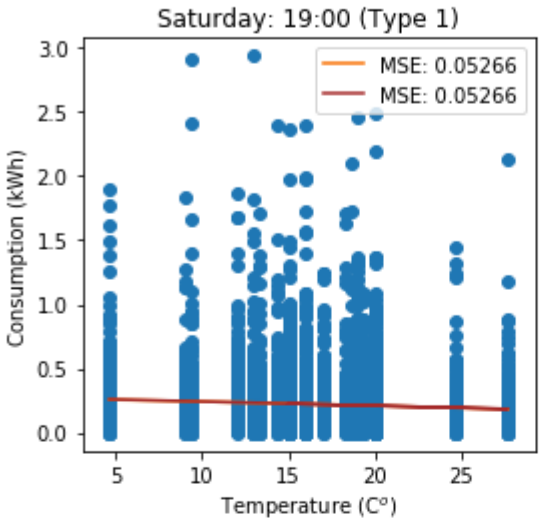

Saturday: 17:00 (Type 2)
MSE: 0.12203
MSE: 0.12206
Consumption (kWh)
Temperature (C°)

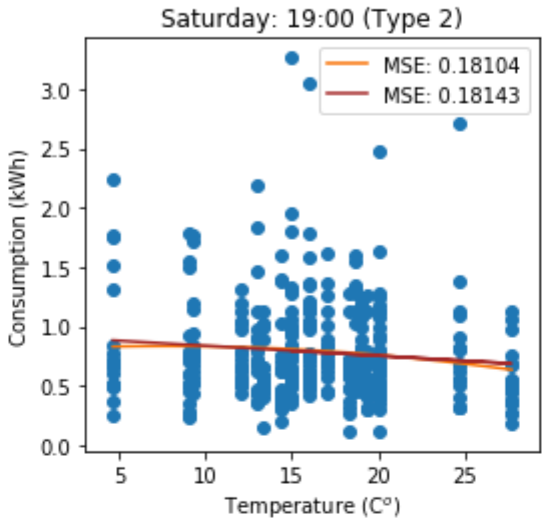

Saturday: 17:00 (Type 3)
MSE: 0.16582
MSE: 0.16818
Consumption (kWh)
Temperature (C°)

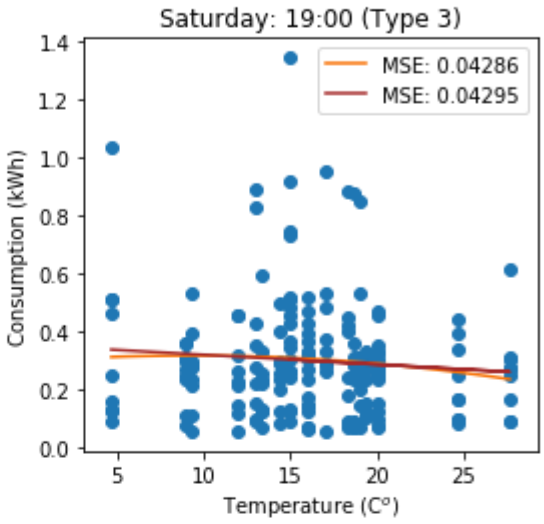

Saturday: 18:00 (Type 1)
MSE: 0.04576
MSE: 0.04576
Consumption (kWh)
Temperature (C°)

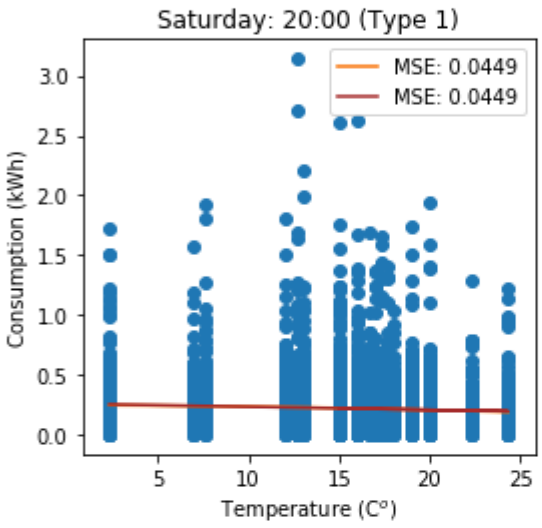

Saturday: 18:00 (Type 2)
MSE: 0.15721
MSE: 0.15753
Consumption (kWh)
Temperature (C°)

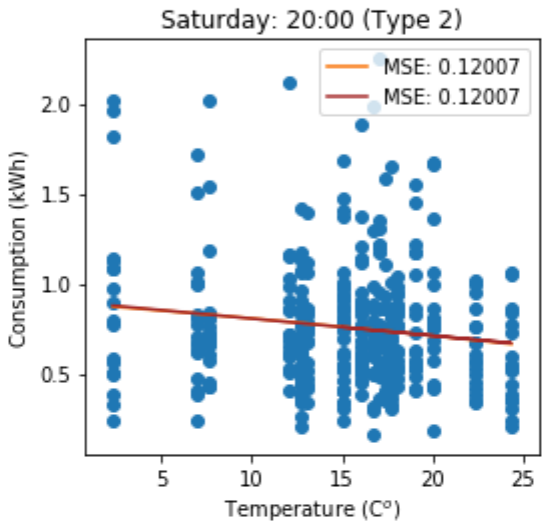

Saturday: 18:00 (Type 3)
MSE: 0.06176
MSE: 0.06176
Consumption (kWh)
Temperature (C°)

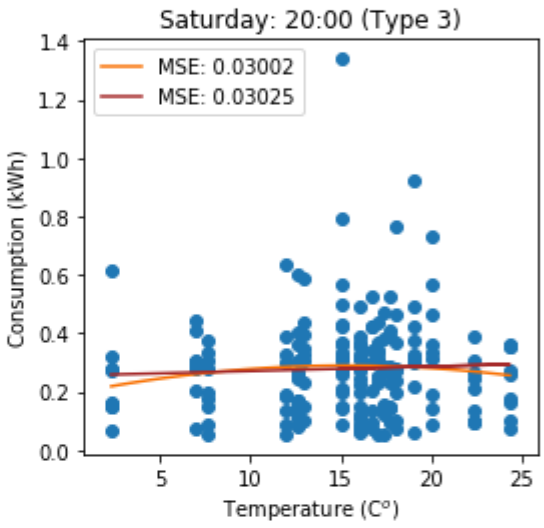

Saturday: 19:00 (Type 1)
MSE: 0.05266
MSE: 0.05266
Consumption (kWh)
Temperature (C°)

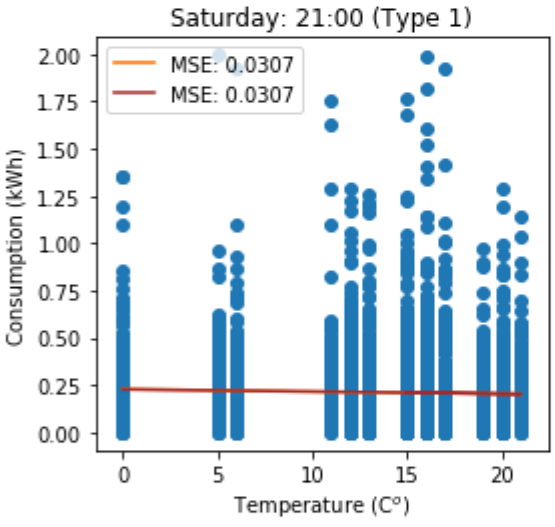

Saturday: 19:00 (Type 2)
MSE: 0.18104
MSE: 0.18143
Consumption (kWh)
Temperature (C°)

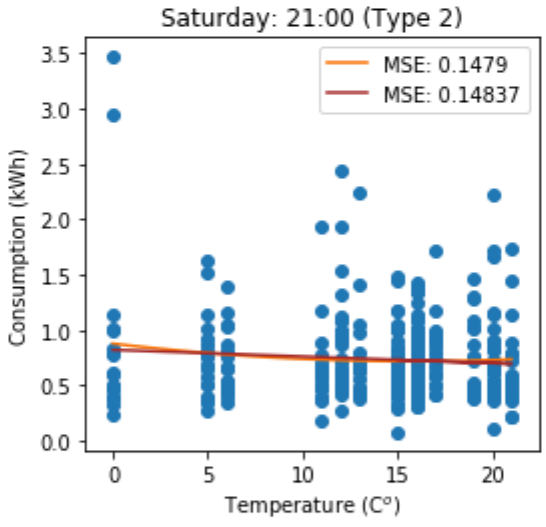

Saturday: 19:00 (Type 3)
MSE: 0.04286
MSE: 0.04295
Consumption (kWh)
Temperature (C°)

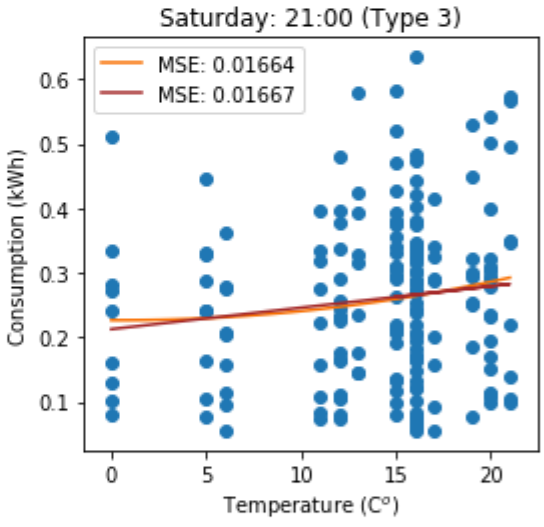

Saturday: 20:00 (Type 1)
MSE: 0.0449
MSE: 0.0449
Consumption (kWh)
Temperature (C°)

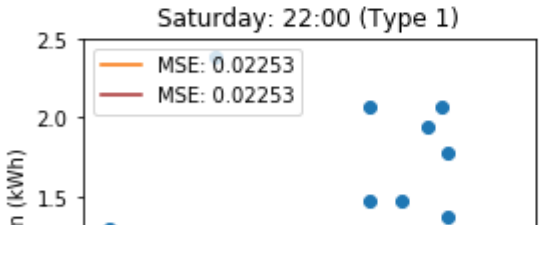

Saturday: 20:00 (Type 2)
MSE: 0.12007
MSE: 0.12007
Consumption (kWh)
Temperature (C°)

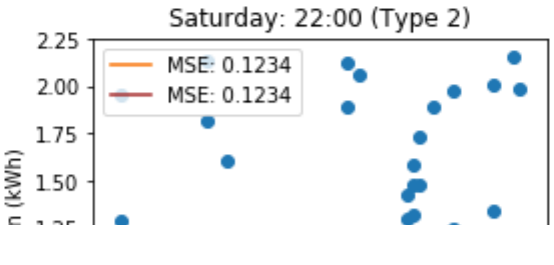

Saturday: 20:00 (Type 3)
MSE: 0.03002
MSE: 0.03025
Consumption (kWh)
Temperature (C°)

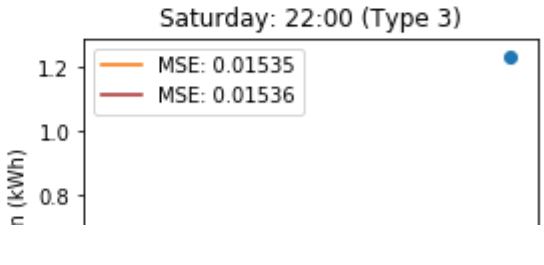

Saturday: 21:00 (Type 1)
MSE: 0.0307
MSE: 0.0307
Consumption (kWh)
Temperature (C°)

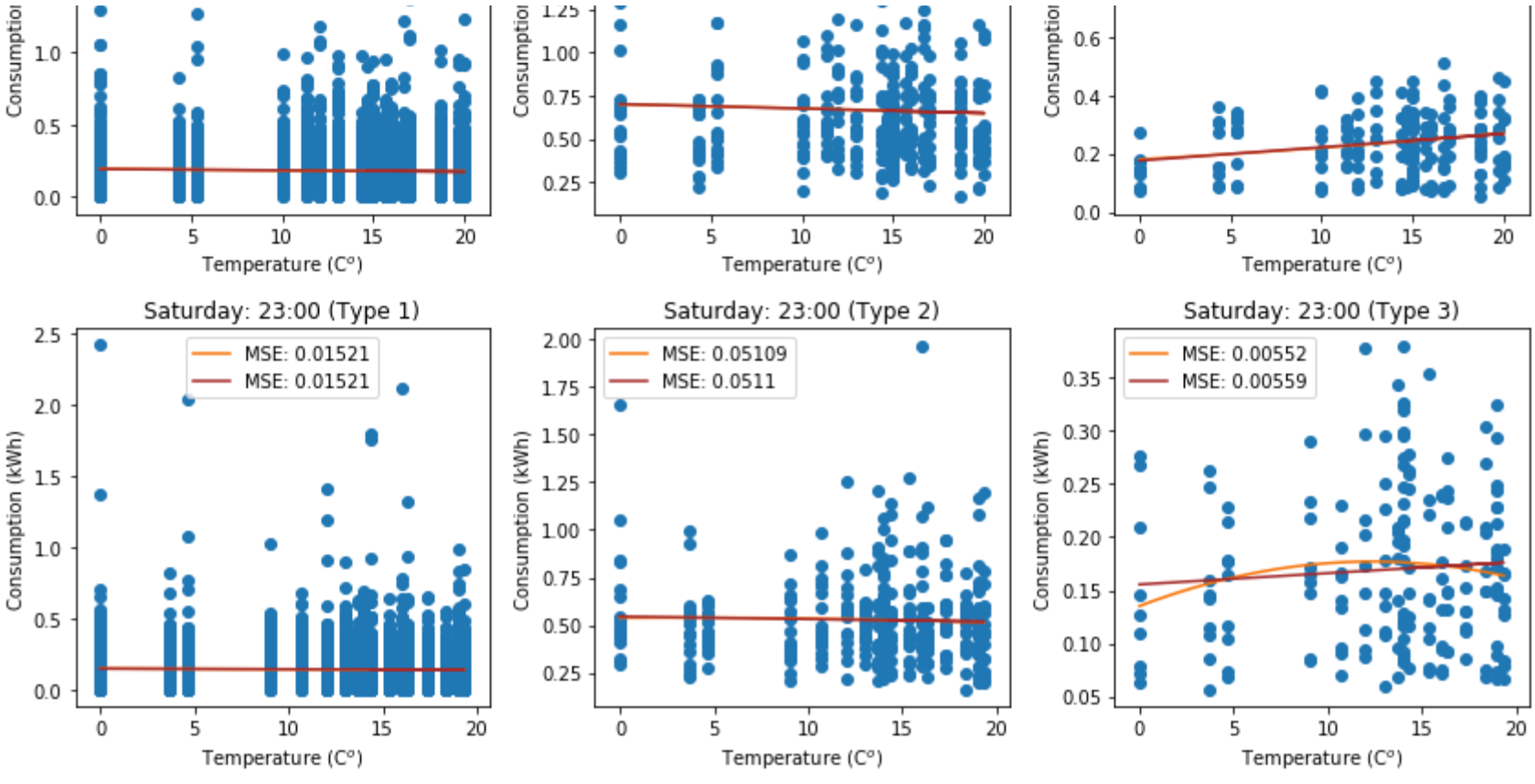

Saturday: 21:00 (Type 2)
MSE: 0.1479
MSE: 0.14837
Consumption (kWh)
Temperature (C°)


Saturday: 21:00 (Type 3)
MSE: 0.01664
MSE: 0.01667
Consumption (kWh)
Temperature (C°)

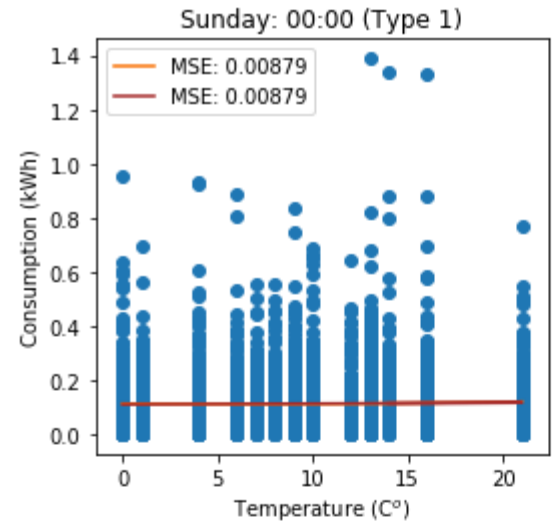

Saturday: 22:00 (Type 1)
MSE: 0.02253
MSE: 0.02253

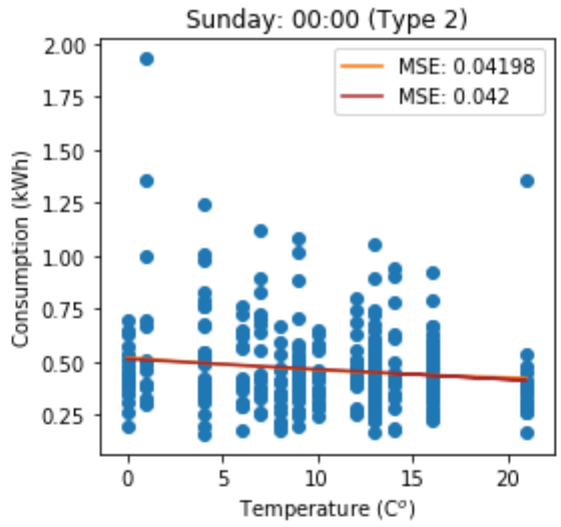

Saturday: 22:00 (Type 2)
MSE: 0.1234
MSE: 0.1234

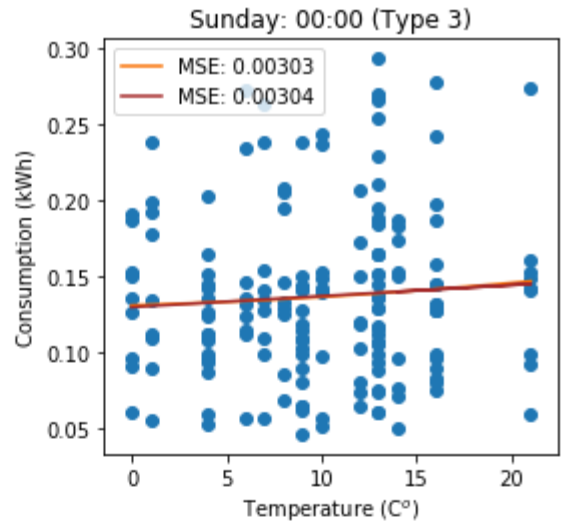

Saturday: 22:00 (Type 3)
MSE: 0.01535
MSE: 0.01536

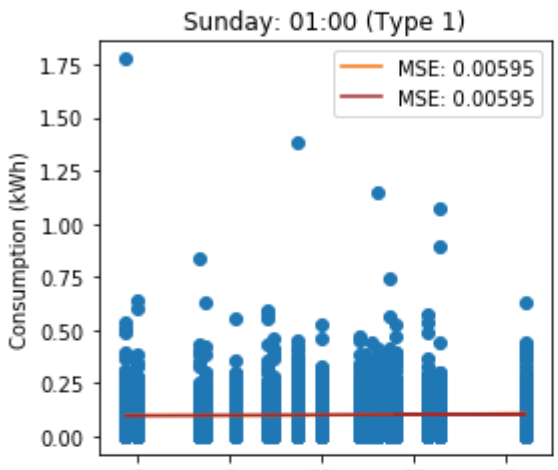

Saturday: 23:00 (Type 1)
MSE: 0.01521
MSE: 0.01521
Saturday: 23:00 (Type 2)
MSE: 0.05109
MSE: 0.0511
Saturday: 23:00 (Type 3)
MSE: 0.00552
MSE: 0.00559
Consumption (kWh)
Temperature (C°)

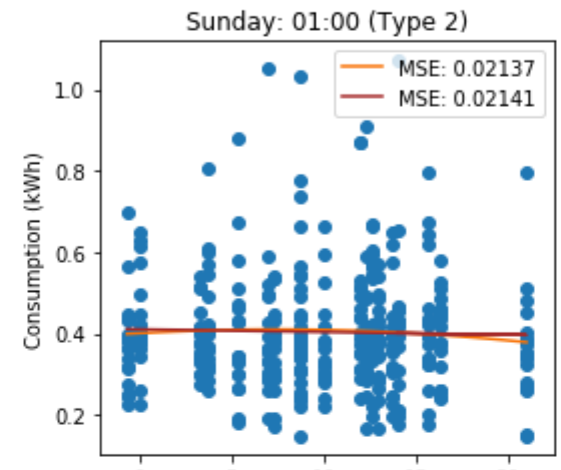
In [39]:
```
# Sunday hourly regressions in warm seasons
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
day_of_week = 6 # 0 is Monday, 6 is Sunday
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        if i <= 9:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_datetime
(z.GMT).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': 0' + str(i) + ':00 (Type ' + str(g+1) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
        else:
            z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
            y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].House
)]
            y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for the corr
esponding day of a week
            y.set_index('GMT', inplace = True)
            y = y.fillna(y.mean()).values
            y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

            x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z.GMT
).dt.dayofweek == day_of_week)].values
            x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct the x
 variable
            xp = np.linspace(min(x), max(x), 50) # polynomial regression
            z = np.polyfit(x, y, 2)
            p = np.poly1d(z)
            ax_Ntou[-1].plot(xp, p(xp), color = 'xkcd:orange', label = 'MSE: ' + str(round(mean_squared_error
(y, p(x)), 5)))
            x = x.reshape(-1, 1)
            y = y.reshape(-1, 1)
            lm.fit(x, y)
            ax_Ntou[-1].plot(x, lm.predict(x), color = 'brown', label = 'MSE: ' + str(round(mean_squared_erro
r(y, lm.predict(x)), 5)))
            ax_Ntou[-1].scatter(x, y)
            ax_Ntou[-1].set_title(weeks[day_of_week] + ': ' + str(i) + ':00 (Type ' + str(g+1) + ')')
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            ax_Ntou[-1].legend()
plt.tight_layout()
```

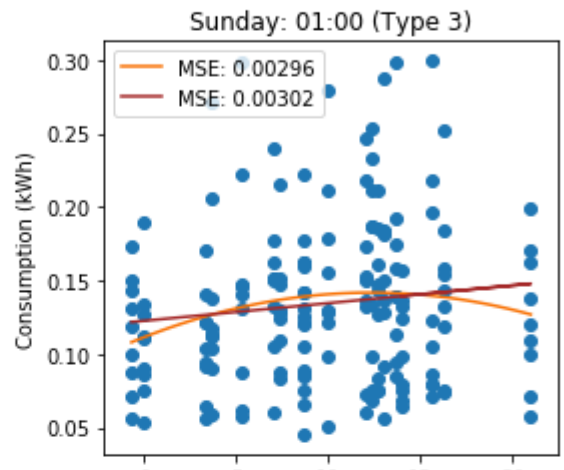

Sunday: 00:00 (Type 1)
MSE: 0.00879
MSE: 0.00879
Consumption (kWh)
Temperature (C°)

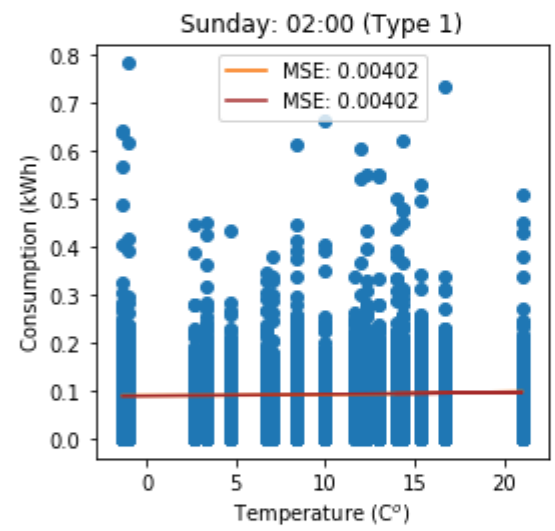

Sunday: 00:00 (Type 2)
MSE: 0.04198
MSE: 0.042
Consumption (kWh)
Temperature (C°)

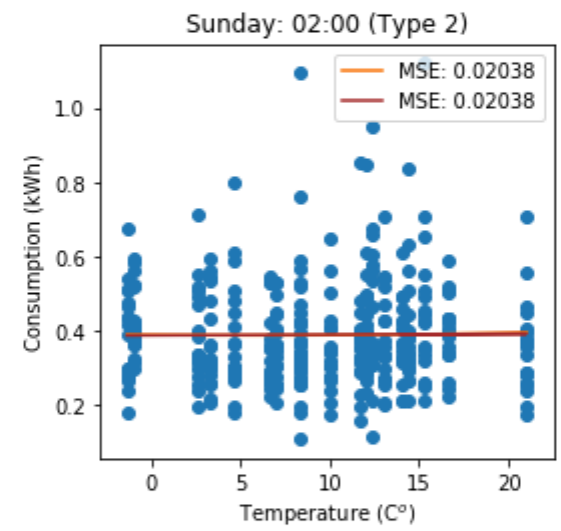

Sunday: 00:00 (Type 3)
MSE: 0.00303
MSE: 0.00304
Consumption (kWh)
Temperature (C°)

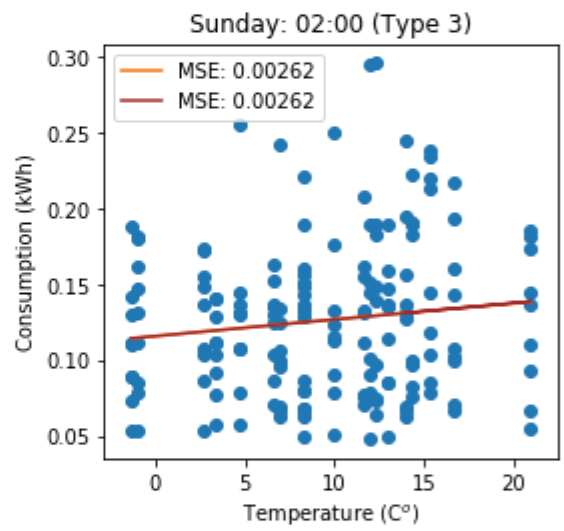

Sunday: 01:00 (Type 1)
MSE: 0.00595
MSE: 0.00595
Consumption (kWh)
Temperature (C°)

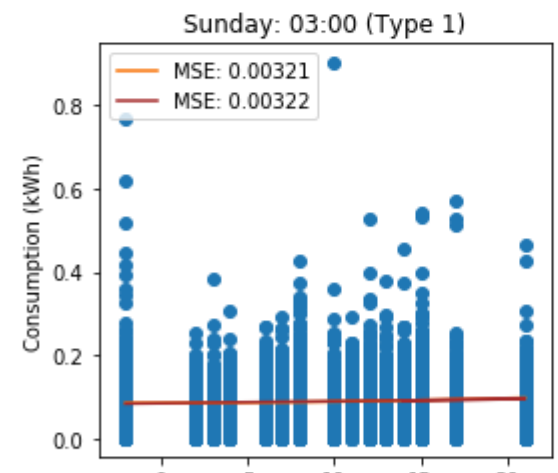

Sunday: 01:00 (Type 2)
MSE: 0.02137
MSE: 0.02141
Consumption (kWh)
Temperature (C°)

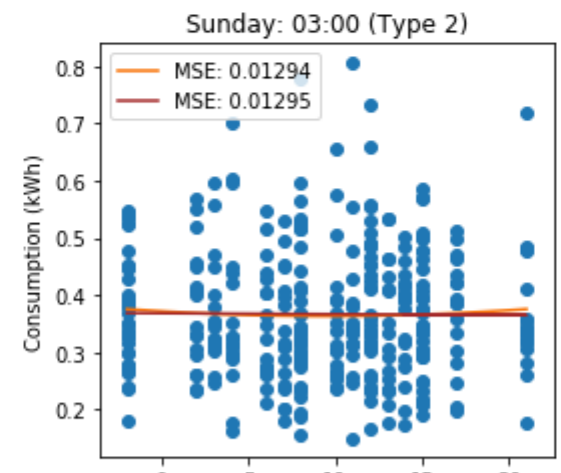

Sunday: 01:00 (Type 3)
MSE: 0.00296
MSE: 0.00302
Consumption (kWh)
Temperature (C°)

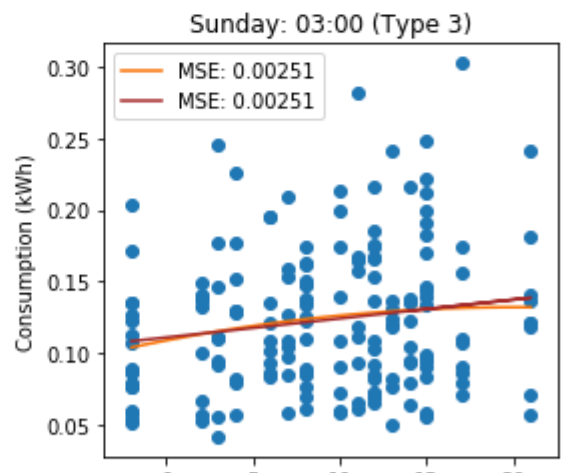

Sunday: 02:00 (Type 1)
MSE: 0.00402
MSE: 0.00402
Consumption (kWh)
Temperature (C°)

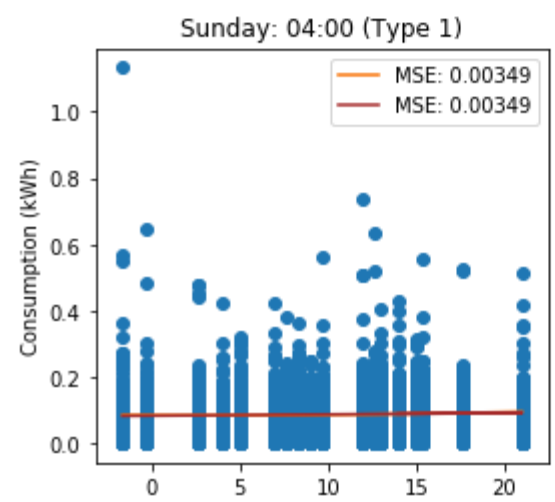

Sunday: 02:00 (Type 2)
MSE: 0.02038
MSE: 0.02038
Consumption (kWh)
Temperature (C°)

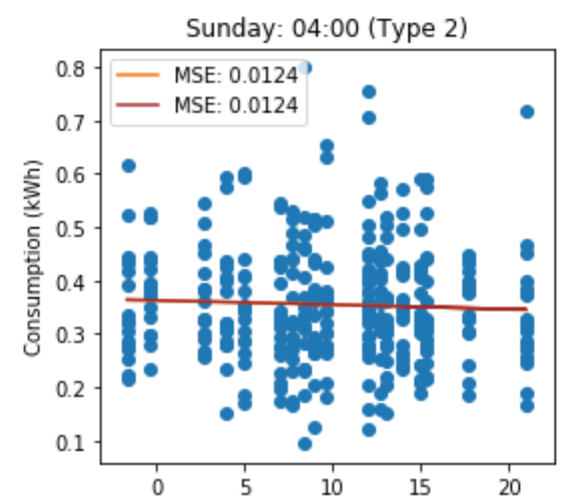

Sunday: 02:00 (Type 3)
MSE: 0.00262
MSE: 0.00262
Consumption (kWh)
Temperature (C°)

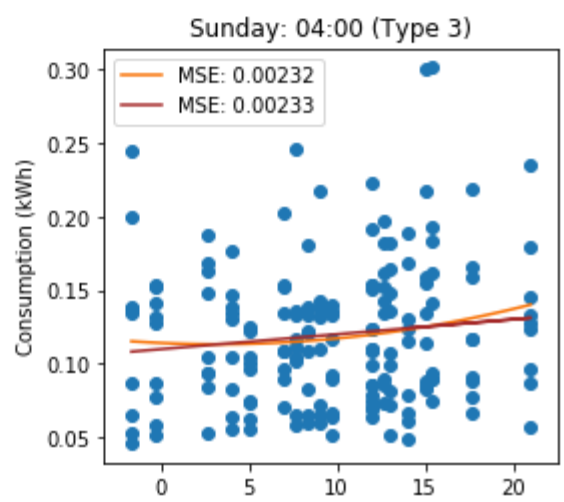

Sunday: 03:00 (Type 1)
MSE: 0.00321
MSE: 0.00322
Consumption (kWh)
Temperature (C°)

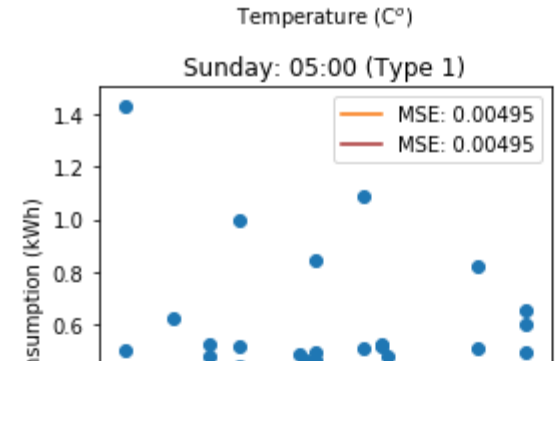

Sunday: 03:00 (Type 2)
MSE: 0.01294
MSE: 0.01295
Consumption (kWh)
Temperature (C°)

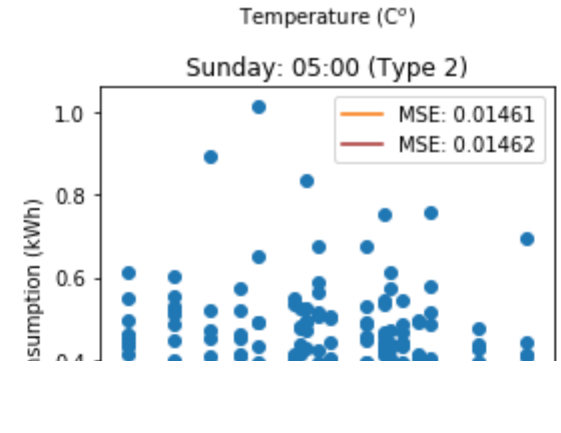

Sunday: 03:00 (Type 3)
MSE: 0.00251
MSE: 0.00251
Consumption (kWh)
Temperature (C°)

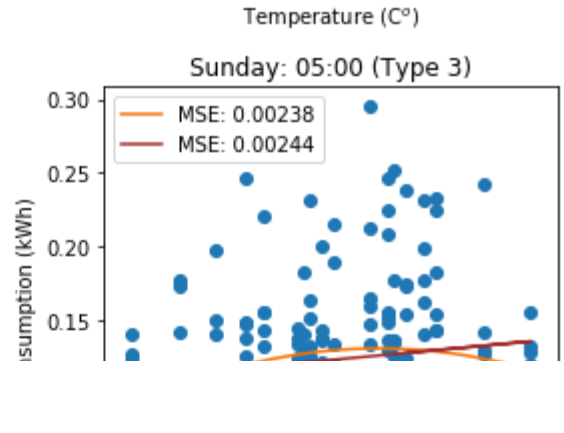

Sunday: 04:00 (Type 1)
MSE: 0.00349
MSE: 0.00349
Consumption (kWh)
Temperature (C°)

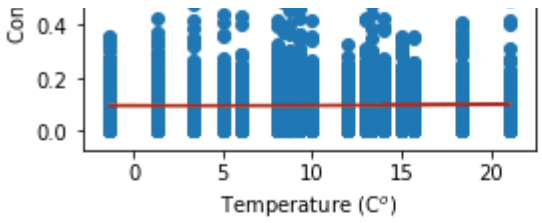

Sunday: 04:00 (Type 2)
MSE: 0.0124
MSE: 0.0124
Consumption (kWh)
Temperature (C°)

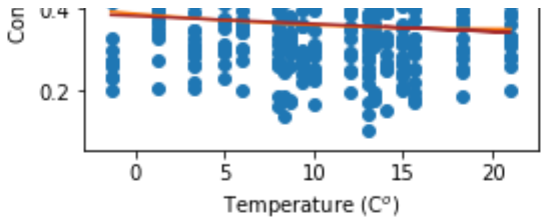

Sunday: 04:00 (Type 3)
MSE: 0.00232
MSE: 0.00233
Consumption (kWh)
Temperature (C°)

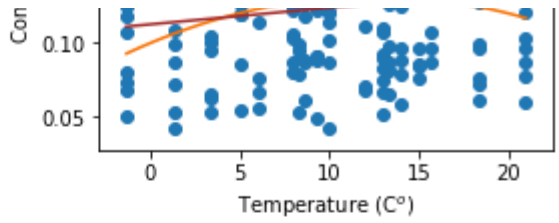

Sunday: 05:00 (Type 1)
MSE: 0.00495
MSE: 0.00495

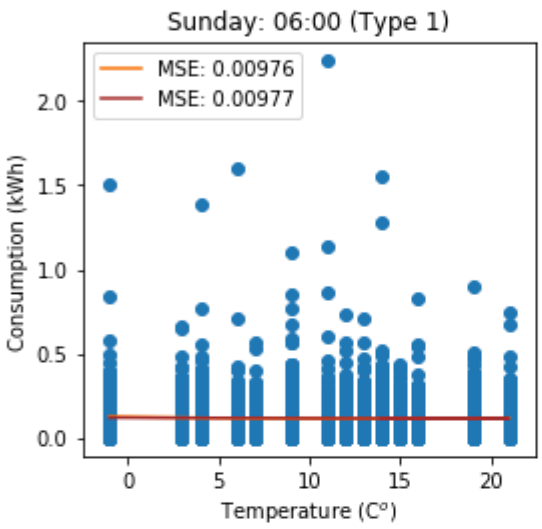

Sunday: 05:00 (Type 2)
MSE: 0.01461
MSE: 0.01462

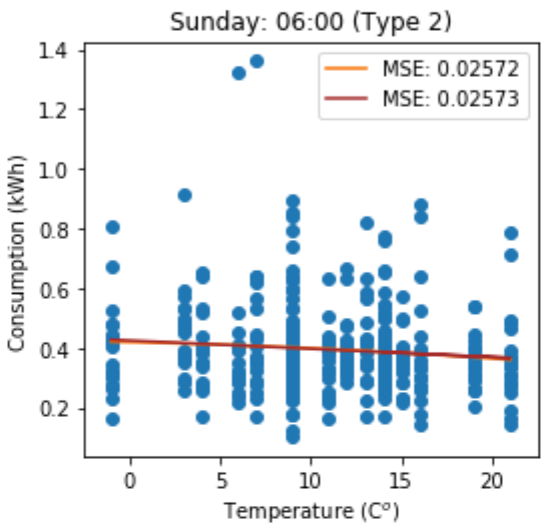

Sunday: 05:00 (Type 3)
MSE: 0.00238
MSE: 0.00244

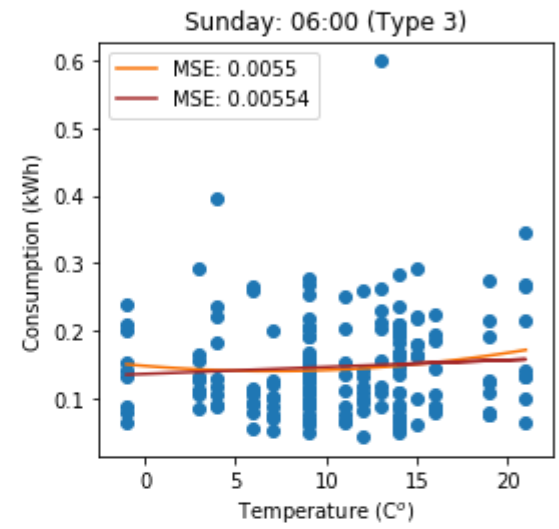

Temperature (C°)

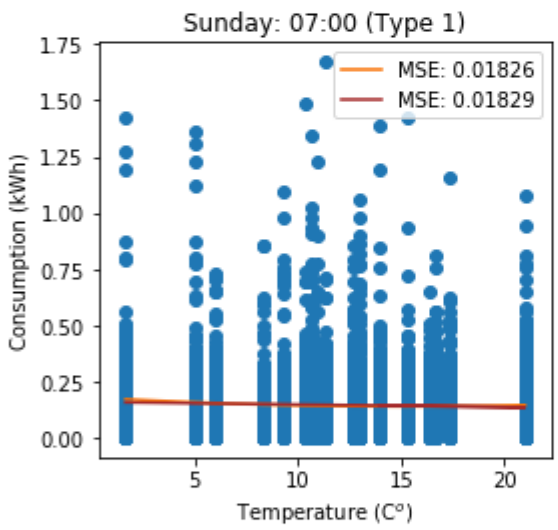

Temperature (C°)

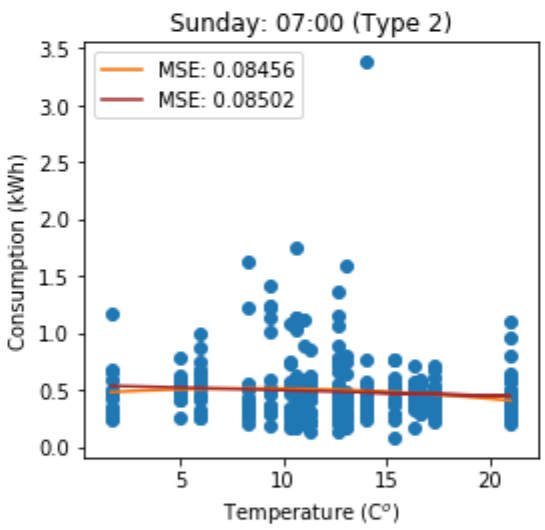

Temperature (C°)

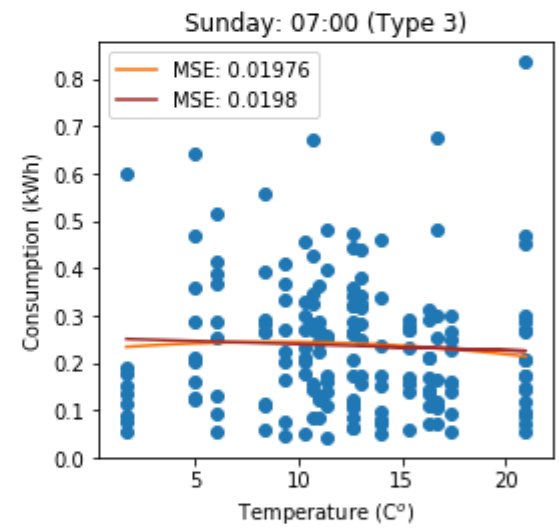

Sunday: 06:00 (Type 1)
MSE: 0.00976
MSE: 0.00977
Consumption (kWh)
Temperature (C°)

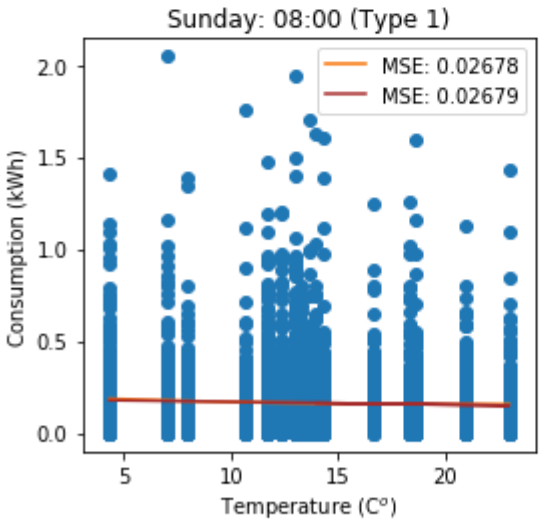

Sunday: 06:00 (Type 2)
MSE: 0.02572
MSE: 0.02573
Consumption (kWh)
Temperature (C°)

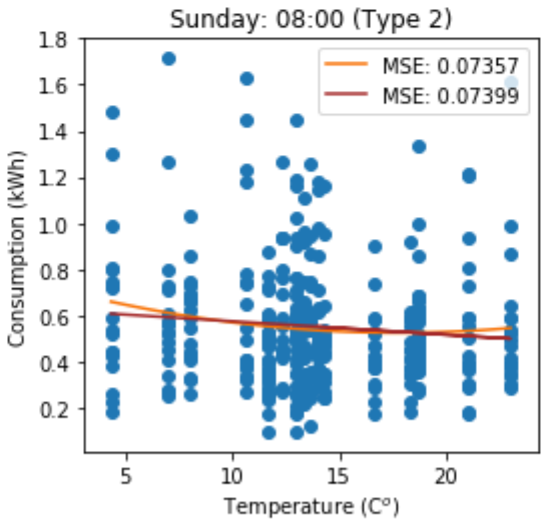

Sunday: 06:00 (Type 3)
MSE: 0.0055
MSE: 0.00554
Consumption (kWh)
Temperature (C°)

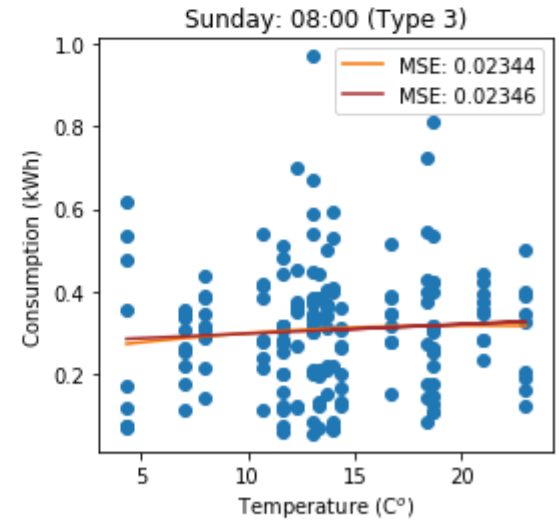

Sunday: 07:00 (Type 1)
MSE: 0.01826
MSE: 0.01829
Consumption (kWh)
Temperature (C°)

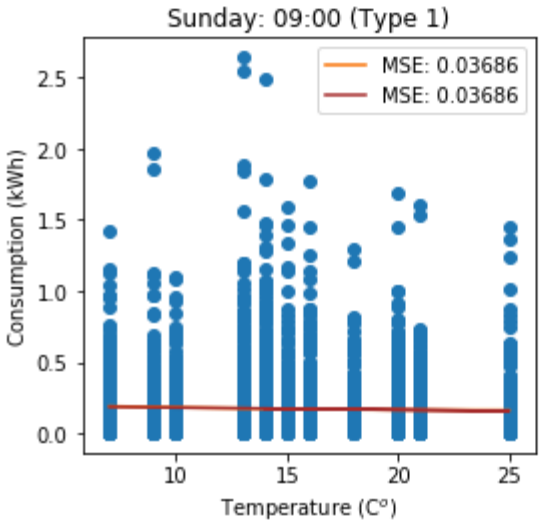

Sunday: 07:00 (Type 2)
MSE: 0.08456
MSE: 0.08502
Consumption (kWh)
Temperature (C°)

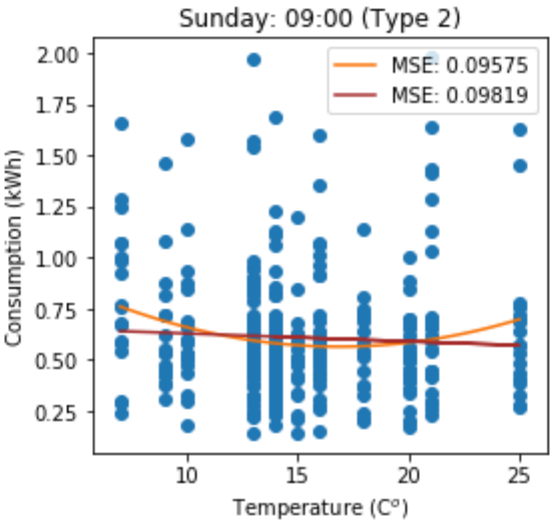

Sunday: 07:00 (Type 3)
MSE: 0.01976
MSE: 0.0198
Consumption (kWh)
Temperature (C°)

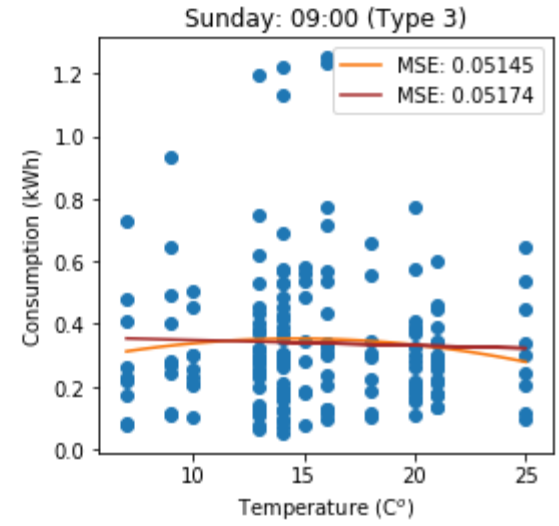

Sunday: 08:00 (Type 1)
MSE: 0.02678
MSE: 0.02679
Consumption (kWh)
Temperature (C°)

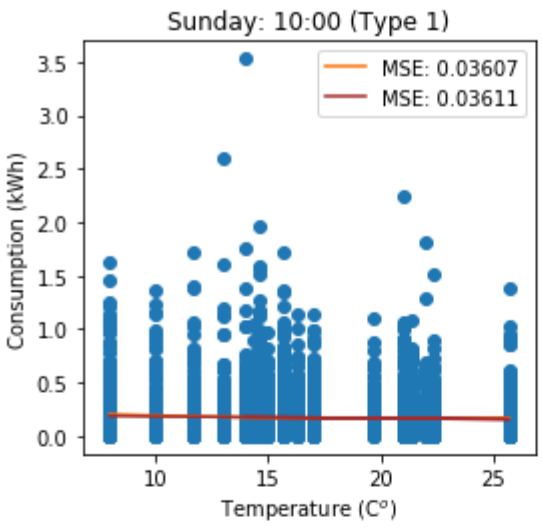

Sunday: 08:00 (Type 2)
MSE: 0.07357
MSE: 0.07399
Consumption (kWh)
Temperature (C°)

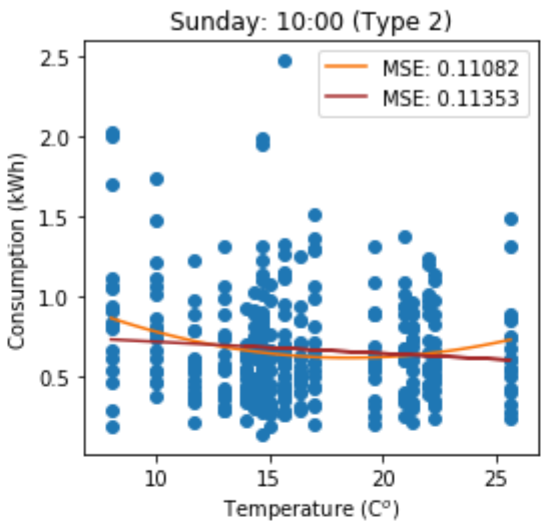

Sunday: 08:00 (Type 3)
MSE: 0.02344
MSE: 0.02346
Consumption (kWh)
Temperature (C°)

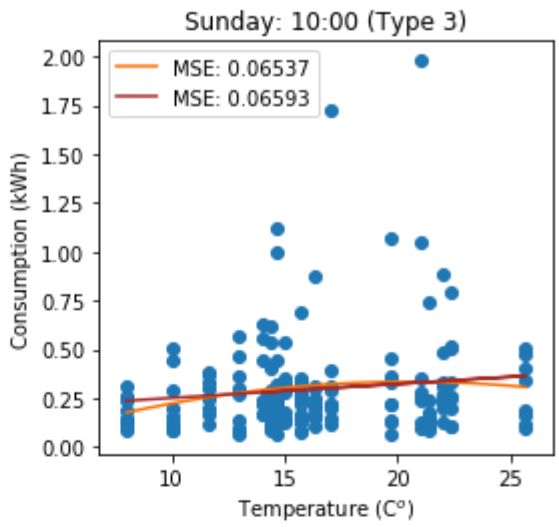

Sunday: 09:00 (Type 1)
MSE: 0.03686
MSE: 0.03686
Consumption (kWh)
Temperature (C°)


Sunday: 09:00 (Type 2)
MSE: 0.09575
MSE: 0.09819
Consumption (kWh)
Temperature (C°)


Sunday: 09:00 (Type 3)
MSE: 0.05145
MSE: 0.05174
Consumption (kWh)
Temperature (C°)

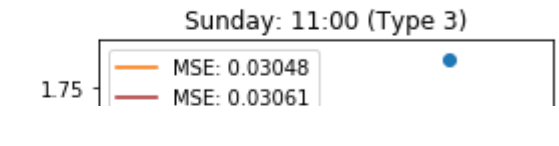

Sunday: 10:00 (Type 1)
MSE: 0.03607
MSE: 0.03611
Consumption (kWh)
Temperature (C°)

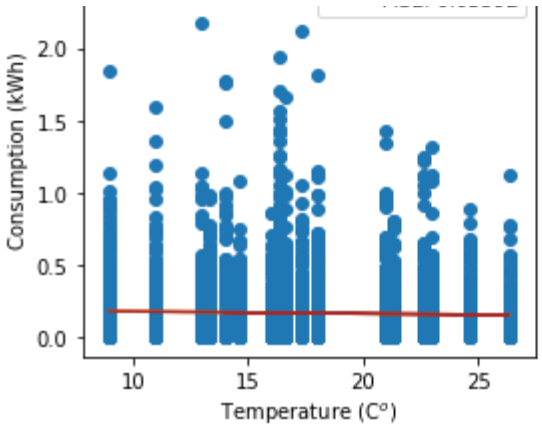

Sunday: 10:00 (Type 2)
MSE: 0.11082
MSE: 0.11353
Consumption (kWh)
Temperature (C°)

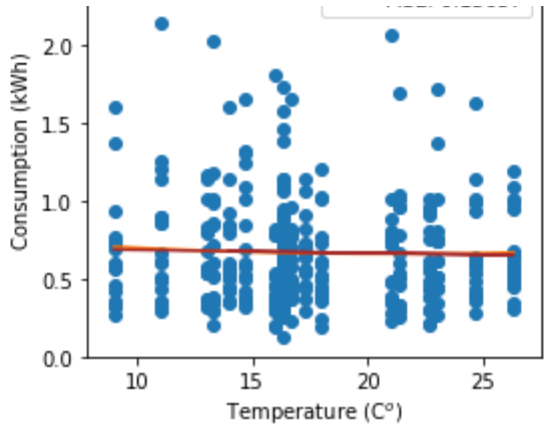

Sunday: 10:00 (Type 3)
MSE: 0.06537
MSE: 0.06593
Consumption (kWh)
Temperature (C°)

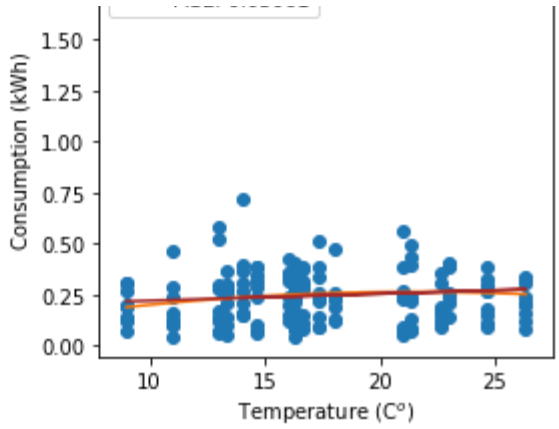

Sunday: 11:00 (Type 1)
MSE: 0.03592
MSE: 0.03592

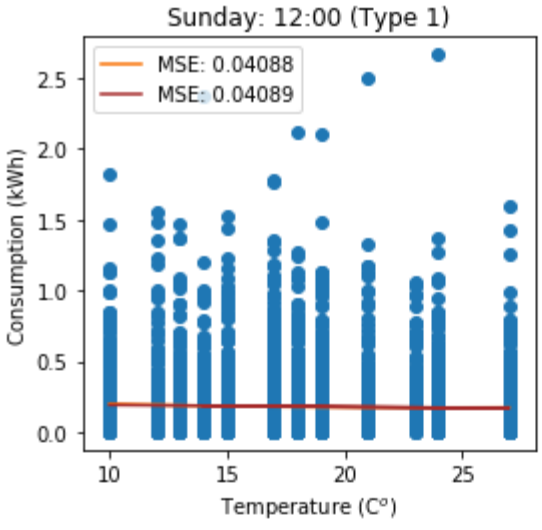

Sunday: 11:00 (Type 2)
MSE: 0.13652
MSE: 0.13657

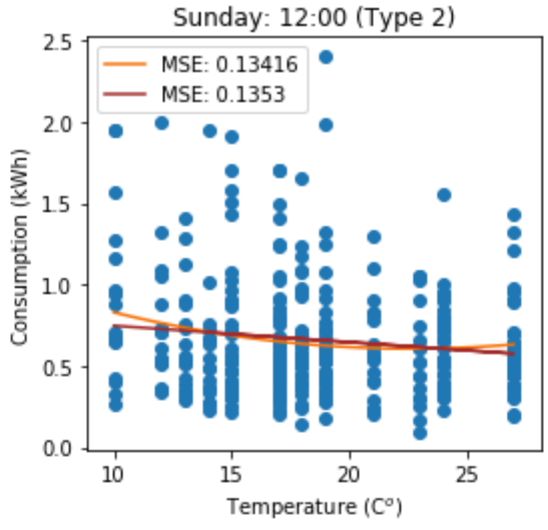

Sunday: 11:00 (Type 3)
MSE: 0.03048
MSE: 0.03061

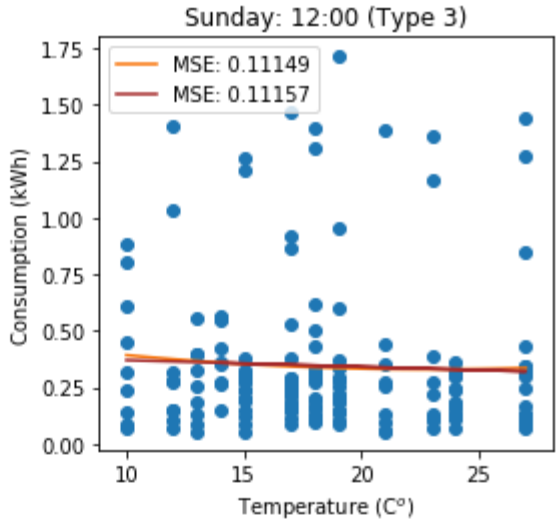

Consumption (kWh)
Temperature (C°)

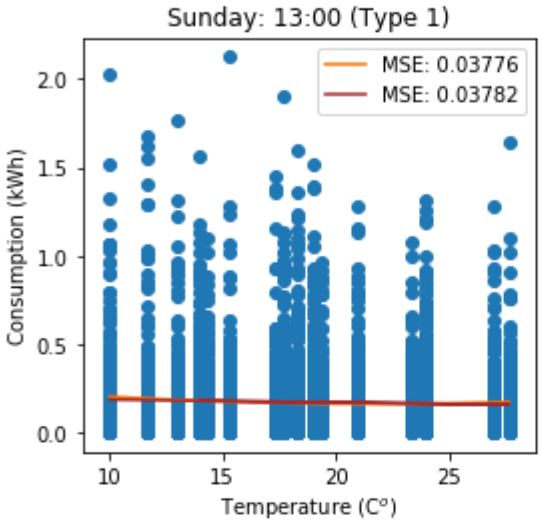

Consumption (kWh)
Temperature (C°)

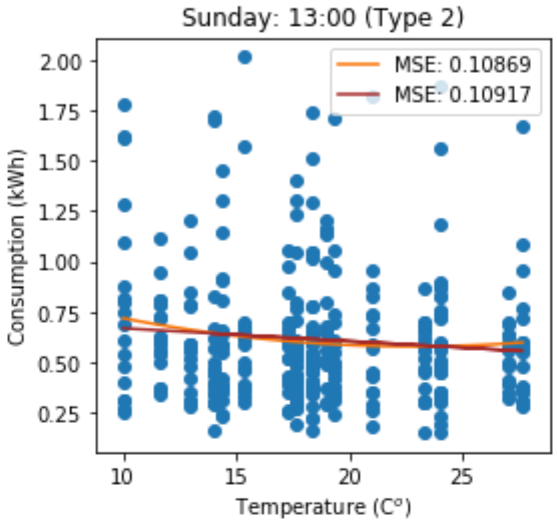

Consumption (kWh)
Temperature (C°)

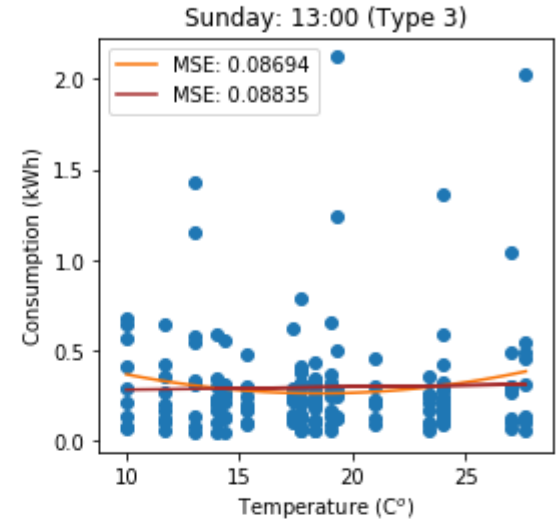

Sunday: 12:00 (Type 1)
MSE: 0.04088
MSE: 0.04089
Consumption (kWh)
Temperature (C°)

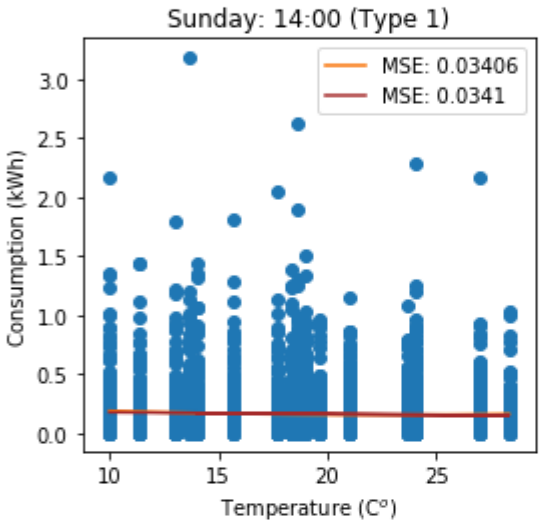

Sunday: 12:00 (Type 2)
MSE: 0.13416
MSE: 0.1353
Consumption (kWh)
Temperature (C°)

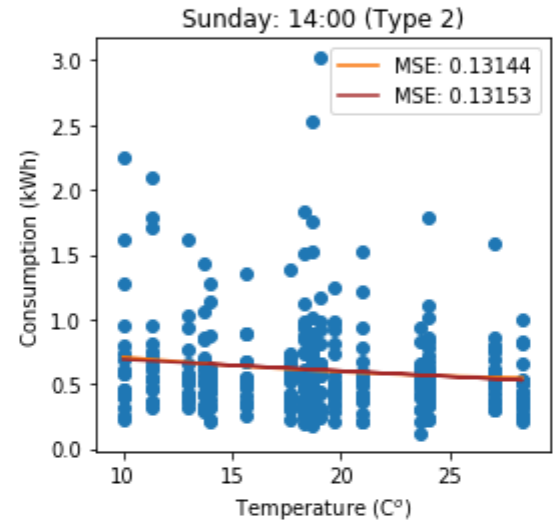

Sunday: 12:00 (Type 3)
MSE: 0.11149
MSE: 0.11157
Consumption (kWh)
Temperature (C°)

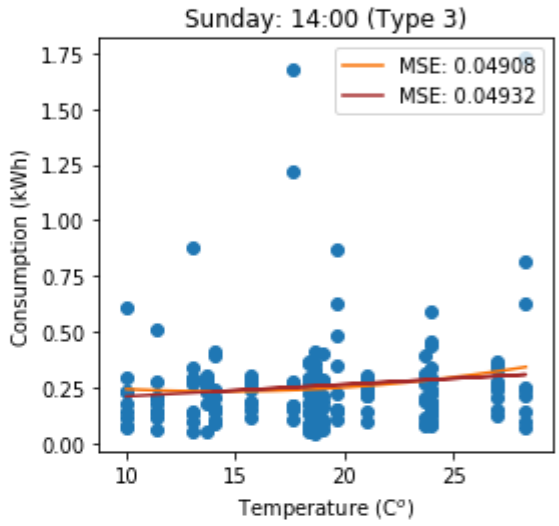

Sunday: 13:00 (Type 1)
MSE: 0.03776
MSE: 0.03782
Consumption (kWh)
Temperature (C°)

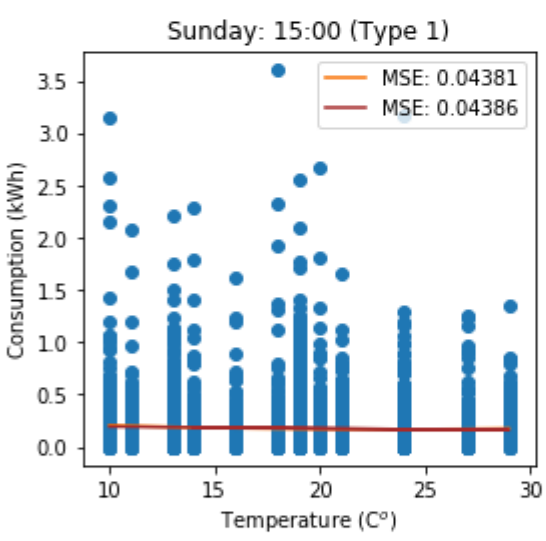

Sunday: 13:00 (Type 2)
MSE: 0.10869
MSE: 0.10917
Consumption (kWh)
Temperature (C°)

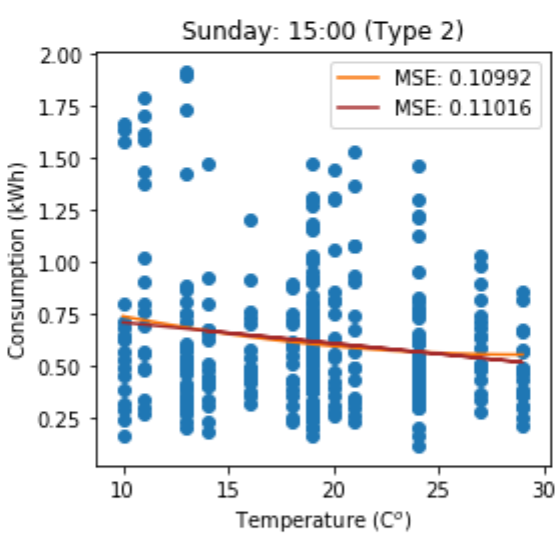

Sunday: 13:00 (Type 3)
MSE: 0.08694
MSE: 0.08835
Consumption (kWh)
Temperature (C°)

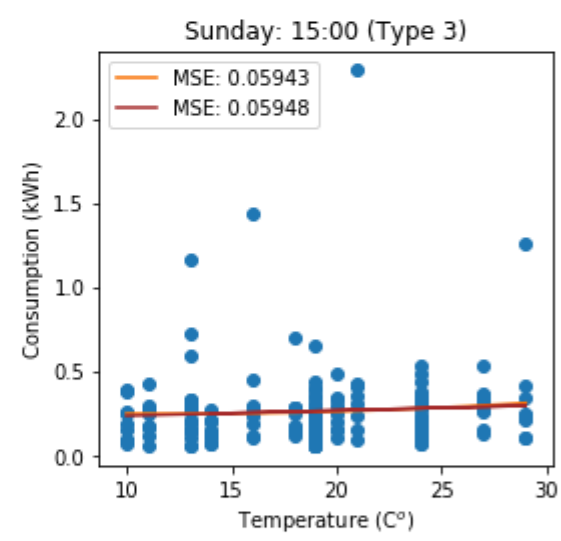

Sunday: 14:00 (Type 1)
MSE: 0.03406
MSE: 0.0341
Consumption (kWh)
Temperature (C°)

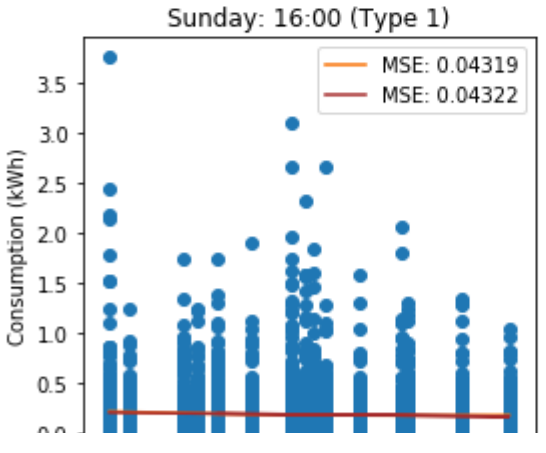

Sunday: 14:00 (Type 2)
MSE: 0.13144
MSE: 0.13153
Consumption (kWh)
Temperature (C°)

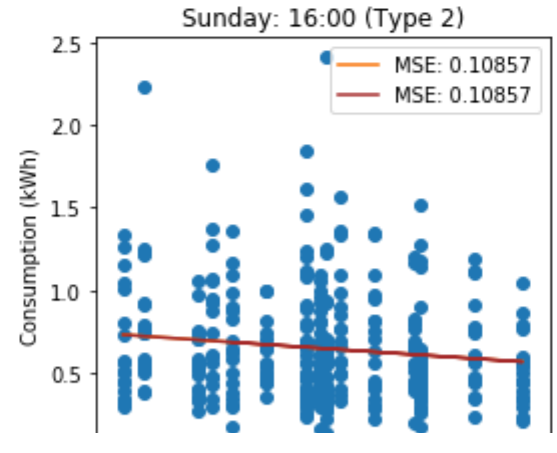

Sunday: 14:00 (Type 3)
MSE: 0.04908
MSE: 0.04932
Consumption (kWh)
Temperature (C°)

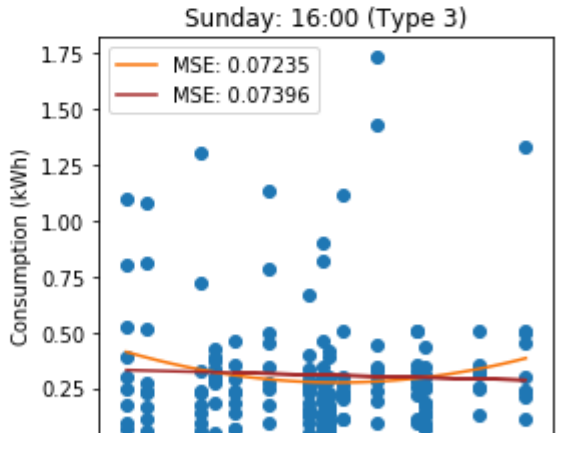

Sunday: 15:00 (Type 1)
MSE: 0.04381
MSE: 0.04386
Consumption (kWh)
Temperature (C°)

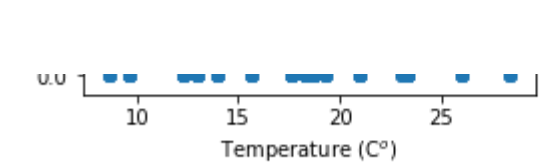

Sunday: 15:00 (Type 2)
MSE: 0.10992
MSE: 0.11016
Consumption (kWh)
Temperature (C°)

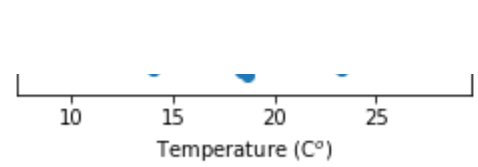

Sunday: 15:00 (Type 3)
MSE: 0.05943
MSE: 0.05948
Consumption (kWh)
Temperature (C°)

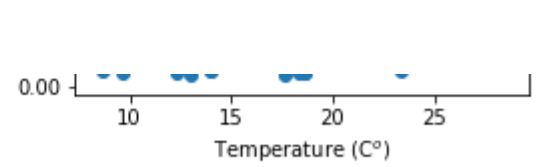

Sunday: 16:00 (Type 1)
MSE: 0.04319
MSE: 0.04322
Consumption (kWh)

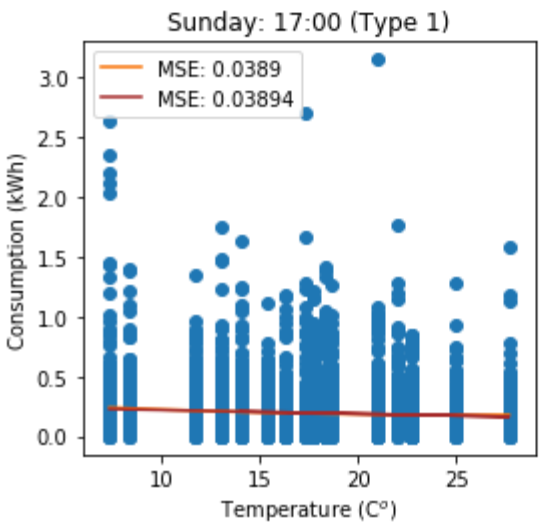

Sunday: 16:00 (Type 2)
MSE: 0.10857
MSE: 0.10857
Consumption (kWh)

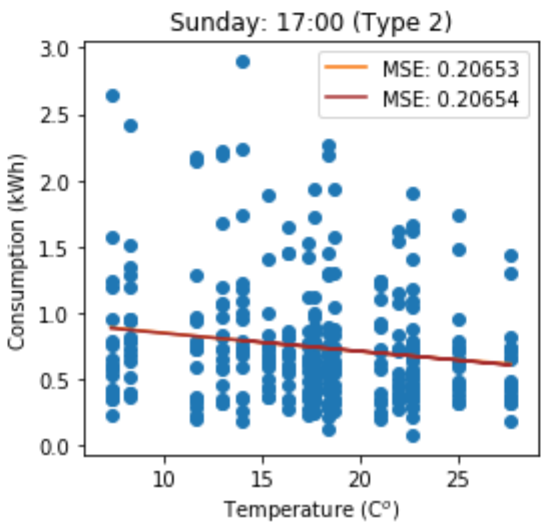

Sunday: 16:00 (Type 3)
MSE: 0.07235
MSE: 0.07396
Consumption (kWh)

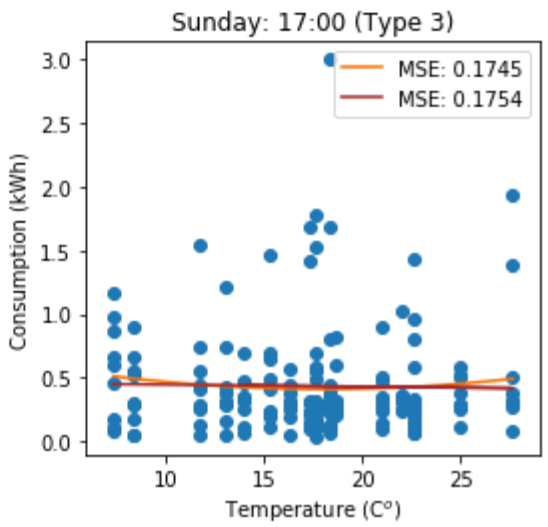

Temperature (C°)

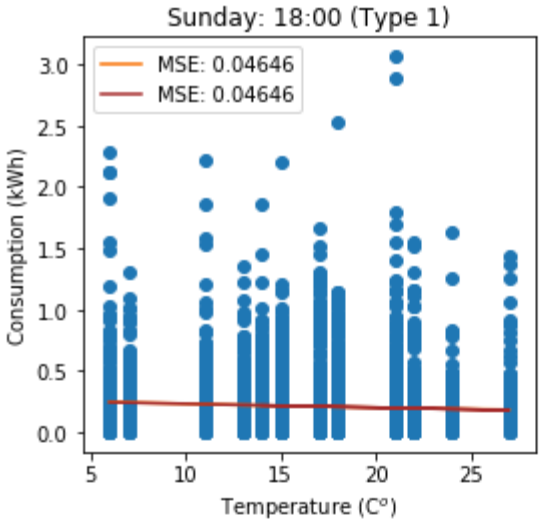

Temperature (C°)

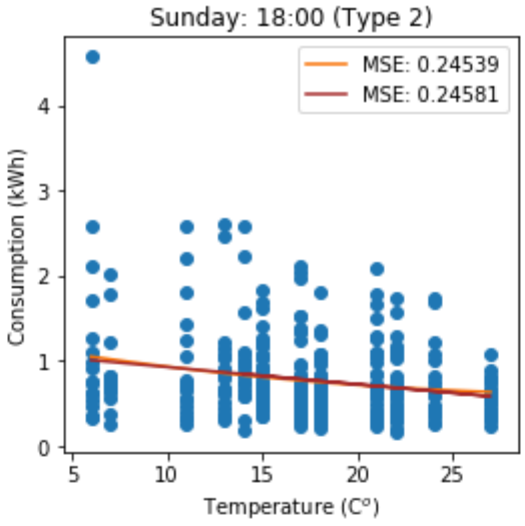

Temperature (C°)

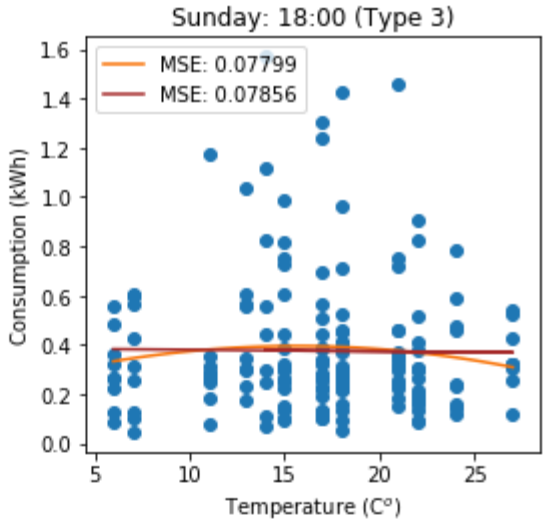

Sunday: 17:00 (Type 1)
MSE: 0.0389
MSE: 0.03894
Consumption (kWh)
Temperature (C°)

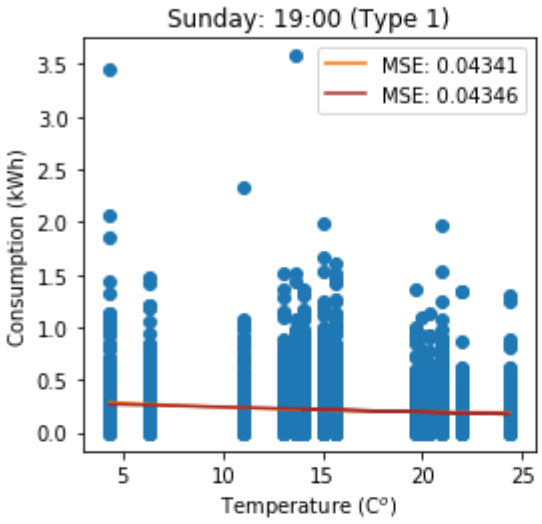

Sunday: 17:00 (Type 2)
MSE: 0.20653
MSE: 0.20654
Consumption (kWh)
Temperature (C°)

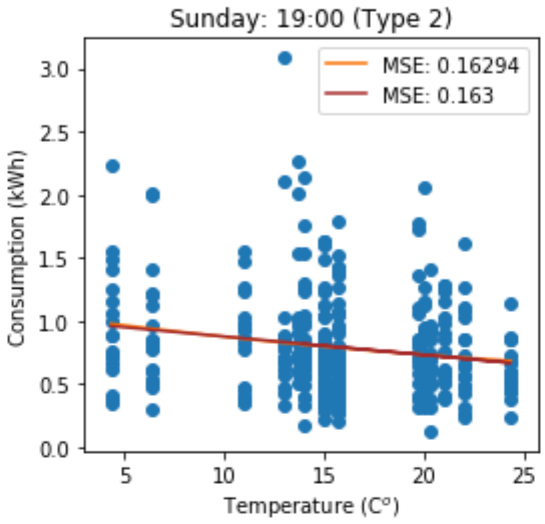

Sunday: 17:00 (Type 3)
MSE: 0.1745
MSE: 0.1754
Consumption (kWh)
Temperature (C°)

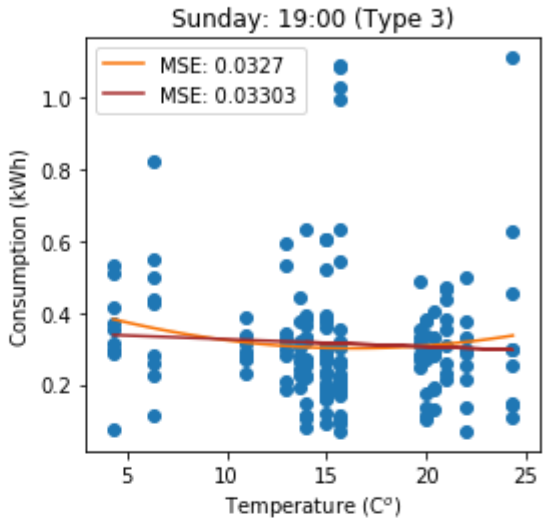

Sunday: 18:00 (Type 1)
MSE: 0.04646
MSE: 0.04646
Consumption (kWh)
Temperature (C°)

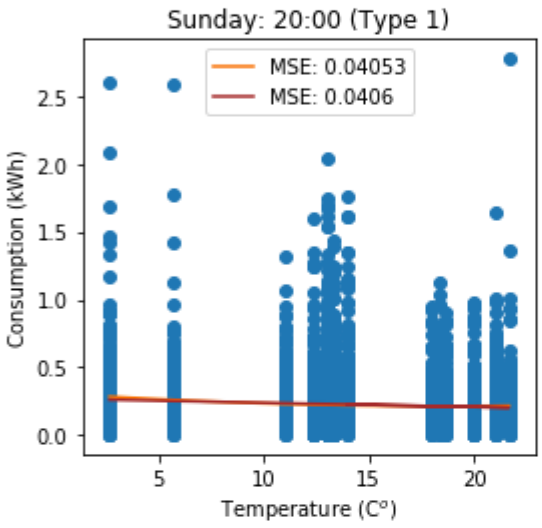

Sunday: 18:00 (Type 2)
MSE: 0.24539
MSE: 0.24581
Consumption (kWh)
Temperature (C°)

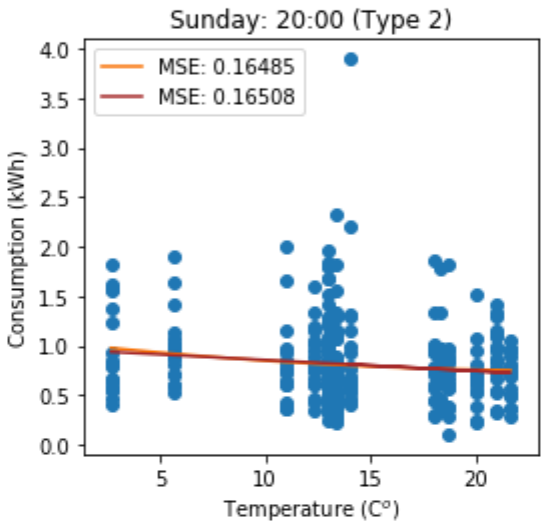

Sunday: 18:00 (Type 3)
MSE: 0.07799
MSE: 0.07856
Consumption (kWh)
Temperature (C°)

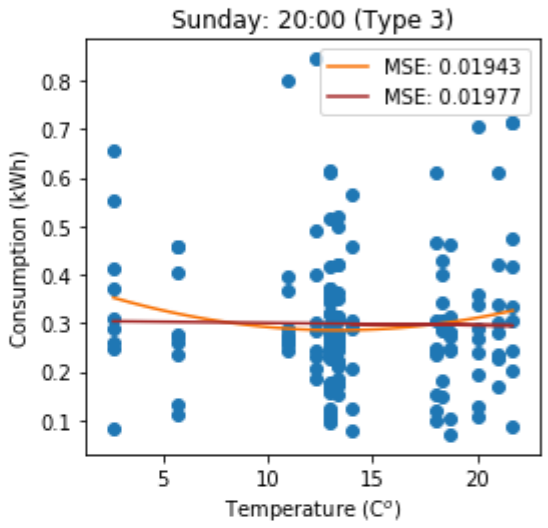

Sunday: 19:00 (Type 1)
MSE: 0.04341
MSE: 0.04346
Consumption (kWh)
Temperature (C°)

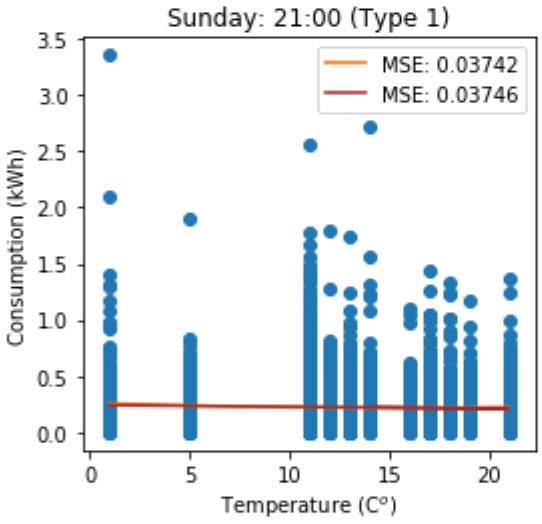

Sunday: 19:00 (Type 2)
MSE: 0.16294
MSE: 0.163
Consumption (kWh)
Temperature (C°)

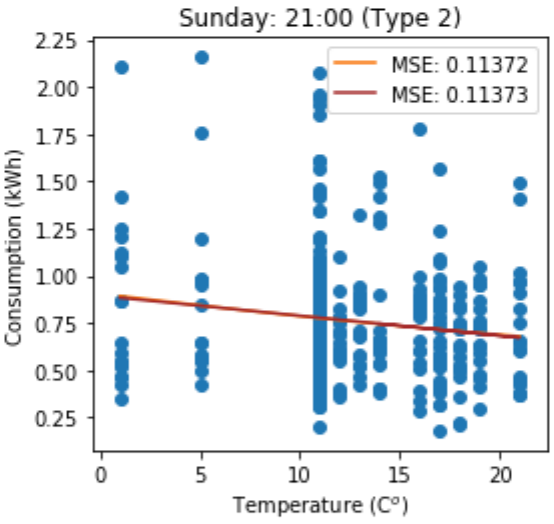

Sunday: 19:00 (Type 3)
MSE: 0.0327
MSE: 0.03303
Consumption (kWh)
Temperature (C°)

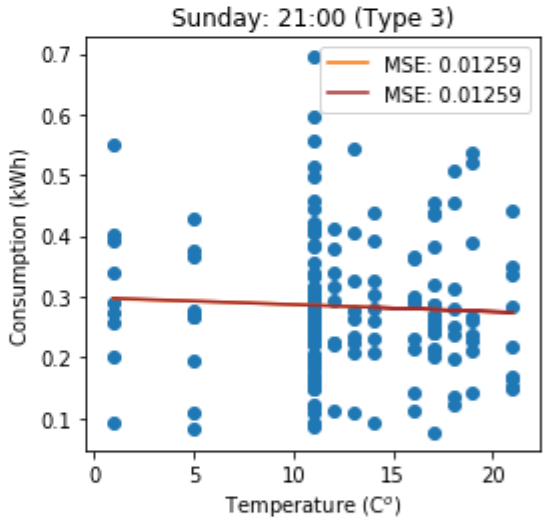

Sunday: 20:00 (Type 1)
MSE: 0.04053
MSE: 0.0406
Consumption (kWh)
Temperature (C°)


Sunday: 20:00 (Type 2)
MSE: 0.16485
MSE: 0.16508
Consumption (kWh)
Temperature (C°)


Sunday: 20:00 (Type 3)
MSE: 0.01943
MSE: 0.01977
Consumption (kWh)
Temperature (C°)


Sunday: 21:00 (Type 1)
MSE: 0.03742
MSE: 0.03746
Consumption (kWh)
Temperature (C°)

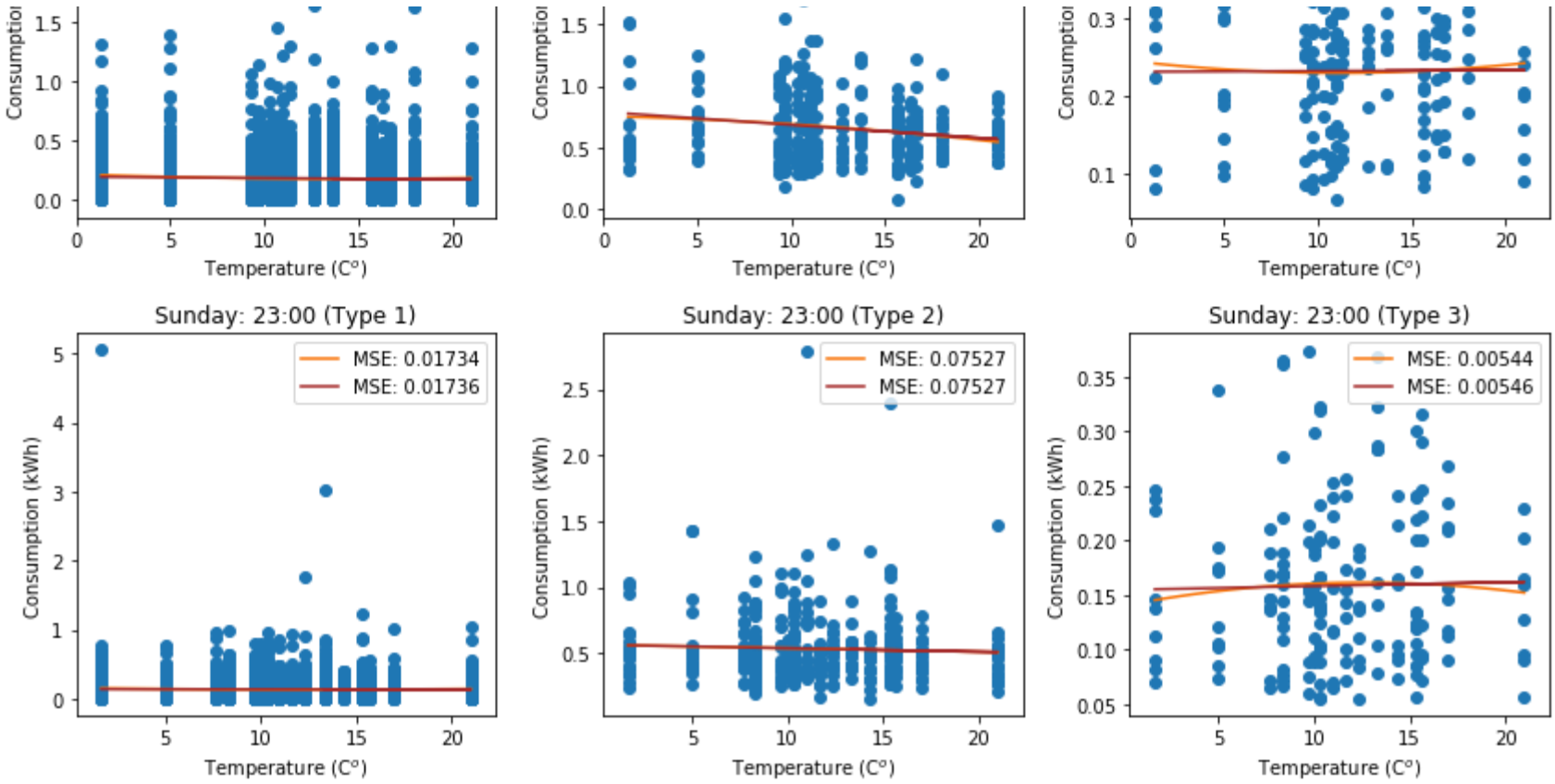

Sunday: 21:00 (Type 2)
MSE: 0.11372
MSE: 0.11373
Consumption (kWh)
Temperature (C°)


Sunday: 21:00 (Type 3)
MSE: 0.01259
MSE: 0.01259
Consumption (kWh)
Temperature (C°)

Sunday: 22:00 (Type 1)
MSE: 0.02613
MSE: 0.02616
Temperature (C°)

Sunday: 22:00 (Type 2)
MSE: 0.10039
MSE: 0.10048
Temperature (C°)

Sunday: 22:00 (Type 3)
MSE: 0.00786
MSE: 0.00787
Temperature (C°)



4. Groups' Price Responsiveness - K-mean based

The way we analyze the group price responsiveness has three steps:

1) for each hosehold group, build a new dataframe, the first column is GMT, the next columns are user ids in the group, the next column is temperature, then the price, day of a week, hour of a day, then the predicted consumption, and last the price responsiveness

2) build a function that can generate predicted consumption based on the day of a week, temp, hour to choose the coefficients and then make a prediction.

3) plot the properties based on from the smallest unit (day of week, hour, temp) to (day of week, hour), (day of week, temp), (hour, temp), (day of week), (hour), (temp) so we will have very comprehensive price responsiveness of three different groups

In [7]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
n_house = 1025 # total household numbers
houseLabel = [] # store the 24hour group labels for each hourshold
for i in range(1025):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and fill in null value with mean
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
    else:
        x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
# We obtain the house Label shown below
# print(houseLabel)

# Group the house based on their 24 hour labels
houseLableDf = pd.DataFrame()
houseLableDf['House'] = df_tou1h_nf_warm.columns[1:]
houseLableDf['Label'] = houseLabel
houseGroupDf = houseLableDf.groupby('Label').size().reset_index(name='counts')
houseGroupDf = houseGroupDf.sort_values(by=['counts'], ascending=False)
houseGroupDf
```

Out[7]:

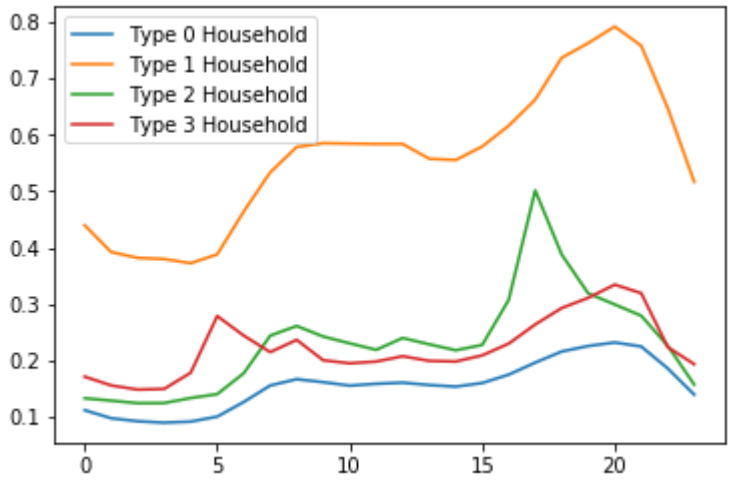
	Label	counts
129	00000120000000120012000	314
508	12112002122212001120012 2	19
140	00000120000000120101200 0	9
36	00000020000000120012000	8
399	12000120000000120012002	6
147	00000120000000121001200 0	6
126	00000120000000120000200 0	6
231	00000122100000120012000	5
149	00000120000000121101200 0	5
137	00000120000000120021200 0	4
575	21221211211121212212121 1	4
125	00000120000000120000000 0	4
365	12000002122212001120012 2	4
131	00000120000000120012020	4
127	00000120000000120001000 0	4
397	12000120000000120000012 2	4
209	00000120120000120012000	4
202	00000120100000120012000	3
1	00000000000000120012000	3
134	00000120000000120020000 0	3
121	00000120000000101101200 0	3
28	00000002122212001120000 0	3
389	12000102122212001120012 2	3
163	00000120000010120012000	3
168	00000120000012001120012 2	3
483	12112000122212001120012 2	2
117	00000120000000020012000	2
364	12000002122212001120012 0	2
115	00000120000000001120000 0	2
402	12000120000000120121202 2	2
...	...	...
236	00000122100210121001200 0	1
235	00000122100010120120012 2	1
233	00000122100000120120012 0	1
232	00000122100000120012022	1
230	00000122002010101120012 2	1
229	00000122000012002220012 2	1
228	00000122000000121120000 0	1
227	00000122000000120121200 0	1
226	00000122000000120001200 0	1
225	00000122000000120000012 0	1
224	00000122000000101120002 0	1
223	00000121200000120001200 0	1
222	00000121120010100120002 2	1
219	00000120122010120000000 0	1
203	00000120100000120120000 0	1
218	00000120122002001120002 0	1
217	00000120122002000020202 0	1
216	00000120122000120001200 0	1
215	00000120120200121101200 0	1

	Label	counts
214	000001201200120011012000	1
213	000001201200101200202000	1
212	000001201200020001200120	1
211	000001201200001201012000	1
210	000001201200001200200120	1
208	000001201200000010012000	1
207	000001201000120011202000	1
206	000001201000120010012000	1
205	000001201000020011012000	1
204	000001201000001211012000	1
583	221120021222120111121111	1

584 rows × 2 columns

```
<Figure size 936x6480 with 0 Axes>
```

In [8]:
```
for i in range(4):
    load = []
    for j in range(24):
        if j <= 9:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
        else:
            x = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
    plt.plot(load, label = 'Type ' + str(i) + ' Household')
plt.legend()
plt.show()
```



In [13]:
```
# Store warm season houshold labels
house_list_type0 = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[0]].House)
house_list_type1 = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[1]].House)
house_list_type2 = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[2]].House)
```

In [57]:
```
# Type 0 household
# build a new dataframe with columns: GMT, user ids in the group
df_type0 = df_tou1h_event_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[0]].
House)]
df_type0 = pd.merge(df_type0, df_wea1h_event_warm, on = 'GMT') # add weather
df_type0 = pd.merge(df_type0, df_tariff_1h_event_warm, on = 'GMT') # add price
df_type0['Day of week'] = pd.to_datetime(df_type0.GMT).dt.dayofweek # add day of week
df_type0['Hour of day'] = pd.to_datetime(df_type0.GMT).dt.hour # add hour of day
df_type0['Household type'] = 0

# Type 1 household
# build a new dataframe with columns: GMT, user ids in the group
df_type1 = df_tou1h_event_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[1]].
House)]
df_type1 = pd.merge(df_type1, df_wea1h_event_warm, on = 'GMT') # add weather
df_type1 = pd.merge(df_type1, df_tariff_1h_event_warm, on = 'GMT') # add price
df_type1['Day of week'] = pd.to_datetime(df_type1.GMT).dt.dayofweek # add day of week
df_type1['Hour of day'] = pd.to_datetime(df_type1.GMT).dt.hour # add hour of day
df_type1['Household type'] = 1

# Type 2 household
# build a new dataframe with columns: GMT, user ids in the group
df_type2 = df_tou1h_event_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[2]].
House)]
df_type2 = pd.merge(df_type2, df_wea1h_event_warm, on = 'GMT') # add weather
df_type2 = pd.merge(df_type2, df_tariff_1h_event_warm, on = 'GMT') # add price
df_type2['Day of week'] = pd.to_datetime(df_type2.GMT).dt.dayofweek # add day of week
df_type2['Hour of day'] = pd.to_datetime(df_type2.GMT).dt.hour # add hour of day
df_type2['Household type'] = 2
```

With the household type and feature data ready, we need to build functions that can predict the consumptions during these time periods. Since we are considering group properties, the above users are all in the first group, for each time period, we only have one consumption predicted for the whole group, but since the regression coefficients depend on day of week, temperature and hour of day, so it should be quite different for different time.

In [58]:
```
def predict(dfRegression, dayOfWeek, hour, temperature):
    '''
    Predict the consumption duruing event time based on
    the regression, day of week, hour of a day and temperature
    '''
    weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
    output = dfRegression[weeks[dayOfWeek]].iloc[hour](temperature)
    return output
```

In [59]:
```
# store the regression coefficients
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df_regression_0 = {} # store the regression coefficients
df_regression_1 = {}
df_regression_2 = {}
# df_regression_3 = {}
lm = LinearRegression()
for g in range(3): # doing this for three groups
    for day_of_week in range(7):
        coef_temp = []
        for i in range(24):
            if i <= 9:
                z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]
                y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].H
ouse)]
                y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Mond
ay
                y.set_index('GMT', inplace = True)
                y = y.fillna(y.mean()).values
                y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable
                x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains('0' + str(i) + ':00:00')) & (pd.to_date
time(z.GMT).dt.dayofweek == day_of_week)].values
                x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct th
e x variable
                z = np.polyfit(x, y, 2)
                p = np.poly1d(z)
                coef_temp.append(p)
            else:
                z = df_tou1h_nf_warm[df_tou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]
                y_temp = z[['GMT'] + list(houseLableDf[(houseLableDf.Label == houseGroupDf.Label.iloc[g]) ].H
ouse)]
                y = y_temp[pd.to_datetime(z.GMT).dt.dayofweek == day_of_week] # find the users' data for Mond
ay
                y.set_index('GMT', inplace = True)
                y = y.fillna(y.mean()).values
                y = np.reshape(y, y.shape[0] * y.shape[1], order = 'F') # reconstruct the y variable

                x = df_wea1h_nf_warm.TempC[(df_wea1h.GMT.str.contains(str(i) + ':00:00')) & (pd.to_datetime(z
.GMT).dt.dayofweek == day_of_week)].values
                x = np.tile(x, reps = sum(houseLableDf.Label == houseGroupDf.Label.iloc[g])) # reconstruct th
e x variable
                z = np.polyfit(x, y, 2)
                p = np.poly1d(z)
                coef_temp.append(p)
        if g == 0:
            df_regression_0[weeks[day_of_week]] = coef_temp
        elif g == 1:
            df_regression_1[weeks[day_of_week]] = coef_temp
        else:
            df_regression_2[weeks[day_of_week]] = coef_temp
df_regression_0 = pd.DataFrame(df_regression_0) # type 0 regression coefficient
df_regression_1 = pd.DataFrame(df_regression_1) # type 1
df_regression_2 = pd.DataFrame(df_regression_2) # type 2
```

In [49]:
```
# type 0 event time consumption prediction
pred_result = []
for i in range(df_type0.shape[0]):
    pred_result.append(predict(df_regression_0, df_type0['Day of week'].iloc[i], df_type0['Hour of day'].iloc
[i], df_type0['TempC'].iloc[i]))
df_type0['Predicted consumption'] = pred_result
# price responsiveness elementwise
df_type0_res = df_type0.copy()
df_type0_res.iloc[:,1:houseGroupDf.counts.iloc[0] + 1] = df_type0.iloc[:,1:houseGroupDf.counts.iloc[0] + 1].s
ub(df_type0['Predicted consumption'], axis = 0)

# type 1 event time consumption prediction
pred_result = []
for i in range(df_type1.shape[0]):
    pred_result.append(predict(df_regression_1, df_type1['Day of week'].iloc[i], df_type1['Hour of day'].iloc
[i], df_type1['TempC'].iloc[i]))
df_type1['Predicted consumption'] = pred_result

# price responsiveness elementwise
df_type1_res = df_type1.copy()
df_type1_res.iloc[:,1:houseGroupDf.counts.iloc[1] + 1] = df_type1.iloc[:,1:houseGroupDf.counts.iloc[1] + 1].s
ub(df_type1['Predicted consumption'], axis = 0)

# type 2 event time consumption prediction
pred_result = []
for i in range(df_type2.shape[0]):
    pred_result.append(predict(df_regression_2, df_type2['Day of week'].iloc[i], df_type2['Hour of day'].iloc
[i], df_type2['TempC'].iloc[i]))
df_type2['Predicted consumption'] = pred_result

# price responsiveness elementwise
df_type2_res = df_type2.copy()
df_type2_res.iloc[:,1:houseGroupDf.counts.iloc[2] + 1] = df_type2.iloc[:,1:houseGroupDf.counts.iloc[2] + 1].s
ub(df_type2['Predicted consumption'], axis = 0)
```

To visualize the price responsiveness, we need to reformat the data and melt them to a long dataframe, and all the user ID columns will be extracted to form a individual dataframe with adding household group label and we concatenate all of them in a long dataframe.

In [50]:
```
houseGroupDf.counts.iloc[0]
```

Out[50]:
```
318
```

In [59]:
```
df_type0_res.head()
```

Out[59]:

	GMT	D0001	D0008	D0010	D0012	D0013	D0016	D0018	D0021	D0024	...	D1023	D1024	
0	2013-04-05 19:00:00	0.059165	0.309165	-0.210835	0.185165	0.742165	0.214165	-0.165835	-0.176835	0.022165	...	0.004165	0.250165	2
1	2013-04-05 20:00:00	0.060997	0.144997	-0.245003	-0.055003	0.108997	0.189997	-0.205003	-0.207003	-0.004003	...	-0.046003	-0.031003	(
2	2013-04-05 21:00:00	0.039187	NaN	-0.215813	-0.073813	0.152187	0.042187	-0.163813	-0.176813	0.035187	...	-0.080813	-0.037813	-
3	2013-04-08 10:00:00	0.056350	-0.039650	-0.171650	0.133350	-0.028650	-0.100650	-0.072650	-0.137650	-0.116650	...	-0.073650	-0.051650	8
4	2013-04-08 11:00:00	-0.036222	-0.076222	-0.174222	-0.023222	0.079778	-0.103222	-0.083222	-0.042222	-0.113222	...	-0.134222	-0.079222	9

5 rows × 327 columns

In [52]:
```
df_type0_res
```

Out[52]:

	GMT	D0001	D0008	D0010	D0012	D0013	D0016	D0018	D0021	D0024	...	D1023	D1024
0	2013-04-05 19:00:00	0.059165	0.309165	-0.210835	0.185165	0.742165	0.214165	-0.165835	-0.176835	0.022165	...	0.004165	0.250165
1	2013-04-05 20:00:00	0.060997	0.144997	-0.245003	-0.055003	0.108997	0.189997	-0.205003	-0.207003	-0.004003	...	-0.046003	-0.031003
2	2013-04-05 21:00:00	0.039187	NaN	-0.215813	-0.073813	0.152187	0.042187	-0.163813	-0.176813	0.035187	...	-0.080813	-0.037813
3	2013-04-08 10:00:00	0.056350	-0.039650	-0.171650	0.133350	-0.028650	-0.100650	-0.072650	-0.137650	-0.116650	...	-0.073650	-0.051650
4	2013-04-08 11:00:00	-0.036222	-0.076222	-0.174222	-0.023222	0.079778	-0.103222	-0.083222	-0.042222	-0.113222	...	-0.134222	-0.079222
5	2013-04-08 12:00:00	-0.021420	-0.066420	-0.183420	0.685580	-0.053420	-0.113420	-0.125420	-0.151420	-0.119420	...	-0.137420	-0.099420
6	2013-04-11 01:00:00	-0.020985	0.011015	-0.091985	0.046015	-0.003985	-0.018985	-0.040985	-0.092985	0.056015	...	0.023015	0.045015
7	2013-04-11 02:00:00	0.029225	0.038225	-0.089775	0.045225	0.003225	-0.017775	-0.006775	-0.089775	0.037225	...	-0.041775	0.017225
8	2013-04-11 03:00:00	-0.027269	0.040731	-0.088269	0.022731	0.015731	-0.016269	-0.042269	-0.088269	-0.016269	...	-0.061269	0.012731
9	2013-04-13 16:00:00	-0.160185	-0.110185	-0.202185	0.263815	0.047815	-0.125185	-0.164185	-0.187185	-0.127185	...	-0.185185	-0.068185
10	2013-04-13 17:00:00	-0.104299	-0.107299	-0.226299	0.162701	0.108701	-0.158299	-0.149299	-0.212299	-0.166299	...	-0.199299	-0.092299
11	2013-04-13 18:00:00	-0.001917	-0.127917	-0.230917	0.356083	0.026083	0.007083	-0.180917	0.119083	0.071083	...	-0.181917	-0.038917
12	2013-04-13 19:00:00	-0.137588	-0.136588	-0.246588	0.229412	0.012412	0.217412	-0.195588	-0.194588	-0.088588	...	-0.230588	-0.057588
13	2013-04-13 20:00:00	-0.152610	-0.007610	-0.239610	0.235390	0.069390	-0.075610	-0.163610	-0.175610	0.014390	...	-0.193610	0.028390
14	2013-04-13 21:00:00	-0.100988	0.192012	-0.215988	0.169012	0.038012	0.141012	-0.128988	-0.172988	0.031012	...	-0.098988	-0.030988
15	2013-04-16 04:00:00	0.021026	0.066026	-0.093974	0.022026	0.014026	-0.000974	-0.011974	-0.093974	-0.024974	...	-0.037974	-0.019974
16	2013-04-16 05:00:00	-0.037351	0.021649	-0.102351	0.042649	0.008649	-0.008351	-0.051351	-0.103351	-0.029351	...	-0.086351	-0.020351
17	2013-04-16 06:00:00	-0.021102	-0.042102	-0.134102	0.021898	0.072898	0.316898	-0.054102	-0.134102	0.085898	...	-0.084102	-0.080102
18	2013-04-16 07:00:00	0.085523	0.013523	-0.161477	0.141523	0.975523	0.035523	-0.027477	-0.161477	0.646523	...	-0.136477	-0.064477
19	2013-04-16 08:00:00	-0.075930	0.154070	-0.168930	0.170070	0.530070	-0.100930	0.276070	-0.169930	-0.006930	...	-0.141930	-0.119930

	GMT	D0001	D0008	D0010	D0012	D0013	D0016	D0018	D0021	D0024	...	D1023	D1024
20	2013-04-16 09:00:00	-0.077061	-0.046061	-0.161061	0.092939	0.905939	-0.093061	0.045939	-0.161061	-0.084061	...	-0.037061	-0.078061
21	2013-04-21 10:00:00	0.214167	-0.070833	-0.151833	0.023167	0.585167	-0.093833	0.724167	-0.136833	0.538167	...	-0.152833	-0.102833
22	2013-04-21 11:00:00	-0.102202	-0.077202	-0.161202	0.001798	1.073798	-0.106202	0.775798	-0.167202	0.025798	...	-0.076202	-0.108202
23	2013-04-21 12:00:00	-0.073161	-0.073161	-0.158161	-0.039161	0.965839	-0.100161	-0.119161	-0.184161	0.619839	...	-0.061161	0.037839
24	2013-04-23 04:00:00	0.022072	0.016072	-0.063928	-0.001928	0.005072	-0.023928	-0.046928	-0.093928	-0.011928	...	-0.054928	-0.005928
25	2013-04-23 05:00:00	0.001783	0.009783	-0.086217	0.022783	-0.014217	0.193783	-0.032217	-0.103217	-0.034217	...	-0.084217	-0.057217
26	2013-04-23 06:00:00	-0.049102	-0.035102	-0.103102	0.050898	0.121898	0.026898	-0.082102	-0.134102	0.366898	...	-0.071102	-0.051102
27	2013-04-23 07:00:00	-0.055967	-0.034967	-0.071967	-0.011967	-0.061967	-0.085967	0.180033	-0.159967	0.381033	...	-0.140967	0.032033
28	2013-04-23 08:00:00	0.087113	-0.071887	-0.051887	0.806113	-0.062887	-0.079887	0.544113	-0.165887	-0.057887	...	-0.133887	-0.101887
29	2013-04-23 09:00:00	0.189463	-0.034537	-0.038537	0.715463	-0.058537	-0.085537	0.172463	-0.156537	-0.084537	...	-0.083537	-0.086537
...	...	...	...	...	...	...	...	...	...	...	...	...	...
471	2013-10-20 04:00:00	-0.004580	0.040420	-0.048580	0.019420	0.017420	-0.020580	-0.026580	-0.090580	-0.017580	...	-0.030580	-0.034580
472	2013-10-20 05:00:00	-0.040567	0.046433	-0.059567	0.037433	0.041433	-0.018567	-0.021567	-0.096567	-0.010567	...	-0.075567	-0.047567
473	2013-10-20 06:00:00	0.016420	-0.018580	-0.074580	0.067420	0.018420	-0.035580	-0.058580	-0.113580	0.016420	...	-0.076580	-0.043580
474	2013-10-20 07:00:00	0.005490	-0.008510	-0.091510	0.177490	-0.030510	0.068490	0.336490	-0.140510	0.064490	...	-0.111510	-0.079510
475	2013-10-20 08:00:00	-0.036917	-0.029917	-0.124917	1.248083	-0.056917	0.142083	0.493083	-0.161917	0.032083	...	-0.131917	-0.099917
476	2013-10-20 09:00:00	0.168666	0.539666	-0.053334	0.057666	0.396666	-0.082334	0.914666	-0.175334	0.042666	...	-0.137334	-0.050334
477	2013-10-20 10:00:00	-0.073856	0.235144	-0.117856	0.532144	-0.005856	-0.091856	-0.063856	-0.168856	-0.092856	...	-0.102856	-0.090856
478	2013-10-20 11:00:00	-0.000578	0.390422	-0.147578	0.148422	0.275422	-0.103578	0.023422	0.004422	-0.052578	...	-0.055578	-0.134578
479	2013-10-20 12:00:00	-0.092161	0.941839	-0.166161	0.021839	0.485839	-0.100161	0.559839	-0.145161	0.072839	...	-0.160161	-0.114161
480	2013-10-20 13:00:00	-0.096463	0.755537	-0.160463	0.008537	0.057537	-0.113463	0.763537	-0.144463	-0.083463	...	-0.123463	0.275537

	GMT	D0001	D0008	D0010	D0012	D0013	D0016	D0018	D0021	D0024	...	D1023	D1024
481	2013-10-20 14:00:00	-0.104833	1.007167	-0.090833	-0.049833	0.496167	-0.097833	0.074167	-0.136833	0.048167	...	-0.061833	-0.06883
482	2013-10-20 15:00:00	-0.072993	0.926007	-0.154993	-0.045993	0.996007	-0.108993	0.135007	-0.173993	-0.034993	...	-0.082993	-0.07999
483	2013-10-20 16:00:00	-0.062334	0.098666	-0.172334	-0.072334	0.266666	-0.117334	-0.073334	-0.190334	0.010666	...	-0.142334	0.016666
484	2013-10-20 17:00:00	-0.064158	0.050842	-0.183158	-0.067158	0.270842	0.016842	-0.072158	-0.201158	-0.028158	...	-0.119158	0.042842
485	2013-10-20 18:00:00	-0.085146	0.241854	-0.187146	-0.048146	0.104854	0.166854	0.195854	-0.219146	0.057854	...	-0.152146	-0.10414
486	2013-10-20 19:00:00	-0.095181	0.114819	-0.195181	0.231819	0.791819	-0.008181	0.104819	-0.217181	0.167819	...	-0.109181	-0.14118
487	2013-10-20 20:00:00	-0.153420	0.090580	-0.199420	0.195580	0.386580	-0.037420	0.199580	-0.219420	0.110580	...	-0.136420	-0.13442
488	2013-10-20 21:00:00	0.114515	0.083515	-0.193485	0.177515	0.141515	0.003515	0.238515	-0.061485	0.030515	...	-0.098485	-0.12048
489	2013-10-20 22:00:00	0.219094	0.084094	-0.153906	0.250094	-0.016906	0.017094	0.220094	-0.132906	-0.034906	...	-0.071906	0.176094
490	2013-10-20 23:00:00	0.181462	0.153462	-0.117538	0.052462	-0.025538	-0.015538	-0.026538	-0.133538	0.013462	...	-0.025538	-0.06753
491	2013-10-21 00:00:00	-0.042715	0.303285	-0.093715	0.039285	-0.002715	-0.039715	0.020285	-0.112715	0.026285	...	-0.033715	-0.04171
492	2013-10-21 01:00:00	-0.024798	0.002202	-0.068798	0.010202	0.020202	-0.013798	-0.014798	-0.098798	0.059202	...	-0.076798	-0.05479
493	2013-10-21 02:00:00	-0.007945	0.023055	-0.069945	0.044055	0.019055	-0.022945	-0.045945	-0.092945	0.035055	...	-0.036945	-0.01994
494	2013-10-21 03:00:00	-0.025181	0.045819	-0.070181	0.024819	0.020819	-0.021181	-0.007181	-0.091181	0.000819	...	-0.074181	-0.02518
495	2013-10-31 02:00:00	-0.006122	0.450878	-0.074122	0.022878	0.011878	-0.016122	-0.033122	-0.092122	0.068878	...	-0.075122	0.003878
496	2013-10-31 03:00:00	-0.005571	0.197429	-0.070571	0.022429	0.011429	-0.008571	-0.006571	-0.088571	0.046429	...	-0.022571	-0.01957
497	2013-10-31 04:00:00	-0.020389	0.161611	-0.072389	0.059611	0.012611	-0.023389	-0.048389	-0.092389	-0.001389	...	-0.073389	-0.01438
498	2013-10-31 05:00:00	-0.042486	0.099514	-0.071486	0.002514	0.004514	-0.031486	0.008514	-0.100486	-0.024486	...	-0.063486	-0.03348
499	2013-10-31 06:00:00	0.012769	-0.024231	-0.106231	0.008769	-0.024231	0.116769	-0.076231	-0.130231	-0.049231	...	-0.091231	-0.04623
500	2013-10-31 07:00:00	-0.017527	-0.036527	-0.140527	1.337473	0.160473	0.335473	-0.077527	-0.149527	-0.003527	...	-0.143527	-0.08852

501 rows × 327 columns

In [60]:
```
# select each user ID and form a seperate dataframe
# first change all the user ID to "Consumption"
new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[0]):
    new_col.append('Consumption')
new_col = new_col + list(df_type0_res.columns)[-8:] # the new column names
df_type0_res.columns = new_col
df_all_res_long = pd.DataFrame()
for i in range(1, houseGroupDf.counts.iloc[0] + 1):
    df_all_res_long = df_all_res_long.append(df_type0_res.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[1]): # for renaming the second type
    new_col.append('Consumption')
new_col = new_col + list(df_type1_res.columns)[-8:] # the new column names
df_type1_res.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[1] + 1):
    df_all_res_long = df_all_res_long.append(df_type1_res.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[2]): # for renaming the third type
    new_col.append('Consumption')
new_col = new_col + list(df_type2_res.columns)[-8:] # the new column names
df_type2_res.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[2] + 1):
    df_all_res_long = df_all_res_long.append(df_type2_res.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

# for clearer visulization, round the temperatures
df_all_res_long['TempC'] = df_all_res_long.TempC.round(0)
df_all_res_long['TempF'] = df_all_res_long.TempF.round(0)
df_all_res_long # ready for the price responsiveness analysis
```

Out[60]:

	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
0	2013-04-05 19:00:00	0.059165	2.0	36.0	0.6720	H3	4	19	0	0.210835
1	2013-04-05 20:00:00	0.060997	1.0	34.0	0.6720	H3	4	20	0	0.245003
2	2013-04-05 21:00:00	0.039187	-1.0	31.0	0.6720	H3	4	21	0	0.216813
3	2013-04-08 10:00:00	0.056350	8.0	46.0	0.6720	H3	0	10	0	0.171650
4	2013-04-08 11:00:00	-0.036222	9.0	48.0	0.6720	H3	0	11	0	0.174222
5	2013-04-08 12:00:00	-0.021420	10.0	49.0	0.6720	H3	0	12	0	0.184420
6	2013-04-11 01:00:00	-0.020985	4.0	39.0	0.6720	H3	3	1	0	0.092985
7	2013-04-11 02:00:00	0.029225	5.0	40.0	0.6720	H3	3	2	0	0.089775
8	2013-04-11 03:00:00	-0.027269	5.0	40.0	0.6720	H3	3	3	0	0.088269
9	2013-04-13 16:00:00	-0.160185	9.0	49.0	0.6720	H6	5	16	0	0.202185
10	2013-04-13 17:00:00	-0.104299	10.0	49.0	0.6720	H6	5	17	0	0.227299
11	2013-04-13 18:00:00	-0.001917	10.0	50.0	0.6720	H6	5	18	0	0.230917
12	2013-04-13 19:00:00	-0.137588	10.0	50.0	0.6720	H6	5	19	0	0.246588
13	2013-04-13 20:00:00	-0.152610	10.0	50.0	0.6720	H6	5	20	0	0.239610
14	2013-04-13 21:00:00	-0.100988	10.0	50.0	0.6720	H6	5	21	0	0.216988
15	2013-04-16 04:00:00	0.021026	8.0	47.0	0.0399	L6	1	4	0	0.093974
16	2013-04-16 05:00:00	-0.037351	9.0	47.0	0.0399	L6	1	5	0	0.103351
17	2013-04-16 06:00:00	-0.021102	9.0	47.0	0.0399	L6	1	6	0	0.134102
18	2013-04-16 07:00:00	0.085523	10.0	49.0	0.0399	L6	1	7	0	0.161477
19	2013-04-16 08:00:00	-0.075930	11.0	51.0	0.0399	L6	1	8	0	0.169930
20	2013-04-16 09:00:00	-0.077061	12.0	53.0	0.0399	L6	1	9	0	0.161061
21	2013-04-21 10:00:00	0.214167	12.0	53.0	0.0399	L3	6	10	0	0.182833
22	2013-04-21 11:00:00	-0.102202	13.0	56.0	0.0399	L3	6	11	0	0.178202
23	2013-04-21 12:00:00	-0.073161	15.0	58.0	0.0399	L3	6	12	0	0.184161
24	2013-04-23 04:00:00	0.022072	9.0	48.0	0.0399	CM	1	4	0	0.093928
25	2013-04-23 05:00:00	0.001783	9.0	48.0	0.0399	CM	1	5	0	0.103217
26	2013-04-23 06:00:00	-0.049102	9.0	48.0	0.0399	CM	1	6	0	0.134102
27	2013-04-23 07:00:00	-0.055967	11.0	51.0	0.0399	CM	1	7	0	0.159967
28	2013-04-23 08:00:00	0.087113	13.0	55.0	0.0399	CM	1	8	0	0.165887

	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
29	2013-04-23 09:00:00	0.189463	15.0	58.0	0.0399	CM	1	9	0	0.156537
...	...	...	...	...	...	...	...	...	...	...
172815	2013-10-20 04:00:00	-0.050036	15.0	60.0	0.0399	CM	6	4	2	0.125036
172816	2013-10-20 05:00:00	0.176306	15.0	60.0	0.0399	CM	6	5	2	0.129694
172817	2013-10-20 06:00:00	0.005842	15.0	60.0	0.0399	CM	6	6	2	0.150158
172818	2013-10-20 07:00:00	-0.157141	15.0	60.0	0.0399	CM	6	7	2	0.236141
172819	2013-10-20 08:00:00	-0.141628	15.0	60.0	0.0399	CM	6	8	2	0.312628
172820	2013-10-20 09:00:00	0.041124	15.0	60.0	0.0399	CM	6	9	2	0.352876
172821	2013-10-20 10:00:00	0.646515	15.0	60.0	0.0399	CM	6	10	2	0.303485
172822	2013-10-20 11:00:00	0.059131	15.0	60.0	0.0399	CM	6	11	2	0.243869
172823	2013-10-20 12:00:00	-0.043581	15.0	60.0	0.0399	CM	6	12	2	0.352581
172824	2013-10-20 13:00:00	0.025959	15.0	60.0	0.0399	CM	6	13	2	0.280041
172825	2013-10-20 14:00:00	0.063204	15.0	60.0	0.0399	CM	6	14	2	0.231796
172826	2013-10-20 15:00:00	0.048402	15.0	60.0	0.0399	CM	6	15	2	0.247598
172827	2013-10-20 16:00:00	0.002518	15.0	60.0	0.6720	CM	6	16	2	0.295482
172828	2013-10-20 17:00:00	0.448101	15.0	60.0	0.6720	CM	6	17	2	0.417899
172829	2013-10-20 18:00:00	-0.031712	15.0	60.0	0.6720	CM	6	18	2	0.394712
172830	2013-10-20 19:00:00	0.023072	15.0	60.0	0.6720	CM	6	19	2	0.301928
172831	2013-10-20 20:00:00	-0.073839	15.0	60.0	0.6720	CM	6	20	2	0.286839
172832	2013-10-20 21:00:00	-0.133330	15.0	60.0	0.6720	CM	6	21	2	0.281330
172833	2013-10-20 22:00:00	-0.164549	15.0	60.0	0.0399	CM	6	22	2	0.231549
172834	2013-10-20 23:00:00	-0.077590	15.0	59.0	0.0399	CM	6	23	2	0.160590
172835	2013-10-21 00:00:00	-0.068899	15.0	59.0	0.0399	CM	0	0	2	0.138899
172836	2013-10-21 01:00:00	-0.057239	15.0	59.0	0.0399	CM	0	1	2	0.134239
172837	2013-10-21 02:00:00	-0.049731	15.0	59.0	0.0399	CM	0	2	2	0.127731
172838	2013-10-21 03:00:00	-0.063047	15.0	59.0	0.0399	CM	0	3	2	0.132047
172839	2013-10-31 02:00:00	-0.052853	13.0	56.0	0.6720	H3	3	2	2	0.126853
172840	2013-10-31 03:00:00	-0.048703	13.0	56.0	0.6720	H3	3	3	2	0.127703
172841	2013-10-31 04:00:00	-0.076388	13.0	56.0	0.6720	H3	3	4	2	0.141388
172842	2013-10-31 05:00:00	-0.071596	13.0	56.0	0.0399	L3	3	5	2	0.153596

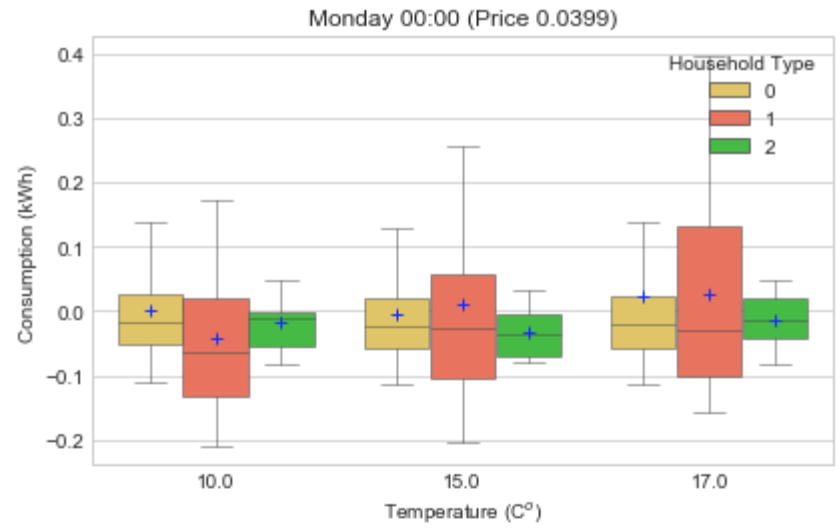
	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
172843	2013-10-31 06:00:00	0.040278	13.0	56.0	0.0399	L3	3	6	2	0.196722
172844	2013-10-31 07:00:00	0.120918	13.0	56.0	0.0399	L3	3	7	2	0.255082

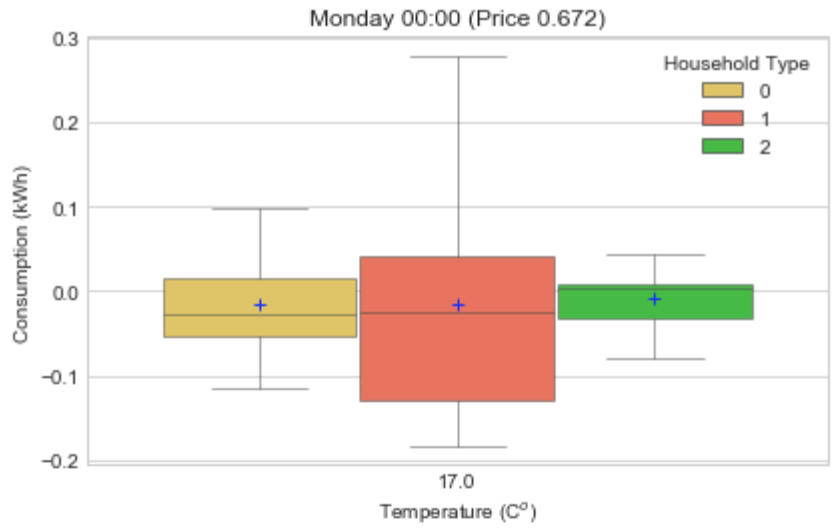
172845 rows × 10 columns

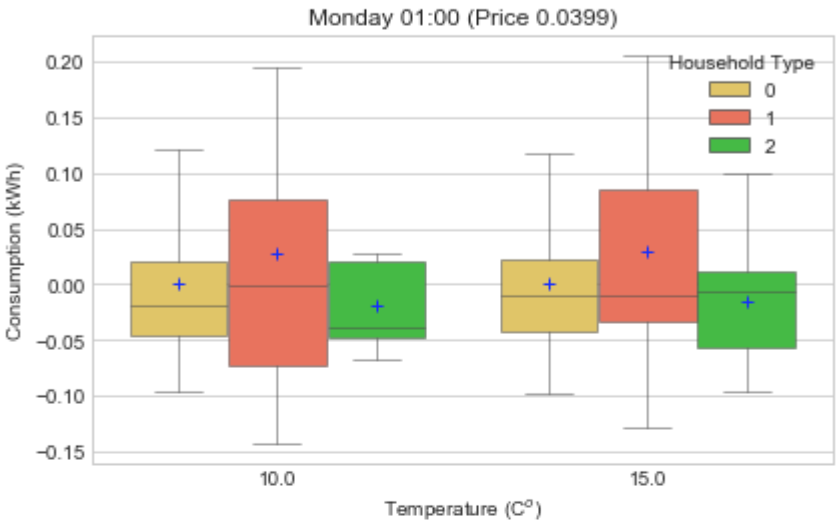
Then plot the properties based on from the smallest unit (day of week, hour, temp) to (day of week, hour), (day of week, temp), (hour, temp), (day of week), (hour), (temp) for different prices (high and low) so we will have very comprehensive price responsiveness of three different groups

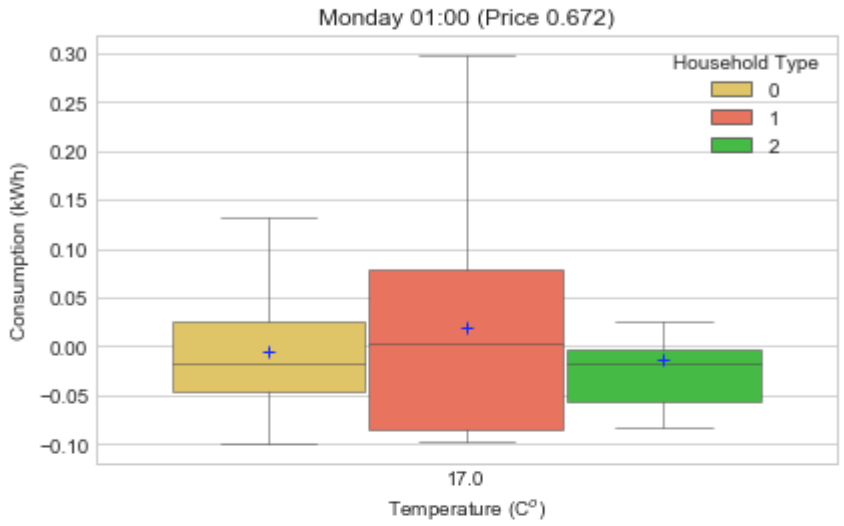
a) Day of week - hour - temperature : prices

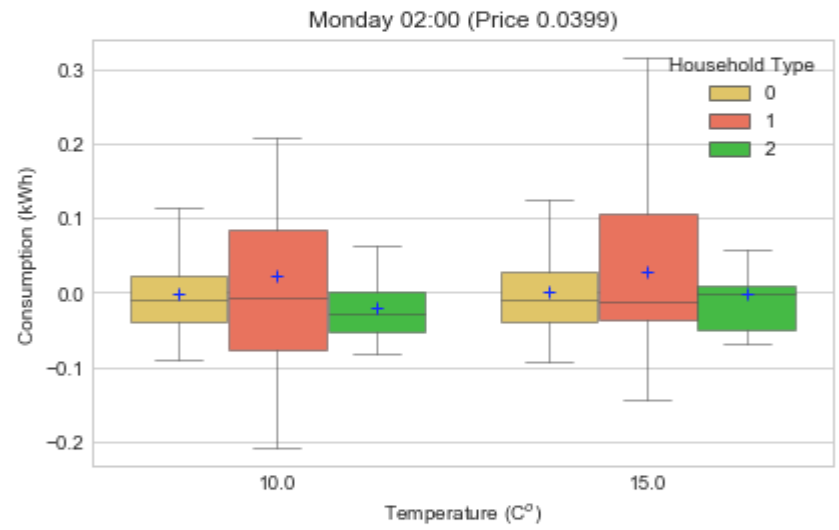
In [61]:
```
# Monday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 0 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

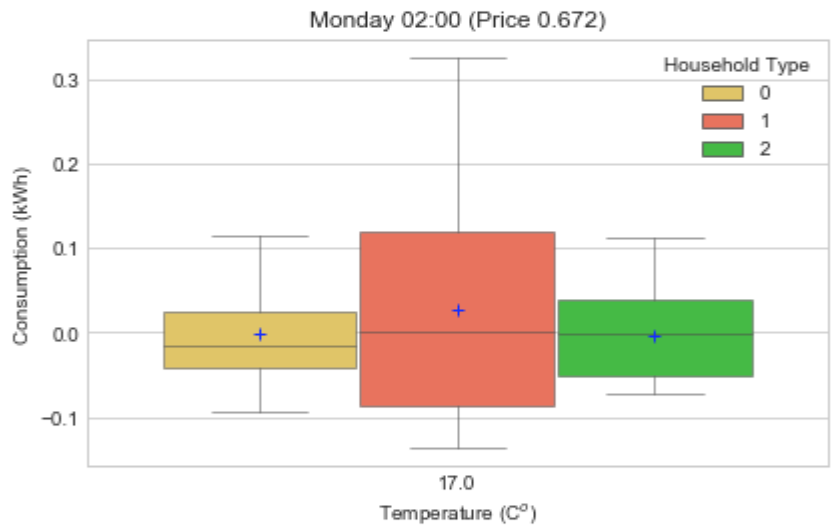

Monday 00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
10.0
15.0
17.0
Temperature (C°)

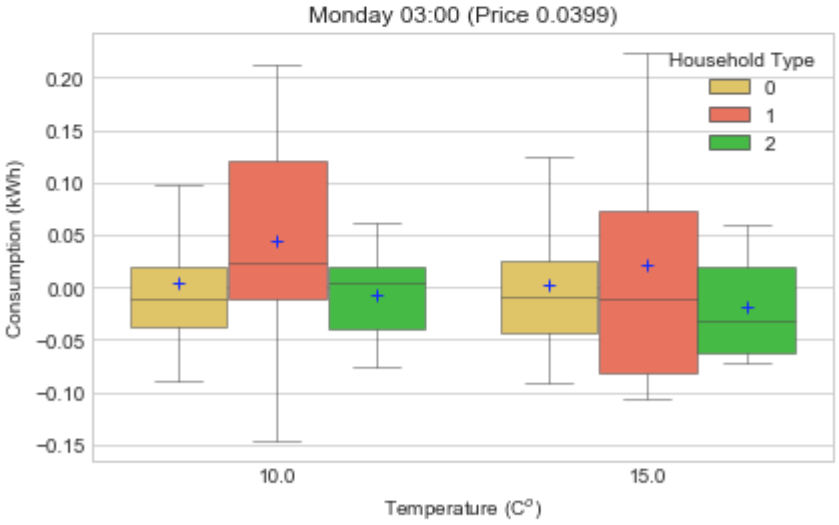

Monday 00:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
17.0
Temperature (C°)

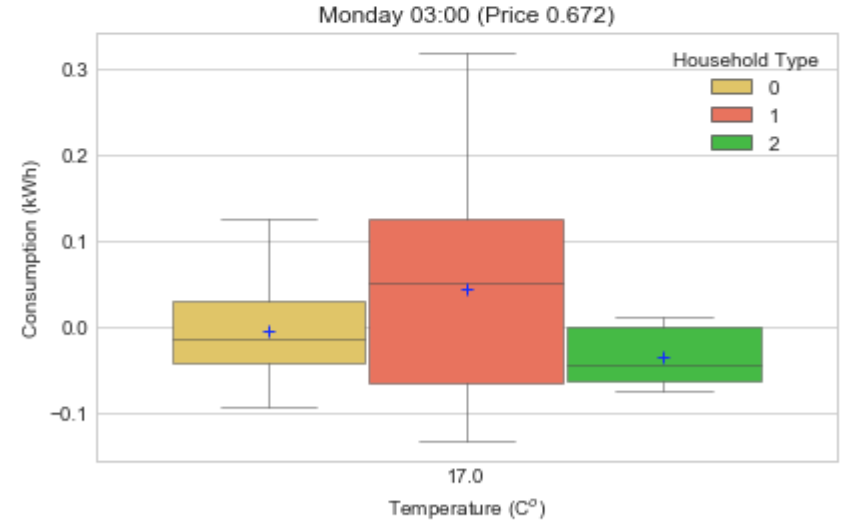

Monday 01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
10.0
15.0
Temperature (C°)

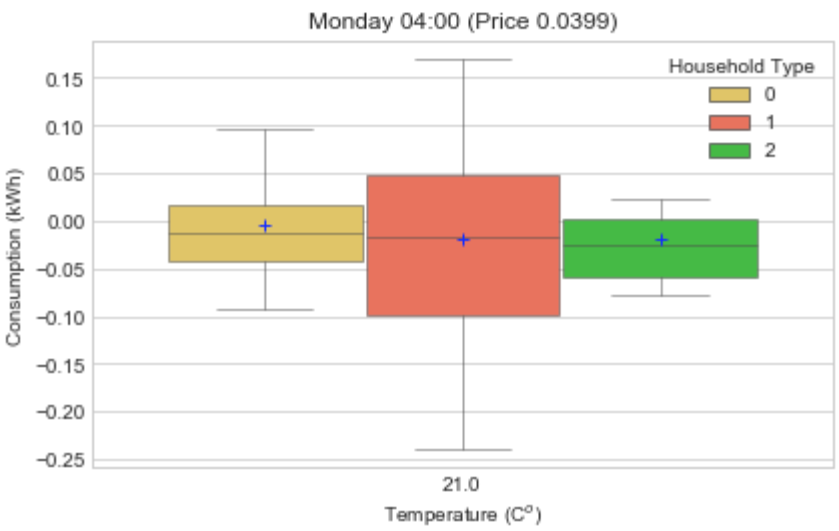

Monday 01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
17.0
Temperature (C°)

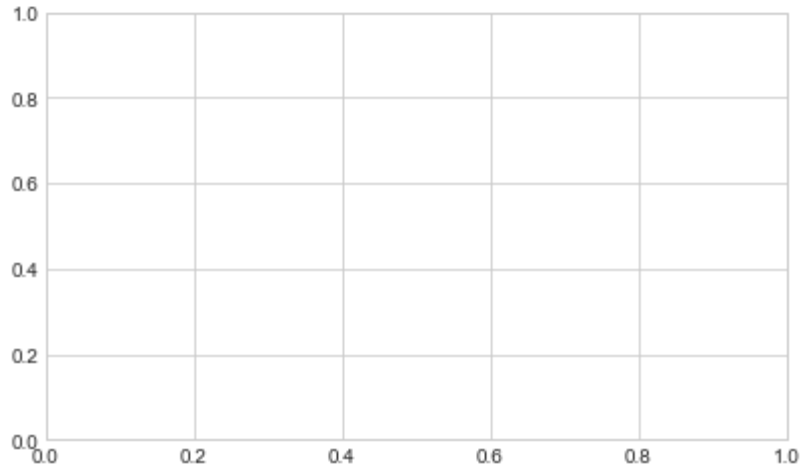

Monday 02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
10.0
15.0
Temperature (C°)


Monday 02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
17.0
Temperature (C°)

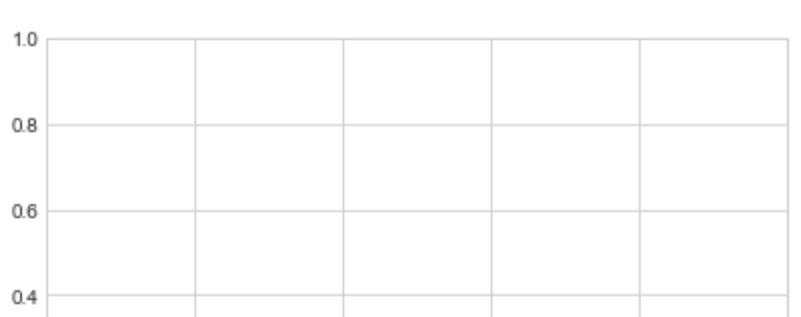

Monday 03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
10.0
15.0
Temperature (C°)

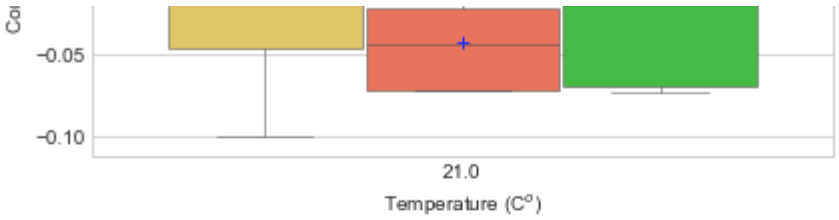

Monday 03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
17.0
Temperature (C°)


Monday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
21.0
Temperature (C°)

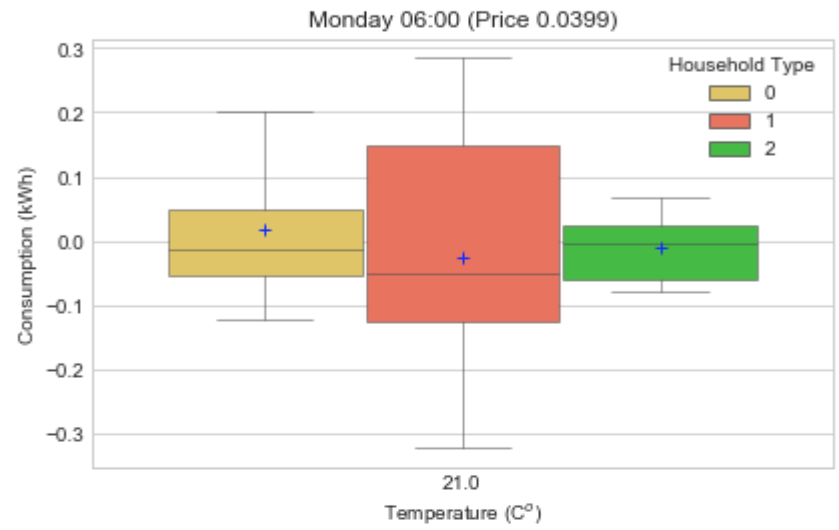

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

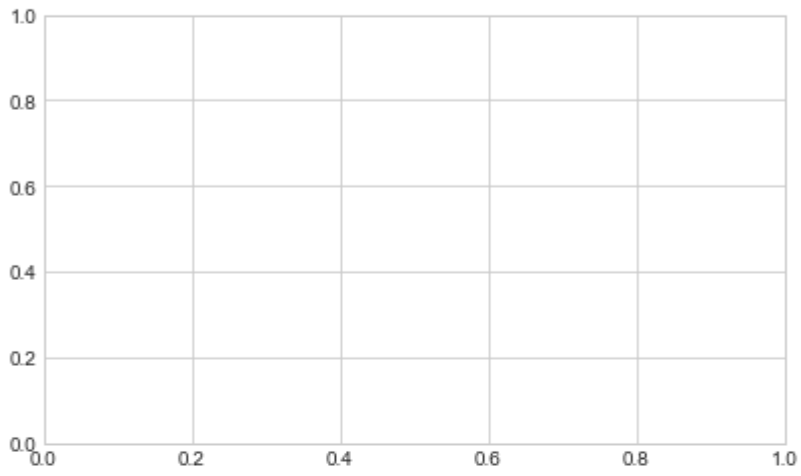

Monday 05:00 (Price 0.0399)
Household Type
0
1
2
0.15
0.10
0.05
0.00

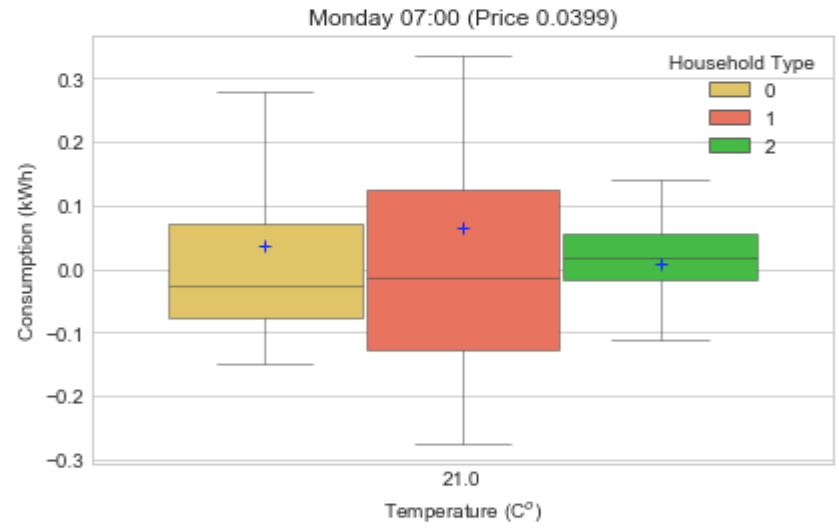

1.0
0.8
0.6
0.4

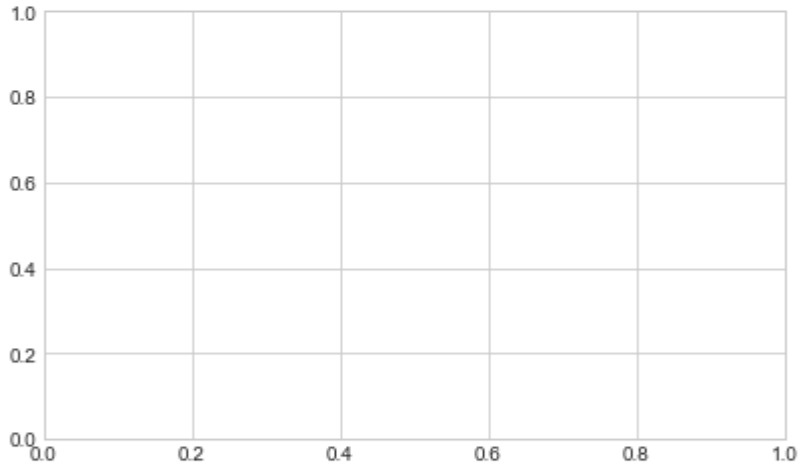

−0.05
−0.10
21.0
Temperature (C°)

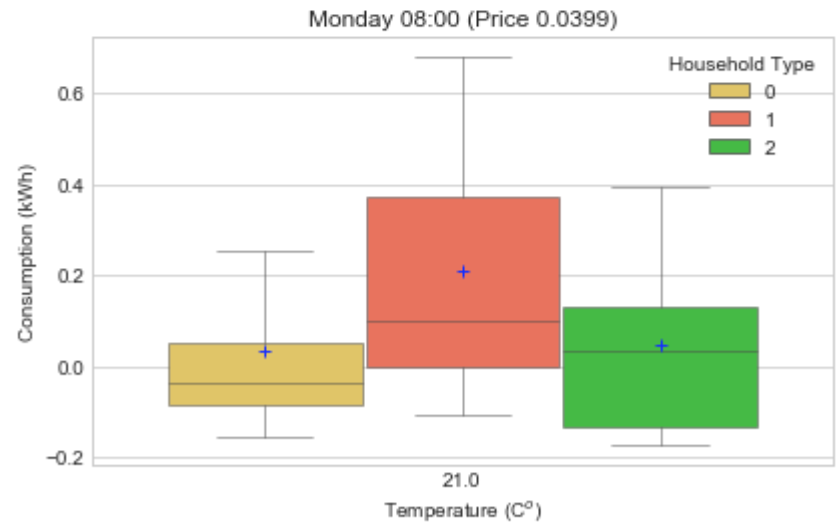

0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

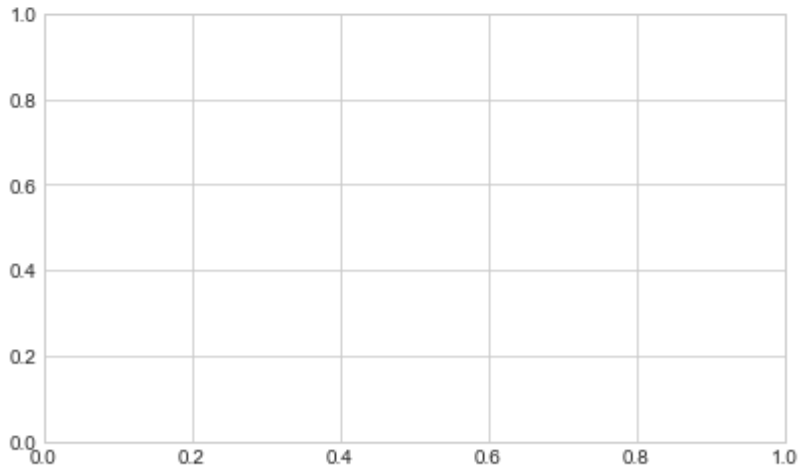

Monday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
21.0
Temperature (C°)

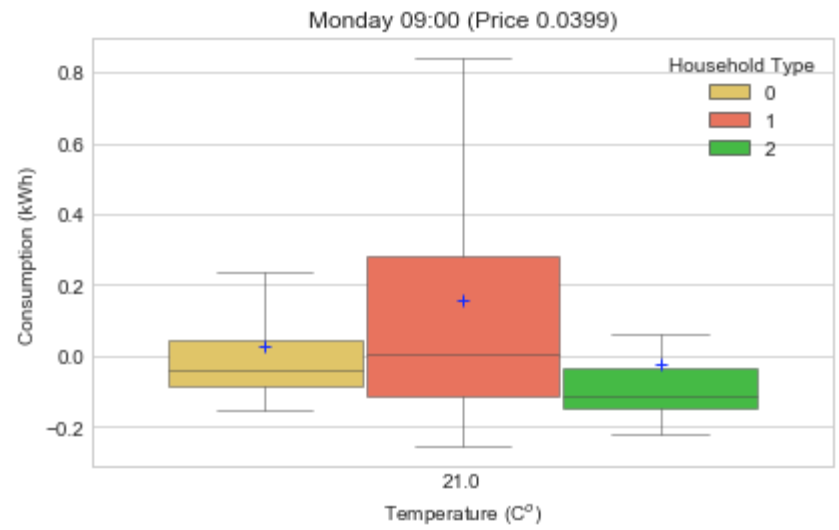

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

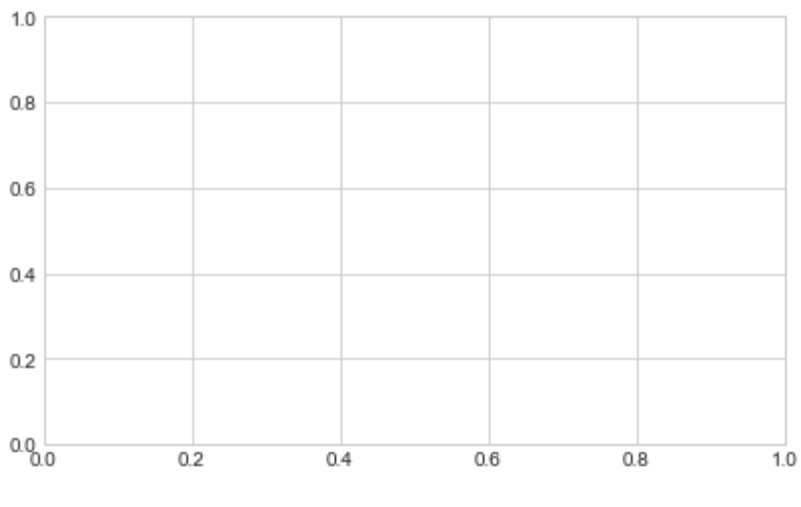

Monday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
21.0
Temperature (C°)

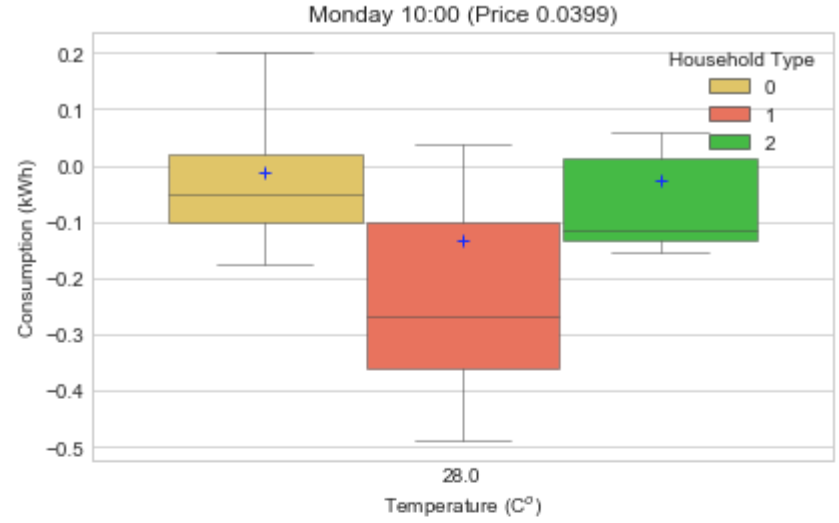

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

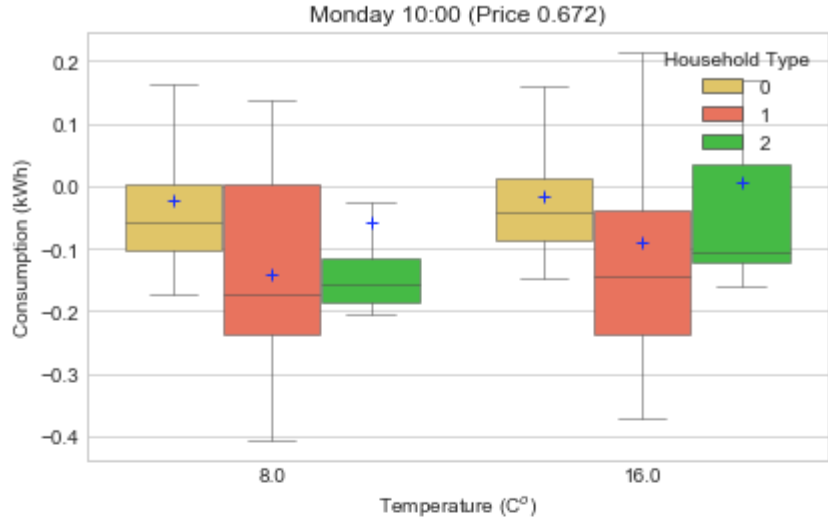

Monday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
21.0
Temperature (C°)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

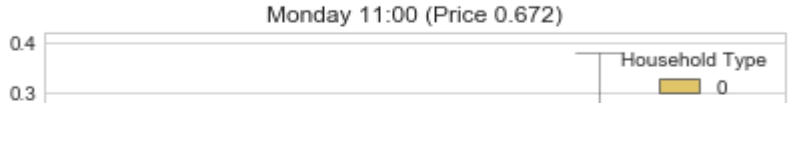

Monday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
21.0
Temperature (C°)

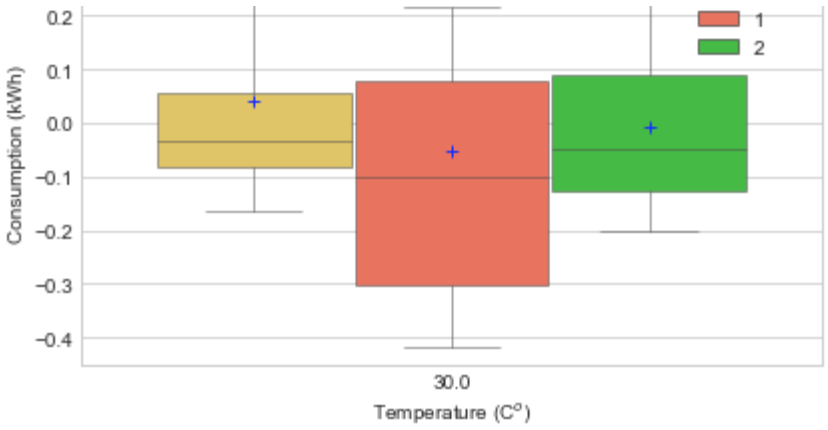

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

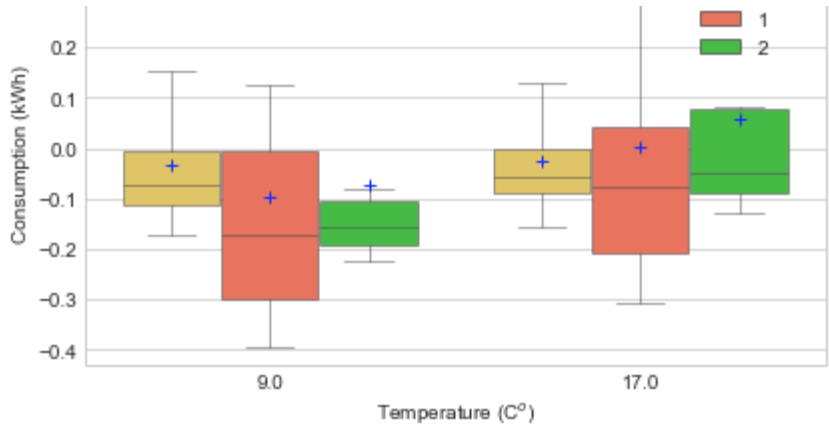

Monday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
28.0
Temperature (C°)

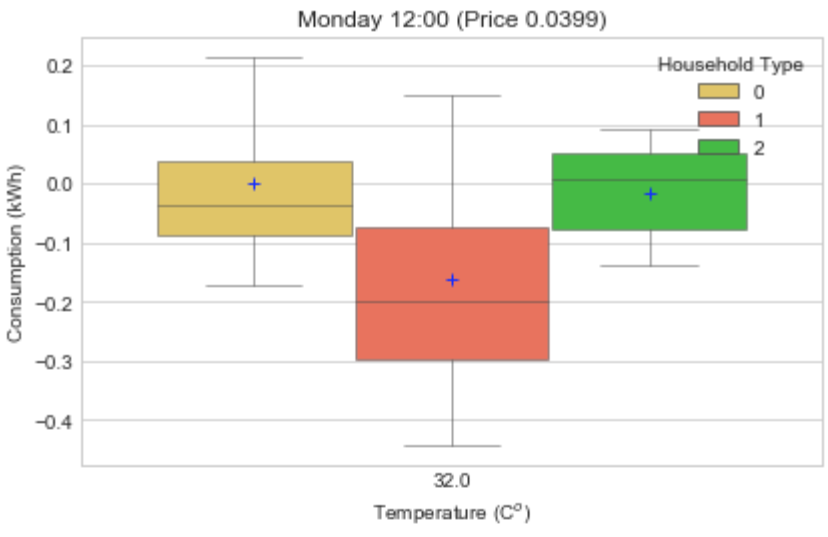

Monday 10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
8.0
16.0
Temperature (C°)

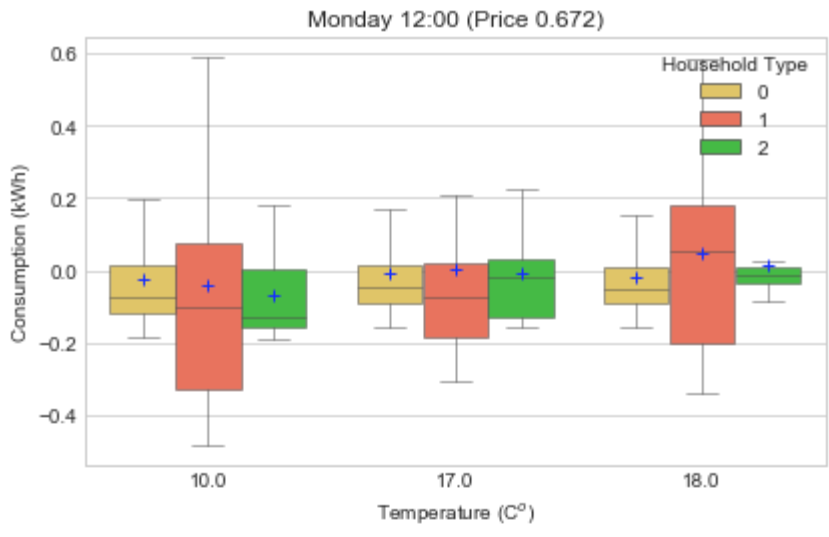

Monday 11:00 (Price 0.0399)
Household Type
0
0.3

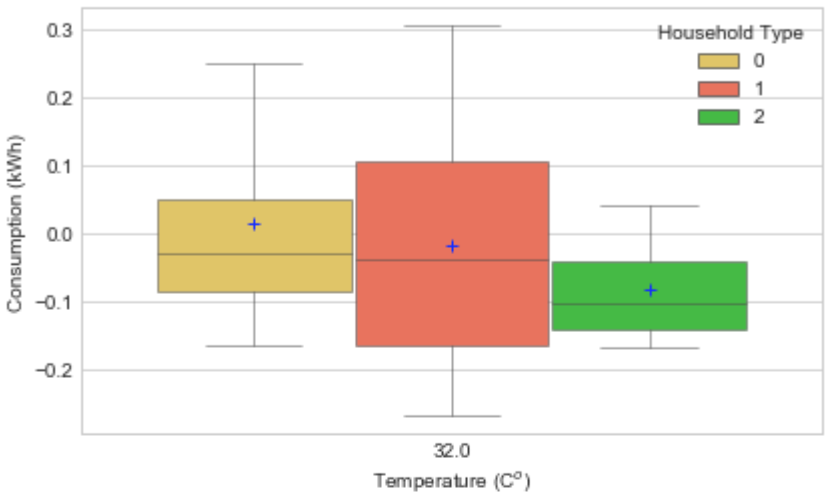

Monday 11:00 (Price 0.672)
Household Type
0
0.4
0.3

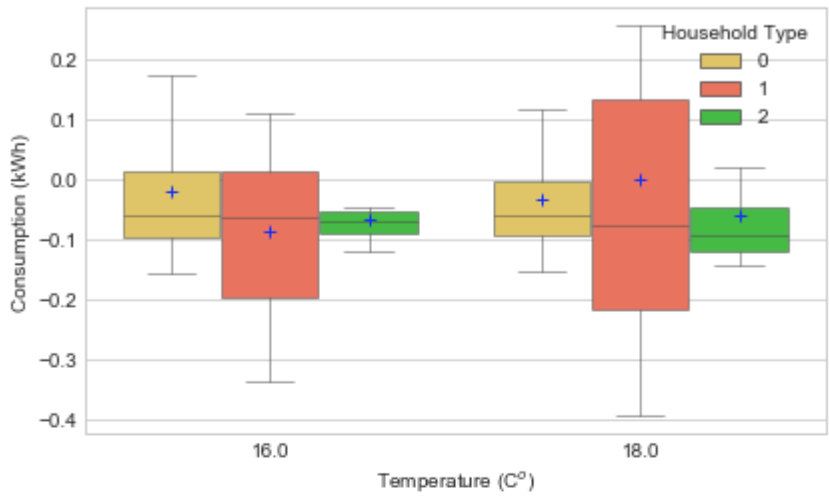

Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
1
2
30.0
Temperature (C°)

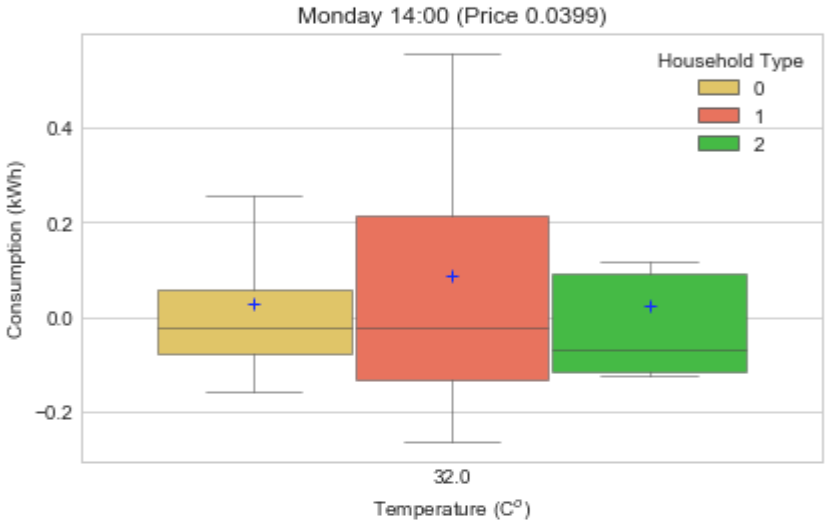

Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
1
2
9.0
17.0
Temperature (C°)

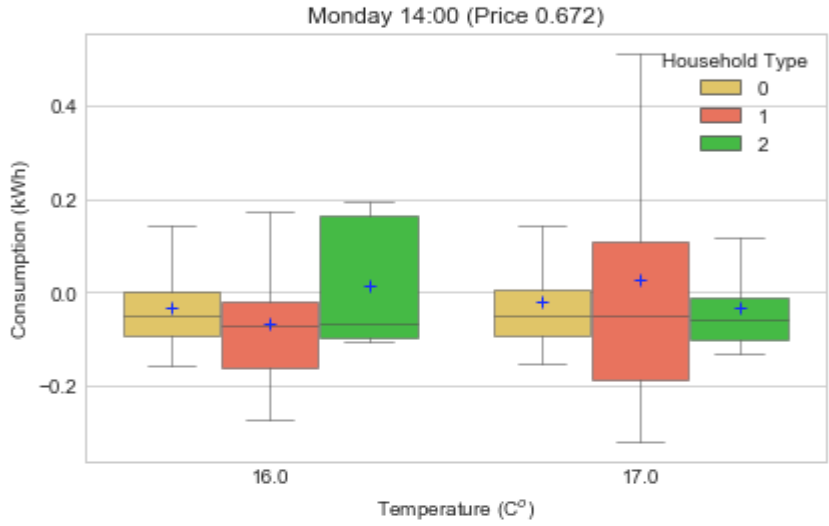

Monday 12:00 (Price 0.0399)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
32.0
Temperature (C°)

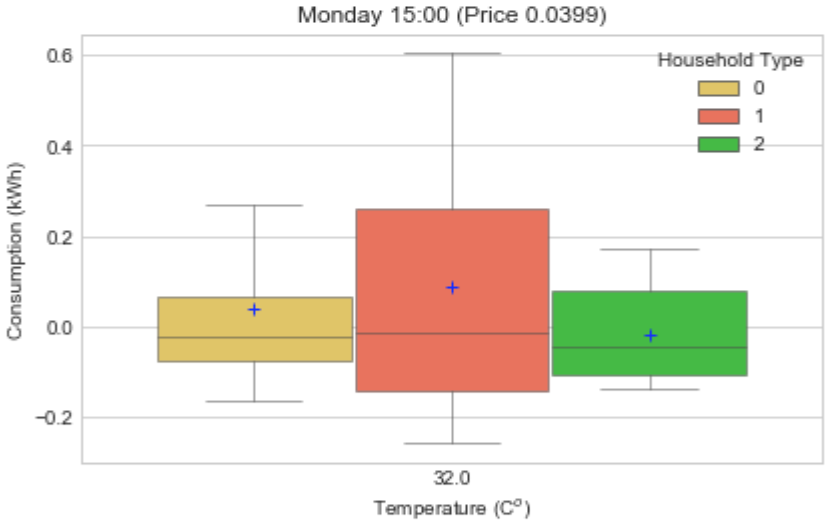

Monday 12:00 (Price 0.672)
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
10.0
17.0
18.0
Temperature (C°)

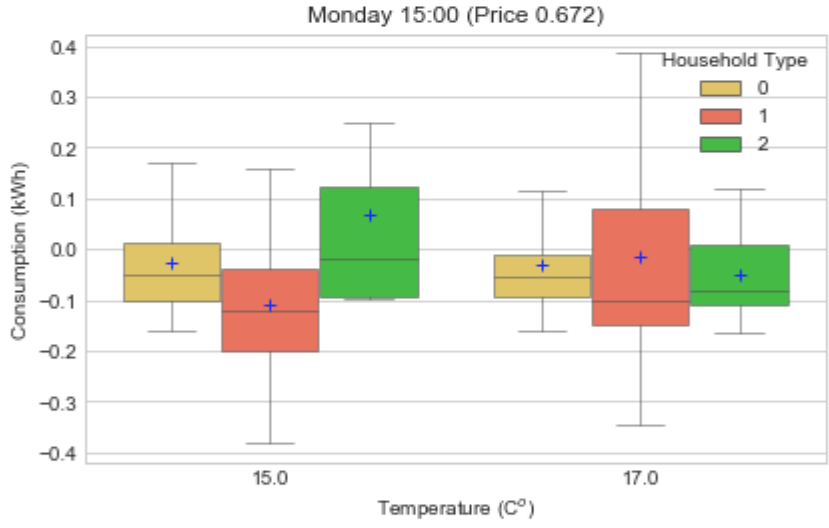

Monday 13:00 (Price 0.0399)
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
Household Type
0
1
2
32.0
Temperature (C°)

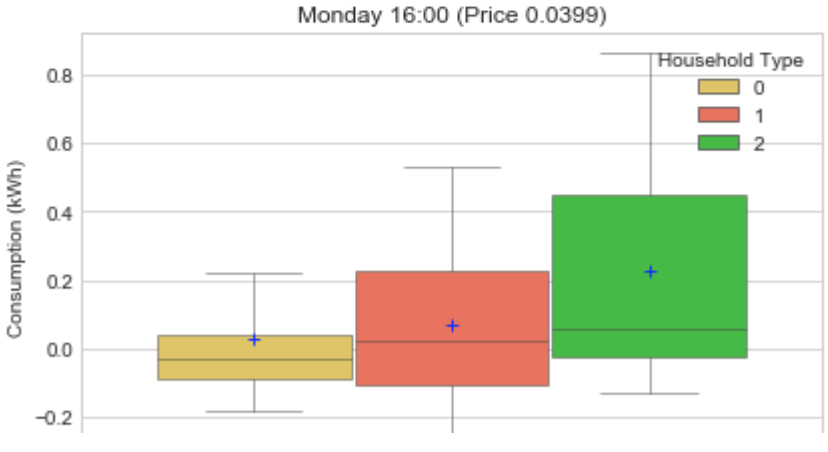

Monday 13:00 (Price 0.672)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
16.0
18.0
Temperature (C°)

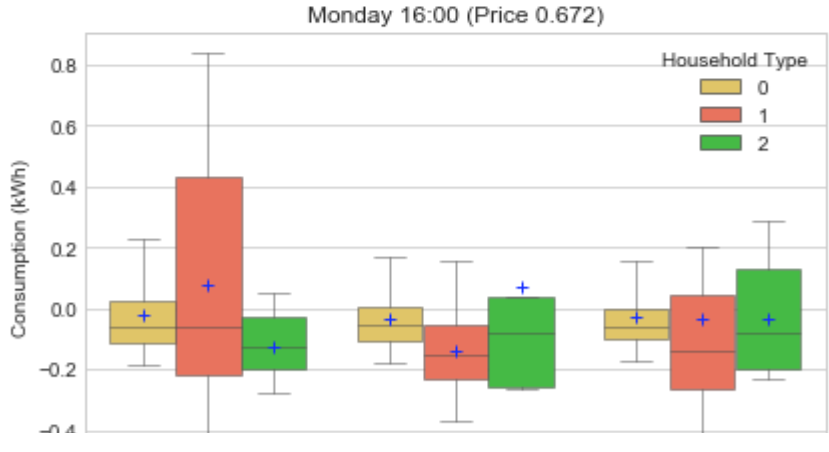

Monday 14:00 (Price 0.0399)
Consumption (kWh)
0.4
0.2
0.0
−0.2
Household Type
0
1
2
32.0
Temperature (C°)

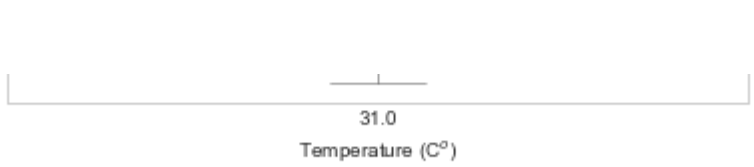

Monday 14:00 (Price 0.672)
Consumption (kWh)
0.4
0.2
0.0
−0.2
Household Type
0
1
2
16.0
17.0
Temperature (C°)


Monday 15:00 (Price 0.0399)
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
Household Type
0
1
2
32.0
Temperature (C°)

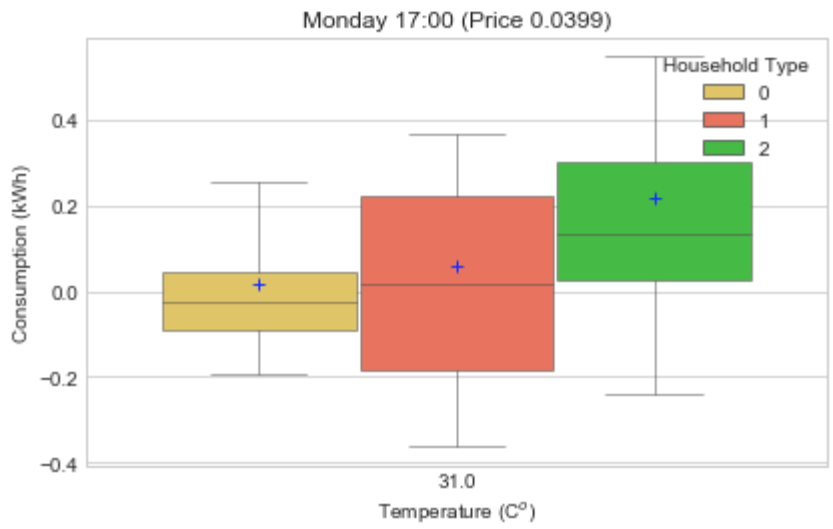

Monday 15:00 (Price 0.672)
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
15.0
17.0
Temperature (C°)

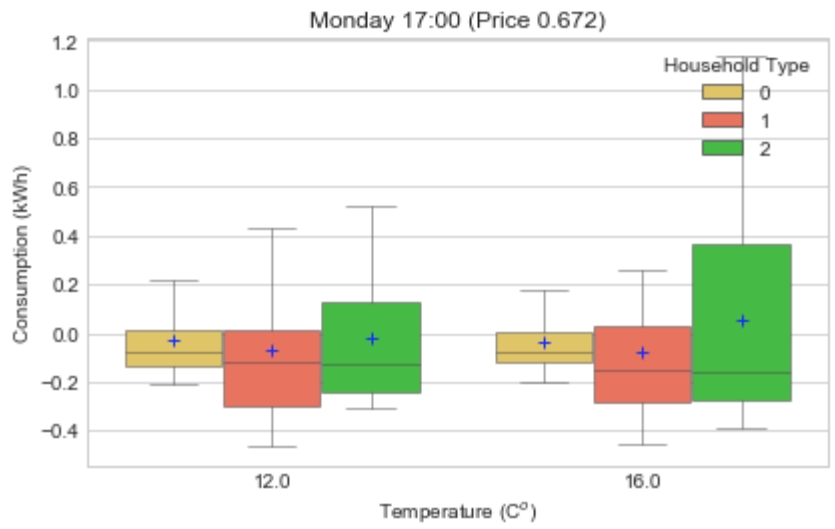

Monday 16:00 (Price 0.0399)
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
Household Type
0
1
2

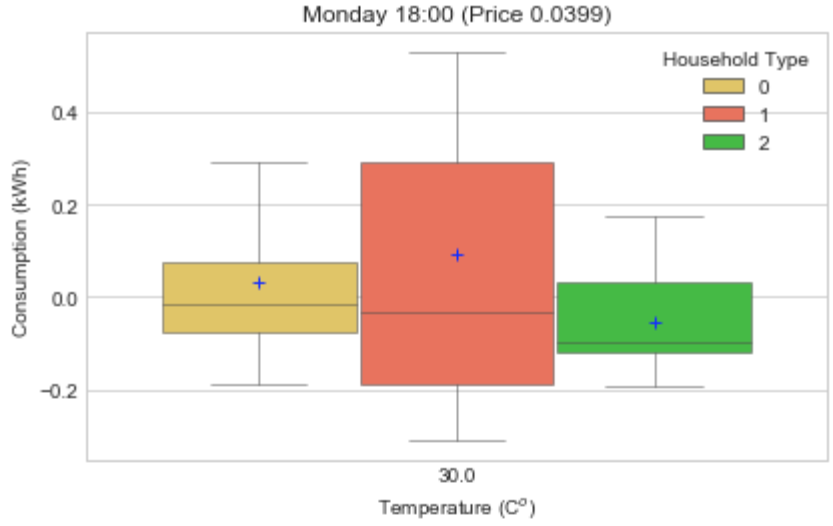

Monday 16:00 (Price 0.672)
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2

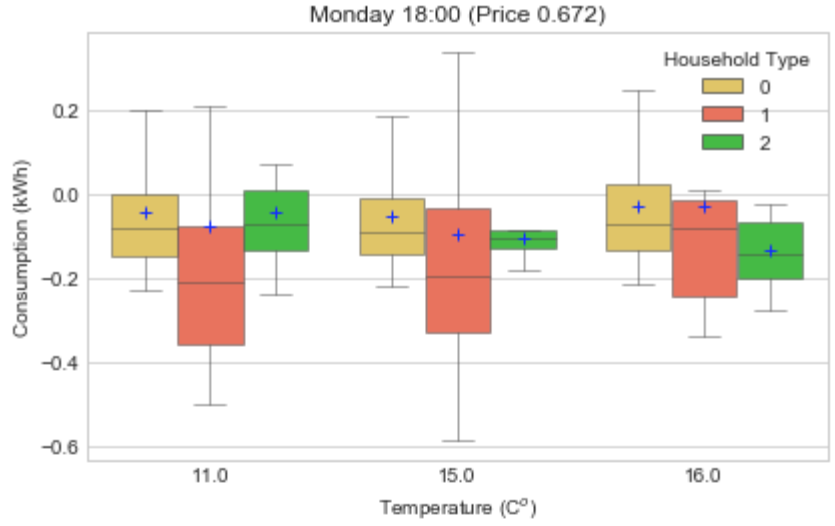

31.0
Temperature (C°)

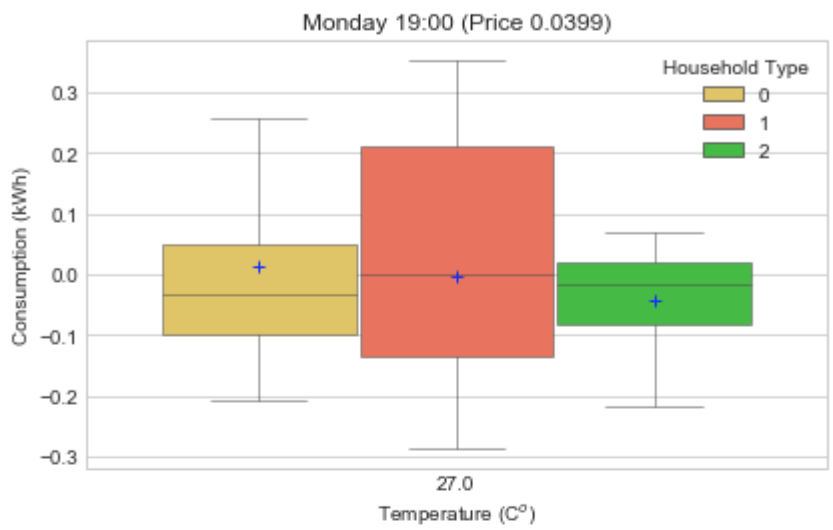

12.0
15.0
16.0
Temperature (C°)

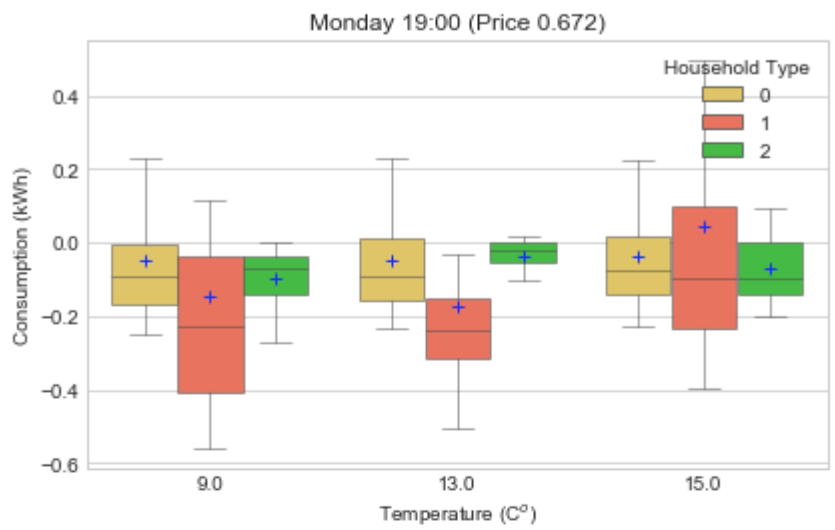

Monday 17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
31.0
Temperature (C°)

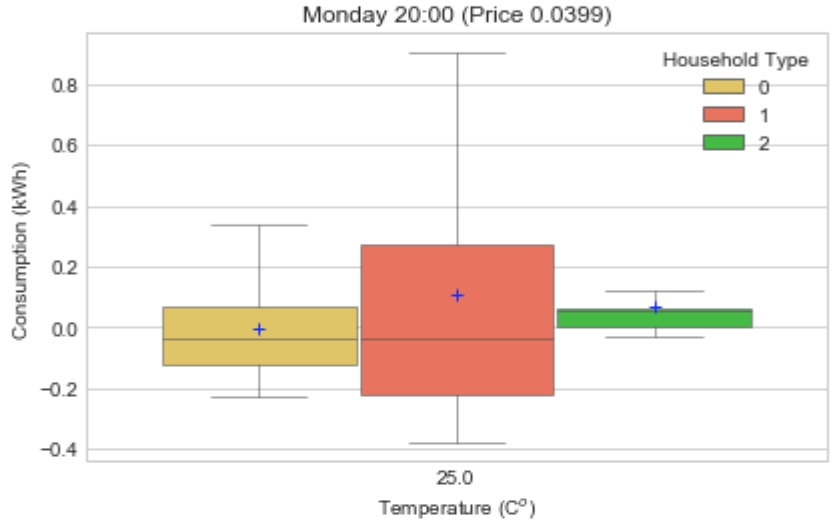

Monday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
12.0
16.0
Temperature (C°)

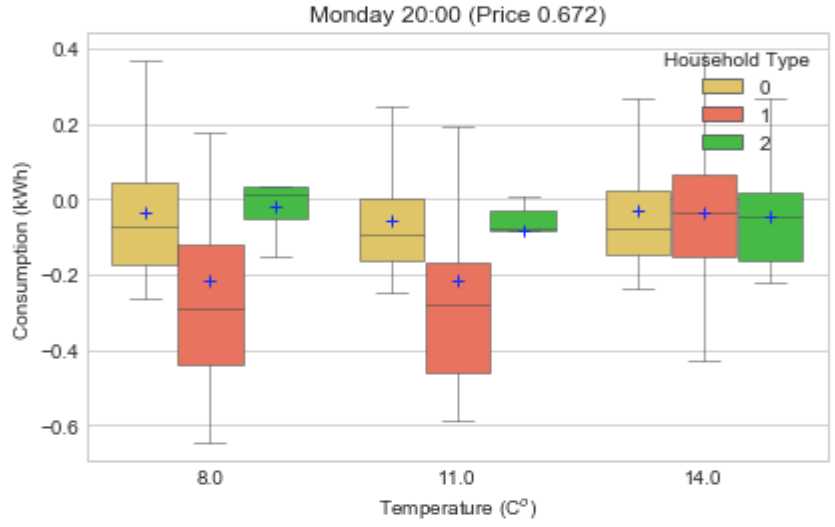

Monday 18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
30.0
Temperature (C°)

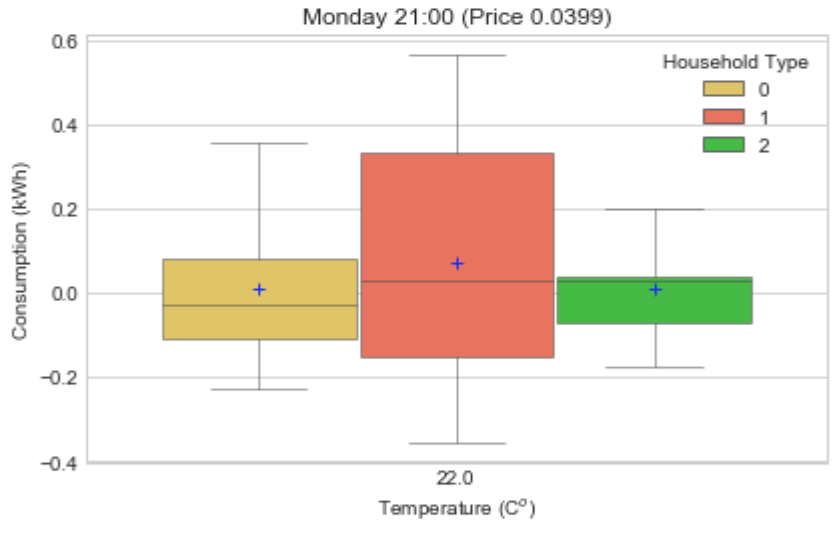

Monday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.0
−0.2
−0.4
−0.6
11.0
15.0
16.0
Temperature (C°)

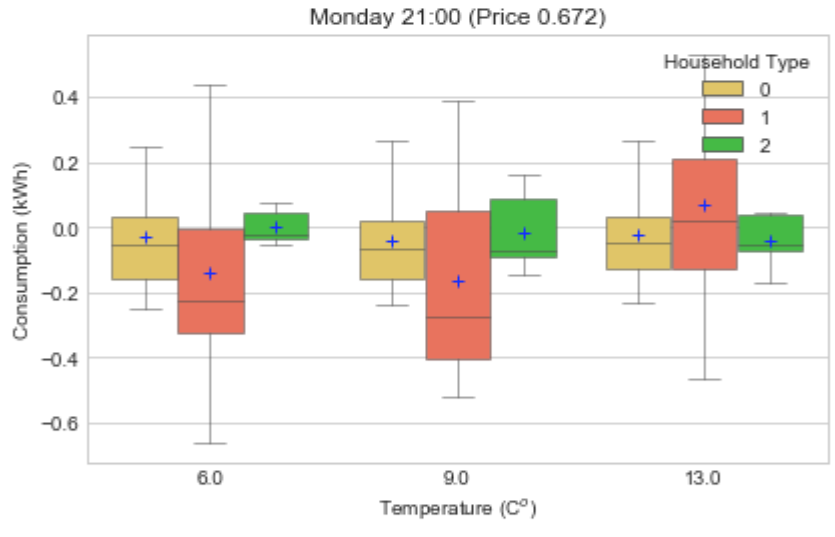

Monday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
27.0
Temperature (C°)


Monday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
−0.6
9.0
13.0
15.0
Temperature (C°)


Monday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
25.0
Temperature (C°)

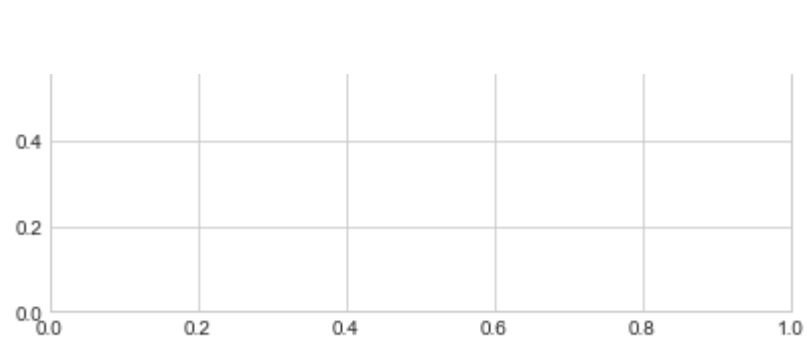

Monday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
−0.6
8.0
11.0
14.0
Temperature (C°)

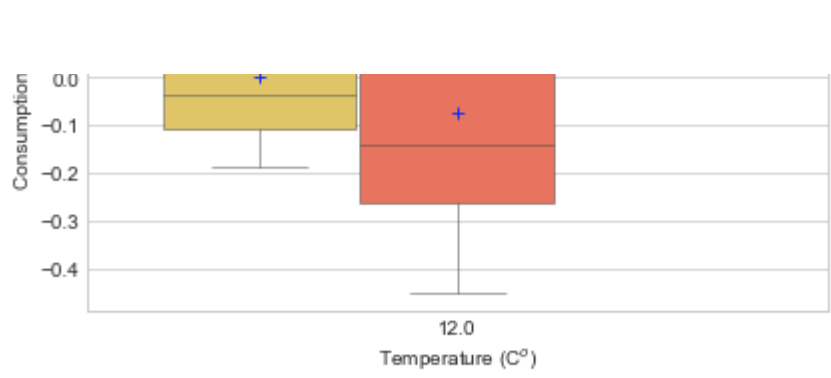

Monday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
22.0
Temperature (C°)

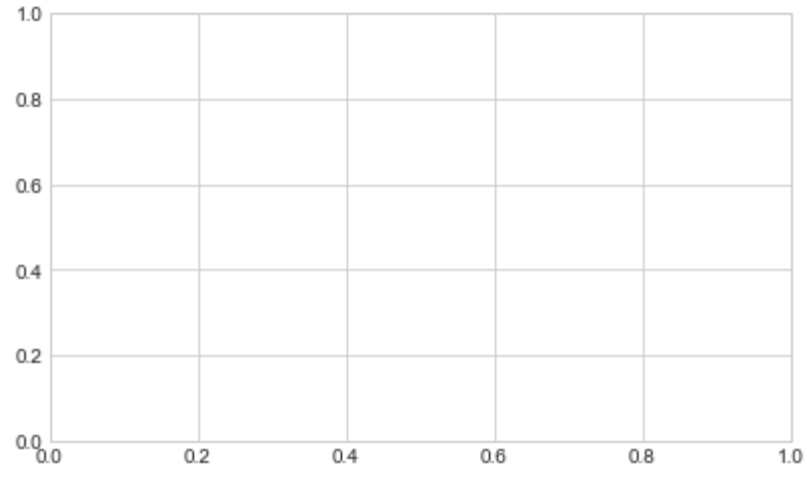

Monday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
−0.6
6.0
9.0
13.0
Temperature (C°)

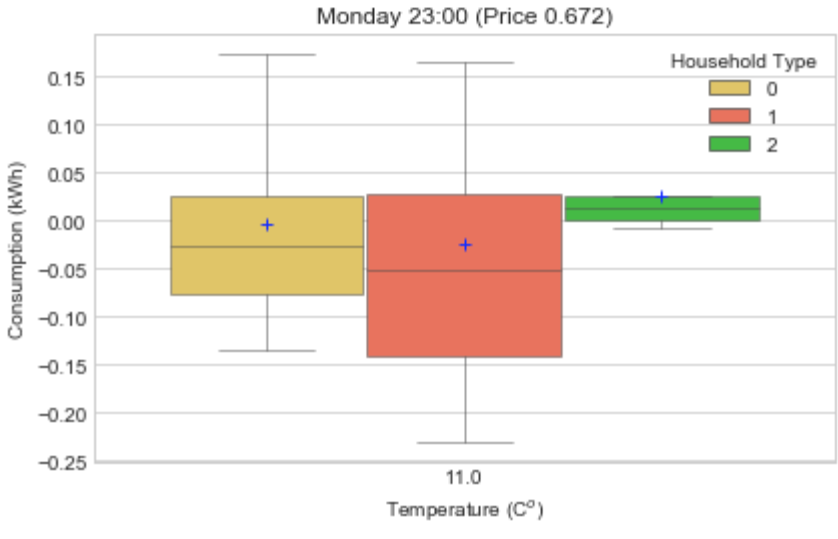
1.0
0.8
0.6

Monday 22:00 (Price 0.672)
Household Type
0
1
2
0.4
0.3
0.2
0.1

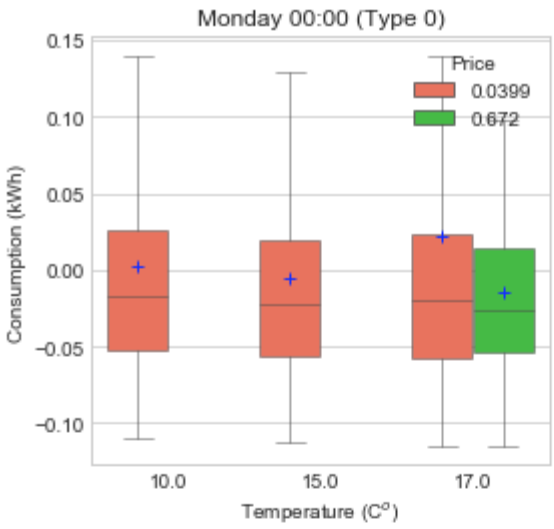

0.0
0.2
0.4
0.6
0.8
1.0

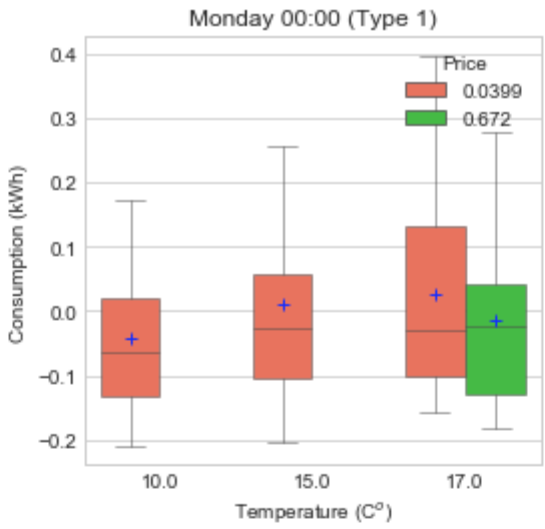

Consumption
0.0
−0.1
−0.2
−0.3
−0.4
12.0
Temperature (C°)

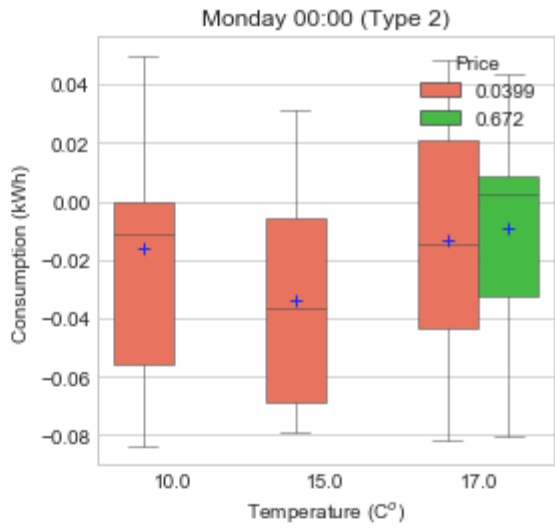

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

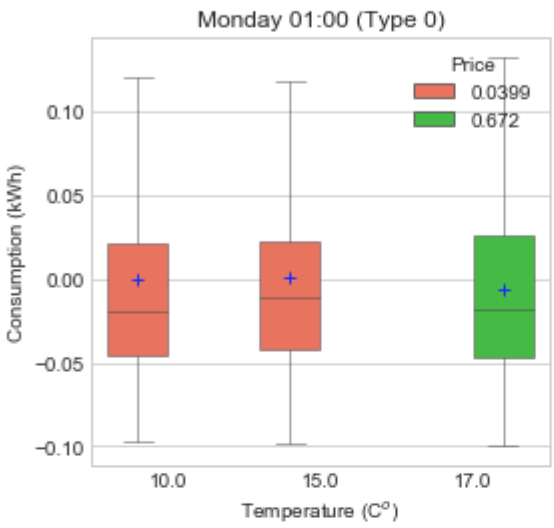

Monday 23:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
11.0
Temperature (C°)

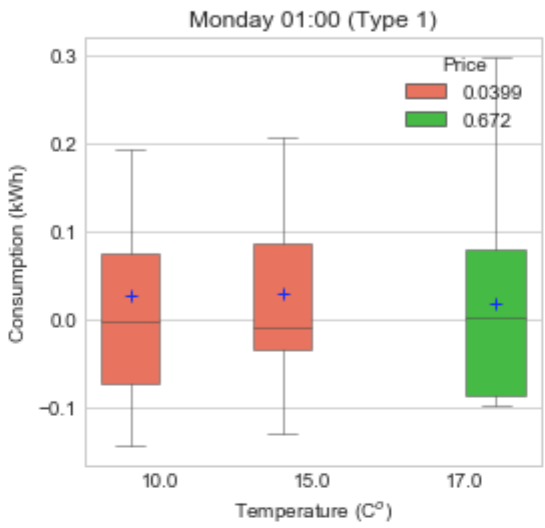
In [62]:
```
# Price comparison version
# Monday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 0 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

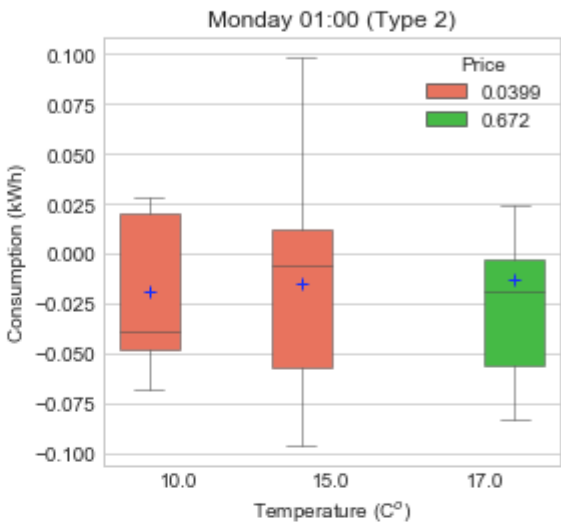

Monday 00:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

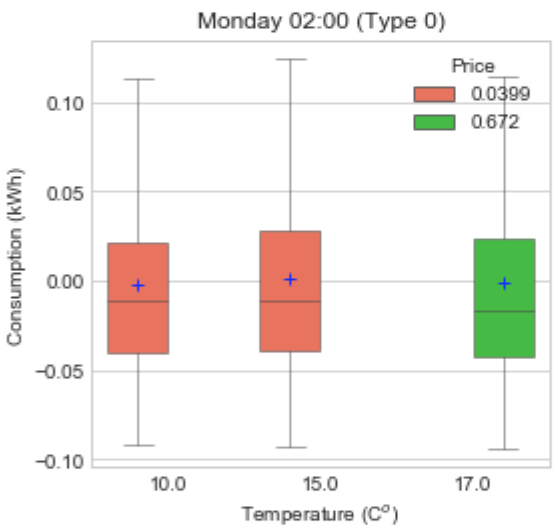

Monday 00:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

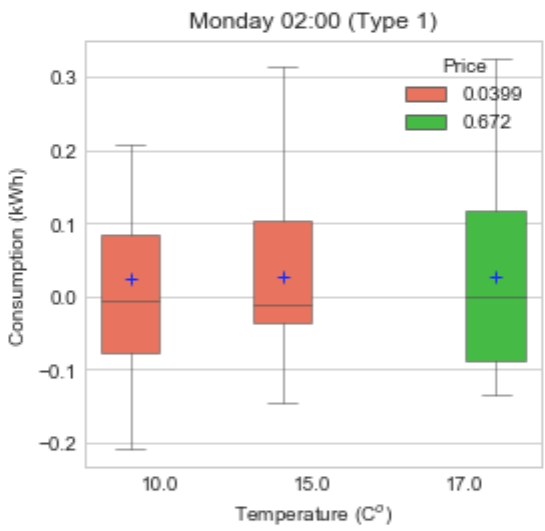

Monday 00:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

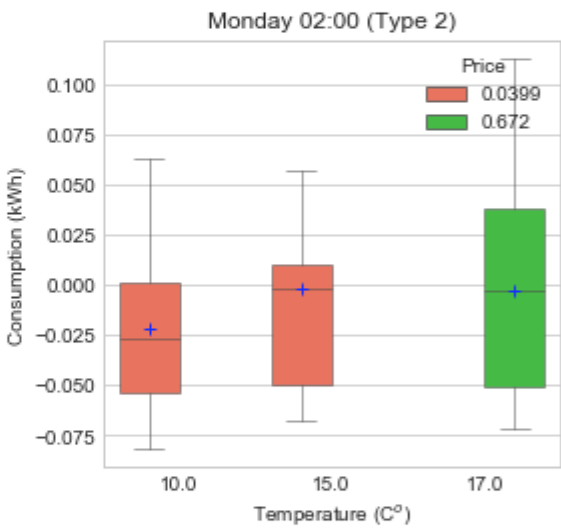

Monday 01:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

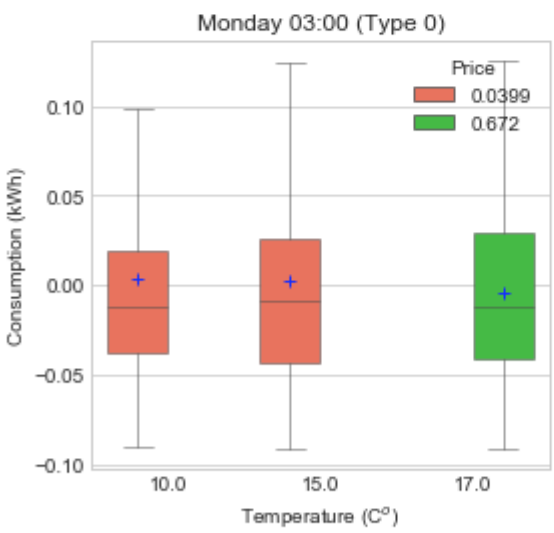

Monday 01:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

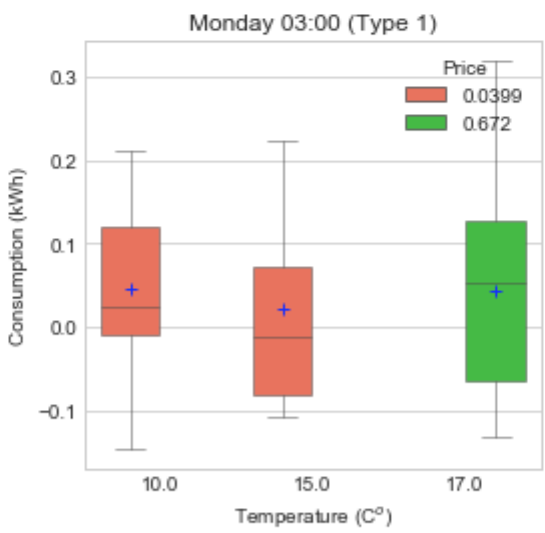

Monday 01:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

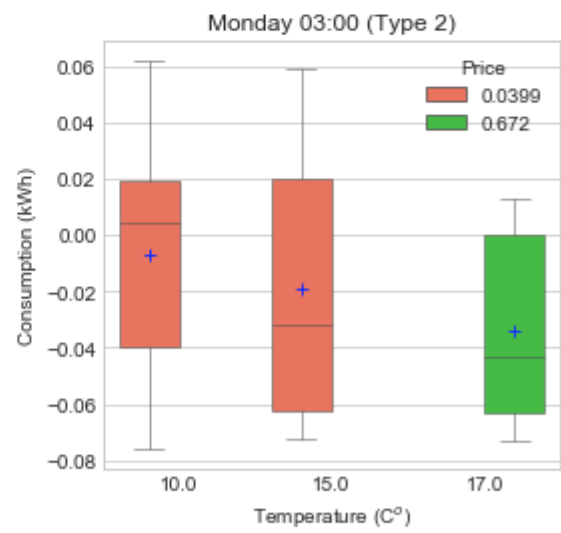

Monday 02:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

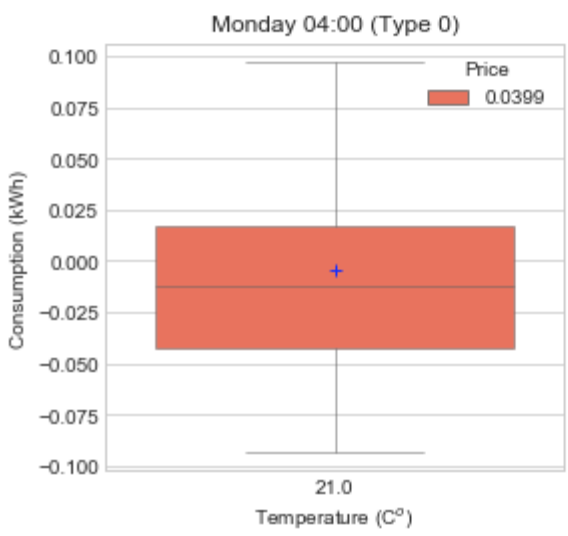

Monday 02:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

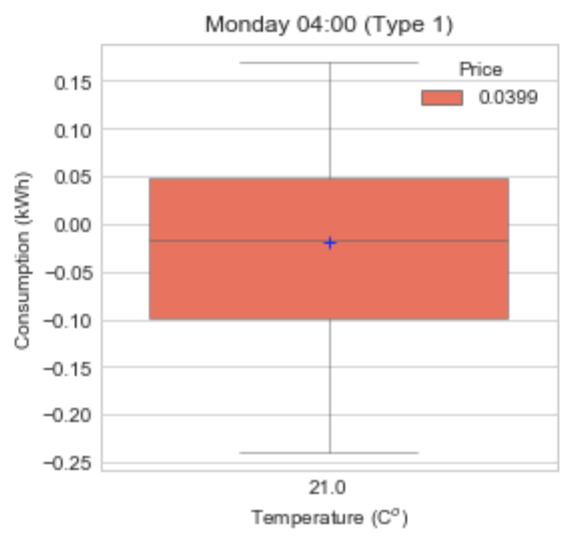

Monday 02:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

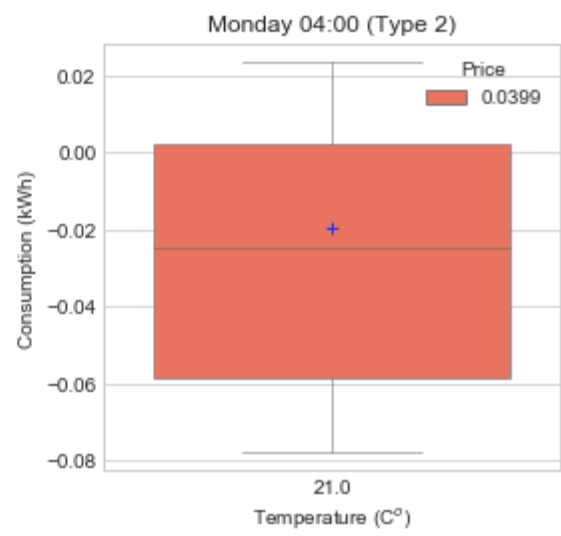

Monday 03:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

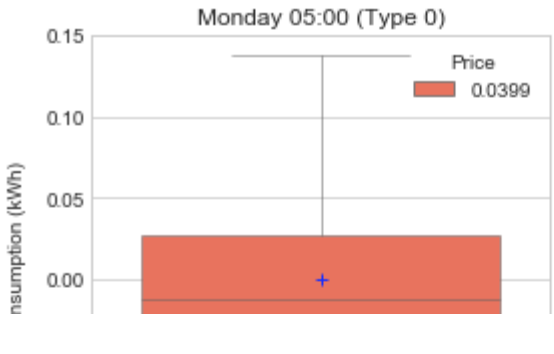

Monday 03:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

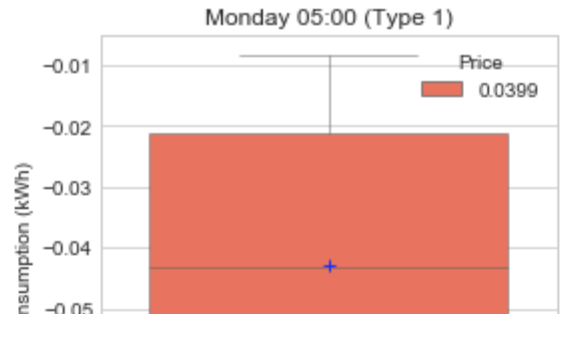

Monday 03:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
17.0

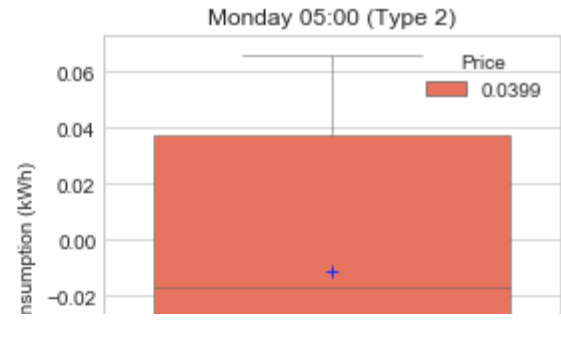

Monday 04:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
21.0

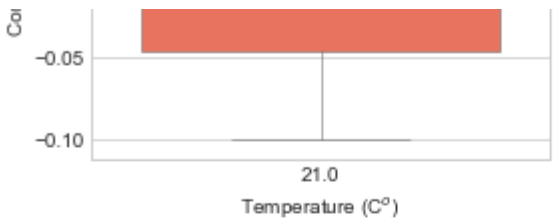

Monday 04:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
21.0

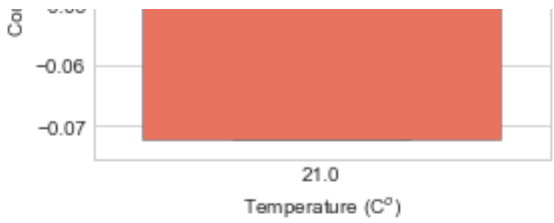

Monday 04:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
21.0

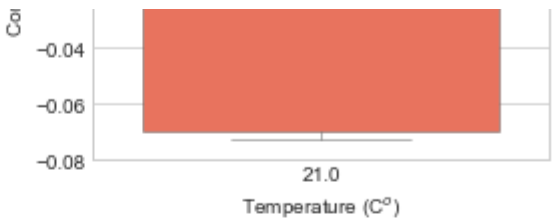

Monday 05:00 (Type 0)
Price
0.0399
Consumption (kWh)

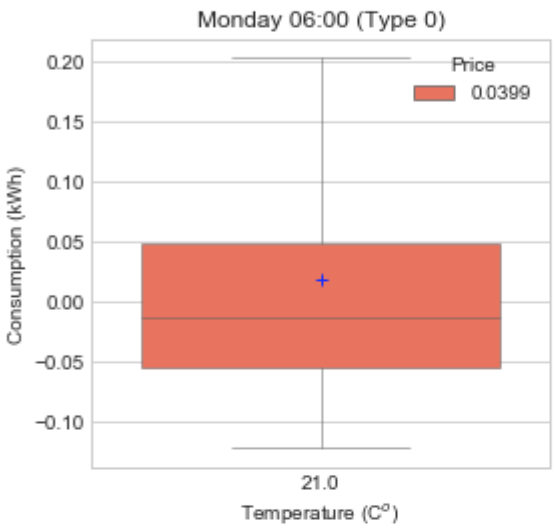

Monday 05:00 (Type 1)
Price
0.0399
Consumption (kWh)

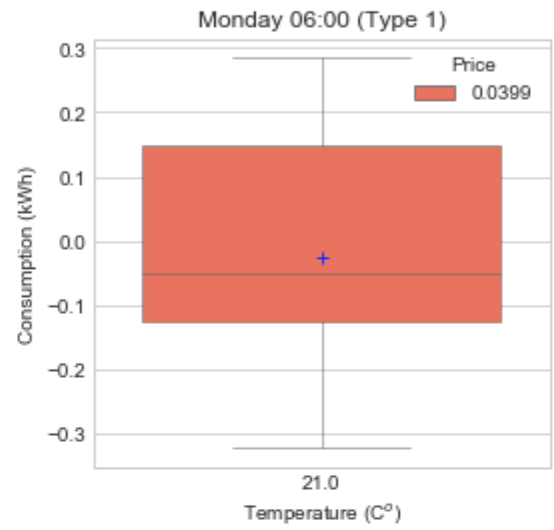

Monday 05:00 (Type 2)
Price
0.0399
Consumption (kWh)

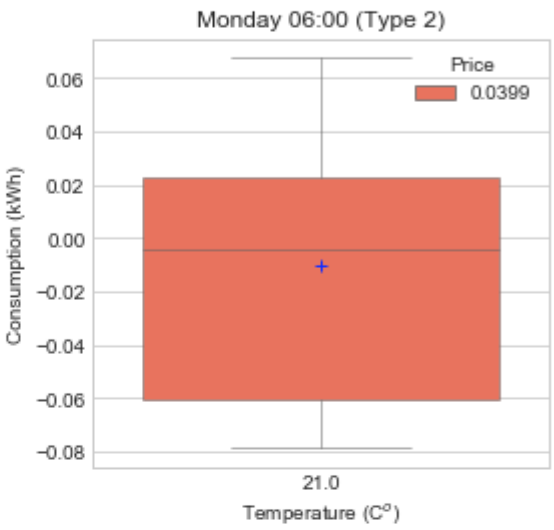

−0.05
−0.10
21.0
Temperature (C°)

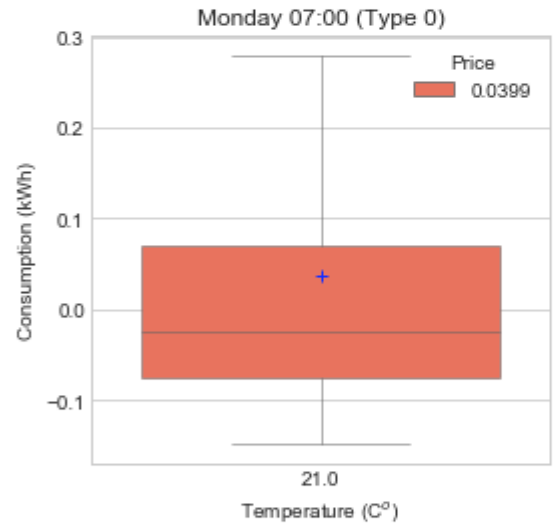

−0.06
−0.07
21.0
Temperature (C°)

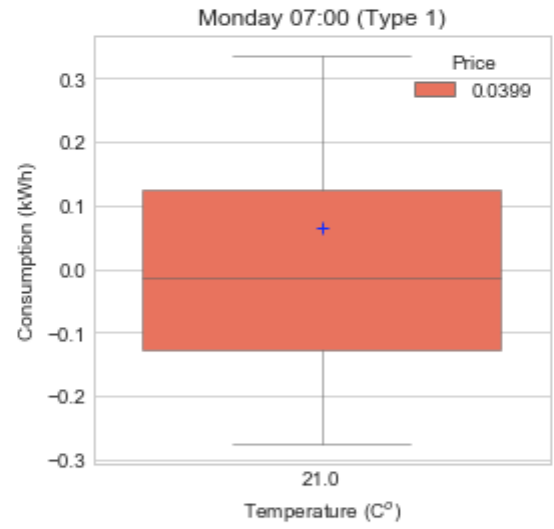

−0.04
−0.06
−0.08
21.0
Temperature (C°)

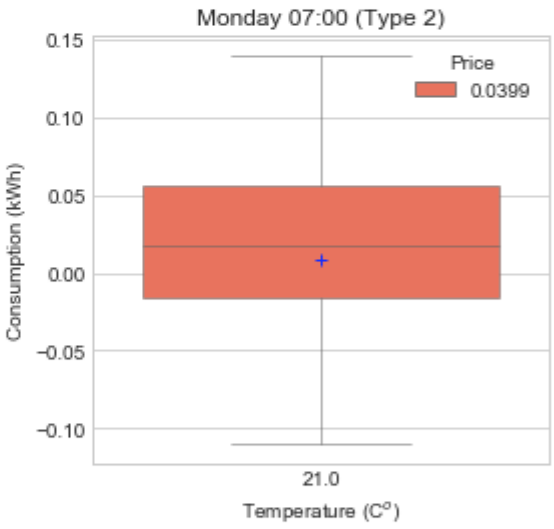

Monday 06:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
21.0
Temperature (C°)

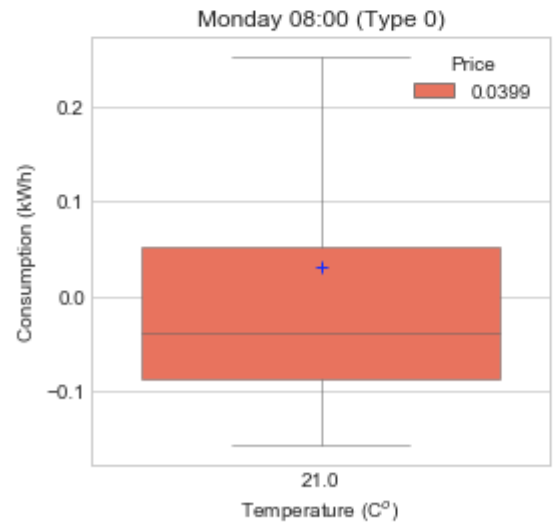

Monday 06:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
21.0
Temperature (C°)

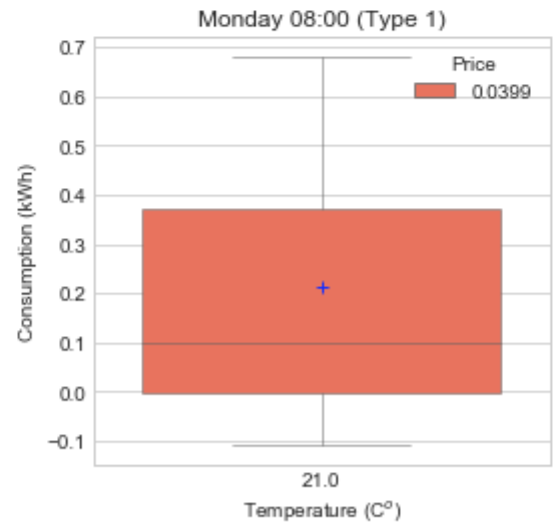

Monday 06:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
21.0
Temperature (C°)

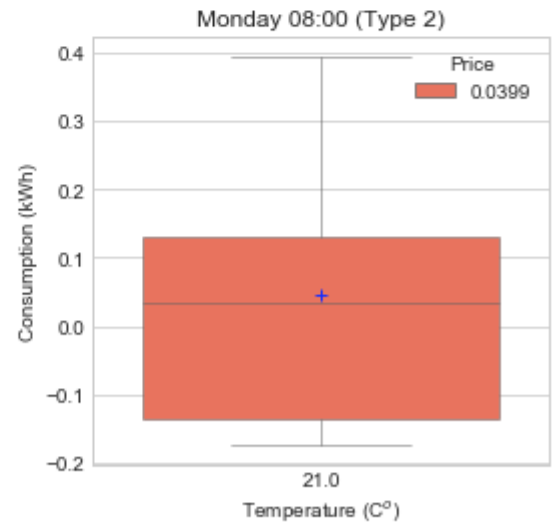

Monday 07:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
21.0
Temperature (C°)

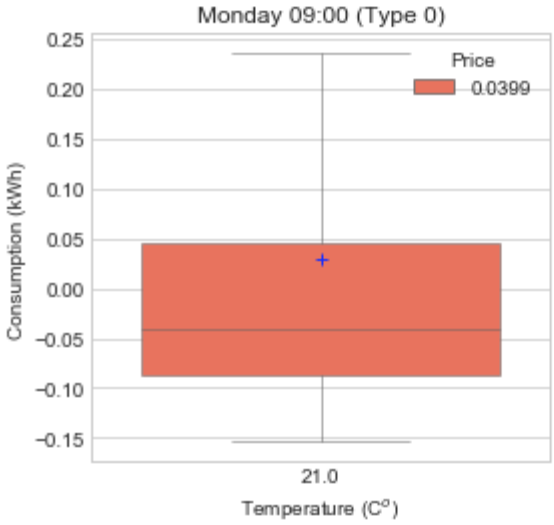

Monday 07:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
21.0
Temperature (C°)

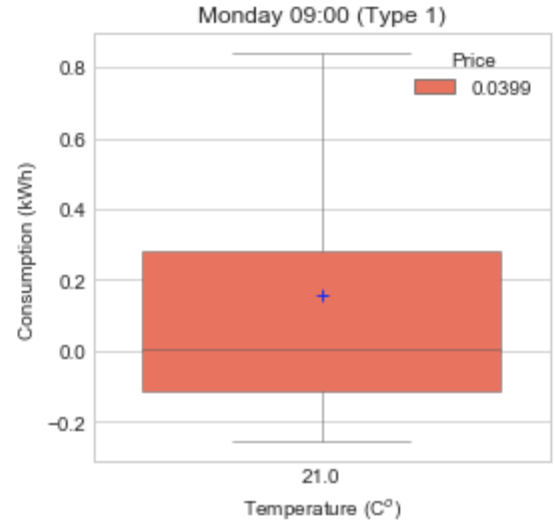

Monday 07:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
21.0
Temperature (C°)

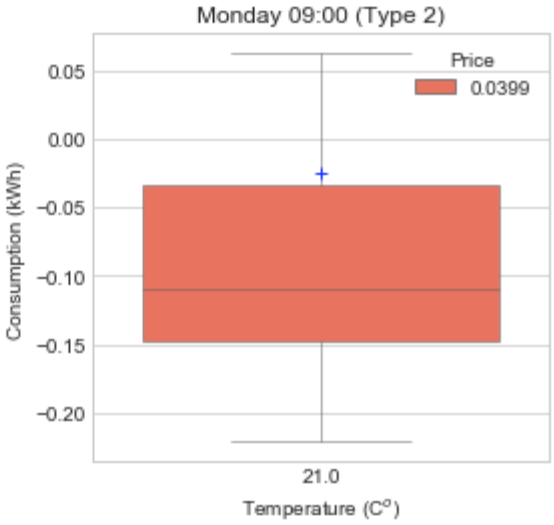

Monday 08:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
21.0
Temperature (C°)

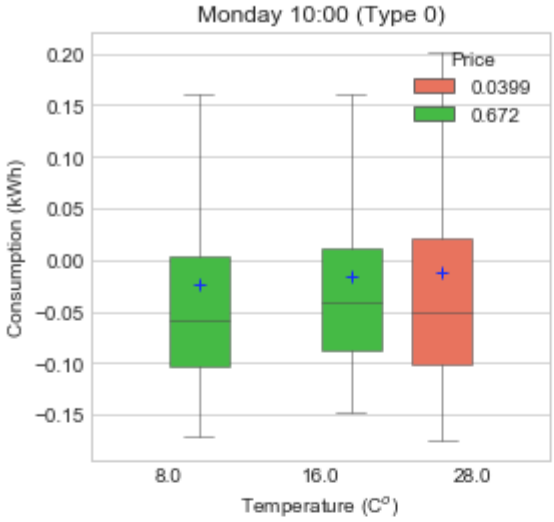

Monday 08:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
21.0
Temperature (C°)

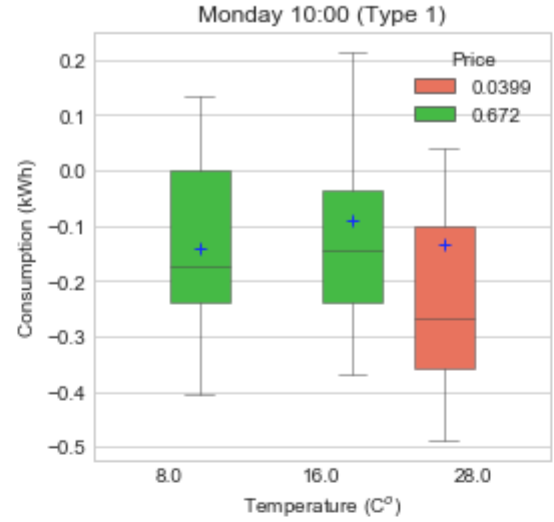

Monday 08:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
21.0
Temperature (C°)

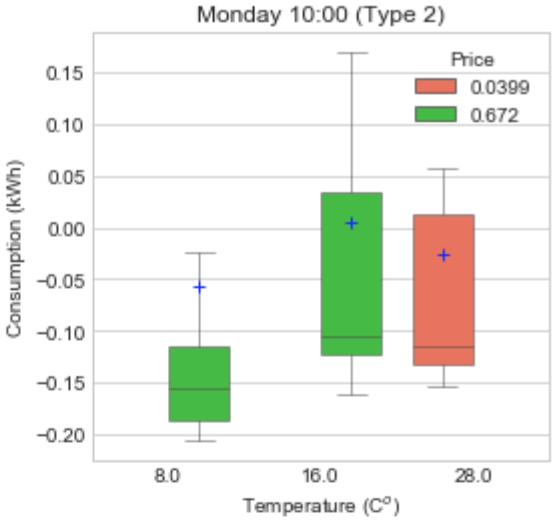

Monday 09:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
21.0
Temperature (C°)


Monday 09:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
21.0
Temperature (C°)


Monday 09:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.05
0.00
−0.05
−0.10
−0.15
−0.20
21.0
Temperature (C°)


Monday 10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
8.0
16.0
28.0
Temperature (C°)

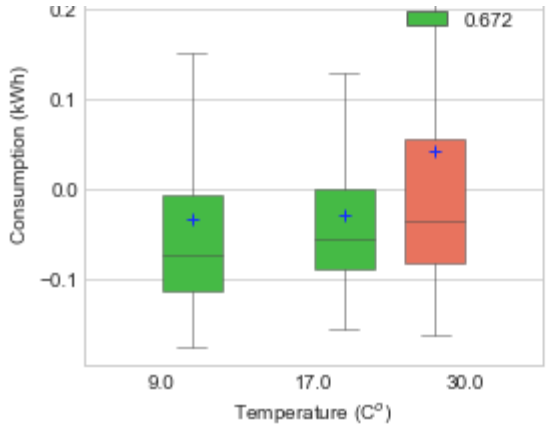

Monday 10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
8.0
16.0
28.0
Temperature (C°)

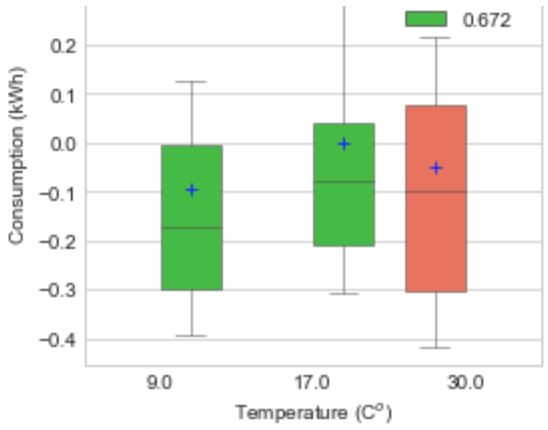

Monday 10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
8.0
16.0
28.0
Temperature (C°)

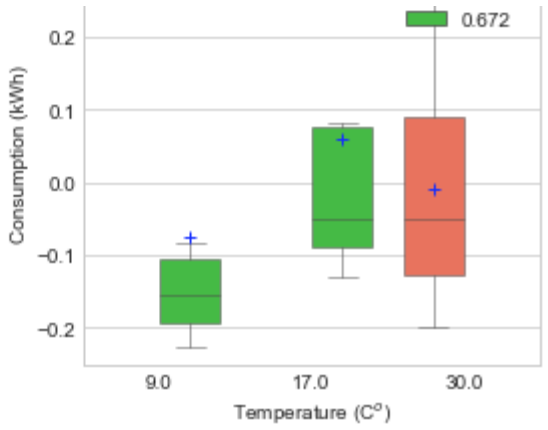
Monday 11:00 (Type 0)
Price
0.0399

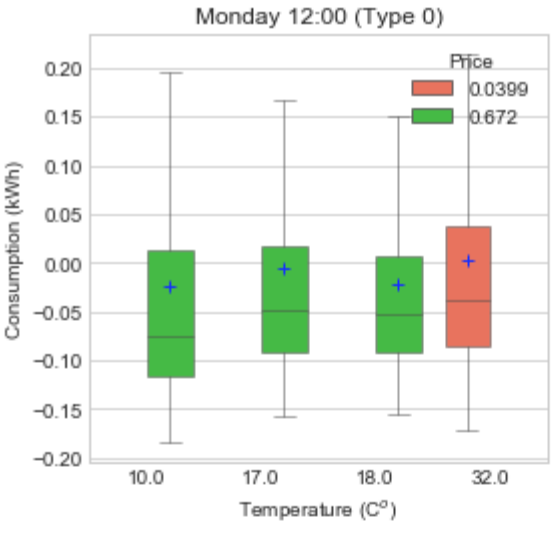
Monday 11:00 (Type 1)
Price
0.0399
0.4
0.3

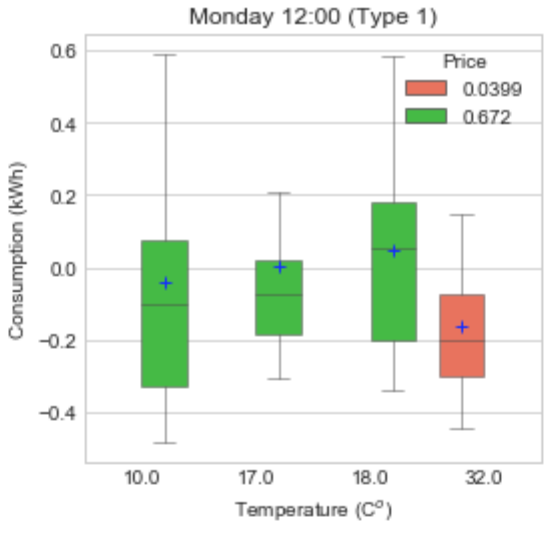
Monday 11:00 (Type 2)
Price
0.0399
0.3

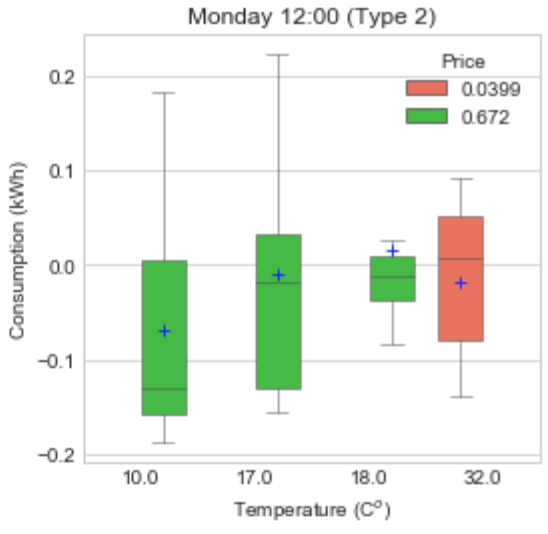

0.672
Consumption (kWh)
Temperature (C°)
9.0
17.0
30.0

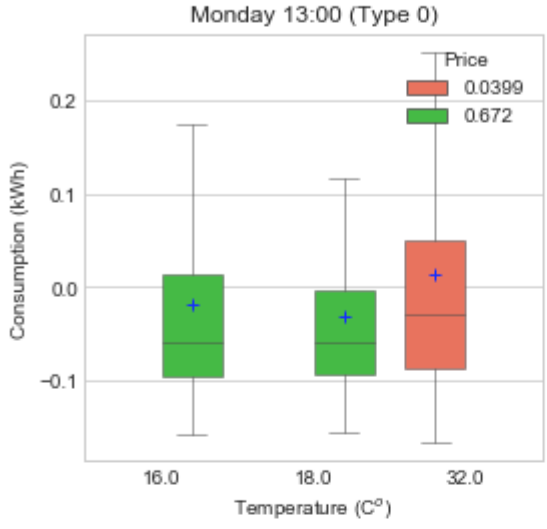

0.672
Consumption (kWh)
Temperature (C°)
9.0
17.0
30.0

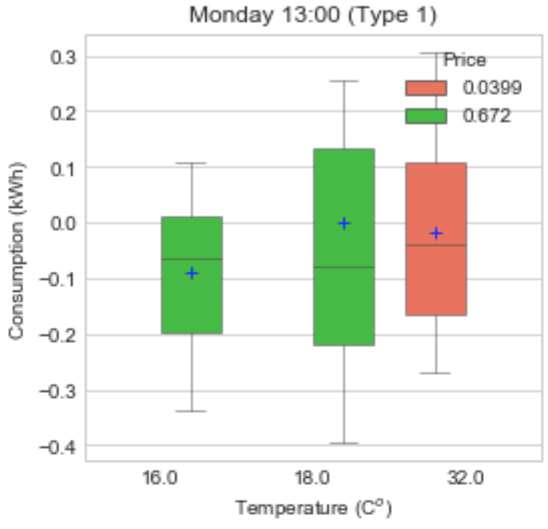

0.672
Consumption (kWh)
Temperature (C°)
9.0
17.0
30.0

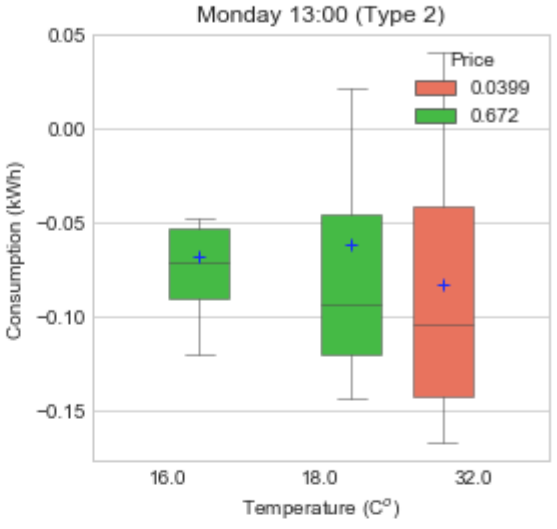

Monday 12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
18.0
32.0

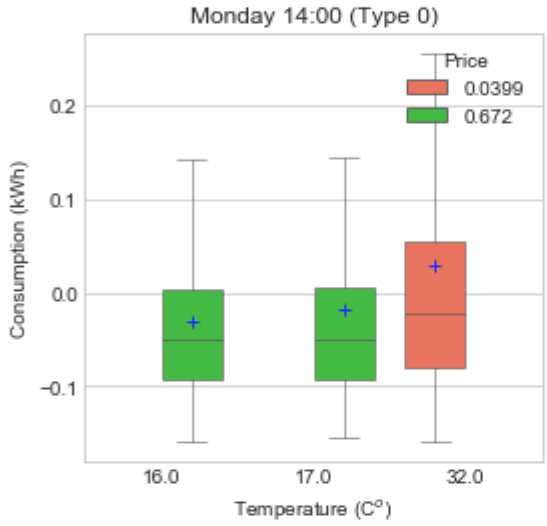

Monday 12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
18.0
32.0

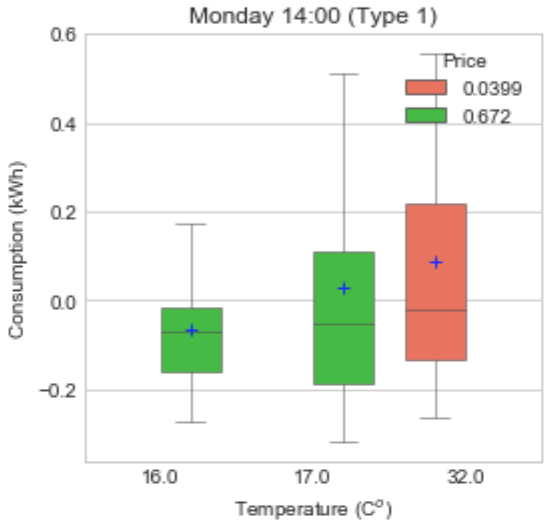

Monday 12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
18.0
32.0

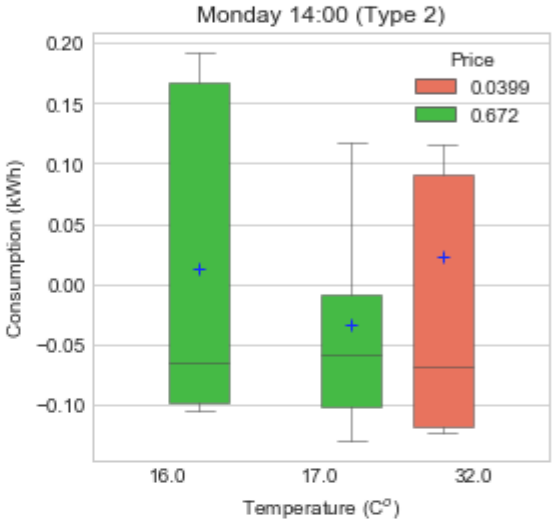

Monday 13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
18.0
32.0

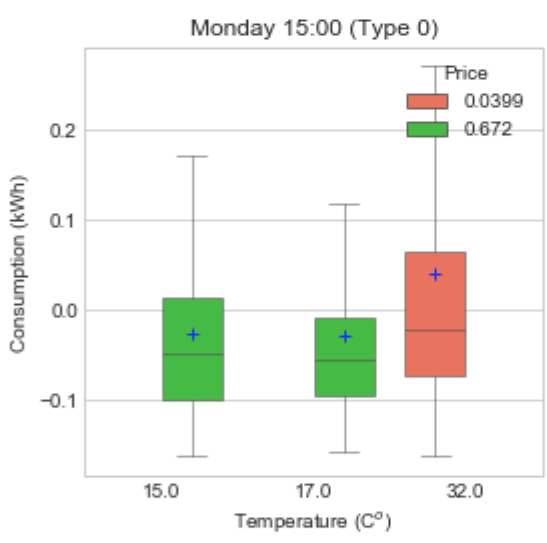

Monday 13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
18.0
32.0

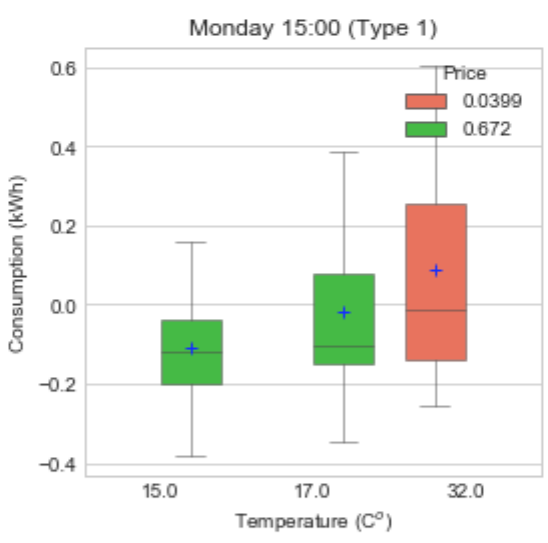

Monday 13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
18.0
32.0

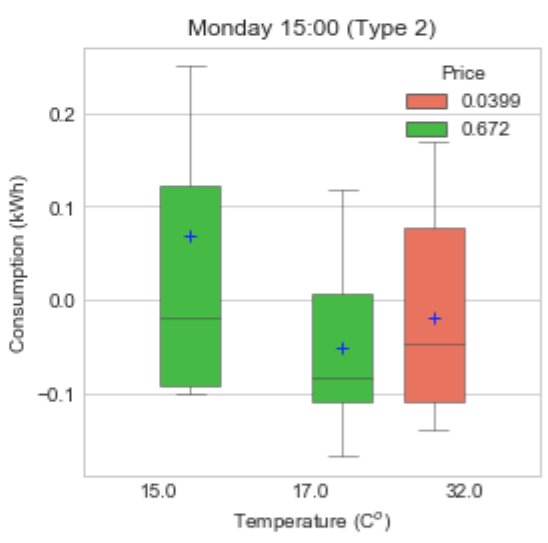

Monday 14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
17.0
32.0

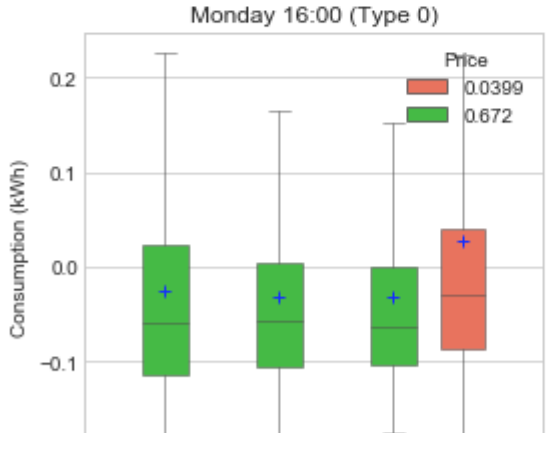

Monday 14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
17.0
32.0

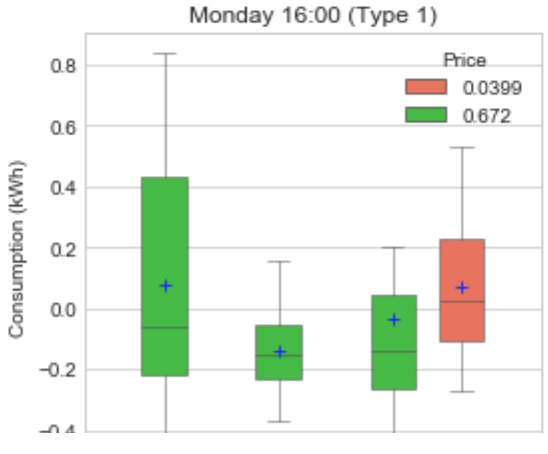

Monday 14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
16.0
17.0
32.0

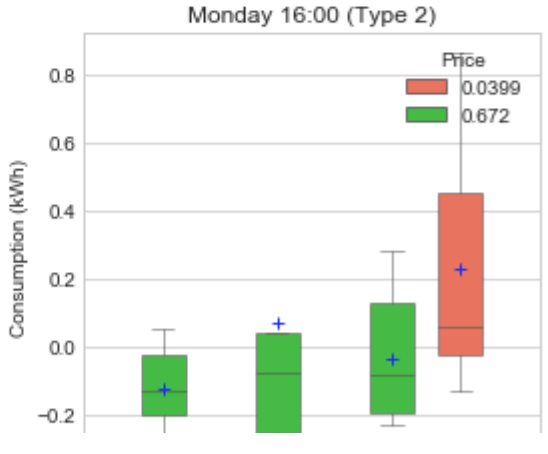

Monday 15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
32.0

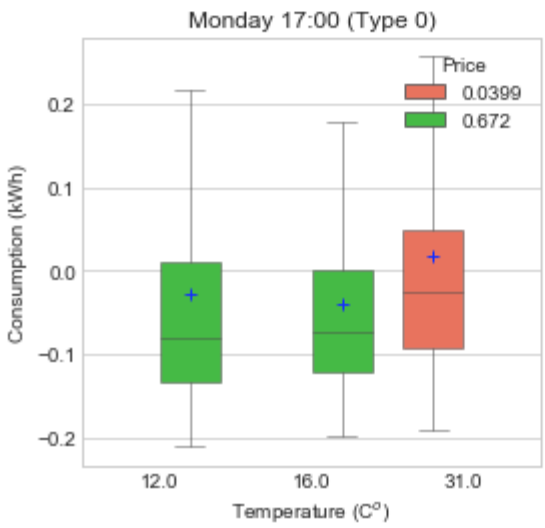

Monday 15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
32.0

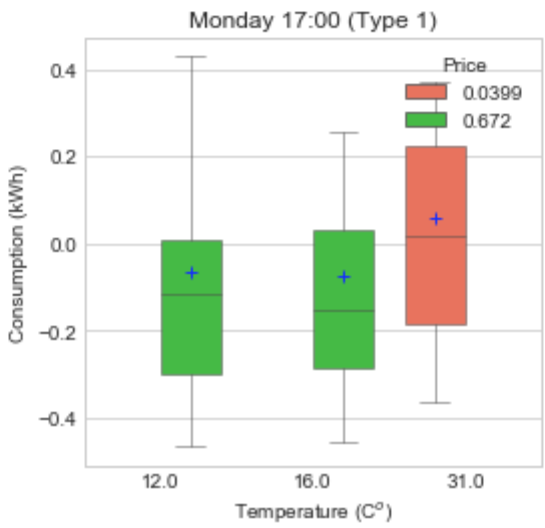

Monday 15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
32.0

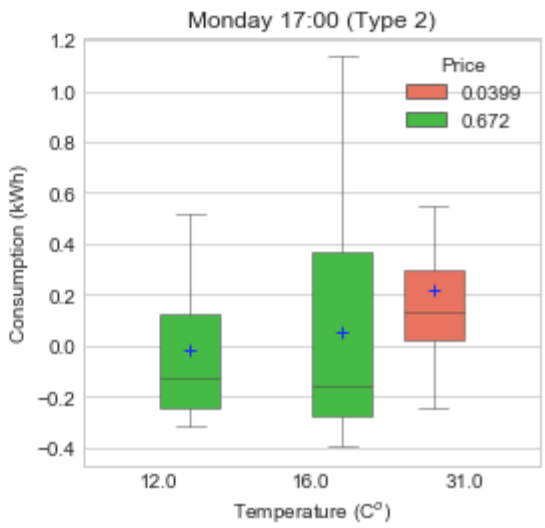

Monday 16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

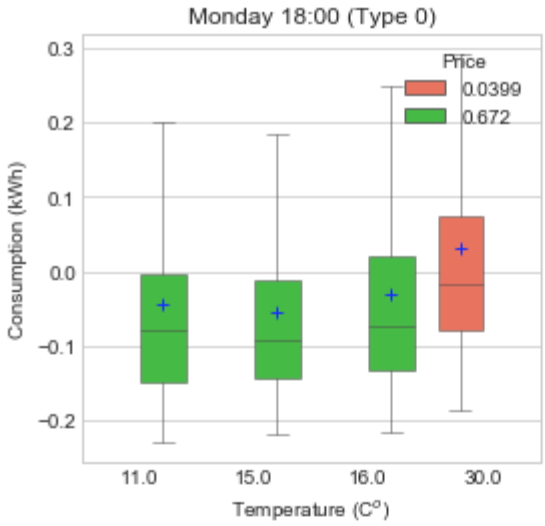

Monday 16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

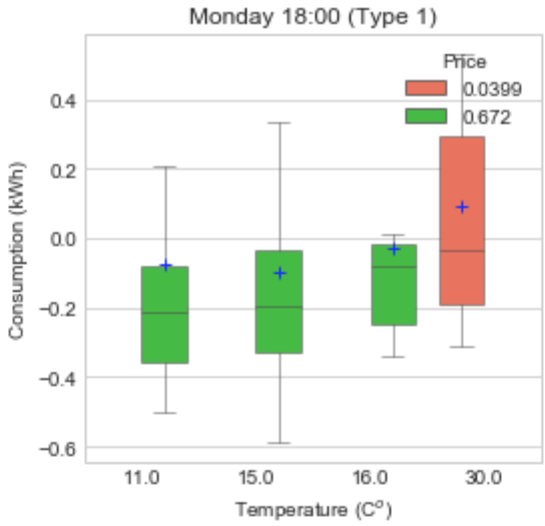

Monday 16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

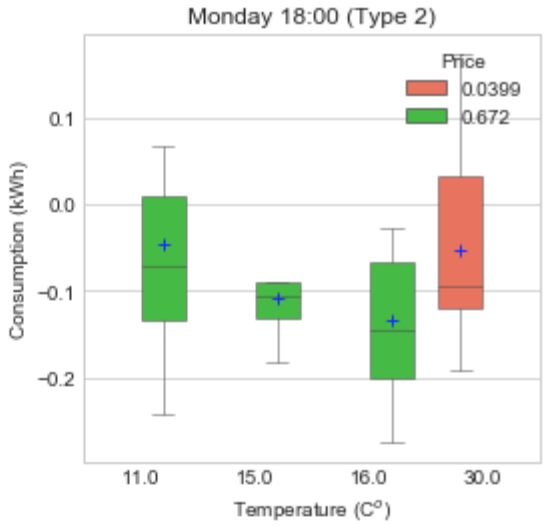

Monday 17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

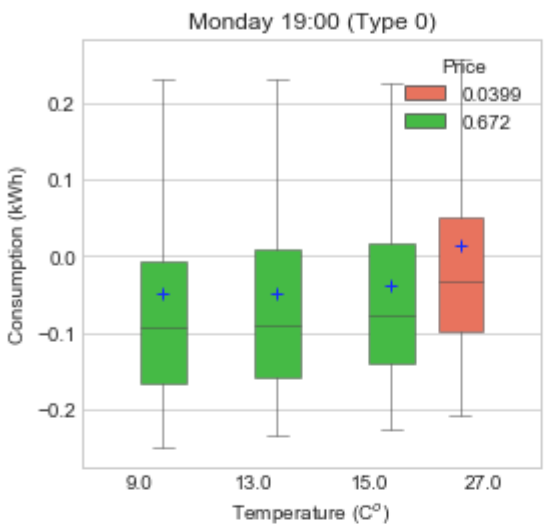

Monday 17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

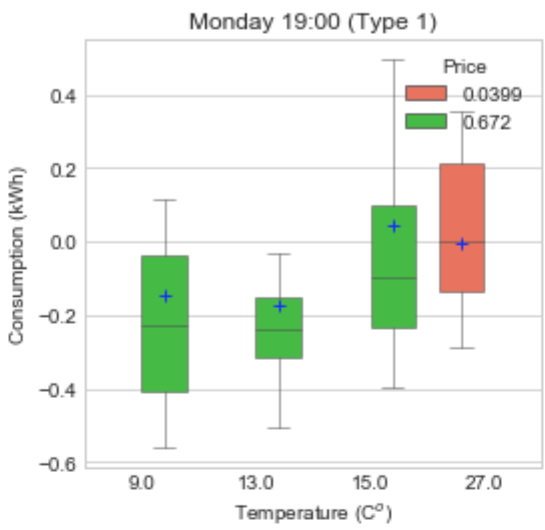

Monday 17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

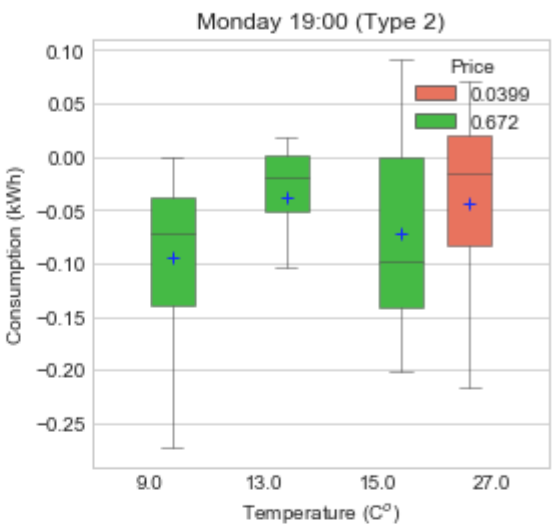

Monday 18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

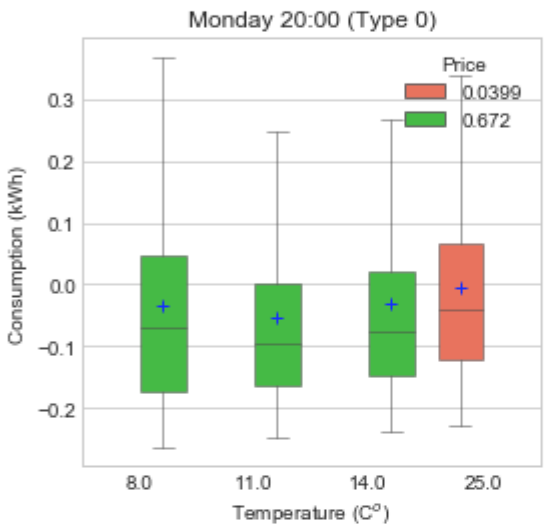

Monday 18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

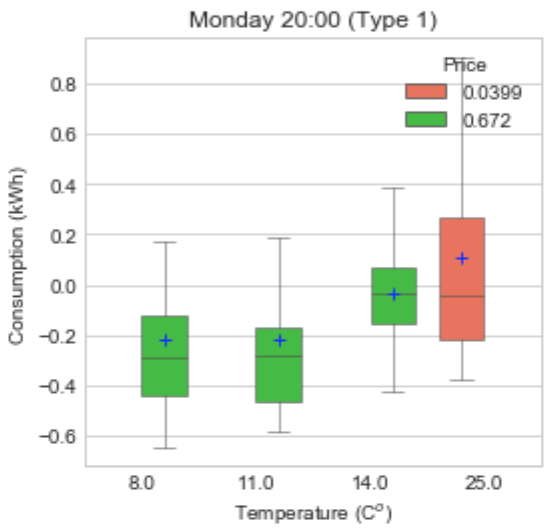

Monday 18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

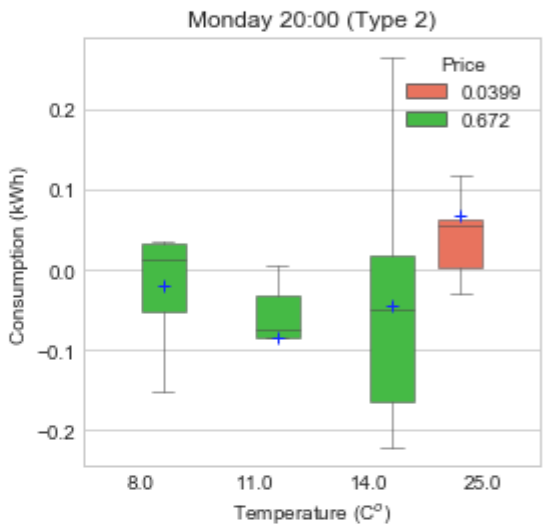

Monday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

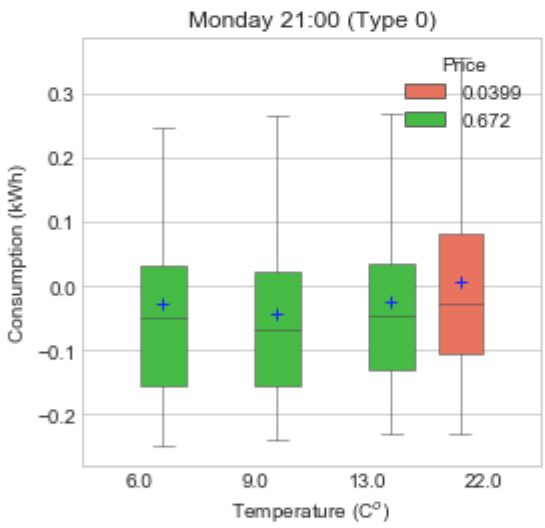

Monday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

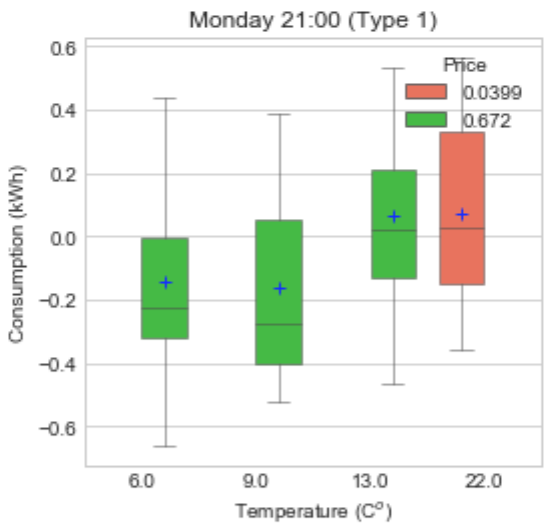

Monday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

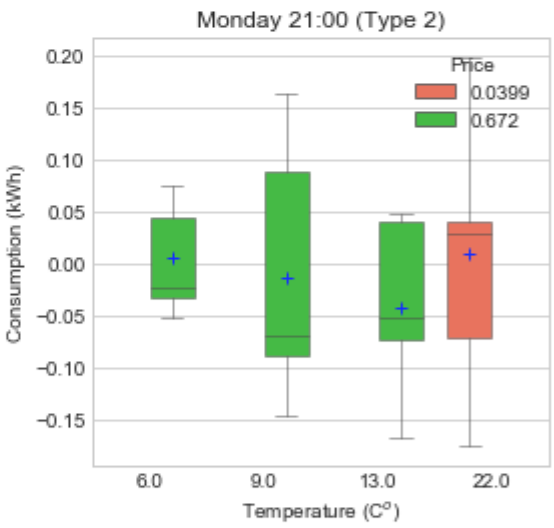

Monday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

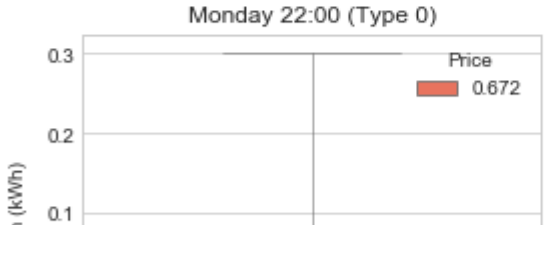

Monday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

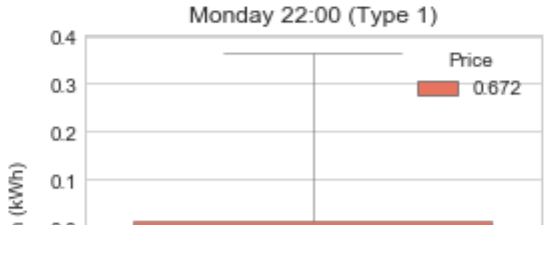

Monday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

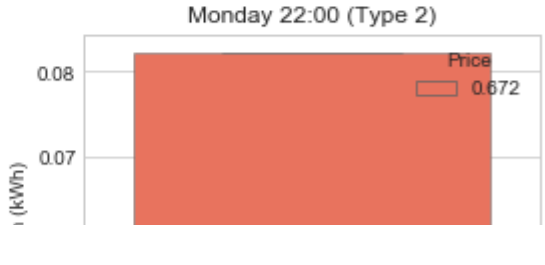

Monday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

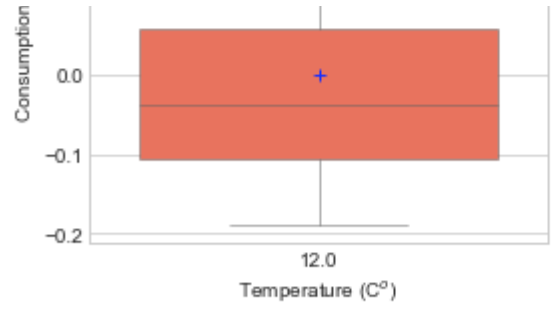

Monday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

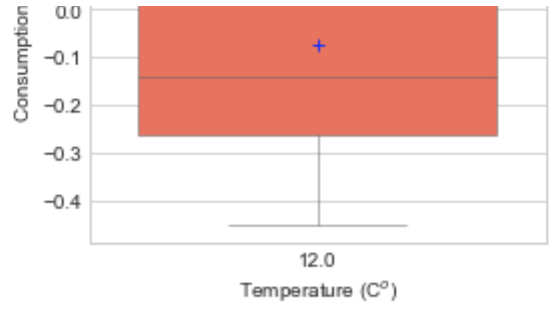

Monday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

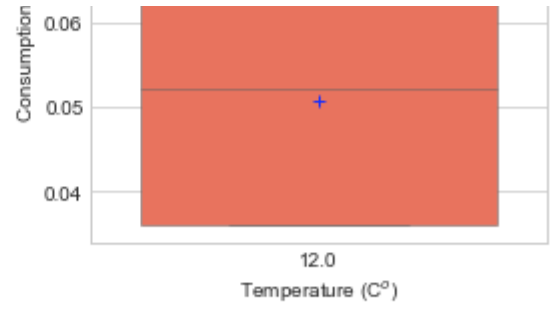

Monday 22:00 (Type 0)
Price
0.672

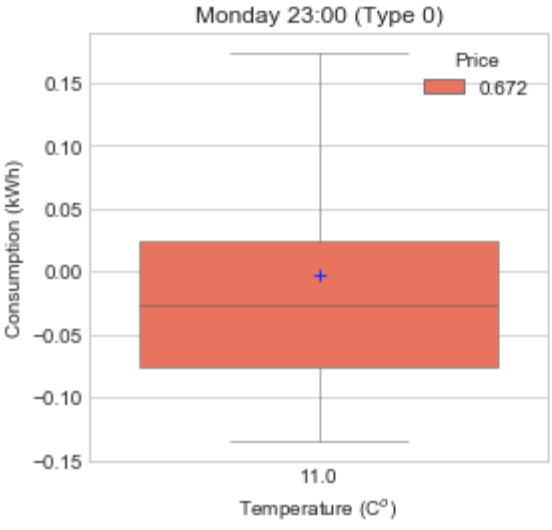

Monday 22:00 (Type 1)
Price
0.672

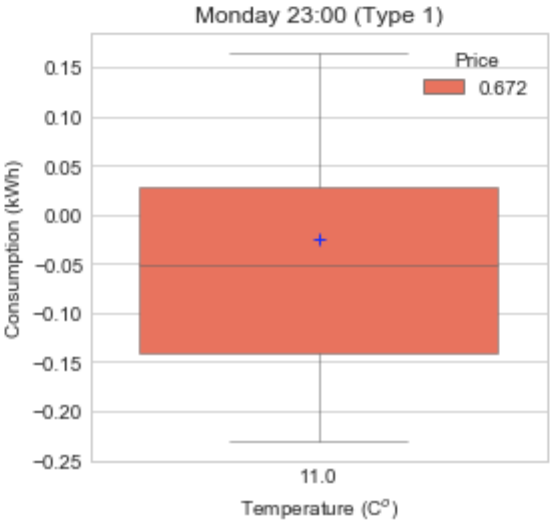

Monday 22:00 (Type 2)
Price
0.672

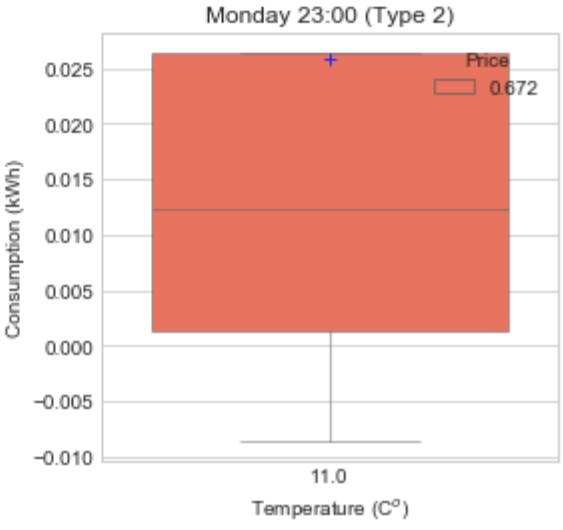

Consumption
0.0
−0.1
−0.2
12.0
Temperature (C°)


Consumption
0.0
−0.1
−0.2
−0.3
−0.4
12.0
Temperature (C°)

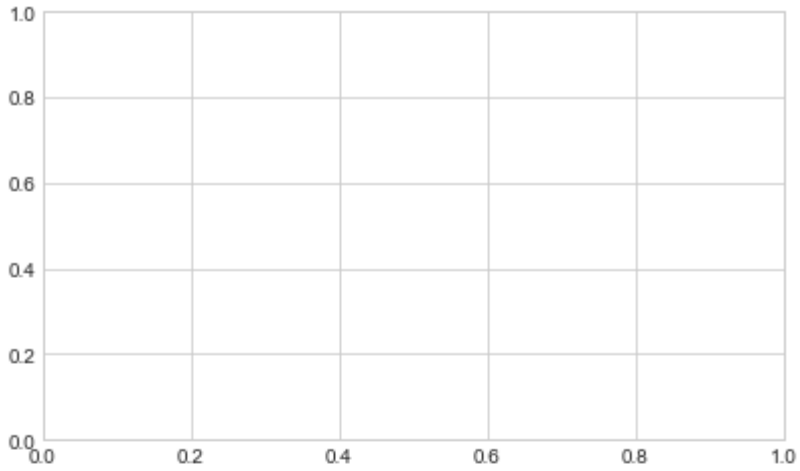

Consumption
0.06
0.05
0.04
12.0
Temperature (C°)

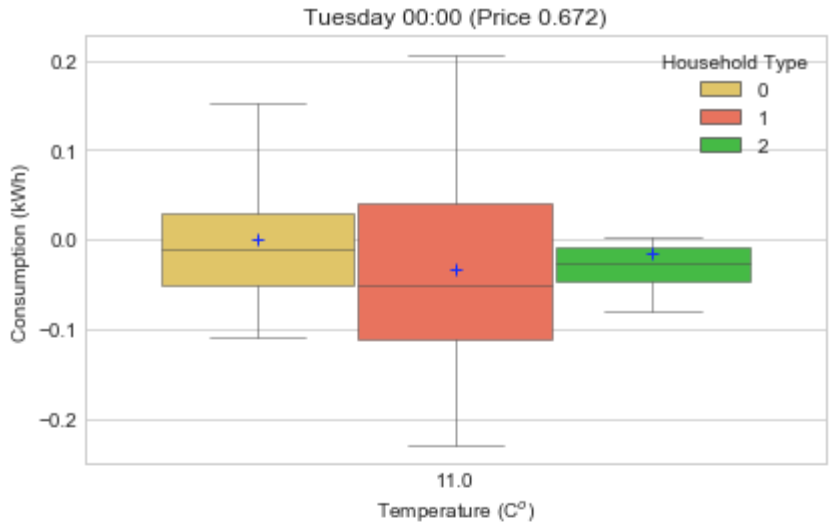

Monday 23:00 (Type 0)
Price
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
11.0
Temperature (C°)

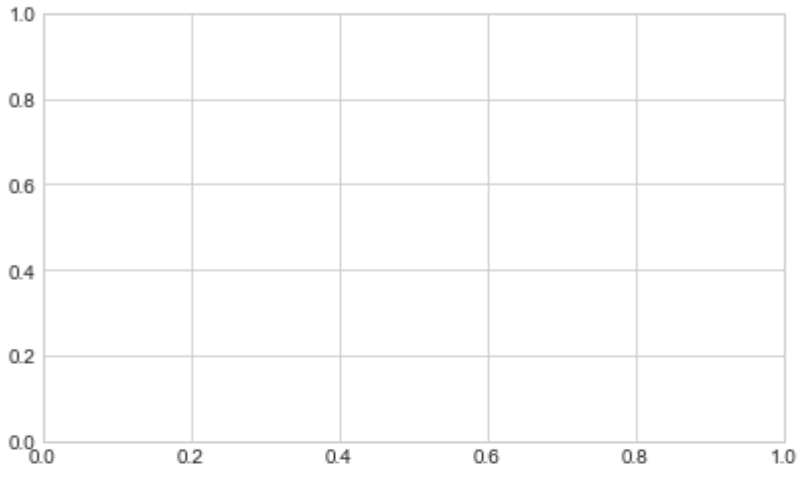

Monday 23:00 (Type 1)
Price
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
11.0
Temperature (C°)

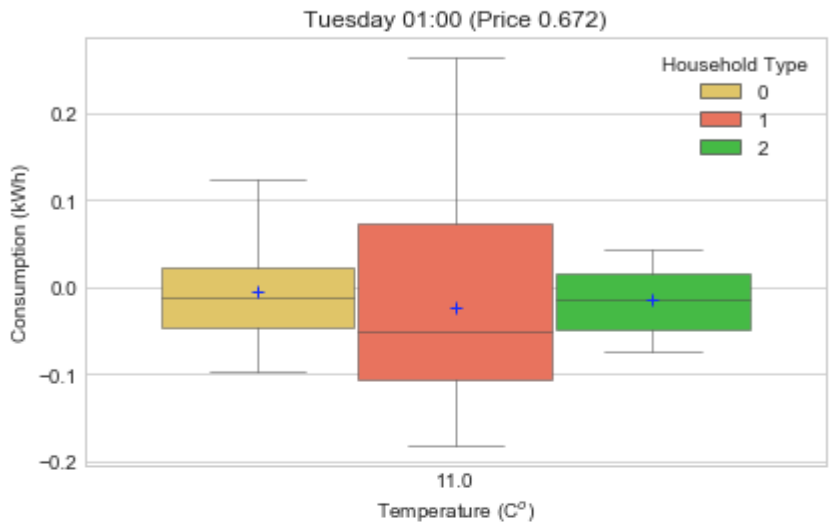

Monday 23:00 (Type 2)
Price
0.672
Consumption (kWh)
0.025
0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
11.0
Temperature (C°)

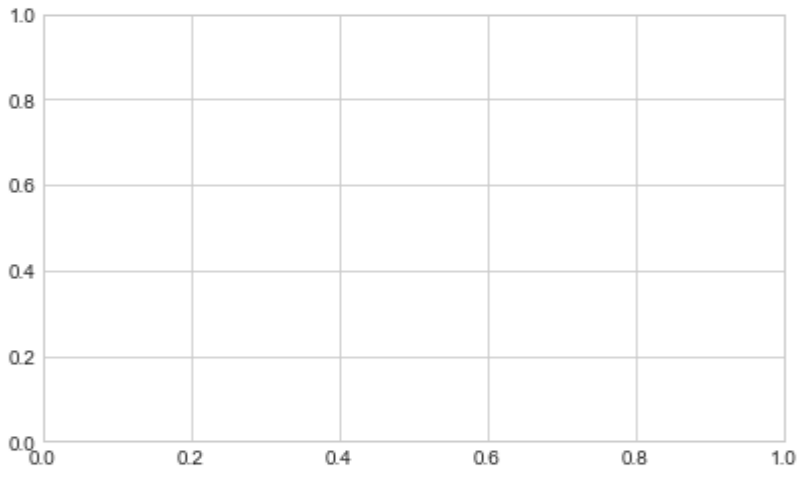
In [63]:
```
# Tuesday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 1 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

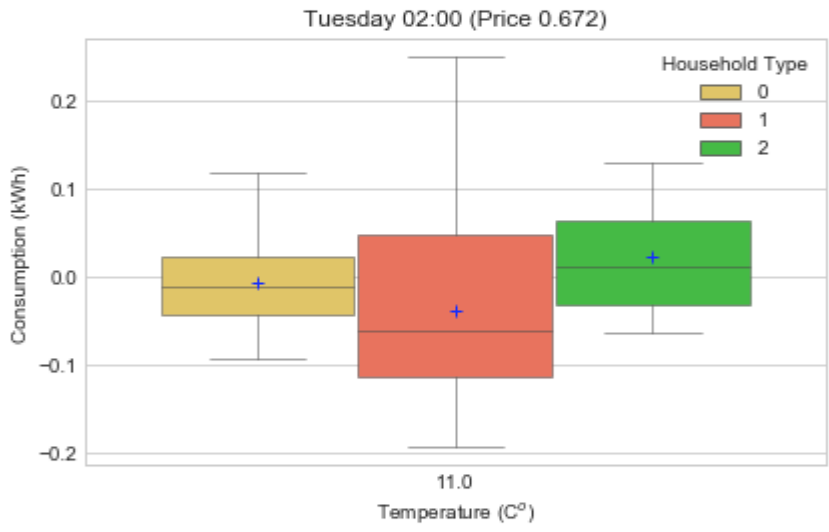

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

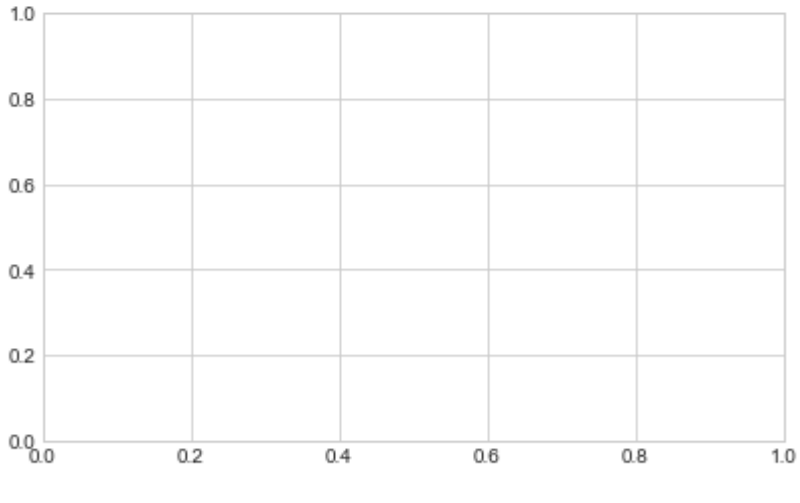

Tuesday 00:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

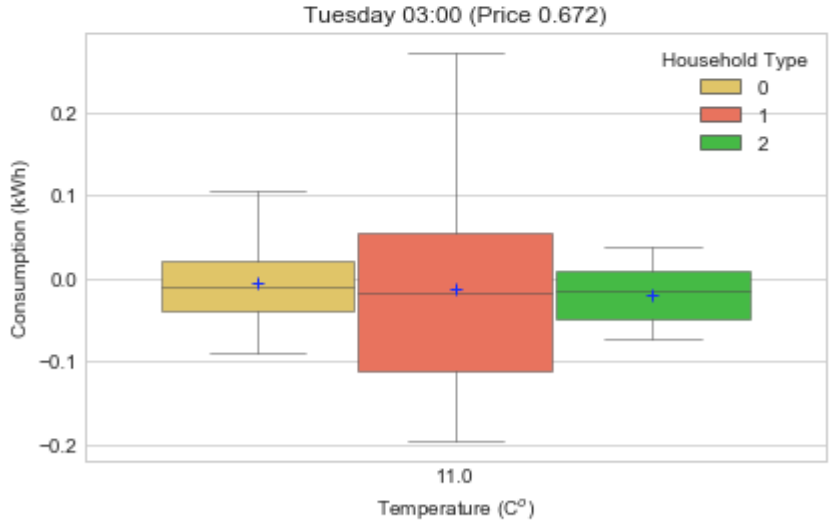

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

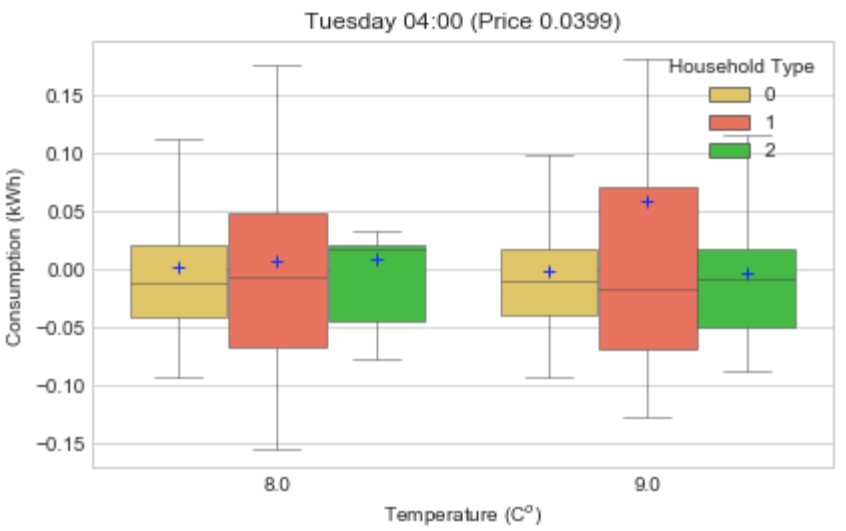

Tuesday 01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

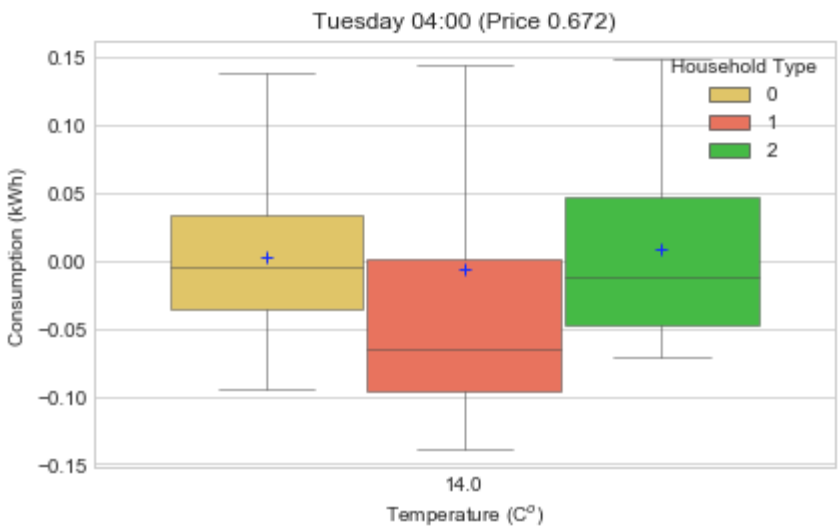

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Tuesday 02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

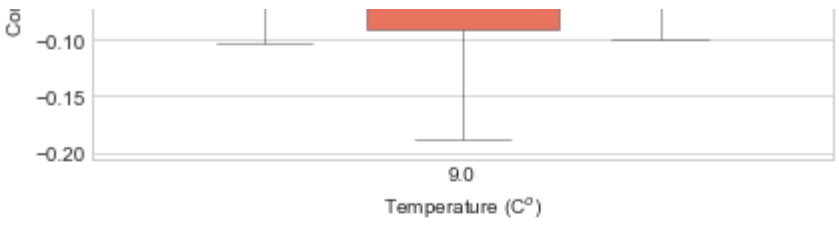

Tuesday 03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

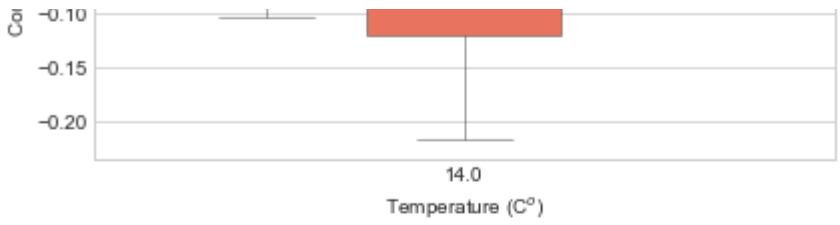

Tuesday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
8.0
9.0
Temperature (C°)

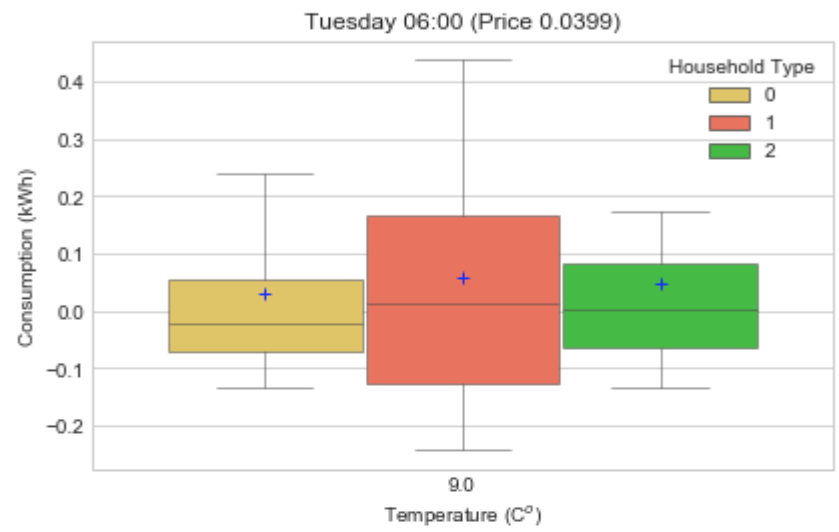

Tuesday 04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
14.0
Temperature (C°)

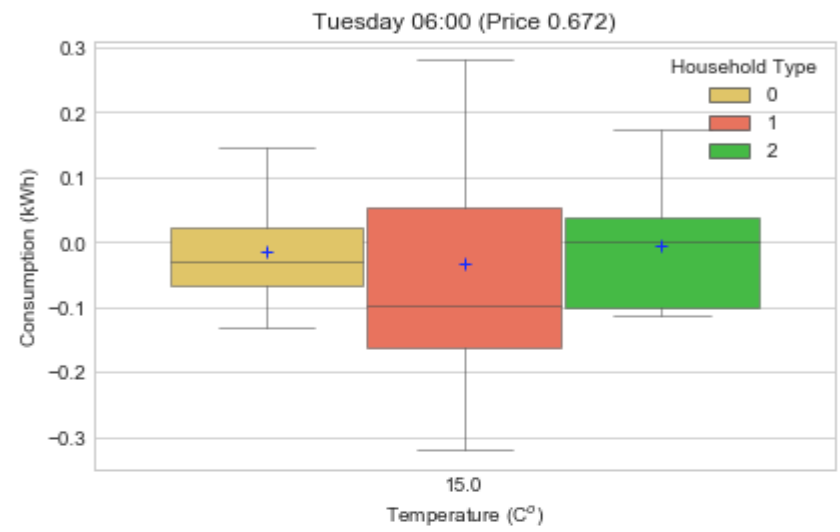

Tuesday 05:00 (Price 0.0399)
Household Type
0
1
2
0.15
0.10
0.05
0.00
−0.05

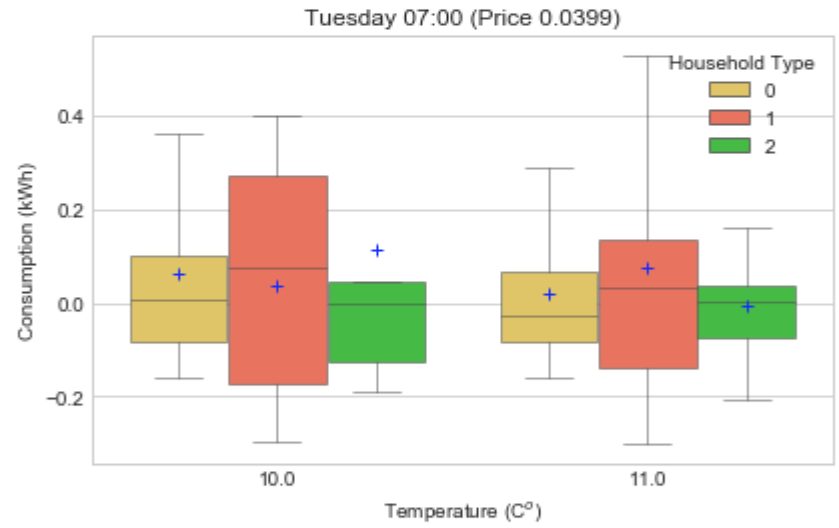

Tuesday 05:00 (Price 0.672)
Household Type
0
1
2
0.15
0.10
0.05
0.00
−0.05

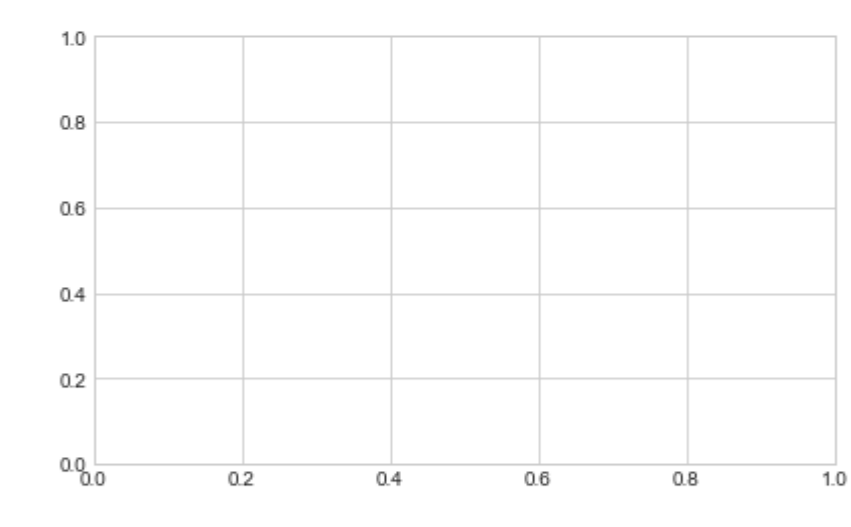

−0.10
−0.15
−0.20
9.0
Temperature (C°)

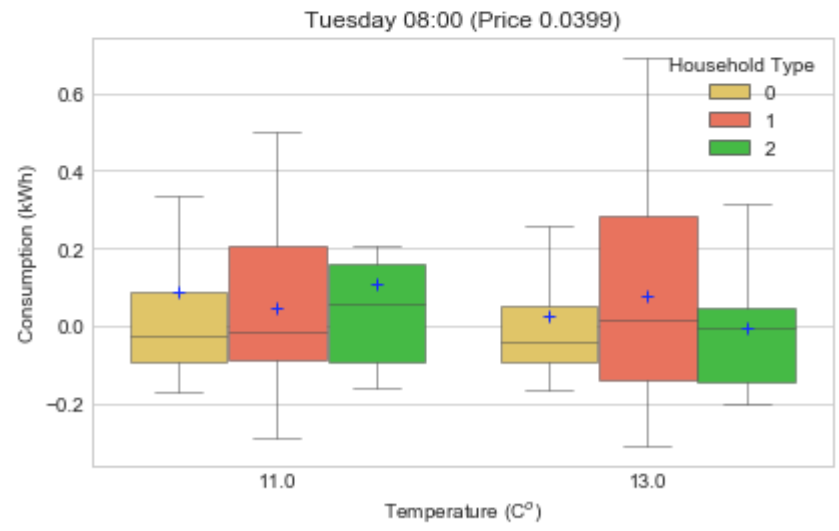

−0.10
−0.15
−0.20
14.0
Temperature (C°)

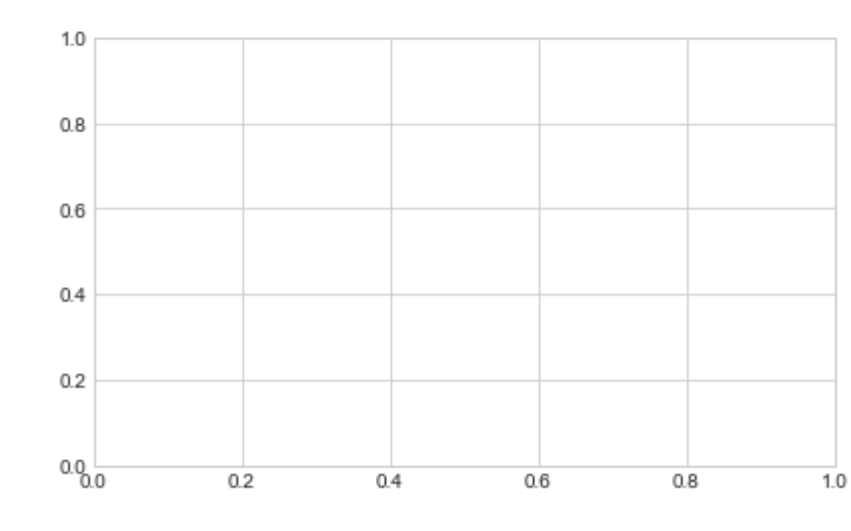

Tuesday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
9.0
Temperature (C°)

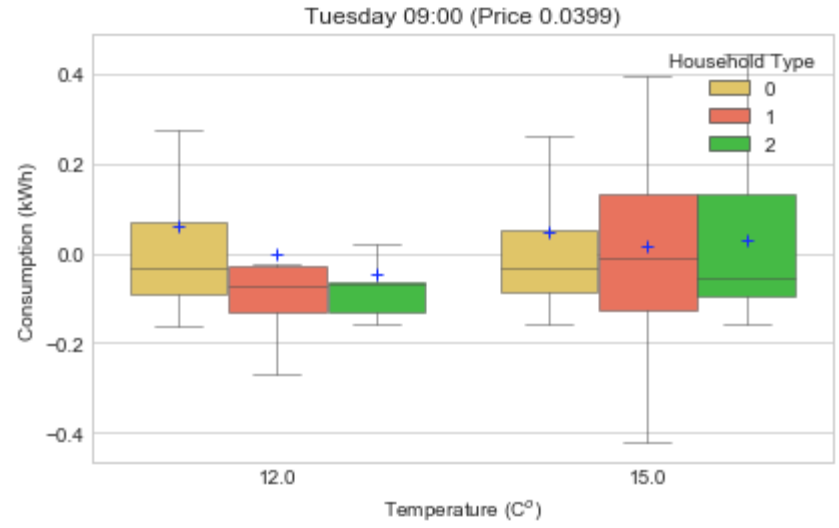

Tuesday 06:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
15.0
Temperature (C°)

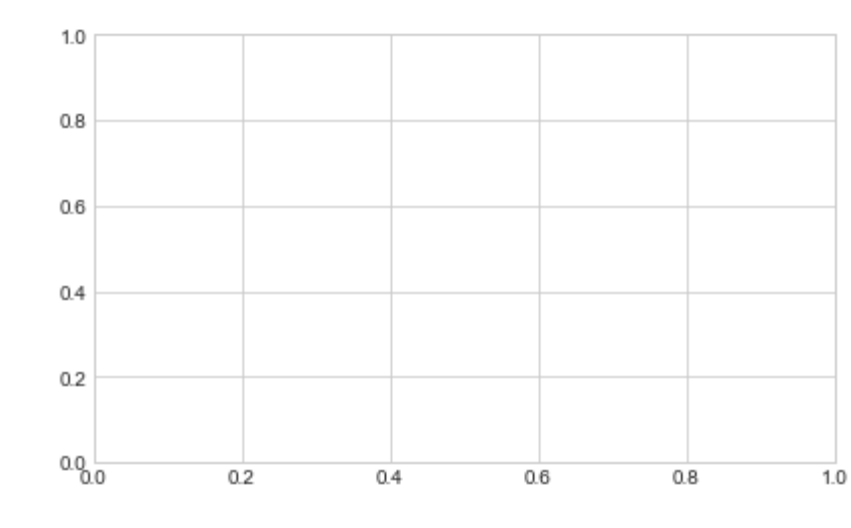

Tuesday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
10.0
11.0
Temperature (C°)

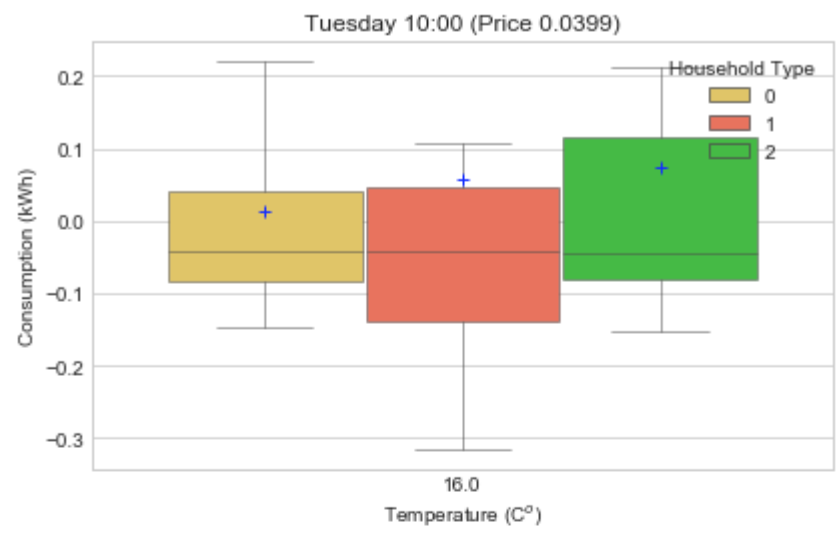

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

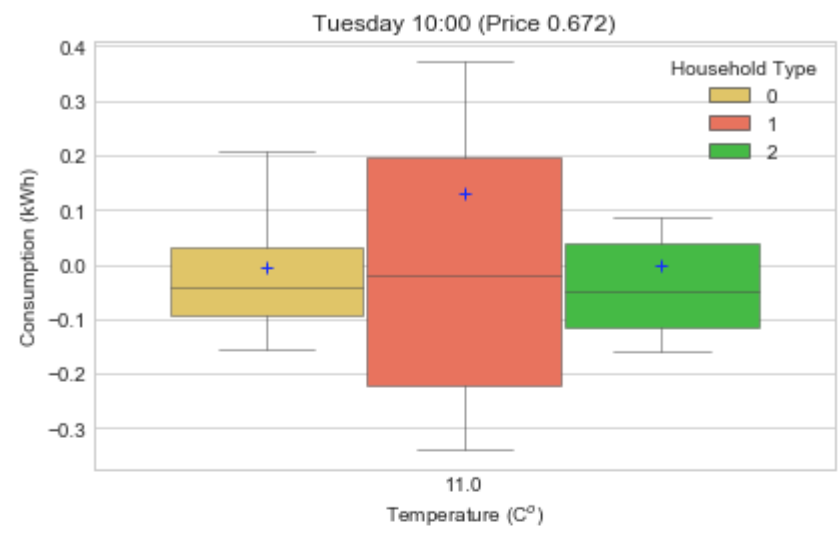

Tuesday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
11.0
13.0
Temperature (C°)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Tuesday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
12.0
15.0
Temperature (C°)

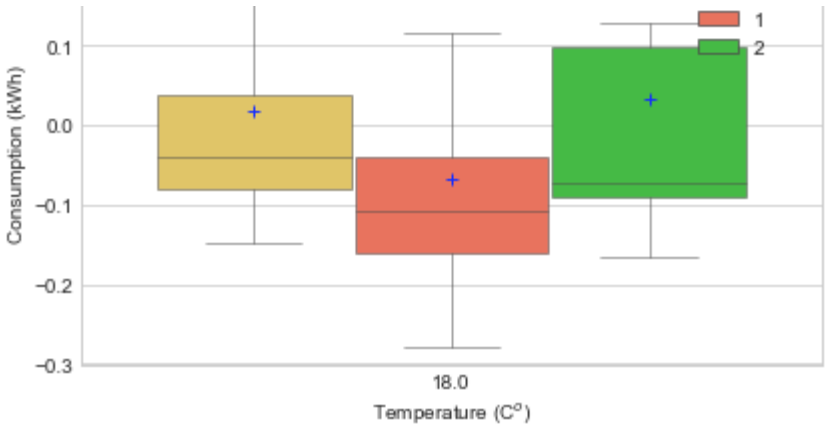

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

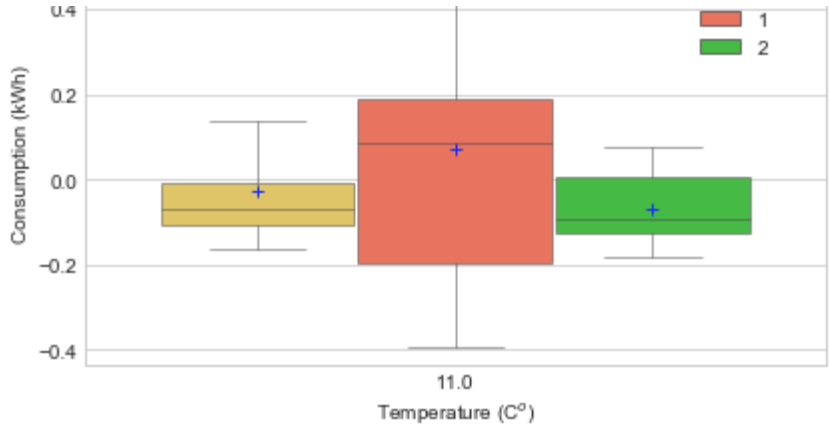

Tuesday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
16.0
Temperature (C°)

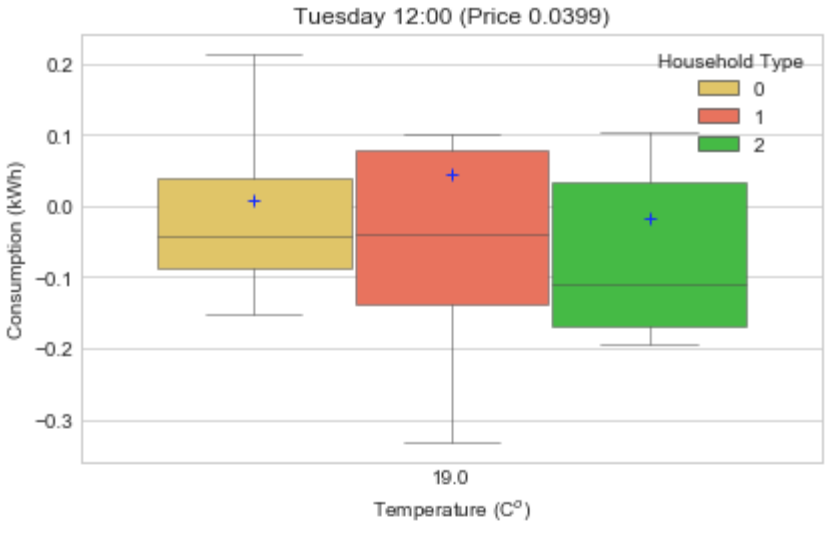

Tuesday 10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
11.0
Temperature (C°)

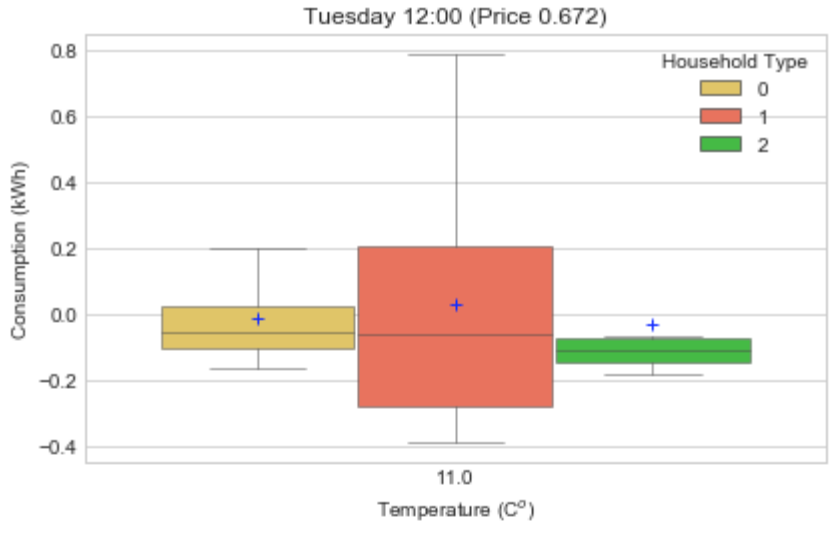
Tuesday 11:00 (Price 0.0399)
Household Type
0
Consumption (kWh)
0.2

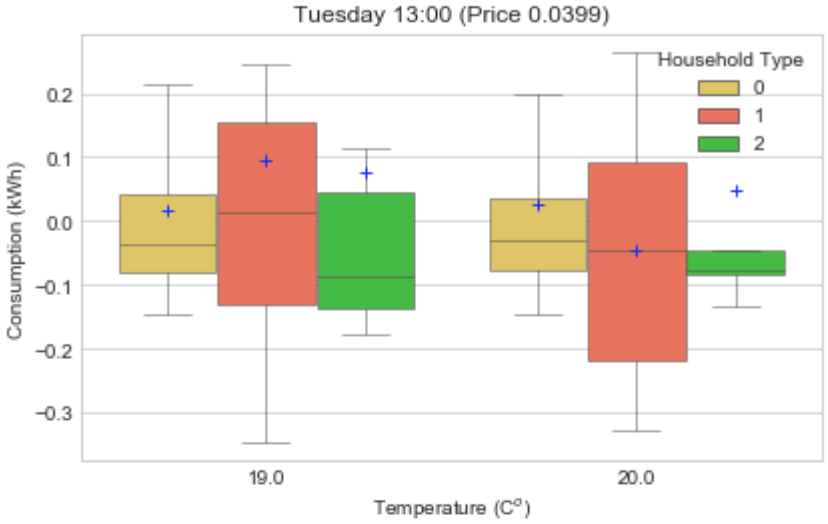
Tuesday 11:00 (Price 0.672)
Household Type
0

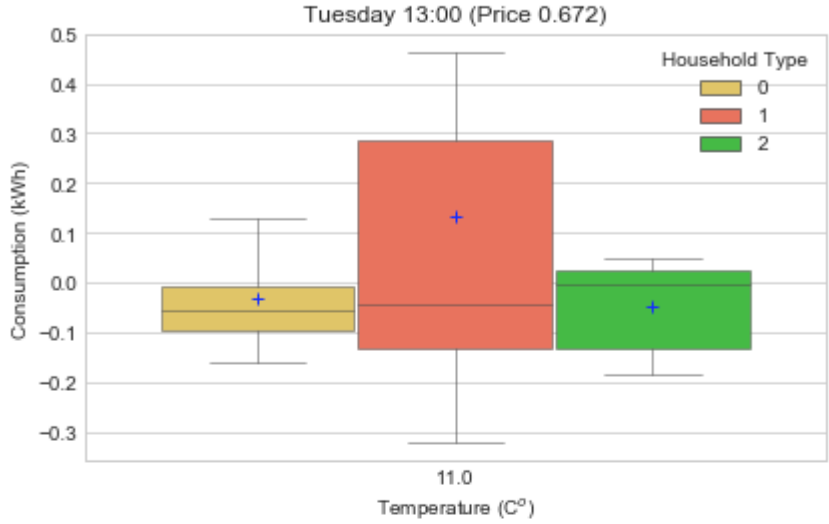

Consumption (kWh)
0.1
0.0
−0.1
−0.2
−0.3
1
2
18.0
Temperature (C°)

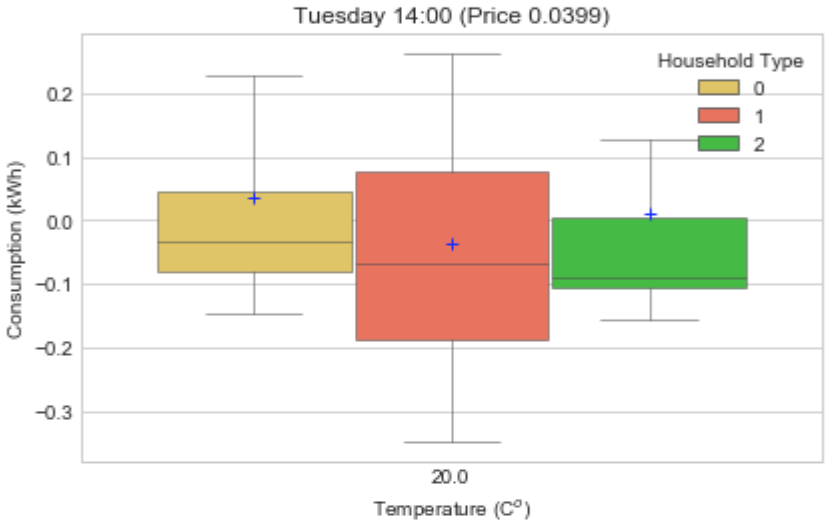

Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
1
2
11.0
Temperature (C°)

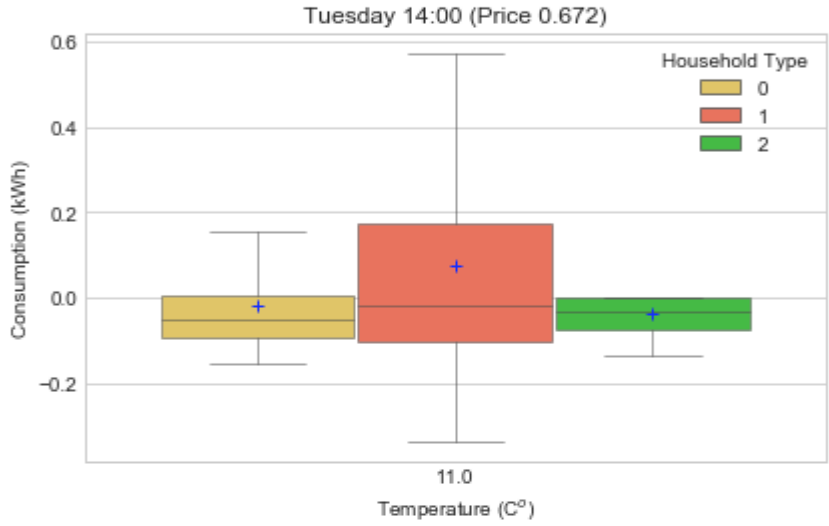

Tuesday 12:00 (Price 0.0399)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2
19.0
Temperature (C°)

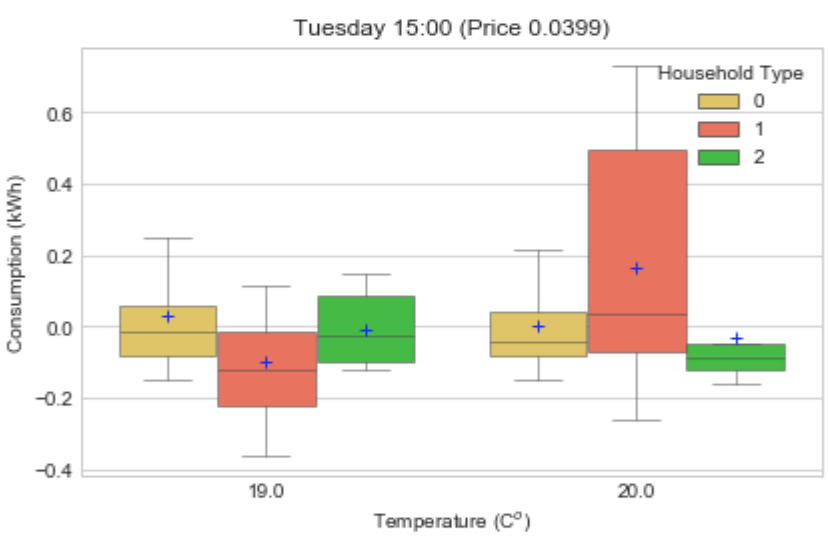

Tuesday 12:00 (Price 0.672)
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
11.0
Temperature (C°)

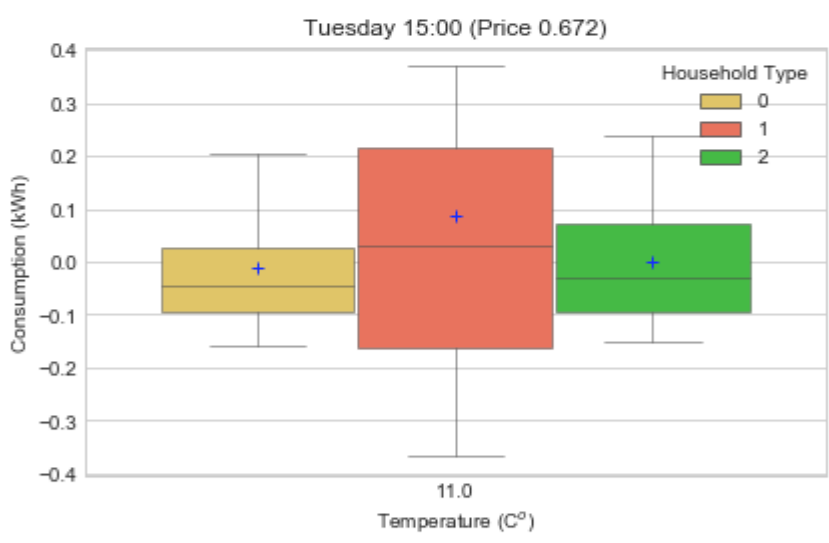

Tuesday 13:00 (Price 0.0399)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2
19.0
20.0
Temperature (C°)

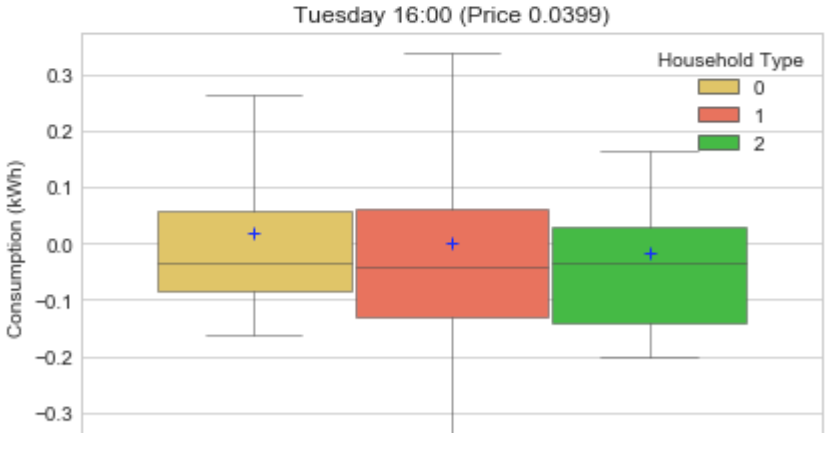

Tuesday 13:00 (Price 0.672)
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2
11.0
Temperature (C°)

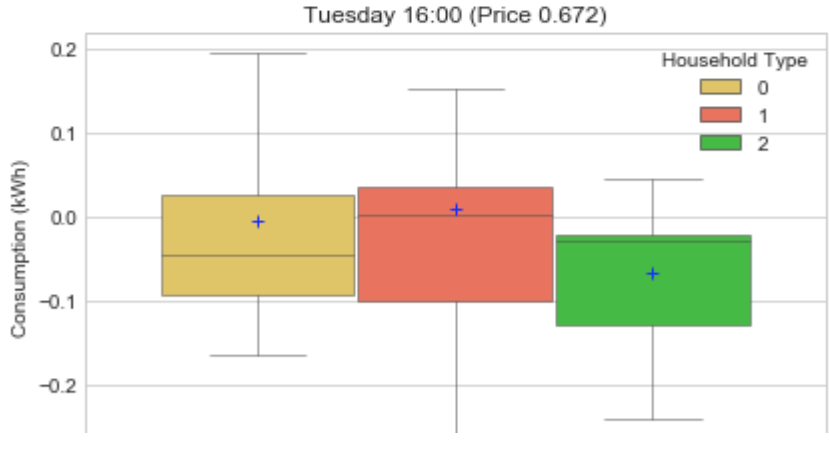

Tuesday 14:00 (Price 0.0399)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2
20.0
Temperature (C°)

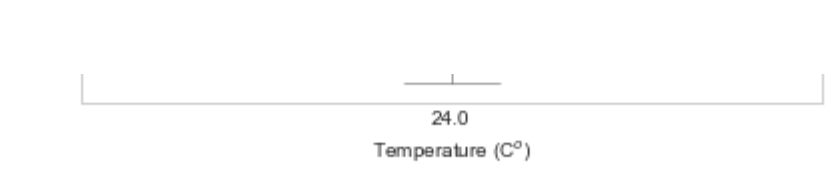

Tuesday 14:00 (Price 0.672)
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
Household Type
0
1
2
11.0
Temperature (C°)

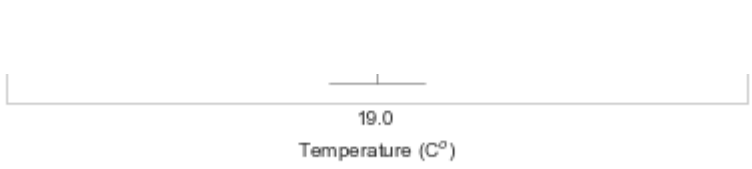

Tuesday 15:00 (Price 0.0399)
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
19.0
20.0
Temperature (C°)

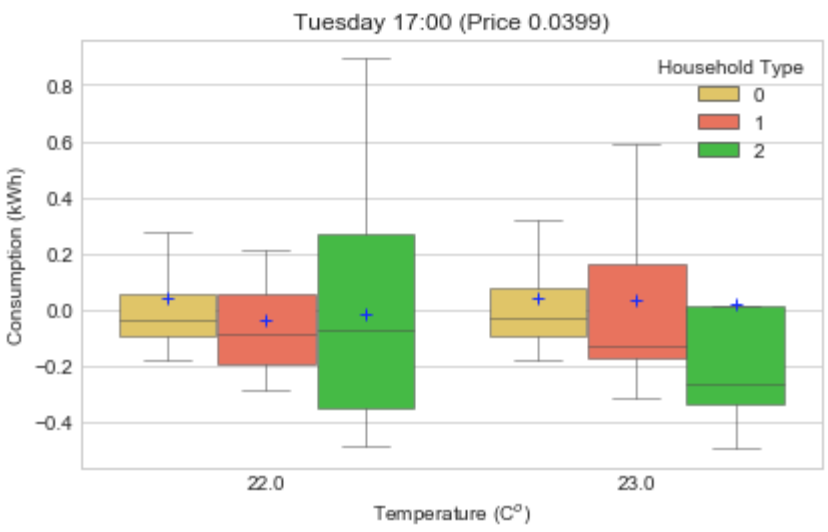

Tuesday 15:00 (Price 0.672)
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
11.0
Temperature (C°)

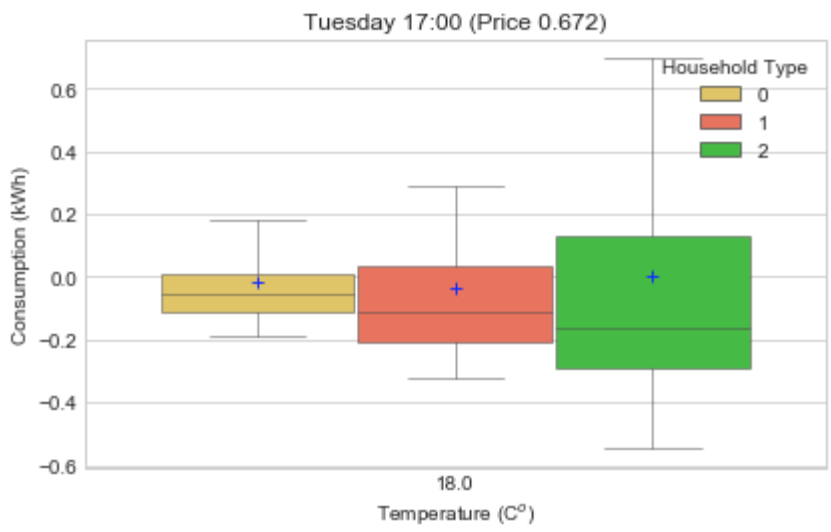

Tuesday 16:00 (Price 0.0399)
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2

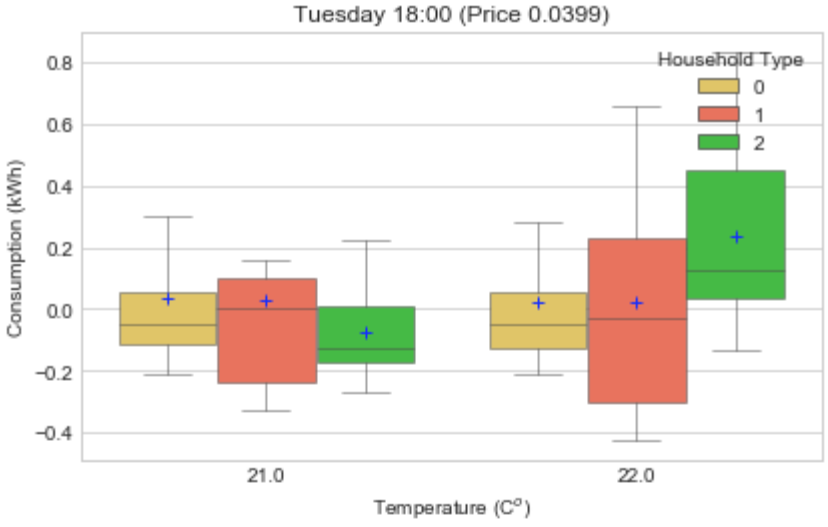

Tuesday 16:00 (Price 0.672)
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
Household Type
0
1
2

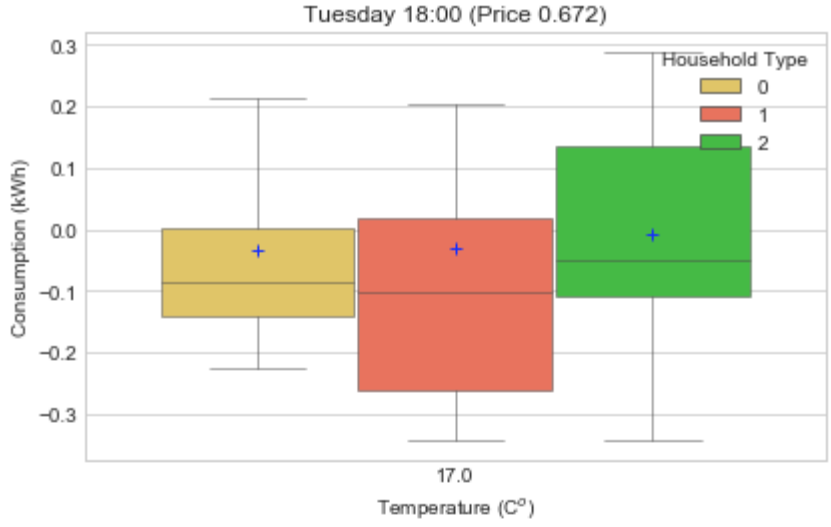

24.0
Temperature (C°)

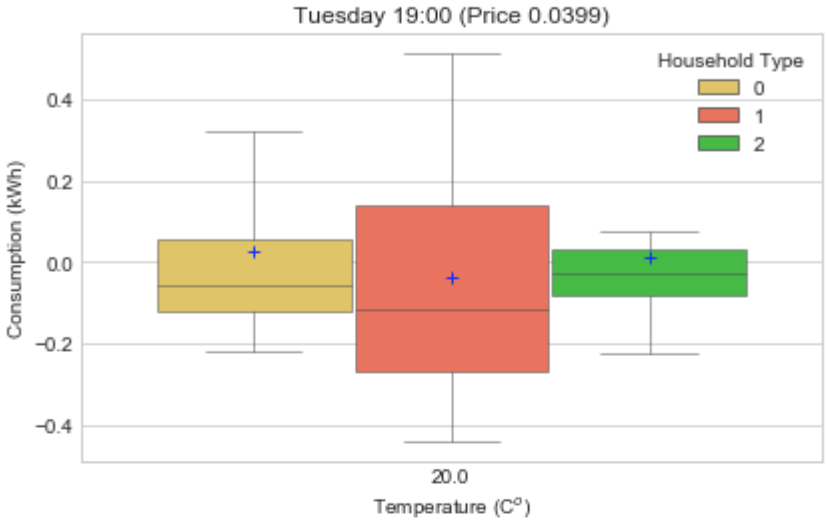

19.0
Temperature (C°)

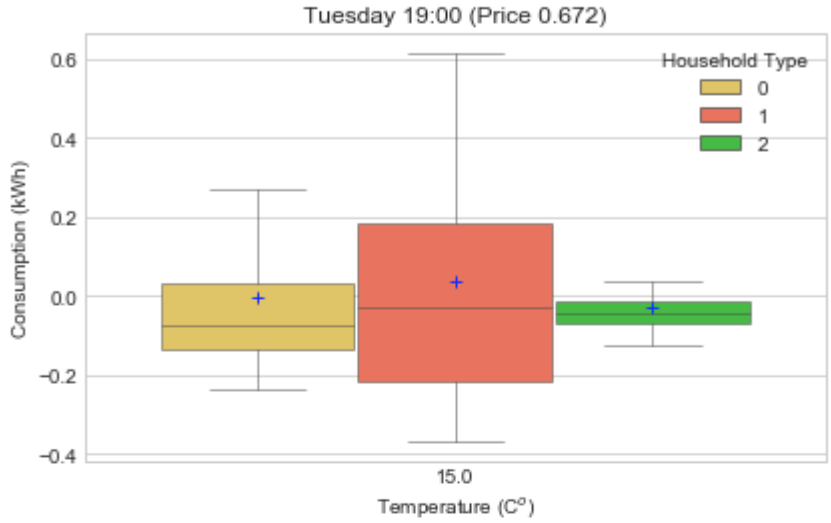

Tuesday 17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
22.0
23.0
Temperature (C°)

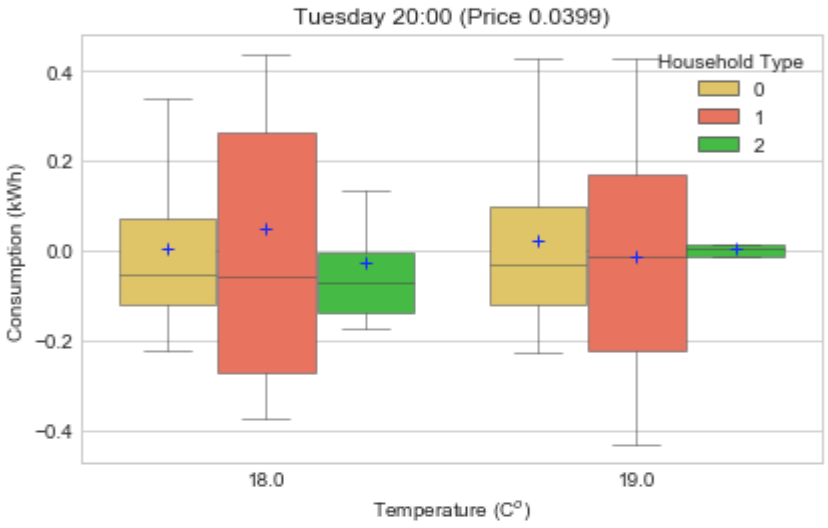

Tuesday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
18.0
Temperature (C°)

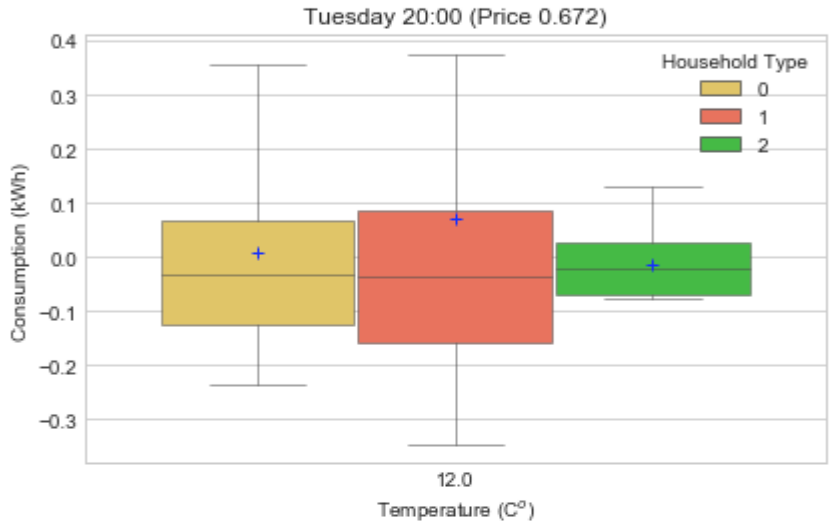

Tuesday 18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
21.0
22.0
Temperature (C°)

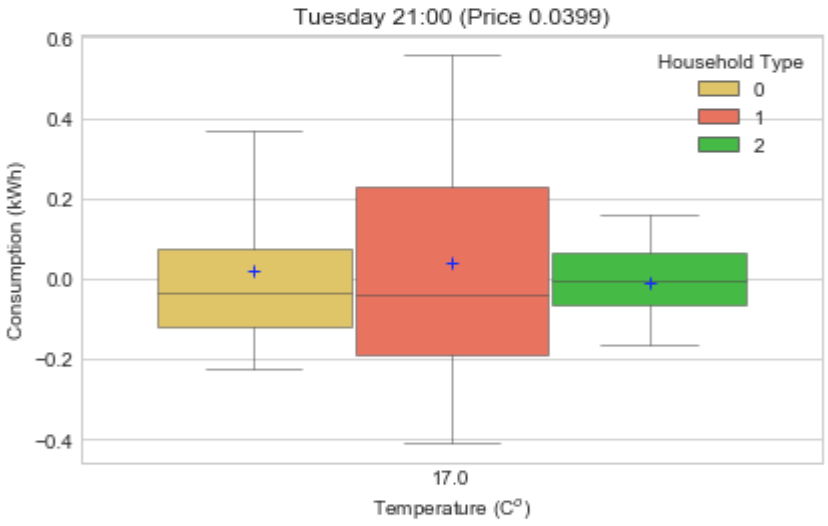

Tuesday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
17.0
Temperature (C°)

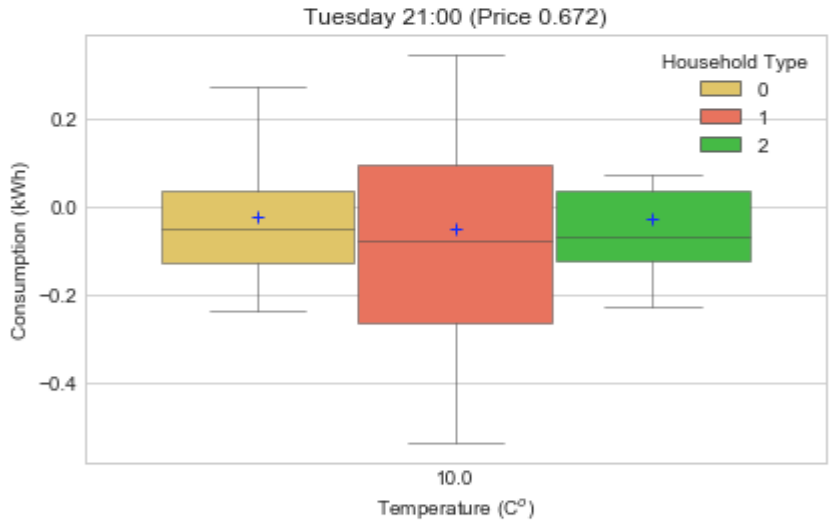

Tuesday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
20.0
Temperature (C°)

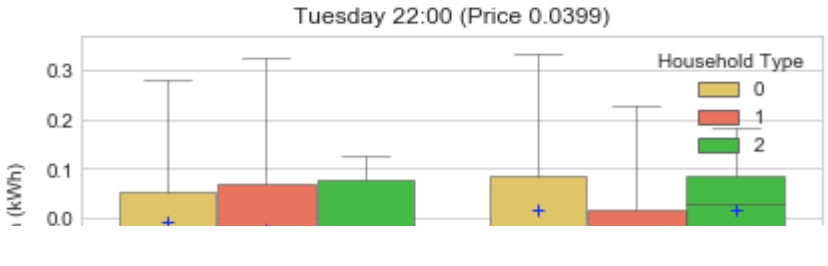

Tuesday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
Temperature (C°)

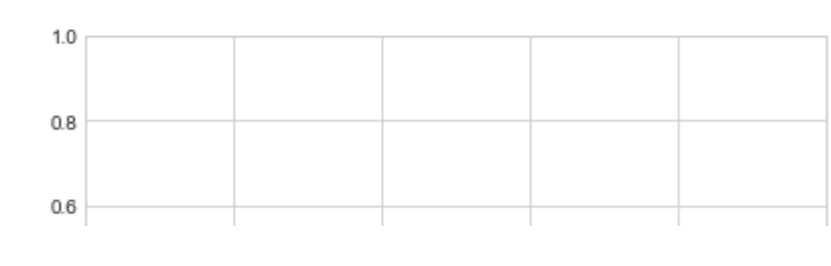

Tuesday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
18.0
19.0
Temperature (C°)

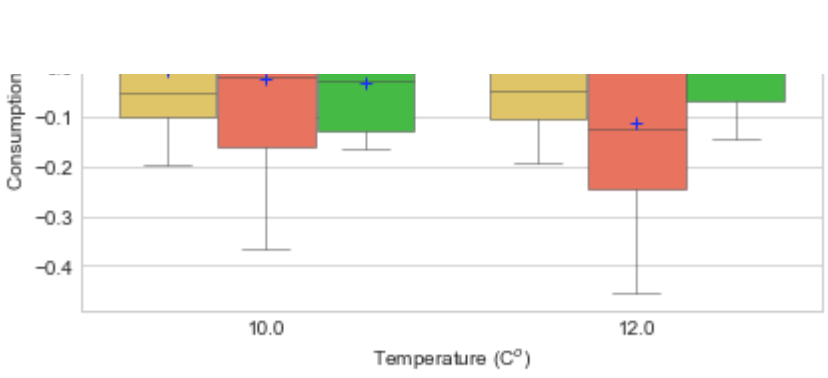

Tuesday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
12.0
Temperature (C°)

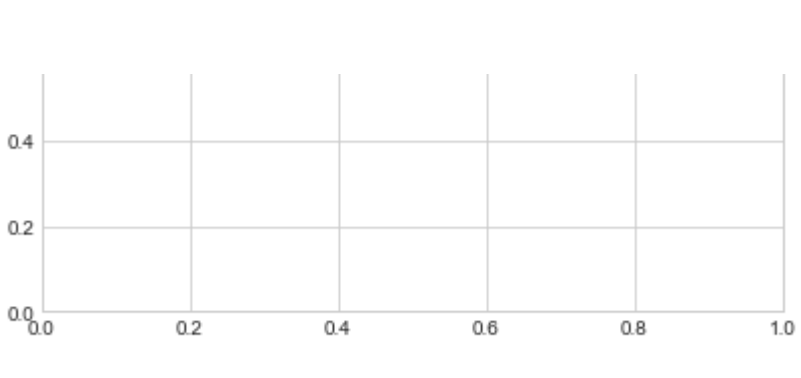

Tuesday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
17.0
Temperature (C°)

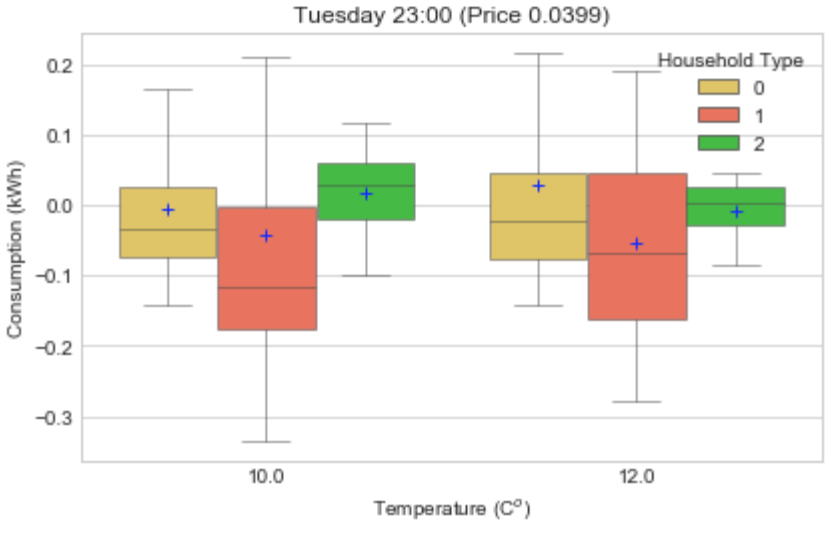

Tuesday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.0
−0.2
−0.4
10.0
Temperature (C°)

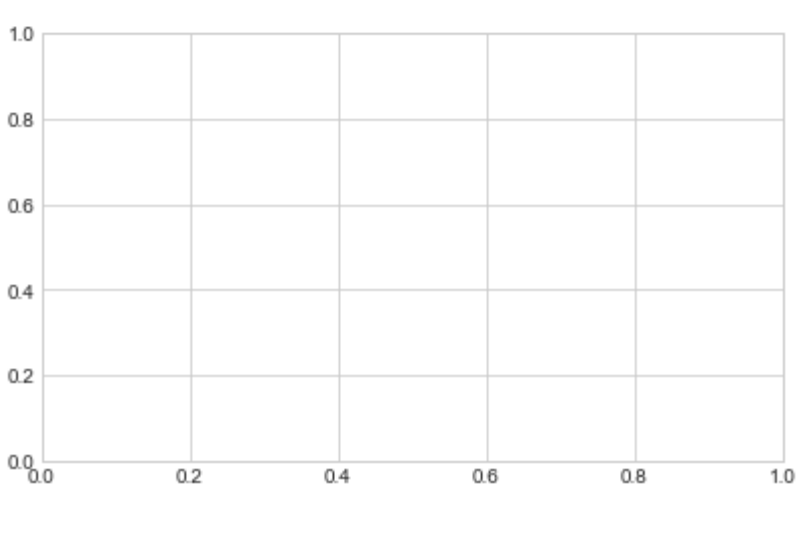

Tuesday 22:00 (Price 0.0399)
Household Type
0
1
2
0.3
0.2
0.1
0.0


1.0
0.8
0.6

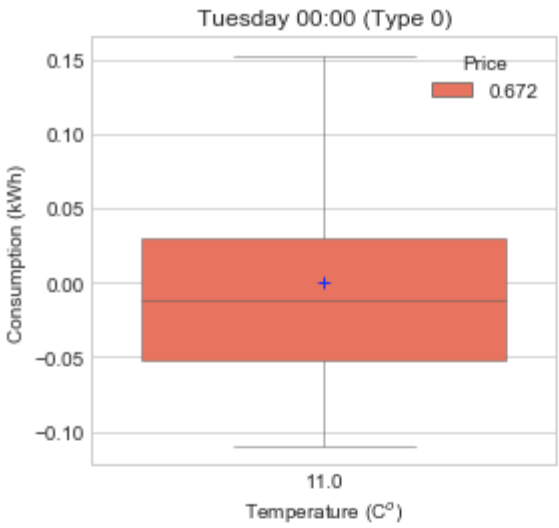

Consumption (kWh)
−0.1
−0.2
−0.3
−0.4
10.0
12.0
Temperature (C°)

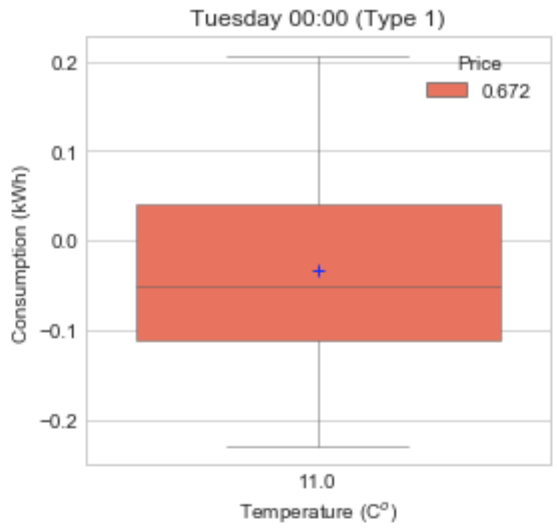

0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

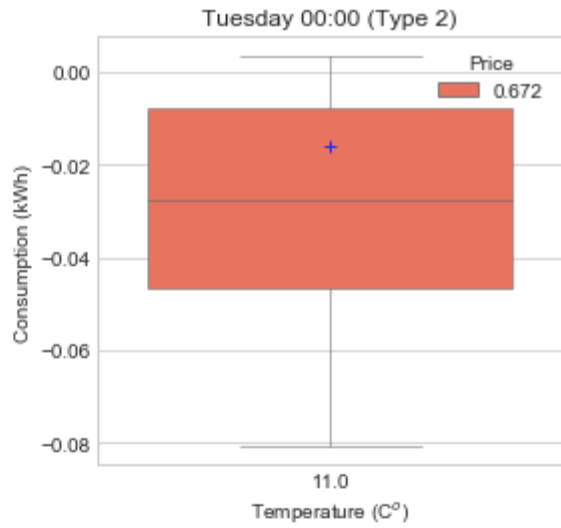

Tuesday 23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
10.0
12.0
Temperature (C°)

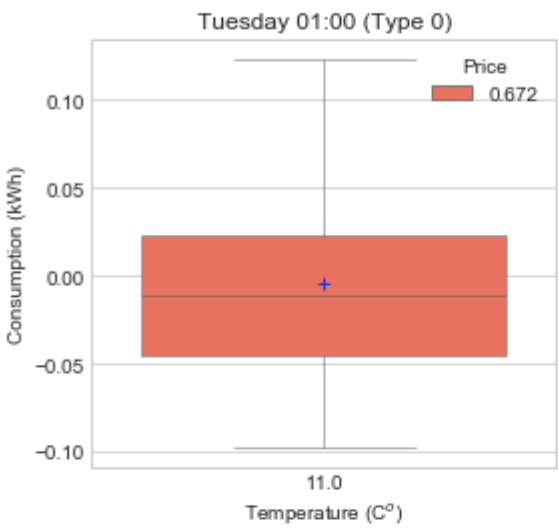

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

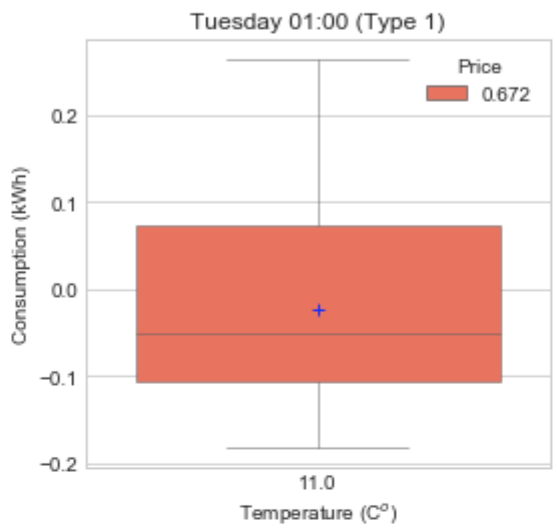
In [64]:
```
# Price comparison
# Tuesday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 1 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

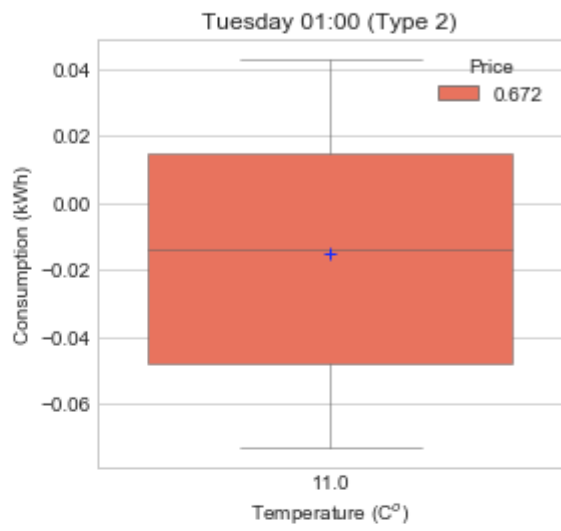

Tuesday 00:00 (Type 0)
Price
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
11.0
Temperature (C°)

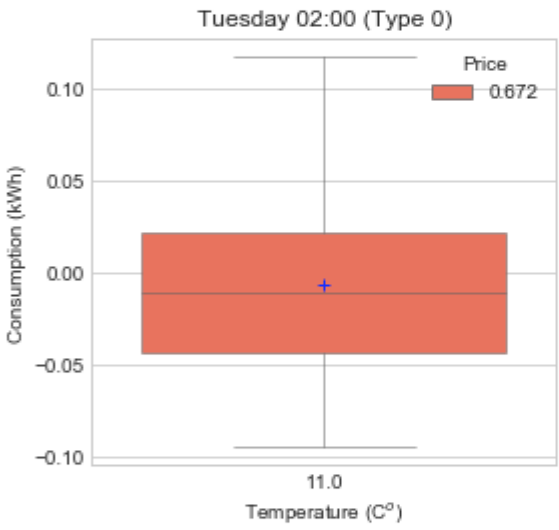

Tuesday 00:00 (Type 1)
Price
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

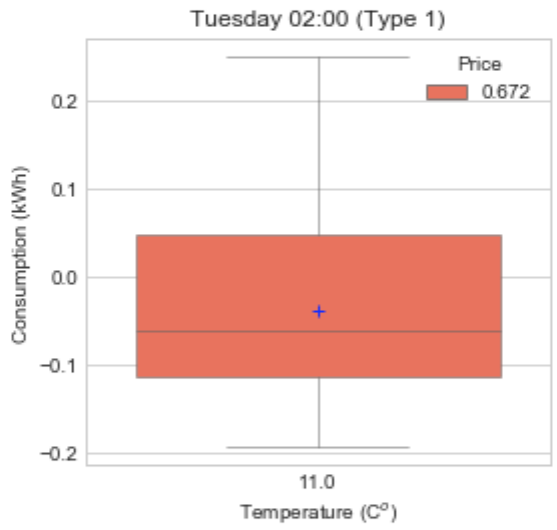

Tuesday 00:00 (Type 2)
Price
0.672
Consumption (kWh)
0.00
−0.02
−0.04
−0.06
−0.08
11.0
Temperature (C°)

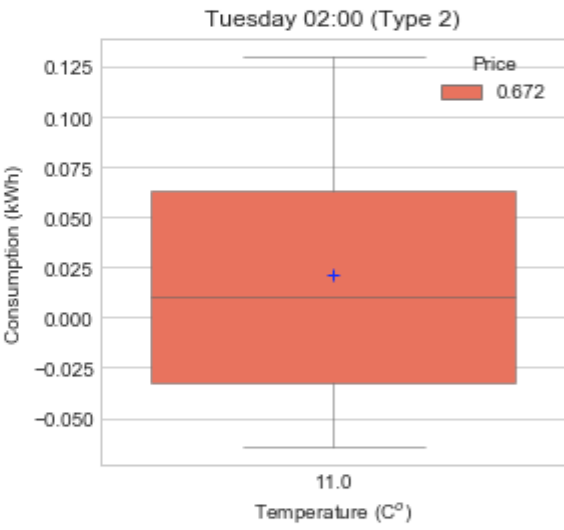

Tuesday 01:00 (Type 0)
Price
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
11.0
Temperature (C°)

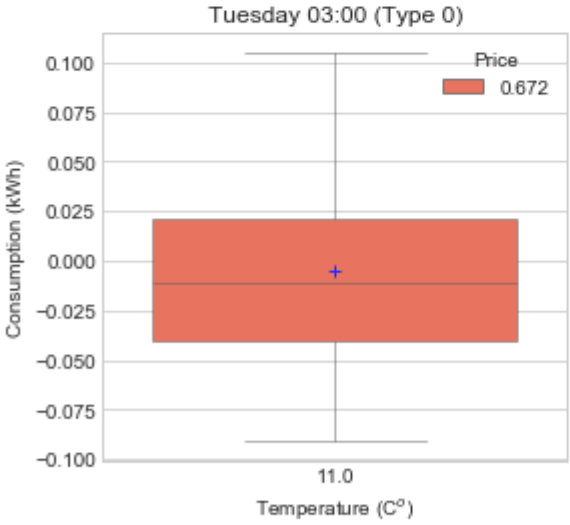

Tuesday 01:00 (Type 1)
Price
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

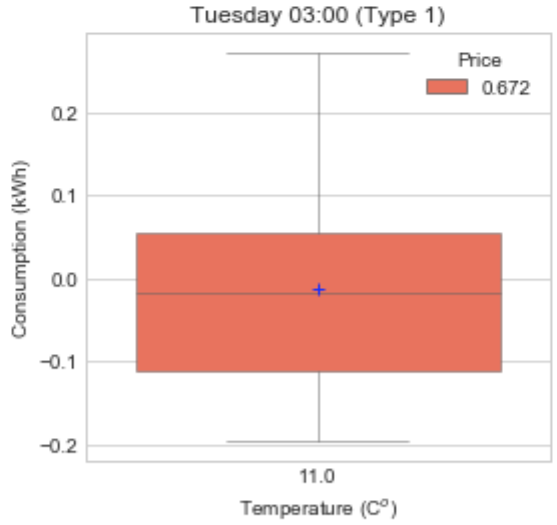

Tuesday 01:00 (Type 2)
Price
0.672
Consumption (kWh)
0.04
0.02
0.00
−0.02
−0.04
−0.06
11.0
Temperature (C°)

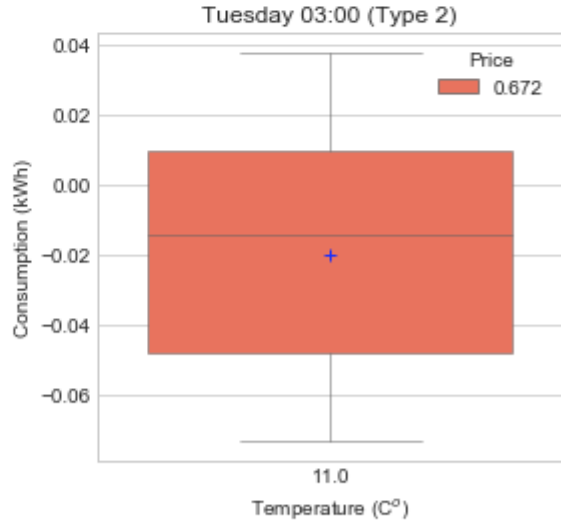

Tuesday 02:00 (Type 0)
Price
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
11.0
Temperature (C°)

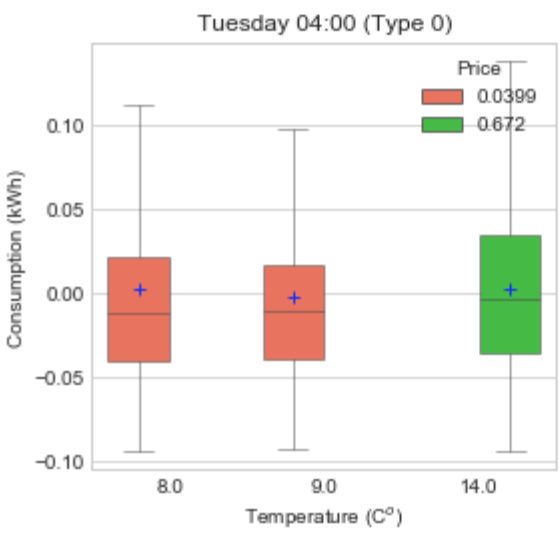

Tuesday 02:00 (Type 1)
Price
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

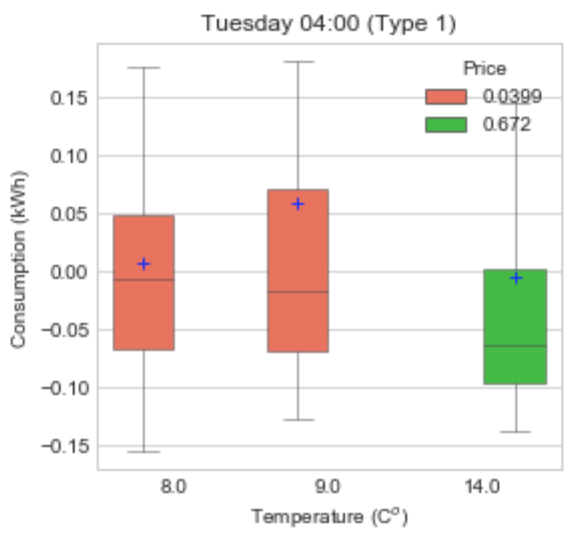

Tuesday 02:00 (Type 2)
Price
0.672
Consumption (kWh)
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
11.0
Temperature (C°)

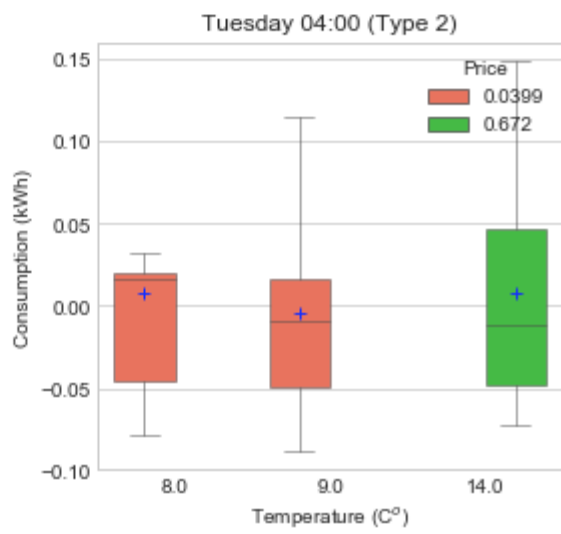

Tuesday 03:00 (Type 0)
Price
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
11.0
Temperature (C°)

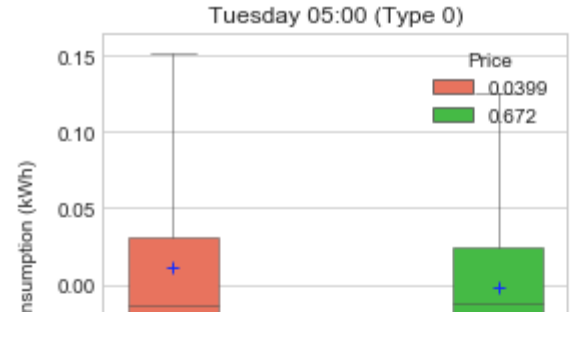

Tuesday 03:00 (Type 1)
Price
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

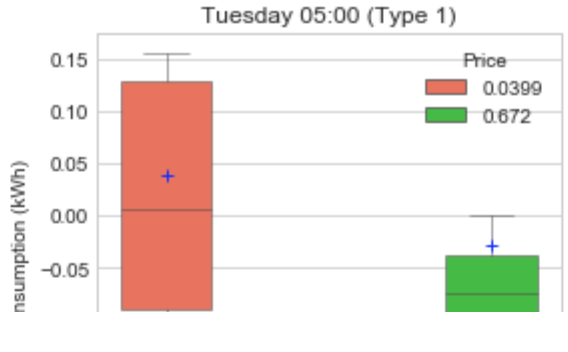

Tuesday 03:00 (Type 2)
Price
0.672
Consumption (kWh)
0.04
0.02
0.00
−0.02
−0.04
−0.06
11.0
Temperature (C°)

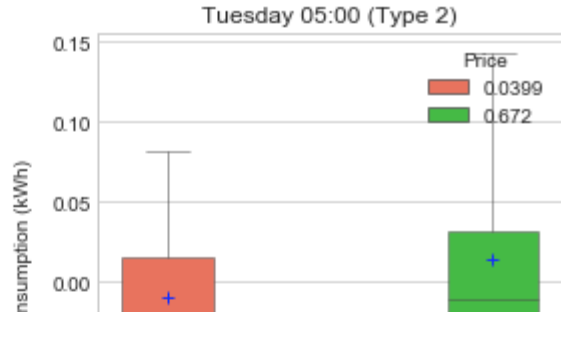

Tuesday 04:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
8.0
9.0
14.0
Temperature (C°)

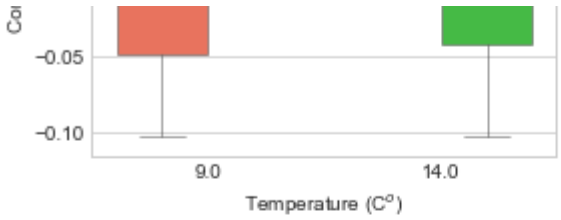

Tuesday 04:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
8.0
9.0
14.0
Temperature (C°)

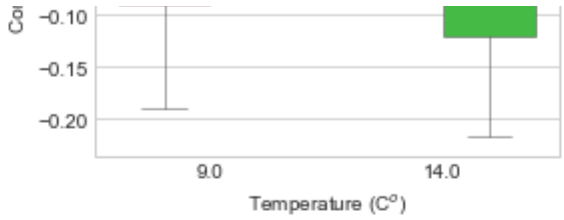

Tuesday 04:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
8.0
9.0
14.0
Temperature (C°)

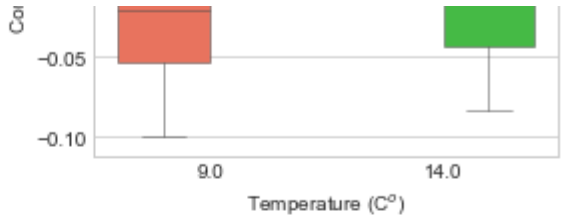

Tuesday 05:00 (Type 0)
Price
0.0399
0.672
0.15
0.10
0.05
0.00

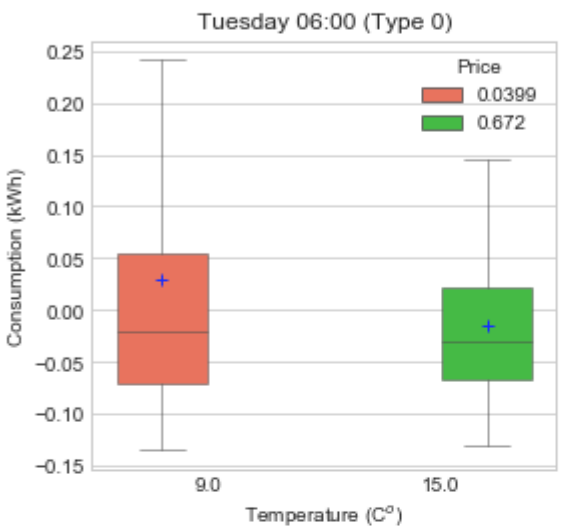

Tuesday 05:00 (Type 1)
Price
0.0399
0.672
0.15
0.10
0.05
0.00
−0.05

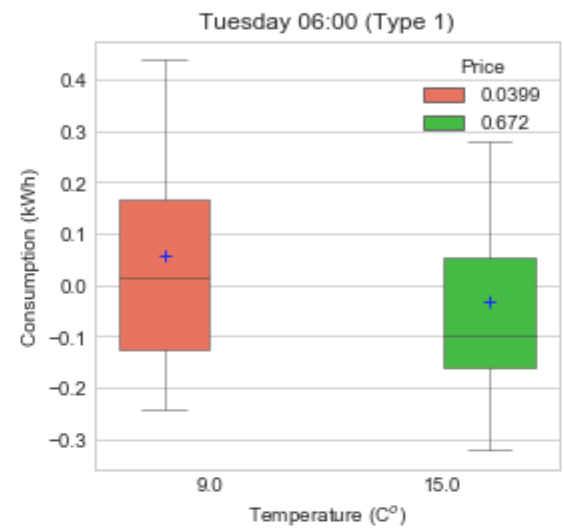

Tuesday 05:00 (Type 2)
Price
0.0399
0.672
0.15
0.10
0.05
0.00

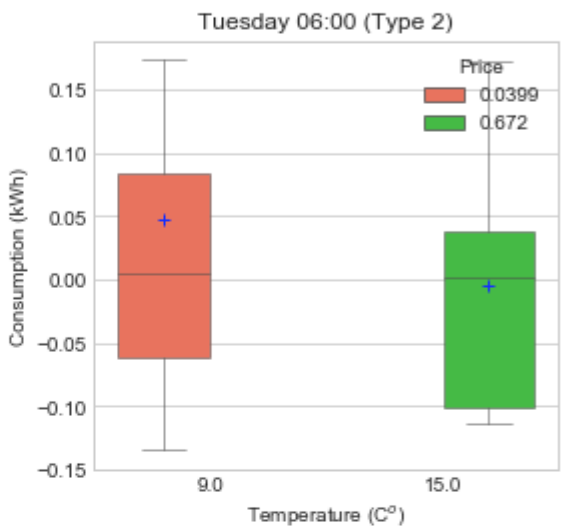

−0.05
−0.10
9.0
14.0
Temperature (C°)

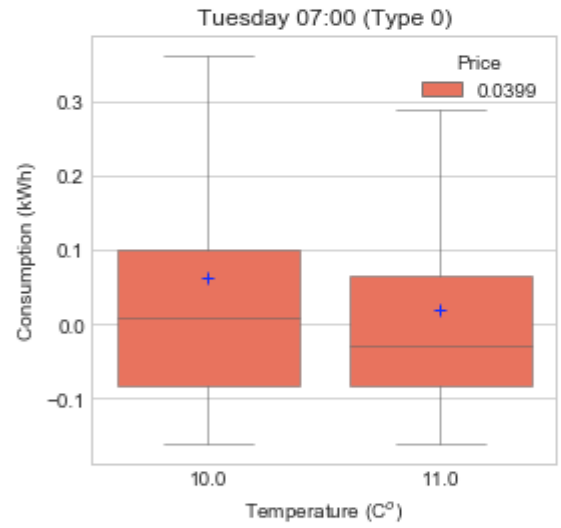

−0.10
−0.15
−0.20
9.0
14.0
Temperature (C°)

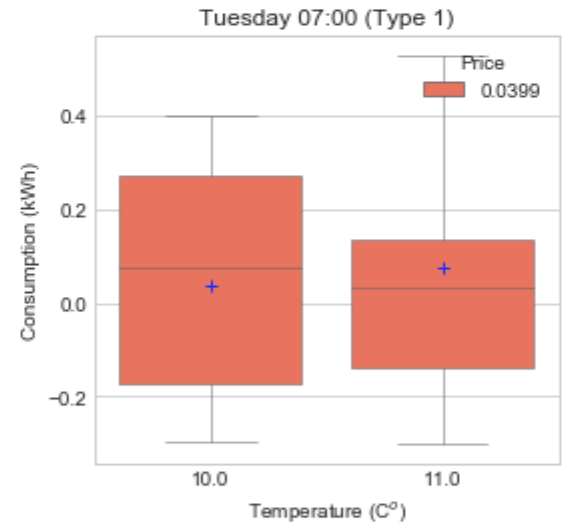

−0.05
−0.10
9.0
14.0
Temperature (C°)

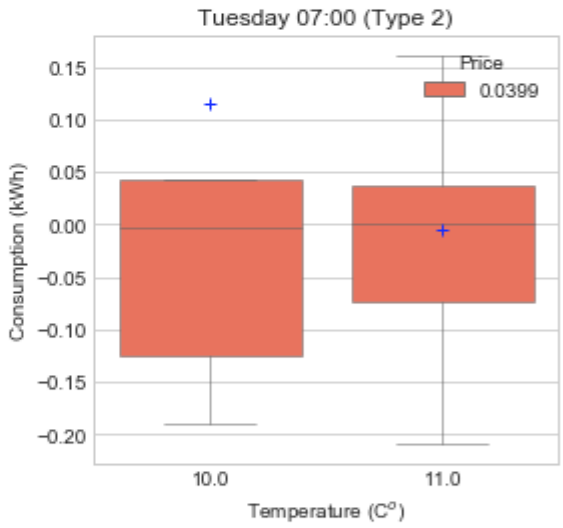

Tuesday 06:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
9.0
15.0
Temperature (C°)

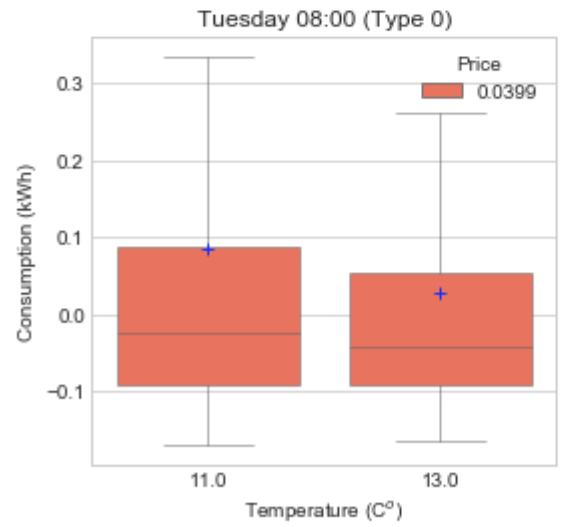

Tuesday 06:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
9.0
15.0
Temperature (C°)

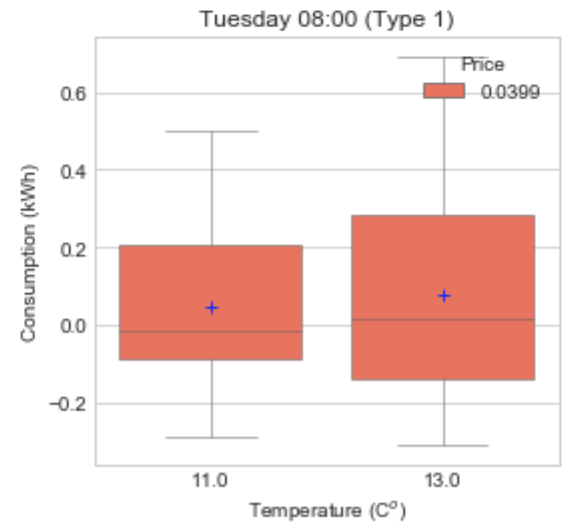

Tuesday 06:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
9.0
15.0
Temperature (C°)

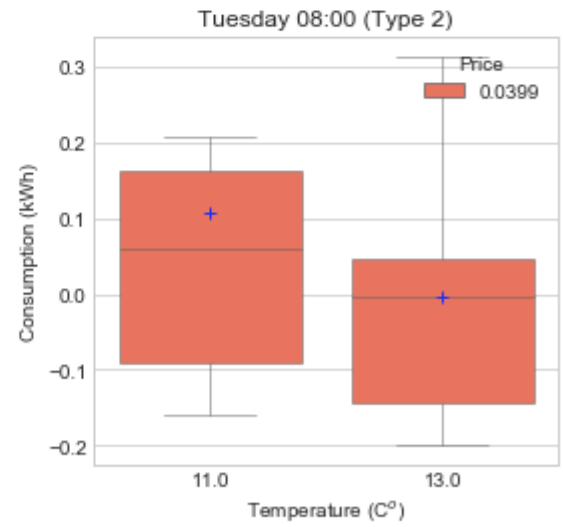

Tuesday 07:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
10.0
11.0
Temperature (C°)

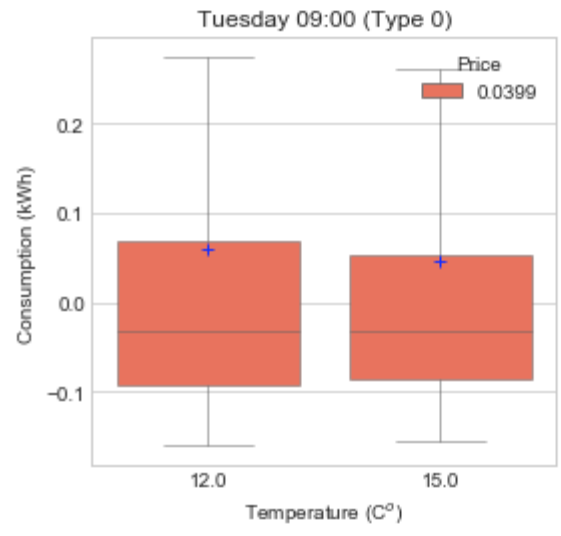

Tuesday 07:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.2
0.0
−0.2
10.0
11.0
Temperature (C°)

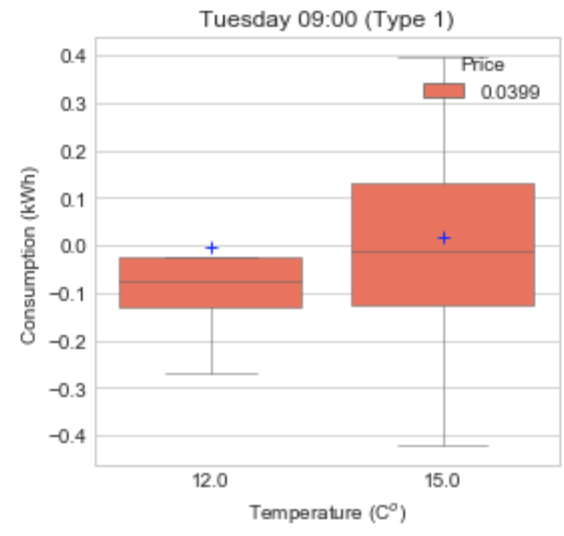

Tuesday 07:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
10.0
11.0
Temperature (C°)

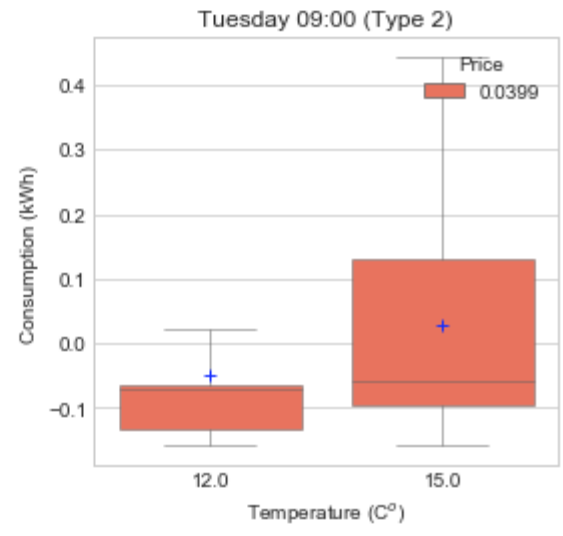

Tuesday 08:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
11.0
13.0
Temperature (C°)

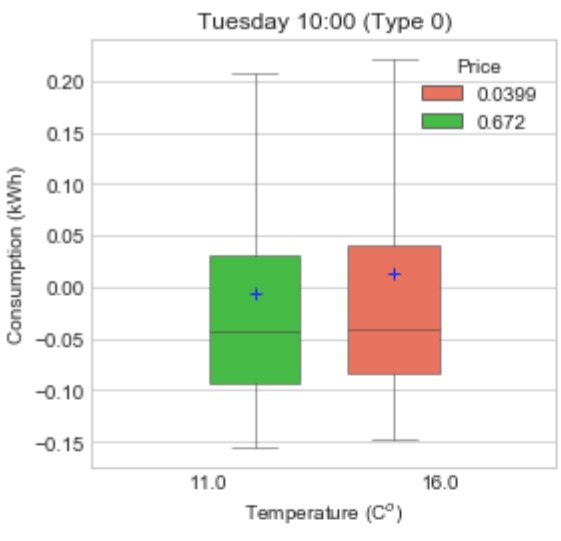

Tuesday 08:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
11.0
13.0
Temperature (C°)

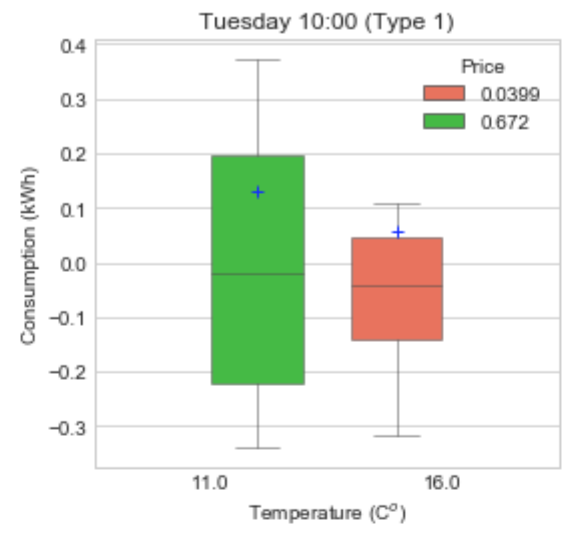

Tuesday 08:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
11.0
13.0
Temperature (C°)

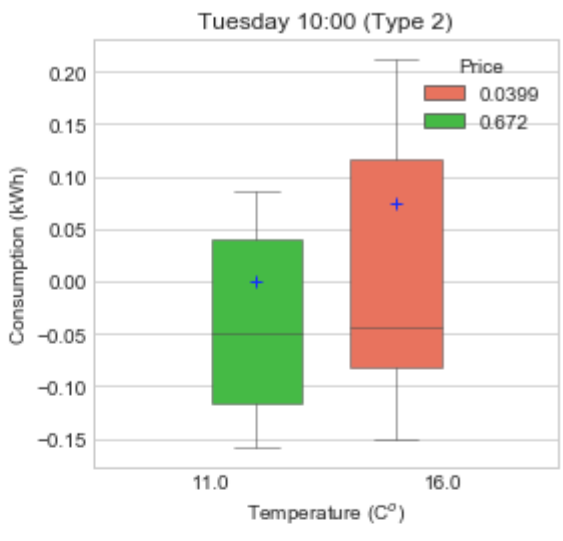

Tuesday 09:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
12.0
15.0
Temperature (C°)

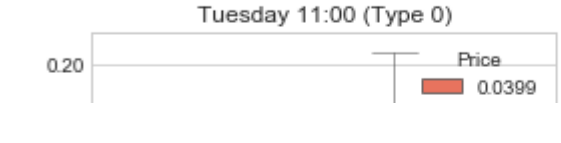

Tuesday 09:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
12.0
15.0
Temperature (C°)


Tuesday 09:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
12.0
15.0
Temperature (C°)


Tuesday 10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
11.0
16.0
Temperature (C°)

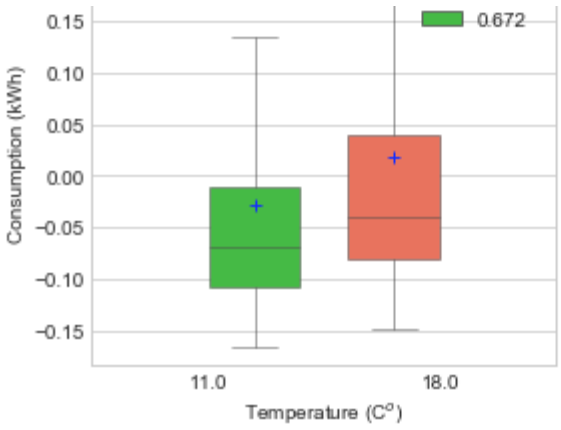

Tuesday 10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
11.0
16.0
Temperature (C°)

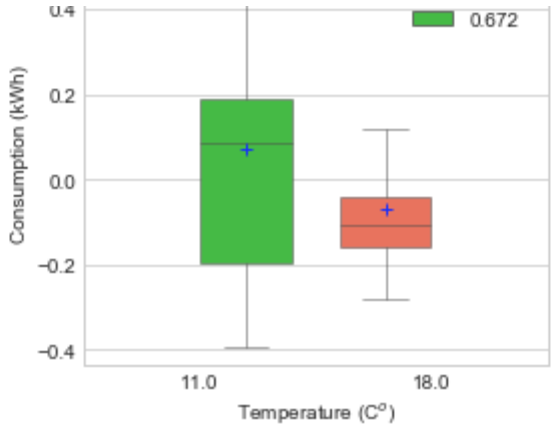

Tuesday 10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
11.0
16.0
Temperature (C°)

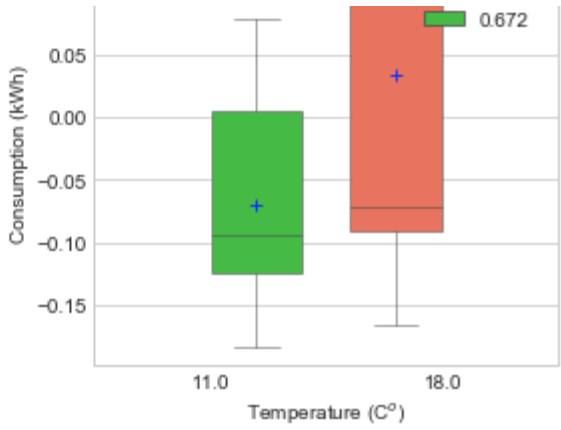

Tuesday 11:00 (Type 0)
Price
0.0399
0.20

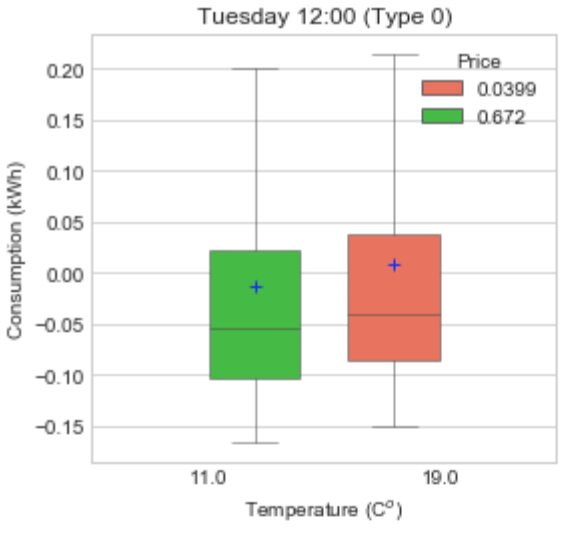

Tuesday 11:00 (Type 1)
Price
0.0399

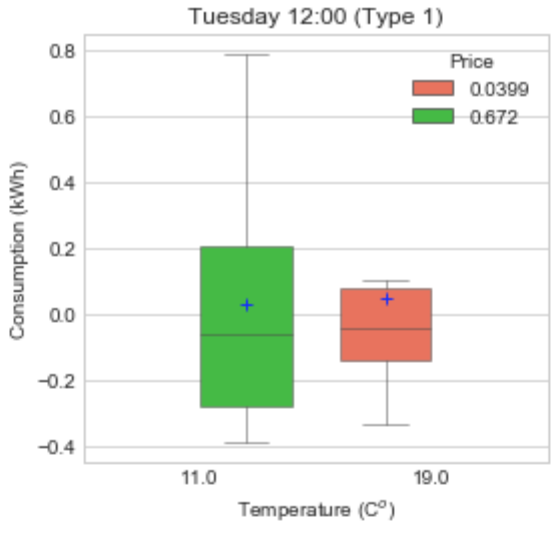

Tuesday 11:00 (Type 2)
Price
0.0399
0.10

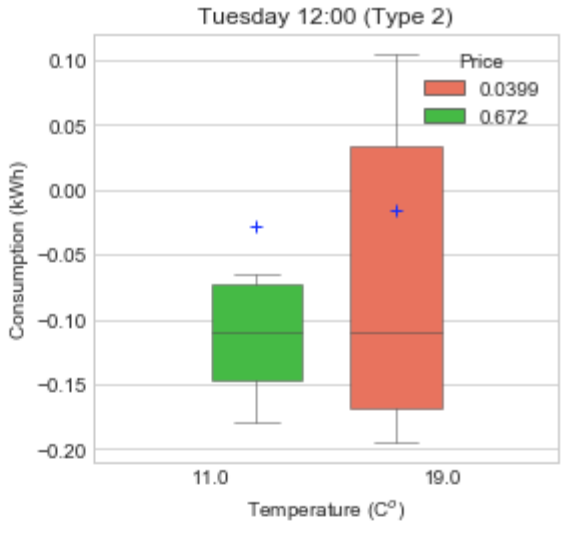

0.672
Consumption (kWh)
Temperature (C°)
11.0
18.0

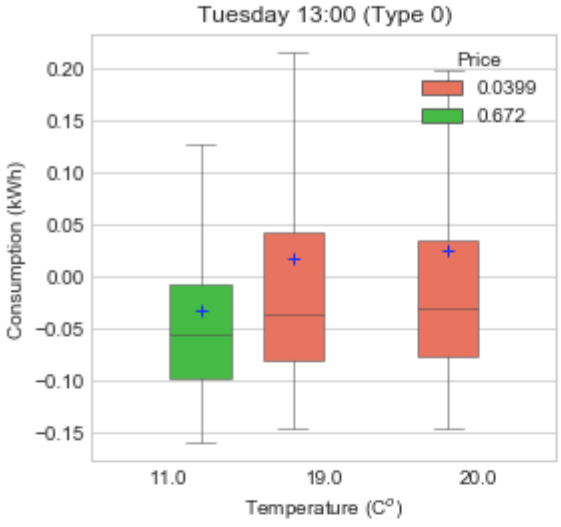

0.672
Consumption (kWh)
Temperature (C°)
11.0
18.0

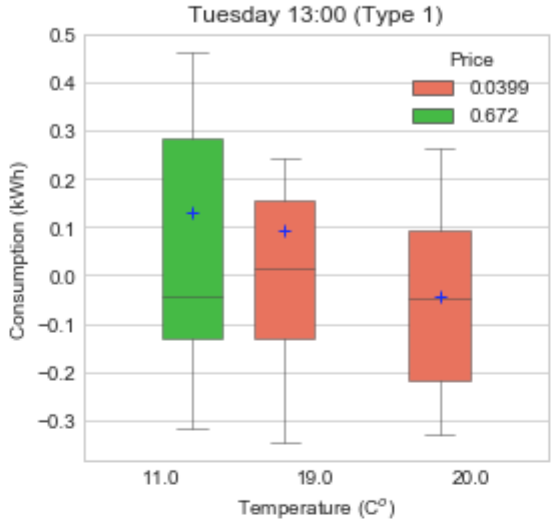

0.672
Consumption (kWh)
Temperature (C°)
11.0
18.0

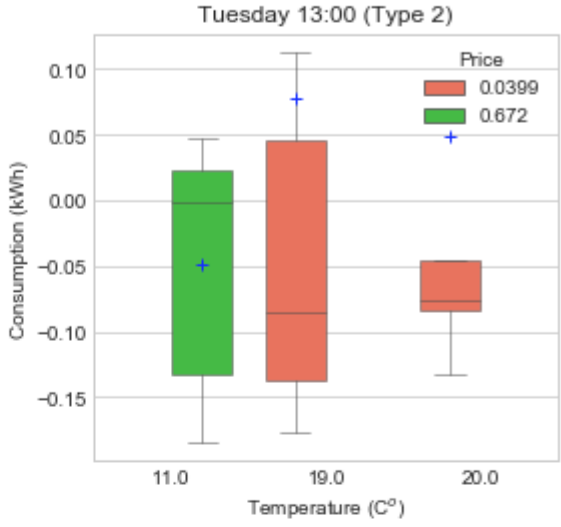

Tuesday 12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0

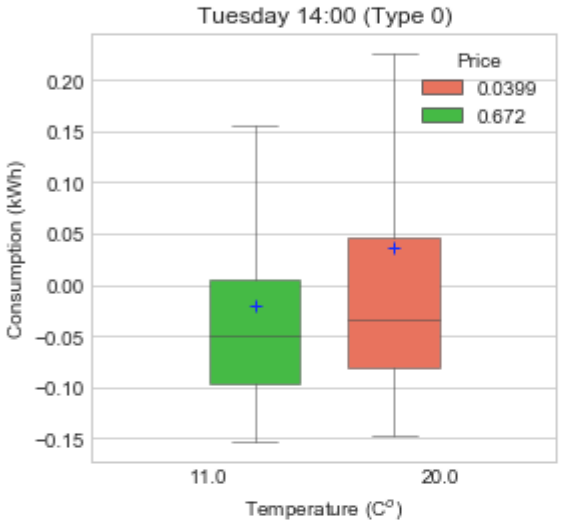

Tuesday 12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0

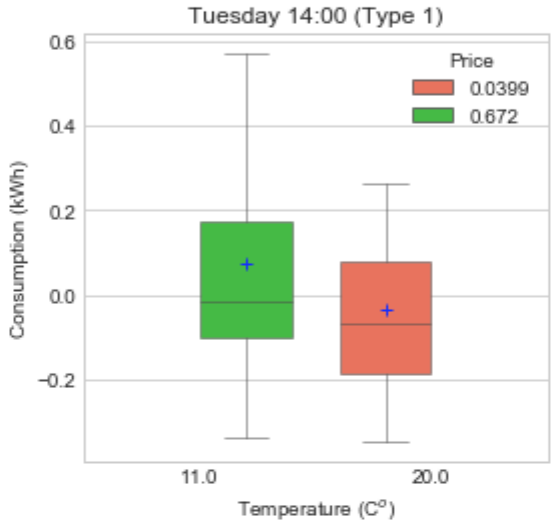

Tuesday 12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0

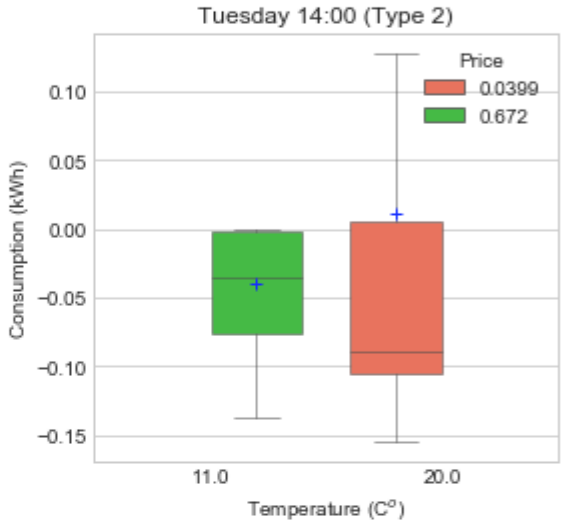

Tuesday 13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0

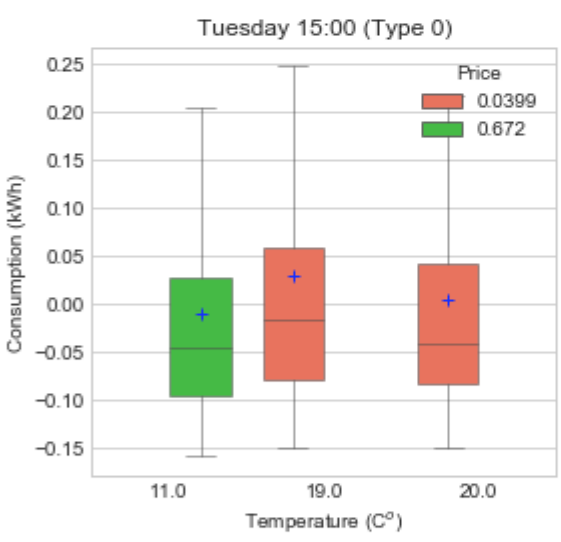

Tuesday 13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0

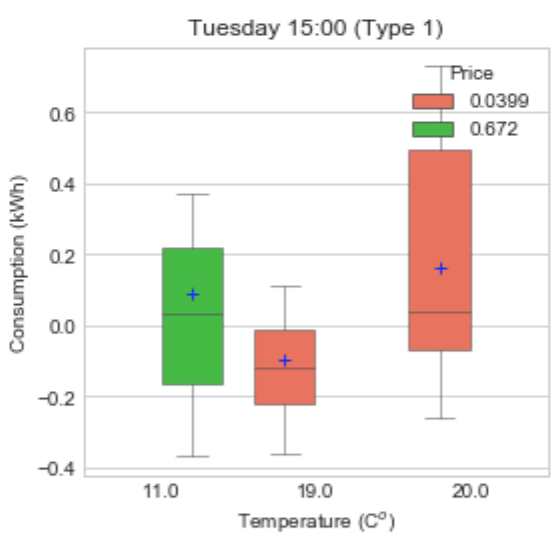

Tuesday 13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0

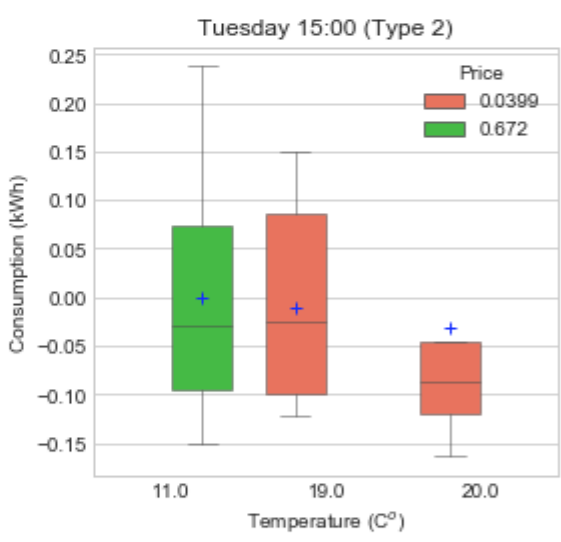

Tuesday 14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
20.0

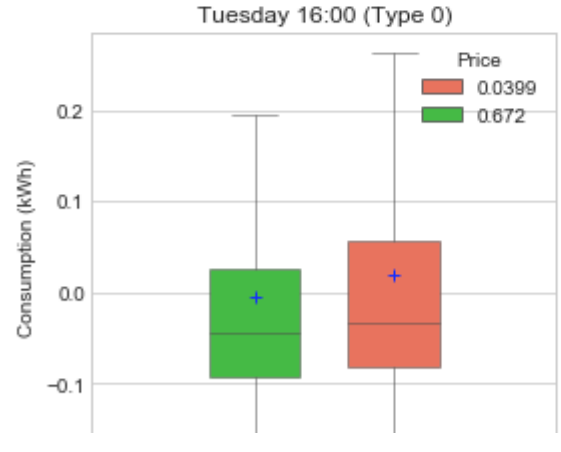

Tuesday 14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
20.0

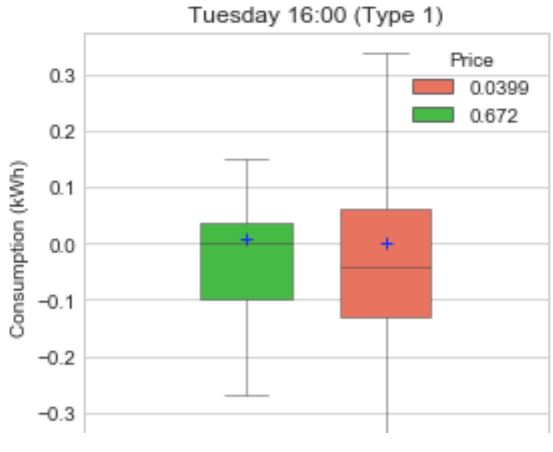

Tuesday 14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
20.0

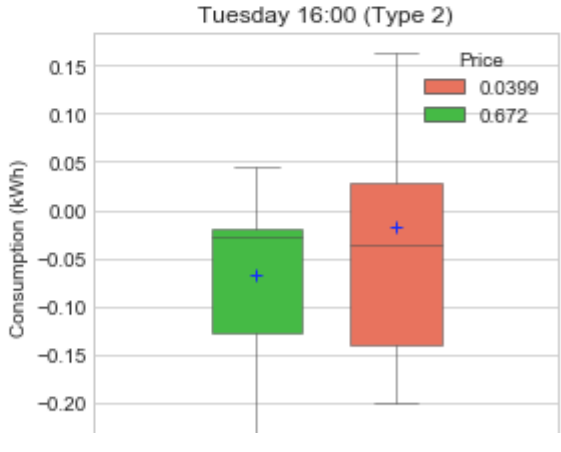

Tuesday 15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0


Tuesday 15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0

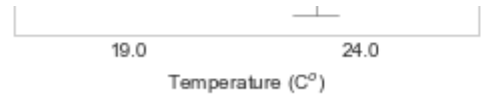

Tuesday 15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
11.0
19.0
20.0

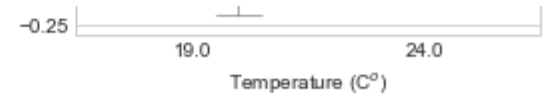

Tuesday 16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

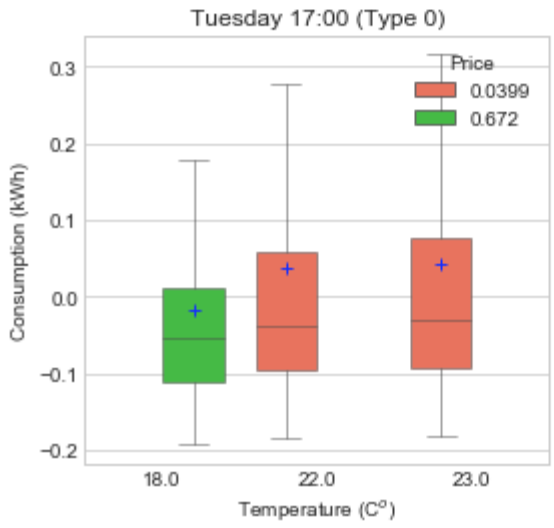

Tuesday 16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

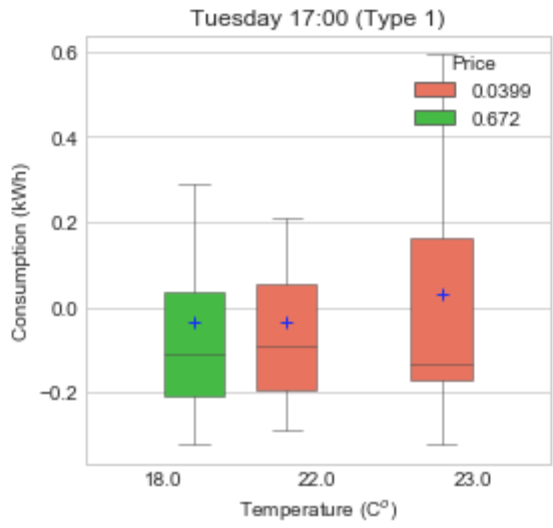

Tuesday 16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

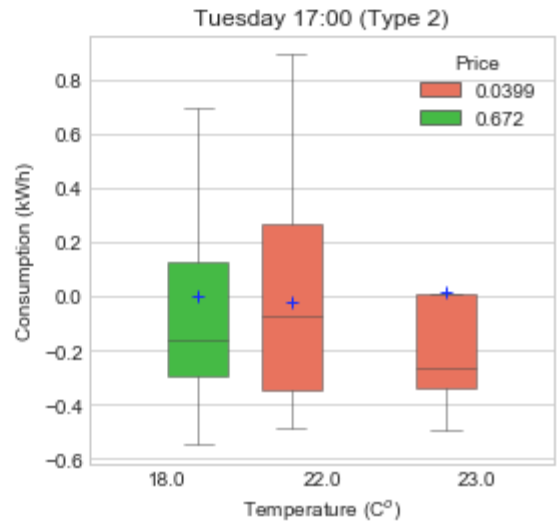

19.0
24.0
Temperature (C°)

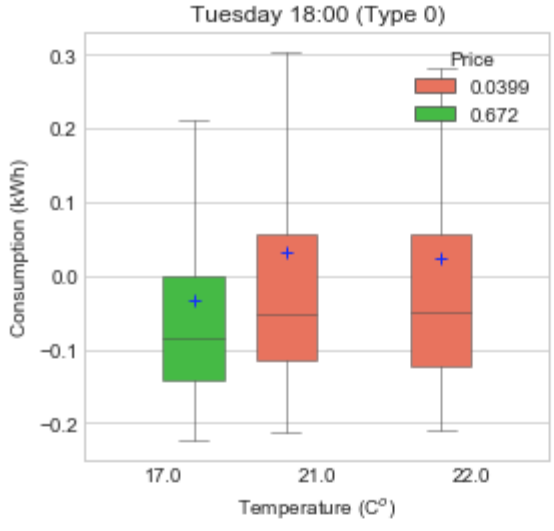

19.0
24.0
Temperature (C°)

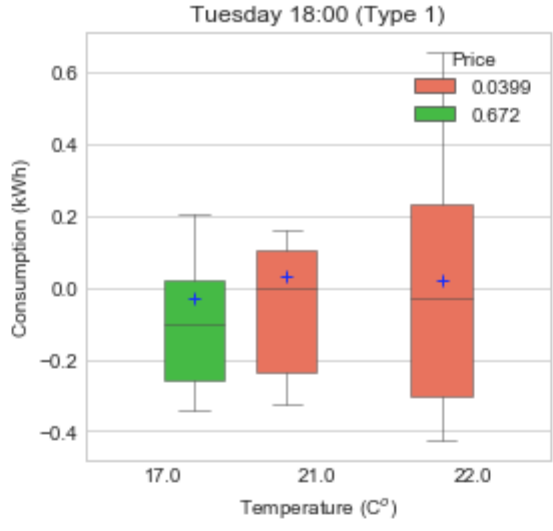

−0.25
19.0
24.0
Temperature (C°)

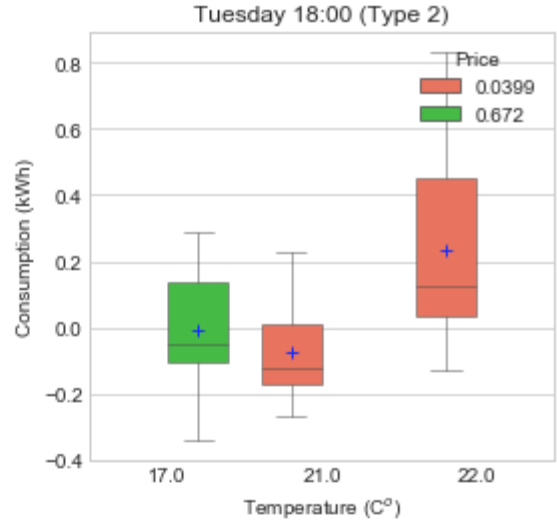

Tuesday 17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
18.0
22.0
23.0
Temperature (C°)

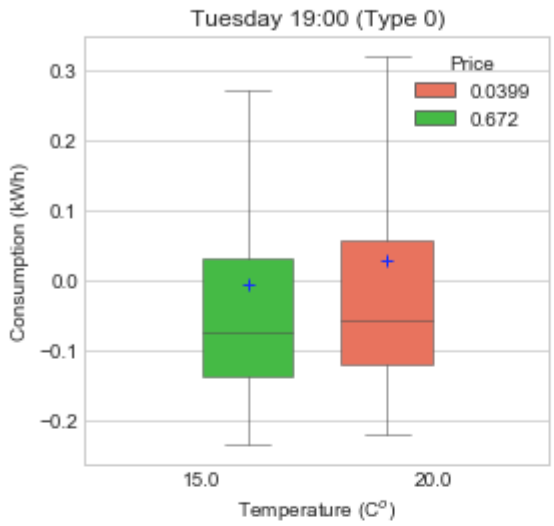

Tuesday 17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
18.0
22.0
23.0
Temperature (C°)

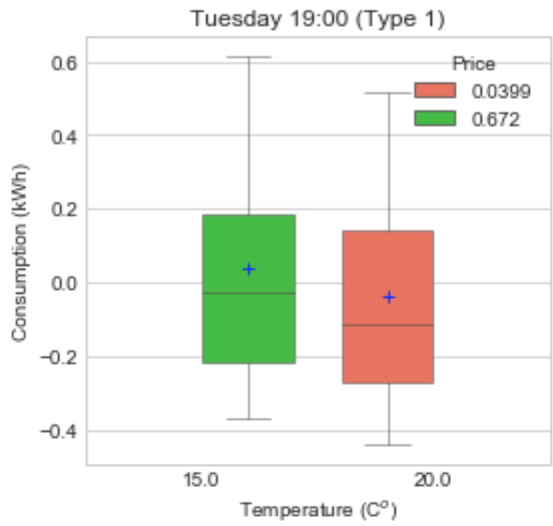

Tuesday 17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
18.0
22.0
23.0
Temperature (C°)

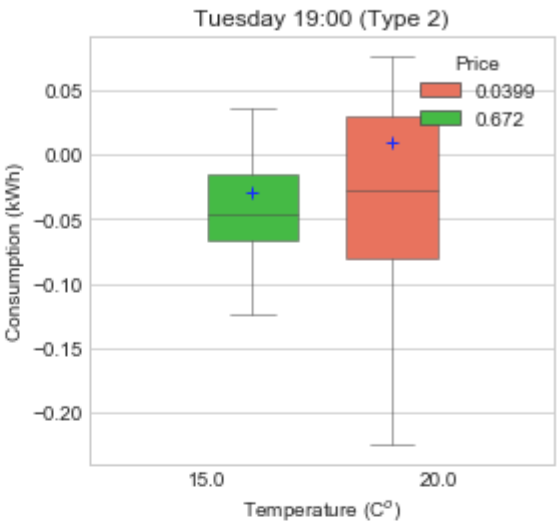

Tuesday 18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
17.0
21.0
22.0
Temperature (C°)

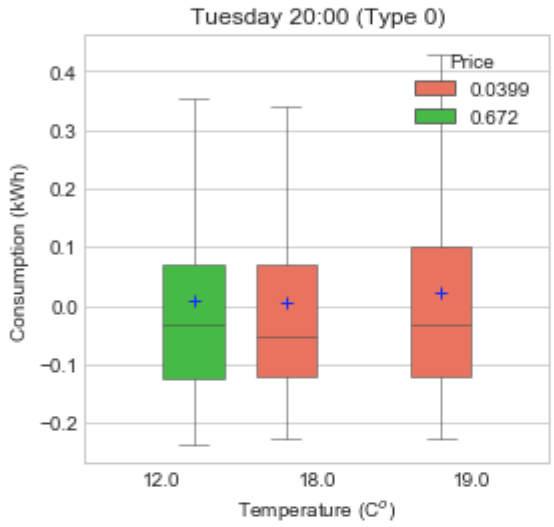

Tuesday 18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
17.0
21.0
22.0
Temperature (C°)

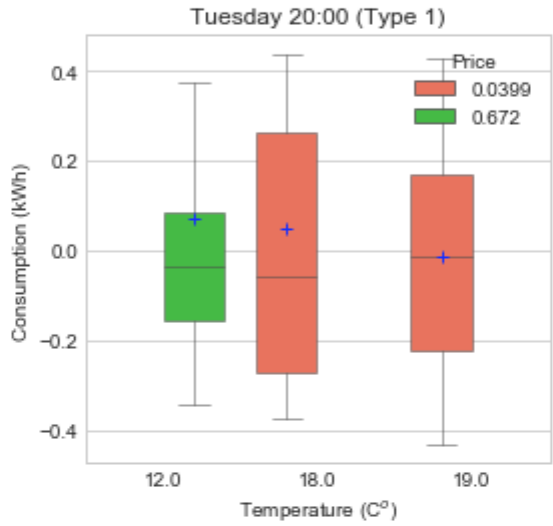

Tuesday 18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
17.0
21.0
22.0
Temperature (C°)

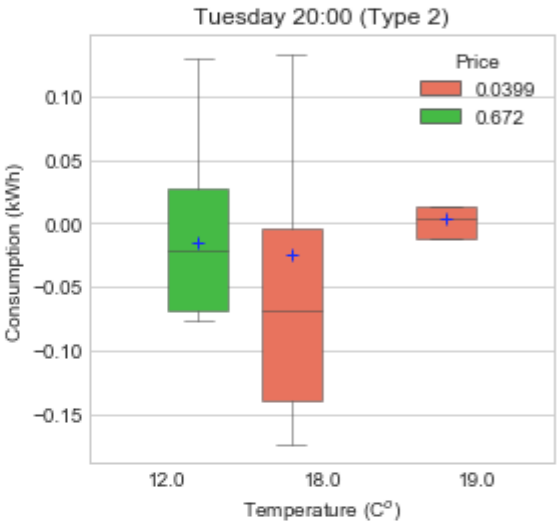

Tuesday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
15.0
20.0
Temperature (C°)

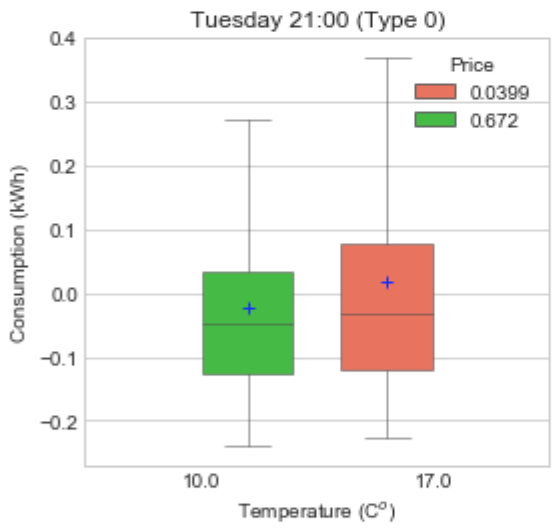

Tuesday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
20.0
Temperature (C°)

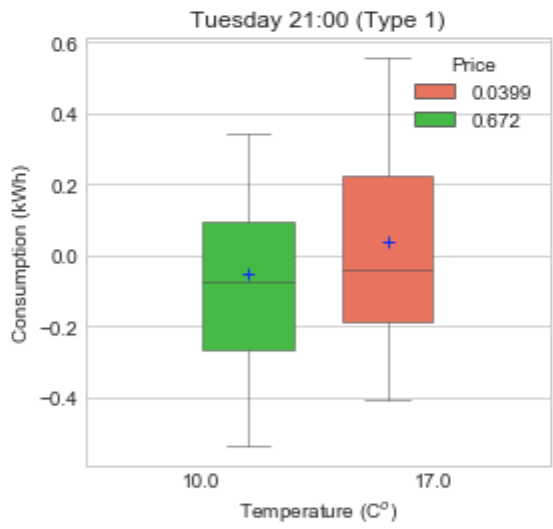

Tuesday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.05
0.00
−0.05
−0.10
−0.15
−0.20
15.0
20.0
Temperature (C°)

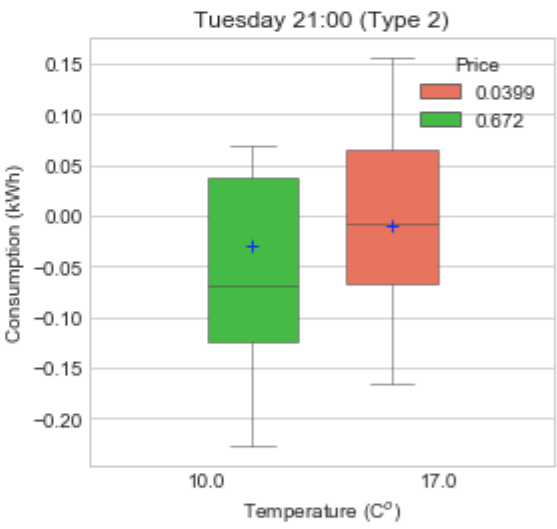

Tuesday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
12.0
18.0
19.0
Temperature (C°)

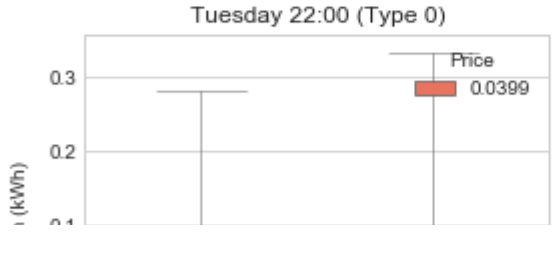

Tuesday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
12.0
18.0
19.0
Temperature (C°)

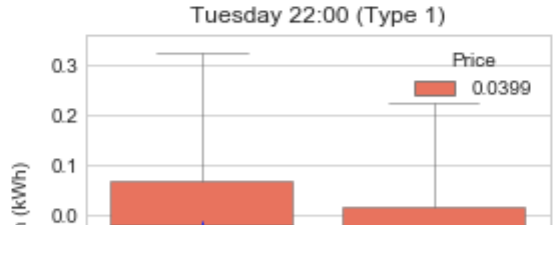

Tuesday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
12.0
18.0
19.0
Temperature (C°)

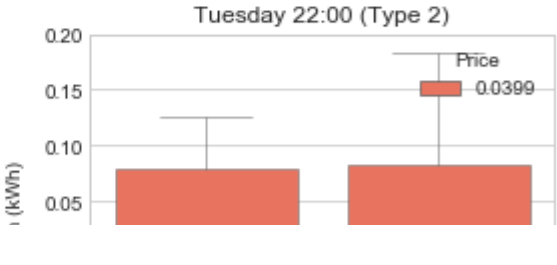

Tuesday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
10.0
17.0
Temperature (C°)

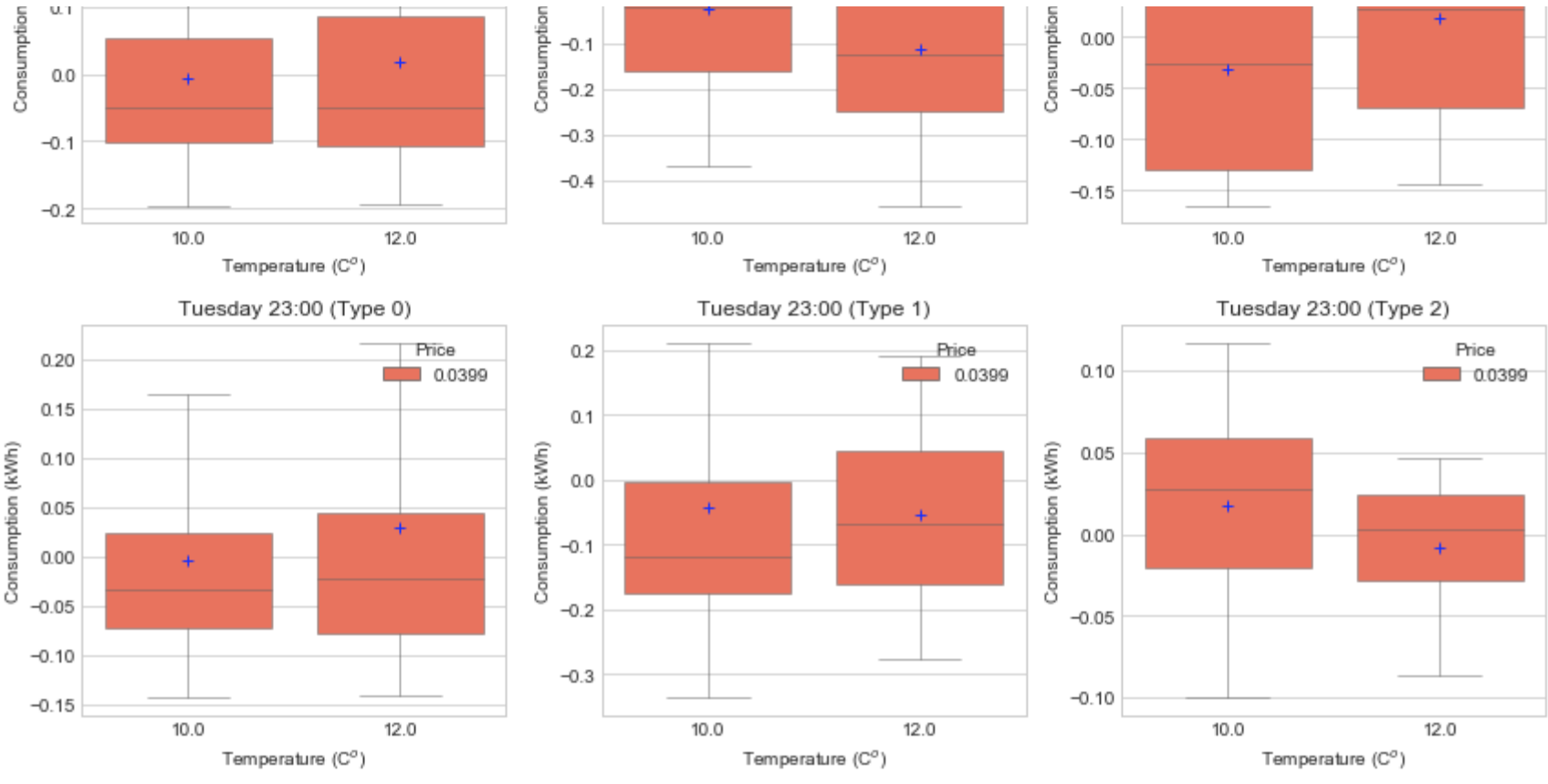

Tuesday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
17.0
Temperature (C°)


Tuesday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
10.0
17.0
Temperature (C°)

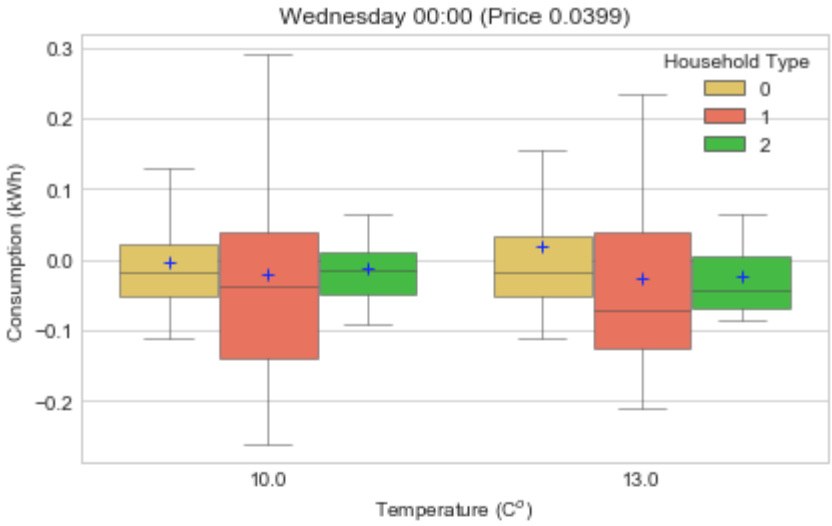

Tuesday 22:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
Temperature (C°)

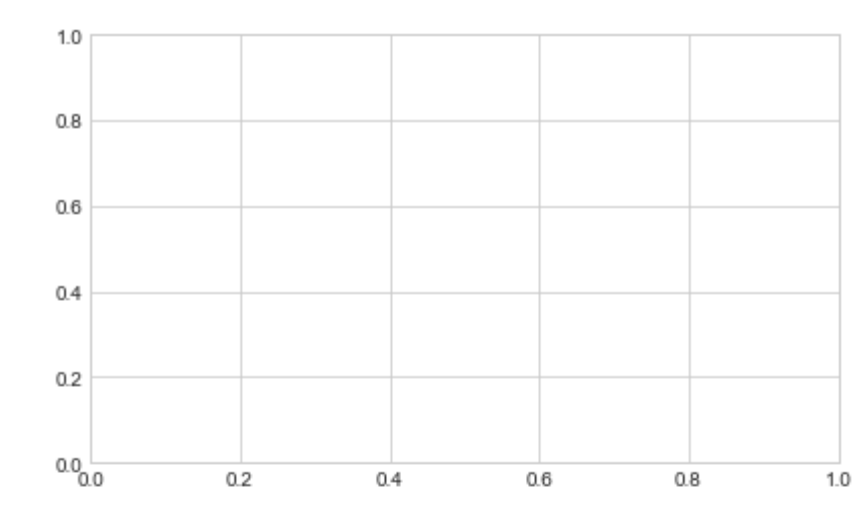

Tuesday 22:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
Temperature (C°)

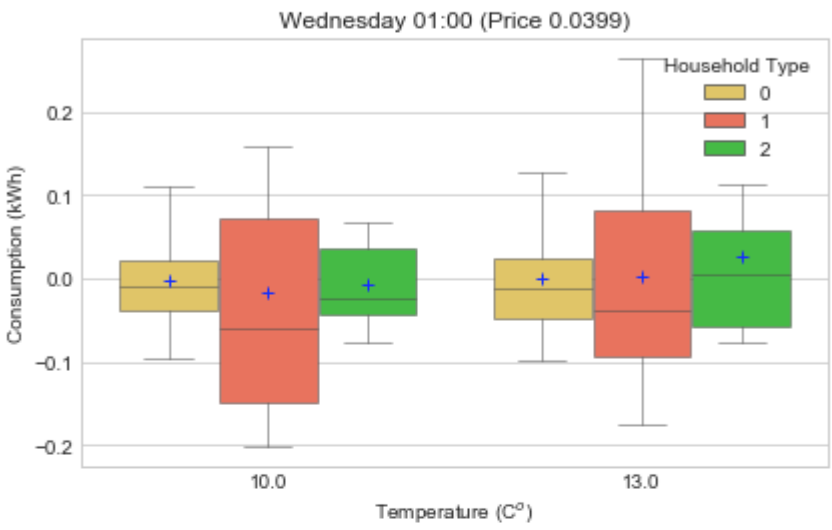

Tuesday 22:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
Temperature (C°)

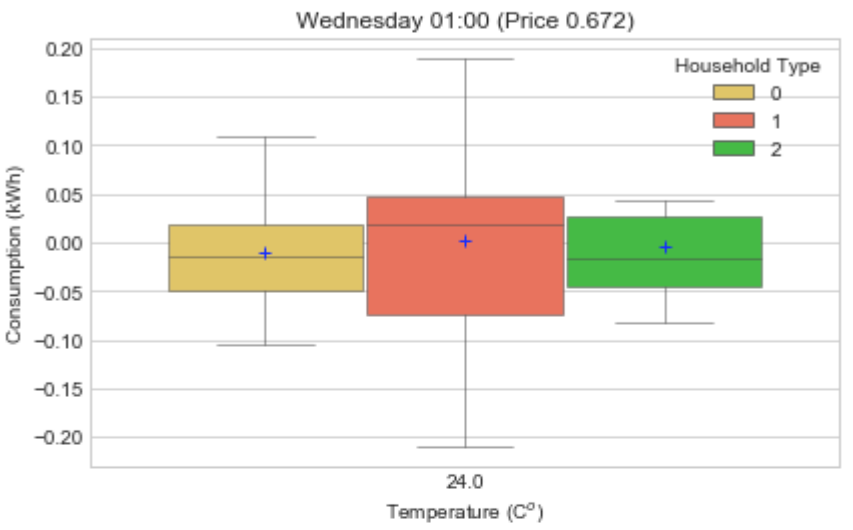

Consumption
Temperature (C°)
10.0
12.0
Tuesday 23:00 (Type 0)
Tuesday 23:00 (Type 1)
Tuesday 23:00 (Type 2)
Consumption (kWh)
Price
0.0399

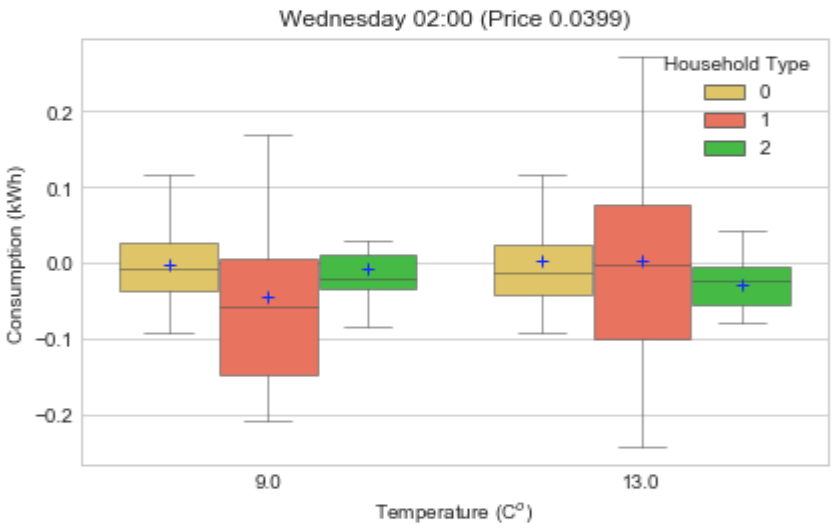
In [65]:
```
# Wednesday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 2 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

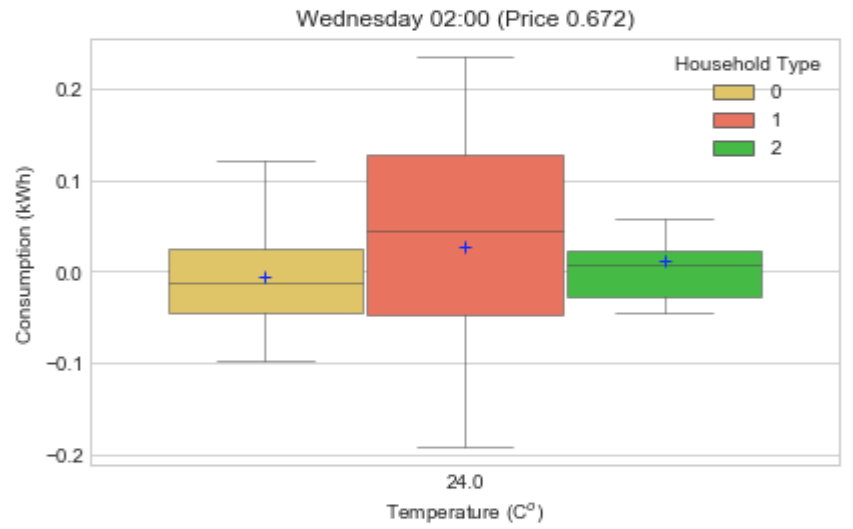

Wednesday 00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
10.0
13.0
Temperature (C°)

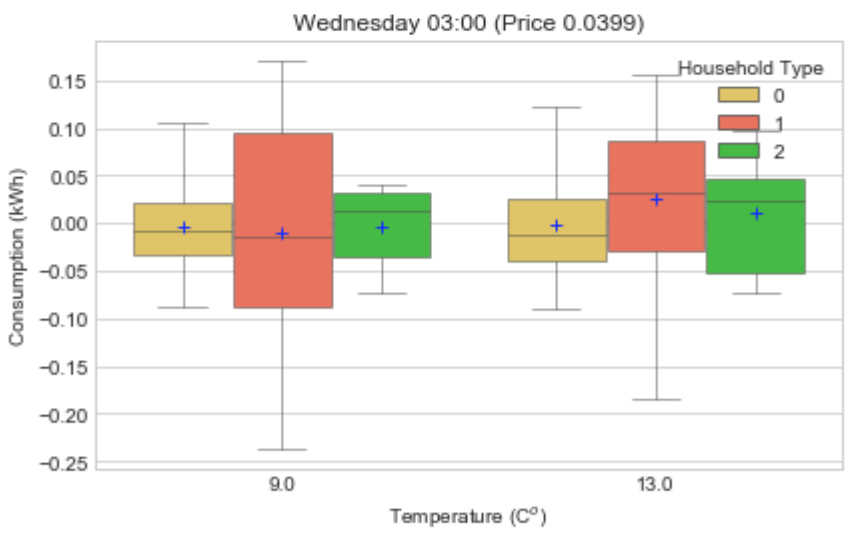

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

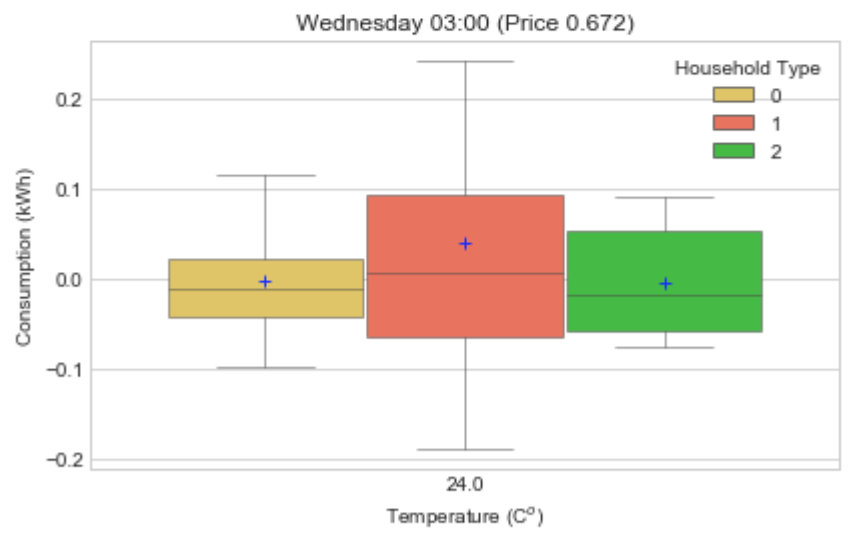

Wednesday 01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
10.0
13.0
Temperature (C°)

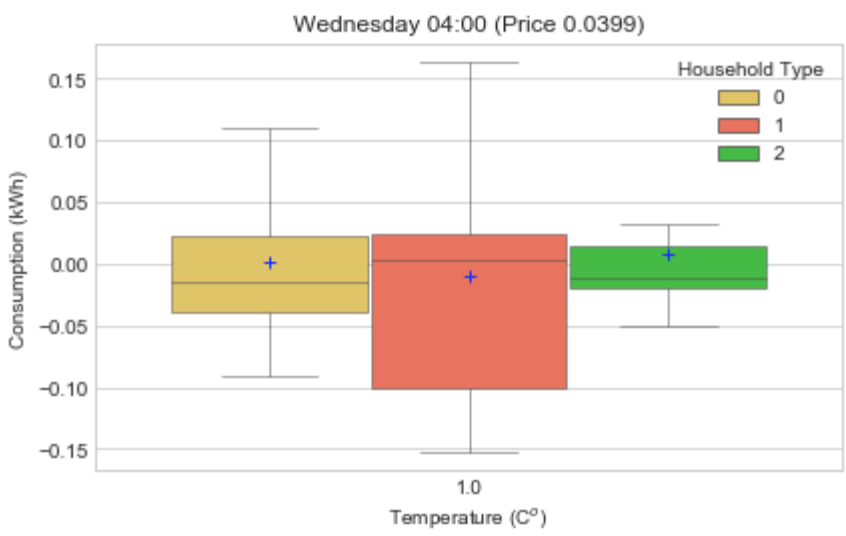

Wednesday 01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
24.0
Temperature (C°)

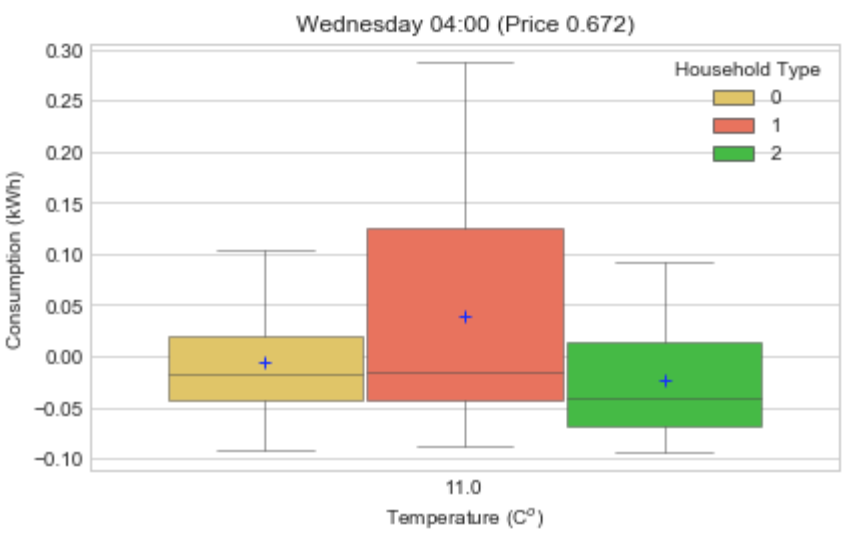

Wednesday 02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
9.0
13.0
Temperature (C°)


Wednesday 02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
24.0
Temperature (C°)


Wednesday 03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
9.0
13.0
Temperature (C°)

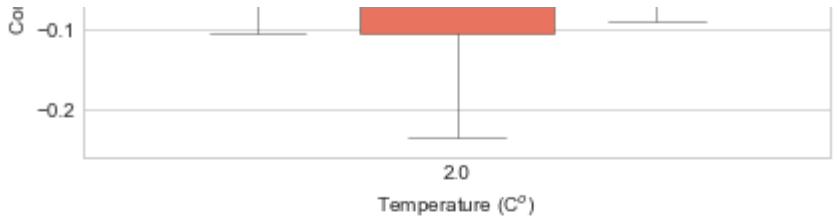

Wednesday 03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
24.0
Temperature (C°)

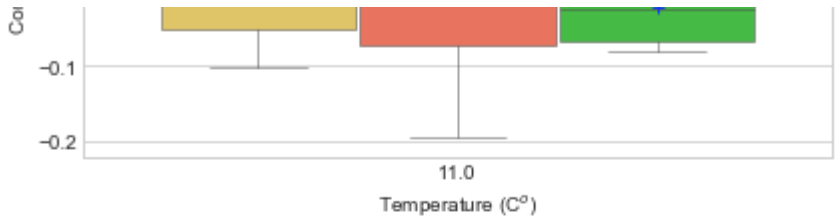

Wednesday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
1.0
Temperature (C°)

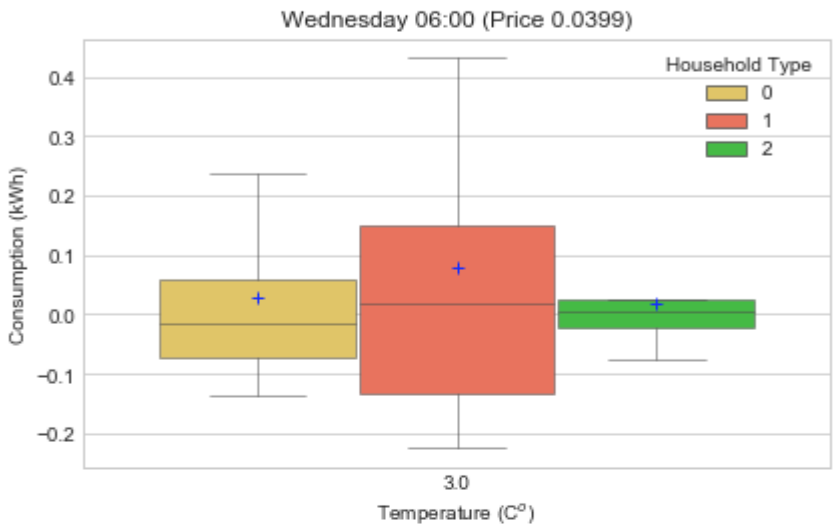

Wednesday 04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
11.0
Temperature (C°)

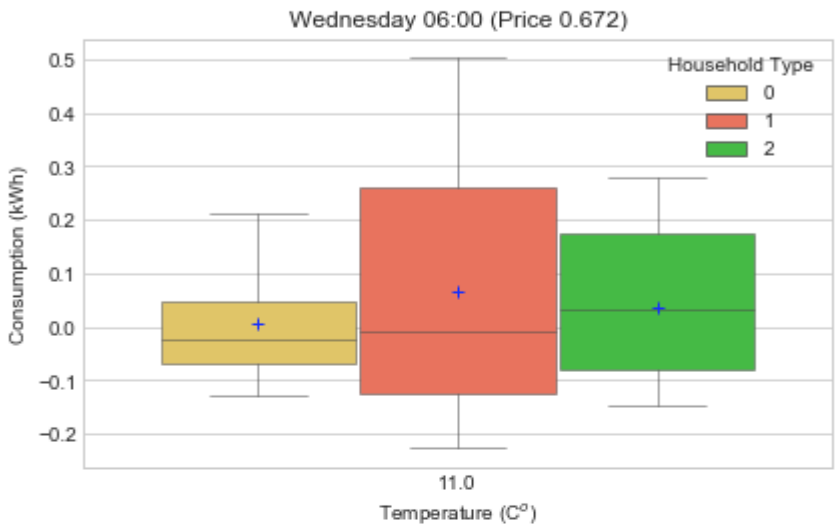
Wednesday 05:00 (Price 0.0399)
Household Type
0
1
2
0.2
0.1
0.0

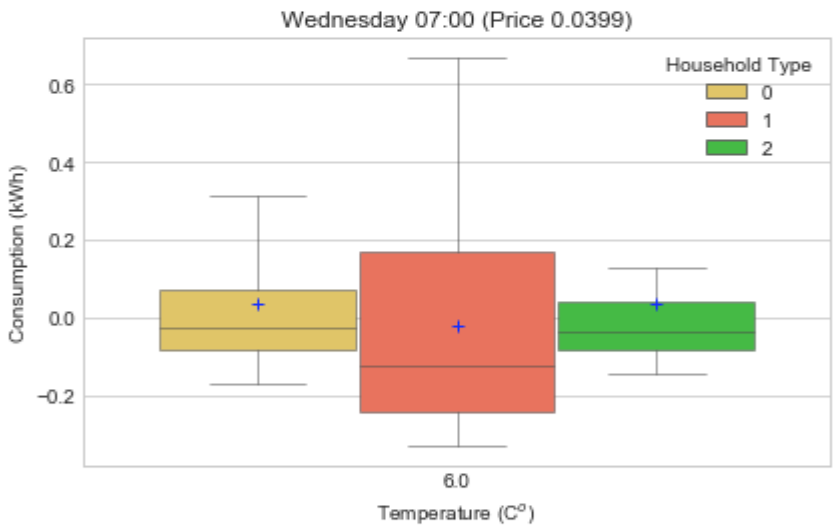
Wednesday 05:00 (Price 0.672)
Household Type
0
1
2
0.3
0.2
0.1
0.0

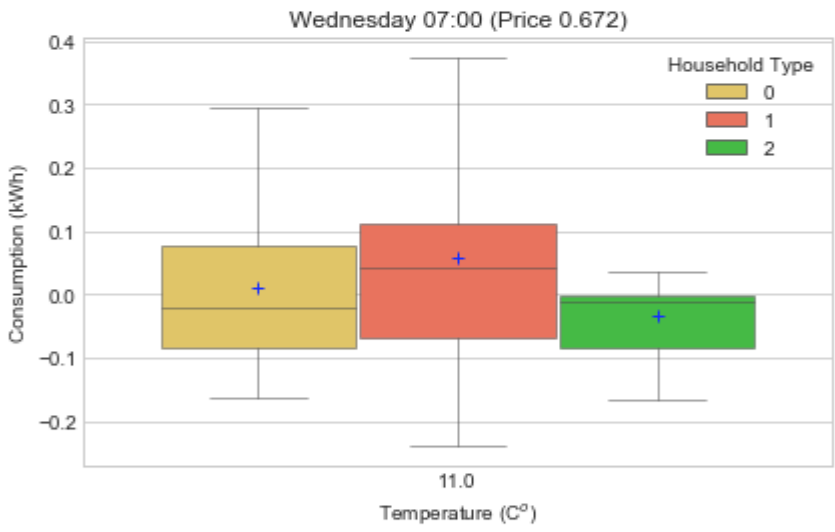

−0.1
−0.2
2.0
Temperature (C°)

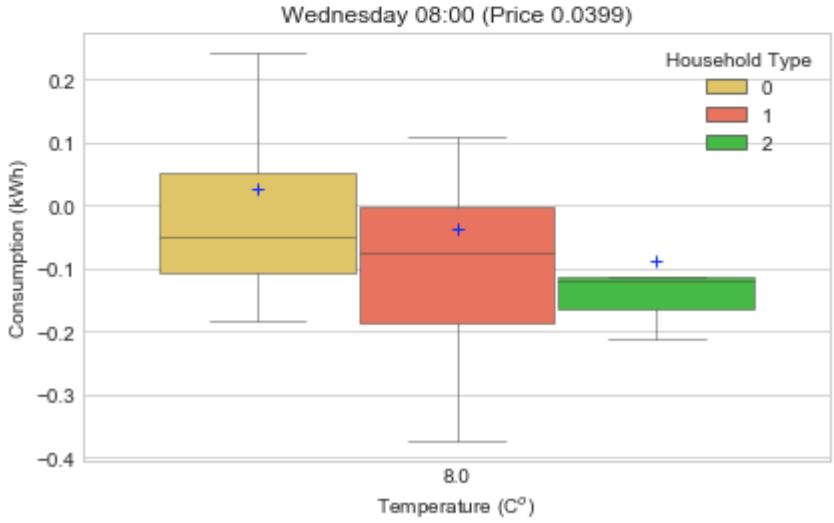

−0.1
−0.2
11.0
Temperature (C°)

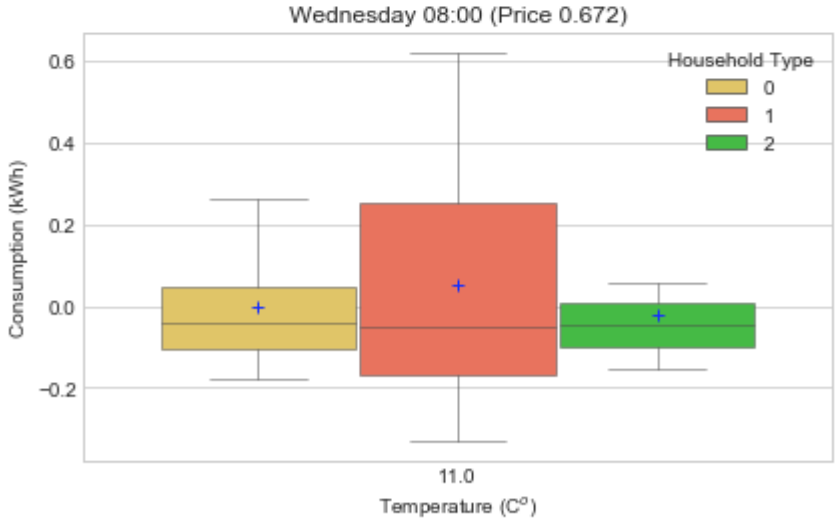

Wednesday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
Temperature (C°)

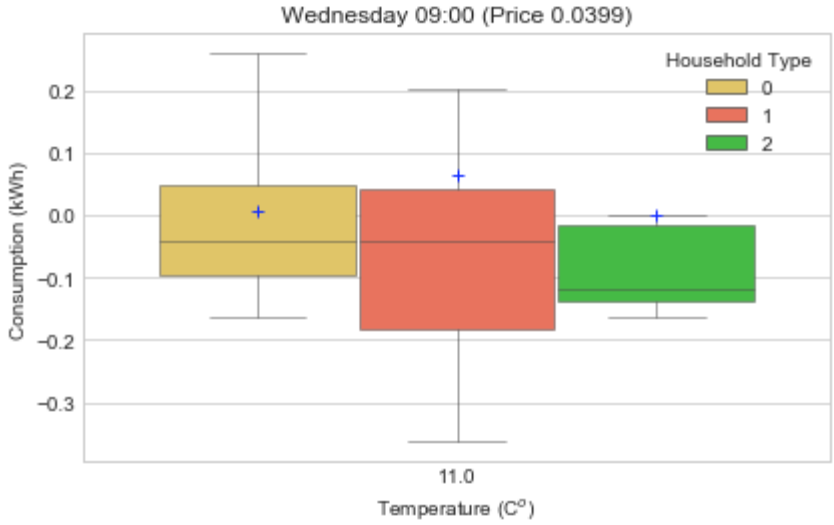

Wednesday 06:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

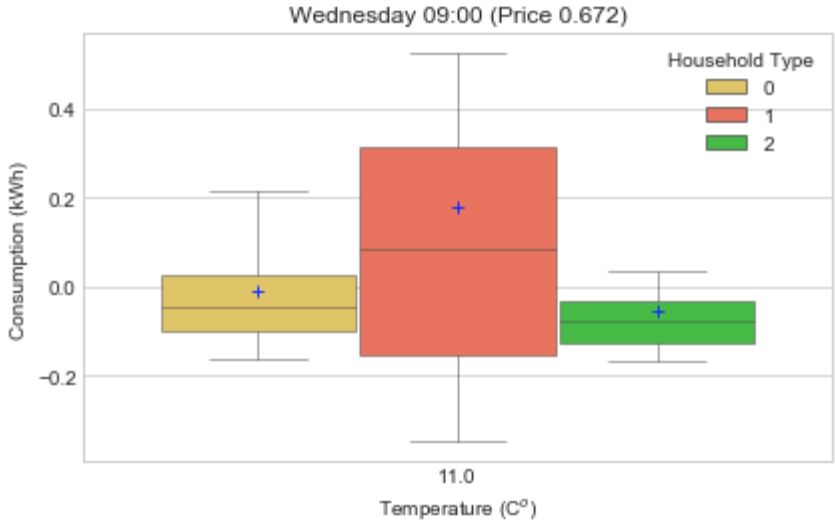

Wednesday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
6.0
Temperature (C°)

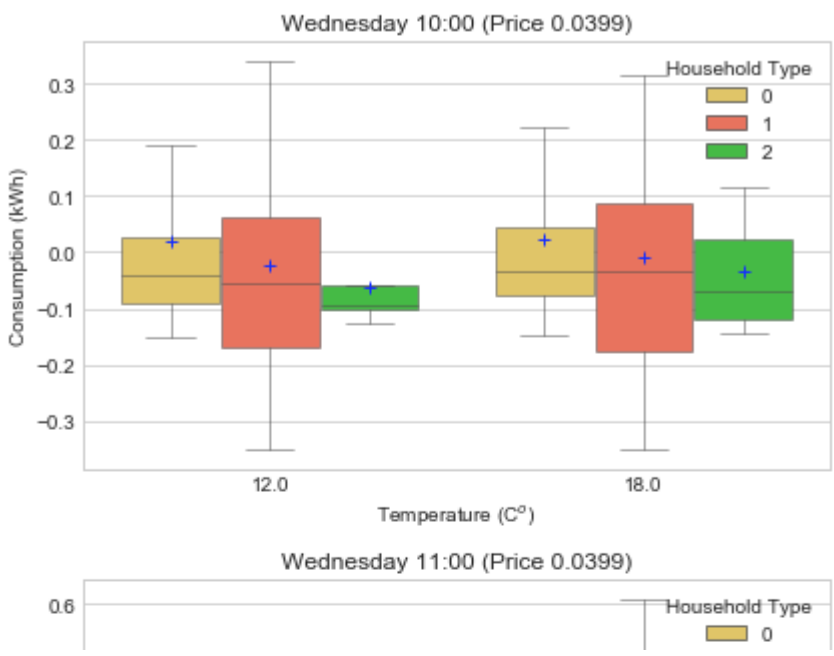

Wednesday 07:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
11.0
Temperature (C°)

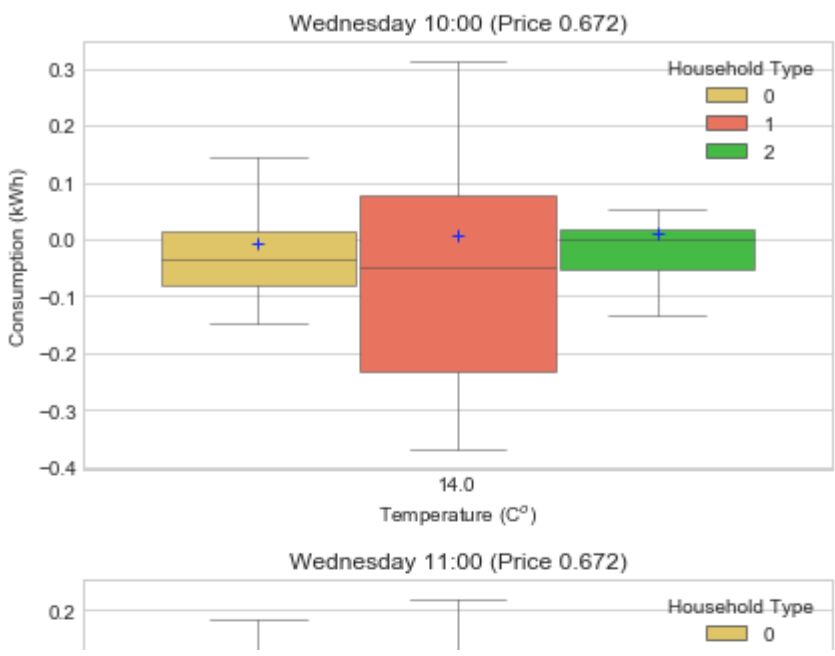

Wednesday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
8.0
Temperature (C°)


Wednesday 08:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
11.0
Temperature (C°)


Wednesday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
11.0
Temperature (C°)

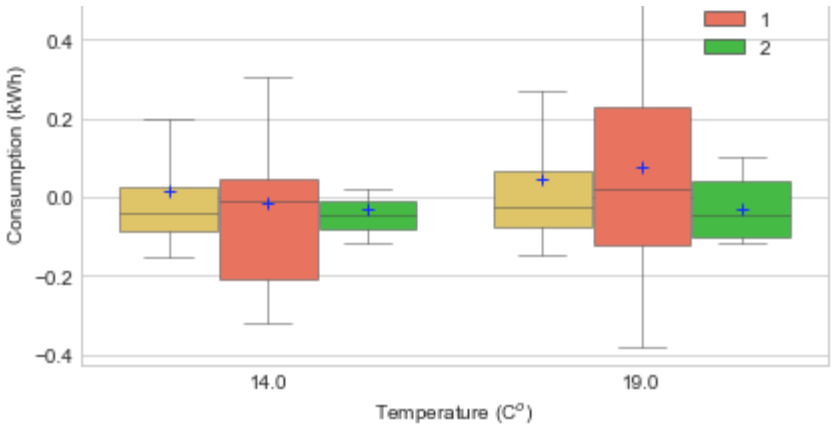

Wednesday 09:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
11.0
Temperature (C°)

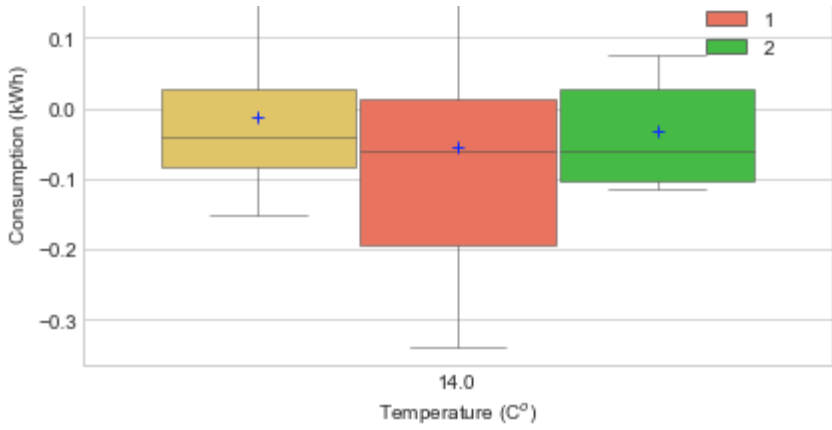

Wednesday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
12.0
18.0
Temperature (C°)

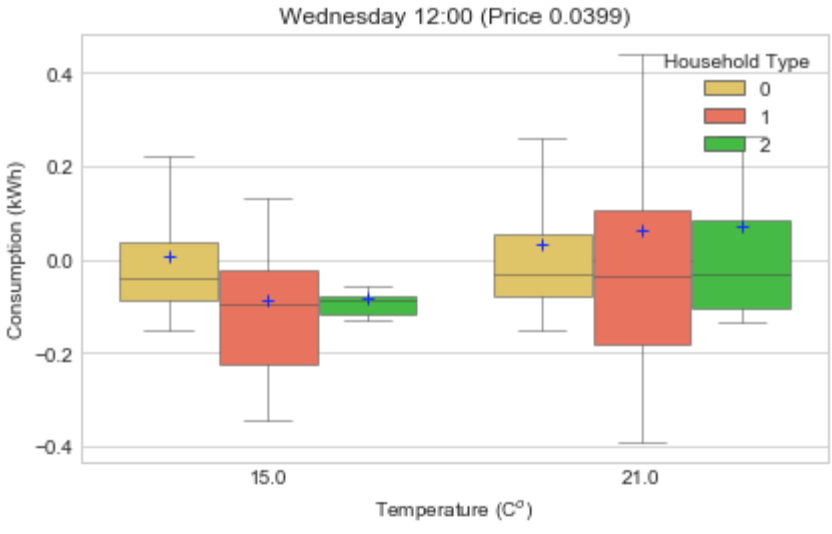

Wednesday 10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
14.0
Temperature (C°)

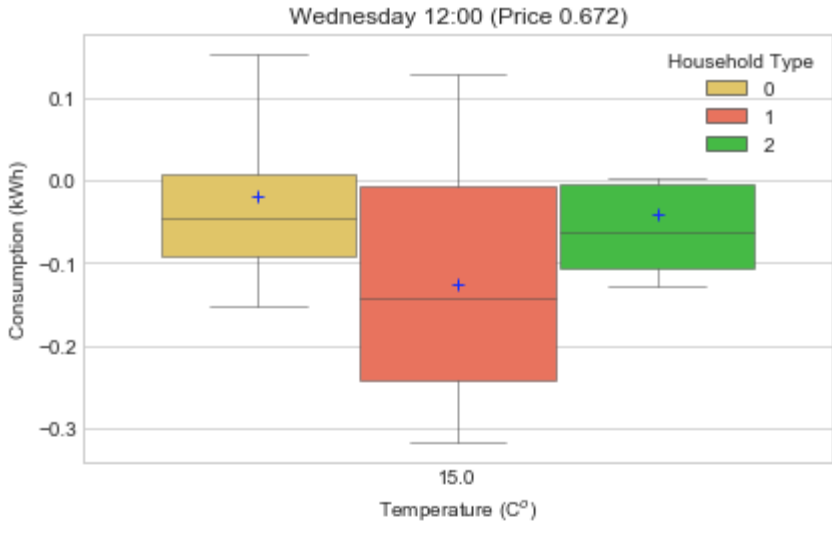
Wednesday 11:00 (Price 0.0399)
Household Type
0
0.6

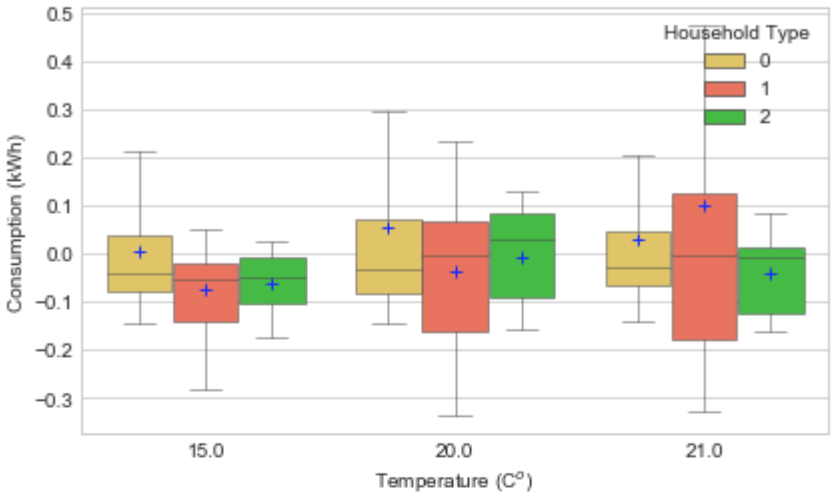
Wednesday 11:00 (Price 0.672)
Household Type
0
0.2

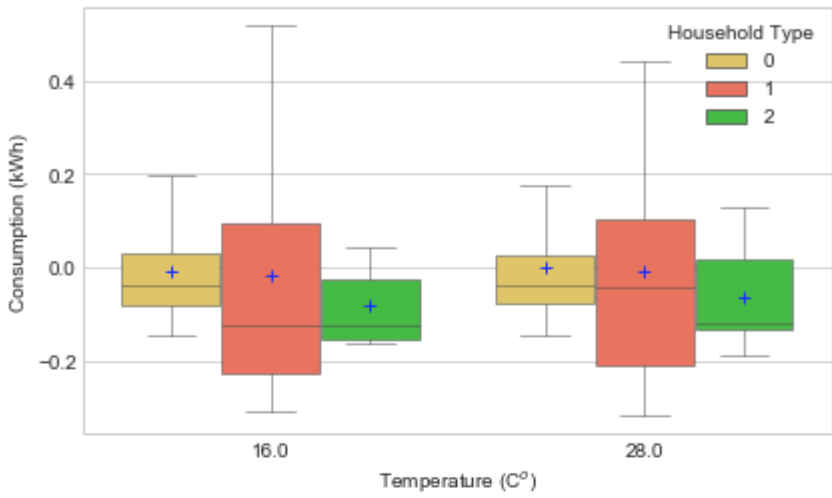

1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
14.0
19.0
Temperature (C°)

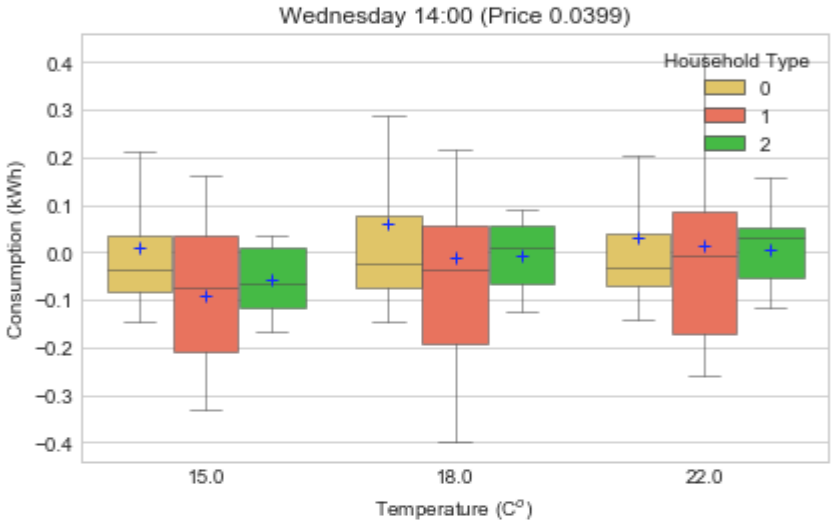

1
2
Consumption (kWh)
0.1
0.0
−0.1
−0.2
−0.3
14.0
Temperature (C°)

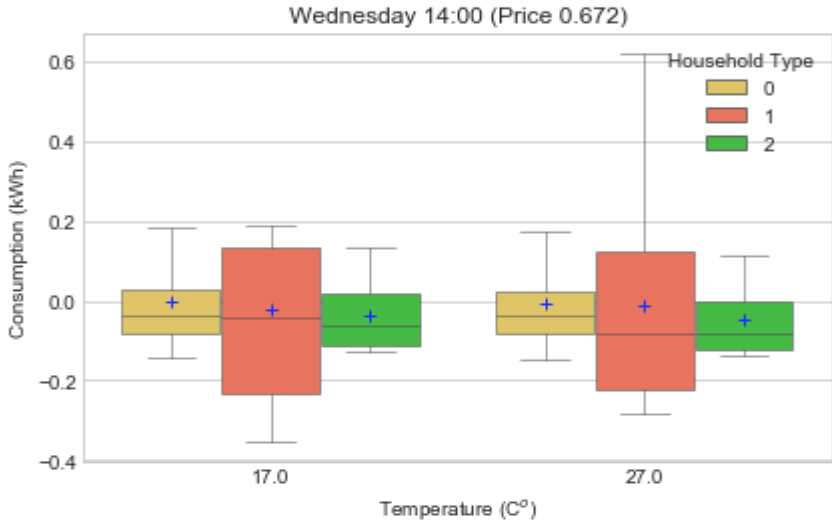

Wednesday 12:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
15.0
21.0
Temperature (C°)

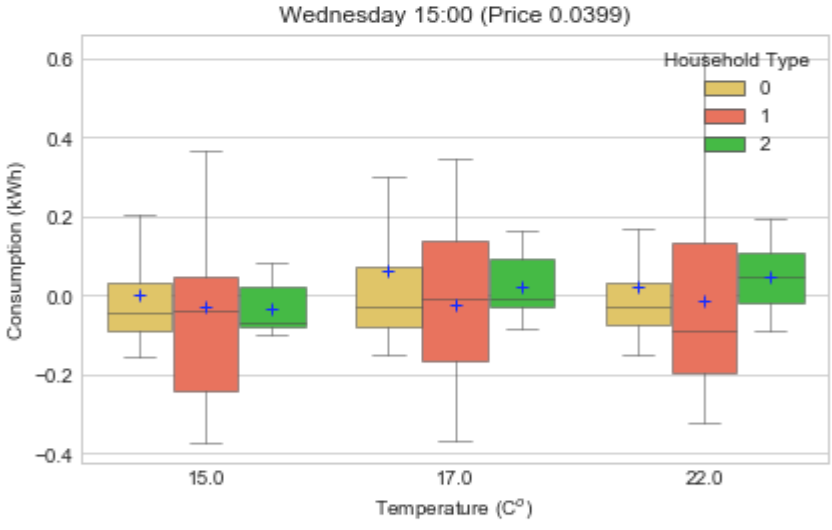

Wednesday 12:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.1
0.0
−0.1
−0.2
−0.3
15.0
Temperature (C°)

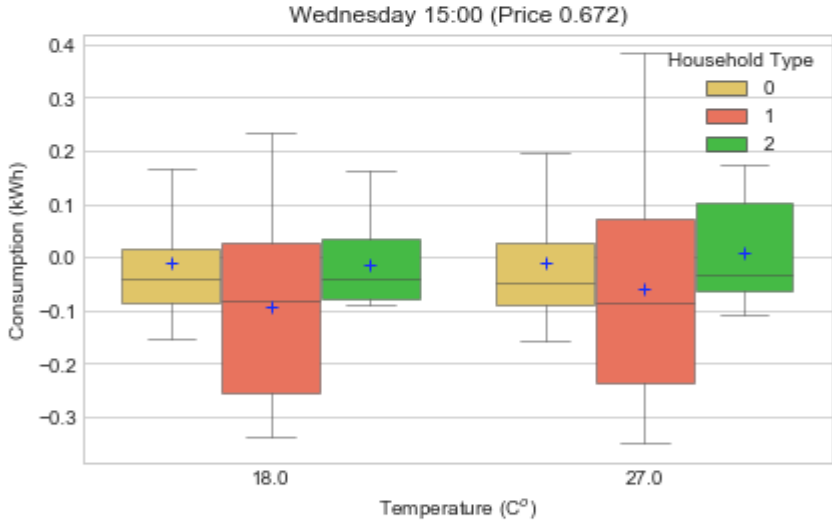

Wednesday 13:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
15.0
20.0
21.0
Temperature (C°)

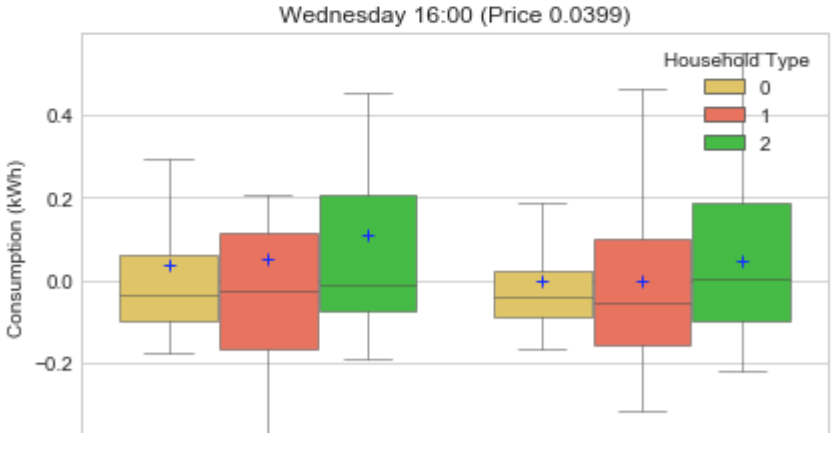

Wednesday 13:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
16.0
28.0
Temperature (C°)

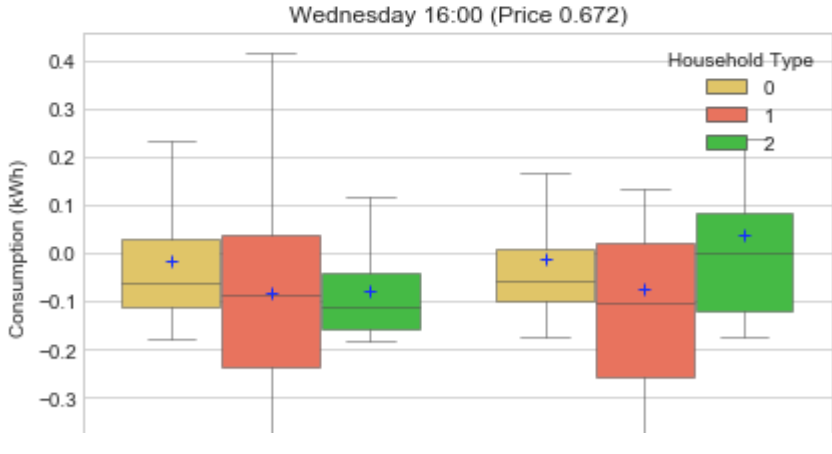

Wednesday 14:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
15.0
18.0
22.0
Temperature (C°)


Wednesday 14:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
17.0
27.0
Temperature (C°)


Wednesday 15:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
17.0
22.0
Temperature (C°)

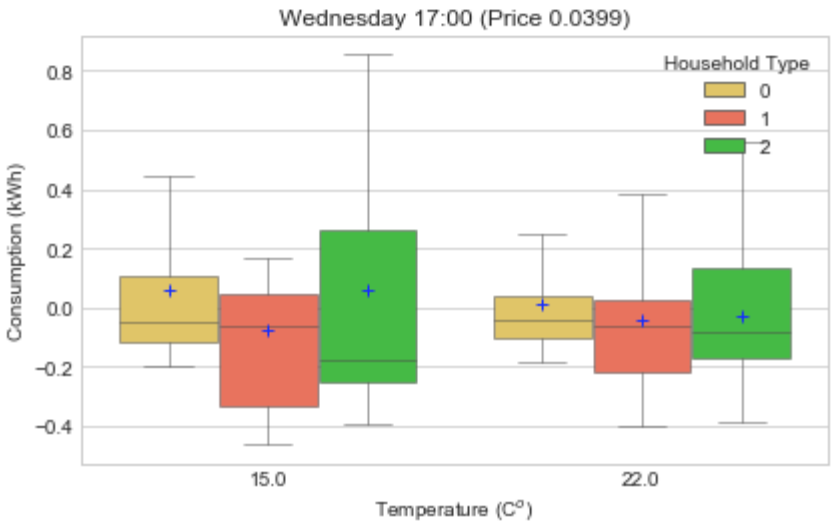

Wednesday 15:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
18.0
27.0
Temperature (C°)

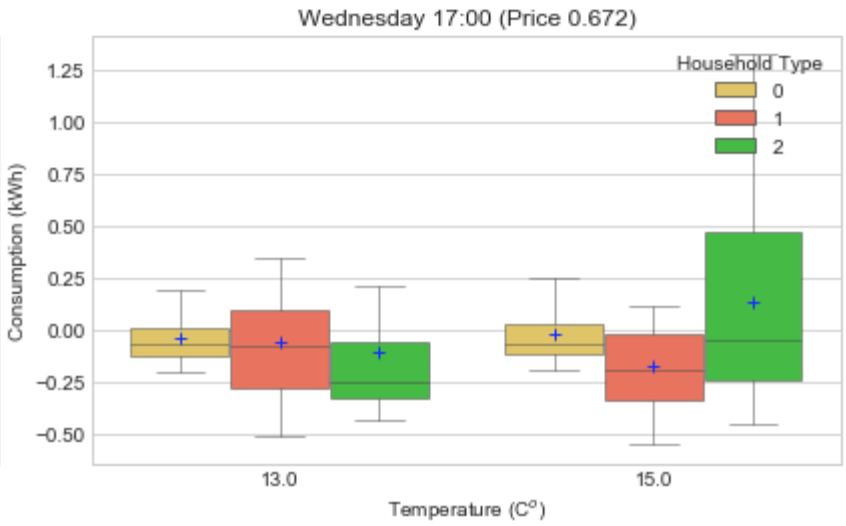

Wednesday 16:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2

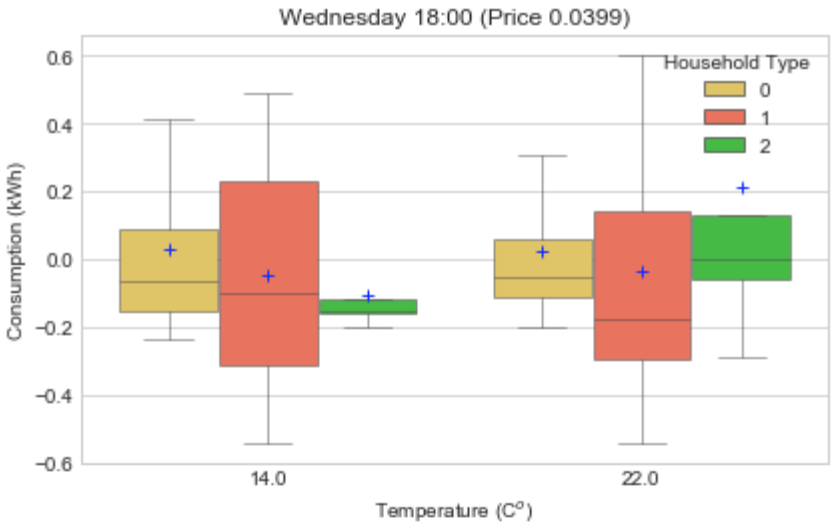

Wednesday 16:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3

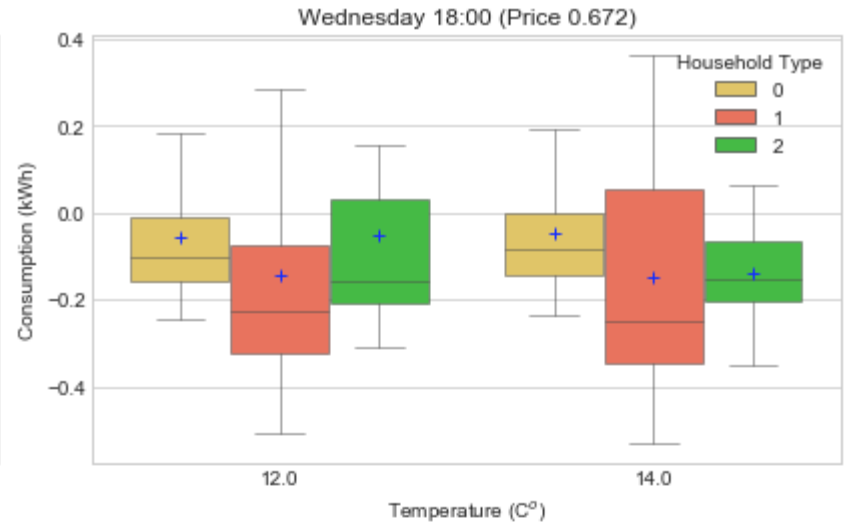

−0.4
15.0
22.0
Temperature (C°)

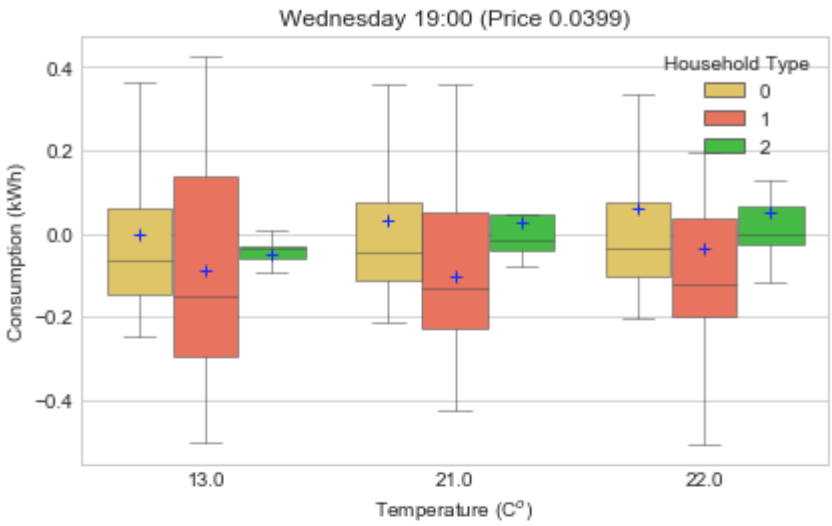

−0.4
14.0
17.0
Temperature (C°)

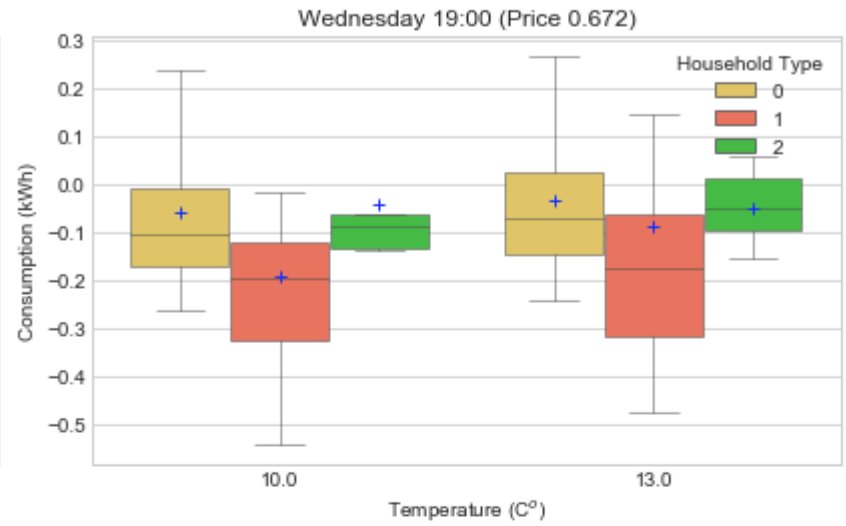

Wednesday 17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
22.0
Temperature (C°)

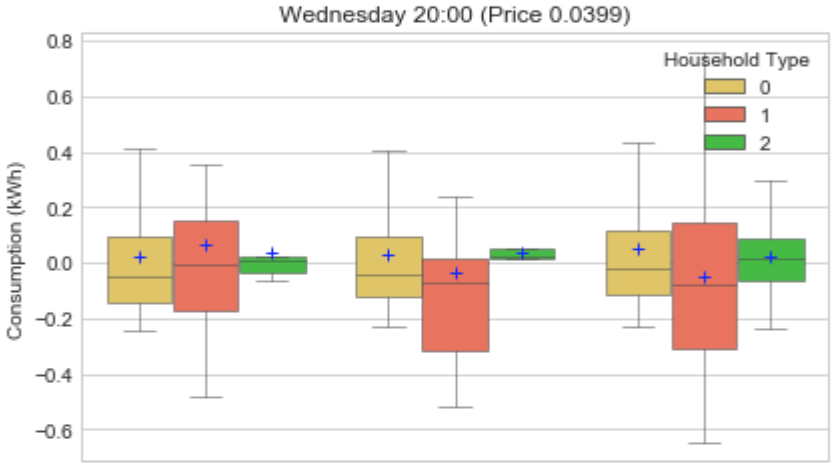

Wednesday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
13.0
15.0
Temperature (C°)

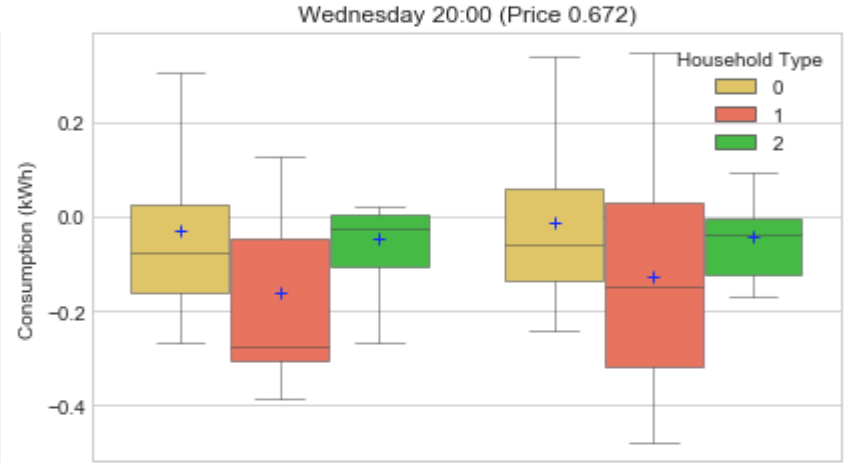

Wednesday 18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
14.0
22.0
Temperature (C°)

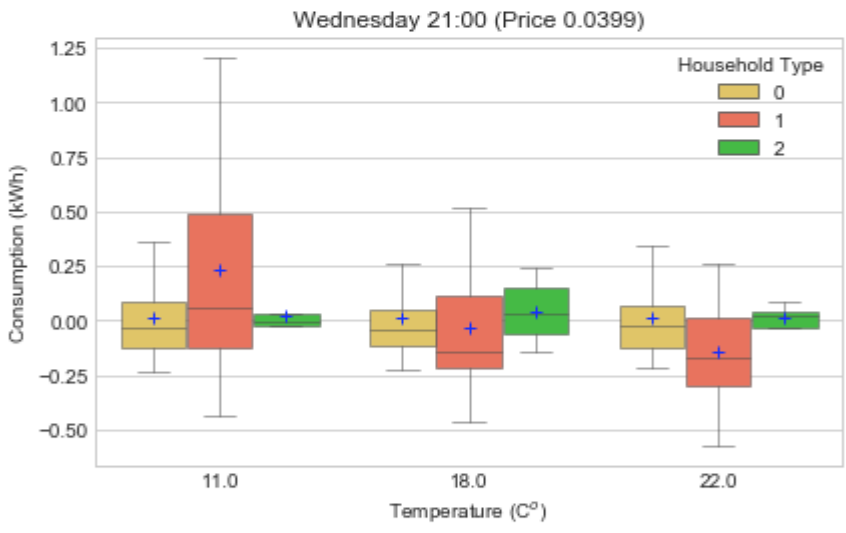

Wednesday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
12.0
14.0
Temperature (C°)

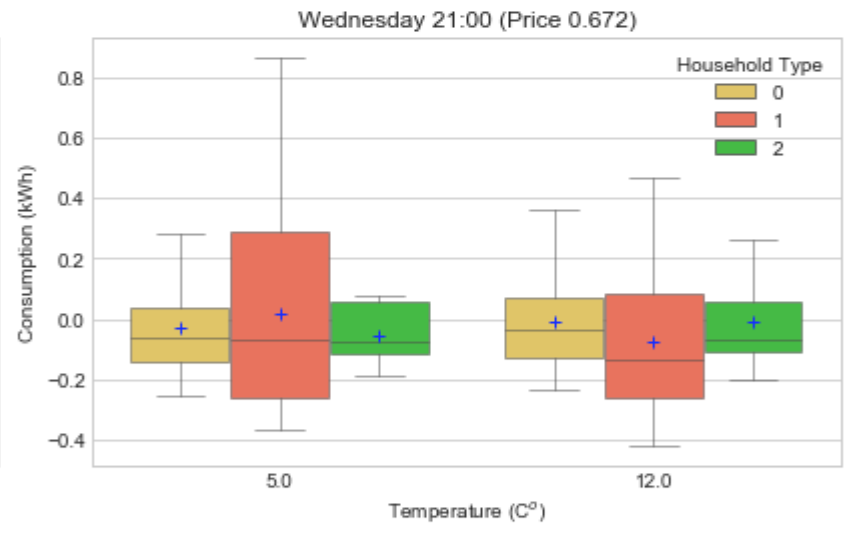

Wednesday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
13.0
21.0
22.0
Temperature (C°)


Wednesday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
10.0
13.0
Temperature (C°)


Wednesday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
12.0
19.0
22.0
Temperature (C°)

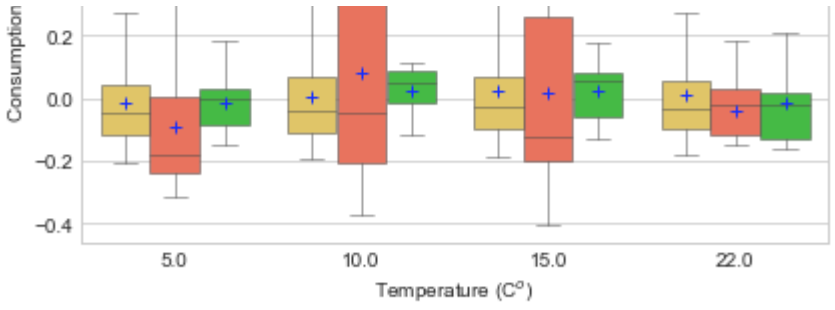

Wednesday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.0
−0.2
−0.4
7.0
13.0
Temperature (C°)

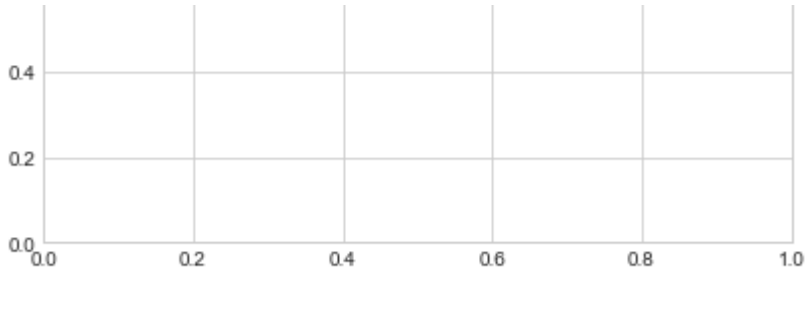

Wednesday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
11.0
18.0
22.0
Temperature (C°)

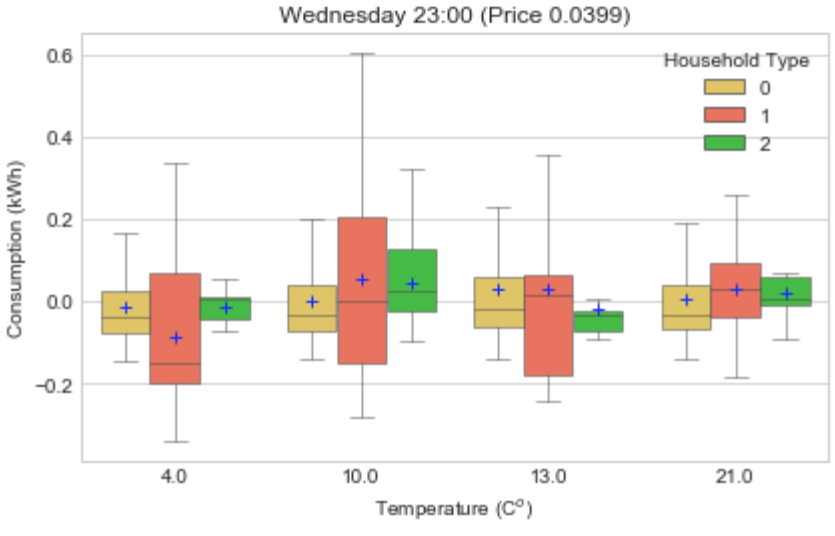

Wednesday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
5.0
12.0
Temperature (C°)

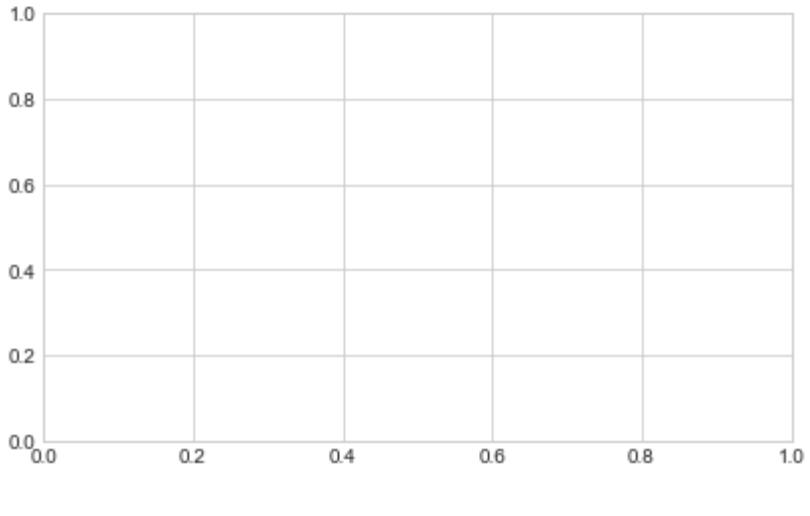
Wednesday 22:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4

1.0
0.8
0.6

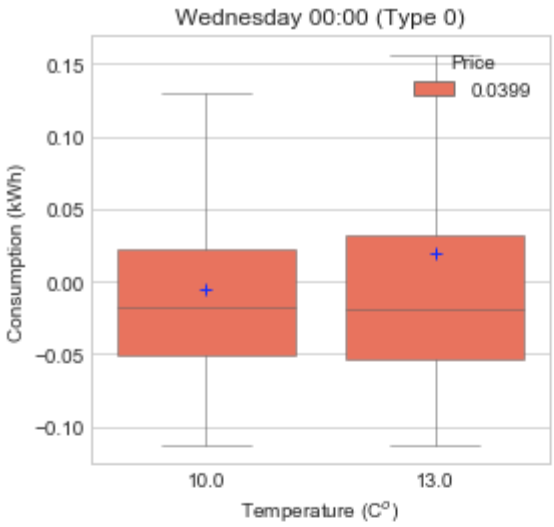

Consumption
0.2
0.0
−0.2
−0.4
5.0
10.0
15.0
22.0
Temperature (C^o)

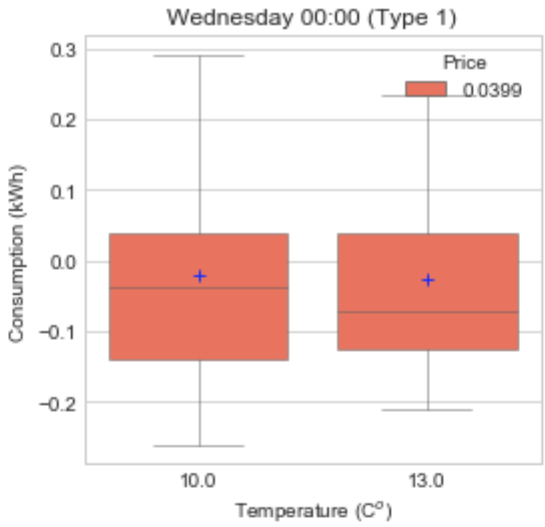

0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

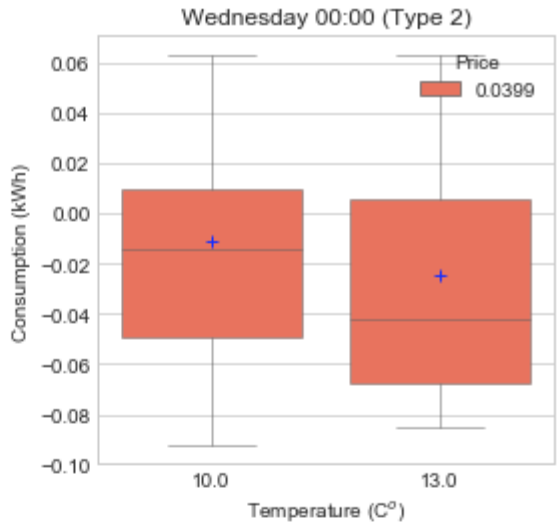

Wednesday 23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
4.0
10.0
13.0
21.0
Temperature (C^o)

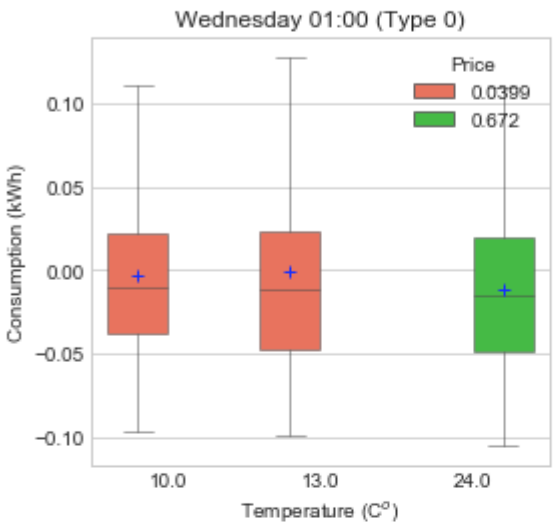

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

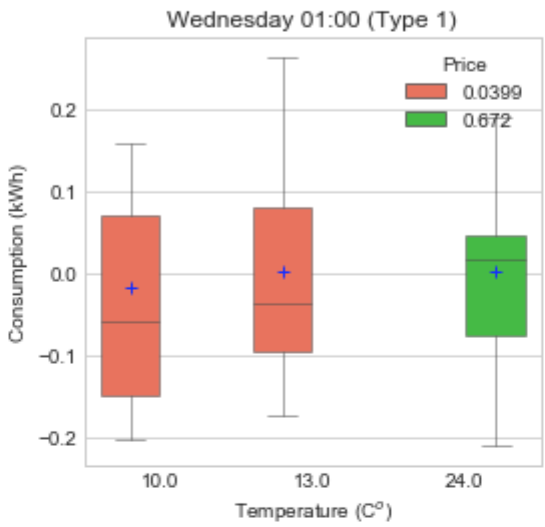
In [66]:
```
# price comparison
# Wednesday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 2 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

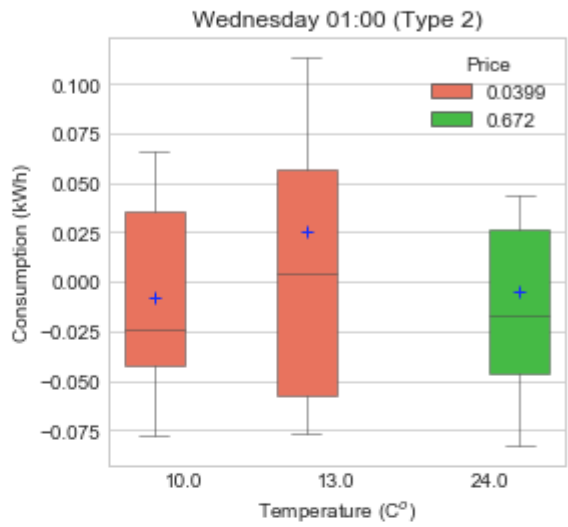

Wednesday 00:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
10.0
13.0
Temperature (C°)

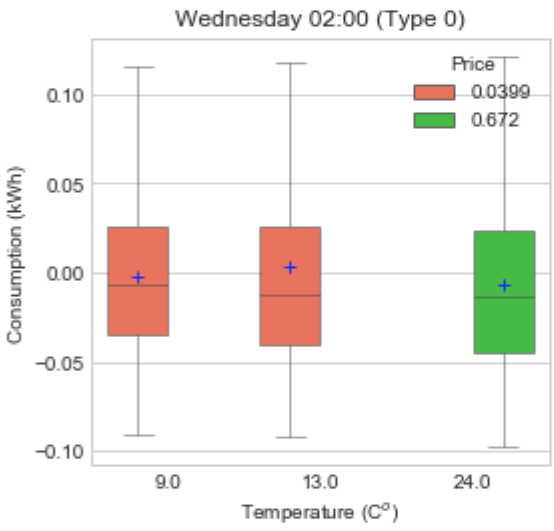

Wednesday 00:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
10.0
13.0
Temperature (C°)

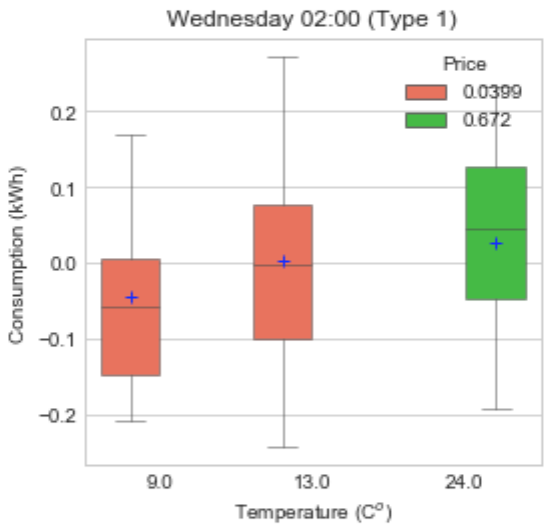

Wednesday 00:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
10.0
13.0
Temperature (C°)

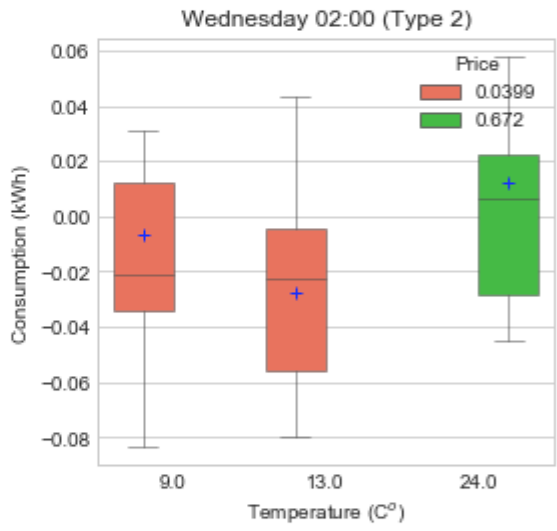

Wednesday 01:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
10.0
13.0
24.0
Temperature (C°)

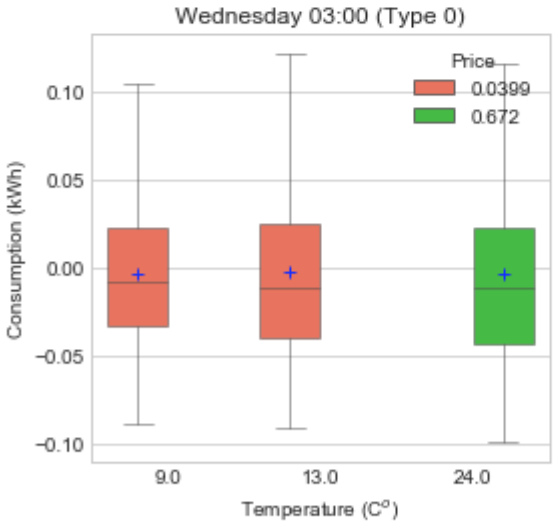

Wednesday 01:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
10.0
13.0
24.0
Temperature (C°)

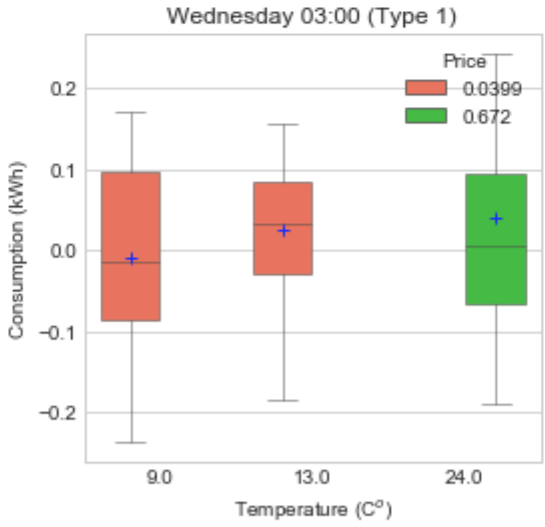

Wednesday 01:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
10.0
13.0
24.0
Temperature (C°)

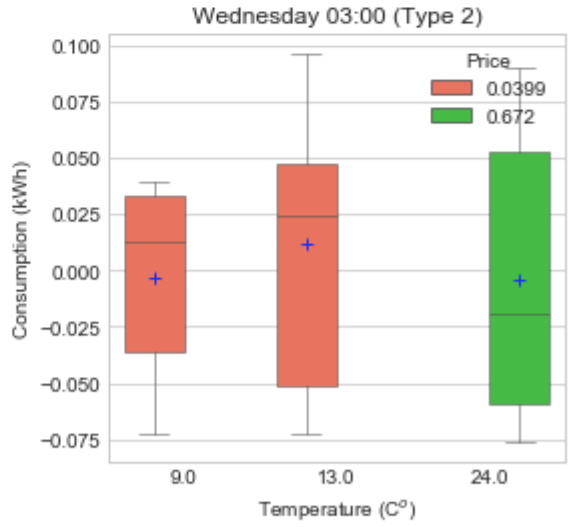

Wednesday 02:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
9.0
13.0
24.0
Temperature (C°)

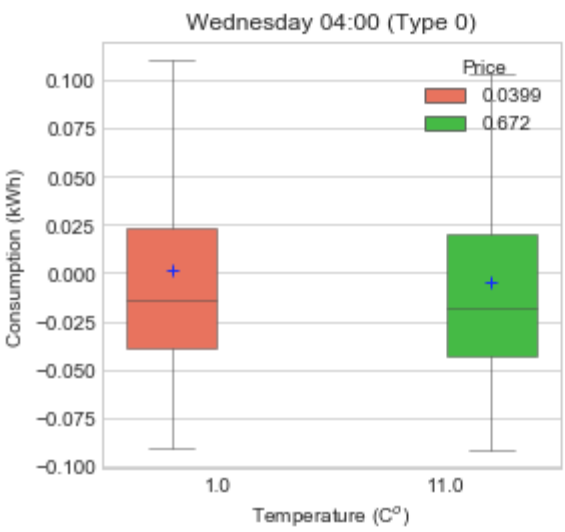

Wednesday 02:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
9.0
13.0
24.0
Temperature (C°)

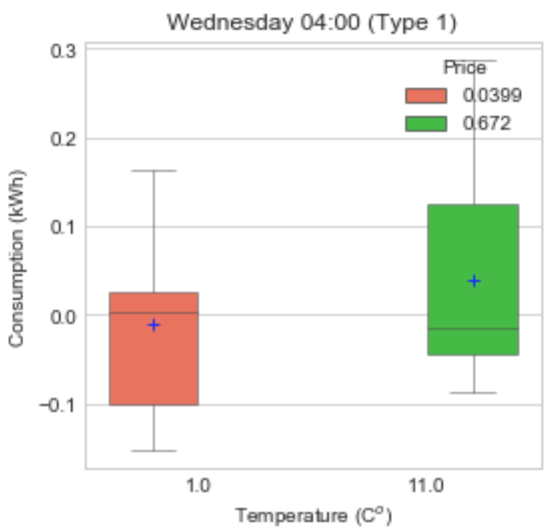

Wednesday 02:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
9.0
13.0
24.0
Temperature (C°)

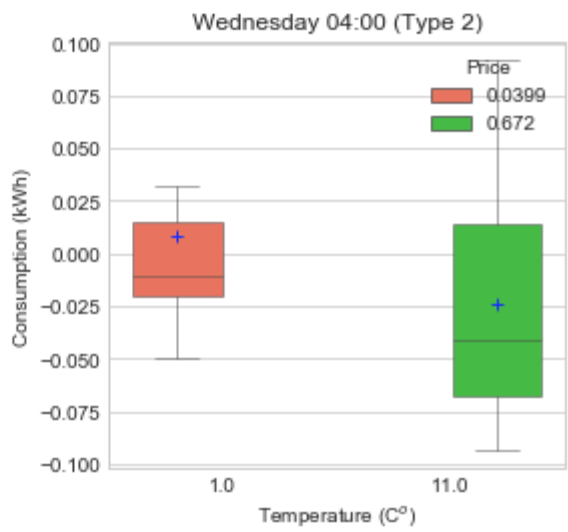

Wednesday 03:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
9.0
13.0
24.0
Temperature (C°)

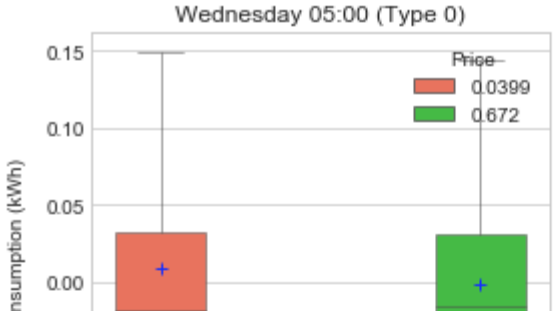

Wednesday 03:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
9.0
13.0
24.0
Temperature (C°)

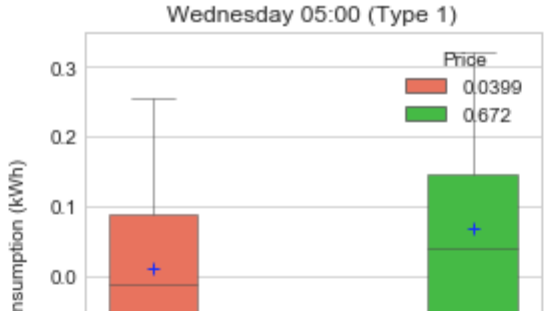

Wednesday 03:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
9.0
13.0
24.0
Temperature (C°)

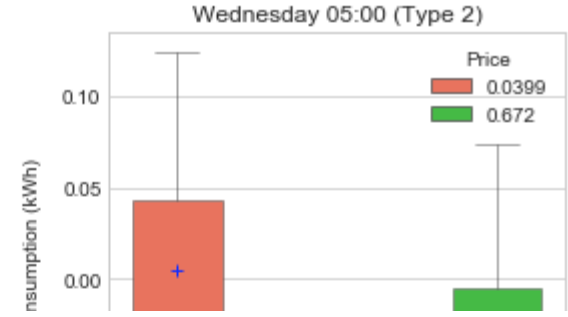

Wednesday 04:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
1.0
11.0
Temperature (C°)

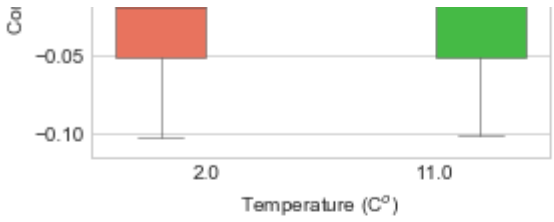

Wednesday 04:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
1.0
11.0
Temperature (C°)

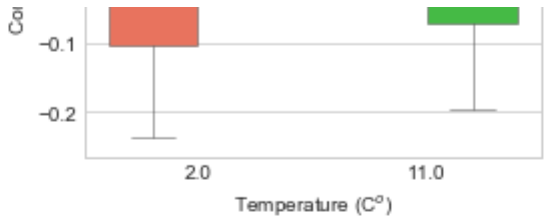

Wednesday 04:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
1.0
11.0
Temperature (C°)

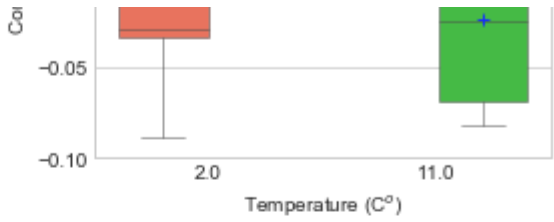

Wednesday 05:00 (Type 0)
Price
0.0399
0.672
nsumption (kWh)
0.15
0.10
0.05
0.00

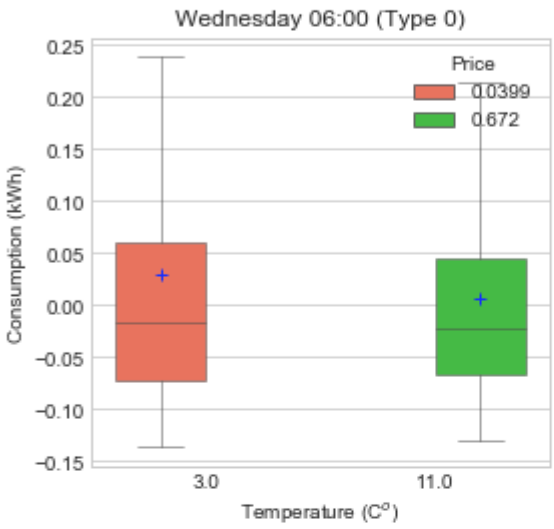

Wednesday 05:00 (Type 1)
Price
0.0399
0.672
nsumption (kWh)
0.3
0.2
0.1
0.0

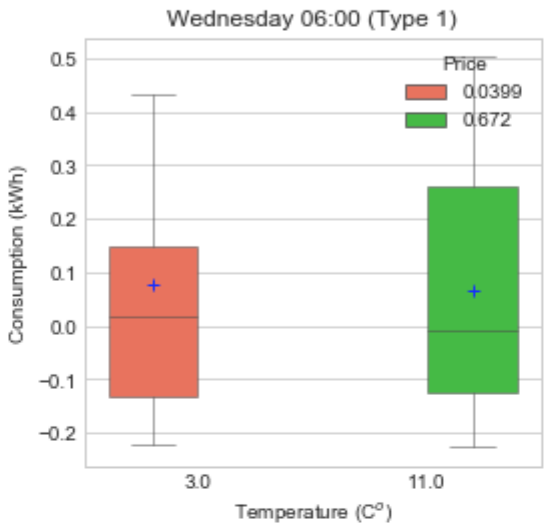

Wednesday 05:00 (Type 2)
Price
0.0399
0.672
nsumption (kWh)
0.10
0.05
0.00

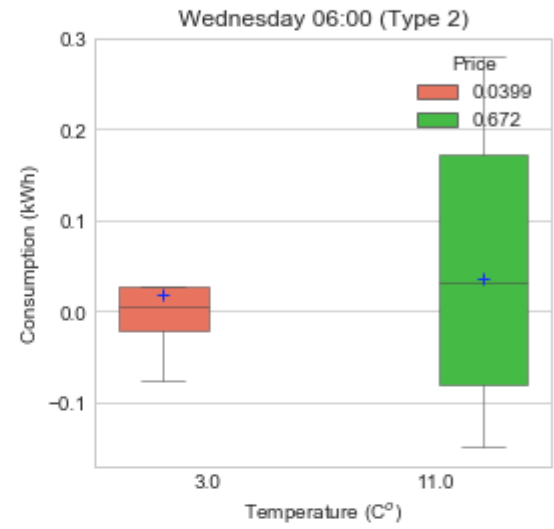

Temperature (C°)

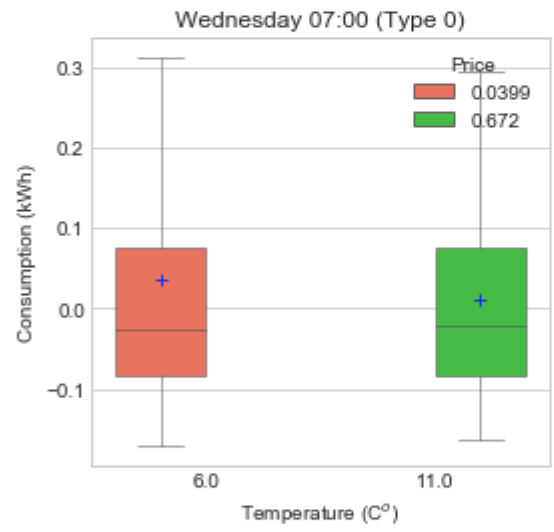

2.0
11.0
Temperature (C°)

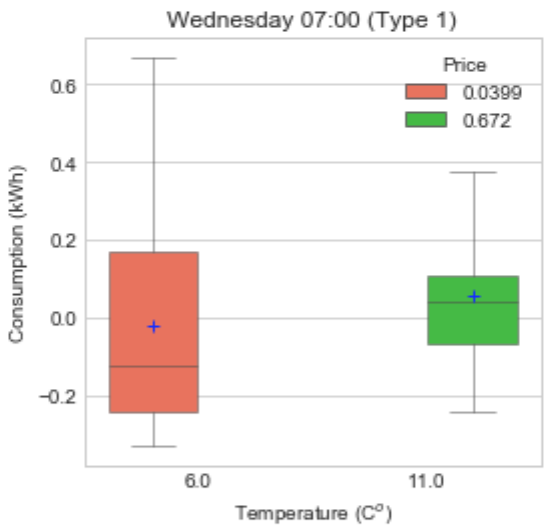

2.0
11.0
Temperature (C°)

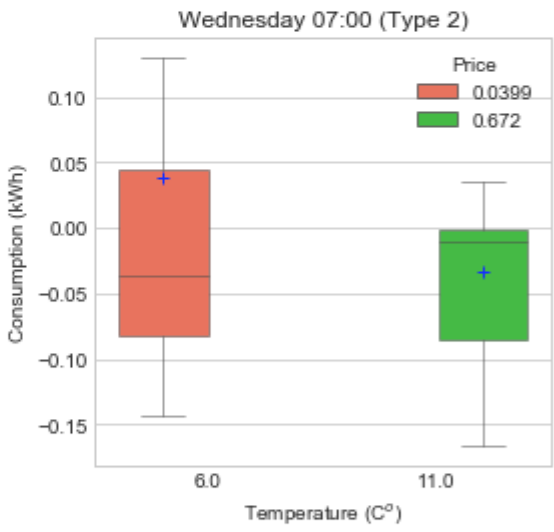

Wednesday 06:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
3.0
11.0
Temperature (C°)

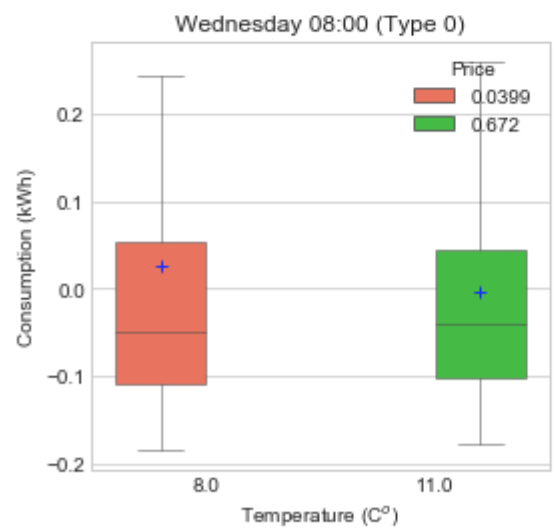

Wednesday 06:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
3.0
11.0
Temperature (C°)

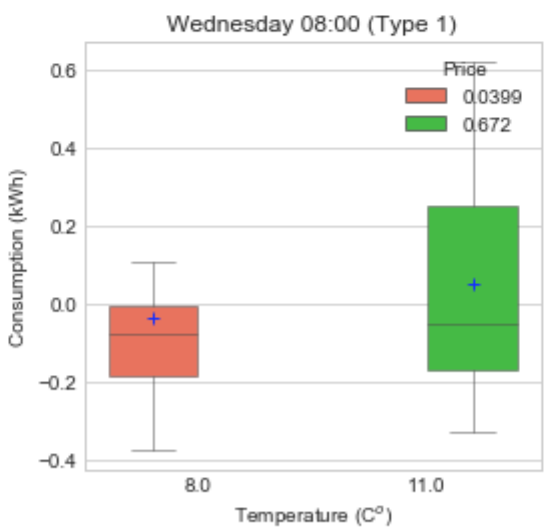

Wednesday 06:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
3.0
11.0
Temperature (C°)

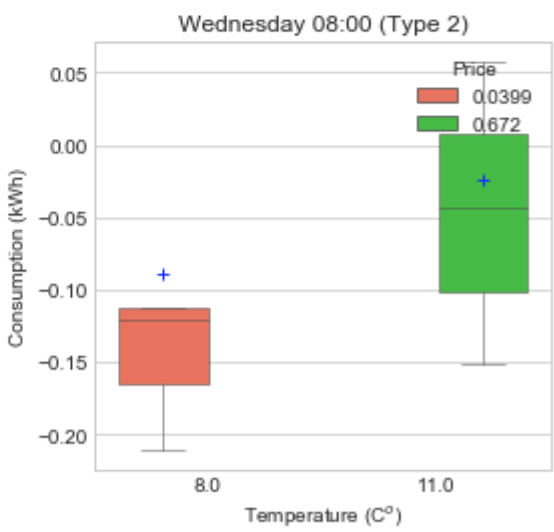

Wednesday 07:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
6.0
11.0
Temperature (C°)

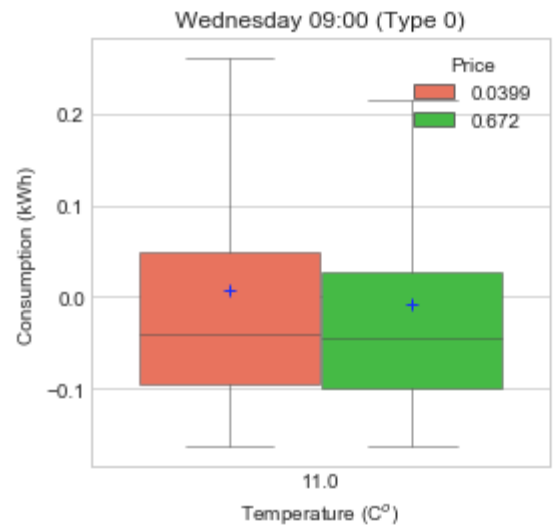

Wednesday 07:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
6.0
11.0
Temperature (C°)

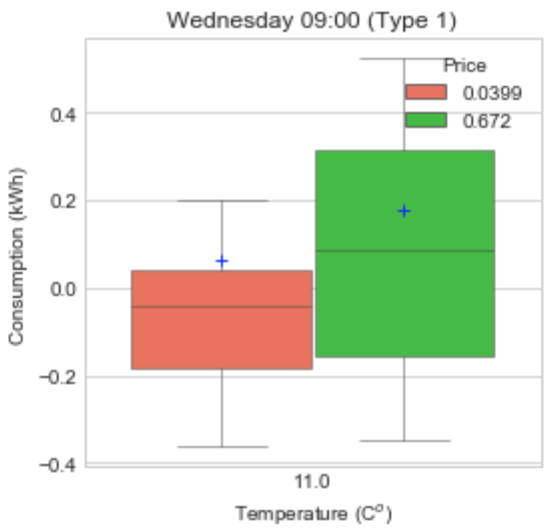

Wednesday 07:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
6.0
11.0
Temperature (C°)

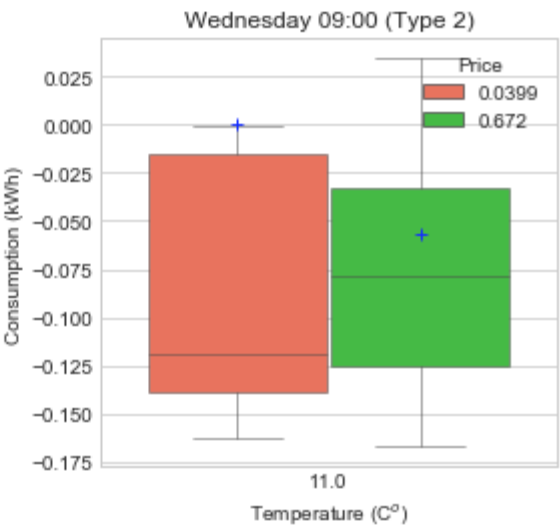

Wednesday 08:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
8.0
11.0
Temperature (C°)

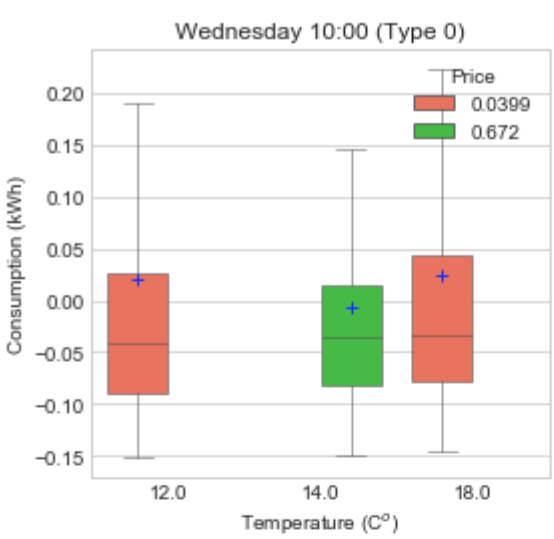

Wednesday 08:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
8.0
11.0
Temperature (C°)

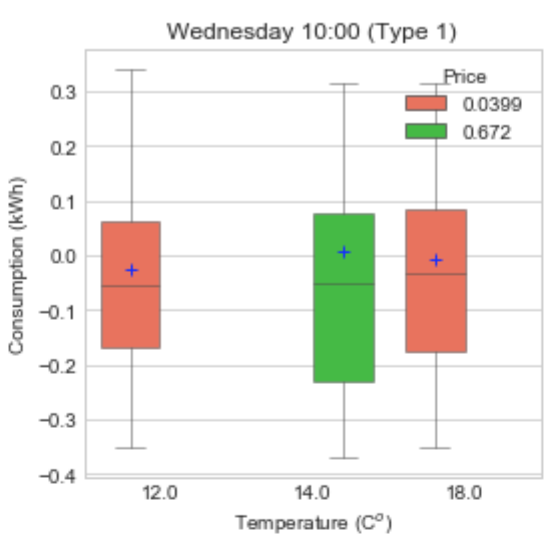

Wednesday 08:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
8.0
11.0
Temperature (C°)

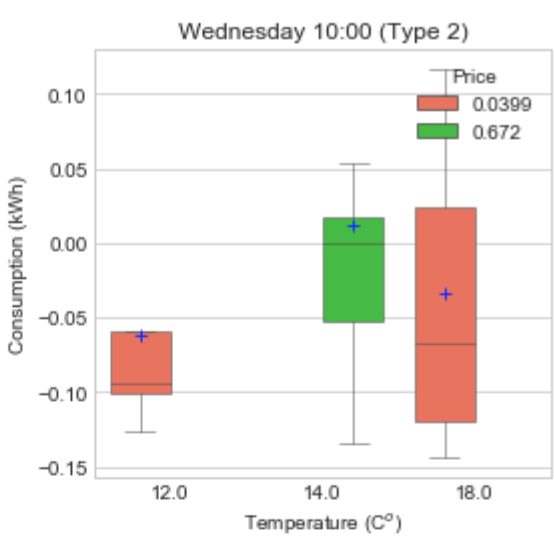

Wednesday 09:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
11.0
Temperature (C°)

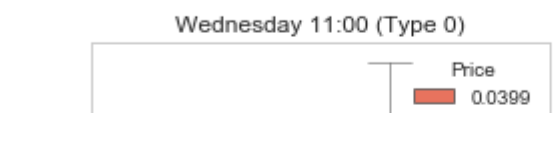

Wednesday 09:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
11.0
Temperature (C°)

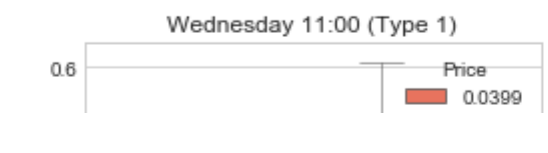

Wednesday 09:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
11.0
Temperature (C°)


Wednesday 10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
18.0
Temperature (C°)

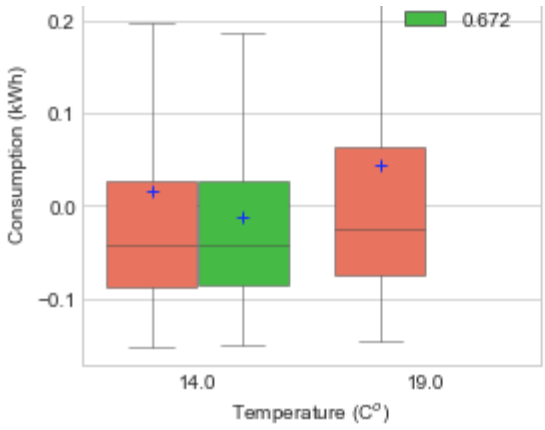

Wednesday 10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
18.0
Temperature (C°)

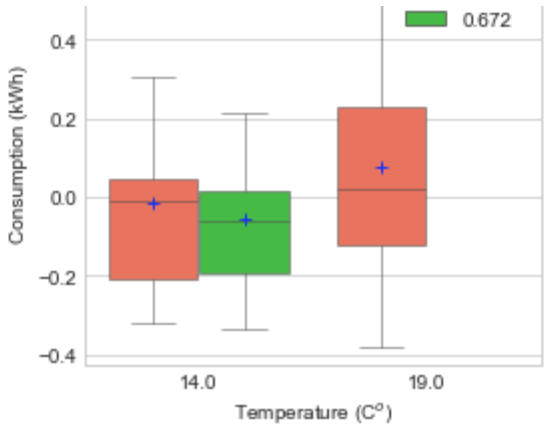

Wednesday 10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
18.0
Temperature (C°)

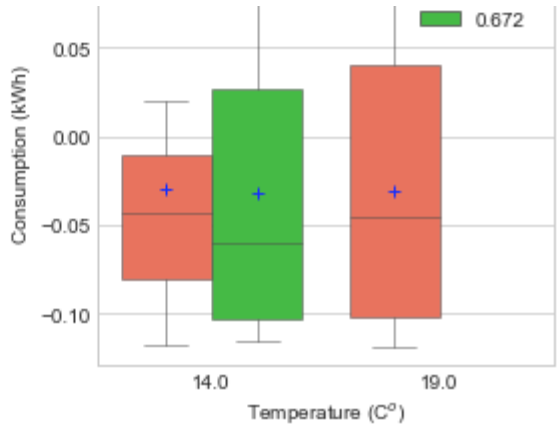

Wednesday 11:00 (Type 0)
Price
0.0399

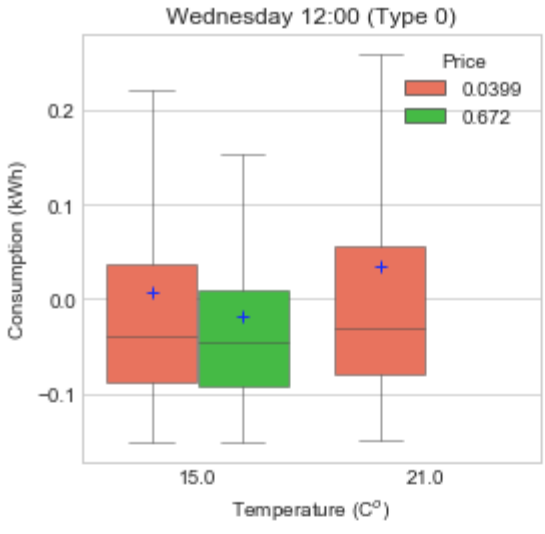

Wednesday 11:00 (Type 1)
Price
0.0399

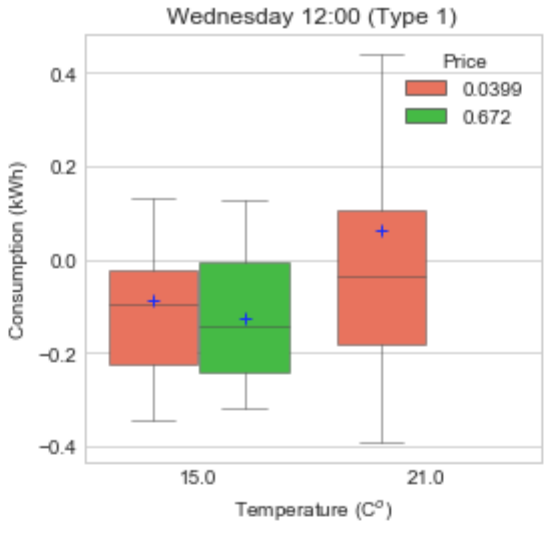

Wednesday 11:00 (Type 2)
Price
0.0399

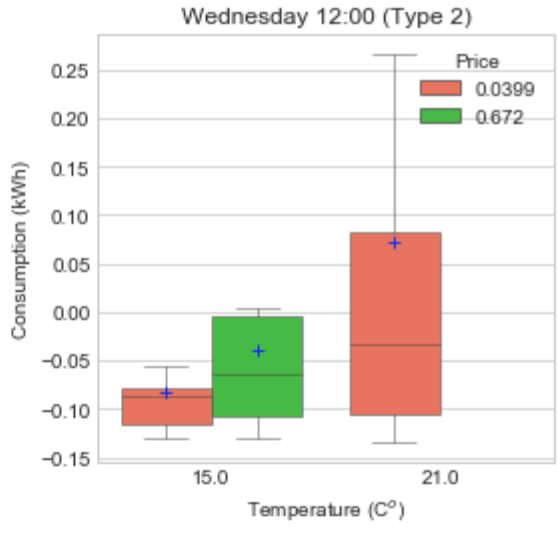

0.672
Consumption (kWh)
Temperature (C°)
14.0
19.0

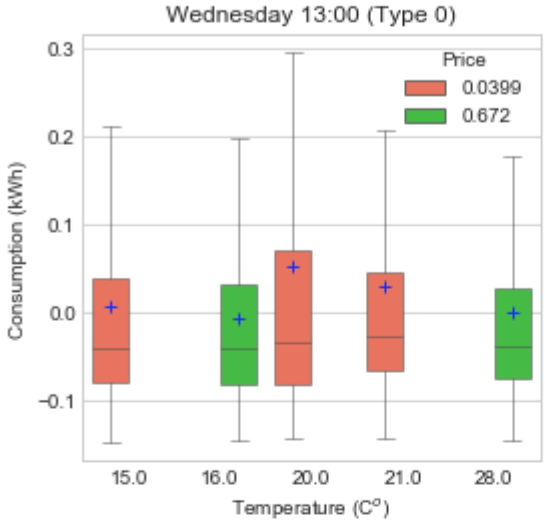

0.672
Consumption (kWh)
Temperature (C°)
14.0
19.0

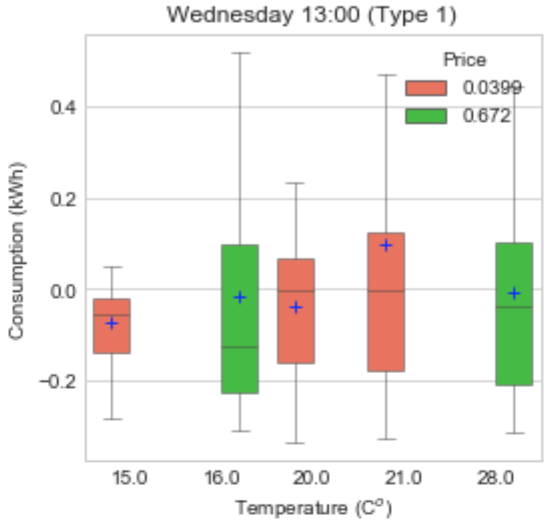

0.672
Consumption (kWh)
Temperature (C°)
14.0
19.0

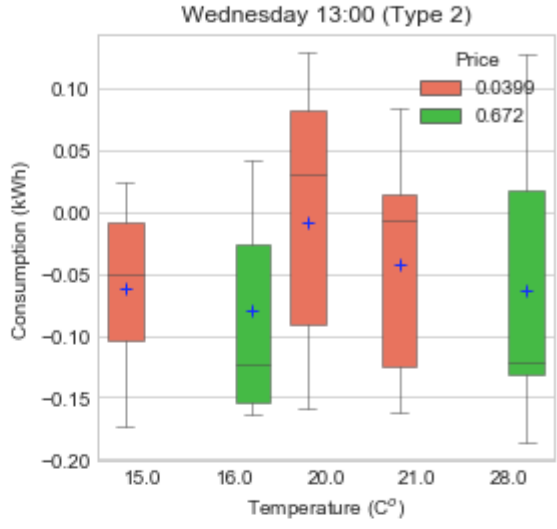

Wednesday 12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
21.0

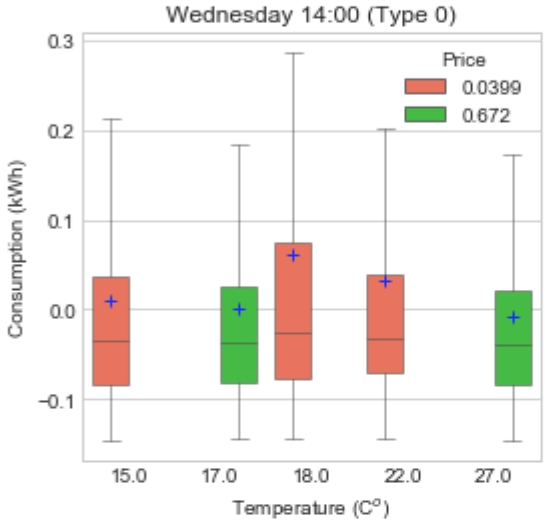

Wednesday 12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
21.0

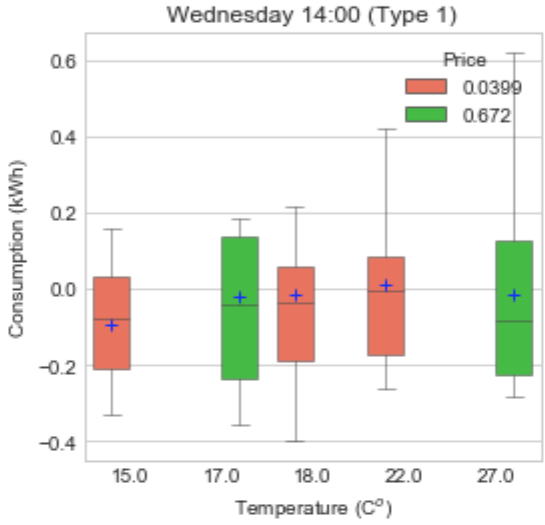

Wednesday 12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
21.0

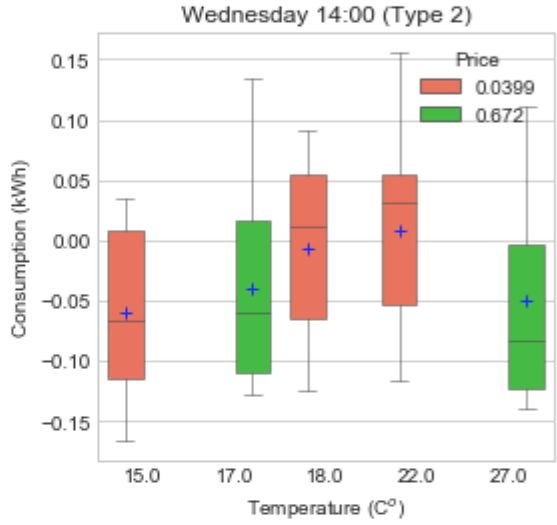

Wednesday 13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
16.0
20.0
21.0
28.0

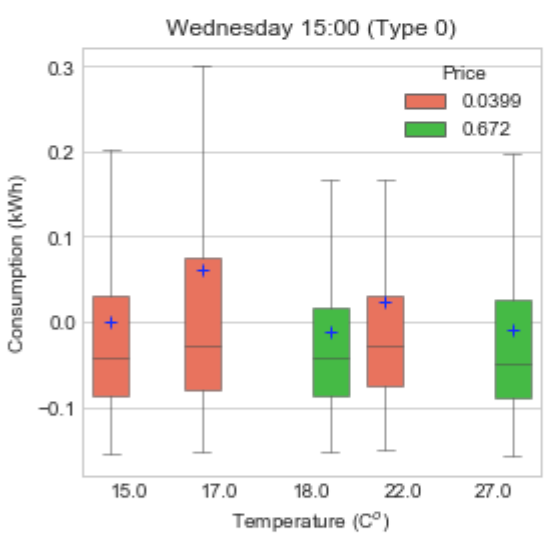

Wednesday 13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
16.0
20.0
21.0
28.0

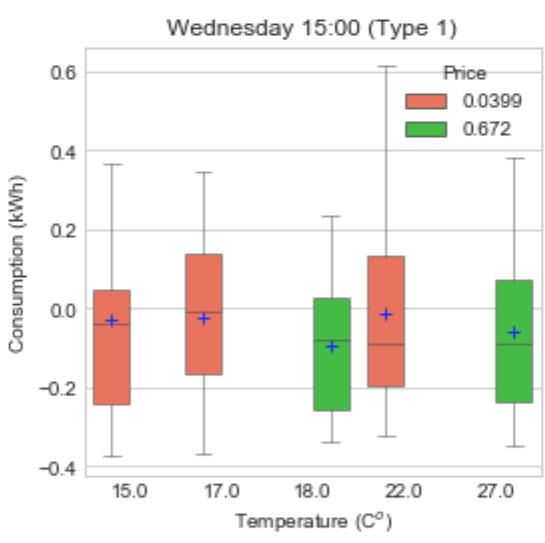

Wednesday 13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
16.0
20.0
21.0
28.0

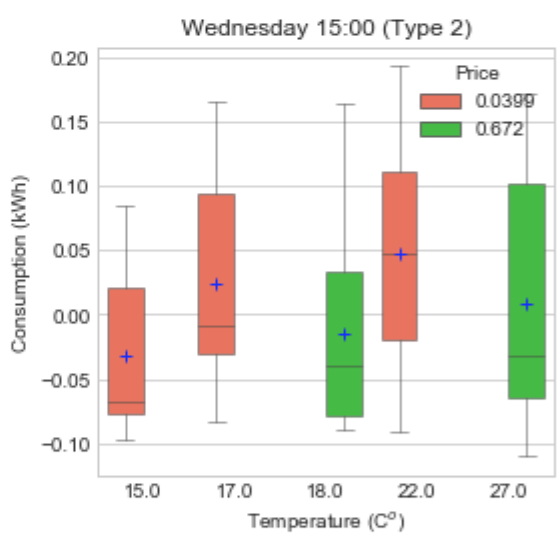

Wednesday 14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

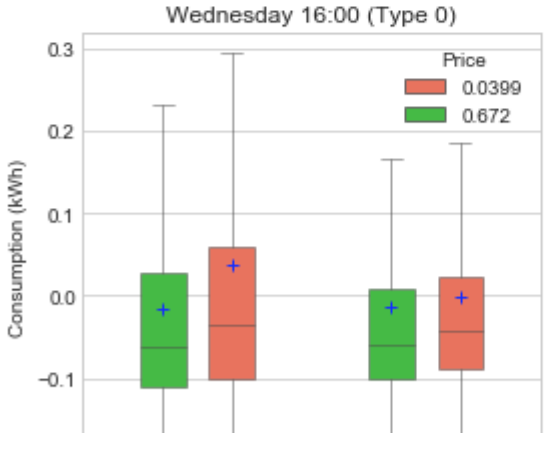

Wednesday 14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

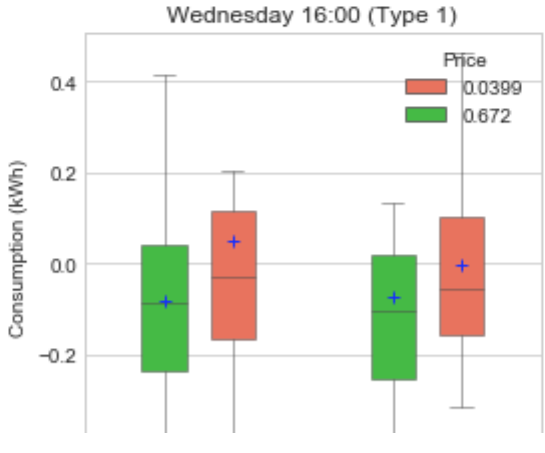

Wednesday 14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

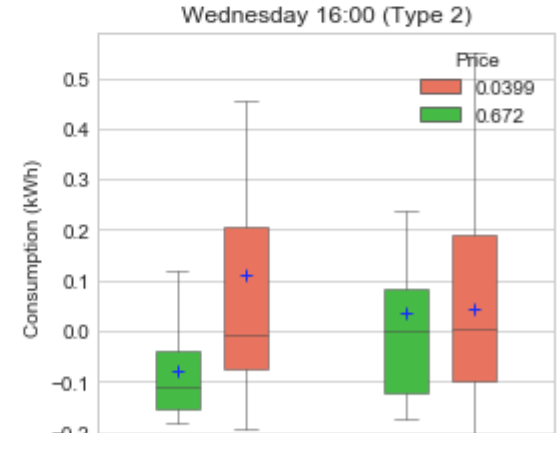

Wednesday 15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

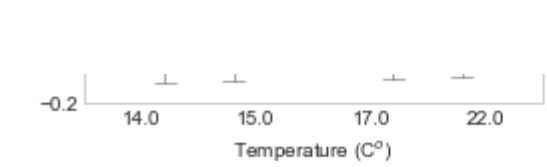

Wednesday 15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

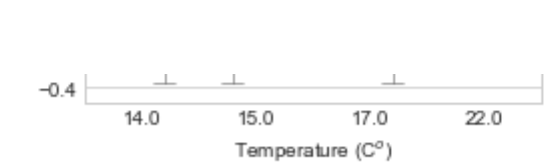

Wednesday 15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
15.0
17.0
18.0
22.0
27.0

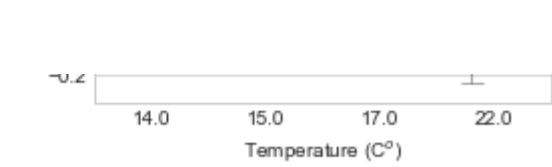

Wednesday 16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

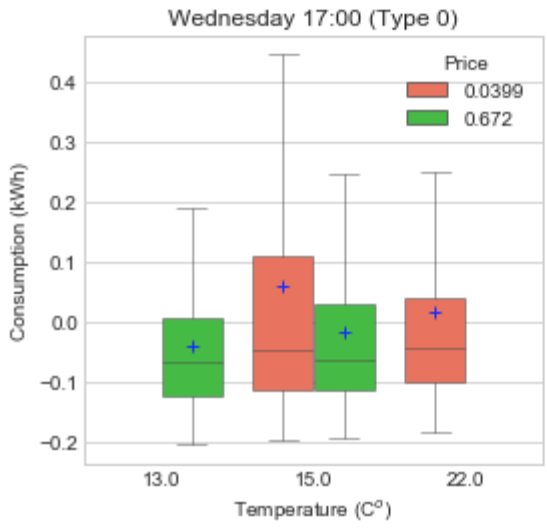

Wednesday 16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

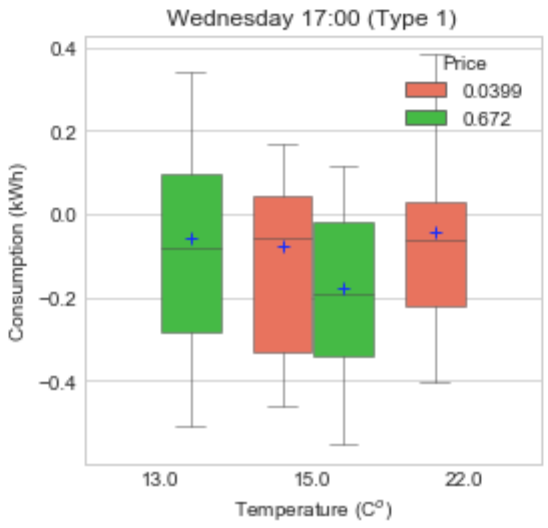

Wednesday 16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

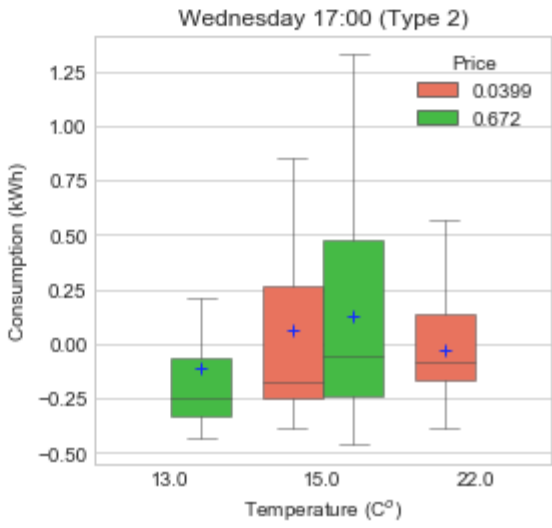

14.0
15.0
17.0
22.0
Temperature (C°)

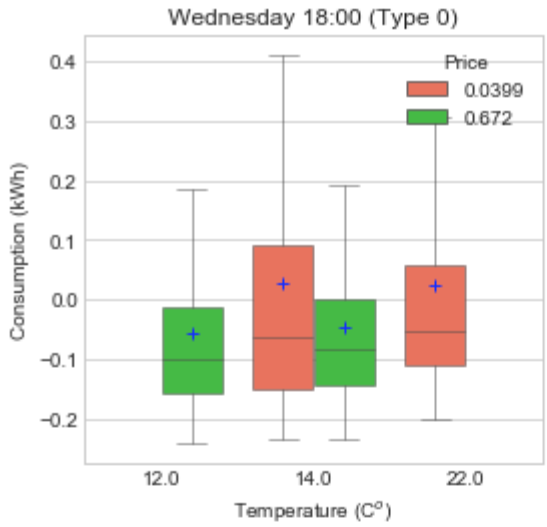

14.0
15.0
17.0
22.0
Temperature (C°)

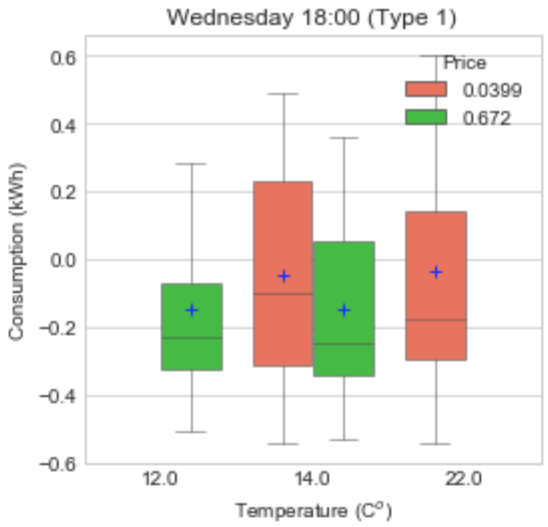

14.0
15.0
17.0
22.0
Temperature (C°)

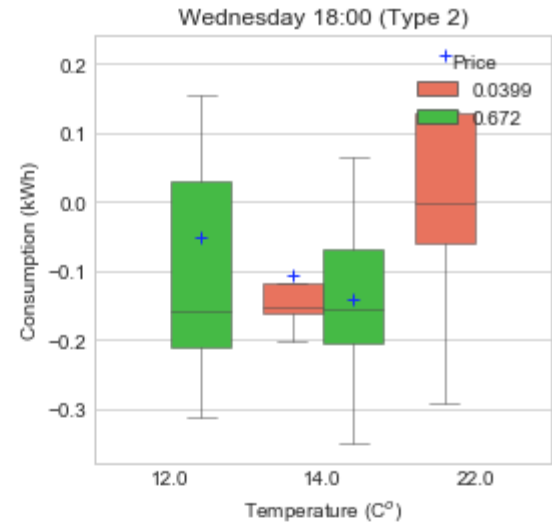

Wednesday 17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
13.0
15.0
22.0
Temperature (C°)

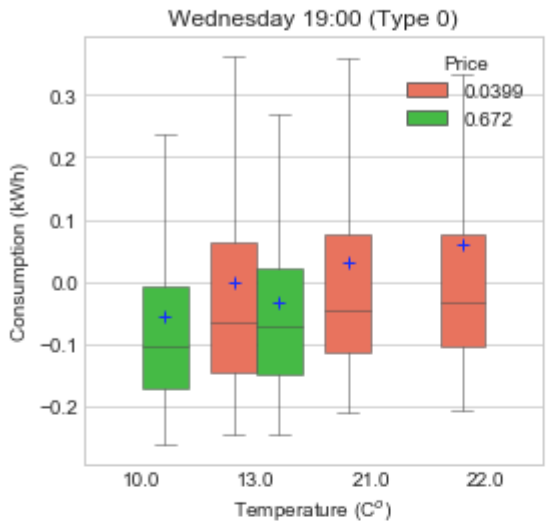

Wednesday 17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
13.0
15.0
22.0
Temperature (C°)

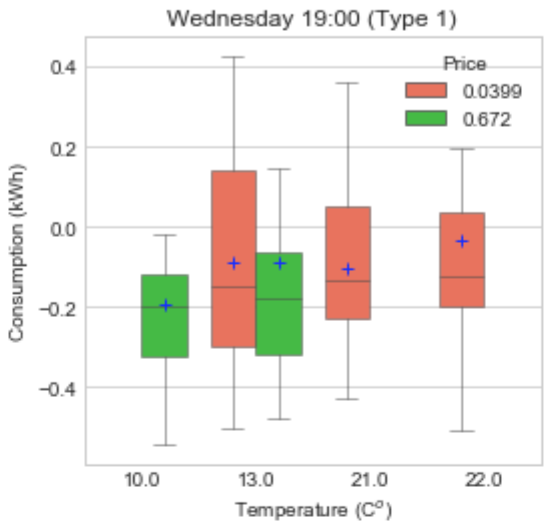

Wednesday 17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
13.0
15.0
22.0
Temperature (C°)

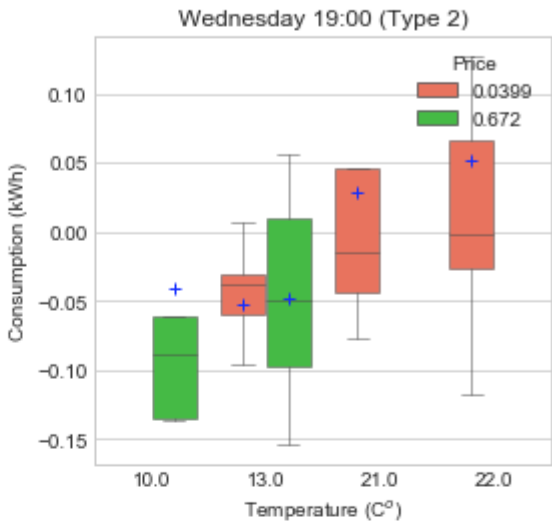

Wednesday 18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
22.0
Temperature (C°)

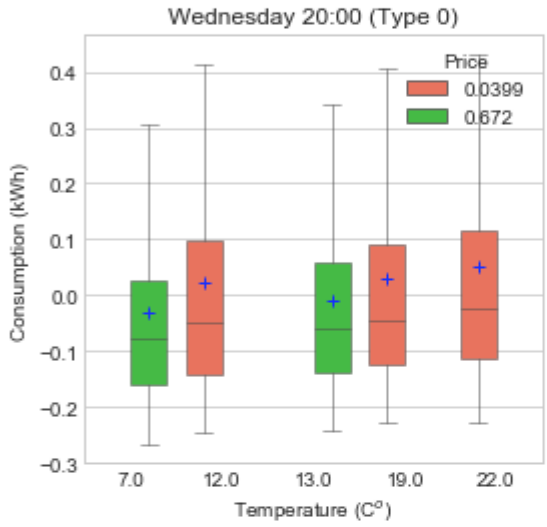

Wednesday 18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
22.0
Temperature (C°)

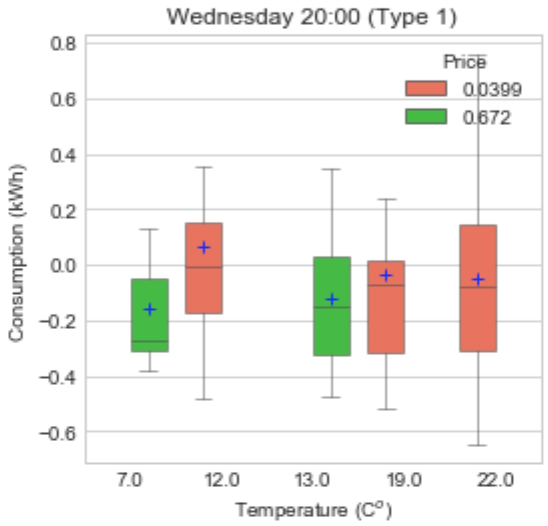

Wednesday 18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
12.0
14.0
22.0
Temperature (C°)

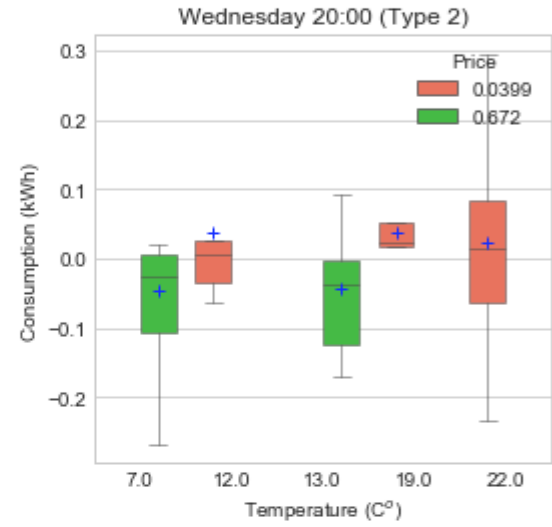

Wednesday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
13.0
21.0
22.0
Temperature (C°)

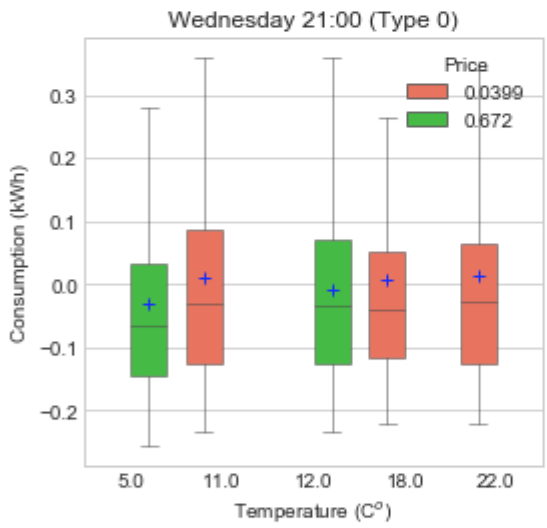

Wednesday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
13.0
21.0
22.0
Temperature (C°)

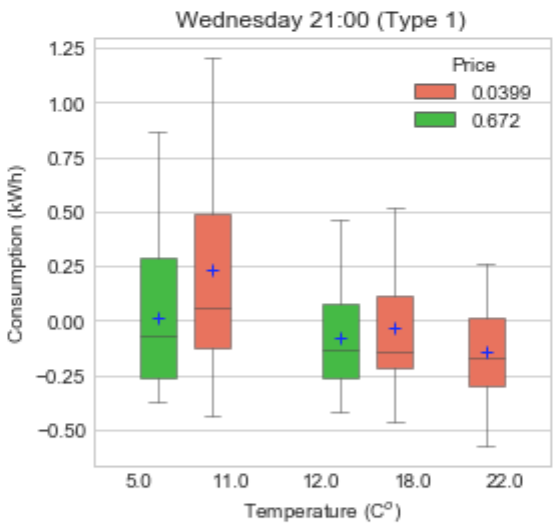

Wednesday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
13.0
21.0
22.0
Temperature (C°)

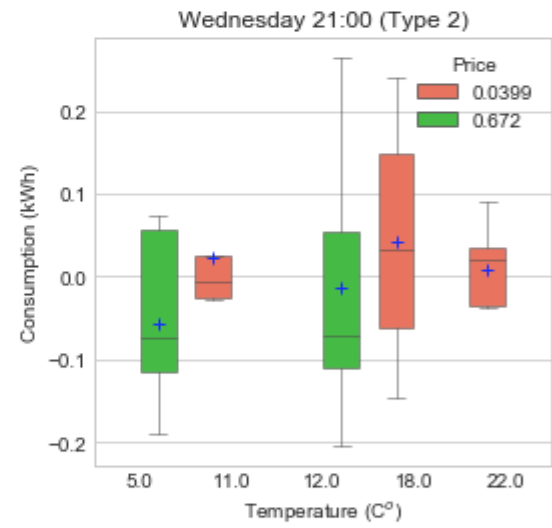

Wednesday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
7.0
12.0
13.0
19.0
22.0
Temperature (C°)

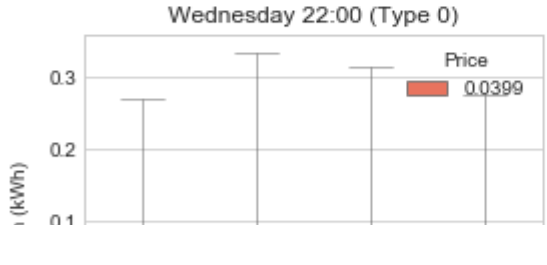

Wednesday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
7.0
12.0
13.0
19.0
22.0
Temperature (C°)

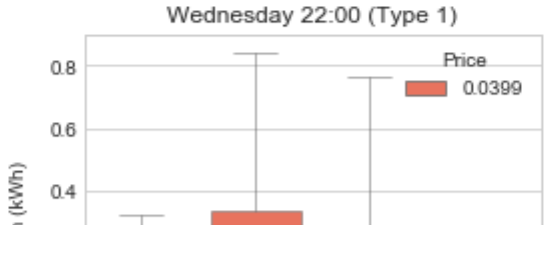

Wednesday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
7.0
12.0
13.0
19.0
22.0
Temperature (C°)

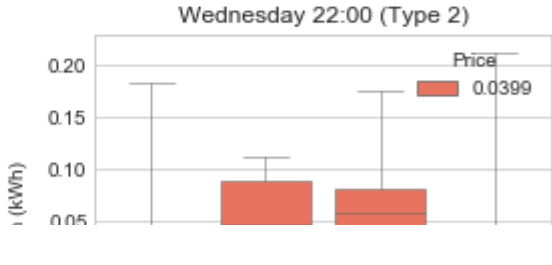

Wednesday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
5.0
11.0
12.0
18.0
22.0
Temperature (C°)

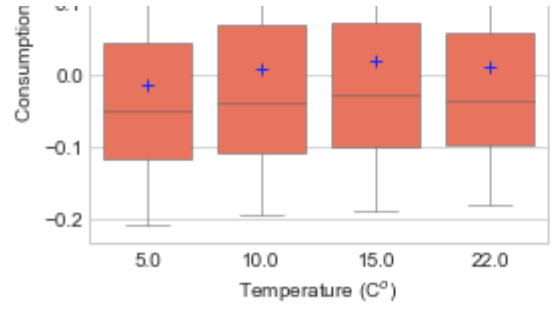

Wednesday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
5.0
11.0
12.0
18.0
22.0
Temperature (C°)

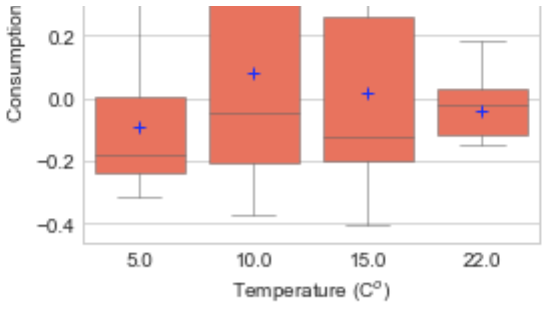

Wednesday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
5.0
11.0
12.0
18.0
22.0
Temperature (C°)

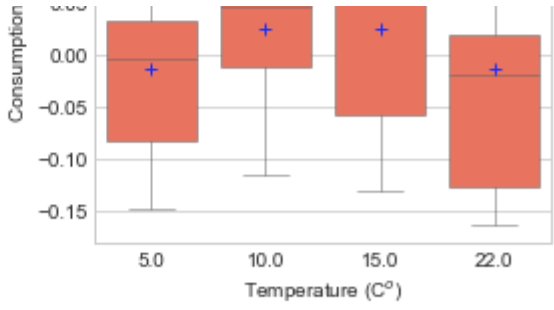

Wednesday 22:00 (Type 0)
Price
0.0399
Temperature (C°)

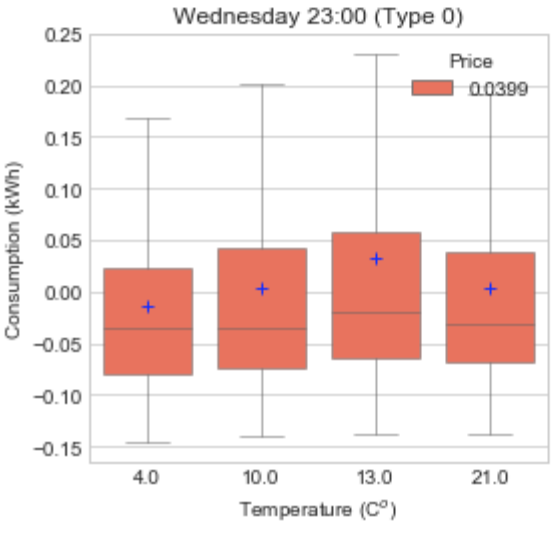

Wednesday 22:00 (Type 1)
Price
0.0399
Temperature (C°)

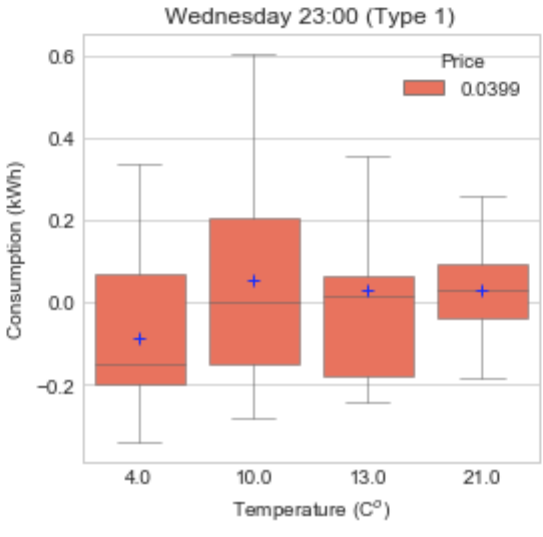

Wednesday 22:00 (Type 2)
Price
0.0399
Temperature (C°)

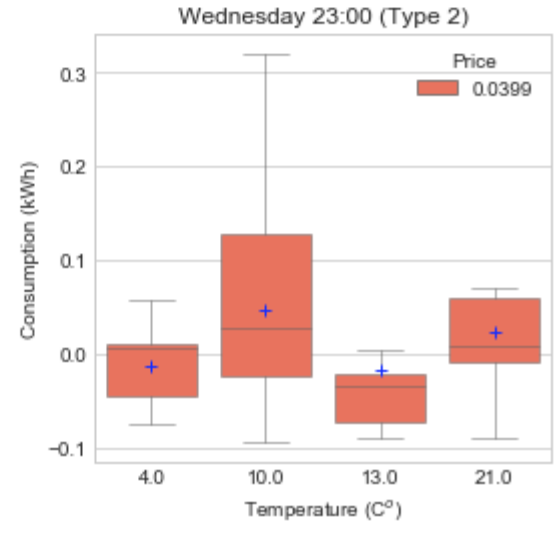

Consumption
0.0
−0.1
−0.2
5.0
10.0
15.0
22.0
Temperature (C°)


Consumption
0.2
0.0
−0.2
−0.4
5.0
10.0
15.0
22.0
Temperature (C°)

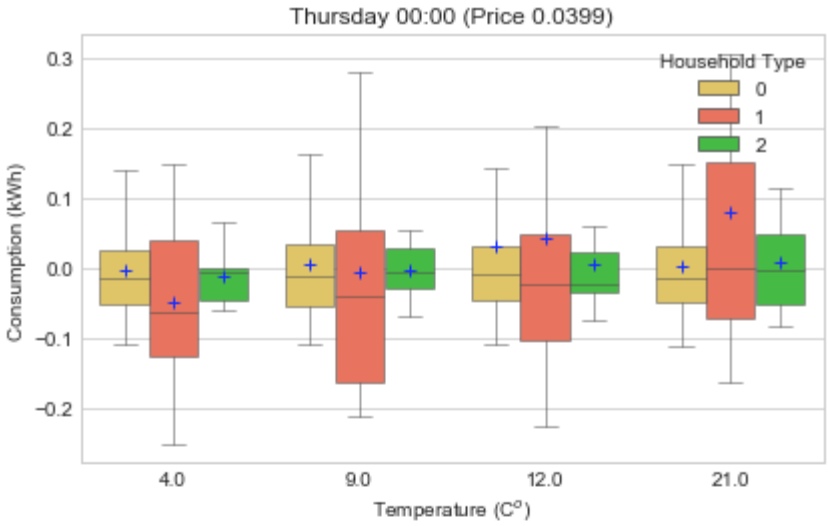

Consumption
0.00
−0.05
−0.10
−0.15
5.0
10.0
15.0
22.0
Temperature (C°)

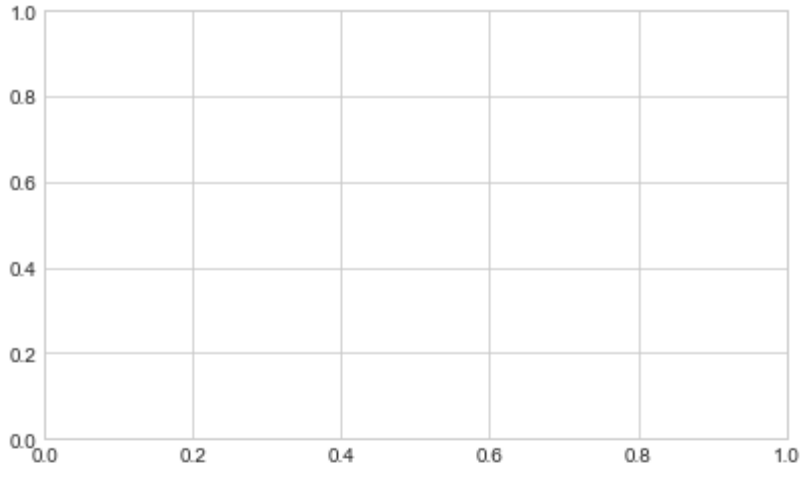

Wednesday 23:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
4.0
10.0
13.0
21.0
Temperature (C°)

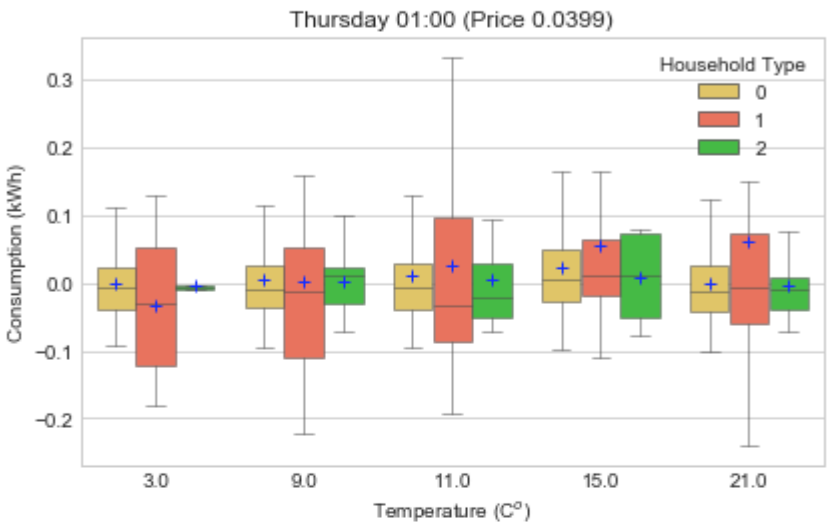

Wednesday 23:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
4.0
10.0
13.0
21.0
Temperature (C°)

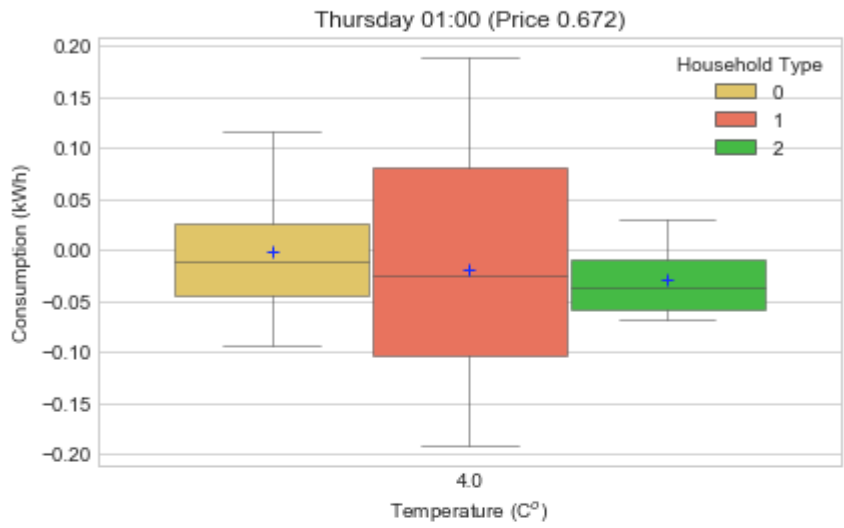

Wednesday 23:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
4.0
10.0
13.0
21.0
Temperature (C°)

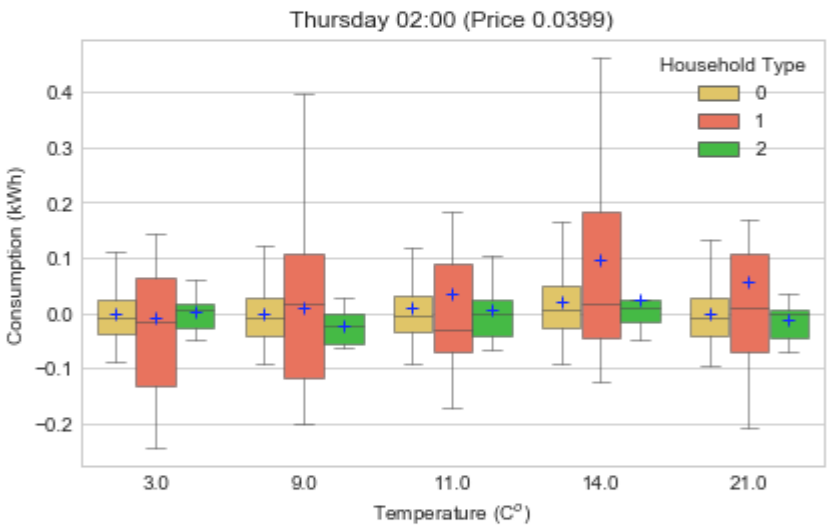
In [67]:
```
# Thursday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 3 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

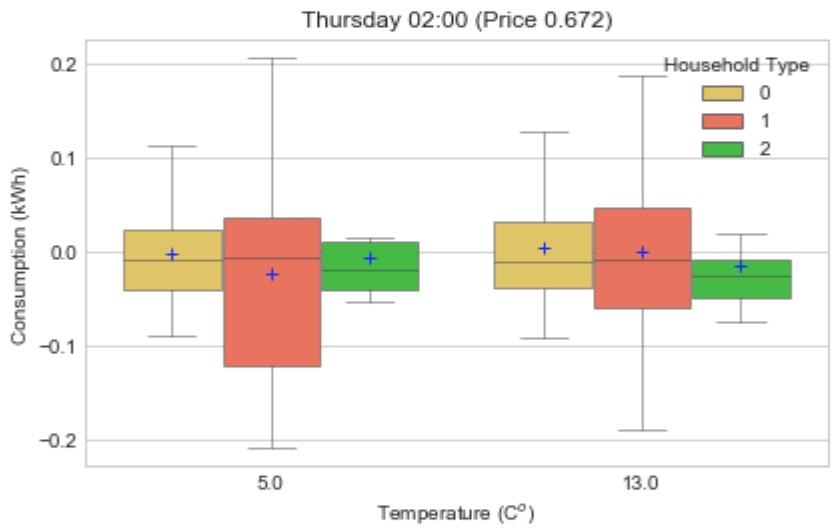

Thursday 00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
9.0
12.0
21.0
Temperature (C°)

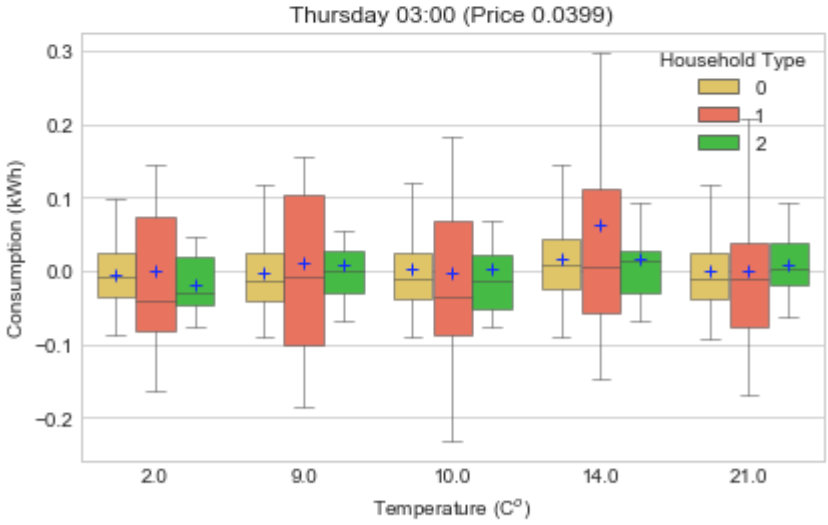

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

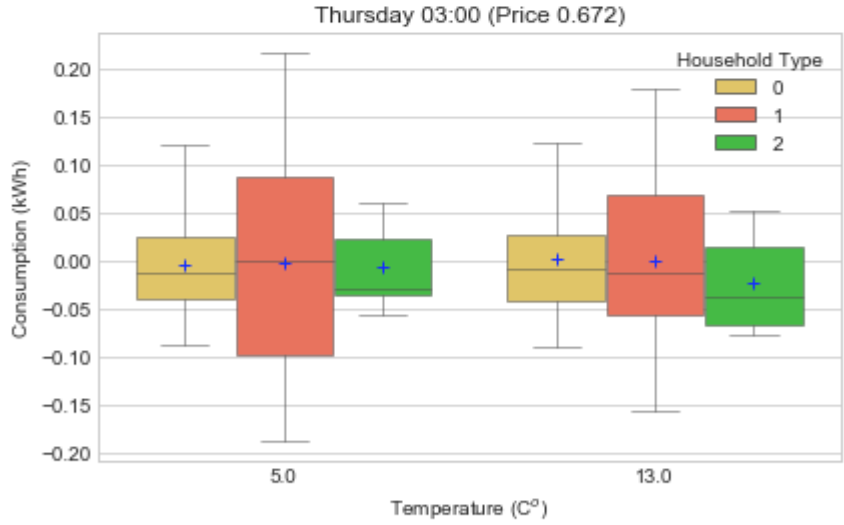

Thursday 01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
9.0
11.0
15.0
21.0
Temperature (C°)

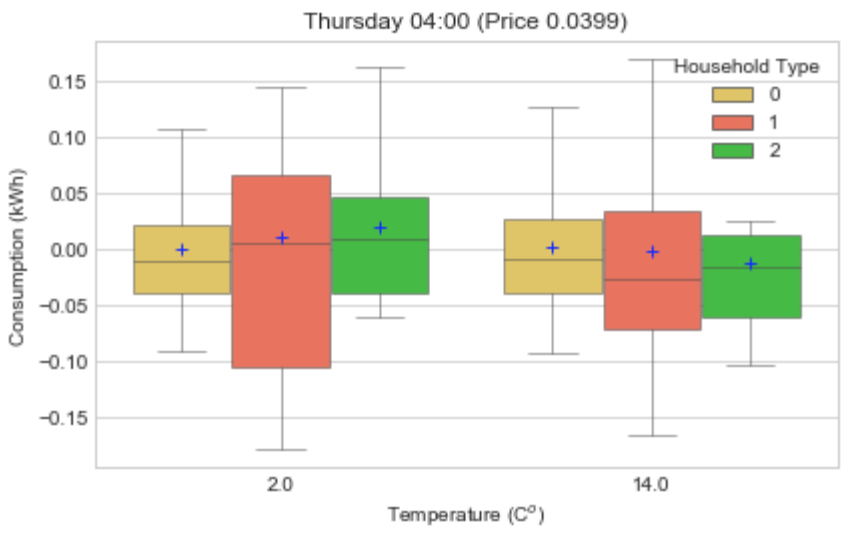

Thursday 01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
4.0
Temperature (C°)

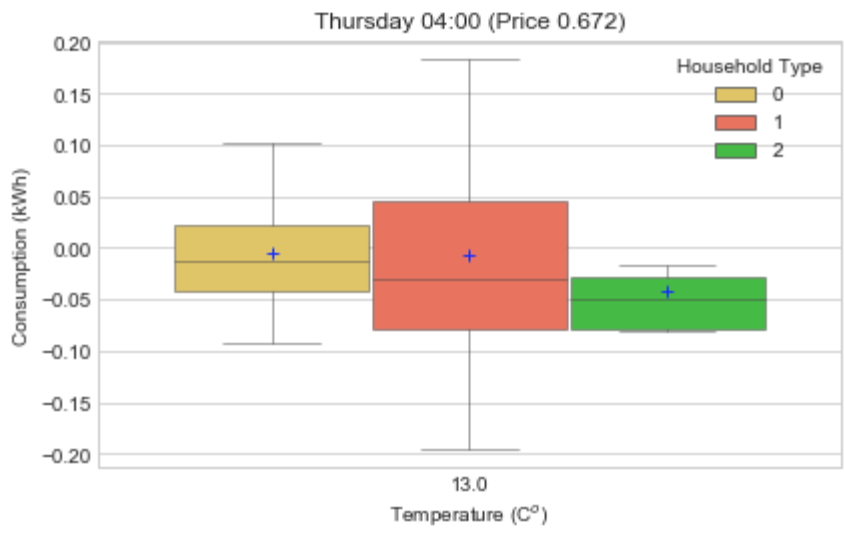

Thursday 02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
9.0
11.0
14.0
21.0
Temperature (C°)

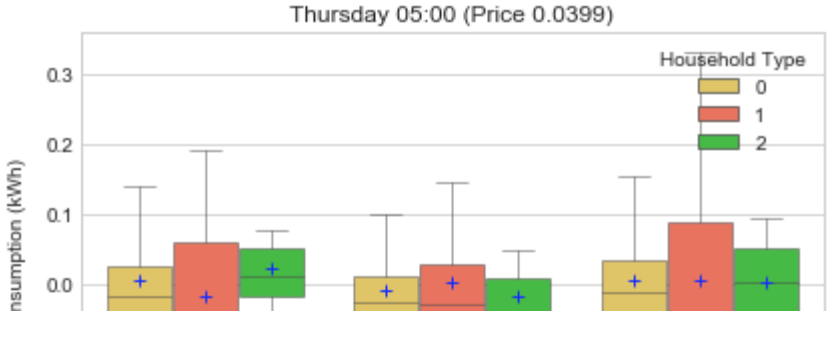

Thursday 02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
5.0
13.0
Temperature (C°)

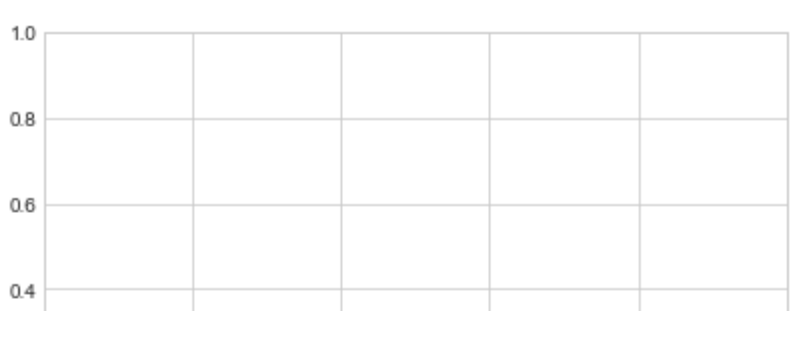

Thursday 03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
2.0
9.0
10.0
14.0
21.0
Temperature (C°)

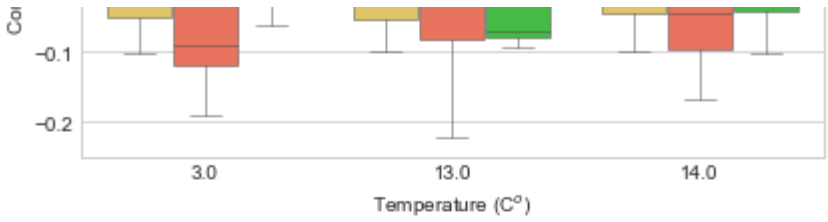

Thursday 03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
5.0
13.0
Temperature (C°)


Thursday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
2.0
14.0
Temperature (C°)

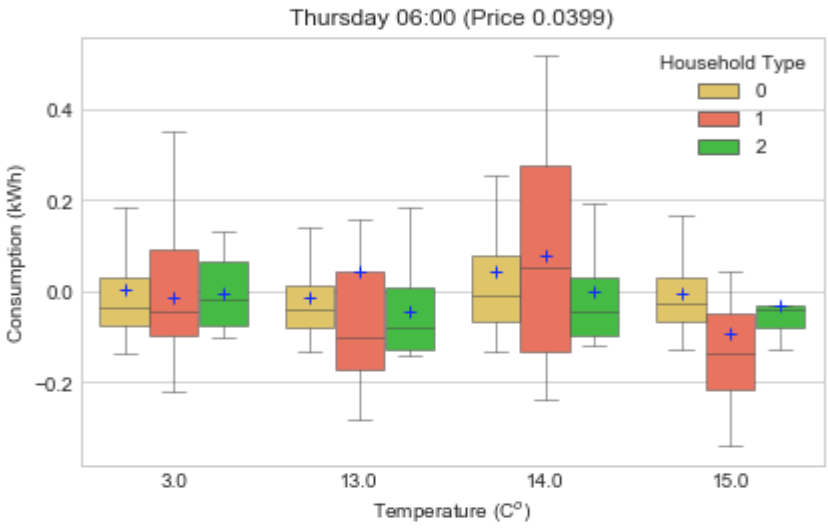

Thursday 04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
13.0
Temperature (C°)

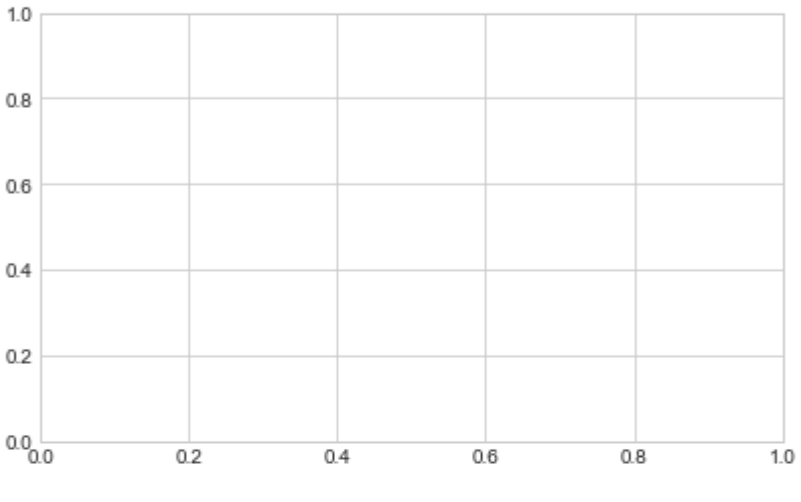

Thursday 05:00 (Price 0.0399)
Household Type
0
1
2
0.3
0.2
0.1
0.0

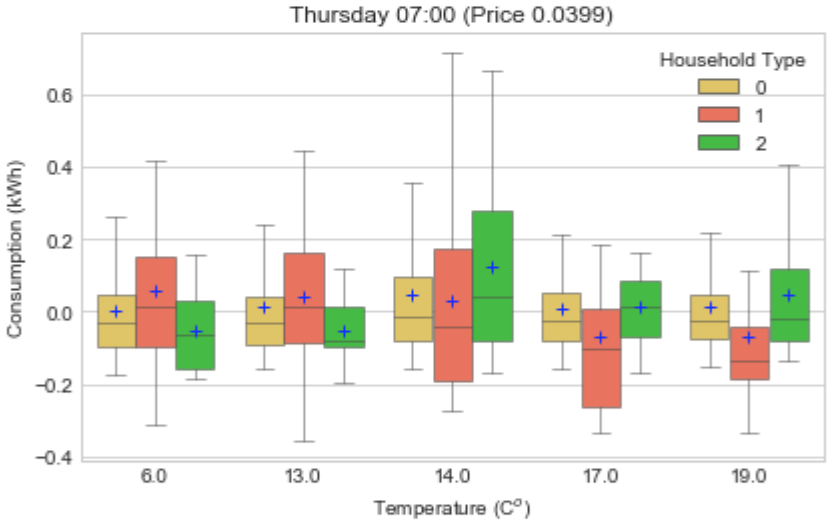

1.0
0.8
0.6
0.4

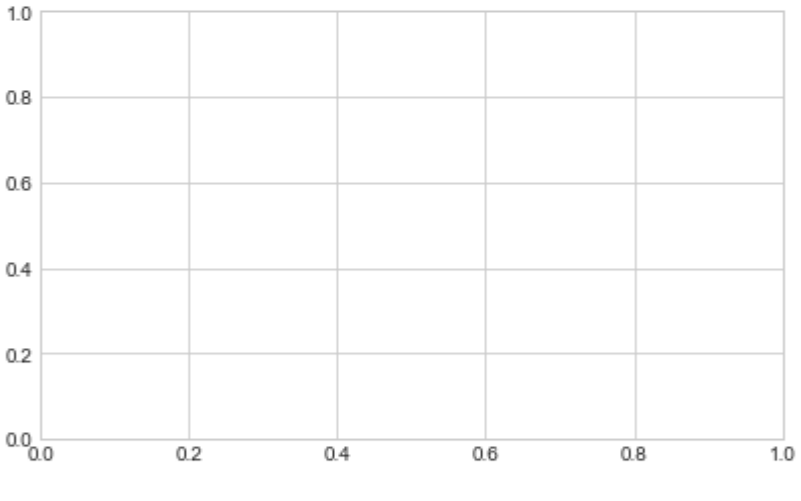

Co
−0.1
−0.2
3.0
13.0
14.0
Temperature (C°)

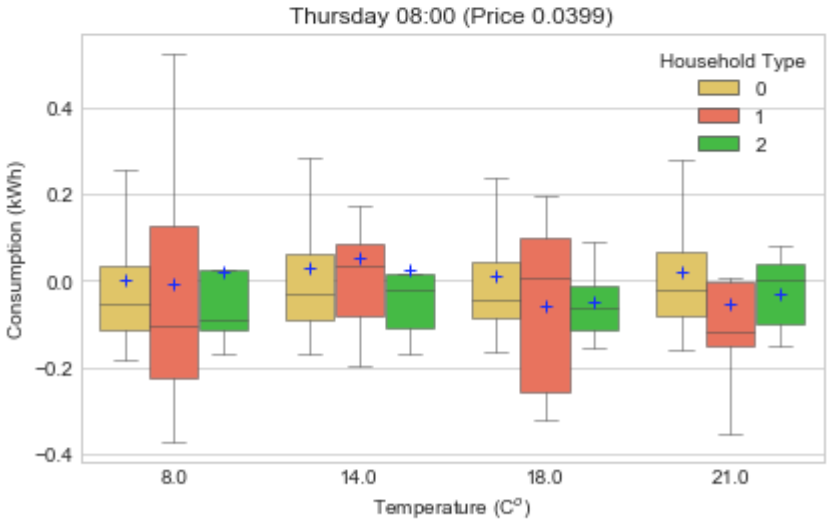

0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

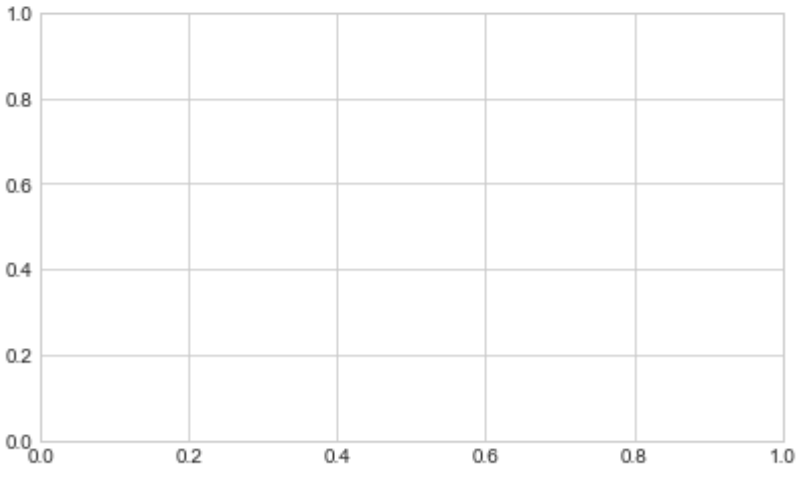

Thursday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
3.0
13.0
14.0
15.0
Temperature (C°)

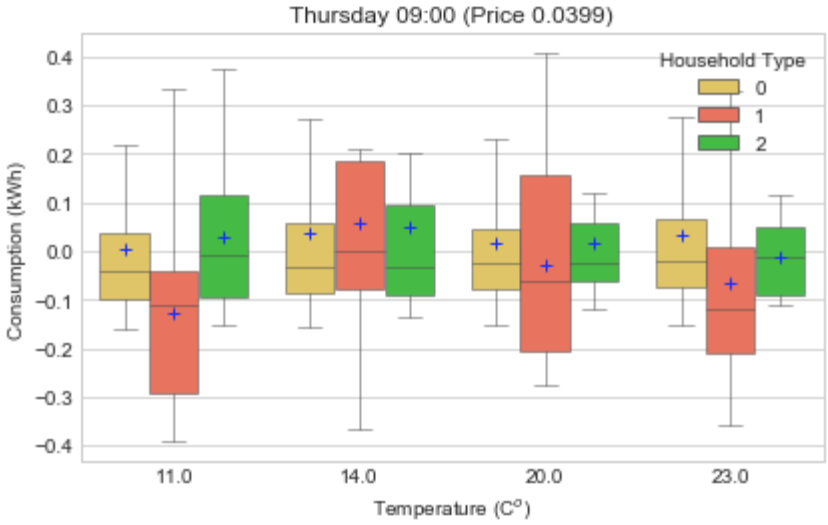

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

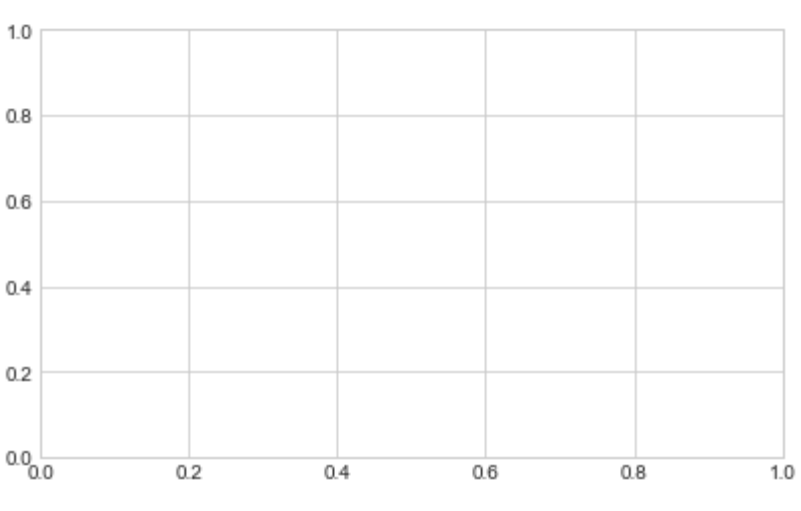

Thursday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
6.0
13.0
14.0
17.0
19.0
Temperature (C°)

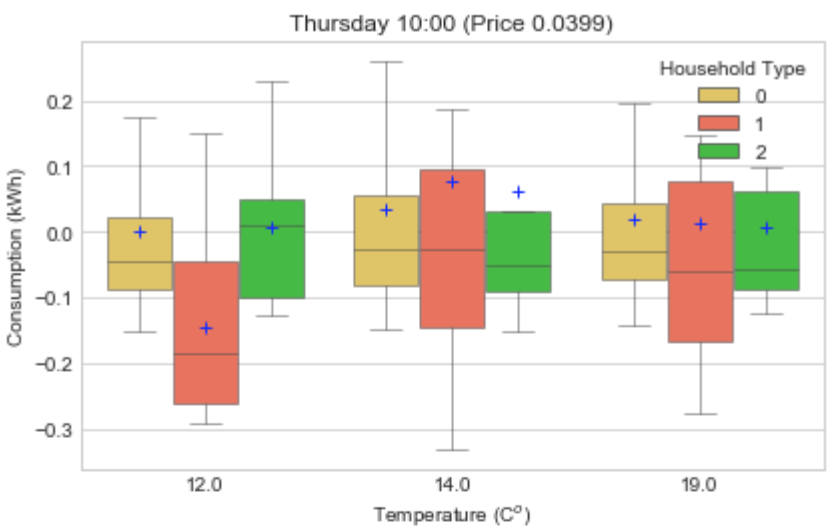

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

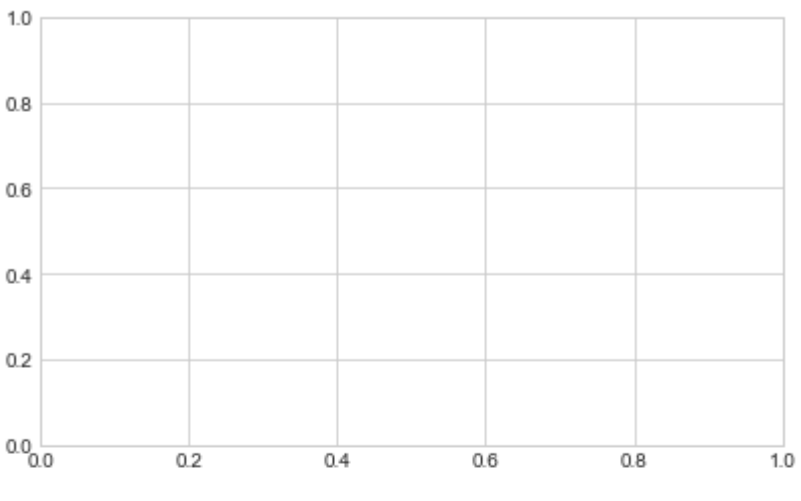

Thursday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
8.0
14.0
18.0
21.0
Temperature (C°)

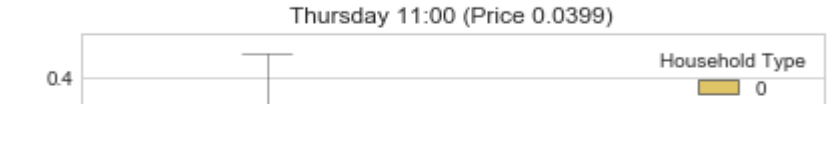

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Thursday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
11.0
14.0
20.0
23.0
Temperature (C°)

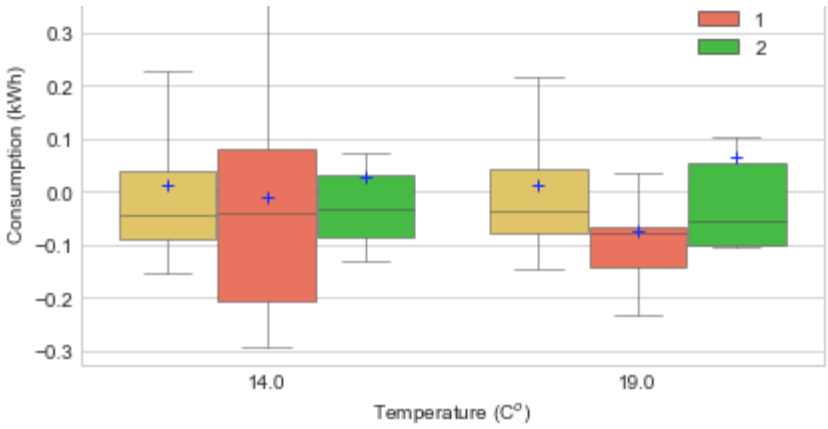

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

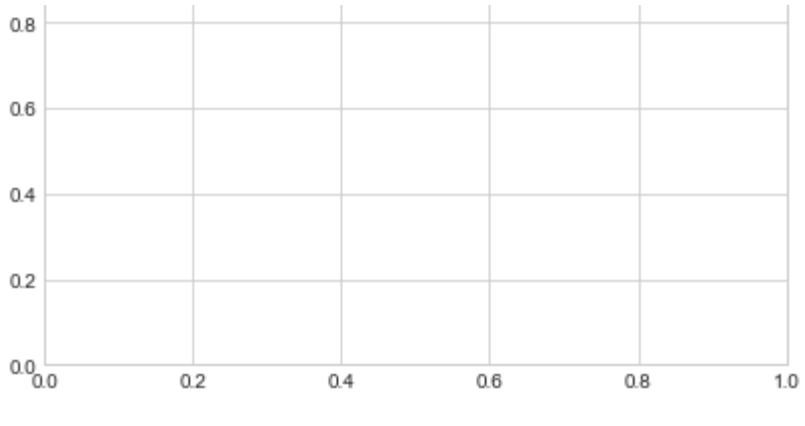

Thursday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
12.0
14.0
19.0
Temperature (C°)

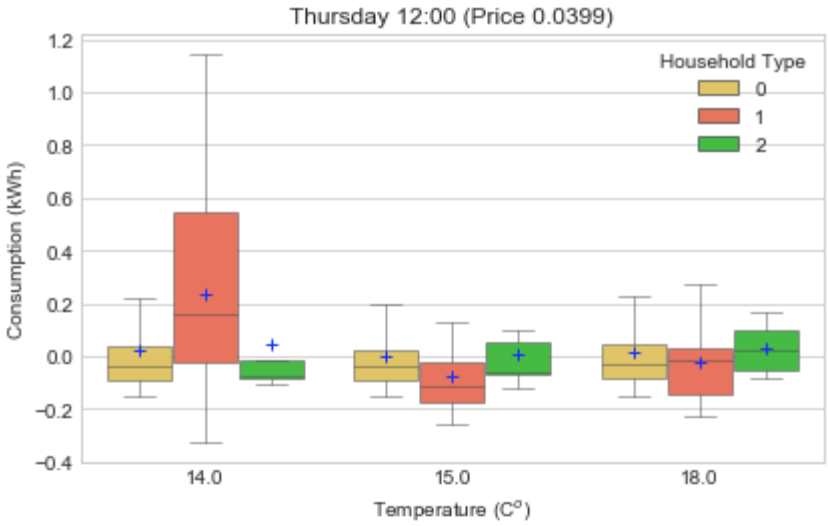

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

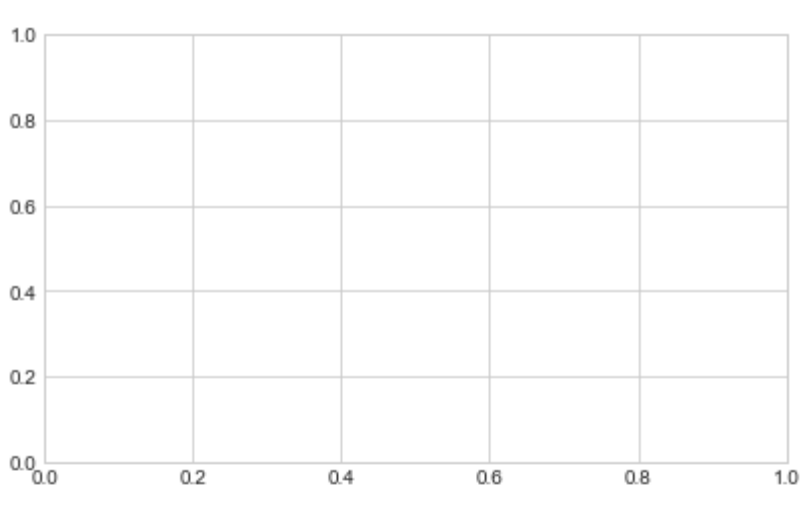

Thursday 11:00 (Price 0.0399)
Household Type
0
0.4

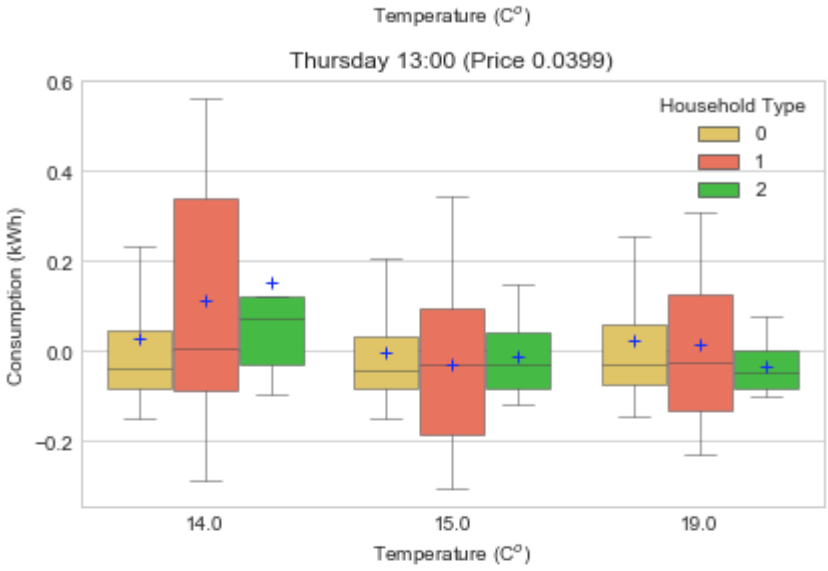
1.0

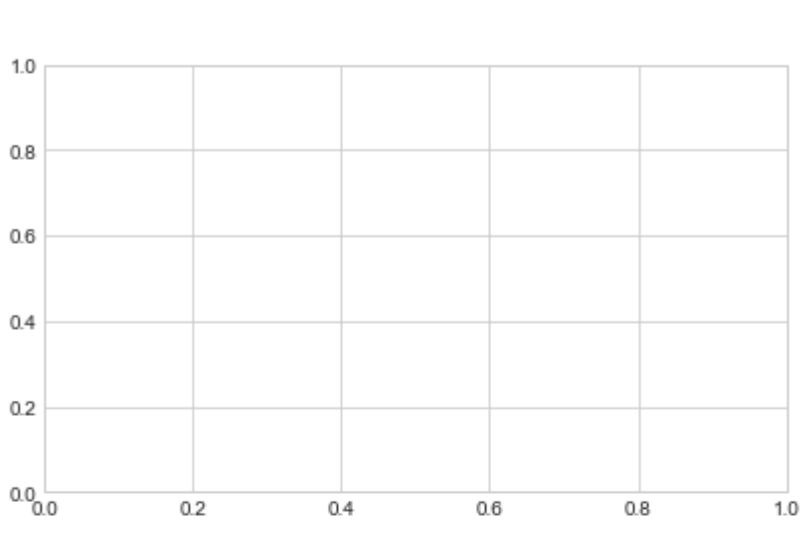

1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
14.0
19.0
Temperature (C°)

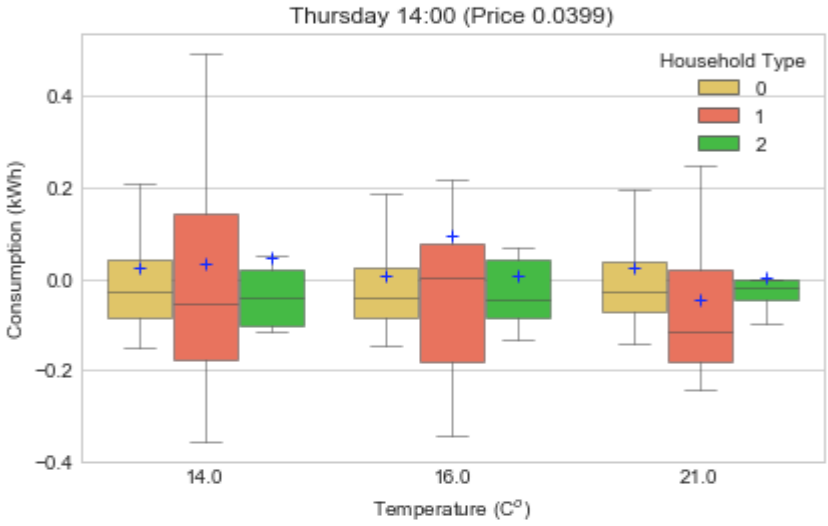

0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

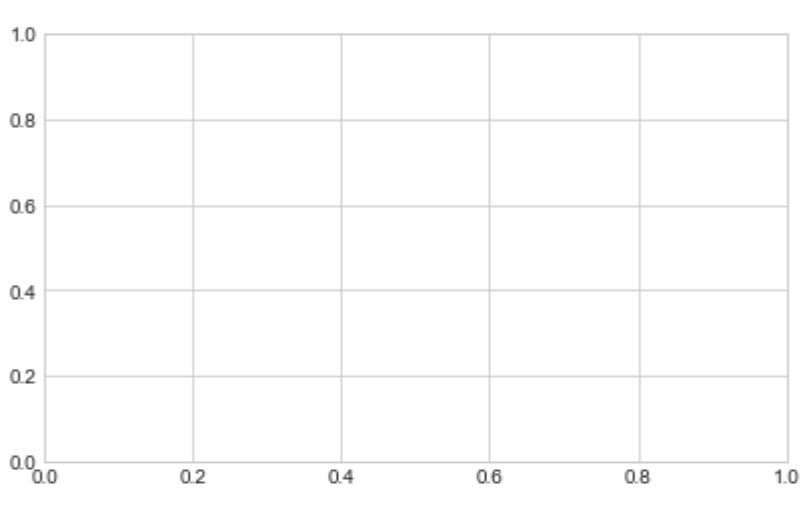

Thursday 12:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
14.0
15.0
18.0
Temperature (C°)

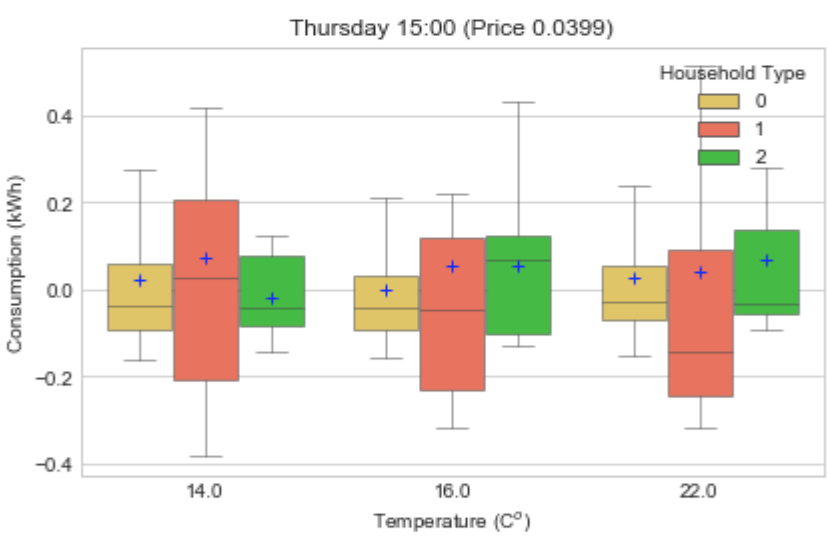

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

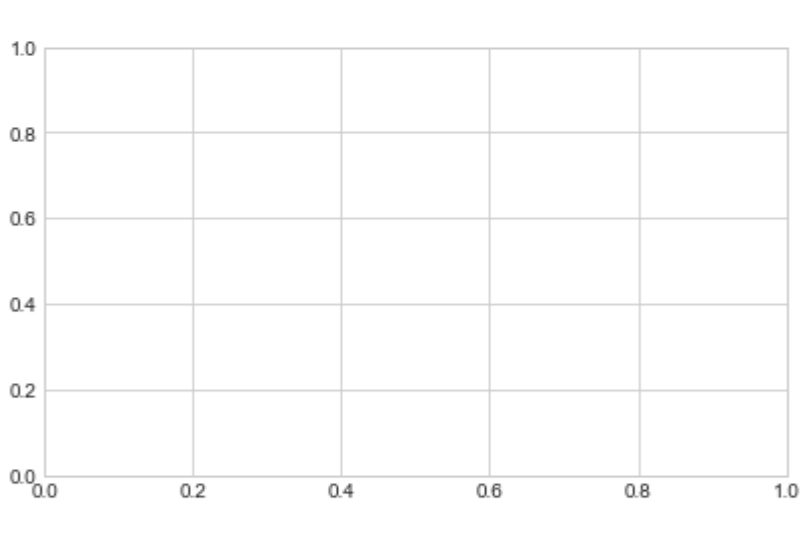

Thursday 13:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
14.0
15.0
19.0
Temperature (C°)

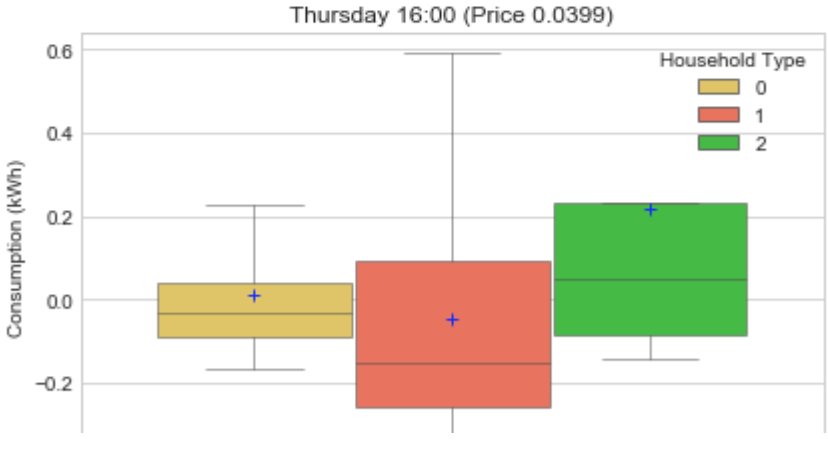

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

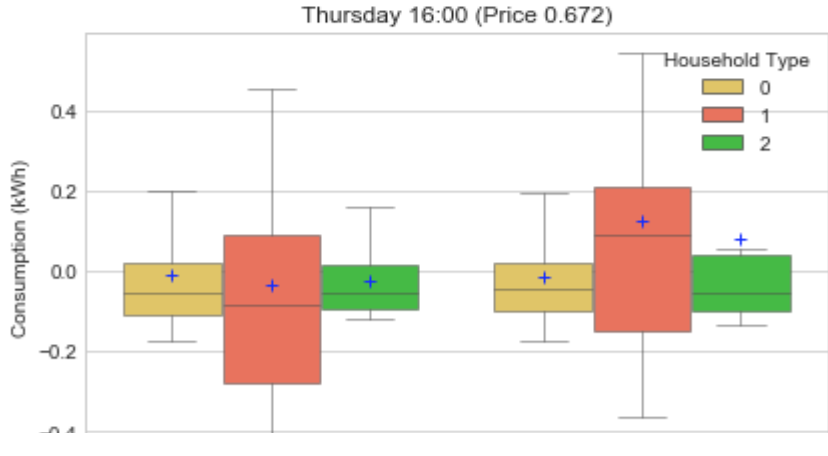

Thursday 14:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
14.0
16.0
21.0
Temperature (C°)

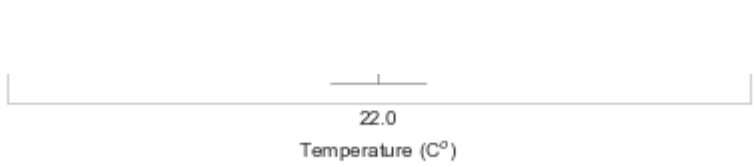

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Thursday 15:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
14.0
16.0
22.0
Temperature (C°)

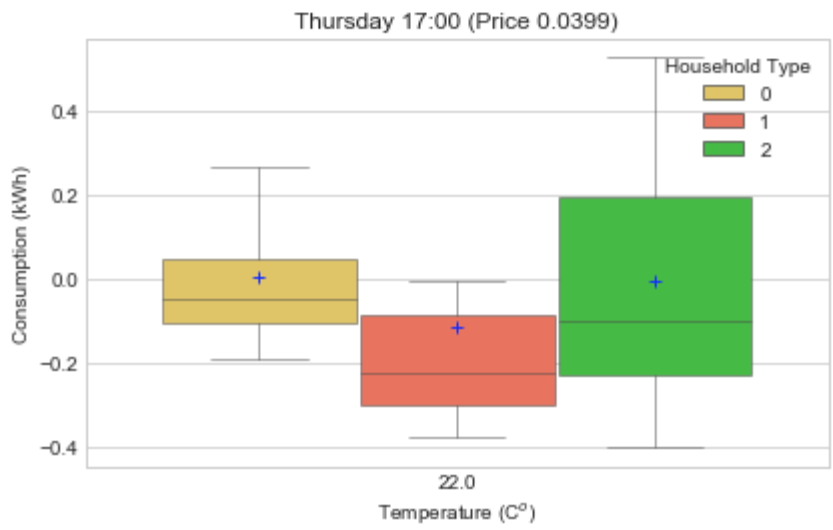

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

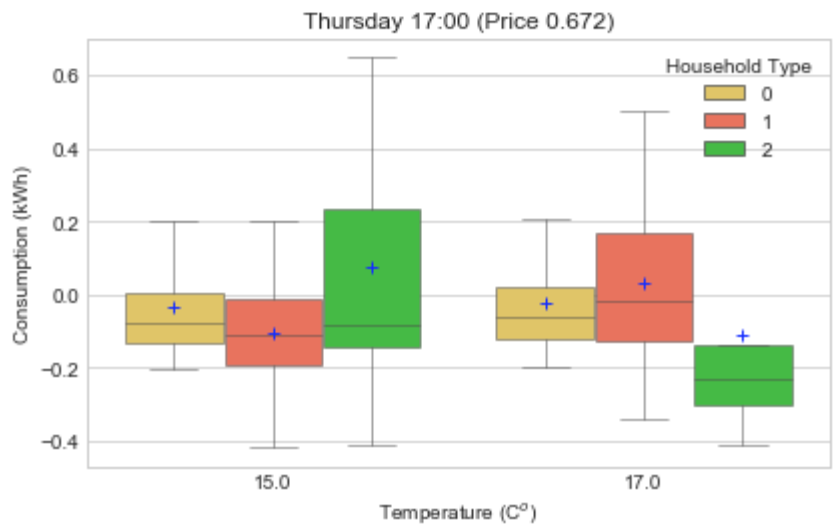

Thursday 16:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2

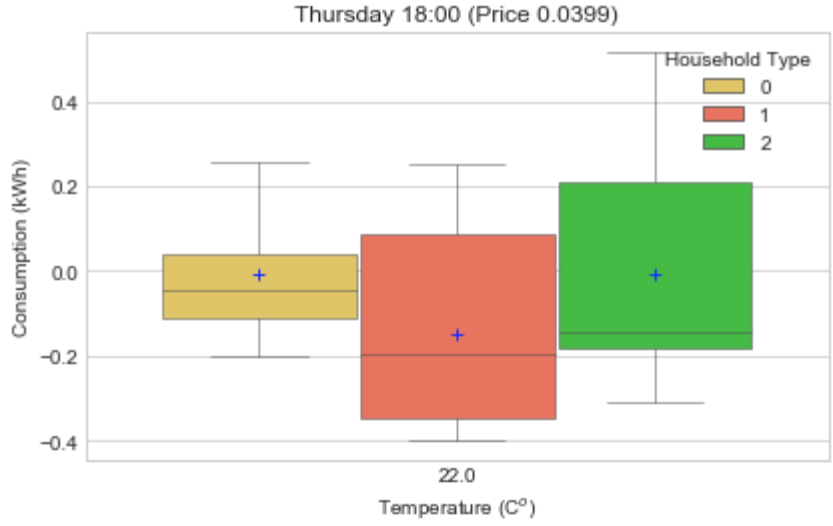

Thursday 16:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2

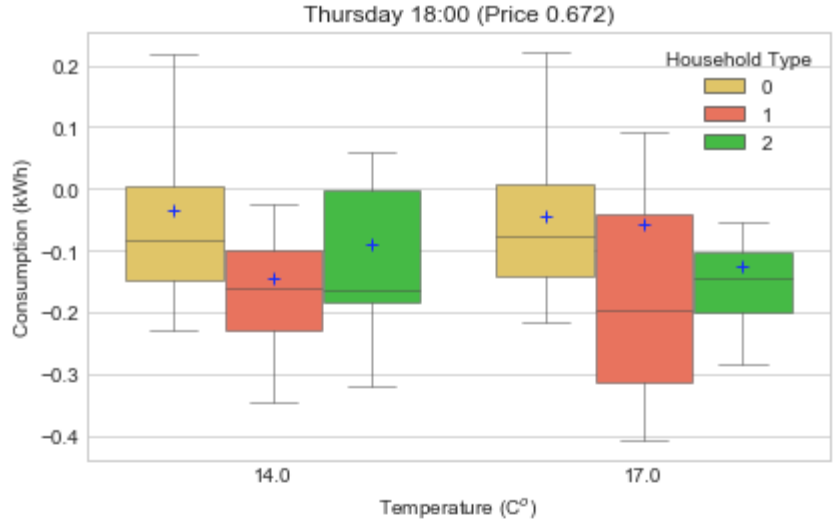

22.0
Temperature (C°)

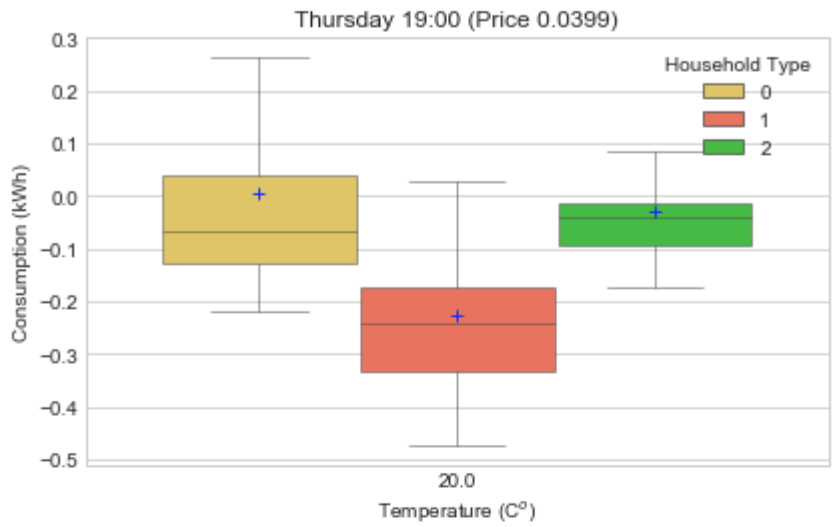

15.0
16.0
Temperature (C°)

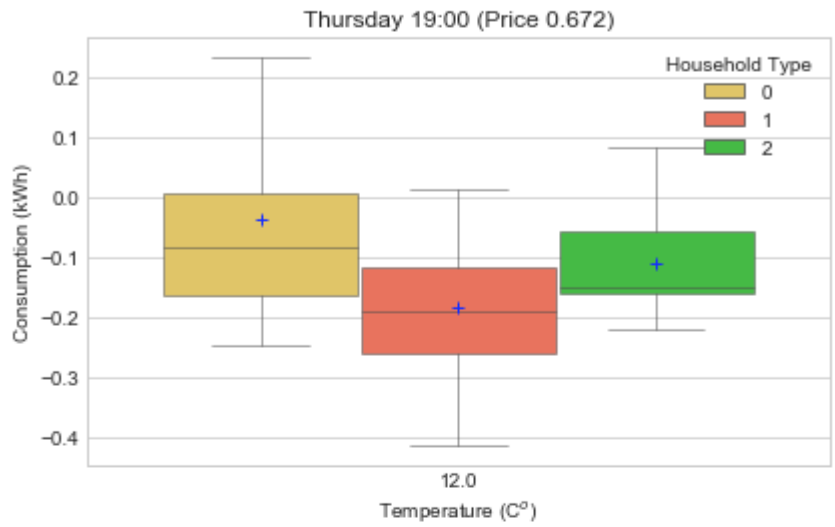

Thursday 17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
22.0
Temperature (C°)

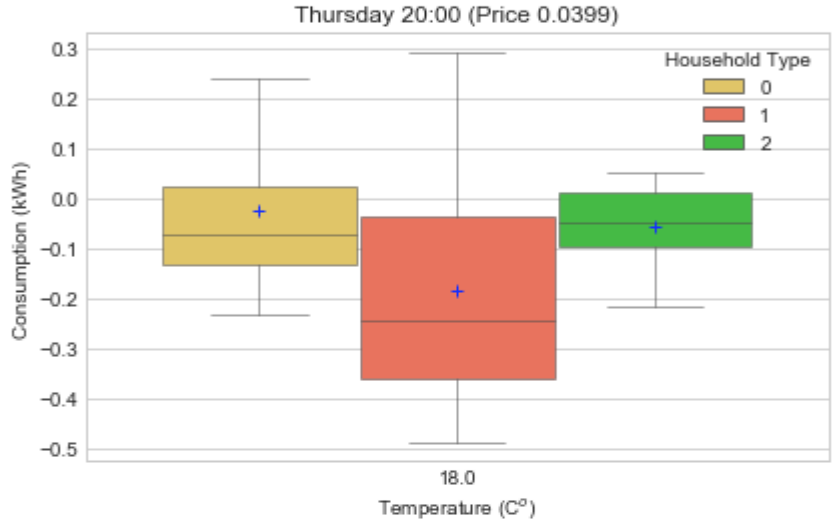

Thursday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
17.0
Temperature (C°)

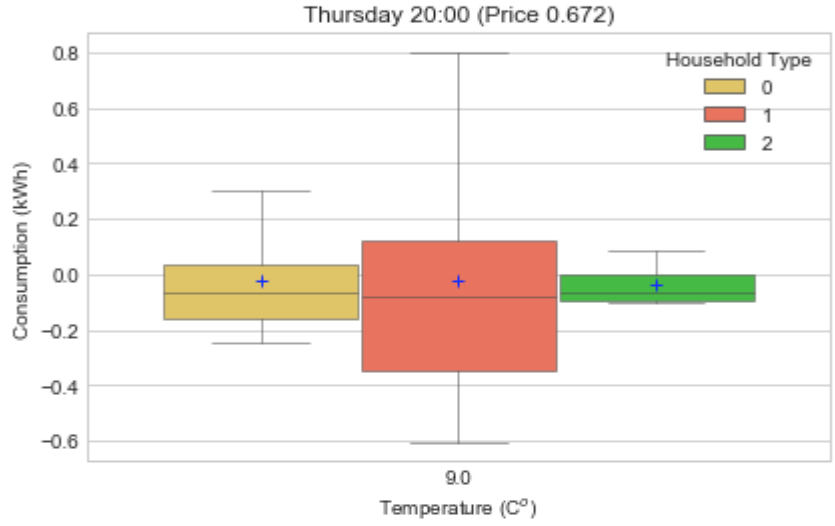

Thursday 18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
22.0
Temperature (C°)

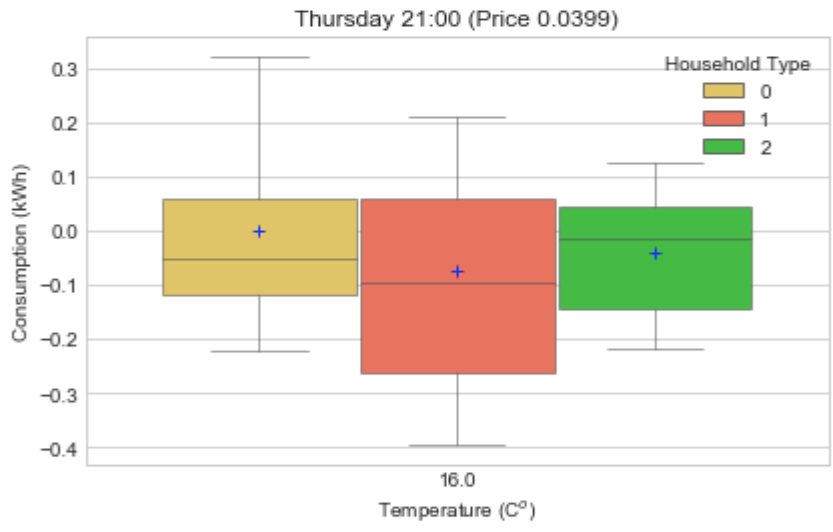

Thursday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
14.0
17.0
Temperature (C°)

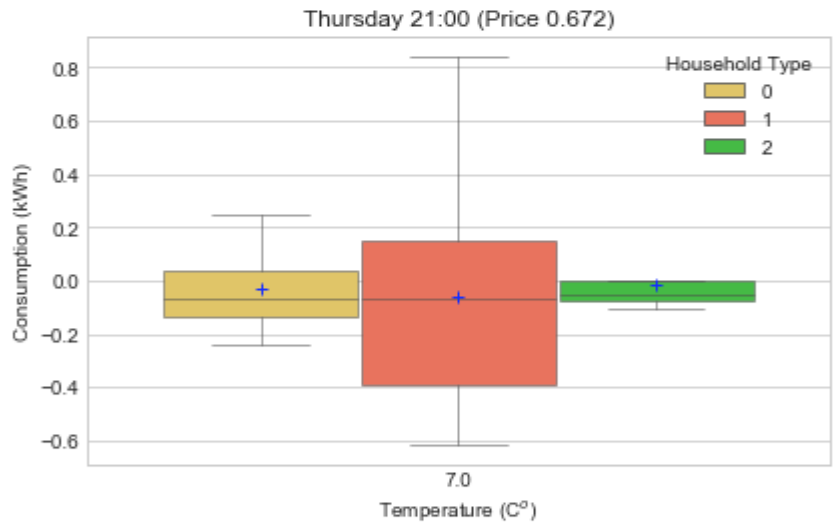

Thursday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
20.0
Temperature (C°)

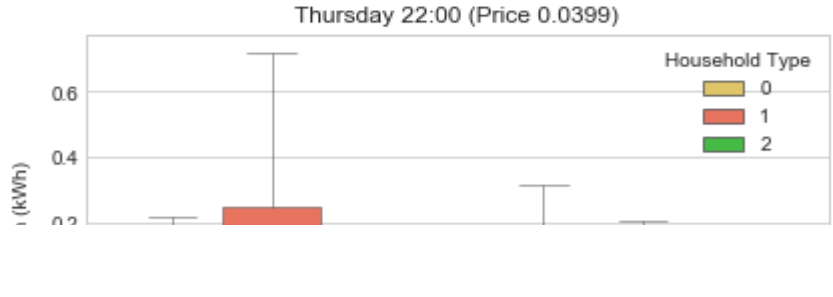

Thursday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
12.0
Temperature (C°)

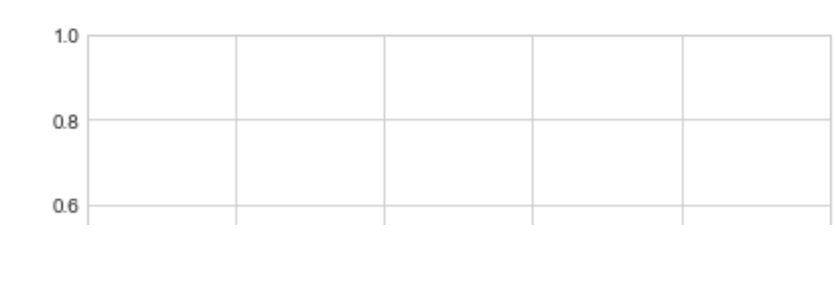

Thursday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
18.0
Temperature (C°)

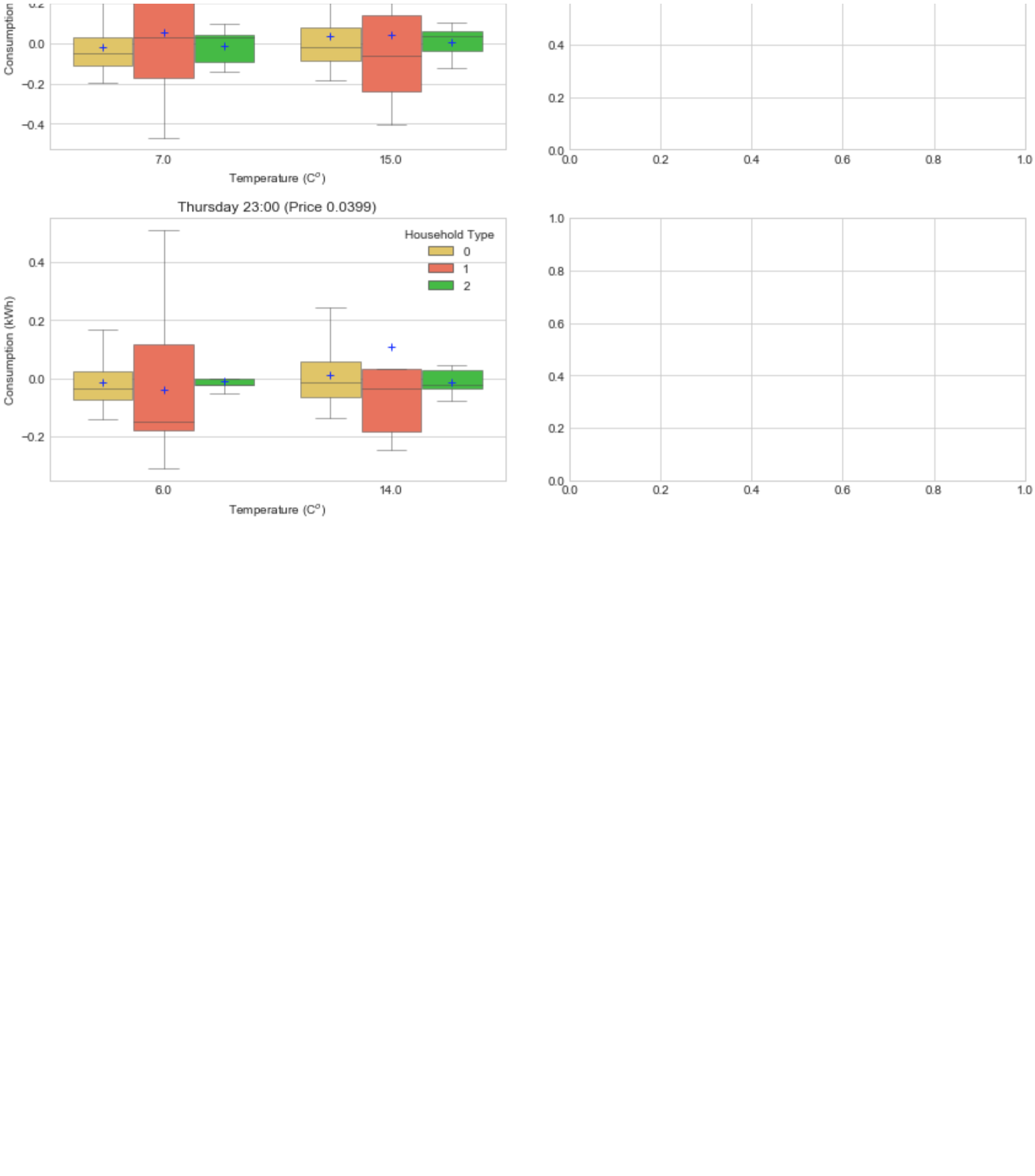

Thursday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
9.0
Temperature (C°)


Thursday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
16.0
Temperature (C°)

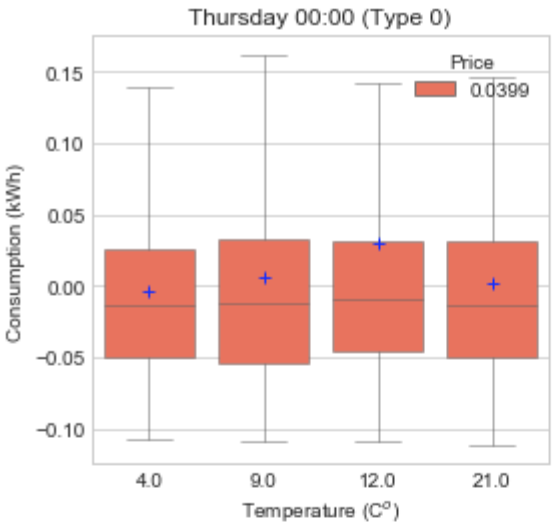

Thursday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
7.0
Temperature (C°)

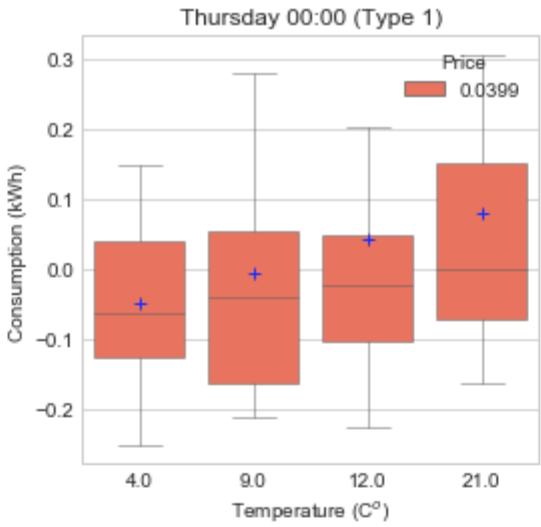

Thursday 22:00 (Price 0.0399)
Household Type
0
1
2
0.6
0.4

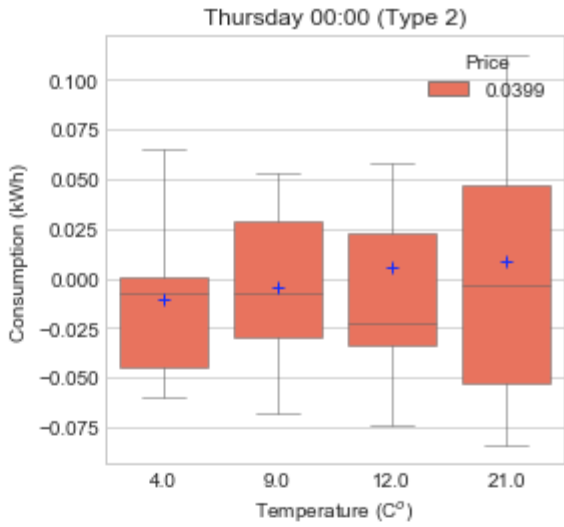

1.0
0.8
0.6

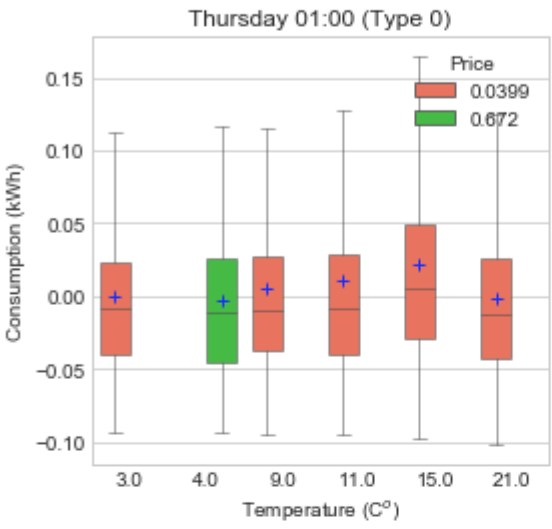

Consumption (kWh)
0.4
0.0
−0.2
−0.4
7.0
15.0
Temperature (C°)
Thursday 23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
6.0
14.0
Temperature (C°)
0.0
0.2
0.4
0.6
0.8
1.0

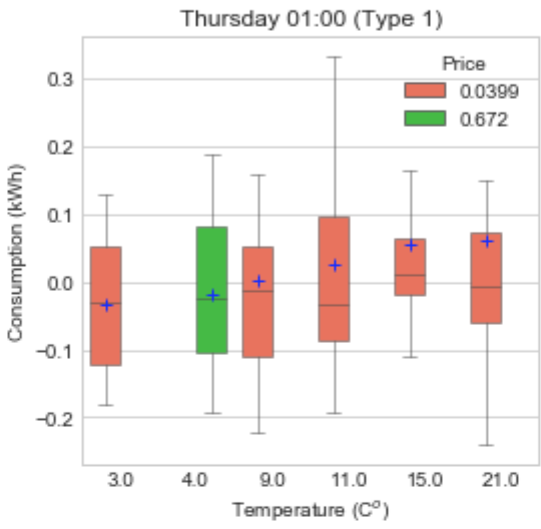
In [68]:
```
# price comparison
# Thursday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 3 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

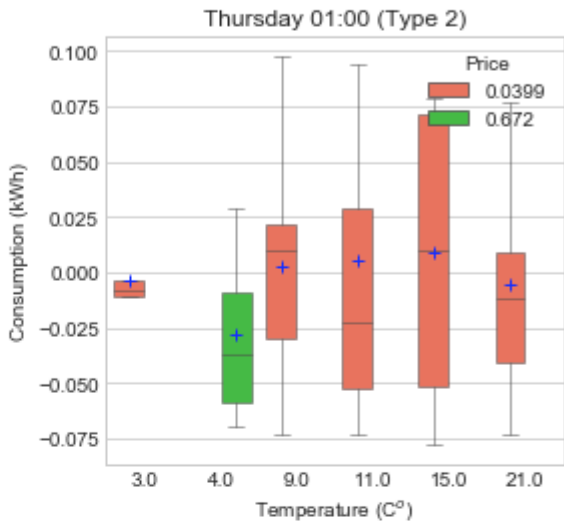

Thursday 00:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
4.0
9.0
12.0
21.0
Temperature (C°)

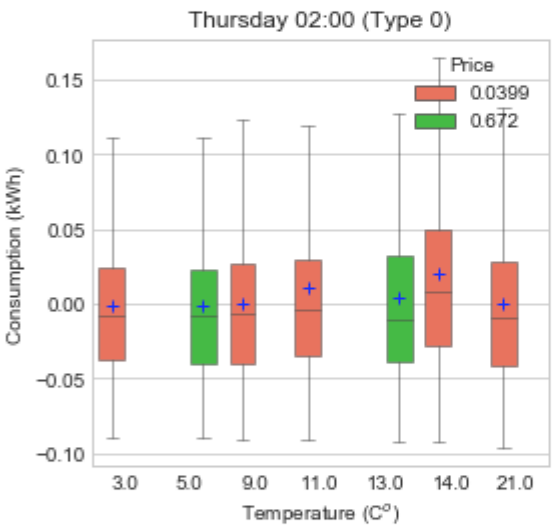

Thursday 00:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
9.0
12.0
21.0
Temperature (C°)

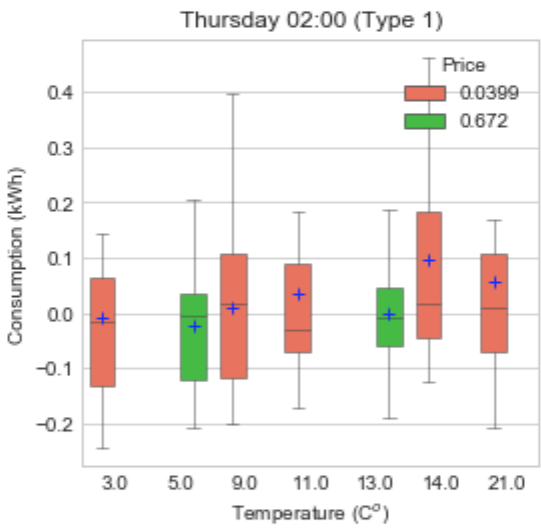

Thursday 00:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
4.0
9.0
12.0
21.0
Temperature (C°)

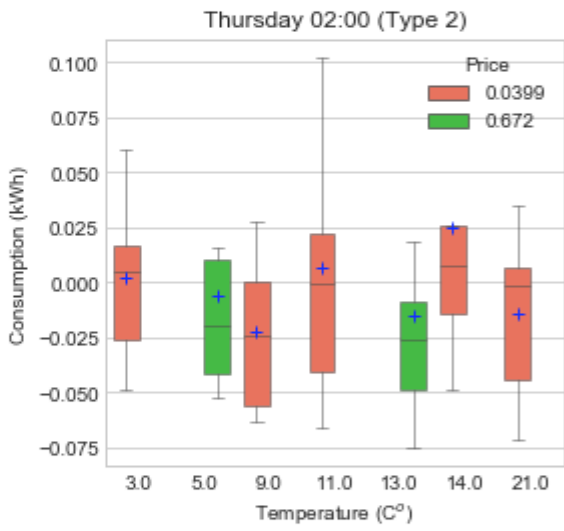

Thursday 01:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
3.0
4.0
9.0
11.0
15.0
21.0
Temperature (C°)

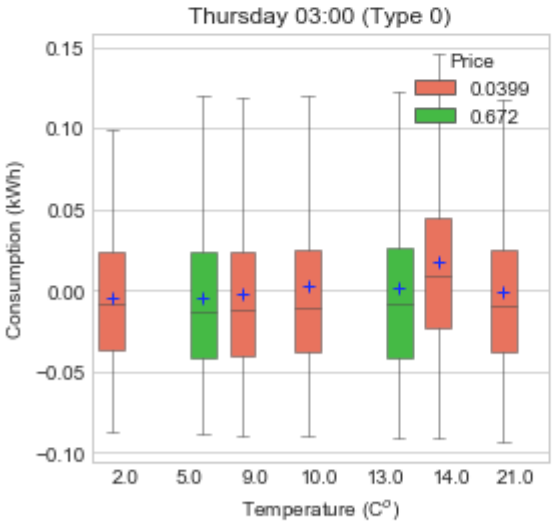

Thursday 01:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
4.0
9.0
11.0
15.0
21.0
Temperature (C°)

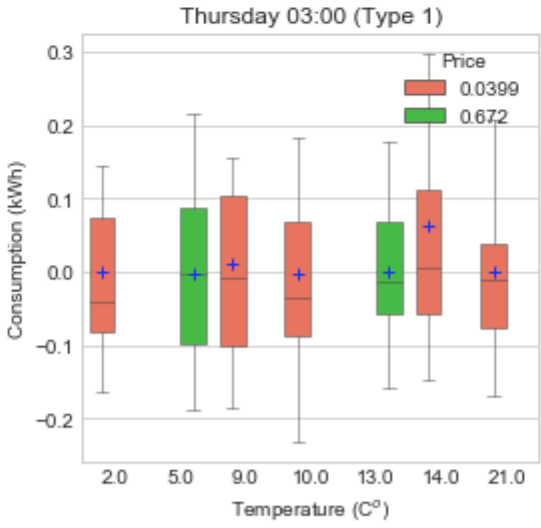

Thursday 01:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
3.0
4.0
9.0
11.0
15.0
21.0
Temperature (C°)

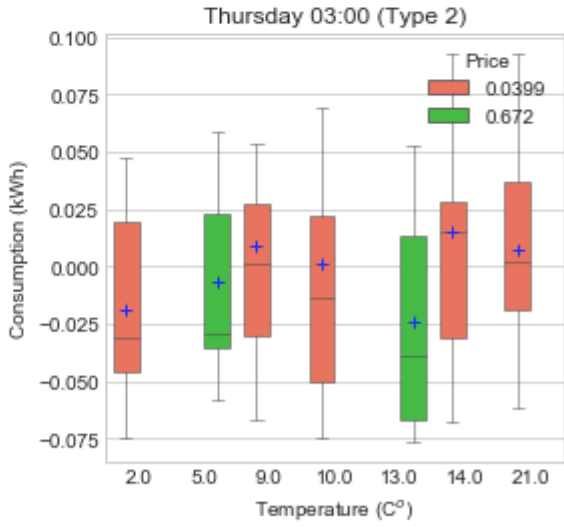

Thursday 02:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
3.0
5.0
9.0
11.0
13.0
14.0
21.0
Temperature (C°)

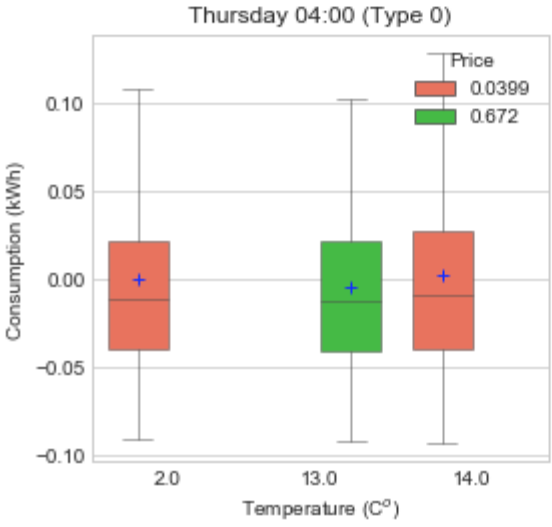

Thursday 02:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
5.0
9.0
11.0
13.0
14.0
21.0
Temperature (C°)

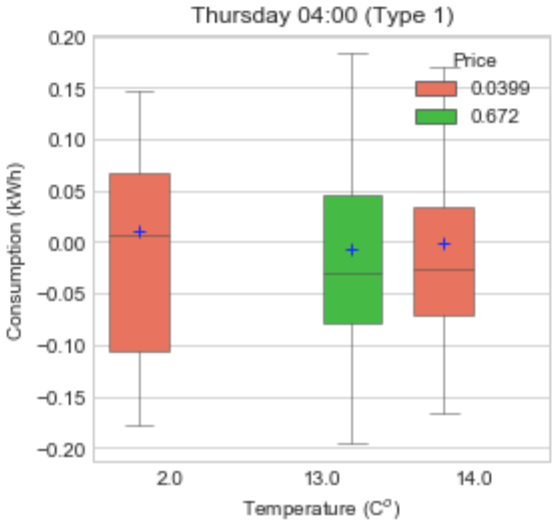

Thursday 02:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
3.0
5.0
9.0
11.0
13.0
14.0
21.0
Temperature (C°)

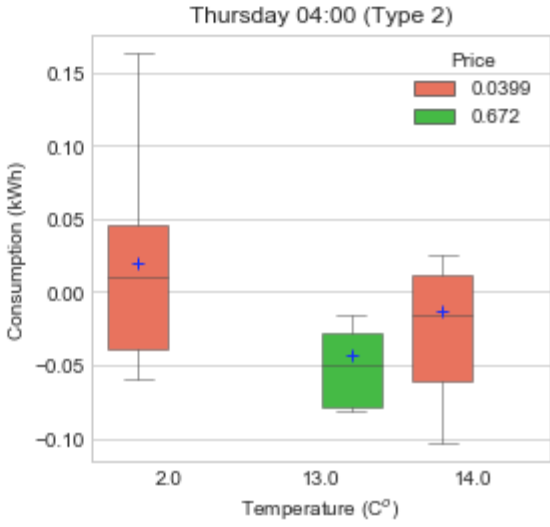

Thursday 03:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
2.0
5.0
9.0
10.0
13.0
14.0
21.0
Temperature (C°)

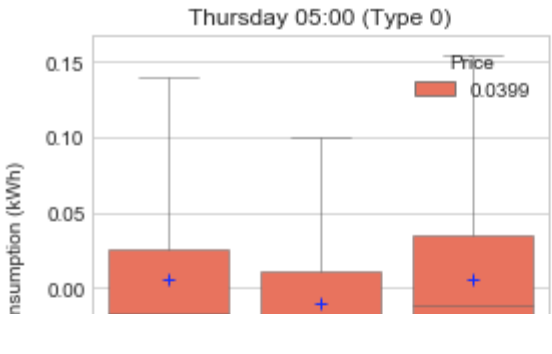

Thursday 03:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
2.0
5.0
9.0
10.0
13.0
14.0
21.0
Temperature (C°)

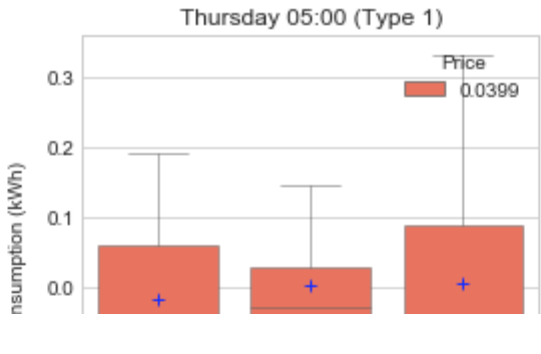

Thursday 03:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
2.0
5.0
9.0
10.0
13.0
14.0
21.0
Temperature (C°)

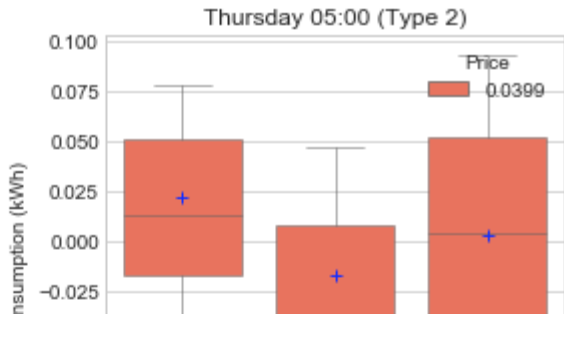

Thursday 04:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
2.0
13.0
14.0
Temperature (C°)

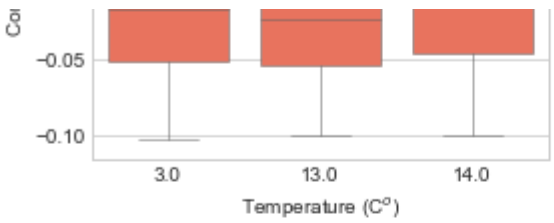

Thursday 04:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2.0
13.0
14.0
Temperature (C°)

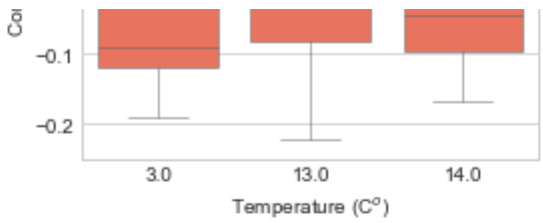

Thursday 04:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
2.0
13.0
14.0
Temperature (C°)

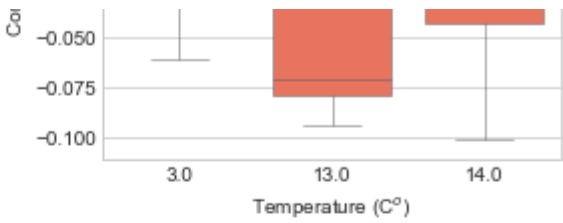

Thursday 05:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00

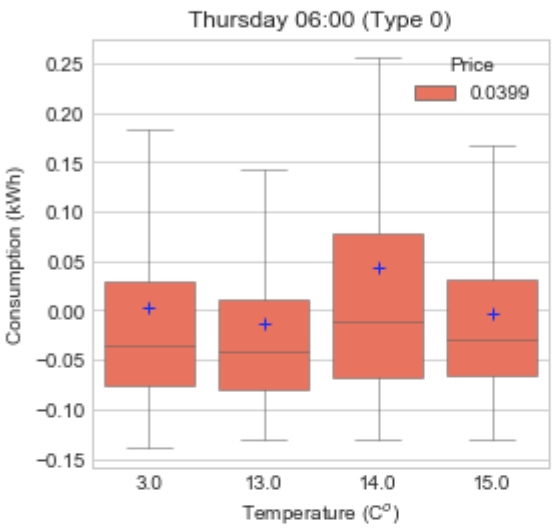

Thursday 05:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0

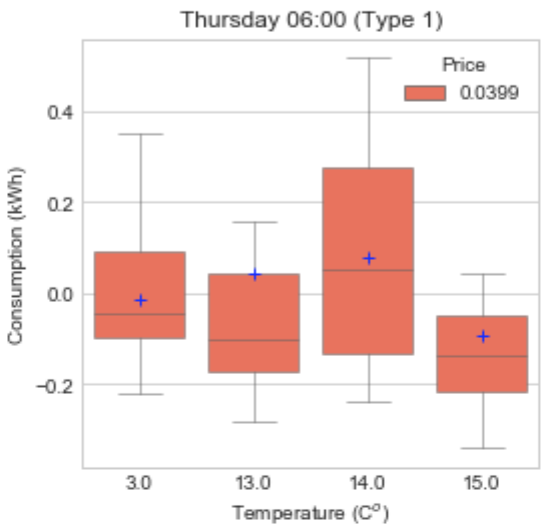

Thursday 05:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025

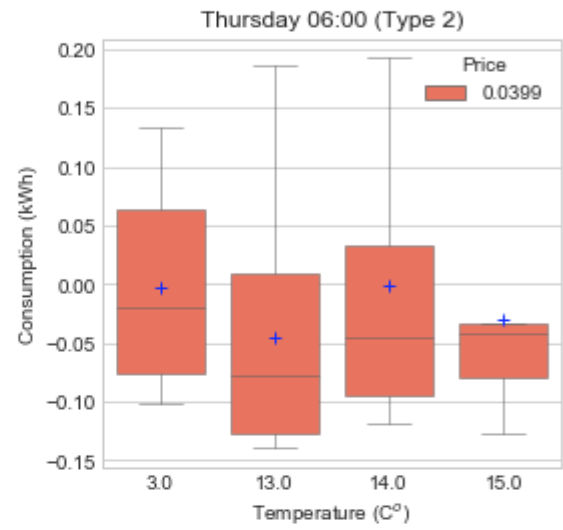

−0.05
−0.10
3.0
13.0
14.0
Temperature (C°)

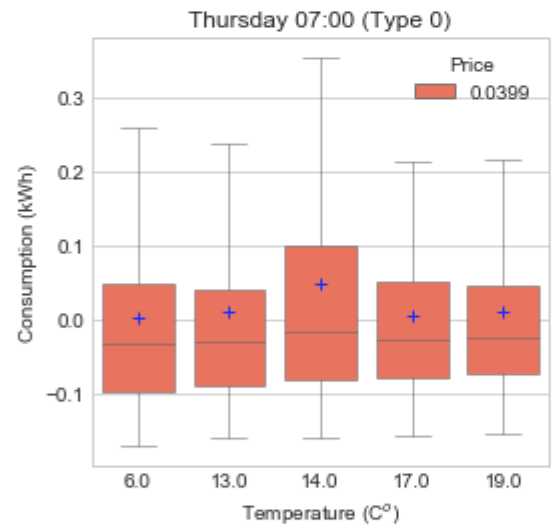

−0.1
−0.2
3.0
13.0
14.0
Temperature (C°)

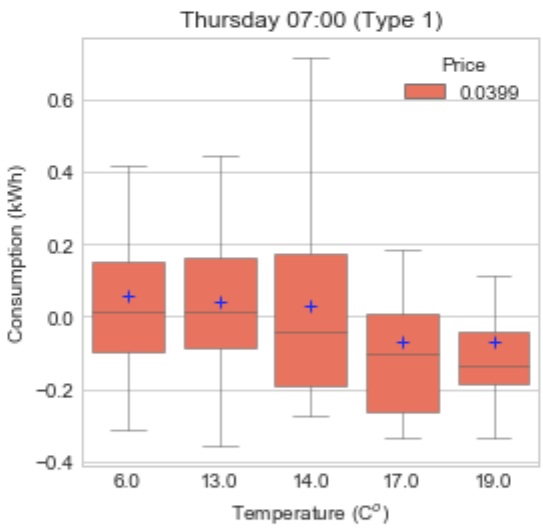

−0.050
−0.075
−0.100
3.0
13.0
14.0
Temperature (C°)

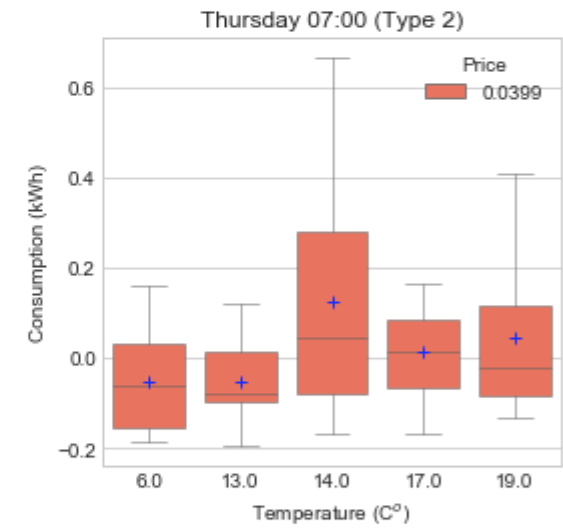

Thursday 06:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
3.0
13.0
14.0
15.0
Temperature (C°)

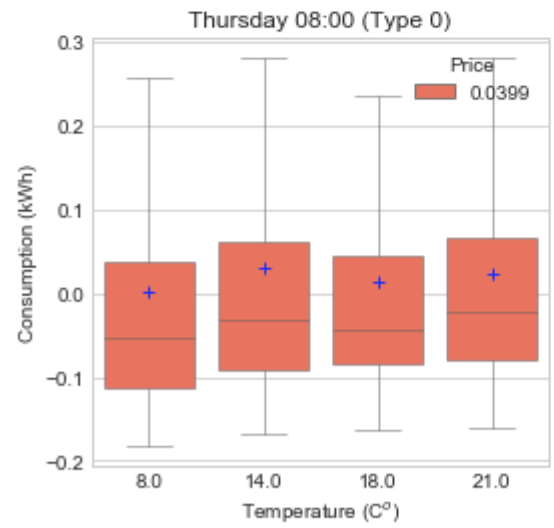

Thursday 06:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.2
0.0
−0.2
3.0
13.0
14.0
15.0
Temperature (C°)

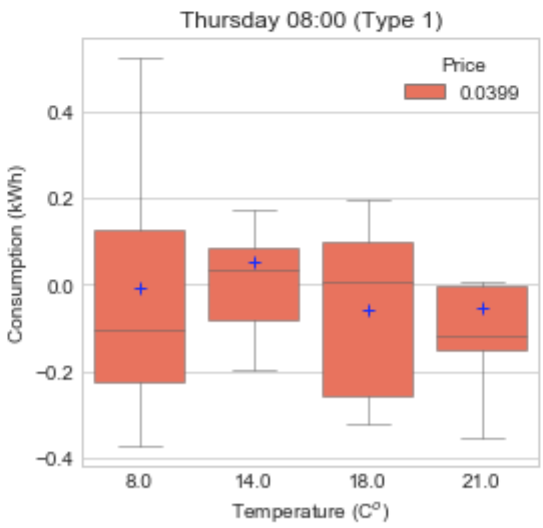

Thursday 06:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
3.0
13.0
14.0
15.0
Temperature (C°)

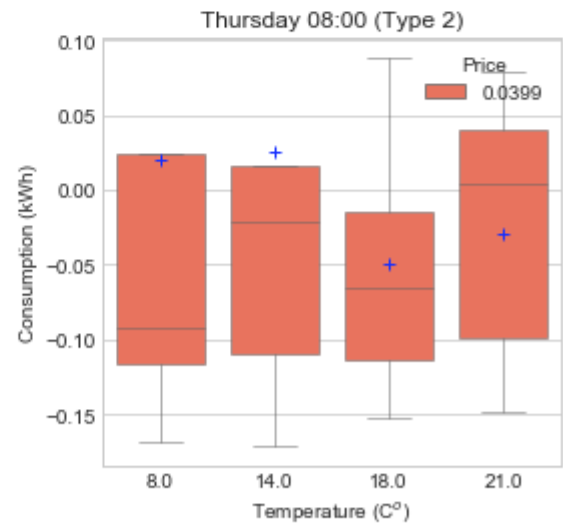

Thursday 07:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
6.0
13.0
14.0
17.0
19.0
Temperature (C°)

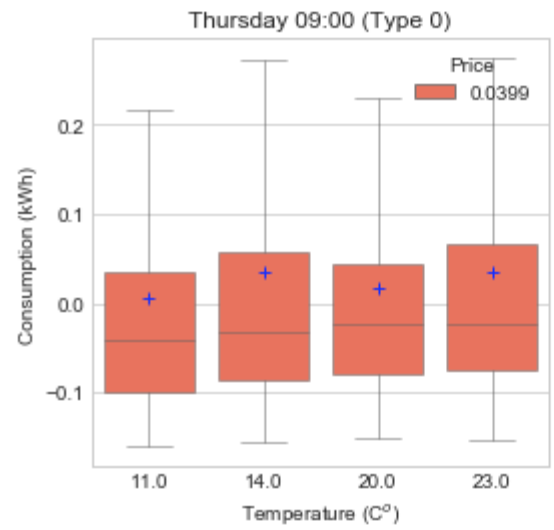

Thursday 07:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
6.0
13.0
14.0
17.0
19.0
Temperature (C°)

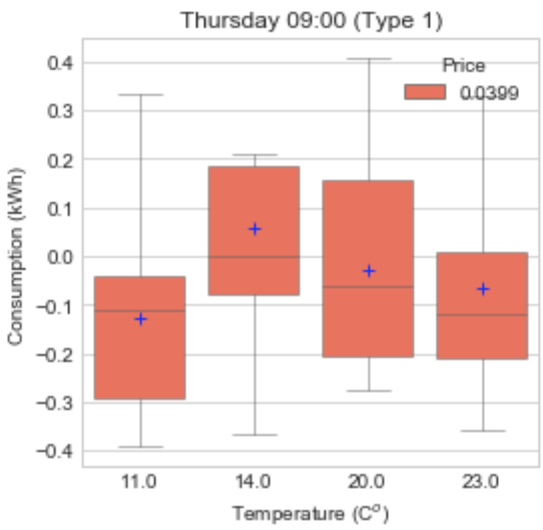

Thursday 07:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
6.0
13.0
14.0
17.0
19.0
Temperature (C°)

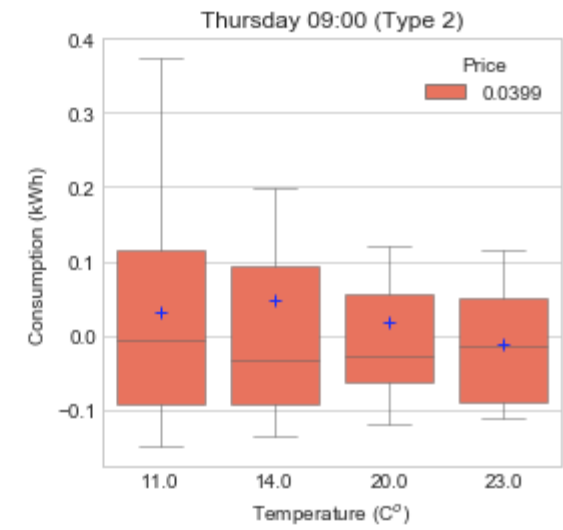

Thursday 08:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
8.0
14.0
18.0
21.0
Temperature (C°)

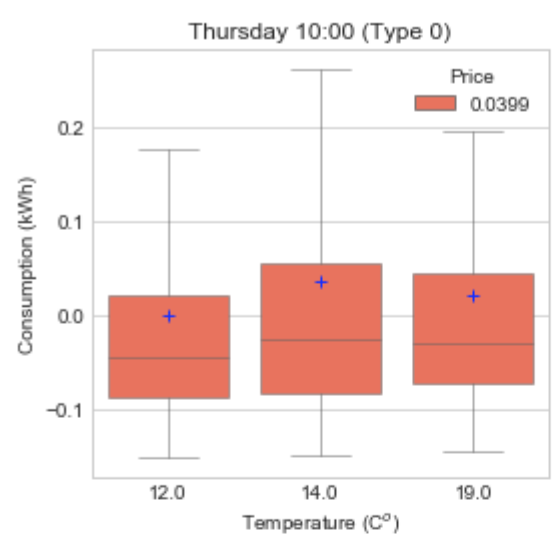

Thursday 08:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
8.0
14.0
18.0
21.0
Temperature (C°)

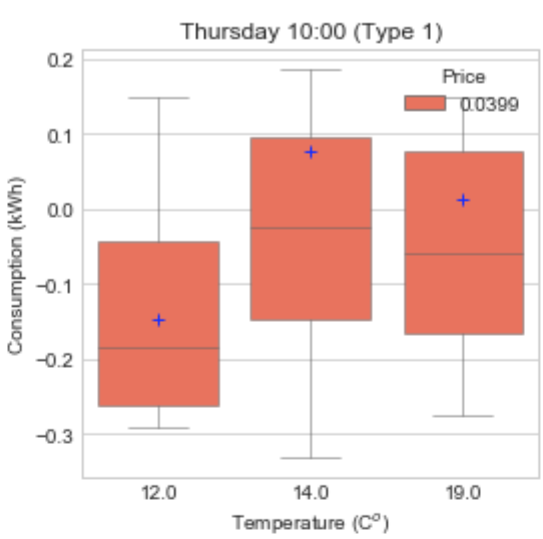

Thursday 08:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
8.0
14.0
18.0
21.0
Temperature (C°)

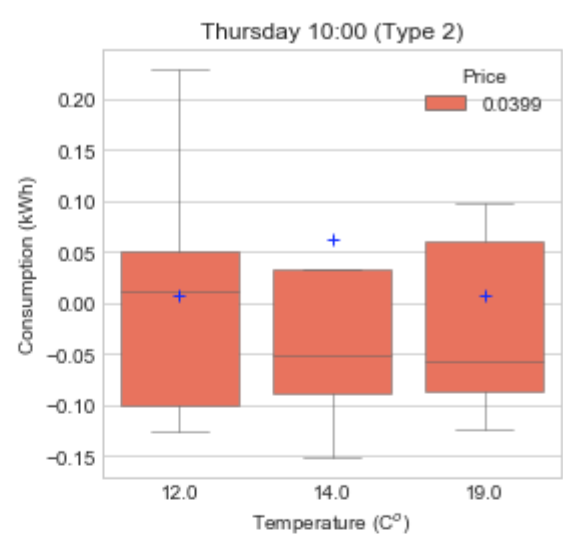

Thursday 09:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
11.0
14.0
20.0
23.0
Temperature (C°)

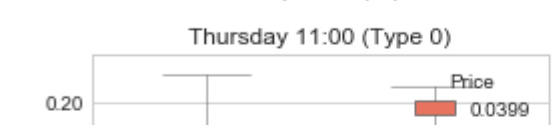

Thursday 09:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
11.0
14.0
20.0
23.0
Temperature (C°)

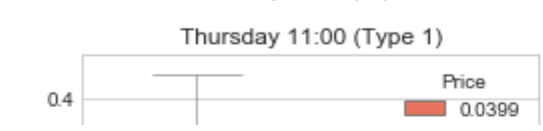

Thursday 09:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
11.0
14.0
20.0
23.0
Temperature (C°)

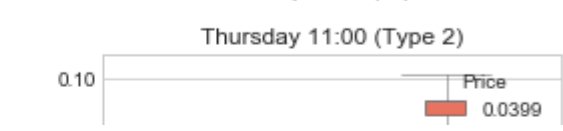

Thursday 10:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
12.0
14.0
19.0
Temperature (C°)

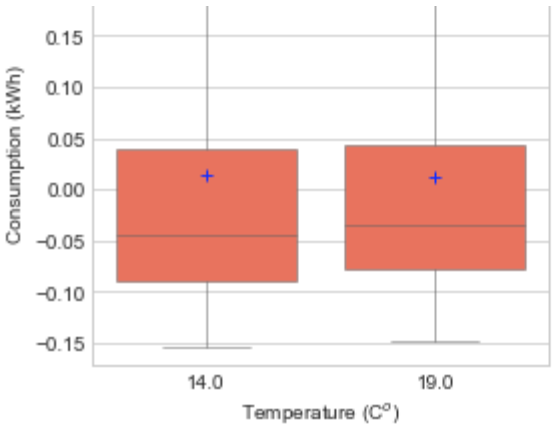

Thursday 10:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
12.0
14.0
19.0
Temperature (C°)

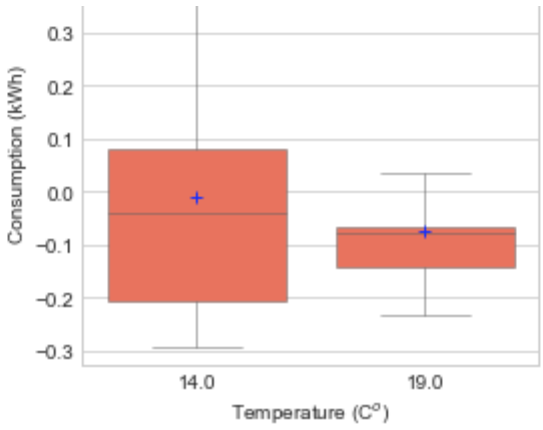

Thursday 10:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
12.0
14.0
19.0
Temperature (C°)

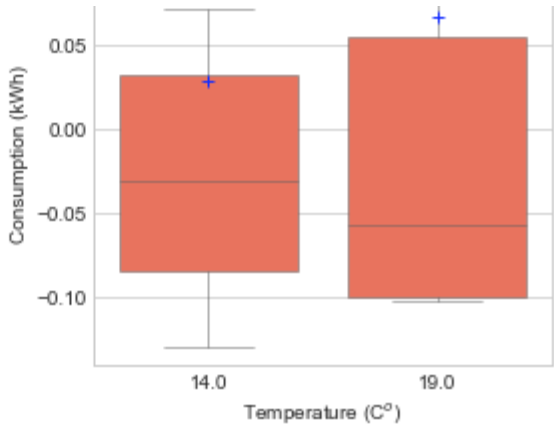

Thursday 11:00 (Type 0)
Price
0.0399
0.20

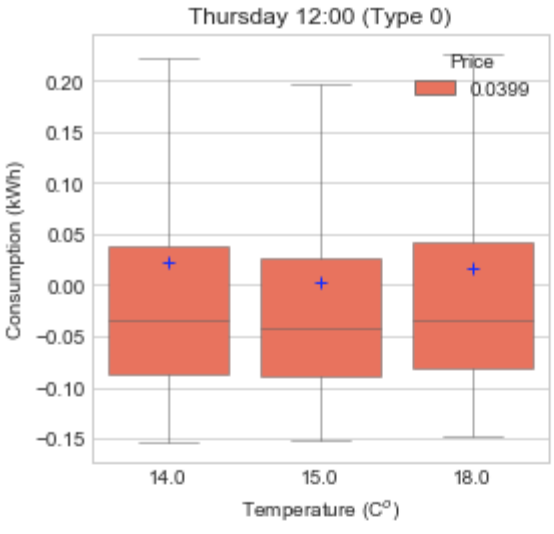

Thursday 11:00 (Type 1)
Price
0.0399
0.4

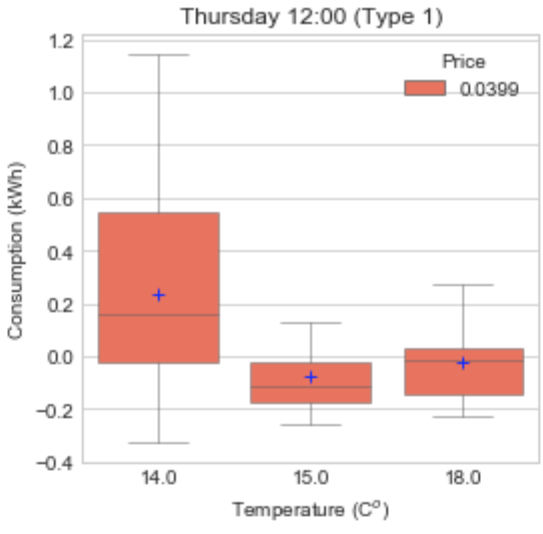

Thursday 11:00 (Type 2)
Price
0.0399
0.10

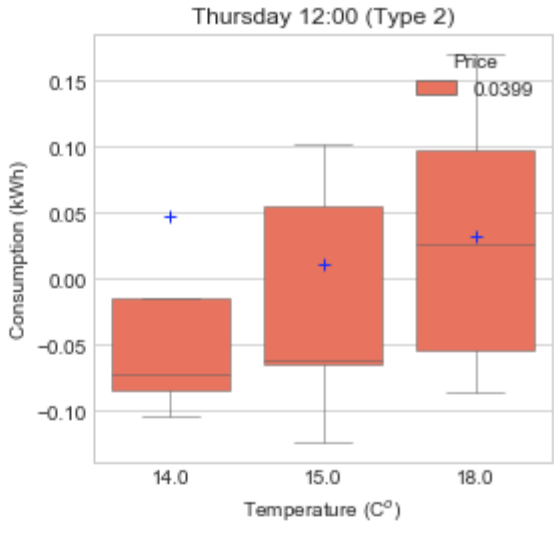

Consumption (kWh)
Temperature (C°)
14.0
19.0

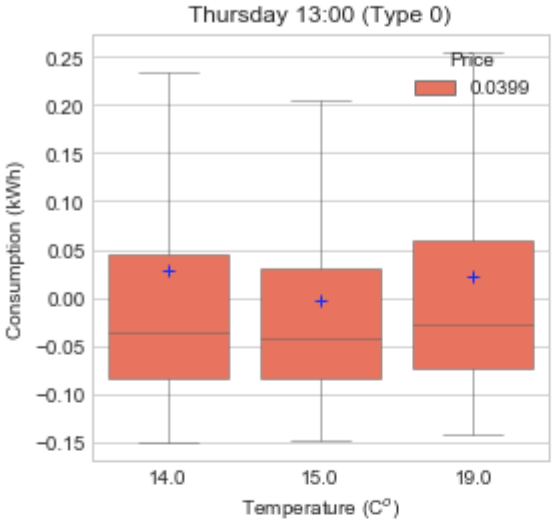

Consumption (kWh)
Temperature (C°)
14.0
19.0

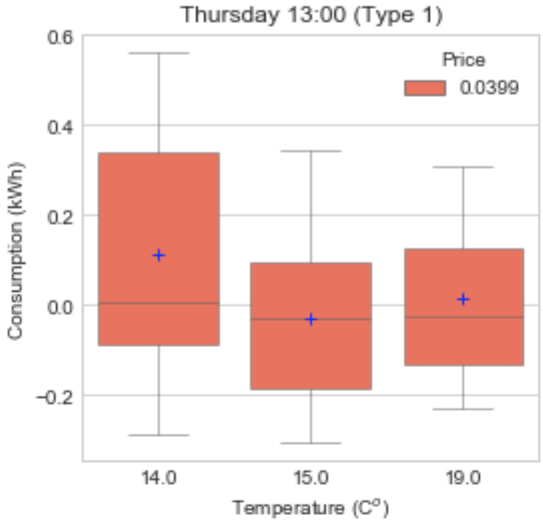

Consumption (kWh)
Temperature (C°)
14.0
19.0

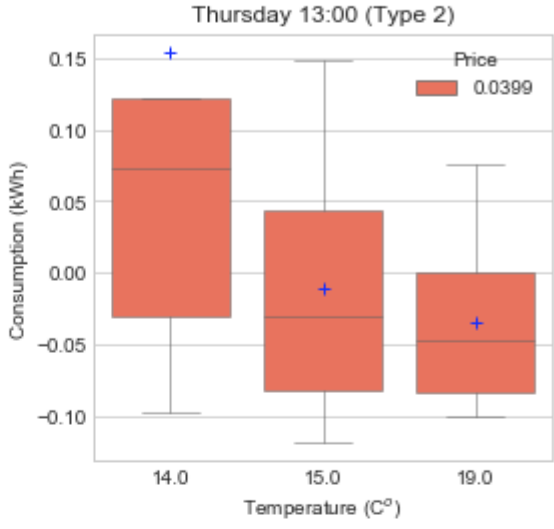

Thursday 12:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
18.0

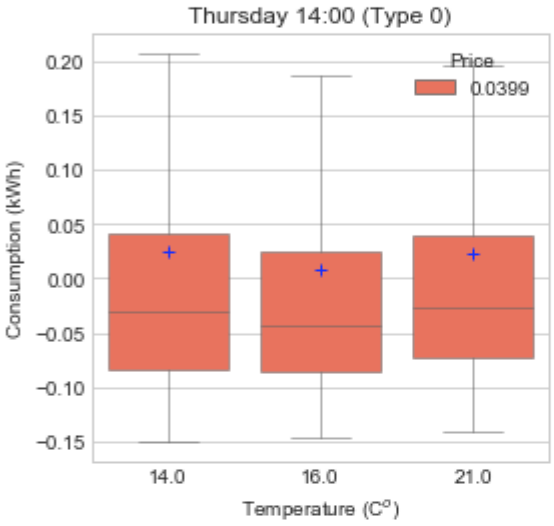

Thursday 12:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
18.0

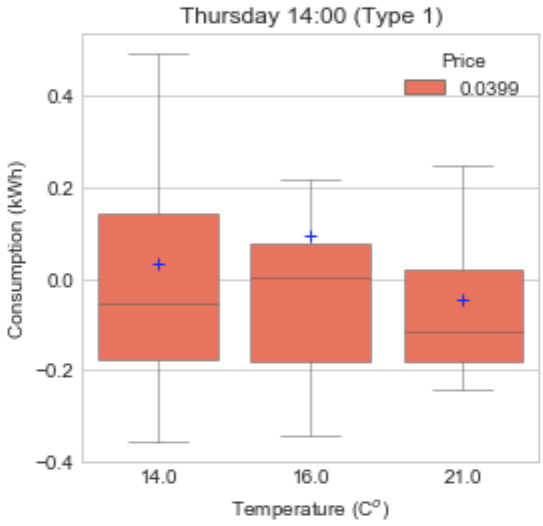

Thursday 12:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
18.0

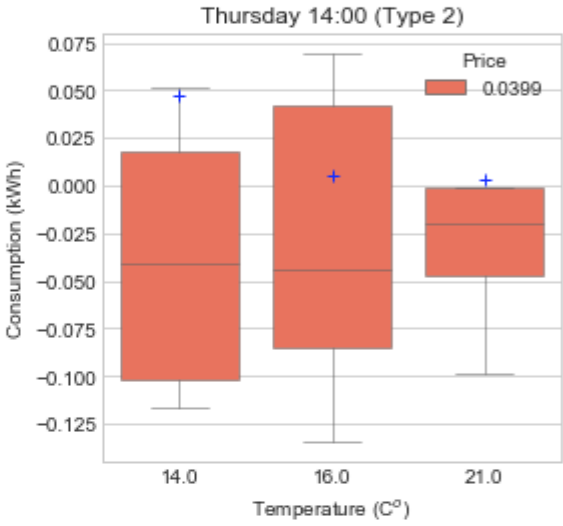

Thursday 13:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
19.0

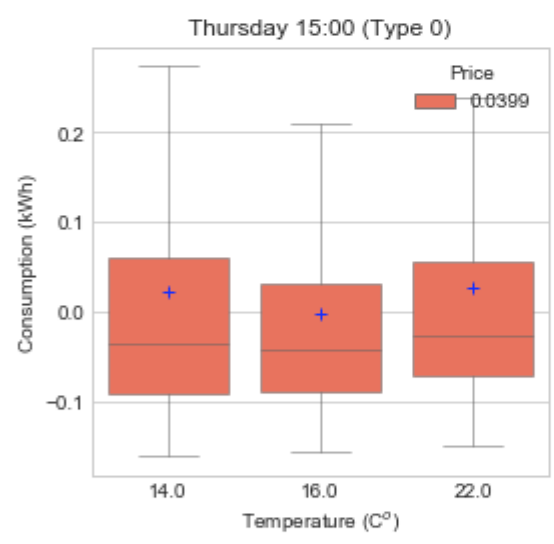

Thursday 13:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
19.0

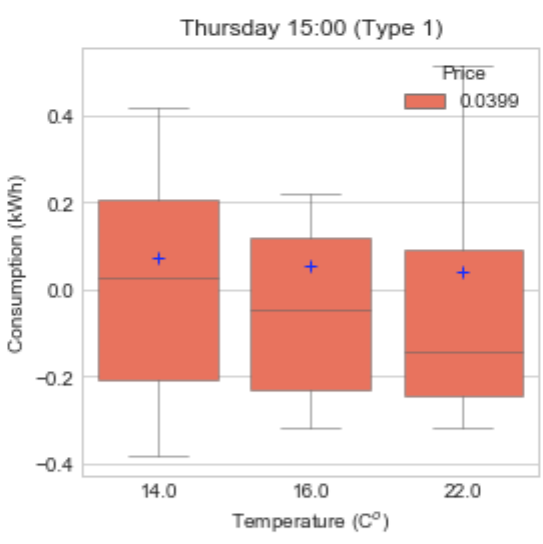

Thursday 13:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
15.0
19.0

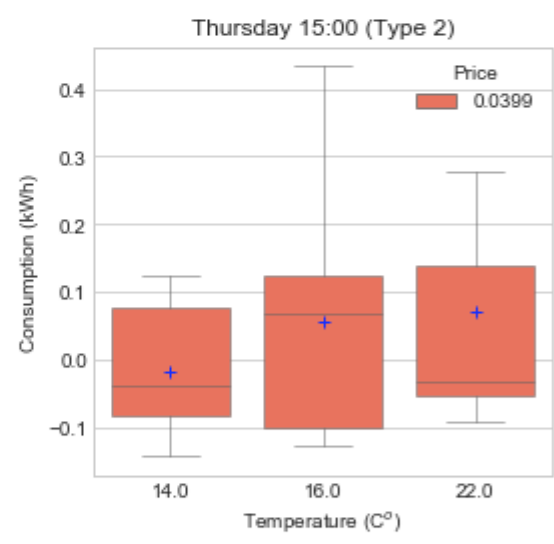

Thursday 14:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
21.0

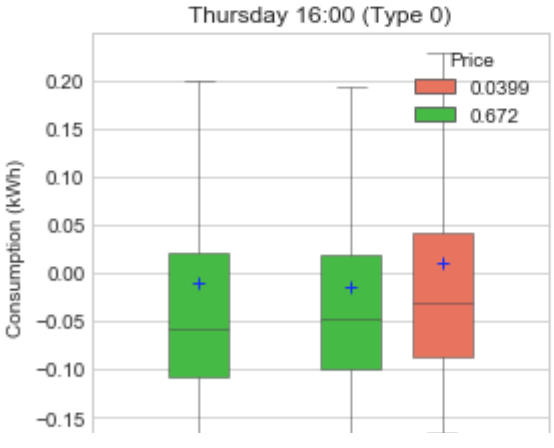

Thursday 14:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
21.0

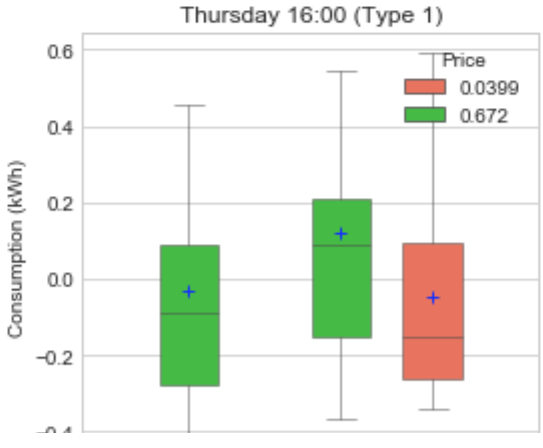

Thursday 14:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
21.0

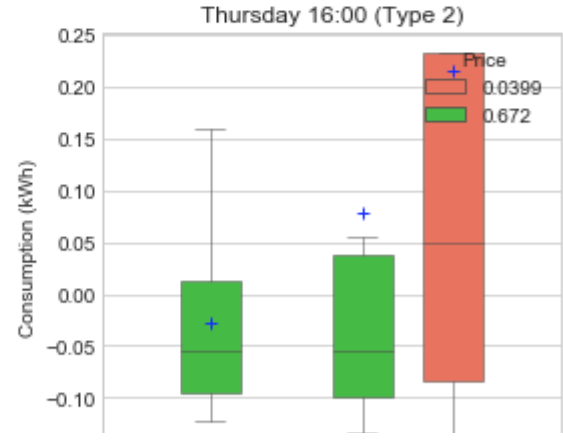

Thursday 15:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
22.0

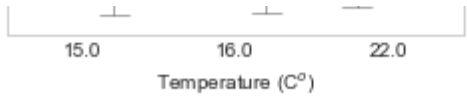

Thursday 15:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
22.0

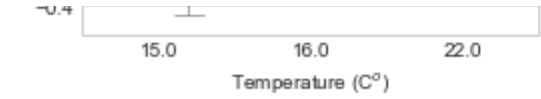

Thursday 15:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
14.0
16.0
22.0


Thursday 16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

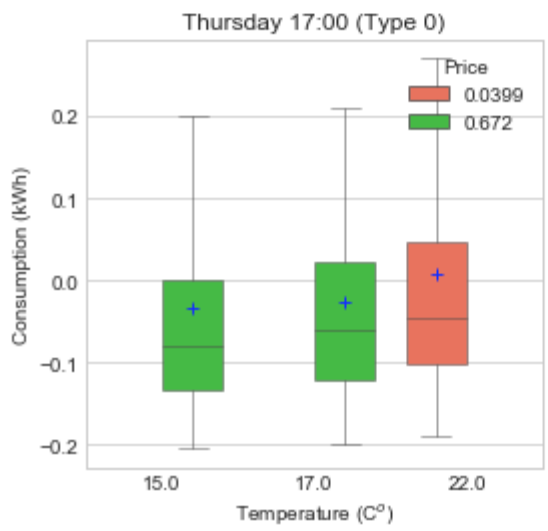

Thursday 16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

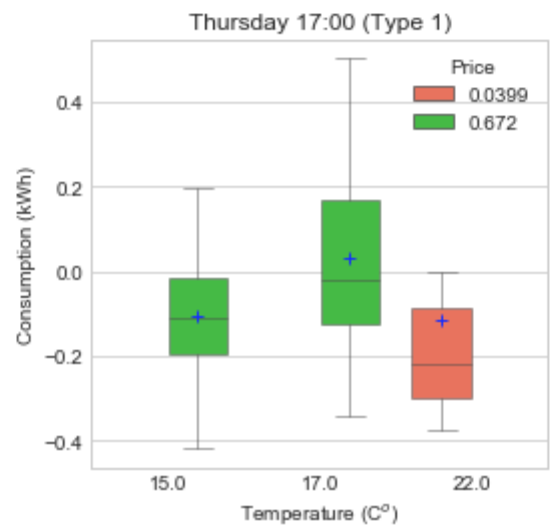

Thursday 16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

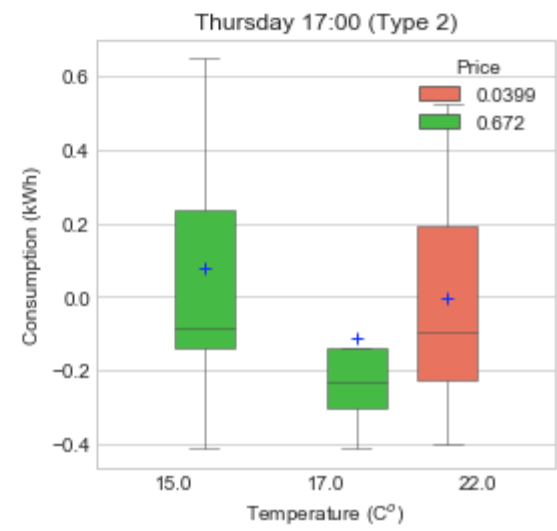

15.0
16.0
22.0
Temperature (C°)

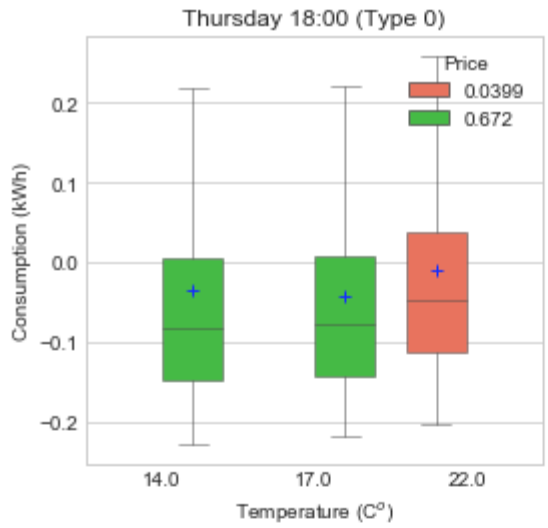

15.0
16.0
22.0
Temperature (C°)

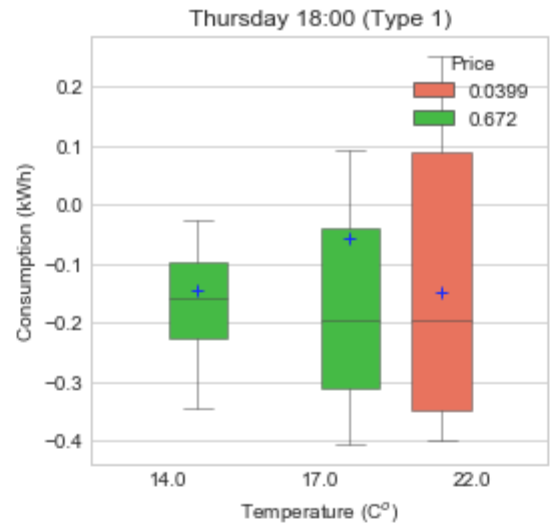

−0.15
15.0
16.0
22.0
Temperature (C°)

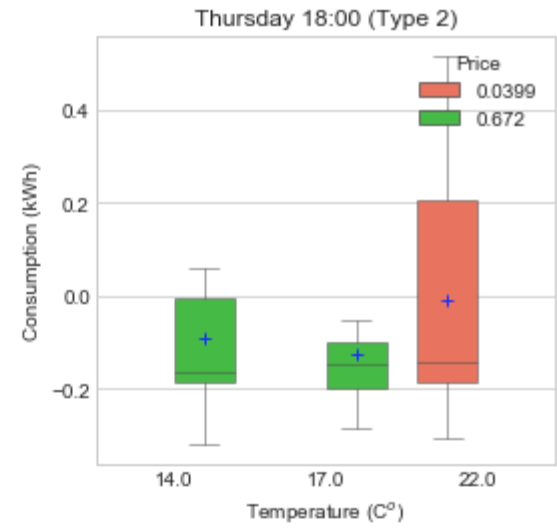

Thursday 17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
15.0
17.0
22.0
Temperature (C°)

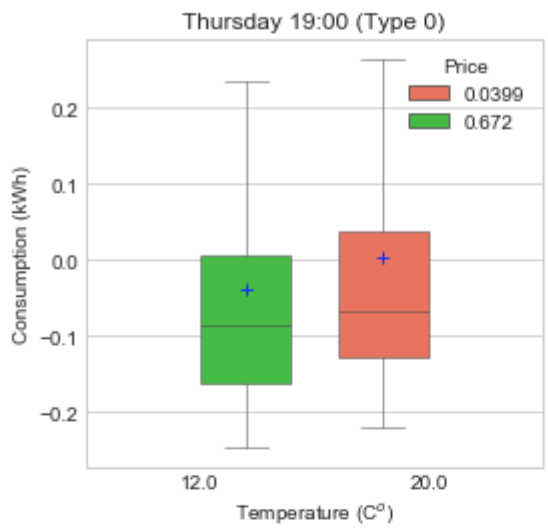

Thursday 17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
15.0
17.0
22.0
Temperature (C°)

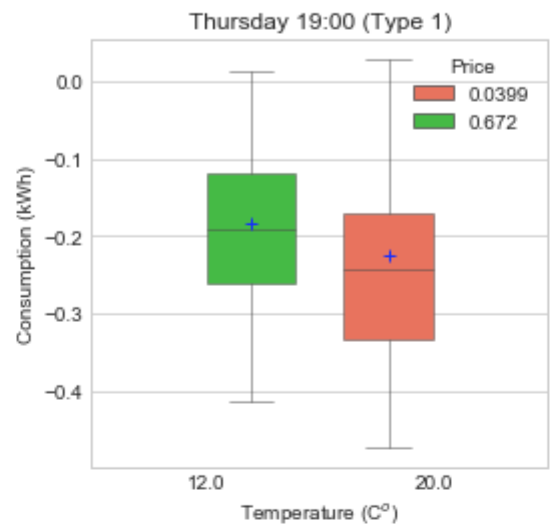

Thursday 17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
15.0
17.0
22.0
Temperature (C°)

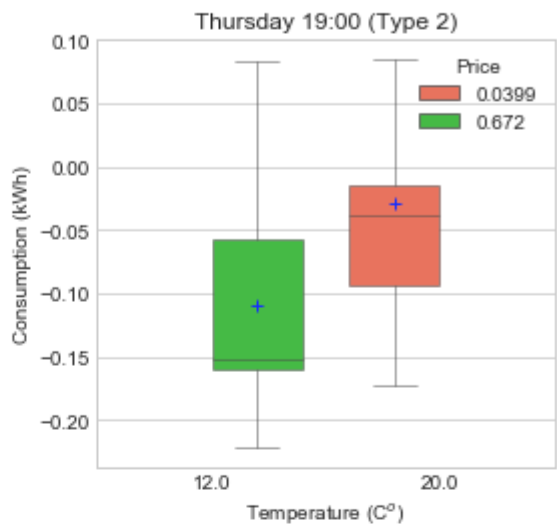

Thursday 18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
14.0
17.0
22.0
Temperature (C°)

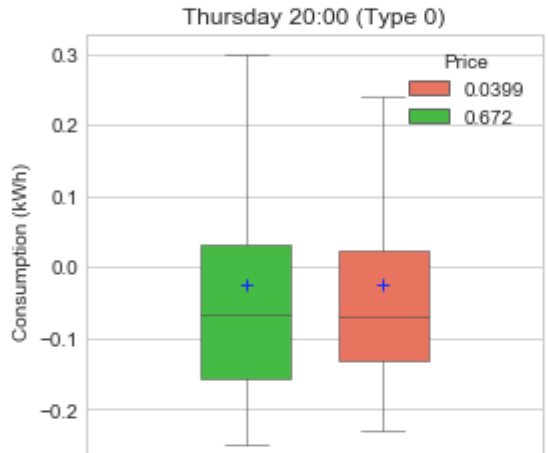

Thursday 18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
14.0
17.0
22.0
Temperature (C°)

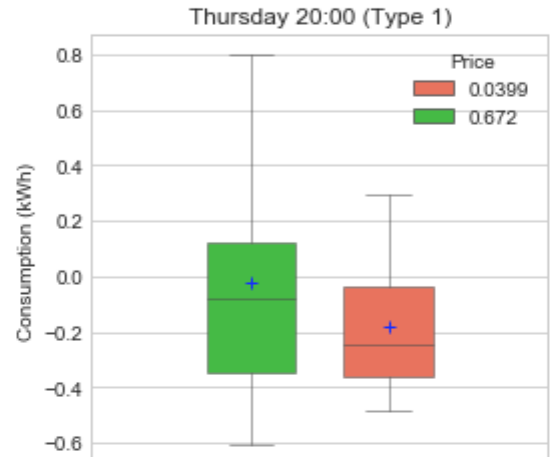

Thursday 18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.2
0.0
−0.2
14.0
17.0
22.0
Temperature (C°)

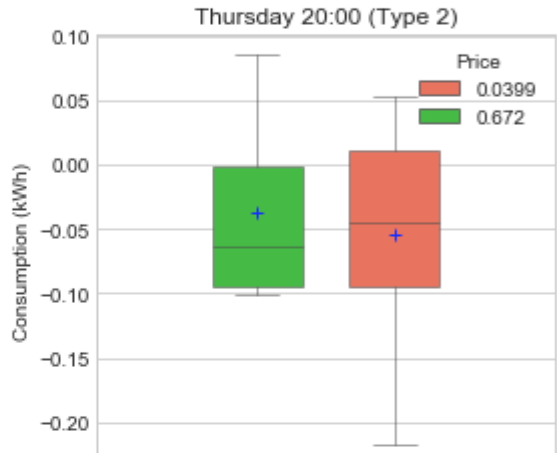

Thursday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
12.0
20.0
Temperature (C°)

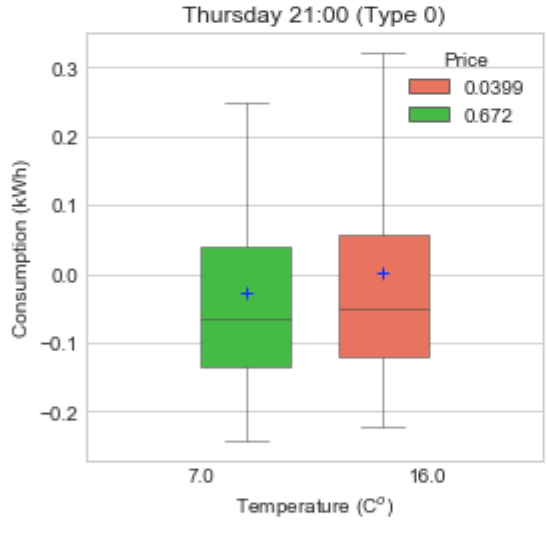

Thursday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.0
−0.1
−0.2
−0.3
−0.4
12.0
20.0
Temperature (C°)

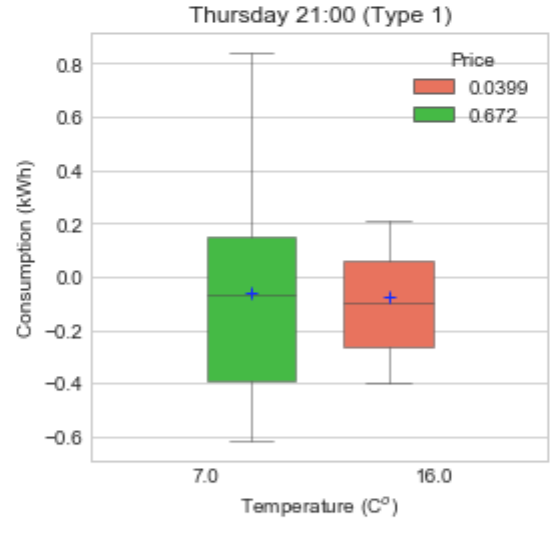

Thursday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
12.0
20.0
Temperature (C°)

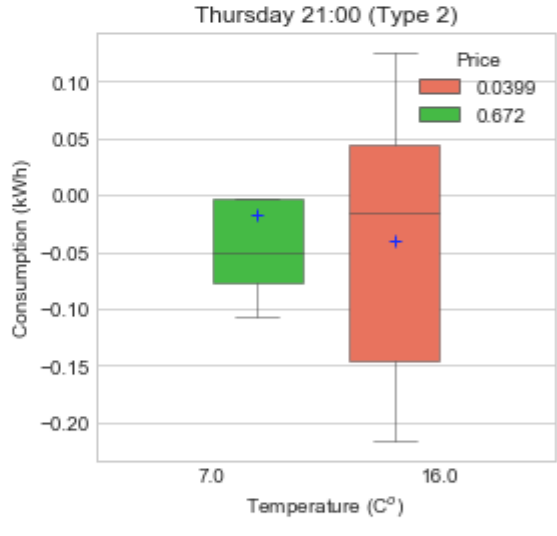

Thursday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
9.0
18.0
Temperature (C°)


Thursday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
9.0
18.0
Temperature (C°)


Thursday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
9.0
18.0
Temperature (C°)


Thursday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0
16.0
Temperature (C°)

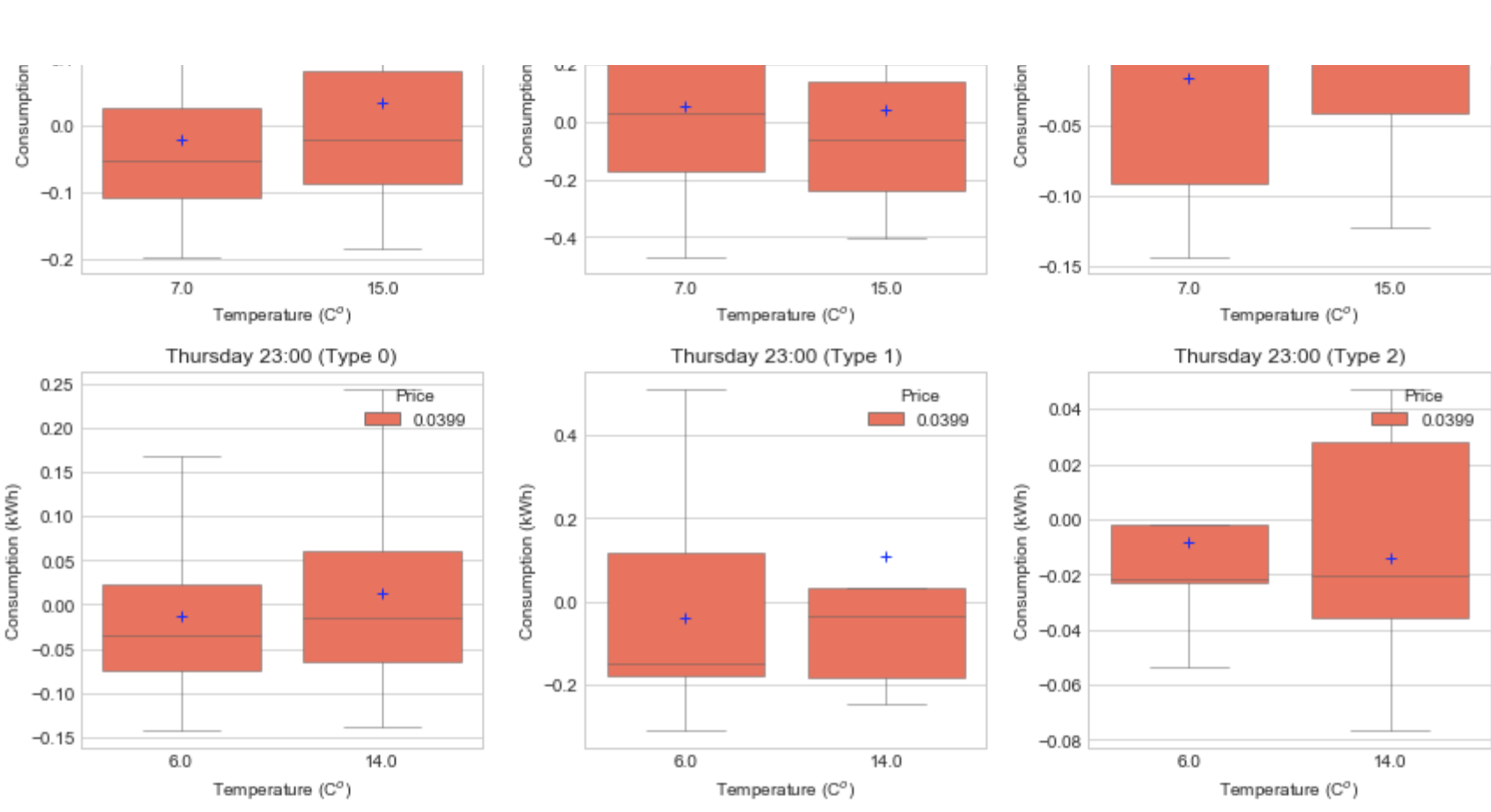

Thursday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
7.0
16.0
Temperature (C°)


Thursday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
7.0
16.0
Temperature (C°)

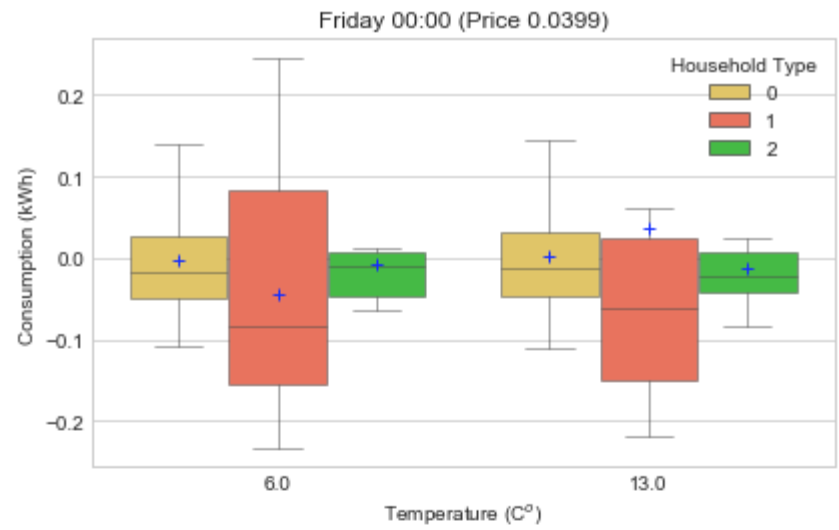
Thursday 22:00 (Type 0)
Price
0.0399
0.3
0.2
0.1

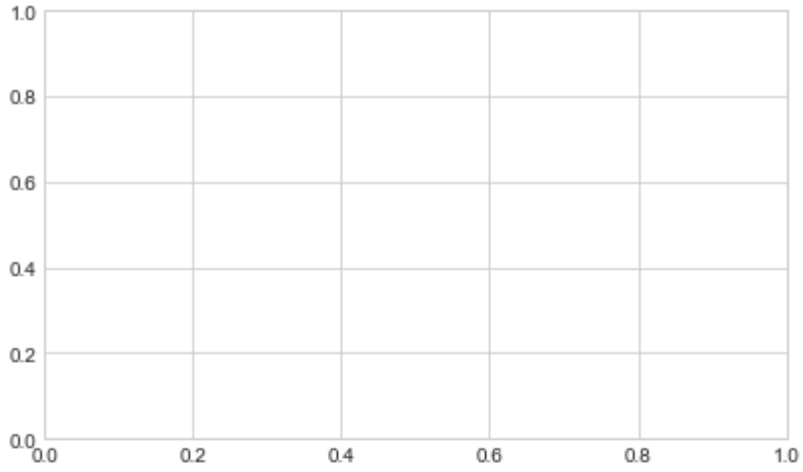
Thursday 22:00 (Type 1)
Price
0.0399
0.6
0.4

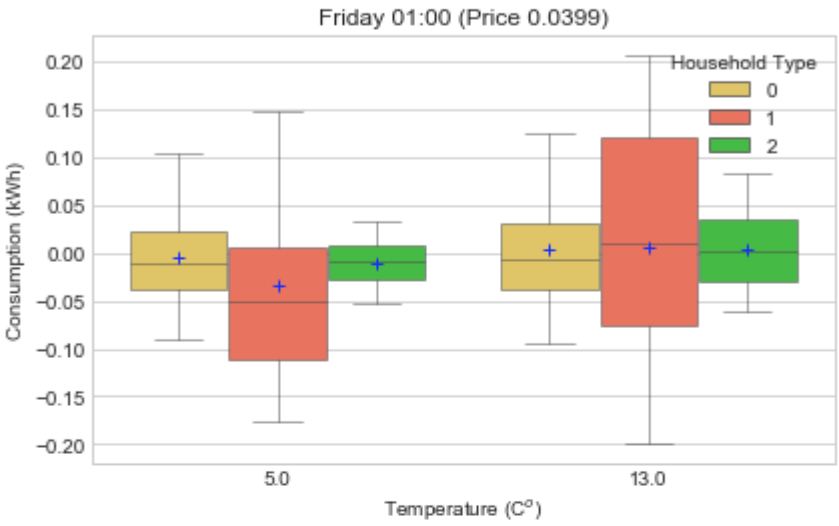
Thursday 22:00 (Type 2)
Price
0.0399
0.10
0.05
0.00

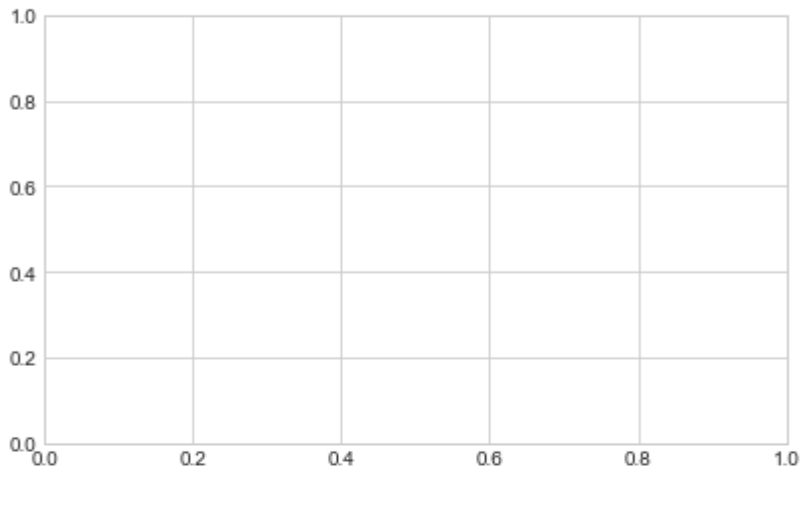

Thursday 23:00 (Type 0)
Thursday 23:00 (Type 1)
Thursday 23:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
6.0
14.0
7.0
15.0

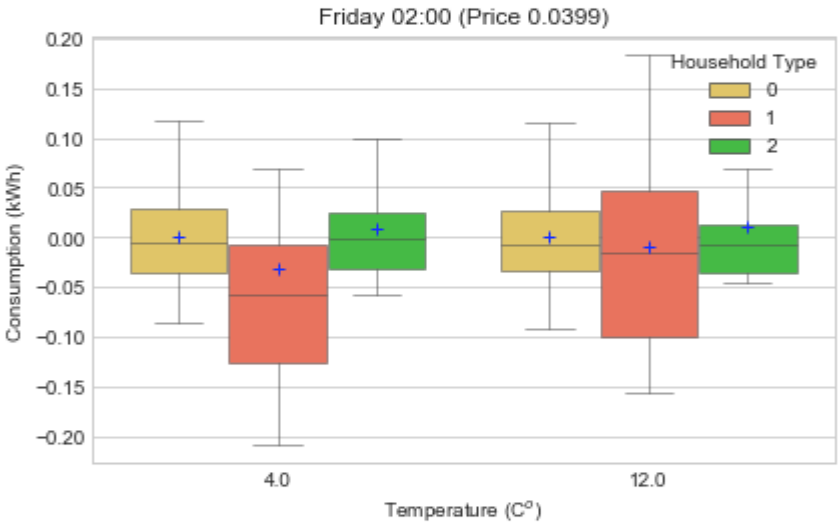
In [69]:
```
# Friday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 4 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers=False, showmeans=True, meanprops={"marker":"+","ma
rkerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers=False, showmeans=True, meanprops={"marker":"+","ma
rkerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

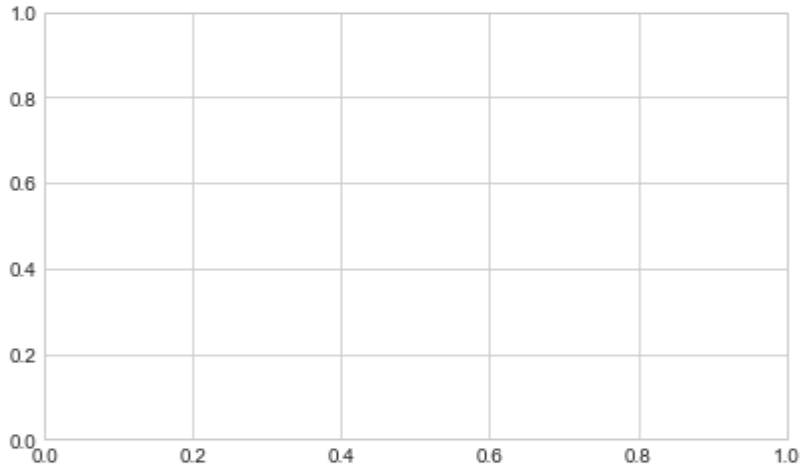

Friday 00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
6.0
13.0
Temperature (C°)

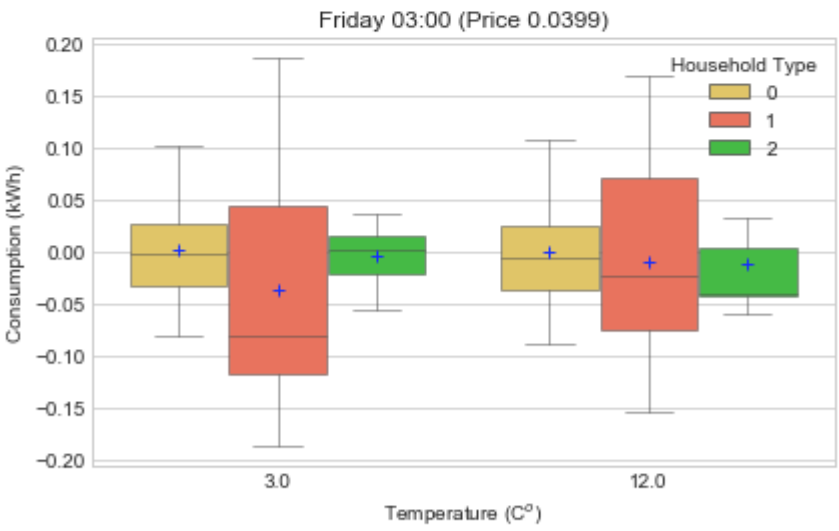

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

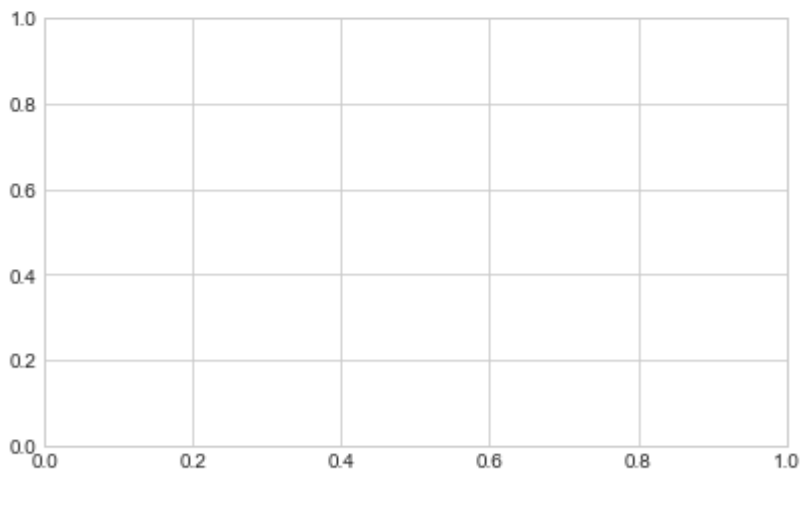

Friday 01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
5.0
13.0
Temperature (C°)

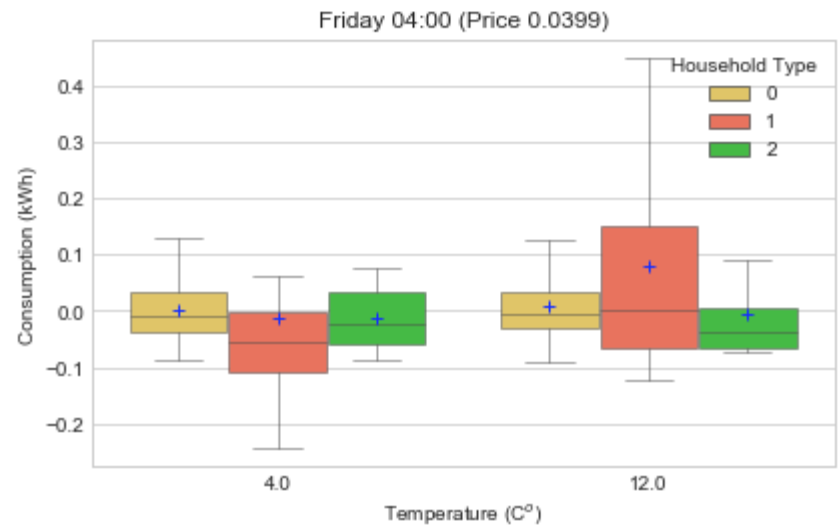

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

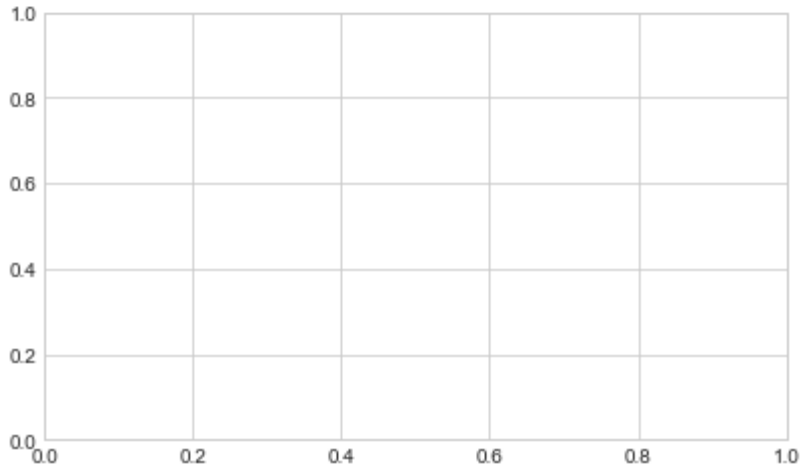

Friday 02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
4.0
12.0
Temperature (C°)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

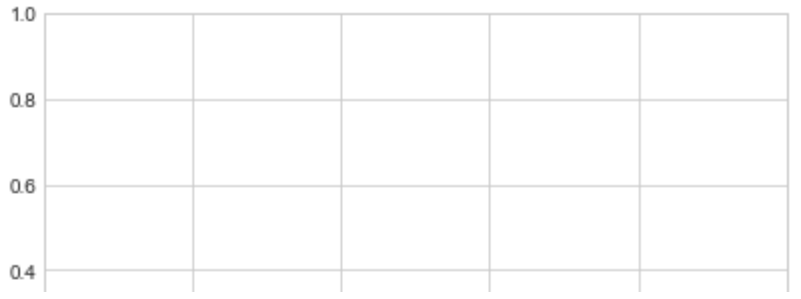

Friday 03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
3.0
12.0
Temperature (C°)

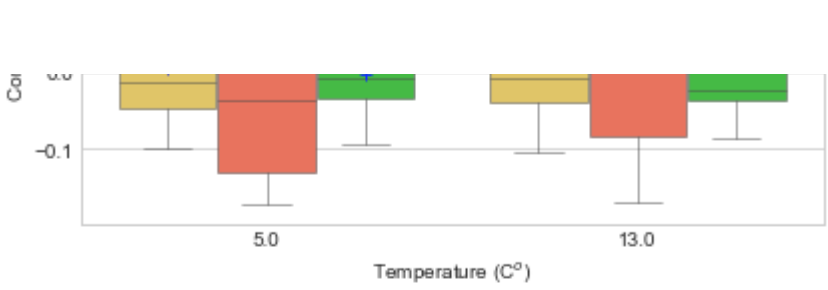

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

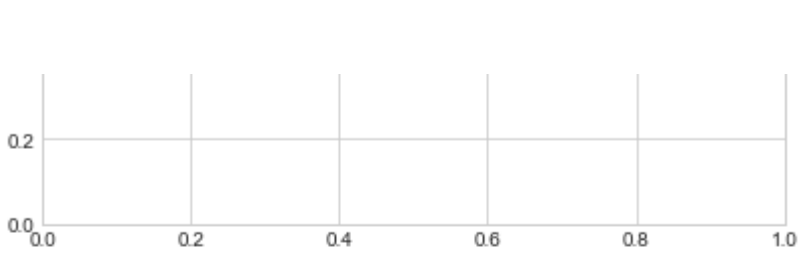

Friday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
12.0
Temperature (C°)

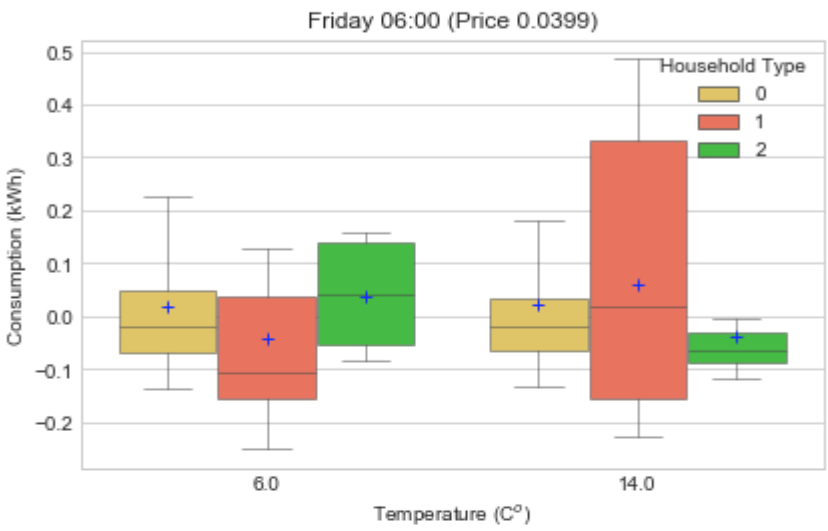

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

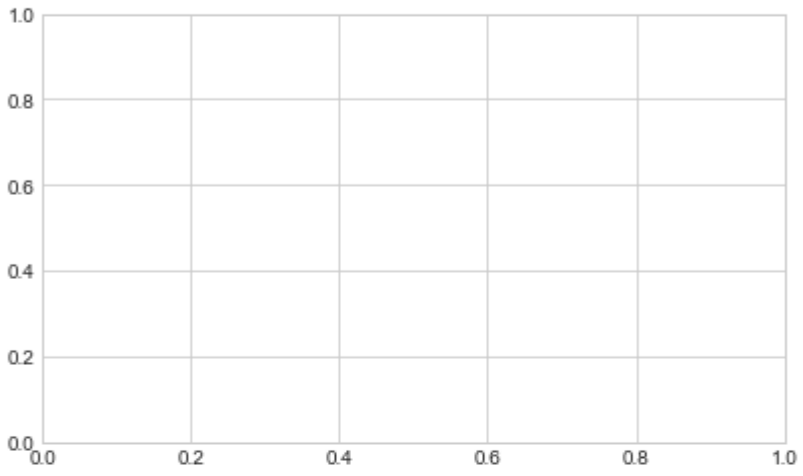

Friday 05:00 (Price 0.0399)
Household Type
0
1
2
0.3
0.2
0.1
0.0

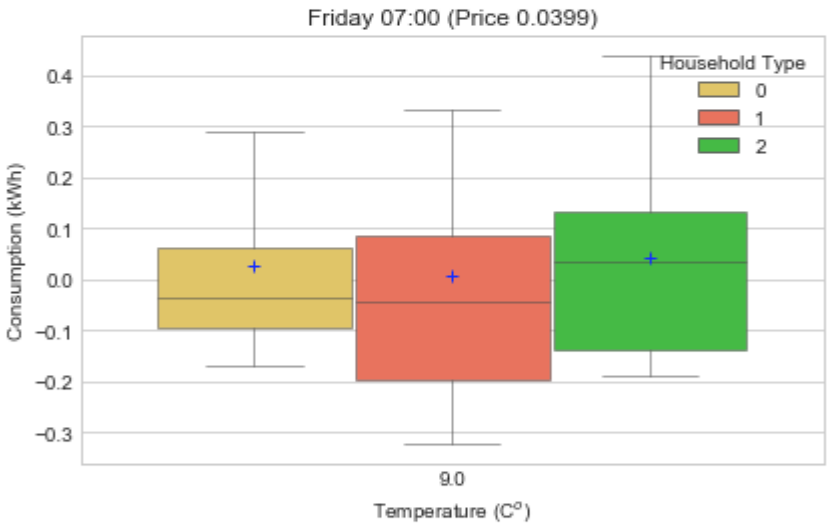

1.0
0.8
0.6
0.4

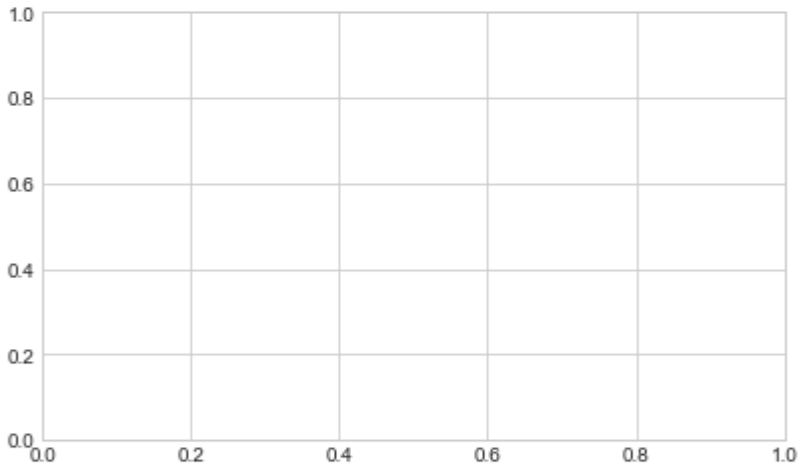

−0.1
5.0
13.0
Temperature (C°)

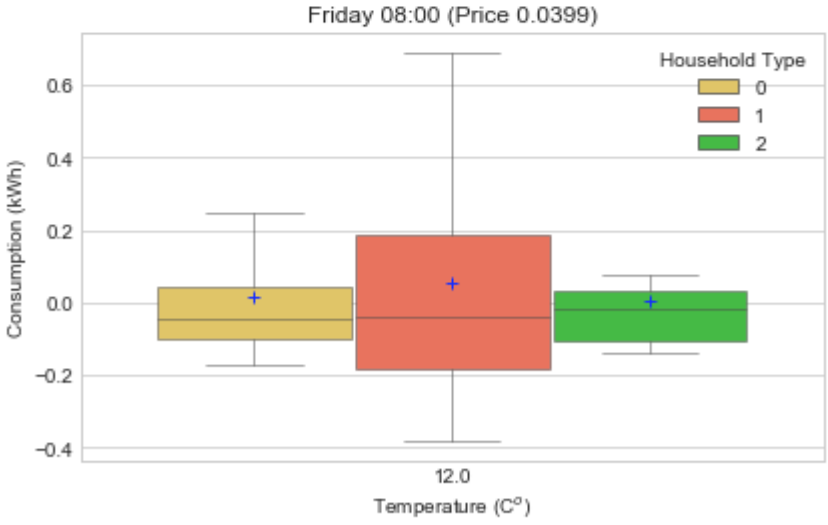

0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

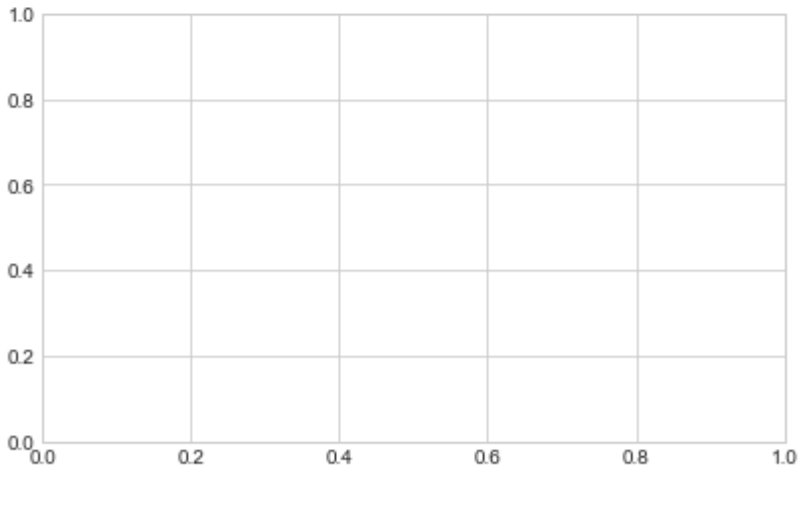

Friday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
6.0
14.0
Temperature (C°)

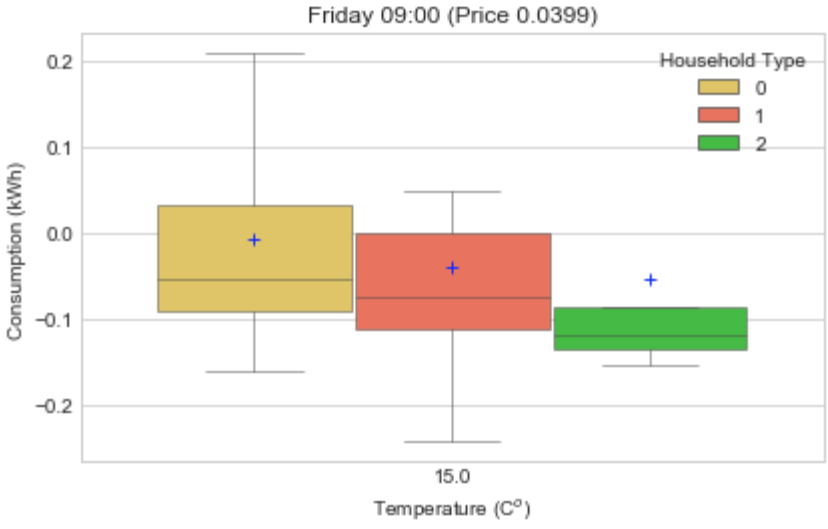

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

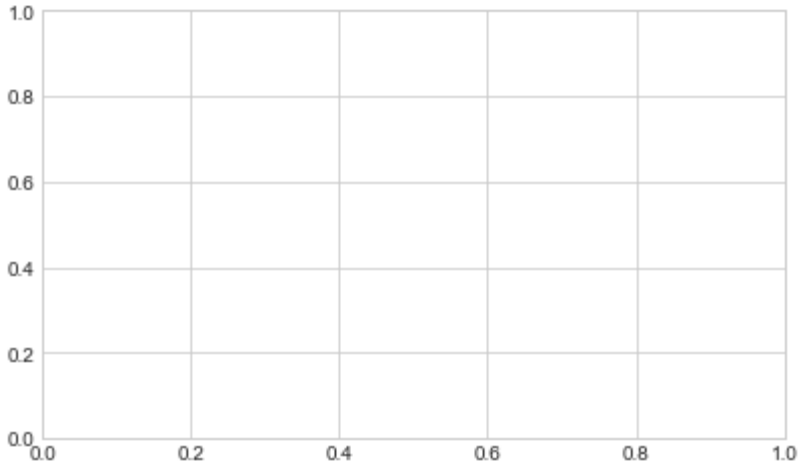

Friday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
9.0
Temperature (C°)

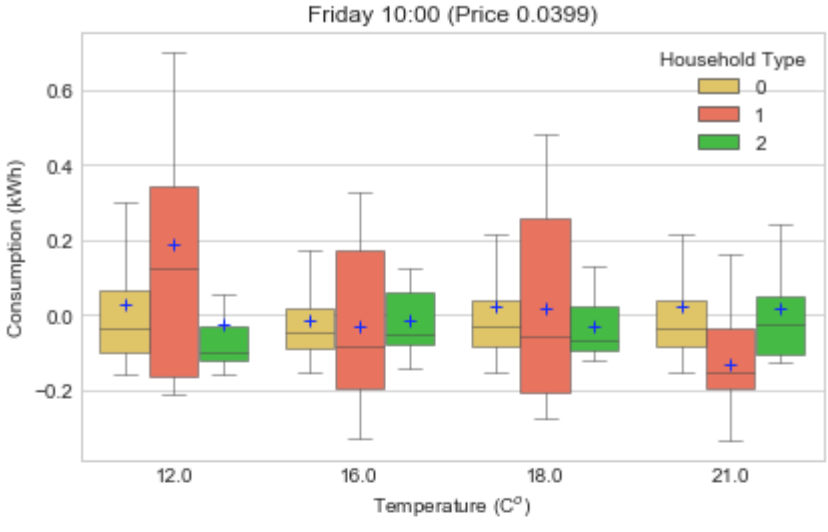

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

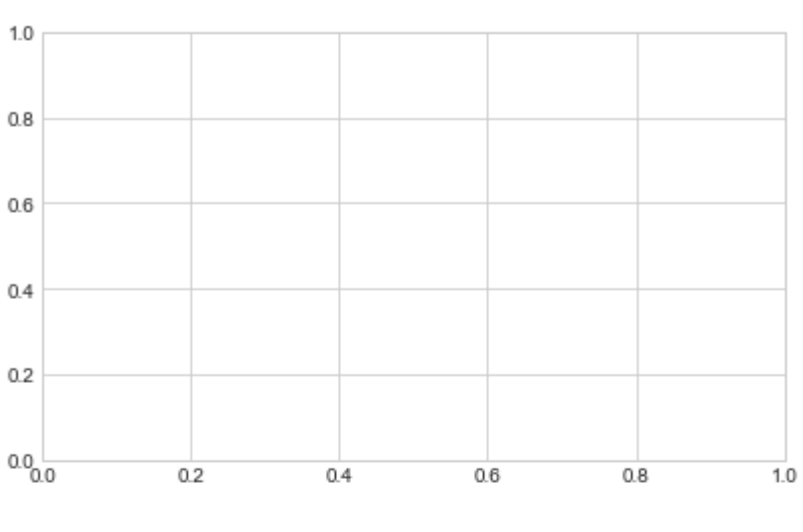

Friday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
12.0
Temperature (C°)

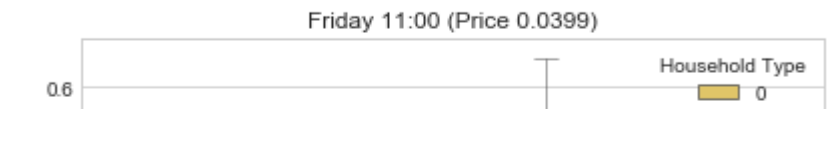

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Friday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
15.0
Temperature (C°)

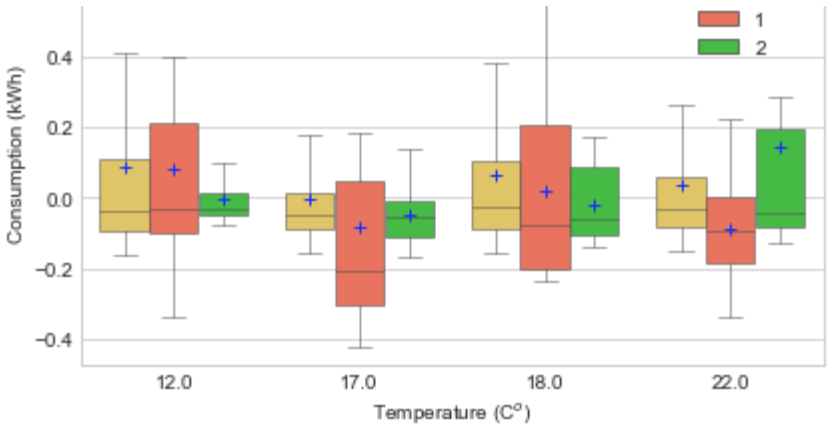

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

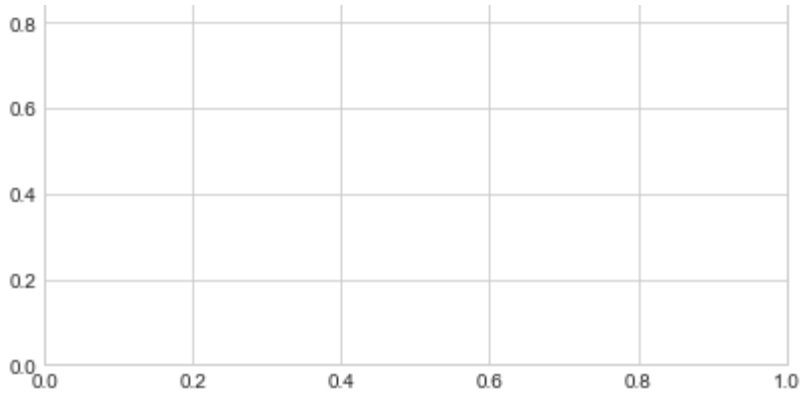

Friday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
12.0
16.0
18.0
21.0
Temperature (C°)

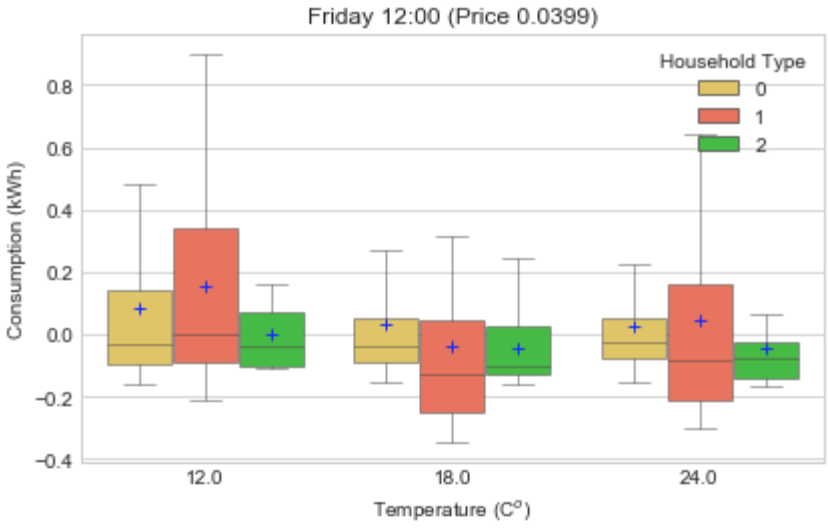

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

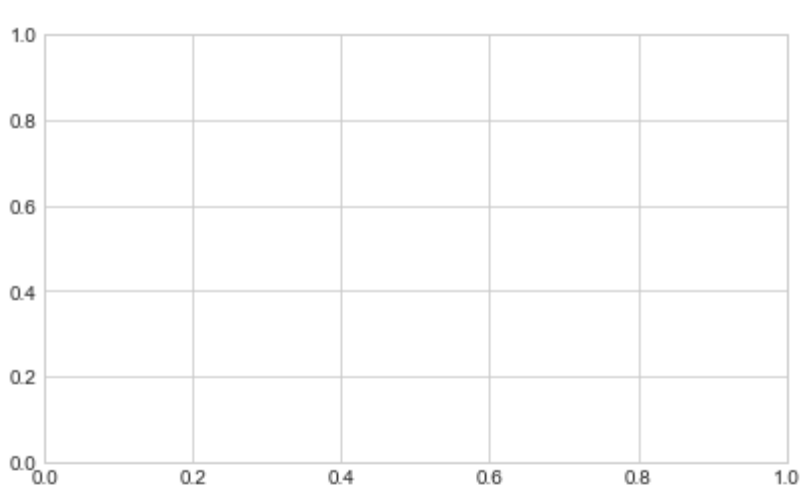

Friday 11:00 (Price 0.0399)
Household Type
0
0.6

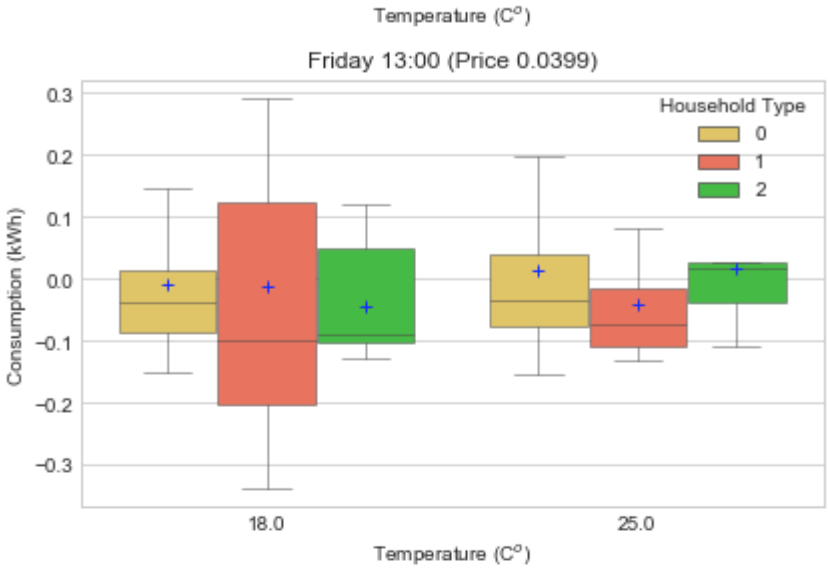
1.0

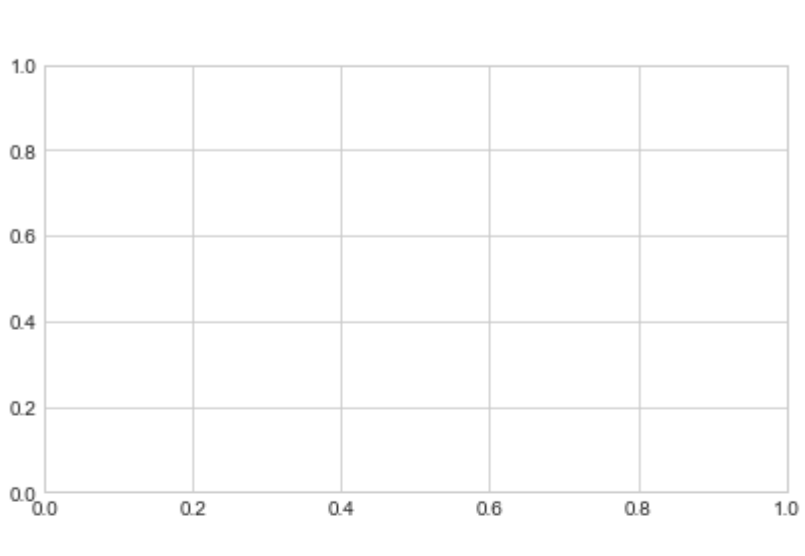

1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
12.0
17.0
18.0
22.0
Temperature (C°)

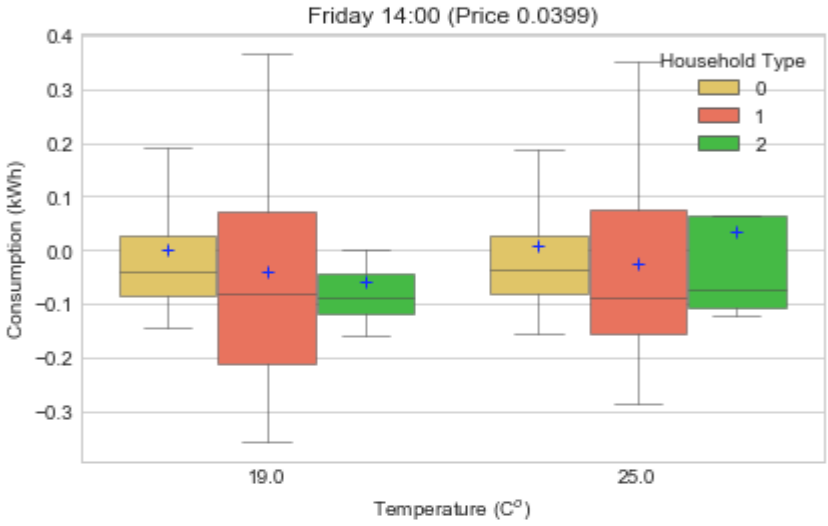

0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

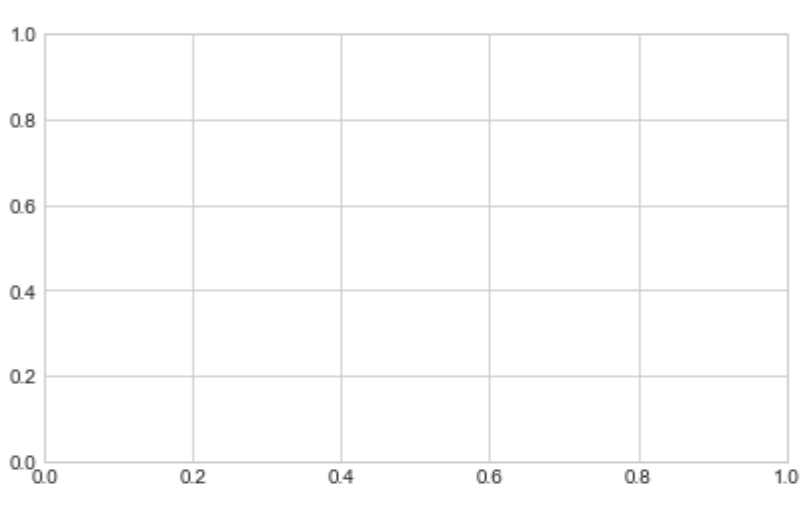

Friday 12:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
12.0
18.0
24.0
Temperature (C°)

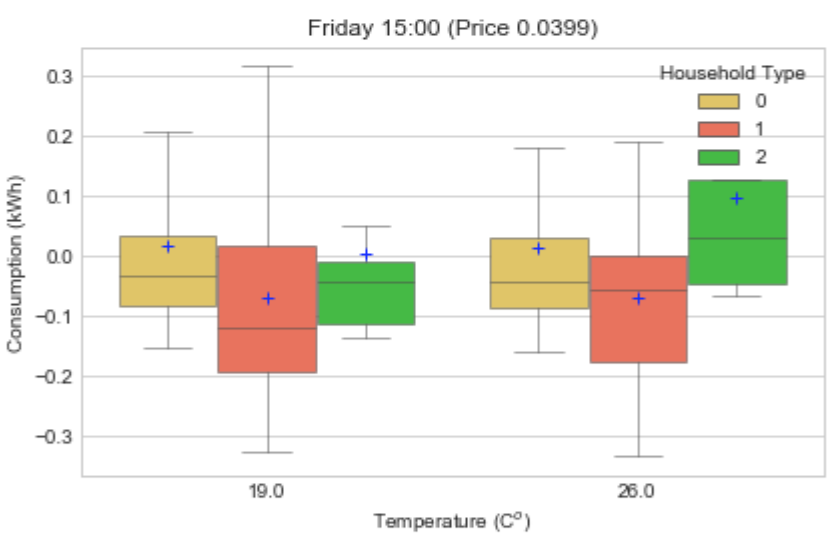

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

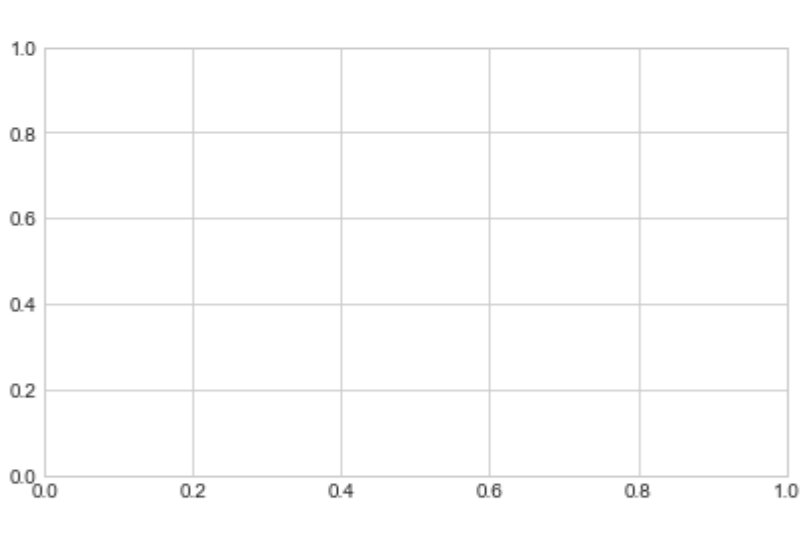

Friday 13:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
18.0
25.0
Temperature (C°)

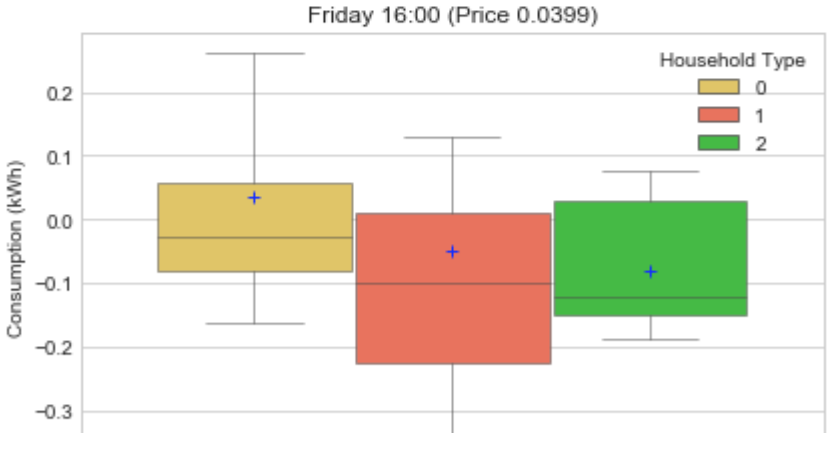

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

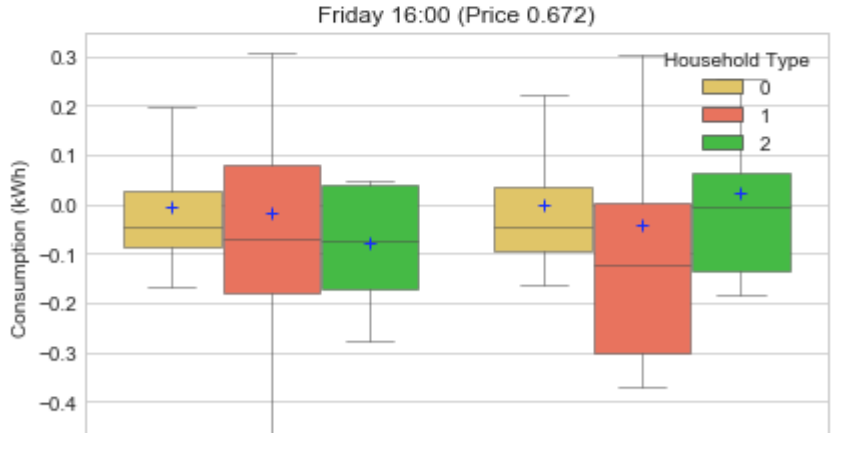

Friday 14:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
19.0
25.0
Temperature (C°)

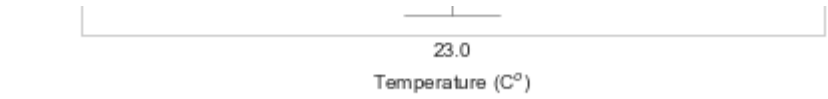

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Friday 15:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
19.0
26.0
Temperature (C°)

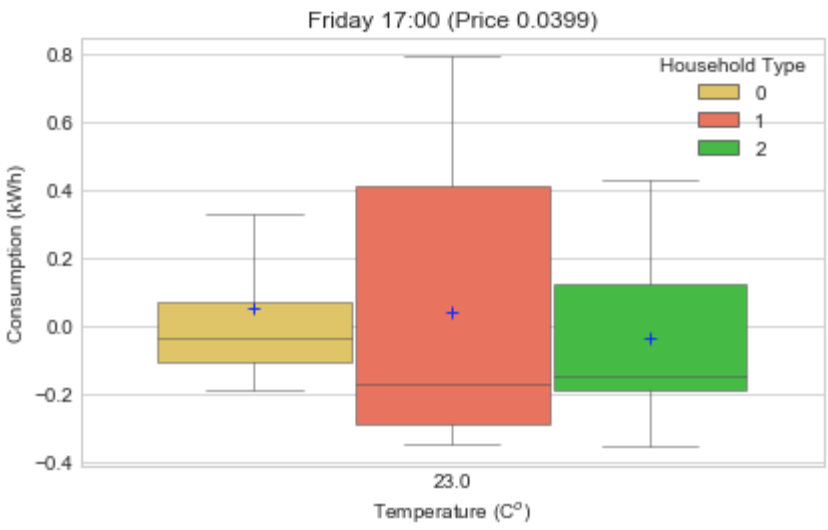

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

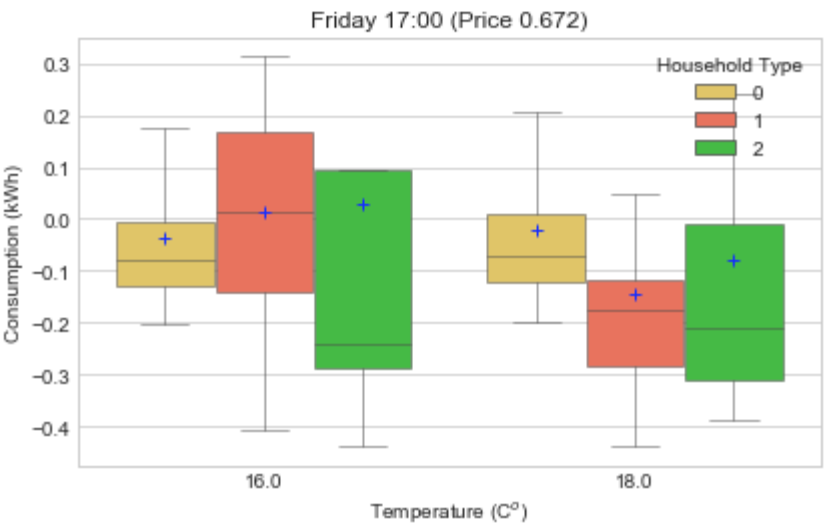

Friday 16:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3

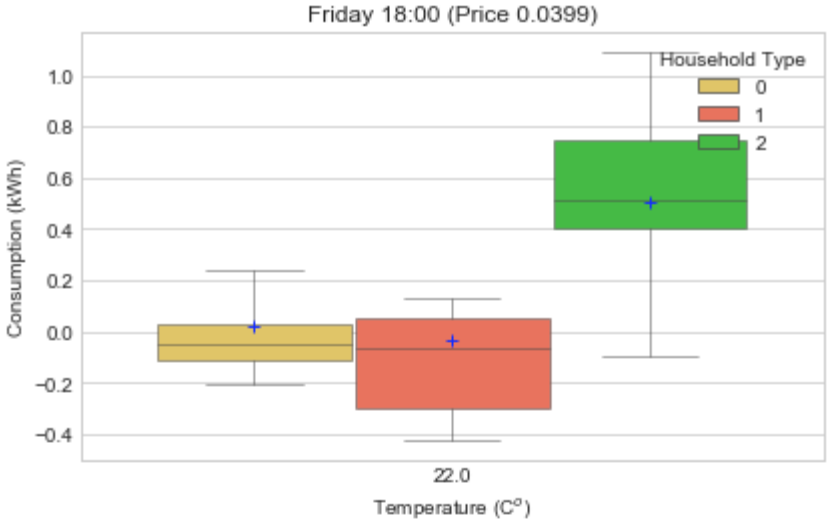

Friday 16:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4

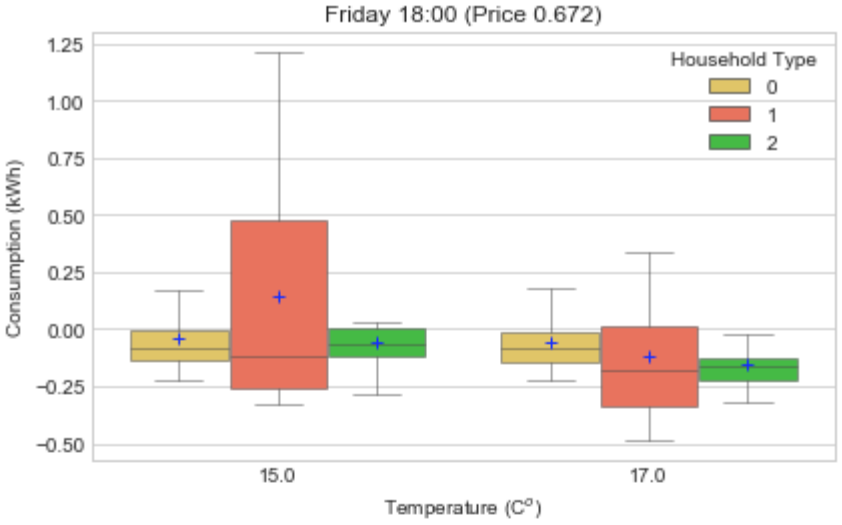

23.0
Temperature (C°)

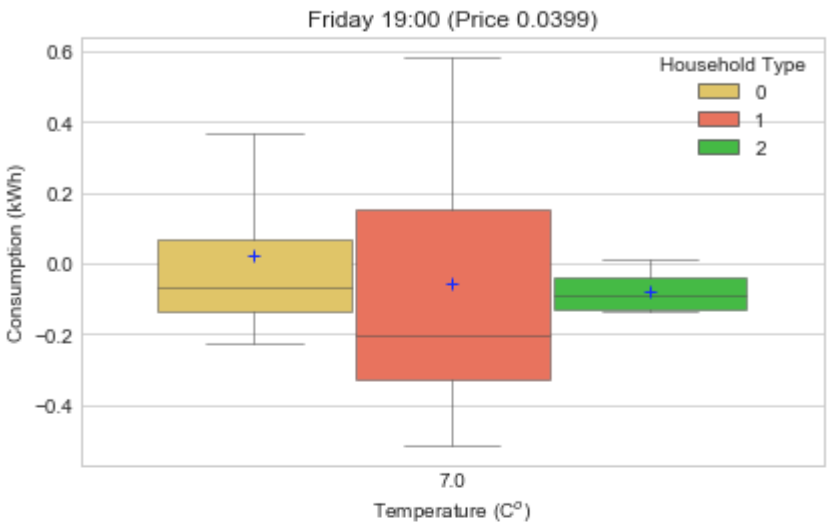

−0.5
18.0
20.0
Temperature (C°)

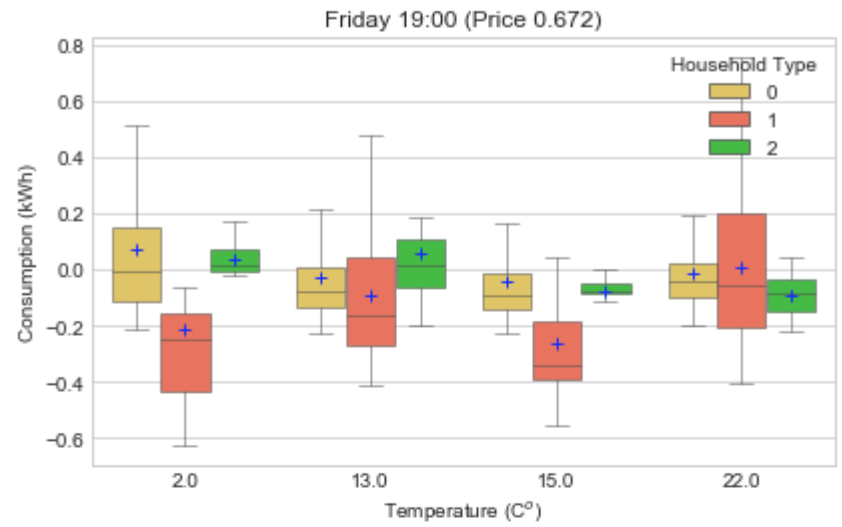

Friday 17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
23.0
Temperature (C°)

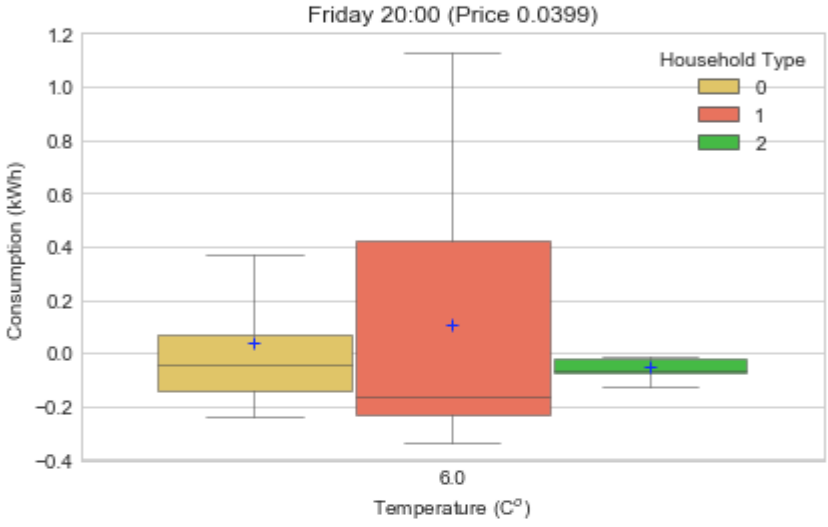

Friday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
16.0
18.0
Temperature (C°)

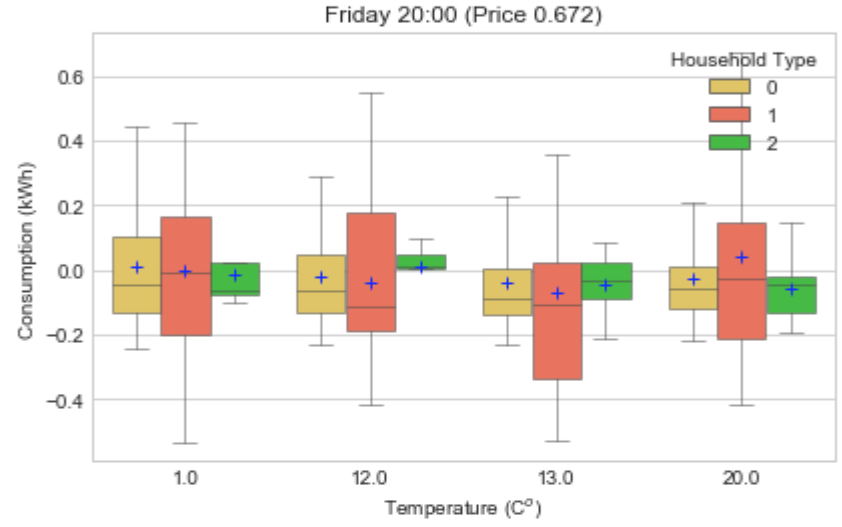

Friday 18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
22.0
Temperature (C°)

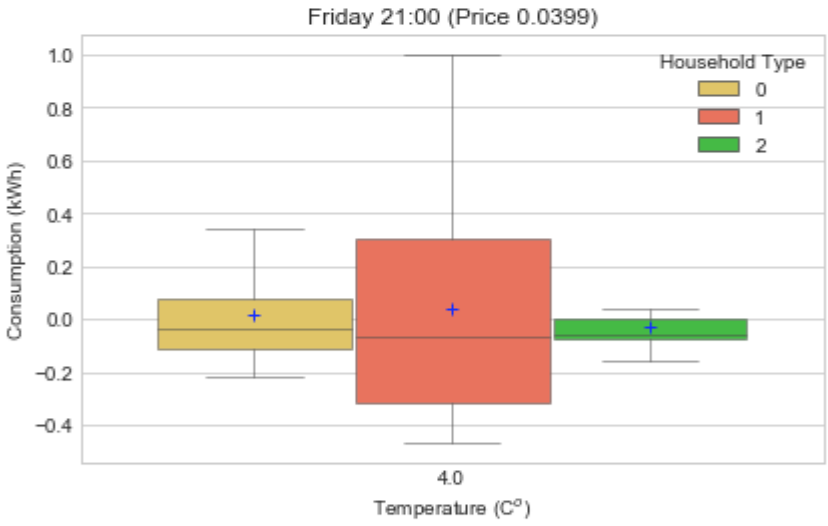

Friday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
15.0
17.0
Temperature (C°)

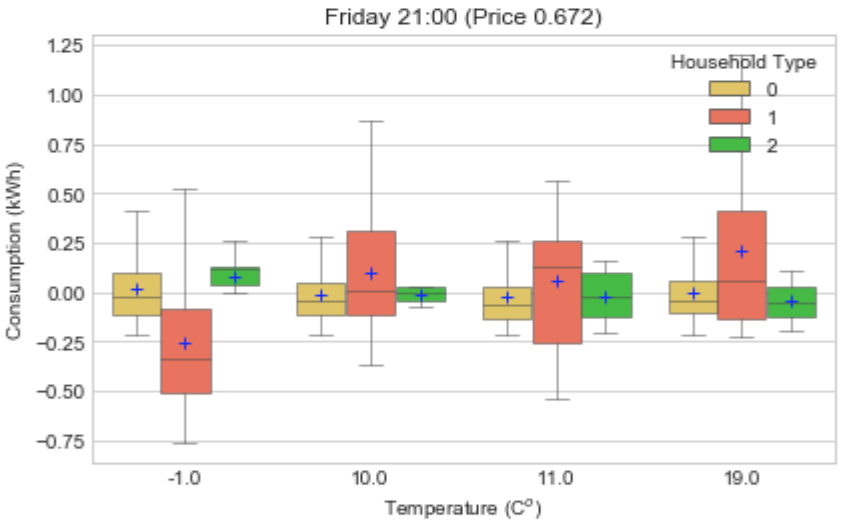

Friday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
7.0
Temperature (C°)


Friday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
2.0
13.0
15.0
22.0
Temperature (C°)


Friday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
6.0
Temperature (C°)

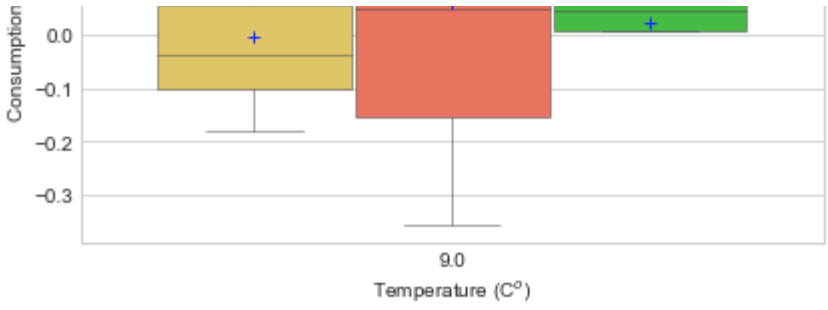

Friday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
1.0
12.0
13.0
20.0
Temperature (C°)

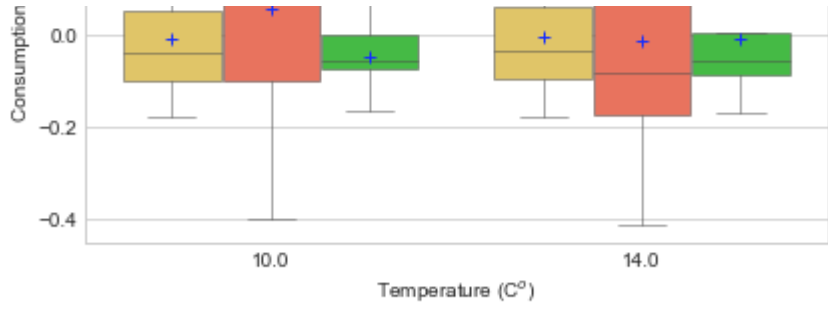

Friday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
4.0
Temperature (C°)

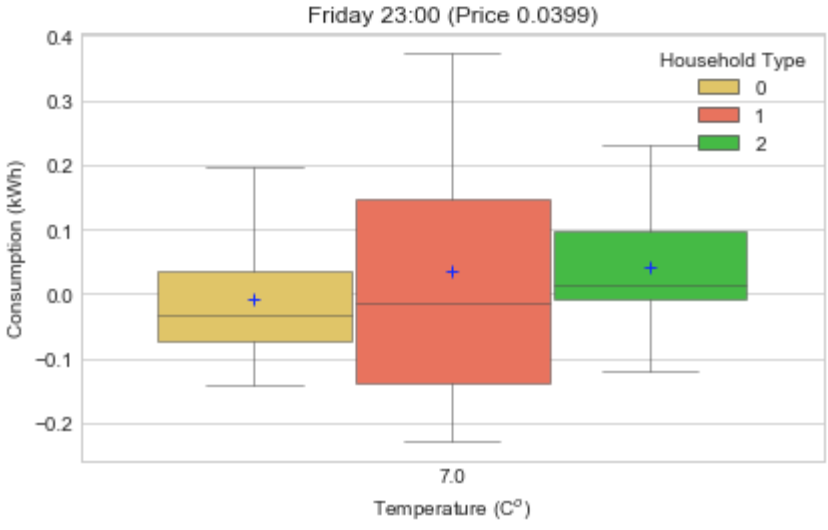

Friday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
-1.0
10.0
11.0
19.0
Temperature (C°)

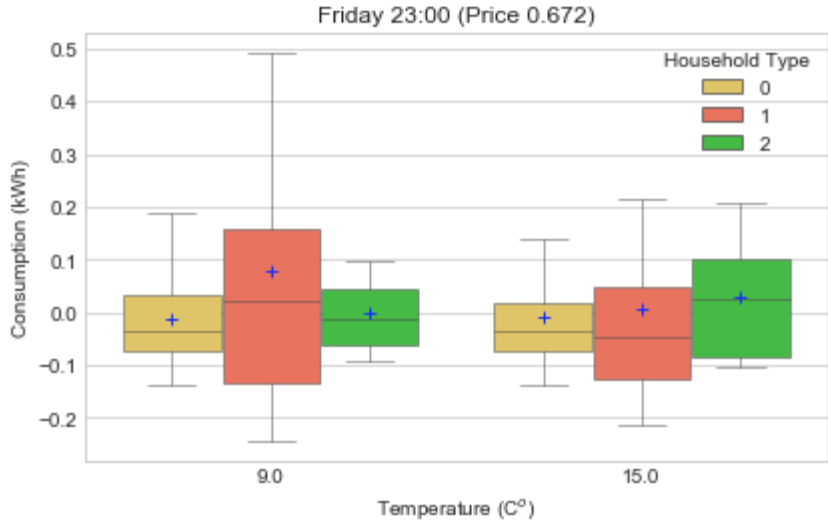
Friday 22:00 (Price 0.0399)
Household Type
0
1
2
0.4
0.3
0.2
0.1

Friday 22:00 (Price 0.672)
Household Type
0
1
2
0.4
0.2

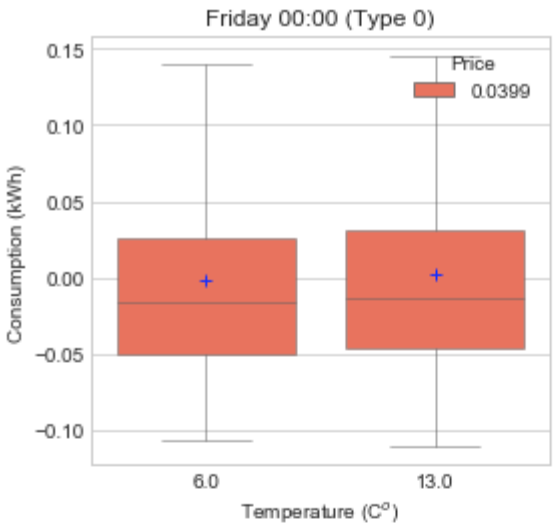

Consumption
0.0
−0.1
−0.2
−0.3
9.0
Temperature (C°)

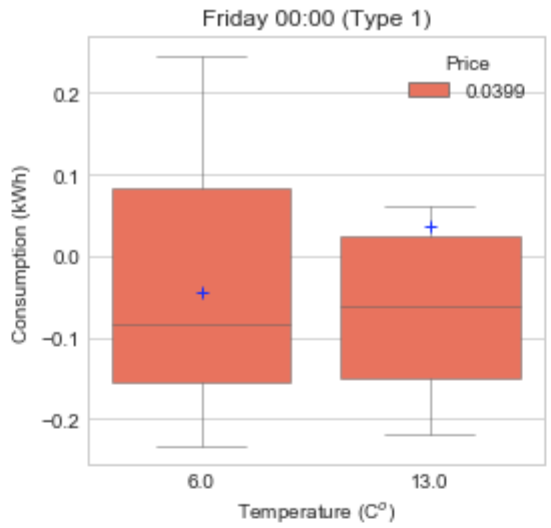

Consumption
0.0
−0.2
−0.4
10.0
14.0
Temperature (C°)

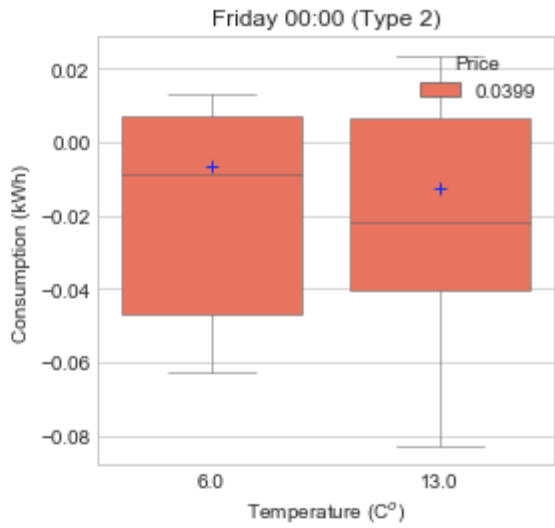

Friday 23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0
Temperature (C°)

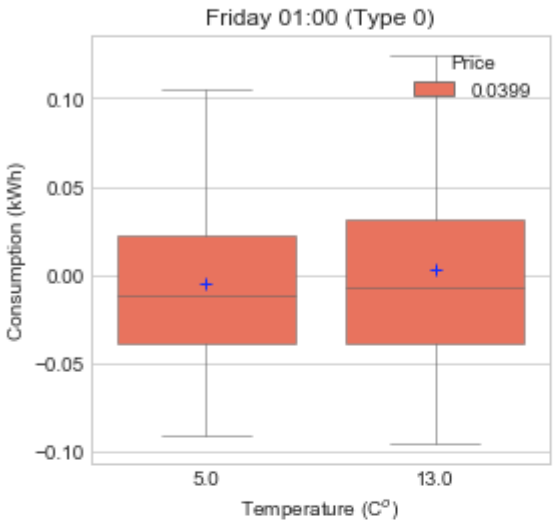

Friday 23:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
9.0
15.0
Temperature (C°)

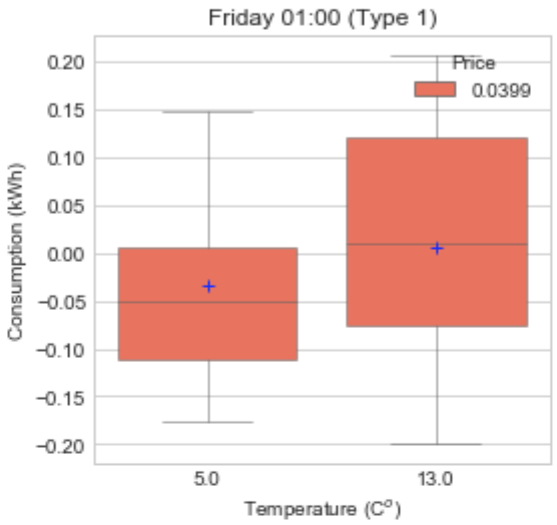
In [70]:
```
# price comparison
# Friday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 4 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers=False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:v
ivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers=False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:v
ivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

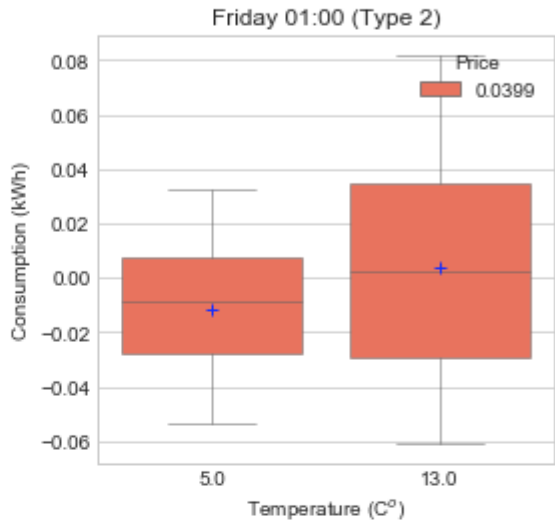

Friday 00:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
6.0
13.0
Temperature (C°)

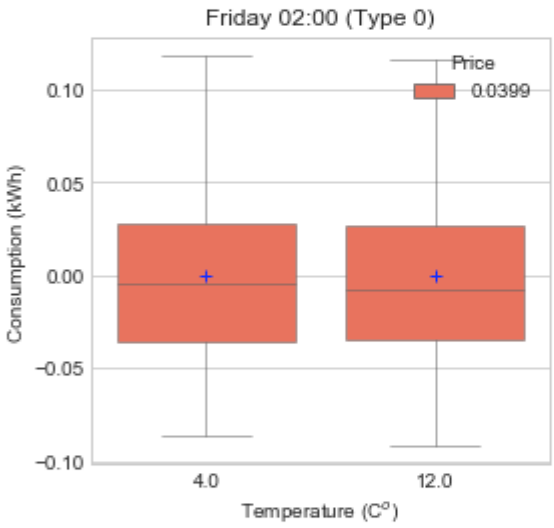

Friday 00:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
6.0
13.0
Temperature (C°)

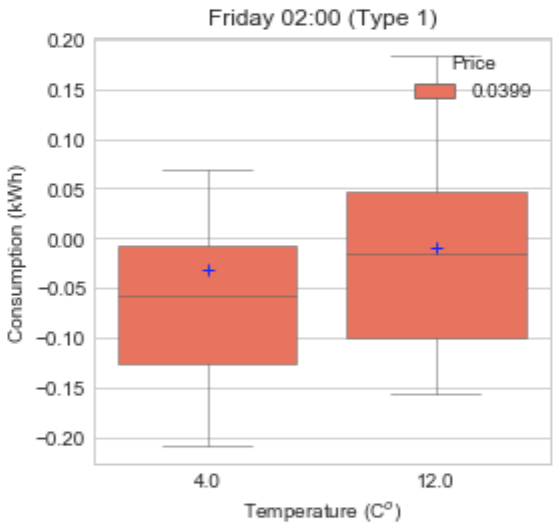

Friday 00:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.02
0.00
−0.02
−0.04
−0.06
−0.08
6.0
13.0
Temperature (C°)

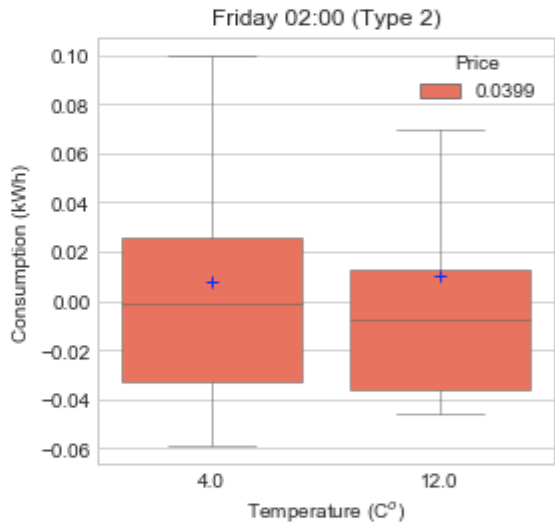

Friday 01:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
5.0
13.0
Temperature (C°)

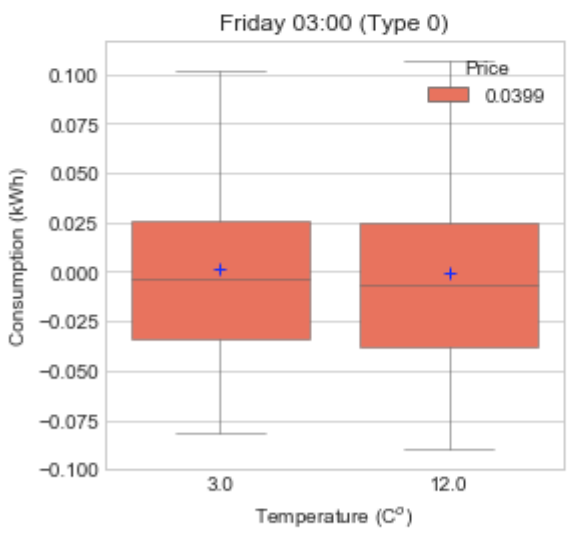

Friday 01:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
5.0
13.0
Temperature (C°)

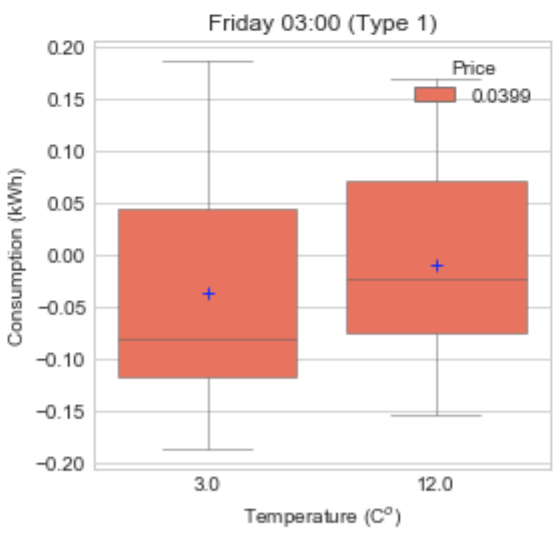

Friday 01:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
5.0
13.0
Temperature (C°)

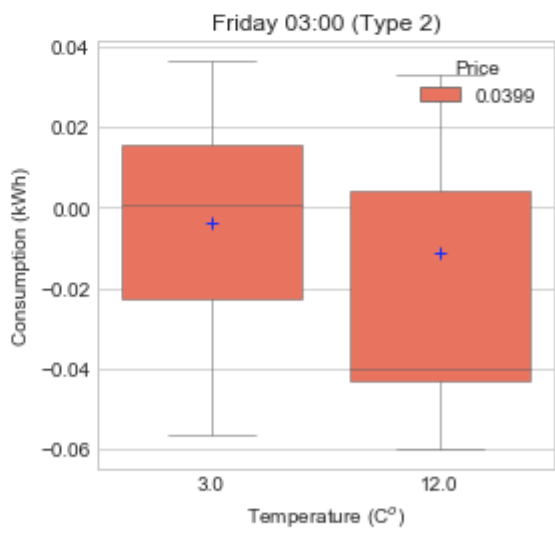

Friday 02:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
4.0
12.0
Temperature (C°)

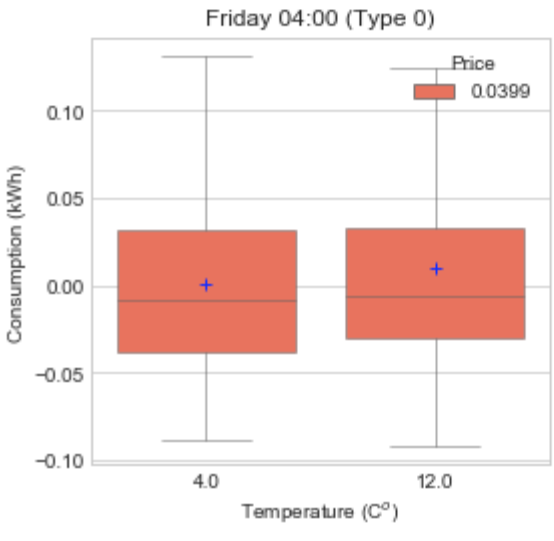

Friday 02:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
4.0
12.0
Temperature (C°)

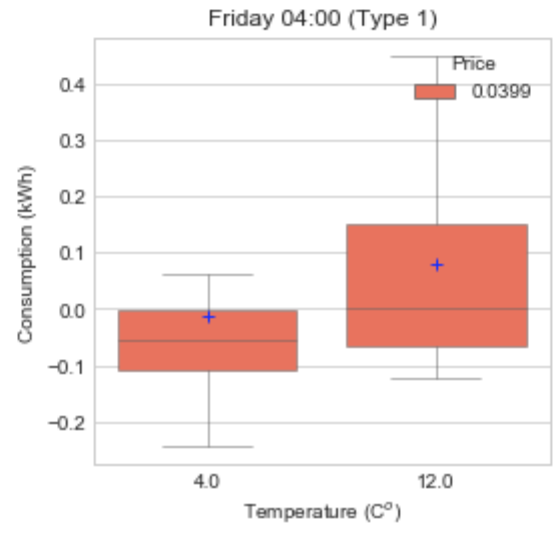

Friday 02:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
4.0
12.0
Temperature (C°)

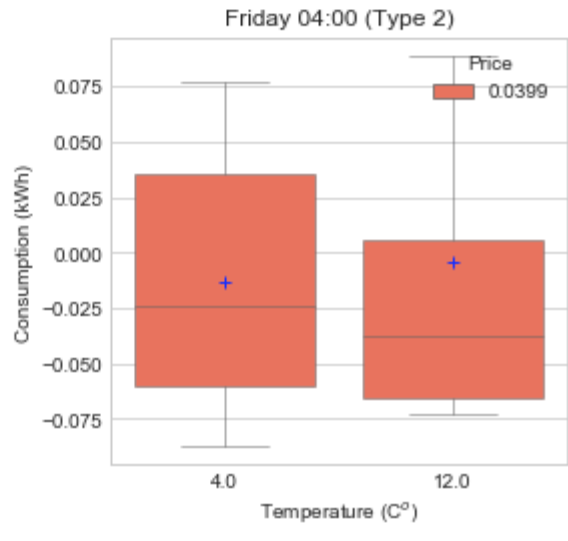

Friday 03:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
3.0
12.0
Temperature (C°)

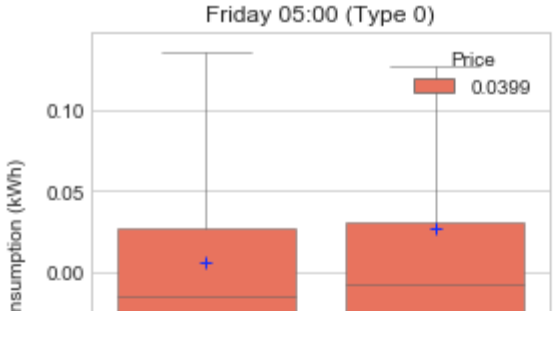

Friday 03:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
3.0
12.0
Temperature (C°)

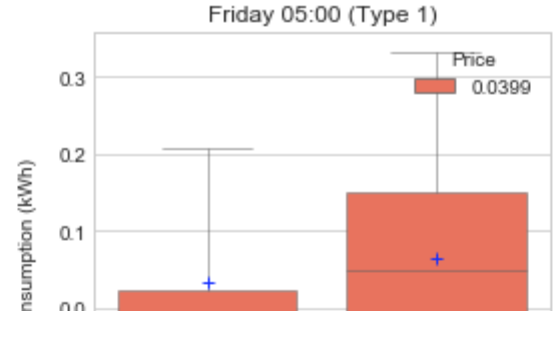

Friday 03:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.04
0.02
0.00
−0.02
−0.04
−0.06
3.0
12.0
Temperature (C°)

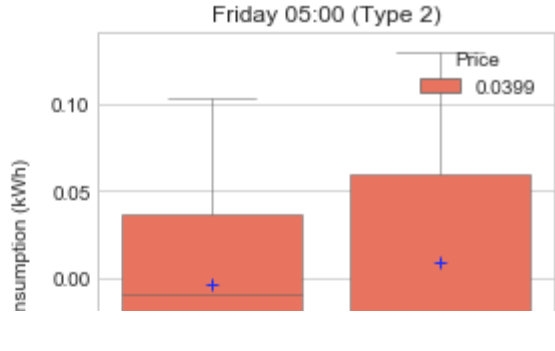

Friday 04:00 (Type 0)
Price
0.0399
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
4.0
12.0
Temperature (C°)

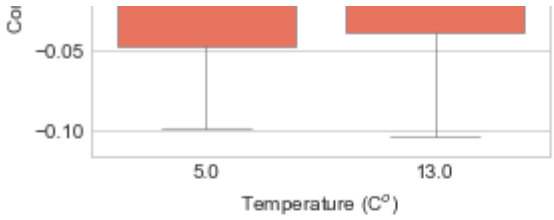

Friday 04:00 (Type 1)
Price
0.0399
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
12.0
Temperature (C°)

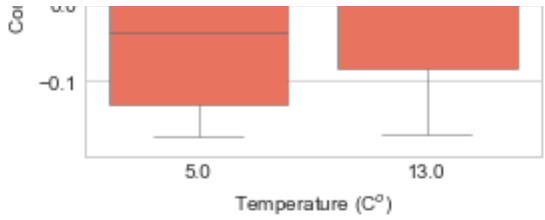

Friday 04:00 (Type 2)
Price
0.0399
Consumption (kWh)
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
4.0
12.0
Temperature (C°)

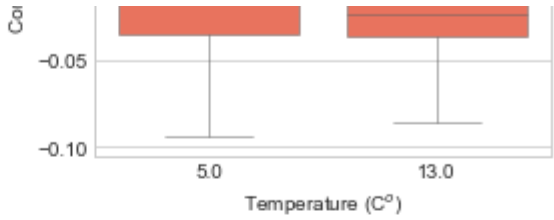

Friday 05:00 (Type 0)
Price
0.0399
0.10
0.05
0.00

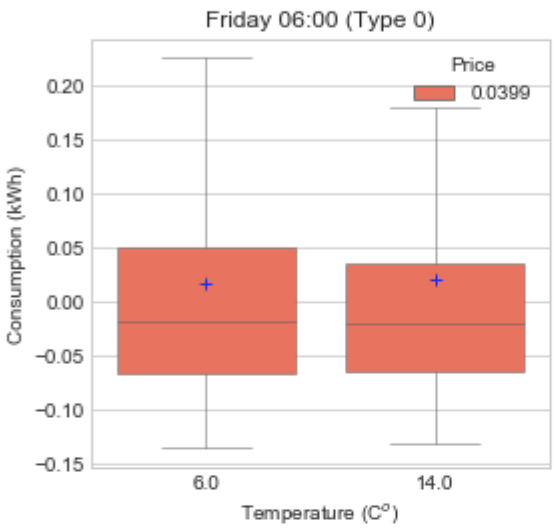

Friday 05:00 (Type 1)
Price
0.0399
0.3
0.2
0.1

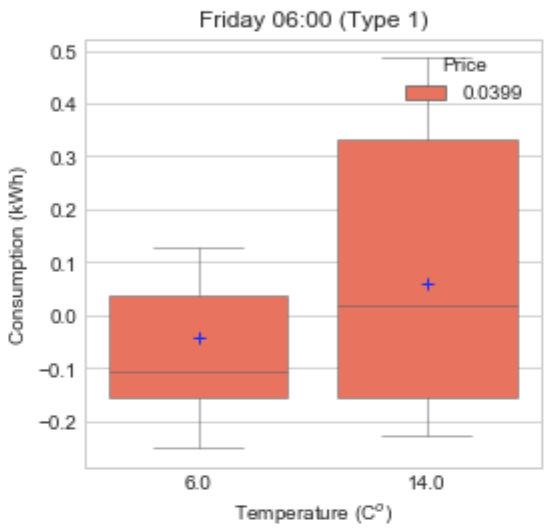

Friday 05:00 (Type 2)
Price
0.0399
0.10
0.05
0.00

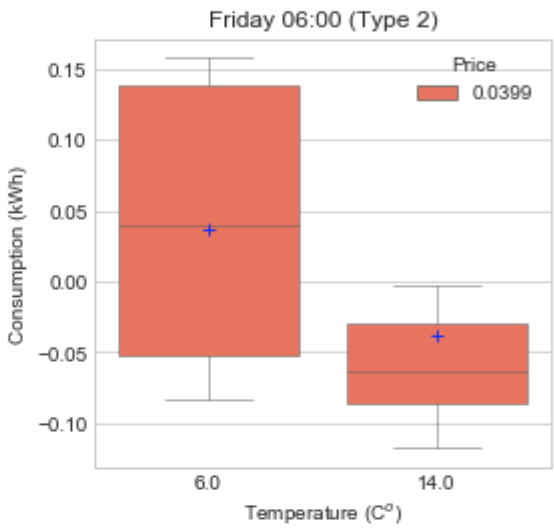

Price
0.0399
5.0
13.0
Temperature (C°)

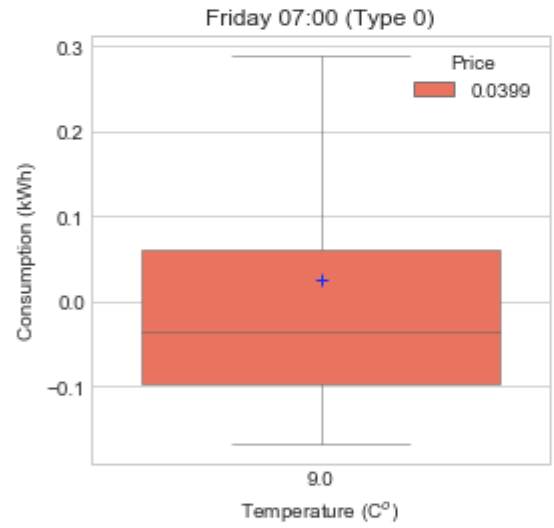

Price
0.0399
5.0
13.0
Temperature (C°)

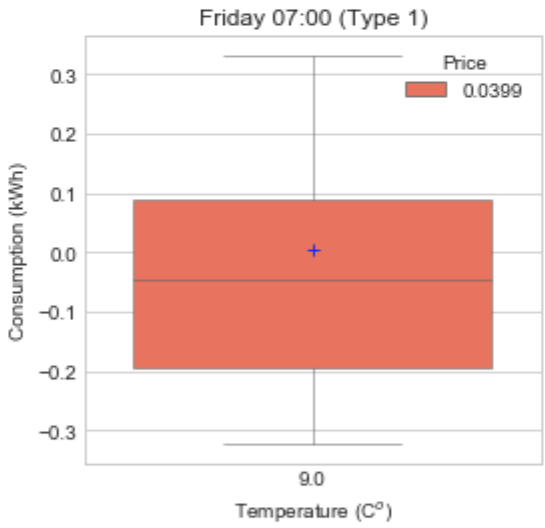

Price
0.0399
5.0
13.0
Temperature (C°)

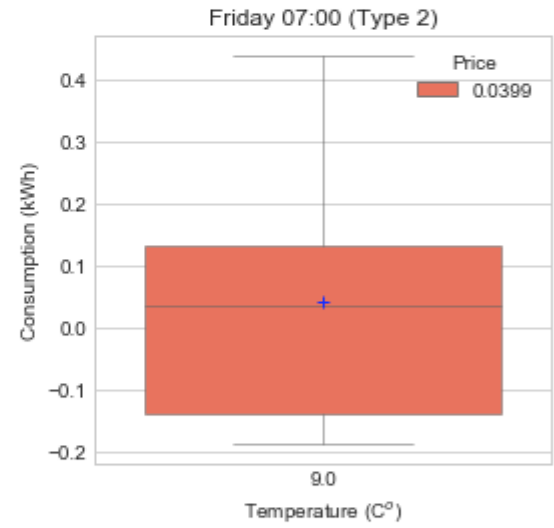

Friday 06:00 (Type 0)
Price
0.0399
Consumption (kWh)
6.0
14.0
Temperature (C°)

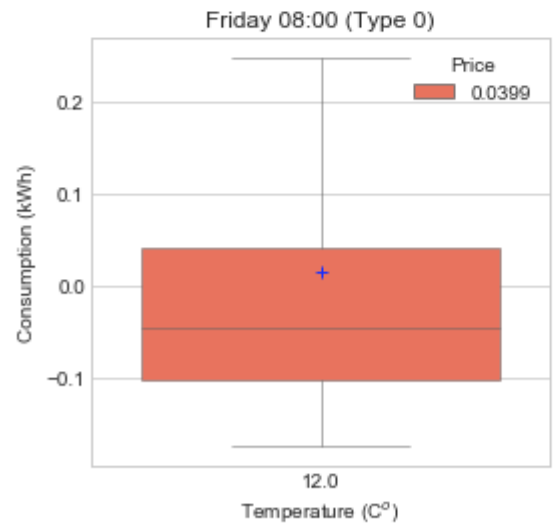

Friday 06:00 (Type 1)
Price
0.0399
Consumption (kWh)
6.0
14.0
Temperature (C°)

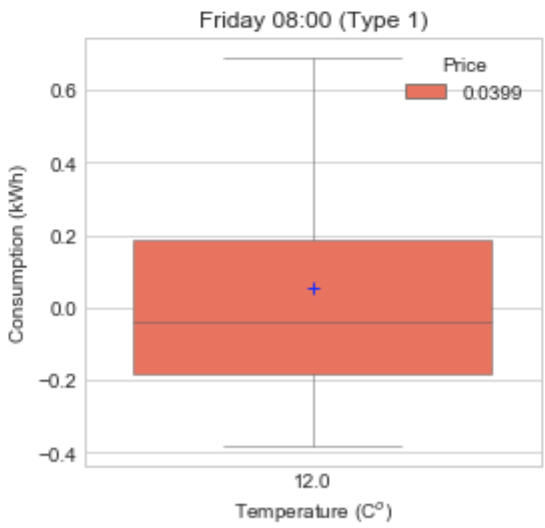

Friday 06:00 (Type 2)
Price
0.0399
Consumption (kWh)
6.0
14.0
Temperature (C°)

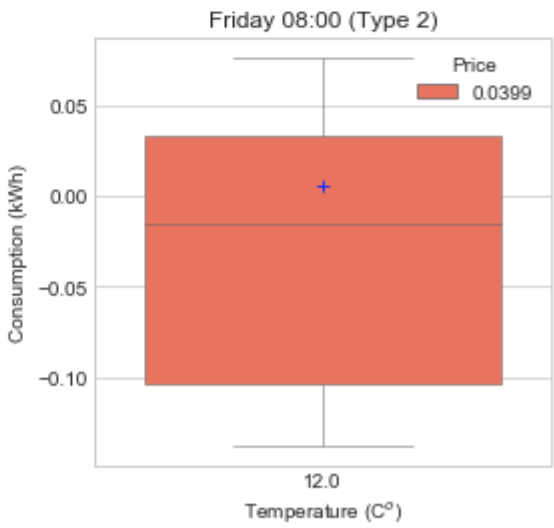

Friday 07:00 (Type 0)
Price
0.0399
Consumption (kWh)
9.0
Temperature (C°)

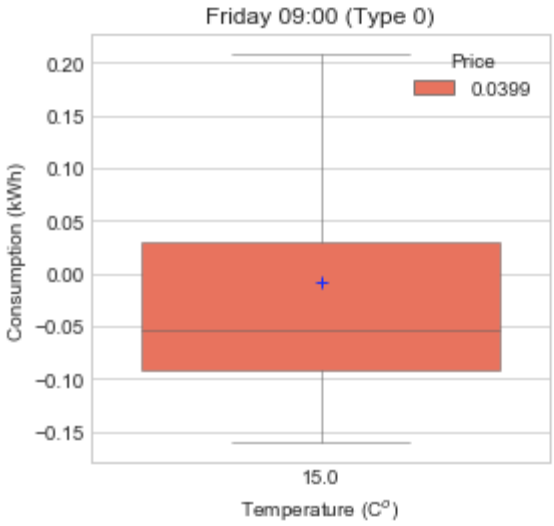

Friday 07:00 (Type 1)
Price
0.0399
Consumption (kWh)
9.0
Temperature (C°)

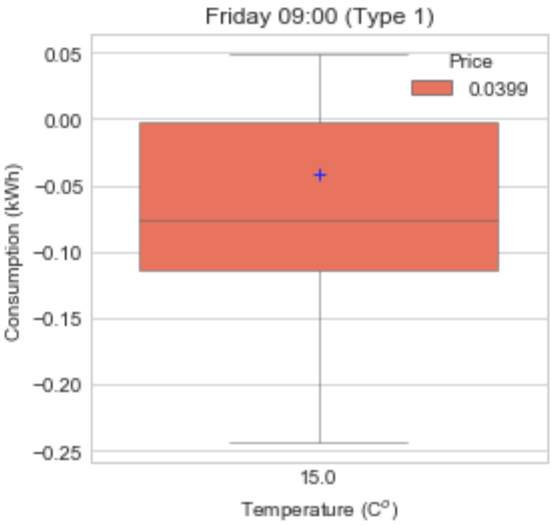

Friday 07:00 (Type 2)
Price
0.0399
Consumption (kWh)
9.0
Temperature (C°)

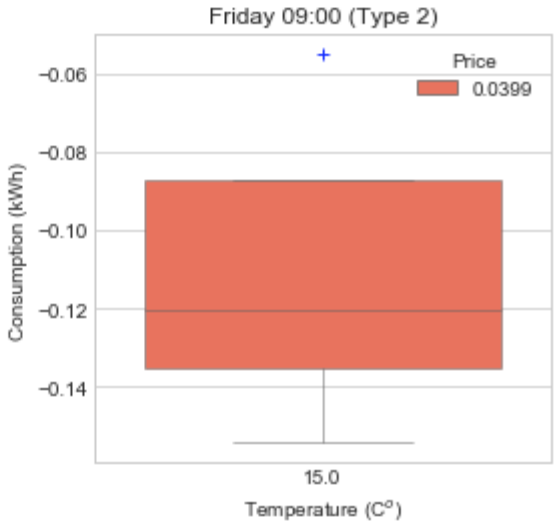

Friday 08:00 (Type 0)
Price
0.0399
Consumption (kWh)
12.0
Temperature (C°)

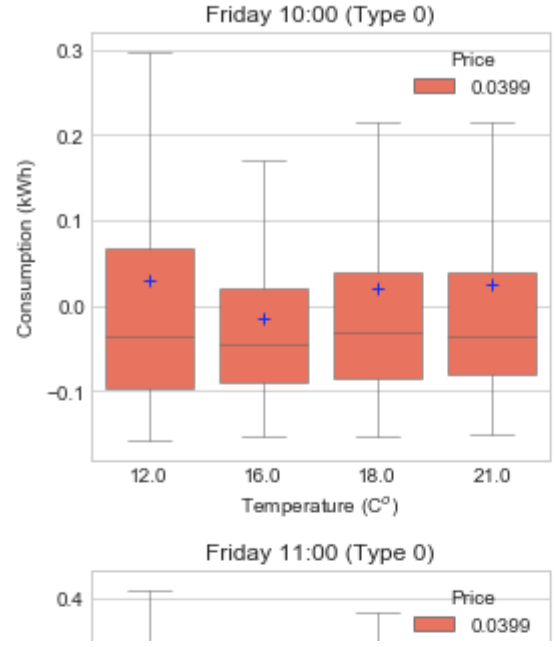

Friday 08:00 (Type 1)
Price
0.0399
Consumption (kWh)
12.0
Temperature (C°)

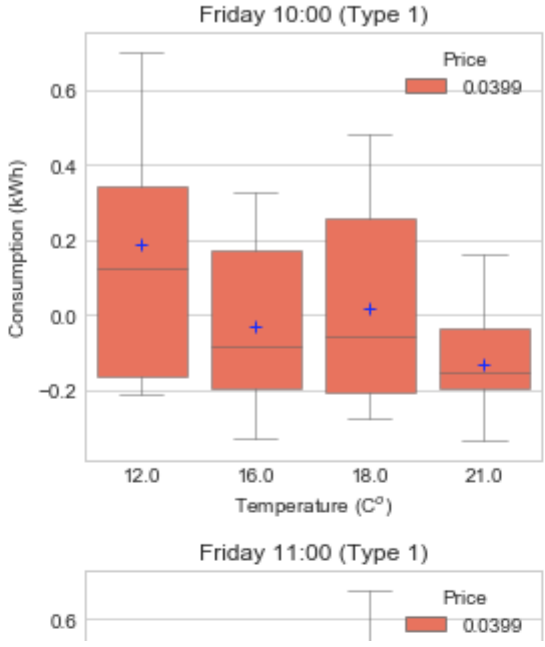

Friday 08:00 (Type 2)
Price
0.0399
Consumption (kWh)
12.0
Temperature (C°)

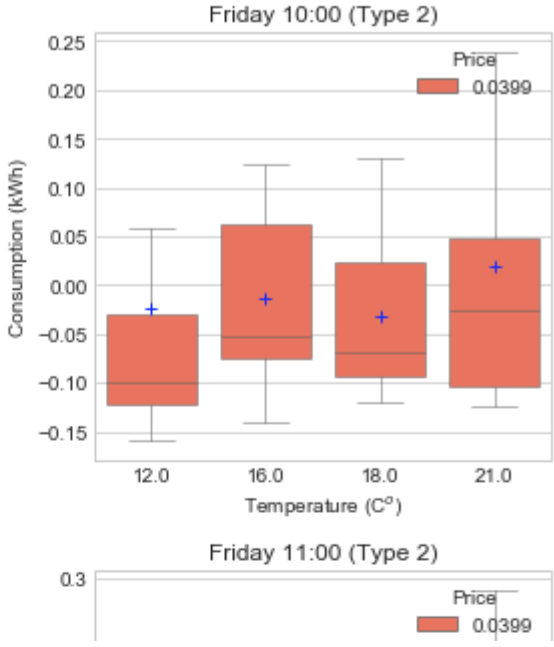

Friday 09:00 (Type 0)
Price
0.0399
Consumption (kWh)
15.0
Temperature (C°)


Friday 09:00 (Type 1)
Price
0.0399
Consumption (kWh)
15.0
Temperature (C°)


Friday 09:00 (Type 2)
Price
0.0399
Consumption (kWh)
15.0
Temperature (C°)


Friday 10:00 (Type 0)
Price
0.0399
Consumption (kWh)
12.0
16.0
18.0
21.0
Temperature (C°)

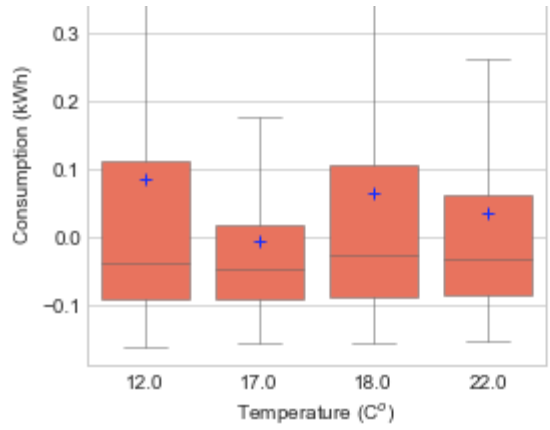

Friday 10:00 (Type 1)
Price
0.0399
Consumption (kWh)
12.0
16.0
18.0
21.0
Temperature (C°)

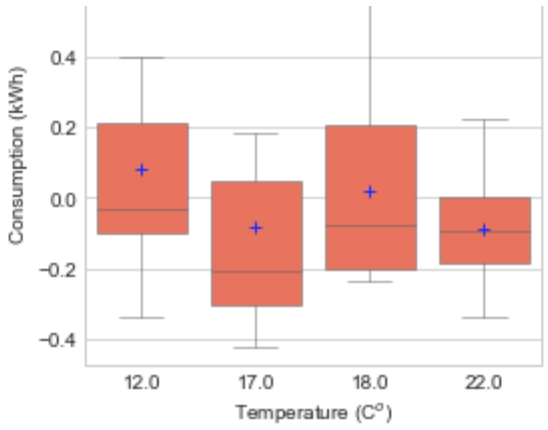

Friday 10:00 (Type 2)
Price
0.0399
Consumption (kWh)
12.0
16.0
18.0
21.0
Temperature (C°)

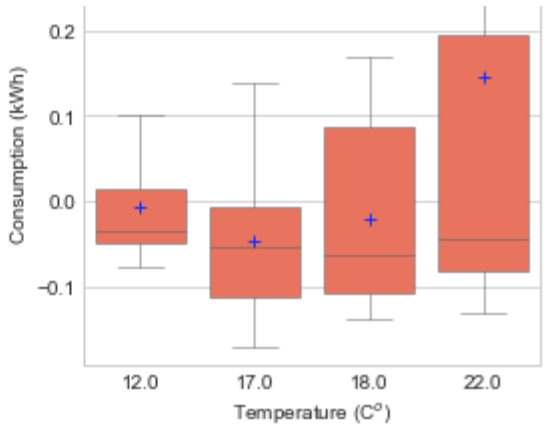
Friday 11:00 (Type 0)
Price
0.0399

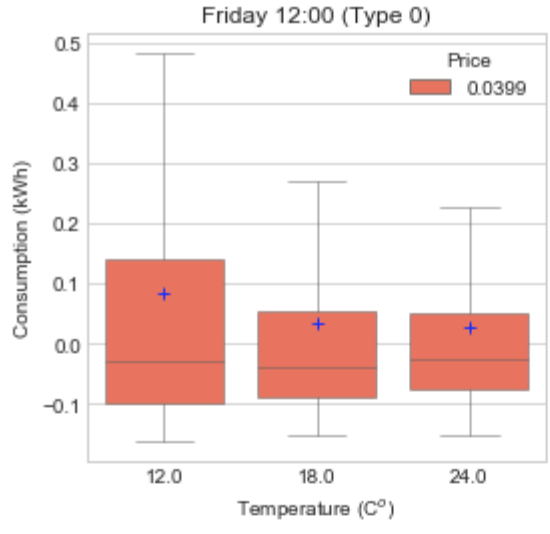
Friday 11:00 (Type 1)
Price
0.0399

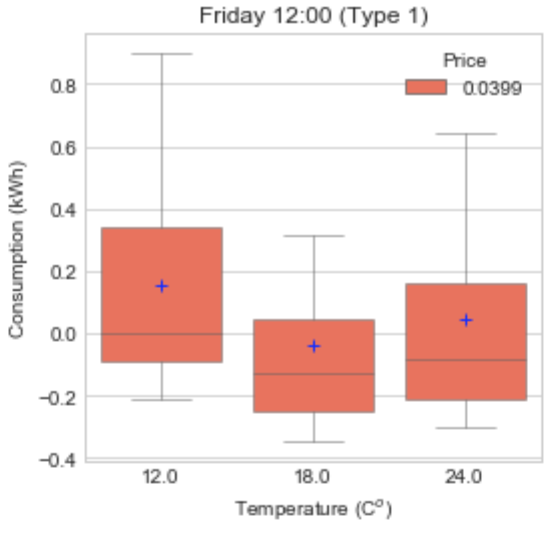
Friday 11:00 (Type 2)
Price
0.0399

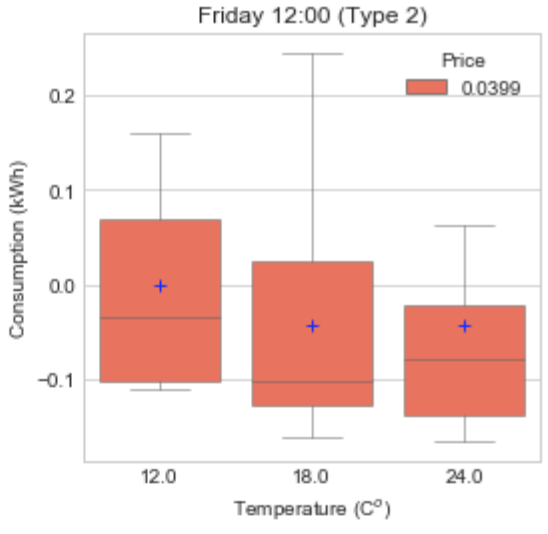

Consumption (kWh)
Temperature (C°)
12.0
17.0
18.0
22.0

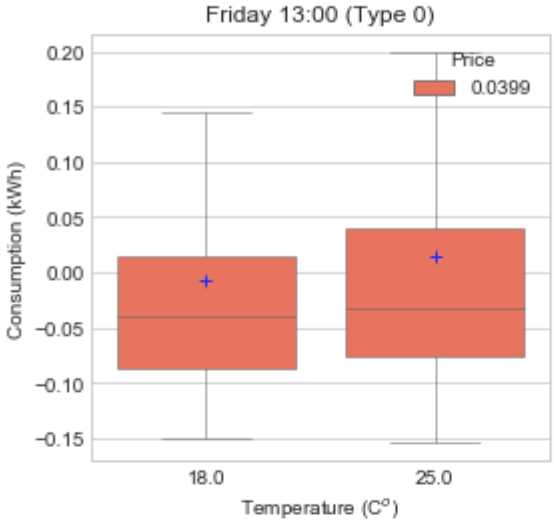

Consumption (kWh)
Temperature (C°)
12.0
17.0
18.0
22.0

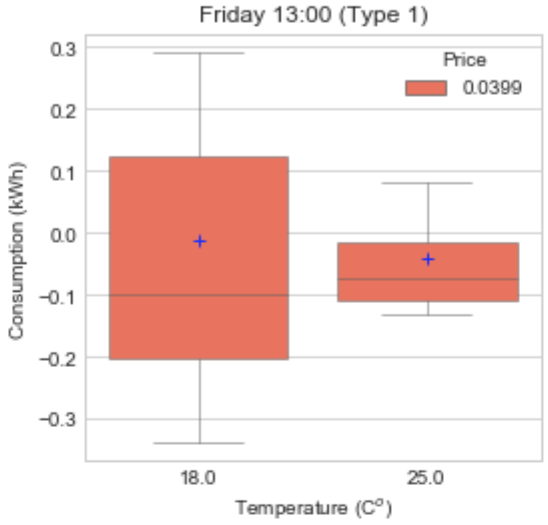

Consumption (kWh)
Temperature (C°)
12.0
17.0
18.0
22.0

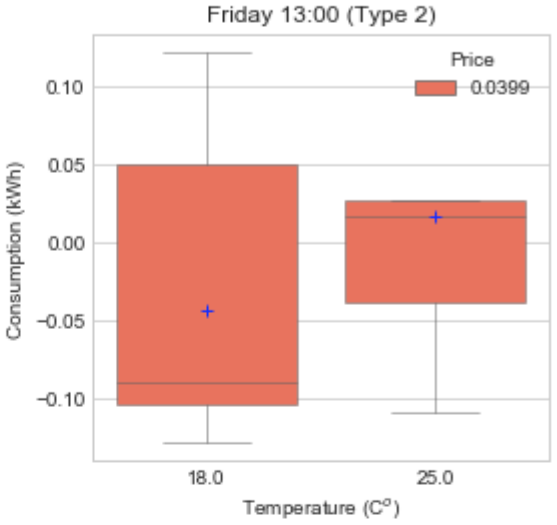

Friday 12:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
12.0
18.0
24.0

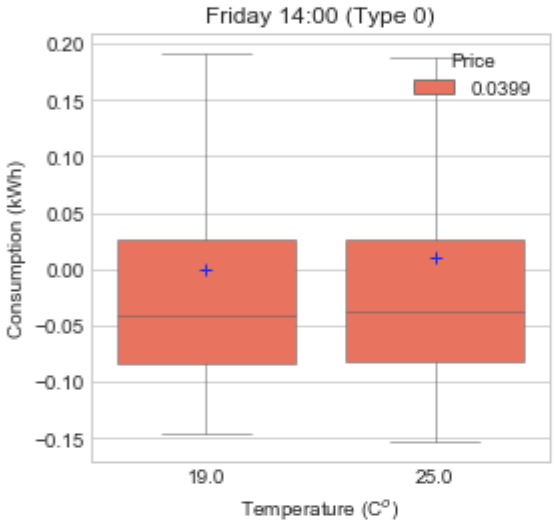

Friday 12:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
12.0
18.0
24.0

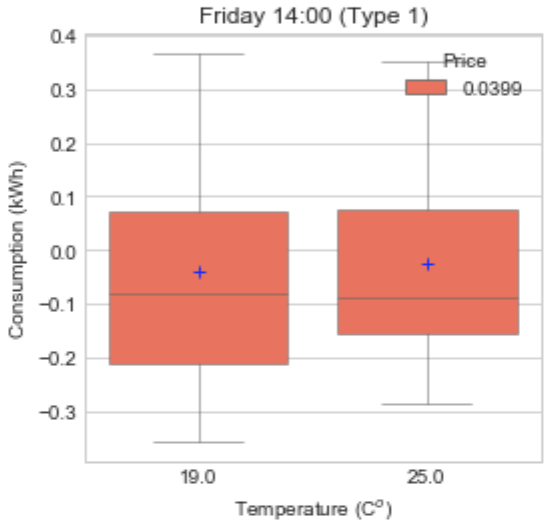

Friday 12:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
12.0
18.0
24.0

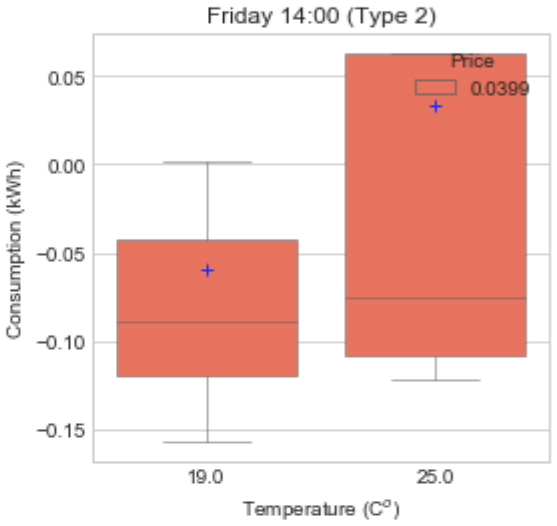

Friday 13:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
18.0
25.0

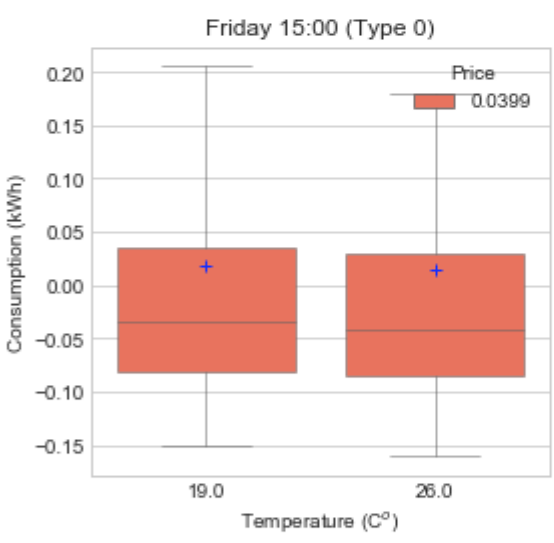

Friday 13:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
18.0
25.0

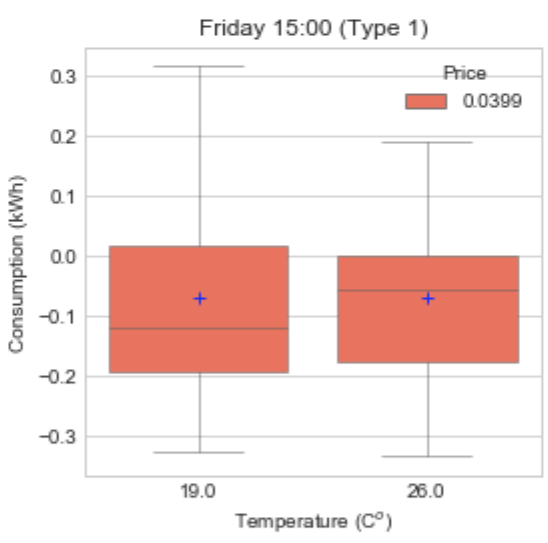

Friday 13:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
18.0
25.0

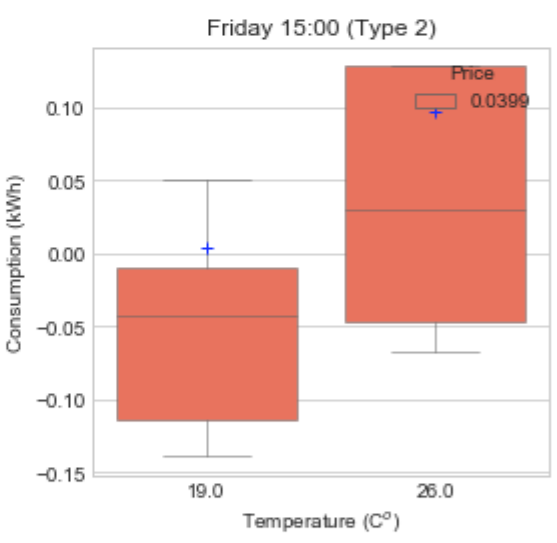

Friday 14:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
25.0

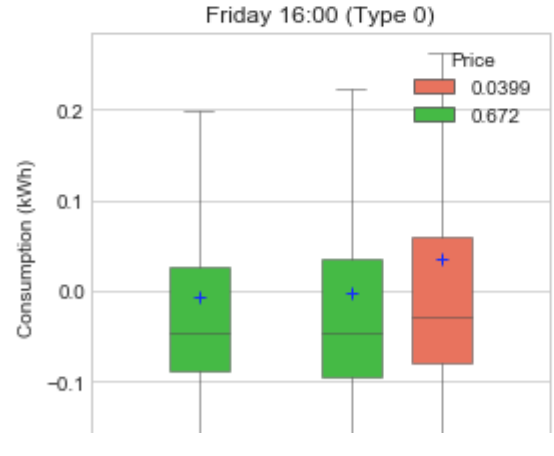

Friday 14:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
25.0

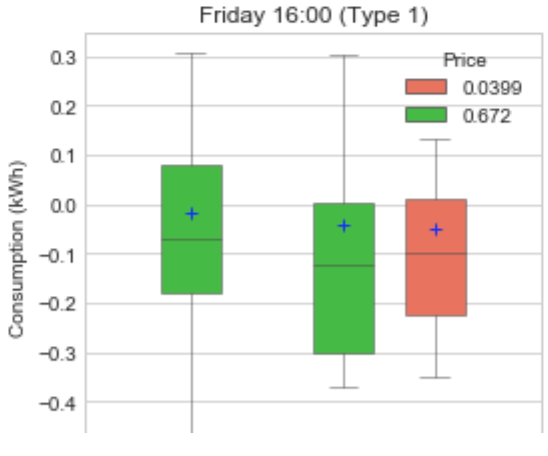

Friday 14:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
25.0

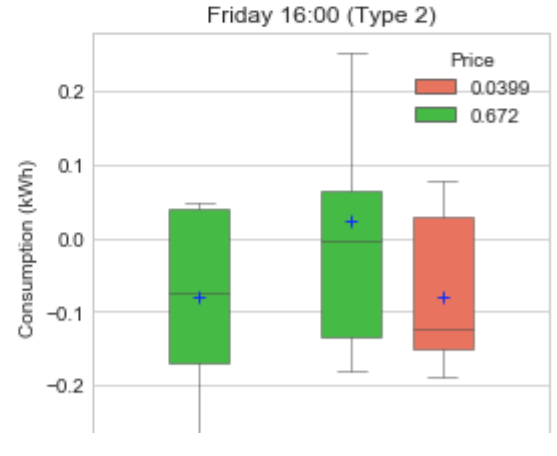

Friday 15:00 (Type 0)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
26.0

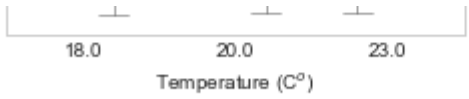

Friday 15:00 (Type 1)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
26.0


Friday 15:00 (Type 2)
Price
0.0399
Consumption (kWh)
Temperature (C°)
19.0
26.0


Friday 16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

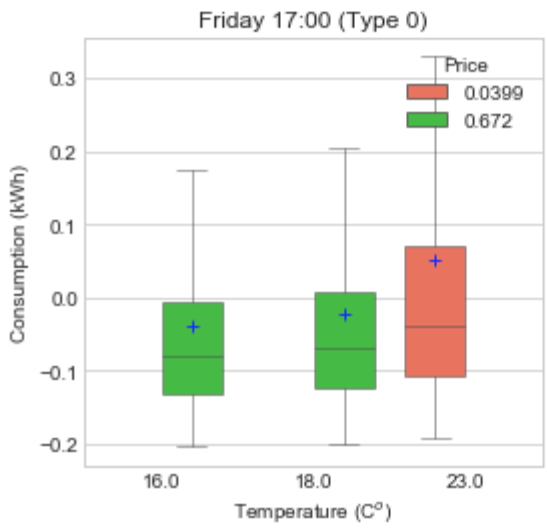

Friday 16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

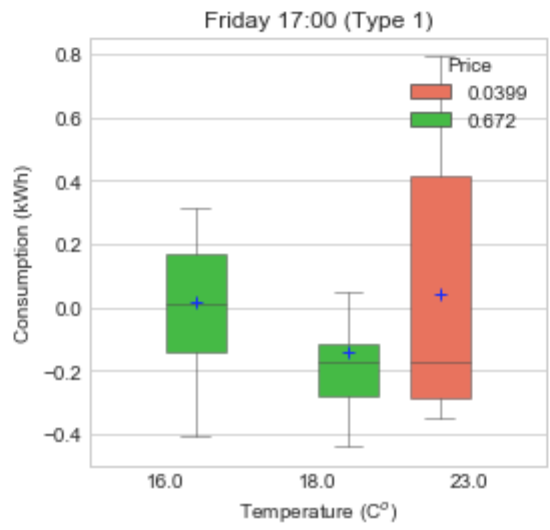

Friday 16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

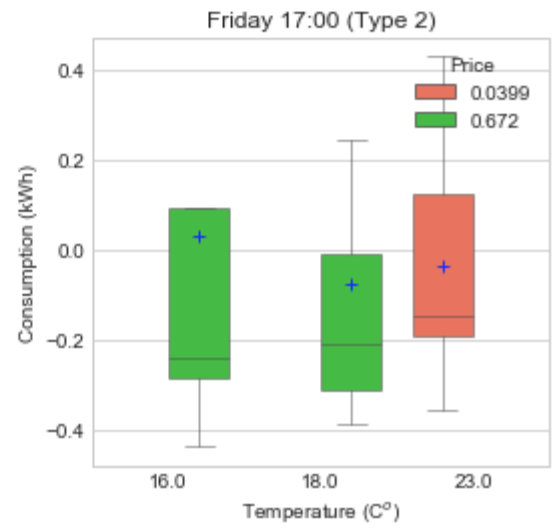

18.0
20.0
23.0
Temperature (C°)

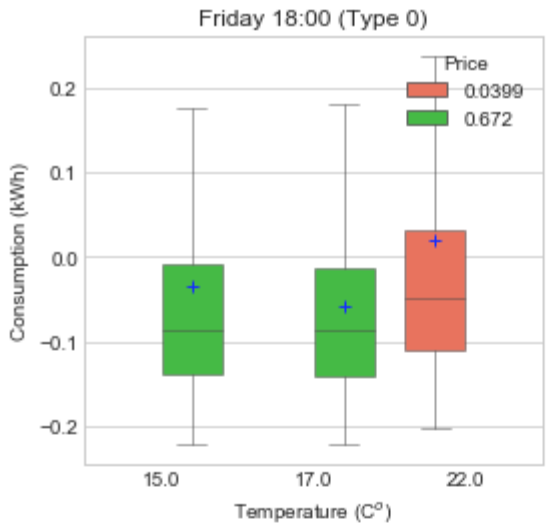

−0.5
18.0
20.0
23.0
Temperature (C°)

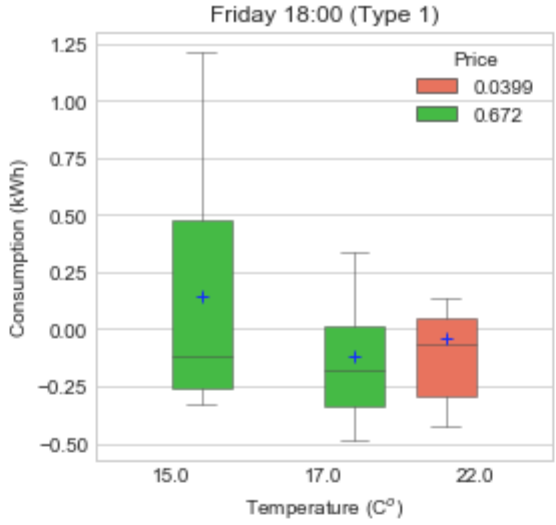

−0.3
18.0
20.0
23.0
Temperature (C°)

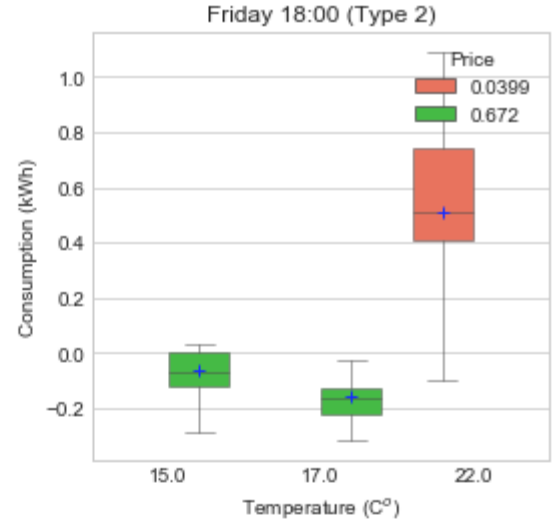

Friday 17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
16.0
18.0
23.0
Temperature (C°)

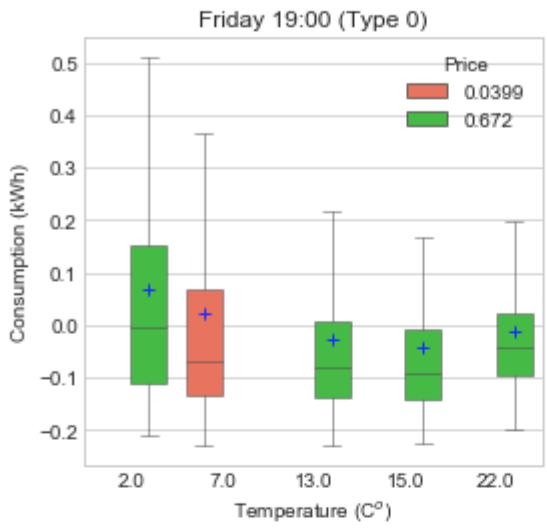

Friday 17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
16.0
18.0
23.0
Temperature (C°)

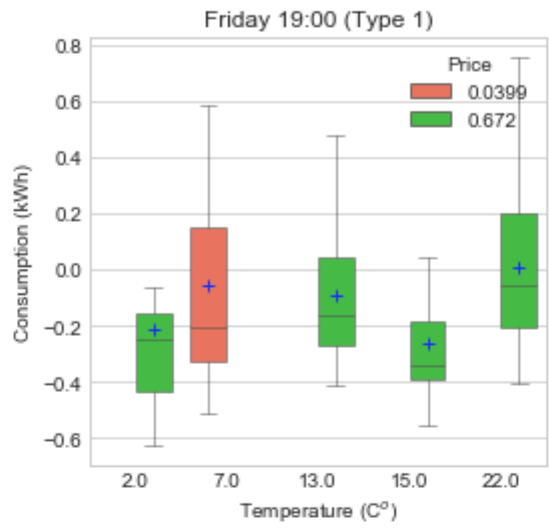

Friday 17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
16.0
18.0
23.0
Temperature (C°)

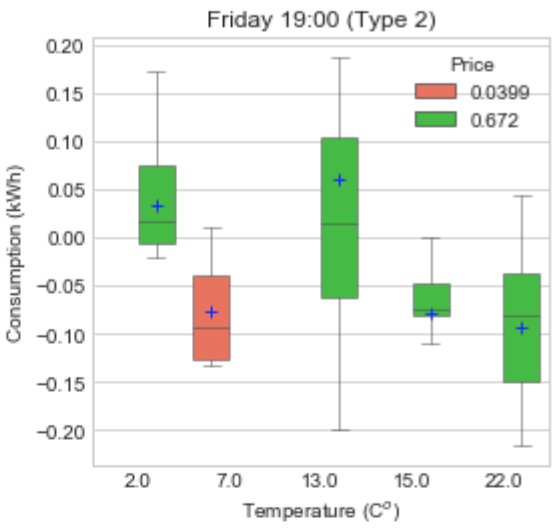

Friday 18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
15.0
17.0
22.0
Temperature (C°)

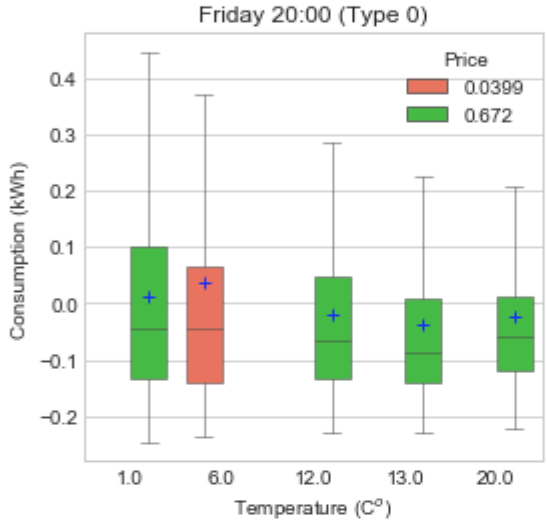

Friday 18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
15.0
17.0
22.0
Temperature (C°)

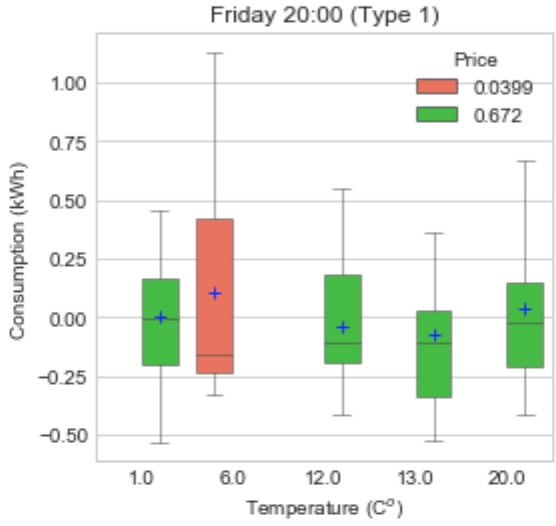

Friday 18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
15.0
17.0
22.0
Temperature (C°)

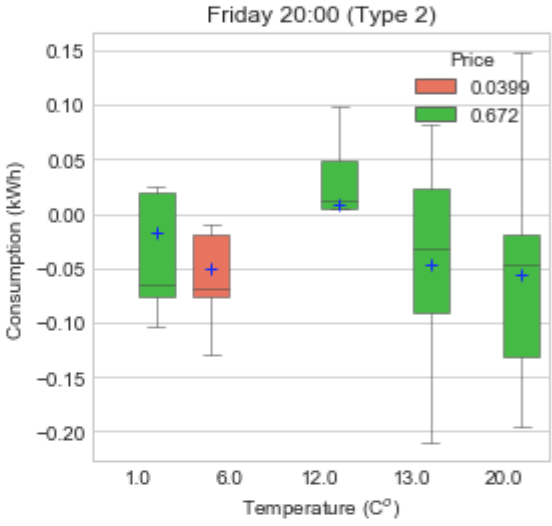

Friday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
2.0
7.0
13.0
15.0
22.0
Temperature (C°)

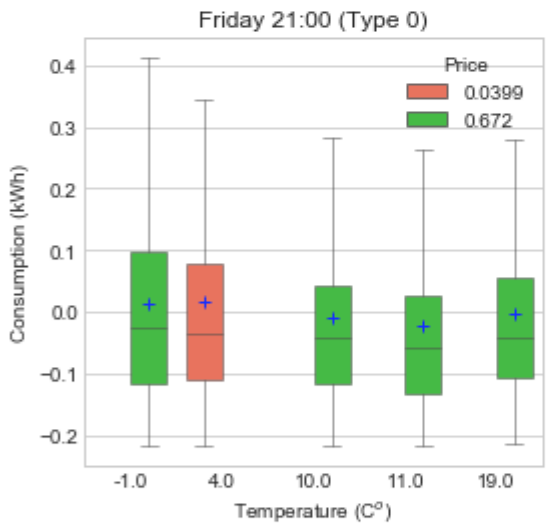

Friday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
2.0
7.0
13.0
15.0
22.0
Temperature (C°)

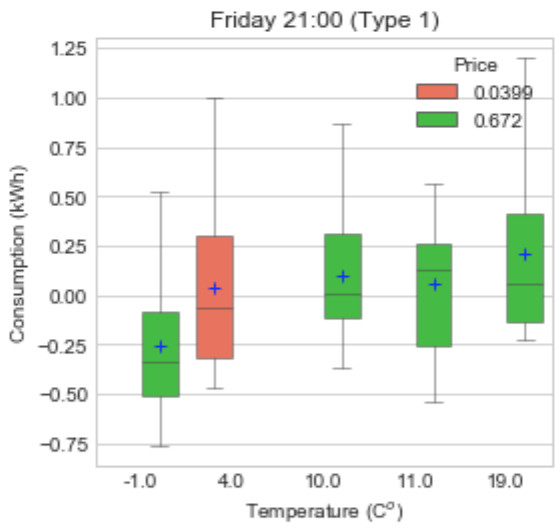

Friday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
2.0
7.0
13.0
15.0
22.0
Temperature (C°)

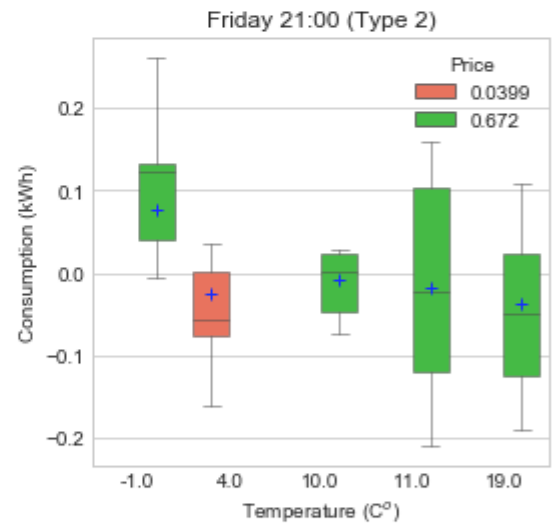

Friday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
1.0
6.0
12.0
13.0
20.0
Temperature (C°)

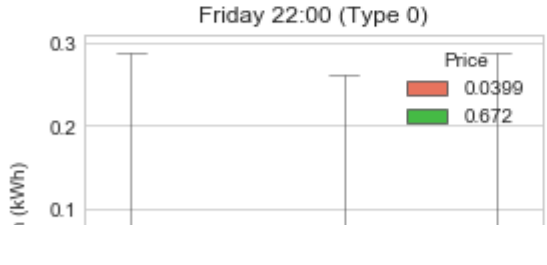

Friday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
1.0
6.0
12.0
13.0
20.0
Temperature (C°)

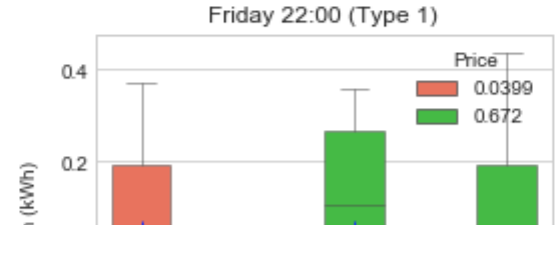

Friday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
1.0
6.0
12.0
13.0
20.0
Temperature (C°)

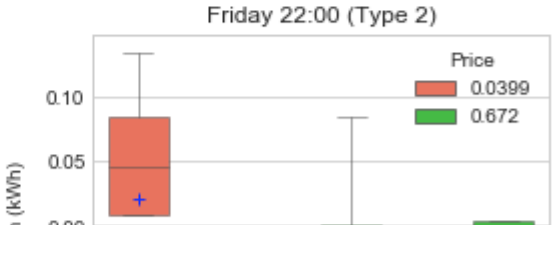

Friday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
-1.0
4.0
10.0
11.0
19.0
Temperature (C°)

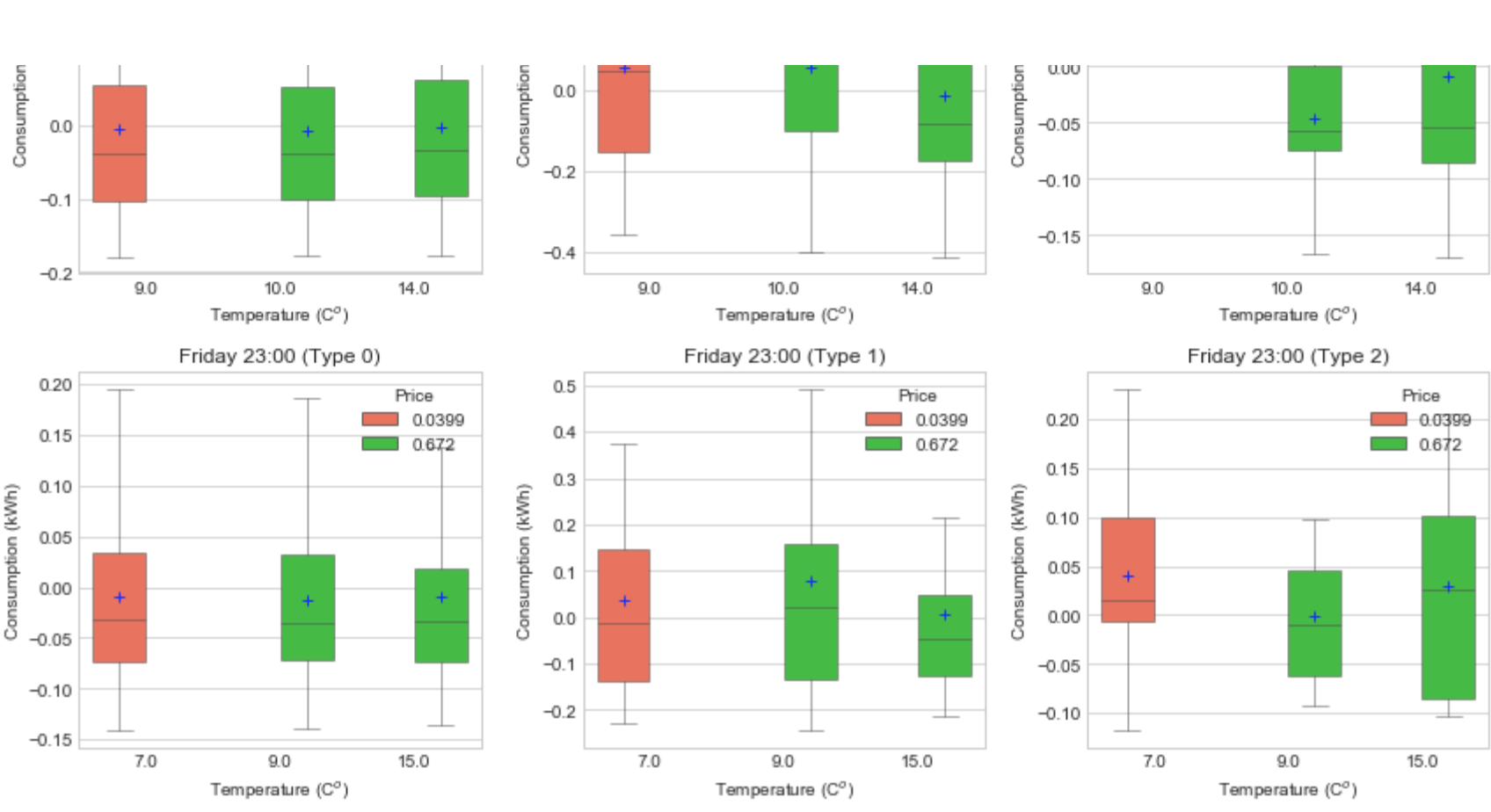

Friday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
-1.0
4.0
10.0
11.0
19.0
Temperature (C°)


Friday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
-1.0
4.0
10.0
11.0
19.0
Temperature (C°)

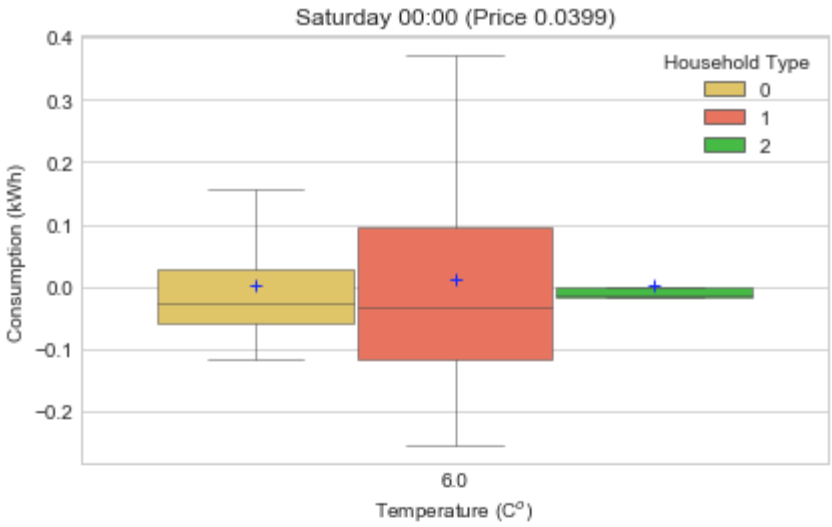

Friday 22:00 (Type 0)
Price
0.0399
0.672
0.3
0.2
0.1

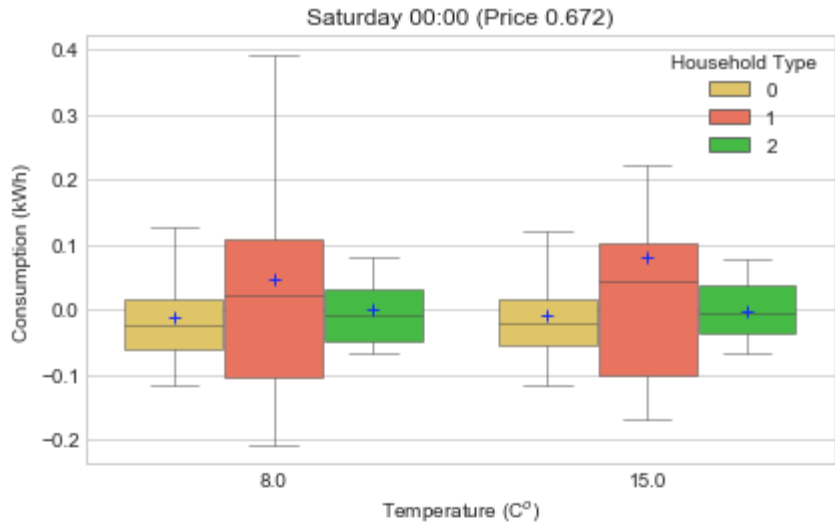

Friday 22:00 (Type 1)
Price
0.0399
0.672
0.4
0.2

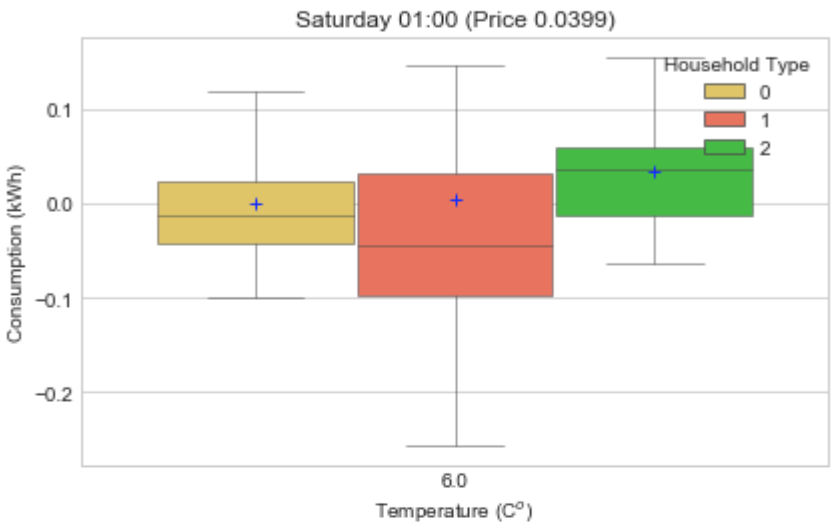

Friday 22:00 (Type 2)
Price
0.0399
0.672
0.10
0.05

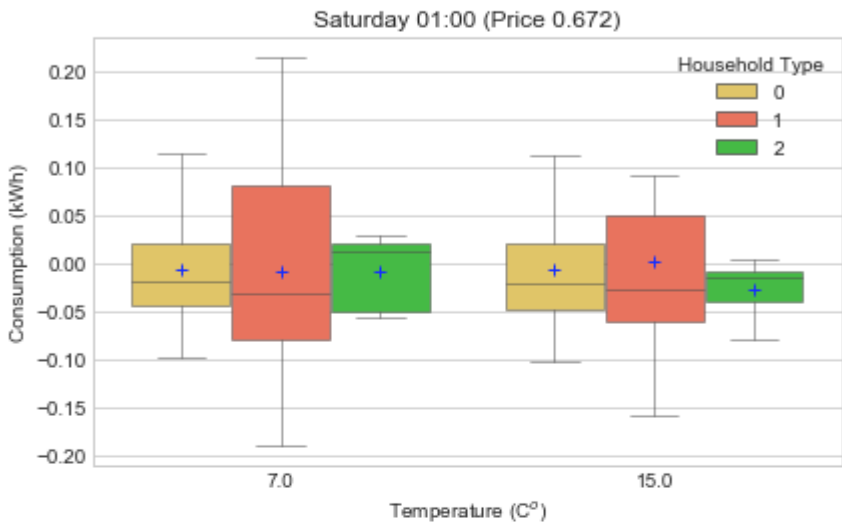

Consumption
Temperature (C°)
9.0
10.0
14.0
Friday 23:00 (Type 0)
Friday 23:00 (Type 1)
Friday 23:00 (Type 2)
Consumption (kWh)
Price
0.0399
0.672
7.0
9.0
15.0

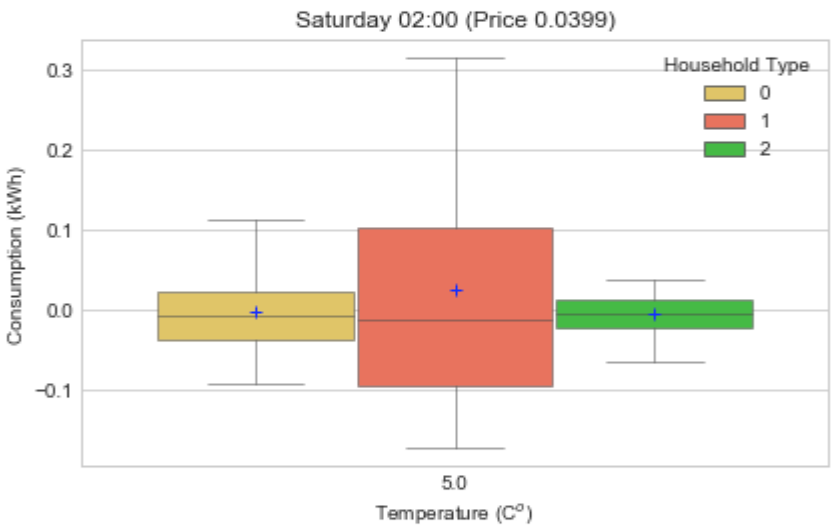
In [71]:
```
# Saturday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 5 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

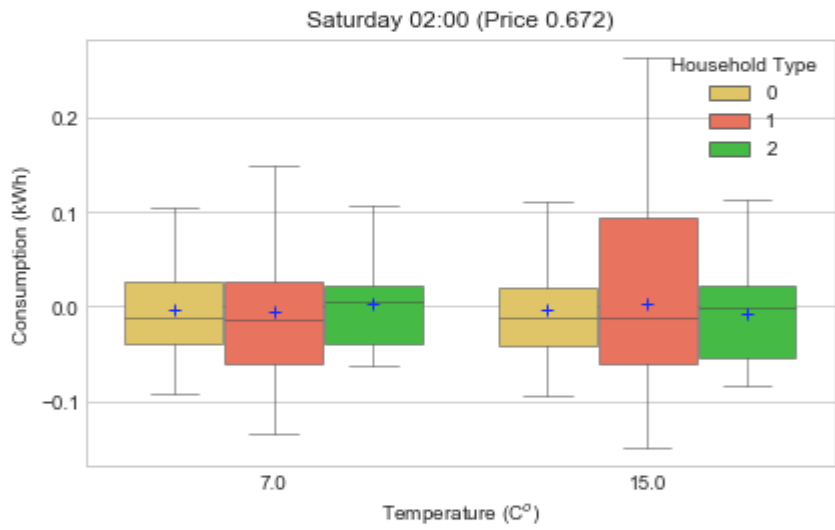

Saturday 00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
6.0
Temperature (C°)

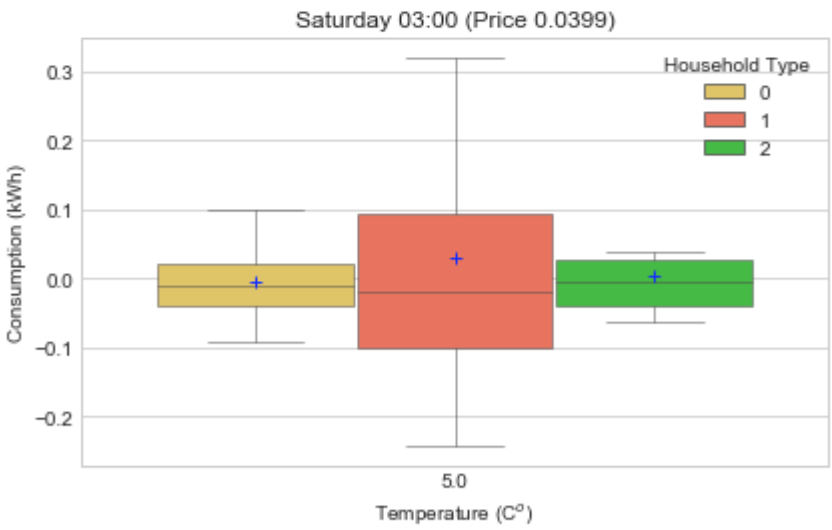

Saturday 00:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
8.0
15.0
Temperature (C°)

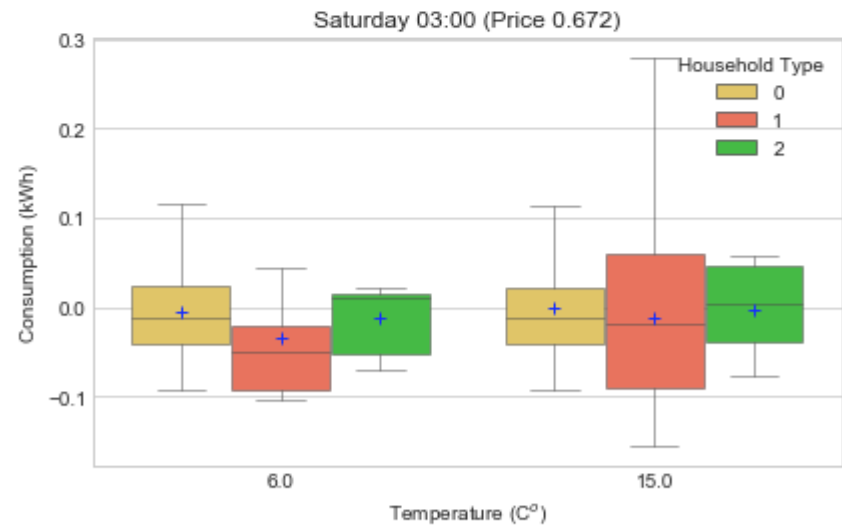

Saturday 01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.1
0.0
−0.1
−0.2
6.0
Temperature (C°)

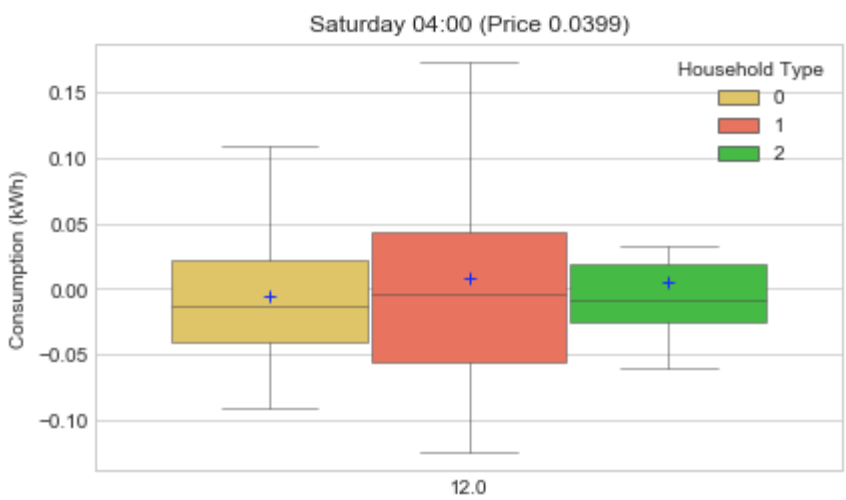

Saturday 01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
7.0
15.0
Temperature (C°)

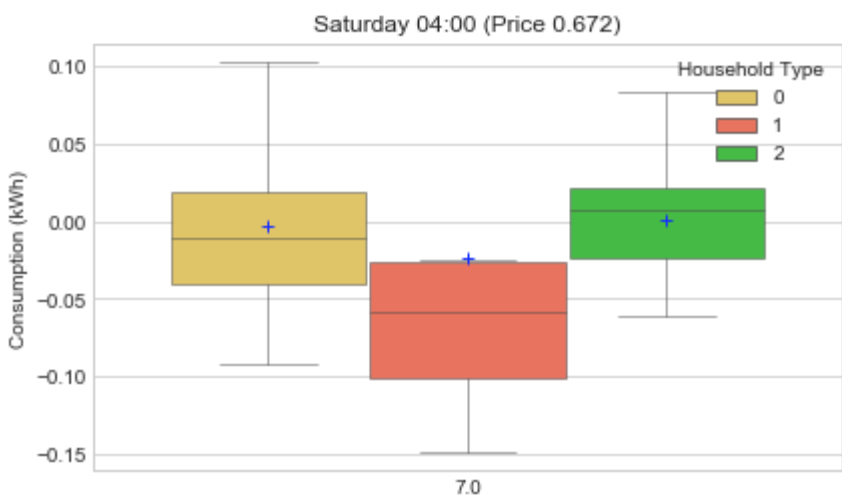

Saturday 02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
5.0
Temperature (C°)

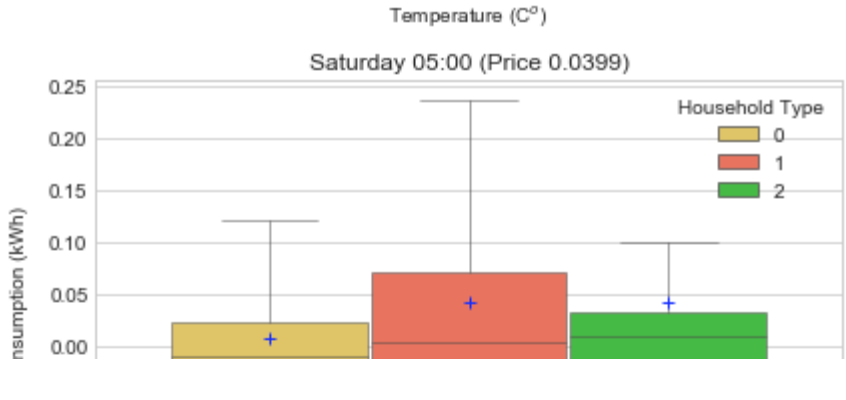

Saturday 02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
7.0
15.0
Temperature (C°)

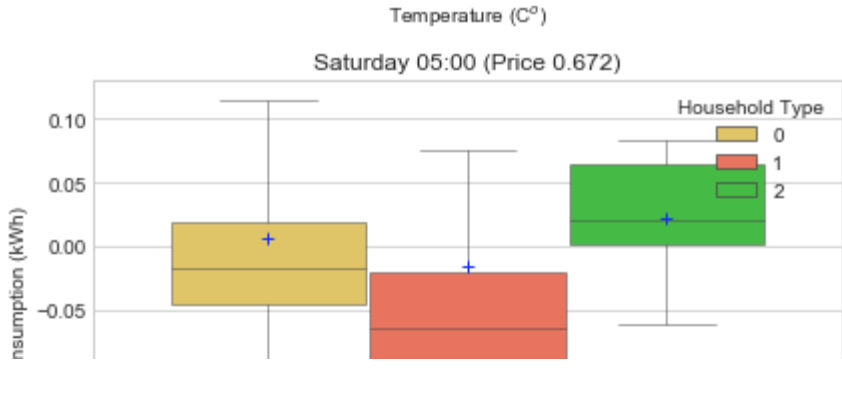

Saturday 03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
5.0
Temperature (C°)

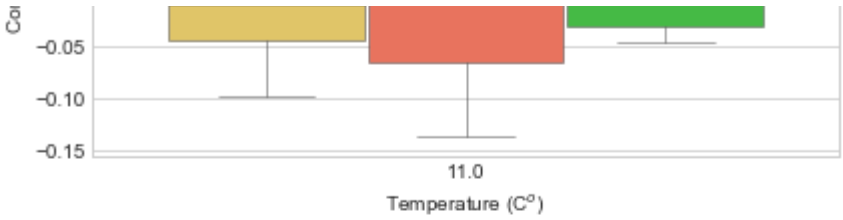

Saturday 03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
6.0
15.0
Temperature (C°)

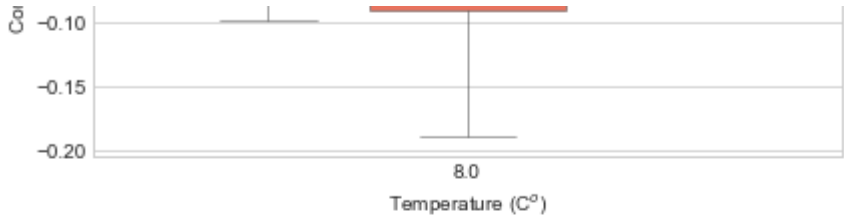

Saturday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
12.0
Temperature (C°)

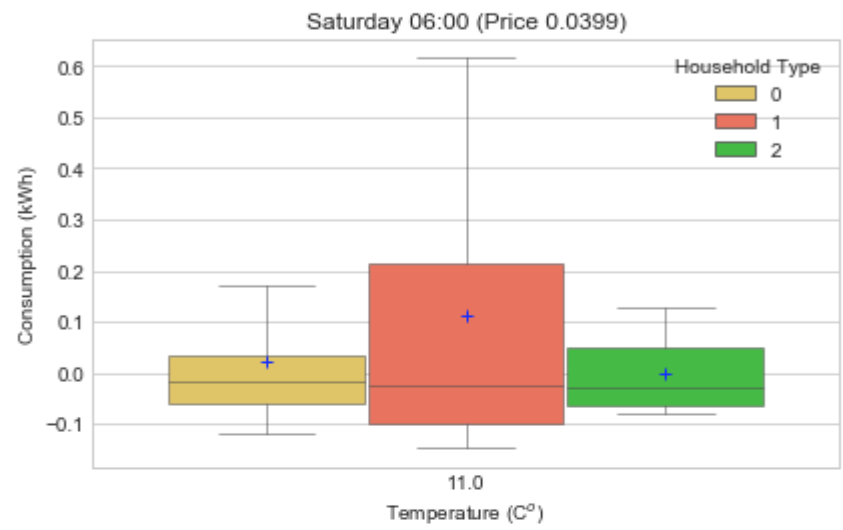

Saturday 04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
−0.15
7.0
Temperature (C°)

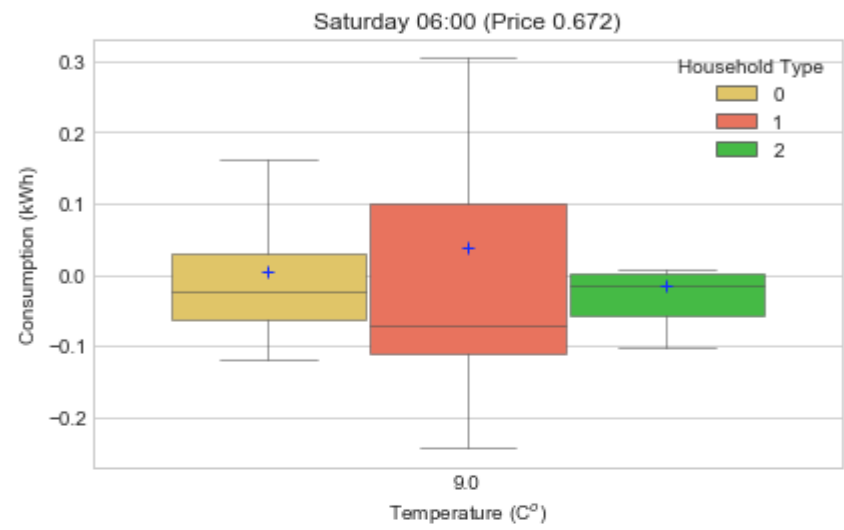

Saturday 05:00 (Price 0.0399)
Household Type
0
1
2
0.25
0.20
0.15
0.10
0.05
0.00
Temperature (C°)

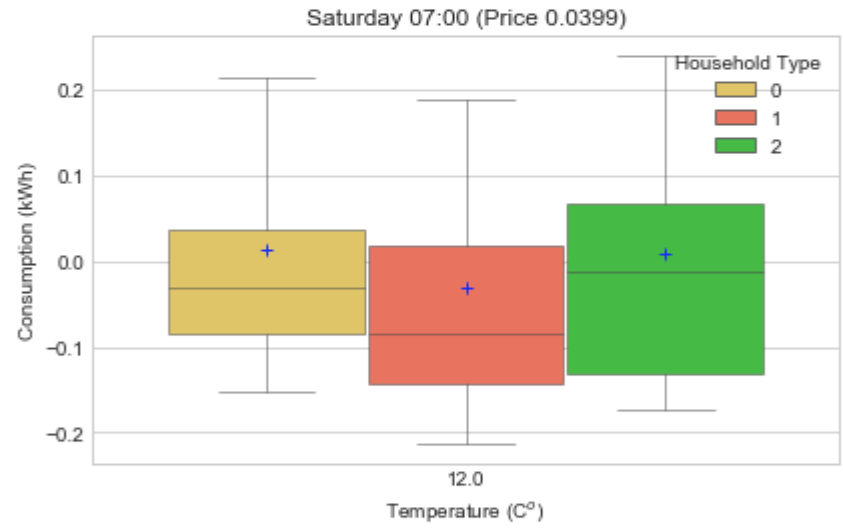

Saturday 05:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.10
0.05
0.00
−0.05
Temperature (C°)

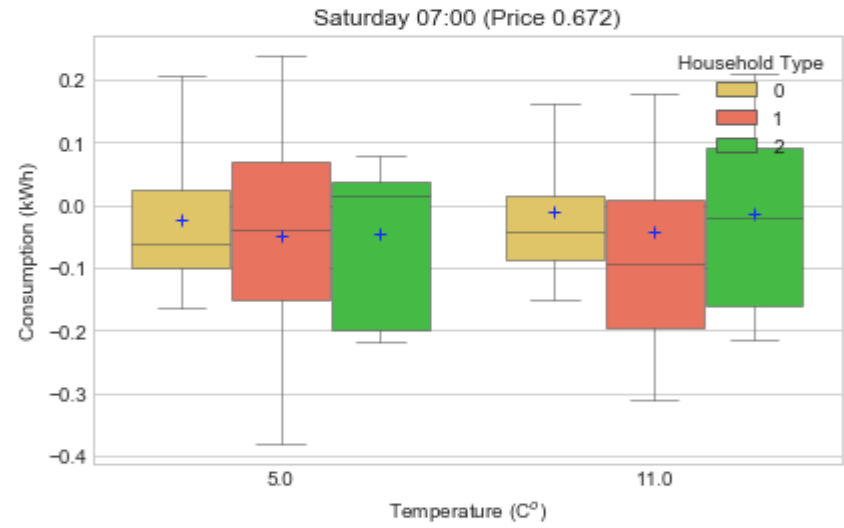

−0.05
−0.10
−0.15
11.0
Temperature (C°)

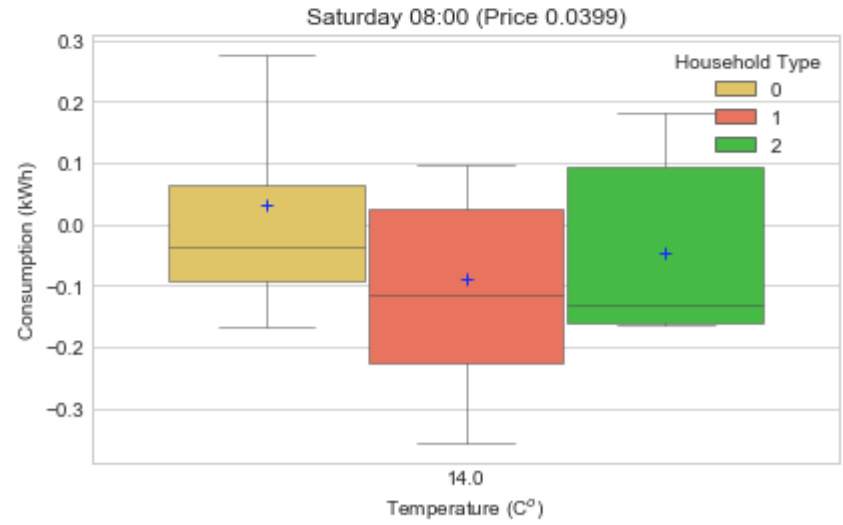

−0.10
−0.15
−0.20
8.0
Temperature (C°)

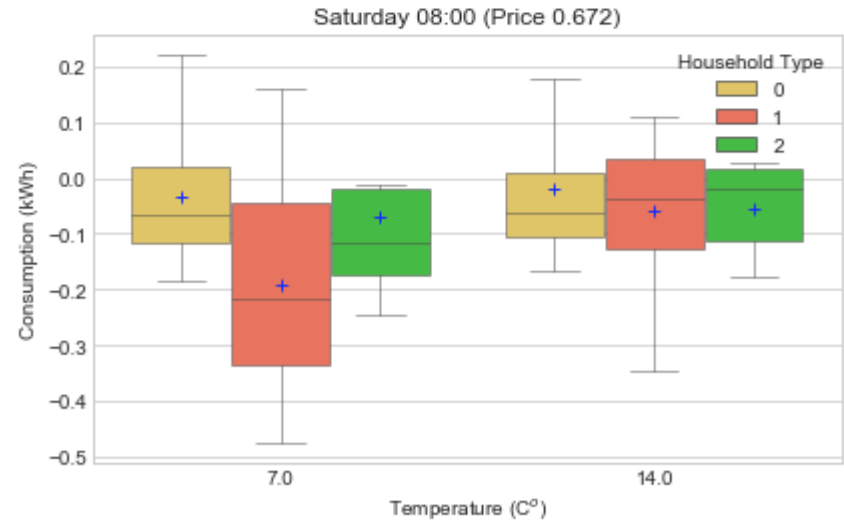

Saturday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
11.0
Temperature (C°)

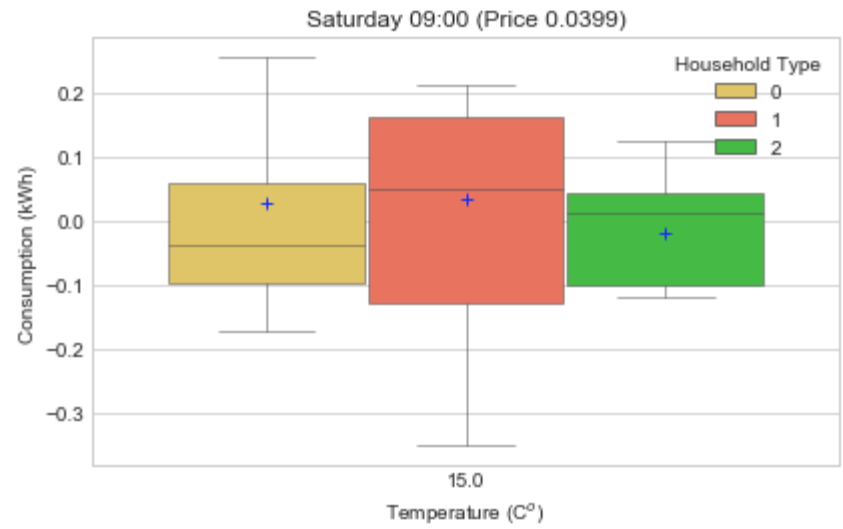

Saturday 06:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
9.0
Temperature (C°)

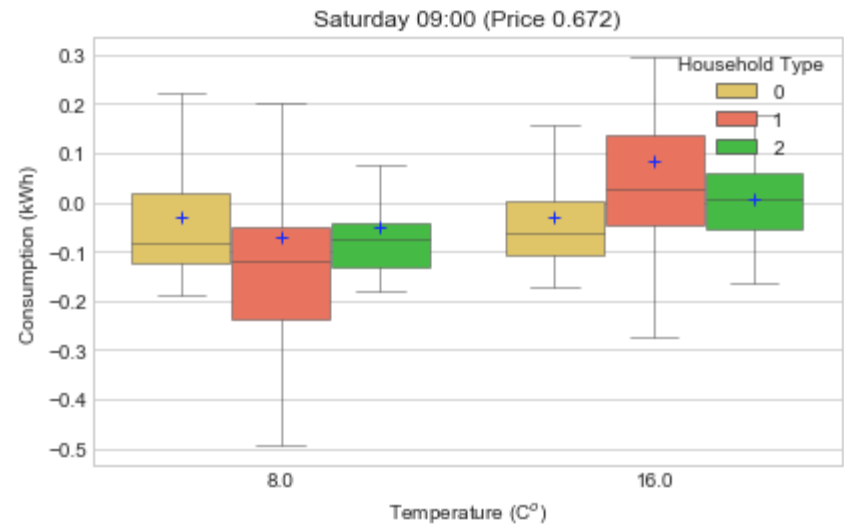

Saturday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
12.0
Temperature (C°)

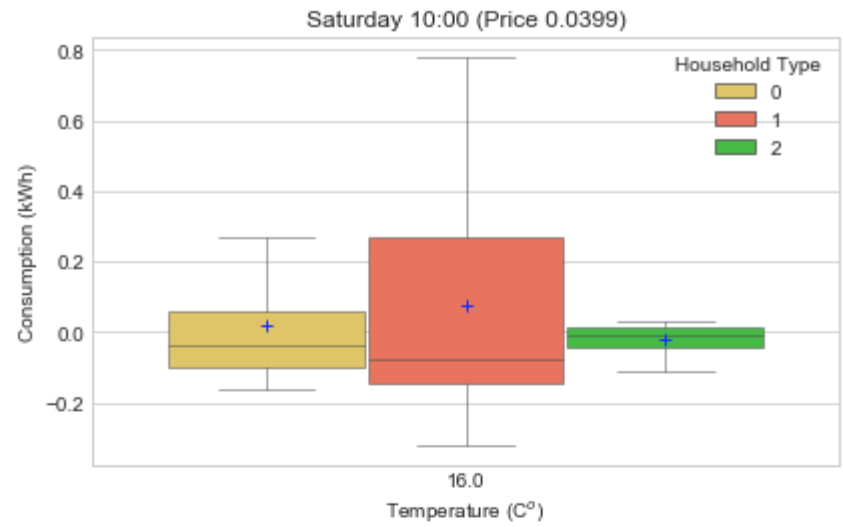

Saturday 07:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
5.0
11.0
Temperature (C°)

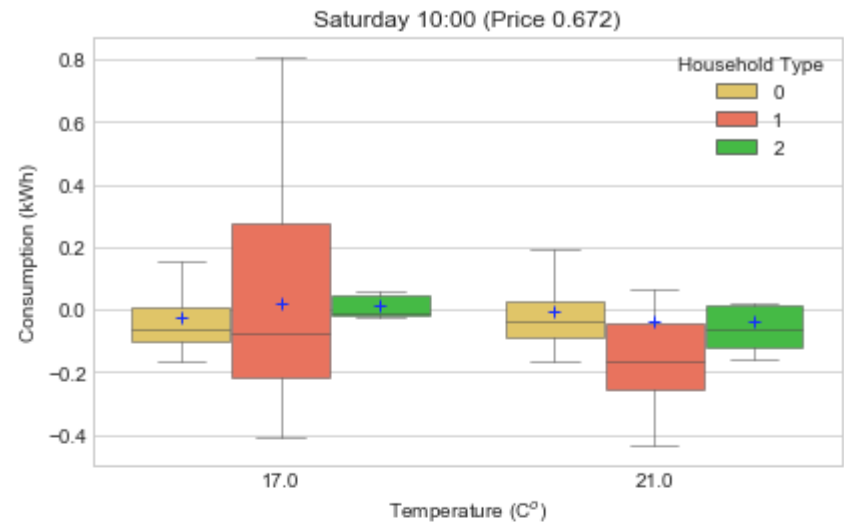

Saturday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
14.0
Temperature (C°)


Saturday 08:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
7.0
14.0
Temperature (C°)


Saturday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
15.0
Temperature (C°)

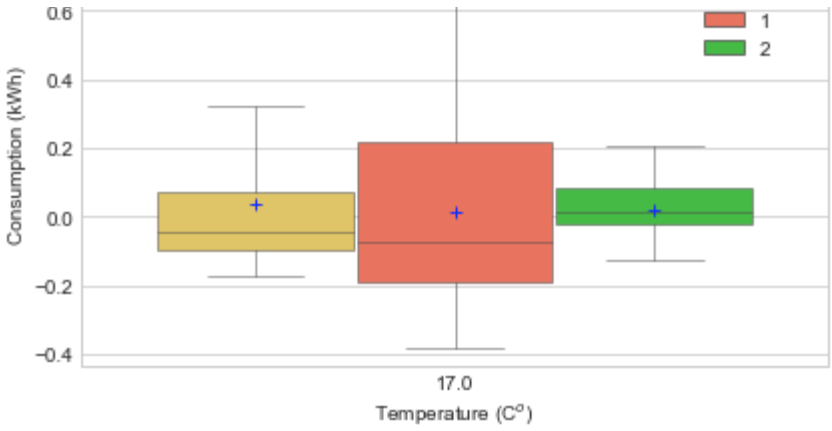

Saturday 09:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
8.0
16.0
Temperature (C°)

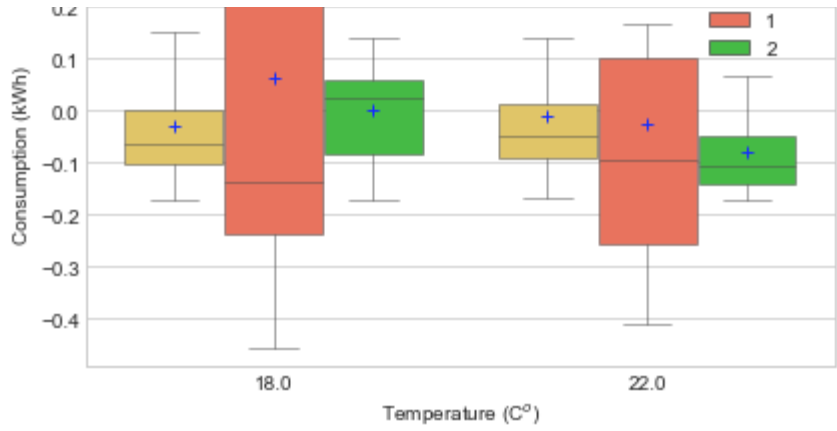

Saturday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
16.0
Temperature (C°)

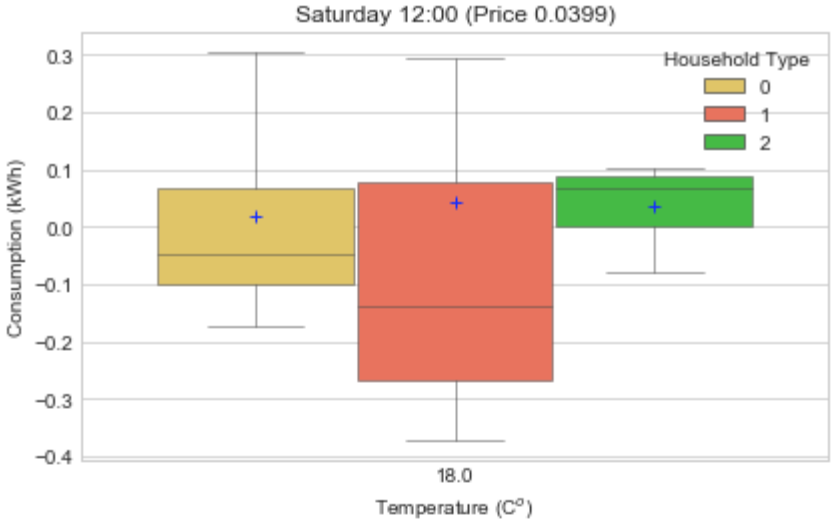

Saturday 10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
17.0
21.0
Temperature (C°)

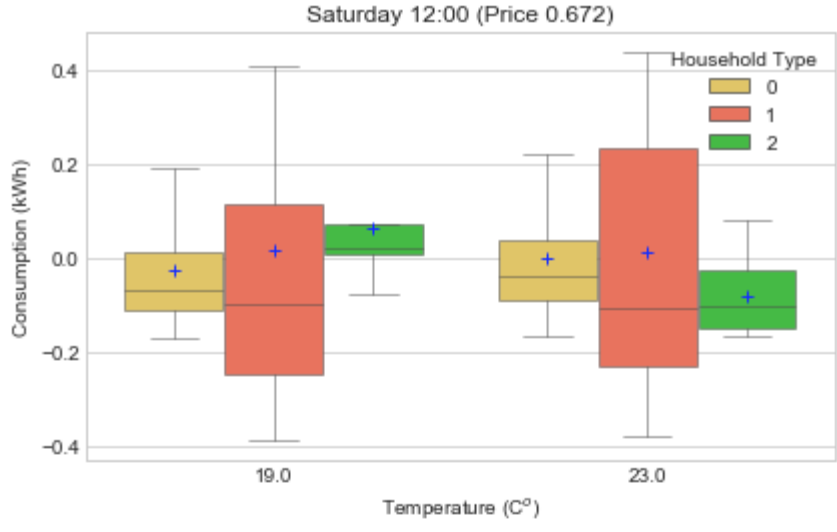

Saturday 11:00 (Price 0.0399)
Household Type
0
0.8

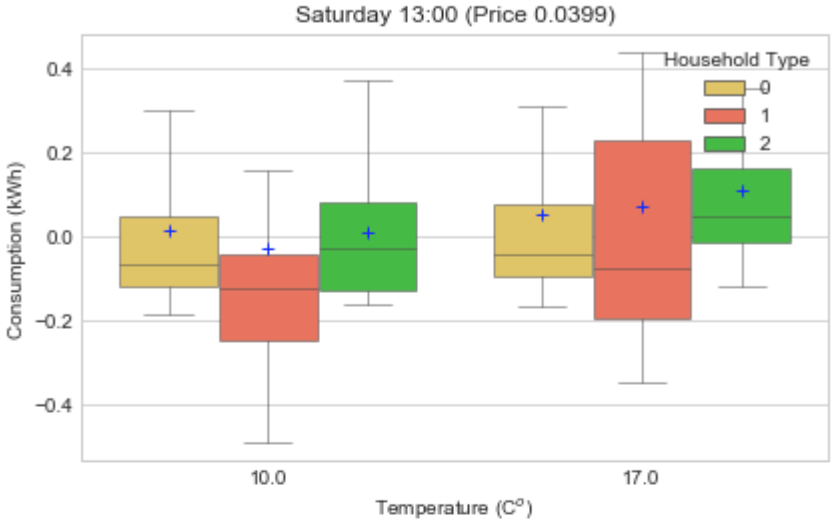

Saturday 11:00 (Price 0.672)
Household Type
0
0.3

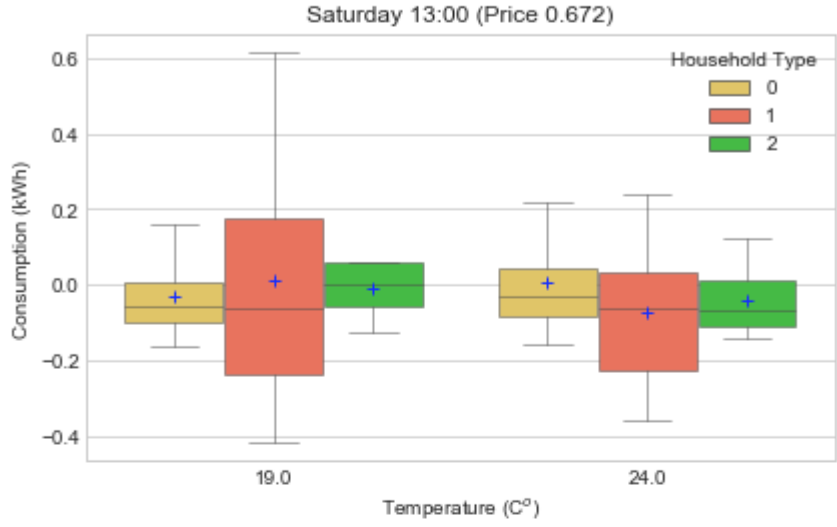

Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
1
2
17.0
Temperature (C°)

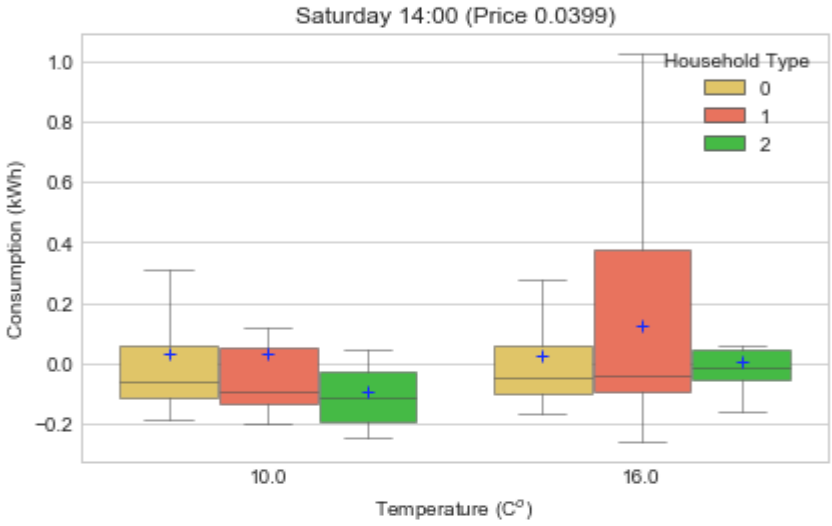

Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
1
2
18.0
22.0
Temperature (C°)

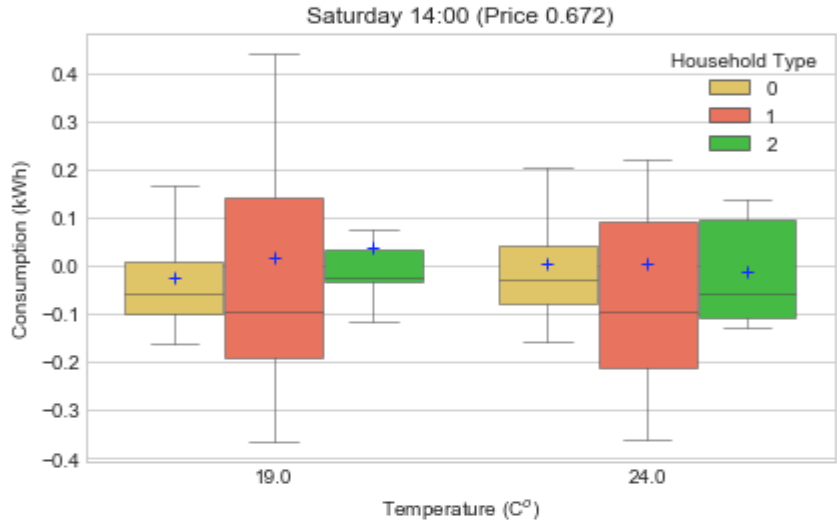

Saturday 12:00 (Price 0.0399)
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
18.0
Temperature (C°)

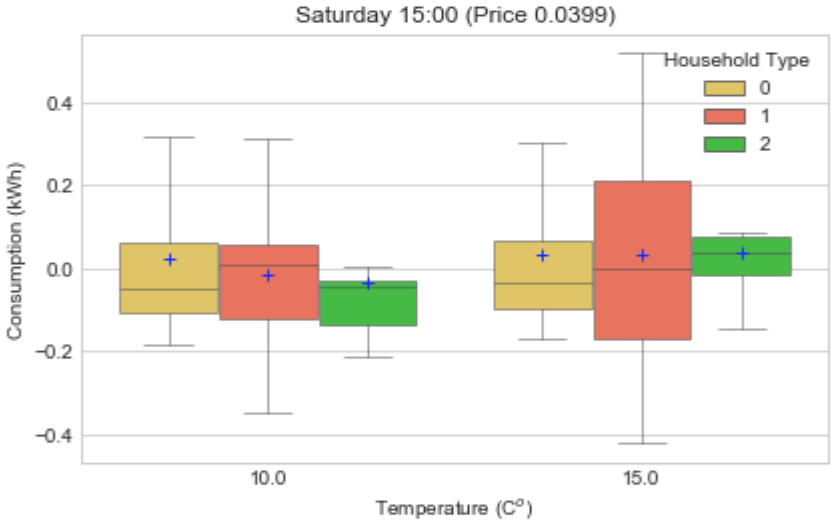

Saturday 12:00 (Price 0.672)
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
19.0
23.0
Temperature (C°)

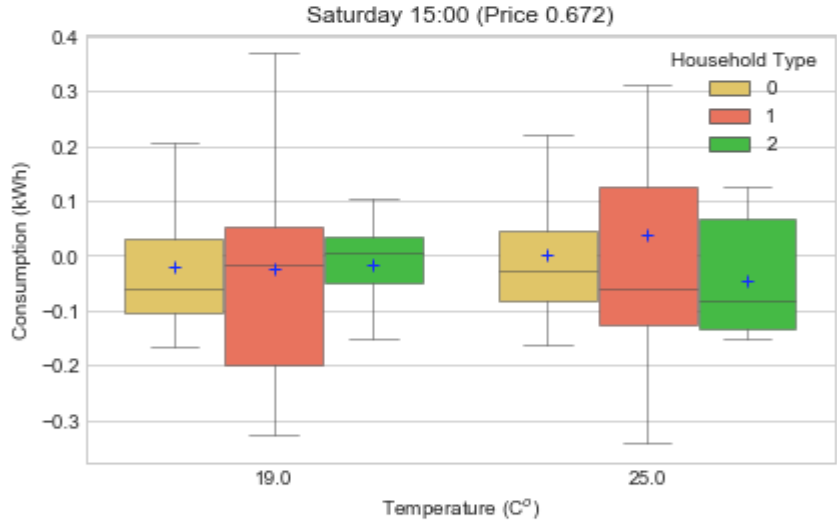

Saturday 13:00 (Price 0.0399)
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
10.0
17.0
Temperature (C°)

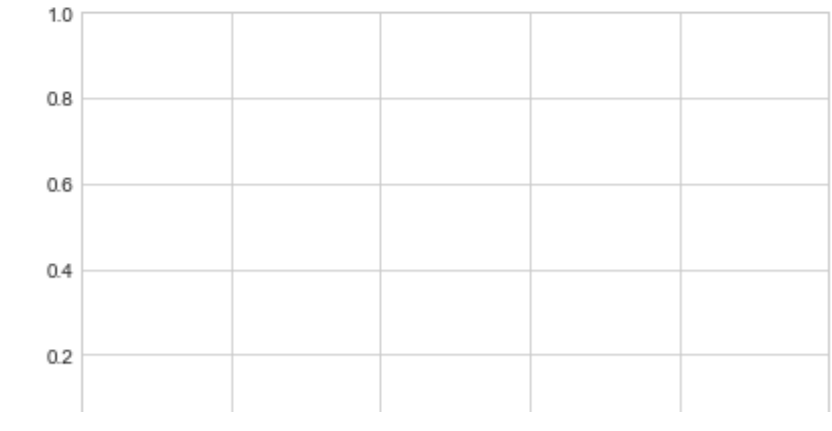

Saturday 13:00 (Price 0.672)
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
19.0
24.0
Temperature (C°)

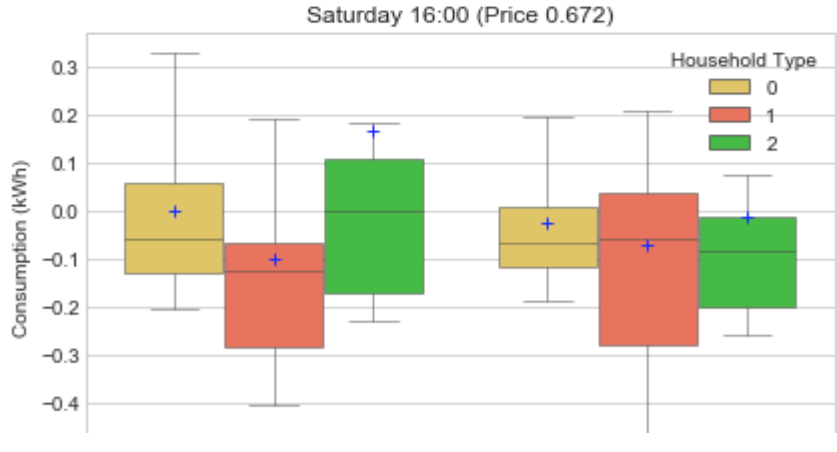

Saturday 14:00 (Price 0.0399)
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
Household Type
0
1
2
10.0
16.0
Temperature (C°)


Saturday 14:00 (Price 0.672)
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2
19.0
24.0
Temperature (C°)


Saturday 15:00 (Price 0.0399)
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
Household Type
0
1
2
10.0
15.0
Temperature (C°)

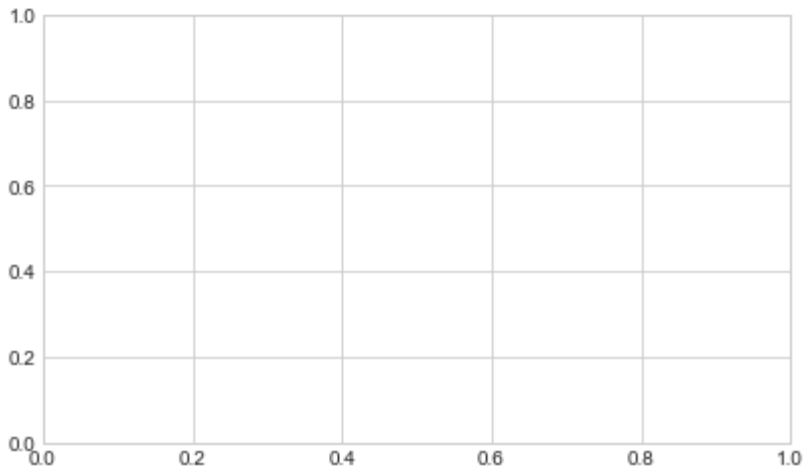

Saturday 15:00 (Price 0.672)
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Household Type
0
1
2
19.0
25.0
Temperature (C°)

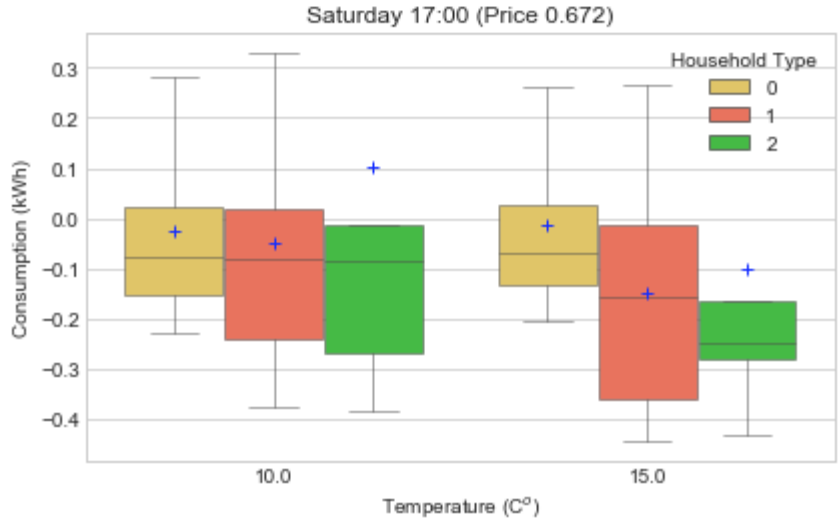

1.0
0.8
0.6
0.4
0.2

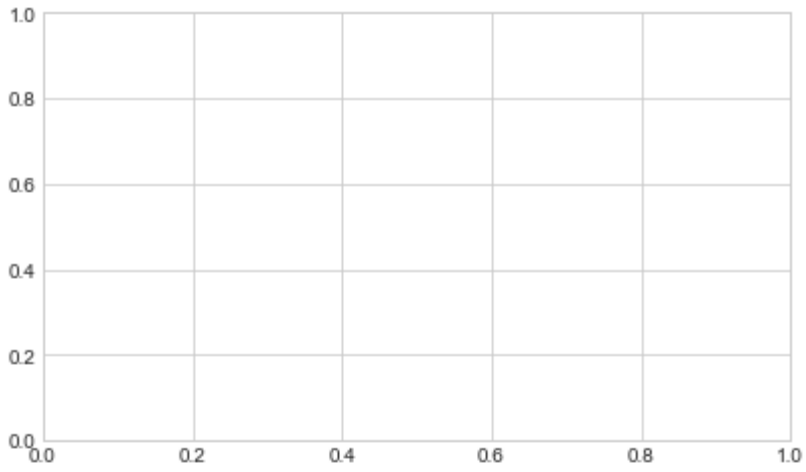

Saturday 16:00 (Price 0.672)
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
Household Type
0
1
2

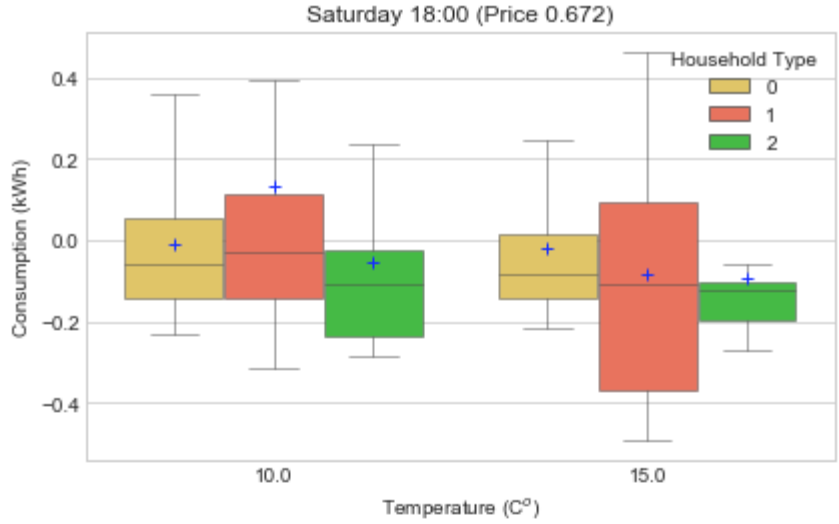

0.0
0.2
0.4
0.6
0.8
1.0

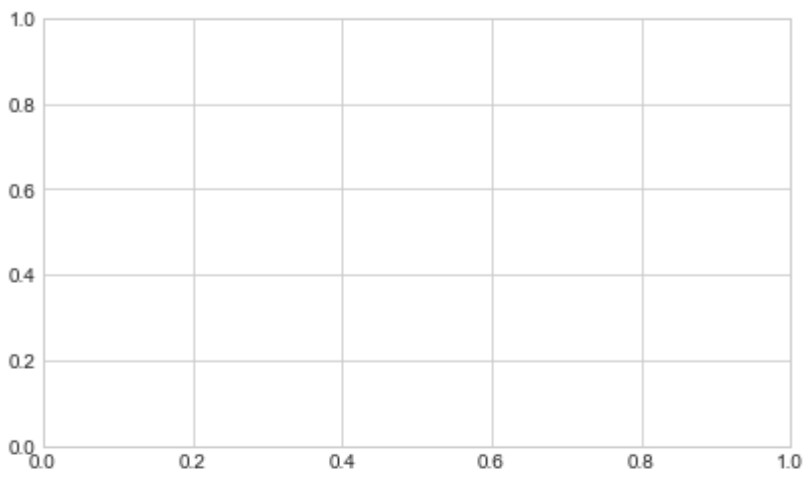

−0.5
9.0
15.0
Temperature (C°)

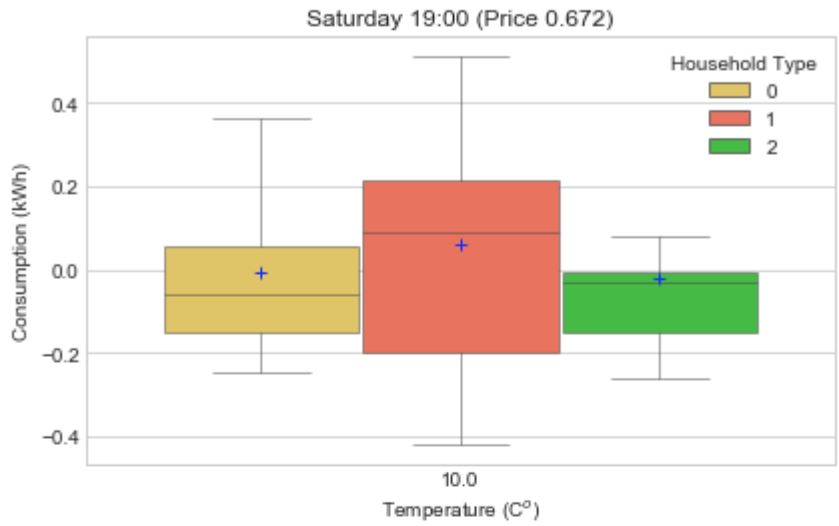

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

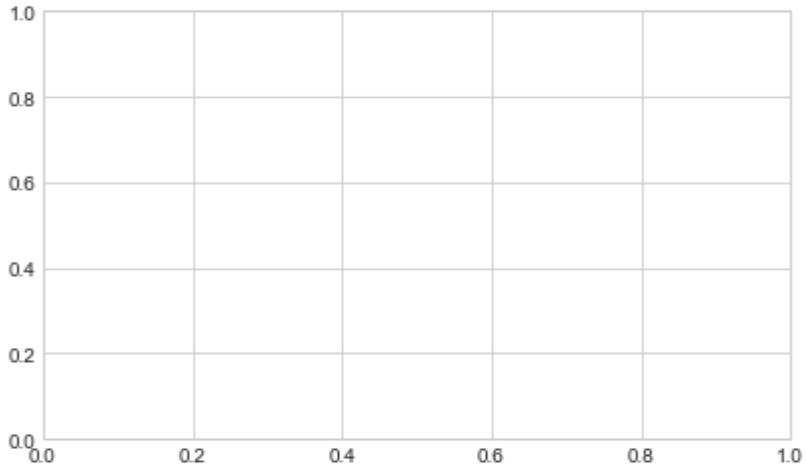

Saturday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
10.0
15.0
Temperature (C°)

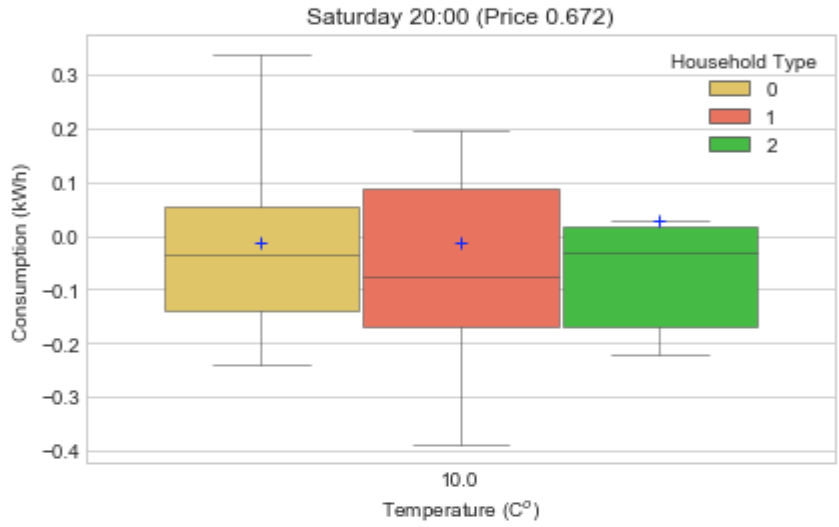

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

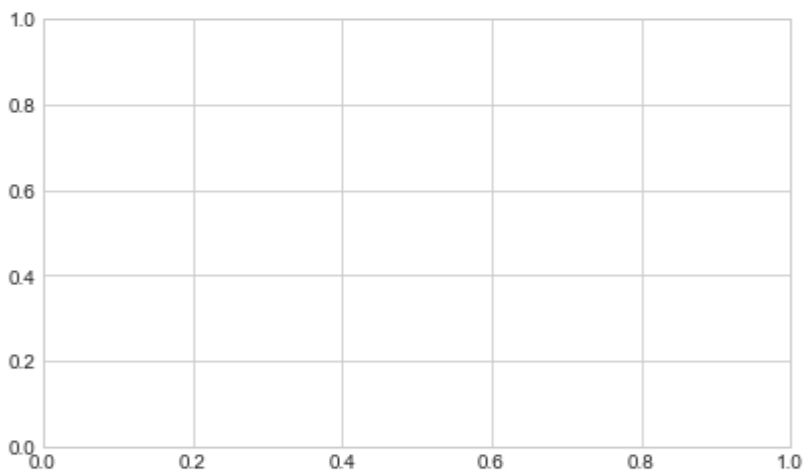

Saturday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
10.0
15.0
Temperature (C°)

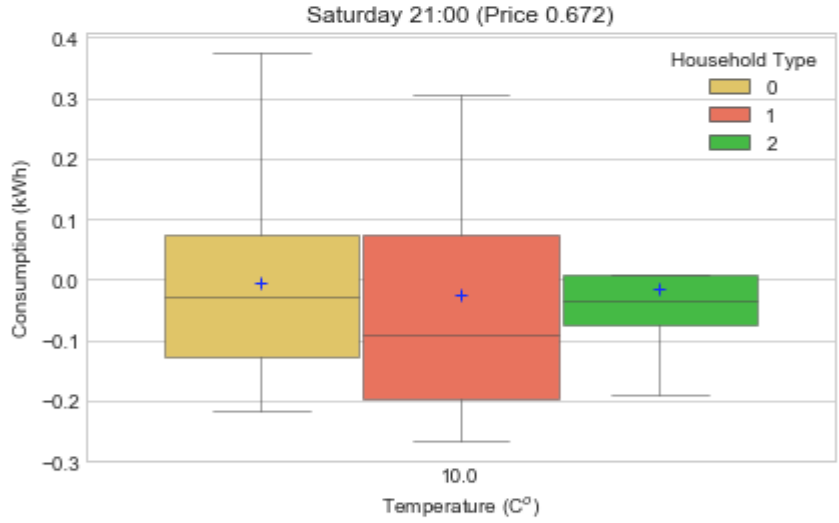

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Saturday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
10.0
Temperature (C°)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

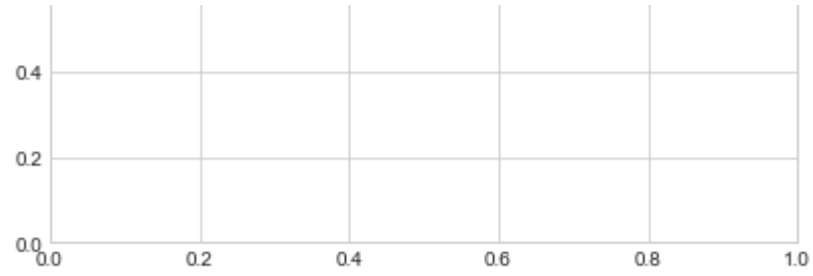

Saturday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
10.0
Temperature (C°)

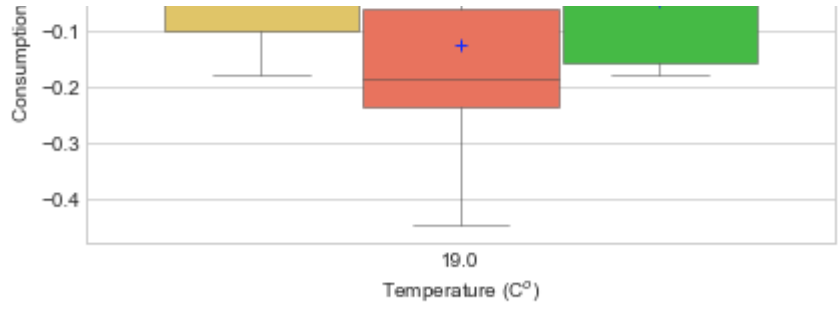

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

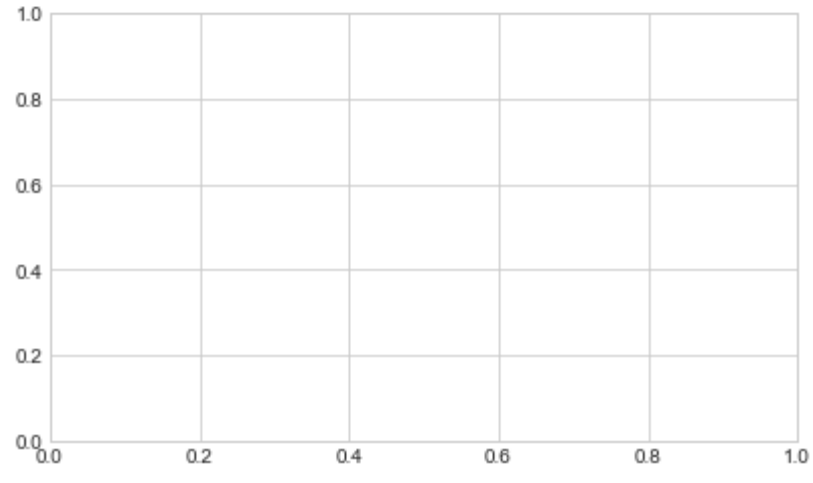

Saturday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
10.0
Temperature (C°)

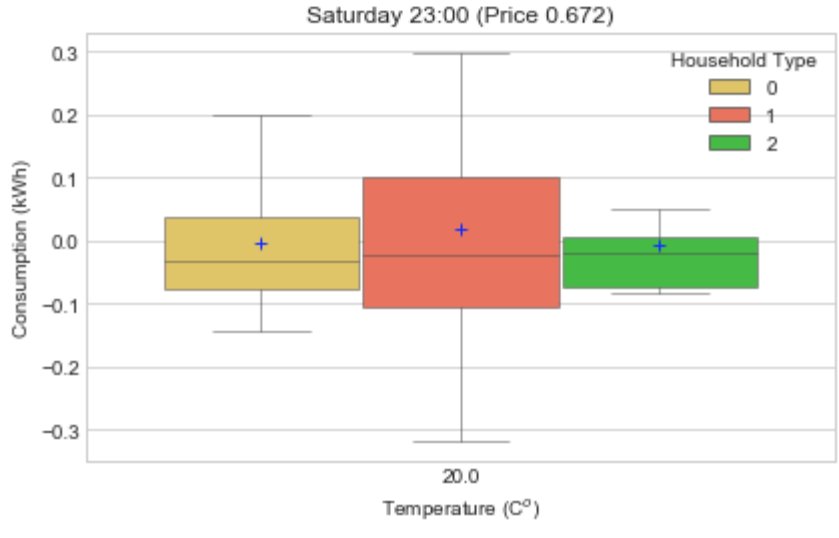
1.0
0.8
0.6

Saturday 22:00 (Price 0.672)
Household Type
0
1
2
0.2
0.1
0.0

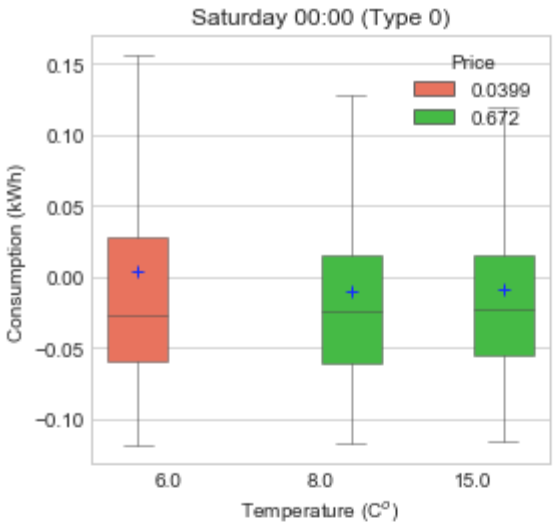

0.0
0.2
0.4
0.6
0.8
1.0

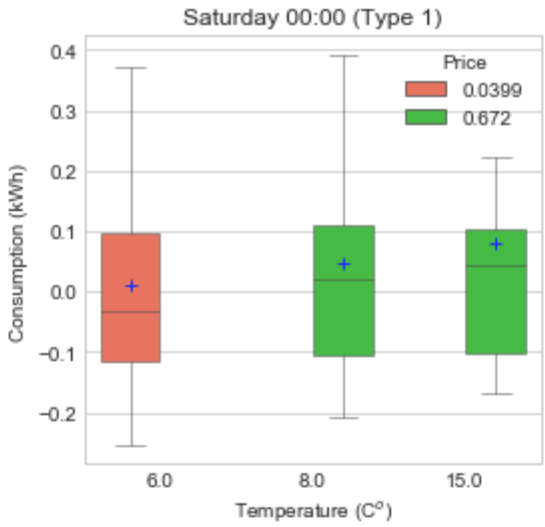

Consumption
−0.1
−0.2
−0.3
−0.4
19.0
Temperature (C°)

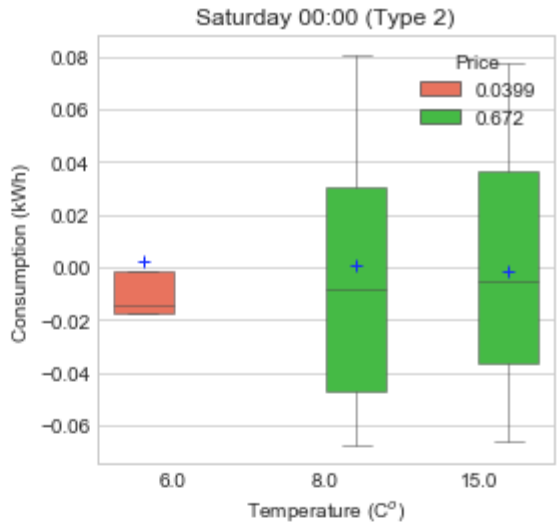

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

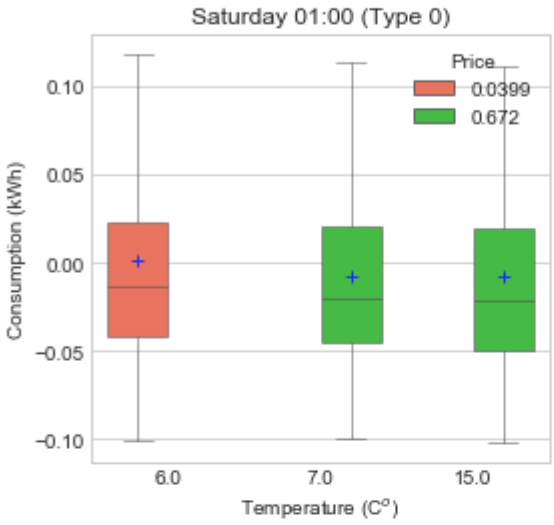

Saturday 23:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
20.0
Temperature (C°)

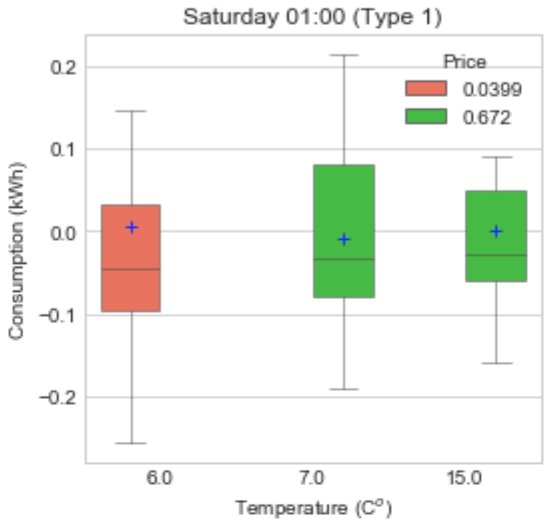
In [72]:
```
# price comparison
# Saturday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 5 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

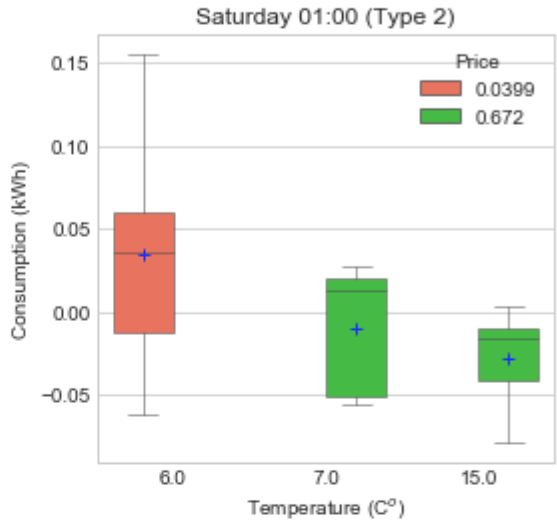

Saturday 00:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
6.0
8.0
15.0
Temperature (C°)

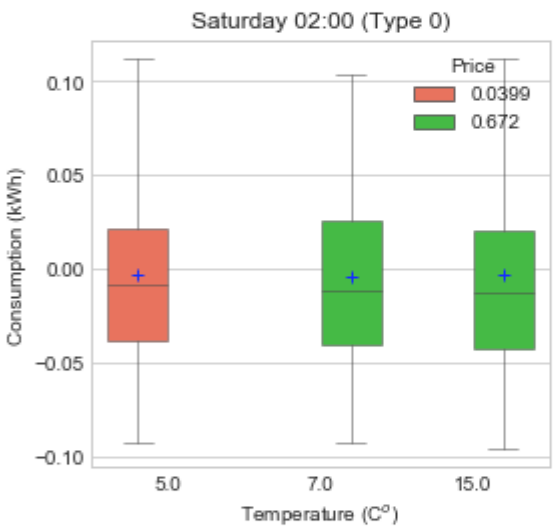

Saturday 00:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
6.0
8.0
15.0
Temperature (C°)

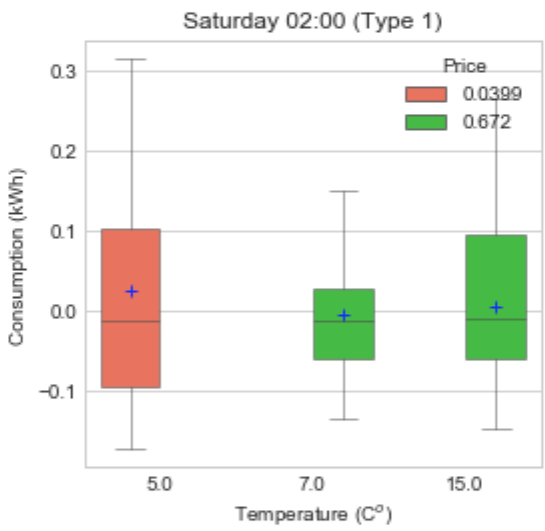

Saturday 00:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
6.0
8.0
15.0
Temperature (C°)

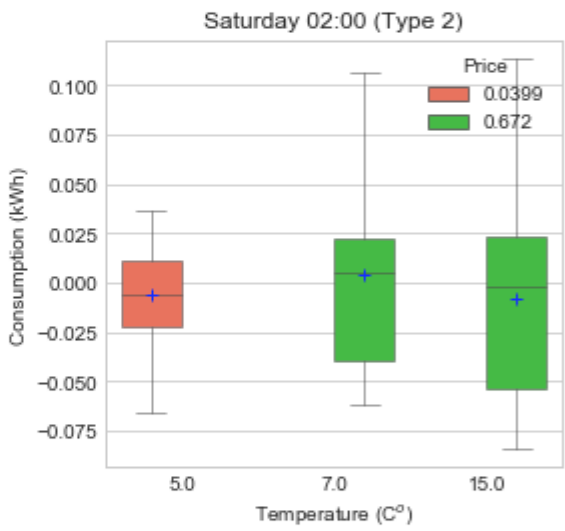

Saturday 01:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
6.0
7.0
15.0
Temperature (C°)

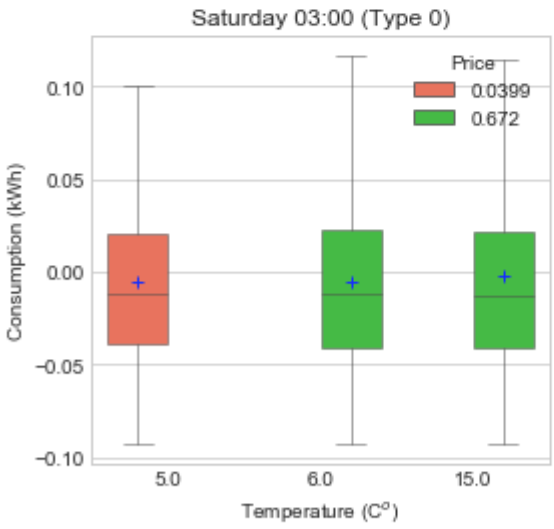

Saturday 01:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
6.0
7.0
15.0
Temperature (C°)

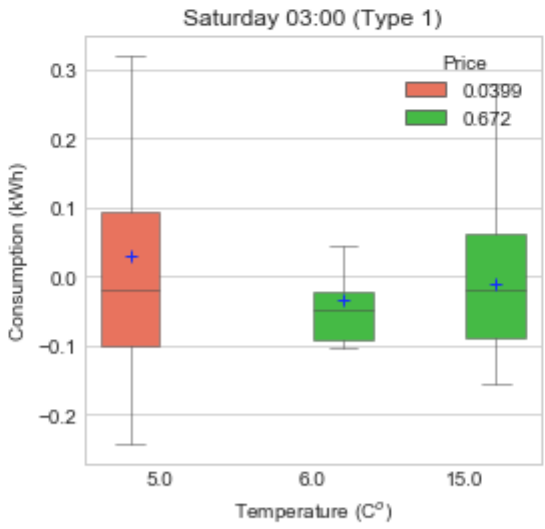

Saturday 01:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
6.0
7.0
15.0
Temperature (C°)

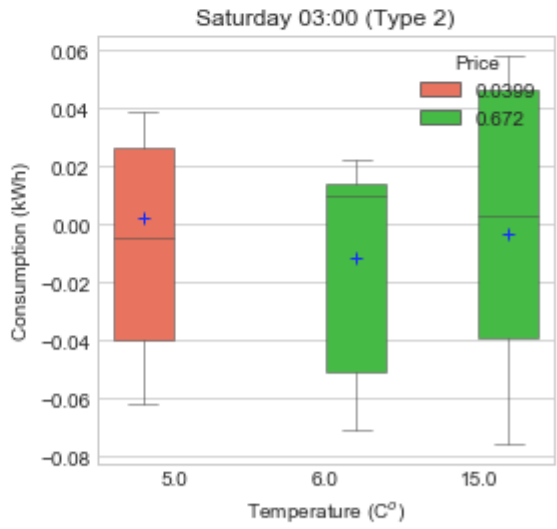

Saturday 02:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
5.0
7.0
15.0
Temperature (C°)

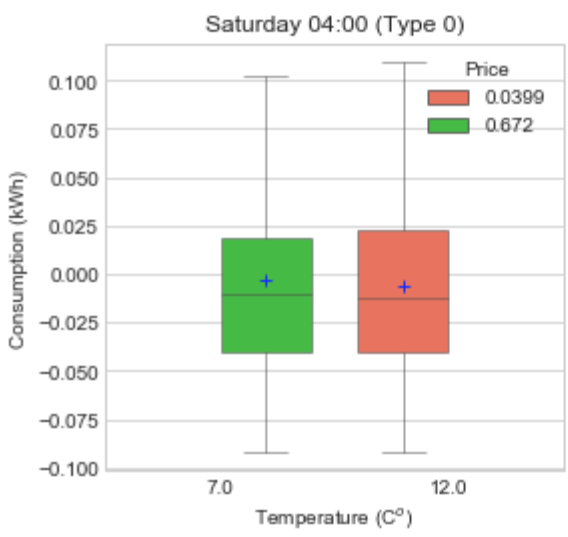

Saturday 02:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
5.0
7.0
15.0
Temperature (C°)

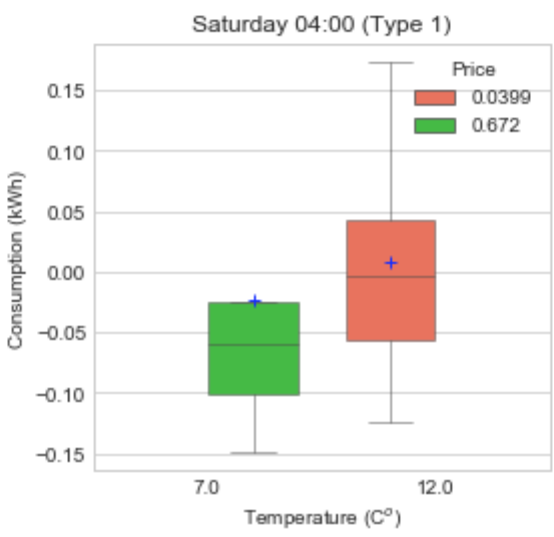

Saturday 02:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
5.0
7.0
15.0
Temperature (C°)

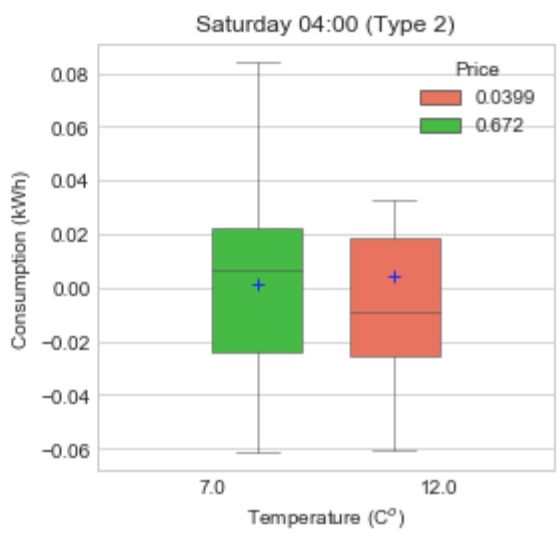

Saturday 03:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
5.0
6.0
15.0
Temperature (C°)

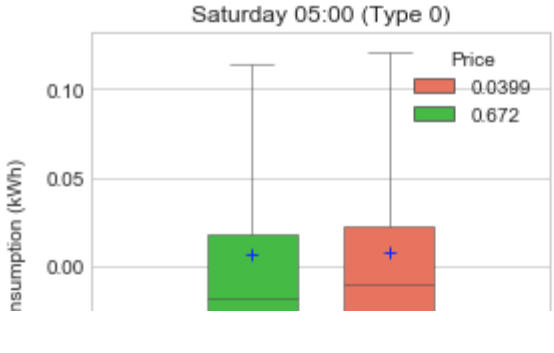

Saturday 03:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
5.0
6.0
15.0
Temperature (C°)

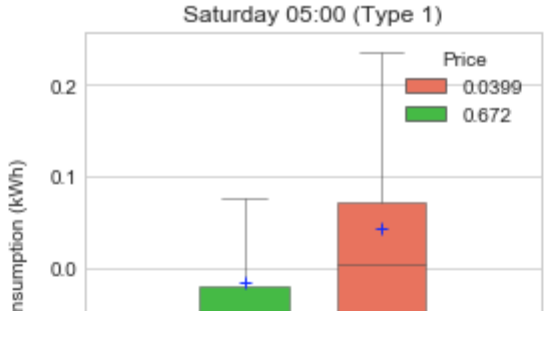

Saturday 03:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
5.0
6.0
15.0
Temperature (C°)

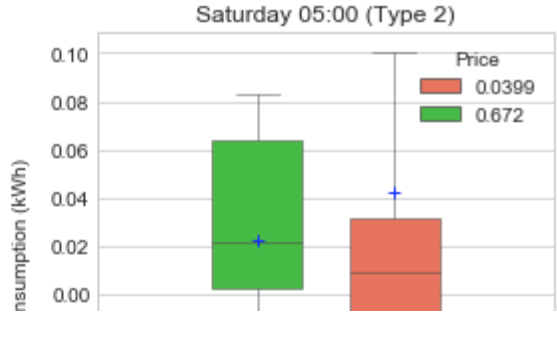

Saturday 04:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
7.0
12.0
Temperature (C°)

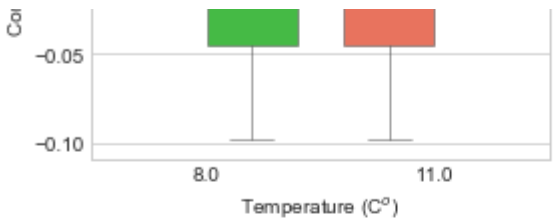

Saturday 04:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
7.0
12.0
Temperature (C°)

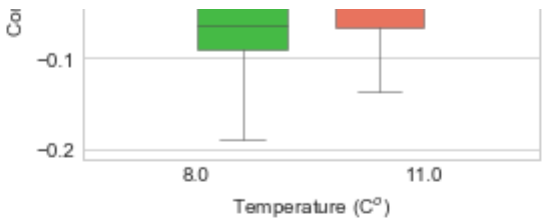

Saturday 04:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
7.0
12.0
Temperature (C°)

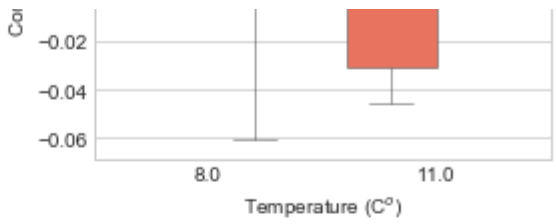

Saturday 05:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
Temperature (C°)

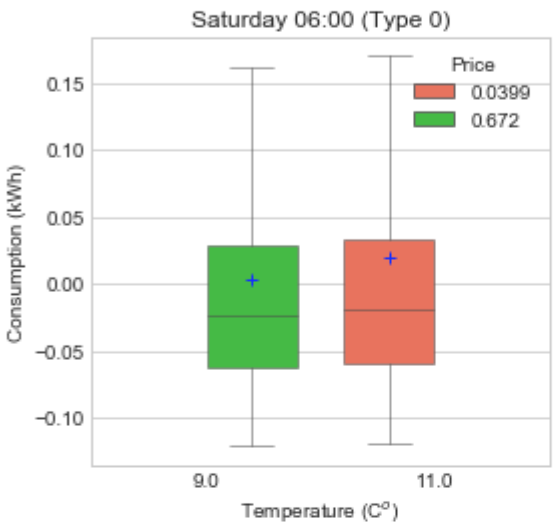

Saturday 05:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
Temperature (C°)

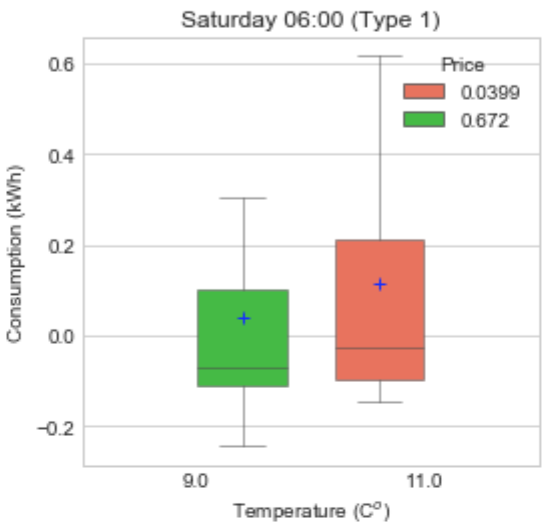

Saturday 05:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.08
0.06
0.04
0.02
0.00
Temperature (C°)

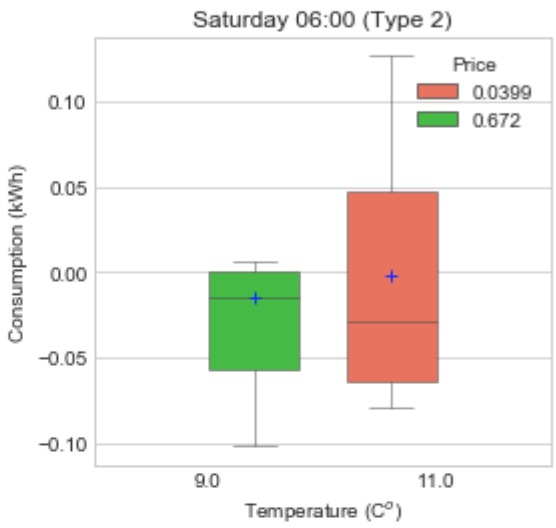

−0.05
−0.10
8.0
11.0
Temperature (C°)

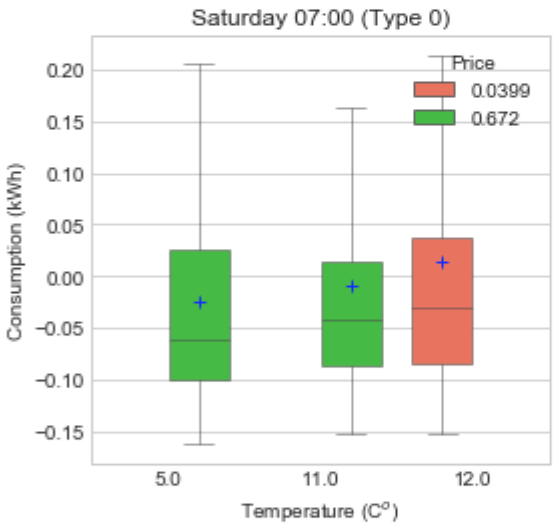

−0.1
−0.2
8.0
11.0
Temperature (C°)

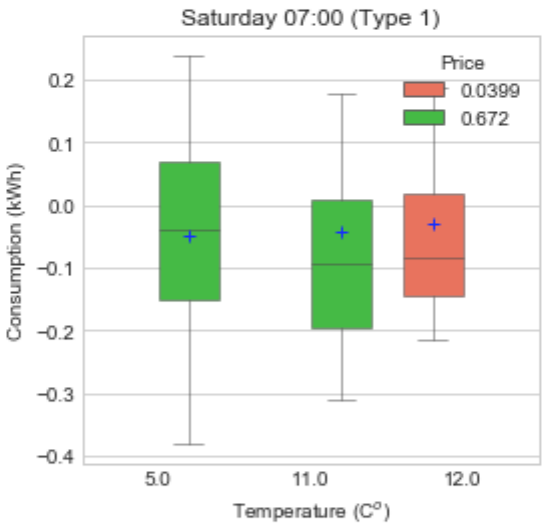

−0.02
−0.04
−0.06
8.0
11.0
Temperature (C°)

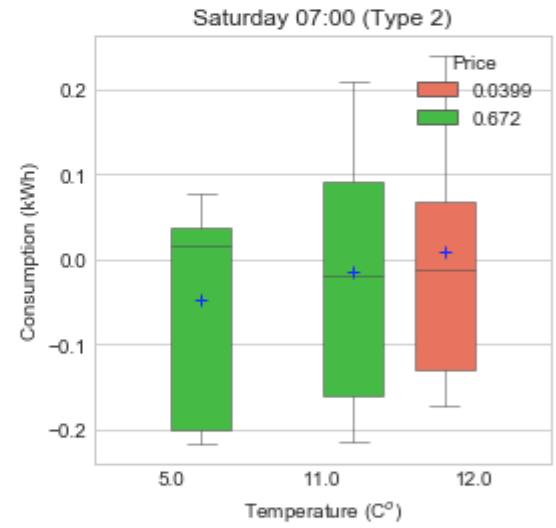

Saturday 06:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
9.0
11.0
Temperature (C°)

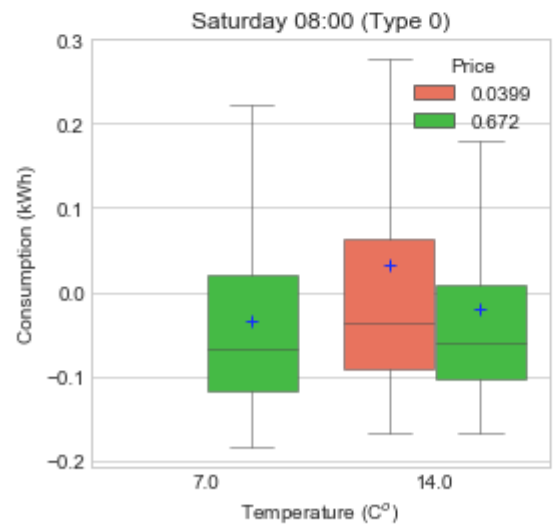

Saturday 06:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
9.0
11.0
Temperature (C°)

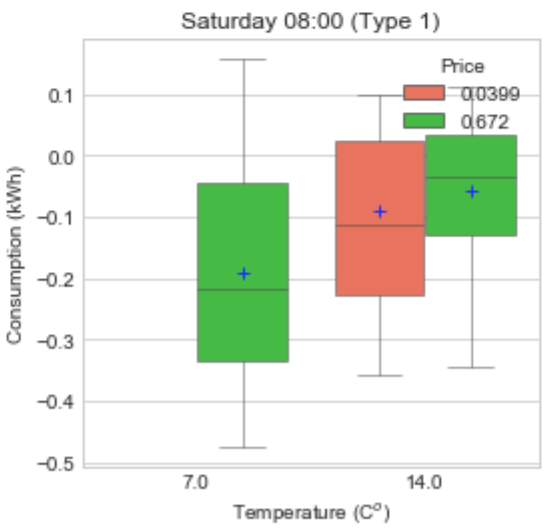

Saturday 06:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.10
0.05
0.00
−0.05
−0.10
9.0
11.0
Temperature (C°)

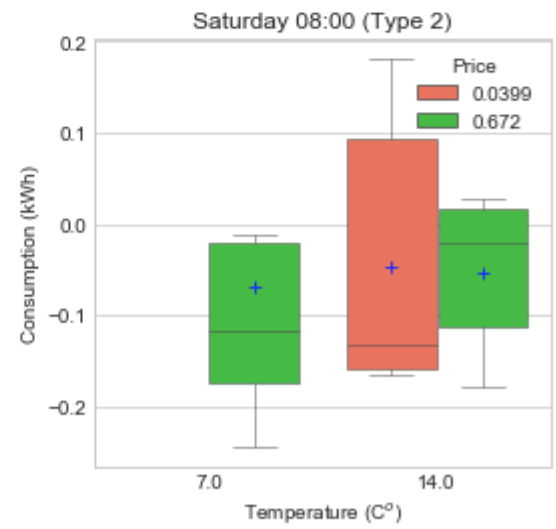

Saturday 07:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
5.0
11.0
12.0
Temperature (C°)

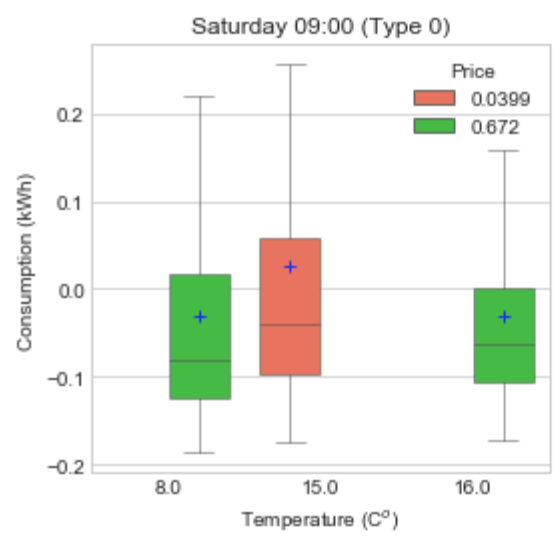

Saturday 07:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
5.0
11.0
12.0
Temperature (C°)

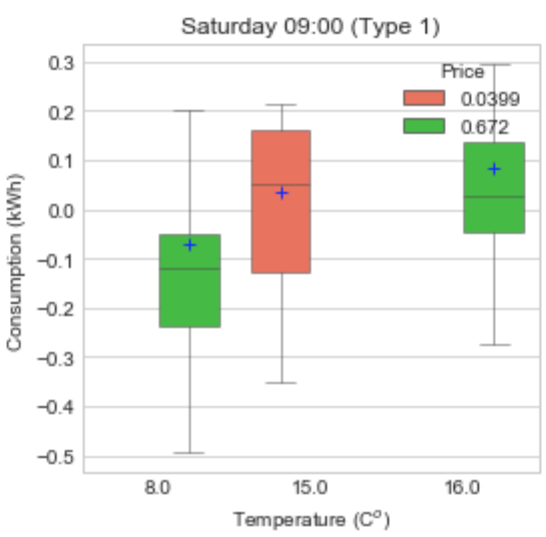

Saturday 07:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
5.0
11.0
12.0
Temperature (C°)

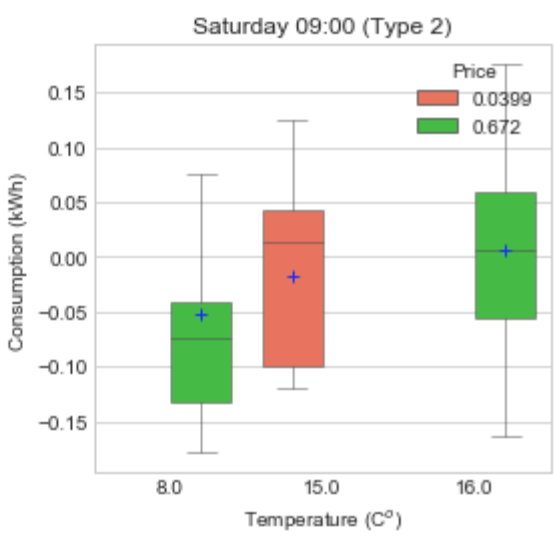

Saturday 08:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0
14.0
Temperature (C°)

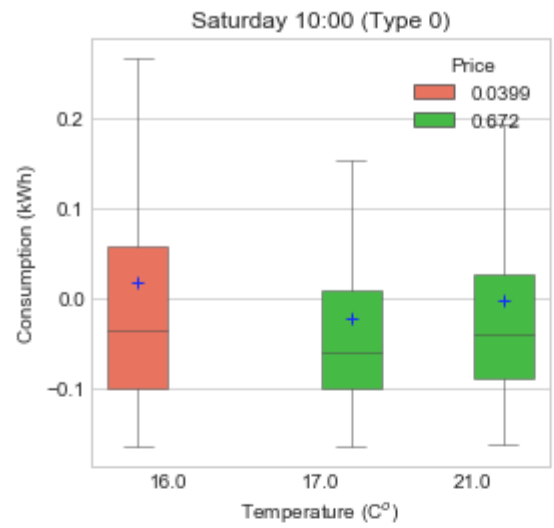

Saturday 08:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
7.0
14.0
Temperature (C°)

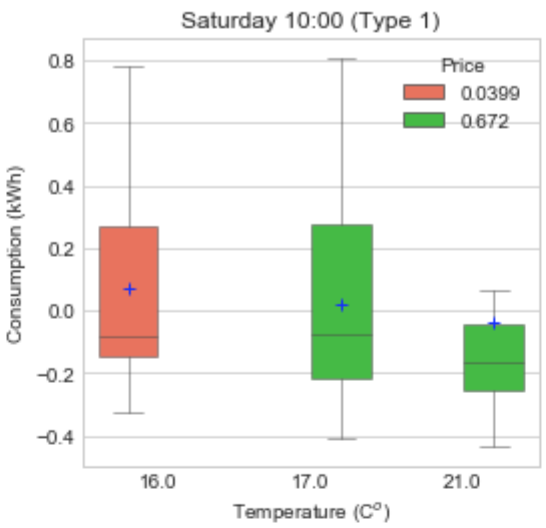

Saturday 08:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
7.0
14.0
Temperature (C°)

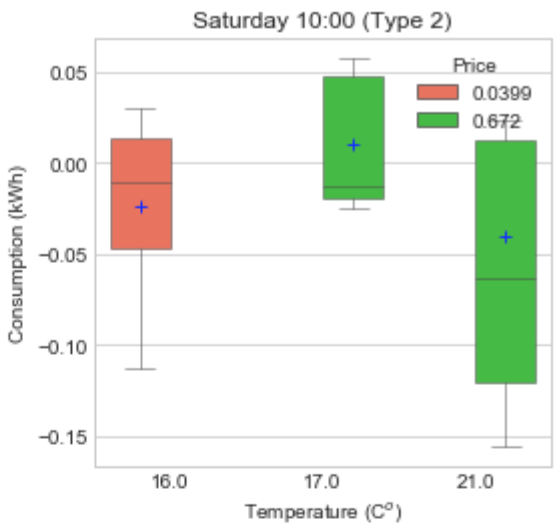

Saturday 09:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
−0.2
8.0
15.0
16.0
Temperature (C°)


Saturday 09:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
8.0
15.0
16.0
Temperature (C°)

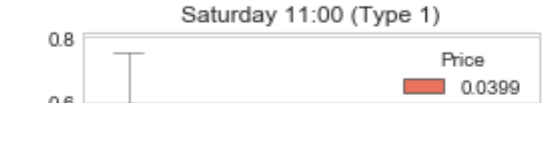

Saturday 09:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
8.0
15.0
16.0
Temperature (C°)


Saturday 10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.2
0.1
0.0
−0.1
16.0
17.0
21.0
Temperature (C°)

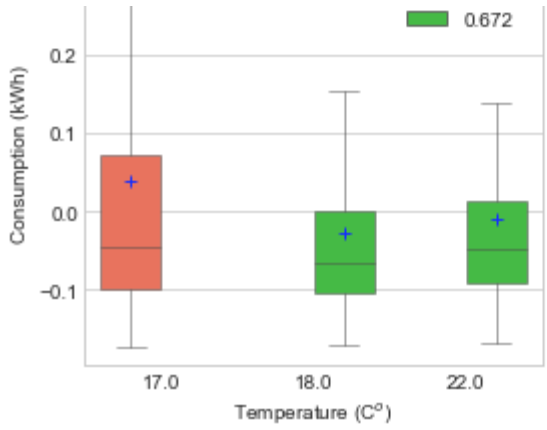

Saturday 10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
16.0
17.0
21.0
Temperature (C°)

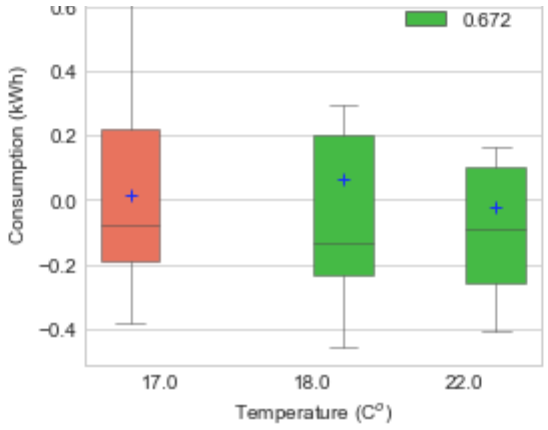

Saturday 10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.05
0.00
−0.05
−0.10
−0.15
16.0
17.0
21.0
Temperature (C°)

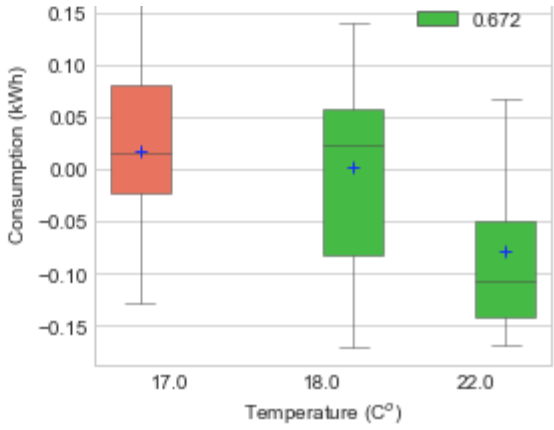

Saturday 11:00 (Type 0)
Price
0.0399
0.3

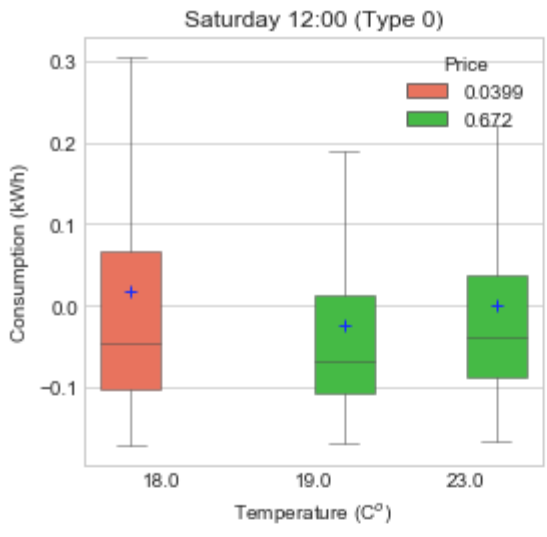

Saturday 11:00 (Type 1)
Price
0.0399
0.8

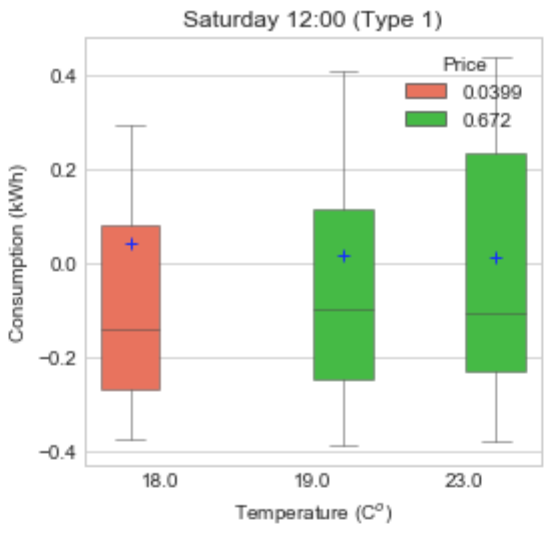

Saturday 11:00 (Type 2)
Price
0.0399
0.20

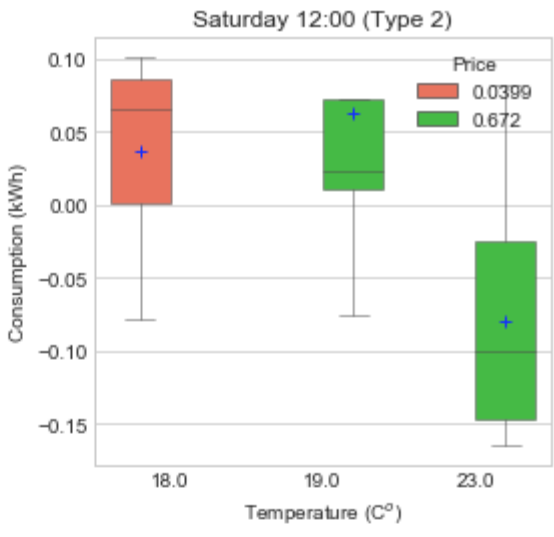

0.672
Consumption (kWh)
Temperature (C°)
17.0
18.0
22.0

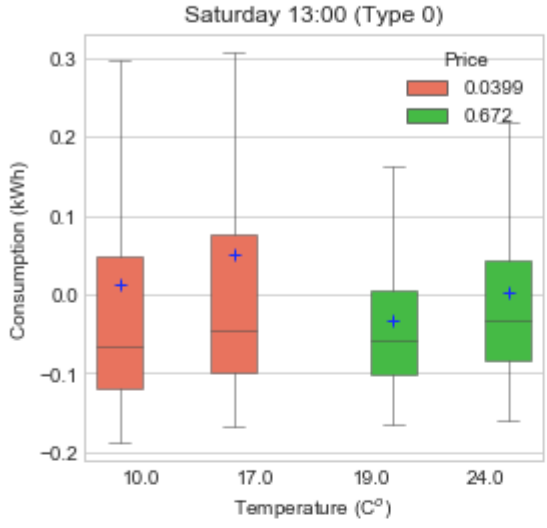

0.672
Consumption (kWh)
Temperature (C°)
17.0
18.0
22.0

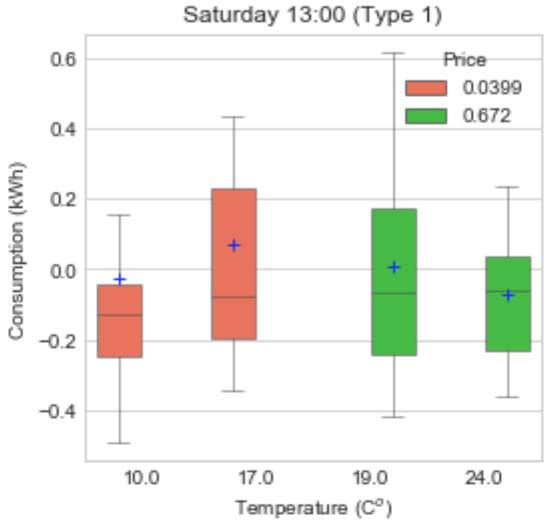

0.672
Consumption (kWh)
Temperature (C°)
17.0
18.0
22.0

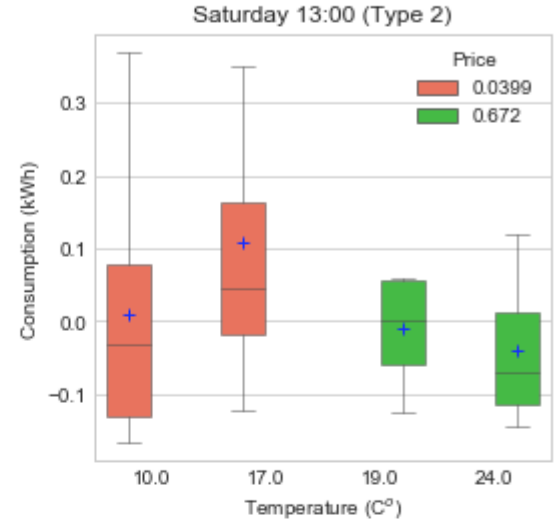

Saturday 12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
18.0
19.0
23.0

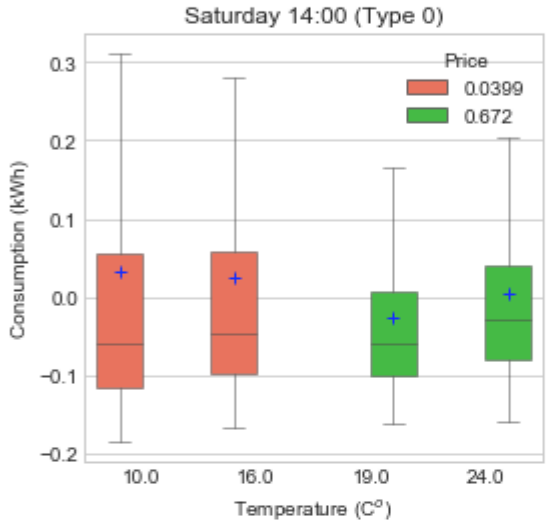

Saturday 12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
18.0
19.0
23.0

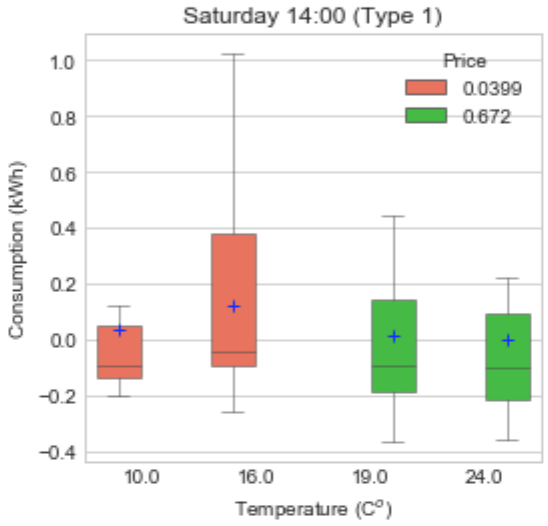

Saturday 12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
18.0
19.0
23.0

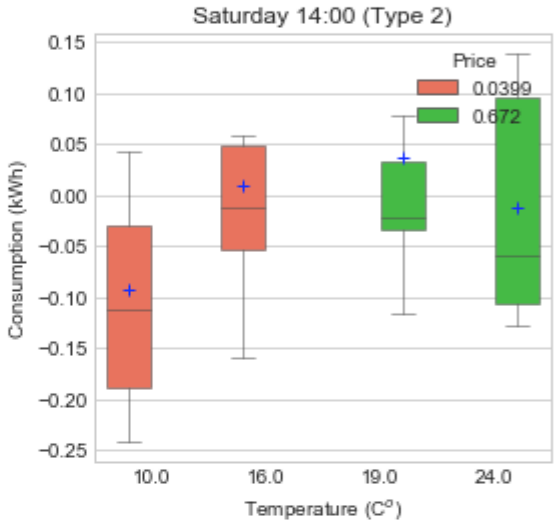

Saturday 13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
19.0
24.0

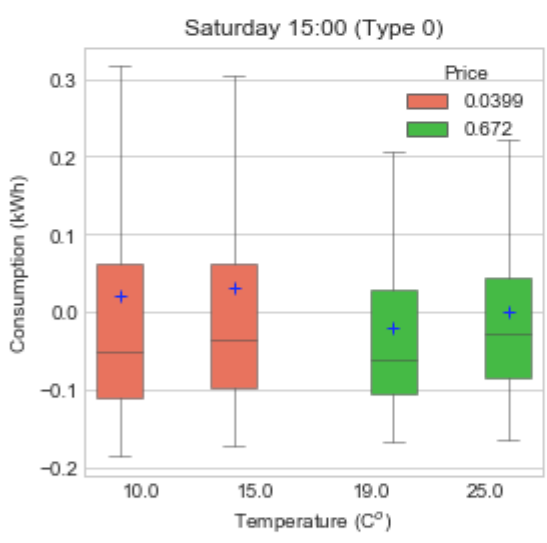

Saturday 13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
19.0
24.0

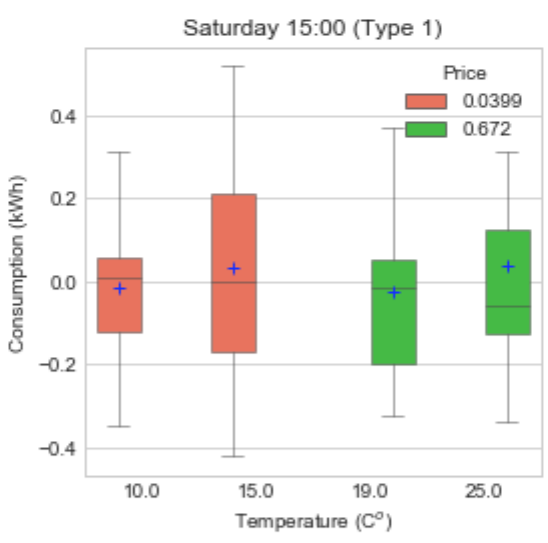

Saturday 13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
17.0
19.0
24.0

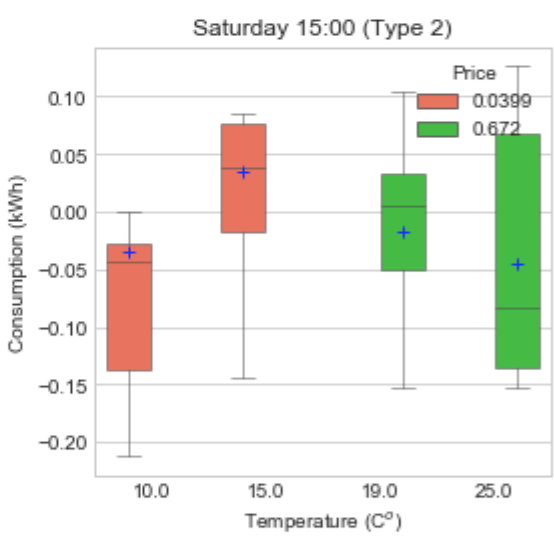

Saturday 14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
16.0
19.0
24.0

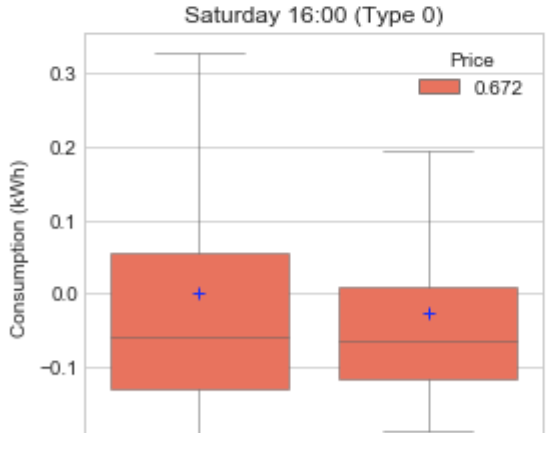

Saturday 14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
16.0
19.0
24.0

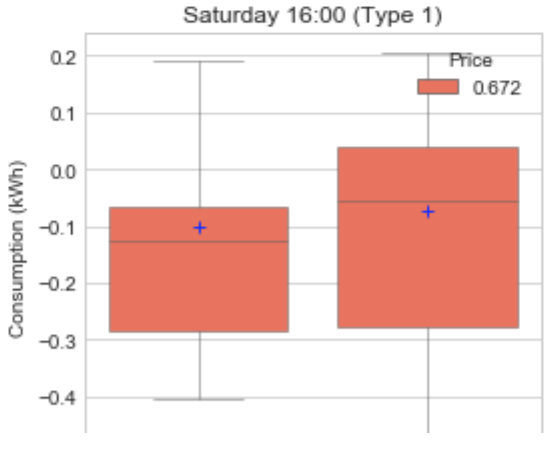

Saturday 14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
16.0
19.0
24.0

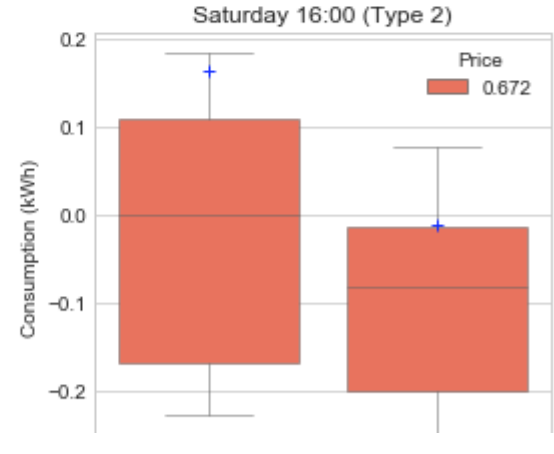

Saturday 15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
19.0
25.0


Saturday 15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
19.0
25.0


Saturday 15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
19.0
25.0


Saturday 16:00 (Type 0)
Price
0.672
Consumption (kWh)

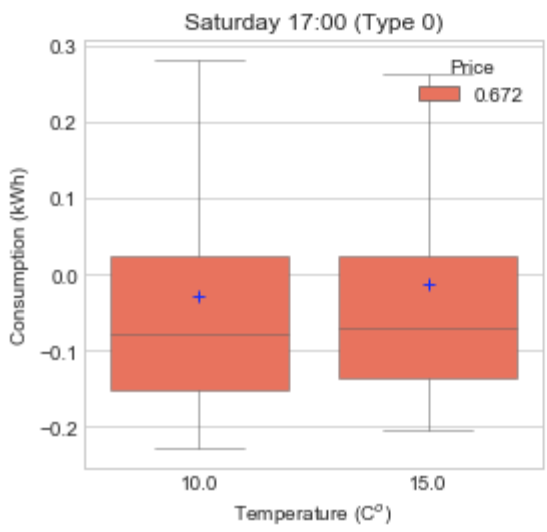

Saturday 16:00 (Type 1)
Price
0.672
Consumption (kWh)

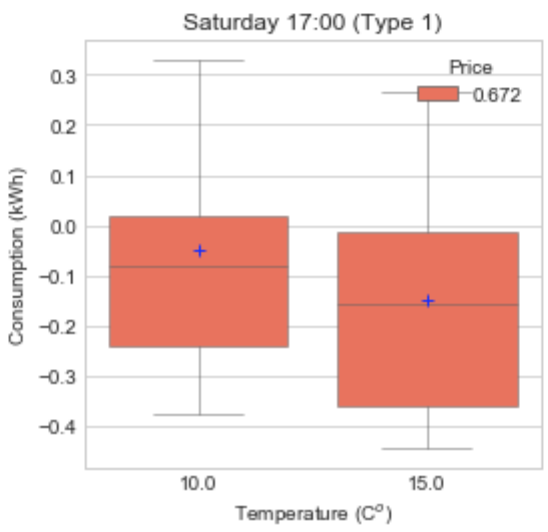

Saturday 16:00 (Type 2)
Price
0.672
Consumption (kWh)

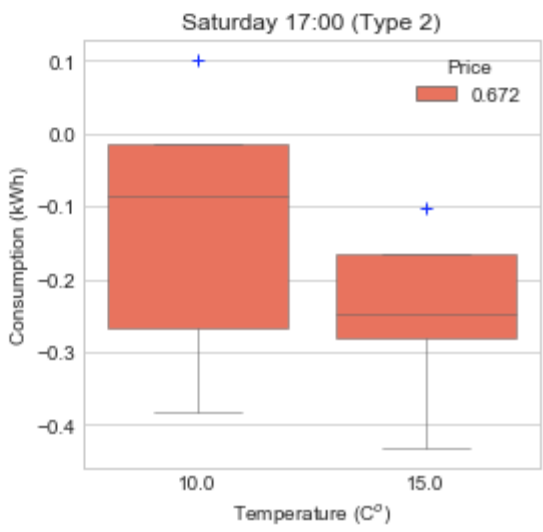

9.0
15.0
Temperature (C°)

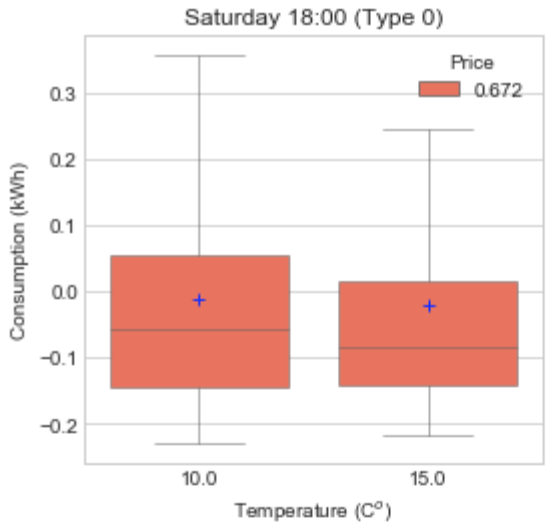

9.0
15.0
Temperature (C°)

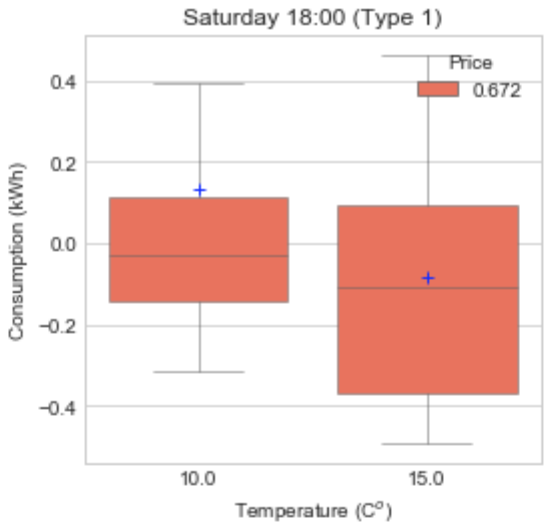

9.0
15.0
Temperature (C°)

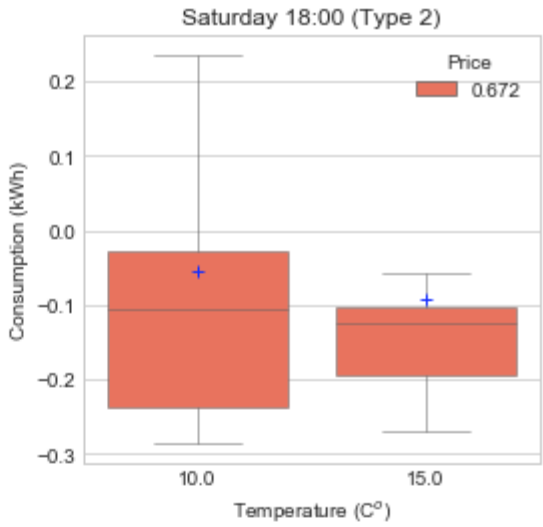

Saturday 17:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

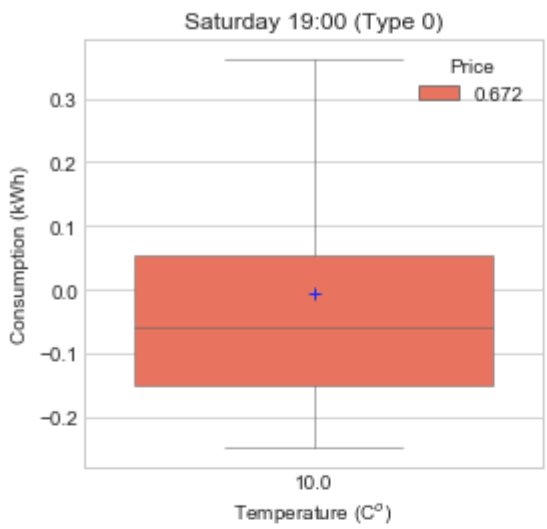

Saturday 17:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

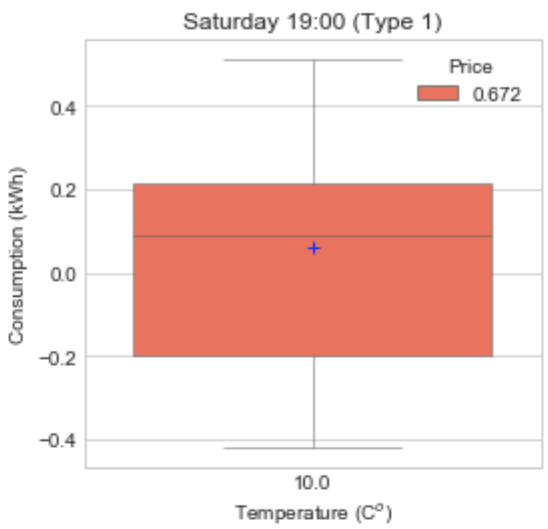

Saturday 17:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

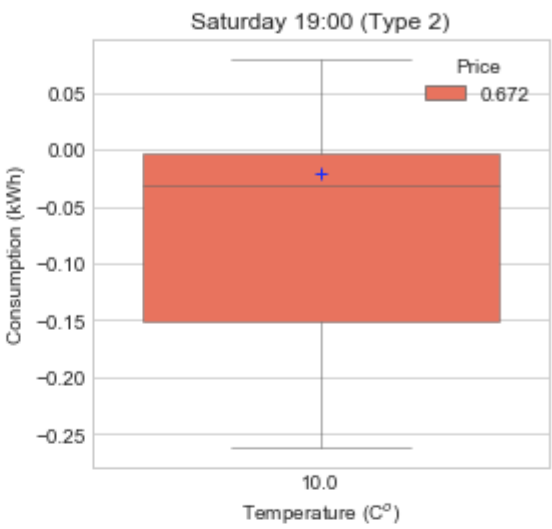

Saturday 18:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

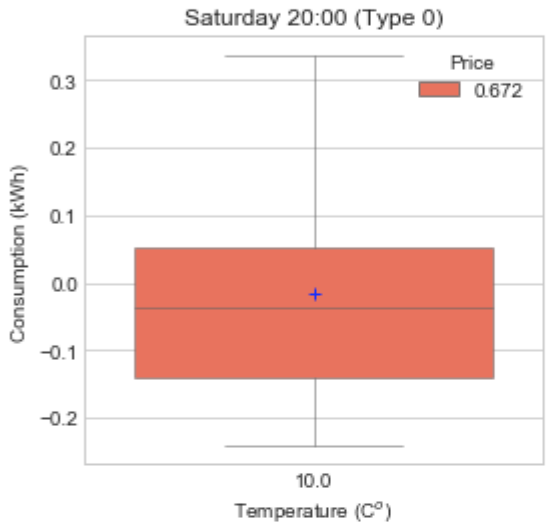

Saturday 18:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

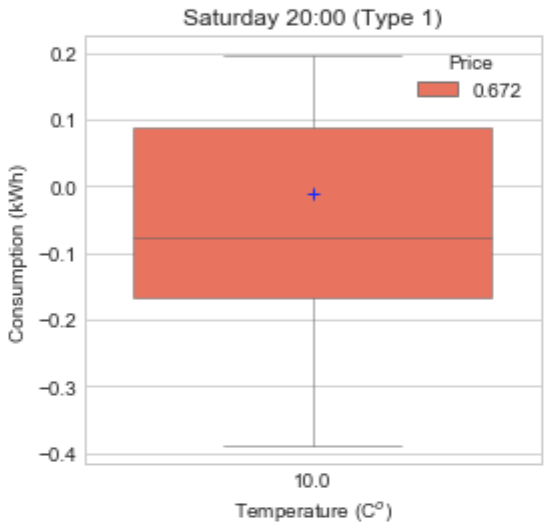

Saturday 18:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
15.0
Temperature (C°)

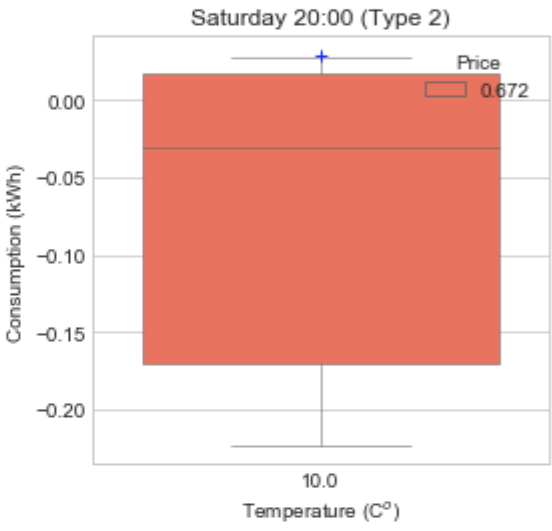

Saturday 19:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

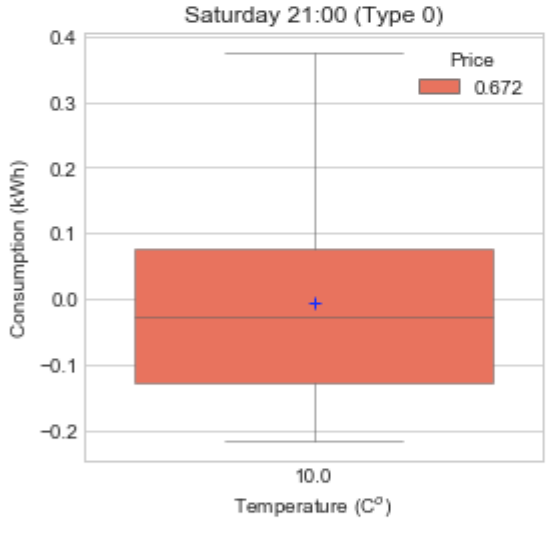

Saturday 19:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

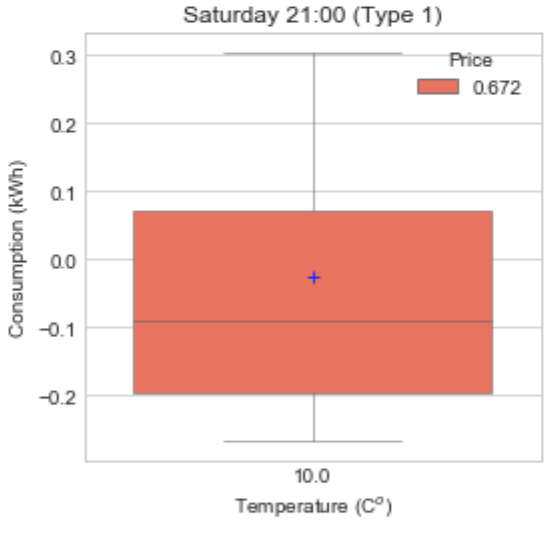

Saturday 19:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

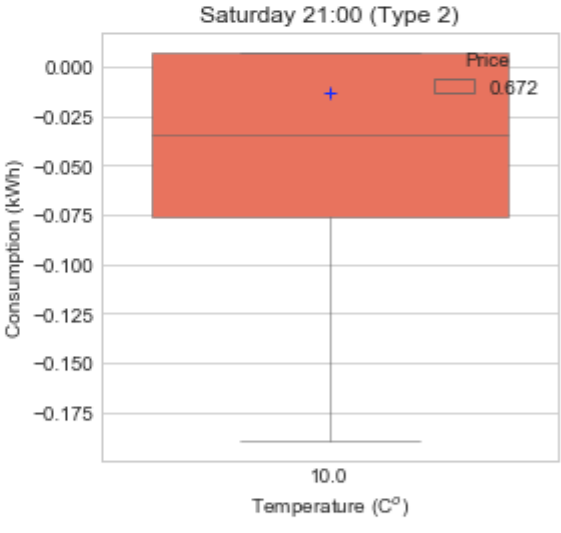

Saturday 20:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)


Saturday 20:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

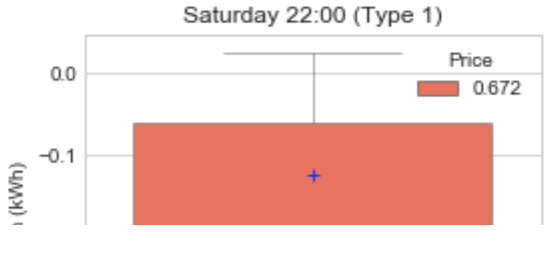

Saturday 20:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)


Saturday 21:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

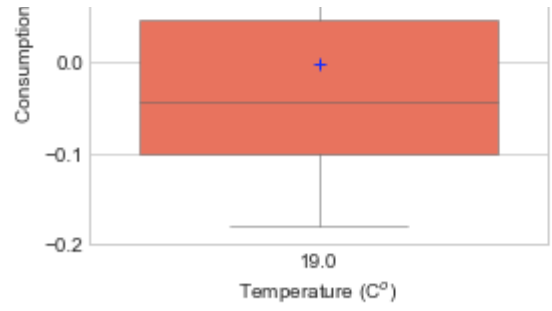

Saturday 21:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

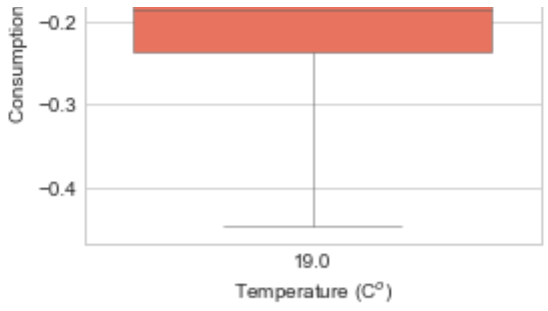

Saturday 21:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
Temperature (C°)

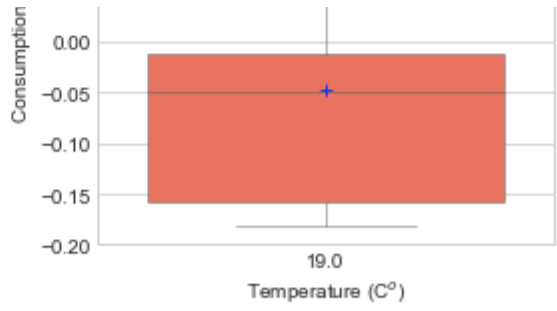

Saturday 22:00 (Type 0)
Price
0.672

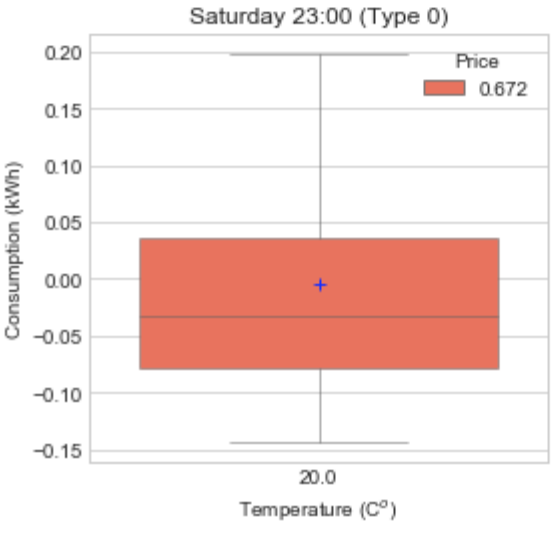

Saturday 22:00 (Type 1)
Price
0.672

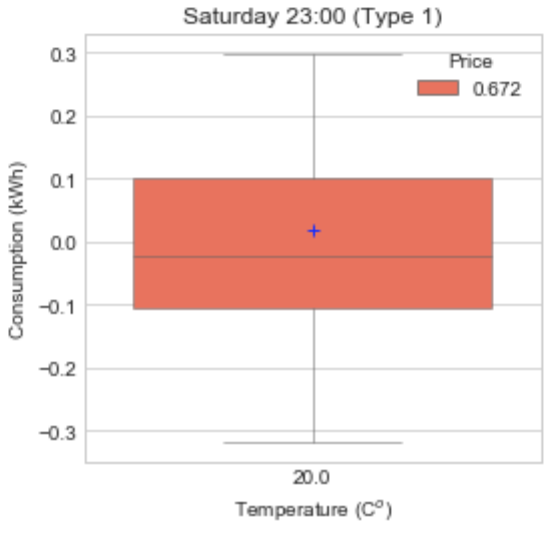

Saturday 22:00 (Type 2)
Price
0.672

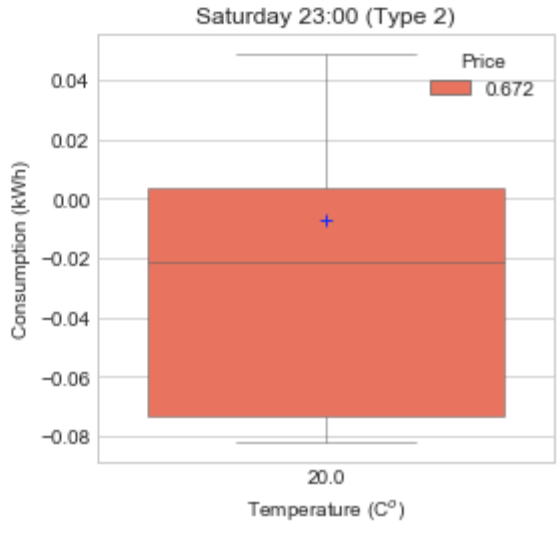

Consumption
0.0
−0.1
−0.2
19.0
Temperature (C°)


Consumption
−0.2
−0.3
−0.4
19.0
Temperature (C°)

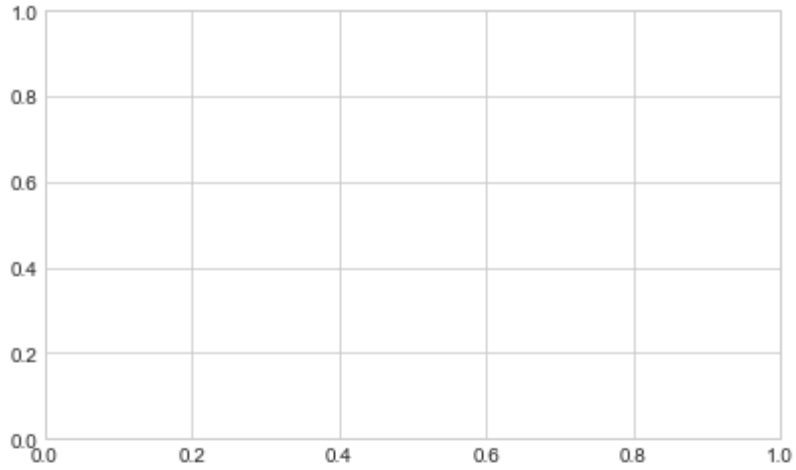

Consumption
0.00
−0.05
−0.10
−0.15
−0.20
19.0
Temperature (C°)

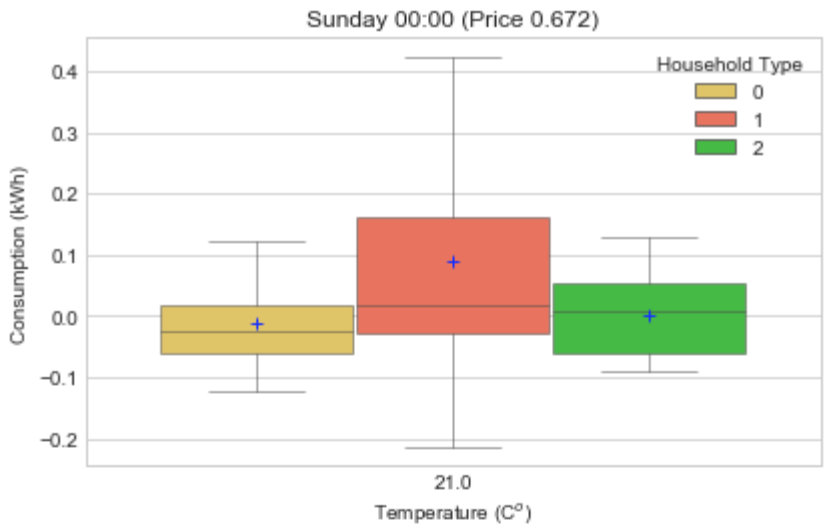

Saturday 23:00 (Type 0)
Price
0.672
Consumption (kWh)
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
20.0
Temperature (C°)

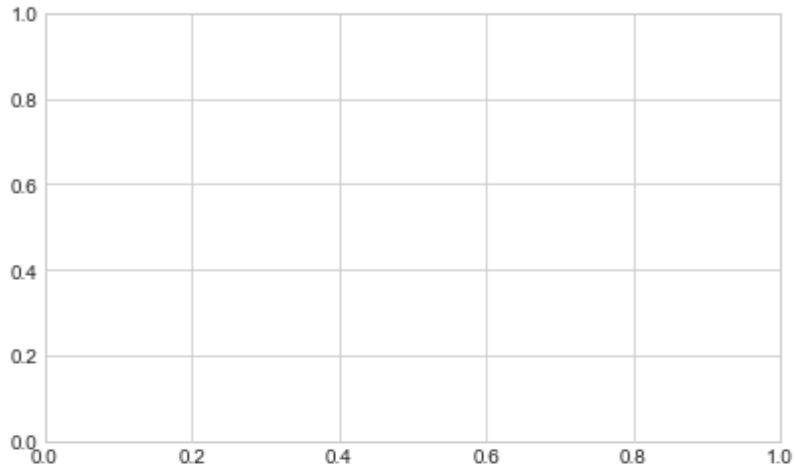

Saturday 23:00 (Type 1)
Price
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
20.0
Temperature (C°)

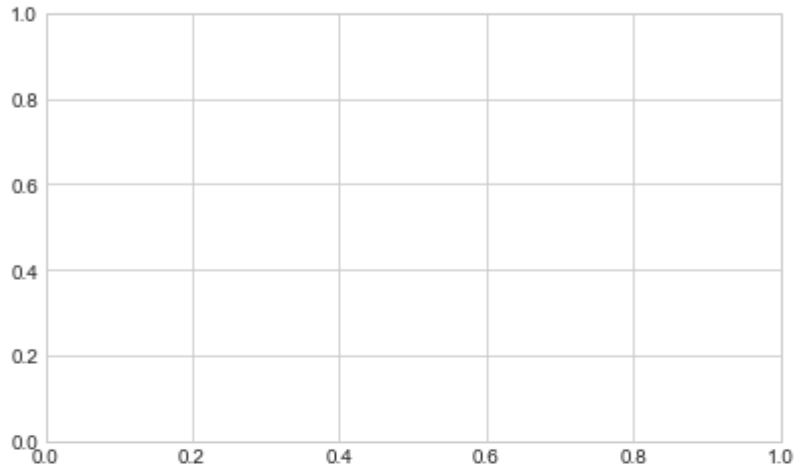

Saturday 23:00 (Type 2)
Price
0.672
Consumption (kWh)
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
20.0
Temperature (C°)

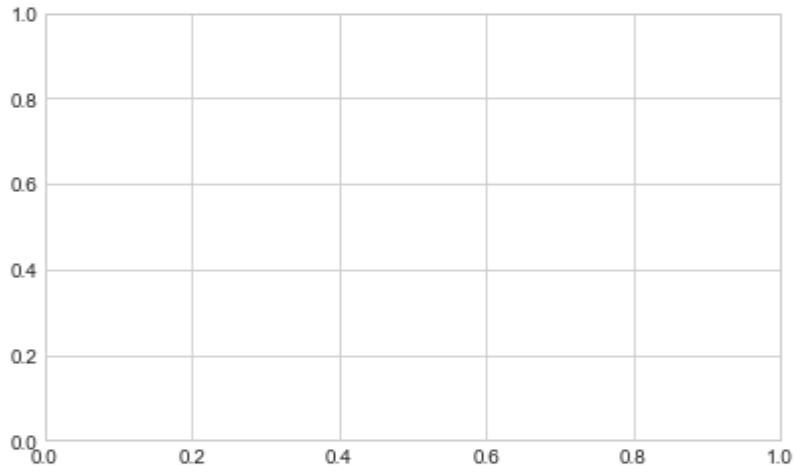
In [73]:
```
# Sunday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 6 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Price'] == prices[p])]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Price ' + str(prices[p]) +
')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day_hour, ax=ax_Ntou[-1
], palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+",
"markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Price ' + str(prices[p]) +
')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Household Type')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

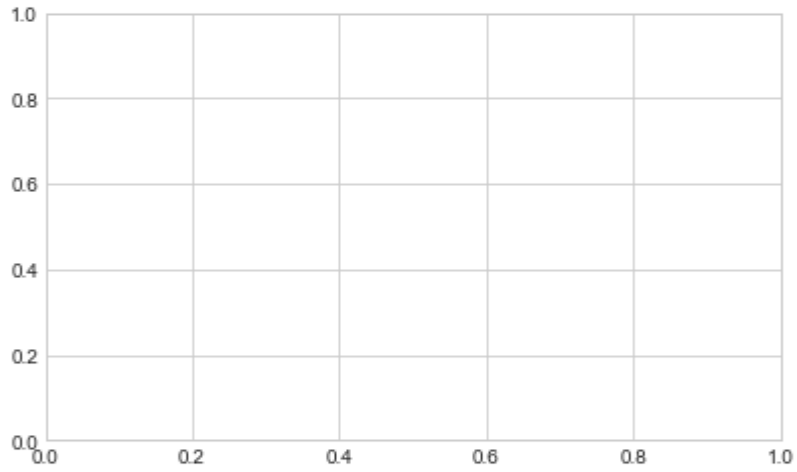


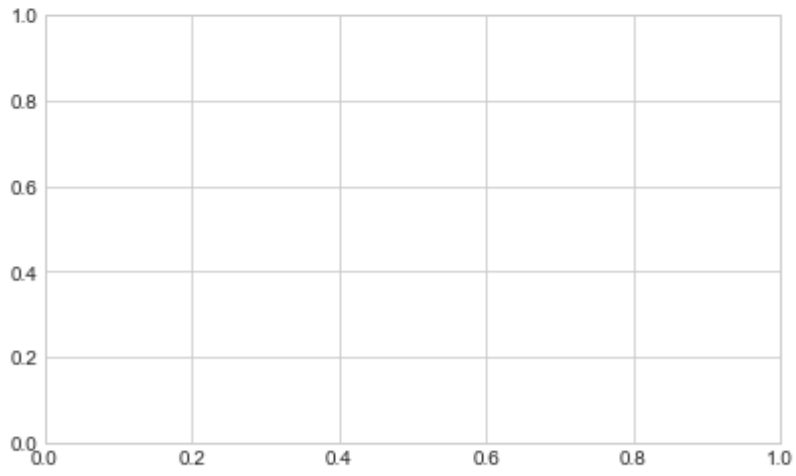

Sunday 00:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
21.0
Temperature (C°)

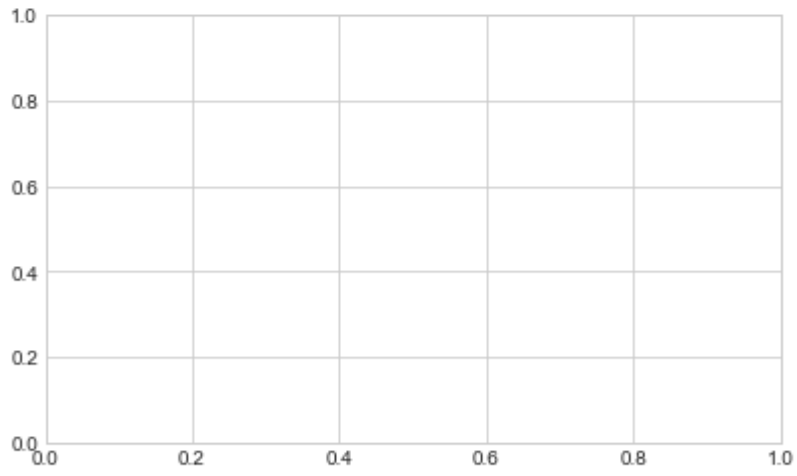


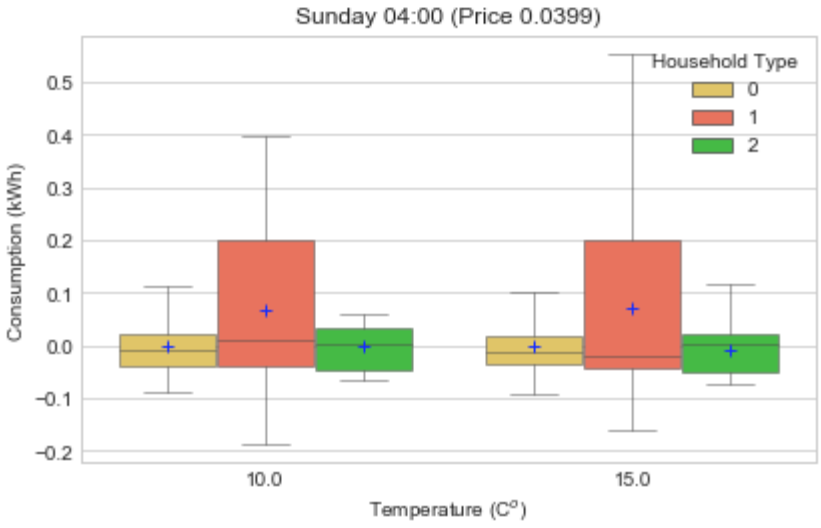


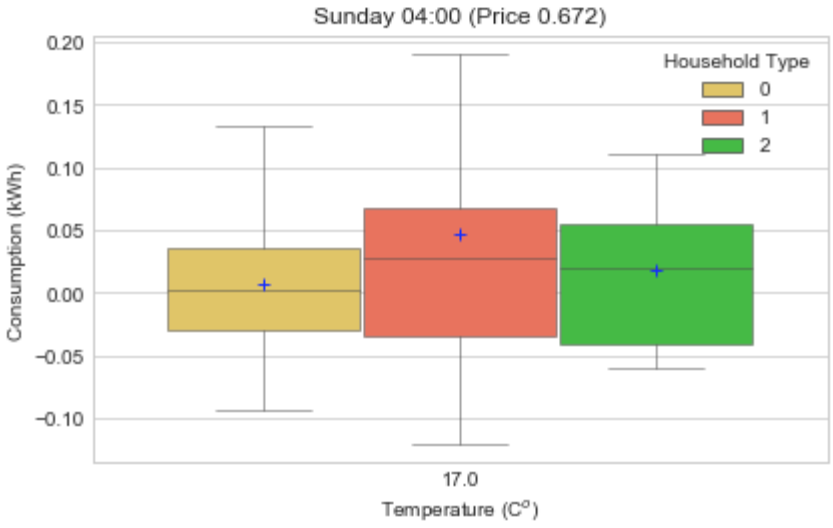




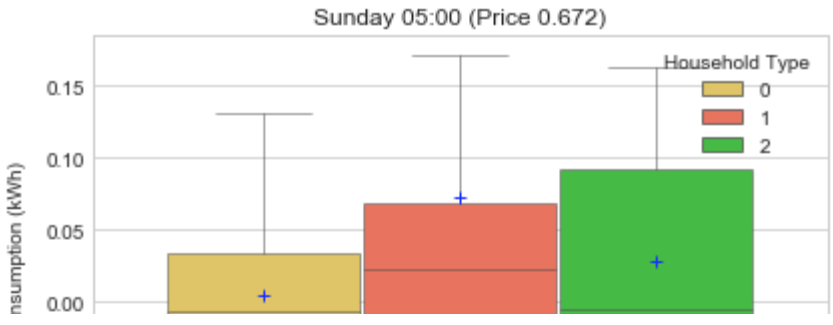


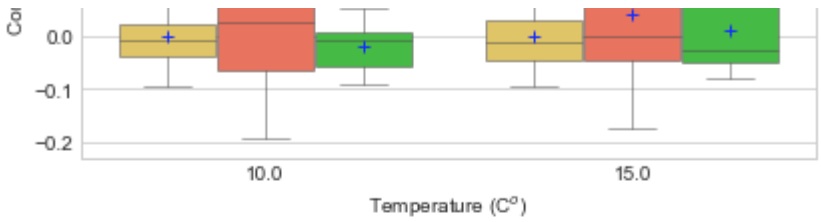


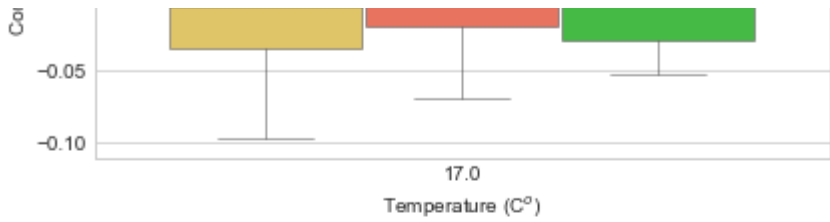

Sunday 04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
Temperature (C°)

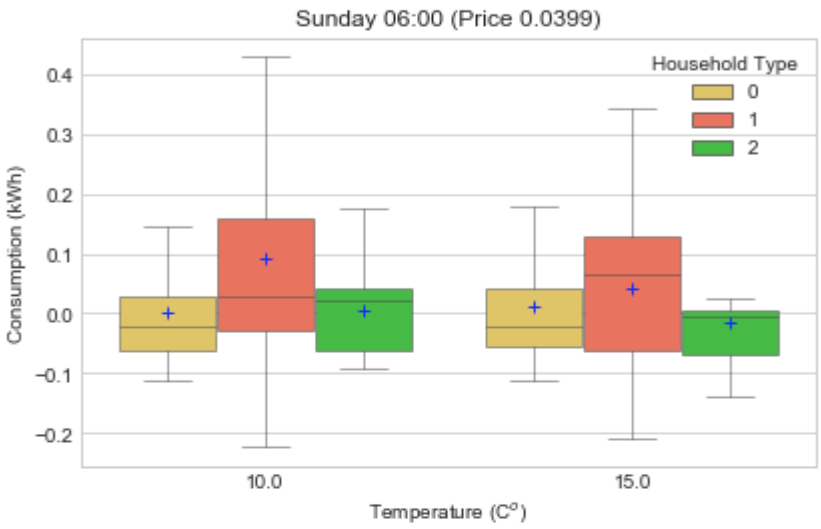

Sunday 04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
17.0
Temperature (C°)

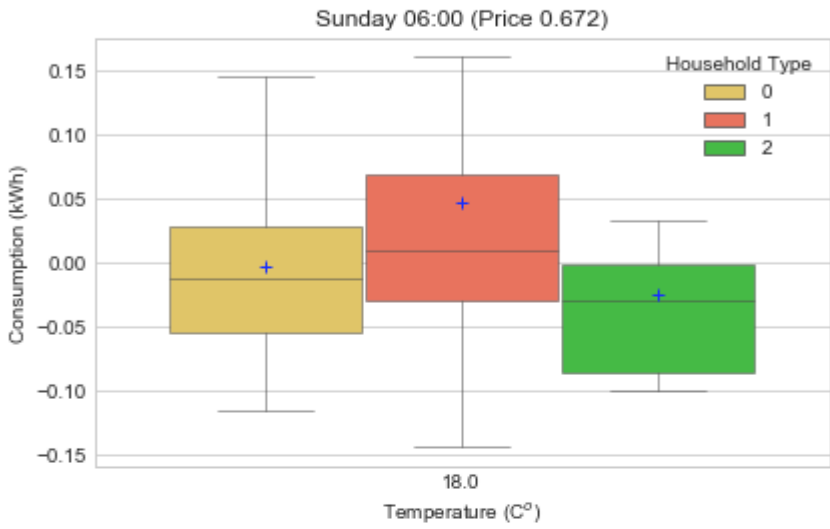

Sunday 05:00 (Price 0.0399)
Household Type
0
1
2

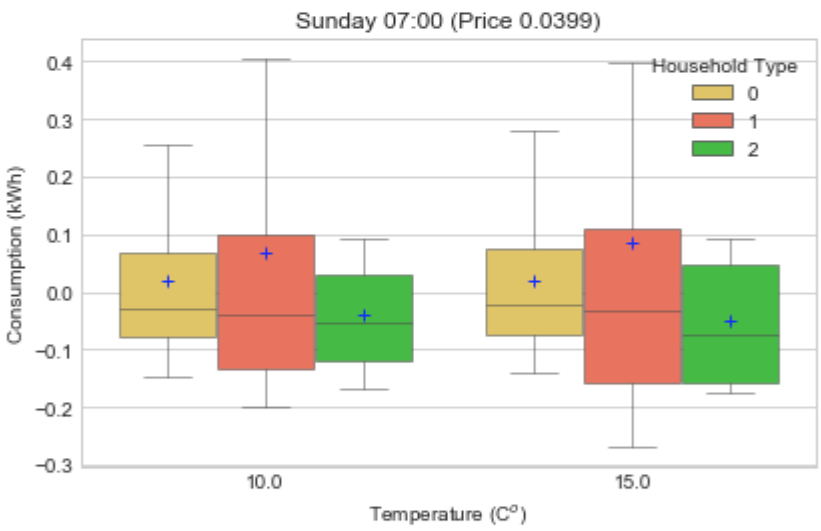

Sunday 05:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)

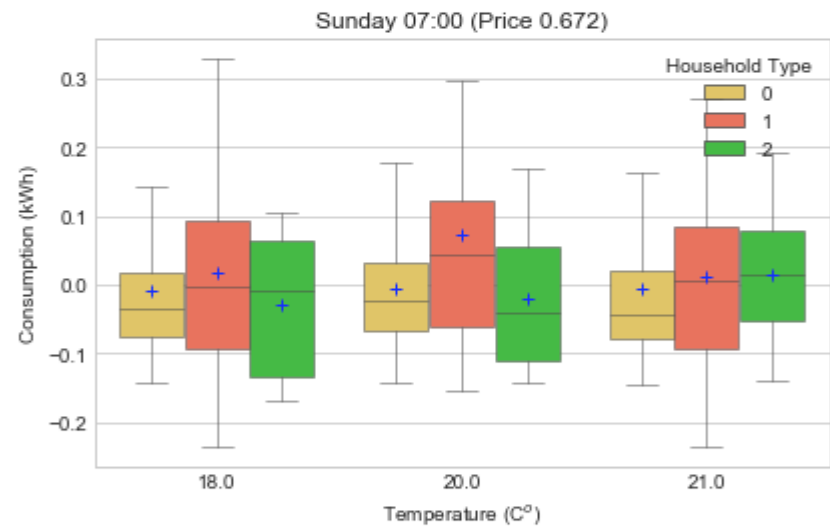

Temperature (C°)

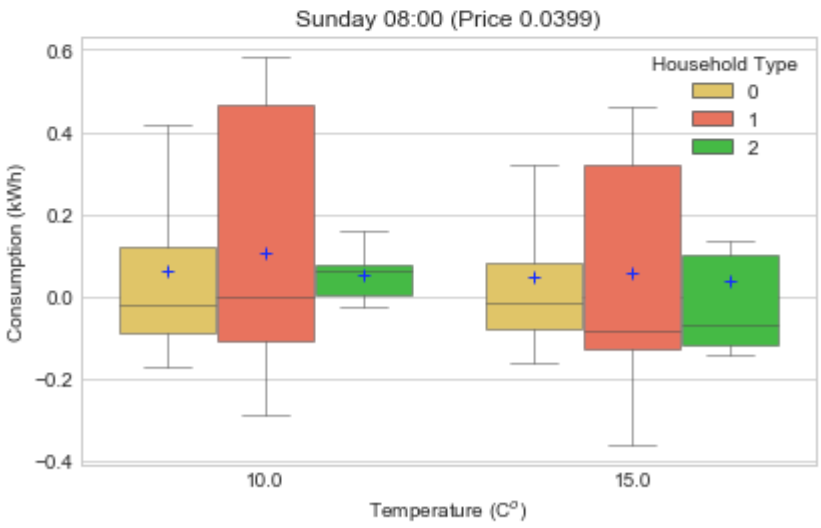

17.0
Temperature (C°)

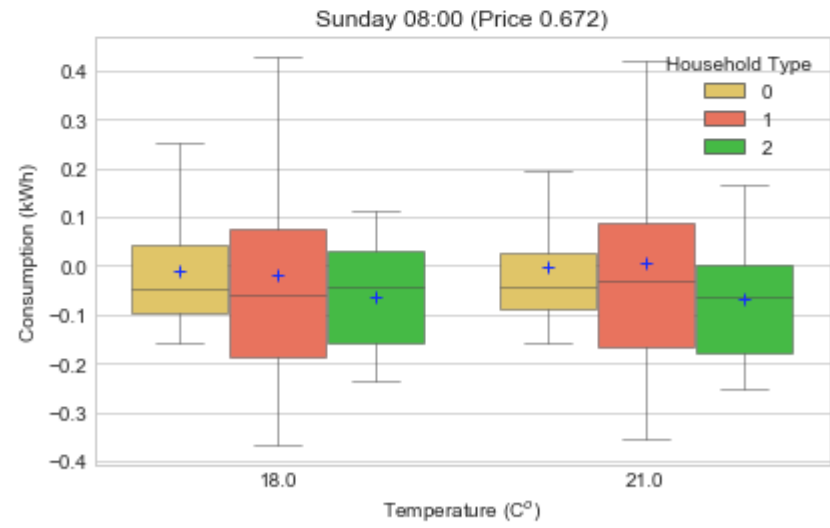

Sunday 06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
Temperature (C°)

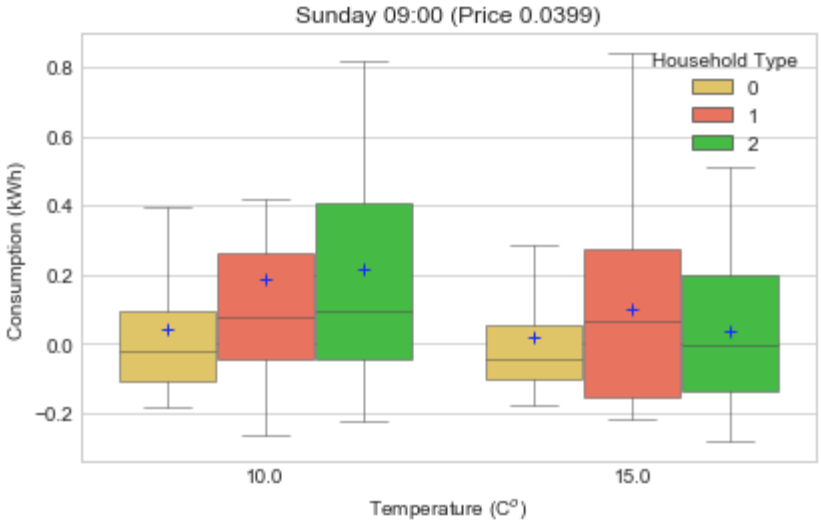

Sunday 06:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
18.0
Temperature (C°)

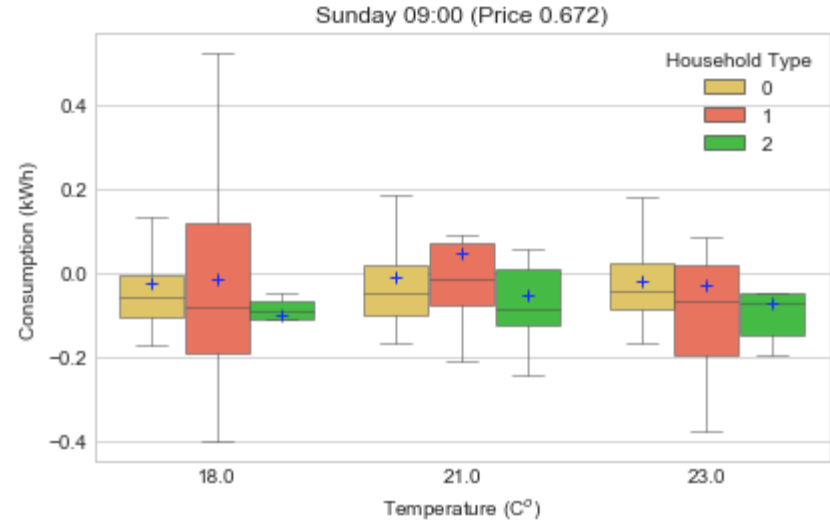

Sunday 07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
Temperature (C°)

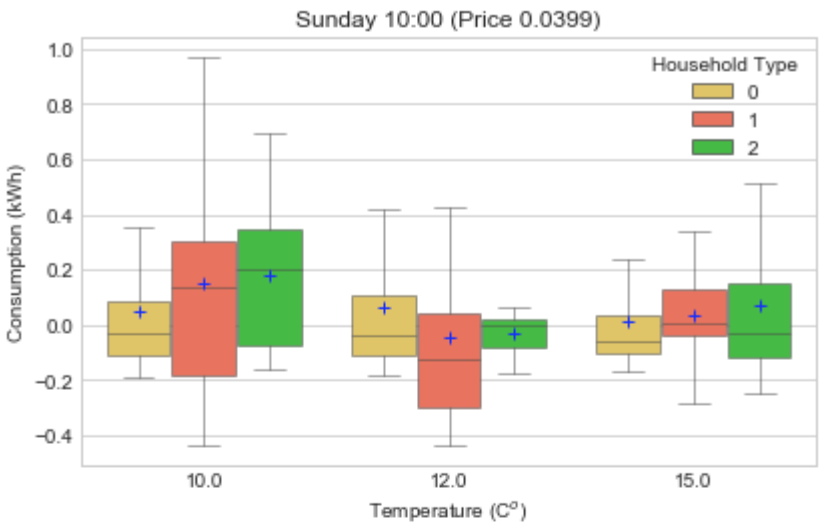

Sunday 07:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
18.0
20.0
21.0
Temperature (C°)

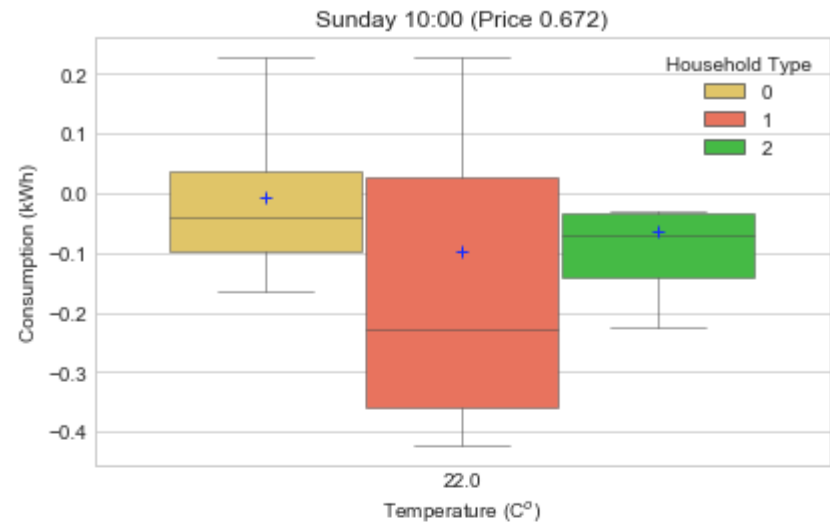

Sunday 08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
Temperature (C°)


Sunday 08:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
18.0
21.0
Temperature (C°)


Sunday 09:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
Temperature (C°)

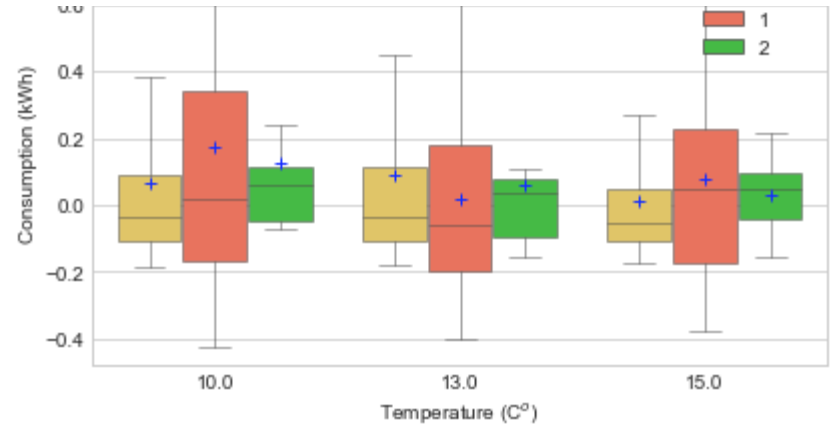

Sunday 09:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
18.0
21.0
23.0
Temperature (C°)

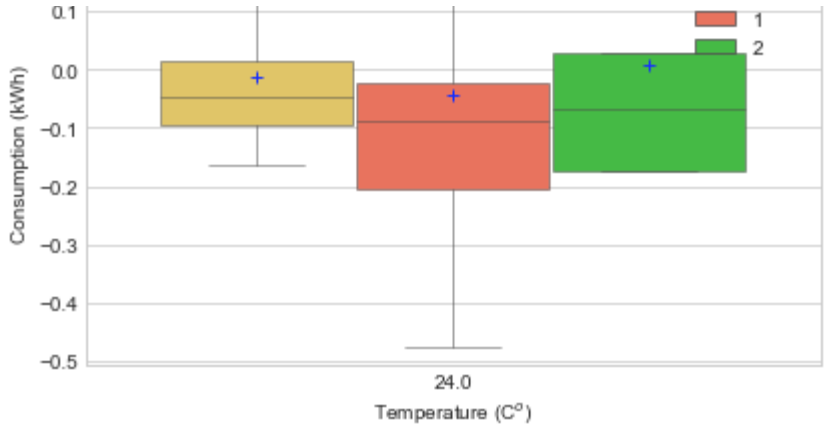

Sunday 10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
10.0
12.0
15.0
Temperature (C°)

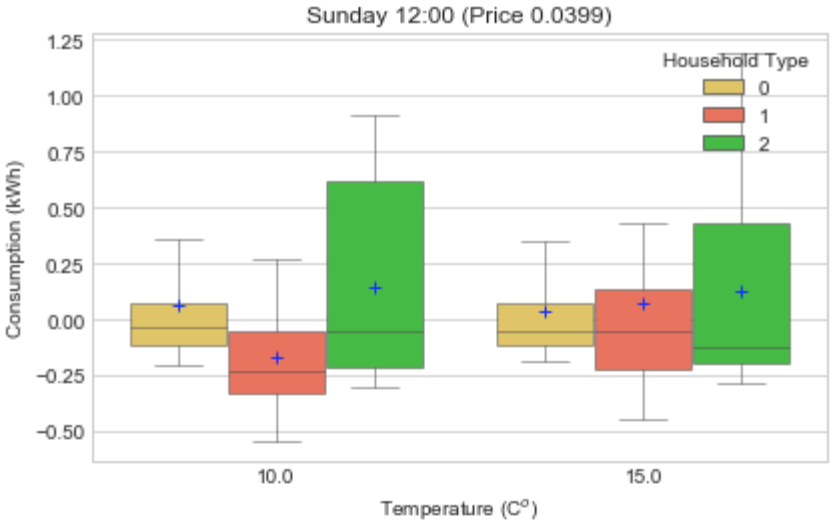

Sunday 10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
22.0
Temperature (C°)

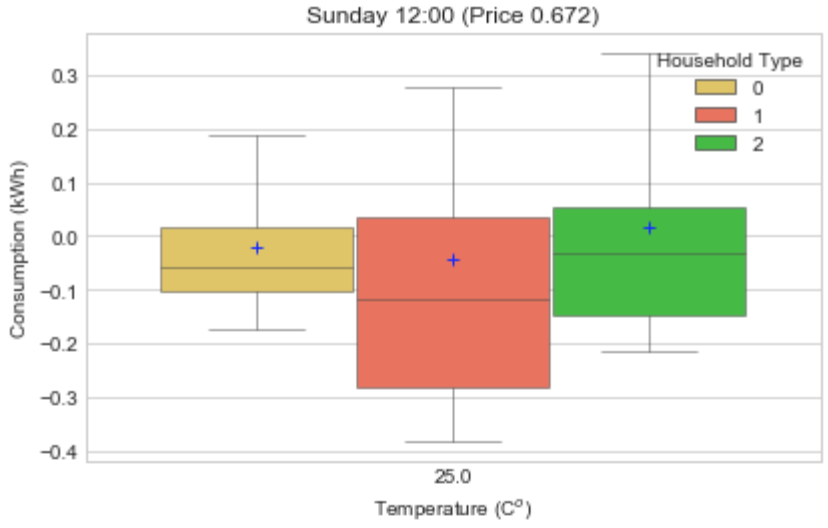
Sunday 11:00 (Price 0.0399)
Household Type
0

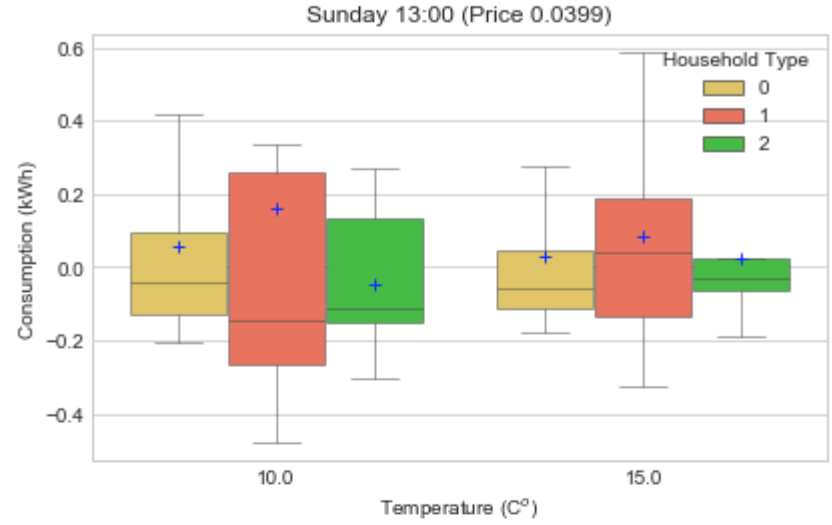
Sunday 11:00 (Price 0.672)
Household Type
0

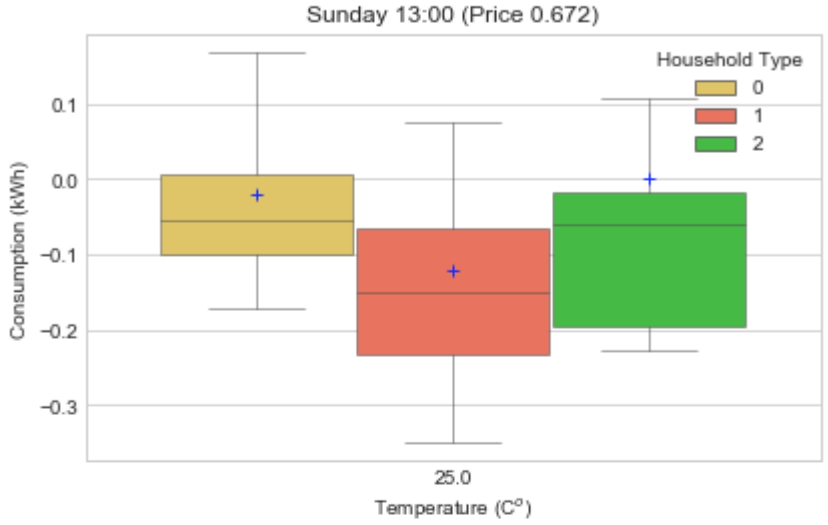

Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
10.0
13.0
15.0

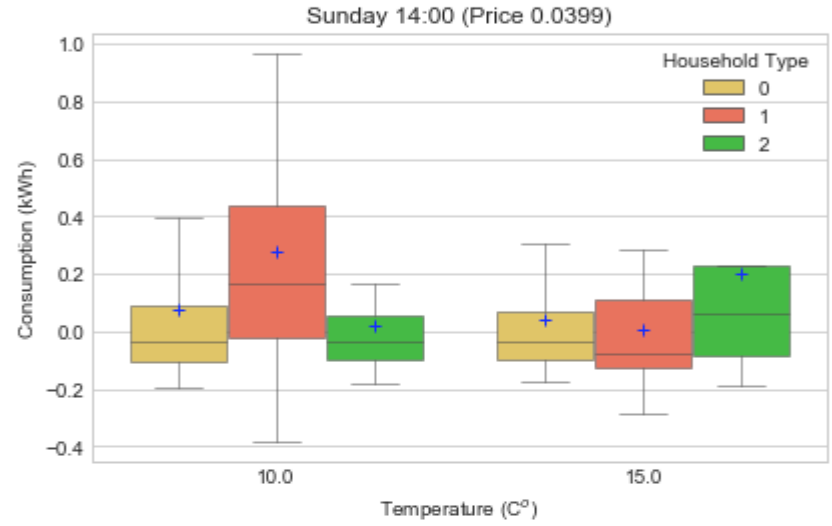

Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
24.0

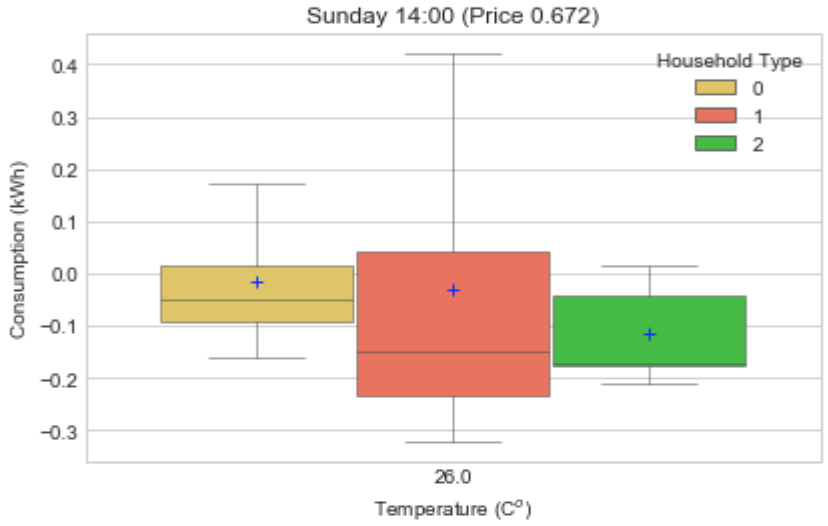

Sunday 12:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
10.0
15.0

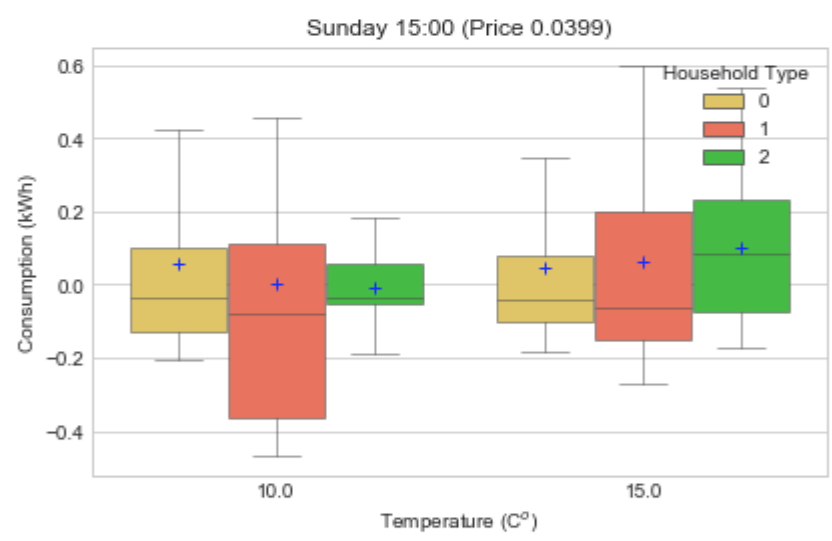

Sunday 12:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
25.0

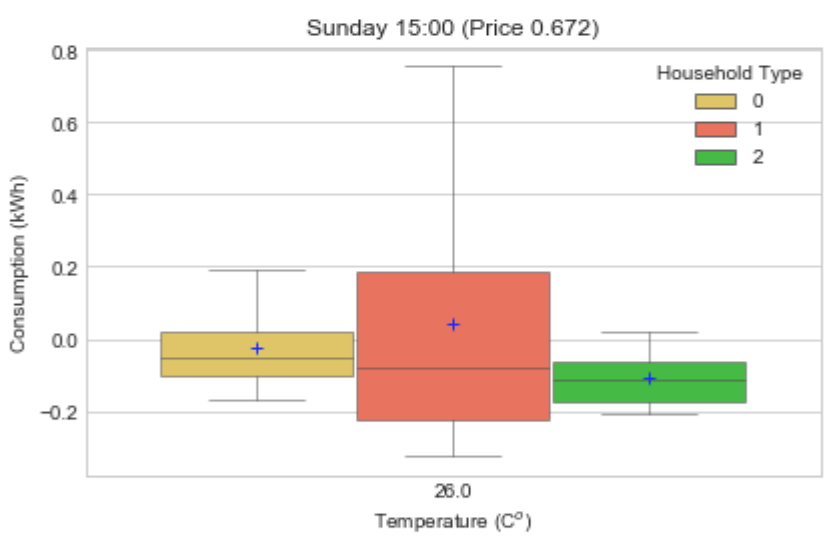

Sunday 13:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
10.0
15.0

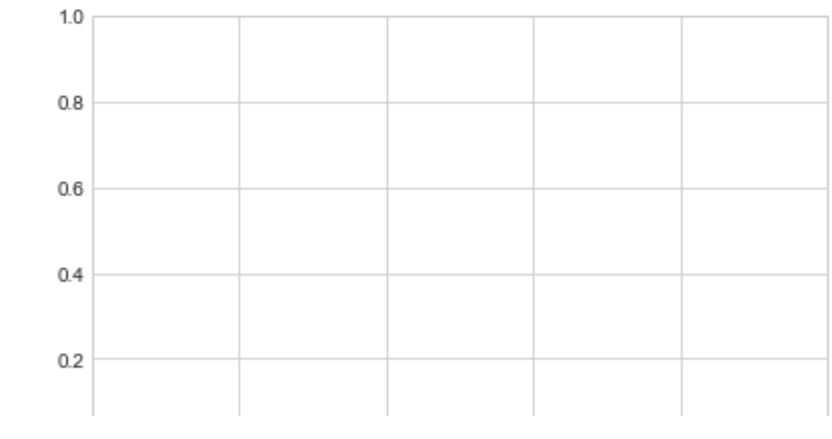

Sunday 13:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
25.0

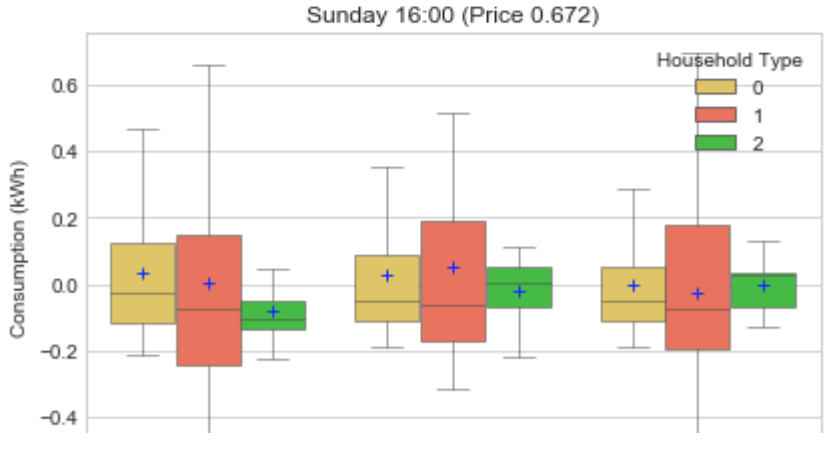

Sunday 14:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
10.0
15.0

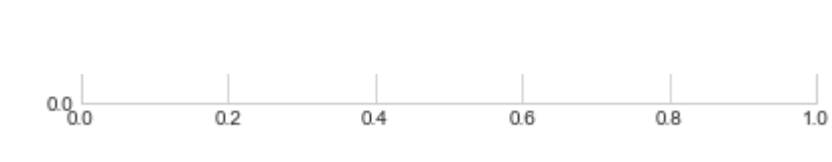

Sunday 14:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
26.0

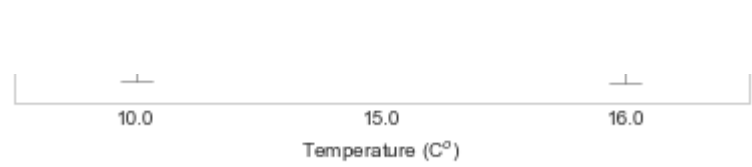

Sunday 15:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
10.0
15.0

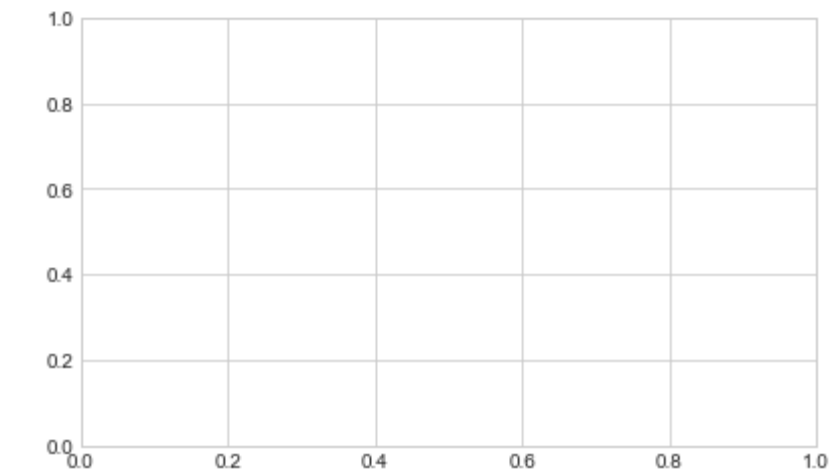

Sunday 15:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)
26.0

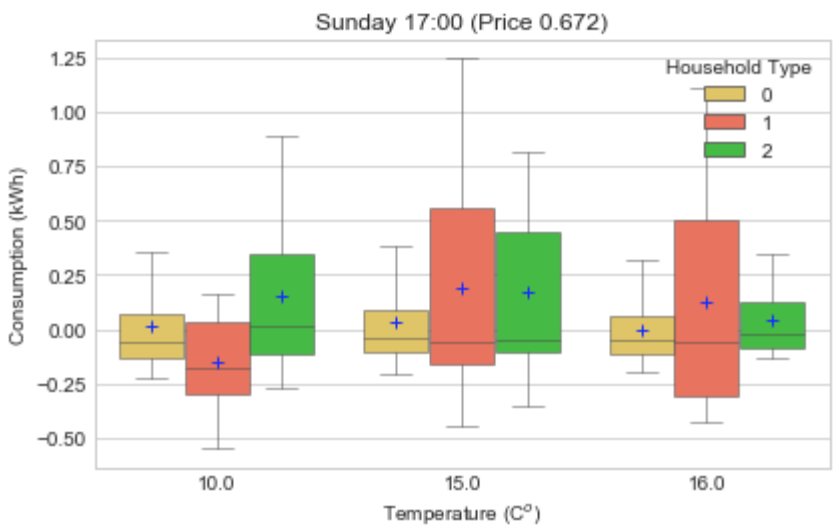


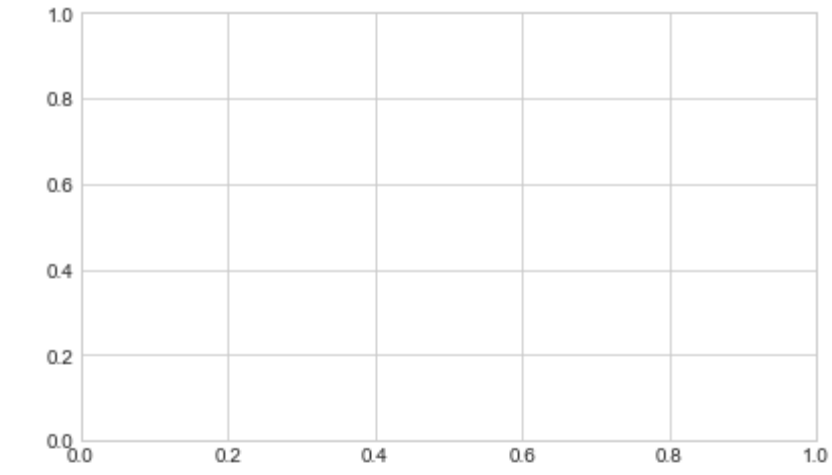

Sunday 16:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)

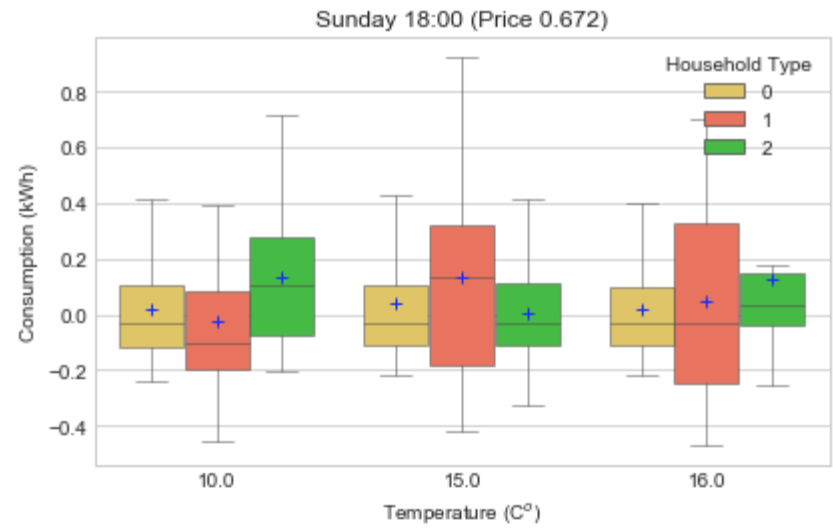


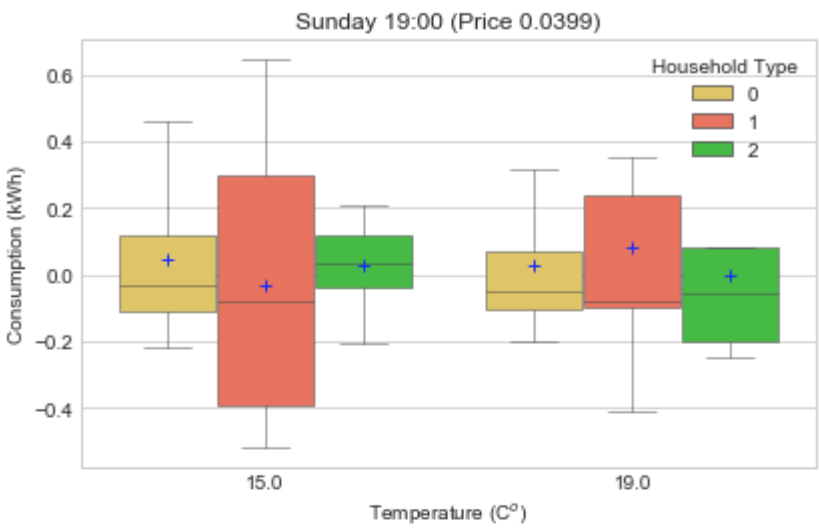

10.0
15.0
16.0
Temperature (C°)

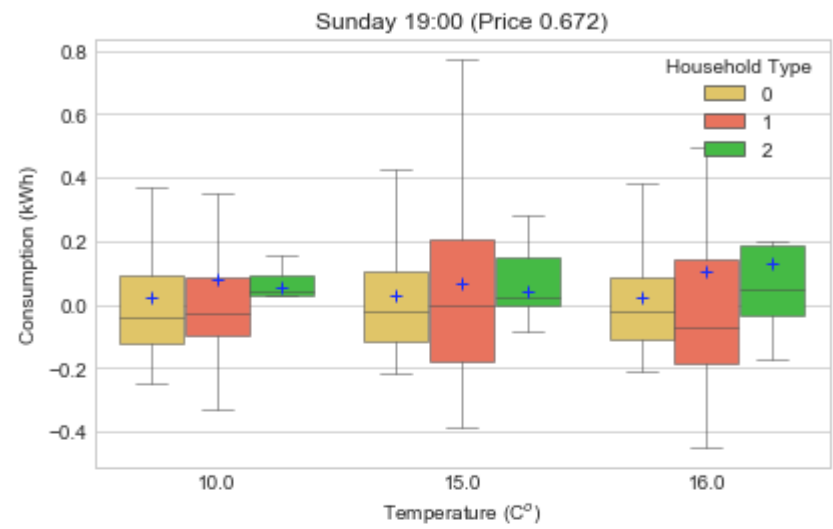


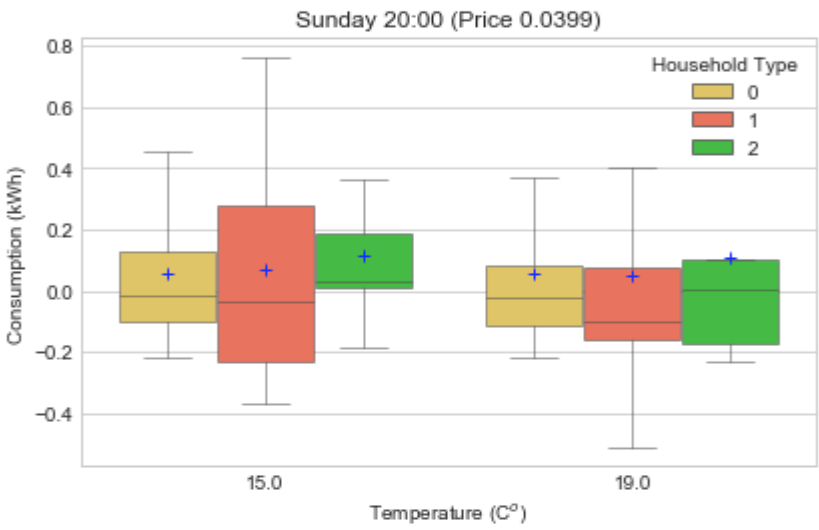

Sunday 17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

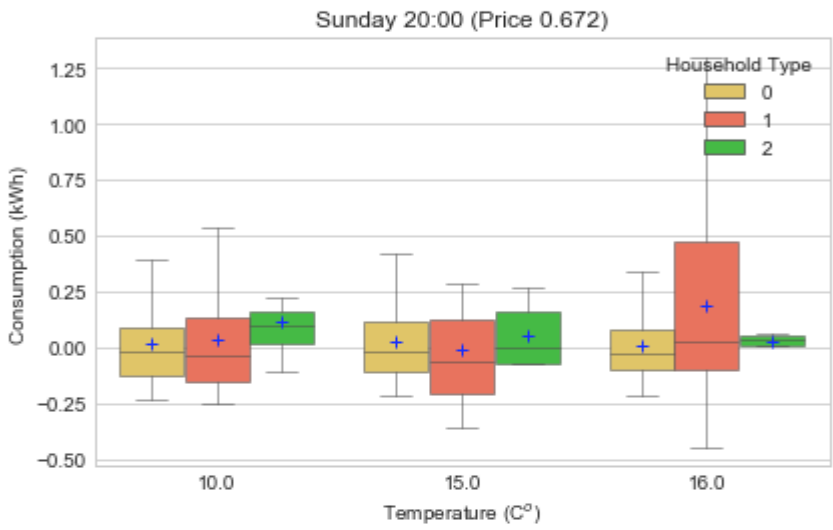


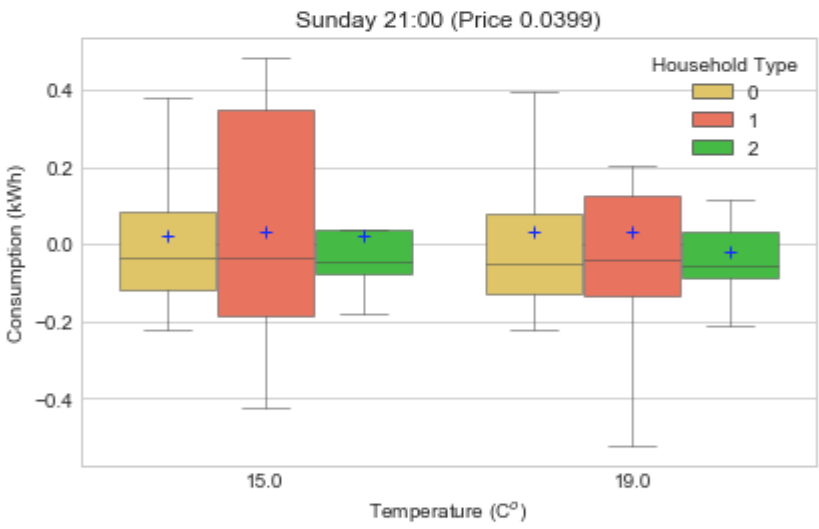

Sunday 18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

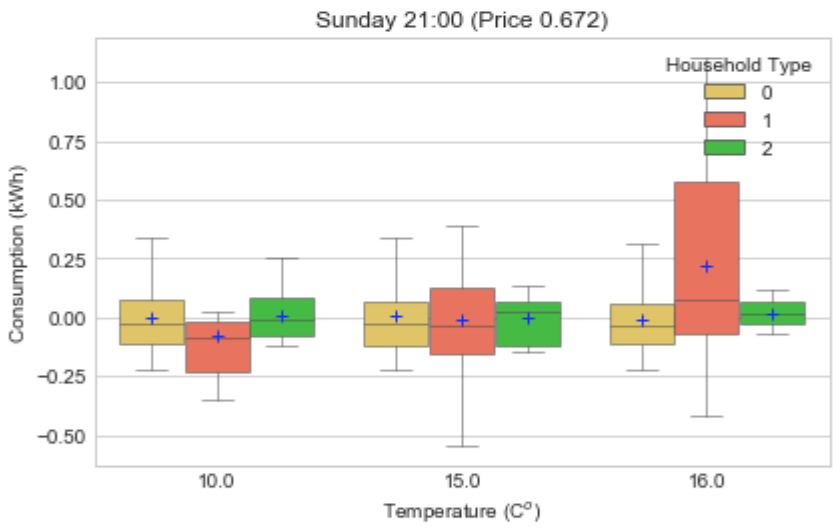

Sunday 19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
15.0
19.0
Temperature (C°)


Sunday 19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)


Sunday 20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
15.0
19.0
Temperature (C°)

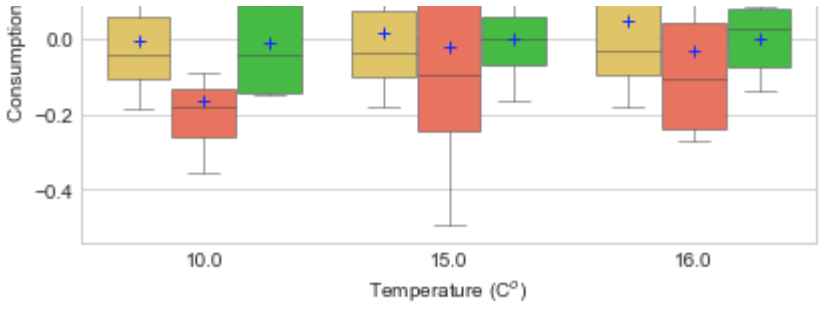

Sunday 20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

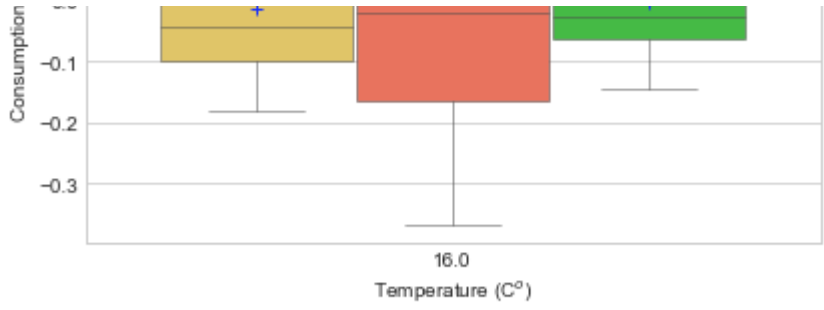

Sunday 21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
15.0
19.0
Temperature (C°)

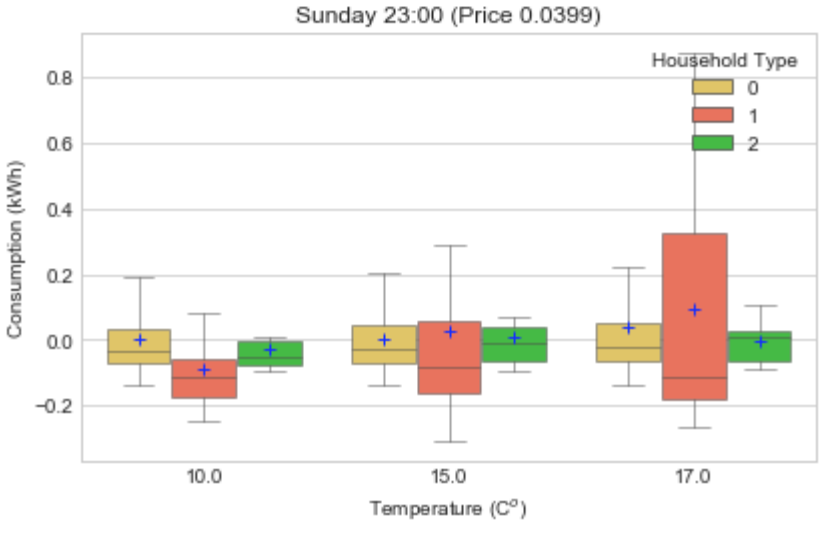

Sunday 21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

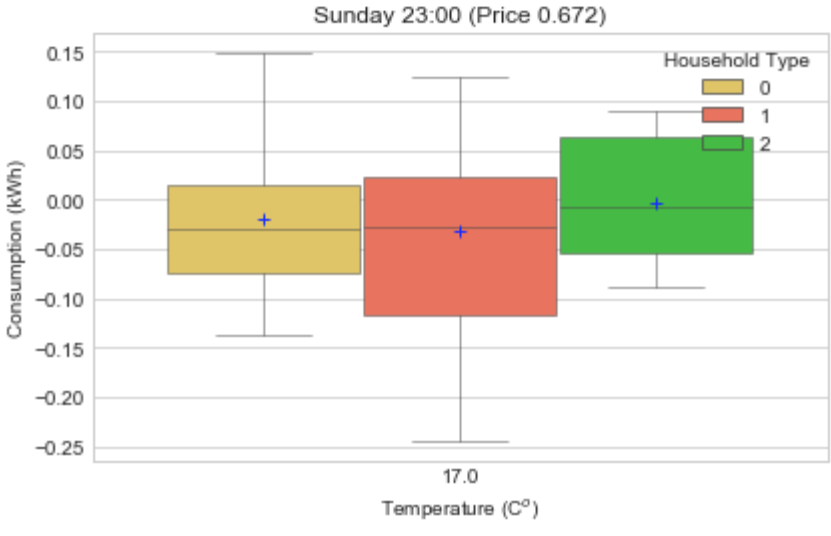
Sunday 22:00 (Price 0.0399)
Household Type
0
1
2

Sunday 22:00 (Price 0.672)
Household Type
0
1
2

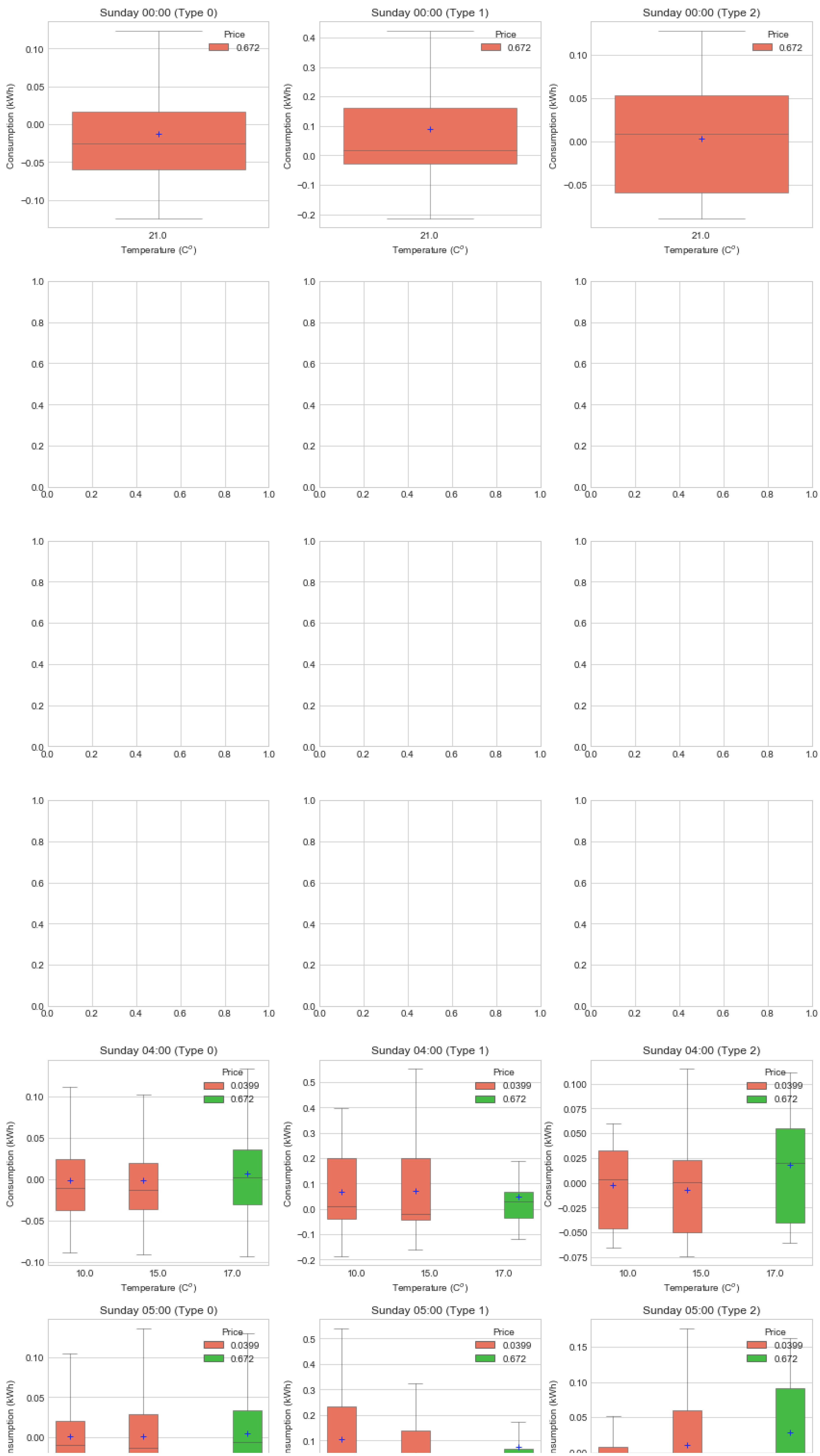

Consumption
0.0
−0.2
−0.4
10.0
15.0
16.0
Temperature (C°)


Consumption
−0.1
−0.2
−0.3
16.0
Temperature (C°)

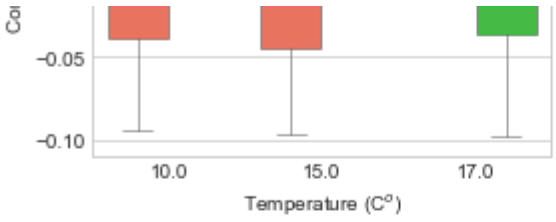

Sunday 23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
10.0
15.0
17.0
Temperature (C°)

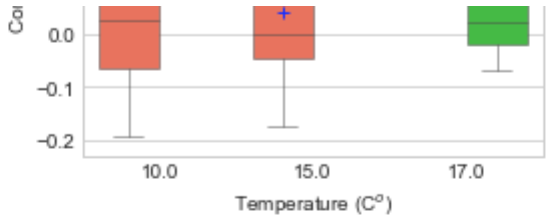

Sunday 23:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
17.0
Temperature (C°)

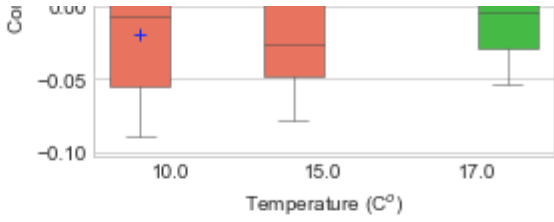
In [74]:
```
# price comparison
# Sunday hourly price responsiveness with different temperature and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
day_of_week = 6 # 0 is Monday, 6 is Sunday
df_day = df_all_res_long[df_all_res_long['Day of week'] == day_of_week]
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_day_hour = df_day[(df_day['Hour of day'] == i) & (df_day['Household type'] == g)]
        if df_day_hour.shape[0] >= 1:
            if i <= 9:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' 0' + str(i) + ':00 (Type ' + str(g) + ')' )
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
            else:
                sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day_hour, ax=ax_Ntou[-1], palett
e=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkc
d:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
                ax_Ntou[-1].set_title(weeks[day_of_week] + ' ' + str(i) + ':00 (Type ' + str(g) + ')')
                ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
                ax_Ntou[-1].set_ylabel('Consumption (kWh)')
                l = ax_Ntou[-1].legend()
                l.set_title('Price')
                for i,artist in enumerate(ax_Ntou[-1].artists):
                    artist.set_linewidth(0.5)
                    # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                    # Loop over them here, and use the same colour as above
                    for j in range(i*6,i*6+6):
                        line = ax_Ntou[-1].lines[j]
                        line.set_linewidth(0.5)
                ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

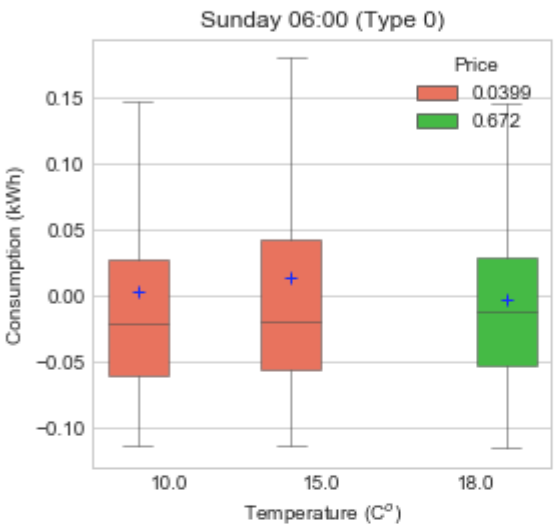


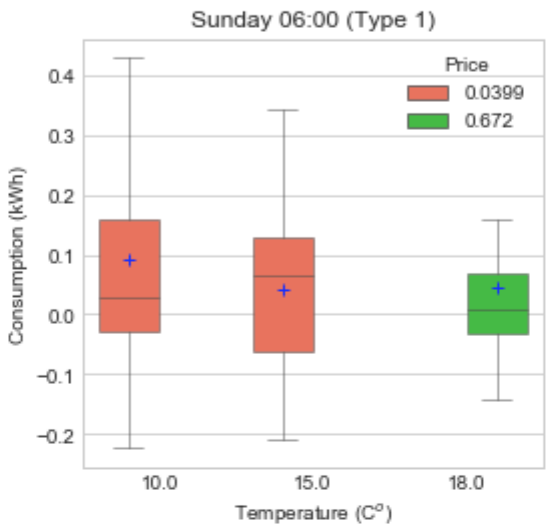
Sunday 00:00 (Type 0)
Sunday 00:00 (Type 1)
Sunday 00:00 (Type 2)
Price
0.672
Consumption (kWh)
Temperature (C^o)
21.0
Sunday 04:00 (Type 0)
Sunday 04:00 (Type 1)
Sunday 04:00 (Type 2)
Price
0.0399
0.672
10.0
15.0
17.0
Sunday 05:00 (Type 0)
Sunday 05:00 (Type 1)
Sunday 05:00 (Type 2)

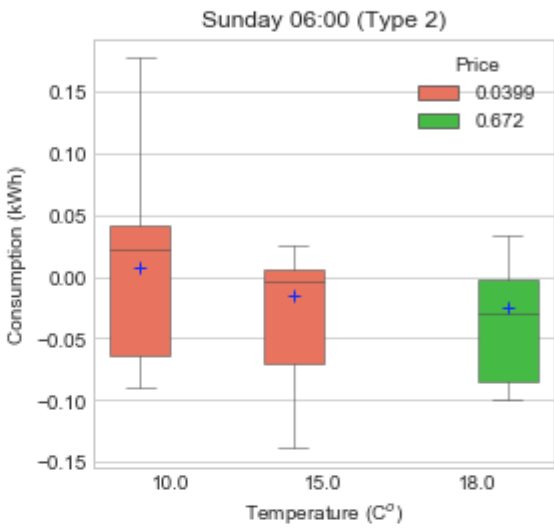

Temperature (C°)

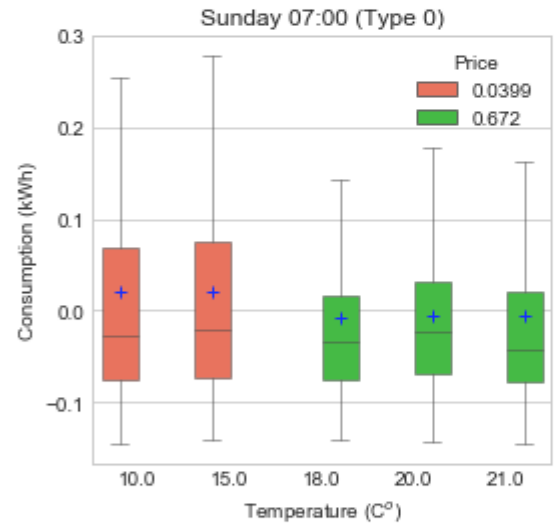

10.0
15.0
17.0
Temperature (C°)

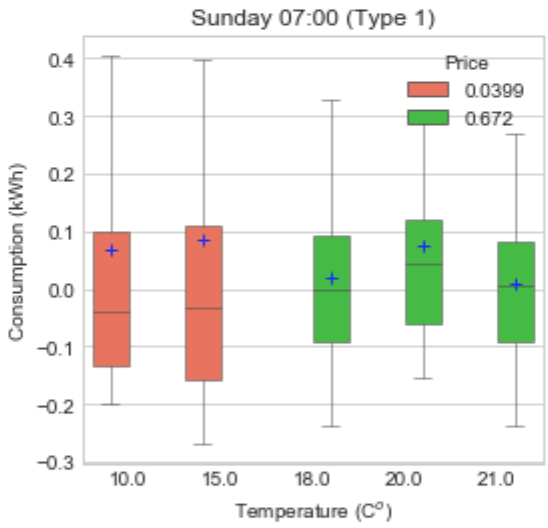

10.0
15.0
17.0
Temperature (C°)

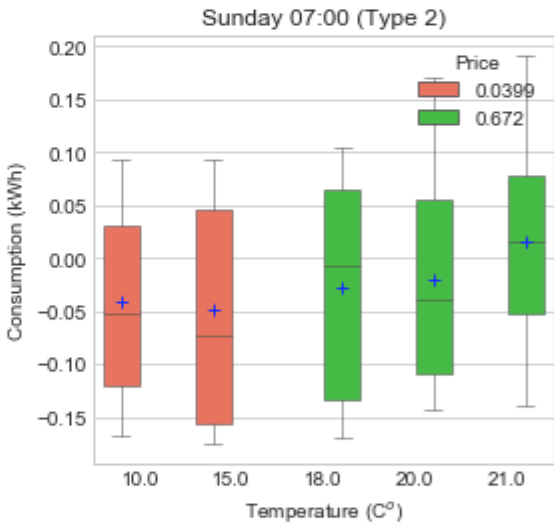

Sunday 06:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
Temperature (C°)

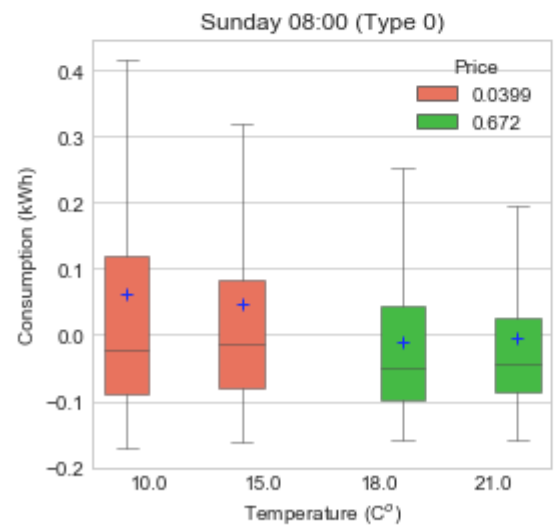

Sunday 06:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
Temperature (C°)

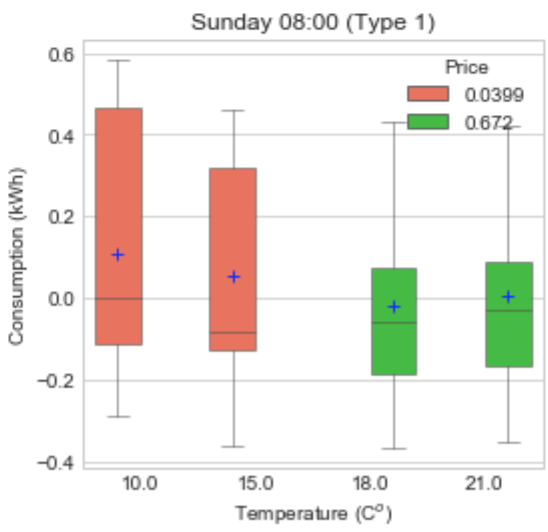

Sunday 06:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
Temperature (C°)

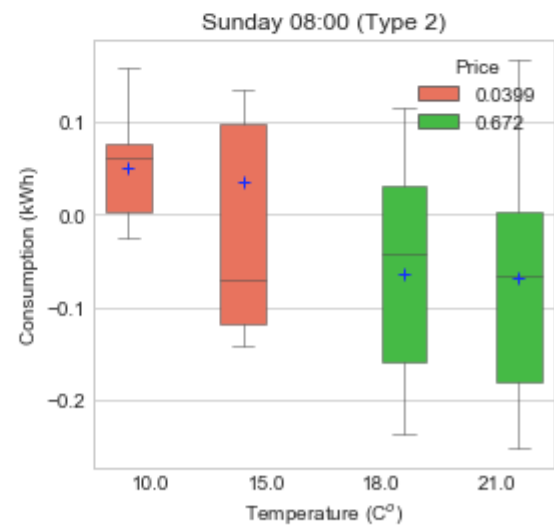

Sunday 07:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
20.0
21.0
Temperature (C°)

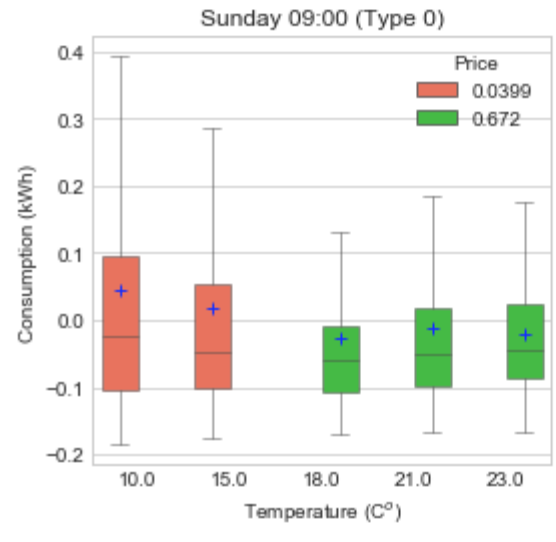

Sunday 07:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
20.0
21.0
Temperature (C°)

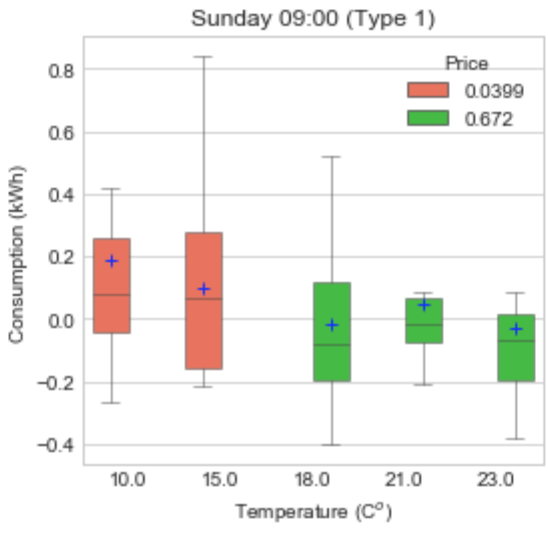

Sunday 07:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
20.0
21.0
Temperature (C°)

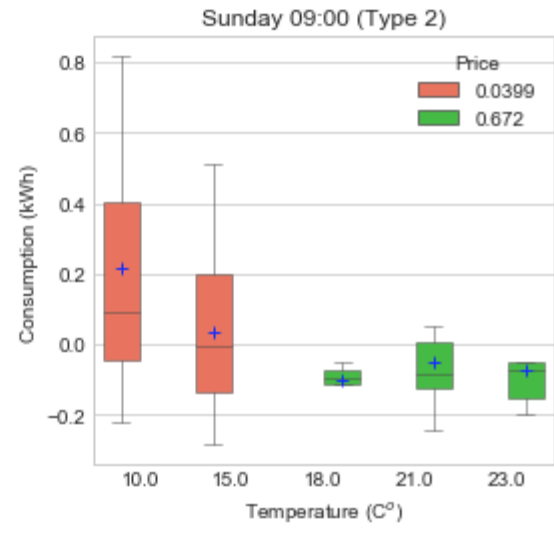

Sunday 08:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
Temperature (C°)

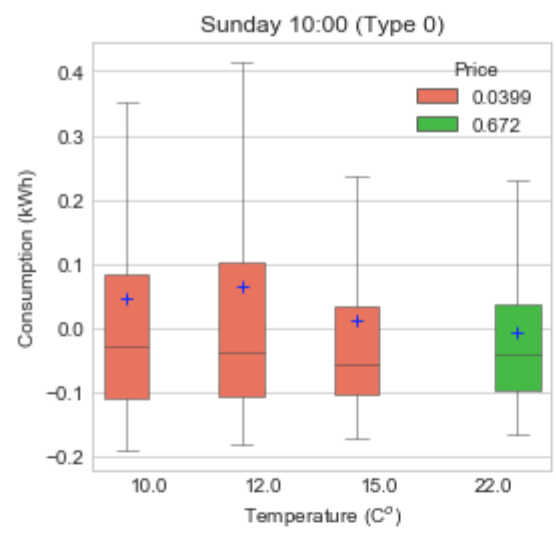

Sunday 08:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
Temperature (C°)

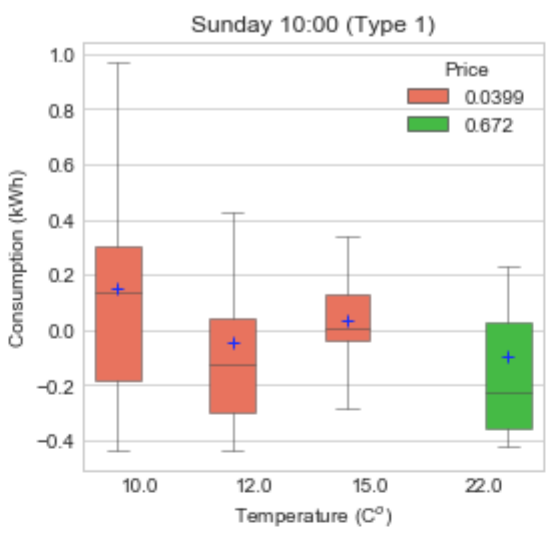

Sunday 08:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
Temperature (C°)

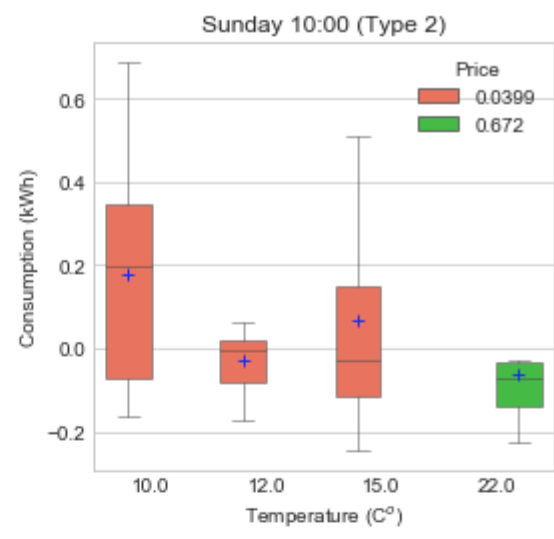

Sunday 09:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
23.0
Temperature (C°)


Sunday 09:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
23.0
Temperature (C°)


Sunday 09:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
18.0
21.0
23.0
Temperature (C°)


Sunday 10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
12.0
15.0
22.0
Temperature (C°)

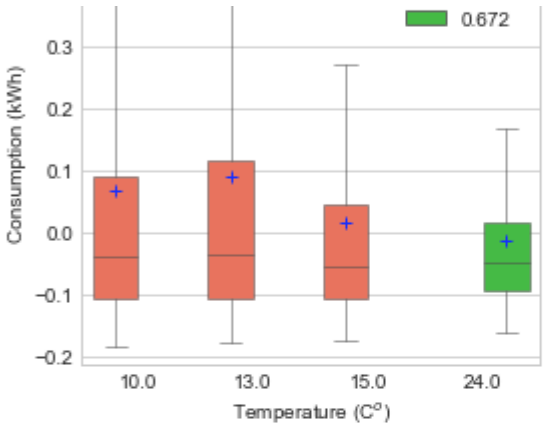

Sunday 10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
12.0
15.0
22.0
Temperature (C°)

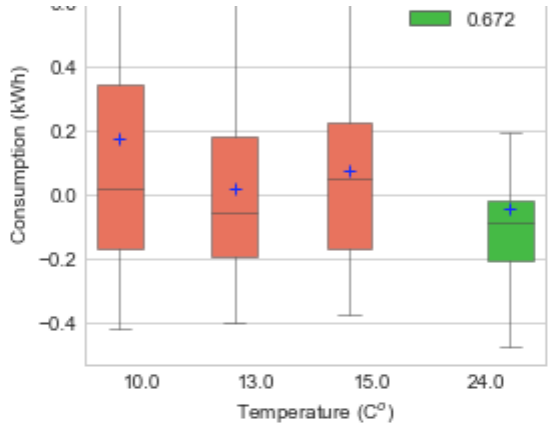

Sunday 10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
12.0
15.0
22.0
Temperature (C°)

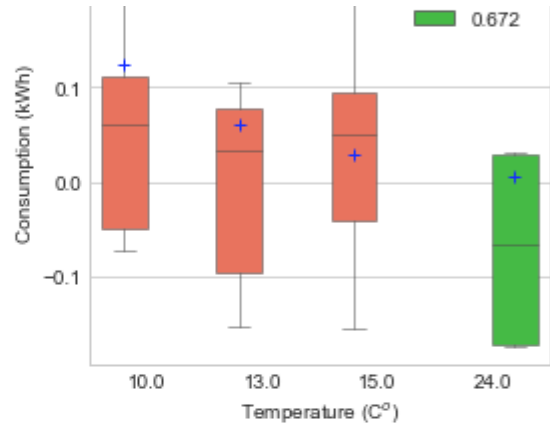

Sunday 11:00 (Type 0)
Price
0.0399

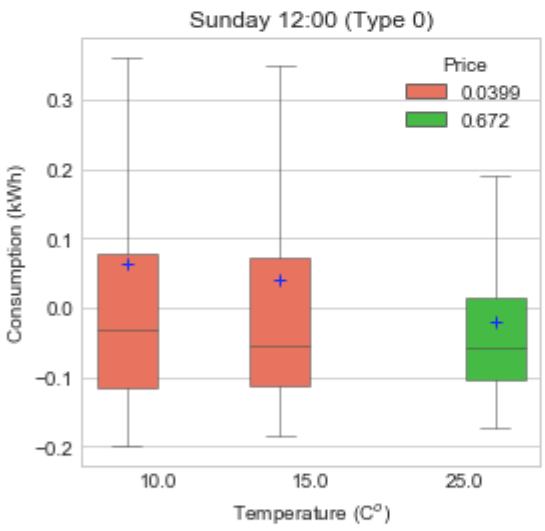

Sunday 11:00 (Type 1)
Price
0.0399

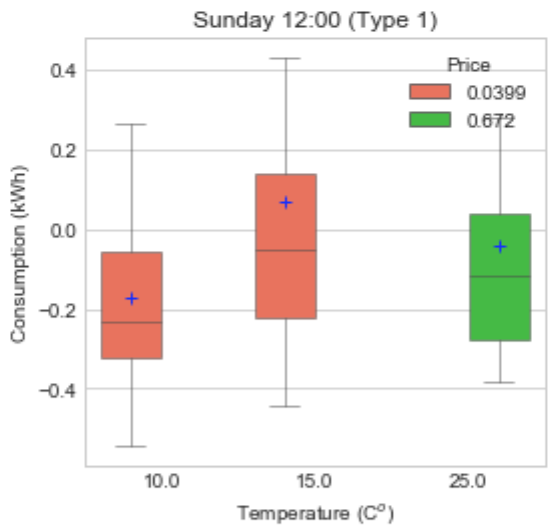

Sunday 11:00 (Type 2)
Price
0.0399

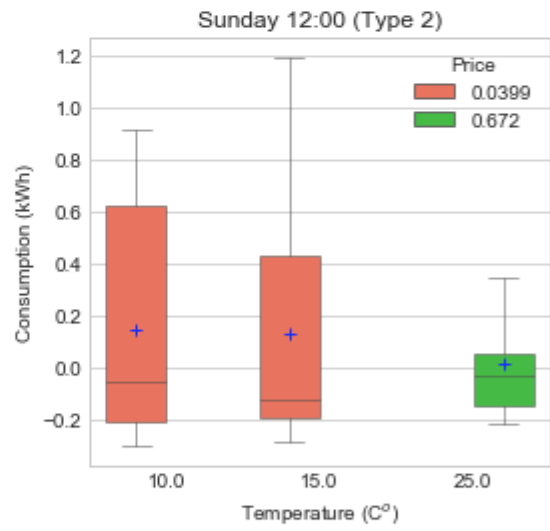

Price
0.672
Consumption (kWh)
Temperature (C°)
10.0
13.0
15.0
24.0

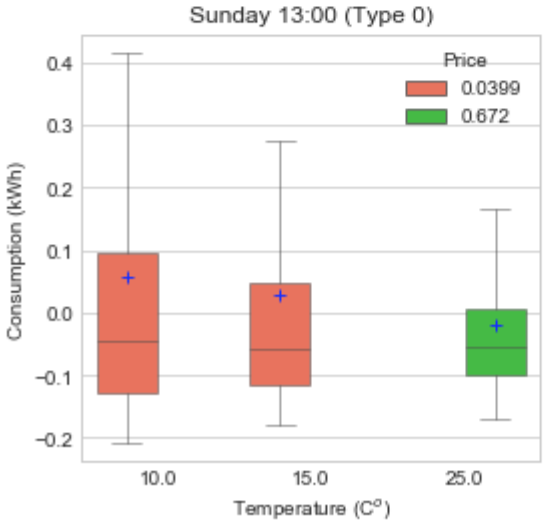

Price
0.672
Consumption (kWh)
Temperature (C°)
10.0
13.0
15.0
24.0

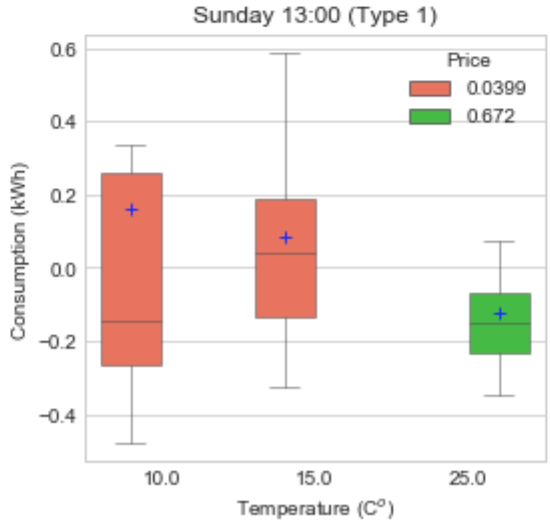

Price
0.672
Consumption (kWh)
Temperature (C°)
10.0
13.0
15.0
24.0

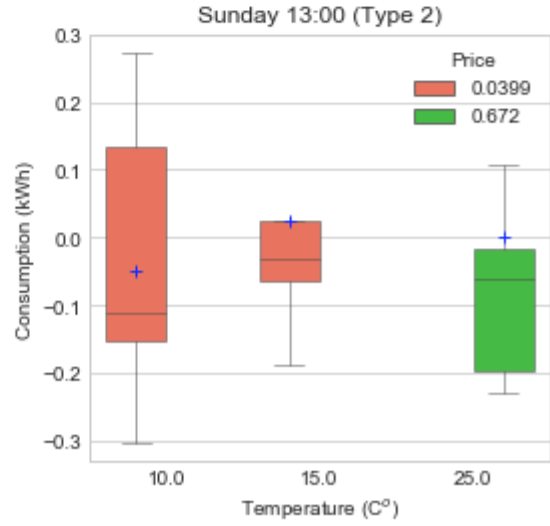

Sunday 12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

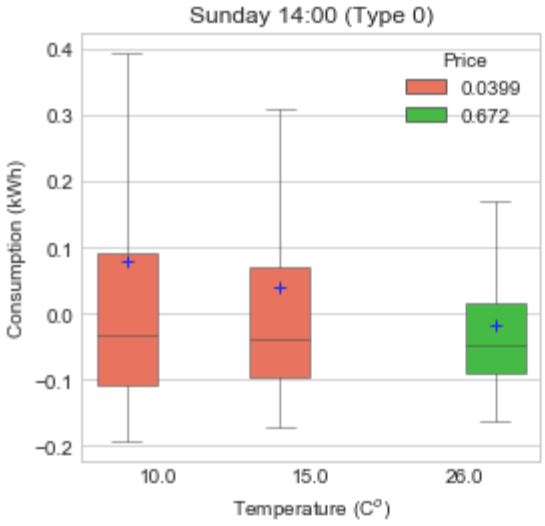

Sunday 12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

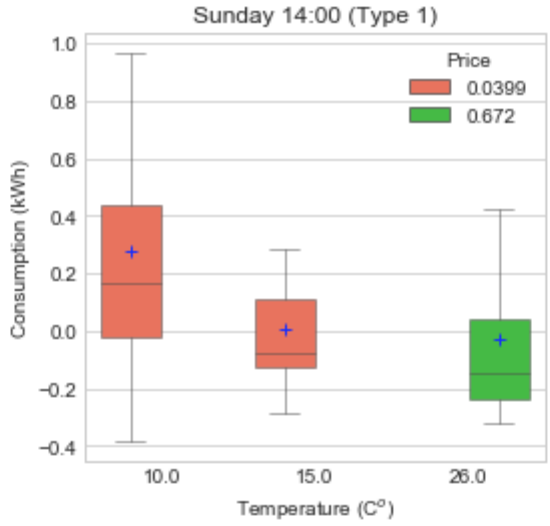

Sunday 12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

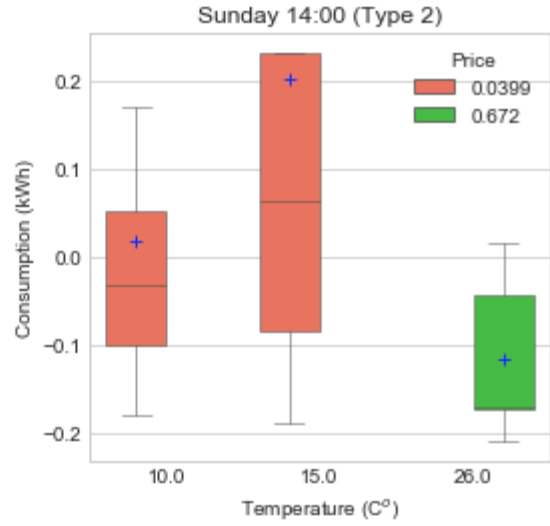

Sunday 13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

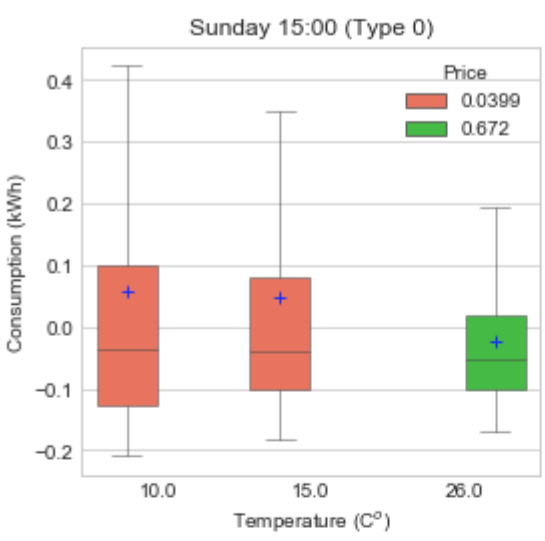

Sunday 13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

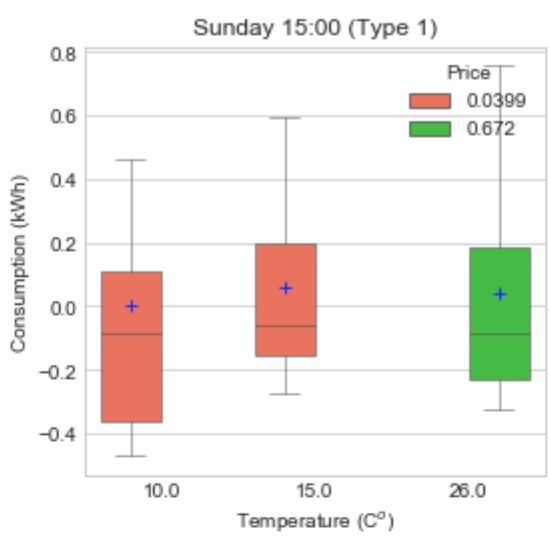

Sunday 13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
25.0

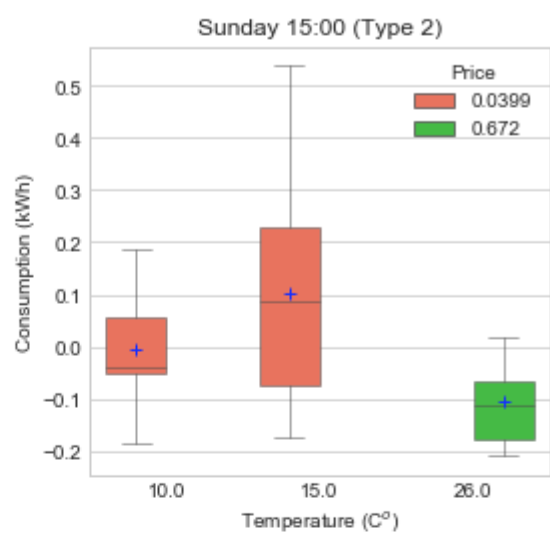

Sunday 14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0

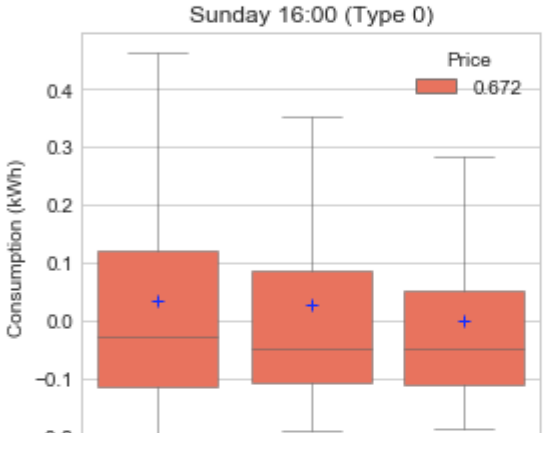

Sunday 14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0

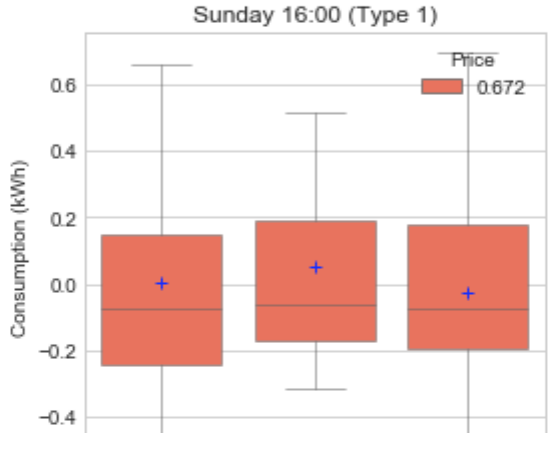

Sunday 14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0

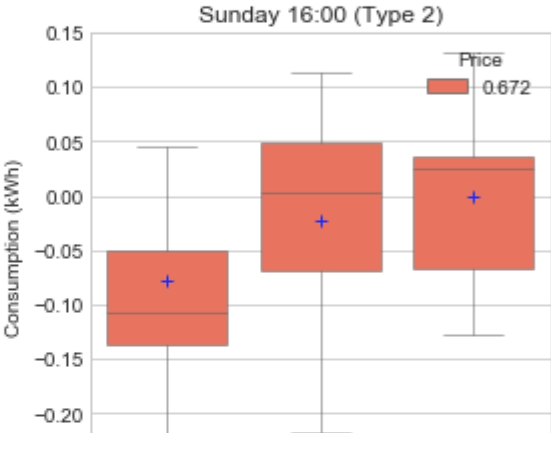

Sunday 15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0

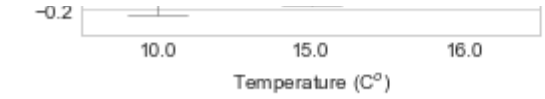

Sunday 15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0

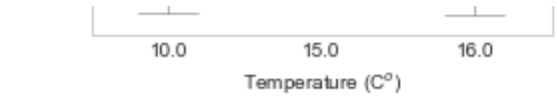

Sunday 15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)
10.0
15.0
26.0


Sunday 16:00 (Type 0)
Price
0.672
Consumption (kWh)

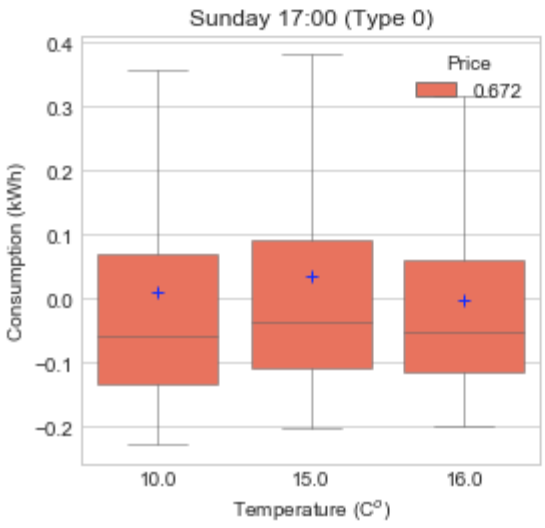

Sunday 16:00 (Type 1)
Price
0.672
Consumption (kWh)

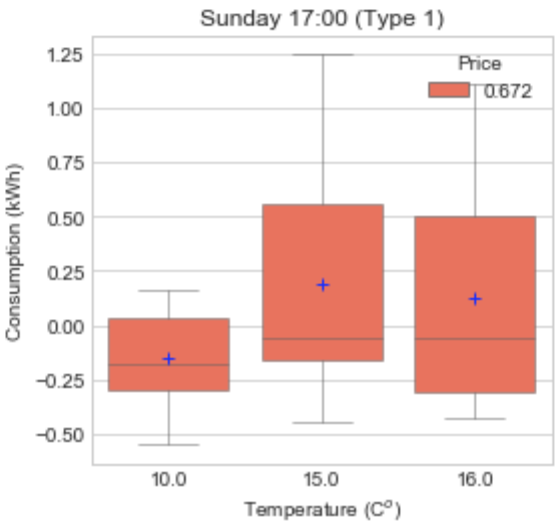

Sunday 16:00 (Type 2)
Price
0.672
Consumption (kWh)

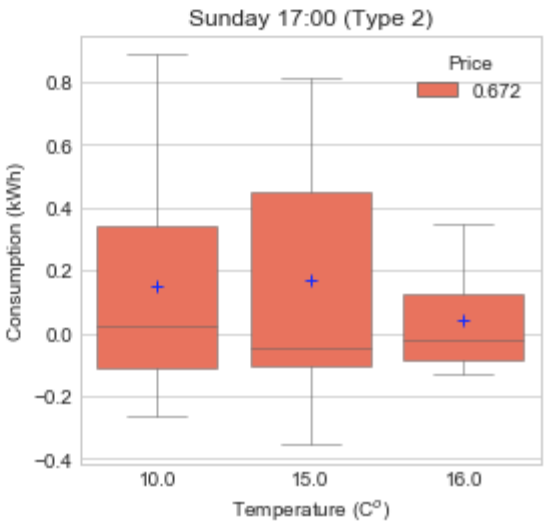

10.0
15.0
16.0
Temperature (C°)

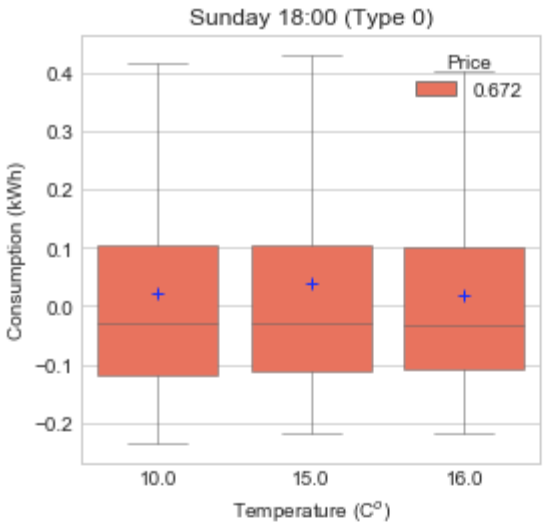

10.0
15.0
16.0
Temperature (C°)

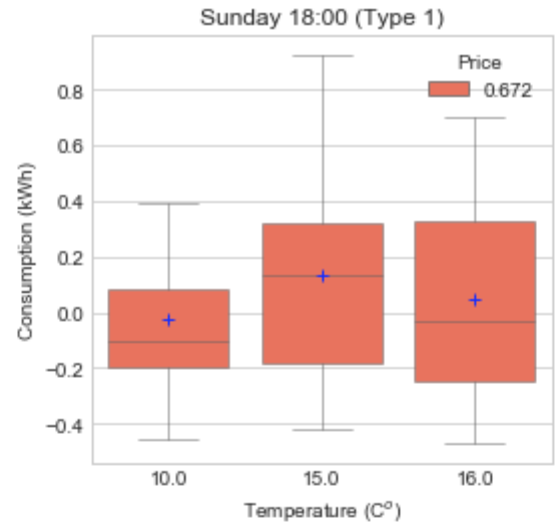

10.0
15.0
16.0
Temperature (C°)

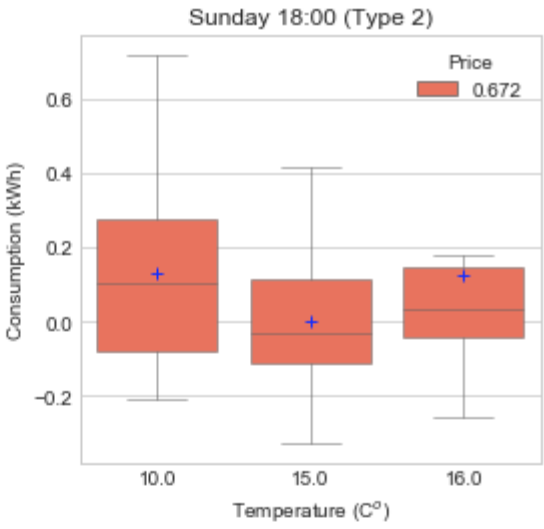

Sunday 17:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

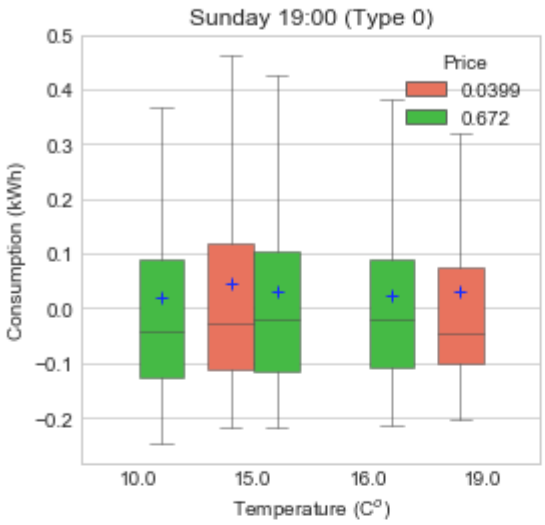

Sunday 17:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

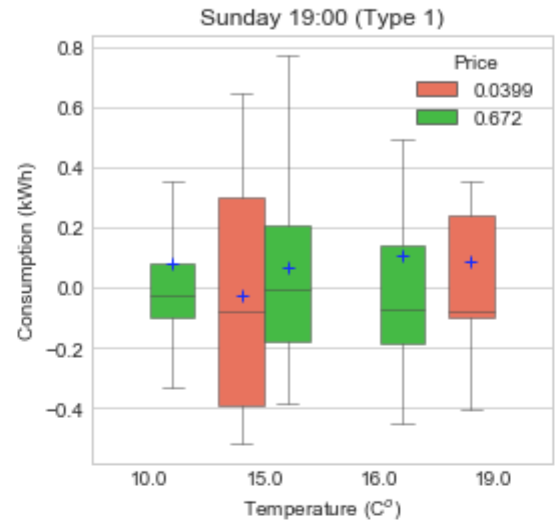

Sunday 17:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

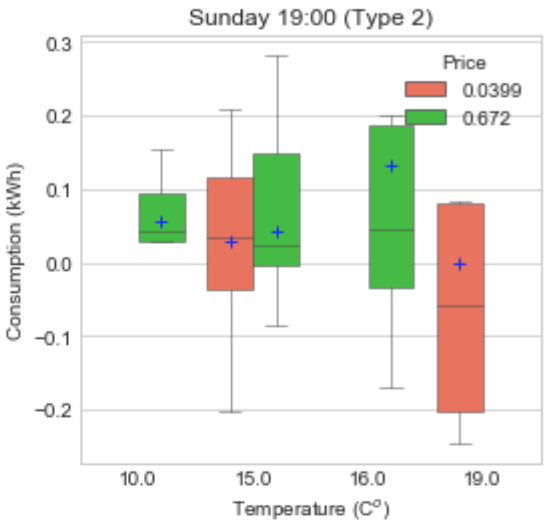

Sunday 18:00 (Type 0)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

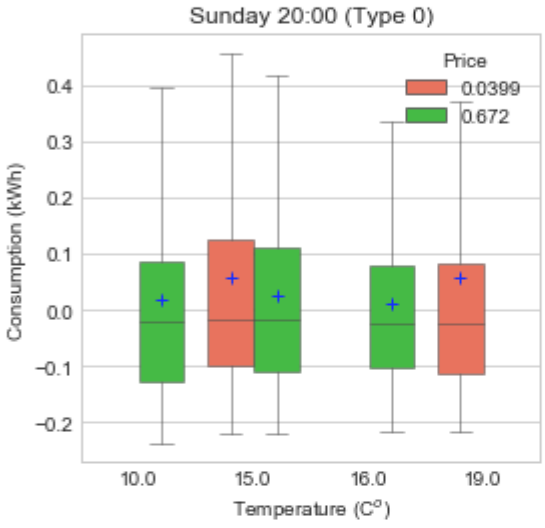

Sunday 18:00 (Type 1)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

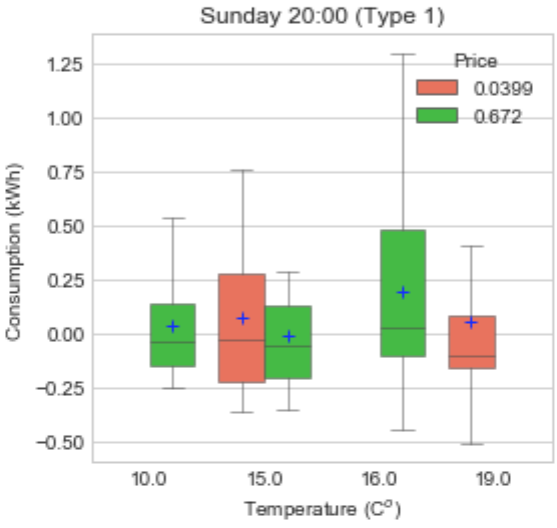

Sunday 18:00 (Type 2)
Price
0.672
Consumption (kWh)
10.0
15.0
16.0
Temperature (C°)

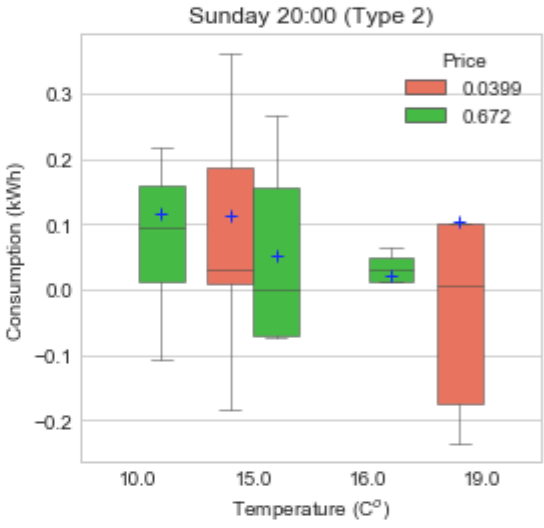

Sunday 19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

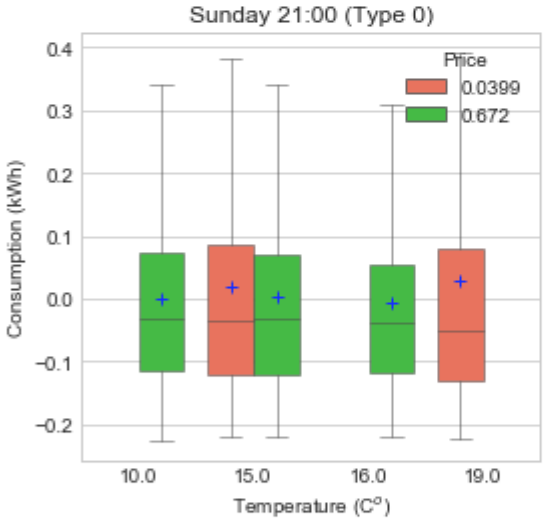

Sunday 19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

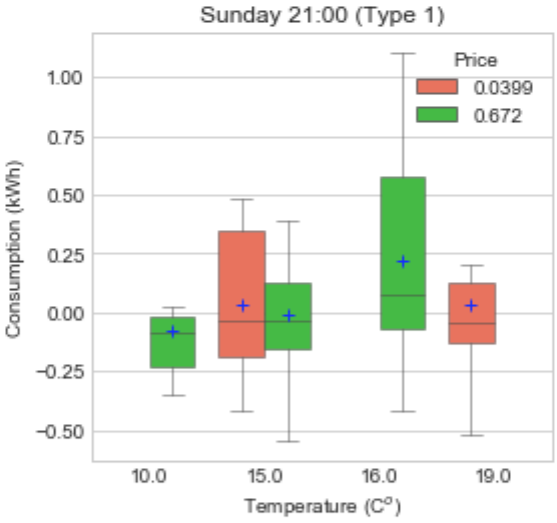

Sunday 19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

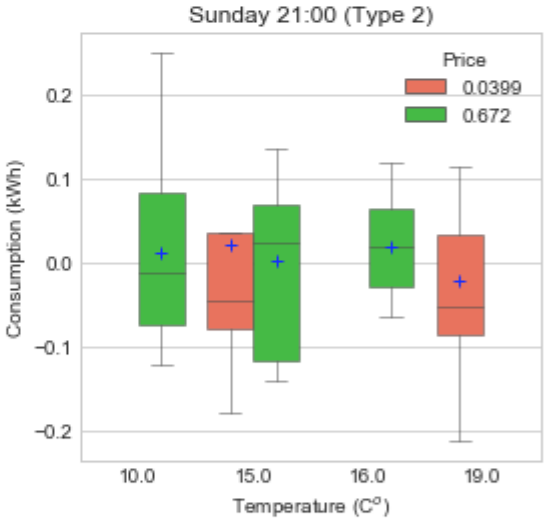

Sunday 20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

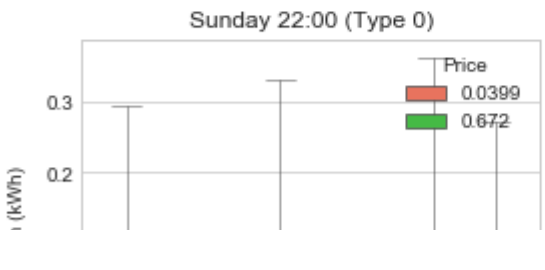

Sunday 20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

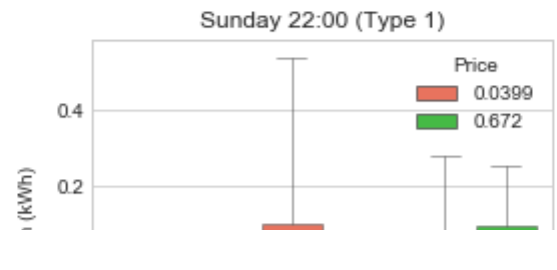

Sunday 20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

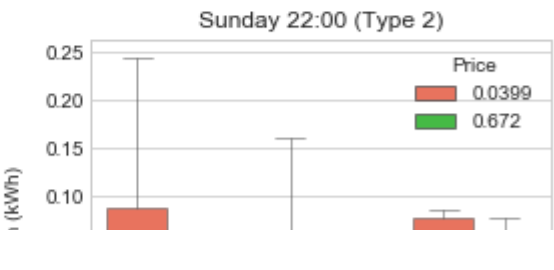

Sunday 21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)

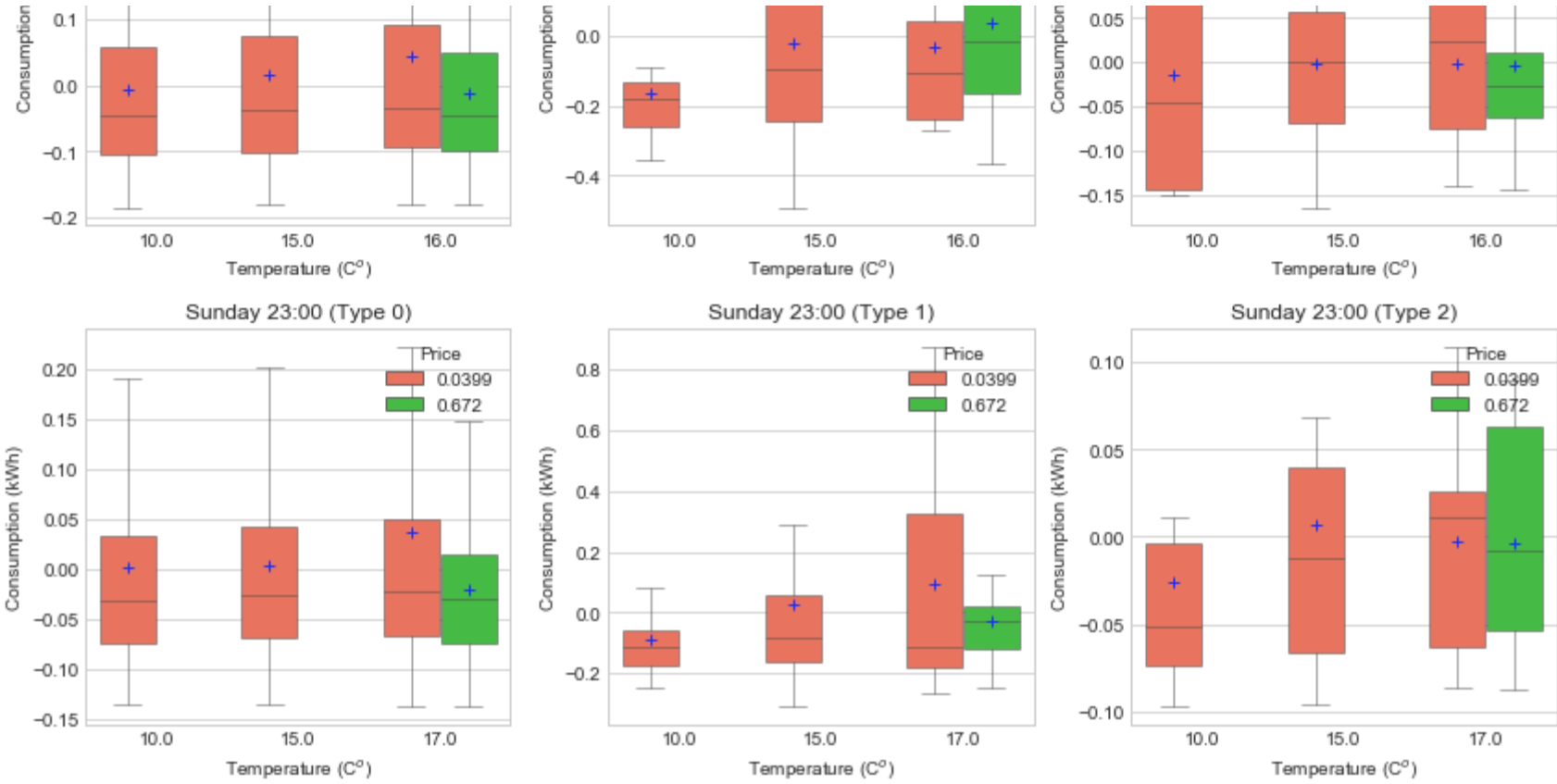

Sunday 21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)


Sunday 21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
10.0
15.0
16.0
19.0
Temperature (C°)


Sunday 22:00 (Type 0)
Price
0.0399
0.672


Sunday 22:00 (Type 1)
Price
0.0399
0.672

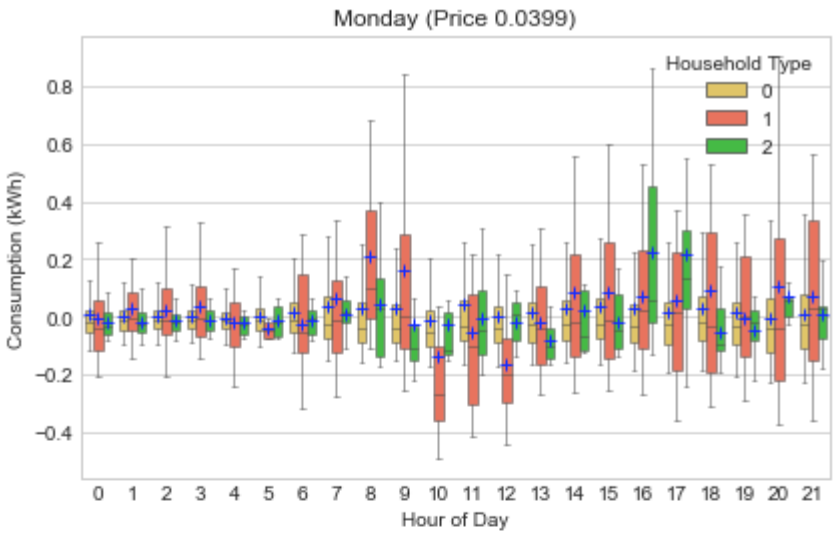

Sunday 22:00 (Type 2)
Price
0.0399
0.672

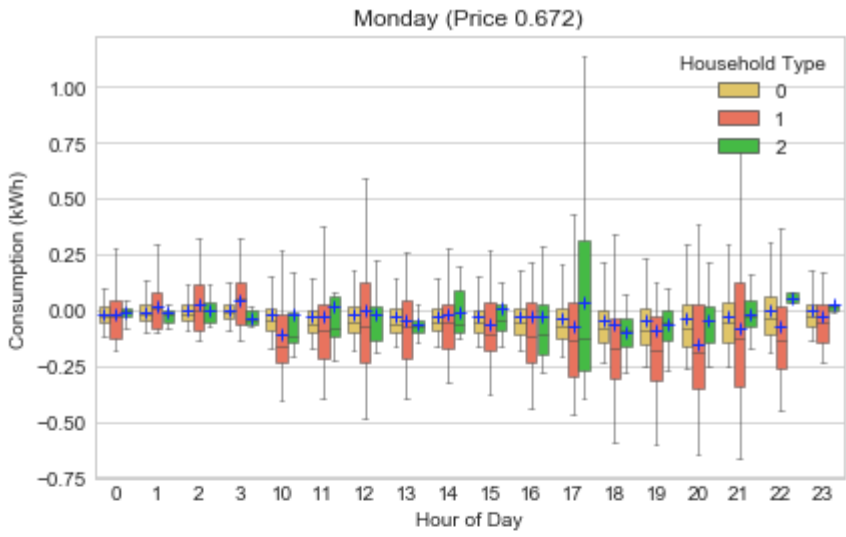


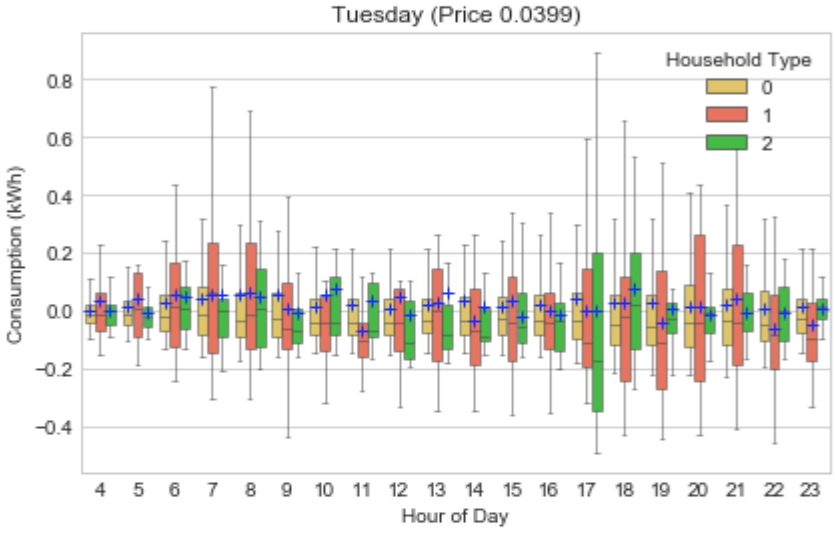
type2 may not be a good candidate for DR; type0 and type1 are good candidates but the program design must pay attention to the temperature impact on the demand responsiveness, we can expect with the same temperature, lower price would increase the consumption, higher price would decrease consumption, but this may not be true in different temperature situations. Conclusion: demand response plan must consider temperature impact, i.e., differentiated based on temperature situations instead of use a unified demand response plan for all temperatures.

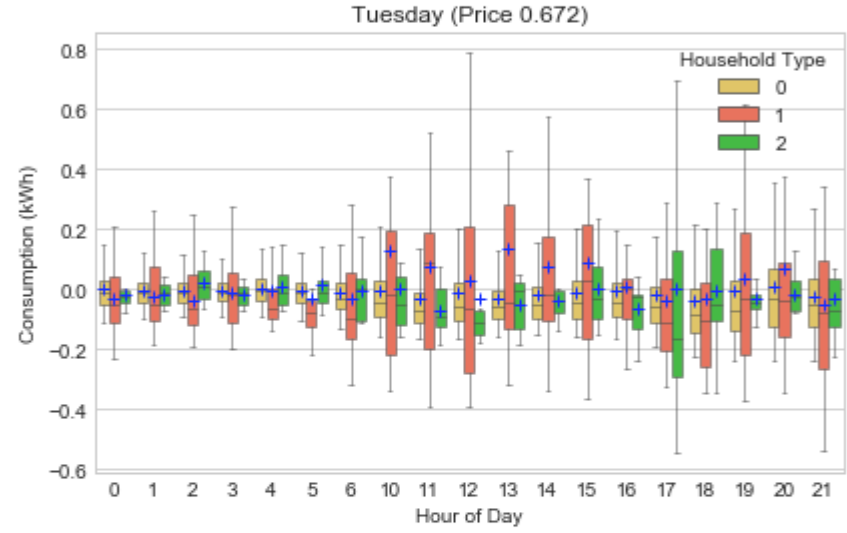
b) Day of week - hour : prices

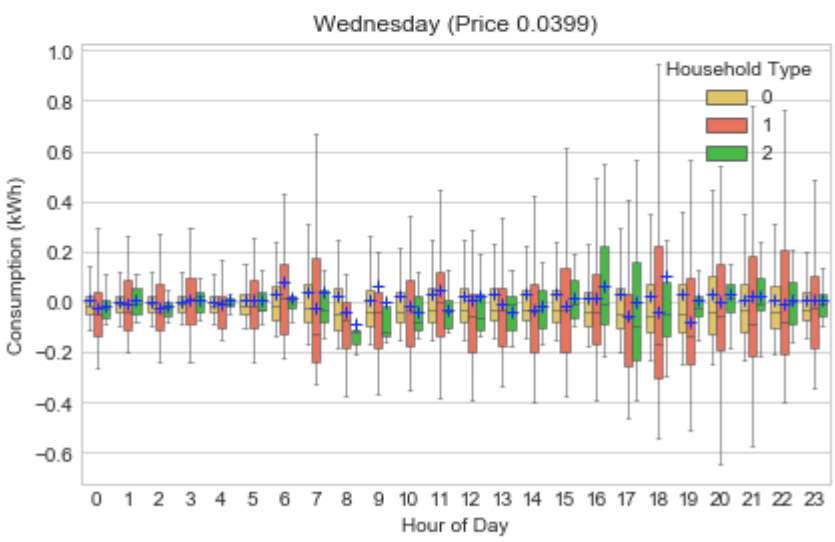
In [75]:
```
# hourly price responsiveness over days of week and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    for day_of_week in range(7):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * day_of_week + (p + 1)))
        df_day = df_all_res_long[(df_all_res_long['Day of week'] == day_of_week) & (df_all_res_long['Price']
== prices[p])]
        if df_day.shape[0] >= 1:
            sns.boxplot(x="Hour of day", y="Consumption", hue="Household type", data=df_day, ax=ax_Ntou[-1],
palette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","mar
kerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
            ax_Ntou[-1].set_title(weeks[day_of_week]  + ' (Price ' + str(prices[p]) + ')' )
            ax_Ntou[-1].set_xlabel(r'Hour of Day')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Household Type')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

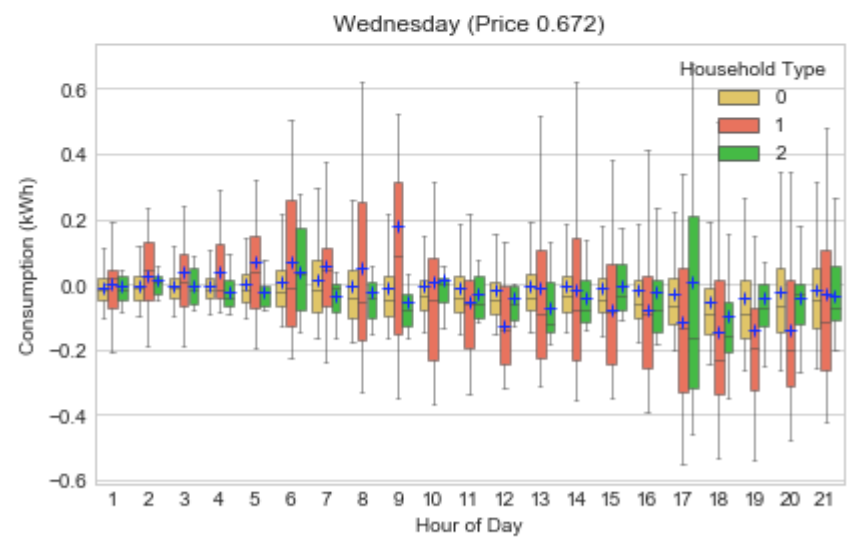

Monday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

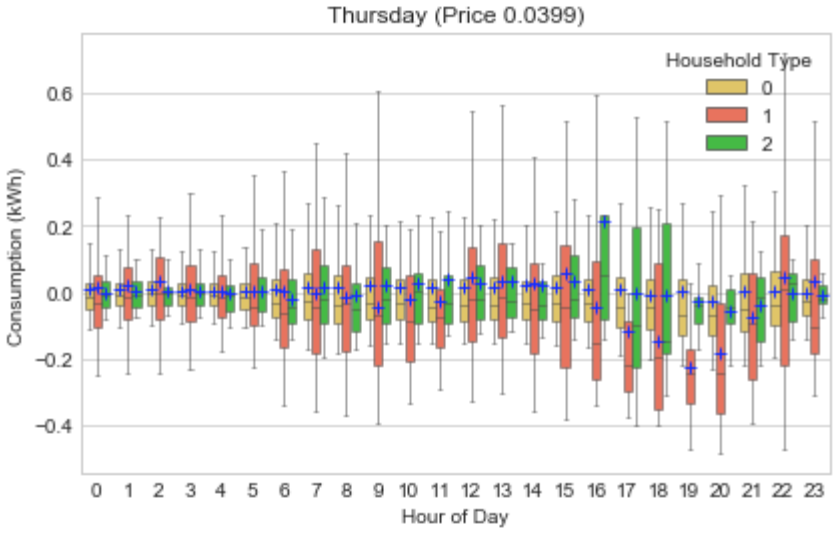

Monday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

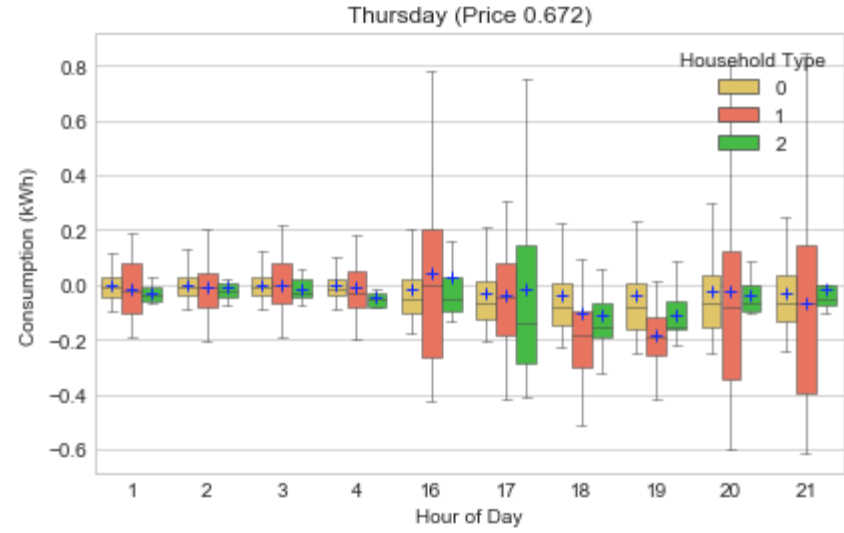

Tuesday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

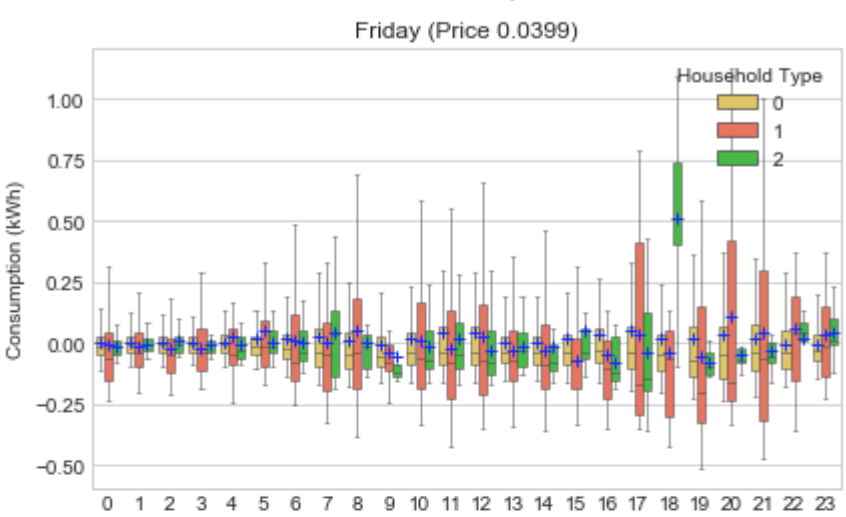

Tuesday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

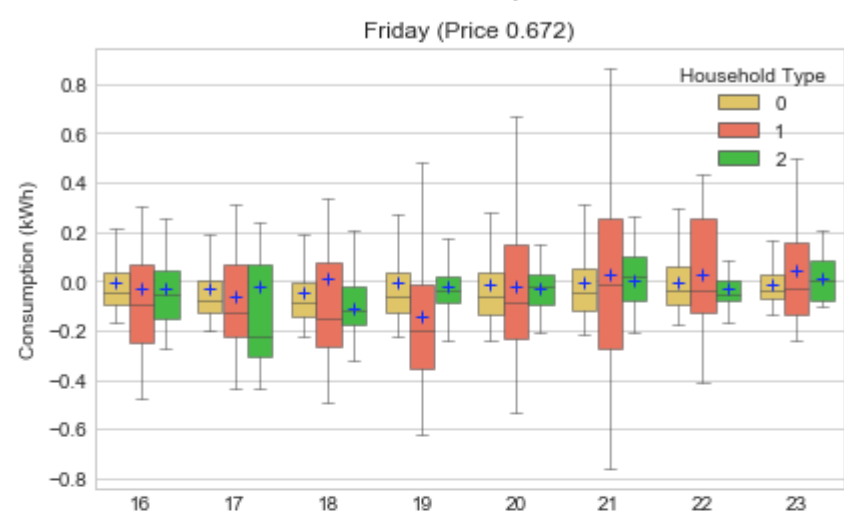

Wednesday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Hour of Day


Wednesday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Hour of Day


Thursday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

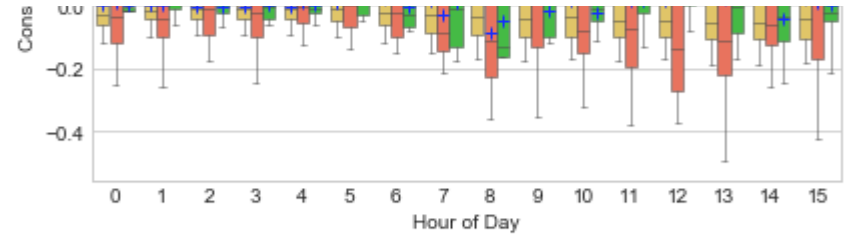

Thursday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

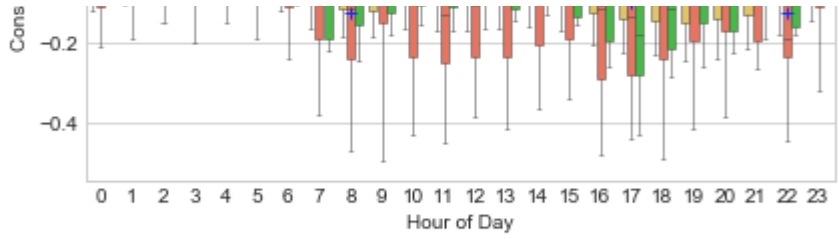

Friday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

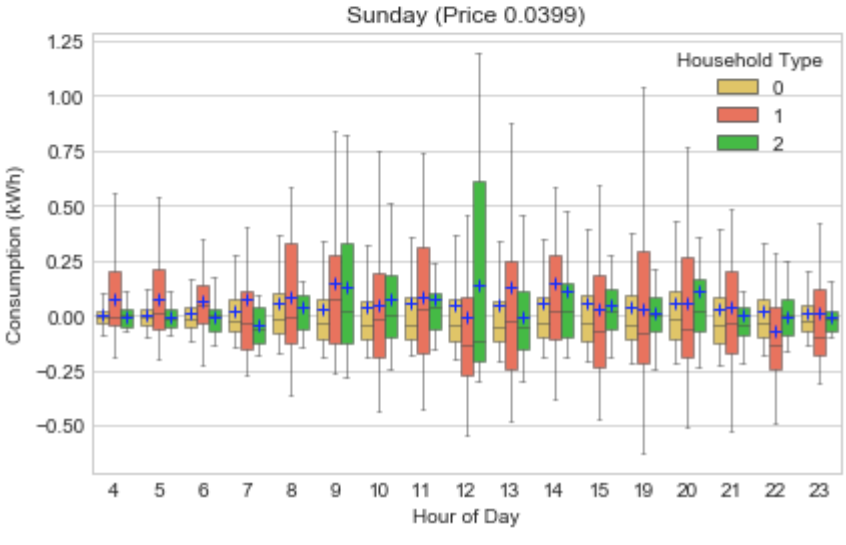

Friday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Hour of Day

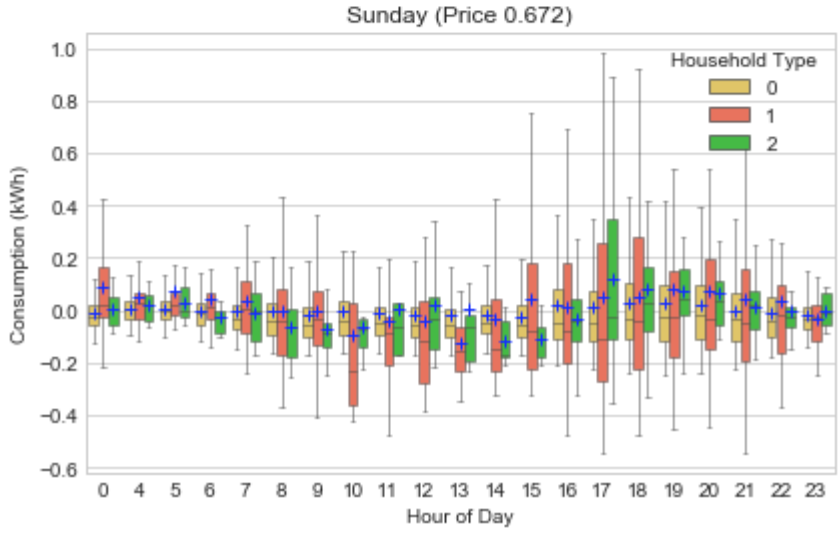
Saturday (Price 0.0399)
Household Type
0
1
2

Saturday (Price 0.672)
Household Type
0
1
2

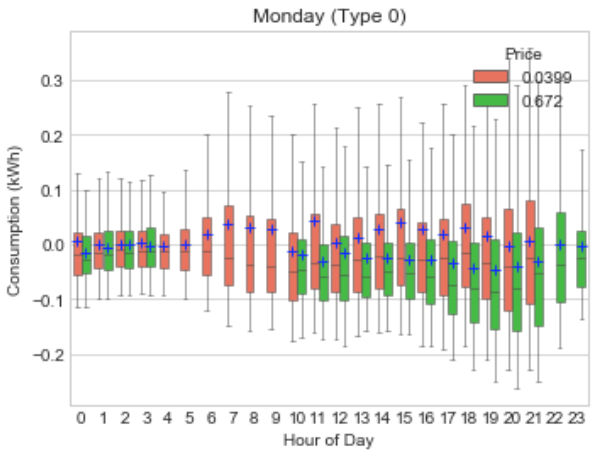

Cons
−0.2
−0.4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Hour of Day

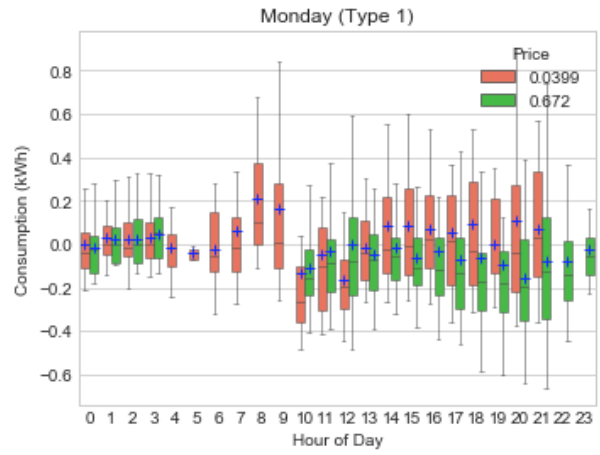

Cons
−0.2
−0.4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day

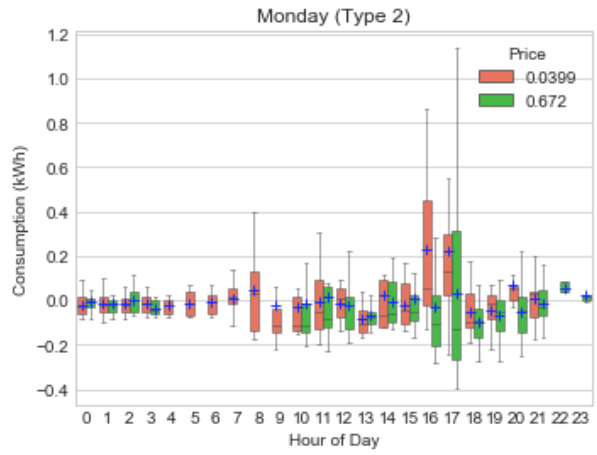

Sunday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
4 5 6 7 8 9 10 11 12 13 14 15 19 20 21 22 23
Hour of Day

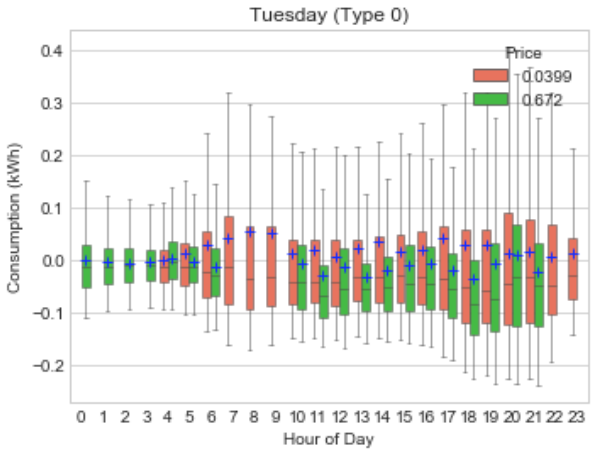

Sunday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day

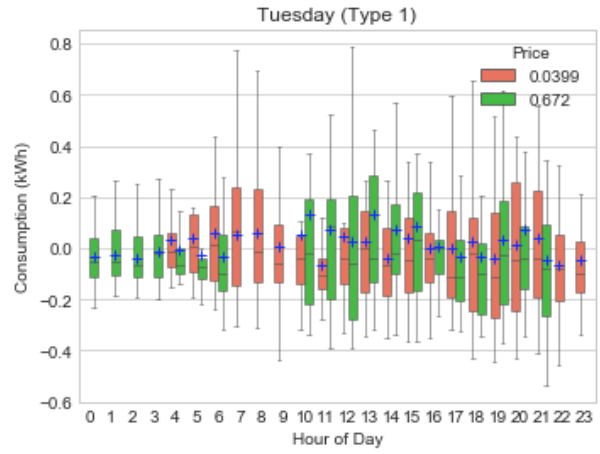
In [76]:
```
# price comparison
# hourly price responsiveness over days of week and prices
fig_all = plt.figure(figsize = (15,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    for day_of_week in range(7):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * day_of_week + (g + 1)))
        df_day = df_all_res_long[(df_all_res_long['Day of week'] == day_of_week) & (df_all_res_long['Household type'] == g)]
        if df_day.shape[0] >= 1:
            sns.boxplot(x="Hour of day", y="Consumption", hue="Price", data=df_day, ax=ax_Ntou[-1], palette=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
            ax_Ntou[-1].set_title(weeks[day_of_week]  + ' (Type ' + str(g) + ')' )
            ax_Ntou[-1].set_xlabel(r'Hour of Day')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Price')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

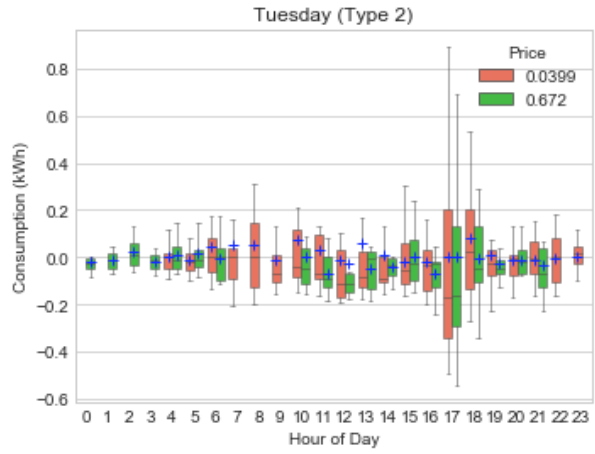

Monday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

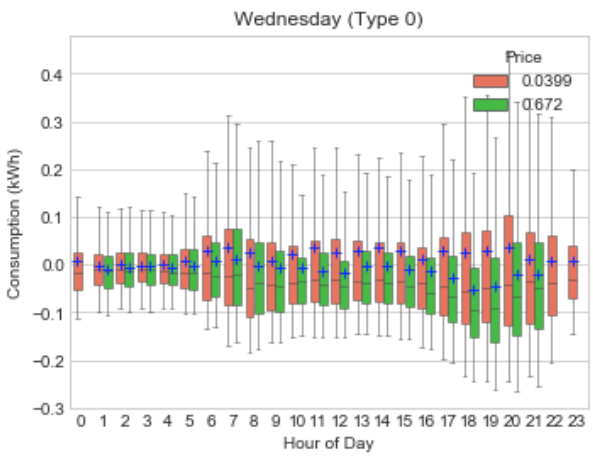

Monday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

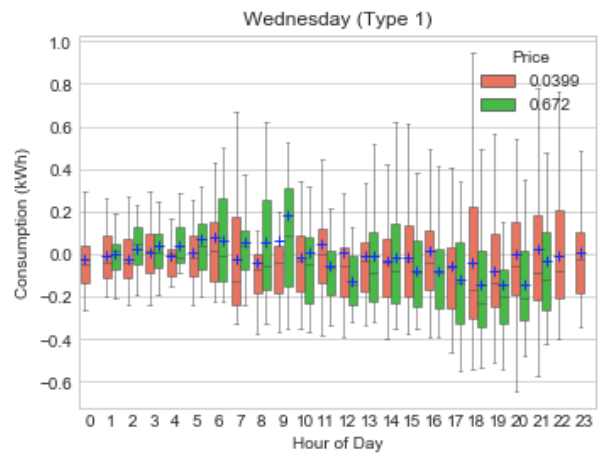

Monday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

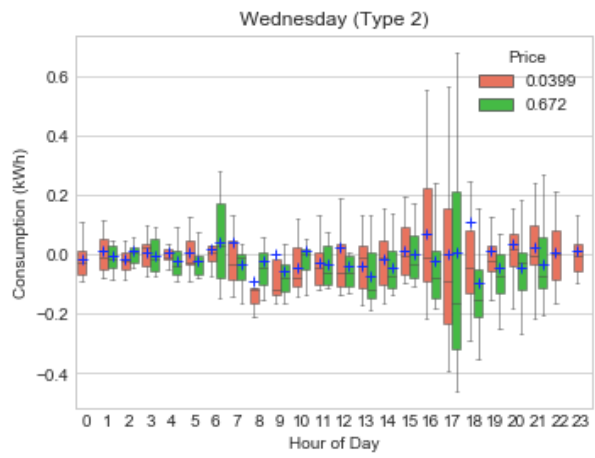

Tuesday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

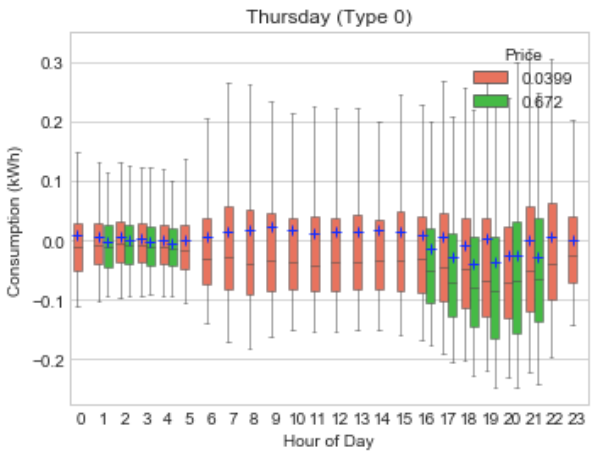

Tuesday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

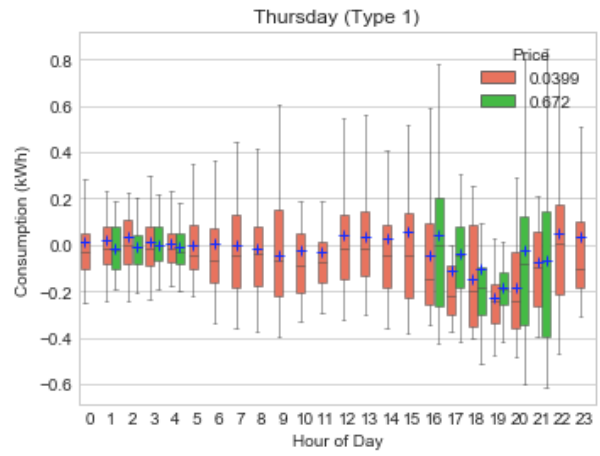

Tuesday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

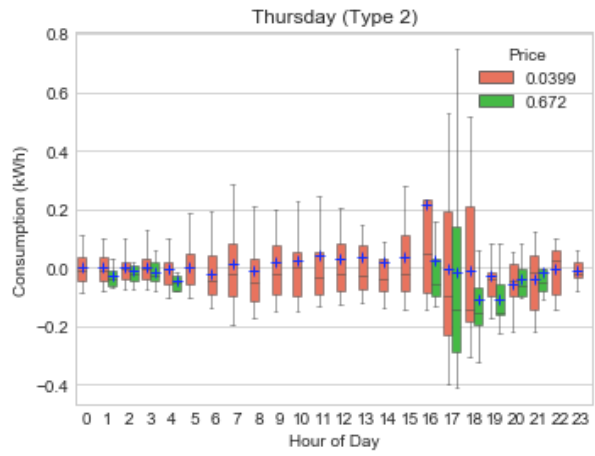

Wednesday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

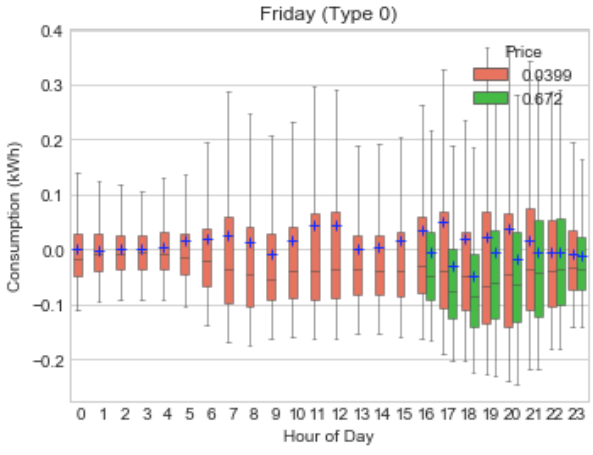

Wednesday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

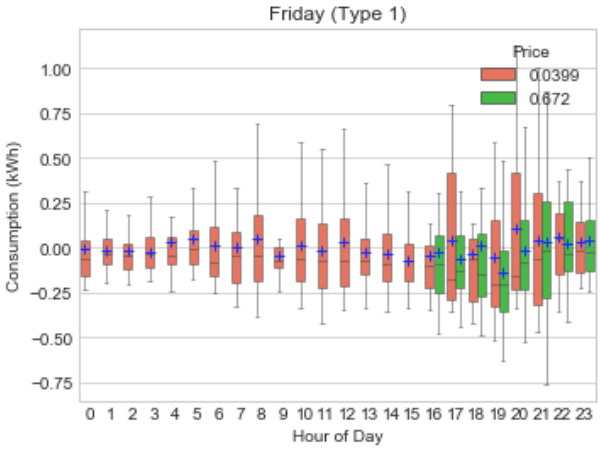

Wednesday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

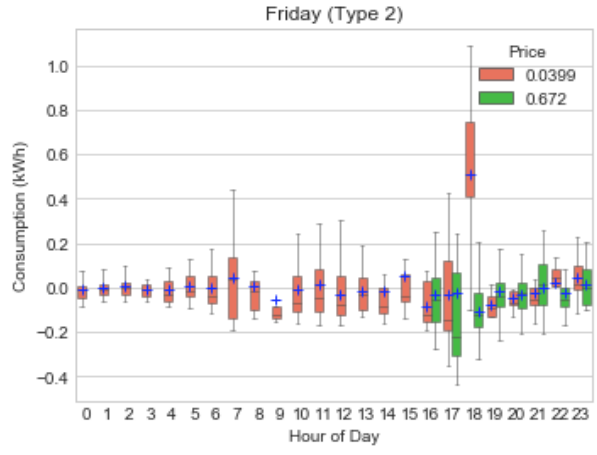

Thursday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

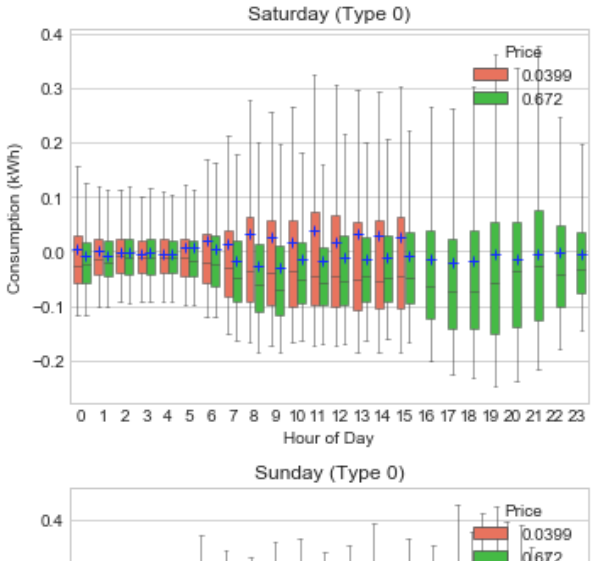

Thursday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

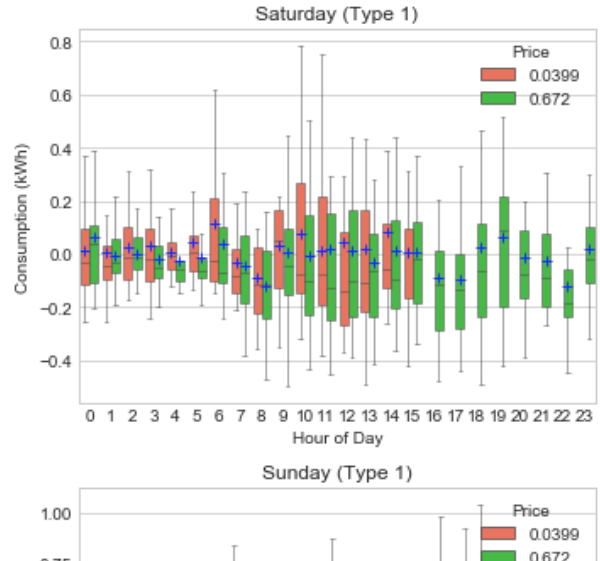

Thursday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

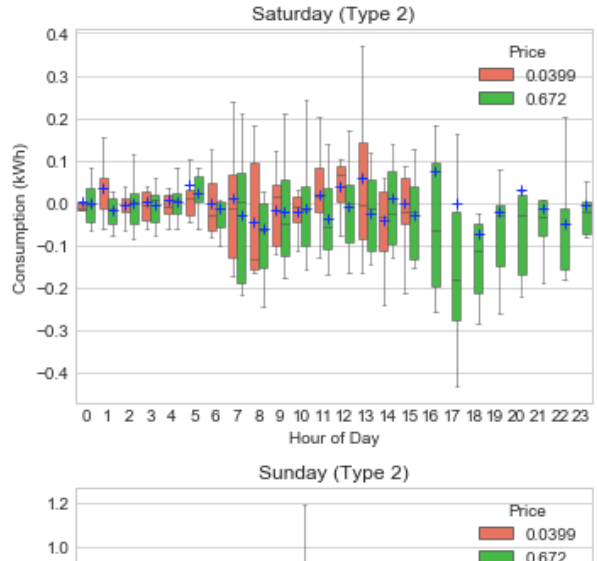

Friday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

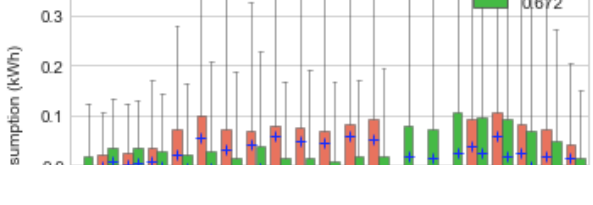

Friday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

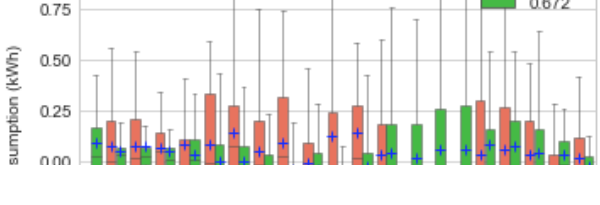

Friday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

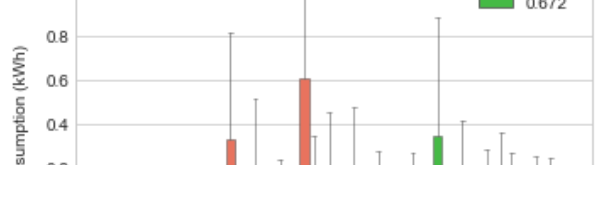

Saturday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

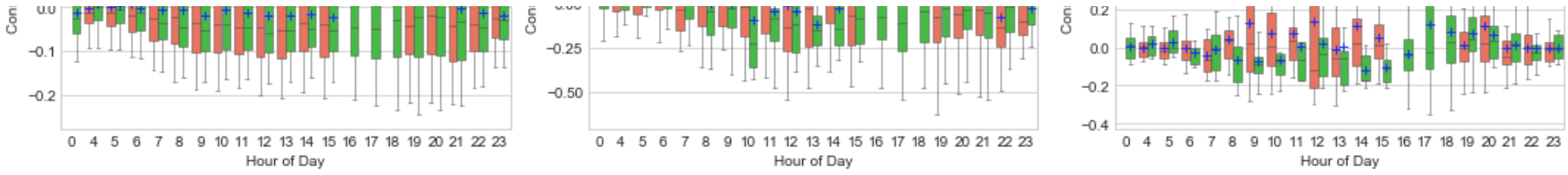

Saturday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day


Saturday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day


Sunday (Type 0)
Price
0.0399
0.672
Hour of Day


Sunday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Hour of Day

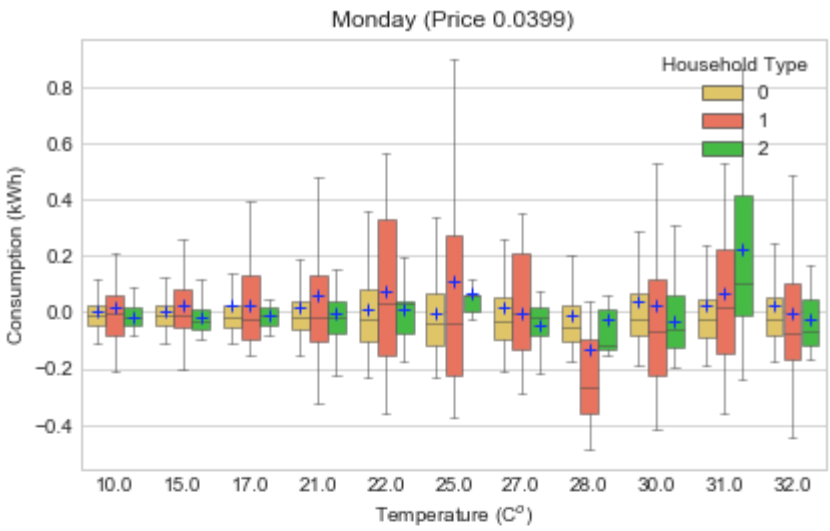

Sunday (Type 2)
Price
0.0399
0.672
Consumption (kWh)

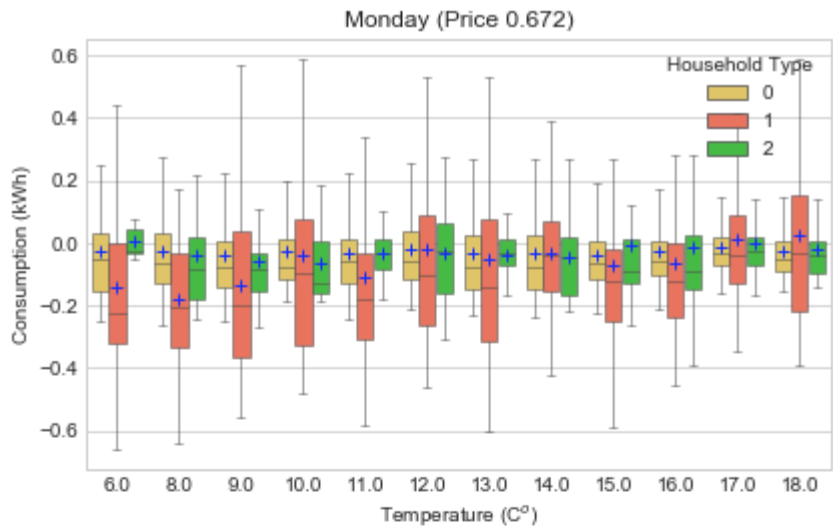


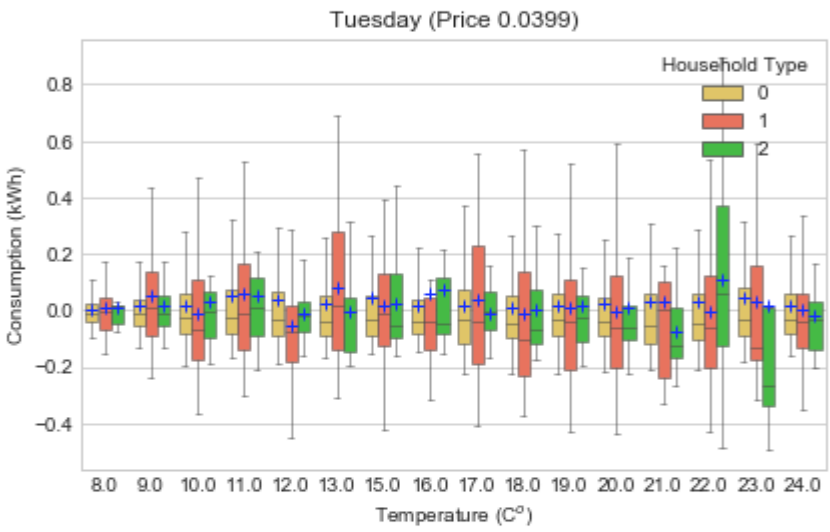
Again, type 0 shows better price reponsiveness than type1 and 2.

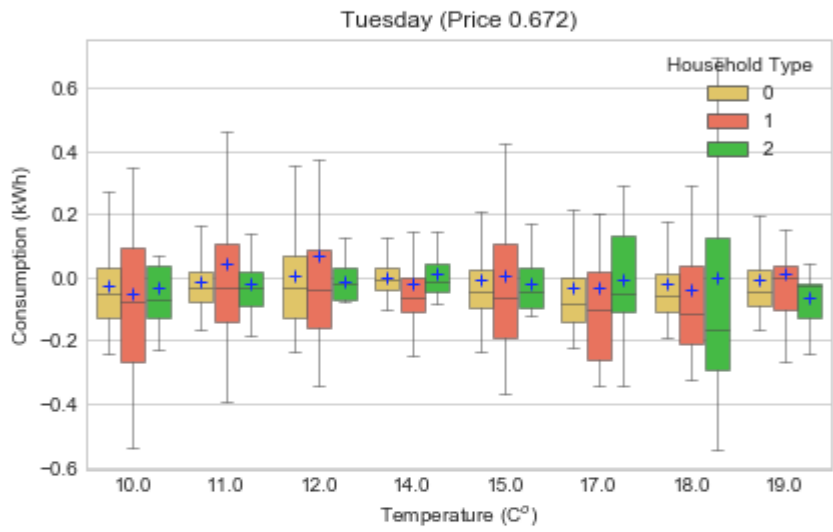
c) Day of week - temperature : prices

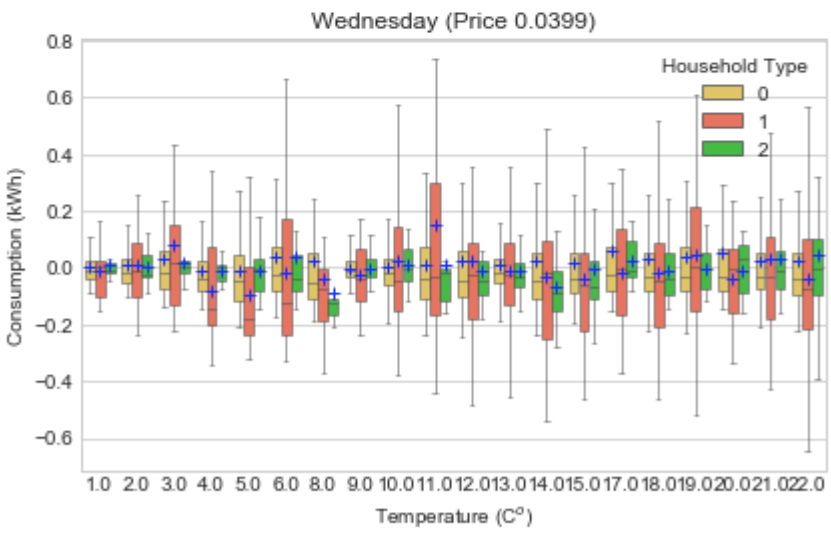
In [77]:
```
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    for day_of_week in range(7):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * day_of_week + (p + 1)))
        df_day = df_all_res_long[(df_all_res_long['Day of week'] == day_of_week) & (df_all_res_long['Price']
== prices[p])]
        if df_day.shape[0] >= 1:
            sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_day, ax=ax_Ntou[-1], palett
e=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfac
ecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
            ax_Ntou[-1].set_title(weeks[day_of_week]  + ' (Price ' + str(prices[p]) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Household Type')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

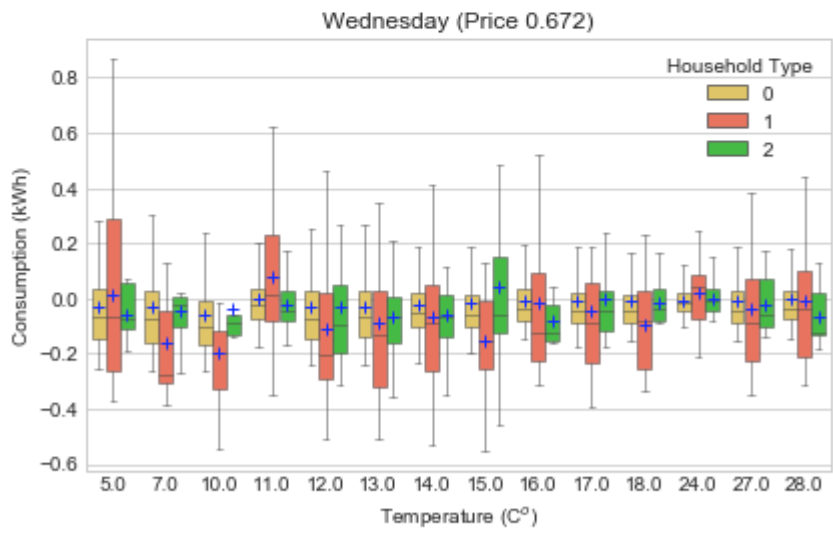

Monday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

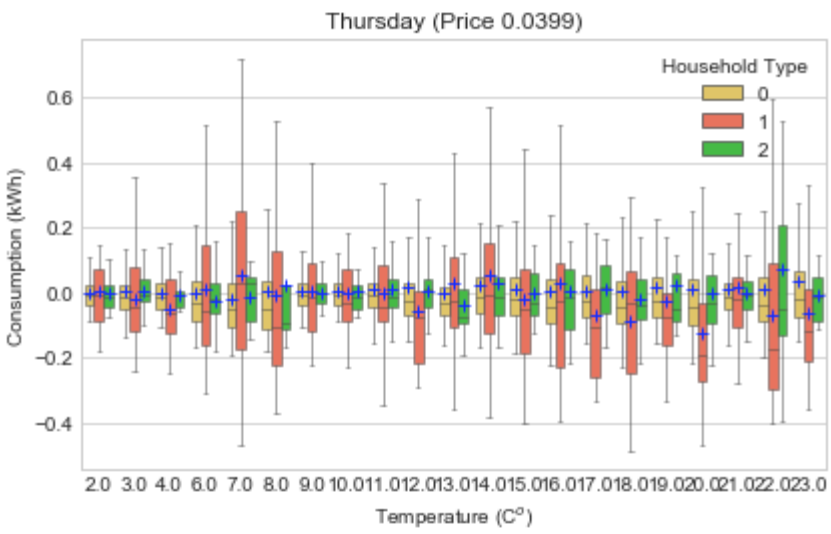

Monday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

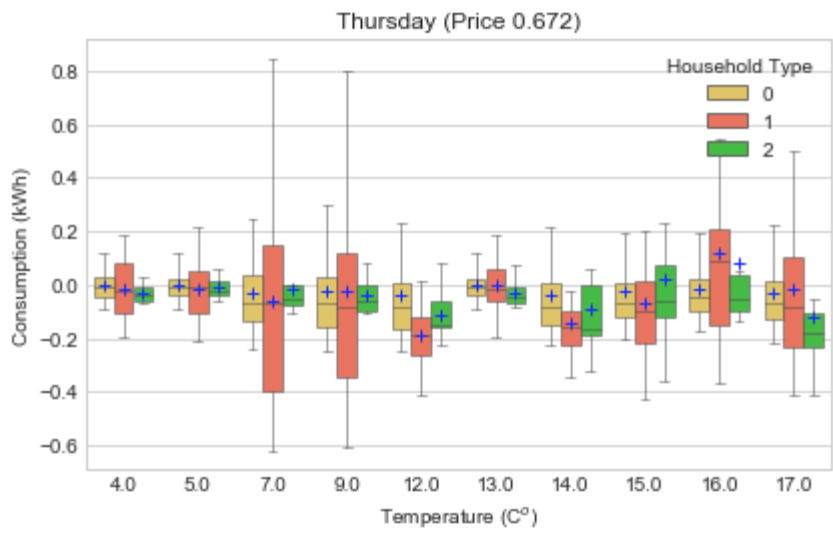

Tuesday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

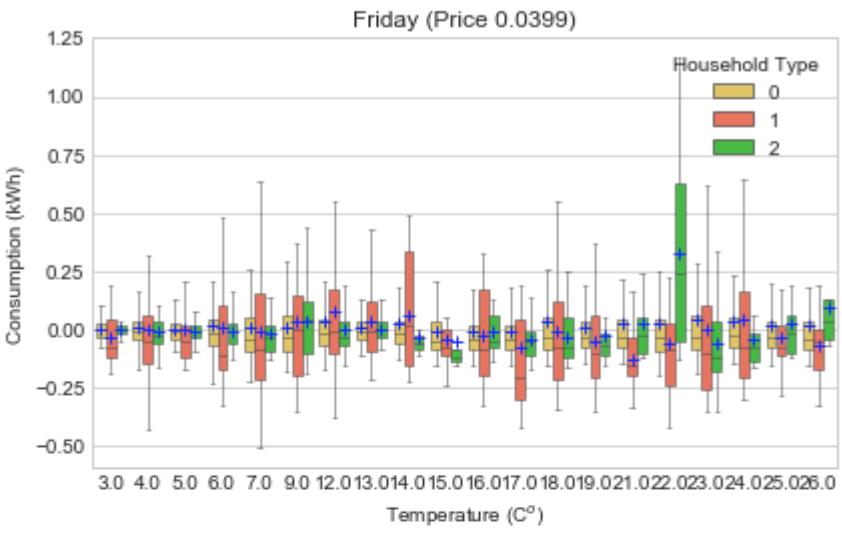

Tuesday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

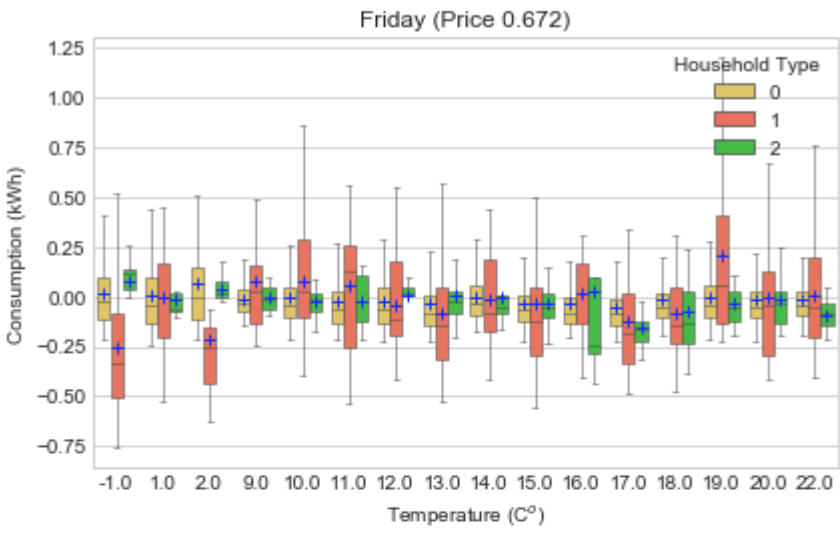

Wednesday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

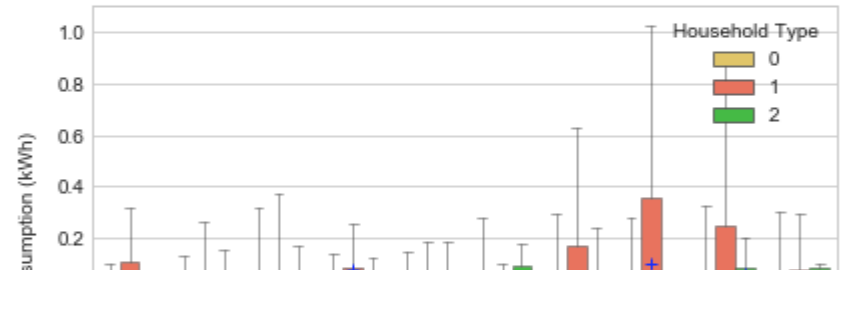

Wednesday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

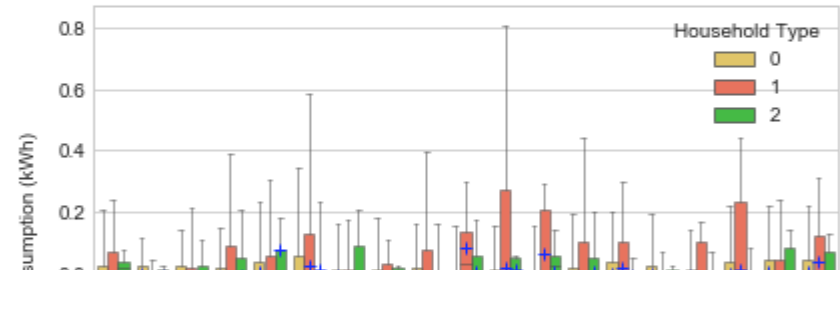

Thursday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

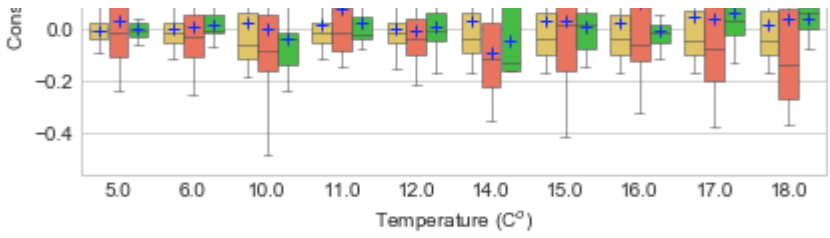

Thursday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

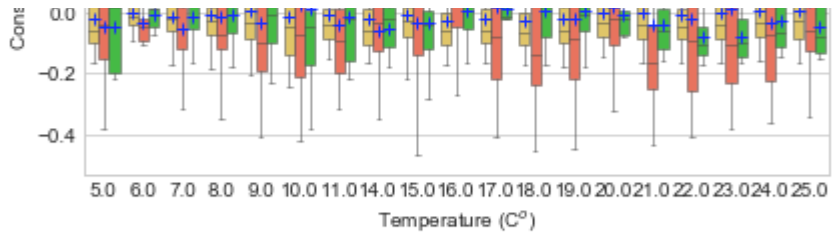

Friday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

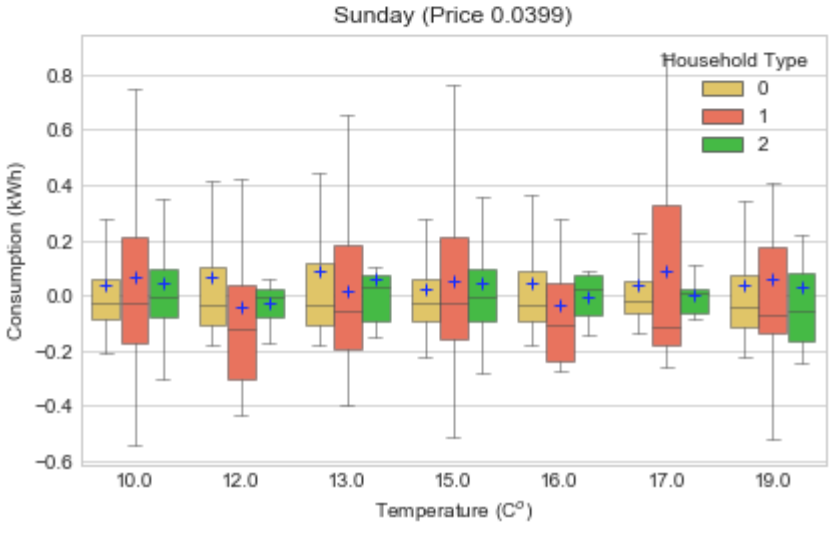

Friday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C^o)

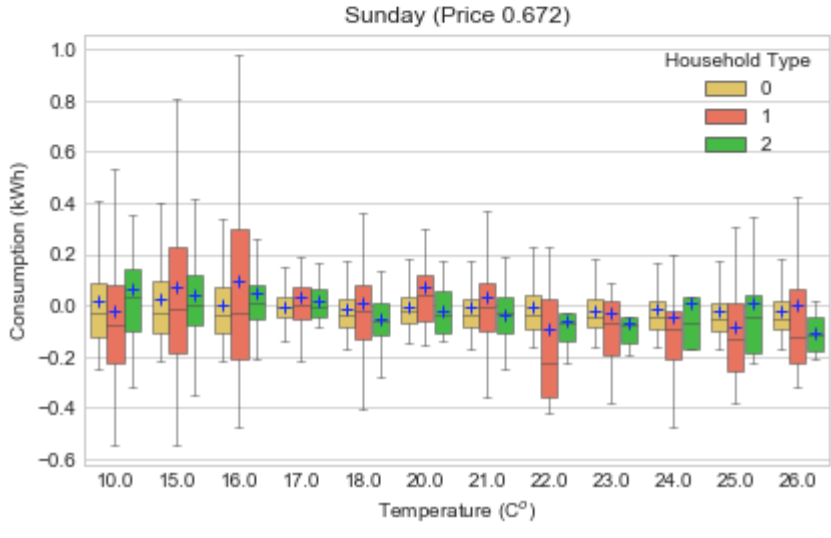

Saturday (Price 0.0399)
Household Type
0
1
2
Temperature (C^o)


Saturday (Price 0.672)
Household Type
0
1
2
Temperature (C^o)

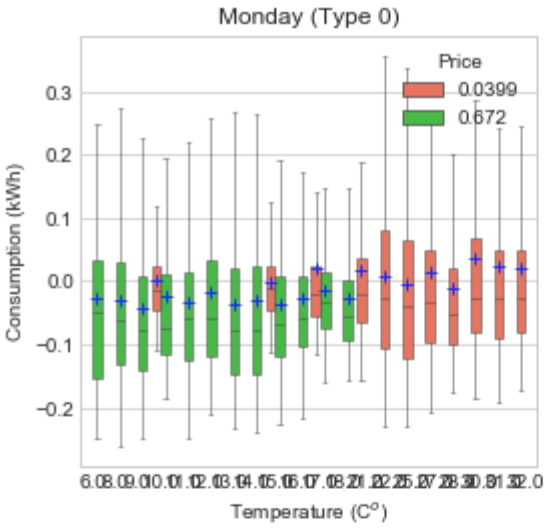

Cons
0.0
−0.2
−0.4
5.0
6.0
10.0
11.0
12.0
14.0
15.0
16.0
17.0
18.0
Temperature (C^o)

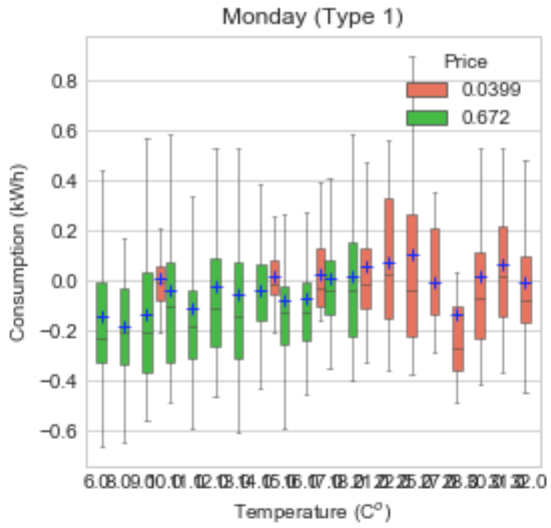

Cons
0.0
−0.2
−0.4
5.0
6.0
7.0
8.0
9.0
10.0
11.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0
24.0
25.0
Temperature (C^o)

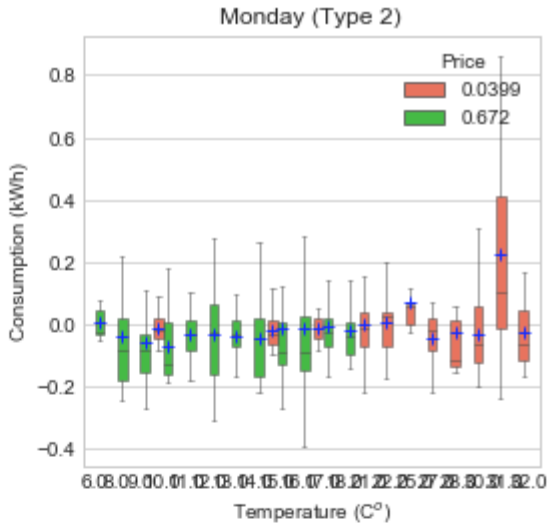

Sunday (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
10.0
12.0
13.0
15.0
16.0
17.0
19.0
Temperature (C^o)

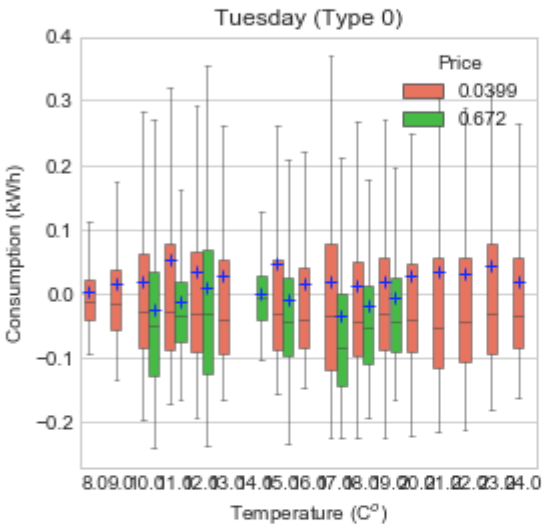

Sunday (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
10.0
15.0
16.0
17.0
18.0
20.0
21.0
22.0
23.0
24.0
25.0
26.0
Temperature (C^o)

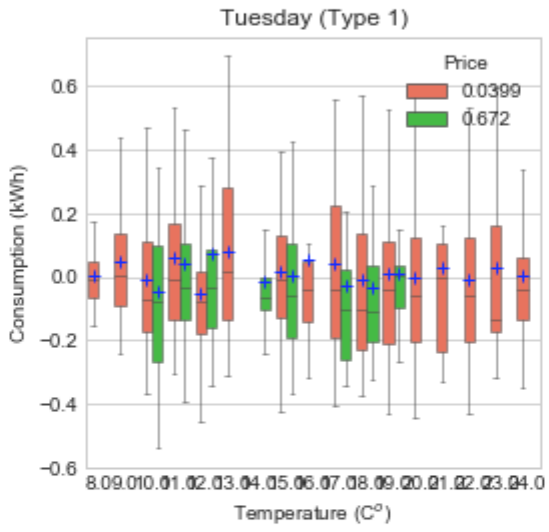
In [78]:
```
# price comparision
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
weeks = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    for day_of_week in range(7):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3 * day_of_week + (g + 1)))
        df_day = df_all_res_long[(df_all_res_long['Day of week'] == day_of_week) & (df_all_res_long['Household type'] == g)]
        if df_day.shape[0] >= 1:
            sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_day, ax=ax_Ntou[-1], palette=['tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
            ax_Ntou[-1].set_title(weeks[day_of_week]  + ' (Type ' + str(g) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Price')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

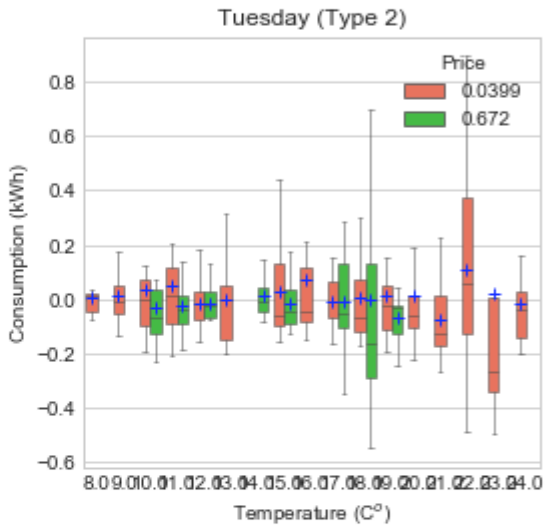

Monday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

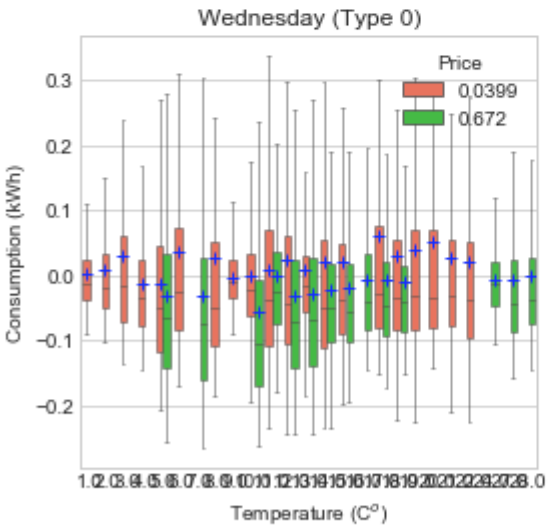

Monday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

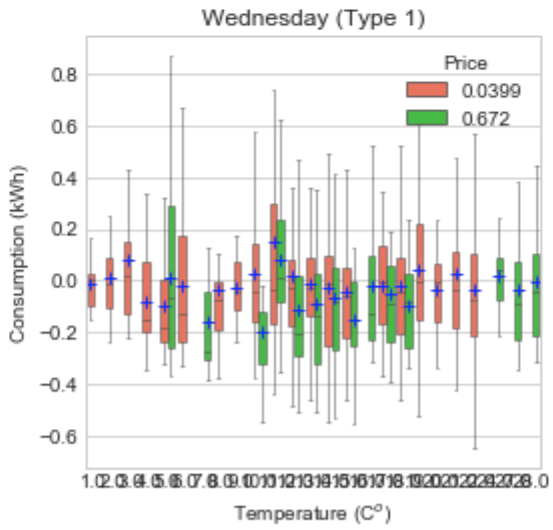

Monday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

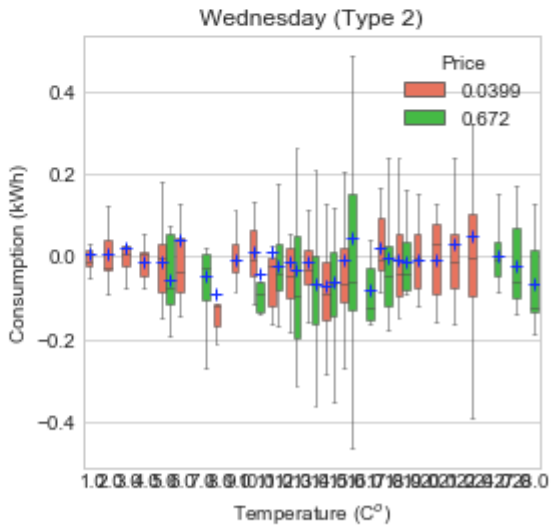

Tuesday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

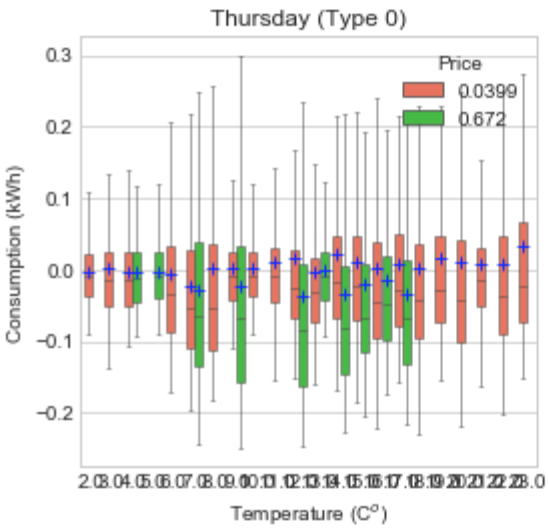

Tuesday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

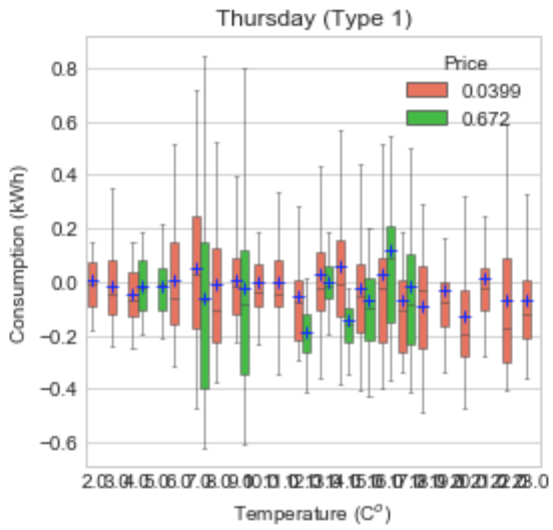

Tuesday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

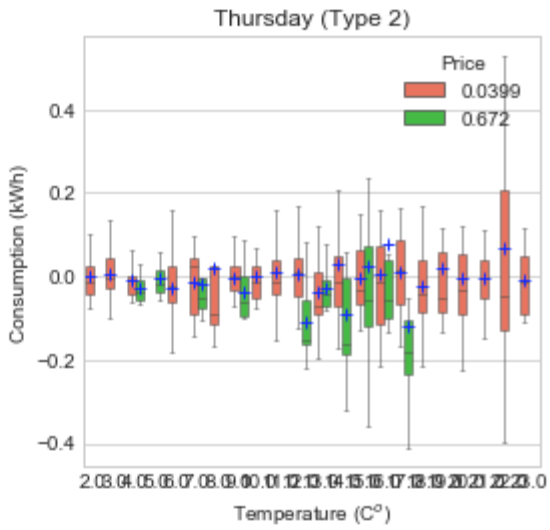

Wednesday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

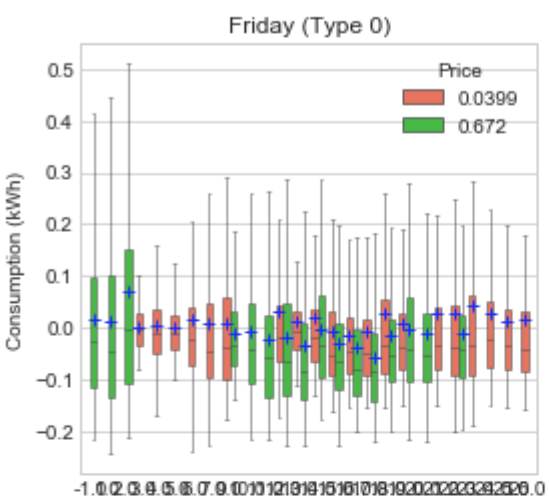

Wednesday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

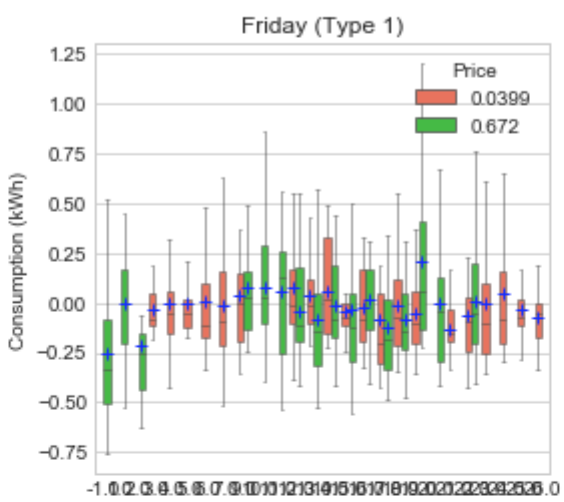

Wednesday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

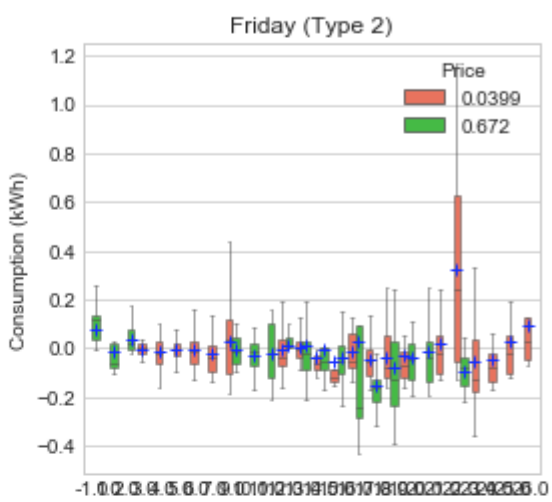

Thursday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

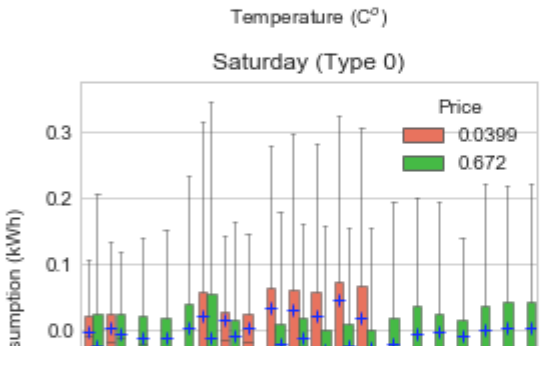

Thursday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

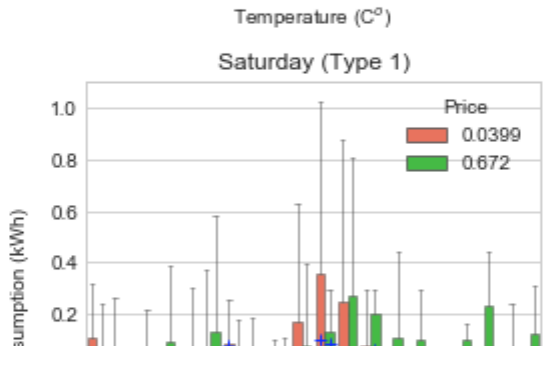

Thursday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

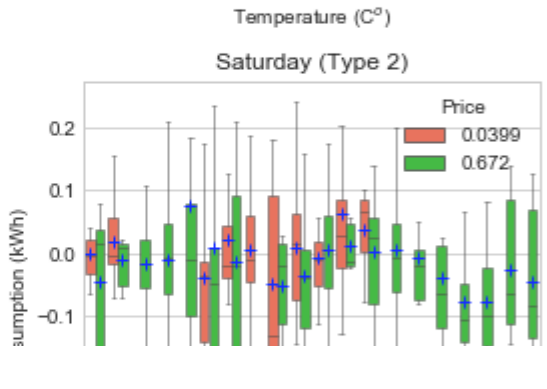

Friday (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

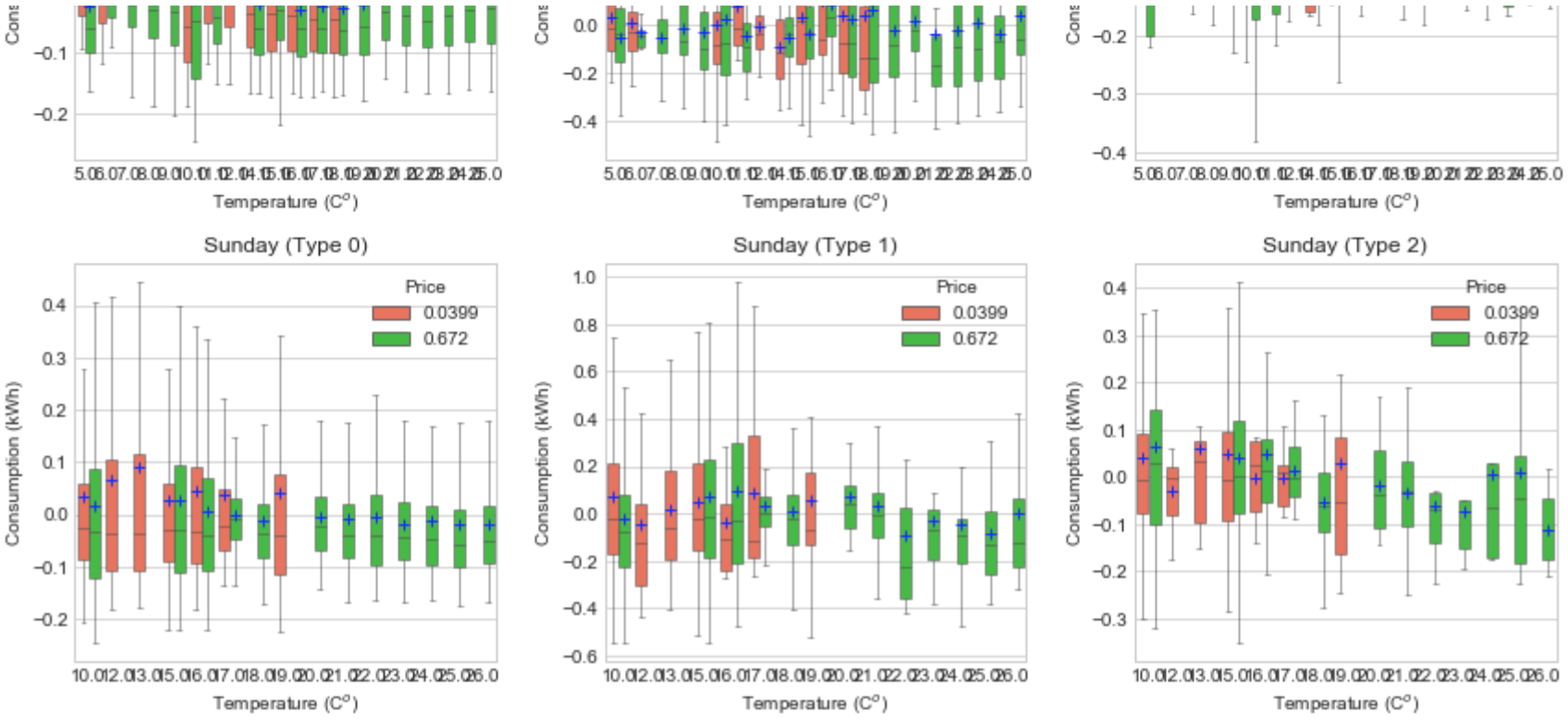

Friday (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)


Friday (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)


Saturday (Type 0)
Price
0.0399
0.672

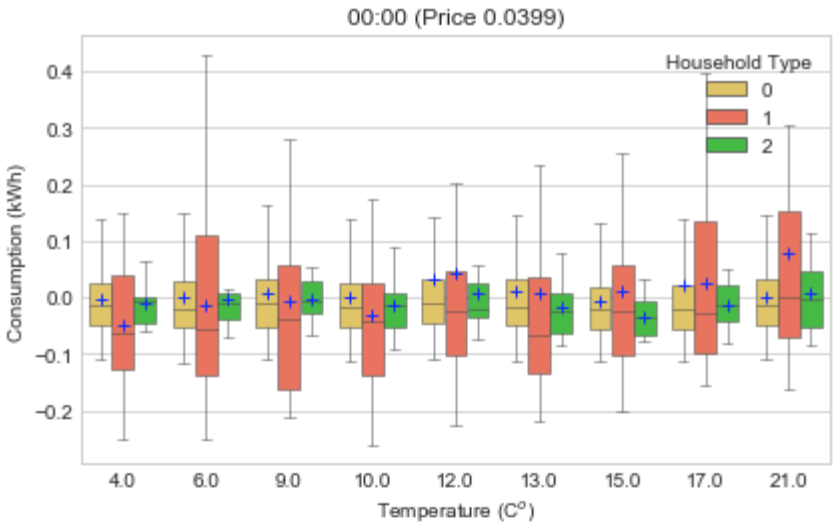

Saturday (Type 1)
Price
0.0399
0.672

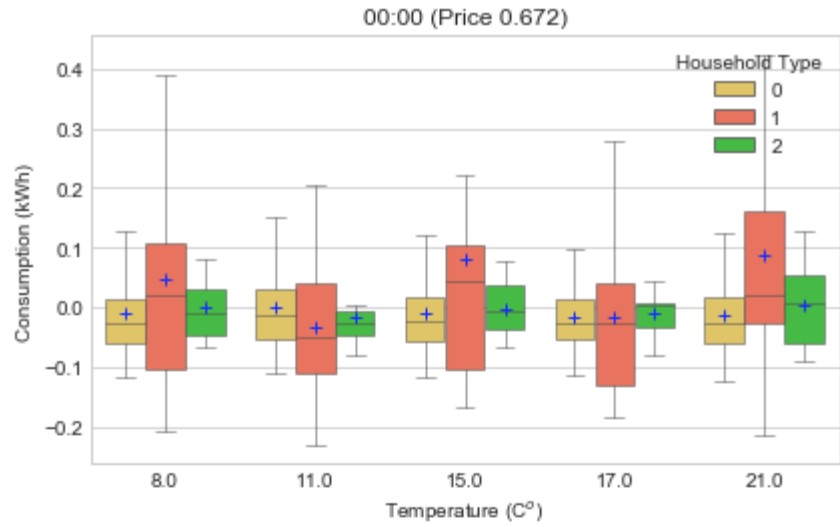

Saturday (Type 2)
Price
0.0399
0.672

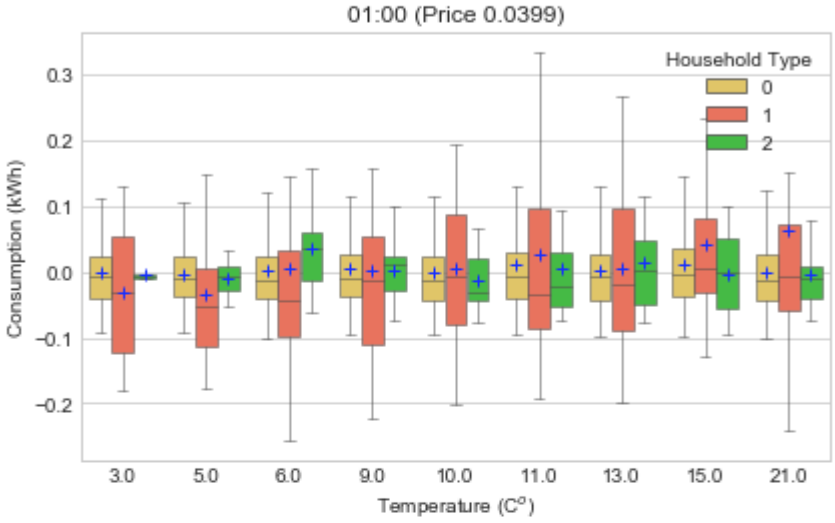


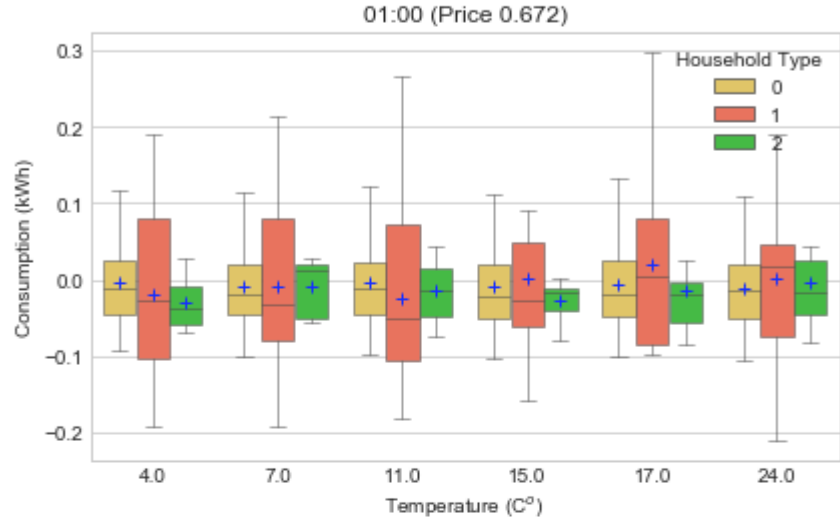
d) Hour of day - temperature : prices

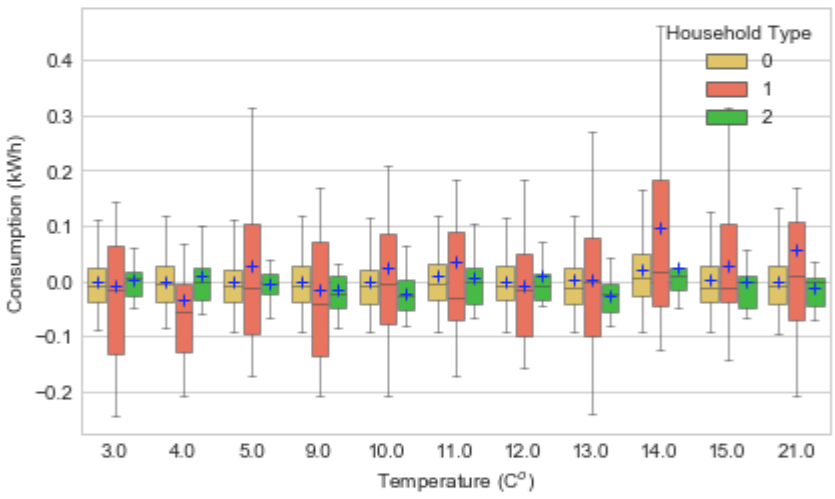
In [79]:
```
# hourly price responsiveness over temperatures and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 2, 2 * i + (p + 1)))
        df_hour = df_all_res_long[(df_all_res_long['Hour of day'] == i) & (df_all_res_long['Price'] == prices
[p])]
#         min_consumption = df_hour['Consumption'].min()
#         max_consumption = df_hour['Consumption'].max()
        if df_hour.shape[0] >= 1:
            sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_hour, ax=ax_Ntou[-1], palet
te=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfa
cecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
            if i < 9:
                ax_Ntou[-1].set_title('0' + str(i) + ':00' + ' (Price ' + str(prices[p]) + ')' )
            else:
                ax_Ntou[-1].set_title(str(i) + ':00' + ' (Price ' + str(prices[p]) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Household Type')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```

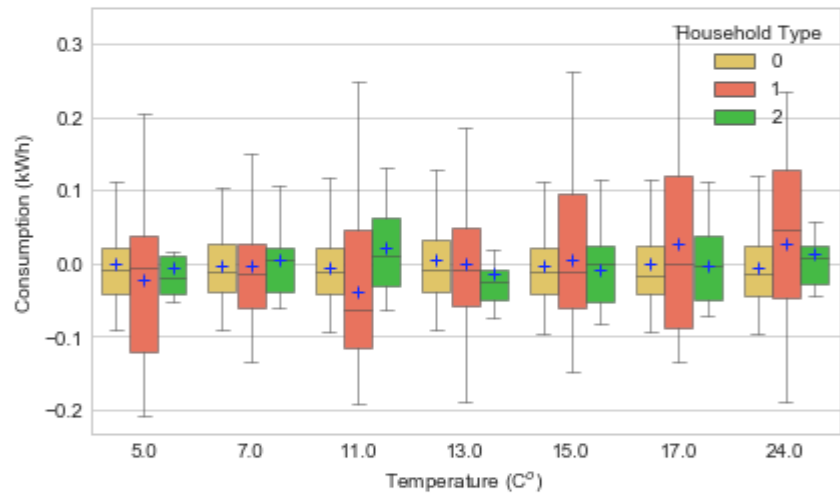

00:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
6.0
9.0
10.0
12.0
13.0
15.0
17.0
21.0
Temperature (C°)

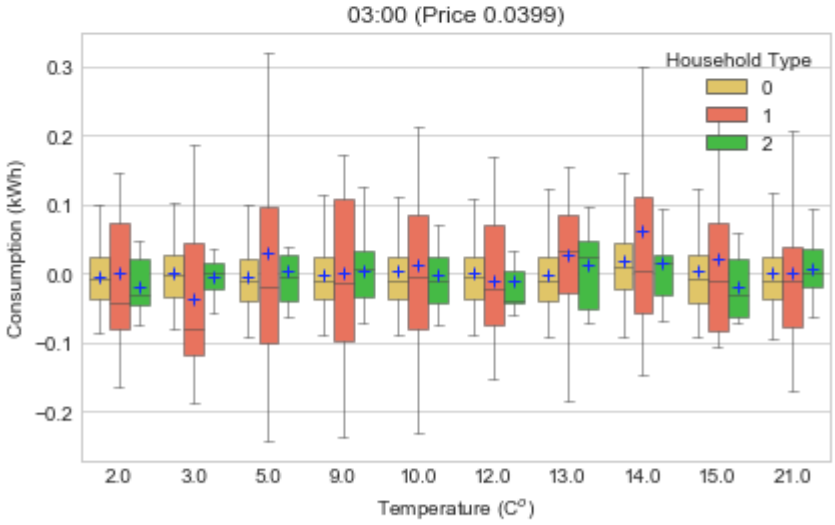

00:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
8.0
11.0
15.0
17.0
21.0
Temperature (C°)

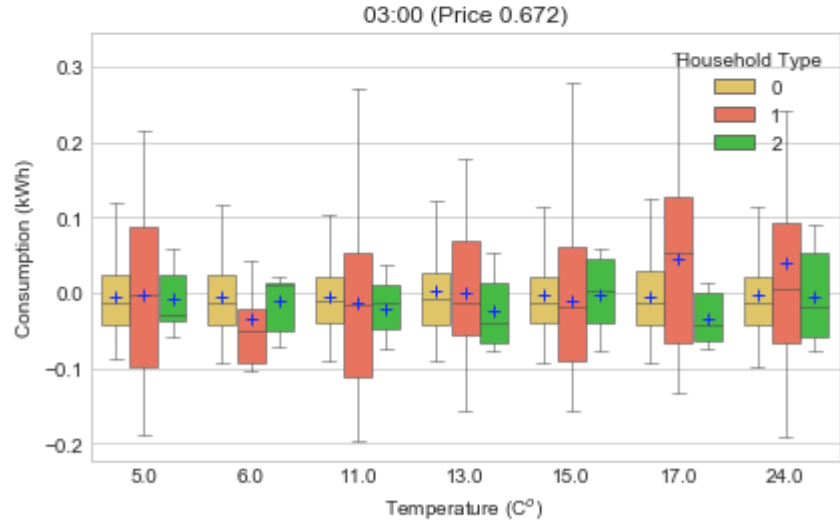

01:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
5.0
6.0
9.0
10.0
11.0
13.0
15.0
21.0
Temperature (C°)

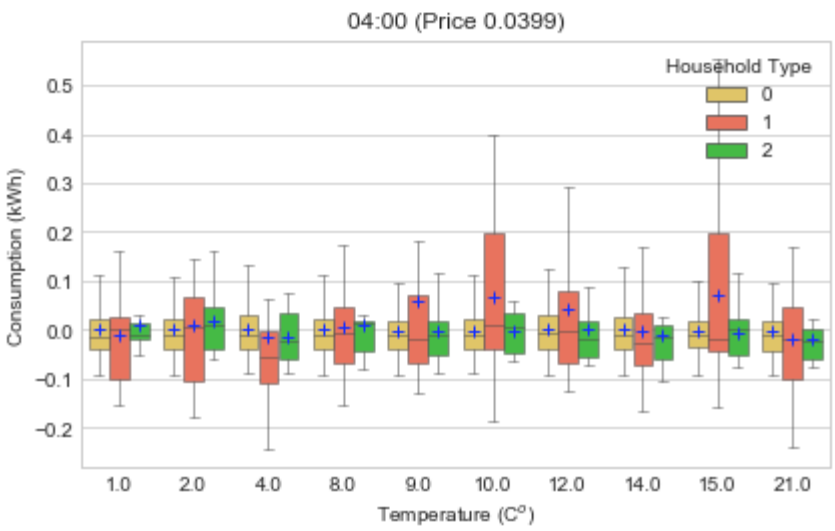

01:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
4.0
7.0
11.0
15.0
17.0
24.0
Temperature (C°)

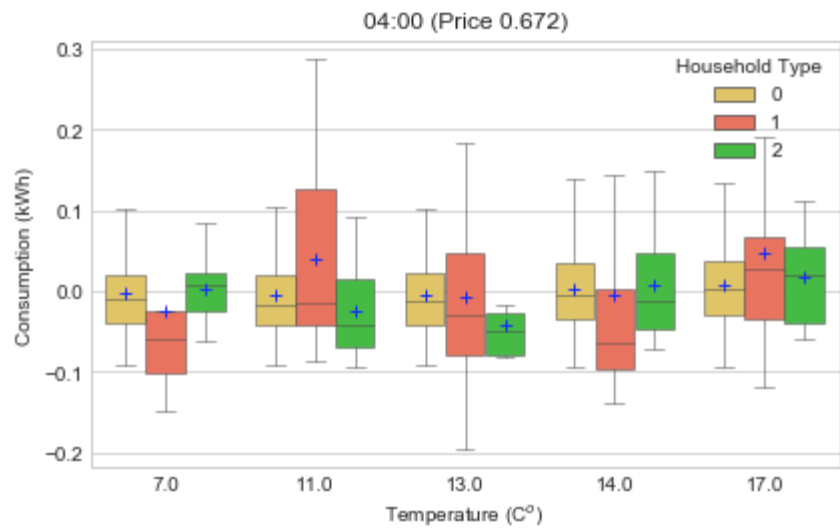

02:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
3.0
4.0
5.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
21.0
Temperature (C°)

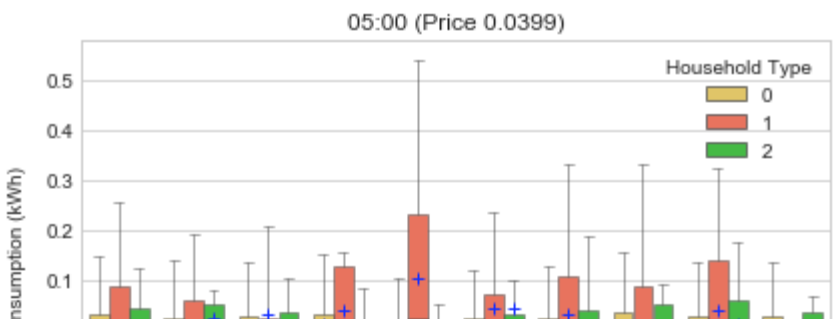

02:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
5.0
7.0
11.0
13.0
15.0
17.0
24.0
Temperature (C°)

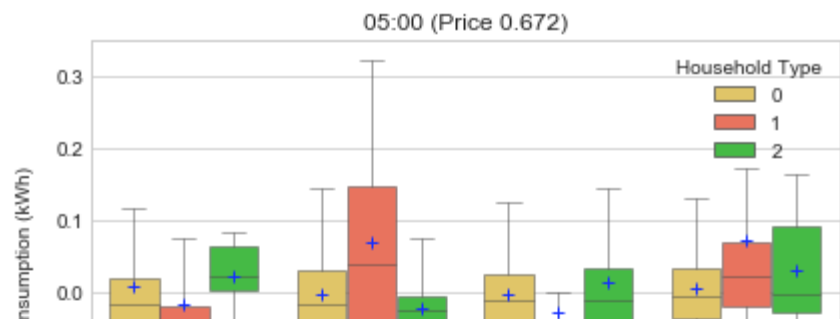

03:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
2.0
3.0
5.0
9.0
10.0
12.0
13.0
14.0
15.0
21.0
Temperature (C°)

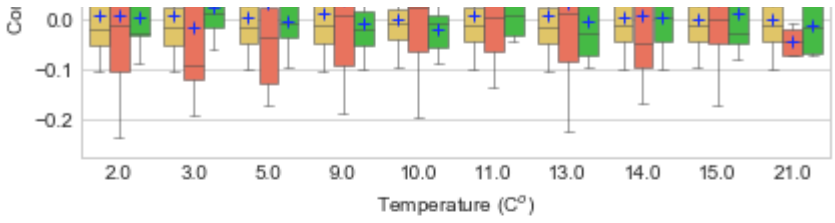

03:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
5.0
6.0
11.0
13.0
15.0
17.0
24.0
Temperature (C°)

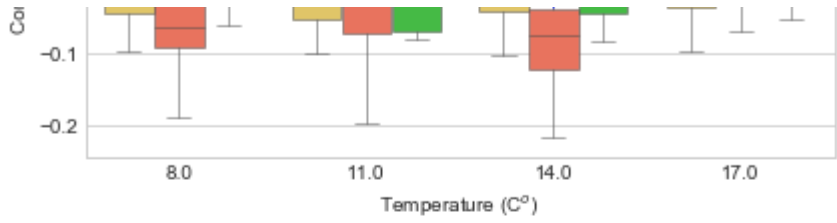

04:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
1.0
2.0
4.0
8.0
9.0
10.0
12.0
14.0
15.0
21.0
Temperature (C°)

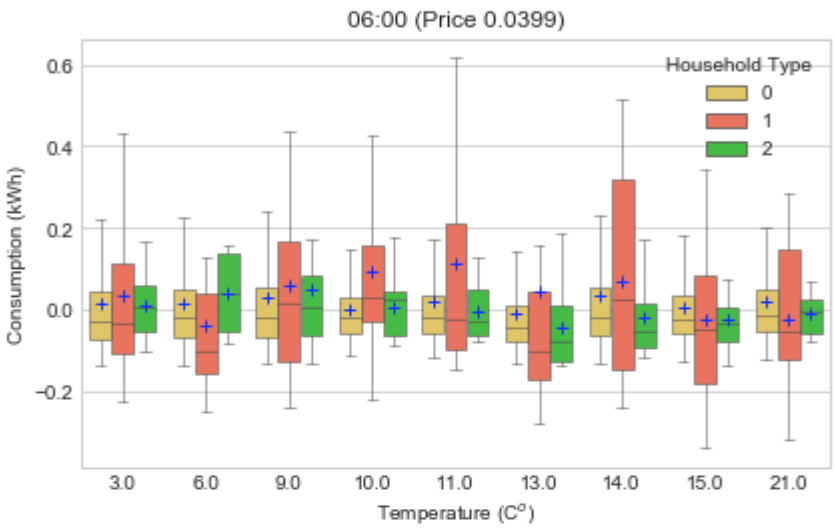

04:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0
11.0
13.0
14.0
17.0
Temperature (C°)

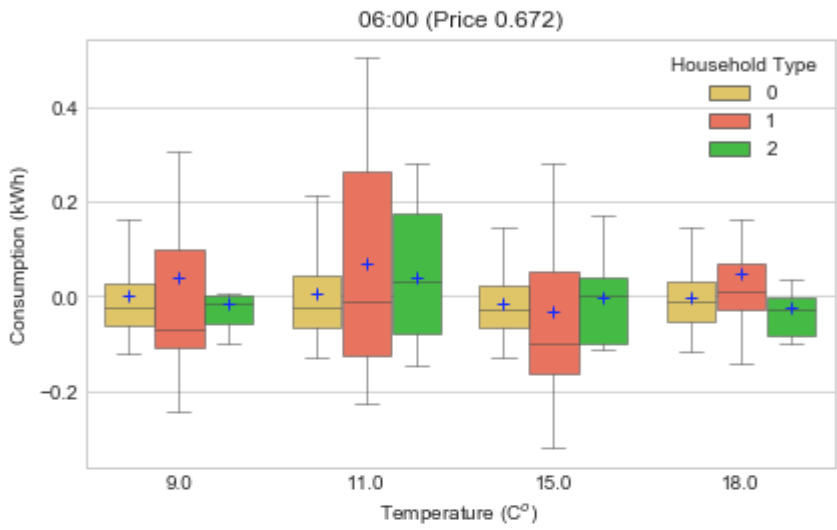

05:00 (Price 0.0399)
Household Type
0
1
2
0.5
0.4
0.3
0.2
0.1

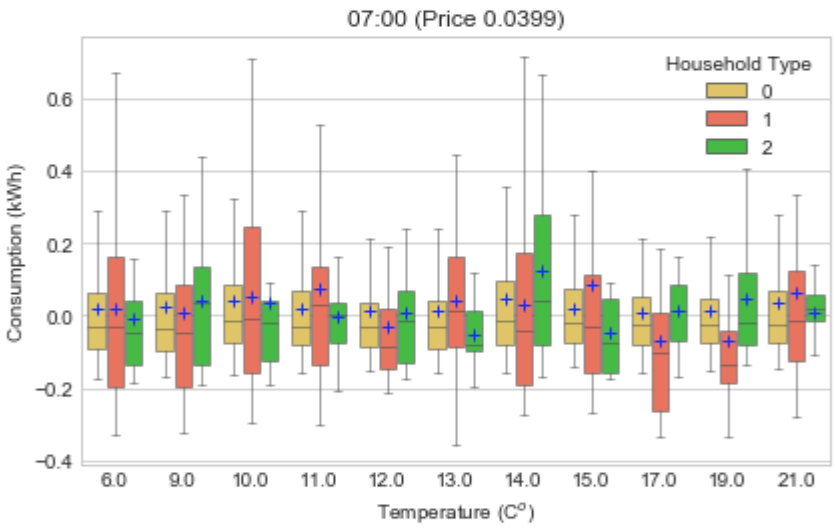

05:00 (Price 0.672)
Household Type
0
1
2
0.3
0.2
0.1
0.0

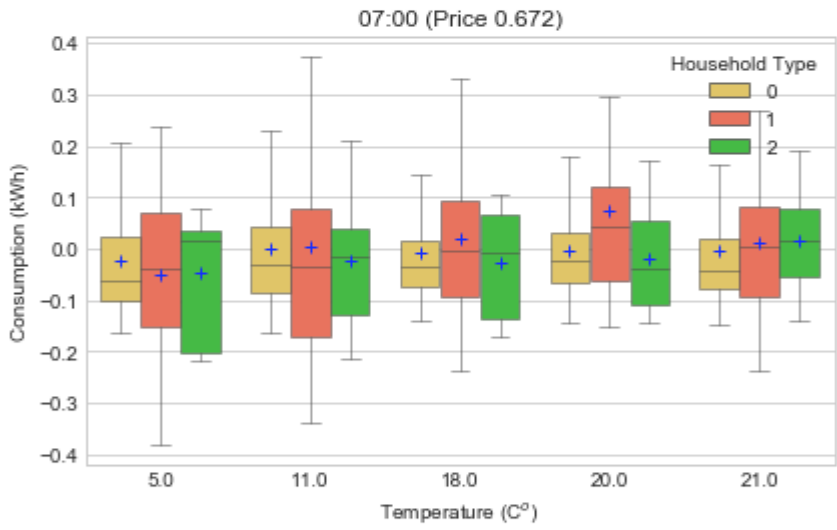

Temperature (C°)

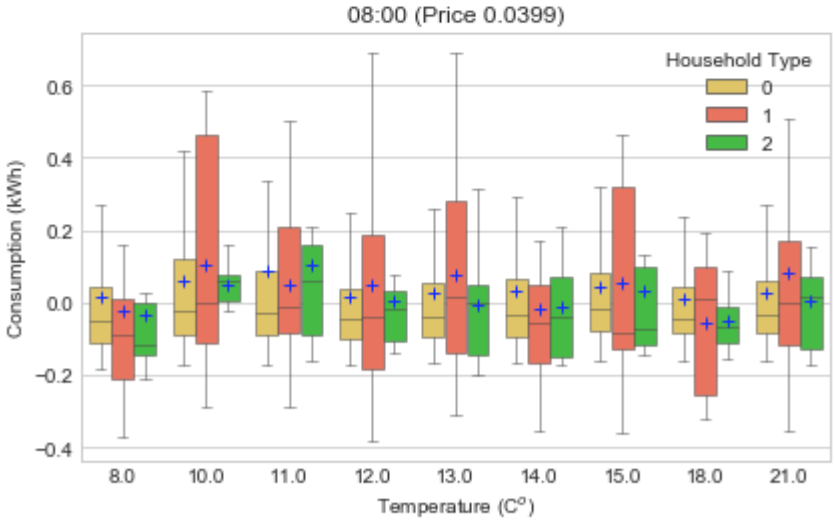

Temperature (C°)

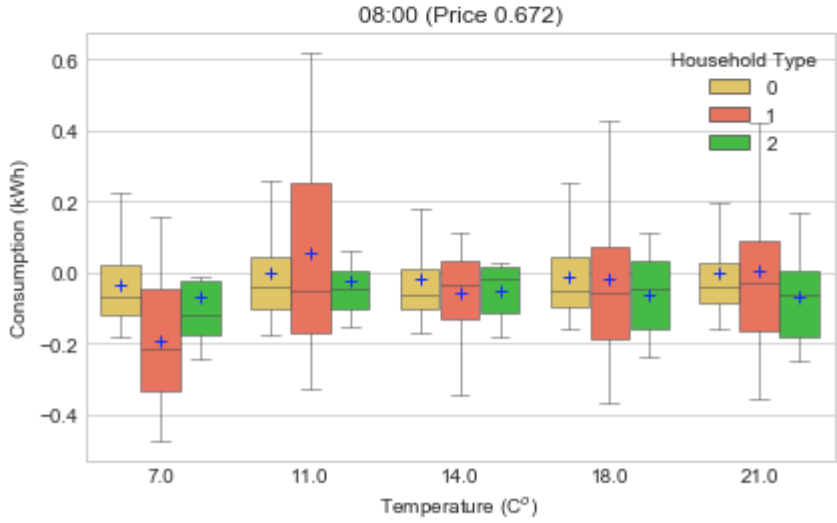

06:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

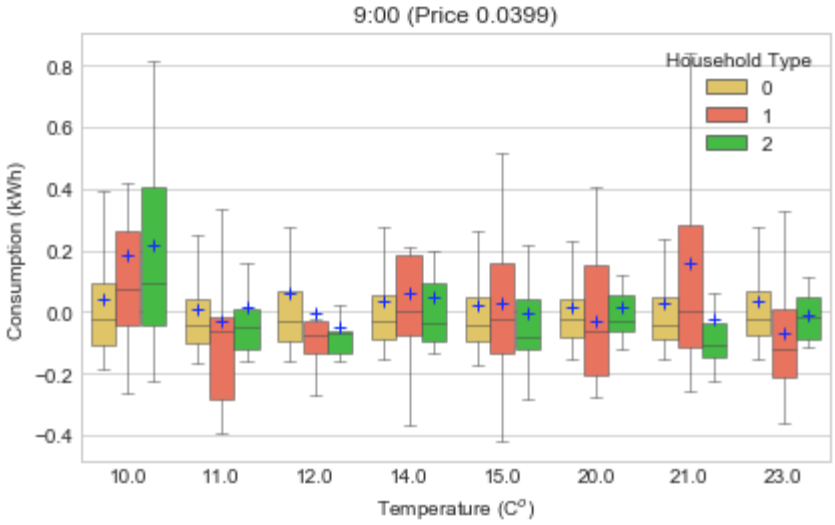

06:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

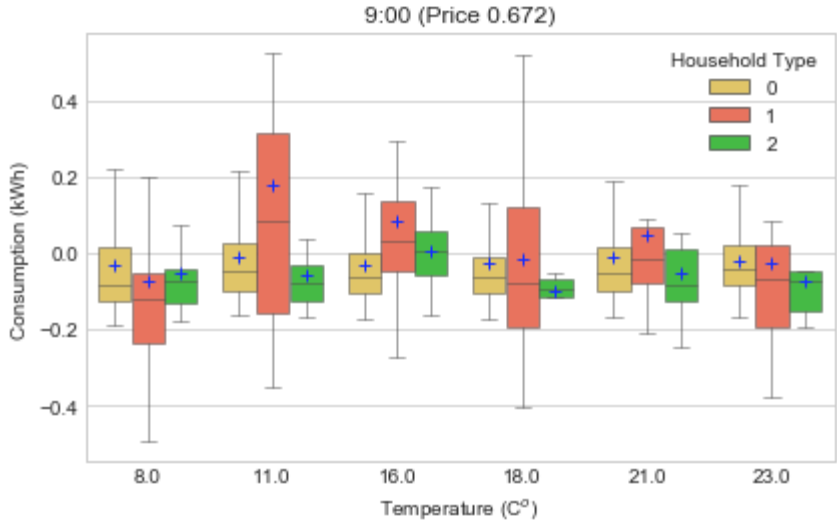

07:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

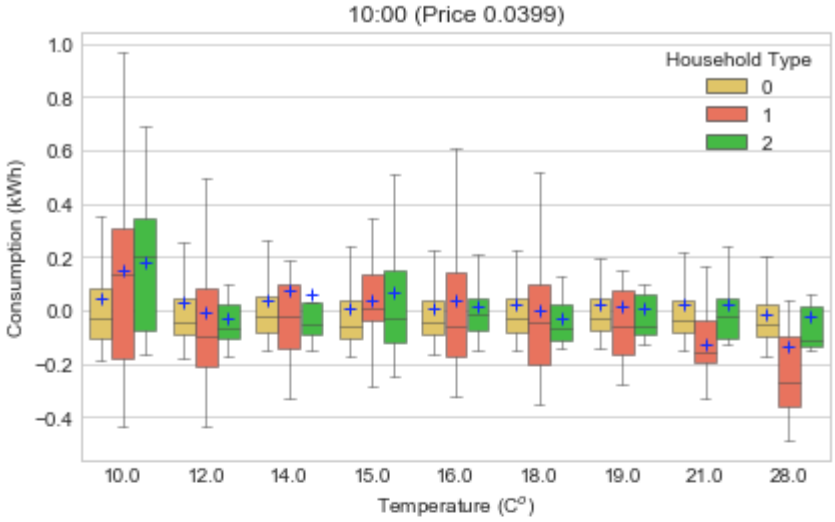

07:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

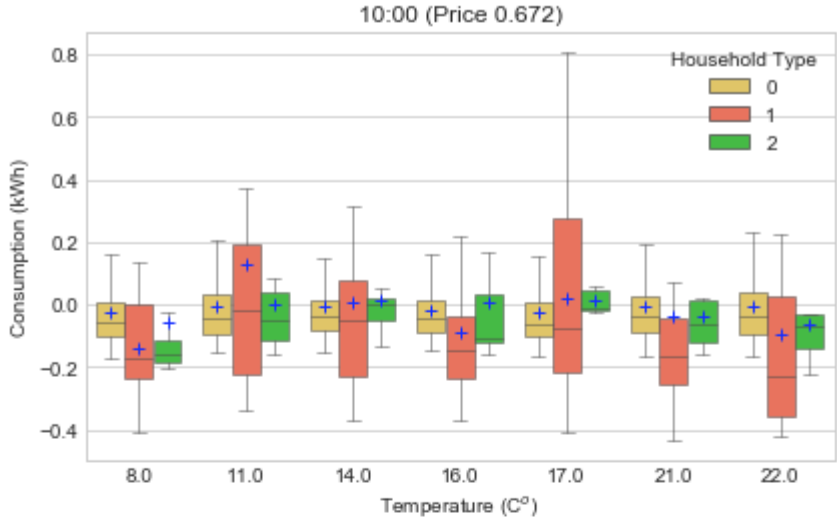

08:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)


08:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)


9:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

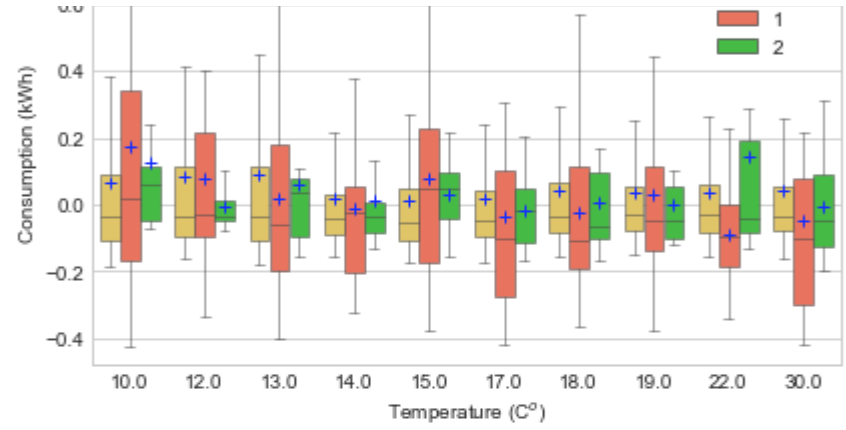

9:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

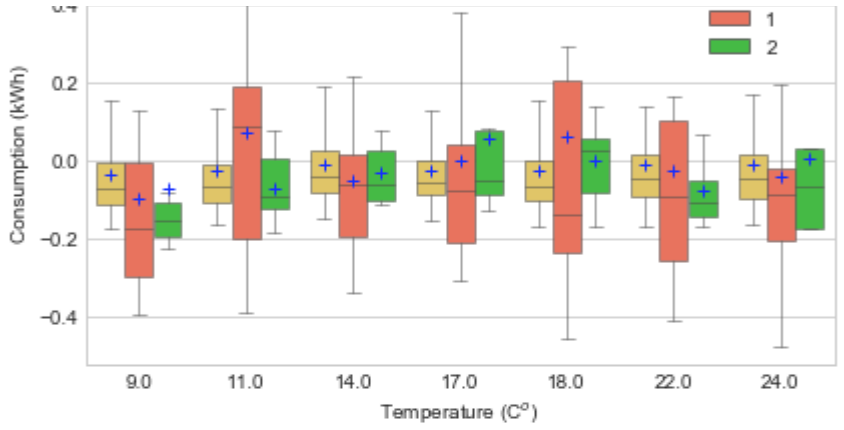

10:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

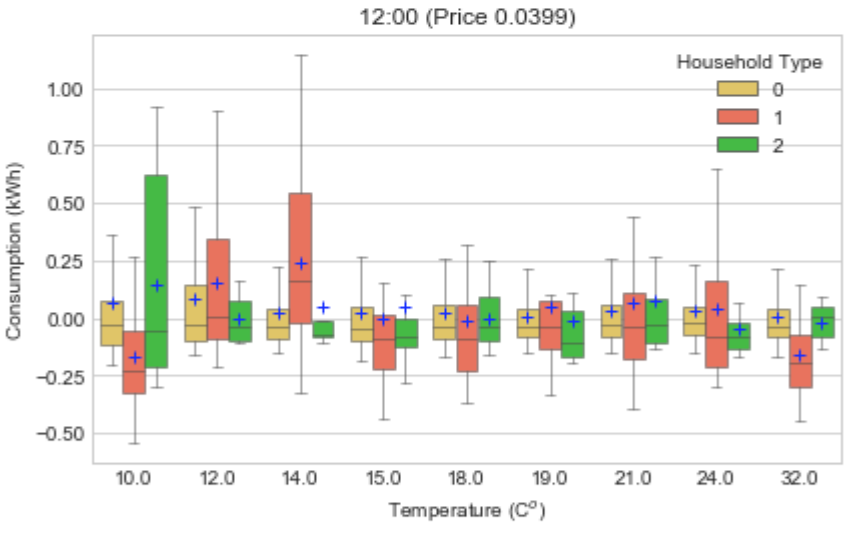

10:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

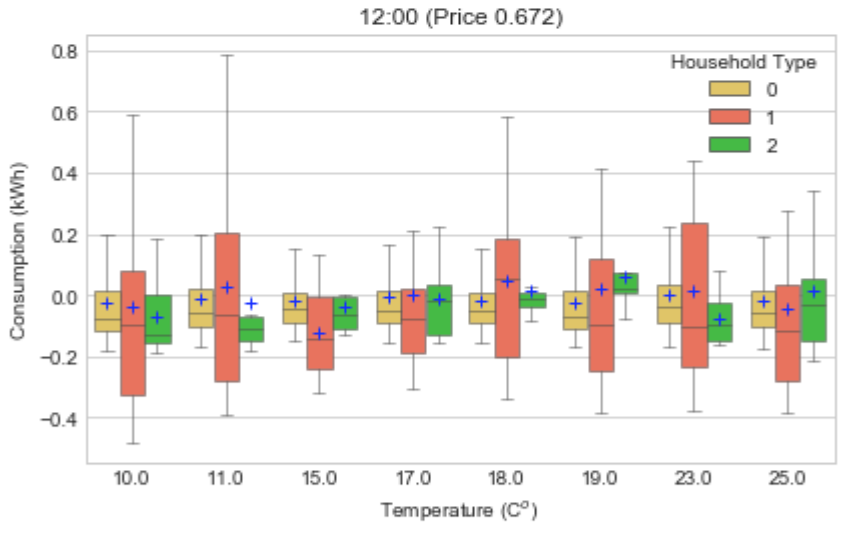
11:00 (Price 0.0399)
Household Type
0
Consumption (kWh)

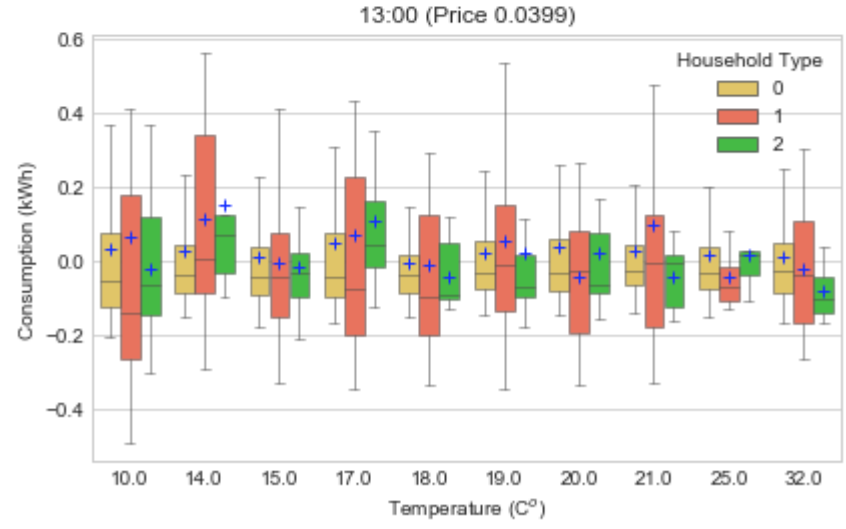
11:00 (Price 0.672)
Household Type
0

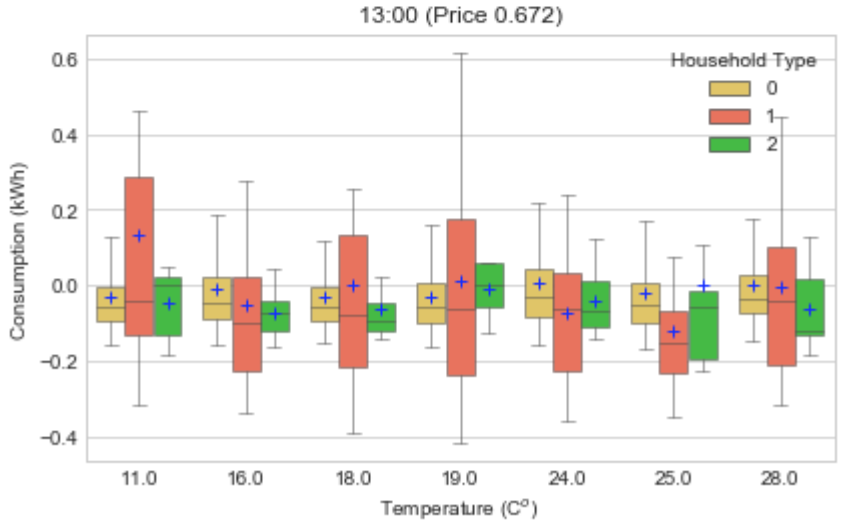

1
2
Consumption (kWh)
0.4
0.2
0.0
−0.2
−0.4
10.0
12.0
13.0
14.0
15.0
17.0
18.0
19.0
22.0
30.0
Temperature (C^o)

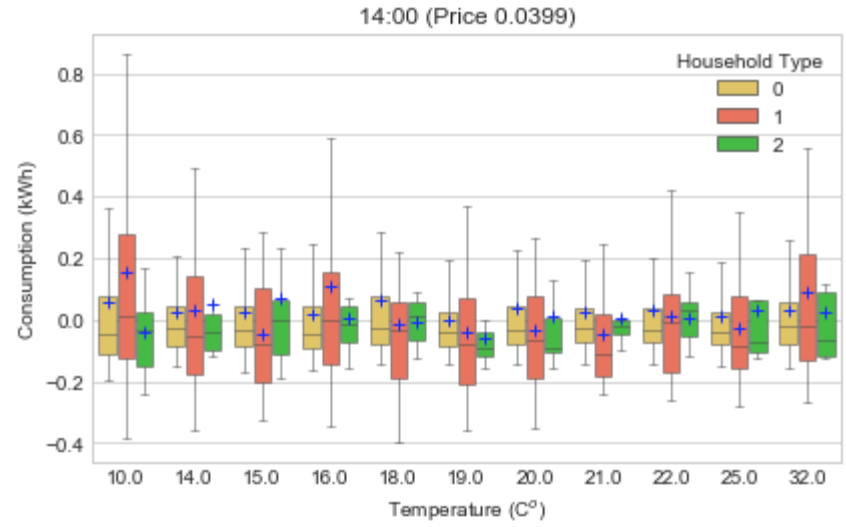

1
2
Consumption (kWh)
0.2
0.0
−0.2
−0.4
9.0
11.0
14.0
17.0
18.0
22.0
24.0
Temperature (C^o)

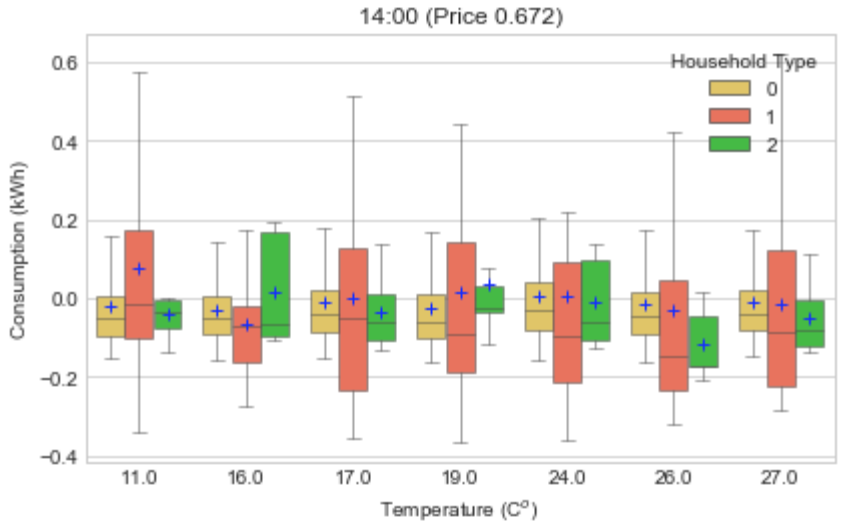

12:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
10.0
12.0
14.0
15.0
18.0
19.0
21.0
24.0
32.0
Temperature (C^o)

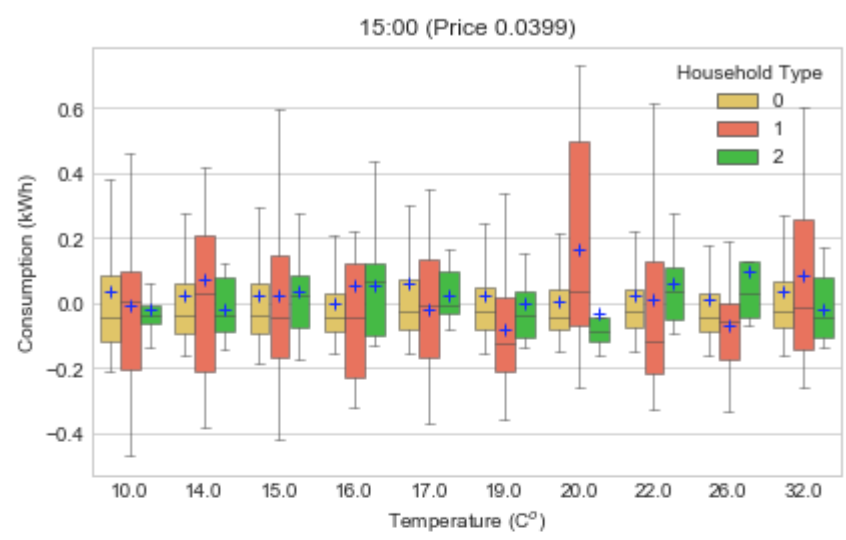

12:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
11.0
15.0
17.0
18.0
19.0
23.0
25.0
Temperature (C^o)

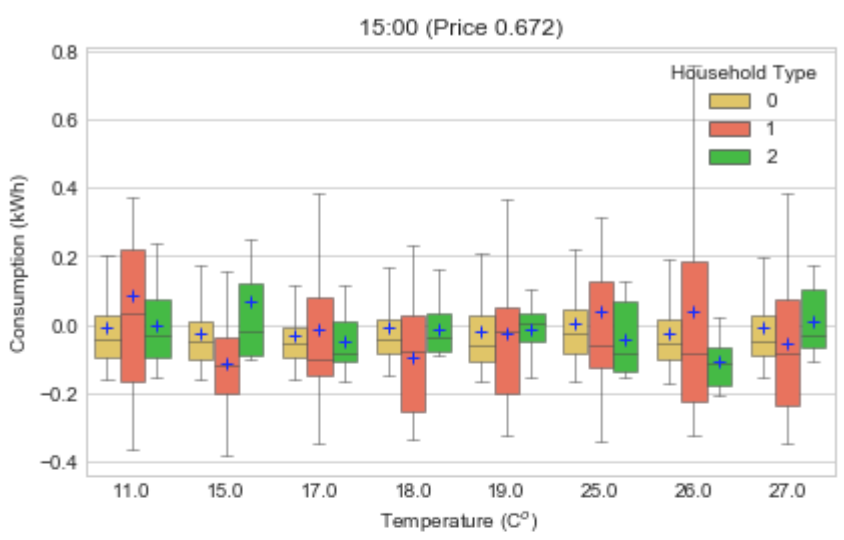

13:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
14.0
15.0
17.0
18.0
19.0
20.0
21.0
25.0
32.0
Temperature (C^o)

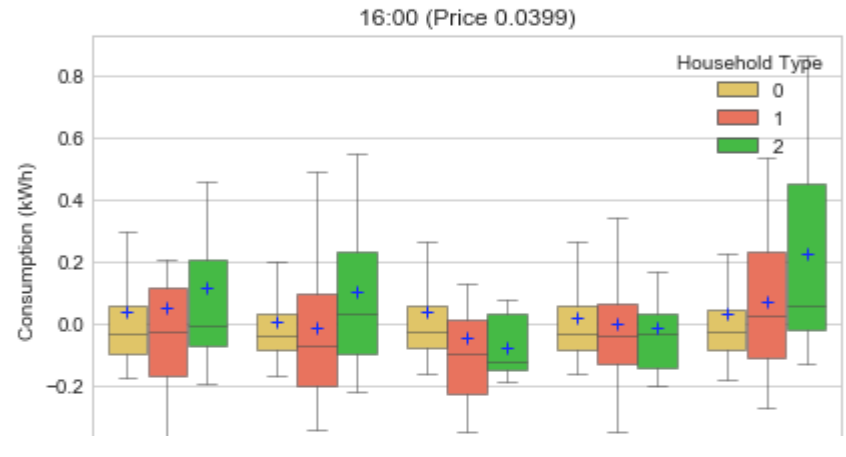

13:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
11.0
16.0
18.0
19.0
24.0
25.0
28.0
Temperature (C^o)

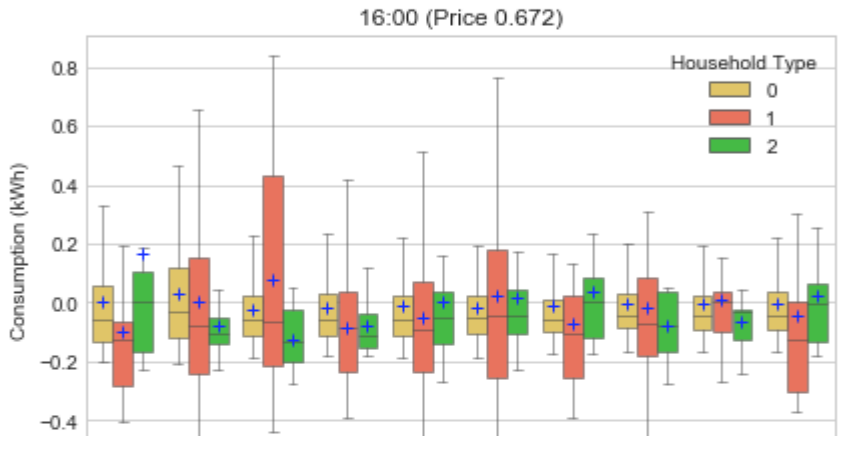

14:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
14.0
15.0
16.0
18.0
19.0
20.0
21.0
22.0
25.0
32.0
Temperature (C^o)


14:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
11.0
16.0
17.0
19.0
24.0
26.0
27.0
Temperature (C^o)

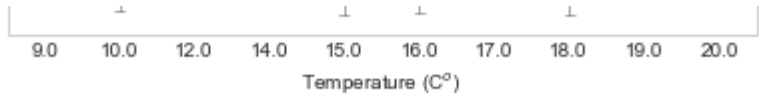

15:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
14.0
15.0
16.0
17.0
19.0
20.0
22.0
26.0
32.0
Temperature (C^o)

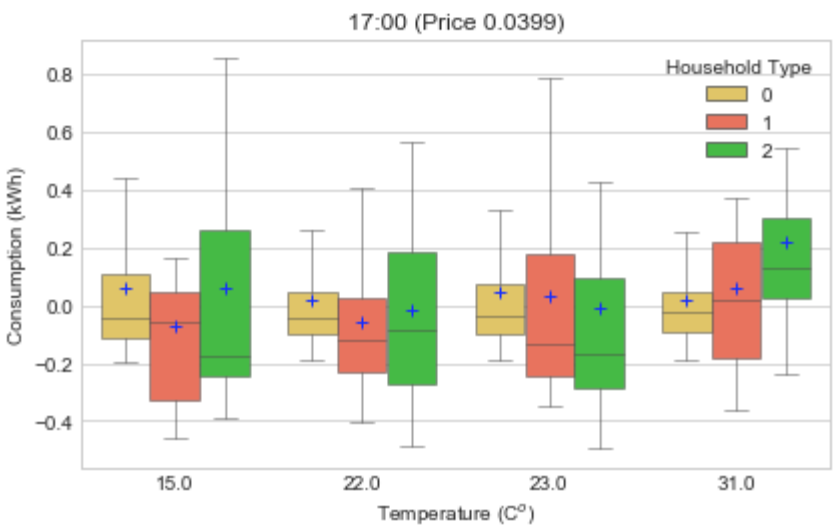

15:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
11.0
15.0
17.0
18.0
19.0
25.0
26.0
27.0
Temperature (C^o)

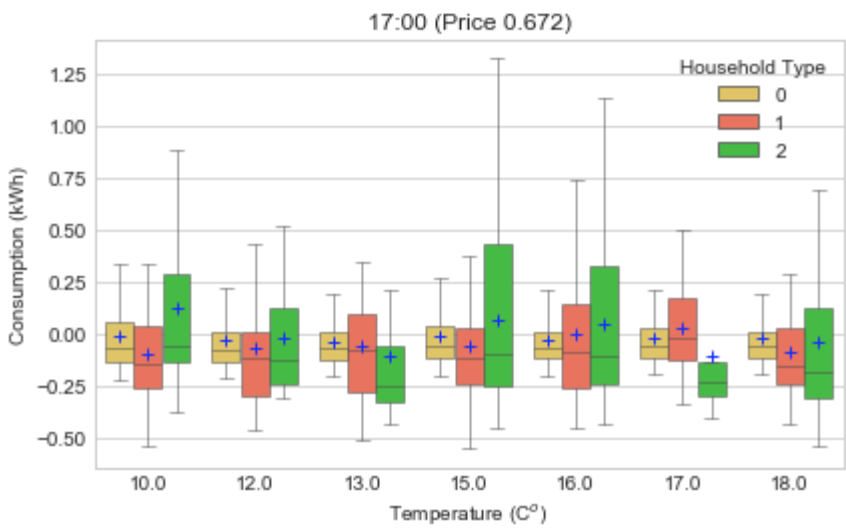

16:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2

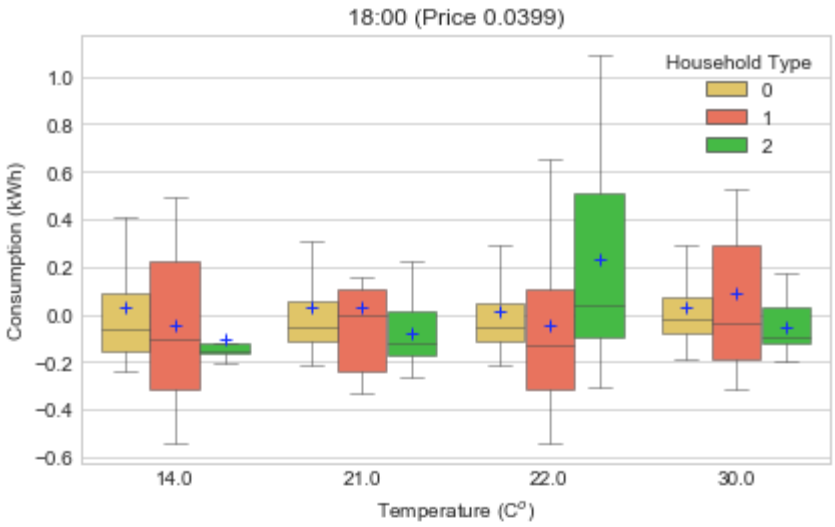

16:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

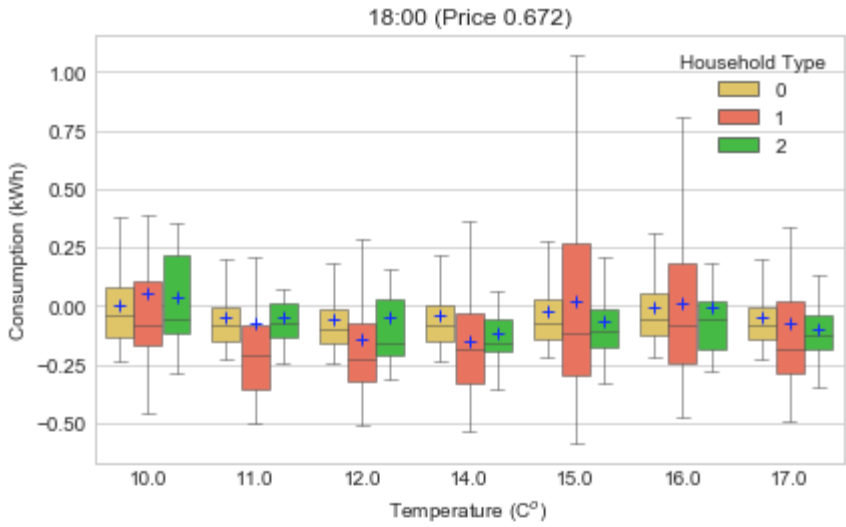

Temperature (C°)

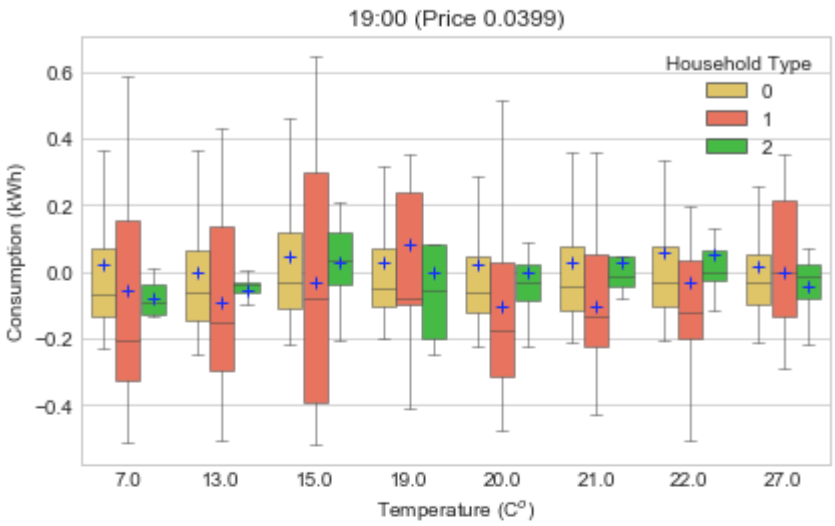

Temperature (C°)

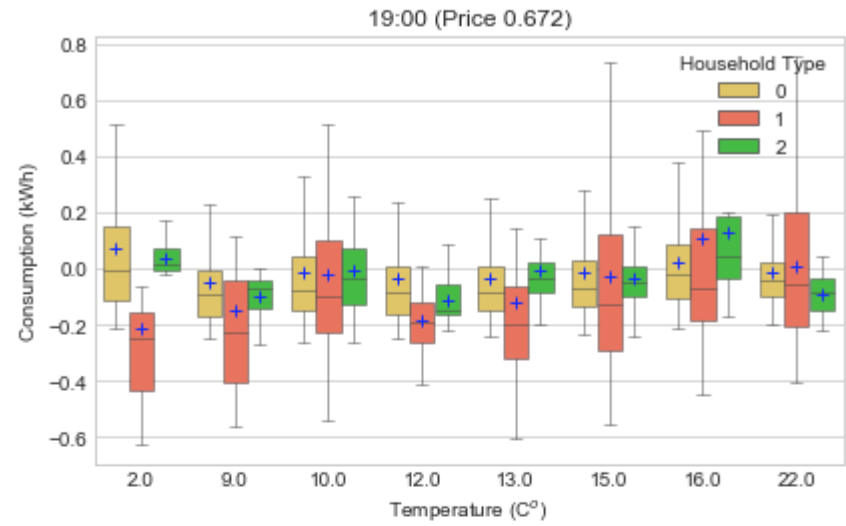

17:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

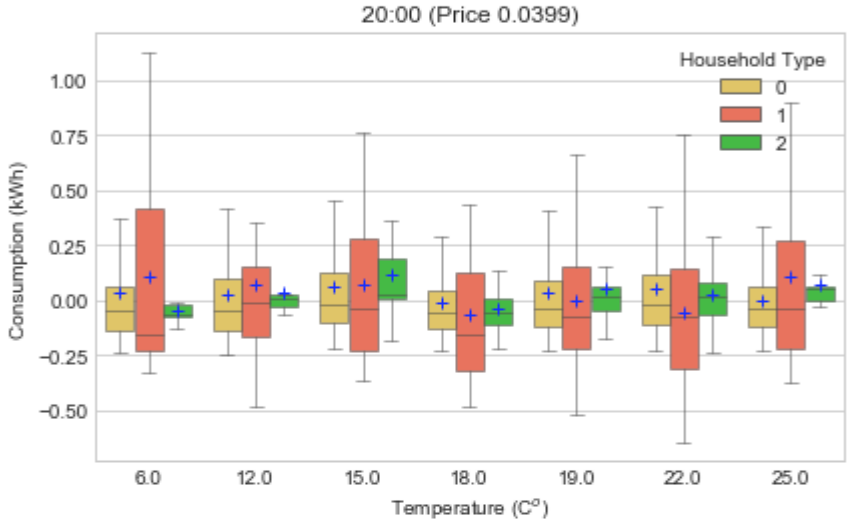

17:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

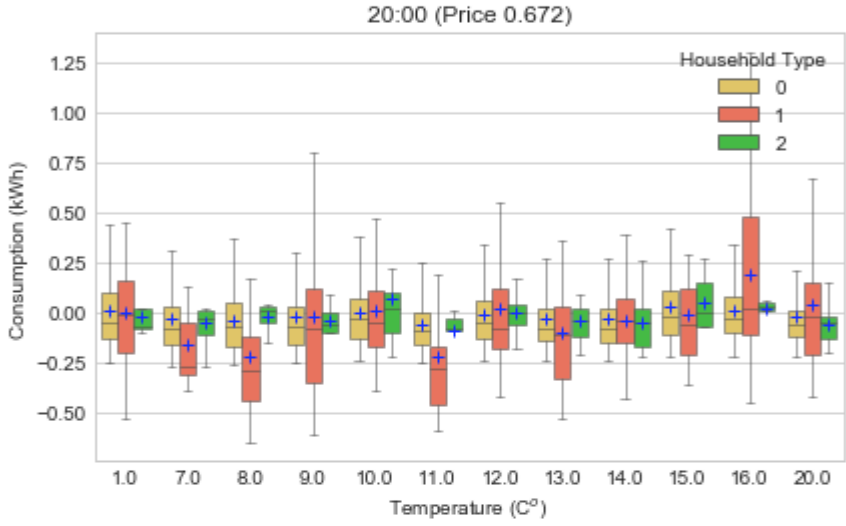

18:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

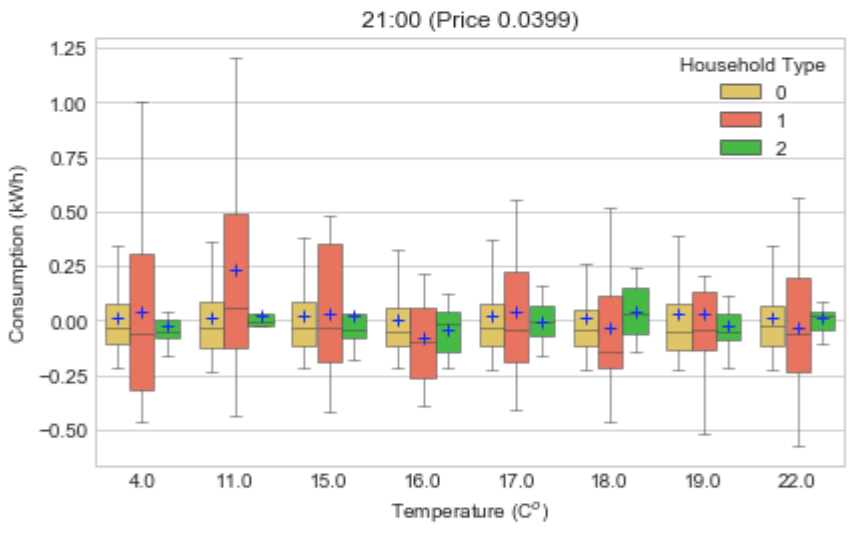

18:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

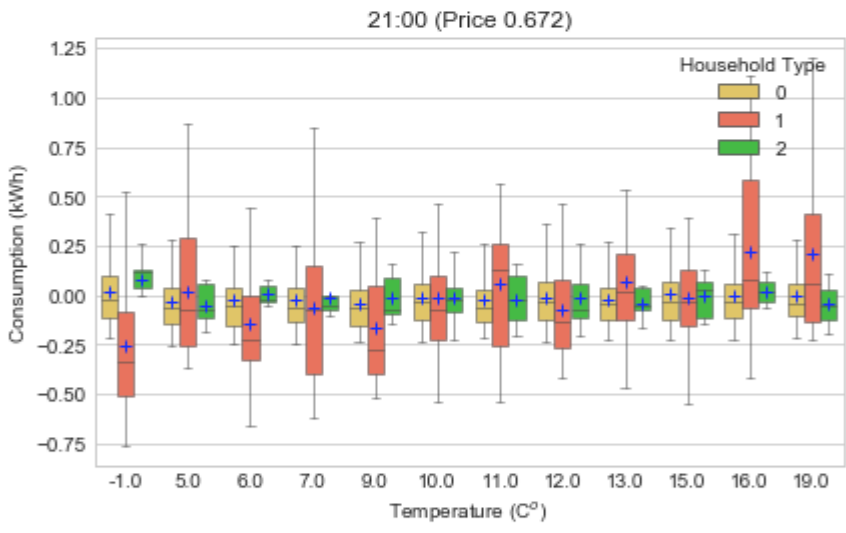

19:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

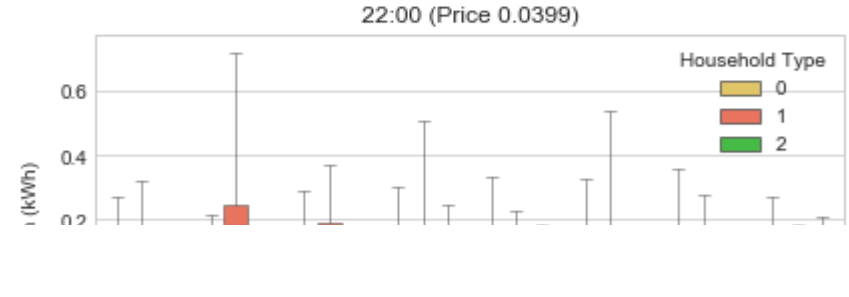

19:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)


20:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

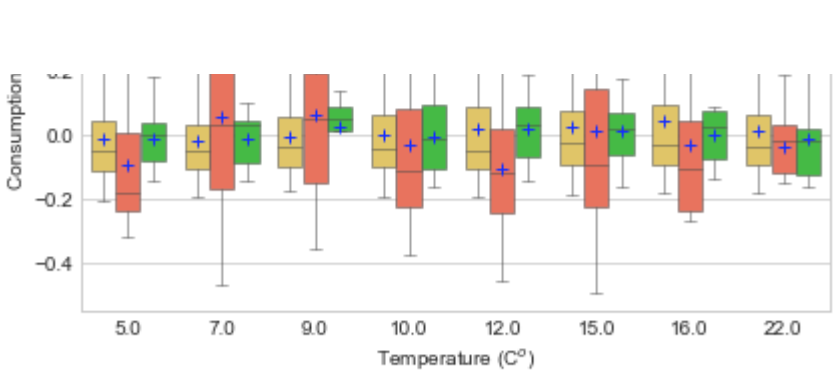

20:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

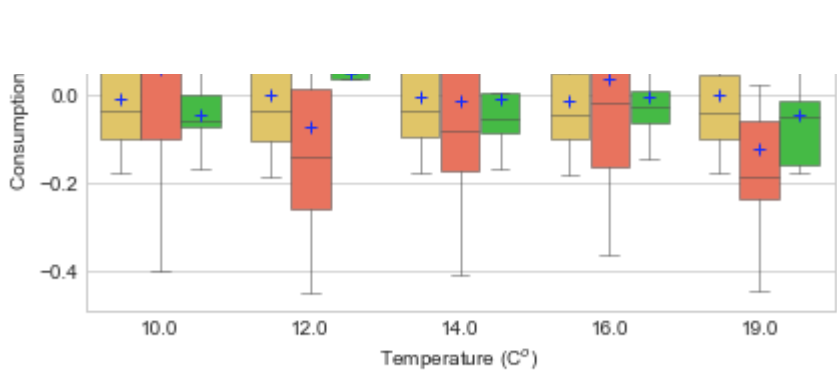

21:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

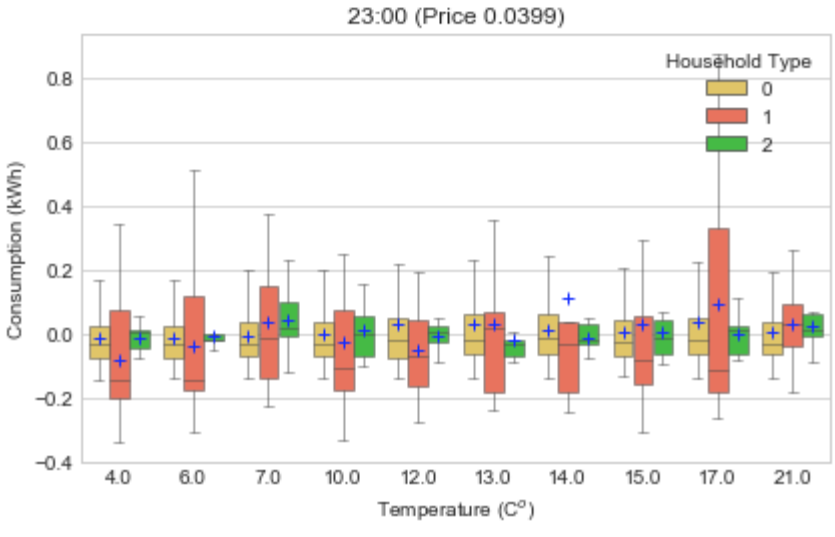

21:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
Temperature (C°)

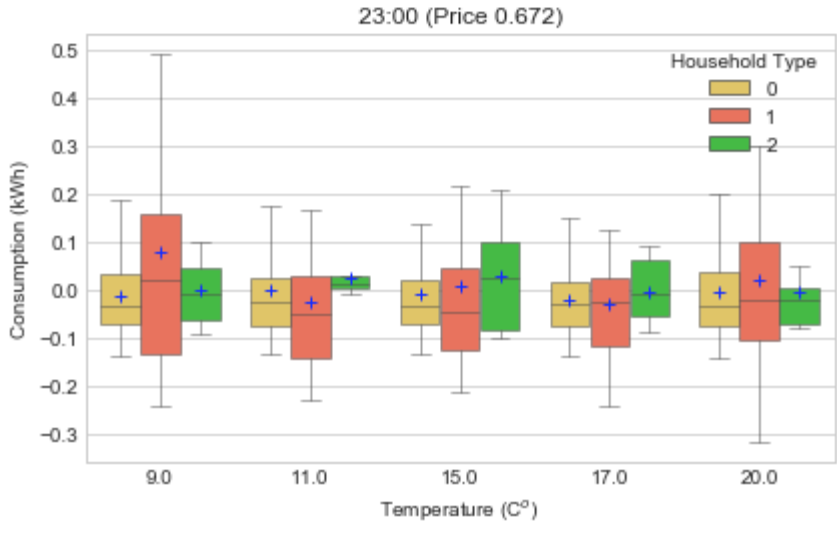

22:00 (Price 0.0399)
Household Type
0
1
2
Temperature (C°)


22:00 (Price 0.672)
Household Type
0
1
2
Temperature (C°)

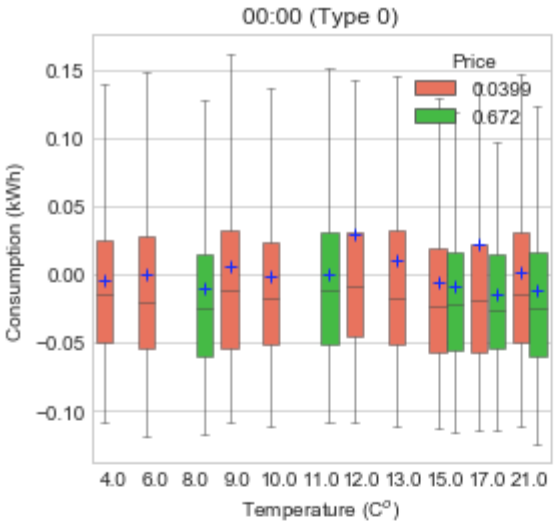

Consumption
0.0
−0.2
−0.4
5.0
7.0
9.0
10.0
12.0
15.0
16.0
22.0
Temperature (C°)

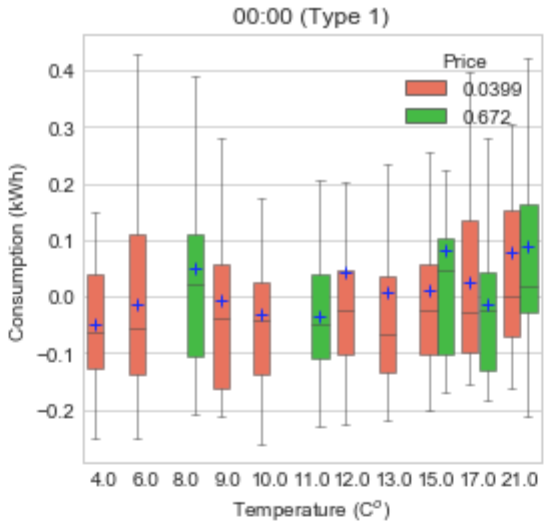

Consumption
0.0
−0.2
−0.4
10.0
12.0
14.0
16.0
19.0
Temperature (C°)

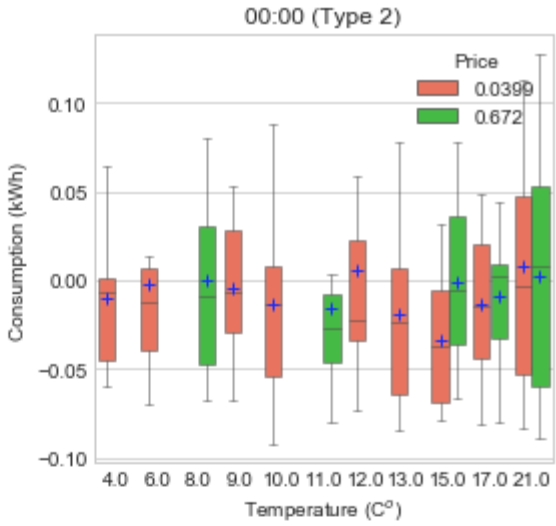

23:00 (Price 0.0399)
Household Type
0
1
2
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
4.0
6.0
7.0
10.0
12.0
13.0
14.0
15.0
17.0
21.0
Temperature (C°)

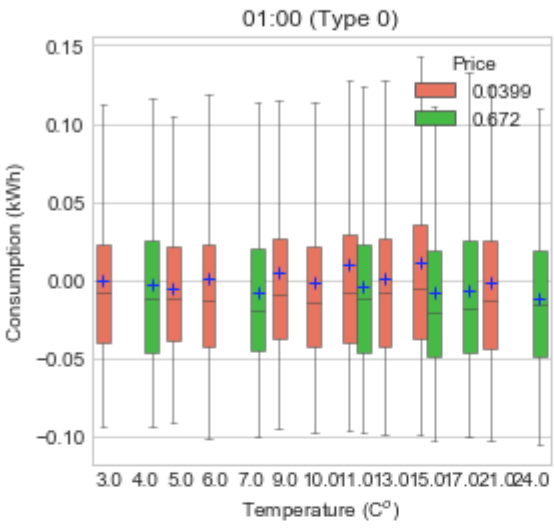

23:00 (Price 0.672)
Household Type
0
1
2
Consumption (kWh)
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
9.0
11.0
15.0
17.0
20.0
Temperature (C°)

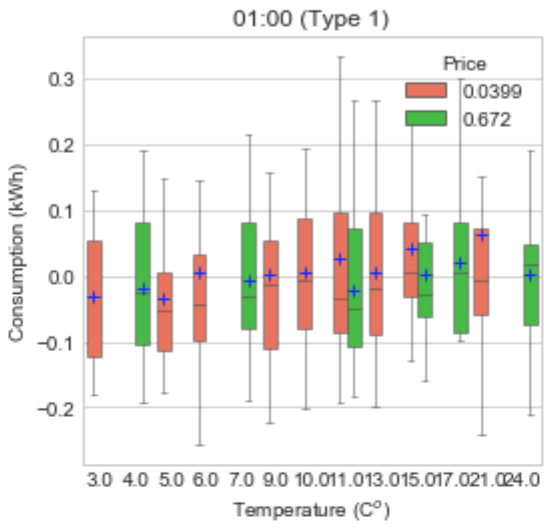
In [80]:
```
# price comparision
# hourly price responsiveness over temperatures and prices
fig_all = plt.figure(figsize = (12,90))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    for i in range(24):
        ax_Ntou.append(fig_all.add_subplot(24, 3, 3* i + (g + 1)))
        df_hour = df_all_res_long[(df_all_res_long['Hour of day'] == i) & (df_all_res_long['Household type']
== g)]
#         min_consumption = df_hour['Consumption'].min()
#         max_consumption = df_hour['Consumption'].max()
        if df_hour.shape[0] >= 1:
            sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_hour, ax=ax_Ntou[-1], palette=['toma
to','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid b
lue", "markeredgecolor":"xkcd:vivid blue"})
            if i < 9:
                ax_Ntou[-1].set_title('0' + str(i) + ':00' + ' (Type ' + str(g) + ')' )
            else:
                ax_Ntou[-1].set_title(str(i) + ':00' + ' (Type ' + str(g) + ')' )
            ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
            ax_Ntou[-1].set_ylabel('Consumption (kWh)')
            l = ax_Ntou[-1].legend()
            l.set_title('Price')
            for i,artist in enumerate(ax_Ntou[-1].artists):
                artist.set_linewidth(0.5)
                # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
                # Loop over them here, and use the same colour as above
                for j in range(i*6,i*6+6):
                    line = ax_Ntou[-1].lines[j]
                    line.set_linewidth(0.5)
            ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```

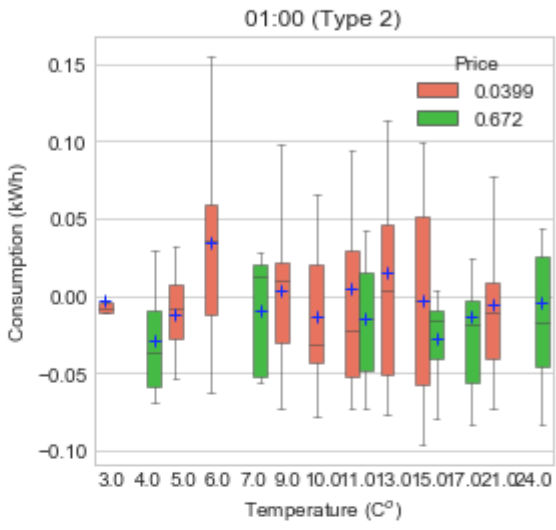

00:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

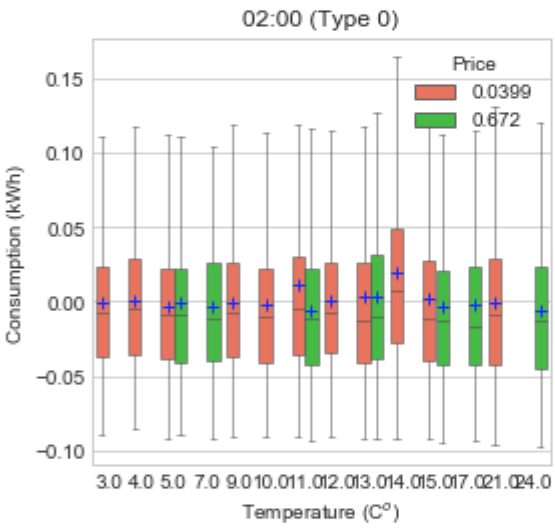

00:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

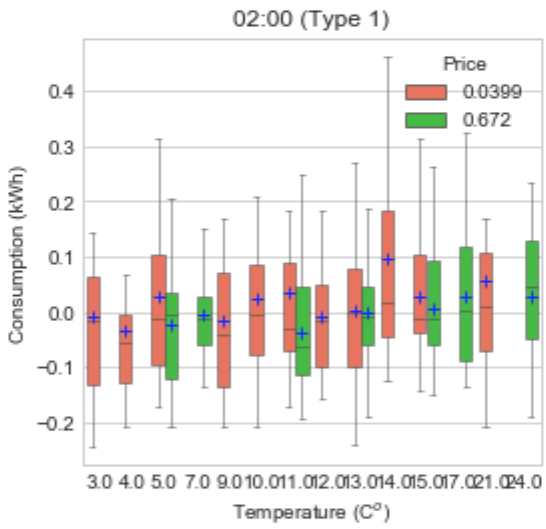

00:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

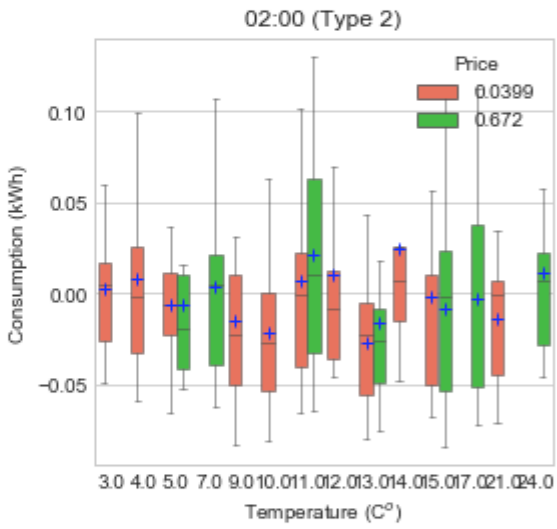

01:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

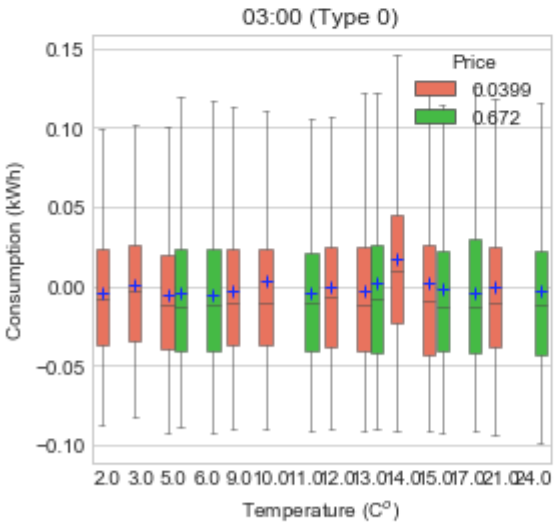

01:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

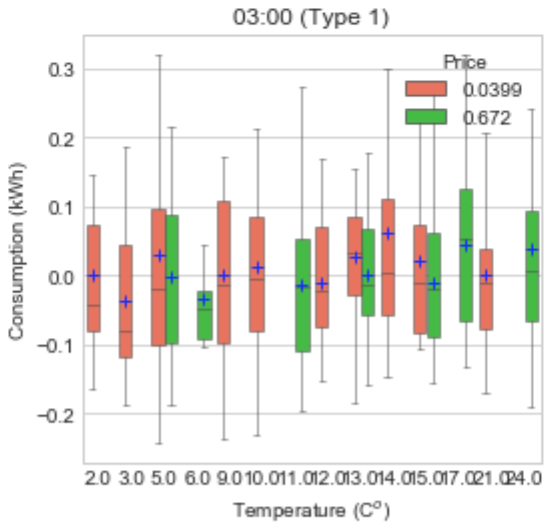

01:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

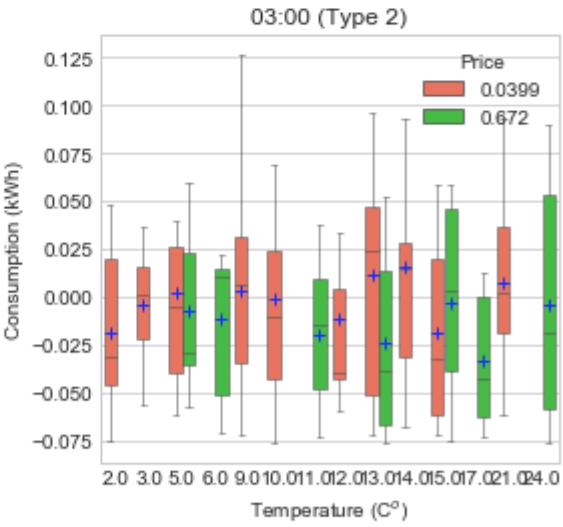

02:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

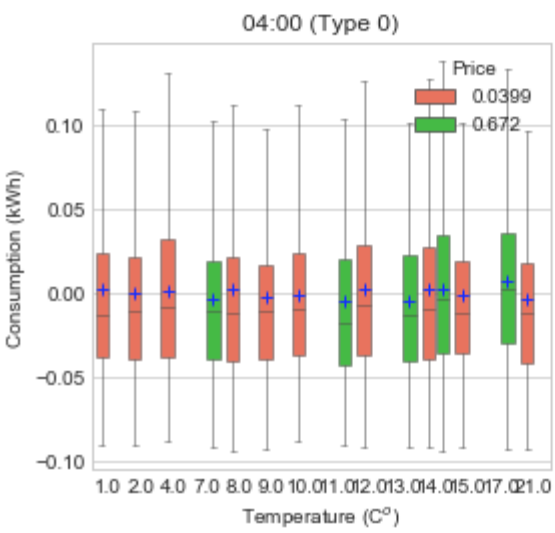

02:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

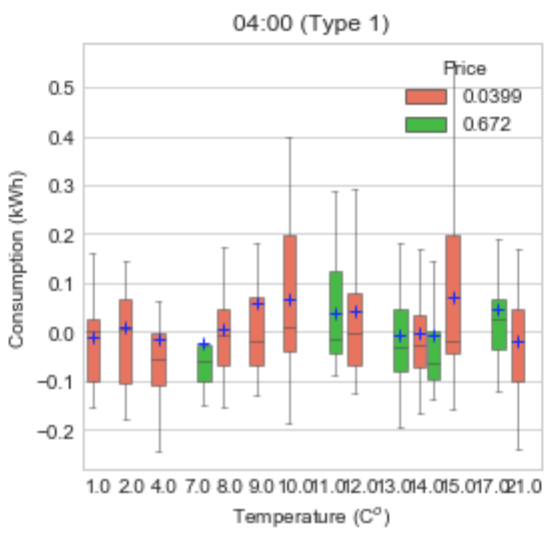

02:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

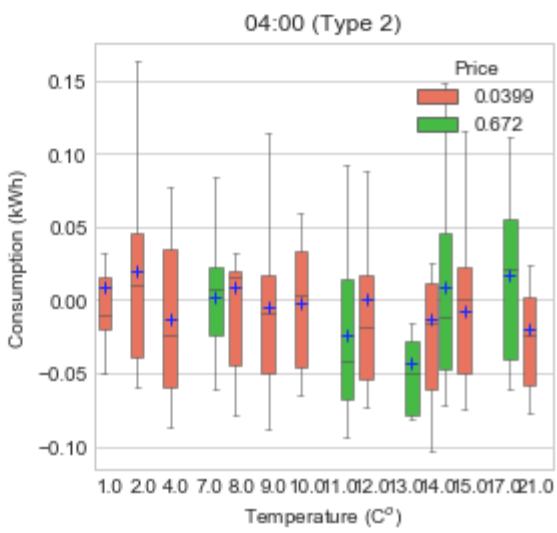

03:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

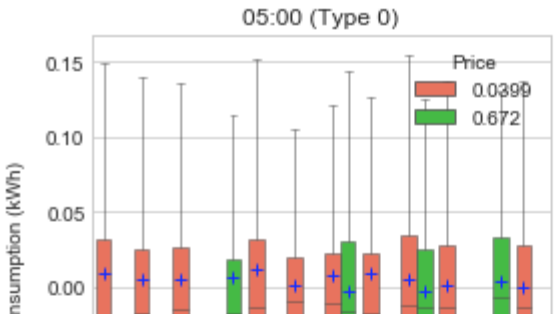

03:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

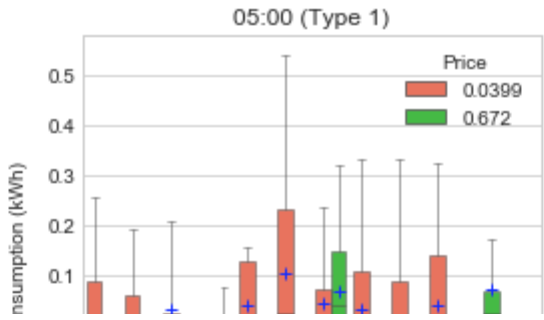

03:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

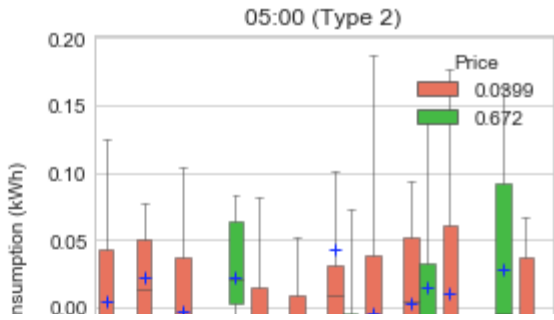

04:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

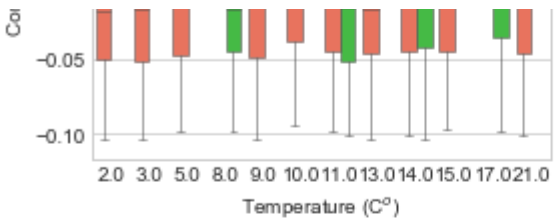

04:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

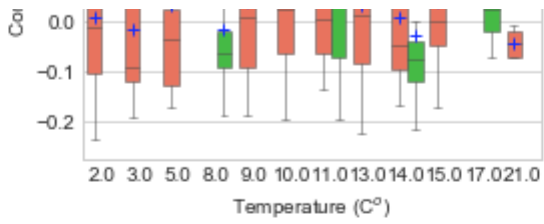

04:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

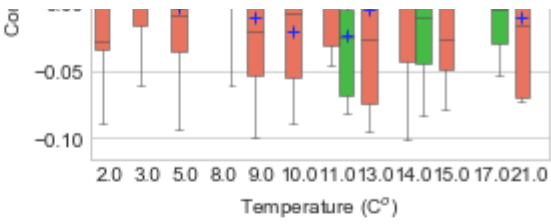

05:00 (Type 0)
Price
0.0399
0.672

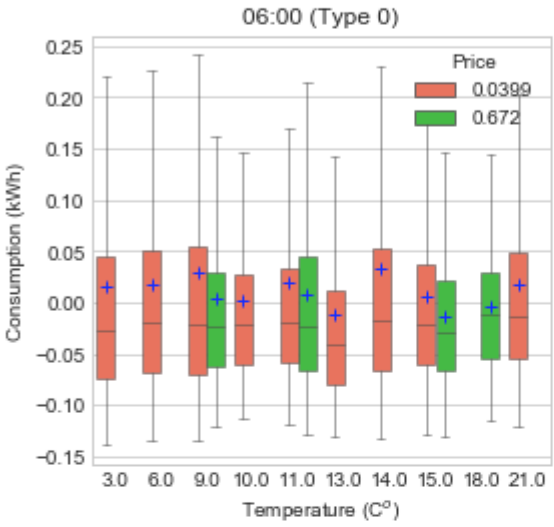

05:00 (Type 1)
Price
0.0399
0.672

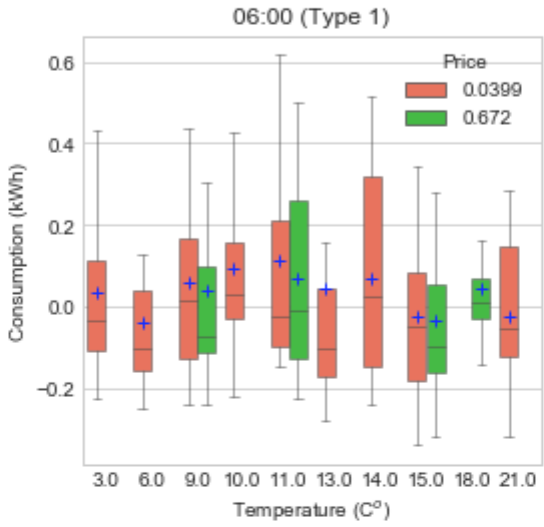

05:00 (Type 2)
Price
0.0399
0.672

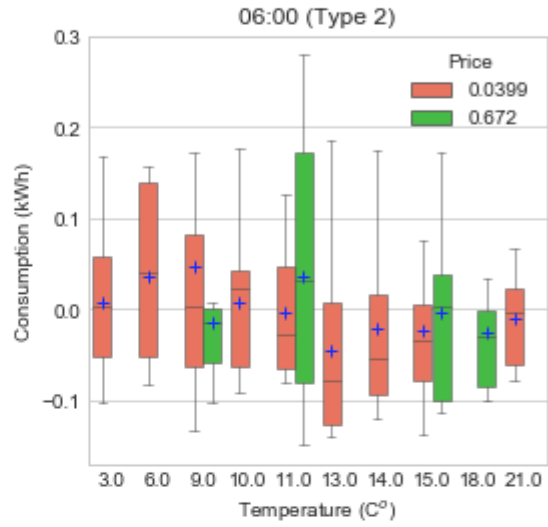

−0.05
−0.10
2.0 3.0 5.0 8.0 9.0 10.0 11.0 13.0 14.0 15.0 17.0 21.0
Temperature (C°)

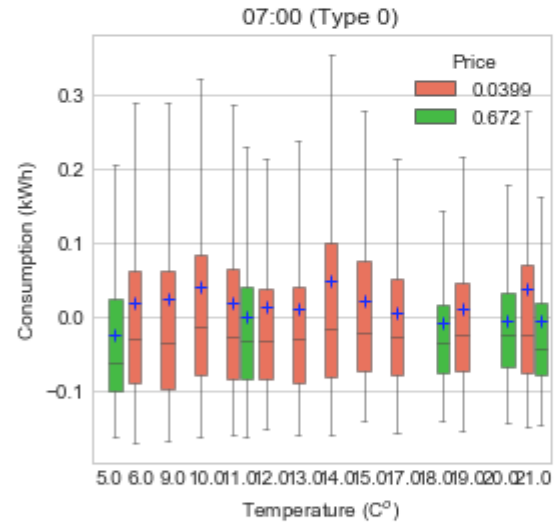

0.0
−0.1
−0.2
2.0 3.0 5.0 8.0 9.0 10.0 11.0 13.0 14.0 15.0 17.0 21.0
Temperature (C°)

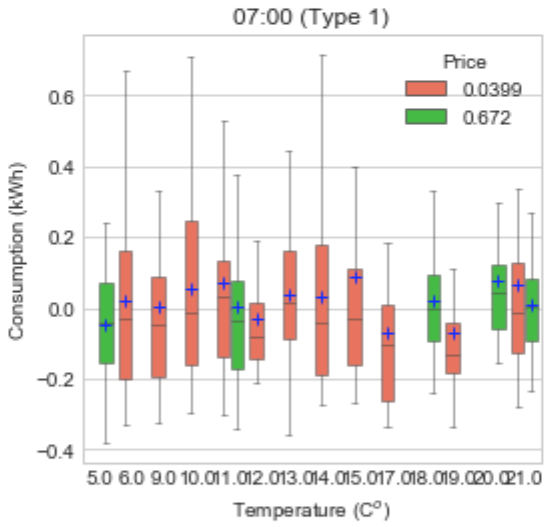

−0.05
−0.10
2.0 3.0 5.0 8.0 9.0 10.0 11.0 13.0 14.0 15.0 17.0 21.0
Temperature (C°)

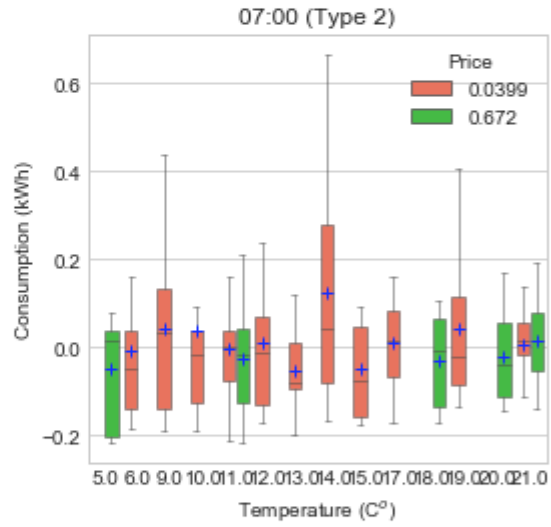

06:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
3.0 6.0 9.0 10.0 11.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

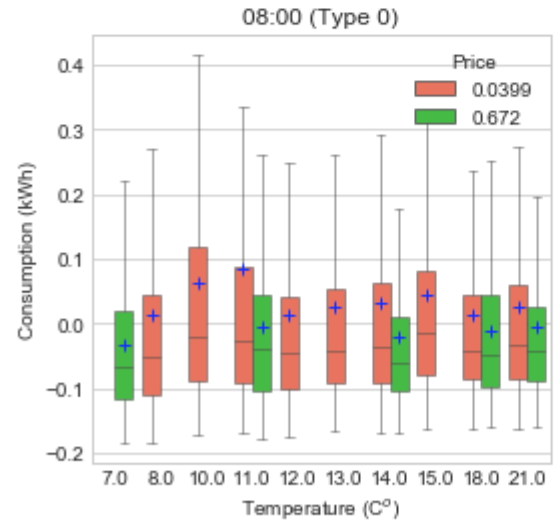

06:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
3.0 6.0 9.0 10.0 11.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

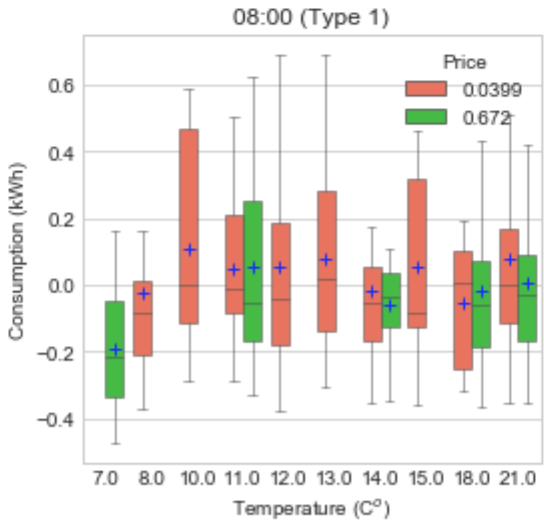

06:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
3.0 6.0 9.0 10.0 11.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

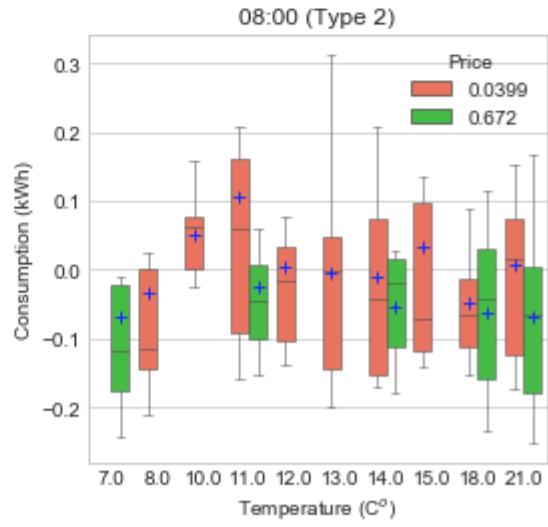

07:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
5.0 6.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 17.0 18.0 19.0 20.0 21.0
Temperature (C°)

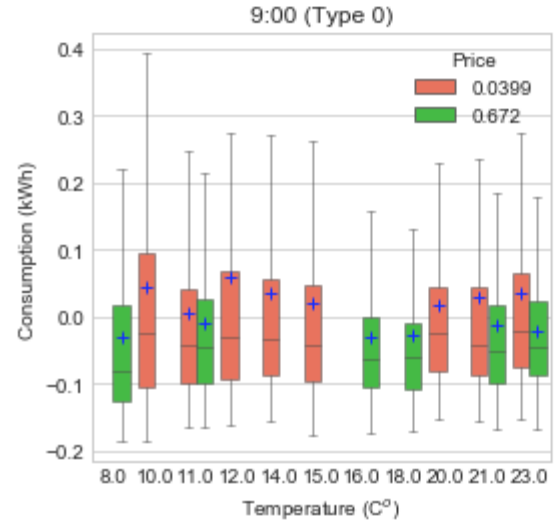

07:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
5.0 6.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 17.0 18.0 19.0 20.0 21.0
Temperature (C°)

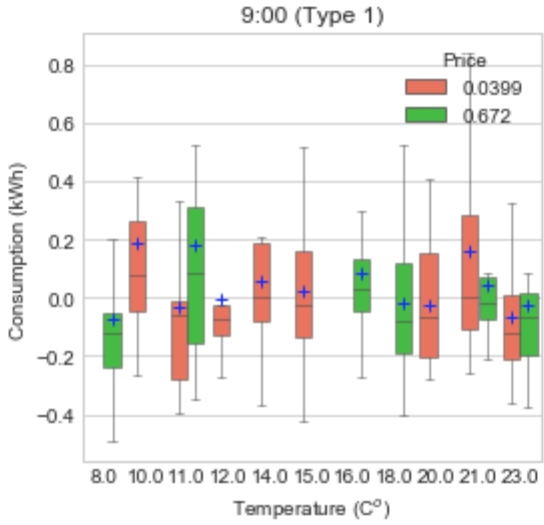

07:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
5.0 6.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 17.0 18.0 19.0 20.0 21.0
Temperature (C°)

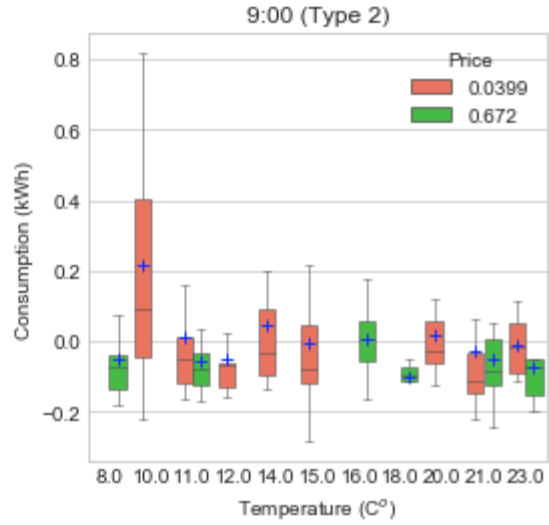

08:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0 8.0 10.0 11.0 12.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

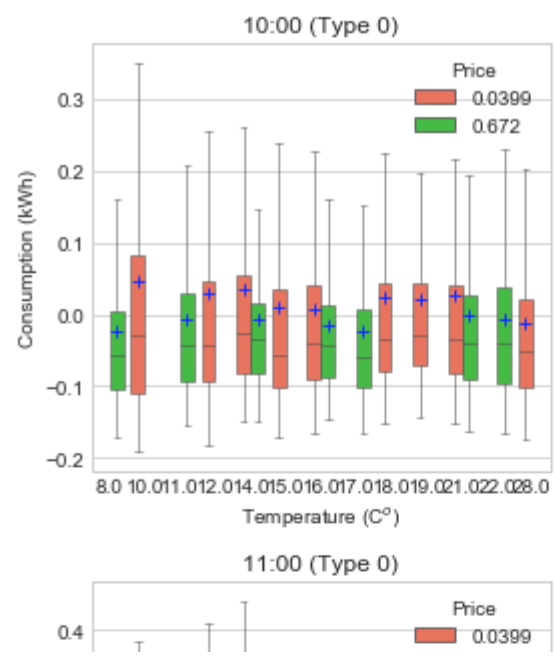

08:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
−0.4
7.0 8.0 10.0 11.0 12.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

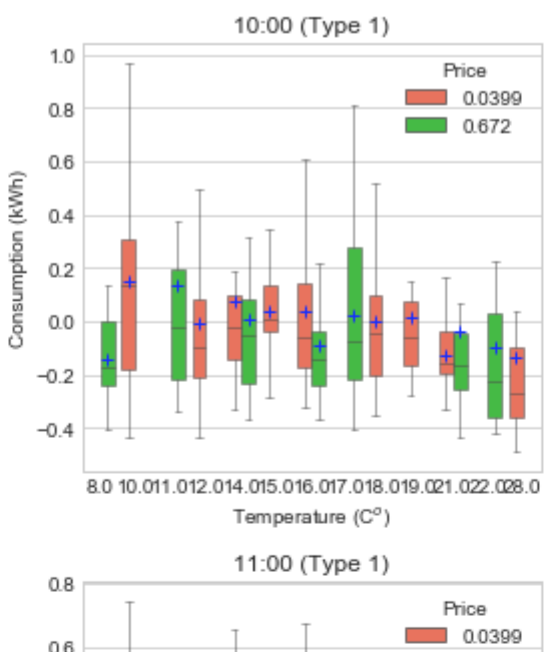

08:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
7.0 8.0 10.0 11.0 12.0 13.0 14.0 15.0 18.0 21.0
Temperature (C°)

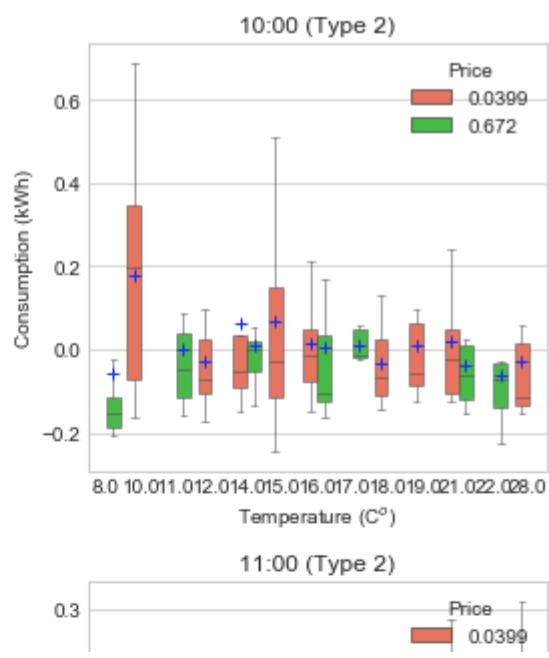

9:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
8.0 10.0 11.0 12.0 14.0 15.0 16.0 18.0 20.0 21.0 23.0
Temperature (C°)


9:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
8.0 10.0 11.0 12.0 14.0 15.0 16.0 18.0 20.0 21.0 23.0
Temperature (C°)


9:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.8
0.6
0.4
0.2
0.0
−0.2
8.0 10.0 11.0 12.0 14.0 15.0 16.0 18.0 20.0 21.0 23.0
Temperature (C°)


10:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
0.3
0.2
0.1
0.0
−0.1
−0.2
8.0 10.0 11.0 12.0 14.0 15.0 16.0 17.0 18.0 19.0 21.0 22.0 28.0
Temperature (C°)

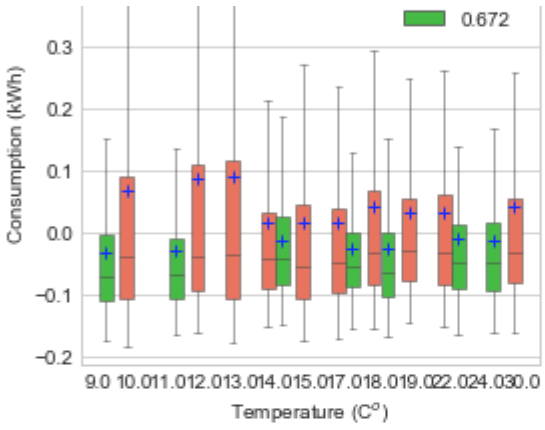

10:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
8.0 10.0 11.0 12.0 14.0 15.0 16.0 17.0 18.0 19.0 21.0 22.0 28.0
Temperature (C°)

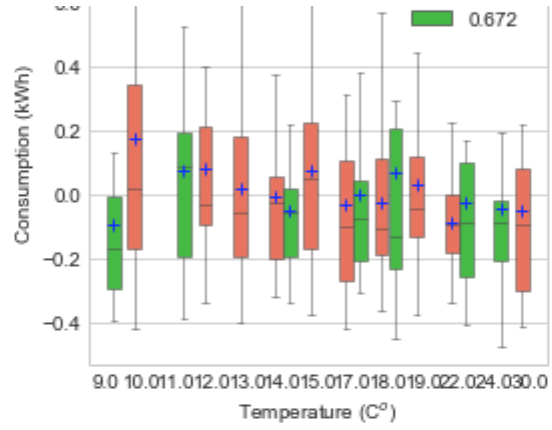

10:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
0.6
0.4
0.2
0.0
−0.2
8.0 10.0 11.0 12.0 14.0 15.0 16.0 17.0 18.0 19.0 21.0 22.0 28.0
Temperature (C°)

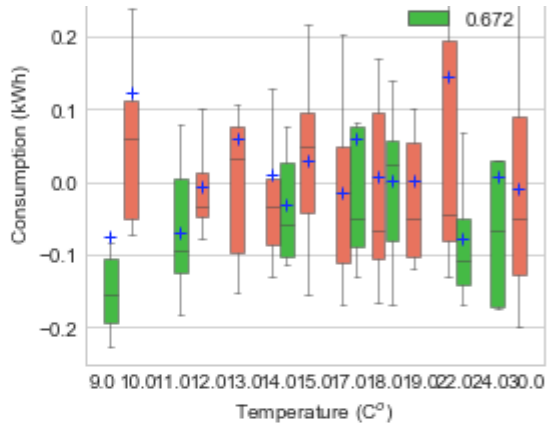
11:00 (Type 0)
Price
0.0399
0.4

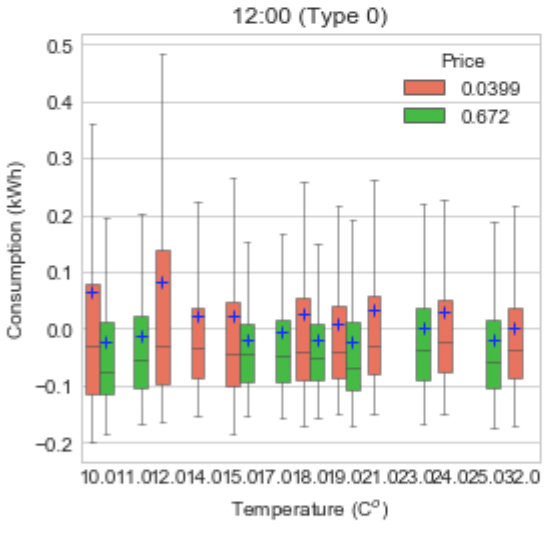
11:00 (Type 1)
Price
0.0399
0.8
0.6

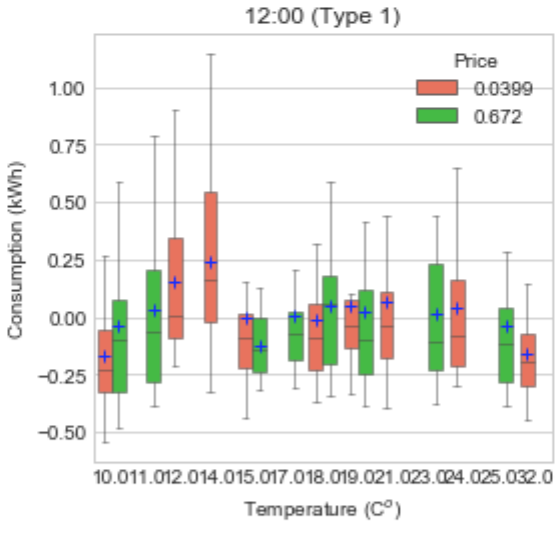
11:00 (Type 2)
Price
0.0399
0.3

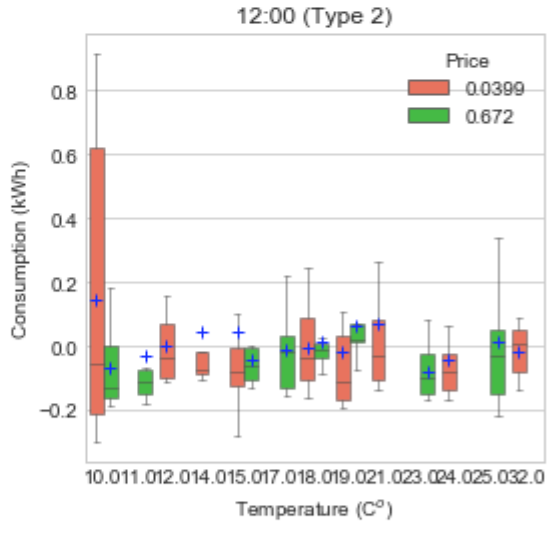

0.672
Consumption (kWh)
Temperature (C°)

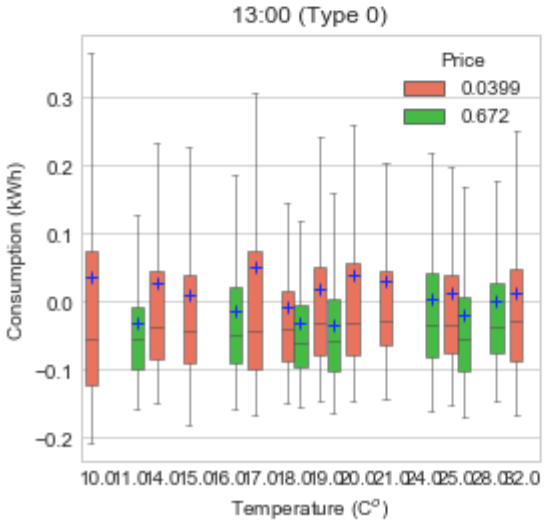

0.672
Consumption (kWh)
Temperature (C°)

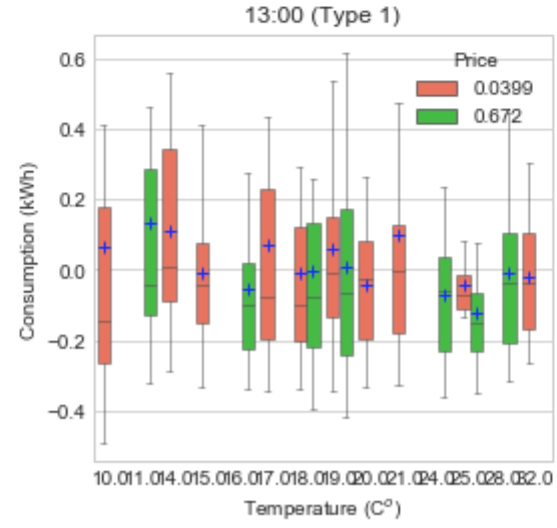

0.672
Consumption (kWh)
Temperature (C°)

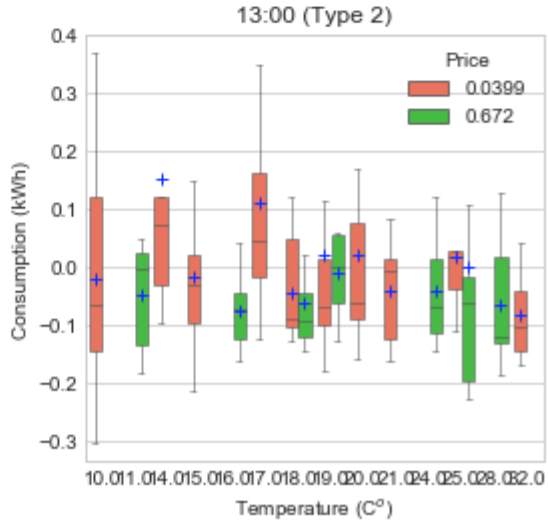

12:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

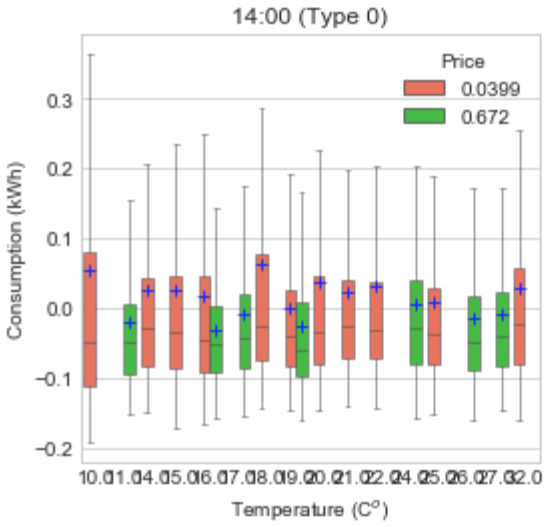

12:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

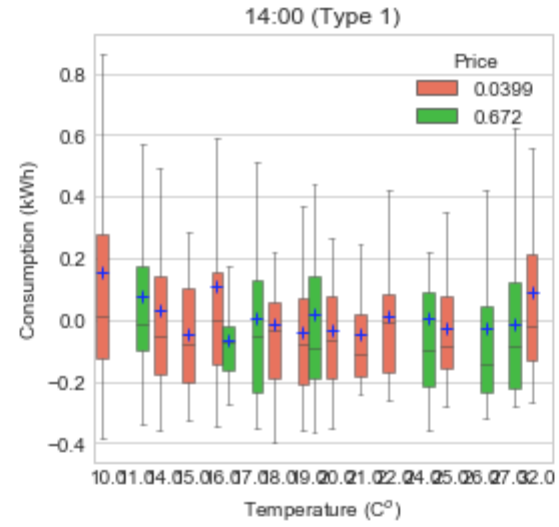

12:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

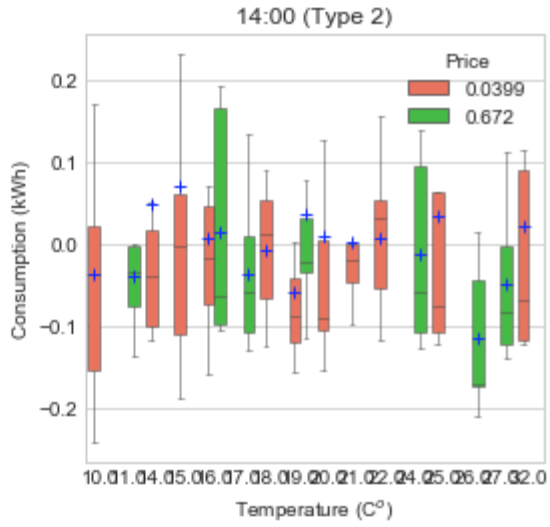

13:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

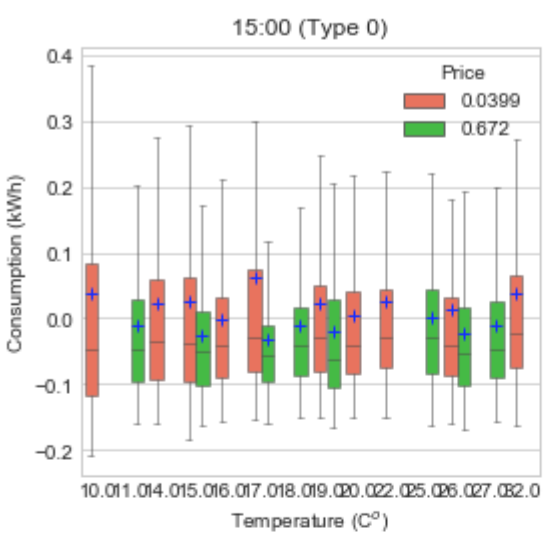

13:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

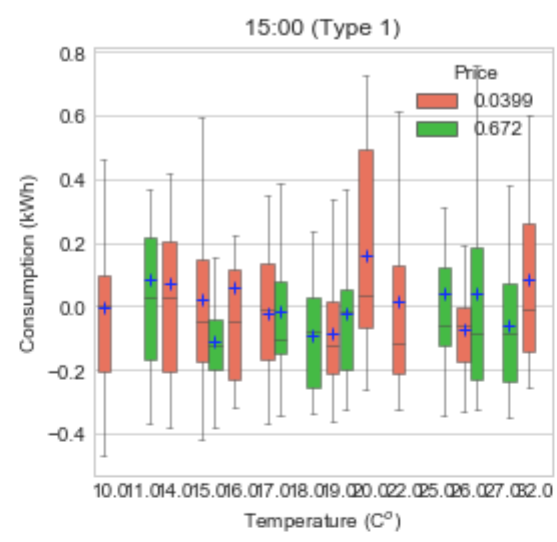

13:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

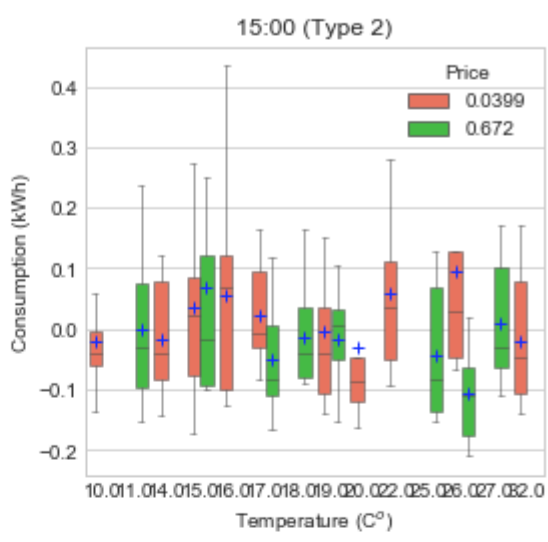

14:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

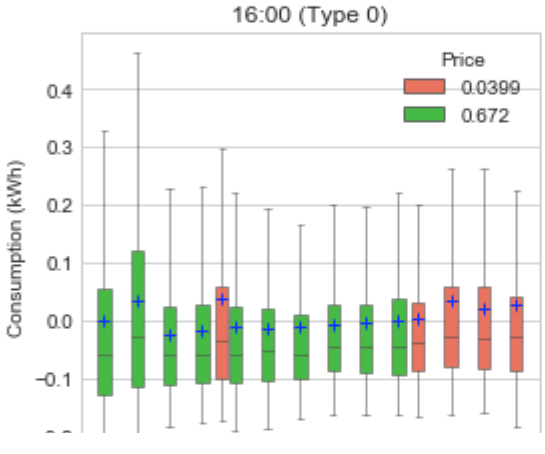

14:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

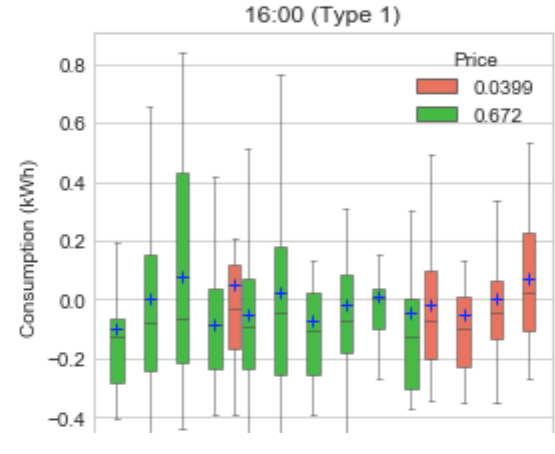

14:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

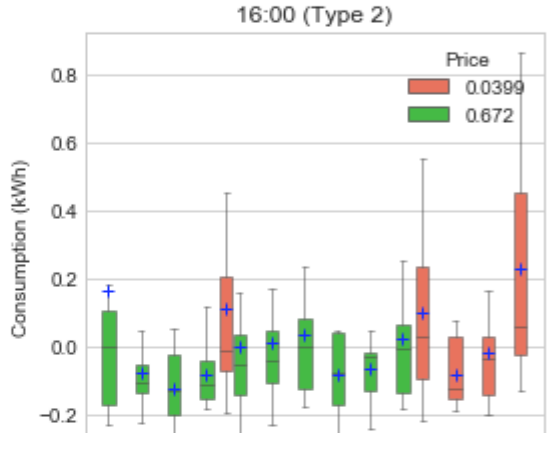

15:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)


15:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

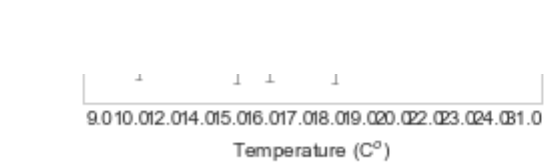

15:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

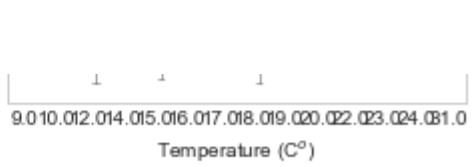

16:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)

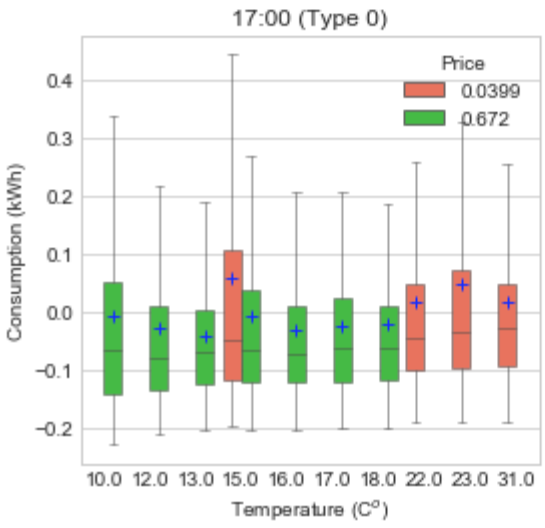

16:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)

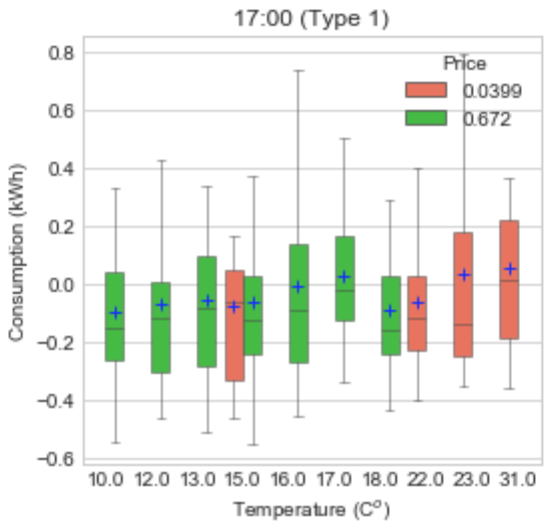

16:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)

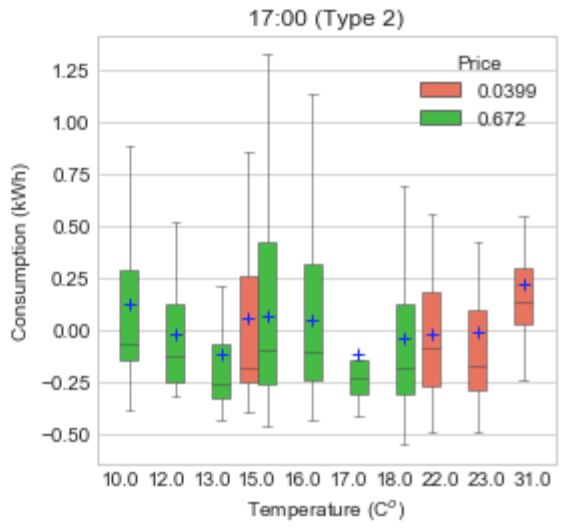

Temperature (C°)

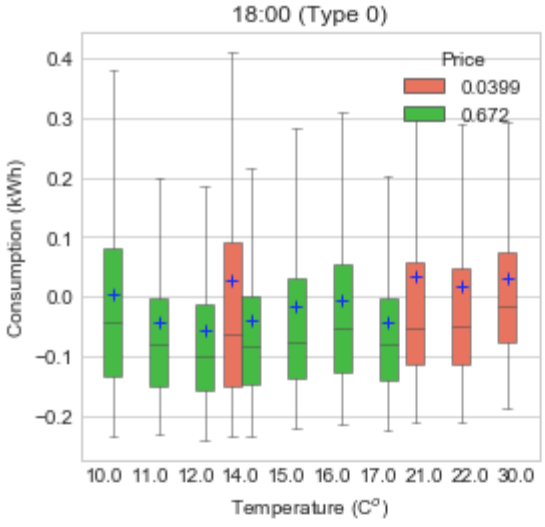

Temperature (C°)

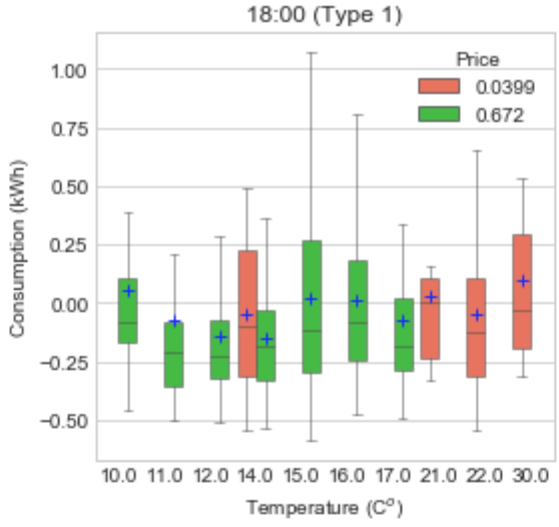

Temperature (C°)

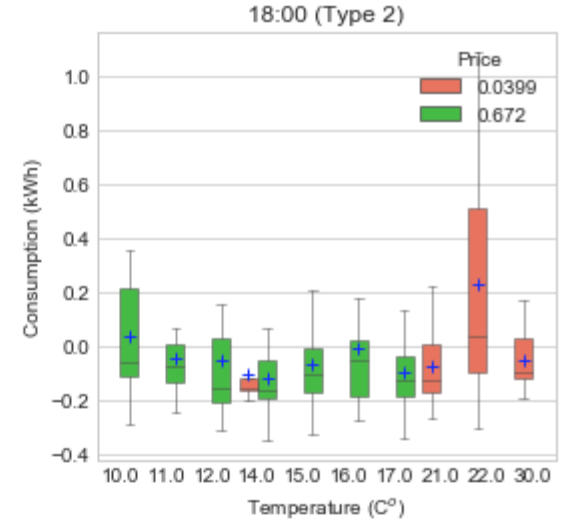

17:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

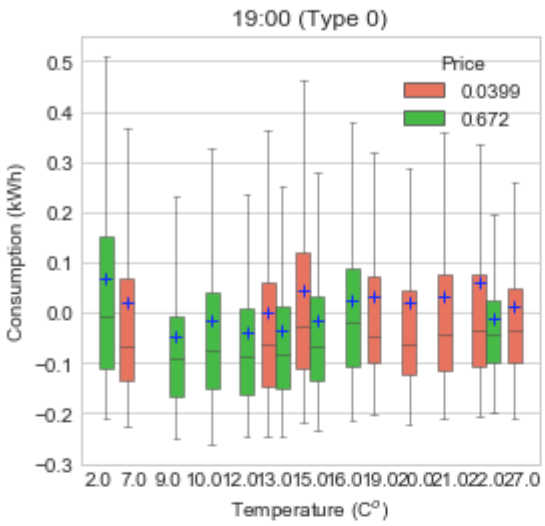

17:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

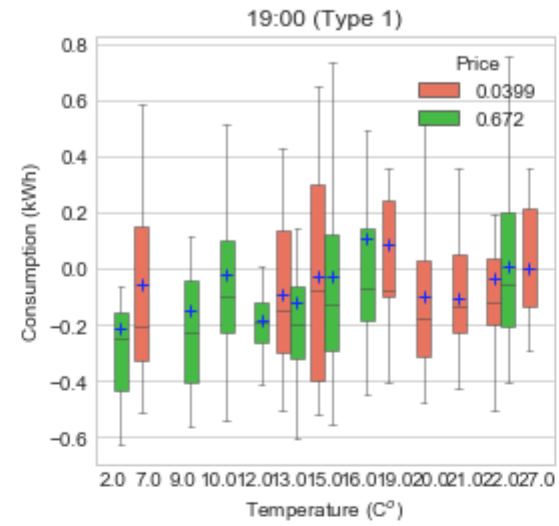

17:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

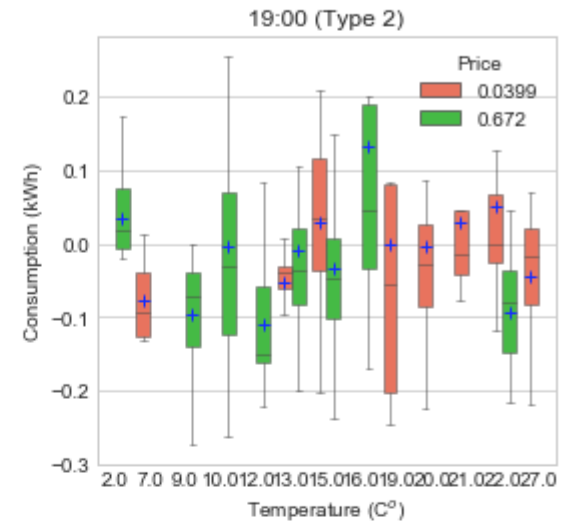

18:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

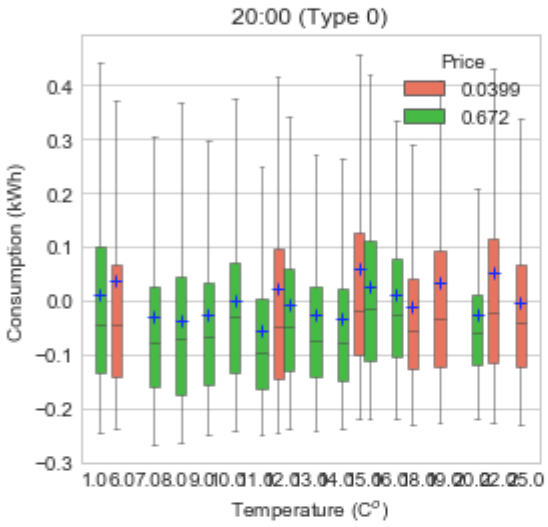

18:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

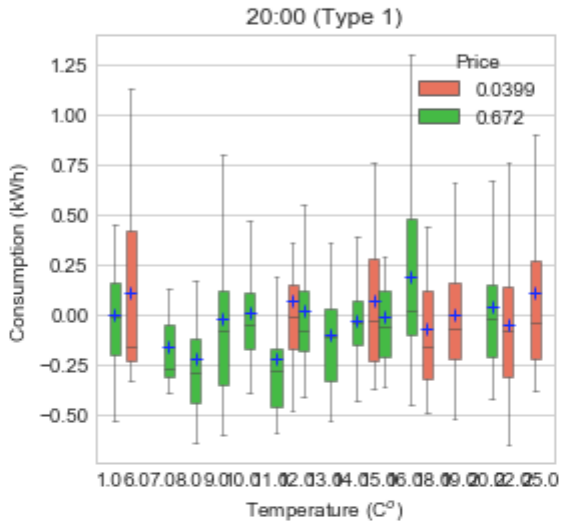

18:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

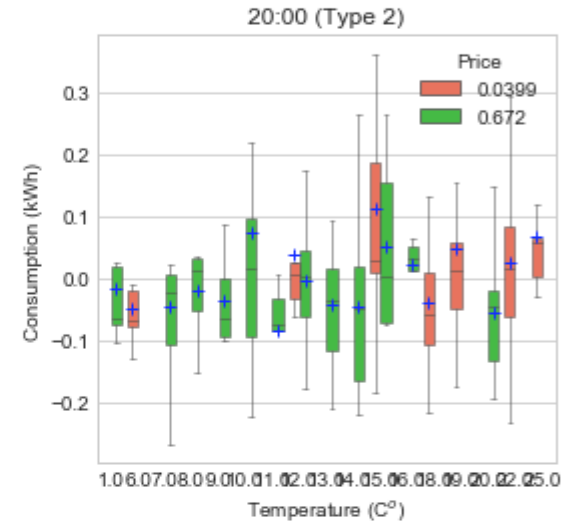

19:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

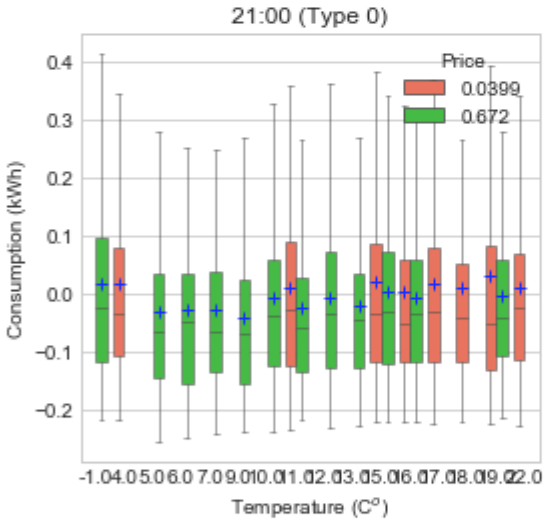

19:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

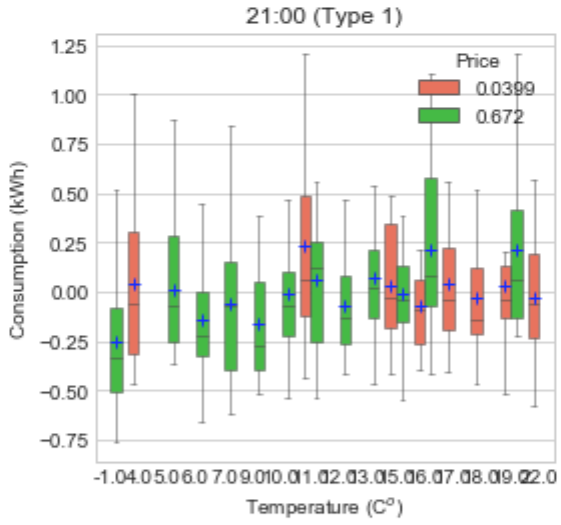

19:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

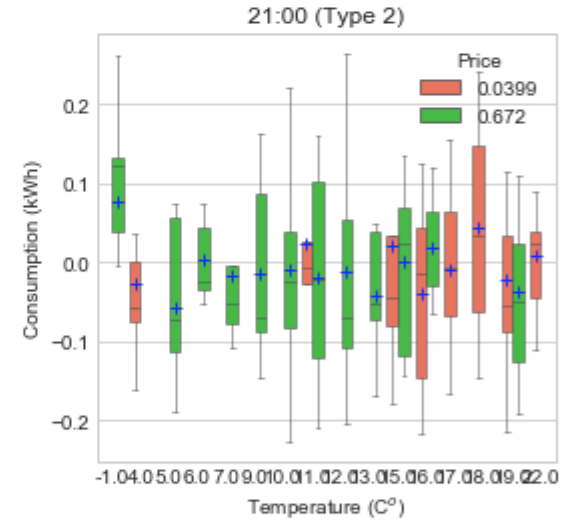

20:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

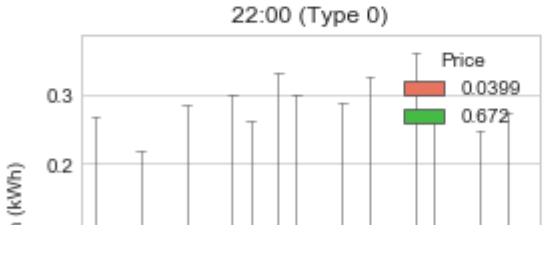

20:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

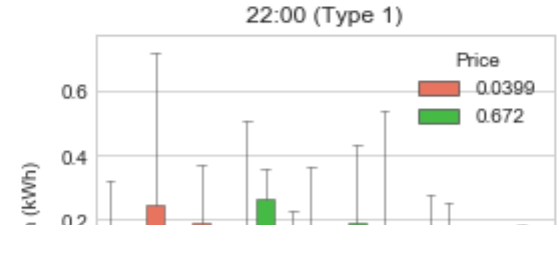

20:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

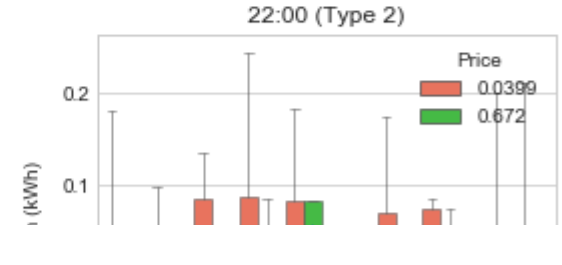

21:00 (Type 0)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)

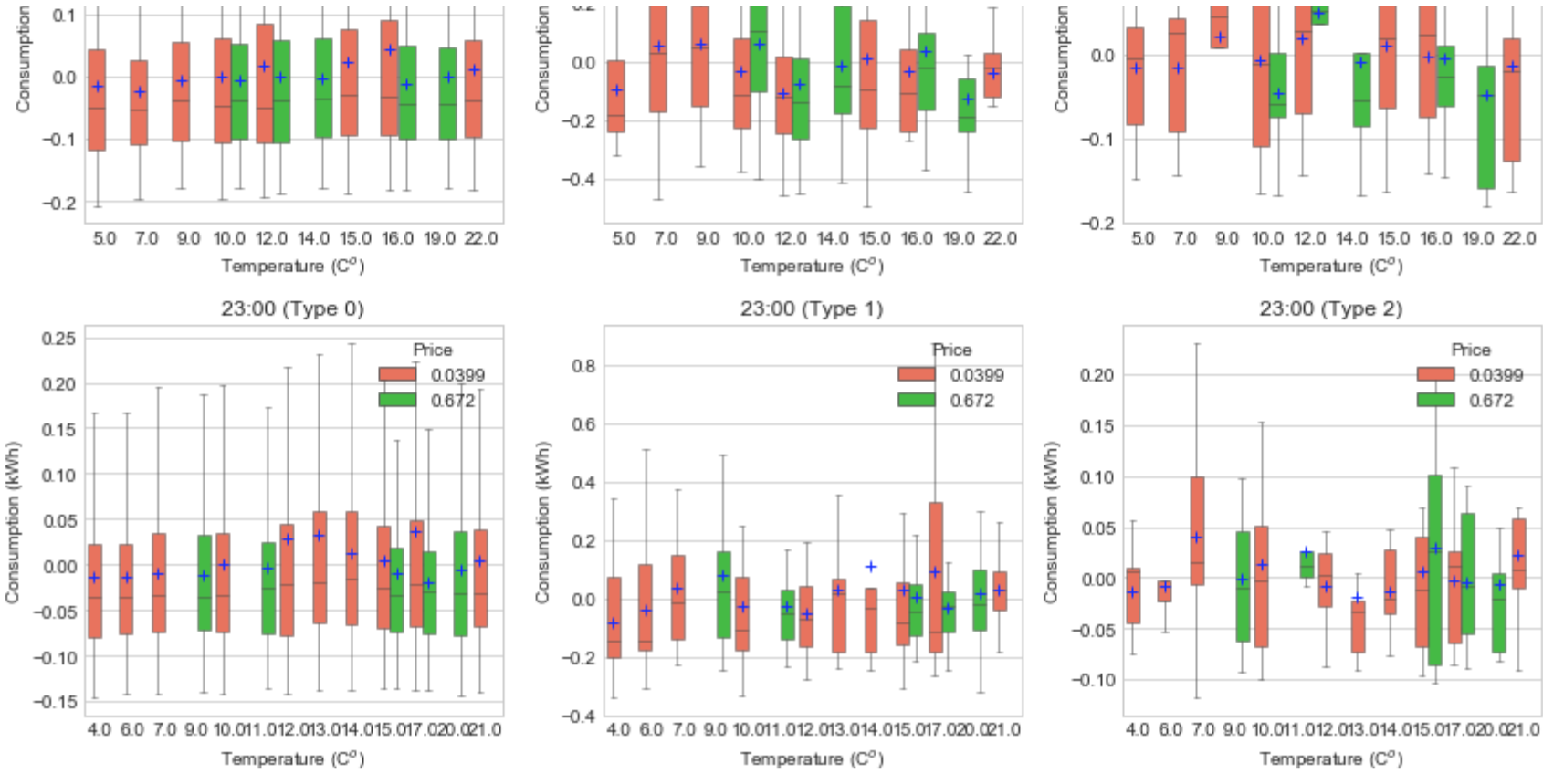

21:00 (Type 1)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)


21:00 (Type 2)
Price
0.0399
0.672
Consumption (kWh)
Temperature (C°)


22:00 (Type 0)
Price
0.0399
0.672

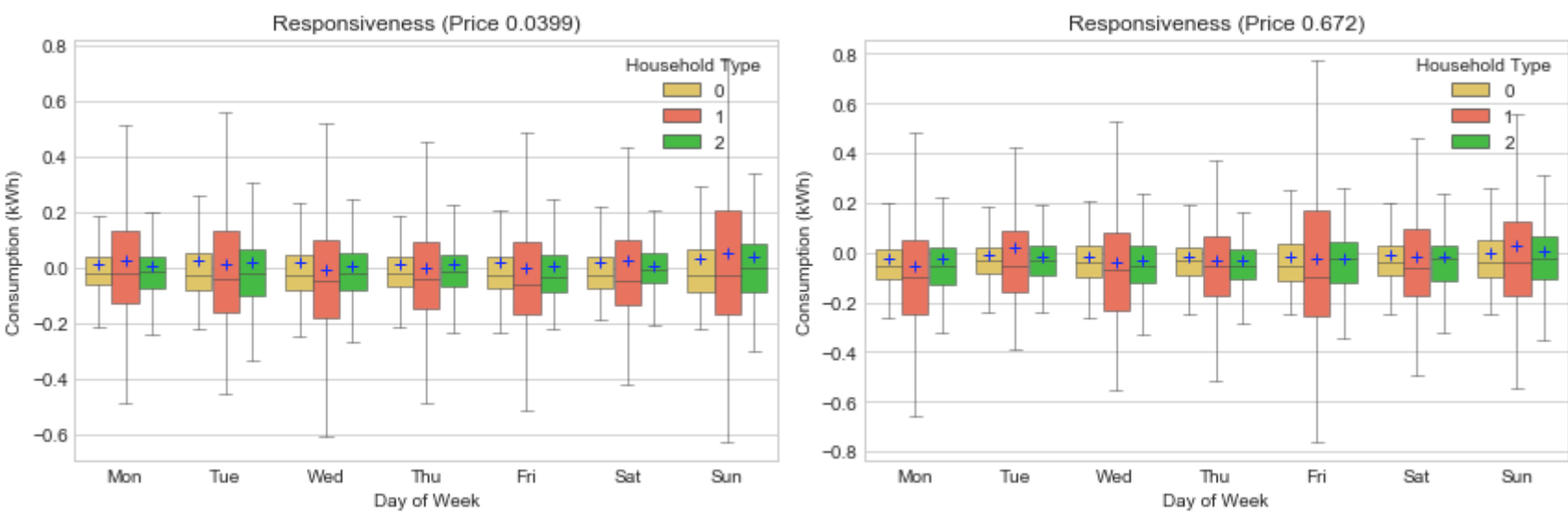

22:00 (Type 1)
Price
0.0399
0.672


22:00 (Type 2)
Price
0.0399
0.672

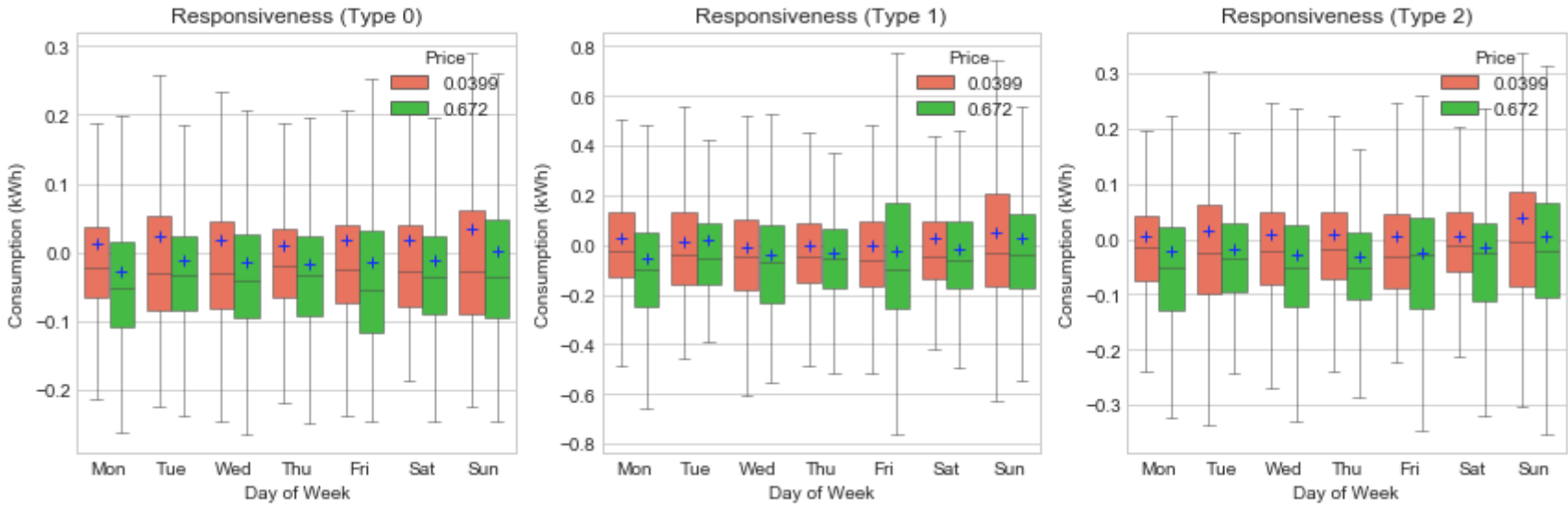


e) Day of week: prices

In [83]:
```
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,4))
ax_Ntou = [] # store subplot objects
weeks = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    ax_Ntou.append(fig_all.add_subplot(1, 2, (p + 1)))
    df_day = df_all_res_long[df_all_res_long['Price'] == prices[p]]
    if df_day.shape[0] >= 1:
        sns.boxplot(x="Day of week", y="Consumption", hue="Household type", data=df_day, ax=ax_Ntou[-1], pale
tte=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerf
acecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Price ' + str(prices[p]) + ')' )
        ax_Ntou[-1].set_xlabel(r'Day of Week')
        ax_Ntou[-1].set_xticklabels(weeks)
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
        l = ax_Ntou[-1].legend()
        l.set_title('Household Type')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```



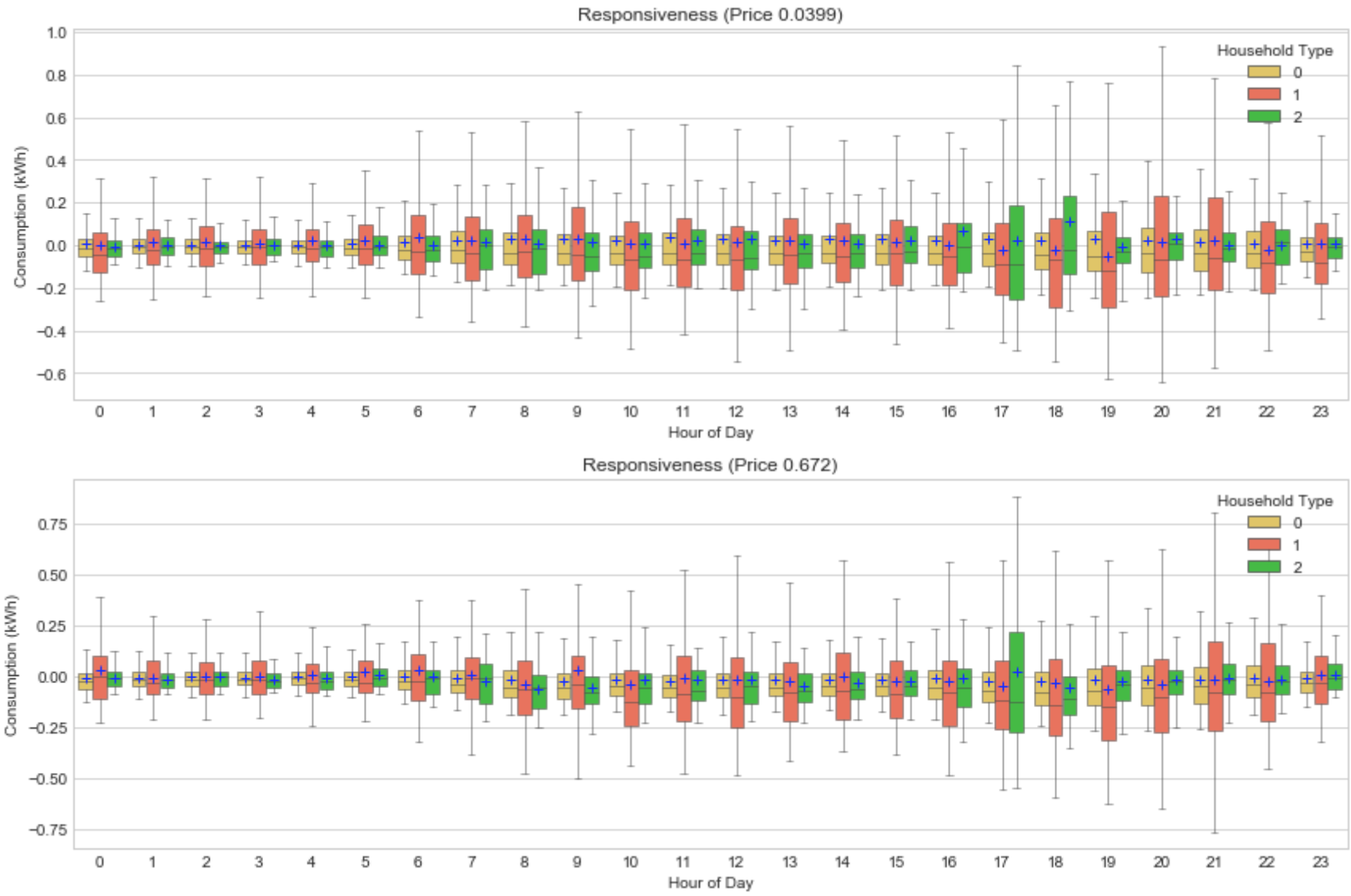
In [84]:
```
# price comparison
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,4))
ax_Ntou = [] # store subplot objects
weeks = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(1, 3, (g + 1)))
    df_day = df_all_res_long[df_all_res_long['Household type'] == g]
    if df_day.shape[0] >= 1:
        sns.boxplot(x="Day of week", y="Consumption", hue="Price", data=df_day, ax=ax_Ntou[-1], palette=['tom
ato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid
 blue", "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Type ' + str(g) + ')' )
        ax_Ntou[-1].set_xlabel(r'Day of Week')
        ax_Ntou[-1].set_xticklabels(weeks)
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
        l = ax_Ntou[-1].legend()
        l.set_title('Price')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```



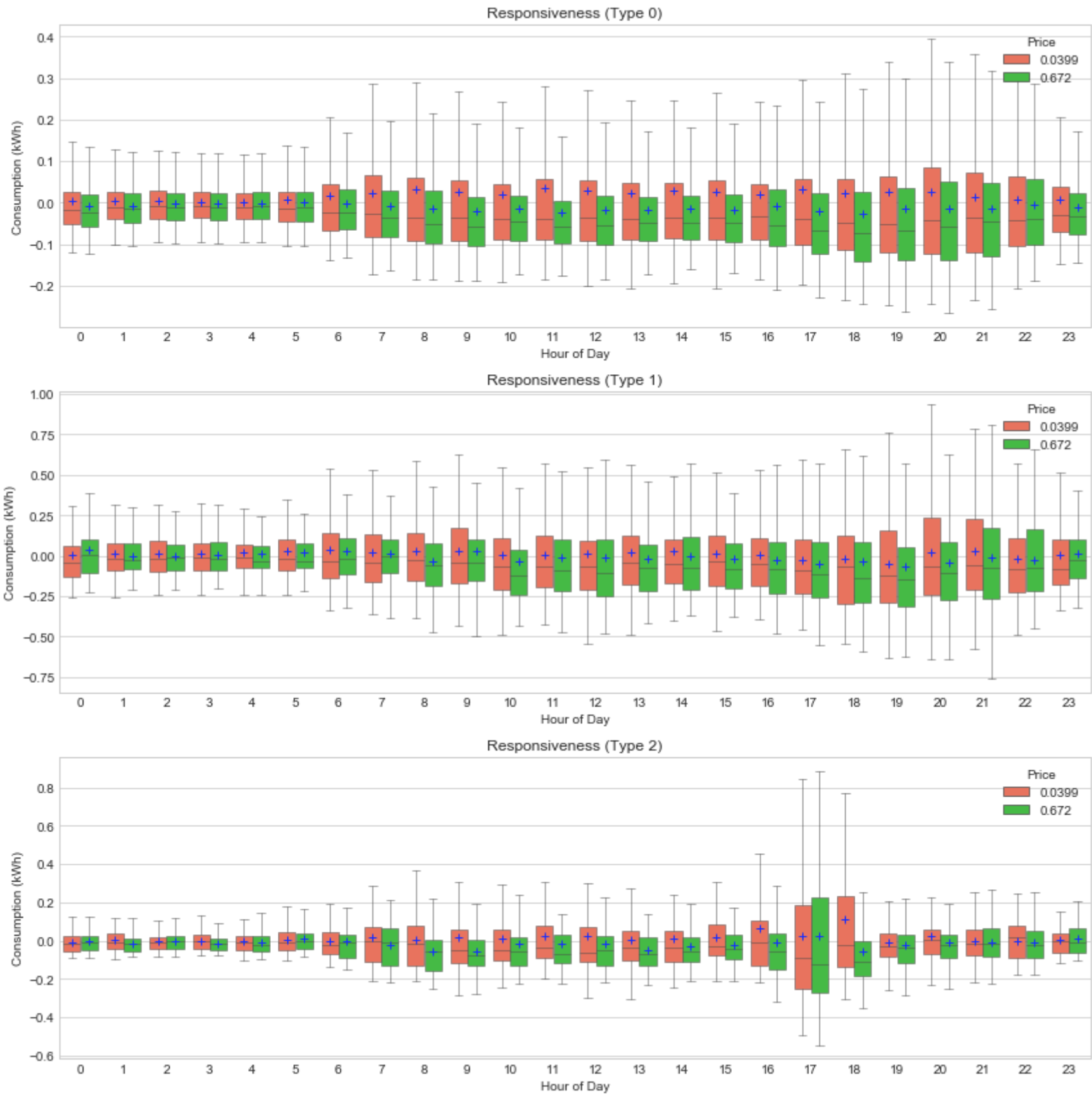
Average consumption kinda increases along days of weeks (from Monday to Sunday)

f) Hour of day: prices

In [85]:
```
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,8))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    ax_Ntou.append(fig_all.add_subplot(2, 1, (p + 1)))
    df_hour = df_all_res_long[df_all_res_long['Price'] == prices[p]]
    if df_hour.shape[0] >= 1:
        sns.boxplot(x="Hour of day", y="Consumption", hue="Household type", data=df_hour, ax=ax_Ntou[-1], pal
ette=['xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","marker
facecolor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Price ' + str(prices[p]) + ')' )
        ax_Ntou[-1].set_xlabel(r'Hour of Day')
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
        l = ax_Ntou[-1].legend()
#         ax_Ntou[-1].yaxis.grid(True)
#         ax_Ntou[-1].xaxis.grid(True)
        l.set_title('Household Type')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```



In [86]:
```
# price comparison
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,12))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(3, 1, (g + 1)))
    df_hour = df_all_res_long[df_all_res_long['Household type'] == g]
    if df_hour.shape[0] >= 1:
        sns.boxplot(x="Hour of day", y="Consumption", hue="Price", data=df_hour, ax=ax_Ntou[-1], palette=['to
mato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid
blue", "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Type ' + str(g) + ')' )
        ax_Ntou[-1].set_xlabel(r'Hour of Day')
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
        l = ax_Ntou[-1].legend()
#         ax_Ntou[-1].yaxis.grid(True)
#         ax_Ntou[-1].xaxis.grid(True)
        l.set_title('Price')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```



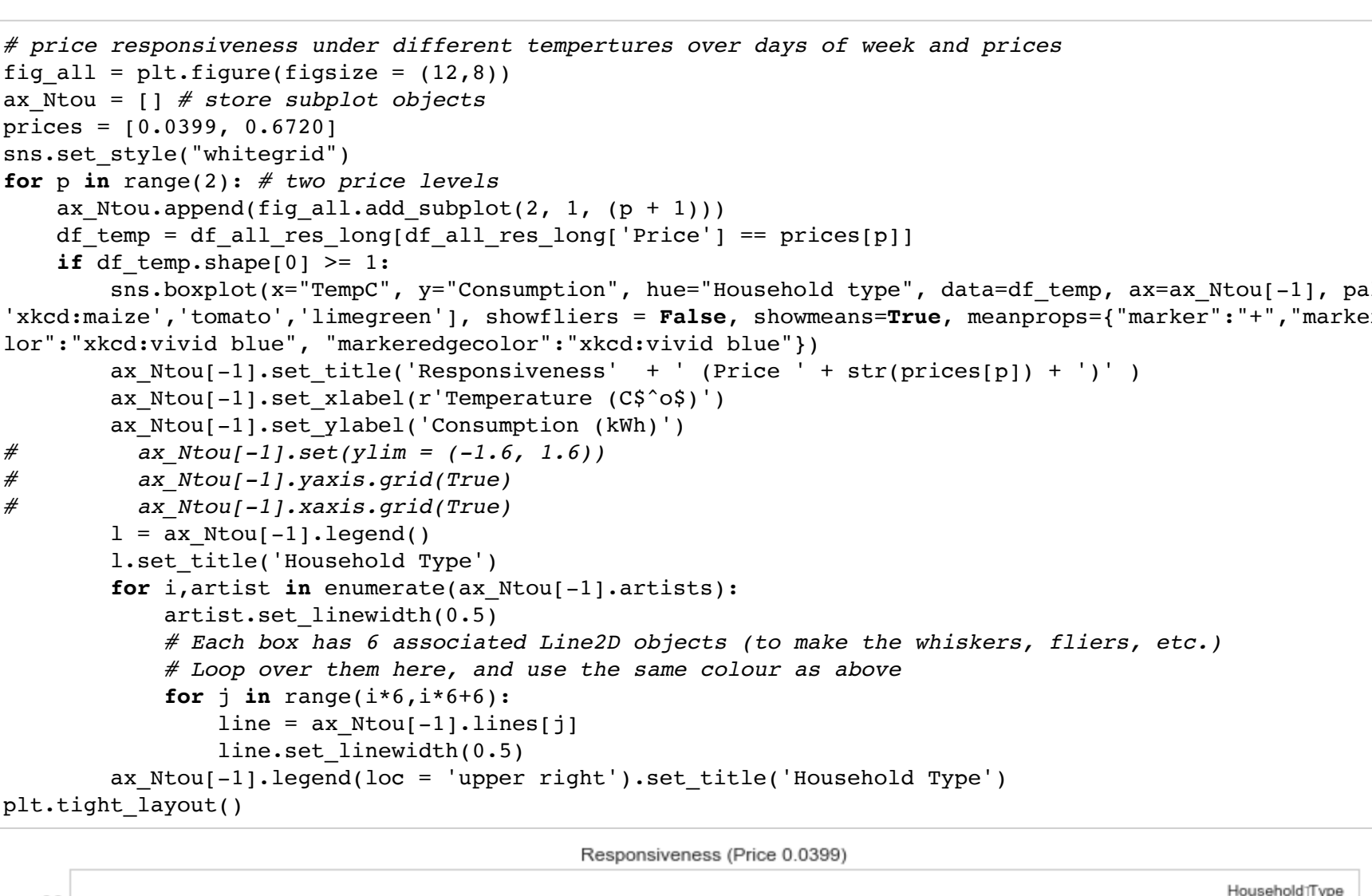
In a day, for low price, the max price responsivenss happens around 5pm - 8pm; for high price the max responsiveness happens around 1pm and 5pm or 9pm.

Midnight doesn't have too much responsiveness potential. Also high price has more impact on the median behvaior for all household groups.

Type 0 is ideal candidate for DR, type1 and 2 could be if design appropriately.

g) Temperature: prices

In [87]:
```
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,8))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for p in range(2): # two price levels
    ax_Ntou.append(fig_all.add_subplot(2, 1, (p + 1)))
    df_temp = df_all_res_long[df_all_res_long['Price'] == prices[p]]
    if df_temp.shape[0] >= 1:
        sns.boxplot(x="TempC", y="Consumption", hue="Household type", data=df_temp, ax=ax_Ntou[-1], palette=[
'xkcd:maize','tomato','limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfaceco
lor":"xkcd:vivid blue", "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Price ' + str(prices[p]) + ')' )
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
#         ax_Ntou[-1].set(ylim = (-1.6, 1.6))
#         ax_Ntou[-1].yaxis.grid(True)
#         ax_Ntou[-1].xaxis.grid(True)
        l = ax_Ntou[-1].legend()
        l.set_title('Household Type')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Household Type')
plt.tight_layout()
```



In [88]:
```
# price comparison
# price responsiveness under different tempertures over days of week and prices
fig_all = plt.figure(figsize = (12,12))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.6720]
sns.set_style("whitegrid")
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(3, 1, (g + 1)))
    df_temp = df_all_res_long[df_all_res_long['Household type'] == g]
    if df_temp.shape[0] >= 1:
        sns.boxplot(x="TempC", y="Consumption", hue="Price", data=df_temp, ax=ax_Ntou[-1], palette=['tomato',
'limegreen'], showfliers = False, showmeans=True, meanprops={"marker":"+","markerfacecolor":"xkcd:vivid blue"
, "markeredgecolor":"xkcd:vivid blue"})
        ax_Ntou[-1].set_title('Responsiveness'  + ' (Type ' + str(g) + ')' )
        ax_Ntou[-1].set_xlabel(r'Temperature (C$^o$)')
        ax_Ntou[-1].set_ylabel('Consumption (kWh)')
#         ax_Ntou[-1].set(ylim = (-1.6, 1.6))
#         ax_Ntou[-1].yaxis.grid(True)
#         ax_Ntou[-1].xaxis.grid(True)
        l = ax_Ntou[-1].legend()
        l.set_title('Price')
        for i,artist in enumerate(ax_Ntou[-1].artists):
            artist.set_linewidth(0.5)
            # Each box has 6 associated Line2D objects (to make the whiskers, fliers, etc.)
            # Loop over them here, and use the same colour as above
            for j in range(i*6,i*6+6):
                line = ax_Ntou[-1].lines[j]
                line.set_linewidth(0.5)
        ax_Ntou[-1].legend(loc = 'upper right').set_title('Price')
plt.tight_layout()
```


Compare to other temperatures, -1 & 2 celcius degrees for type 1 and 31 celcius degree for type 2 have some obvious impact, so when design DR program the temperature impact should be considered.

Additional info

Consumption variance of different type of households over 24 hours (Warm Season).

In [60]:
```
# type 0 event time consumption prediction
pred_result = []
for i in range(df_type0.shape[0]):
    pred_result.append(predict(df_regression_0, df_type0['Day of week'].iloc[i], df_type0['Hour of day'].iloc
[i], df_type0['TempC'].iloc[i]))
df_type0['Predicted consumption'] = pred_result
# price responsiveness elementwise
df_type0_con = df_type0.copy() # store the consumption not the response, just for this additional test
# df_type0_con.iloc[:,1:houseGroupDf.counts.iloc[0] + 1] = df_type0.iloc[:,1:houseGroupDf.counts.iloc[0] +
 1].sub(df_type0['Predicted consumption'], axis = 0)

# type 1 event time consumption prediction
pred_result = []
for i in range(df_type1.shape[0]):
    pred_result.append(predict(df_regression_1, df_type1['Day of week'].iloc[i], df_type1['Hour of day'].iloc
[i], df_type1['TempC'].iloc[i]))
df_type1['Predicted consumption'] = pred_result

# price responsiveness elementwise
df_type1_con = df_type1.copy()
# df_type1_con.iloc[:,1:houseGroupDf.counts.iloc[1] + 1] = df_type1.iloc[:,1:houseGroupDf.counts.iloc[1] +
 1].sub(df_type1['Predicted consumption'], axis = 0)

# type 2 event time consumption prediction
pred_result = []
for i in range(df_type2.shape[0]):
    pred_result.append(predict(df_regression_2, df_type2['Day of week'].iloc[i], df_type2['Hour of day'].iloc
[i], df_type2['TempC'].iloc[i]))
df_type2['Predicted consumption'] = pred_result

# price responsiveness elementwise
df_type2_con = df_type2.copy()
# df_type2_con.iloc[:,1:houseGroupDf.counts.iloc[2] + 1] = df_type2.iloc[:,1:houseGroupDf.counts.iloc[2] +
 1].sub(df_type2['Predicted consumption'], axis = 0)
```

In [61]:
```
df_type0
```

Out[61]:

	GMT	D0000	D0001	D0005	D0007	D0010	D0012	D0013	D0016	D0018	...	D1023	D1024	TempC	TempF	Price	Eve
0	2013-04-05 19:00:00	0.462	0.270	0.555	0.408	0.000	0.396	0.953	0.425	0.045	...	0.215	0.461	2.333333	36.333333	0.6720	H3
1	2013-04-05 20:00:00	0.399	0.306	0.525	0.260	0.000	0.190	0.354	0.435	0.040	...	0.199	0.214	0.666667	33.666667	0.6720	H3
2	2013-04-05 21:00:00	0.366	0.256	0.533	0.269	0.001	0.143	0.369	0.259	0.053	...	0.136	0.179	-1.000000	31.000000	0.6720	H3
3	2013-04-08 10:00:00	0.587	0.228	0.439	0.441	0.000	0.305	0.143	0.071	0.099	...	0.098	0.120	8.000000	46.333333	0.6720	H3
4	2013-04-08 11:00:00	0.384	0.138	0.447	2.823	0.000	0.151	0.254	0.071	0.091	...	0.040	0.095	9.000000	47.666667	0.6720	H3
5	2013-04-08 12:00:00	0.369	0.163	0.201	0.302	0.001	0.870	0.131	0.071	0.059	...	0.047	0.085	10.000000	49.000000	0.6720	H3
6	2013-04-11 01:00:00	0.097	0.072	0.147	0.170	0.001	0.139	0.089	0.074	0.052	...	0.116	0.138	4.333333	39.333333	0.6720	H3
7	2013-04-11 02:00:00	0.111	0.119	0.138	0.146	0.000	0.135	0.093	0.072	0.083	...	0.048	0.107	4.666667	39.666667	0.6720	H3
8	2013-04-11 03:00:00	0.113	0.061	0.150	0.168	0.000	0.111	0.104	0.072	0.046	...	0.027	0.101	5.000000	40.000000	0.6720	H3
9	2013-04-13 16:00:00	0.346	0.042	0.447	0.276	0.000	0.466	0.250	0.077	0.038	...	0.017	0.134	9.333333	48.666667	0.6720	H6
10	2013-04-13 17:00:00	0.488	0.123	0.496	0.281	0.001	0.390	0.336	0.069	0.078	...	0.028	0.135	9.666667	49.333333	0.6720	H6
11	2013-04-13 18:00:00	0.613	0.229	0.528	0.561	0.000	0.587	0.257	0.238	0.050	...	0.049	0.192	10.000000	50.000000	0.6720	H6
12	2013-04-13 19:00:00	1.085	0.109	0.418	0.308	0.000	0.476	0.259	0.464	0.051	...	0.016	0.189	10.000000	50.000000	0.6720	H6
13	2013-04-13 20:00:00	1.885	0.087	0.421	0.303	0.000	0.475	0.309	0.164	0.076	...	0.046	0.268	10.000000	50.000000	0.6720	H6
14	2013-04-13 21:00:00	0.449	0.116	0.375	0.250	0.001	0.386	0.255	0.358	0.088	...	0.118	0.186	10.000000	50.000000	0.6720	H6
15	2013-04-16 04:00:00	0.135	0.115	0.157	0.254	0.000	0.116	0.108	0.093	0.082	...	0.056	0.074	8.333333	47.000000	0.0399	L6
16	2013-04-16 05:00:00	1.072	0.066	0.192	0.192	0.001	0.146	0.112	0.095	0.052	...	0.017	0.083	8.666667	47.000000	0.0399	L6
17	2013-04-16 06:00:00	0.301	0.113	0.168	0.164	0.000	0.156	0.207	0.451	0.080	...	0.050	0.054	9.000000	47.000000	0.0399	L6
18	2013-04-16 07:00:00	1.223	0.247	0.327	0.276	0.000	0.303	1.137	0.197	0.134	...	0.025	0.097	10.000000	49.000000	0.0399	L6
19	2013-04-16 08:00:00	1.037	0.094	0.218	0.274	0.001	0.340	0.700	0.069	0.446	...	0.028	0.050	11.000000	51.000000	0.0399	L6

	GMT	D0000	D0001	D0005	D0007	D0010	D0012	D0013	D0016	D0018	...	D1023	D1024	TempC	TempF	Price	Eve
20	2013-04-16 09:00:00	0.750	0.084	0.253	0.385	0.000	0.254	1.067	0.068	0.207	...	0.124	0.083	12.000000	53.000000	0.0399	L6
21	2013-04-21 10:00:00	0.401	0.397	1.015	0.367	0.031	0.206	0.768	0.089	0.907	...	0.030	0.080	11.666667	53.333333	0.0399	L3
22	2013-04-21 11:00:00	1.910	0.076	0.408	0.434	0.017	0.180	1.252	0.072	0.954	...	0.102	0.070	13.333333	55.666667	0.0399	L3
23	2013-04-21 12:00:00	0.859	0.111	0.393	0.795	0.026	0.145	1.150	0.084	0.065	...	0.123	0.222	15.000000	58.000000	0.0399	L3
24	2013-04-23 04:00:00	0.133	0.116	0.169	0.328	0.030	0.092	0.099	0.070	0.047	...	0.039	0.088	9.000000	48.000000	0.0399	CM
25	2013-04-23 05:00:00	0.095	0.105	0.231	0.308	0.017	0.126	0.089	0.297	0.071	...	0.019	0.046	9.000000	48.000000	0.0399	CM
26	2013-04-23 06:00:00	0.154	0.085	0.304	0.270	0.031	0.185	0.256	0.161	0.052	...	0.063	0.083	9.000000	48.000000	0.0399	CM
27	2013-04-23 07:00:00	0.209	0.104	0.262	0.432	0.088	0.148	0.098	0.074	0.340	...	0.019	0.192	11.000000	51.333333	0.0399	CM
28	2013-04-23 08:00:00	0.325	0.253	0.250	1.073	0.114	0.972	0.103	0.086	0.710	...	0.032	0.064	13.000000	54.666667	0.0399	CM
29	2013-04-23 09:00:00	0.242	0.346	0.373	0.301	0.118	0.872	0.098	0.071	0.329	...	0.073	0.070	15.000000	58.000000	0.0399	CM
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
471	2013-10-20 04:00:00	0.328	0.087	0.130	0.210	0.043	0.111	0.109	0.071	0.065	...	0.061	0.057	15.000000	60.000000	0.0399	CM
472	2013-10-20 05:00:00	0.089	0.056	0.162	0.342	0.037	0.134	0.138	0.078	0.075	...	0.021	0.049	15.000000	60.000000	0.0399	CM
473	2013-10-20 06:00:00	0.084	0.130	0.183	0.333	0.039	0.181	0.132	0.078	0.055	...	0.037	0.070	15.000000	60.000000	0.0399	CM
474	2013-10-20 07:00:00	0.085	0.146	0.303	0.344	0.049	0.318	0.110	0.209	0.477	...	0.029	0.061	15.000000	60.000000	0.0399	CM
475	2013-10-20 08:00:00	0.085	0.125	0.334	0.319	0.037	1.410	0.105	0.304	0.655	...	0.030	0.062	15.000000	60.000000	0.0399	CM
476	2013-10-20 09:00:00	0.104	0.344	0.214	0.257	0.122	0.233	0.572	0.093	1.090	...	0.038	0.125	15.000000	60.000000	0.0399	CM
477	2013-10-20 10:00:00	0.482	0.097	0.229	0.335	0.053	0.703	0.165	0.079	0.107	...	0.068	0.080	15.000000	60.000000	0.0399	CM
478	2013-10-20 11:00:00	0.360	0.175	0.325	0.280	0.028	0.324	0.451	0.072	0.199	...	0.120	0.041	15.000000	60.000000	0.0399	CM
479	2013-10-20 12:00:00	0.329	0.092	0.472	0.371	0.018	0.206	0.670	0.084	0.744	...	0.024	0.070	15.000000	60.000000	0.0399	CM
480	2013-10-20 13:00:00	0.317	0.084	0.571	0.263	0.020	0.189	0.238	0.067	0.944	...	0.057	0.456	15.000000	60.000000	0.0399	CM

	GMT	D0000	D0001	D0005	D0007	D0010	D0012	D0013	D0016	D0018	...	D1023	D1024	TempC	TempF	Price	Eve
481	2013-10-20 14:00:00	0.291	0.067	0.460	0.380	0.081	0.122	0.668	0.074	0.246	...	0.110	0.103	15.000000	60.000000	0.0399	CM
482	2013-10-20 15:00:00	0.370	0.110	0.464	0.281	0.028	0.137	1.179	0.074	0.318	...	0.100	0.103	15.000000	60.000000	0.0399	CM
483	2013-10-20 16:00:00	0.504	0.128	0.496	0.279	0.018	0.118	0.457	0.073	0.117	...	0.048	0.207	15.000000	60.000000	0.6720	CM
484	2013-10-20 17:00:00	0.602	0.137	0.520	0.254	0.018	0.134	0.472	0.218	0.129	...	0.082	0.244	15.000000	60.000000	0.6720	CM
485	2013-10-20 18:00:00	0.372	0.134	0.550	0.451	0.032	0.171	0.324	0.386	0.415	...	0.067	0.115	15.000000	60.000000	0.6720	CM
486	2013-10-20 19:00:00	0.512	0.122	0.492	0.365	0.022	0.449	1.009	0.209	0.322	...	0.108	0.076	15.000000	60.000000	0.6720	CM
487	2013-10-20 20:00:00	0.382	0.066	0.562	0.280	0.020	0.415	0.606	0.182	0.419	...	0.083	0.085	15.000000	60.000000	0.6720	CM
488	2013-10-20 21:00:00	0.715	0.335	0.528	0.280	0.027	0.398	0.362	0.224	0.459	...	0.122	0.100	15.000000	60.000000	0.6720	CM
489	2013-10-20 22:00:00	0.293	0.400	0.481	0.388	0.027	0.431	0.164	0.198	0.401	...	0.109	0.357	15.000000	59.666667	0.0399	CM
490	2013-10-20 23:00:00	0.255	0.317	0.508	0.295	0.018	0.188	0.110	0.120	0.109	...	0.110	0.068	15.000000	59.333333	0.0399	CM
491	2013-10-21 00:00:00	0.162	0.070	0.408	0.271	0.019	0.152	0.110	0.073	0.133	...	0.079	0.071	15.000000	59.000000	0.0399	CM
492	2013-10-21 01:00:00	0.102	0.074	0.392	0.167	0.030	0.109	0.119	0.085	0.084	...	0.022	0.044	15.000000	59.000000	0.0399	CM
493	2013-10-21 02:00:00	0.074	0.085	0.301	0.143	0.023	0.137	0.112	0.070	0.047	...	0.056	0.073	15.000000	59.000000	0.0399	CM
494	2013-10-21 03:00:00	0.277	0.066	0.129	0.163	0.021	0.116	0.112	0.070	0.084	...	0.017	0.066	15.000000	59.000000	0.0399	CM
495	2013-10-31 02:00:00	0.092	0.086	0.146	0.143	0.018	0.115	0.104	0.076	0.059	...	0.017	0.096	13.000000	56.000000	0.6720	H3
496	2013-10-31 03:00:00	0.042	0.085	0.134	0.141	0.020	0.113	0.102	0.082	0.084	...	0.068	0.071	13.000000	56.000000	0.6720	H3
497	2013-10-31 04:00:00	0.074	0.072	0.167	0.141	0.020	0.152	0.105	0.069	0.044	...	0.019	0.078	13.000000	56.000000	0.6720	H3
498	2013-10-31 05:00:00	0.034	0.058	0.156	0.259	0.029	0.103	0.105	0.069	0.109	...	0.037	0.067	13.000000	56.000000	0.0399	L3
499	2013-10-31 06:00:00	0.087	0.143	0.134	0.308	0.024	0.139	0.106	0.247	0.054	...	0.039	0.084	13.000000	56.000000	0.0399	L3
500	2013-10-31 07:00:00	0.061	0.142	0.176	0.324	0.019	1.497	0.320	0.495	0.082	...	0.016	0.071	13.000000	56.000000	0.0399	L3

501 rows × 420 columns

In [62]:
```
# select each user ID and form a seperate dataframe
# first change all the user ID to "Consumption"
new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[0]):
    new_col.append('Consumption')
new_col = new_col + list(df_type0_con.columns)[-8:] # the new column names
df_type0_con.columns = new_col
df_all_con_long = pd.DataFrame()
for i in range(1, houseGroupDf.counts.iloc[0] + 1):
    df_all_con_long = df_all_con_long.append(df_type0_con.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[1]): # for renaming the second type
    new_col.append('Consumption')
new_col = new_col + list(df_type1_con.columns)[-8:] # the new column names
df_type1_con.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[1] + 1):
    df_all_con_long = df_all_con_long.append(df_type1_con.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[2]): # for renaming the third type
    new_col.append('Consumption')
new_col = new_col + list(df_type2_con.columns)[-8:] # the new column names
df_type2_con.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[2] + 1):
    df_all_con_long = df_all_con_long.append(df_type2_con.iloc[:,[0,i,-8,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

# for clearer visulization, round the temperatures
df_all_con_long['TempC'] = df_all_con_long.TempC.round(0)
df_all_con_long['TempF'] = df_all_con_long.TempF.round(0)
df_all_con_long # ready for the price responsiveness analysis
```

Out[62]:

	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
0	2013-04-05 19:00:00	0.462	2.0	36.0	0.6720	H3	4	19	0	0.243337
1	2013-04-05 20:00:00	0.399	1.0	34.0	0.6720	H3	4	20	0	0.270188
2	2013-04-05 21:00:00	0.366	-1.0	31.0	0.6720	H3	4	21	0	0.237944
3	2013-04-08 10:00:00	0.587	8.0	46.0	0.6720	H3	0	10	0	0.202466
4	2013-04-08 11:00:00	0.384	9.0	48.0	0.6720	H3	0	11	0	0.203218
5	2013-04-08 12:00:00	0.369	10.0	49.0	0.6720	H3	0	12	0	0.217232
6	2013-04-11 01:00:00	0.097	4.0	39.0	0.6720	H3	3	1	0	0.109336
7	2013-04-11 02:00:00	0.111	5.0	40.0	0.6720	H3	3	2	0	0.104794
8	2013-04-11 03:00:00	0.113	5.0	40.0	0.6720	H3	3	3	0	0.103372
9	2013-04-13 16:00:00	0.346	9.0	49.0	0.6720	H6	5	16	0	0.238432
10	2013-04-13 17:00:00	0.488	10.0	49.0	0.6720	H6	5	17	0	0.264226
11	2013-04-13 18:00:00	0.613	10.0	50.0	0.6720	H6	5	18	0	0.269092
12	2013-04-13 19:00:00	1.085	10.0	50.0	0.6720	H6	5	19	0	0.289251
13	2013-04-13 20:00:00	1.885	10.0	50.0	0.6720	H6	5	20	0	0.278164
14	2013-04-13 21:00:00	0.449	10.0	50.0	0.6720	H6	5	21	0	0.251333
15	2013-04-16 04:00:00	0.135	8.0	47.0	0.0399	L6	1	4	0	0.110253
16	2013-04-16 05:00:00	1.072	9.0	47.0	0.0399	L6	1	5	0	0.123750
17	2013-04-16 06:00:00	0.301	9.0	47.0	0.0399	L6	1	6	0	0.165405
18	2013-04-16 07:00:00	1.223	10.0	49.0	0.0399	L6	1	7	0	0.192275
19	2013-04-16 08:00:00	1.037	11.0	51.0	0.0399	L6	1	8	0	0.194820
20	2013-04-16 09:00:00	0.750	12.0	53.0	0.0399	L6	1	9	0	0.190140
21	2013-04-21 10:00:00	0.401	12.0	53.0	0.0399	L3	6	10	0	0.216245
22	2013-04-21 11:00:00	1.910	13.0	56.0	0.0399	L3	6	11	0	0.218322
23	2013-04-21 12:00:00	0.859	15.0	58.0	0.0399	L3	6	12	0	0.232024
24	2013-04-23 04:00:00	0.133	9.0	48.0	0.0399	CM	1	4	0	0.110499
25	2013-04-23 05:00:00	0.095	9.0	48.0	0.0399	CM	1	5	0	0.123756
26	2013-04-23 06:00:00	0.154	9.0	48.0	0.0399	CM	1	6	0	0.165405
27	2013-04-23 07:00:00	0.209	11.0	51.0	0.0399	CM	1	7	0	0.191424
28	2013-04-23 08:00:00	0.325	13.0	55.0	0.0399	CM	1	8	0	0.192104

	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
29	2013-04-23 09:00:00	0.242	15.0	58.0	0.0399	CM	1	9	0	0.186690
...	...	...	...	...	...	...	...	...	...	...
225420	2013-10-20 04:00:00	0.132	15.0	60.0	0.0399	CM	6	4	2	0.120845
225421	2013-10-20 05:00:00	0.096	15.0	60.0	0.0399	CM	6	5	2	0.130691
225422	2013-10-20 06:00:00	0.206	15.0	60.0	0.0399	CM	6	6	2	0.152508
225423	2013-10-20 07:00:00	0.234	15.0	60.0	0.0399	CM	6	7	2	0.241426
225424	2013-10-20 08:00:00	0.345	15.0	60.0	0.0399	CM	6	8	2	0.254882
225425	2013-10-20 09:00:00	0.231	15.0	60.0	0.0399	CM	6	9	2	0.259246
225426	2013-10-20 10:00:00	0.135	15.0	60.0	0.0399	CM	6	10	2	0.282440
225427	2013-10-20 11:00:00	0.127	15.0	60.0	0.0399	CM	6	11	2	0.329020
225428	2013-10-20 12:00:00	0.212	15.0	60.0	0.0399	CM	6	12	2	0.341375
225429	2013-10-20 13:00:00	0.148	15.0	60.0	0.0399	CM	6	13	2	0.274778
225430	2013-10-20 14:00:00	0.182	15.0	60.0	0.0399	CM	6	14	2	0.287356
225431	2013-10-20 15:00:00	0.265	15.0	60.0	0.0399	CM	6	15	2	0.342283
225432	2013-10-20 16:00:00	0.333	15.0	60.0	0.6720	CM	6	16	2	0.532782
225433	2013-10-20 17:00:00	0.237	15.0	60.0	0.6720	CM	6	17	2	0.375501
225434	2013-10-20 18:00:00	0.192	15.0	60.0	0.6720	CM	6	18	2	0.253967
225435	2013-10-20 19:00:00	0.154	15.0	60.0	0.6720	CM	6	19	2	0.259465
225436	2013-10-20 20:00:00	0.216	15.0	60.0	0.6720	CM	6	20	2	0.262970
225437	2013-10-20 21:00:00	0.203	15.0	60.0	0.6720	CM	6	21	2	0.264596
225438	2013-10-20 22:00:00	0.187	15.0	60.0	0.0399	CM	6	22	2	0.174913
225439	2013-10-20 23:00:00	0.177	15.0	59.0	0.0399	CM	6	23	2	0.124752
225440	2013-10-21 00:00:00	0.132	15.0	59.0	0.0399	CM	0	0	2	0.119303
225441	2013-10-21 01:00:00	0.098	15.0	59.0	0.0399	CM	0	1	2	0.115503
225442	2013-10-21 02:00:00	0.161	15.0	59.0	0.0399	CM	0	2	2	0.115871
225443	2013-10-21 03:00:00	0.130	15.0	59.0	0.0399	CM	0	3	2	0.117904
225444	2013-10-31 02:00:00	0.163	13.0	56.0	0.6720	H3	3	2	2	0.116686
225445	2013-10-31 03:00:00	0.090	13.0	56.0	0.6720	H3	3	3	2	0.115433
225446	2013-10-31 04:00:00	0.127	13.0	56.0	0.6720	H3	3	4	2	0.110176
225447	2013-10-31 05:00:00	0.141	13.0	56.0	0.0399	L3	3	5	2	0.128823

	GMT	Consumption	TempC	TempF	Price	Event_tags	Day of week	Hour of day	Household type	Predicted consumption
225448	2013-10-31 06:00:00	0.089	13.0	56.0	0.0399	L3	3	6	2	0.149063
225449	2013-10-31 07:00:00	0.182	13.0	56.0	0.0399	L3	3	7	2	0.257514

225450 rows × 10 columns

In [63]:
```
 # variance of type 0, 1, 2
var_type0_high = []
var_type1_high = []
var_type2_high = []
var_type0_low = []
var_type1_low = []
var_type2_low = []
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 0) & (df_all_con_long['Price'] == 0.6720)]
    var_type0_high.append(df_hour_con_long['Consumption'].std())
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 0) & (df_all_con_long['Price'] == 0.0399)]
    var_type0_low.append(df_hour_con_long['Consumption'].std())
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 1) & (df_all_con_long['Price'] == 0.6720)]
    var_type1_high.append(df_hour_con_long['Consumption'].std())
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 1) & (df_all_con_long['Price'] == 0.0399)]
    var_type1_low.append(df_hour_con_long['Consumption'].std())
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 2) & (df_all_con_long['Price'] == 0.6720)]
    var_type2_high.append(df_hour_con_long['Consumption'].std())
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 2) & (df_all_con_long['Price'] == 0.0399)]
    var_type2_low.append(df_hour_con_long['Consumption'].std())
```

Now we get the standard deviation of all type housholds in different pricing over 24 hours.

In [65]:
```
# non-event data
# Type 0 household
# build a new dataframe with columns: GMT, user ids in the group
df_type0_nf = df_tou1h_nf_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[0]].
House)]
df_type0_nf = pd.merge(df_type0_nf, df_wea1h_nf_warm, on = 'GMT') # add weather
df_type0_nf['Price'] = 0.1176
df_type0_nf['Day of week'] = pd.to_datetime(df_type0_nf.GMT).dt.dayofweek # add day of week
df_type0_nf['Hour of day'] = pd.to_datetime(df_type0_nf.GMT).dt.hour # add hour of day
df_type0_nf['Household type'] = 0

# Type 1 household
# build a new dataframe with columns: GMT, user ids in the group
df_type1_nf = df_tou1h_nf_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[1]].
House)]
df_type1_nf = pd.merge(df_type1_nf, df_wea1h_nf_warm, on = 'GMT') # add weather
df_type1_nf['Price'] = 0.1176
df_type1_nf['Day of week'] = pd.to_datetime(df_type1_nf.GMT).dt.dayofweek # add day of week
df_type1_nf['Hour of day'] = pd.to_datetime(df_type1_nf.GMT).dt.hour # add hour of day
df_type1_nf['Household type'] = 1

# Type 2 household
# build a new dataframe with columns: GMT, user ids in the group
df_type2_nf = df_tou1h_nf_warm[['GMT'] + list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[2]].
House)]
df_type2_nf = pd.merge(df_type2_nf, df_wea1h_nf_warm, on = 'GMT') # add weather
df_type2_nf['Price'] = 0.1176
df_type2_nf['Day of week'] = pd.to_datetime(df_type2_nf.GMT).dt.dayofweek # add day of week
df_type2_nf['Hour of day'] = pd.to_datetime(df_type2_nf.GMT).dt.hour # add hour of day
df_type2_nf['Household type'] = 2
```

In [66]:
```
# add dummy column to use code in the next cell
# type 0 event time consumption prediction
df_type0_nf['Predicted consumption'] = 0
# price responsiveness elementwise
df_type0_con_nf = df_type0_nf.copy() # store the consumption not the response, just for this additional test
# df_type0_con.iloc[:,1:houseGroupDf.counts.iloc[0] + 1] = df_type0.iloc[:,1:houseGroupDf.counts.iloc[0] +
 1].sub(df_type0['Predicted consumption'], axis = 0)

# type 1 event time consumption prediction
df_type1_nf['Predicted consumption'] = 0
# price responsiveness elementwise
df_type1_con_nf = df_type1_nf.copy()
# df_type1_nf_con.iloc[:,1:houseGroupDf.counts.iloc[1] + 1] = df_type1_nf.iloc[:,1:houseGroupDf.counts.iloc
[1] + 1].sub(df_type1_nf['Predicted consumption'], axis = 0)

# type 2 event time consumption prediction
df_type2_nf['Predicted consumption'] = 0
# price responsiveness elementwise
df_type2_con_nf = df_type2_nf.copy()
# df_type2_con.iloc[:,1:houseGroupDf.counts.iloc[2] + 1] = df_type2.iloc[:,1:houseGroupDf.counts.iloc[2] +
 1].sub(df_type2['Predicted consumption'], axis = 0)
```

In [67]:
```
# select each user ID and form a seperate dataframe
# first change all the user ID to "Consumption"
new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[0]):
    new_col.append('Consumption')
new_col = new_col + list(df_type0_con_nf.columns)[-7:] # the new column names
df_type0_con_nf.columns = new_col
df_all_con_long = pd.DataFrame()
for i in range(1, houseGroupDf.counts.iloc[0] + 1):
    df_all_con_long = df_all_con_long.append(df_type0_con_nf.iloc[:,[0,i,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[1]): # for renaming the second type
    new_col.append('Consumption')
new_col = new_col + list(df_type1_con_nf.columns)[-7:] # the new column names
df_type1_con_nf.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[1] + 1):
    df_all_con_long = df_all_con_long.append(df_type1_con_nf.iloc[:,[0,i,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

new_col = ['GMT']
for i in range(0, houseGroupDf.counts.iloc[2]): # for renaming the third type
    new_col.append('Consumption')
new_col = new_col + list(df_type2_con_nf.columns)[-7:] # the new column names
df_type2_con_nf.columns = new_col
for i in range(1, houseGroupDf.counts.iloc[2] + 1):
    df_all_con_long = df_all_con_long.append(df_type2_con_nf.iloc[:,[0,i,-7,-6,-5,-4,-3,-2,-1]], ignore_index
= True)

# for clearer visulization, round the temperatures
df_all_con_long['TempC'] = df_all_con_long.TempC.round(0)
df_all_con_long['TempF'] = df_all_con_long.TempF.round(0)
df_all_con_long # ready for the price responsiveness analysis
```

Out[67]:

	GMT	Consumption	TempC	TempF	Price	Day of week	Hour of day	Household type	Predicted consumption
0	2013-04-01 00:00:00	0.110	-3.0	27.0	0.1176	0	0	0	0
1	2013-04-01 01:00:00	0.063	-3.0	27.0	0.1176	0	1	0	0
2	2013-04-01 02:00:00	0.115	-3.0	27.0	0.1176	0	2	0	0
3	2013-04-01 03:00:00	0.204	-3.0	27.0	0.1176	0	3	0	0
4	2013-04-01 04:00:00	0.124	-3.0	27.0	0.1176	0	4	0	0
5	2013-04-01 05:00:00	0.084	-2.0	28.0	0.1176	0	5	0	0
6	2013-04-01 06:00:00	0.062	-2.0	28.0	0.1176	0	6	0	0
7	2013-04-01 07:00:00	0.109	0.0	32.0	0.1176	0	7	0	0
8	2013-04-01 08:00:00	0.062	2.0	35.0	0.1176	0	8	0	0
9	2013-04-01 09:00:00	0.110	4.0	39.0	0.1176	0	9	0	0
10	2013-04-01 10:00:00	0.061	5.0	40.0	0.1176	0	10	0	0
11	2013-04-01 11:00:00	0.071	5.0	42.0	0.1176	0	11	0	0
12	2013-04-01 12:00:00	0.115	6.0	43.0	0.1176	0	12	0	0
13	2013-04-01 13:00:00	0.066	6.0	43.0	0.1176	0	13	0	0
14	2013-04-01 14:00:00	0.110	6.0	42.0	0.1176	0	14	0	0
15	2013-04-01 15:00:00	0.070	6.0	42.0	0.1176	0	15	0	0
16	2013-04-01 16:00:00	0.372	5.0	40.0	0.1176	0	16	0	0
17	2013-04-01 17:00:00	0.515	3.0	38.0	0.1176	0	17	0	0
18	2013-04-01 18:00:00	0.208	2.0	36.0	0.1176	0	18	0	0
19	2013-04-01 19:00:00	0.242	1.0	34.0	0.1176	0	19	0	0
20	2013-04-01 20:00:00	0.151	0.0	33.0	0.1176	0	20	0	0
21	2013-04-01 21:00:00	0.284	-1.0	31.0	0.1176	0	21	0	0
22	2013-04-01 22:00:00	0.081	-1.0	30.0	0.1176	0	22	0	0
23	2013-04-01 23:00:00	0.122	-2.0	30.0	0.1176	0	23	0	0
24	2013-04-02 00:00:00	0.062	-2.0	29.0	0.1176	1	0	0	0
25	2013-04-02 01:00:00	0.065	-2.0	29.0	0.1176	1	1	0	0
26	2013-04-02 02:00:00	0.121	-1.0	29.0	0.1176	1	2	0	0
27	2013-04-02 03:00:00	0.065	-1.0	29.0	0.1176	1	3	0	0
28	2013-04-02 04:00:00	0.301	-1.0	29.0	0.1176	1	4	0	0

	GMT	Consumption	TempC	TempF	Price	Day of week	Hour of day	Household type	Predicted consumption
29	2013-04-02 05:00:00	0.107	-2.0	28.0	0.1176	1	5	0	0
...	...	...	...	...	...	...	...	...	...
1501170	2013-10-29 18:00:00	0.568	9.0	49.0	0.1176	1	18	2	0
1501171	2013-10-29 19:00:00	0.282	9.0	49.0	0.1176	1	19	2	0
1501172	2013-10-29 20:00:00	0.287	9.0	49.0	0.1176	1	20	2	0
1501173	2013-10-29 21:00:00	0.240	9.0	49.0	0.1176	1	21	2	0
1501174	2013-10-29 22:00:00	0.320	9.0	49.0	0.1176	1	22	2	0
1501175	2013-10-29 23:00:00	0.289	10.0	50.0	0.1176	1	23	2	0
1501176	2013-10-30 00:00:00	0.208	10.0	50.0	0.1176	2	0	2	0
1501177	2013-10-30 01:00:00	0.176	10.0	50.0	0.1176	2	1	2	0
1501178	2013-10-30 02:00:00	0.092	10.0	50.0	0.1176	2	2	2	0
1501179	2013-10-30 03:00:00	0.103	10.0	50.0	0.1176	2	3	2	0
1501180	2013-10-30 04:00:00	0.164	10.0	50.0	0.1176	2	4	2	0
1501181	2013-10-30 05:00:00	0.087	10.0	50.0	0.1176	2	5	2	0
1501182	2013-10-30 06:00:00	0.110	10.0	50.0	0.1176	2	6	2	0
1501183	2013-10-30 07:00:00	0.273	10.0	50.0	0.1176	2	7	2	0
1501184	2013-10-30 08:00:00	0.353	10.0	50.0	0.1176	2	8	2	0
1501185	2013-10-30 09:00:00	0.226	10.0	50.0	0.1176	2	9	2	0
1501186	2013-10-30 10:00:00	0.148	10.0	50.0	0.1176	2	10	2	0
1501187	2013-10-30 11:00:00	0.099	10.0	50.0	0.1176	2	11	2	0
1501188	2013-10-30 12:00:00	0.144	10.0	50.0	0.1176	2	12	2	0
1501189	2013-10-30 13:00:00	0.128	10.0	50.0	0.1176	2	13	2	0
1501190	2013-10-30 14:00:00	0.102	10.0	50.0	0.1176	2	14	2	0
1501191	2013-10-30 15:00:00	0.179	10.0	50.0	0.1176	2	15	2	0
1501192	2013-10-30 16:00:00	0.171	10.0	50.0	0.1176	2	16	2	0
1501193	2013-10-30 17:00:00	1.060	10.0	50.0	0.1176	2	17	2	0
1501194	2013-10-30 18:00:00	0.533	10.0	50.0	0.1176	2	18	2	0
1501195	2013-10-30 19:00:00	0.243	10.0	50.0	0.1176	2	19	2	0
1501196	2013-10-30 20:00:00	0.294	10.0	50.0	0.1176	2	20	2	0
1501197	2013-10-30 21:00:00	0.279	10.0	50.0	0.1176	2	21	2	0

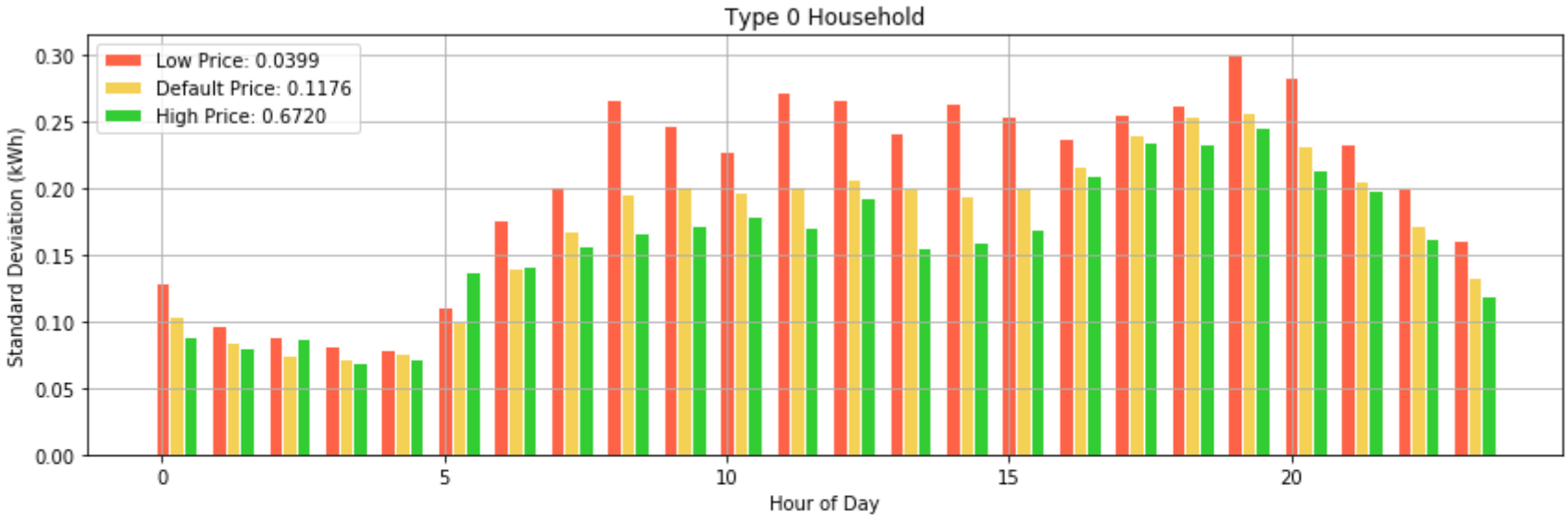
	GMT	Consumption	TempC	TempF	Price	Day of week	Hour of day	Household type	Predicted consumption
1501198	2013-10-30 22:00:00	0.257	11.0	52.0	0.1176	2	22	2	0
1501199	2013-10-30 23:00:00	0.355	12.0	54.0	0.1176	2	23	2	0

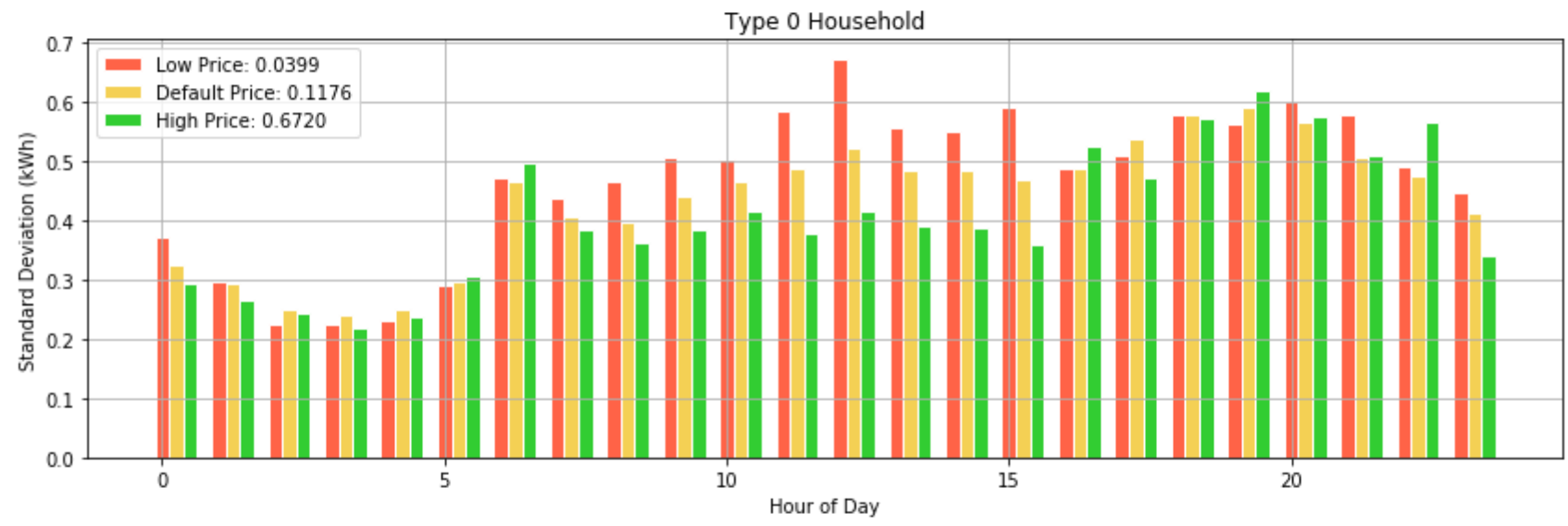
1501200 rows × 9 columns

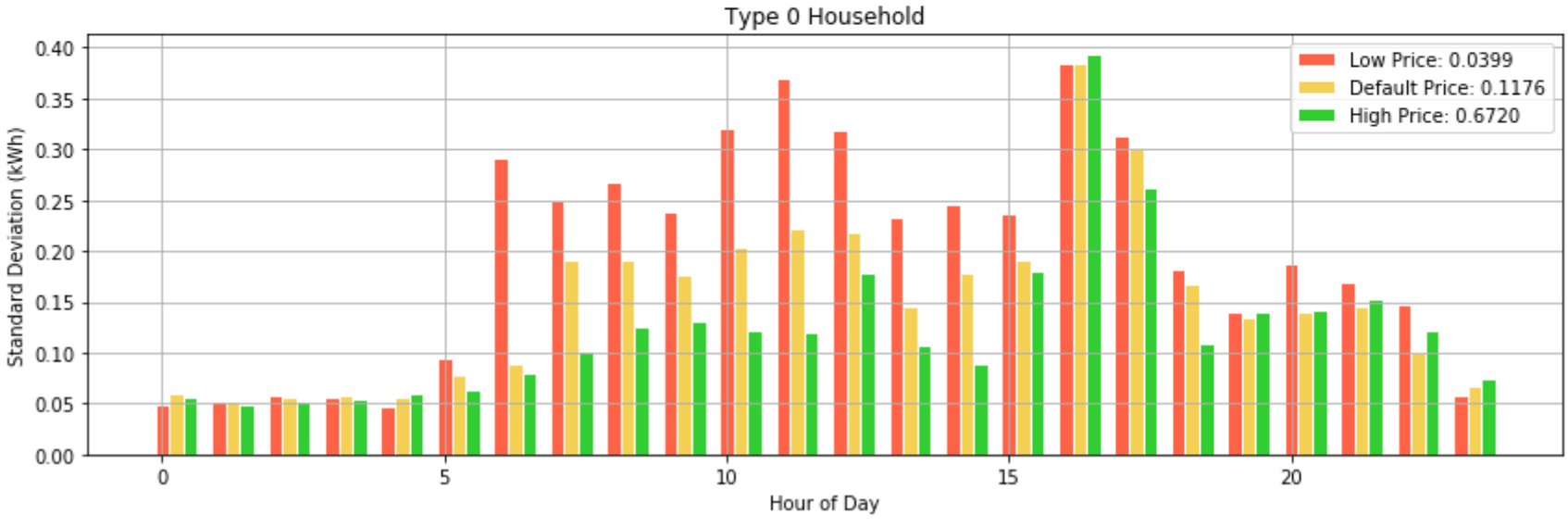
In [68]:
```
 # variance of type 0, 1, 2
var_type0_default = []
var_type1_default = []
var_type2_default = []
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 0) & (df_all_con_long['Price'] == 0.1176)]
    var_type0_default.append(df_hour_con_long['Consumption'].std())
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 1) & (df_all_con_long['Price'] == 0.1176)]
    var_type1_default.append(df_hour_con_long['Consumption'].std())
for i in range(24):
    df_hour_con_long = df_all_con_long[(df_all_con_long['Hour of day'] == i) & (df_all_con_long['Household ty
pe'] == 2) & (df_all_con_long['Price'] == 0.1176)]
    var_type2_default.append(df_hour_con_long['Consumption'].std())
```

In [ ]:
```
# plot the grouped bar charts based on price, for 3 household types
fig_all = plt.figure(figsize = (12,12))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.1176, 0.6720]
# set width of bar
barWidth = 0.25
# Set position of bar on X axis
r1 = np.arange(len(var_type0_high))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(3, 1, (g + 1)))
    # Make the plot
    if g == 0:
        ax_Ntou[-1].bar(r1, var_type0_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type0_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type0_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    if g == 1:
        ax_Ntou[-1].bar(r1, var_type1_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type1_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type1_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    if g == 2:
        ax_Ntou[-1].bar(r1, var_type2_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type2_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type2_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    ax_Ntou[-1].grid()
    ax_Ntou[-1].set_title('Responsiveness'  + ' (Type ' + str(g) + ')' )
    ax_Ntou[-1].set_xlabel('Hour of Day')
    ax_Ntou[-1].set_ylabel('Standard Deviation (kWh)')
    ax_Ntou[-1].legend()
plt.tight_layout()
```

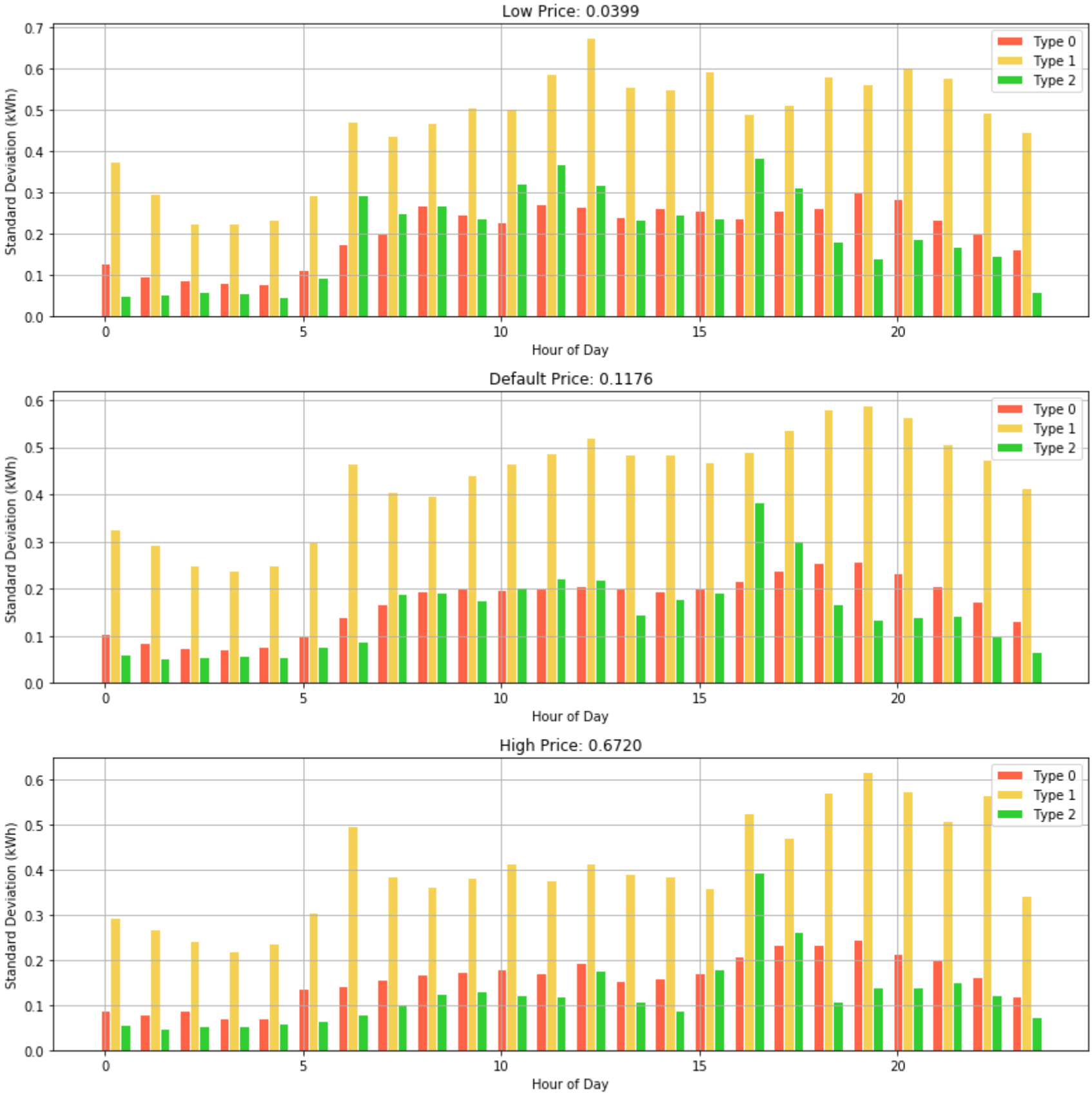
In [69]:
```
# plot the grouped bar charts based on price, for 3 household types
fig_all = plt.figure(figsize = (12,12))
ax_N'tou = [] # store subplot objects
prices = [0.0399, 0.1176, 0.6720]
# set width of bar
barWidth = 0.25
# Set position of bar on X axis
r1 = np.arange(len(var_type0_high))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(3, 1, (g + 1)))
    # Make the plot
    if g == 0:
        ax_Ntou[-1].bar(r1, var_type0_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type0_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type0_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    if g == 1:
        ax_Ntou[-1].bar(r1, var_type1_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type1_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type1_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    if g == 2:
        ax_Ntou[-1].bar(r1, var_type2_low, color='tomato', width=barWidth, edgecolor='white', label='Low Pric
e: 0.0399')
        ax_Ntou[-1].bar(r2, var_type2_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Default Price: 0.1176')
        ax_Ntou[-1].bar(r3, var_type2_high, color='limegreen', width=barWidth, edgecolor='white', label='High
Price: 0.6720')
        ax_Ntou[-1].set_title('Type ' + str(g) + ' Household')
    ax_Ntou[-1].grid()
    ax_Ntou[-1].set_title('Responsiveness'  + ' (Type ' + str(g) + ')' )
    ax_Ntou[-1].set_xlabel('Hour of Day')
    ax_Ntou[-1].set_ylabel('Standard Deviation (kWh)')
    ax_Ntou[-1].legend()
plt.tight_layout()
```


Type 0 Household
Low Price: 0.0399
Default Price: 0.1176
High Price: 0.6720
Standard Deviation (kWh)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
5
10
15
20
Hour of Day


Type 0 Household
Low Price: 0.0399
Default Price: 0.1176
High Price: 0.6720
Standard Deviation (kWh)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
Hour of Day


Type 0 Household
Low Price: 0.0399
Default Price: 0.1176
High Price: 0.6720
Standard Deviation (kWh)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
5
10
15
20
Hour of Day

In [70]:
```
# plot the grouped bar charts based on price, for 3 household types
fig_all = plt.figure(figsize = (12,12))
ax_Ntou = [] # store subplot objects
prices = [0.0399, 0.1176, 0.6720]
# set width of bar
barWidth = 0.25
# Set position of bar on X axis
r1 = np.arange(len(var_type0_high))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
for g in range(3): # three types
    ax_Ntou.append(fig_all.add_subplot(3, 1, (g + 1)))
    # Make the plot
    if g == 0:
        ax_Ntou[-1].bar(r1, var_type0_low, color='tomato', width=barWidth, edgecolor='white', label='Type 0')
        ax_Ntou[-1].bar(r2, var_type1_low, color='xkcd:maize', width=barWidth, edgecolor='white', label='Type
1')
        ax_Ntou[-1].bar(r3, var_type2_low, color='limegreen', width=barWidth, edgecolor='white', label='Type
 2')
        ax_Ntou[-1].set_title('Low Price: 0.0399')
    if g == 1:
        ax_Ntou[-1].bar(r1, var_type0_default, color='tomato', width=barWidth, edgecolor='white', label='Type
0')
        ax_Ntou[-1].bar(r2, var_type1_default, color='xkcd:maize', width=barWidth, edgecolor='white', label=
'Type 1')
        ax_Ntou[-1].bar(r3, var_type2_default, color='limegreen', width=barWidth, edgecolor='white', label='T
ype 2')
        ax_Ntou[-1].set_title('Default Price: 0.1176')
    if g == 2:
        ax_Ntou[-1].bar(r1, var_type0_high, color='tomato', width=barWidth, edgecolor='white', label='Type 0'
)
        ax_Ntou[-1].bar(r2, var_type1_high, color='xkcd:maize', width=barWidth, edgecolor='white', label='Typ
e 1')
        ax_Ntou[-1].bar(r3, var_type2_high, color='limegreen', width=barWidth, edgecolor='white', label='Type
2')
        ax_Ntou[-1].set_title('High Price: 0.6720')
    ax_Ntou[-1].grid()
    ax_Ntou[-1].set_xlabel('Hour of Day')
    ax_Ntou[-1].set_ylabel('Standard Deviation (kWh)')
    ax_Ntou[-1].legend()
plt.tight_layout()
```



Compare the overlap of household types between two seasons

In [14]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
n_house = 1025 # total household numbers
houseLabel = [] # store the 24hour group labels for each hourshold
for i in range(1025):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_tou1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and fill in null value with mean
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
    else:
        x = df_tou1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.fillna(x.mean()).transpose() # data for the specific hour, and filtering out null value
        # PCA without standardization
        pca_nstd = PCA()
        principleComponents_nstd = pca_nstd.fit_transform(x)
        principleDf_nstd = pd.DataFrame(data = principleComponents_nstd)
        kmeans = KMeans(n_clusters = n, init='k-means++').fit(principleDf_nstd.iloc[:,:3])
        cluster_dict = {}
        houseLabel = np.core.defchararray.add(houseLabel, kmeans.labels_.astype('str'))
# We obtain the house Label shown below
# print(houseLabel)

# Group the house based on their 24 hour labels
houseLableDf = pd.DataFrame()
houseLableDf['House'] = df_tou1h_nf_cold.columns[1:]
houseLableDf['Label'] = houseLabel
houseGroupDf = houseLableDf.groupby('Label').size().reset_index(name='counts')
houseGroupDf = houseGroupDf.sort_values(by=['counts'], ascending=False)
houseGroupDf
```

Out[14]:

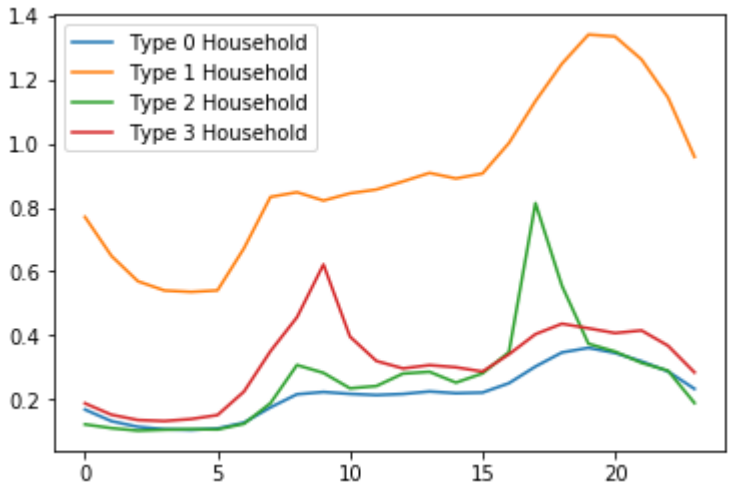
	Label	counts
37	011000000001000011000201	411
297	100211121112122100122020	29
27	011000000001000010000201	10
97	011000000101000011000201	8
43	011000000001000011020201	7
160	011000001112122100122020	7
49	011000000001000011100201	6
319	101000000001000011000201	5
435	222122212220211222211112	5
287	100211101112122100122020	5
190	011000100001000011000201	5
28	011000000001000010100201	4
52	011000000001000011120201	4
208	011001000001000011000201	4
62	011000000001000111000201	4
147	011000001101000011000201	4
42	011000000001000011002201	4
360	111000000001000011000201	4
384	111000000112122100122020	4
318	101000000001000011000200	4
337	101000000112122100122020	3
46	011000000001000011022201	3
140	011000001001000011000201	3
247	100000021112122100122020	3
119	011000000112000011000201	3
85	011000000011000011000201	3
76	011000000001120011000201	3
61	011000000001000101000201	3
111	011000000111000011000201	3
400	111000021112122100122020	3
...	...	...
138	011000000211000011000201	1
137	011000000120211222211121	1
136	011000000120122111000200	1
135	011000000112122200122020	1
134	011000000112122111000201	1
162	011000001112122100122201	1
163	011000001211000011000201	1
164	011000001220211222212020	1
178	011000021112120100122001	1
189	011000022101022011000201	1
188	011000022101000011000201	1
187	011000021222111222122201	1
186	011000021112122110122001	1
185	011000021112122102122201	1
184	011000021112122100122021	1
183	011000021112122100122020	1
180	011000021112122100112020	1
179	011000021112122000100201	1
177	011000021101122100122021	1

	Label	counts
165	01100001222021122222200 1	1
176	01100002110102210000020 1	1
175	01100002110102201012020 1	1
174	01100002110100001100020 1	1
173	01100002100110001100002 1	1
171	01100002011200010012202 1	1
170	01100002000102001100020 1	1
169	01100002000100010012200 1	1
168	01100002000100001102200 1	1
166	01100002000100001000020 1	1
444	22221212122021122221102 0	1

445 rows × 2 columns

```
<Figure size 936x6480 with 0 Axes>
```

In [15]:
```
for i in range(4):
    load = []
    for j in range(24):
        if j <= 9:
            x = df_tou1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains('0' + str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
        else:
            x = df_tou1h_nf_cold[df_tou1h_nf_cold.GMT.str.contains(str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.fillna(x.mean())
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
    plt.plot(load, label = 'Type ' + str(i) + ' Household')
plt.legend()
plt.show()
```



In [16]:
```
# Store warm season houshold labels
house_list_type0_warm = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[0]].House)
house_list_type1_warm = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[1]].House)
house_list_type2_warm = list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[2]].House)
```

In [17]:
```
len(house_list_type0)
```

Out[17]: 314

In [18]:
```
len(house_list_type0_warm)
```

Out[18]: 411

In [20]:
```
len(set(house_list_type0) & set(house_list_type0_warm))
```

Out[20]: 292

In [21]:
```
292/314*100
```

Out[21]: 92.99363057324841

In [22]:
```
292/411*100
```

Out[22]: 71.04622871046229

In [23]:
```
len(set(house_list_type1) & set(house_list_type1_warm))
```

Out[23]: 5

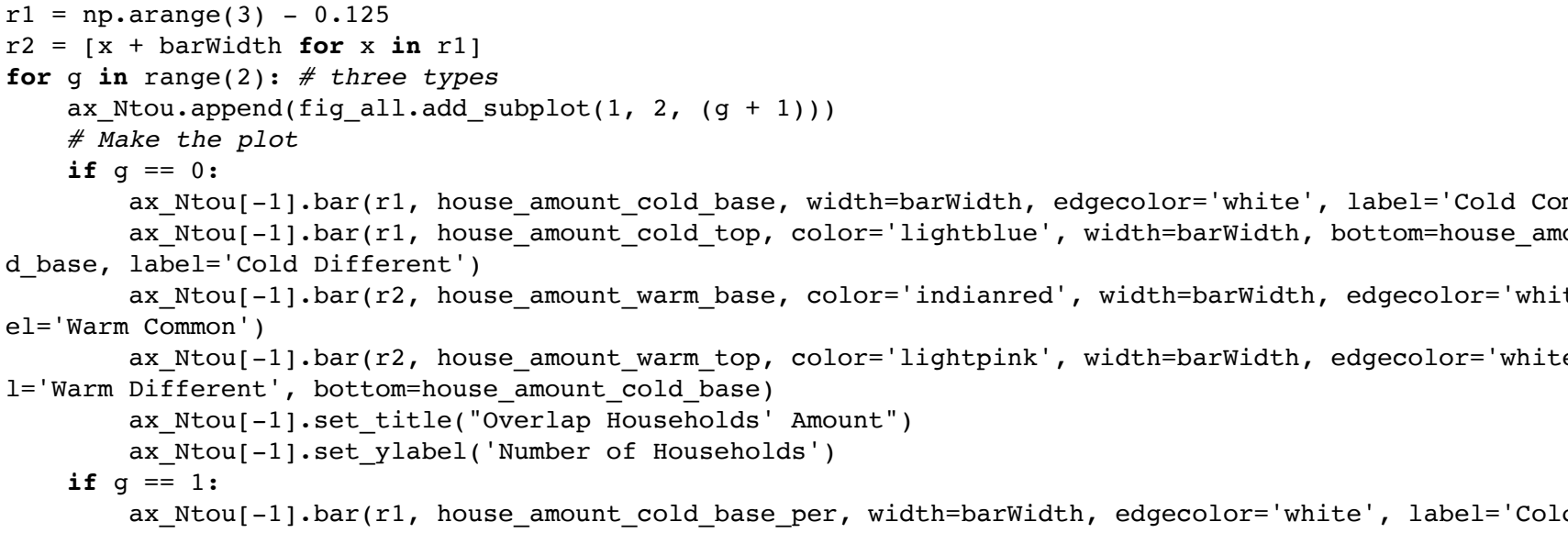
In [24]:
```
len(house_list_type1)
```

Out[24]: 19

In [25]:
```
len(house_list_type1_warm)
```

Out[25]: 29

In [55]:
```
# plot the grouped bar charts based on price, for 3 household types
house_list_common_type0 = set(house_list_type0) & set(house_list_type0_warm)
house_list_common_type1 = set(house_list_type1) & set(house_list_type1_warm)
house_list_common_type2 = set(house_list_type2) & set(house_list_type2_warm)
house_amount_cold_base = [len(house_list_common_type0), len(house_list_common_type1), len(house_list_common_t
ype2)] # type0, type1, type2
house_amount_cold_top = [len(house_list_type0)-len(house_list_common_type0), len(house_list_type1)-len(house_
list_common_type1), len(house_list_type2)-len(house_list_common_type2)] # type0, type1, type2
house_amount_warm_base = [len(house_list_common_type0), len(house_list_common_type1), len(house_list_common_t
ype2)] # type0, type1, type2
house_amount_warm_top = [len(house_list_type0_warm)-len(house_list_common_type0), len(house_list_type1_warm)-
len(house_list_common_type1), len(house_list_type2_warm)-len(house_list_common_type2)] # type0, type1, type2
# percentage
house_amount_cold_base_per = [len(house_list_common_type0)/len(house_list_type0)*100, len(house_list_common_t
ype1)/len(house_list_type1)*100, len(house_list_common_type2)/len(house_list_type2)*100] # type0, type1, type
2
house_amount_cold_top_per = [(len(house_list_type0)-len(house_list_common_type0))/len(house_list_type0)*100,
(len(house_list_type1)-len(house_list_common_type1))/len(house_list_type1)*100, (len(house_list_type2)-len(ho
use_list_common_type2))/len(house_list_type2)*100] # type0, type1, type2
house_amount_warm_base_per = [len(house_list_common_type0)/len(house_list_type0_warm)*100, len(house_list_com
mon_type1)/len(house_list_type1_warm)*100, len(house_list_common_type2)/len(house_list_type2_warm)*100] # typ
e0, type1, type2
house_amount_warm_top_per = [(len(house_list_type0_warm)-len(house_list_common_type0))/len(house_list_type0_w
arm)*100, (len(house_list_type1_warm)-len(house_list_common_type1))/len(house_list_type1_warm)*100, (len(hous
e_list_type2_warm)-len(house_list_common_type2))/len(house_list_type2_warm)*100] # type0, type1, type2
fig_all = plt.figure(figsize = (12,4))
ax_Ntou = [] # store subplot objects
# set width of bar
barWidth = 0.25
# Set position of bar on X axis
r1 = np.arange(3) - 0.125
r2 = [x + barWidth for x in r1]
for g in range(2): # three types
    ax_Ntou.append(fig_all.add_subplot(1, 2, (g + 1)))
    # Make the plot
    if g == 0:
        ax_Ntou[-1].bar(r1, house_amount_cold_base, width=barWidth, edgecolor='white', label='Cold Common')
        ax_Ntou[-1].bar(r1, house_amount_cold_top, color='lightblue', width=barWidth, bottom=house_amount_col
d_base, label='Cold Different')
        ax_Ntou[-1].bar(r2, house_amount_warm_base, color='indianred', width=barWidth, edgecolor='white', lab
el='Warm Common')
        ax_Ntou[-1].bar(r2, house_amount_warm_top, color='lightpink', width=barWidth, edgecolor='white', labe
l='Warm Different', bottom=house_amount_cold_base)
        ax_Ntou[-1].set_title("Overlap Households' Amount")
        ax_Ntou[-1].set_ylabel('Number of Households')
    if g == 1:
        ax_Ntou[-1].bar(r1, house_amount_cold_base_per, width=barWidth, edgecolor='white', label='Cold Commo
n')
        ax_Ntou[-1].bar(r1, house_amount_cold_top_per, color='lightblue', width=barWidth, bottom=house_amount
_cold_base_per, label='Cold Different')
        ax_Ntou[-1].bar(r2, house_amount_warm_base_per, color='indianred', width=barWidth, edgecolor='white',
label='Warm Common')
        ax_Ntou[-1].bar(r2, house_amount_warm_top_per, color='lightpink', width=barWidth, edgecolor='white',
label='Warm Different', bottom=house_amount_warm_base_per)
        ax_Ntou[-1].set_title("Overlap Households' Percentage")
        ax_Ntou[-1].set_ylabel('Percentage (%)')
    ax_Ntou[-1].grid()
    ax_Ntou[-1].set_xticks([0,1,2])
    ax_Ntou[-1].set_xlabel('Household Type')
    ax_Ntou[-1].legend()
plt.tight_layout()
```



The above figure tells us, if we use cold season to do principle components clustering and use the clustered households in the warm season, we will have some households different from those clustered based on warm season data. Such difference percentage increases along type 0, 1, 2. For type 0, the overlap is actually more than 70% in both clustered type 0 using different season data, which could mean that if we want to decide the right consumer groups to target, and we could conduct experiment for the cold season, and use the group for the warm seaon in order save some experiment cost and computation time, if the error of mis clustering is acceptable.