Preference Learning - Part V (non-TOU)

1. Principle Component Analysis (PCA)

2. K-mean clustering with/without normalization

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
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)]
```

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)]
```

In [7]:
```
# due to the null values of Ntou data, even if we use mean value to replace the null, it happens that some us
ers' consumption in a time period of days are always null, which leads to null average value, to get rid of t
he problem, we delete these customers.
df_Ntou1h_nf_cold = df_Ntou1h_nf_cold.dropna(axis=1)
df_Ntou1h_nf_warm = df_Ntou1h_nf_warm.dropna(axis=1)
```

In [8]:
```
df_Ntou1h_nf_warm
```

Out[8]:

	GMT	N0000	N0002	N0003	N0009	N0010	N0011	N0013	N0014	N0016	...	N4153	N4154	N4162	N4165	N4166	N4167	N4
2160	2013-04-01 00:00:00	0.334	0.030	0.087	0.664	6.788	0.036	0.482	1.215	0.019	...	0.373	0.104	0.071	0.055	0.037	0.451	0.1
2161	2013-04-01 01:00:00	0.353	0.029	0.072	0.563	0.544	0.036	0.389	1.129	0.010	...	0.147	0.102	0.064	0.068	0.082	0.599	0.0
2162	2013-04-01 02:00:00	0.409	0.064	0.072	0.542	0.538	0.013	0.338	1.118	0.011	...	0.162	0.103	0.050	0.067	0.037	0.406	0.1
2163	2013-04-01 03:00:00	0.543	0.094	0.126	0.569	0.477	0.040	0.250	1.147	0.010	...	0.174	0.088	0.049	0.060	0.037	0.517	0.1
2164	2013-04-01 04:00:00	0.423	0.091	0.072	0.531	0.472	0.040	0.248	1.164	0.397	...	0.131	0.050	0.048	0.070	0.041	0.608	0.1
2165	2013-04-01 05:00:00	0.457	0.095	0.063	0.541	0.488	0.012	0.256	1.180	0.229	...	0.111	0.145	0.058	0.055	0.807	2.614	0.0
2166	2013-04-01 06:00:00	0.463	0.078	0.117	0.925	0.447	0.135	0.245	1.266	0.014	...	0.166	0.121	0.131	0.060	3.171	1.058	0.1
2167	2013-04-01 07:00:00	0.459	0.029	0.189	1.319	0.551	0.382	0.245	1.225	0.010	...	0.140	0.047	0.049	0.063	3.383	1.249	0.1
2168	2013-04-01 08:00:00	0.500	0.619	0.655	1.240	1.314	0.510	0.242	1.164	0.010	...	0.456	0.048	0.753	0.157	1.696	1.802	0.0
2169	2013-04-01 09:00:00	0.483	0.484	0.753	1.072	2.752	0.453	0.354	1.416	0.132	...	1.235	0.183	0.183	0.222	0.176	2.936	0.1
2170	2013-04-01 10:00:00	0.438	0.475	1.001	0.876	3.169	0.587	1.372	2.290	0.110	...	1.411	0.104	1.244	0.116	0.471	3.470	0.1
2171	2013-04-01 11:00:00	1.500	0.409	2.669	0.881	3.664	0.349	0.879	1.413	0.090	...	0.131	0.383	1.208	0.087	0.145	1.545	0.0
2172	2013-04-01 12:00:00	0.480	0.150	0.707	0.975	4.454	0.016	1.906	1.148	0.054	...	0.095	0.165	0.907	0.056	0.065	1.390	0.1
2173	2013-04-01 13:00:00	0.598	0.421	2.174	0.829	5.084	0.059	0.524	1.182	0.010	...	0.099	0.107	0.976	0.056	0.056	1.553	0.0
2174	2013-04-01 14:00:00	0.546	0.182	0.311	0.823	4.636	0.190	0.376	1.168	0.010	...	0.125	0.110	0.654	0.072	0.036	1.290	0.0
2175	2013-04-01 15:00:00	0.401	0.091	0.579	1.009	4.505	0.327	0.529	1.249	0.041	...	0.120	0.211	0.639	0.059	0.036	1.017	0.1
2176	2013-04-01 16:00:00	0.371	0.534	0.489	1.098	3.538	0.244	0.376	1.264	0.022	...	0.102	0.082	0.655	0.111	0.489	0.974	0.0
2177	2013-04-01 17:00:00	1.647	0.239	0.523	0.629	2.536	0.242	0.384	1.183	0.072	...	0.071	0.048	0.714	0.075	2.264	0.687	0.0
2178	2013-04-01 18:00:00	0.750	0.310	1.663	0.609	2.491	0.830	1.431	1.200	0.066	...	0.087	0.048	1.120	0.109	3.963	0.256	0.1
2179	2013-04-01 19:00:00	0.434	0.238	1.254	1.807	2.745	0.194	0.723	1.276	0.097	...	0.096	0.048	0.615	0.164	3.775	0.955	0.1
2180	2013-04-01 20:00:00	0.331	0.122	0.680	0.800	3.045	0.241	0.587	1.229	0.279	...	0.059	0.049	0.783	0.139	3.466	2.183	0.2

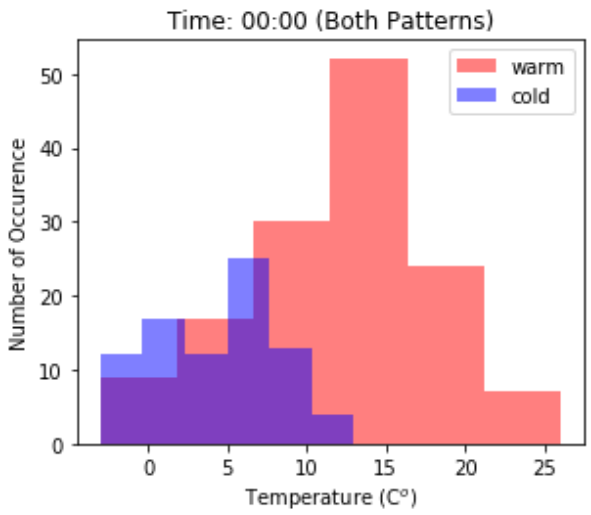
	GMT	N0000	N0002	N0003	N0009	N0010	N0011	N0013	N0014	N0016	...	N4153	N4154	N4162	N4165	N4166	N4167	N4
2181	2013-04-01 21:00:00	0.526	0.092	0.540	0.775	2.866	0.158	0.590	1.354	0.220	...	0.770	0.658	0.740	0.210	2.440	0.743	0.2
2182	2013-04-01 22:00:00	0.543	0.055	0.335	0.668	3.126	0.084	0.594	1.170	0.141	...	0.203	0.628	0.095	0.100	0.091	0.697	0.2
2183	2013-04-01 23:00:00	0.523	0.029	0.217	0.658	2.572	0.057	0.475	2.037	0.059	...	0.169	0.099	0.106	0.070	0.137	0.557	0.1
2184	2013-04-02 00:00:00	0.315	0.030	0.282	0.588	3.034	0.036	0.408	2.157	0.010	...	0.084	0.094	0.109	0.054	0.037	0.584	0.1
2185	2013-04-02 01:00:00	0.294	0.094	0.245	0.542	2.521	0.040	0.528	1.116	0.033	...	0.112	0.089	0.089	0.066	0.070	0.594	0.1
2186	2013-04-02 02:00:00	0.333	0.092	0.169	0.531	1.006	0.011	0.374	1.140	0.010	...	0.072	0.971	0.086	0.058	0.047	0.475	0.0
2187	2013-04-02 03:00:00	0.410	0.091	0.181	0.583	0.527	0.041	0.324	1.125	0.011	...	0.090	0.535	0.084	0.053	0.036	0.548	0.1
2188	2013-04-02 04:00:00	0.496	0.076	0.170	0.555	0.478	0.033	0.252	1.166	0.011	...	0.122	0.100	0.077	0.073	0.037	0.641	0.1
2189	2013-04-02 05:00:00	0.454	0.030	0.177	0.513	0.527	0.020	0.247	1.217	0.197	...	0.080	0.189	0.074	0.053	0.819	2.456	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7242	2013-10-29 18:00:00	0.225	0.653	0.683	0.712	0.754	0.167	0.576	0.553	0.086	...	0.101	0.079	0.336	0.144	0.014	0.415	0.3
7243	2013-10-29 19:00:00	0.384	0.514	1.296	0.745	2.880	0.872	0.574	0.667	0.253	...	0.108	0.108	0.507	0.266	0.015	0.766	0.4
7244	2013-10-29 20:00:00	0.219	0.336	1.168	0.645	4.793	0.184	0.547	0.628	0.309	...	0.074	0.154	0.324	0.221	0.061	1.515	0.3
7245	2013-10-29 21:00:00	0.216	0.345	0.651	0.611	4.406	0.140	0.711	0.648	0.325	...	0.540	0.158	0.545	0.202	0.014	2.711	0.2
7246	2013-10-29 22:00:00	0.294	0.097	0.471	0.635	4.246	0.150	0.631	0.430	0.160	...	0.415	0.154	0.397	0.208	0.014	1.543	0.2
7247	2013-10-29 23:00:00	0.336	0.036	0.185	0.613	3.382	0.240	0.578	1.344	0.175	...	0.260	0.153	0.079	0.109	0.061	0.405	0.2
7248	2013-10-30 00:00:00	0.186	0.125	0.175	0.610	1.124	0.038	0.563	0.449	0.092	...	0.208	0.124	0.097	0.116	0.014	0.195	0.2
7249	2013-10-30 01:00:00	0.200	0.163	0.091	0.599	1.233	0.037	0.545	0.474	0.011	...	0.173	0.079	0.079	0.099	0.014	0.069	0.1
7250	2013-10-30 02:00:00	0.170	0.100	0.142	0.579	1.136	0.039	0.520	0.467	0.010	...	0.196	0.082	0.109	0.100	0.059	0.143	0.1
7251	2013-10-30 03:00:00	0.188	0.037	0.098	0.584	0.473	0.028	0.476	0.516	0.048	...	0.181	0.086	0.067	0.107	0.016	0.079	0.2
7252	2013-10-30 04:00:00	0.367	0.106	0.097	0.543	0.421	0.021	0.477	0.395	0.088	...	0.182	0.094	0.090	0.088	0.015	0.119	0.1
7253	2013-10-30 05:00:00	0.359	0.074	0.113	0.585	0.294	0.039	0.391	0.469	0.010	...	0.187	0.079	0.066	0.109	0.046	0.110	0.2

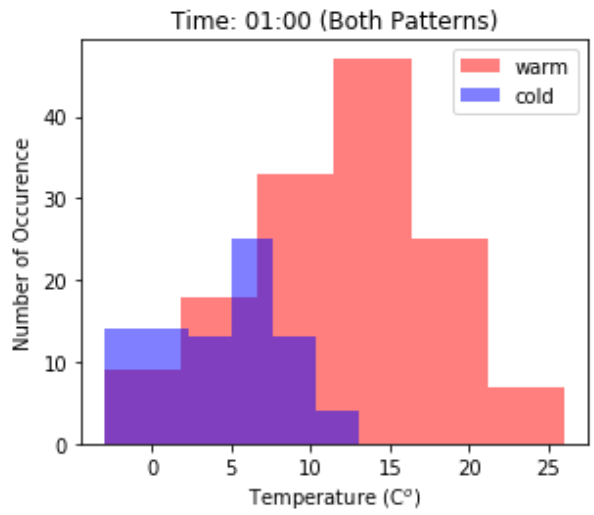
	GMT	N0000	N0002	N0003	N0009	N0010	N0011	N0013	N0014	N0016	...	N4153	N4154	N4162	N4165	N4166	N4167	N4
7254	2013-10-30 06:00:00	0.376	0.035	0.094	0.652	0.219	0.132	0.343	0.414	0.010	...	0.175	0.079	0.089	0.083	0.028	1.261	0.2
7255	2013-10-30 07:00:00	1.075	0.075	0.233	0.568	0.244	0.153	0.233	0.462	0.039	...	0.366	0.081	1.084	0.106	0.014	0.300	0.2
7256	2013-10-30 08:00:00	0.175	0.613	0.900	0.602	0.435	0.063	0.380	0.473	0.169	...	0.404	0.079	0.998	0.085	0.029	0.470	0.3
7257	2013-10-30 09:00:00	0.193	0.306	0.455	0.558	1.347	0.241	1.195	0.606	0.010	...	0.237	0.073	0.195	0.113	0.046	0.570	0.1
7258	2013-10-30 10:00:00	0.213	0.368	0.348	0.552	1.402	0.462	0.746	1.783	0.010	...	0.191	0.074	0.273	0.212	0.014	0.657	0.1
7259	2013-10-30 11:00:00	0.139	0.300	0.158	0.599	2.952	0.543	1.656	1.076	0.023	...	0.182	0.072	0.090	0.226	0.014	2.740	0.0
7260	2013-10-30 12:00:00	0.190	0.571	0.175	0.561	1.897	0.146	0.726	0.879	0.094	...	0.152	0.077	0.583	0.229	0.061	0.549	0.0
7261	2013-10-30 13:00:00	0.202	0.298	0.159	0.539	1.274	0.037	0.335	0.816	0.017	...	0.159	0.078	0.142	0.214	0.014	0.649	0.0
7262	2013-10-30 14:00:00	0.276	0.238	0.210	0.517	1.950	0.038	0.308	0.811	0.023	...	0.191	0.078	0.084	0.235	0.015	0.304	0.0
7263	2013-10-30 15:00:00	0.173	0.323	0.289	0.505	2.830	0.077	0.322	1.009	0.213	...	0.147	0.078	0.411	0.337	0.062	0.339	0.0
7264	2013-10-30 16:00:00	0.848	1.461	0.430	0.525	3.886	0.233	0.394	0.990	0.155	...	0.153	0.081	0.612	0.152	0.014	0.418	0.0
7265	2013-10-30 17:00:00	0.811	1.952	0.682	0.662	1.274	0.182	0.534	1.018	0.016	...	0.185	0.079	0.664	0.131	0.014	1.267	0.0
7266	2013-10-30 18:00:00	1.076	0.910	0.624	0.661	3.004	0.240	0.556	0.960	0.010	...	0.159	0.079	0.555	0.141	0.061	1.342	0.3
7267	2013-10-30 19:00:00	1.054	0.690	0.664	0.725	4.641	0.174	2.119	0.878	0.020	...	0.138	0.103	0.313	0.338	0.014	1.214	0.3
7268	2013-10-30 20:00:00	0.479	0.407	1.320	0.685	4.523	0.155	0.999	0.943	0.332	...	1.295	0.072	0.337	0.248	0.015	0.420	0.3
7269	2013-10-30 21:00:00	0.425	0.054	1.047	0.596	4.085	0.154	0.565	1.051	0.414	...	0.509	0.079	0.525	0.191	0.060	1.583	0.2
7270	2013-10-30 22:00:00	0.580	0.104	0.984	0.632	1.827	0.258	0.572	0.903	0.254	...	0.481	0.101	0.700	0.172	0.014	1.724	0.2
7271	2013-10-30 23:00:00	0.449	0.076	0.187	0.611	1.634	0.137	0.543	1.728	0.082	...	0.214	0.154	0.088	0.124	0.015	0.553	0.2

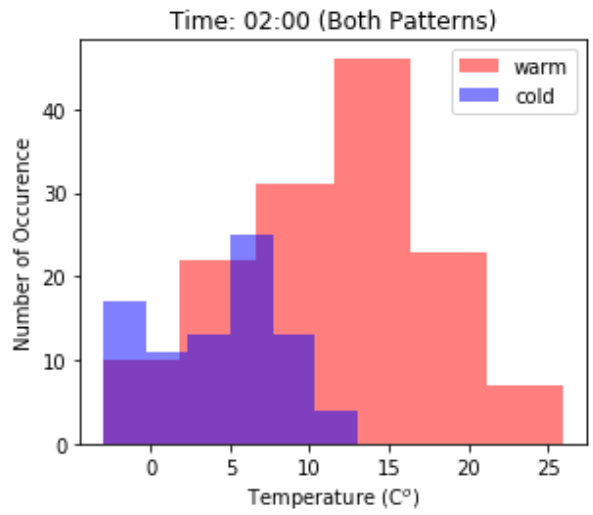
3336 rows × 2019 columns

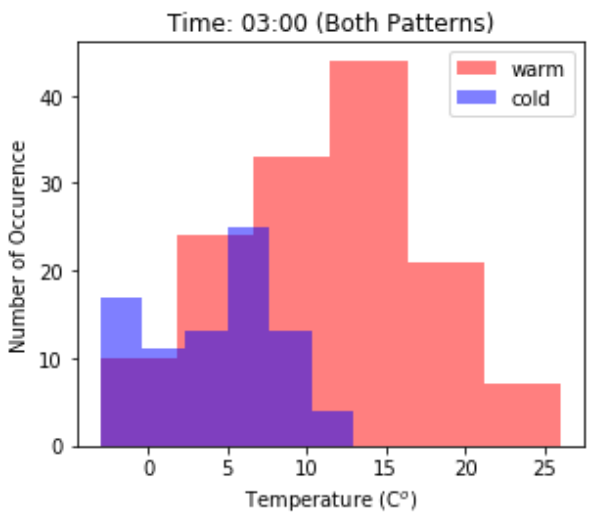
(0) Some basic temperature distribution of the two groups are given below

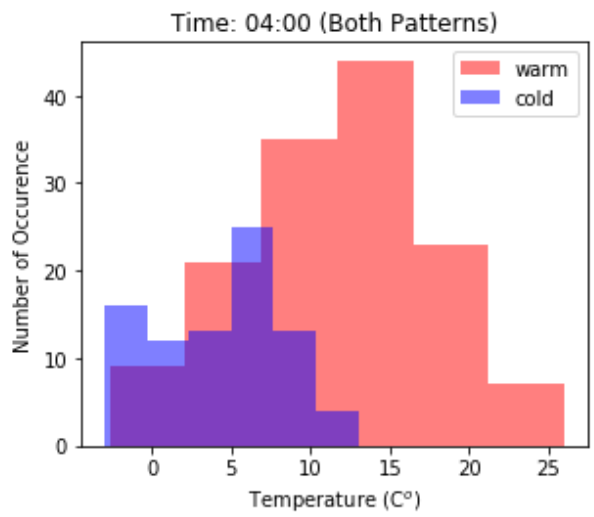
In [8]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.hist(df_wea1h_nf_warm[df_Ntou1h_nf_warm.GMT.str.contains('0' + str(i) + ':00:00')]['TempC'], bins=
6, color = 'red', alpha = 0.5, label = 'warm')
        ax.hist(df_wea1h_nf_cold[df_Ntou1h_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_Ntou1h_nf_warm.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=6, col
or = 'red', alpha = 0.5, label = 'warm')
        ax.hist(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=6, col
or = '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()
```

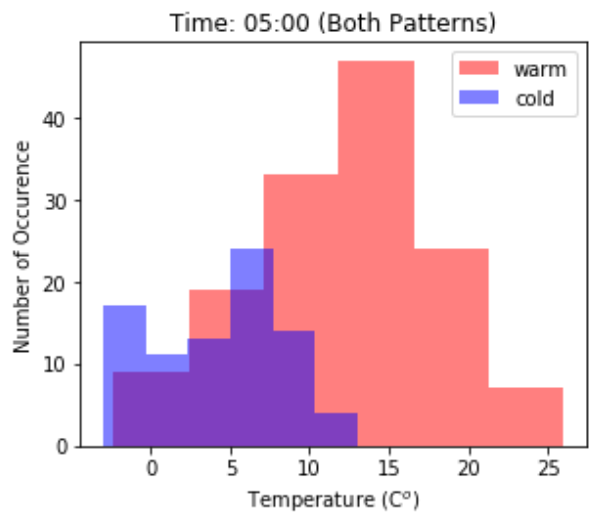

Time: 00:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

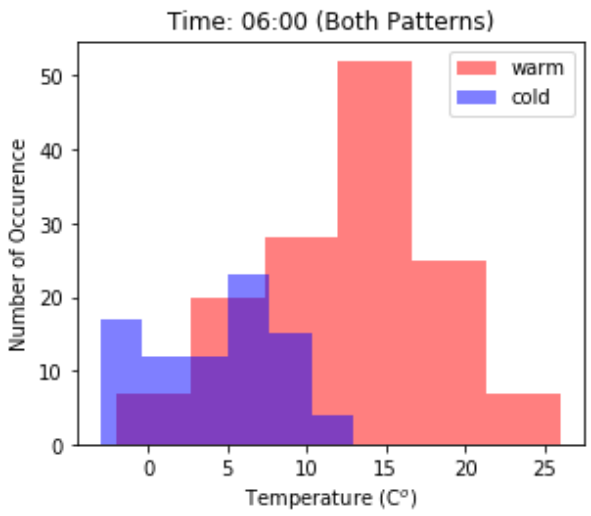

Time: 01:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

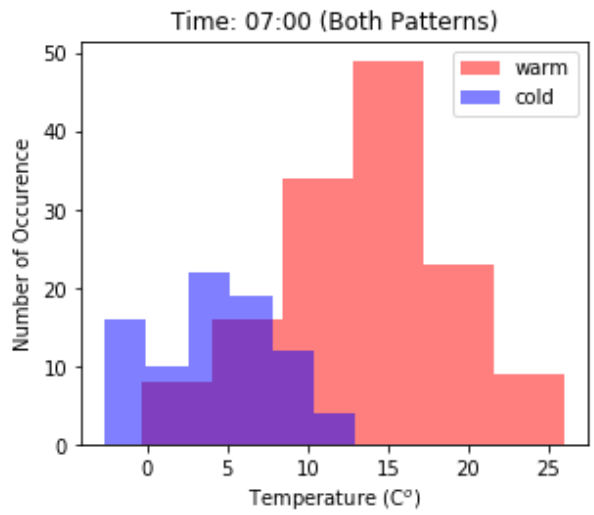

Time: 02:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

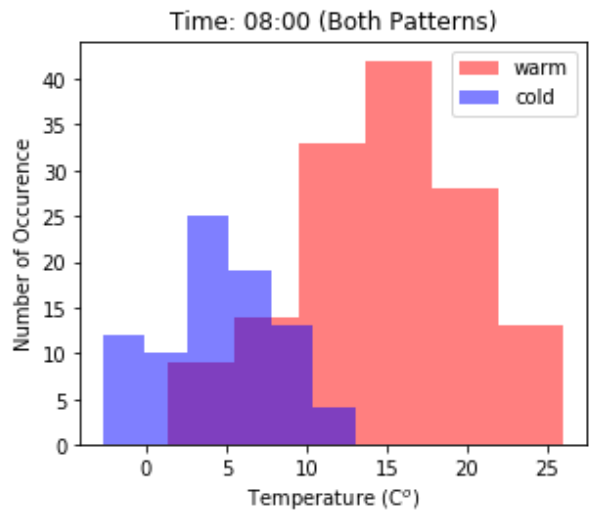

Time: 03:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

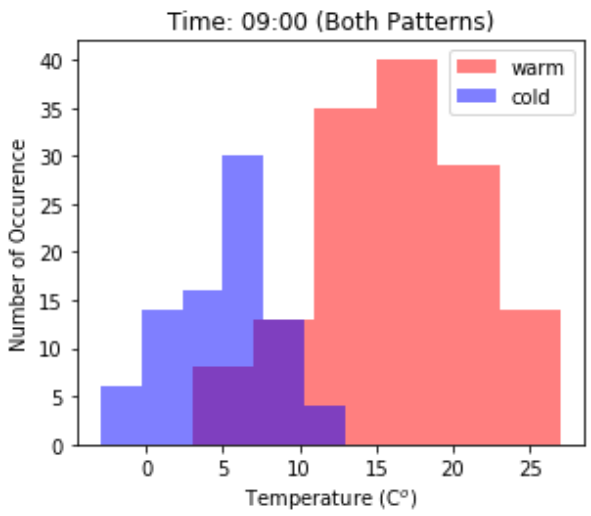

Time: 04:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

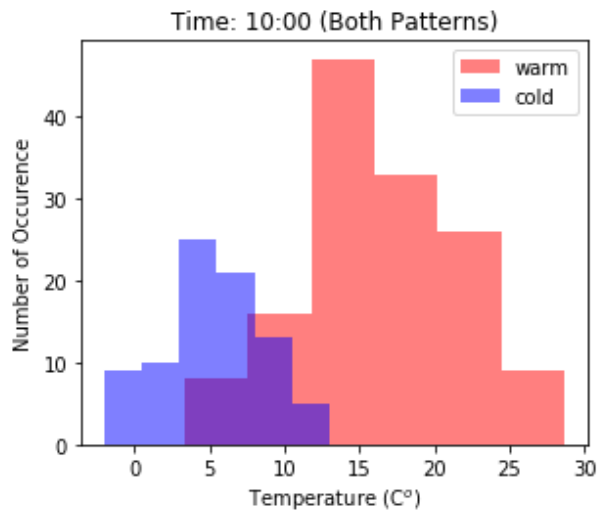

Time: 05:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

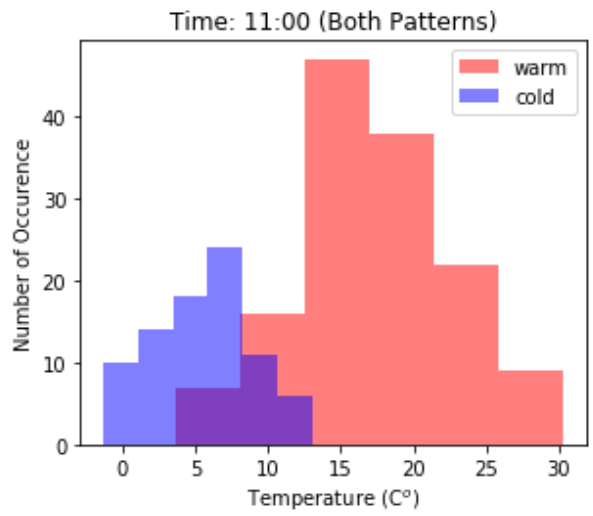

Time: 06:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

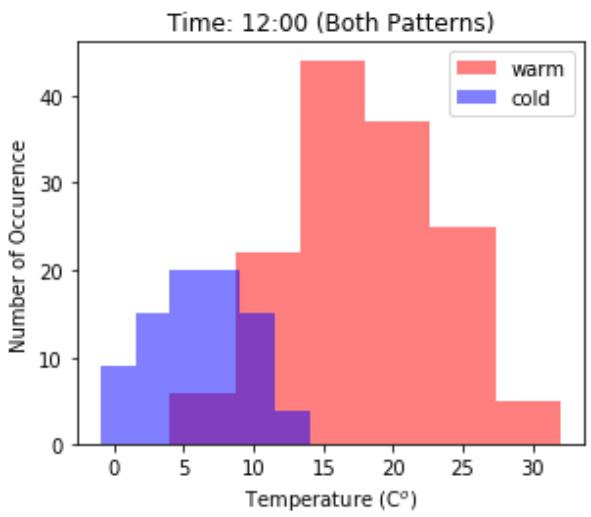

Time: 07:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

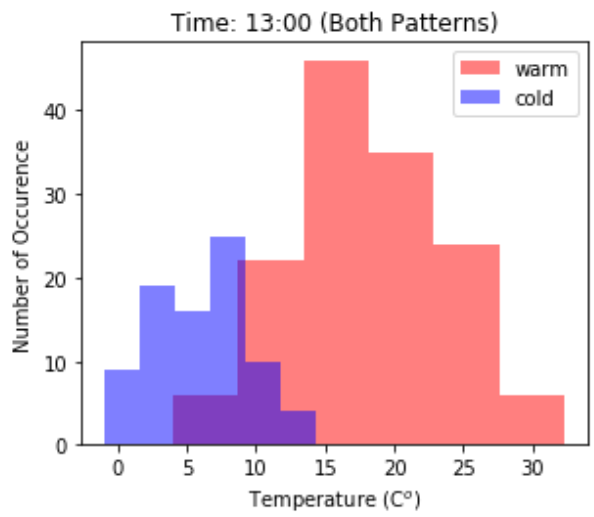

Time: 08:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

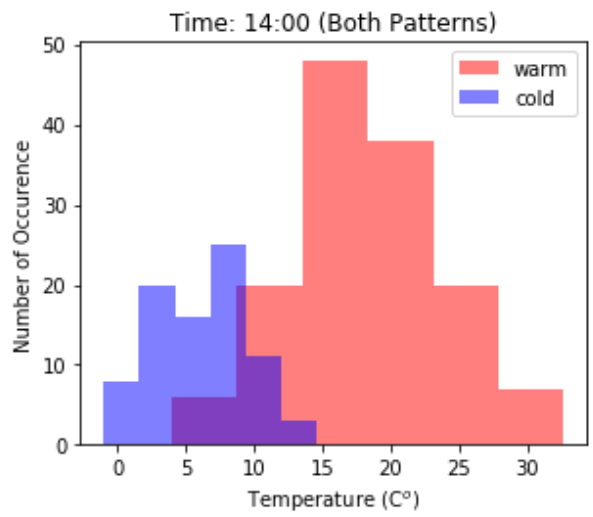

Time: 09:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

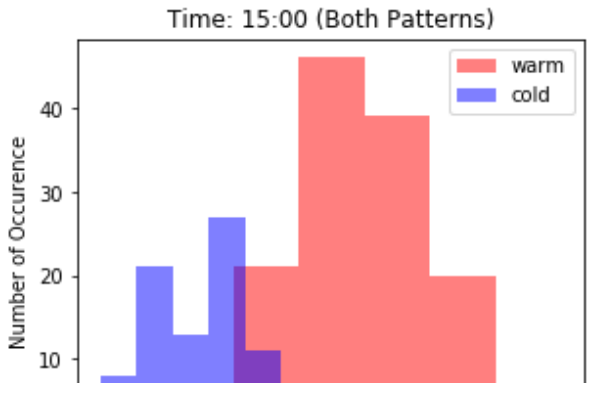

Time: 10:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

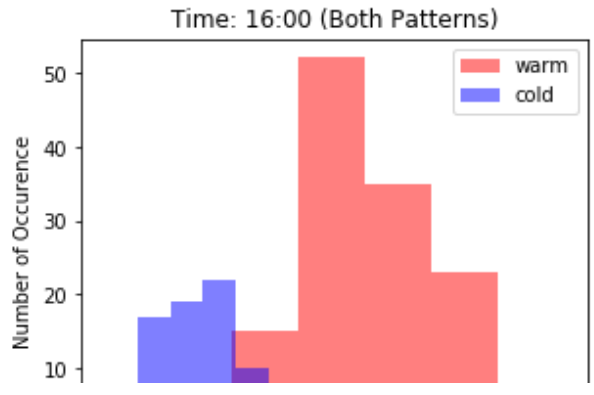

Time: 11:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

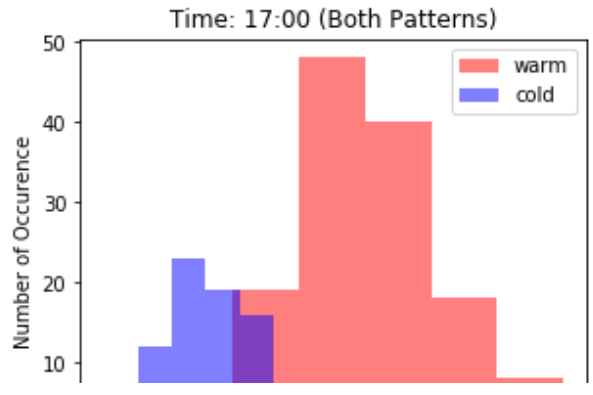

Time: 12:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)

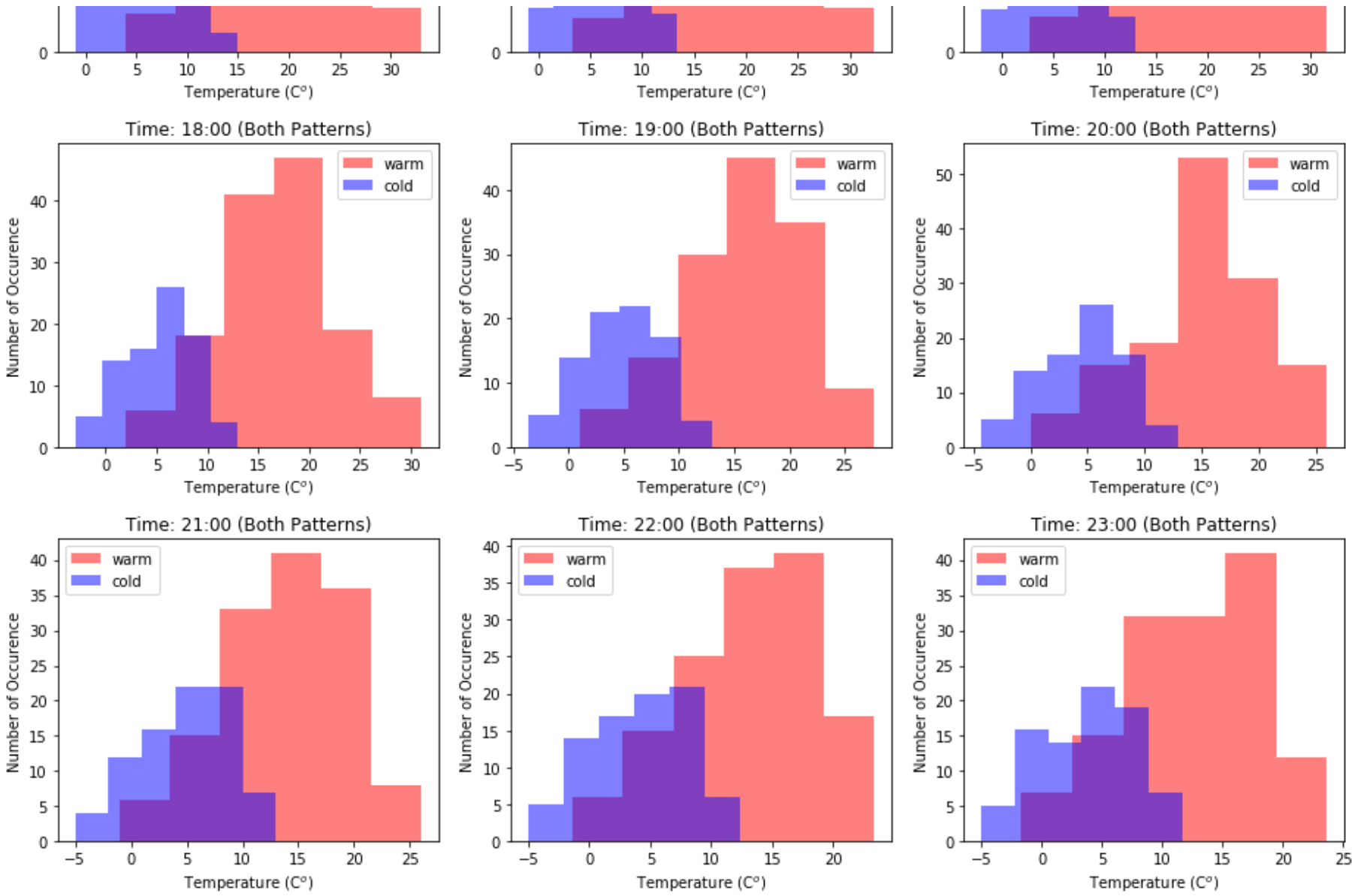

Time: 13:00 (Both Patterns)
warm
cold
Number of Occurence
Temperature (C°)


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 [19]:
```
df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('09:00:00')].transpose()
```

Out[19]:

	9	33	57	105	129	201	273	321	345	417	...	8409	8433	8505	
GMT	2013-01-01 09:00:00	2013-01-02 09:00:00	2013-01-03 09:00:00	2013-01-05 09:00:00	2013-01-06 09:00:00	2013-01-09 09:00:00	2013-01-12 09:00:00	2013-01-14 09:00:00	2013-01-15 09:00:00	2013-01-18 09:00:00	...	2013-12-17 09:00:00	2013-12-18 09:00:00	2013-12-21 09:00:00	2 1 0
N0000	0.448	0.528	0.619	0.607	0.66	0.361	0.486	1.006	0.602	0.967	...	0.248	0.477	0.585	0
N0001	0.228	0.243	0.263	0.234	0.262	0.947	0.099	0.447	1.327	0.304	...	0.464	0.566	0.474	0
N0002	0.238	0.594	0.487	0.316	0.539	0.584	0.478	0.585	0.565	0.548	...	0.226	0.344	0.224	0
N0003	0.598	0.543	0.371	0.351	0.758	0.433	0.318	0.358	0.697	0.594	...	0.585	0.814	0.556	0
N0004	0.305	0.344	0.417	0.115	0.116	0.272	0.504	0.219	0.484	0.305	...	0.201	0.549	0.349	0
N0006	0	0	0	0	0	0	0.25	0.215	0.286	0	...	0	0	0	0
N0007	0.167	0.163	0.166	0.169	0.148	0.263	0.151	0.393	0.35	0.147	...	0.373	0.253	0.398	0
N0008	0.151	0.48	0.627	0.651	0.304	0.481	0.187	1.022	0.562	0.731	...	0.206	0.49	0.629	0
N0010	0.576	0.585	0.714	0.621	0.771	0.843	0.837	3.754	3.375	5.75	...	1.704	2.467	0.58	1
N0011	0.49	0.563	0.491	0.661	0.378	0.441	0.453	0.474	0.44	0.552	...	0.494	0.285	0.523	0
N0012	0.466	0.197	0.194	0.521	0.238	0.084	0.116	0.093	0.142	0.167	...	0.133	0.055	0.309	0
N0013	0.364	0.641	0.501	0.277	0.254	0.236	0.363	0.445	0.245	0.251	...	0.233	0.252	0.308	0
N0014	1.061	1.984	2.97	1.938	1.544	3.066	3.16	2.863	1.696	2.813	...	0.871	0.787	0.803	0
N0015	0.065	0.096	0.103	0.209	0.073	0.073	0.075	0.172	0.065	0.085	...	0.058	0.12	0.044	0
N0017	0.057	0.044	0.045	0.056	0.045	0.063	0.102	0.074	0.168	0.158	...	0.079	0.227	0.134	0
N0018	0.131	0.111	0.093	0.093	0.27	0.31	0.4	0.137	0.168	0.138	...	0.148	0.438	0.464	0
N0020	0.244	0.232	0.127	0.316	0.272	0.228	0.177	0.091	0.051	0.235	...	0.305	0.228	0.251	0
N0022	1.49	1.37	1.206	0.755	0.352	1.307	0.466	1.967	0.604	0.632	...	0.302	0.31	1.158	0
N0023	0.781	0.736	0.361	0	0	4.945	3.371	4.518	4.952	5.247	...	4.389	7.045	3.806	5
N0024	2.745	3.099	1.07	2.223	2.788	1.997	0.52	4.106	1.897	0.509	...	3.185	2.534	1.554	2
N0025	0.295	0.187	0.187	0.186	0.23	0.217	0.157	0.212	0.09	0.272	...	0.127	0.149	0.173	0
N0028	0.243	0.153	0.151	0.19	0.151	0.152	0.189	0.216	0.22	0.165	...	0.306	0.41	0.301	0
N0029	0.374	2.32	3.662	0.878	1.168	3.528	0.822	2.311	3.743	3.258	...	0.885	2.226	0.135	0
N0030	0.155	0.149	0.147	0.131	0.17	1.326	0.221	0.202	0.193	0.113	...	0.136	0.331	0.166	0
N0031	0.069	0.063	0.094	0.05	0.024	0.025	0.024	0.028	0.025	0.035	...	0.091	0.036	0.035	0
N0035	0.416	0.374	0.153	0.739	0.713	0.374	0.312	0.622	0.721	0.324	...	0.905	0.26	0.224	0
N0038	0.633	0.314	1.116	0.147	0.426	0.402	0.636	0.406	0.509	0.32	...	0.356	0.316	0.404	0
N0039	0.206	0.209	0.226	0.139	0.147	0.144	0.277	0.178	0.139	0.135	...	0.269	0.198	0.174	0
N0040	1.77	1.579	1.648	1.636	1.39	1.438	2.196	1.706	2.169	3.855	...	1.999	2.579	2.636	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
N4114	0.132	0.308	0.143	0.992	0.33	0.329	0.056	0.16	0.134	0.052	...	0.131	0.134	0.442	0
N4115	0.222	0.199	0.214	0.254	0.239	0.317	0.217	0.211	0.174	0.218	...	0.52	0.568	0.826	0
N4117	0	0	0	0	0	0	0.111	0.057	0.032	0.362	...	0.167	0.08	0.18	0
N4118	0.074	0.392	0.087	0.069	0.225	0.565	0.309	0.81	0.654	0.934	...	0	0.025	1.042	0
N4120	0.384	0.619	0.391	0.856	0.561	0.702	0.789	0.315	1.116	0.35	...	0.713	0.686	0.882	0
N4122	1.047	0.989	0.668	0.898	0.808	2.304	1.902	1.691	1.681	2.041	...	0.301	0.425	1.55	1
N4123	0.148	0.058	0.061	0.127	0.074	0.052	0.121	0.063	0.122	0.111	...	0.187	0.157	0.148	0
N4128	0.495	0.427	0.443	0.387	0.478	0.266	0.439	0.672	0.194	0.384	...	0.342	0.32	0.291	0
N4129	0.152	0.116	0.136	0.093	0.07	1.795	0.474	1.621	0.553	1.81	...	0.135	0.113	1.974	1
N4130	0.165	0.612	0.662	0.314	1.152	0.225	0.231	0.716	1.109	0.294	...	0.574	0.437	0.493	0
N4131	0.379	0.265	0.156	1.298	0.597	0.311	0.865	2.193	0.281	1.913	...	0.962	0.26	0.48	0
N4134	0.251	0.791	0.772	0.19	0.919	0.739	0.268	0.468	1.083	0.607	...	0.791	0.385	0.161	0
N4136	0.132	0.112	0.102	0.088	0.251	0.199	0.144	0.063	0.107	0.146	...	0.15	0.321	0.056	0
N4137	0.165	0.272	0.526	0.543	0.101	0.062	0.111	0.344	0.333	0.536	...	0.611	0.396	0.344	0
N4139	0.281	0.283	0.26	0.257	0.269	0.272	0.385	0.302	0.305	0.303	...	0.565	0.712	0.61	0
N4140	0.177	0.356	0.601	1.191	0.25	0.201	0.73	0.161	0.187	0.133	...	0.527	0.103	0.615	0
N4141	0.3	0.124	0.3	0.169	0.136	0.292	0.299	0.313	0.266	0.169	...	1.078	0.311	0.221	0
N4145	0.217	0.28	0.08	0.049	0.081	0.086	0.054	0.045	0.05	0.419	...	0.173	0.434	0.408	0

	9	33	57	105	129	201	273	321	345	417	...	8409	8433	8505	
N4148	0.244	0.228	0.567	0.457	0.496	0.191	0.335	0.177	0.372	0.439	...	0.467	0.927	0.374	0
N4149	0.137	0.088	0.084	0.082	0.163	0.174	0.5	0.048	0.134	0.036	...	0.345	0.203	0.249	0
N4150	1.658	1.616	2.407	1.403	1.377	1.748	1.965	1.347	1.889	1.822	...	1.244	2.255	1.103	2
N4152	0.164	0.216	0.081	0.075	1.684	0.094	0.232	0.772	0.011	0.019	...	0.131	0.123	0.104	0
N4154	0.169	0.406	0.385	0.112	0.78	0.059	0.754	0.078	0.272	0.072	...	0.213	0.181	0.166	0
N4157	0.215	0.349	0.648	1.266	0.917	0.885	0.22	0.901	0.649	0.695	...	0.236	1.312	1.639	0
N4159	0.498	0.417	0.484	0.784	0.412	0.412	0.555	0.434	0.46	0.535	...	0.288	0.338	0.492	0
N4163	0.337	0.233	0.386	0.373	0.222	0.423	0.331	0.146	0.154	0.248	...	0.287	0.278	0.076	0
N4164	0.113	0.054	0.091	0.082	0.085	0.1	0.1	0.05	0.086	0.116	...	0.102	0.136	0.113	0
N4165	0.07	0.064	0.064	0.109	0.107	0.114	0.088	0.063	0.066	0.078	...	0.092	0.085	0.08	0
N4166	1.347	0.919	0.986	2.093	0.463	0.485	0.475	0.468	0.598	0.468	...	0.42	0.516	0.669	0
N4172	0.463	0.401	0.076	0.118	0.445	0.089	0.423	0.044	0.275	0.063	...	0.062	0.062	0.419	0

2552 rows × 83 columns

In [20]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Test with 9:00 am TOU data
x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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[20]:

	0	1	2	3	4	5	6	7	8	9	...	73	
0	0.696850	-0.392040	-1.587490	-0.416721	0.019867	0.137950	0.275227	0.460821	-0.205474	-0.119370	...	0.439896	0.542
1	0.106872	4.596487	1.125372	-0.395187	-0.325788	-0.520279	0.289134	-0.768706	-0.330302	0.404762	...	0.471447	-0.38
2	-1.393696	-0.190694	0.456048	-0.618870	0.286823	-0.347357	-0.936676	-0.201528	-0.314644	0.310874	...	0.110387	0.066
3	1.144654	0.081912	-0.119173	1.058585	0.030971	-0.412492	-0.368672	0.042967	-0.031950	-0.666185	...	0.038138	0.395
4	-0.773436	-0.021807	0.918711	-0.779132	1.531794	0.773537	-1.528768	-0.096780	0.646604	-0.567711	...	-0.389100	0.148
5	-5.445869	-1.396856	0.372217	-0.119503	0.927360	0.117048	0.258936	-0.135574	-0.075882	-0.199234	...	0.058762	-0.16
6	-3.300683	0.659808	0.558363	0.031726	0.301340	0.319685	0.625052	0.123881	0.202361	-0.318970	...	-0.356112	-0.11
7	-0.731941	-0.005233	0.744024	-0.013920	-0.199069	0.187361	-0.013915	0.366988	0.022477	0.681687	...	-0.279240	0.514
8	24.719375	-4.070434	4.821968	-4.681213	5.344769	-3.283358	1.501403	1.404125	-0.777467	1.377323	...	-0.223087	0.188
9	-0.903320	0.315381	-0.097327	0.007483	0.170571	0.132724	-0.268336	-0.242824	0.443323	-0.189906	...	0.007418	-0.02
10	-4.449251	0.191881	-0.839857	0.952829	0.299430	-0.050353	-0.204652	-0.170704	-0.154345	0.163083	...	-0.086519	-0.01
11	0.693322	0.162566	-0.283923	-1.281646	0.868102	2.393462	-0.985132	1.134796	1.284451	1.196596	...	-0.631204	-0.52
12	13.274190	-6.397701	-0.113598	0.515948	-3.750213	-0.908997	-0.893462	-1.465207	-0.175129	1.107827	...	0.511405	0.731
13	-5.132877	-0.115572	0.727507	-0.177384	0.220284	-0.023422	0.127272	-0.393454	-0.000739	0.024793	...	-0.034134	-0.07
14	-5.649067	0.138319	0.585896	-0.055718	0.258672	-0.129206	0.319413	-0.125599	0.305849	-0.006578	...	0.160077	-0.02
15	-3.586091	0.729428	-0.547284	0.855367	0.551088	0.503947	0.680350	0.557931	0.277927	0.007076	...	-0.154673	-0.21
16	-4.575991	-0.036125	-0.296820	-0.284802	-0.088017	-0.183515	0.127207	0.042934	0.106336	0.096181	...	-0.151831	0.130
17	5.301446	-2.492304	0.490222	0.753426	-1.617019	0.473946	0.013853	1.558116	0.156758	0.757298	...	-0.542292	0.259
18	45.351415	9.394496	9.217565	-7.011498	-1.328865	1.939653	5.541469	8.859795	-1.441850	-4.785791	...	1.104133	0.207
19	22.565888	3.310685	2.027365	3.639728	-0.984211	-1.416099	0.335722	5.205887	3.049091	-0.698691	...	-1.573733	0.109
20	-4.801660	-0.478947	0.244310	0.205582	-0.018358	-0.023082	0.006272	-0.028737	0.039662	-0.079817	...	0.022354	-0.00
21	-3.319185	1.377947	0.180433	0.037442	-0.173503	-0.224938	0.227637	-0.087274	-0.638858	-0.181766	...	0.229385	-0.08
22	13.888470	-10.821280	4.180467	4.863095	0.732785	5.191961	-2.494163	1.225562	0.406780	0.483653	...	0.888488	0.705
23	-3.548630	0.049964	0.472180	0.345174	-0.298400	-0.133575	-0.329806	0.468743	-0.200892	-0.358924	...	-0.338376	-0.05
24	-6.479144	-0.029389	0.453018	-0.012906	0.159083	-0.033711	0.071763	-0.102691	0.076302	-0.008396	...	0.031484	-0.06
25	0.604123	1.190310	0.789834	0.760391	0.305652	-1.352395	0.178396	-0.154333	0.778207	-0.473147	...	0.114940	-0.21
26	-2.472401	-0.482678	-0.935956	0.481854	0.039207	0.139094	-0.449509	-0.324320	0.219680	-0.307150	...	0.142949	0.274
27	-3.618672	-0.240506	-0.105126	-0.475714	0.278861	-0.237425	0.059422	-0.001408	0.207320	-0.193689	...	0.151424	0.047
28	25.401644	1.456218	-0.452428	1.336540	-1.140297	-0.476787	3.383437	-1.043647	-0.229054	0.643797	...	0.139068	0.302
29	1.997127	0.427698	0.917162	1.024798	-0.245903	0.836485	0.791193	-0.546366	-2.038779	-0.518014	...	0.104002	0.241
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2521	-3.926191	0.194026	-0.179276	-0.385553	-0.397377	0.270865	0.116941	-0.161971	-0.056737	-0.459369	...	-0.311347	-0.09
2522	-2.580257	0.742887	-0.068947	0.517387	0.119106	0.561228	0.330140	0.619638	0.358071	0.039595	...	0.105248	-0.28
2523	-5.464003	0.242233	0.164708	-0.349013	0.565732	-0.134169	0.279379	-0.215719	-0.011160	-0.012135	...	-0.090273	-0.05
2524	-3.639795	-1.970223	0.188345	1.362096	0.756797	0.781278	0.354763	0.061718	-0.525030	0.122009	...	0.018572	-0.41
2525	0.446163	0.334126	-0.315432	1.086870	-0.836385	0.382255	-0.011019	-0.102797	0.896956	0.192627	...	-0.089349	-0.21
2526	13.405413	-3.257119	-2.425468	1.045696	1.237690	-1.046605	2.617127	0.727864	-1.852659	-1.077801	...	-0.600168	-1.07
2527	-5.259591	0.603164	0.127633	0.016877	0.098684	0.205754	0.099751	0.031860	-0.038061	-0.051041	...	-0.026228	-0.15
2528	-1.000221	0.128895	-0.029040	-0.167070	0.415308	-0.293276	-0.401968	-0.707520	0.070975	-0.076244	...	0.110897	0.354
2529	3.037302	-2.183674	-1.453205	1.071054	0.827134	2.050791	2.699747	1.549875	-1.043971	2.124196	...	0.042202	-0.85
2530	1.116998	-0.027708	-0.543724	-0.485067	0.076820	-0.619635	-1.628813	0.240269	-0.587342	-0.157165	...	0.045836	-0.40
2531	3.875259	-1.450202	-0.643770	-0.277516	-0.998481	0.454783	1.432800	1.023305	-0.864239	1.654626	...	0.998811	0.255
2532	3.393949	-0.798296	0.263632	0.614933	1.282059	-1.388269	0.382497	0.551886	-0.146327	-2.470074	...	0.470311	-1.36
2533	-4.852199	0.074888	0.445466	-0.027327	-0.171046	-0.380940	0.266448	0.010828	-0.017821	0.068135	...	0.006598	-0.27
2534	-3.726914	-0.335932	1.457835	0.139601	-0.582365	0.018075	-0.298382	-0.144772	0.309338	0.281115	...	0.456729	-0.21
2535	-2.167411	0.894566	0.366119	0.222347	0.168594	-0.481538	0.771011	0.292038	0.252226	-0.361472	...	-0.256024	-0.13
2536	-2.677111	-0.091009	-2.377999	-0.845420	0.123564	0.172718	-0.184717	0.717734	-0.494401	-1.236634	...	0.357198	0.647
2537	0.147688	3.982390	2.350595	-1.112576	-0.710023	-1.253654	0.841902	-0.597877	0.922652	0.462972	...	0.209354	-0.18
2538	-1.003604	-0.232469	1.012569	0.941552	0.559922	-0.023755	0.570447	0.020736	0.887189	1.131658	...	-0.692308	-0.16
2539	-1.360359	-0.621068	-1.905744	-0.554241	0.372792	-0.277871	0.602383	1.235489	0.107591	0.526025	...	-0.380267	0.242

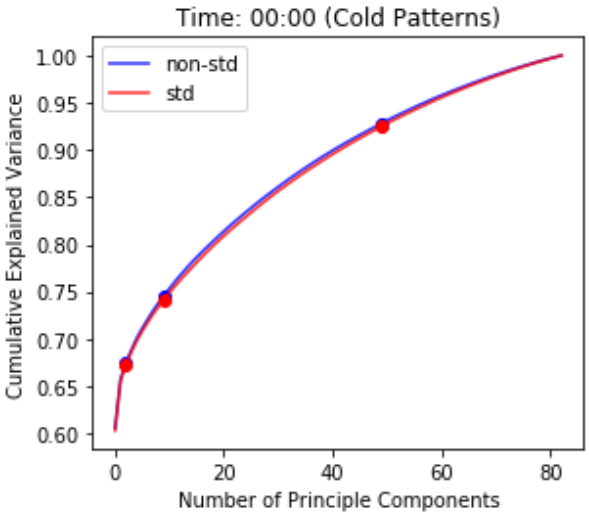
	0	1	2	3	4	5	6	7	8	9	...	73	
2540	-3.208354	0.410025	-0.626284	-0.869626	0.355194	0.433979	-0.014880	-0.642639	0.807228	-0.045501	...	0.038633	0.046
2541	15.129383	-1.004322	-0.740064	2.035567	0.852061	-0.185467	-1.109838	0.046927	-0.123795	-0.291136	...	0.507139	-0.51
2542	-4.484773	-0.440743	-0.256156	-0.059863	-0.536502	-0.457869	0.069165	0.333083	-0.323461	0.123303	...	-0.308431	0.390
2543	-4.767557	-0.743430	-0.456029	0.080625	0.000426	0.057765	-0.127347	0.090443	-0.093506	-0.612455	...	0.504272	-0.47
2544	1.256105	-2.338706	1.488103	1.191418	-0.962439	0.738059	0.031546	0.303843	0.500751	0.301156	...	0.135316	-1.00
2545	-1.151128	0.294099	0.285351	0.328660	-1.047471	0.240591	-0.128327	-0.446528	-0.776249	-0.253193	...	-0.401215	-0.24
2546	-2.944565	-0.062968	-0.002722	-1.052963	-0.335989	-0.056827	0.313068	0.146963	0.470045	-0.212425	...	-0.142360	0.507
2547	-5.821350	-0.059305	0.355248	0.026931	0.160313	-0.026388	0.045502	-0.043106	0.054250	-0.012296	...	0.001389	-0.07
2548	-5.905235	0.090205	0.286259	-0.054929	0.110939	-0.081271	-0.030528	-0.053633	0.002693	0.039946	...	-0.028830	0.016
2549	-0.714634	-1.307532	-0.953467	1.588819	-1.417572	-1.042981	-0.197208	-0.660428	0.322697	0.578401	...	0.491539	-0.48
2550	-3.575326	-0.393457	-2.258284	0.092749	1.245609	-0.553268	-0.651179	0.614406	-0.234088	0.357627	...	-0.315628	-0.07

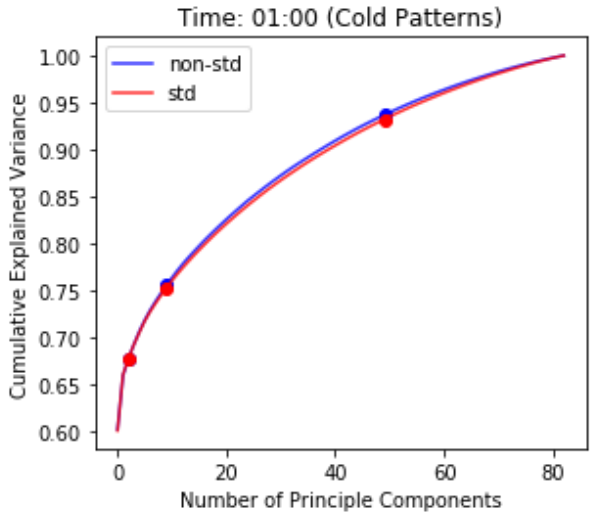
2551 rows × 83 columns

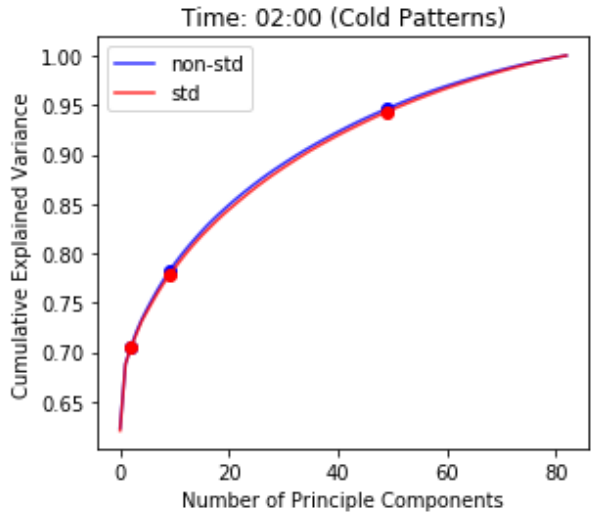
In [15]:
```
sum(x[:,0])
```

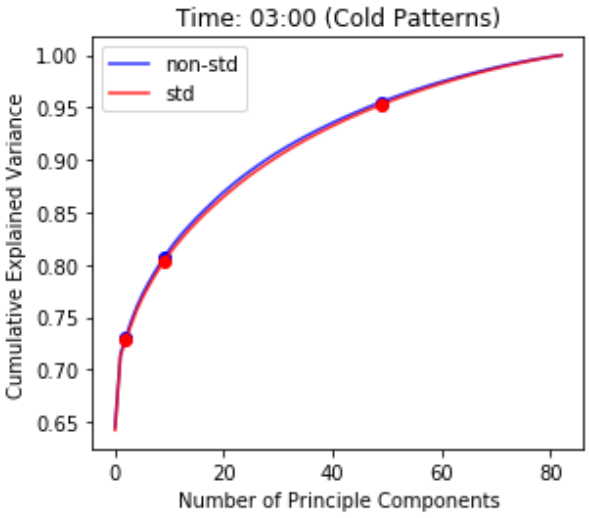
Out[15]: 4.796163466380676e-14

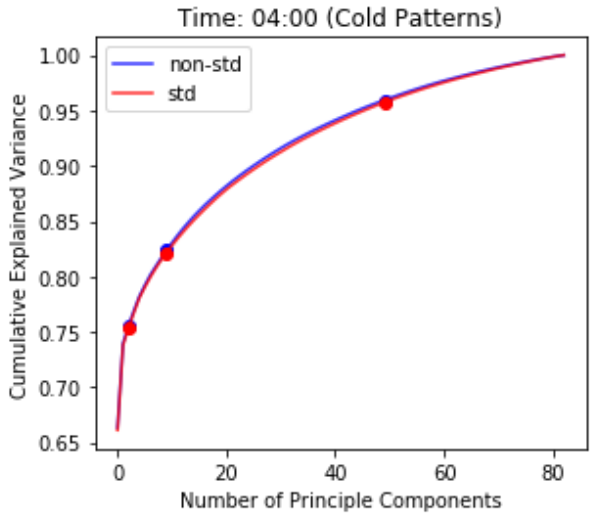
In [21]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Number of Principle Components')
        ax.set_ylabel('Cumulative Explained Variance')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Number of Principle Components')
        ax.set_ylabel('Cumulative Explained Variance')
plt.tight_layout()
```

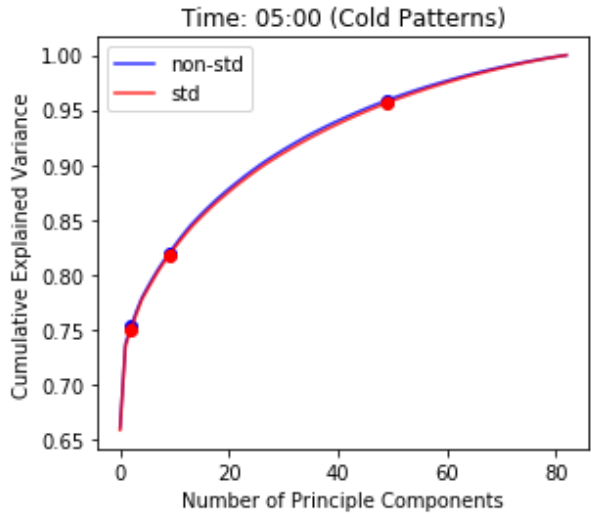

Time: 00:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

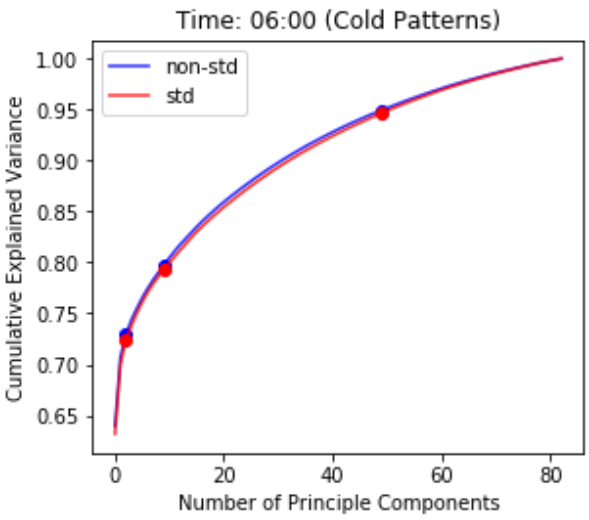

Time: 01:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

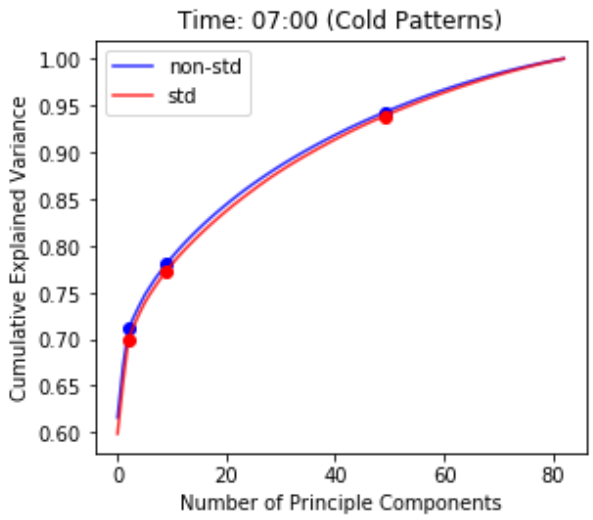

Time: 02:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

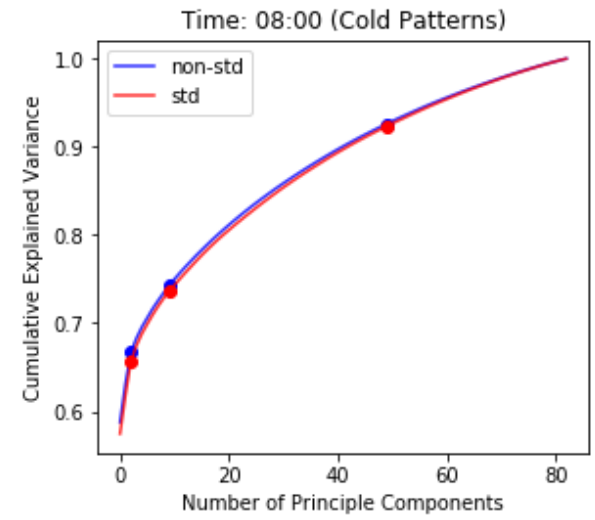

Time: 03:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

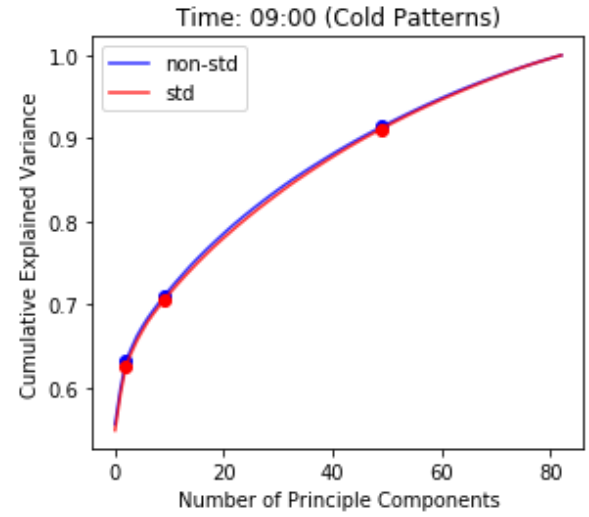

Time: 04:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

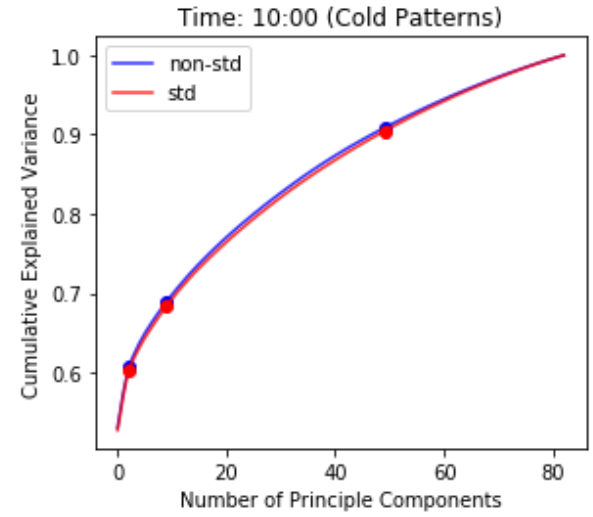

Time: 05:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

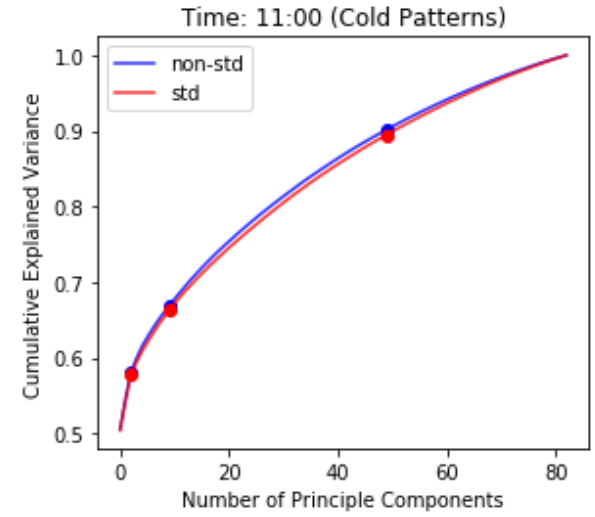

Time: 06:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

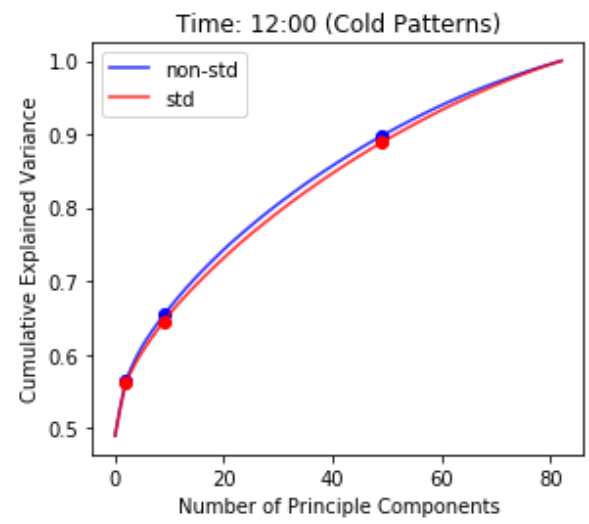

Time: 07:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

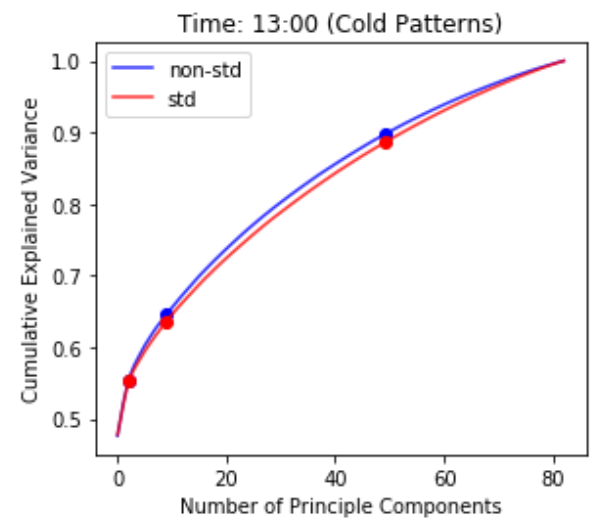

Time: 08:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

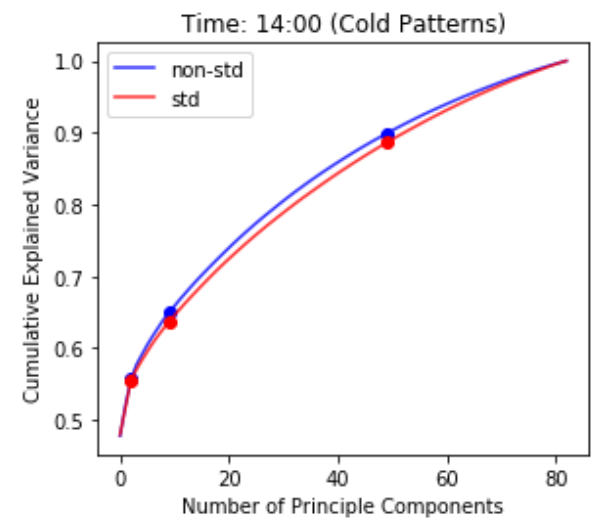

Time: 09:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

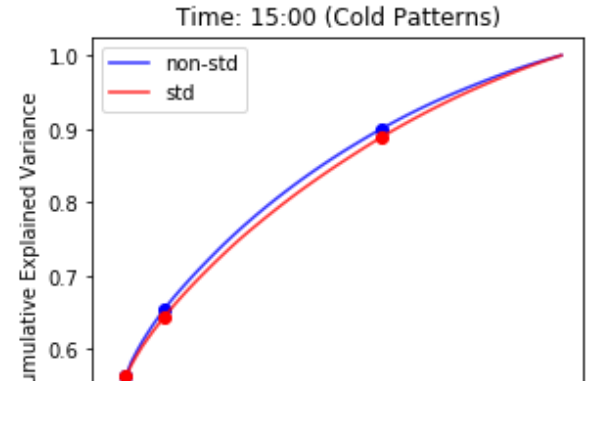

Time: 10:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

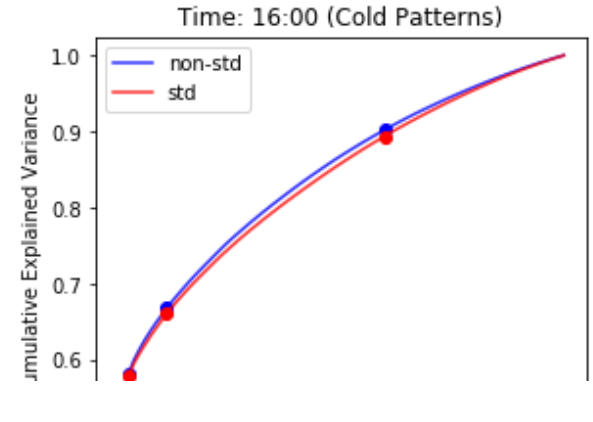

Time: 11:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

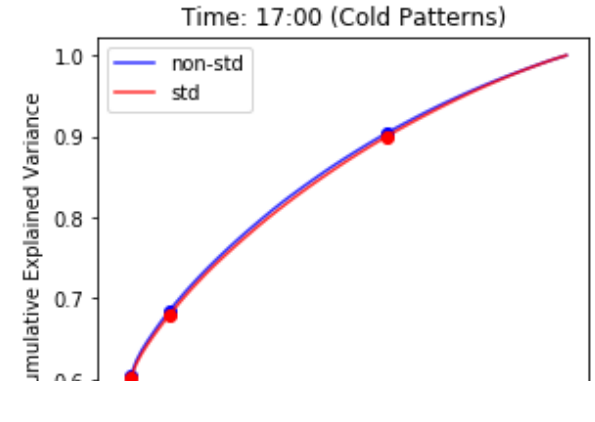

Time: 12:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components

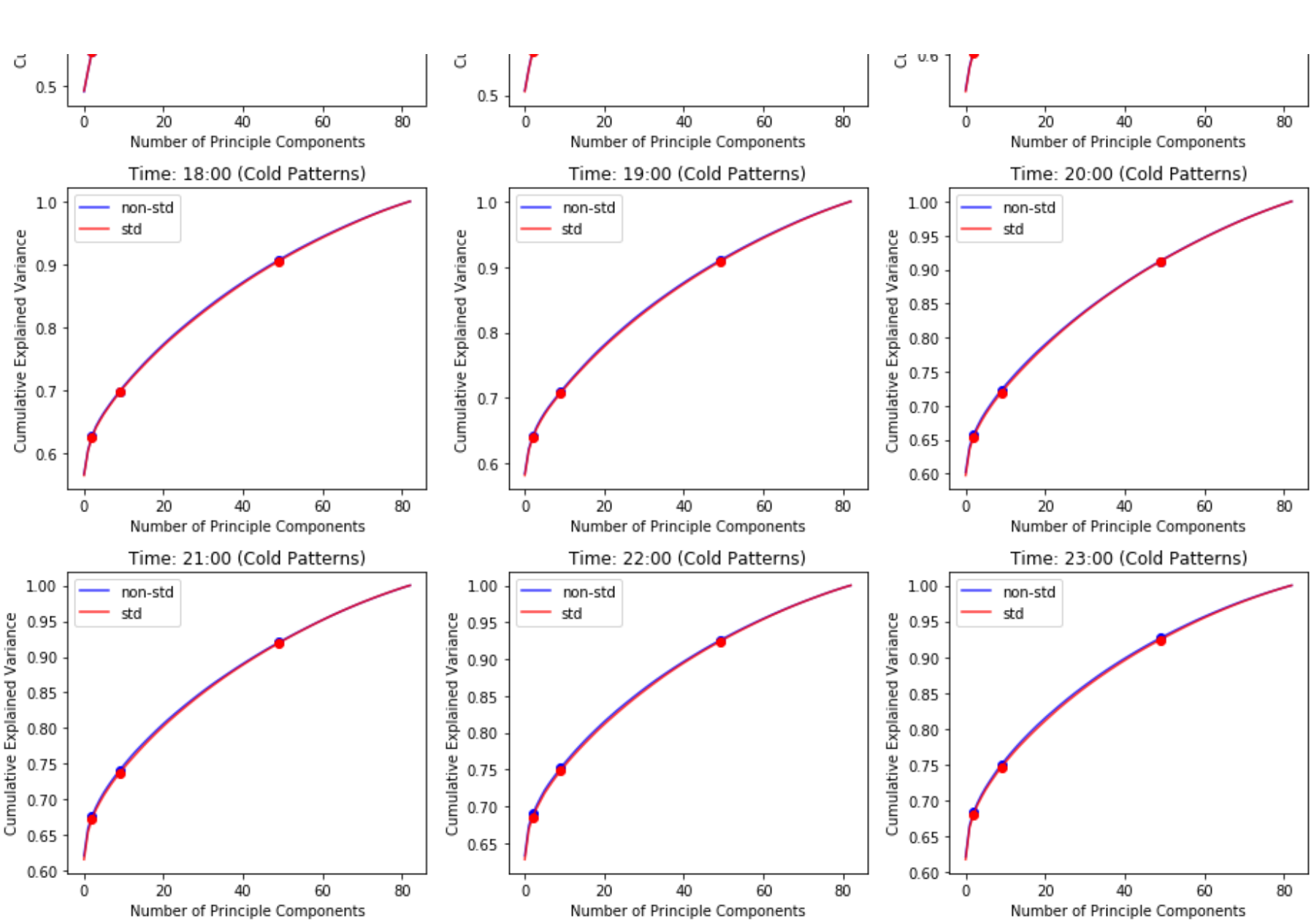

Time: 13:00 (Cold Patterns)
non-std
std
Cumulative Explained Variance
Number of Principle Components


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


Time: 15:00 (Cold Patterns)
non-std
std


Time: 16:00 (Cold Patterns)
non-std
std


Time: 17:00 (Cold Patterns)
non-std
std



The above pictures show that using the first 3 PCs can always explain from 48% to 72% 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)

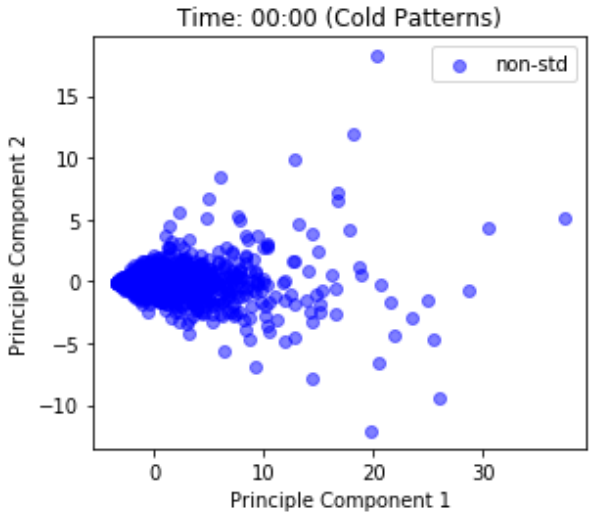
PC3: related to days of week, since the coeffients of Friday and Saturday are skewied to positive area, if there's not correlation, it should always be equally distributed around 0

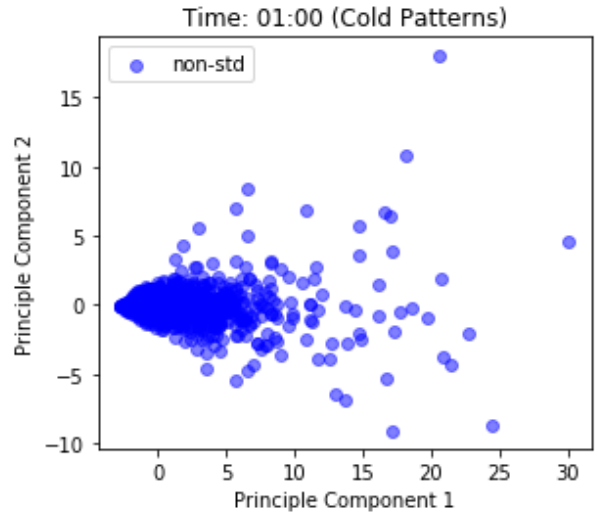
Using the 3 components to do k-mean clustering and we can get several groups

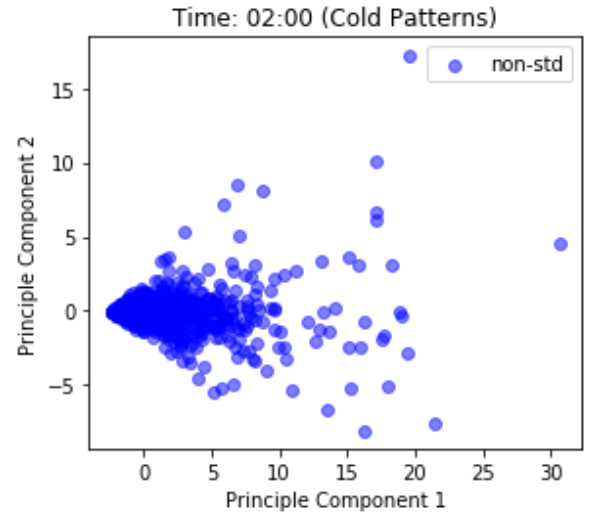
Coloring them in a 3D space (or if the first two PCs leads to some confustion, maybe do 2D with 3 differnt projections)

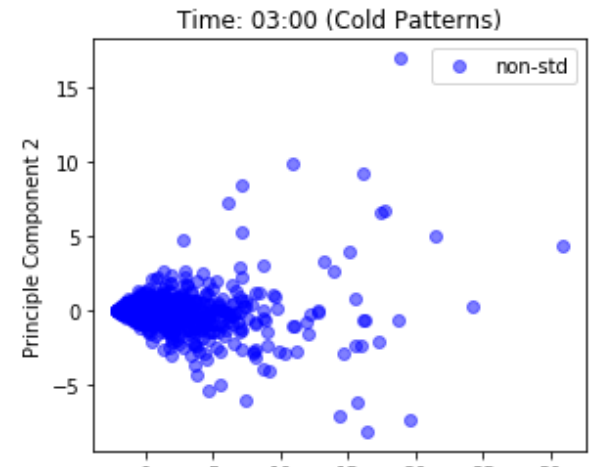
Then we could try to color them in a 2D slice surphase (temp vs day of week, temp)

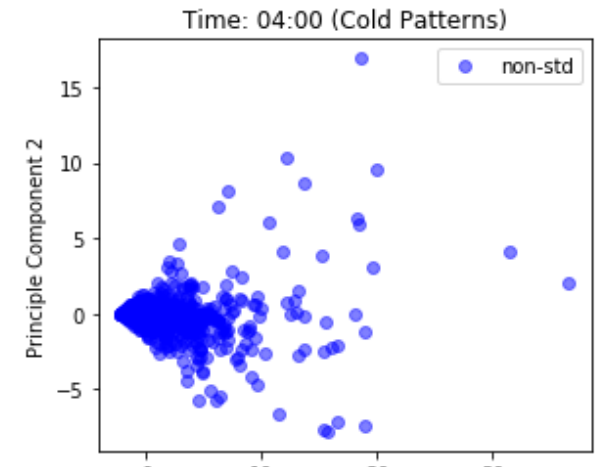
In [22]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

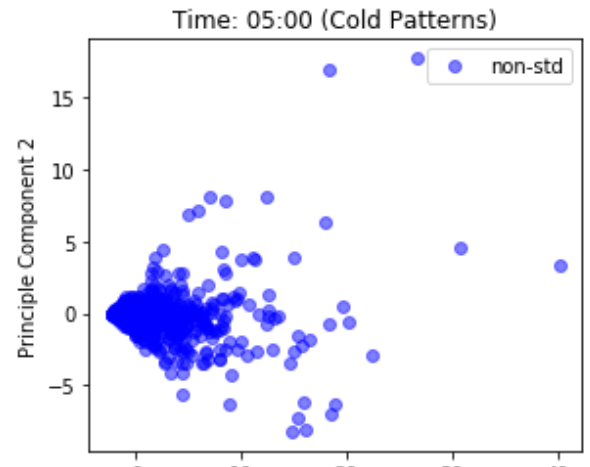

Time: 00:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

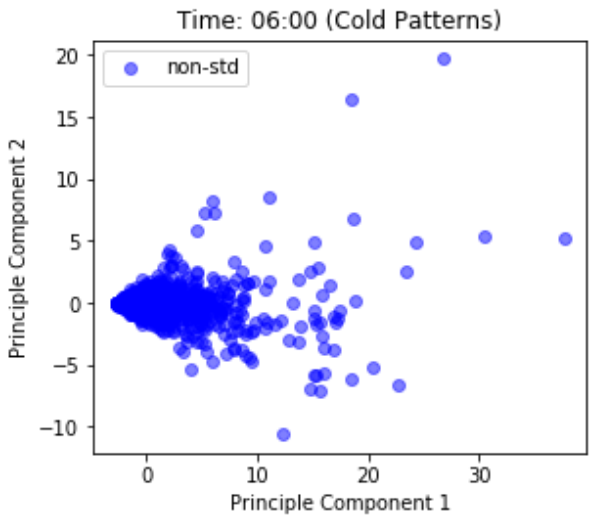

Time: 01:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

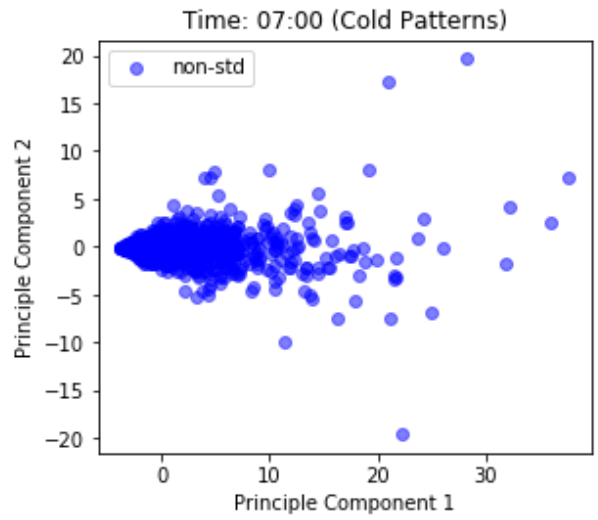

Time: 02:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

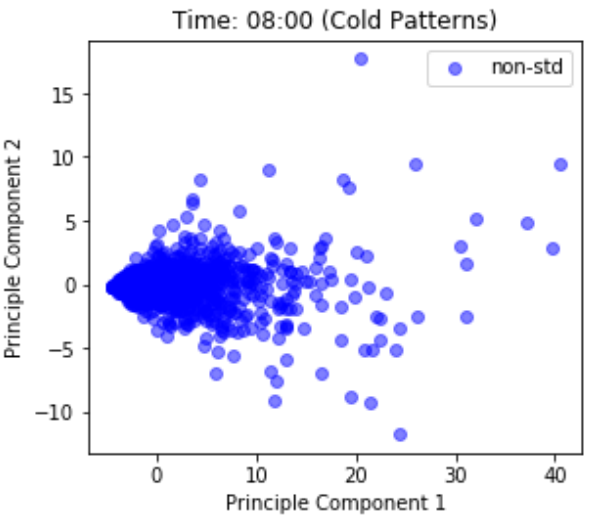

Time: 03:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

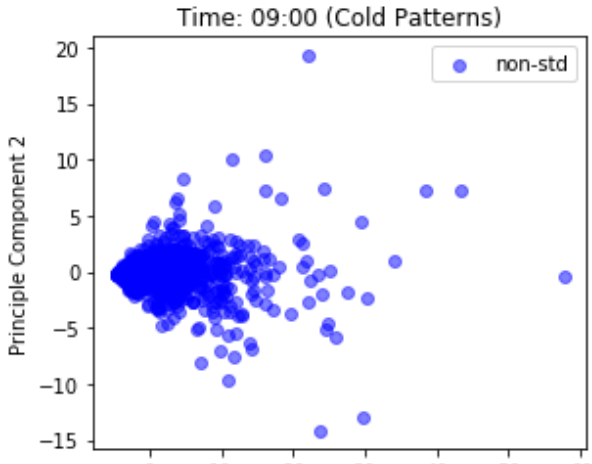

Time: 04:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

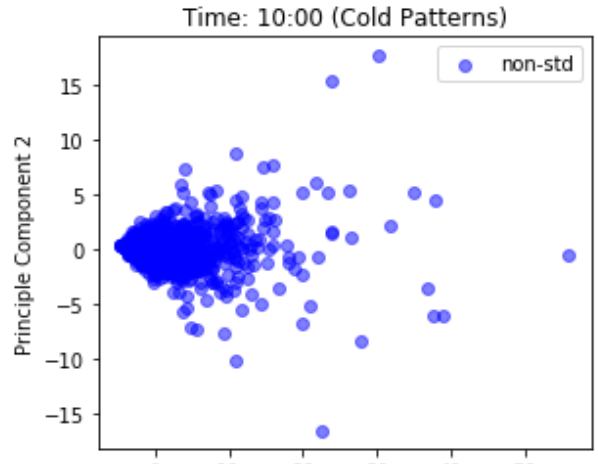

Time: 05:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

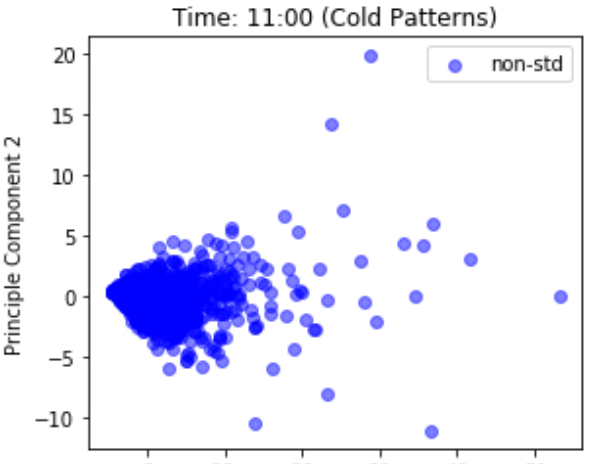

Time: 06:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

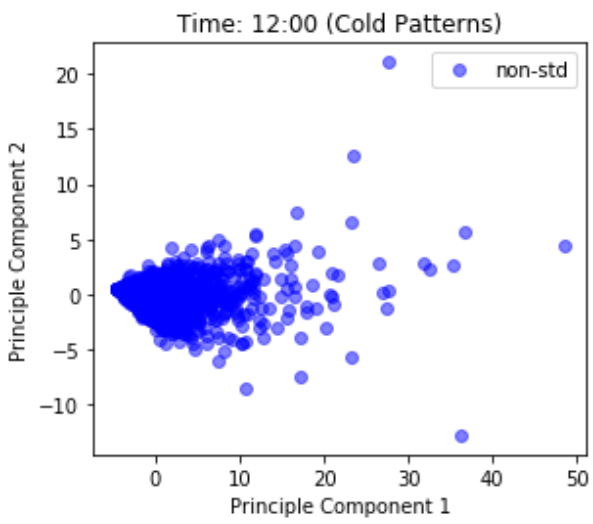

Time: 07:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

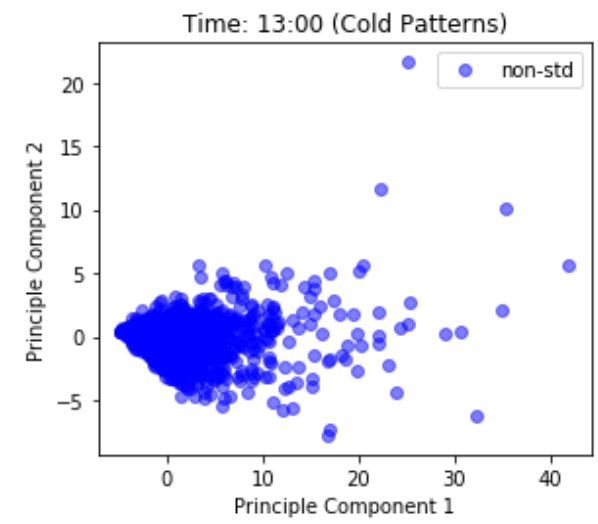

Time: 08:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

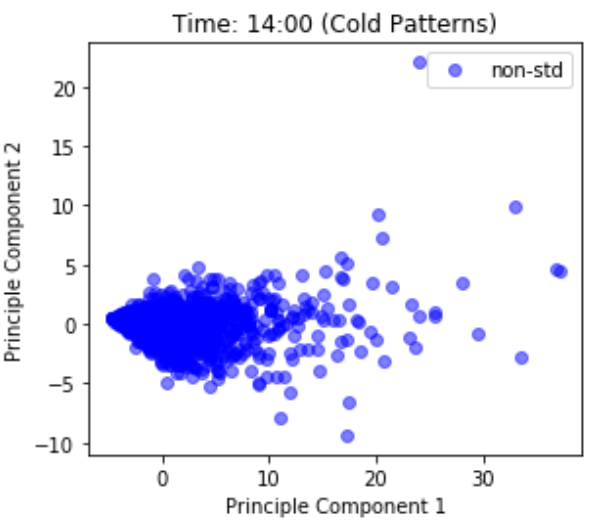

Time: 09:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

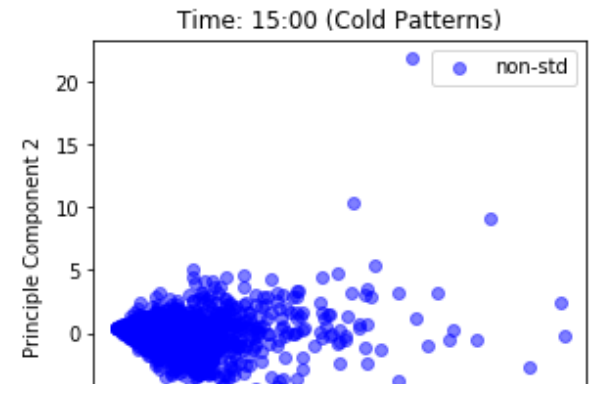

Time: 10:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

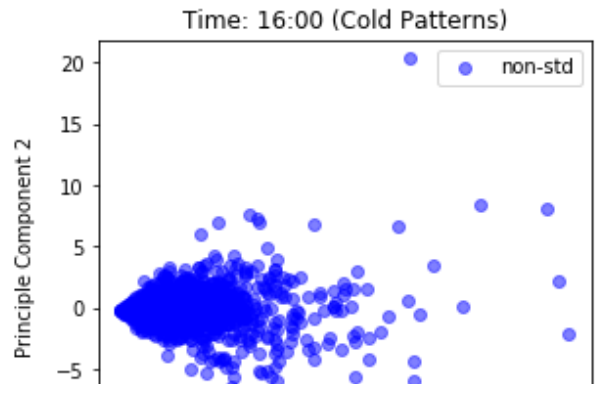

Time: 11:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

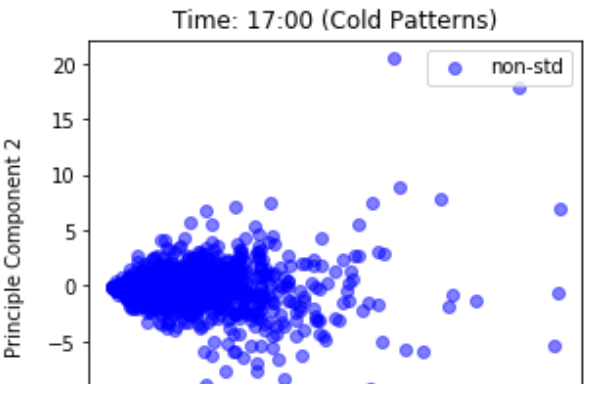

Time: 12:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

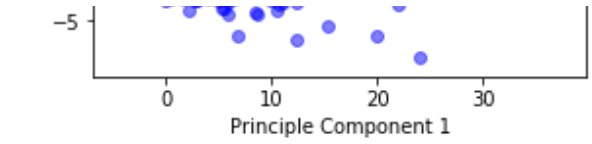

Time: 13:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

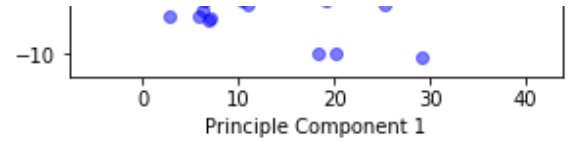

Time: 14:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

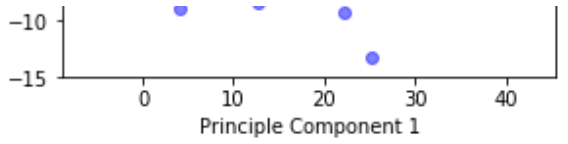

Time: 15:00 (Cold Patterns)
non-std
Principle Component 2

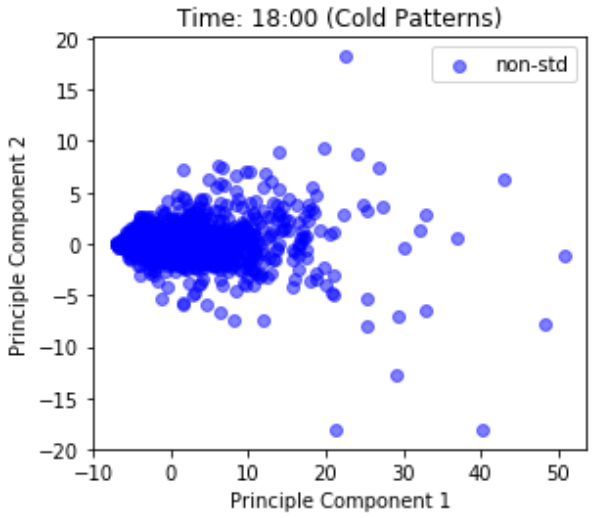

Time: 16:00 (Cold Patterns)
non-std
Principle Component 2

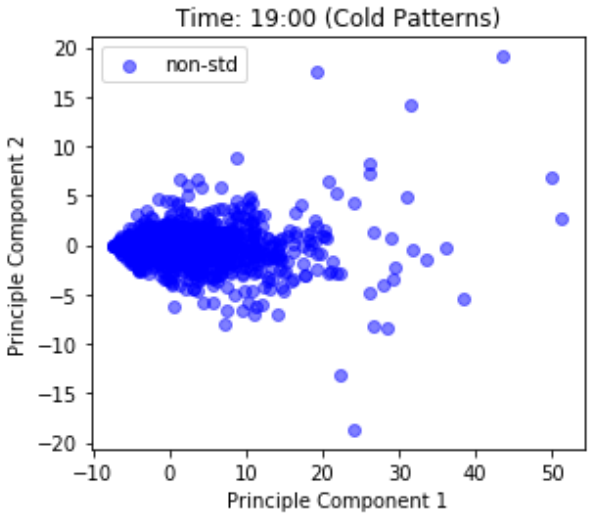

Time: 17:00 (Cold Patterns)
non-std
Principle Component 2

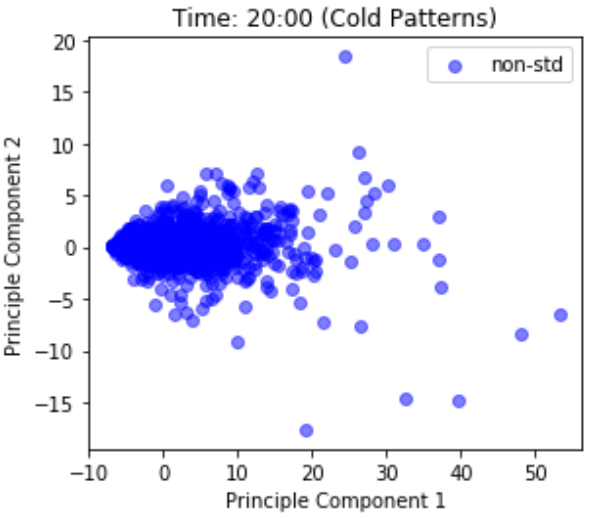

Principle Component 1

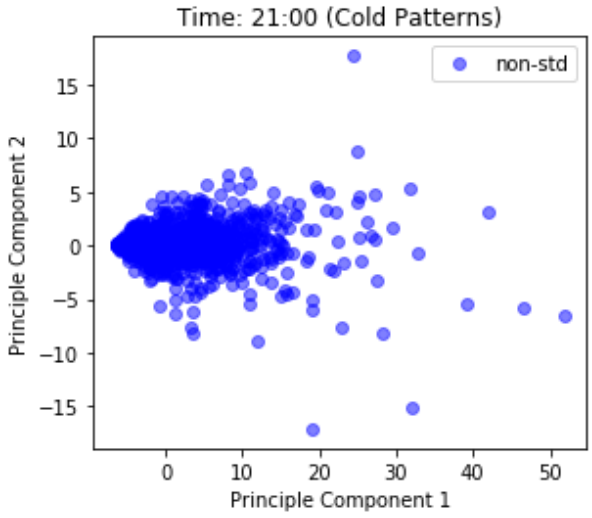

Principle Component 1

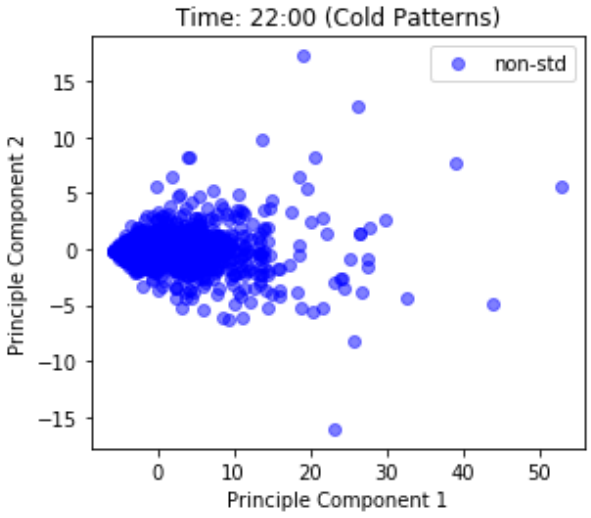

Principle Component 1

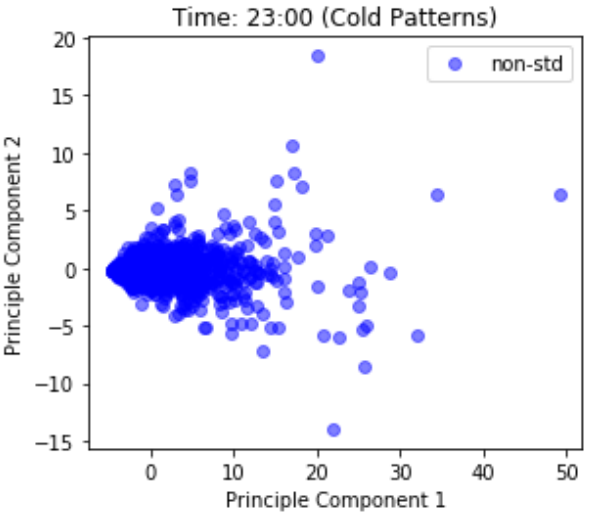

Time: 18:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1


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

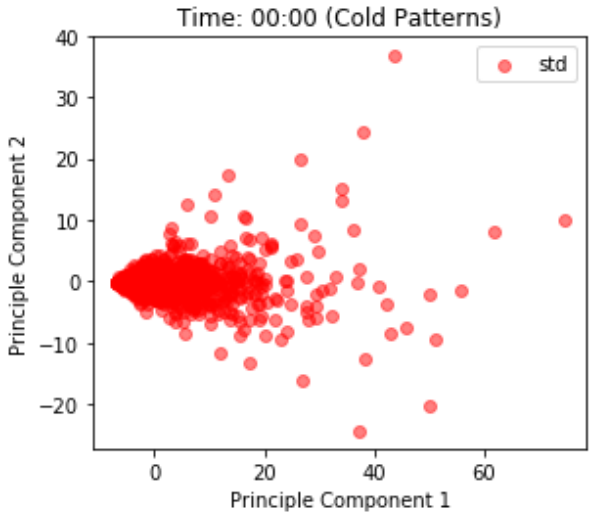

Time: 20:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

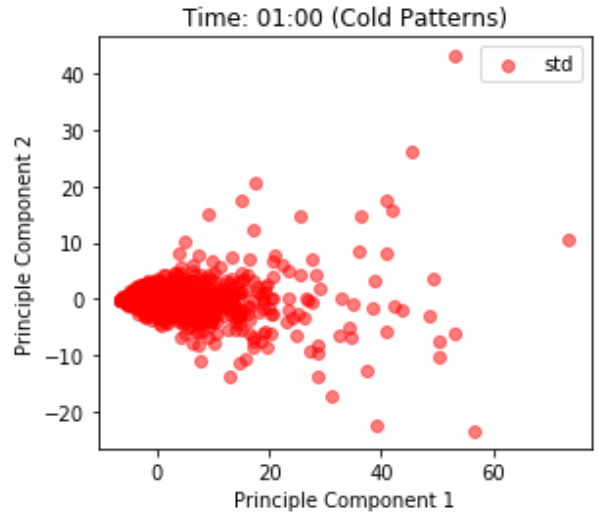

Time: 21:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

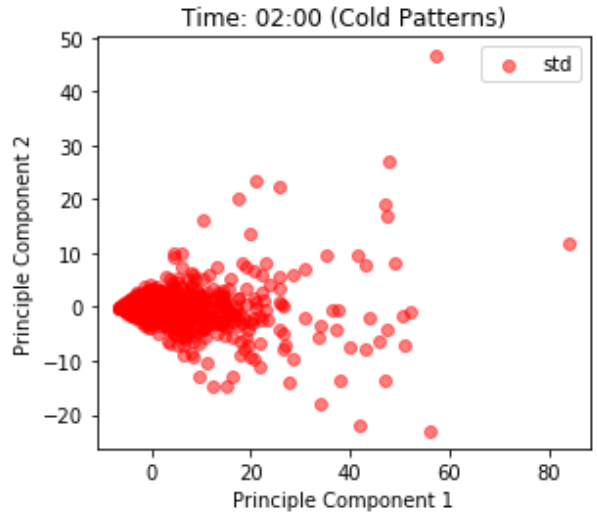

Time: 22:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

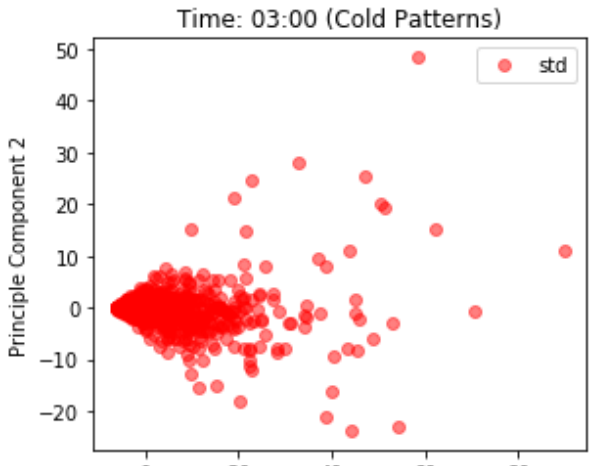

Time: 23:00 (Cold Patterns)
non-std
Principle Component 2
Principle Component 1

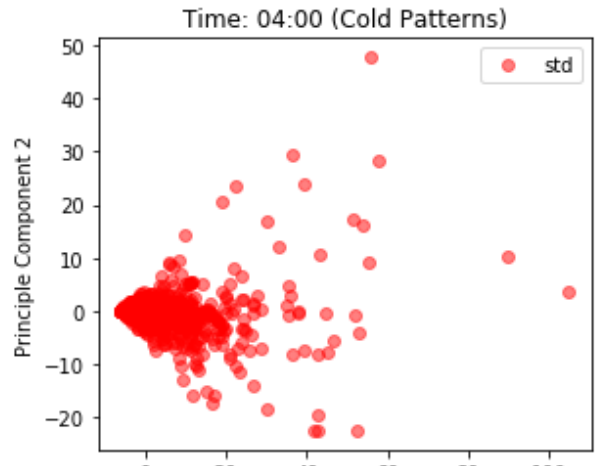
In [23]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

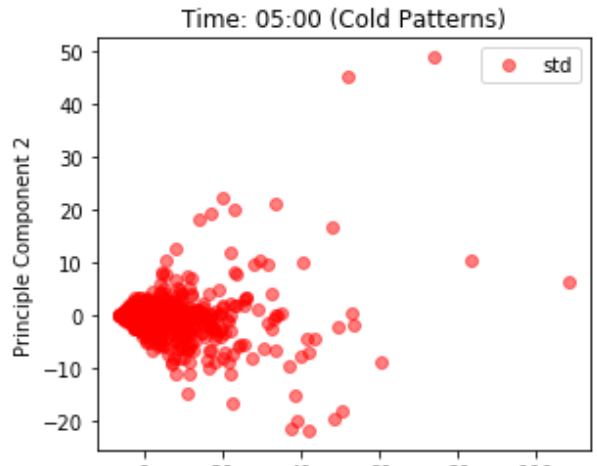

Time: 00:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

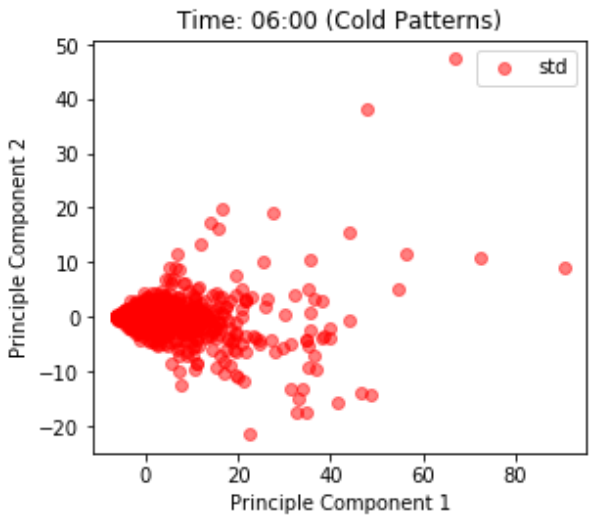

Time: 01:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

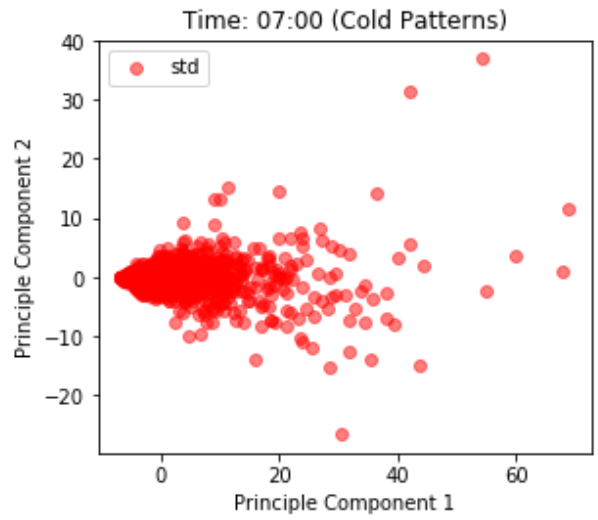

Time: 02:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

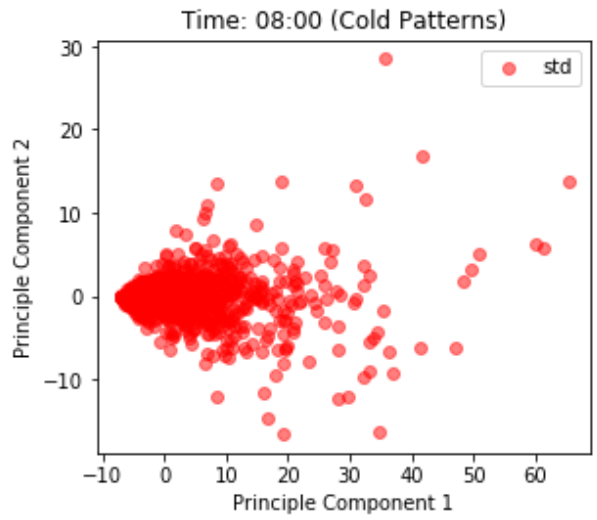

Time: 03:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

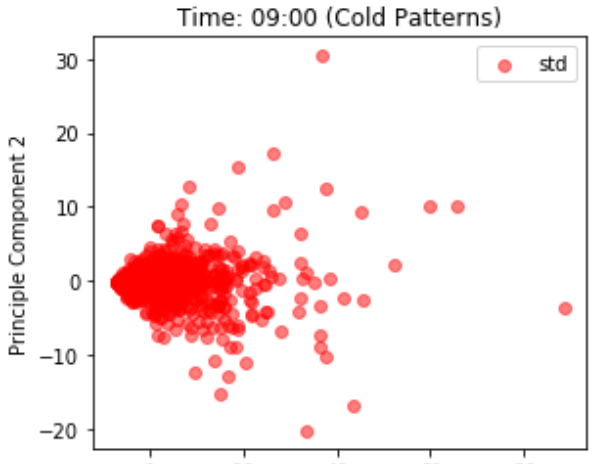

Time: 04:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

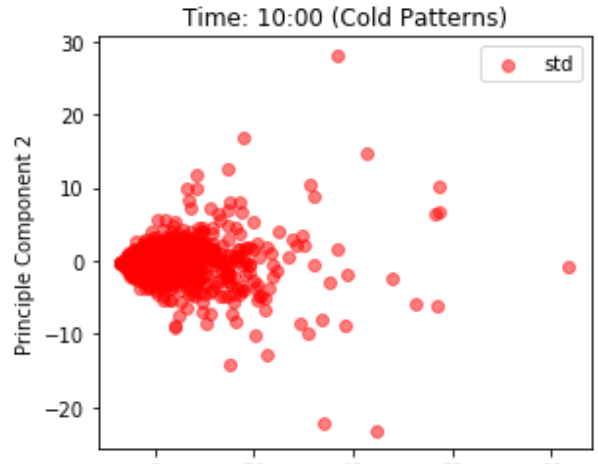

Time: 05:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

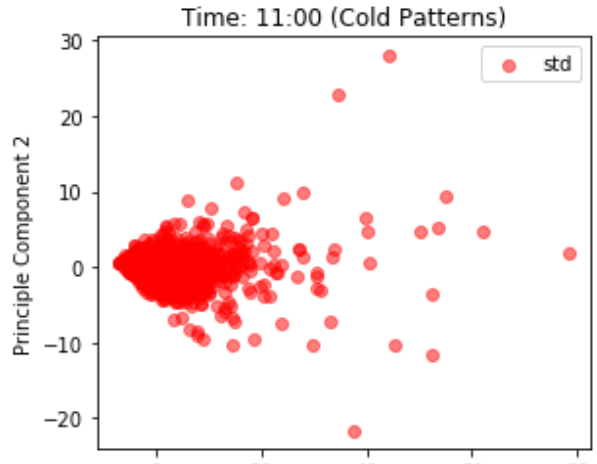

Time: 06:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

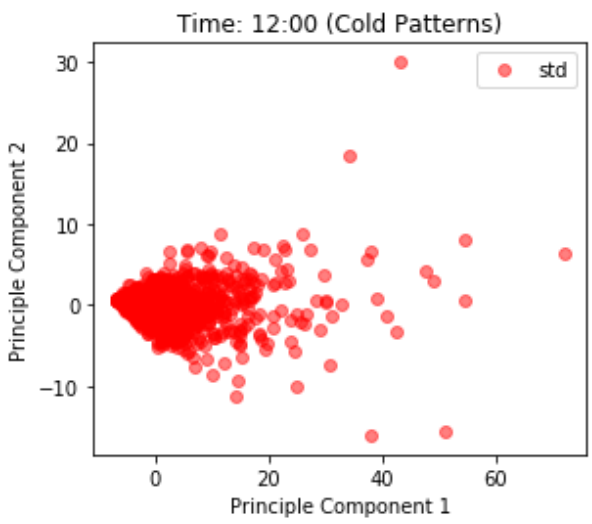

Time: 07:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

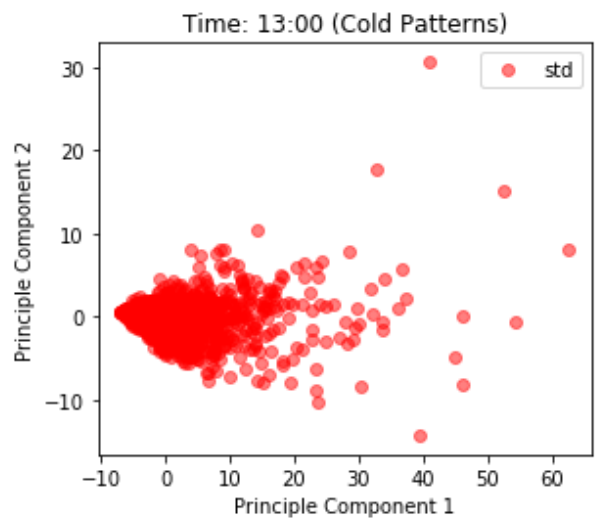

Time: 08:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

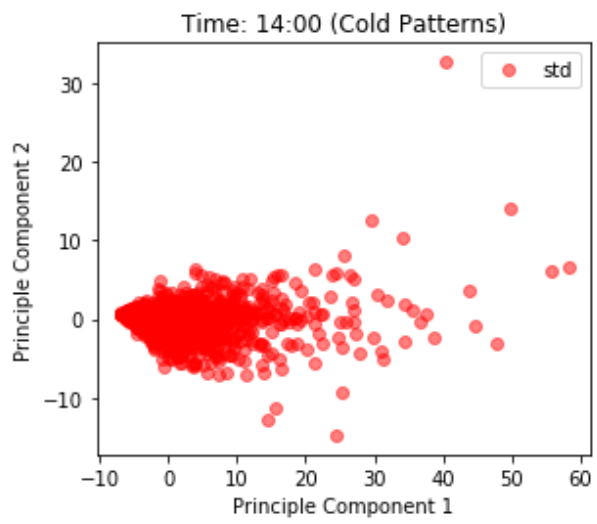

Time: 09:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

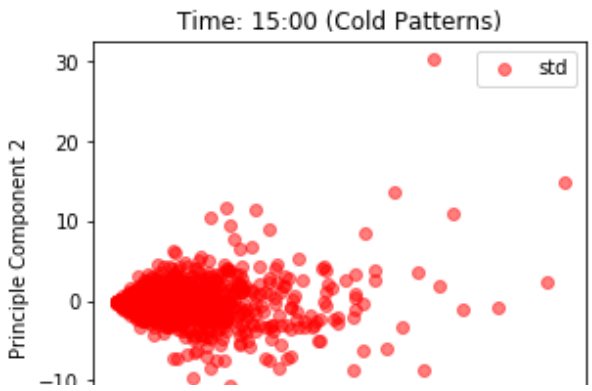

Time: 10:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

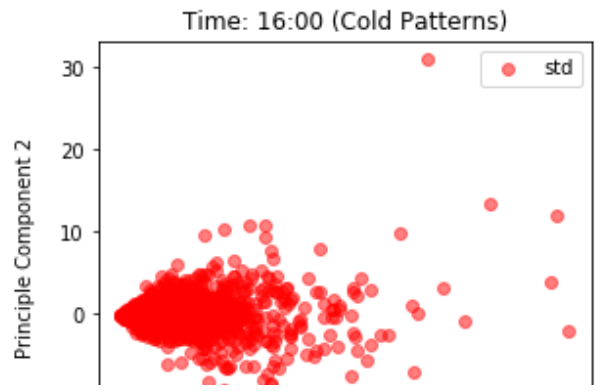

Time: 11:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

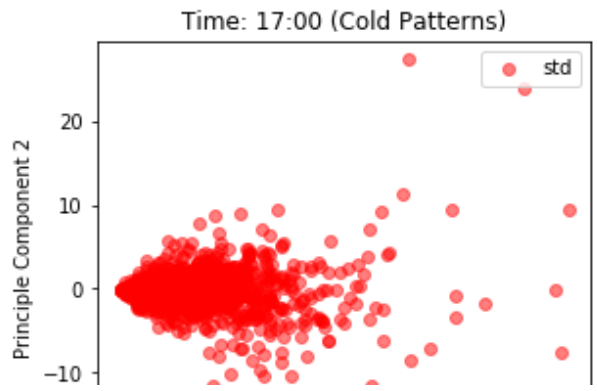

Time: 12:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1

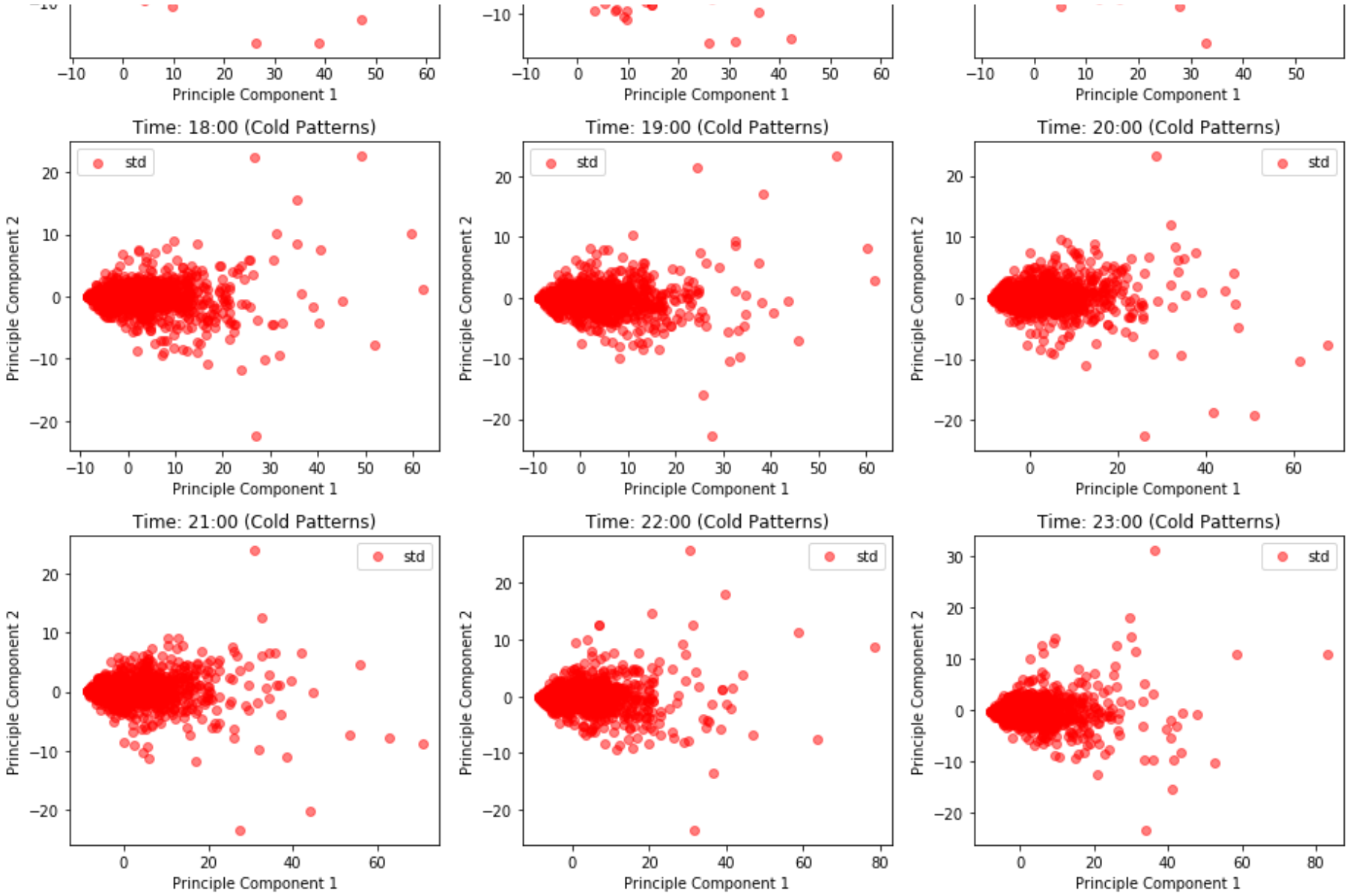

Time: 13:00 (Cold Patterns)
std
Principle Component 2
Principle Component 1


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


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


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

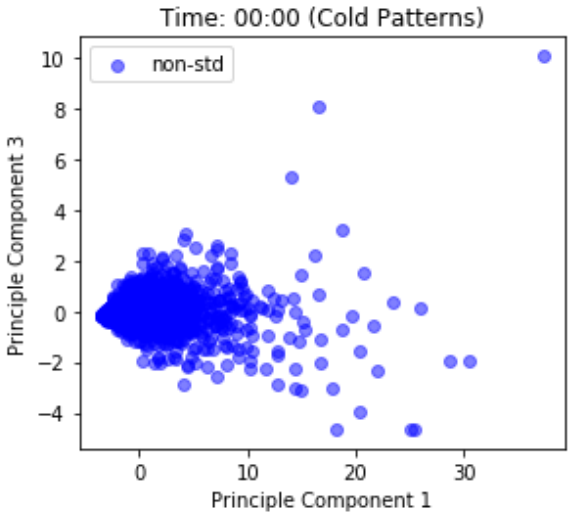

Time: 17:00 (Cold Patterns)
std
Principle Component 2

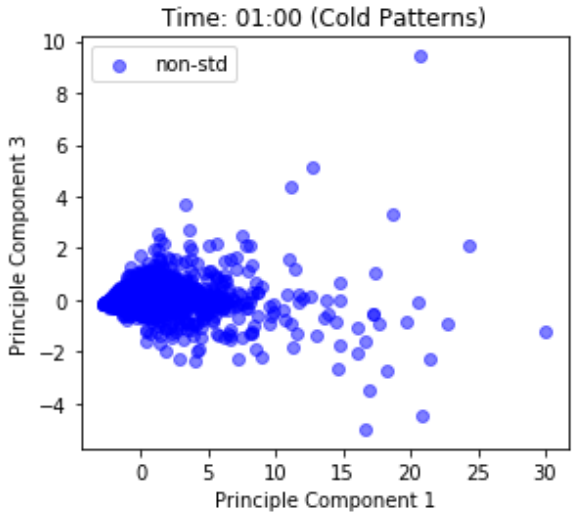


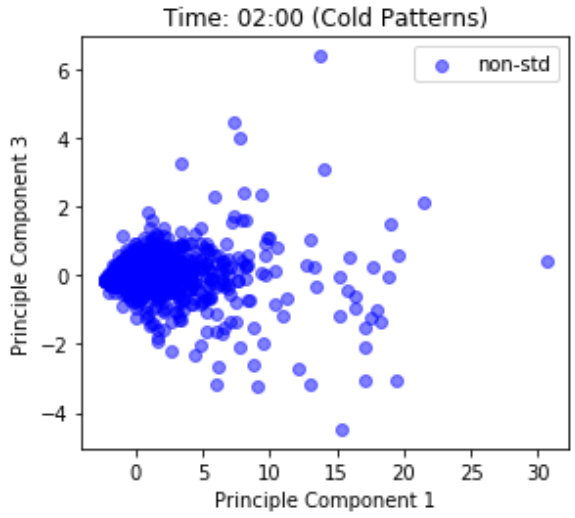
Different from the size-and-variable both standardization in paper 2, we care about the size effect. So we do not perform size standardization. Also in paper2, the author worried about the correlated problems, which wouldn't appear if they didn't use the no-size-scaled PCA to perform K-mean clustering. 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.

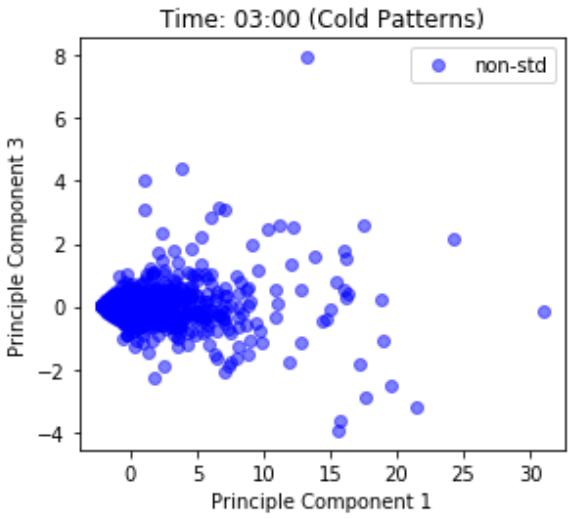
Component 1 and Component 3 relationship is shown below

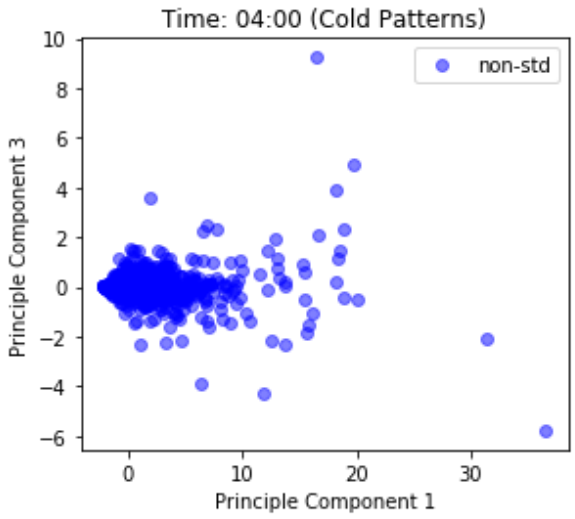
In [24]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

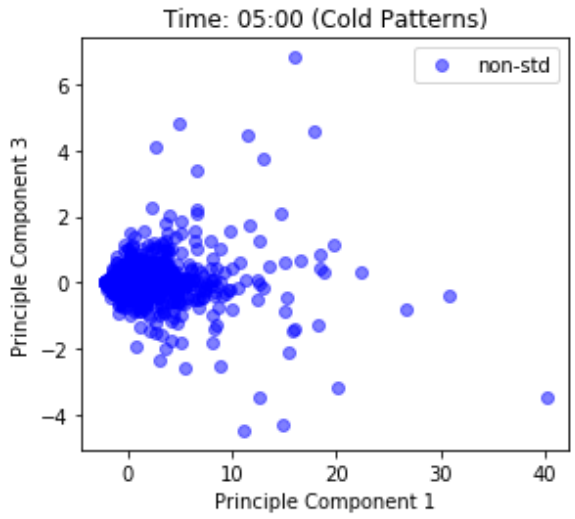

Time: 00:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

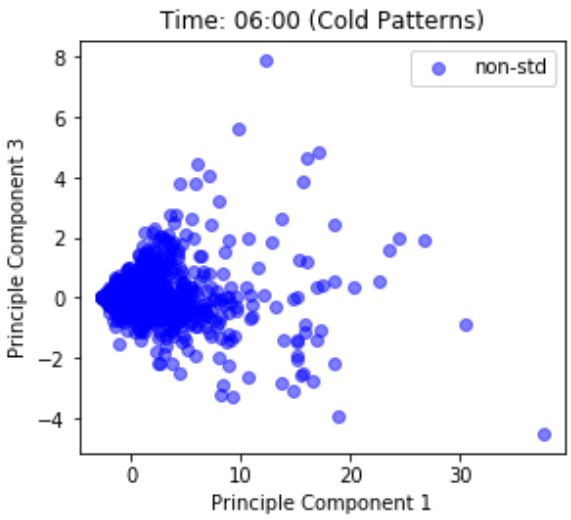

Time: 01:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

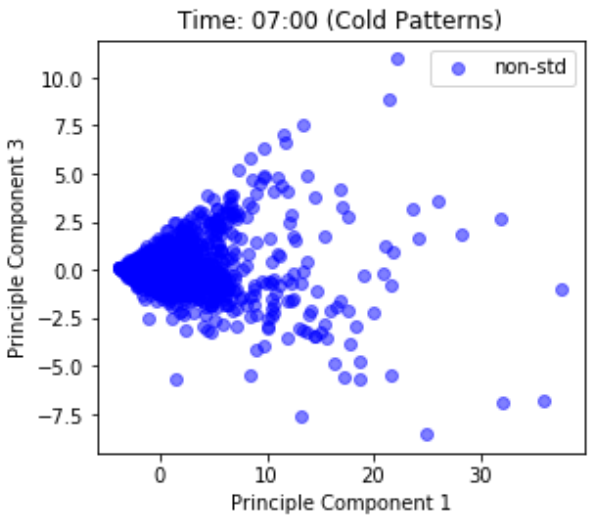

Time: 02:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

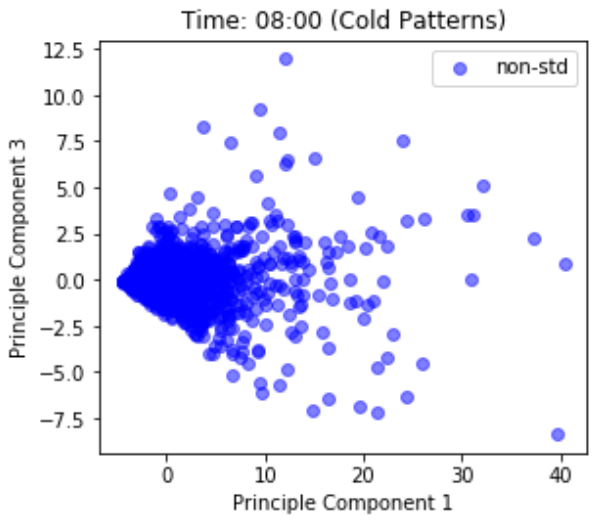

Time: 03:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

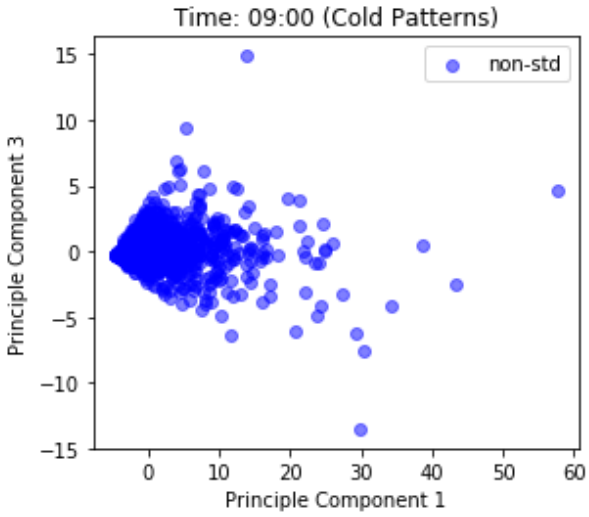

Time: 04:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

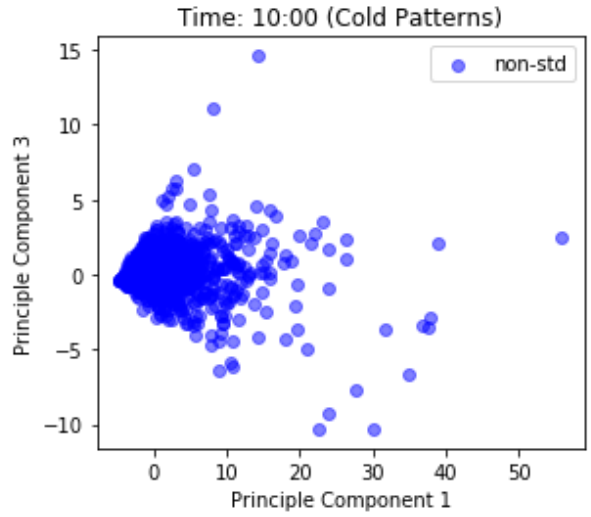

Time: 05:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

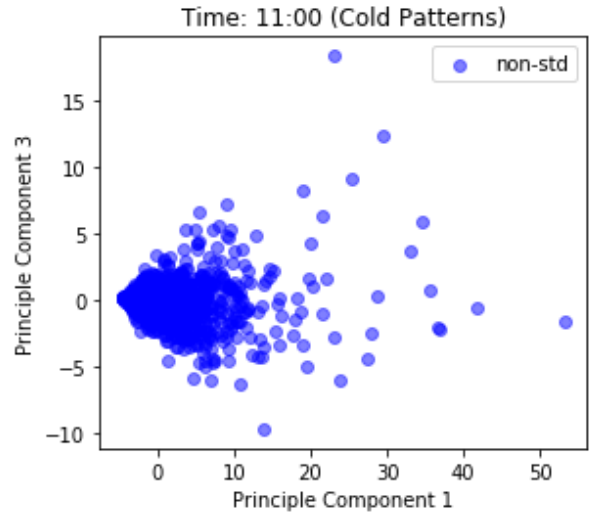

Time: 06:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

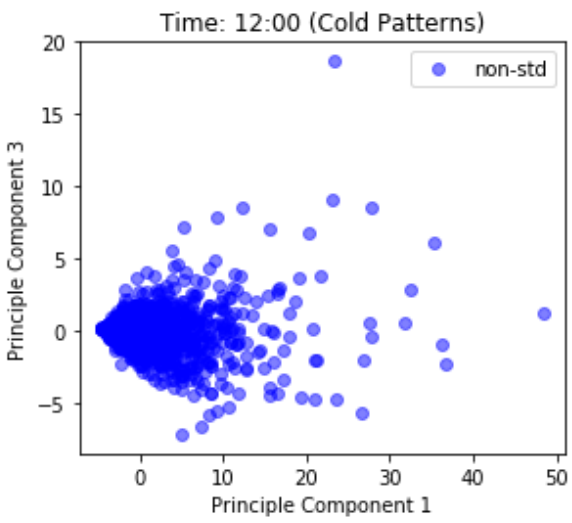

Time: 07:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

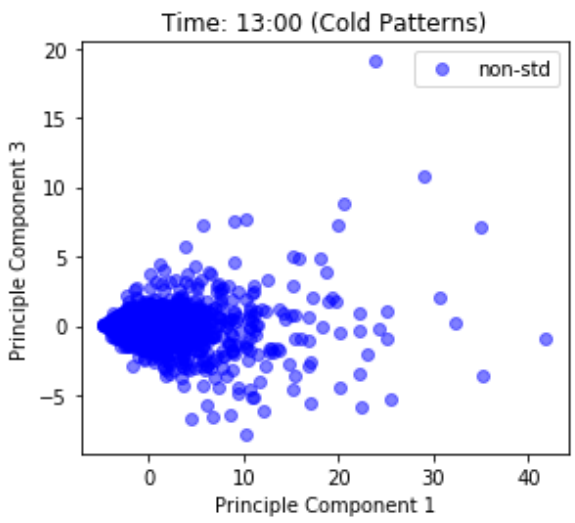

Time: 08:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

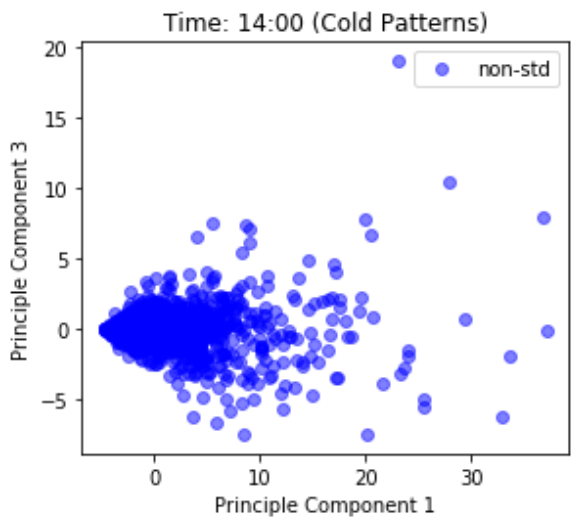

Time: 09:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

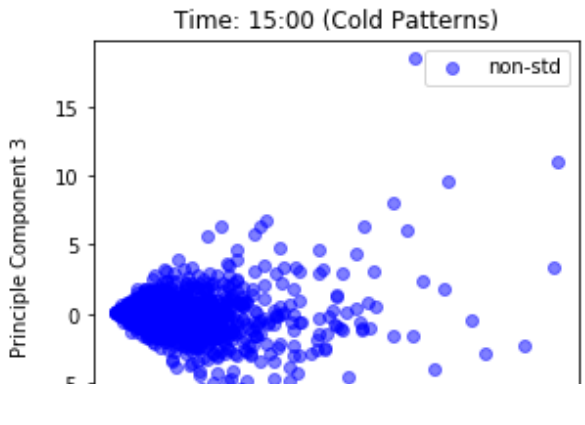

Time: 10:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

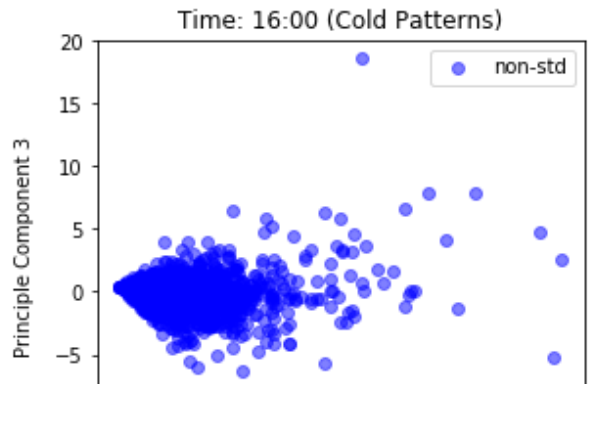

Time: 11:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

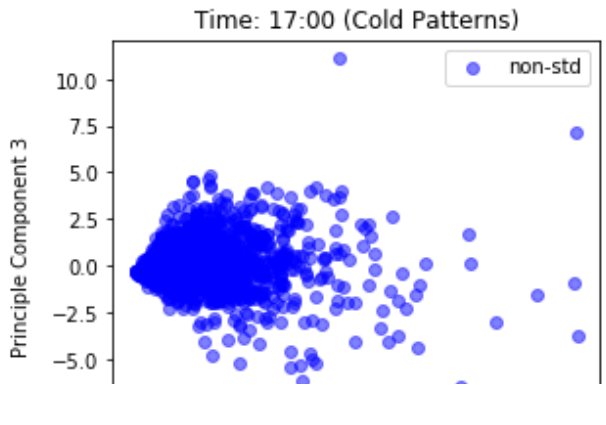

Time: 12:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

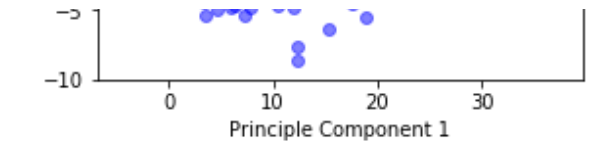

Time: 13:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

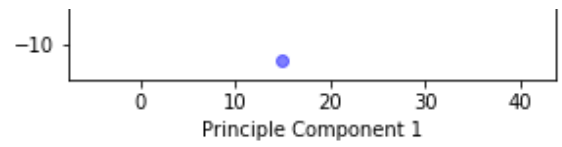

Time: 14:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

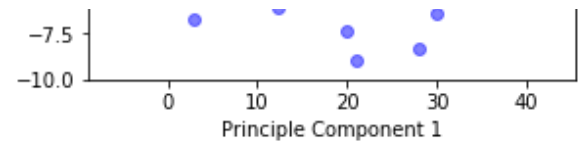

Time: 15:00 (Cold Patterns)
non-std
Principle Component 3

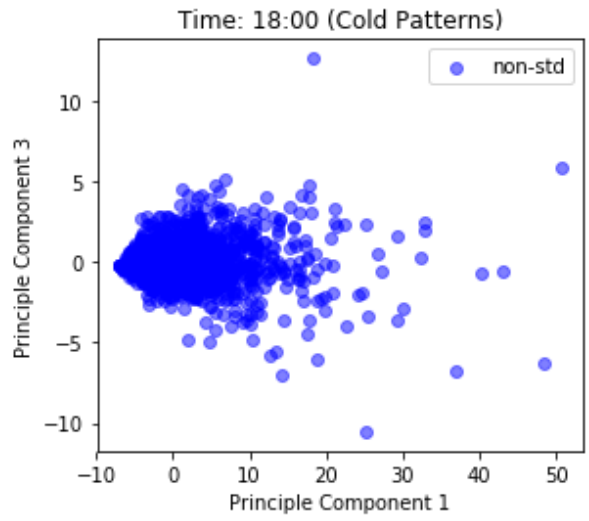

Time: 16:00 (Cold Patterns)
non-std
Principle Component 3

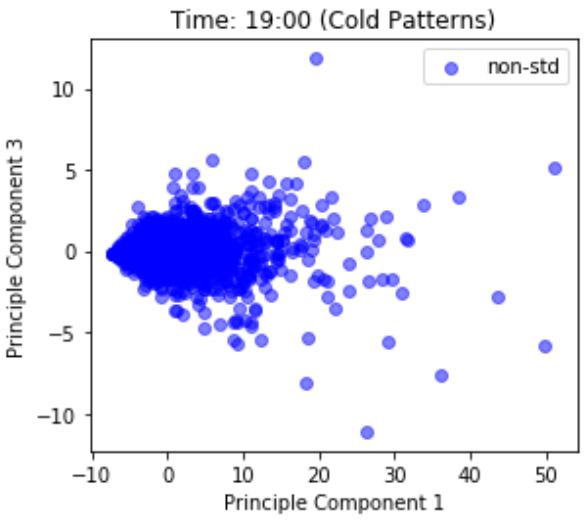

Time: 17:00 (Cold Patterns)
non-std
Principle Component 3

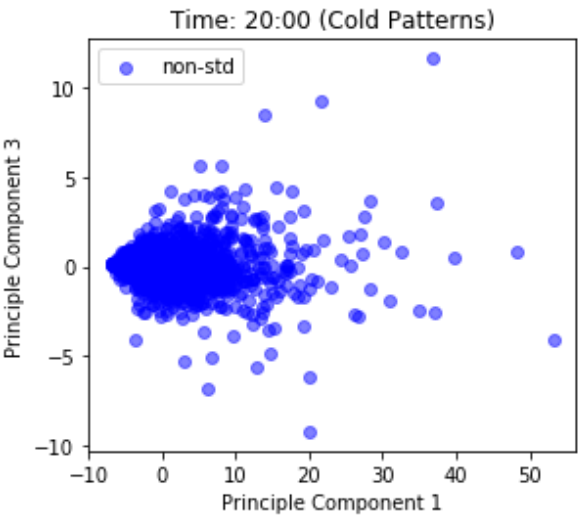

Principle Component 1

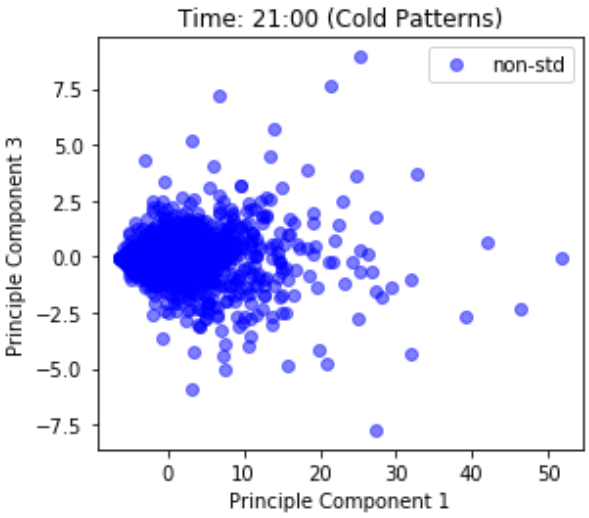

Principle Component 1

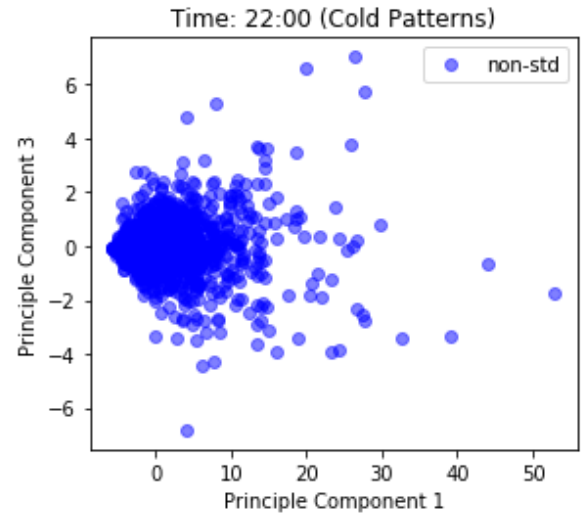

Principle Component 1

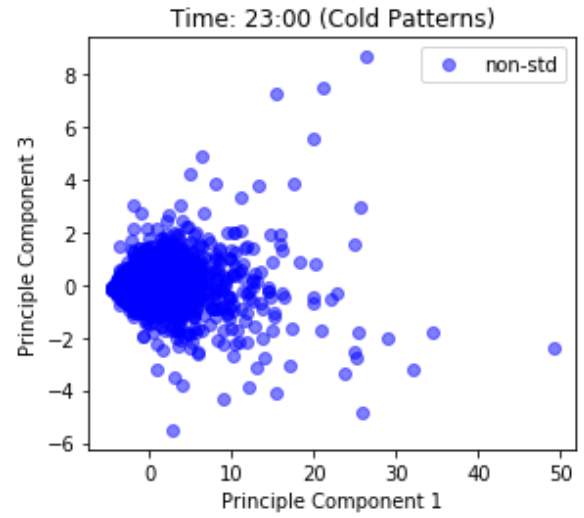

Time: 18:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1


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

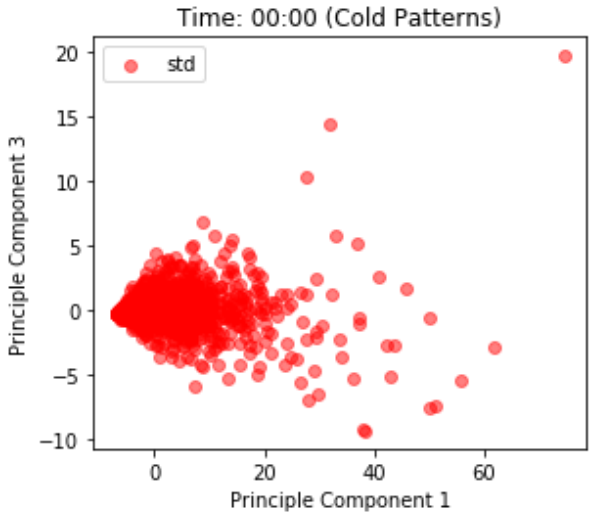

Time: 20:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

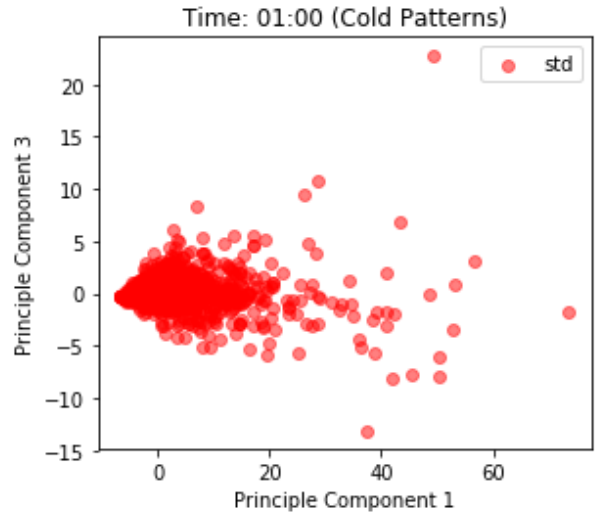

Time: 21:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

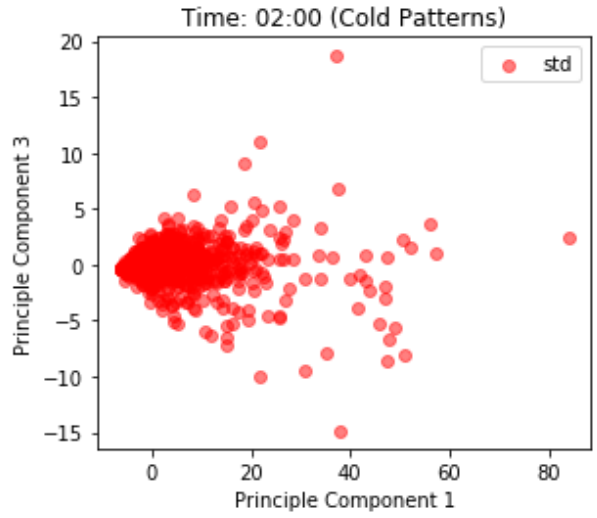

Time: 22:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

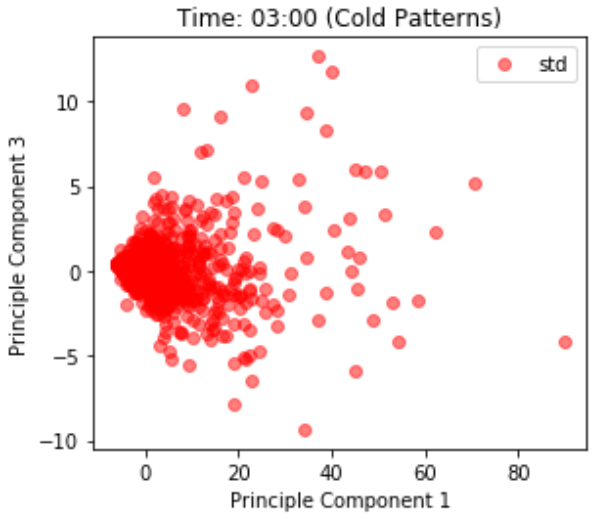

Time: 23:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 1

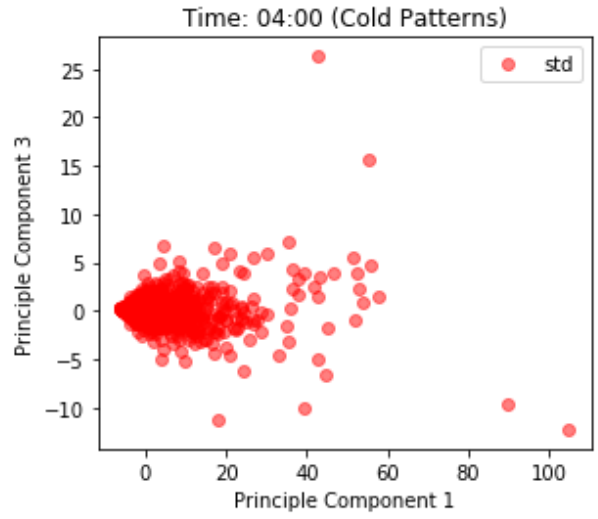
In [25]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 1')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

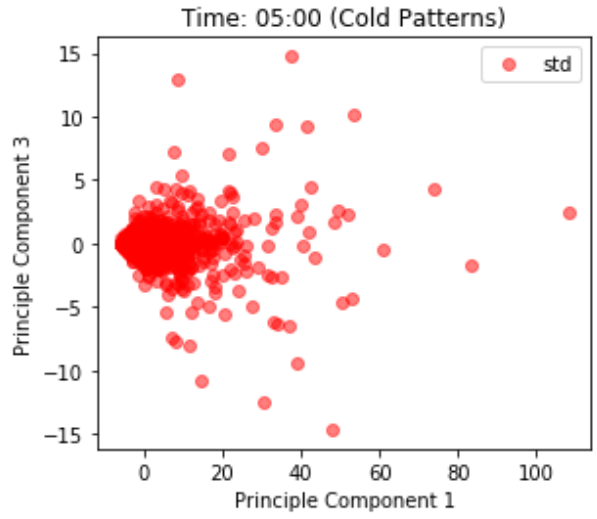

Time: 00:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

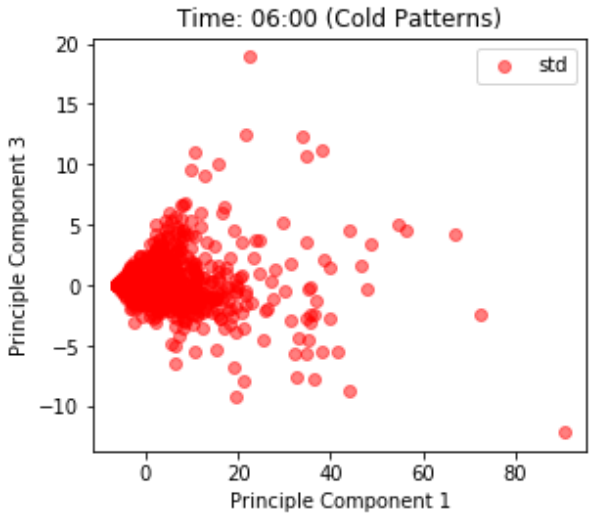

Time: 01:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

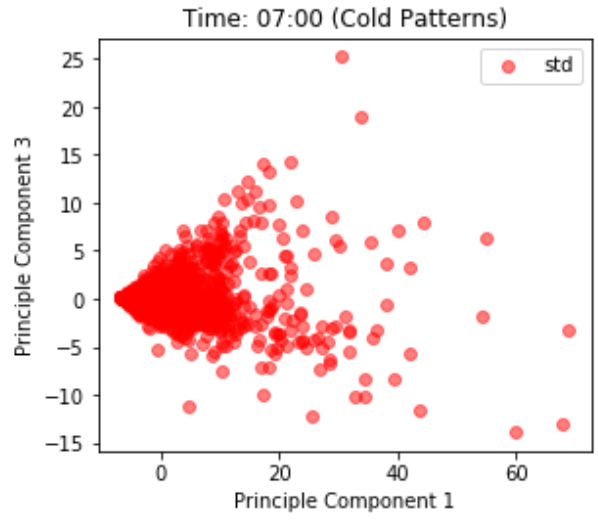

Time: 02:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

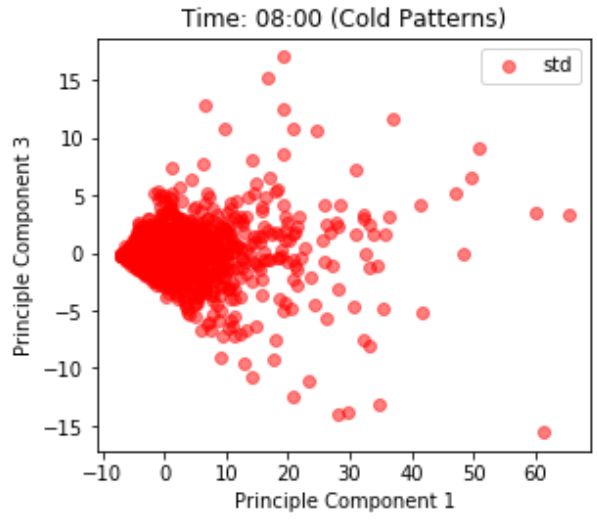

Time: 03:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

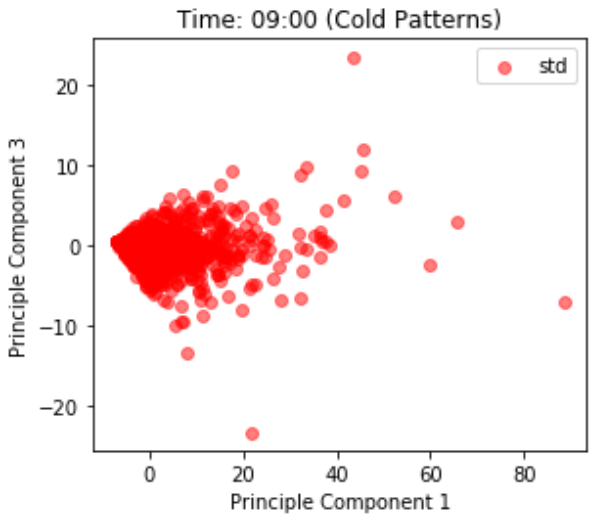

Time: 04:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

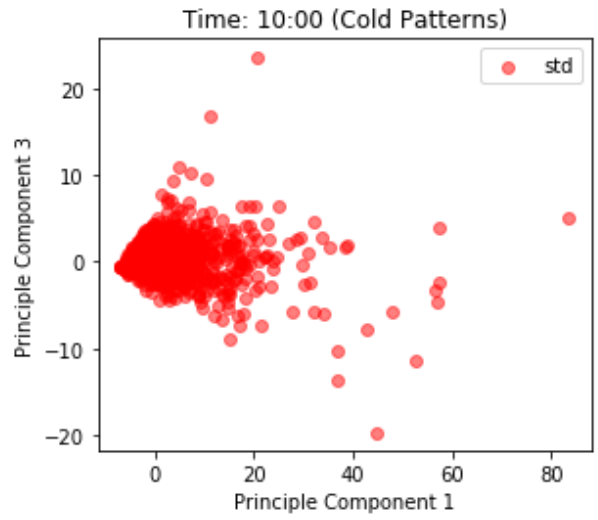

Time: 05:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

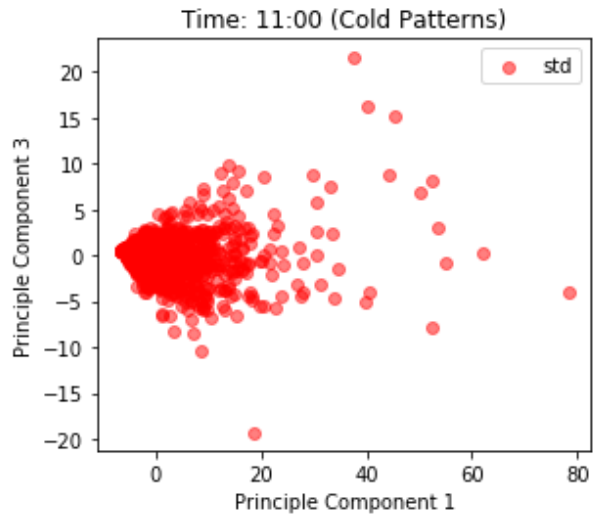

Time: 06:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

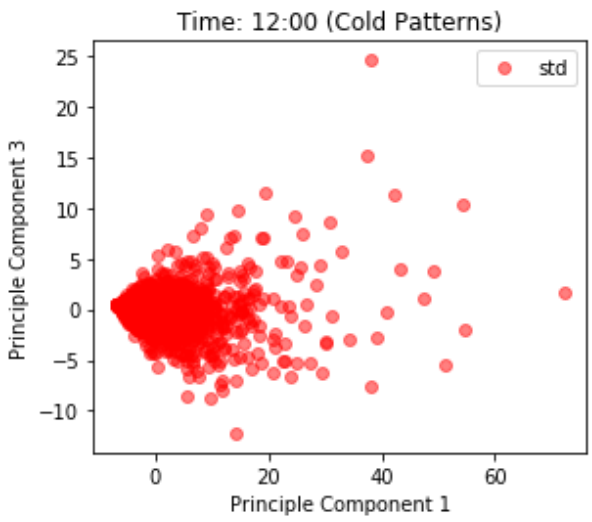

Time: 07:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

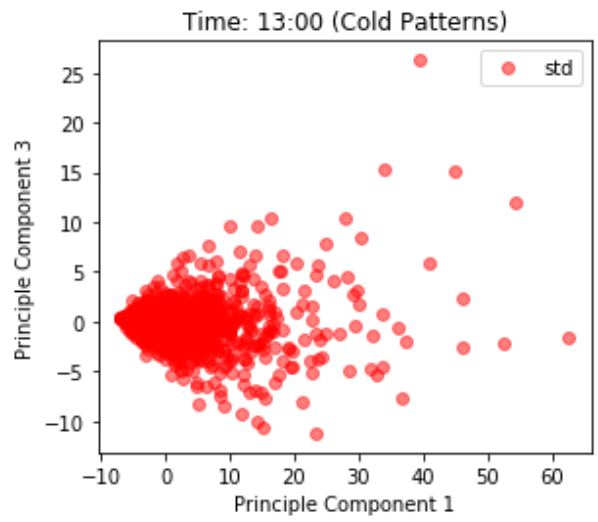

Time: 08:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

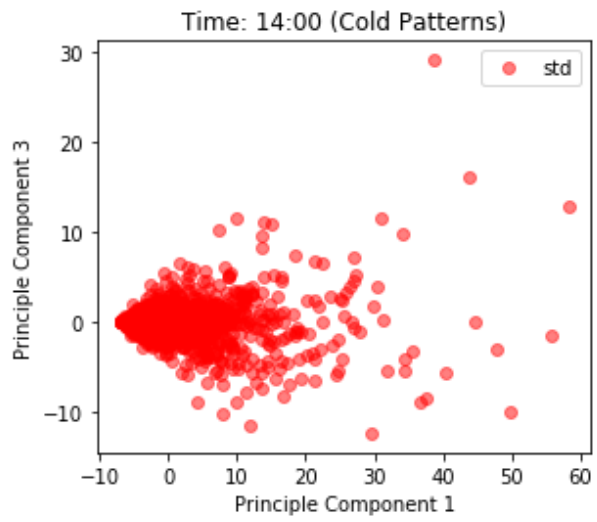

Time: 09:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

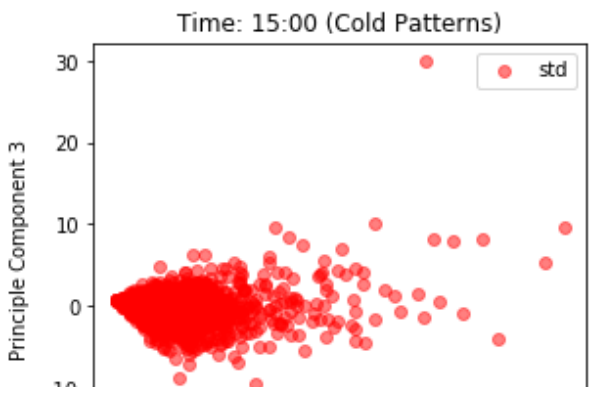

Time: 10:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

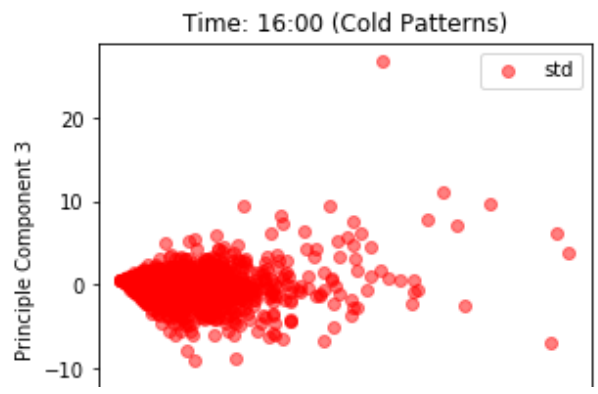

Time: 11:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

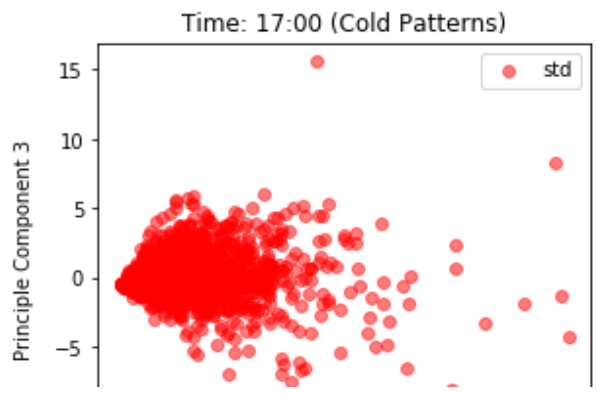

Time: 12:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1

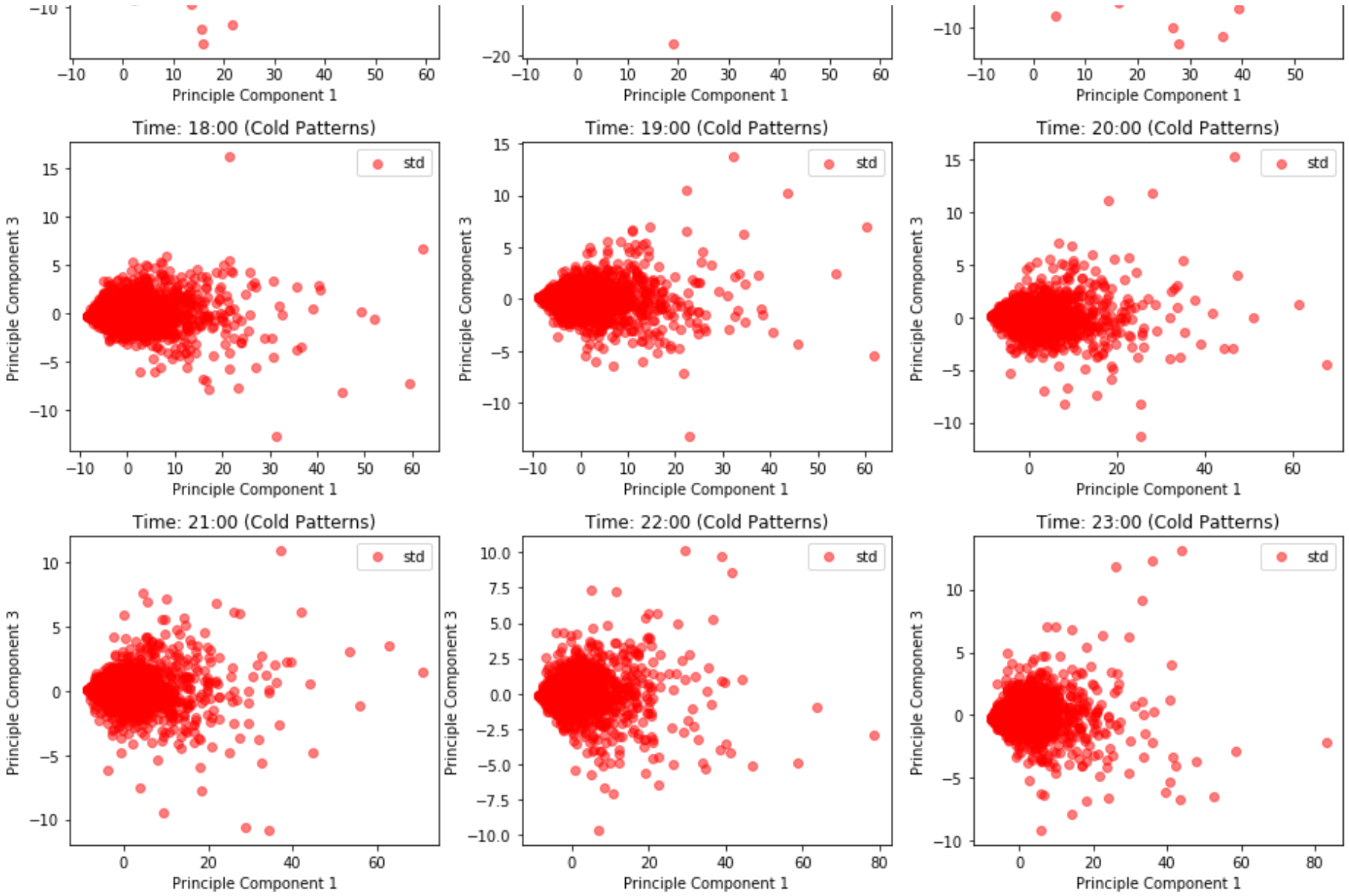

Time: 13:00 (Cold Patterns)
std
Principle Component 3
Principle Component 1


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


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

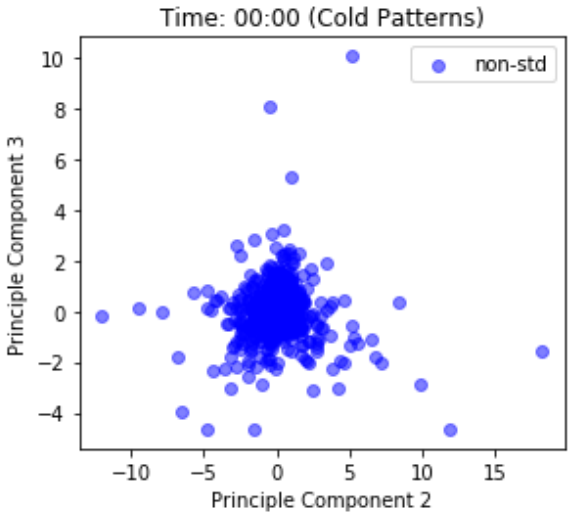

Time: 16:00 (Cold Patterns)
std
Principle Component 3

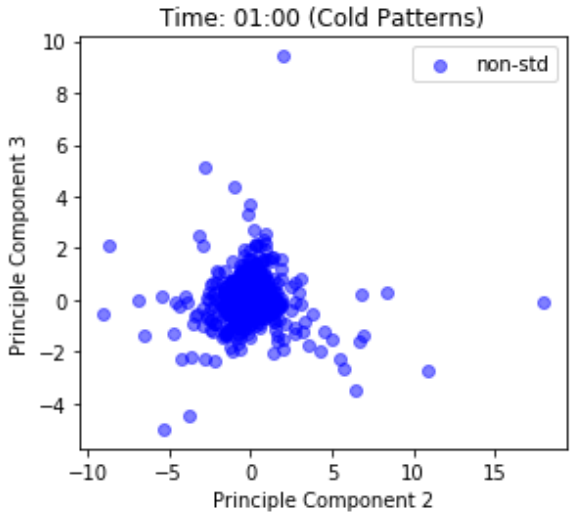

Time: 17:00 (Cold Patterns)
std
Principle Component 3

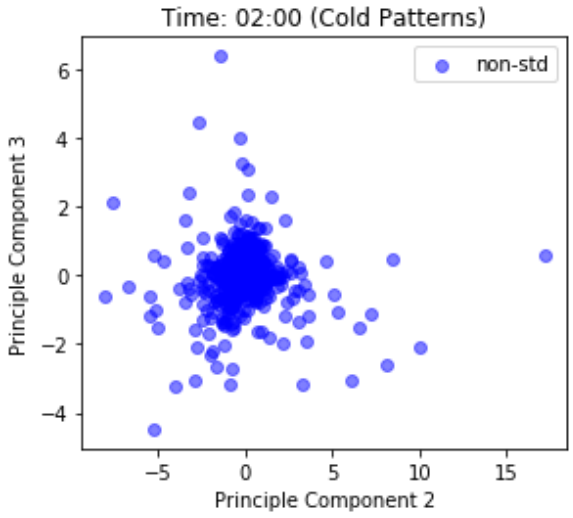


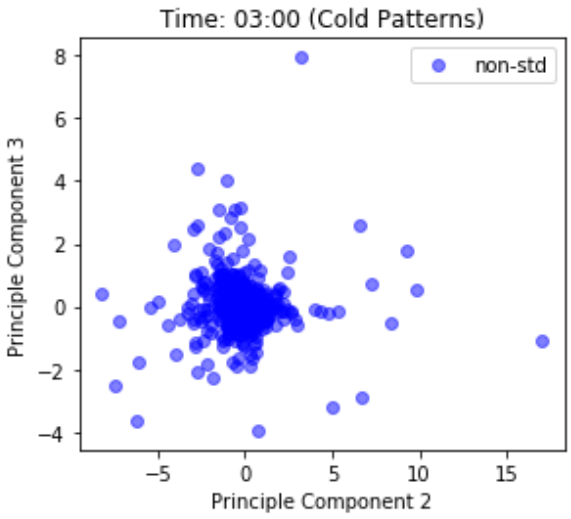
Component 2 and Component 3 relationship is shown below

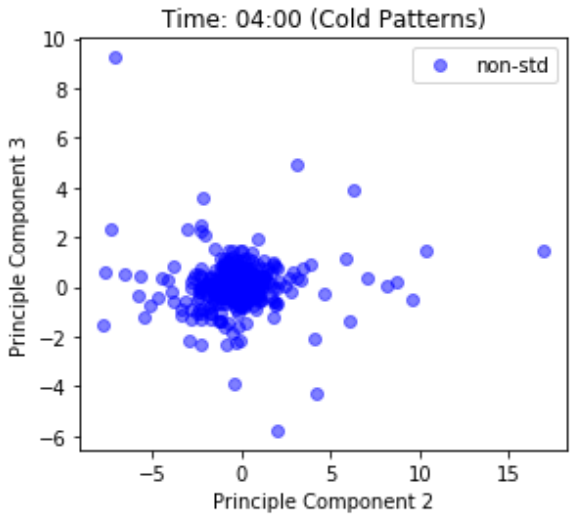
In [26]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

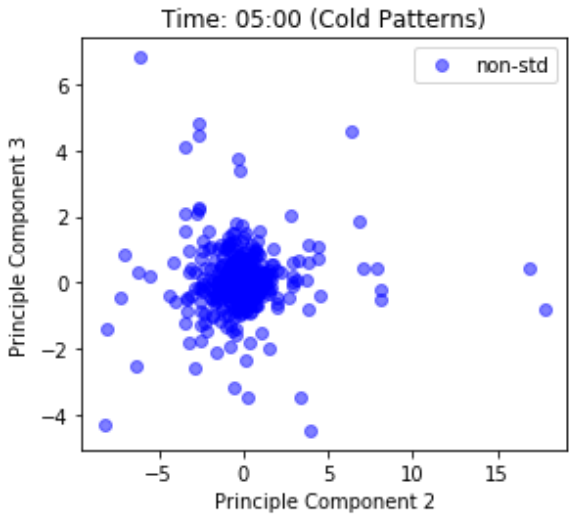

Time: 00:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

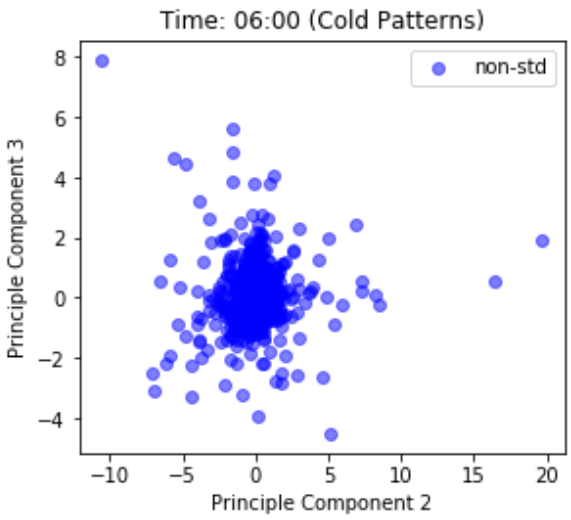

Time: 01:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

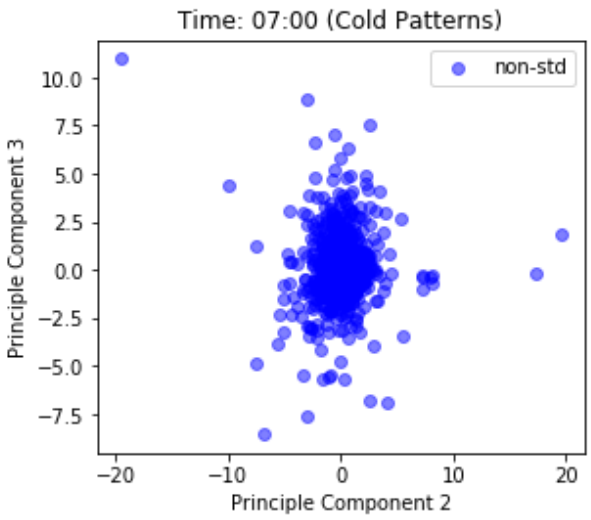

Time: 02:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

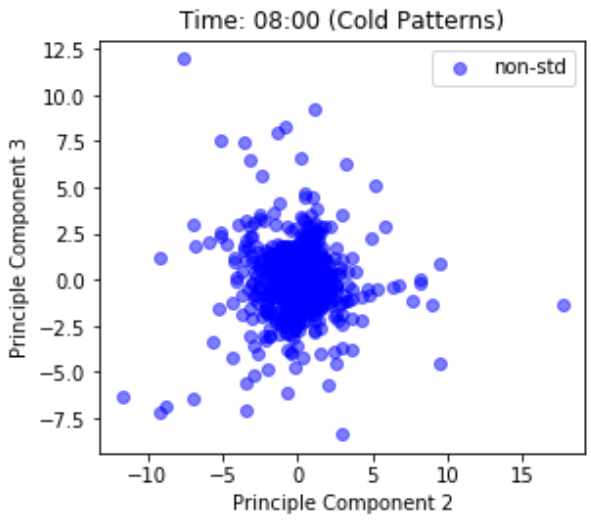

Time: 03:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

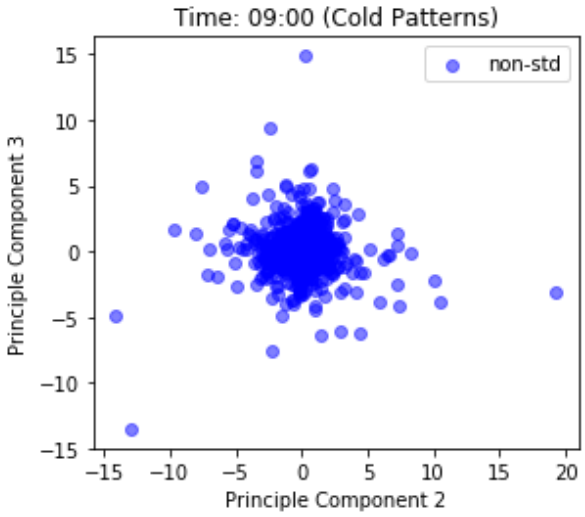

Time: 04:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

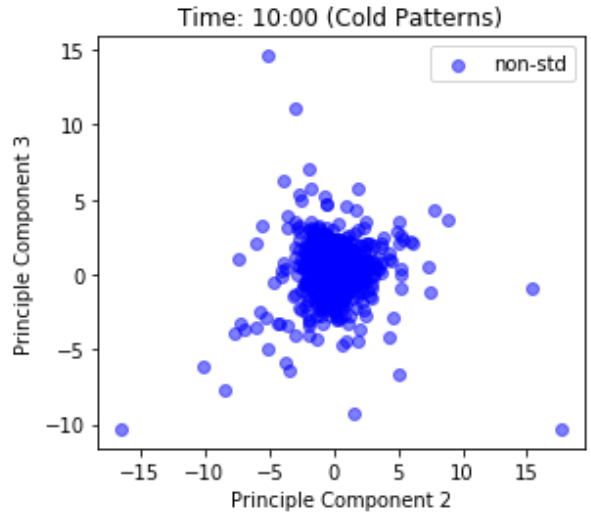

Time: 05:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

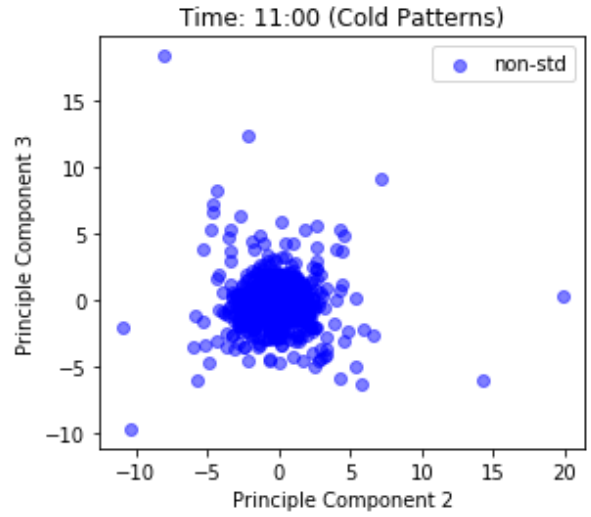

Time: 06:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

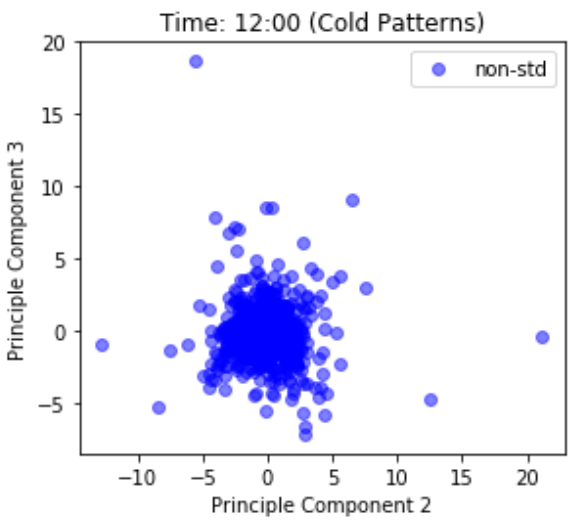

Time: 07:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

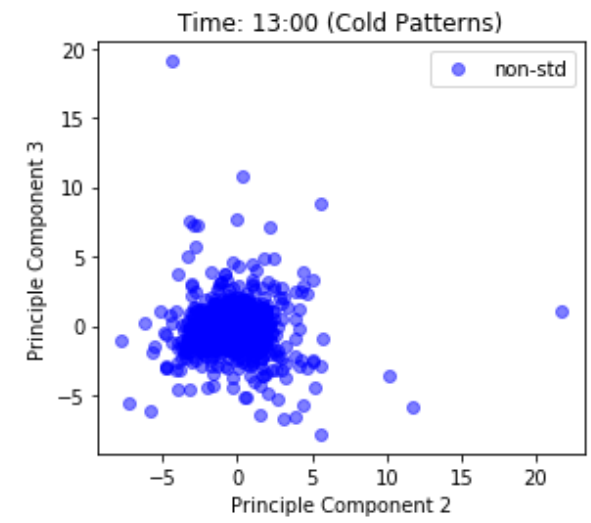

Time: 08:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

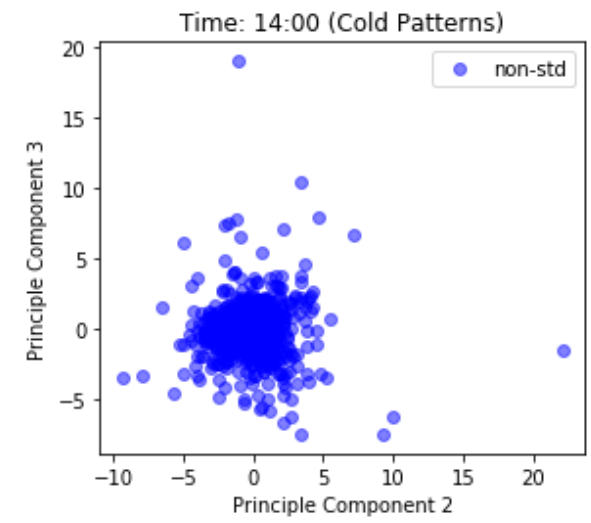

Time: 09:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

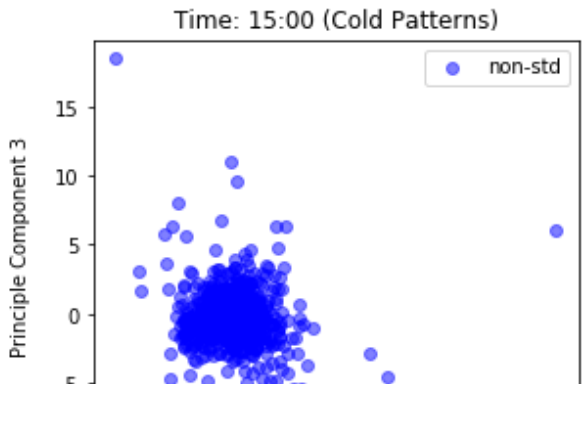

Time: 10:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

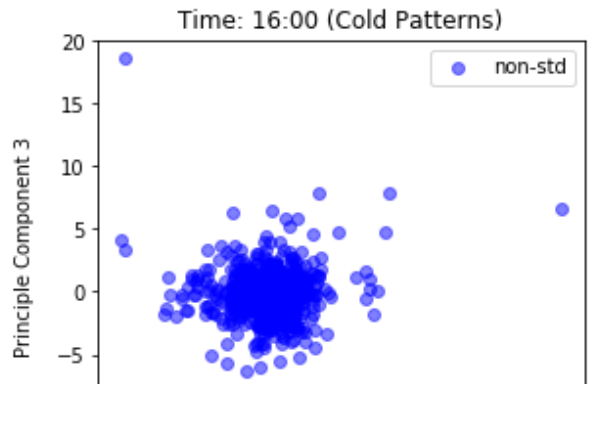

Time: 11:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

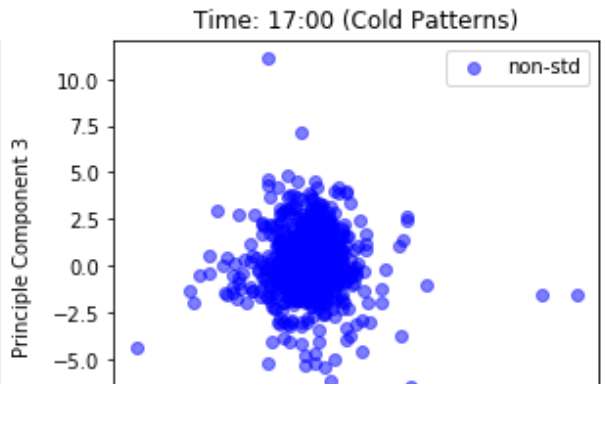

Time: 12:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

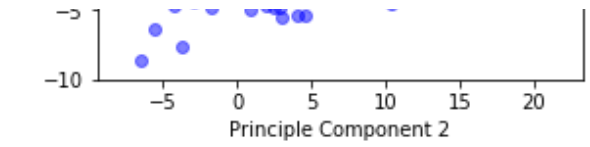

Time: 13:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

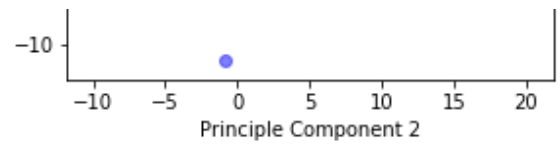

Time: 14:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

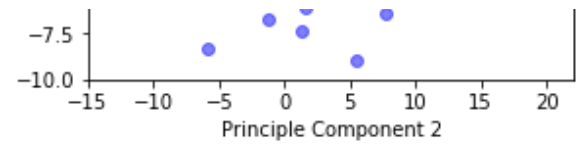

Time: 15:00 (Cold Patterns)
non-std
Principle Component 3

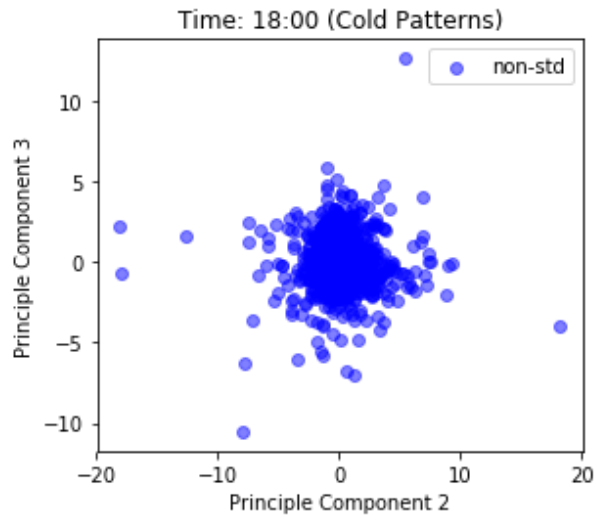

Time: 16:00 (Cold Patterns)
non-std
Principle Component 3

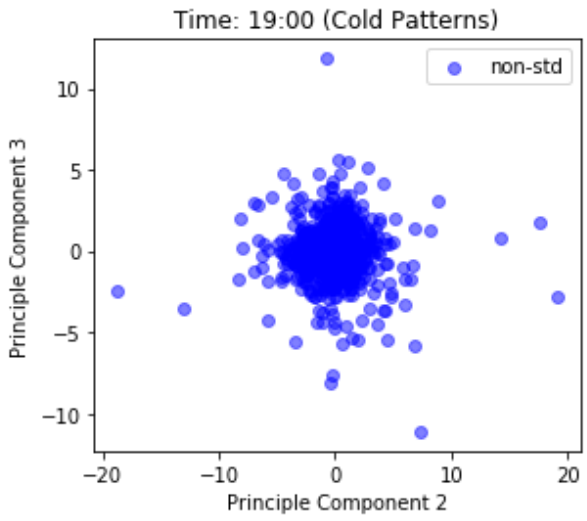

Time: 17:00 (Cold Patterns)
non-std
Principle Component 3

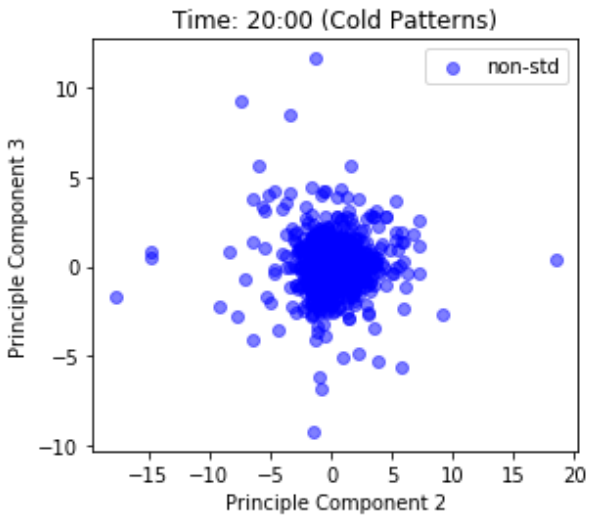

Principle Component 2

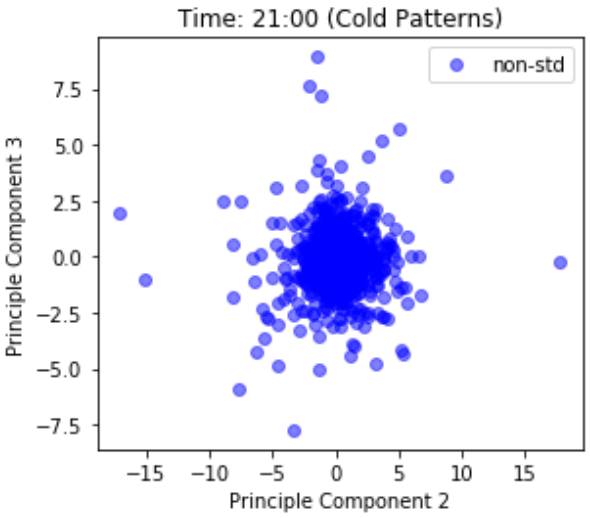

Principle Component 2

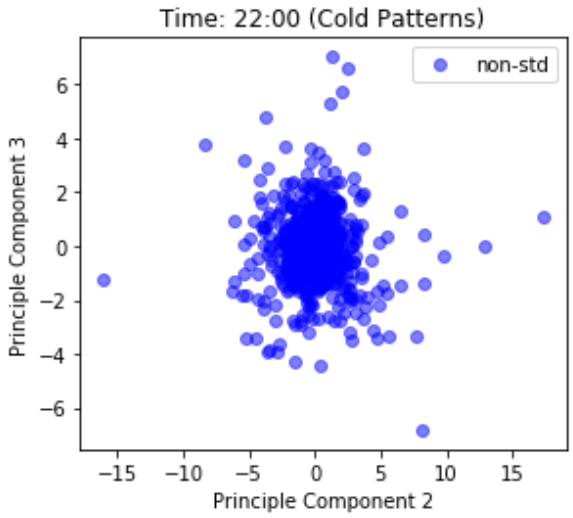

Principle Component 2

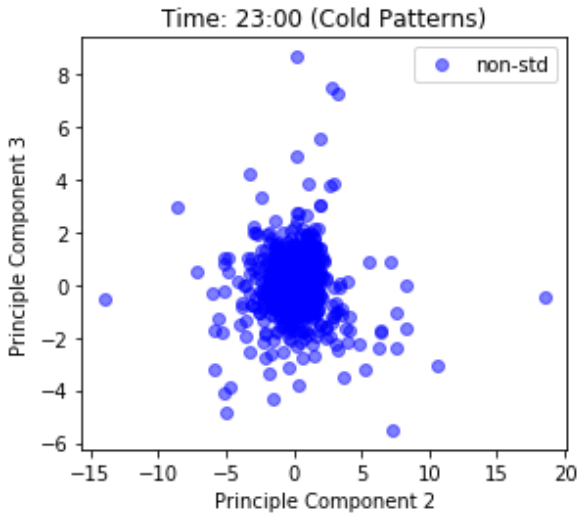

Time: 18:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2


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

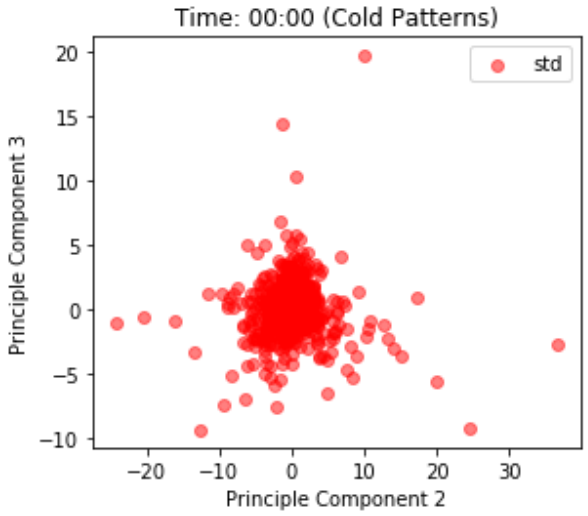

Time: 20:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

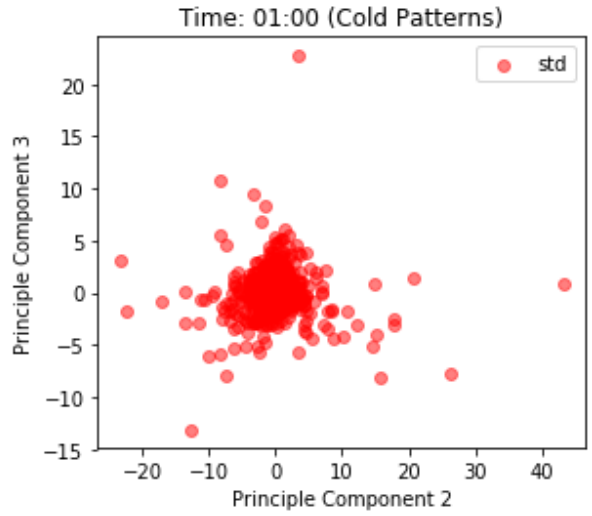

Time: 21:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

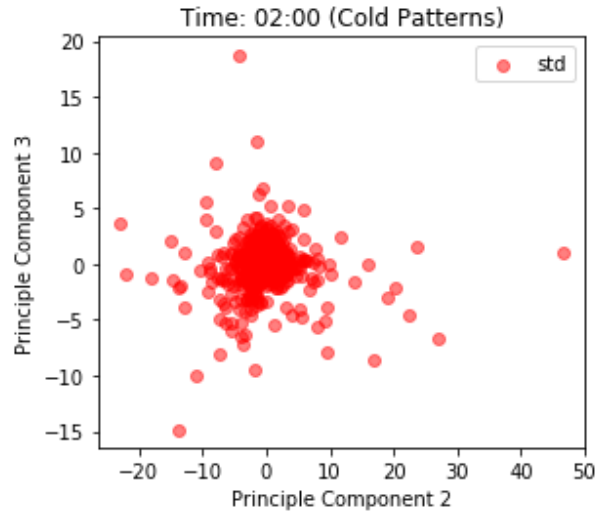

Time: 22:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

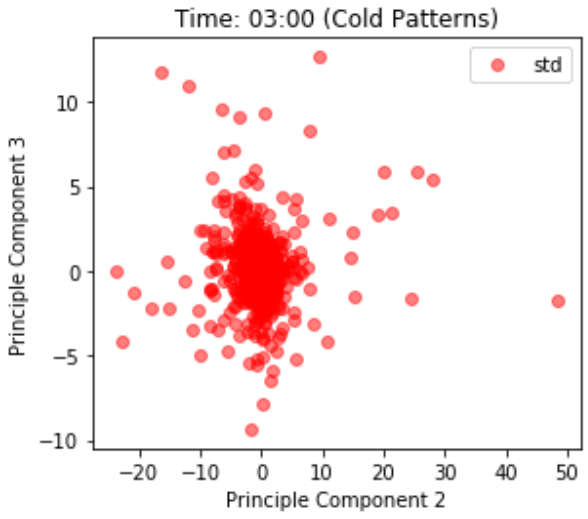

Time: 23:00 (Cold Patterns)
non-std
Principle Component 3
Principle Component 2

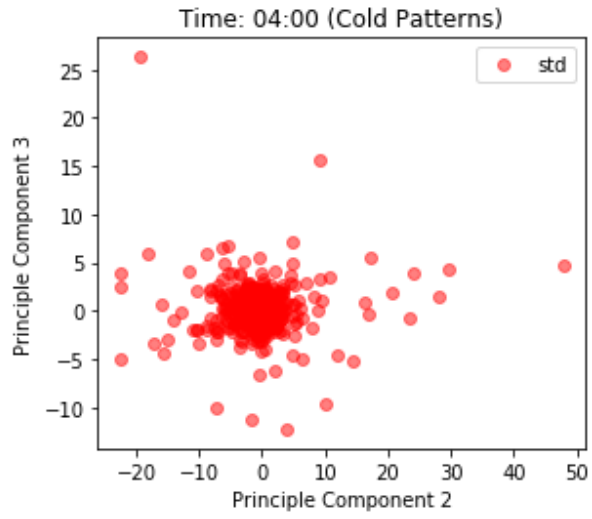
In [27]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Principle Component 2')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

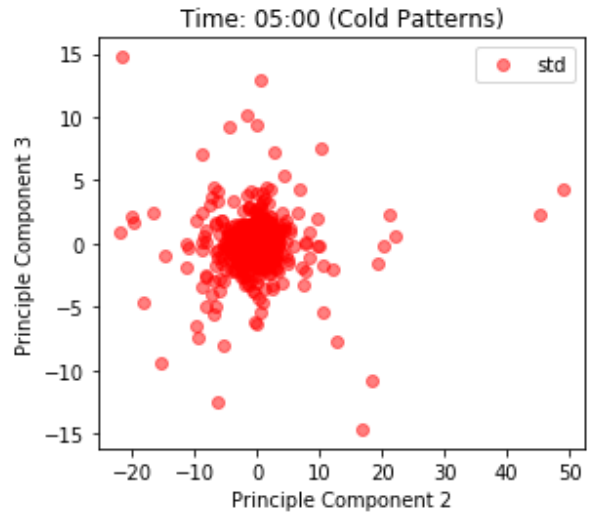

Time: 00:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

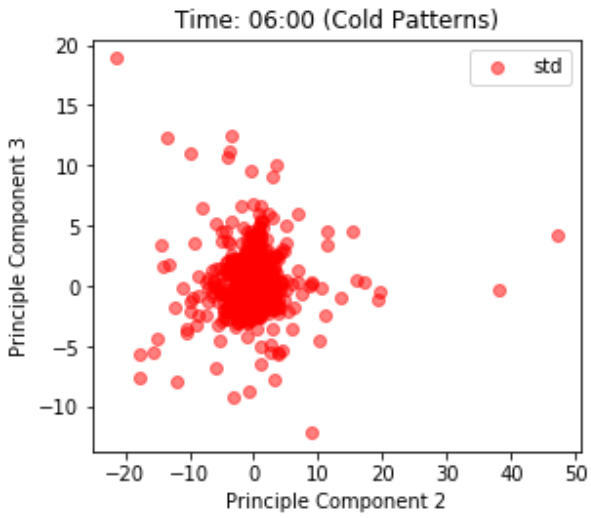

Time: 01:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

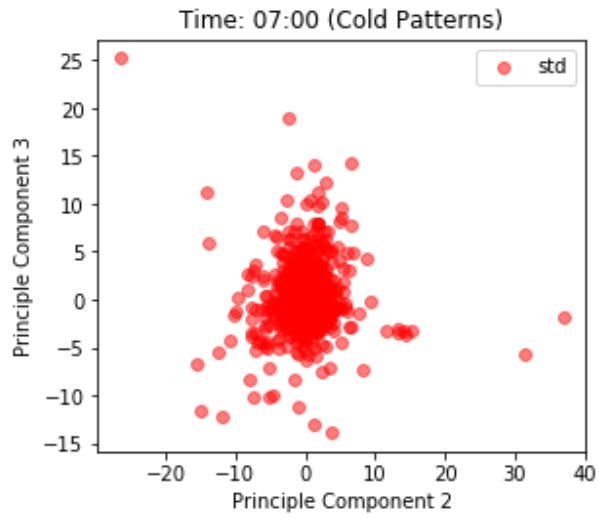

Time: 02:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

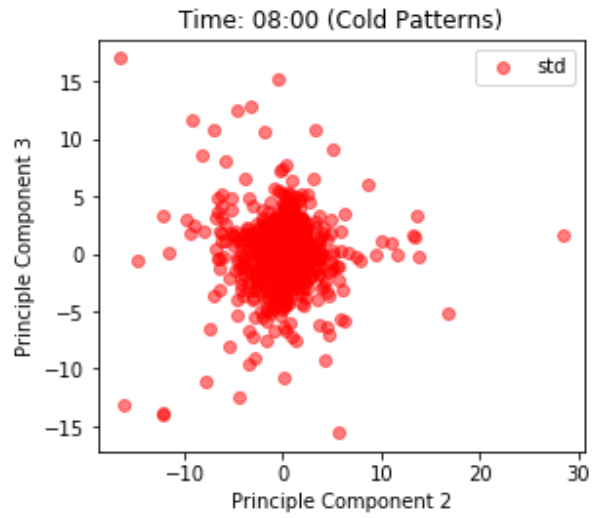

Time: 03:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

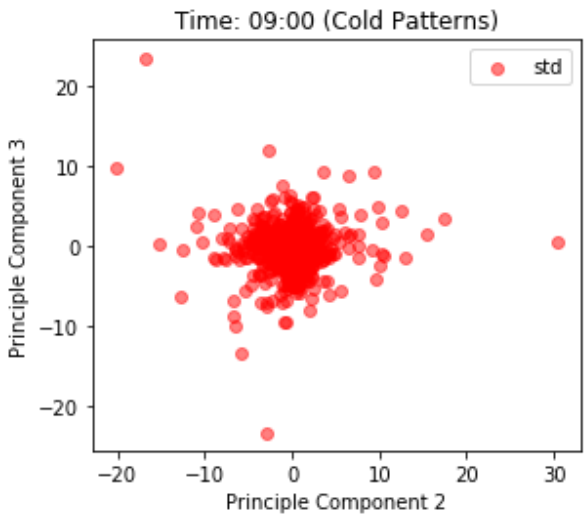

Time: 04:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

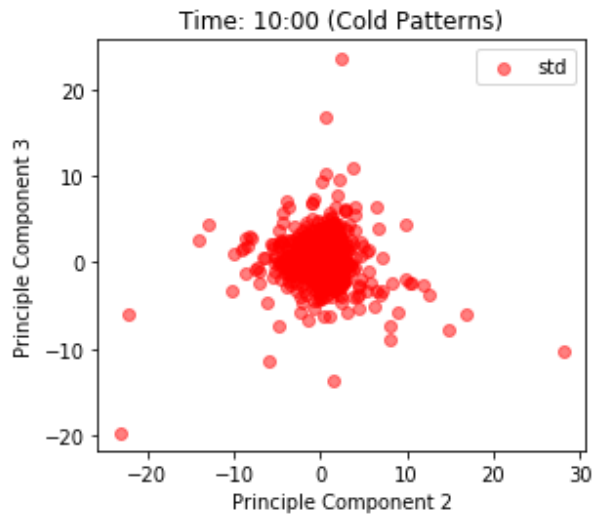

Time: 05:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

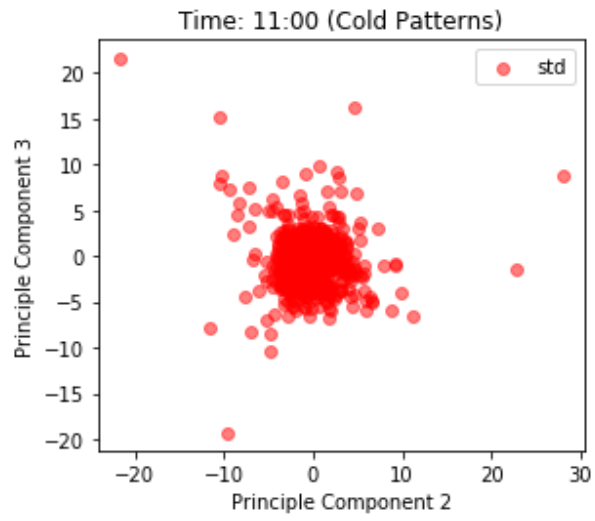

Time: 06:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

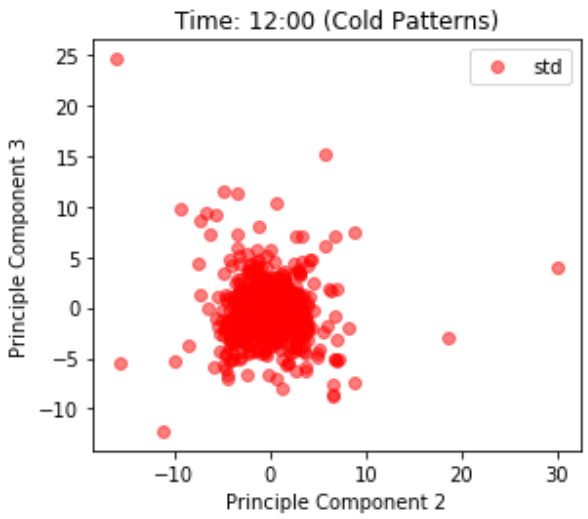

Time: 07:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

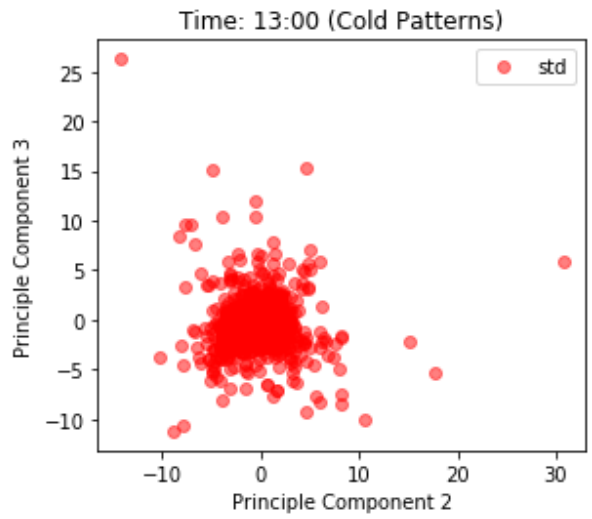

Time: 08:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

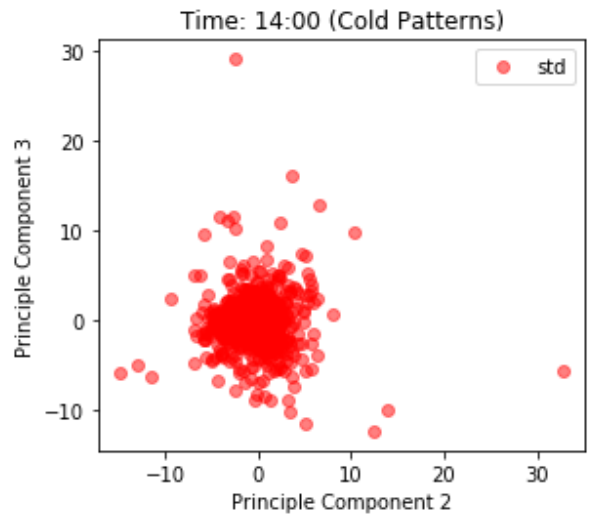

Time: 09:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

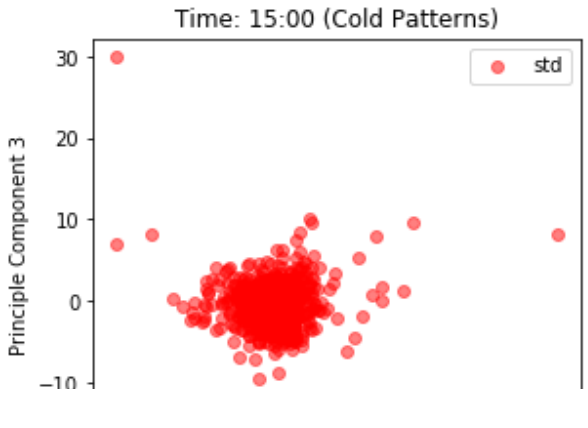

Time: 10:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

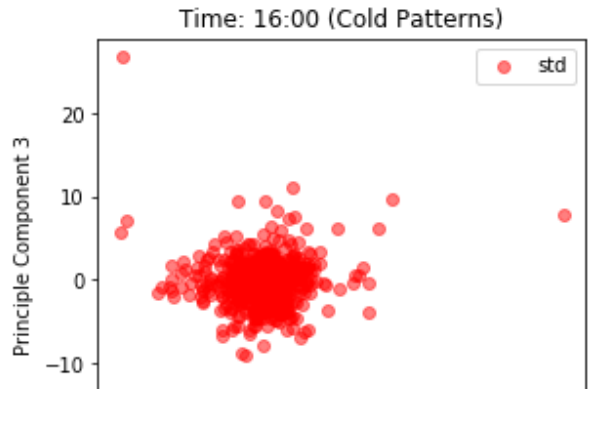

Time: 11:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

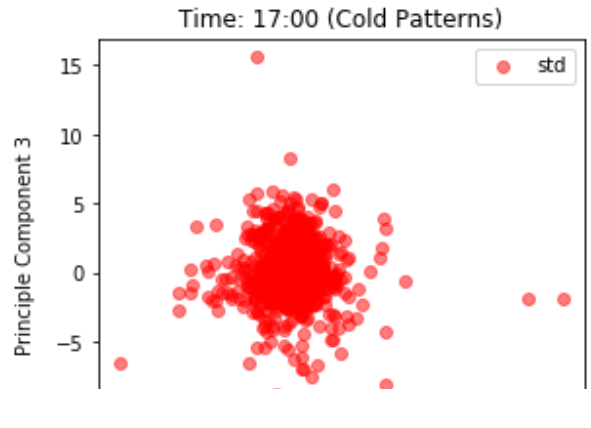

Time: 12:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2

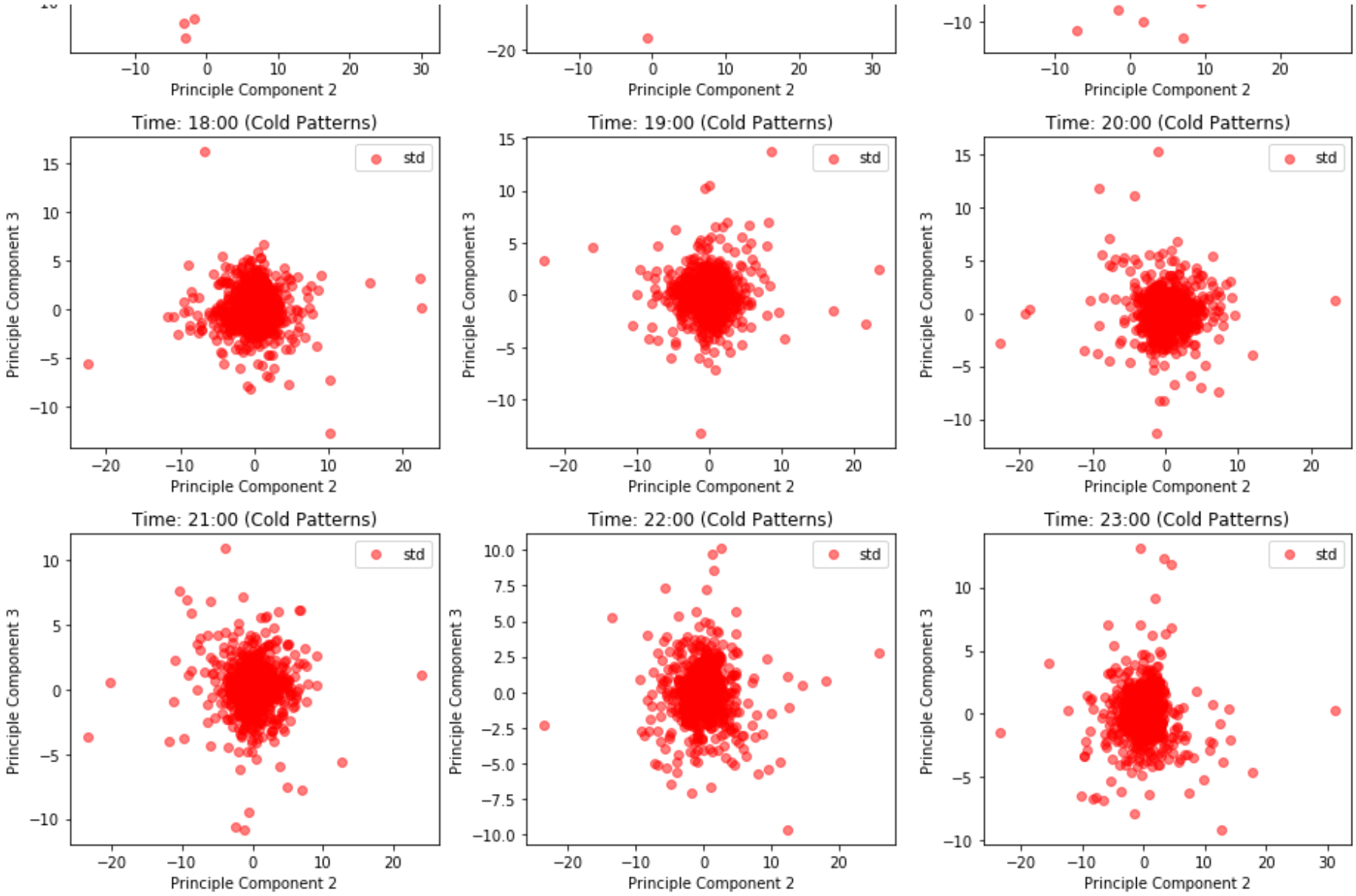

Time: 13:00 (Cold Patterns)
std
Principle Component 3
Principle Component 2


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


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


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


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

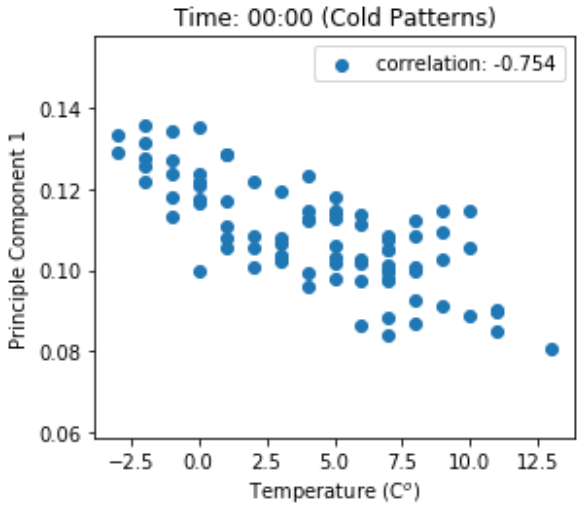


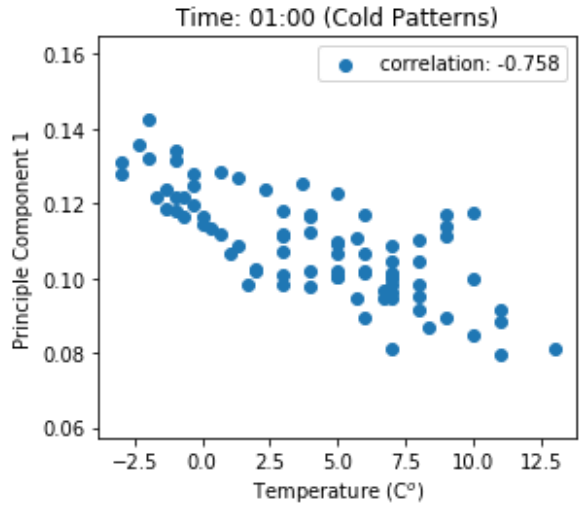
The above three types of plots show that std don't have significant advantage in this case, so we will use non-std in our later analysis.

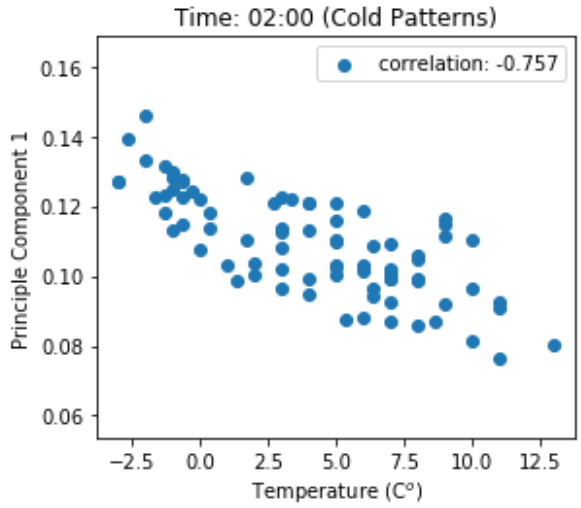
Let's try to interprete PCs' meaning:

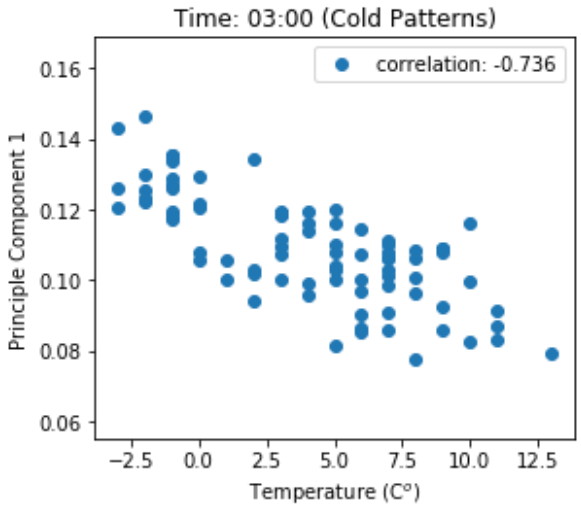
(1) PC1's coefficients vs Temperature:

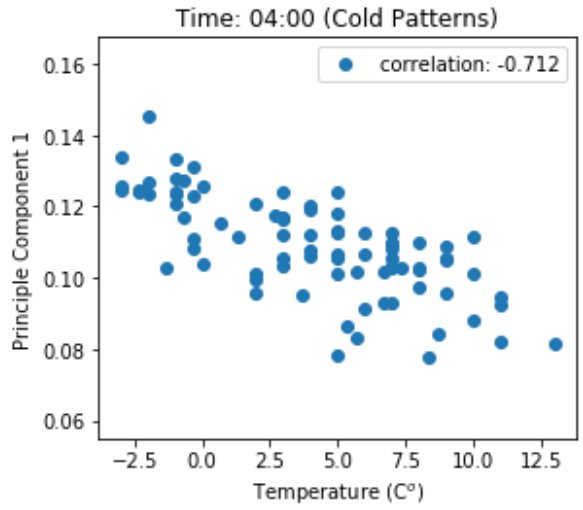
In [28]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # 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_cold.TempC[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')], pca_n
std.components_[0], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[0], df_wea1h_nf_cold
.TempC[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: 0' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 1')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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_cold.TempC[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')], pca_nstd.co
mponents_[0], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[0], df_wea1h_nf_cold.TempC
[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: ' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 1')
plt.tight_layout()
```

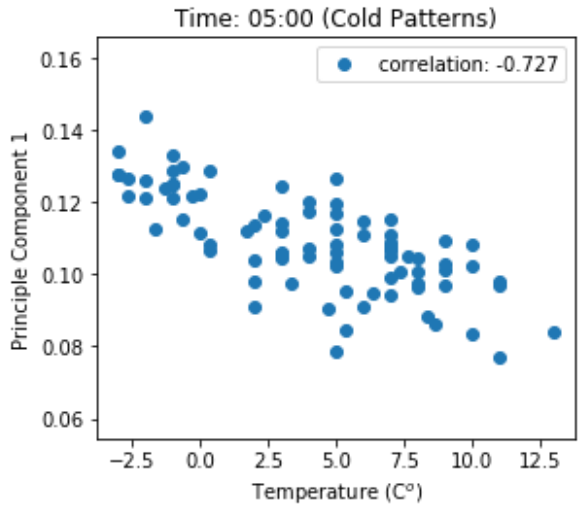

Time: 00:00 (Cold Patterns)
correlation: -0.754
Principle Component 1
Temperature (C°)

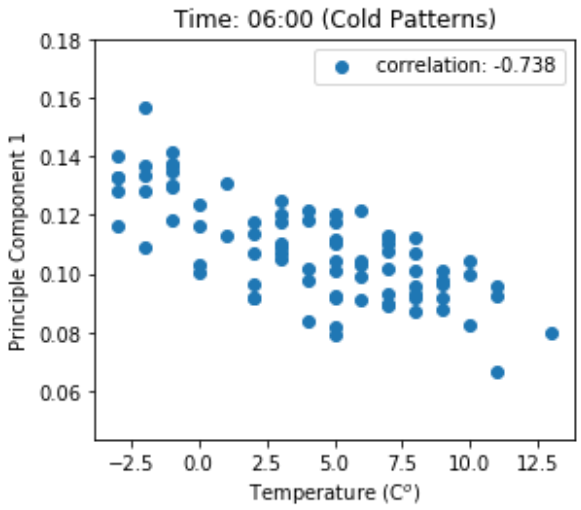

Time: 01:00 (Cold Patterns)
correlation: -0.758
Principle Component 1
Temperature (C°)

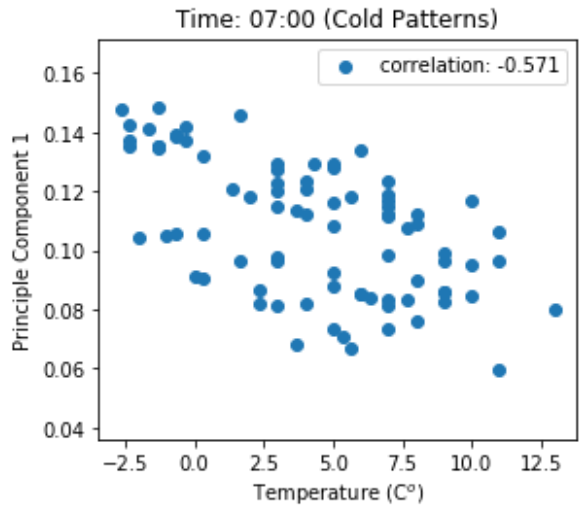

Time: 02:00 (Cold Patterns)
correlation: -0.757
Principle Component 1
Temperature (C°)

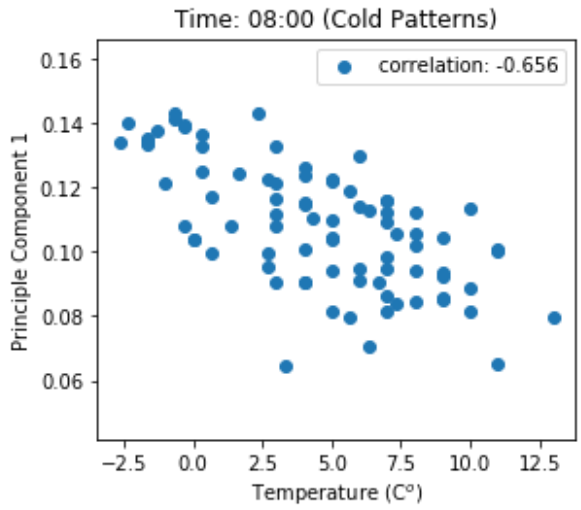

Time: 03:00 (Cold Patterns)
correlation: -0.736
Principle Component 1
Temperature (C°)

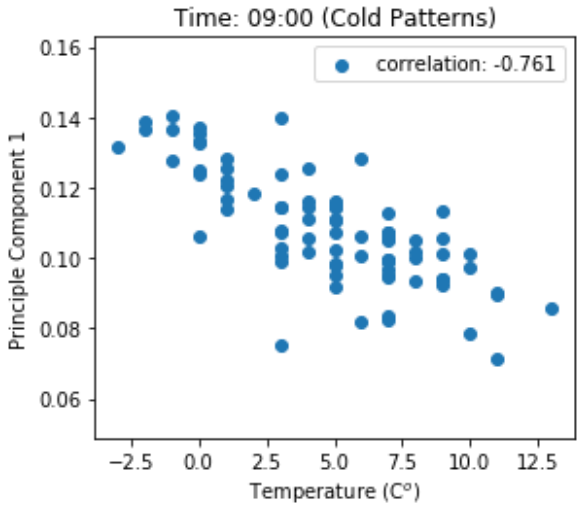

Time: 04:00 (Cold Patterns)
correlation: -0.712
Principle Component 1
Temperature (C°)

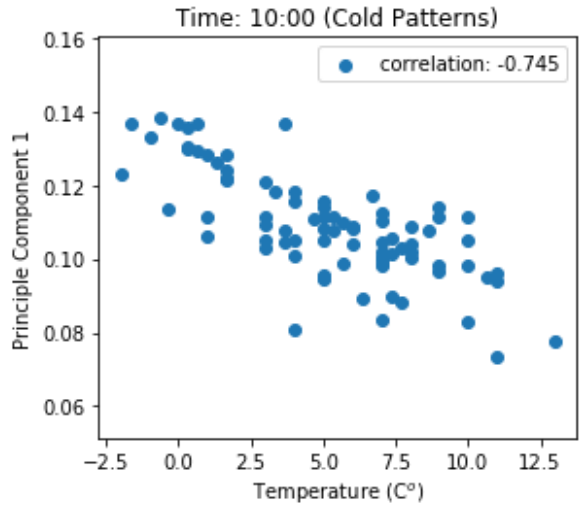

Time: 05:00 (Cold Patterns)
correlation: -0.727
Principle Component 1
Temperature (C°)

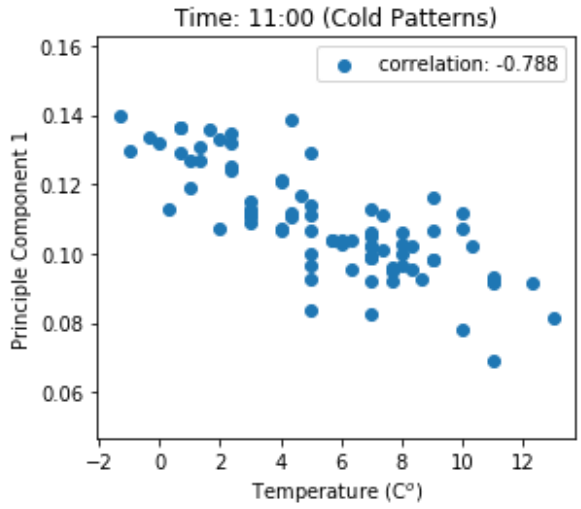

Time: 06:00 (Cold Patterns)
correlation: -0.738
Principle Component 1
Temperature (C°)

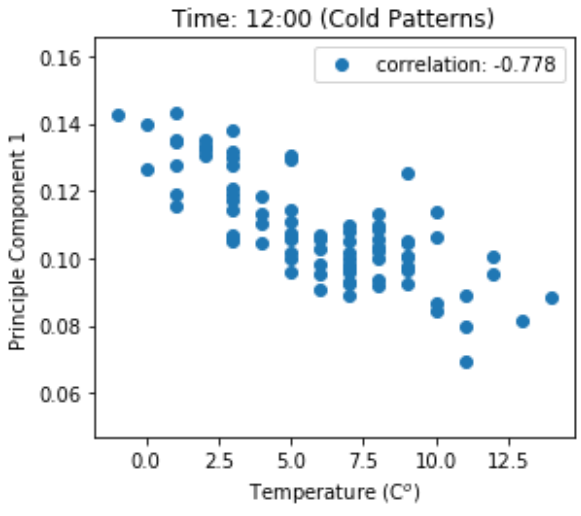

Time: 07:00 (Cold Patterns)
correlation: -0.571
Principle Component 1
Temperature (C°)

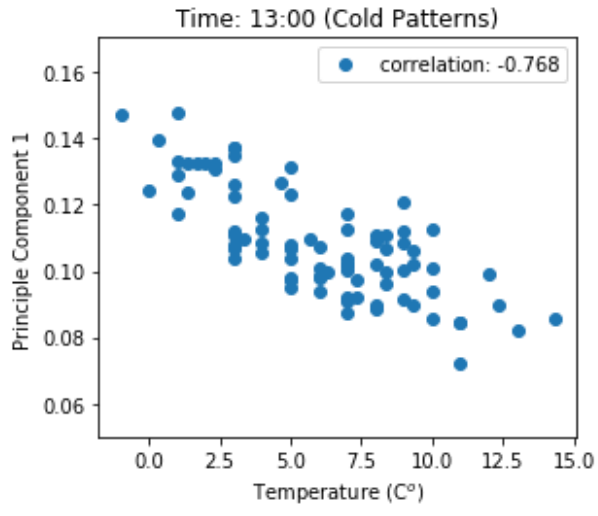

Time: 08:00 (Cold Patterns)
correlation: -0.656
Principle Component 1
Temperature (C°)

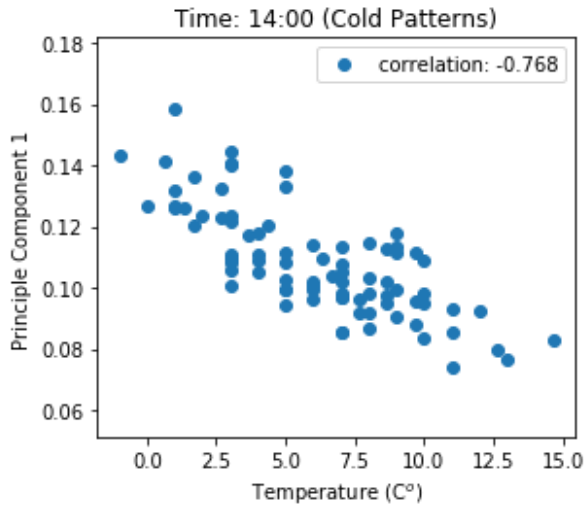

Time: 09:00 (Cold Patterns)
correlation: -0.761
Principle Component 1
Temperature (C°)

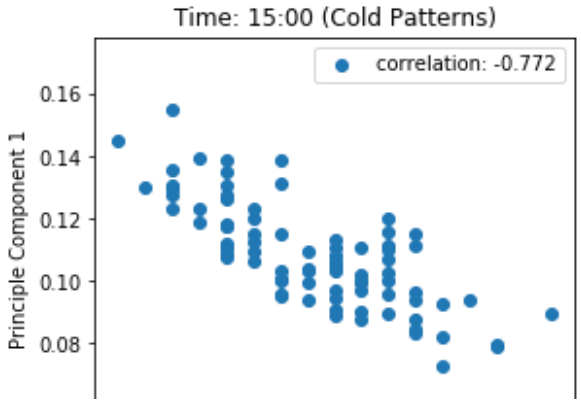

Time: 10:00 (Cold Patterns)
correlation: -0.745
Principle Component 1
Temperature (C°)

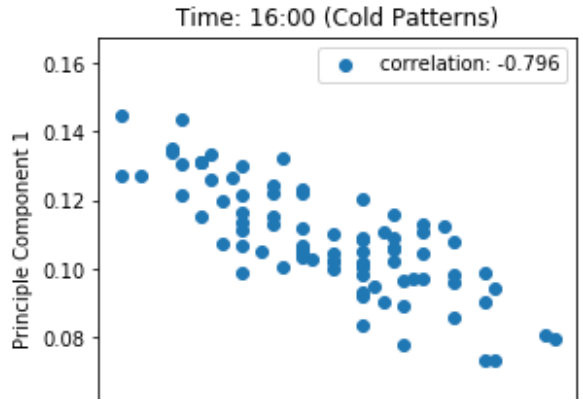

Time: 11:00 (Cold Patterns)
correlation: -0.788
Principle Component 1
Temperature (C°)

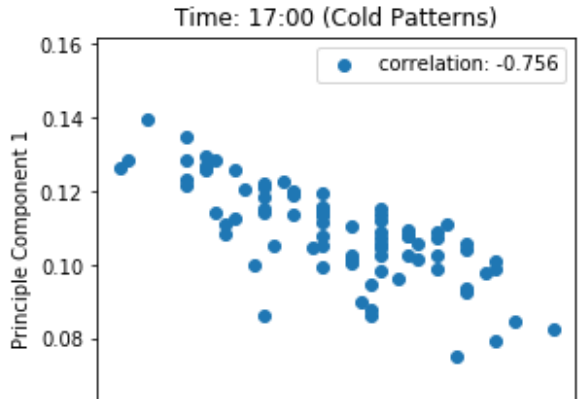

Time: 12:00 (Cold Patterns)
correlation: -0.778
Principle Component 1
Temperature (C°)

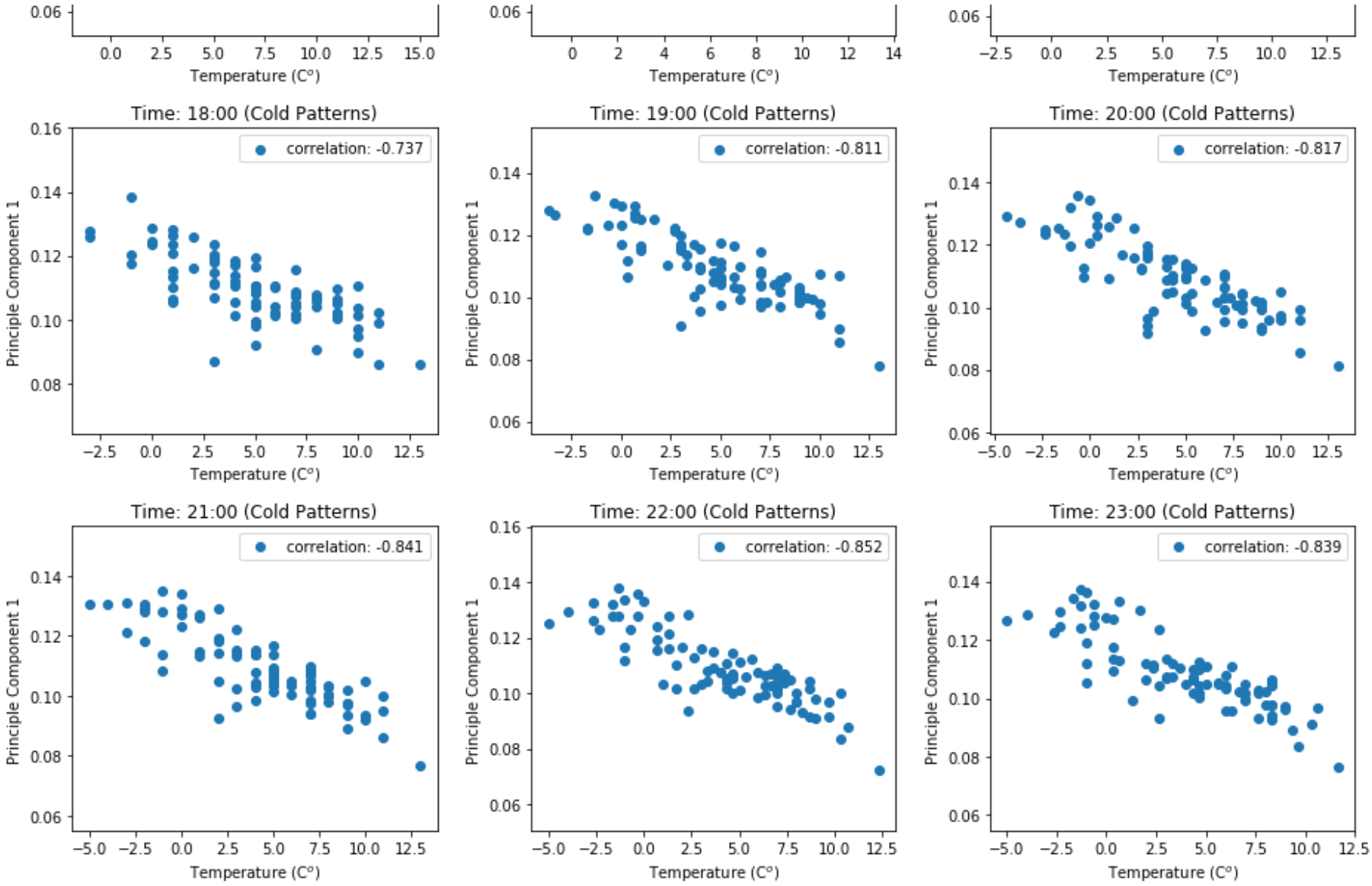

Time: 13:00 (Cold Patterns)
correlation: -0.768
Principle Component 1
Temperature (C°)


Time: 14:00 (Cold Patterns)
correlation: -0.768
Principle Component 1
Temperature (C°)


Time: 15:00 (Cold Patterns)
correlation: -0.772
Principle Component 1

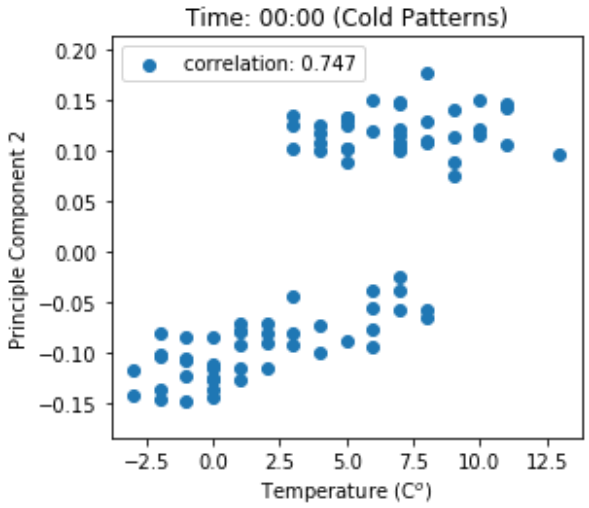

Time: 16:00 (Cold Patterns)
correlation: -0.796
Principle Component 1

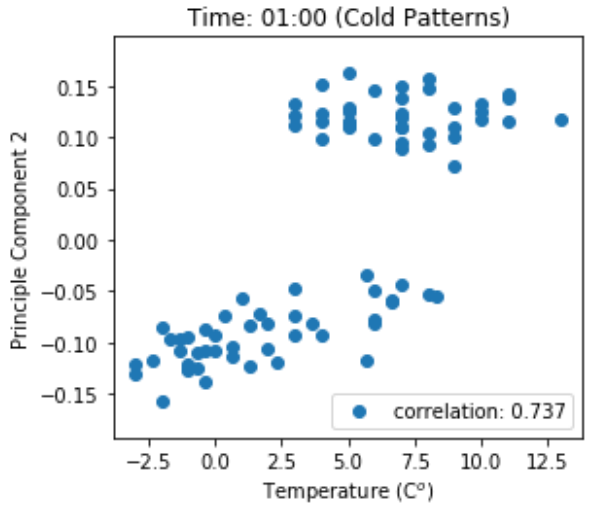

Time: 17:00 (Cold Patterns)
correlation: -0.756
Principle Component 1

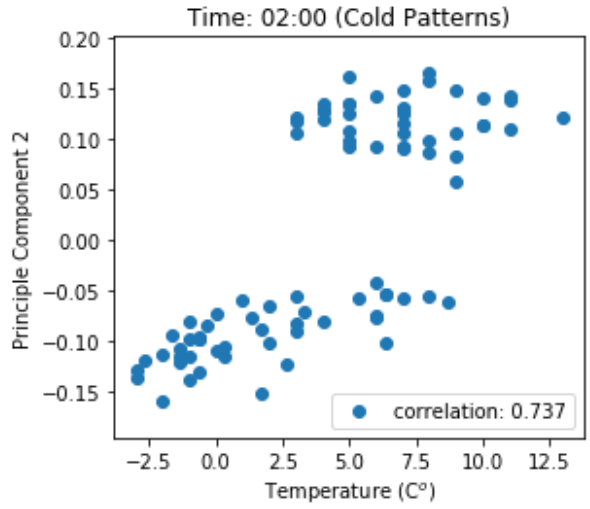


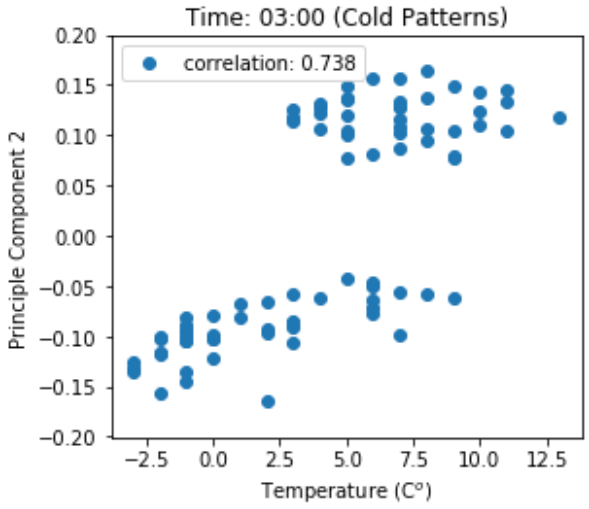
2) PC2's coefficients vs Temperature:

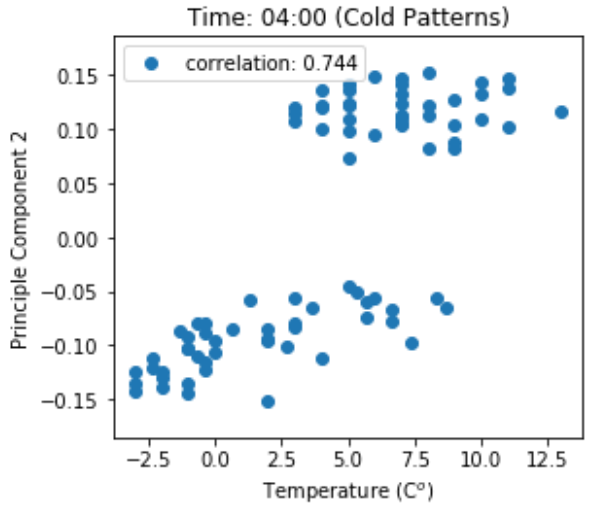
In [29]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.transpose().dropna() # 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_cold.TempC[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')], pca_n
std.components_[1], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[1], df_wea1h_nf_cold
.TempC[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: 0' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 2')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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_cold.TempC[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')], pca_nstd.co
mponents_[1], label = 'correlation: ' + str(round(np.corrcoef(pca_nstd.components_[1], df_wea1h_nf_cold.TempC
[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')])[0][1], 3)))
        ax.set_title('Time: ' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Principle Component 2')
plt.tight_layout()
```

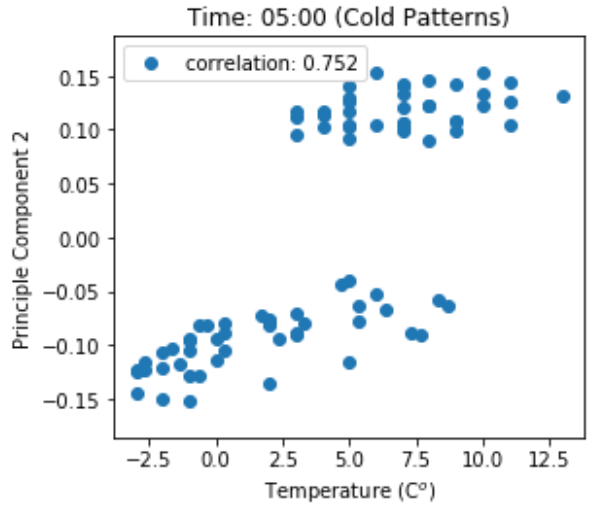

Time: 00:00 (Cold Patterns)
correlation: 0.747
Principle Component 2
Temperature (C°)

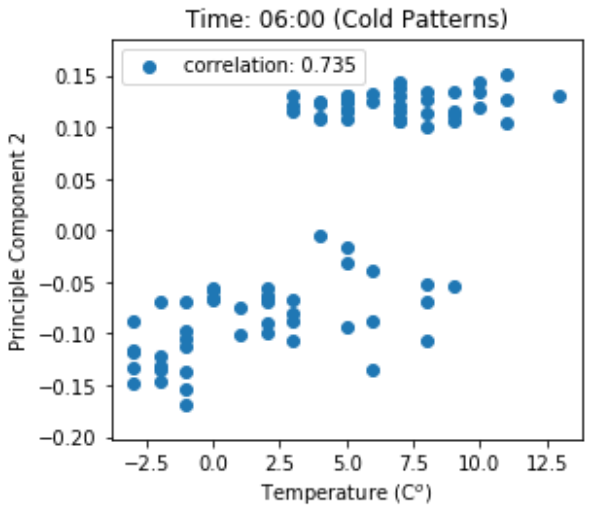

Time: 01:00 (Cold Patterns)
correlation: 0.737
Principle Component 2
Temperature (C°)

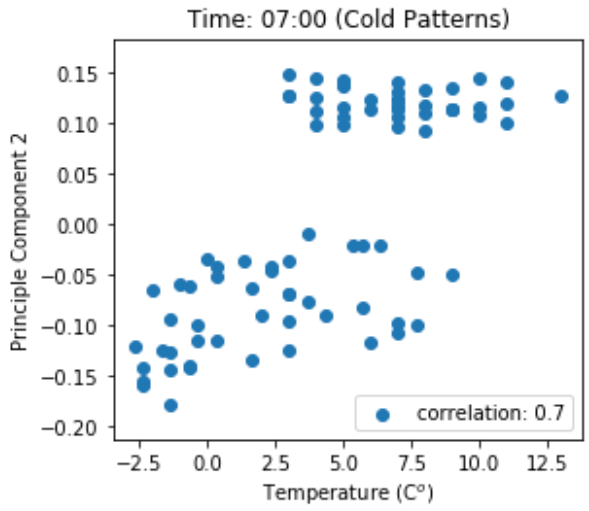

Time: 02:00 (Cold Patterns)
correlation: 0.737
Principle Component 2
Temperature (C°)

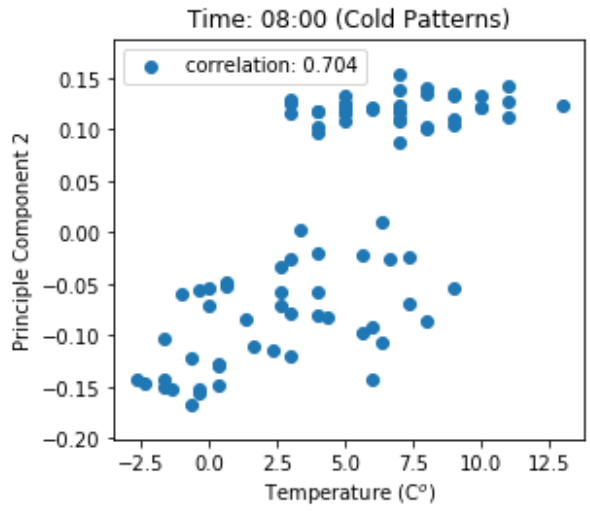

Time: 03:00 (Cold Patterns)
correlation: 0.738
Principle Component 2
Temperature (C°)

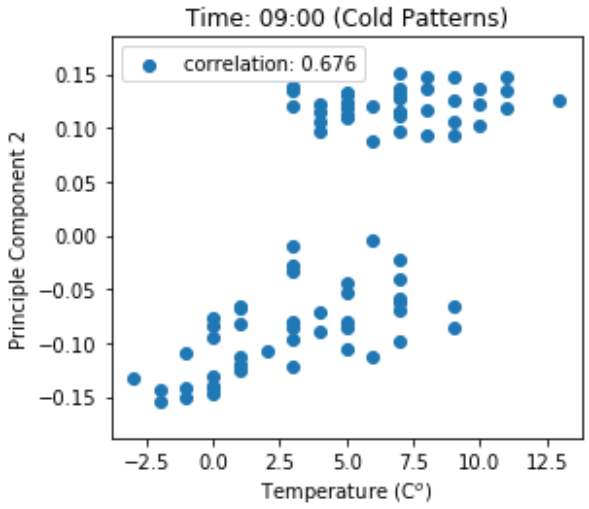

Time: 04:00 (Cold Patterns)
correlation: 0.744
Principle Component 2
Temperature (C°)

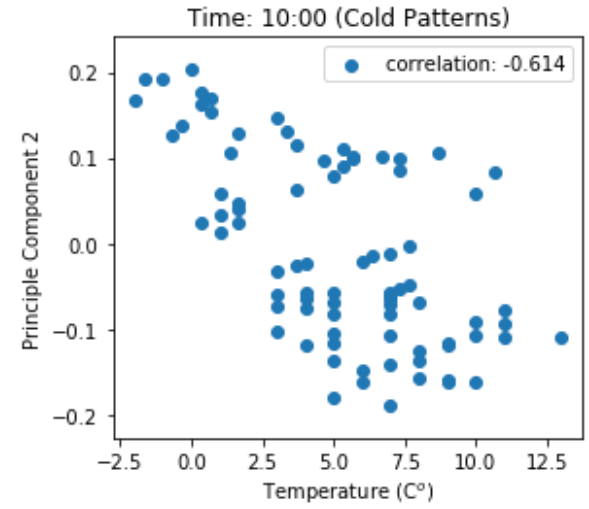

Time: 05:00 (Cold Patterns)
correlation: 0.752
Principle Component 2
Temperature (C°)

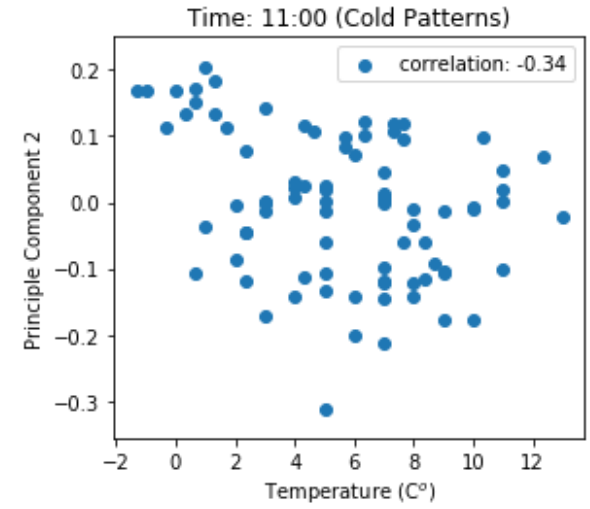

Time: 06:00 (Cold Patterns)
correlation: 0.735
Principle Component 2
Temperature (C°)

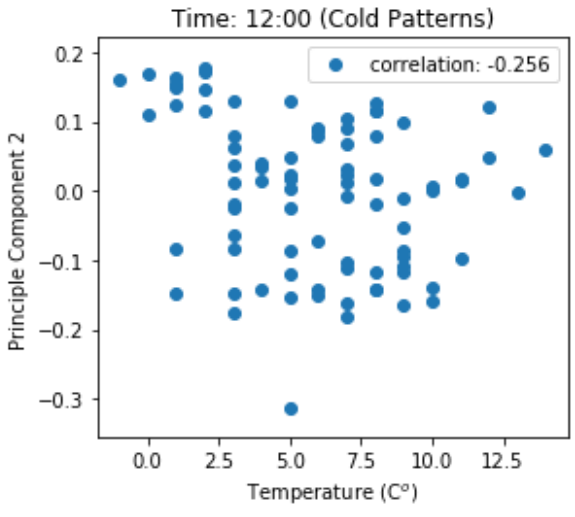

Time: 07:00 (Cold Patterns)
correlation: 0.7
Principle Component 2
Temperature (C°)

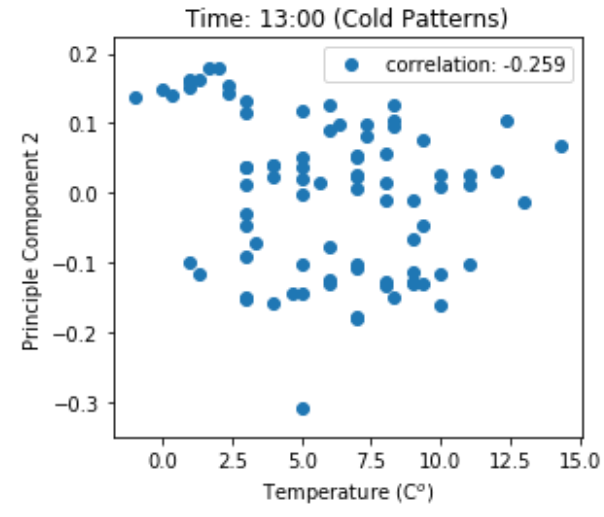

Time: 08:00 (Cold Patterns)
correlation: 0.704
Principle Component 2
Temperature (C°)

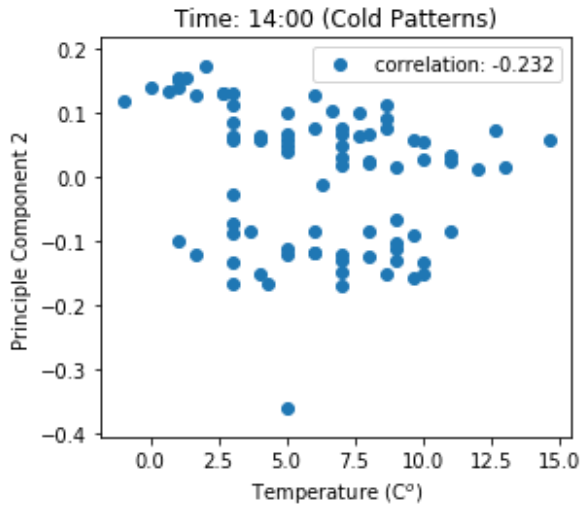

Time: 09:00 (Cold Patterns)
correlation: 0.676
Principle Component 2
Temperature (C°)

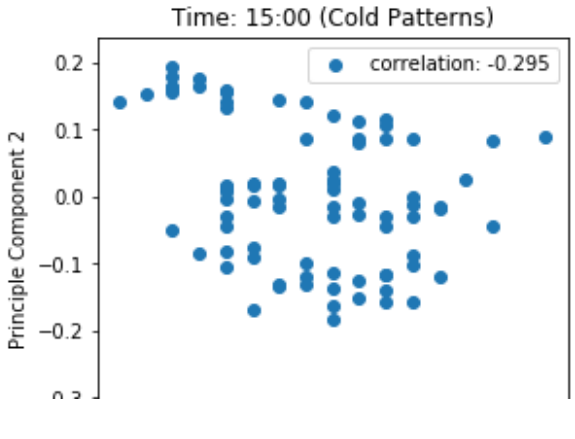

Time: 10:00 (Cold Patterns)
correlation: -0.614
Principle Component 2
Temperature (C°)

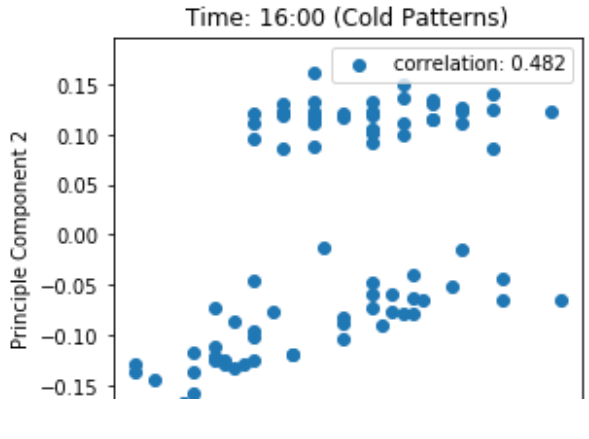

Time: 11:00 (Cold Patterns)
correlation: -0.34
Principle Component 2
Temperature (C°)

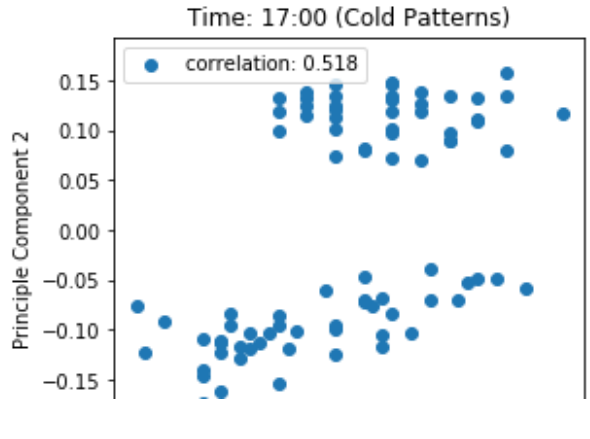

Time: 12:00 (Cold Patterns)
correlation: -0.256
Principle Component 2
Temperature (C°)

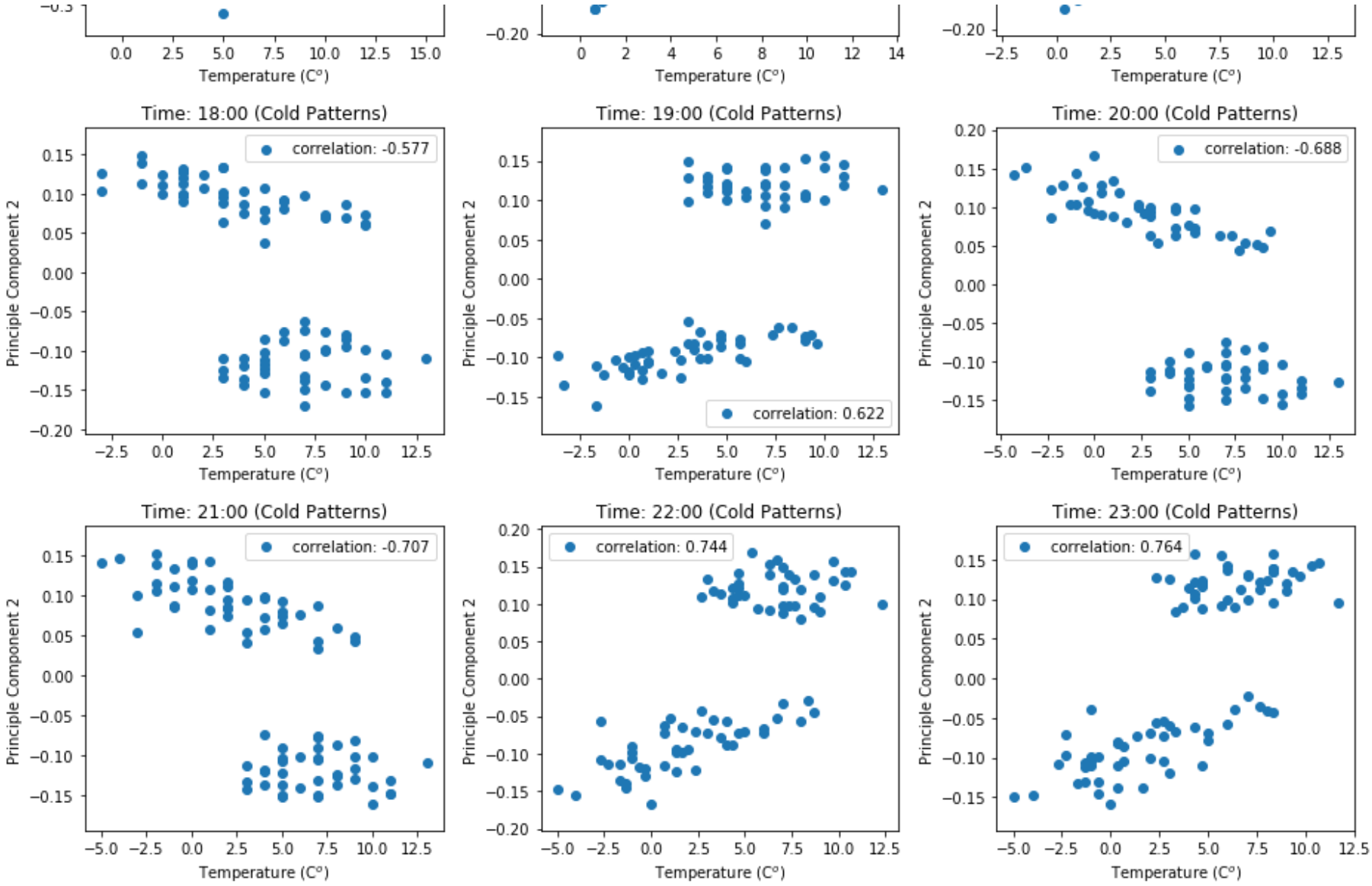

Time: 13:00 (Cold Patterns)
correlation: -0.259
Principle Component 2
Temperature (C°)


Time: 14:00 (Cold Patterns)
correlation: -0.232
Principle Component 2
Temperature (C°)


Time: 15:00 (Cold Patterns)
correlation: -0.295
Principle Component 2


Time: 16:00 (Cold Patterns)
correlation: 0.482
Principle Component 2

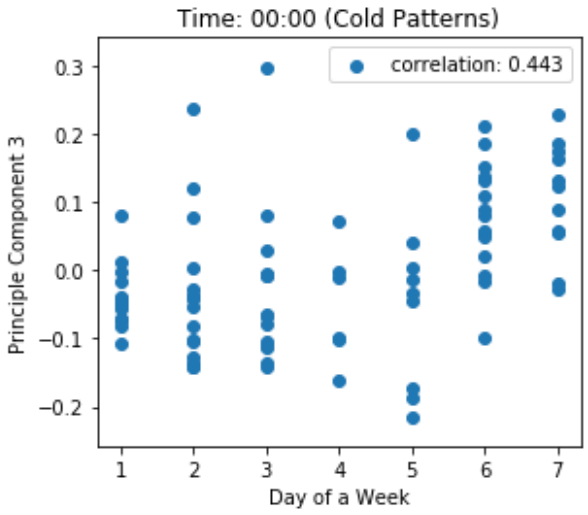

Time: 17:00 (Cold Patterns)
correlation: 0.518
Principle Component 2

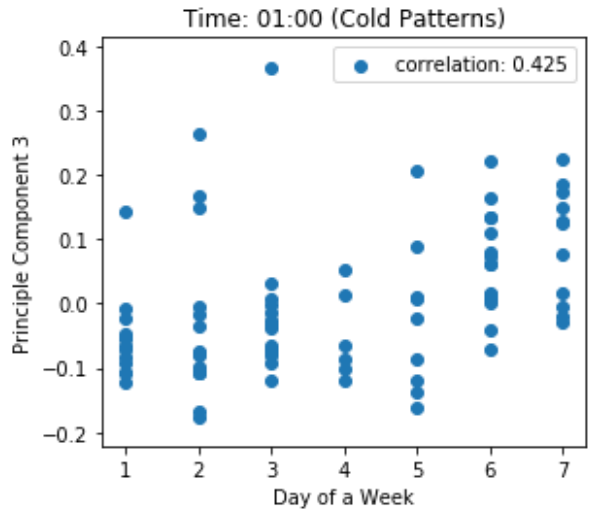


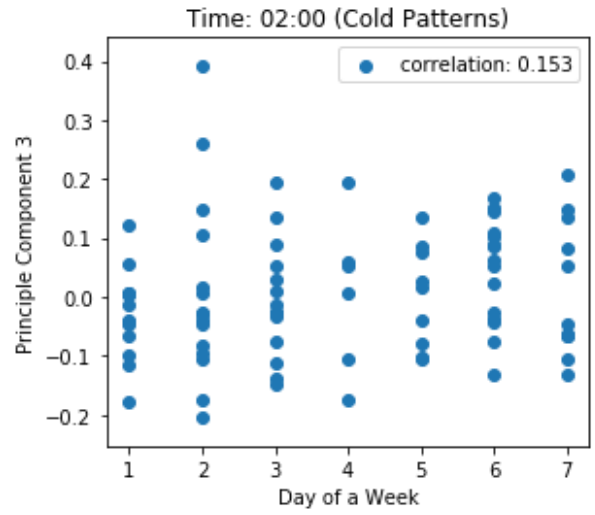
The above figures show that PC2 still has certain amount of correlation with temperature but not as significant as PC1 and temperature have.

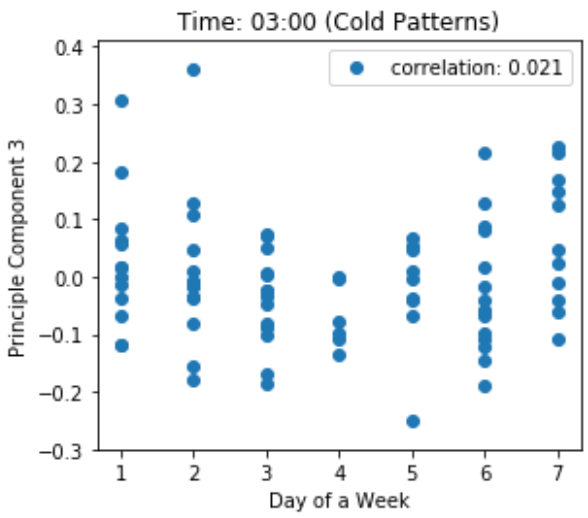
3) PC3's coefficients vs Day of a Week:

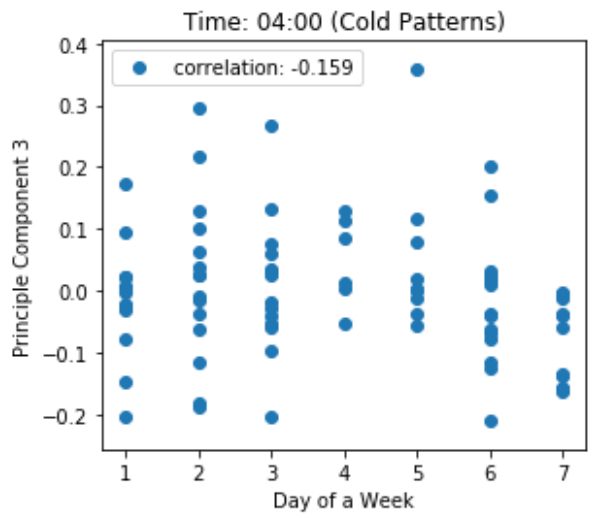
In [30]:
```
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
        x.set_index('GMT', 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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Day of a Week')
        ax.set_ylabel('Principle Component 3')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel('Day of a Week')
        ax.set_ylabel('Principle Component 3')
plt.tight_layout()
```

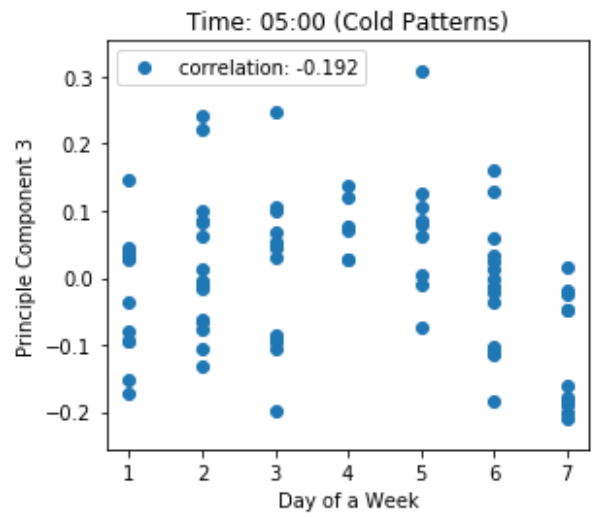

Time: 00:00 (Cold Patterns)
correlation: 0.443
Principle Component 3
Day of a Week

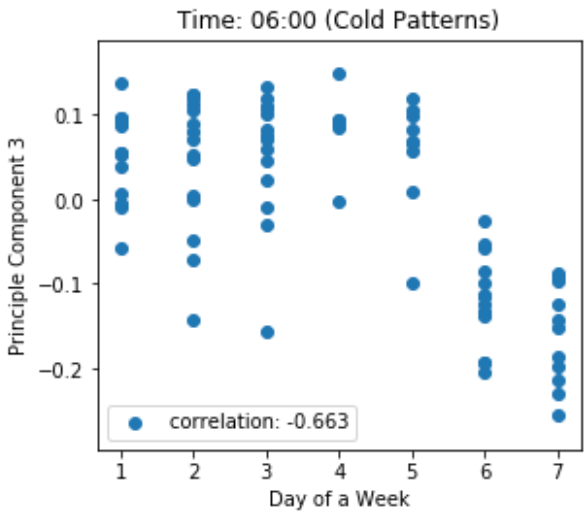

Time: 01:00 (Cold Patterns)
correlation: 0.425
Principle Component 3
Day of a Week

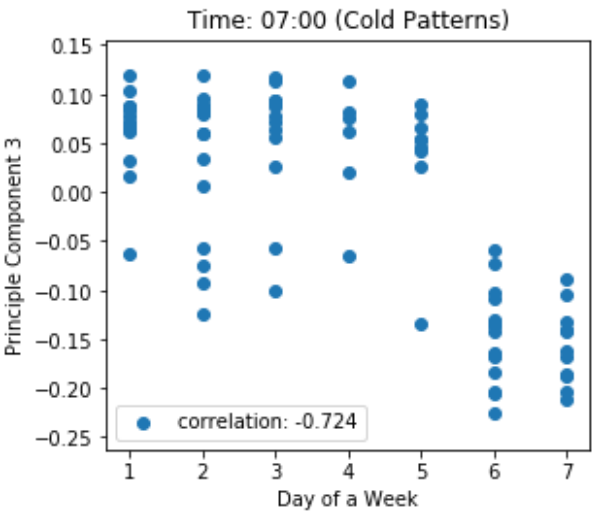

Time: 02:00 (Cold Patterns)
correlation: 0.153
Principle Component 3
Day of a Week

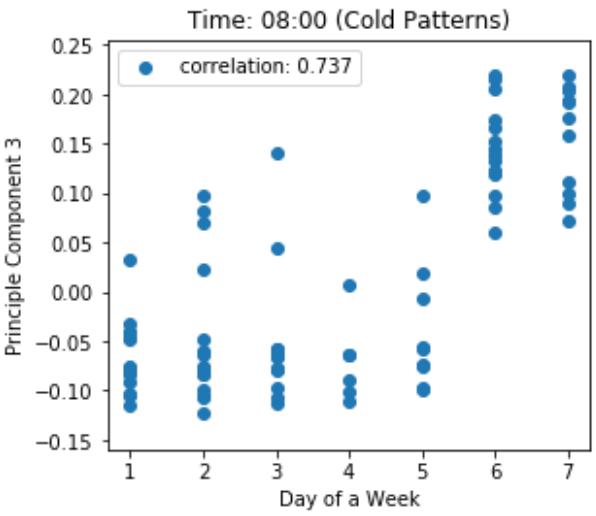

Time: 03:00 (Cold Patterns)
correlation: 0.021
Principle Component 3
Day of a Week

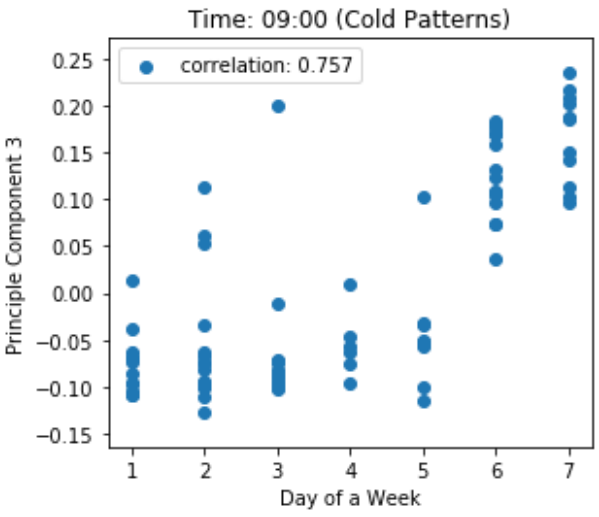

Time: 04:00 (Cold Patterns)
correlation: -0.159
Principle Component 3
Day of a Week

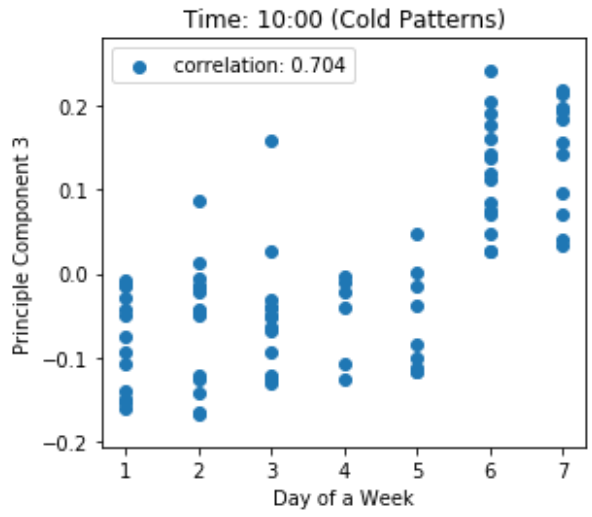

Time: 05:00 (Cold Patterns)
correlation: -0.192
Principle Component 3
Day of a Week

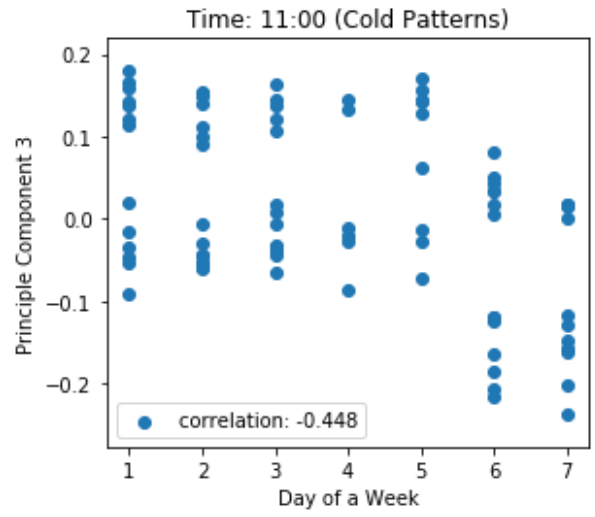

Time: 06:00 (Cold Patterns)
correlation: -0.663
Principle Component 3
Day of a Week

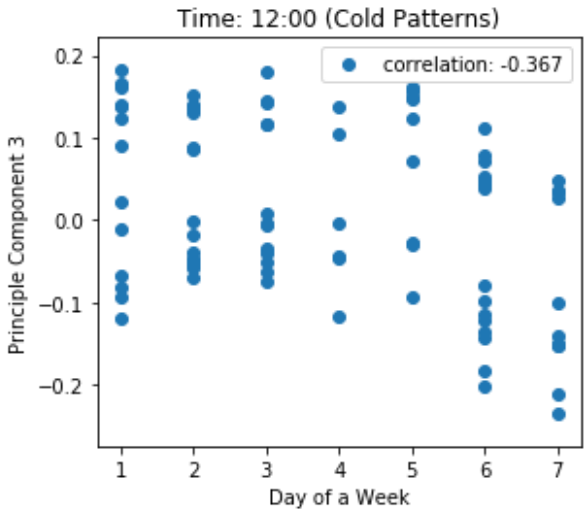

Time: 07:00 (Cold Patterns)
correlation: -0.724
Principle Component 3
Day of a Week

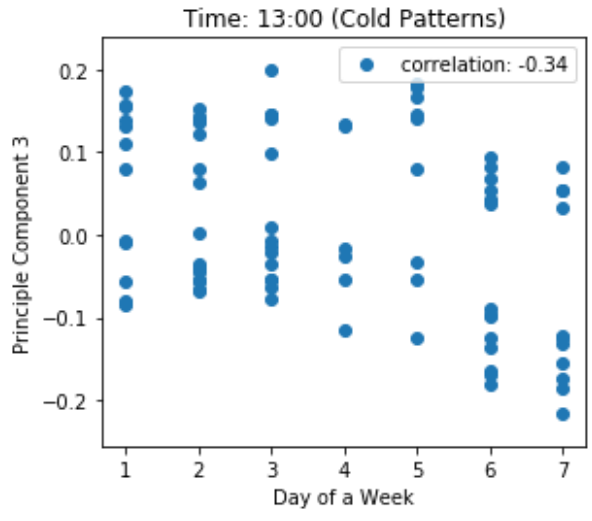

Time: 08:00 (Cold Patterns)
correlation: 0.737
Principle Component 3
Day of a Week

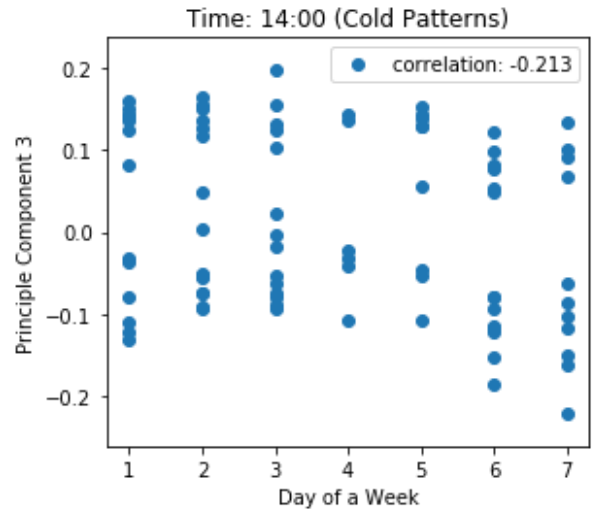

Time: 09:00 (Cold Patterns)
correlation: 0.757
Principle Component 3
Day of a Week

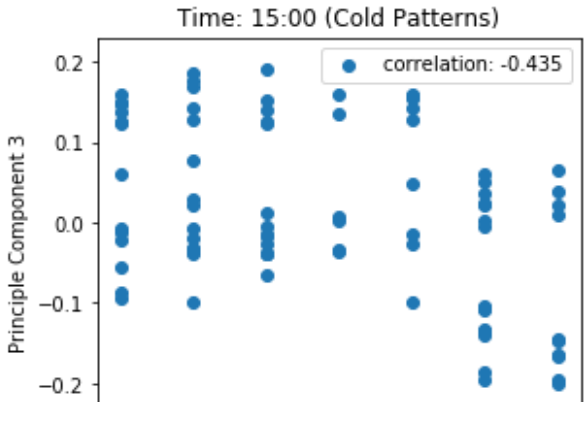

Time: 10:00 (Cold Patterns)
correlation: 0.704
Principle Component 3
Day of a Week

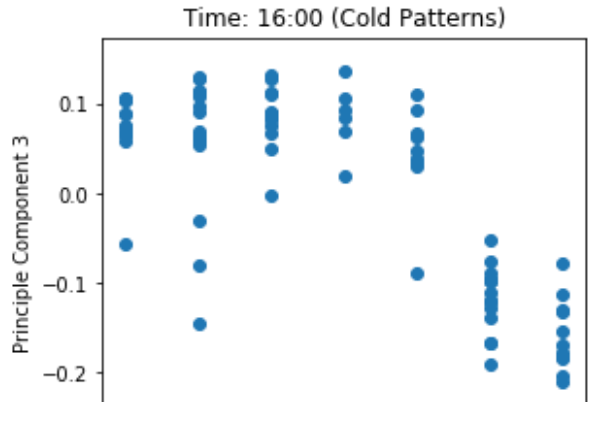

Time: 11:00 (Cold Patterns)
correlation: -0.448
Principle Component 3
Day of a Week

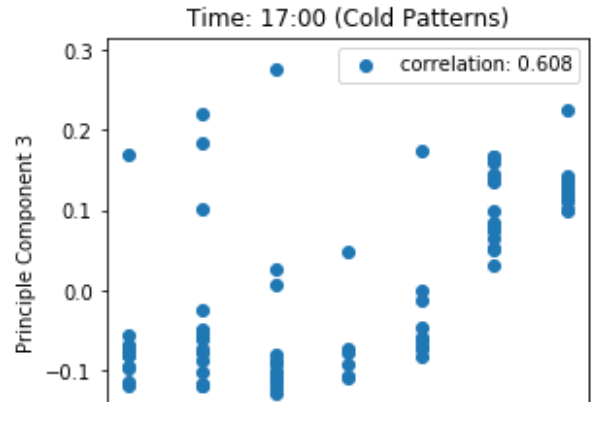

Time: 12:00 (Cold Patterns)
correlation: -0.367
Principle Component 3
Day of a Week

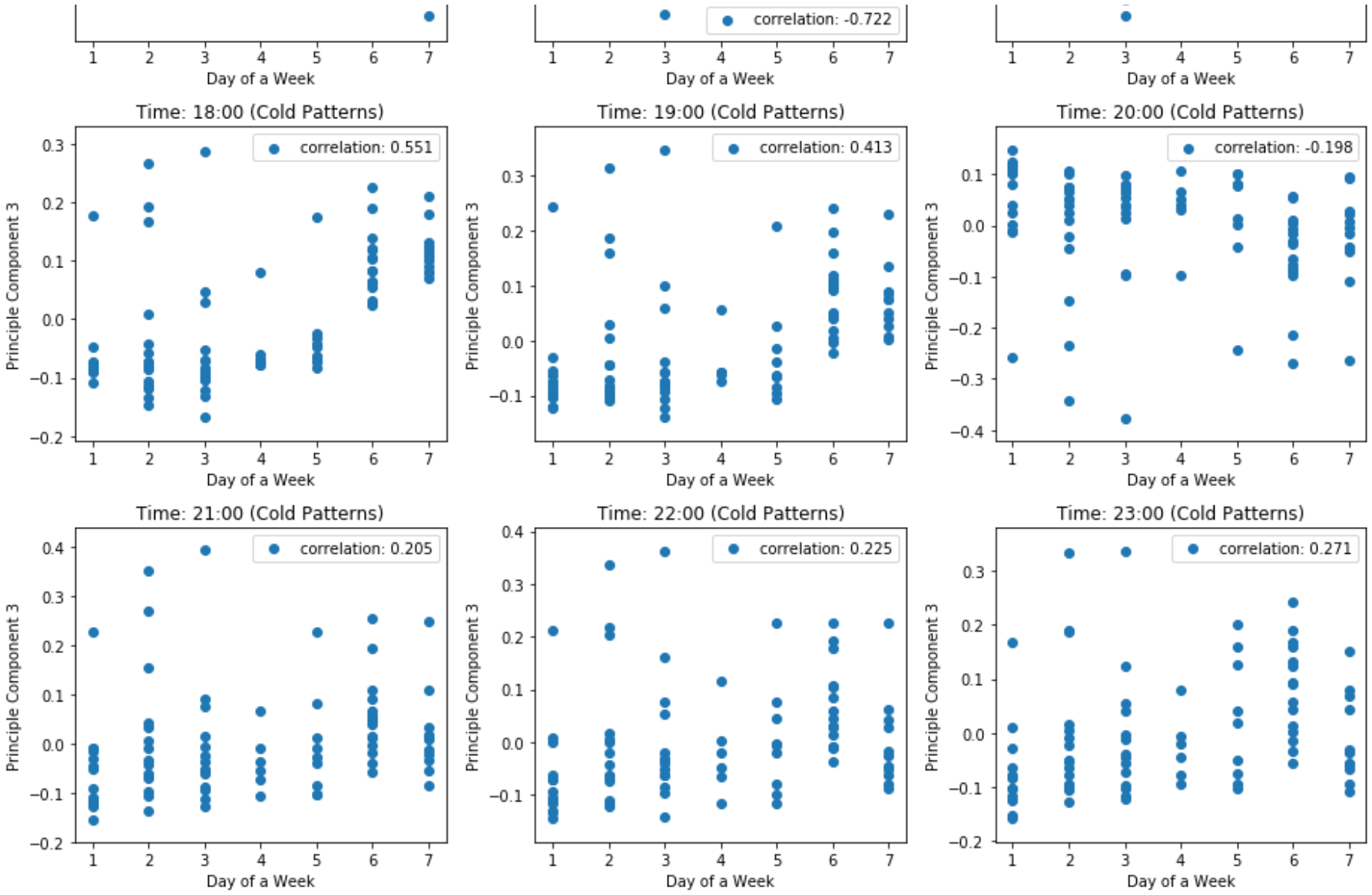

Time: 13:00 (Cold Patterns)
correlation: -0.34
Principle Component 3
Day of a Week


Time: 14:00 (Cold Patterns)
correlation: -0.213
Principle Component 3
Day of a Week


Time: 15:00 (Cold Patterns)
correlation: -0.435
Principle Component 3


Time: 16:00 (Cold Patterns)
Principle Component 3


Time: 17:00 (Cold Patterns)
correlation: 0.608
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 [31]:
```
x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['TempC'].values) # ad
d temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['GMT']).dt.
dayofweek # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['TempC'].values) # ad
d temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['GMT']).dt.
dayofweek # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['TempC'].values) # ad
d temperature column to dataframe
t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['GMT']).dt.
dayofweek # 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')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-31-e88dc4ecde5e> in <module>()
      1 x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]
      2 t = x.dropna(axis = 'columns')
----> 3 t2 = t[t.columns[np.insert(kmeans.labels_ == 0, 0, True)]].copy() # select all data from the n-th c
luster
      4 t2['TempC'] = np.round(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['TempC'].va
lues) # add temperature column to dataframe
      5 t2['DayofWeek'] = pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')][
'GMT']).dt.dayofweek # add day of a week column to dataframe

/anaconda3/lib/python3.6/site-packages/pandas/core/indexes/base.py in __getitem__(self, key)
   2078
   2079         key = com._values_from_object(key)
-> 2080         result = getitem(key)
   2081         if not is_scalar(result):
   2082             return promote(result)

IndexError: boolean index did not match indexed array along dimension 0; dimension is 2552 but corresponding
boolean dimension is 3721
```

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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
#     else:
#         x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
#         ax.set_xlabel('PC2')
#         ax.set_ylabel('PC3')
#         ax.set_zlabel('PC1')
plt.tight_layout()
```

In [44]:
```
x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(0) + ':00:00')]
dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
x.set_index('GMT', inplace = True)
x = x.transpose().dropna()
x
```

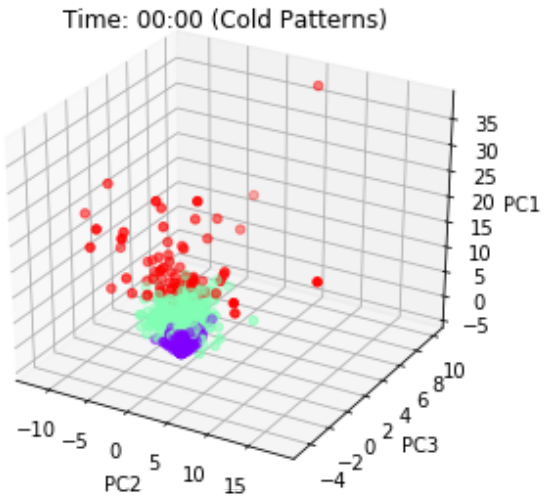
Out[44]:

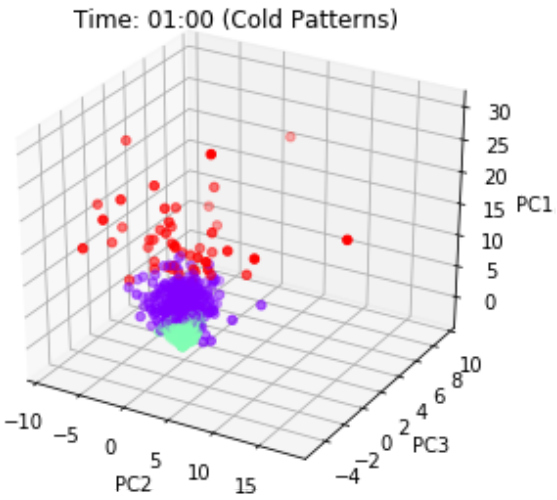
GMT	2013-01-01 00:00:00	2013-01-02 00:00:00	2013-01-03 00:00:00	2013-01-05 00:00:00	2013-01-06 00:00:00	2013-01-09 00:00:00	2013-01-12 00:00:00	2013-01-14 00:00:00	2013-01-15 00:00:00	2013-01-18 00:00:00	...	2013-12-17 00:00:00	2013-12-18 00:00:00	2013-12-21 00:00:00	0
N0000	1.038	0.324	0.475	0.415	0.316	0.315	0.603	0.297	0.309	0.318	...	0.391	0.620	0.549	0
N0001	0.232	0.264	0.267	0.237	0.264	0.195	0.092	0.099	0.240	0.298	...	0.641	0.651	0.653	0
N0002	0.052	0.387	0.101	0.094	0.037	0.085	0.055	0.410	0.100	0.503	...	0.100	0.094	0.041	0
N0003	0.256	0.112	0.133	0.084	0.199	0.074	0.093	0.282	0.055	0.083	...	0.211	0.431	0.221	0
N0004	0.181	1.156	1.322	0.368	0.990	0.235	0.792	0.569	0.106	0.583	...	0.690	0.720	0.818	0
N0005	0.773	0.153	0.244	0.142	0.075	0.204	0.179	0.152	0.146	0.170	...	0.044	0.138	0.275	0
N0006	0.000	0.000	0.000	0.000	0.000	0.000	0.006	0.058	0.047	0.042	...	0.000	0.000	0.000	0
N0007	0.163	0.159	0.161	0.153	0.167	0.167	0.197	0.174	0.173	0.149	...	0.170	0.156	0.167	0
N0008	0.261	0.149	0.134	0.149	0.769	0.123	0.155	0.185	0.181	0.102	...	0.428	0.594	0.564	0
N0009	1.042	0.641	0.780	0.511	0.856	0.478	1.834	0.874	0.545	0.548	...	0.638	0.532	0.612	0
N0010	2.114	4.206	1.762	3.972	1.986	3.647	1.321	3.963	1.336	6.135	...	2.028	0.298	4.585	3
N0011	0.197	0.018	0.015	0.043	0.019	0.036	0.035	0.017	0.039	0.038	...	0.145	0.040	0.029	0
N0012	0.219	0.683	0.382	0.667	0.733	0.134	0.543	0.472	0.781	0.460	...	0.784	0.939	0.769	0
N0013	0.613	0.598	0.579	0.586	0.552	0.622	0.572	0.604	0.569	0.844	...	0.680	1.183	0.585	0
N0014	1.066	1.979	1.265	1.492	1.067	1.483	1.851	1.719	1.451	1.962	...	0.940	1.157	0.972	1
N0015	0.365	0.075	0.080	0.083	0.148	0.254	0.286	0.471	0.293	0.282	...	0.129	0.426	0.389	0
N0016	0.150	0.065	0.244	0.230	0.077	0.075	0.104	0.151	0.010	0.159	...	0.072	0.114	0.010	0
N0017	0.045	0.048	0.054	0.047	0.049	0.128	0.255	0.223	0.313	0.140	...	0.277	0.071	0.081	0
N0018	0.104	0.112	0.095	0.112	0.225	0.135	0.140	0.213	0.218	0.539	...	0.142	0.151	0.173	0
N0020	0.327	0.534	0.161	0.218	0.227	0.144	0.284	0.020	0.143	0.126	...	0.176	0.141	0.313	0
N0021	0.456	0.199	0.448	0.357	0.232	0.171	0.235	0.295	0.263	0.444	...	0.581	0.577	0.180	0
N0022	0.042	0.041	0.269	0.042	0.084	0.094	0.051	0.096	0.097	0.120	...	0.051	0.050	0.093	0
N0023	0.768	0.741	0.064	0.000	0.000	0.073	0.021	0.007	1.707	1.300	...	0.000	0.193	0.772	0
N0024	1.744	0.853	0.862	2.261	2.159	1.331	0.751	1.294	0.551	2.558	...	0.581	0.647	1.635	0
N0025	0.067	0.000	0.004	0.047	0.000	0.000	0.061	0.034	0.072	0.000	...	0.072	0.072	0.072	0
N0026	0.081	0.063	0.882	0.269	0.288	0.153	1.067	0.397	1.876	0.076	...	0.408	0.290	1.864	0
N0027	0.272	0.204	0.212	0.195	0.342	0.270	0.079	0.237	0.222	0.149	...	0.066	0.242	0.127	0
N0028	0.064	0.092	0.082	0.073	0.097	0.066	0.069	0.060	0.085	0.083	...	0.067	0.068	0.086	0
N0029	0.451	0.640	2.712	1.371	0.848	0.834	1.215	0.898	1.098	1.594	...	1.044	1.336	0.949	1
N0030	0.105	0.116	0.121	0.101	0.113	0.115	0.109	0.120	0.141	0.127	...	0.116	0.122	0.109	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
N4138	0.482	0.156	0.243	0.216	0.248	0.175	0.186	0.234	0.138	0.195	...	0.319	0.369	0.424	0
N4139	0.215	0.186	0.201	0.197	0.152	0.155	0.196	0.256	0.159	0.178	...	0.119	0.151	0.085	0
N4140	0.337	0.120	0.072	0.173	0.121	0.097	0.375	0.070	0.110	0.071	...	0.120	0.084	0.227	0
N4141	0.449	0.548	0.575	0.622	0.593	0.625	0.399	0.643	0.809	0.675	...	0.348	0.166	0.386	0
N4142	0.182	0.160	0.212	0.158	0.198	0.256	0.192	0.136	0.224	0.180	...	0.187	0.190	0.191	0
N4143	0.763	0.468	0.074	0.076	0.314	0.663	0.477	0.350	0.484	0.307	...	0.337	0.063	0.071	0
N4144	0.455	0.463	0.425	0.415	0.466	0.800	1.086	0.963	0.932	1.284	...	0.726	0.534	0.864	0
N4145	0.398	0.767	0.112	0.080	0.088	0.041	0.108	0.051	0.051	0.399	...	0.417	0.189	0.208	0
N4146	0.049	0.994	0.773	0.628	1.335	0.075	0.068	0.051	0.048	0.043	...	0.322	0.045	0.131	0
N4147	0.473	0.300	0.413	0.562	0.383	0.300	0.413	0.230	0.383	0.265	...	0.990	0.155	0.474	0
N4148	0.703	0.257	0.324	0.358	0.438	0.217	0.167	0.190	0.282	0.232	...	0.299	0.286	0.144	0
N4149	0.120	0.283	0.156	0.146	0.203	0.267	0.535	0.168	0.269	0.093	...	0.235	0.256	0.359	0
N4150	2.486	1.649	1.522	1.936	2.534	2.684	1.923	1.505	2.214	3.422	...	1.294	1.583	1.331	3
N4152	0.311	0.016	0.028	0.012	0.012	0.018	0.011	0.012	0.011	0.011	...	0.025	0.023	0.059	0
N4153	1.123	0.085	0.100	0.318	0.080	0.146	0.113	0.107	0.110	0.074	...	0.178	0.178	0.131	0
N4154	0.660	0.224	0.223	0.057	0.056	0.138	0.127	0.095	0.274	0.069	...	0.433	0.096	0.138	0
N4156	0.043	0.063	0.042	0.044	0.046	0.051	0.045	0.045	0.043	0.045	...	0.055	0.037	0.044	0
N4157	0.183	0.441	0.325	0.809	0.425	0.388	0.247	0.283	0.370	0.262	...	0.222	0.196	0.345	0

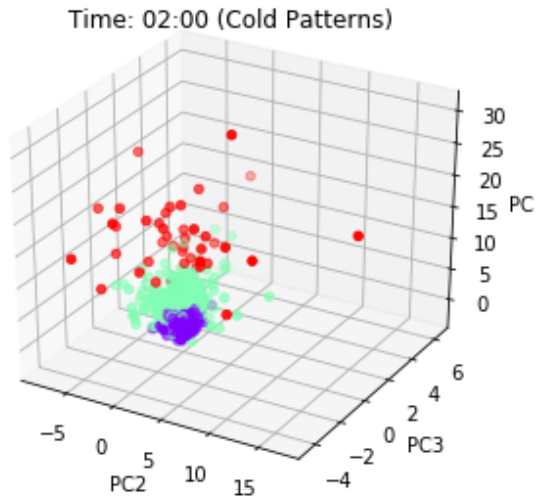
GMT	2013-01-01 00:00:00	2013-01-02 00:00:00	2013-01-03 00:00:00	2013-01-05 00:00:00	2013-01-06 00:00:00	2013-01-09 00:00:00	2013-01-12 00:00:00	2013-01-14 00:00:00	2013-01-15 00:00:00	2013-01-18 00:00:00	...	2013-12-17 00:00:00	2013-12-18 00:00:00	2013-12-21 00:00:00	0
N4158	0.333	0.399	0.214	0.313	0.250	0.201	0.271	0.131	0.262	0.347	...	0.121	0.316	0.229	0
N4159	0.280	0.192	0.205	0.207	0.224	0.212	0.212	0.191	0.192	0.370	...	0.212	0.210	0.273	0
N4162	0.150	0.086	0.080	0.184	0.128	0.081	0.096	0.076	0.088	0.077	...	0.093	0.091	0.074	0
N4163	0.187	0.383	0.509	0.584	0.483	0.386	0.420	0.506	0.489	0.571	...	0.151	0.261	0.072	0
N4164	0.051	0.050	0.039	0.106	0.033	0.030	0.045	0.040	0.025	0.039	...	0.033	0.027	0.042	0
N4165	0.087	0.063	0.067	0.084	0.054	0.084	0.067	0.068	0.066	0.066	...	0.078	0.085	0.100	0
N4166	0.020	0.045	0.051	0.062	0.103	0.063	0.021	0.029	0.200	0.023	...	0.072	0.080	0.021	0
N4167	0.586	0.189	0.234	0.204	0.195	0.201	0.195	0.173	0.254	0.363	...	0.092	0.109	0.184	0
N4168	0.201	0.204	0.197	0.199	0.186	0.197	0.354	0.240	0.192	0.217	...	0.234	0.223	0.251	0
N4170	0.254	0.135	0.149	0.171	0.351	0.346	0.253	0.130	0.193	0.187	...	0.333	0.283	0.109	0
N4171	0.102	0.124	0.169	0.052	0.020	0.094	0.020	0.103	0.101	0.137	...	0.089	0.114	0.044	0
N4172	0.132	0.311	0.283	0.195	0.352	0.142	0.275	0.147	0.188	0.156	...	0.128	0.133	0.282	0

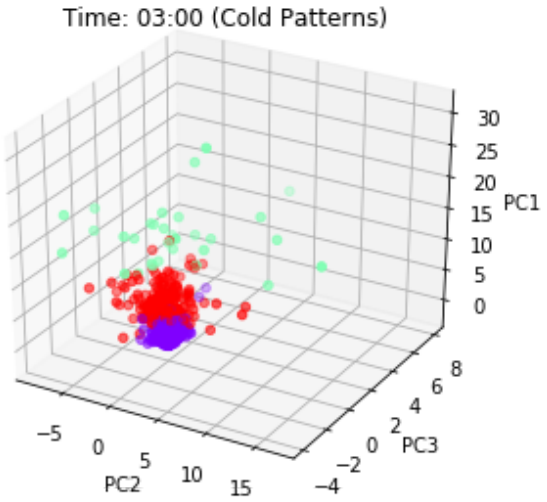
3720 rows × 83 columns

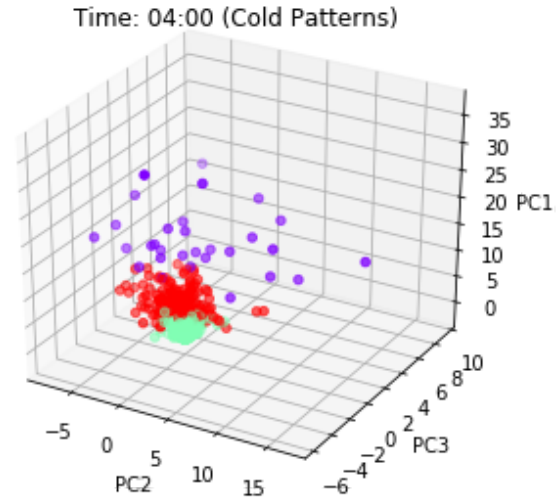
In [32]:
```
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1, projection = '3d')
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':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' + ' (Cold Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold Patterns)')
        ax.set_xlabel('PC2')
        ax.set_ylabel('PC3')
        ax.set_zlabel('PC1')
plt.tight_layout()
```

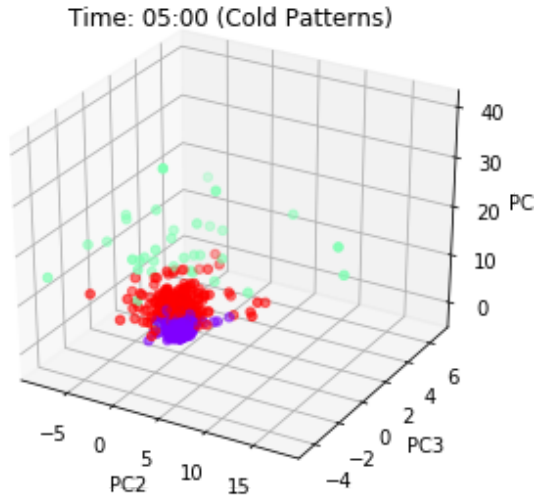

Time: 00:00 (Cold Patterns)
PC1
PC2
PC3

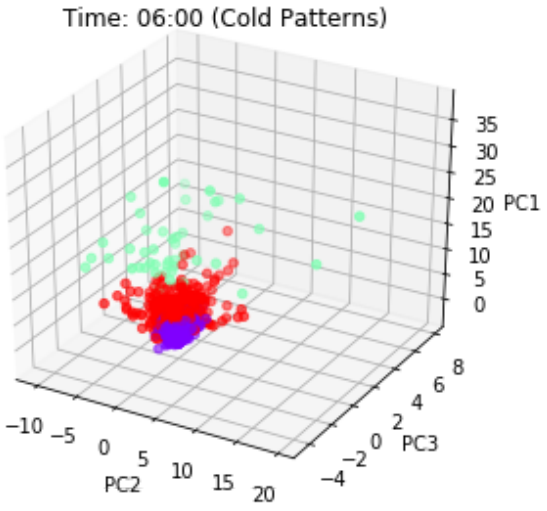

Time: 01:00 (Cold Patterns)
PC1
PC2
PC3

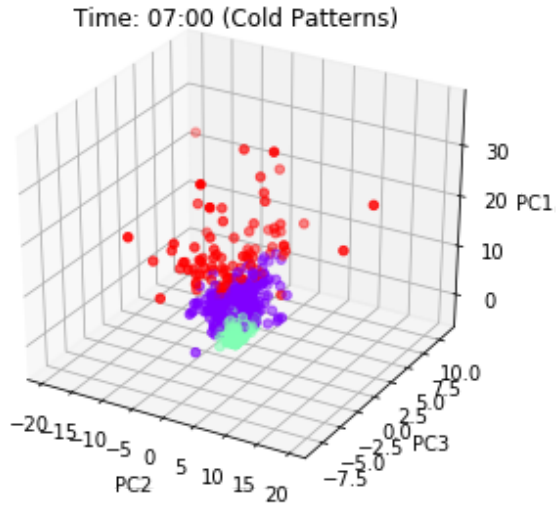

Time: 02:00 (Cold Patterns)
PC
PC2
PC3

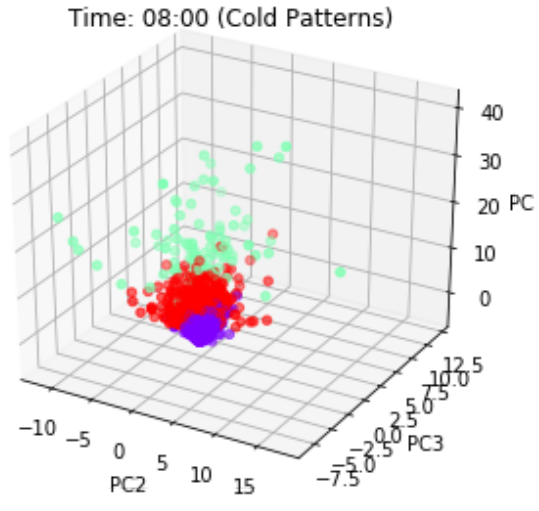

Time: 03:00 (Cold Patterns)
PC1
PC2
PC3

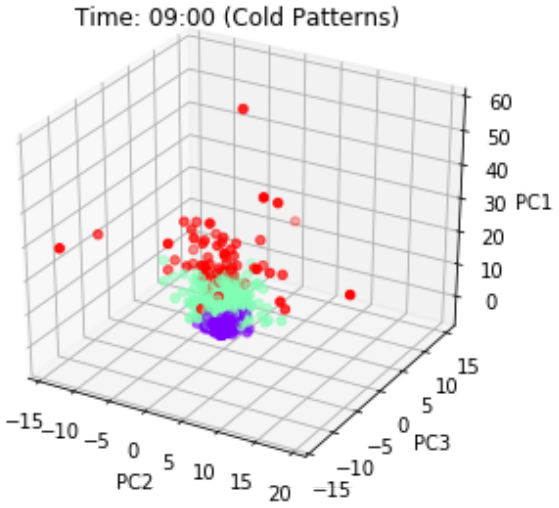

Time: 04:00 (Cold Patterns)
PC1
PC2
PC3

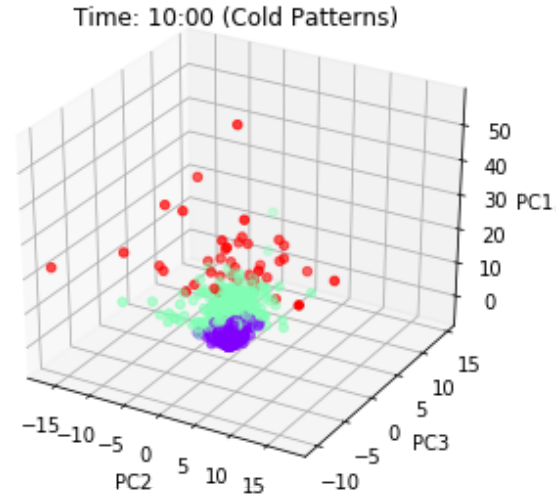

Time: 05:00 (Cold Patterns)
PC
PC2
PC3

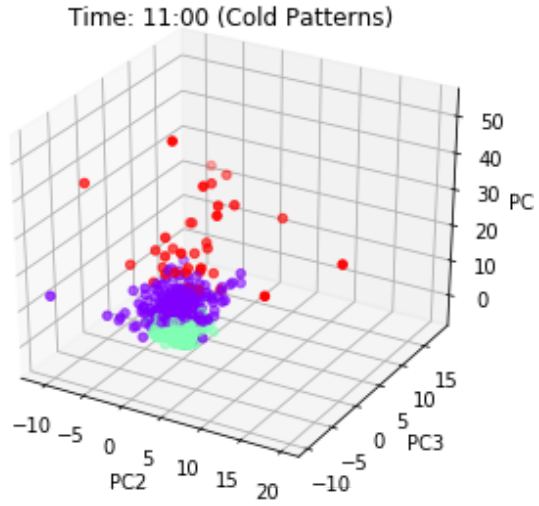

Time: 06:00 (Cold Patterns)
PC1
PC2
PC3

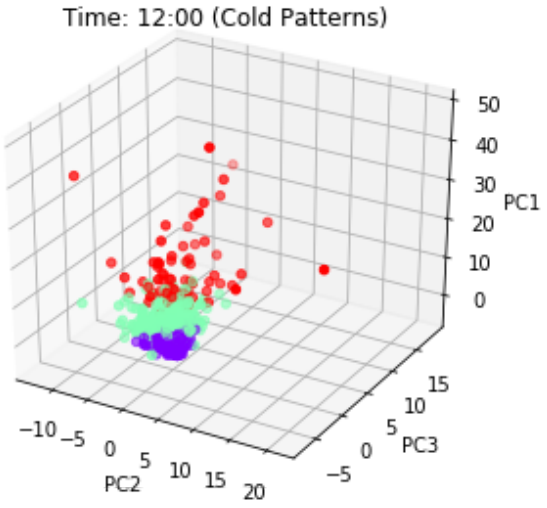

Time: 07:00 (Cold Patterns)
PC1
PC2
PC3

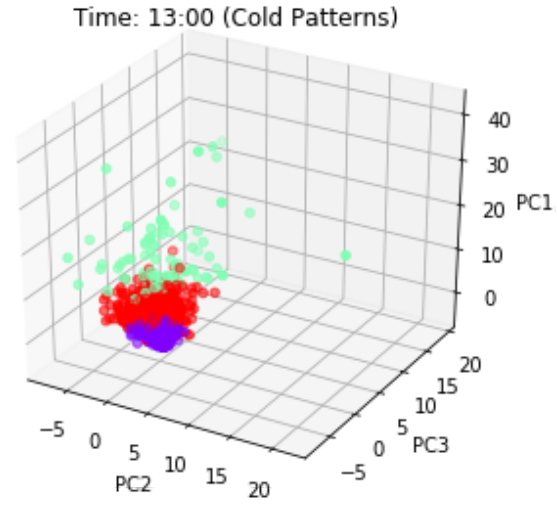

Time: 08:00 (Cold Patterns)
PC
PC2
PC3

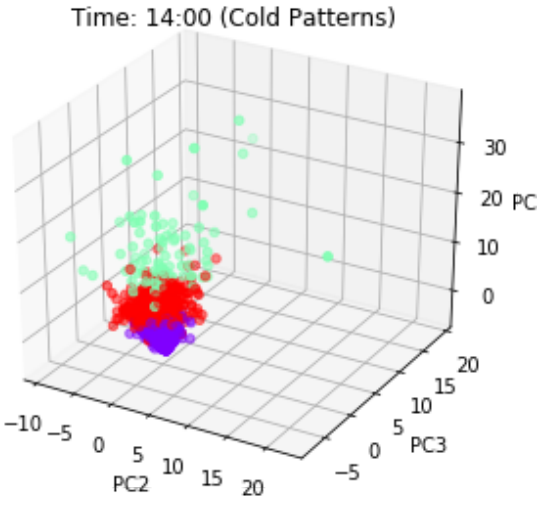

Time: 09:00 (Cold Patterns)
PC1
PC2
PC3

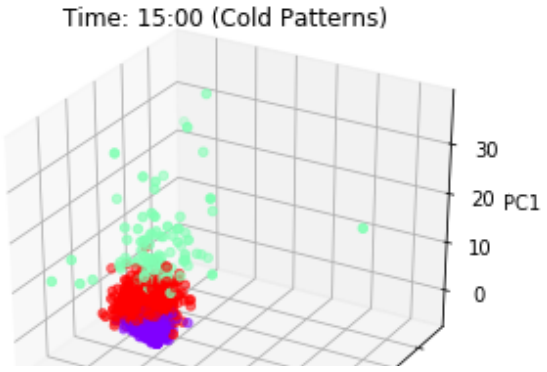

Time: 10:00 (Cold Patterns)
PC1
PC2
PC3

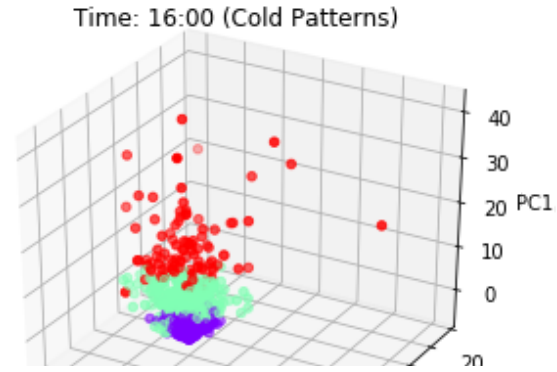

Time: 11:00 (Cold Patterns)
PC
PC2
PC3

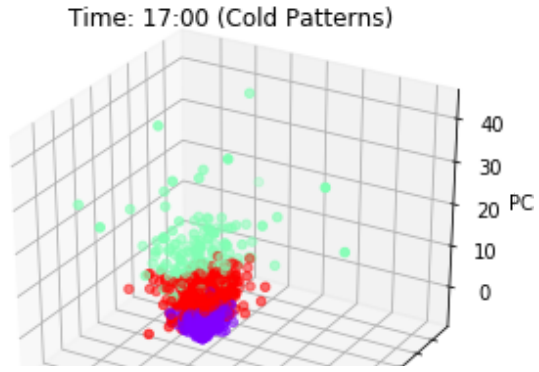

Time: 12:00 (Cold Patterns)
PC1
PC2
PC3

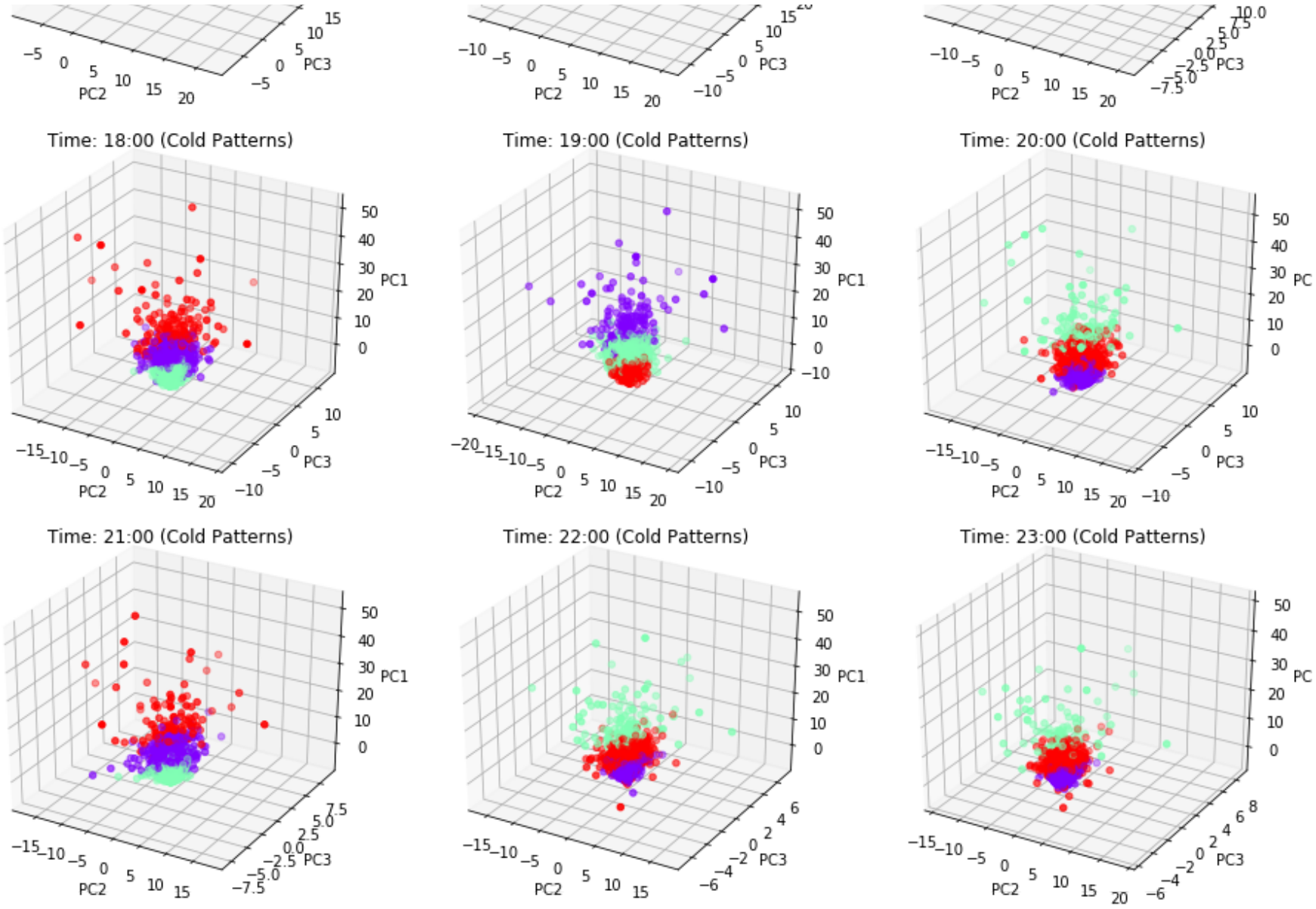

Time: 13:00 (Cold Patterns)
PC1
PC2
PC3


Time: 14:00 (Cold Patterns)
PC
PC2
PC3


Time: 15:00 (Cold Patterns)
PC1


Time: 16:00 (Cold Patterns)
PC1


Time: 17:00 (Cold Patterns)
PC



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 [ ]:
```
x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(0) + ':00:00')]
# dayofWeek= pd.DataFrame(pd.to_datetime(x.GMT).dt.dayofweek)
x.set_index('GMT', inplace = True)
x = x.transpose().dropna()
x
```

In [47]:
```
x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(0) + ':00:00')]
x.set_index('GMT', inplace = True)
t = x.dropna(axis=1) # data used for analyzing consumption distribution
# t = t.dropna(axis=1)
t
```

Out[47]:

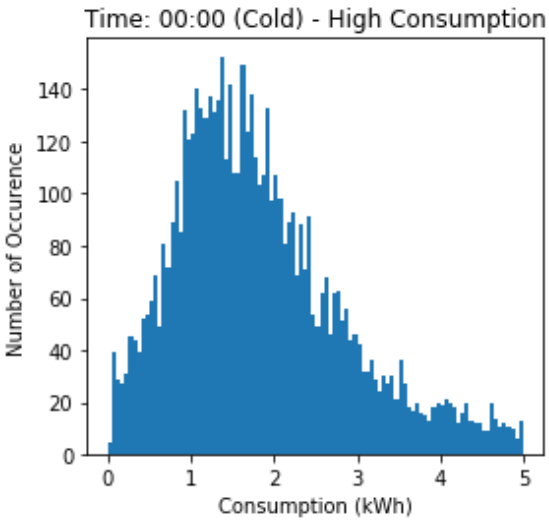
	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	N0009	...	N4162	N4163	N4164	N4165	N4166	N4167	N
GMT																		
2013-01-01 00:00:00	1.038	0.232	0.052	0.256	0.181	0.773	0.000	0.163	0.261	1.042	...	0.150	0.187	0.051	0.087	0.020	0.586	0
2013-01-02 00:00:00	0.324	0.264	0.387	0.112	1.156	0.153	0.000	0.159	0.149	0.641	...	0.086	0.383	0.050	0.063	0.045	0.189	0
2013-01-03 00:00:00	0.475	0.267	0.101	0.133	1.322	0.244	0.000	0.161	0.134	0.780	...	0.080	0.509	0.039	0.067	0.051	0.234	0
2013-01-05 00:00:00	0.415	0.237	0.094	0.084	0.368	0.142	0.000	0.153	0.149	0.511	...	0.184	0.584	0.106	0.084	0.062	0.204	0
2013-01-06 00:00:00	0.316	0.264	0.037	0.199	0.990	0.075	0.000	0.167	0.769	0.856	...	0.128	0.483	0.033	0.054	0.103	0.195	0
2013-01-09 00:00:00	0.315	0.195	0.085	0.074	0.235	0.204	0.000	0.167	0.123	0.478	...	0.081	0.386	0.030	0.084	0.063	0.201	0
2013-01-12 00:00:00	0.603	0.092	0.055	0.093	0.792	0.179	0.006	0.197	0.155	1.834	...	0.096	0.420	0.045	0.067	0.021	0.195	0
2013-01-14 00:00:00	0.297	0.099	0.410	0.282	0.569	0.152	0.058	0.174	0.185	0.874	...	0.076	0.506	0.040	0.068	0.029	0.173	0
2013-01-15 00:00:00	0.309	0.240	0.100	0.055	0.106	0.146	0.047	0.173	0.181	0.545	...	0.088	0.489	0.025	0.066	0.200	0.254	0
2013-01-18 00:00:00	0.318	0.298	0.503	0.083	0.583	0.170	0.042	0.149	0.102	0.548	...	0.077	0.571	0.039	0.066	0.023	0.363	0
2013-01-22 00:00:00	0.590	0.082	0.099	0.236	0.445	0.060	0.024	0.152	0.191	0.658	...	0.074	0.432	0.042	0.103	0.115	0.360	0
2013-01-23 00:00:00	0.296	0.097	0.100	0.083	0.109	0.086	0.000	0.165	0.121	2.331	...	0.075	0.534	0.039	0.063	0.070	0.382	0
2013-01-24 00:00:00	0.577	0.076	0.101	0.168	0.652	0.057	0.000	0.168	0.137	0.768	...	0.162	0.366	0.043	0.066	0.095	0.159	0
2013-01-26 00:00:00	0.275	0.091	0.045	0.103	0.693	0.090	0.002	0.168	0.135	0.601	...	0.078	0.555	0.043	0.081	0.050	0.222	0
2013-01-27 00:00:00	0.534	0.099	0.103	0.130	0.574	0.202	0.047	0.155	0.119	1.435	...	0.081	0.341	0.038	0.104	0.134	0.465	0
2013-01-31 00:00:00	0.545	0.237	0.072	0.189	1.575	0.075	0.007	0.151	0.131	0.525	...	0.102	0.370	0.056	0.075	0.042	0.203	0
2013-02-01 00:00:00	0.394	0.241	0.071	1.082	0.585	0.080	0.079	0.166	0.137	0.522	...	0.080	0.472	0.076	0.079	0.072	0.526	0
2013-02-02 00:00:00	1.248	0.211	0.035	0.187	1.117	0.162	0.054	0.152	0.120	0.618	...	0.075	0.363	0.013	0.078	0.021	0.184	0
2013-02-04 00:00:00	0.333	0.241	0.096	0.061	0.539	0.053	0.047	0.164	0.115	0.688	...	0.090	0.344	0.047	0.077	0.034	0.224	0
2013-02-06 00:00:00	0.313	0.101	0.098	0.212	0.541	0.147	0.005	0.151	0.122	0.615	...	0.075	0.414	0.046	0.068	0.039	0.176	0

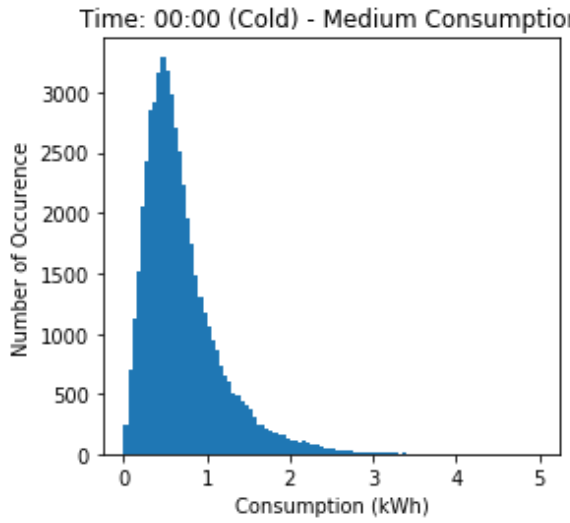
	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	N0009	...	N4162	N4163	N4164	N4165	N4166	N4167	N
GMT																		
2013-02-12 00:00:00	0.357	0.103	0.048	0.203	0.372	0.060	0.080	0.150	0.250	0.709	...	0.057	0.391	0.041	0.069	0.022	0.234	0
2013-02-13 00:00:00	0.342	0.102	0.039	0.117	0.539	0.141	0.064	0.167	0.334	1.750	...	0.073	0.346	0.042	0.110	0.063	0.102	0
2013-02-14 00:00:00	0.303	0.102	0.065	0.146	0.959	0.142	0.136	0.168	0.108	1.630	...	0.075	0.280	0.028	0.064	0.281	0.203	0
2013-02-16 00:00:00	0.325	0.100	0.102	0.195	0.208	0.085	0.113	0.149	0.137	1.134	...	0.092	0.270	0.045	0.080	0.165	0.120	0
2013-02-19 00:00:00	0.317	0.413	0.102	0.232	1.038	0.096	0.075	0.154	0.140	1.634	...	0.074	0.682	0.049	0.069	0.090	0.190	0
2013-02-23 00:00:00	0.621	0.100	0.051	0.220	0.178	0.232	0.027	0.163	0.164	0.768	...	0.073	0.454	0.040	0.062	0.033	0.255	0
2013-02-24 00:00:00	0.508	0.078	0.038	0.180	0.534	0.056	0.067	0.164	0.131	0.679	...	0.068	0.412	0.021	0.063	0.020	0.360	0
2013-02-25 00:00:00	0.496	0.064	0.210	0.120	0.104	0.081	0.056	0.154	0.151	0.689	...	0.080	0.421	0.038	0.124	0.084	0.203	0
2013-03-03 00:00:00	0.543	0.142	0.059	0.247	0.482	0.420	0.099	0.203	0.600	0.670	...	0.073	0.238	0.041	0.095	0.189	0.181	0
2013-03-04 00:00:00	0.320	0.135	0.095	0.085	0.484	0.200	0.048	0.160	0.125	1.036	...	0.072	0.401	0.036	0.082	0.021	0.245	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
2013-11-12 00:00:00	1.057	0.980	0.071	0.230	0.124	0.080	0.000	0.256	0.635	0.665	...	0.077	0.116	0.054	0.101	0.018	0.156	0
2013-11-13 00:00:00	0.297	1.058	0.035	0.091	0.599	0.082	0.000	0.894	0.400	0.805	...	0.077	0.356	0.041	0.108	0.019	0.138	0
2013-11-15 00:00:00	0.405	1.063	0.077	0.093	0.202	0.436	0.000	0.174	0.434	0.531	...	0.076	0.175	0.048	0.085	0.019	0.230	0
2013-11-16 00:00:00	0.388	1.073	0.036	0.078	0.590	0.089	0.000	0.157	0.388	0.629	...	0.085	0.131	0.047	0.098	0.019	0.088	0
2013-11-17 00:00:00	0.456	1.053	0.045	0.211	0.642	0.613	0.000	0.217	0.519	0.648	...	0.161	0.145	0.041	0.134	0.161	0.268	0
2013-11-18 00:00:00	0.563	1.052	0.105	0.204	1.084	0.090	0.000	0.166	0.210	0.936	...	0.195	0.242	0.040	0.102	0.019	0.176	0
2013-11-19 00:00:00	0.676	1.040	0.052	0.098	0.559	0.095	0.000	0.264	0.470	0.587	...	0.169	0.120	0.026	0.099	0.069	0.172	0
2013-11-20 00:00:00	0.570	1.099	0.039	0.090	0.200	0.074	0.000	0.164	0.368	0.520	...	0.167	0.341	0.041	0.067	0.025	0.124	0
2013-11-22 00:00:00	0.387	1.121	0.099	0.086	0.740	0.090	0.000	0.267	0.244	0.498	...	0.146	0.309	0.037	0.069	0.069	0.165	0
2013-11-23 00:00:00	0.363	1.087	0.069	0.102	0.542	0.093	0.000	0.456	0.583	0.595	...	0.112	0.416	0.018	0.205	0.019	0.245	0

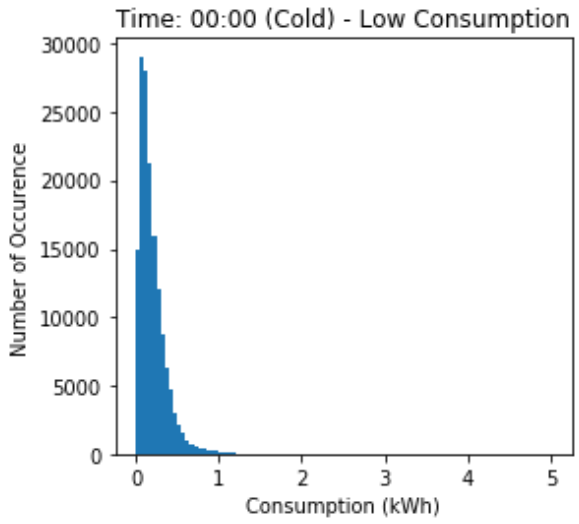
	N0000	N0001	N0002	N0003	N0004	N0005	N0006	N0007	N0008	N0009	...	N4162	N4163	N4164	N4165	N4166	N4167	N
GMT																		
2013-11-24 00:00:00	0.239	1.092	0.105	0.229	0.605	0.368	0.000	0.203	0.281	2.118	...	0.075	0.169	0.031	0.093	0.142	0.247	0
2013-11-25 00:00:00	0.259	1.085	0.113	0.203	0.488	0.096	0.000	0.746	0.154	1.232	...	0.075	0.328	0.040	0.069	0.086	0.098	0
2013-12-02 00:00:00	0.492	1.042	0.036	0.108	0.676	0.066	0.000	0.169	0.423	0.540	...	0.069	0.159	0.043	0.076	0.019	0.341	0
2013-12-03 00:00:00	0.262	1.049	0.082	0.188	0.380	0.092	0.000	0.267	0.670	0.477	...	0.079	0.334	0.051	0.098	0.071	0.143	0
2013-12-05 00:00:00	0.328	1.064	0.103	0.222	0.737	0.122	0.000	0.156	0.211	0.558	...	0.081	0.317	0.044	0.074	0.181	0.337	0
2013-12-06 00:00:00	0.470	1.060	0.085	0.135	0.251	0.098	0.000	0.161	0.589	0.538	...	0.074	0.346	0.041	0.070	0.020	0.133	0
2013-12-09 00:00:00	0.144	1.053	0.046	0.151	0.816	0.098	0.000	0.156	0.238	0.860	...	0.078	0.339	0.044	0.102	0.184	0.292	0
2013-12-10 00:00:00	0.404	1.046	0.036	0.359	0.349	0.091	0.000	0.161	0.228	0.578	...	0.077	0.167	0.046	0.083	0.038	0.101	0
2013-12-11 00:00:00	0.617	0.674	0.059	0.311	0.567	0.086	0.000	0.176	0.175	0.676	...	0.088	0.127	0.037	0.091	0.021	0.196	0
2013-12-14 00:00:00	0.190	0.656	0.099	0.160	0.272	0.133	0.000	0.377	0.369	0.620	...	0.073	0.247	0.036	0.099	0.021	0.222	0
2013-12-17 00:00:00	0.391	0.641	0.100	0.211	0.690	0.044	0.000	0.170	0.428	0.638	...	0.093	0.151	0.033	0.078	0.072	0.092	0
2013-12-18 00:00:00	0.620	0.651	0.094	0.431	0.720	0.138	0.000	0.156	0.594	0.532	...	0.091	0.261	0.027	0.085	0.080	0.109	0
2013-12-21 00:00:00	0.549	0.653	0.041	0.221	0.818	0.275	0.000	0.167	0.564	0.612	...	0.074	0.072	0.042	0.100	0.021	0.184	0
2013-12-22 00:00:00	0.334	0.652	0.107	0.290	0.921	0.690	0.000	0.173	0.473	0.587	...	0.092	0.070	0.046	0.090	0.029	0.224	0
2013-12-23 00:00:00	0.569	0.647	0.066	0.241	0.541	0.099	0.157	0.173	0.681	0.574	...	0.159	0.069	0.050	0.066	0.021	0.261	0
2013-12-24 00:00:00	0.200	0.652	0.039	0.200	0.681	0.288	0.068	0.719	0.190	0.606	...	0.102	0.080	0.030	0.055	0.057	0.158	0
2013-12-25 00:00:00	0.260	0.631	0.040	0.641	0.712	0.158	0.069	0.157	0.154	0.639	...	0.142	0.069	0.035	0.134	0.021	0.140	0
2013-12-27 00:00:00	0.380	0.622	0.107	0.376	0.774	0.161	0.086	0.154	0.343	0.590	...	0.114	0.067	0.035	0.079	0.020	0.403	0
2013-12-28 00:00:00	0.479	0.648	0.098	0.215	0.710	1.021	0.047	0.160	0.490	1.208	...	0.092	0.074	0.027	0.145	0.020	0.232	0
2013-12-31 00:00:00	0.509	0.647	0.066	0.288	0.852	0.075	0.110	0.164	0.297	0.682	...	0.172	0.076	0.021	0.092	0.036	0.311	0

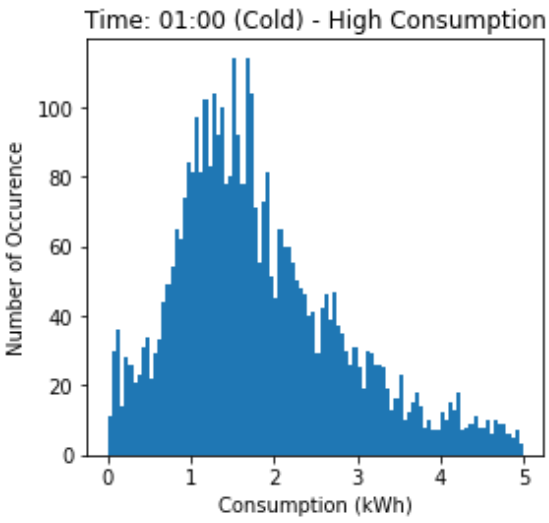
83 rows × 3720 columns

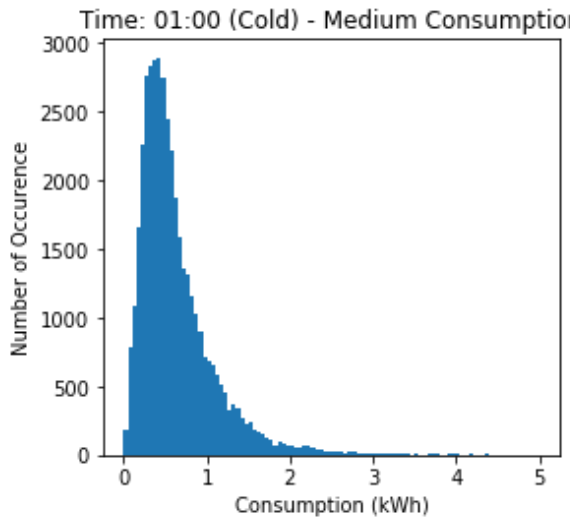
In [33]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
n = 3 # cluster numbers
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
fig_all = plt.figure(figsize = (13,90))
for i in range(24):
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # data used for analyzing consumption distribution
#         t = t.dropna(axis=1)
        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_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:0
0')]['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' + ' (Cold) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Number of Occurence')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # data used for analyzing consumption distribution
#         t = t.dropna(axis=1)
        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_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['T
empC'].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' + ' (Cold) - ' + str(cons_type[k]))
            ax.set_xlabel('Consumption (kWh)')
            ax.set_ylabel('Number of Occurence')
plt.tight_layout()
```

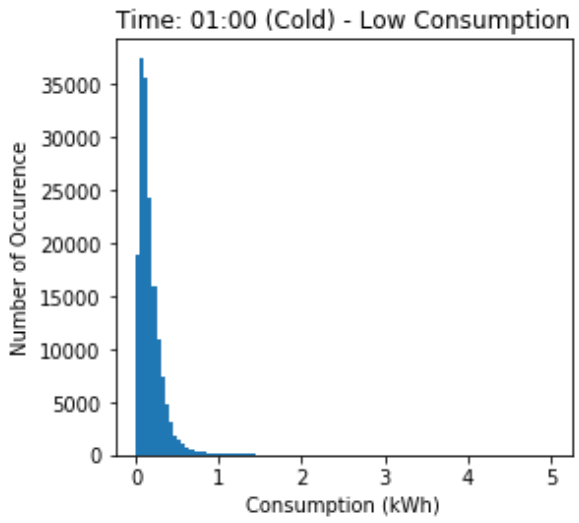

Time: 00:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

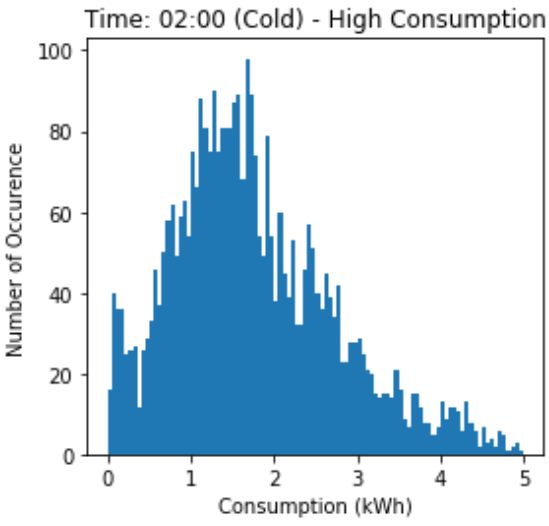

Time: 00:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

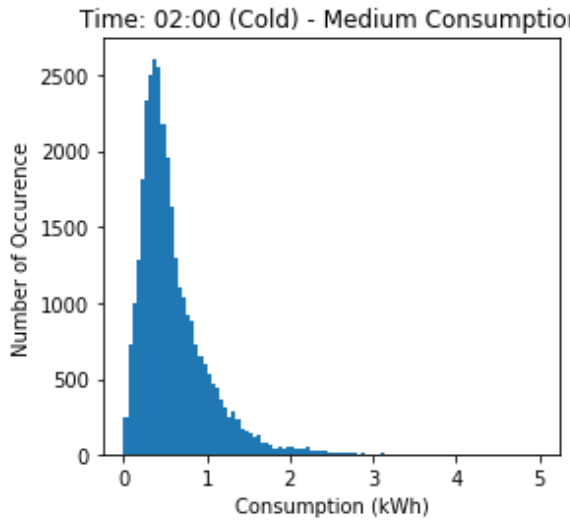

Time: 00:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

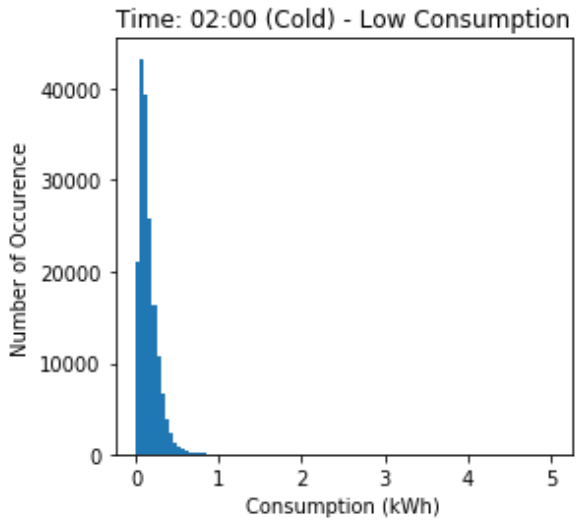

Time: 01:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

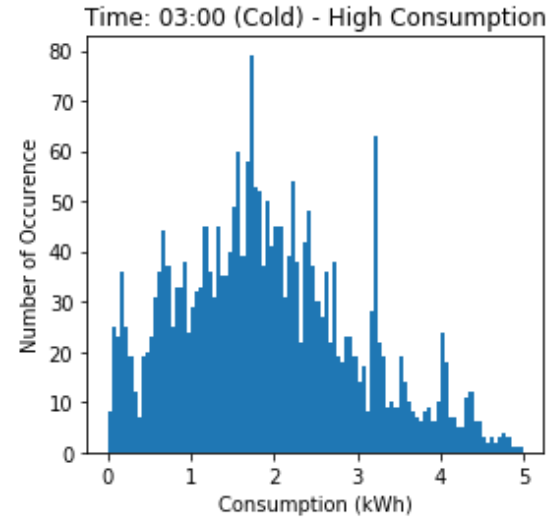

Time: 01:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

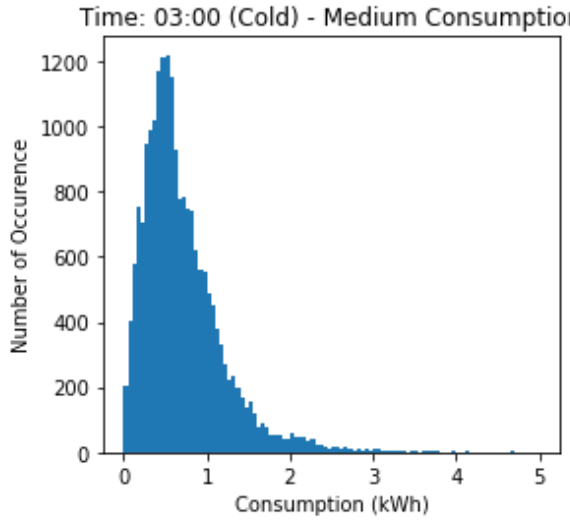

Time: 01:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

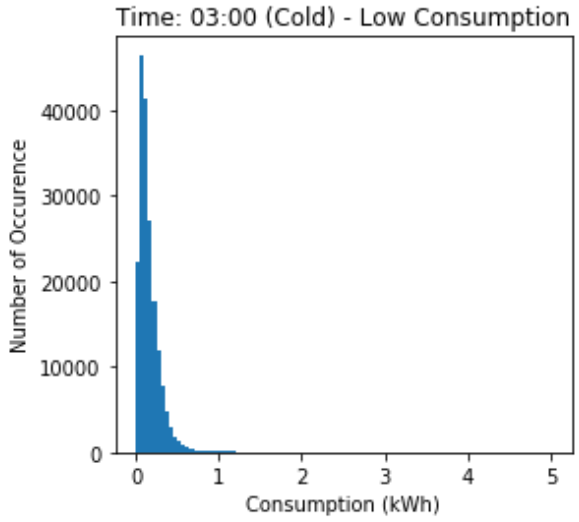

Time: 02:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

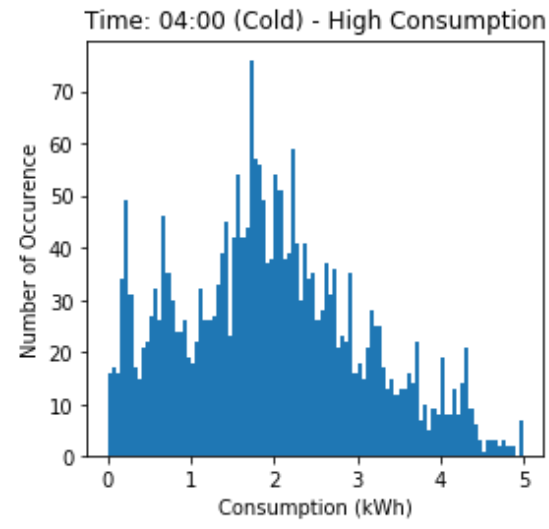

Time: 02:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

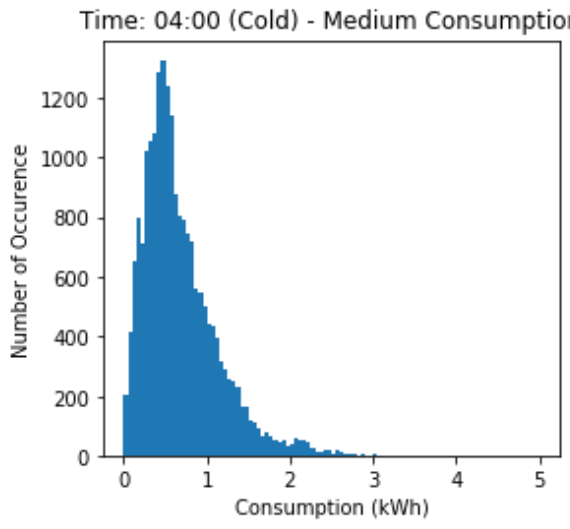

Time: 02:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

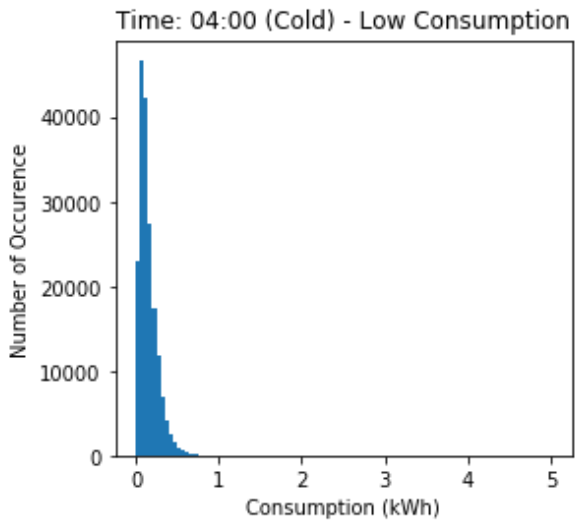

Time: 03:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

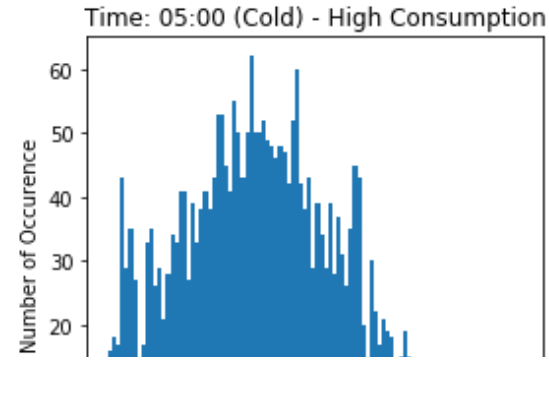

Time: 03:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

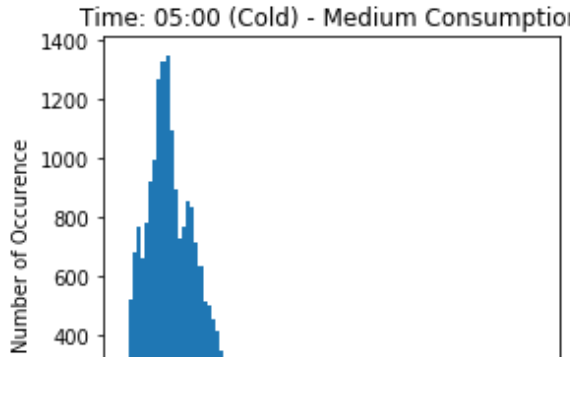

Time: 03:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

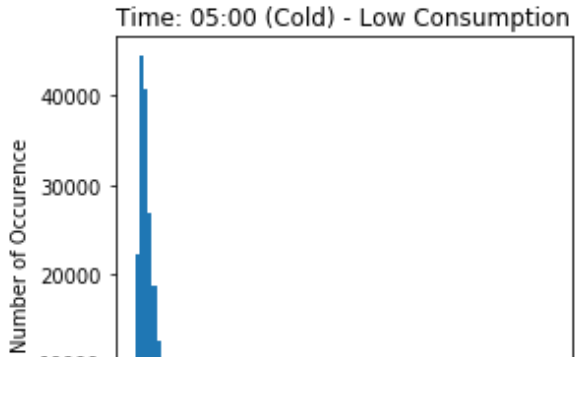

Time: 04:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

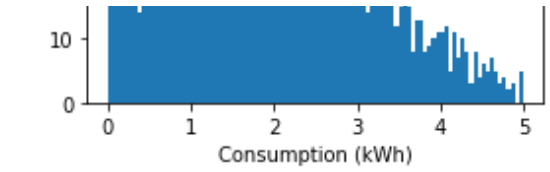

Time: 04:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

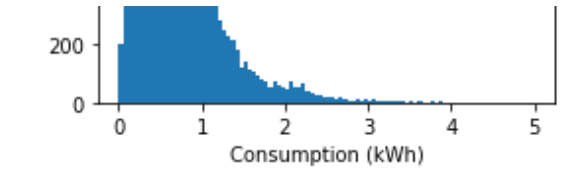

Time: 04:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

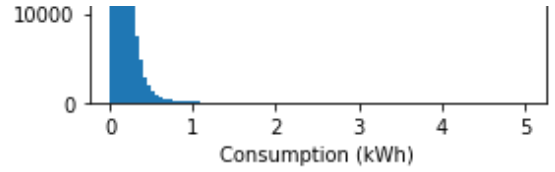

Time: 05:00 (Cold) - High Consumption
Number of Occurence

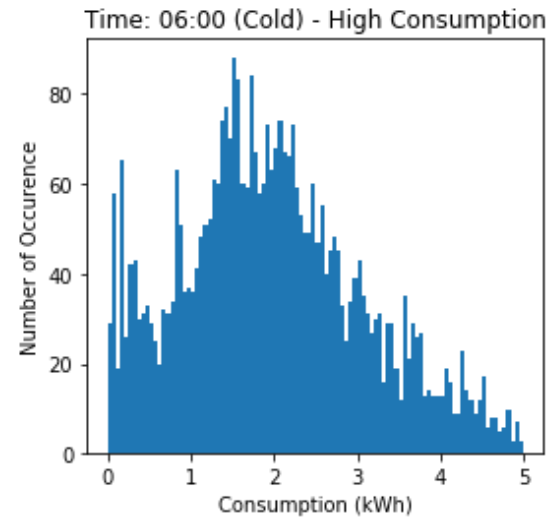

Time: 05:00 (Cold) - Medium Consumption
Number of Occurence

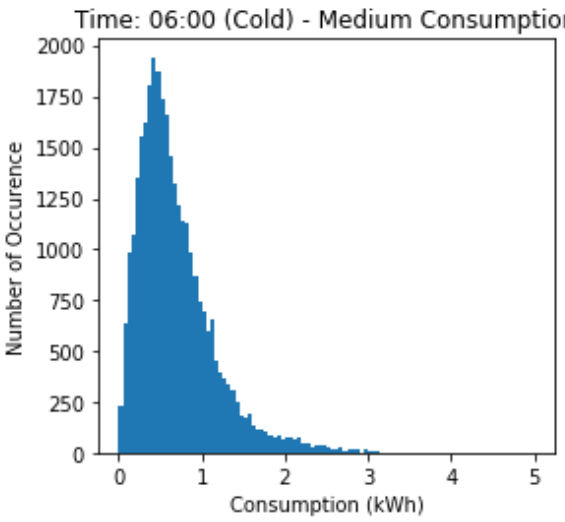

Time: 05:00 (Cold) - Low Consumption
Number of Occurence

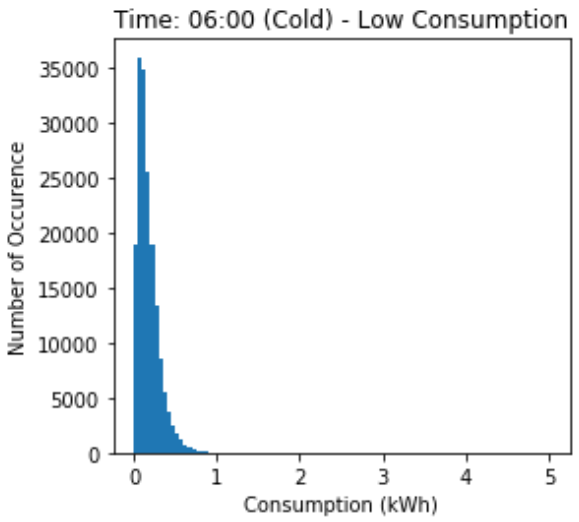

Consumption (kWh)

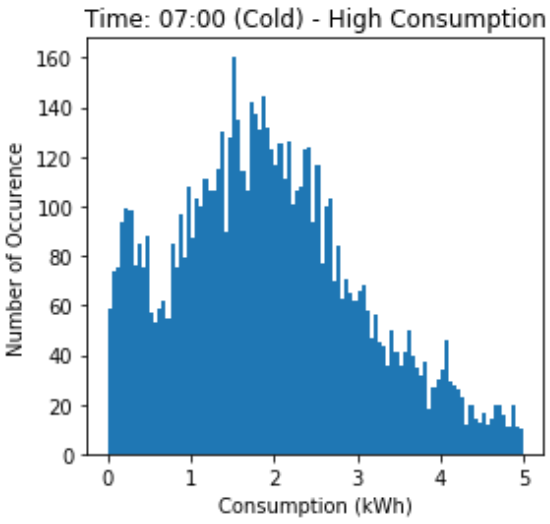

Consumption (kWh)

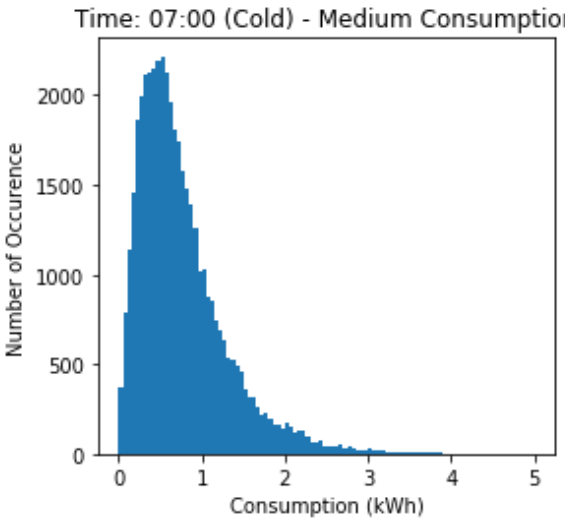

Consumption (kWh)

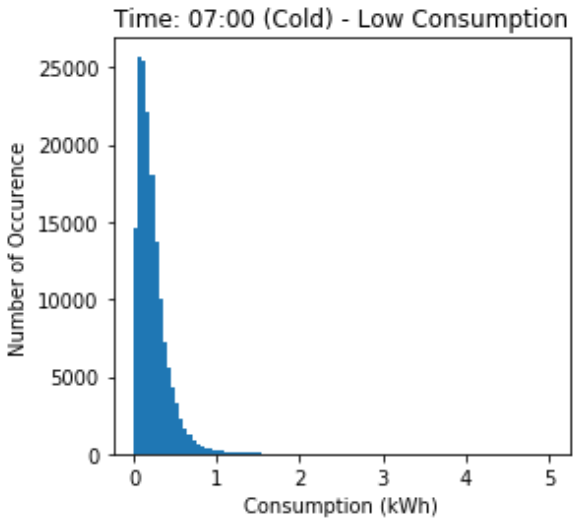

Time: 06:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

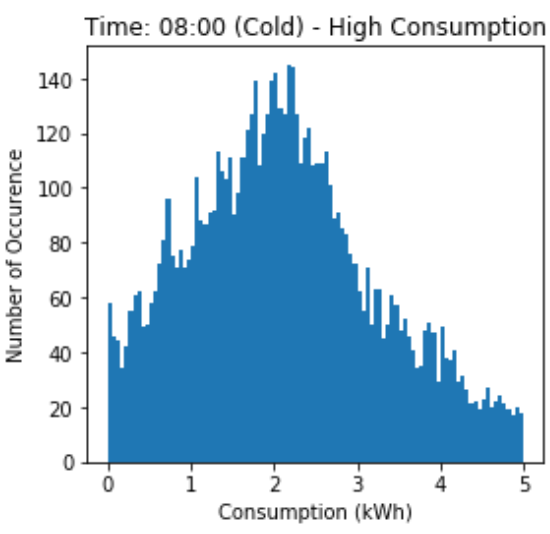

Time: 06:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

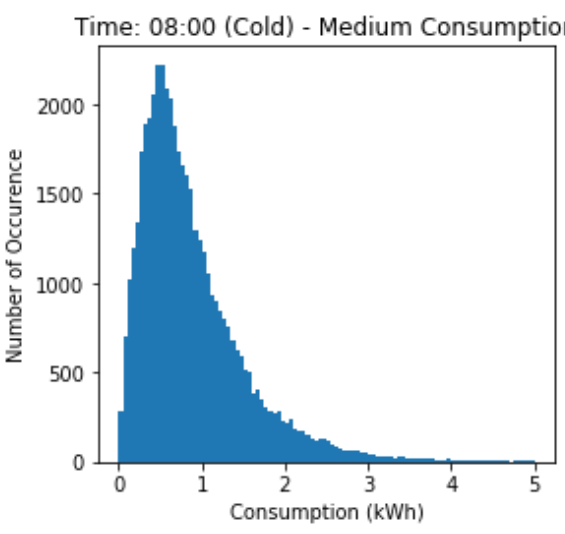

Time: 06:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

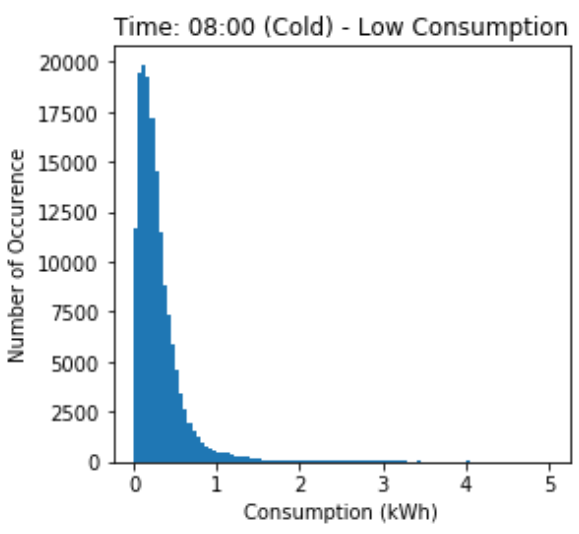

Time: 07:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

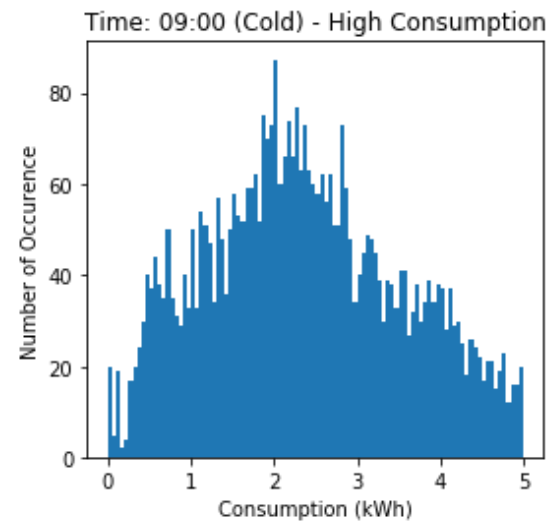

Time: 07:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

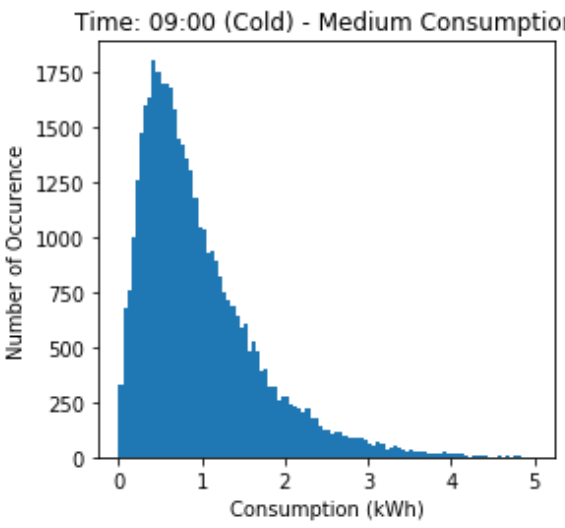

Time: 07:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

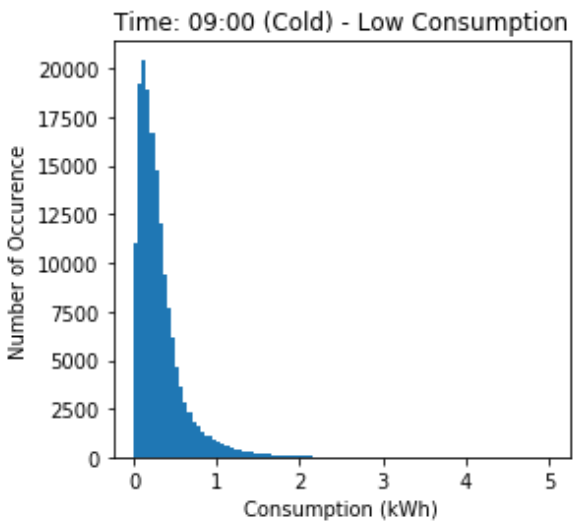

Time: 08:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

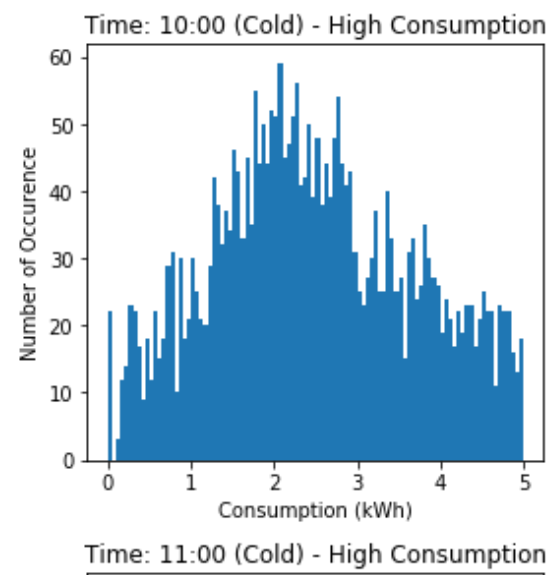

Time: 08:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

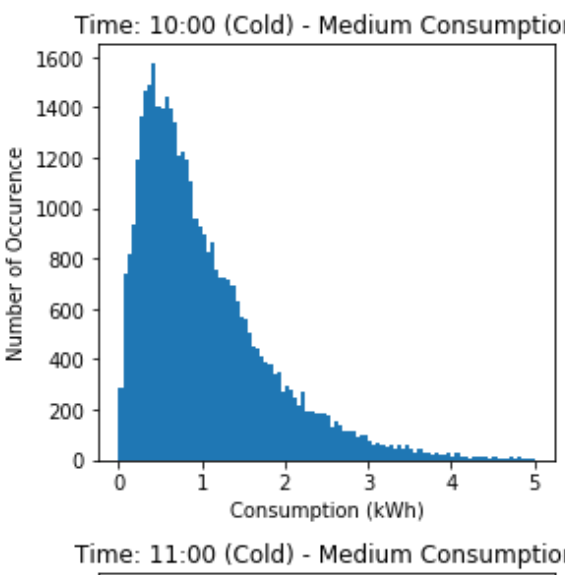

Time: 08:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

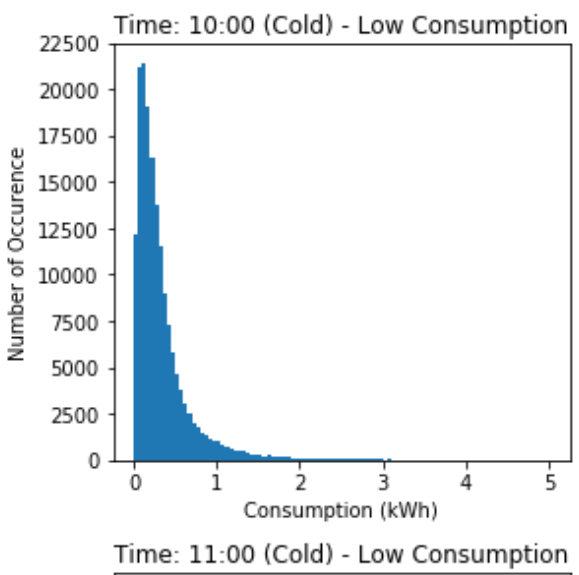

Time: 09:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

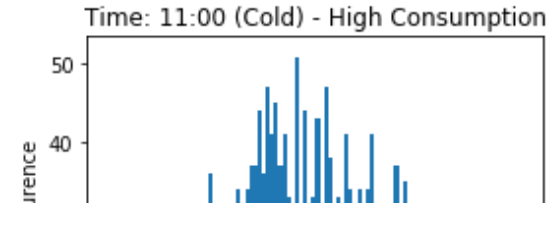

Time: 09:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

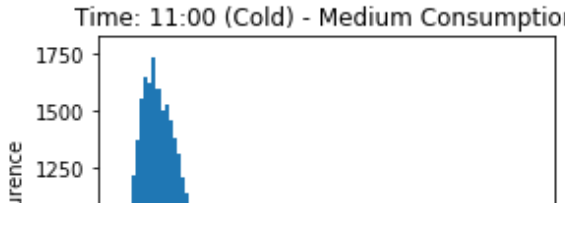

Time: 09:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

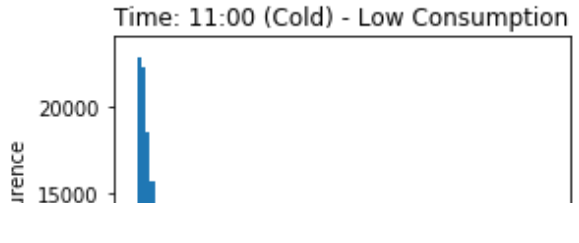

Time: 10:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

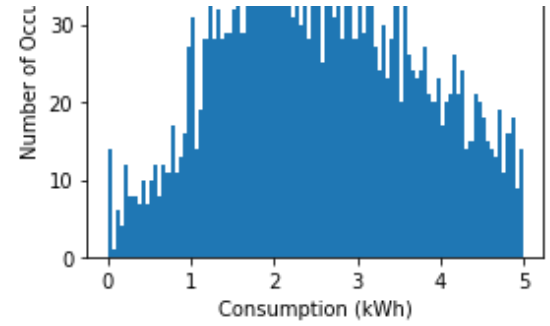

Time: 10:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)


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

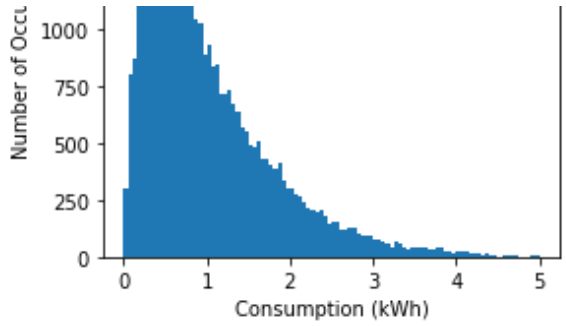

Time: 11:00 (Cold) - High Consumption


Time: 11:00 (Cold) - Medium Consumption

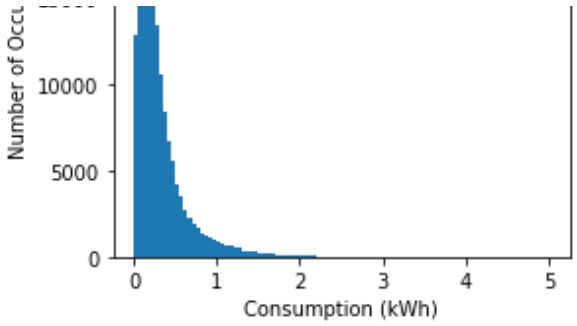

Time: 11:00 (Cold) - Low Consumption



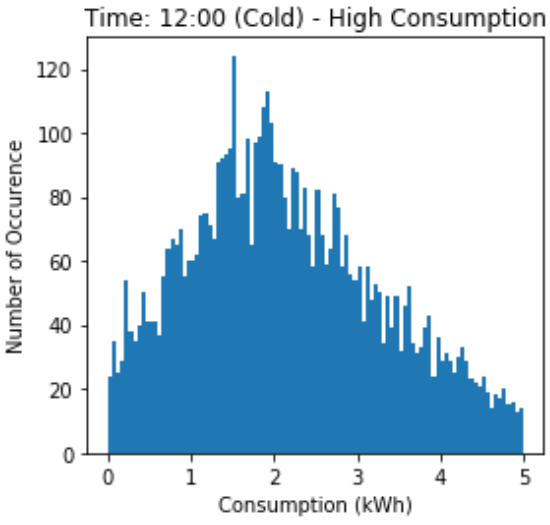
Time: 12:00 (Cold) - High Consumption



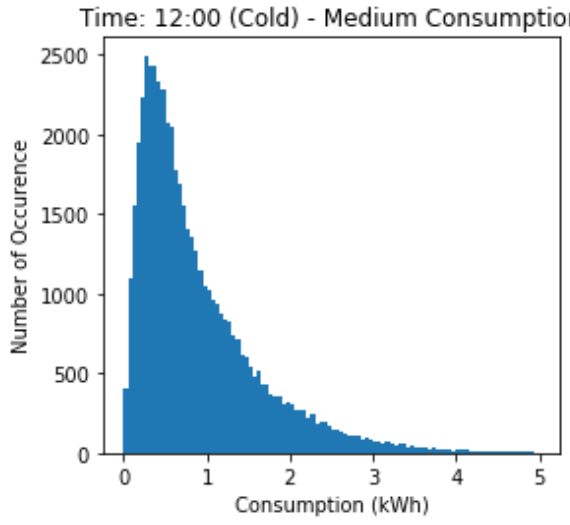
Time: 12:00 (Cold) - Medium Consumption



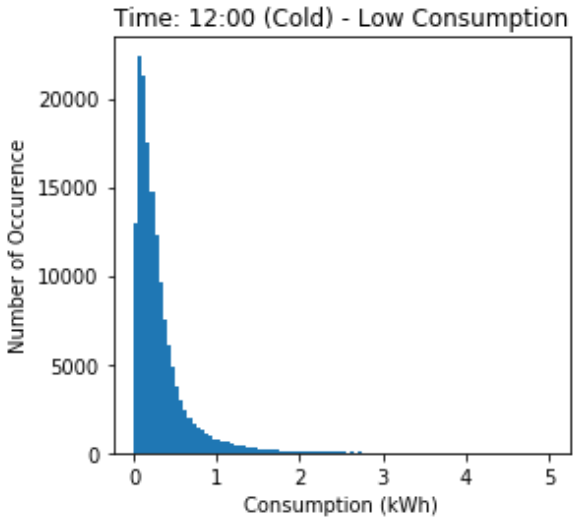
Time: 12:00 (Cold) - Low Consumption



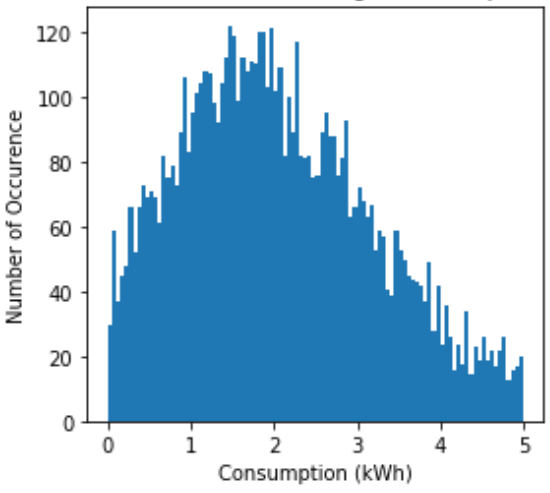
Time: 13:00 (Cold) - High Consumption



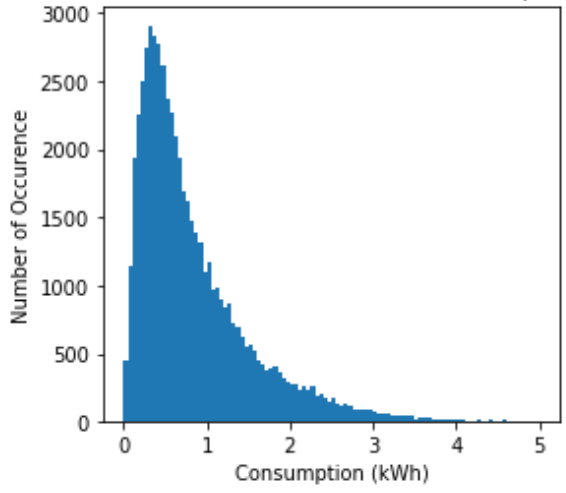
Time: 13:00 (Cold) - Medium Consumption



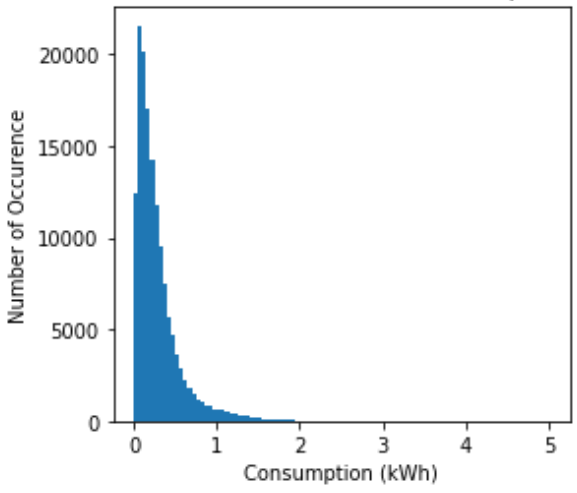
Time: 13:00 (Cold) - Low Consumption



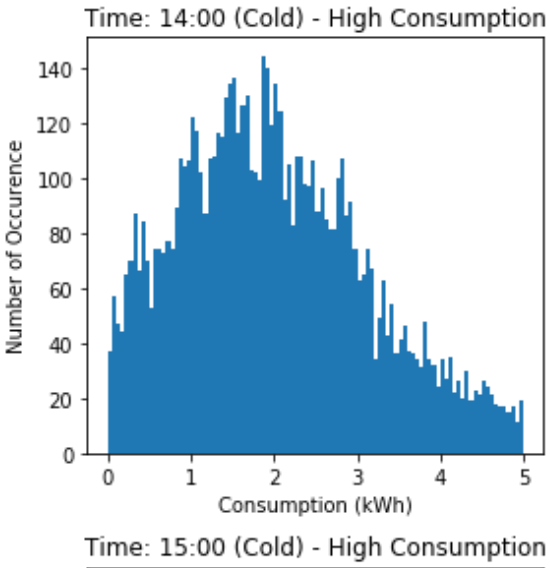
Time: 14:00 (Cold) - High Consumption



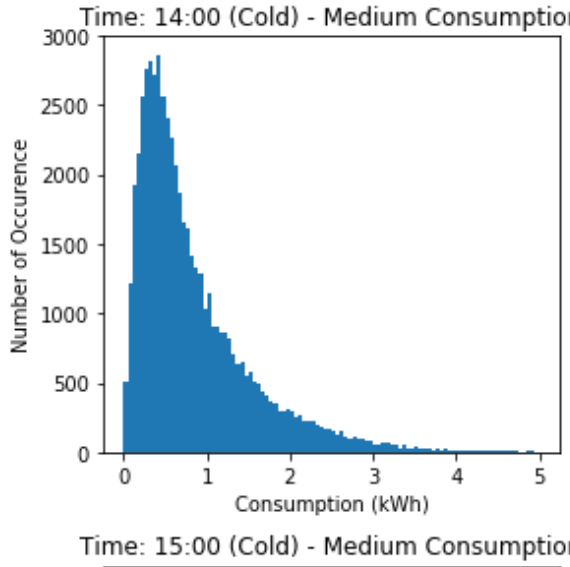
Time: 14:00 (Cold) - Medium Consumption



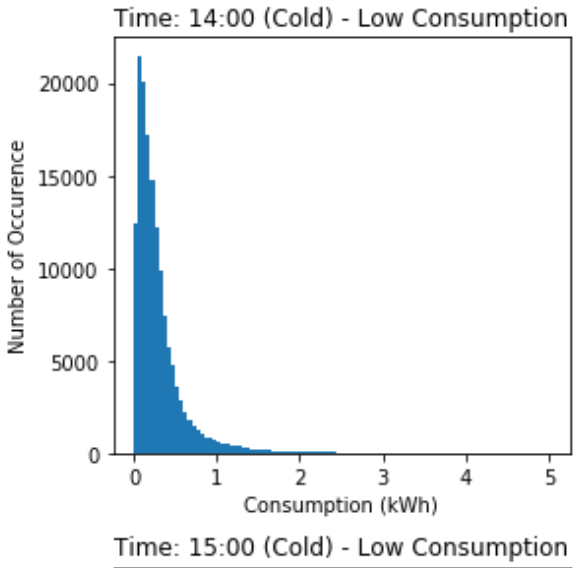
Time: 14:00 (Cold) - Low Consumption



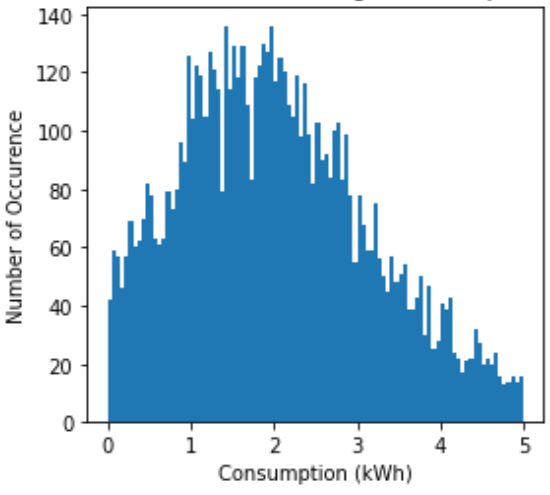
Time: 15:00 (Cold) - High Consumption



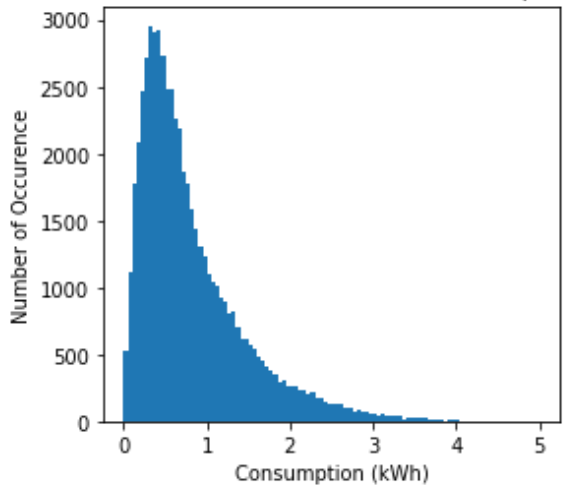
Time: 15:00 (Cold) - Medium Consumption



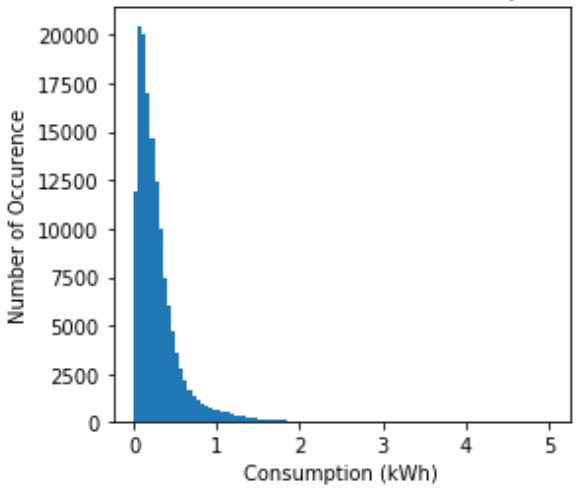
Time: 15:00 (Cold) - Low Consumption



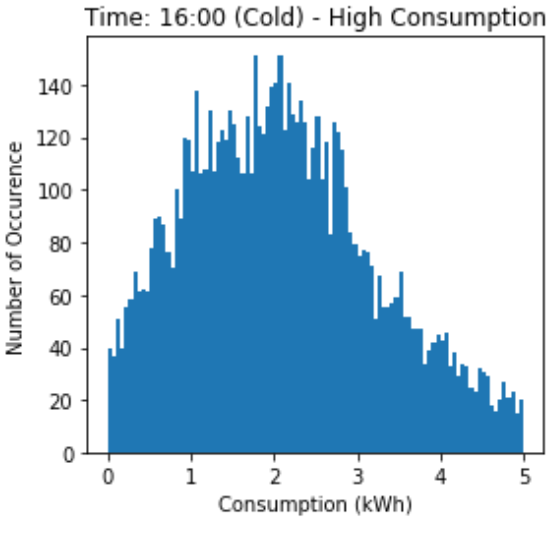
Time: 16:00 (Cold) - High Consumption



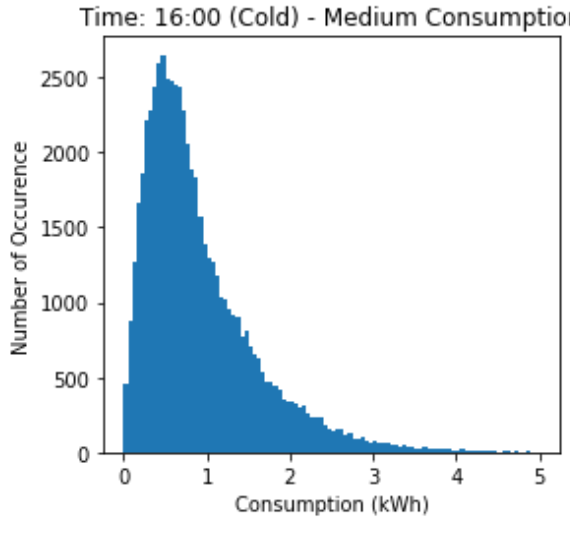
Time: 16:00 (Cold) - Medium Consumption



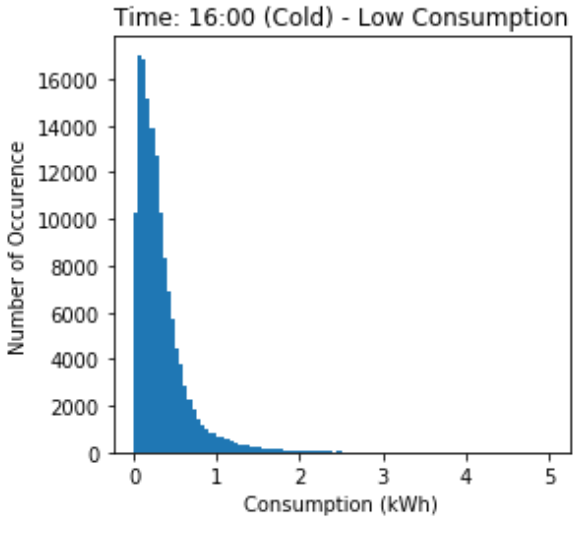
Time: 16:00 (Cold) - Low Consumption



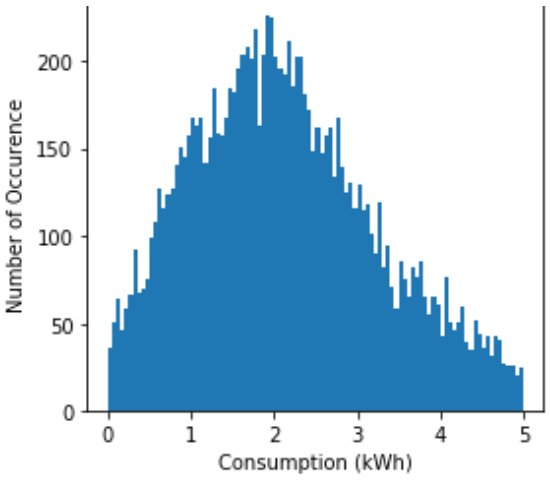
Time: 17:00 (Cold) - High Consumption

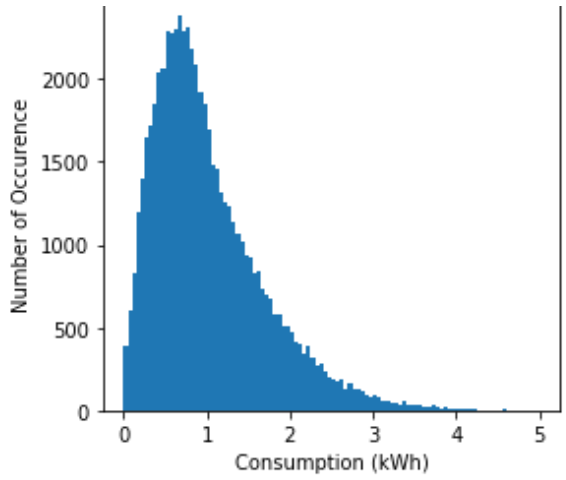


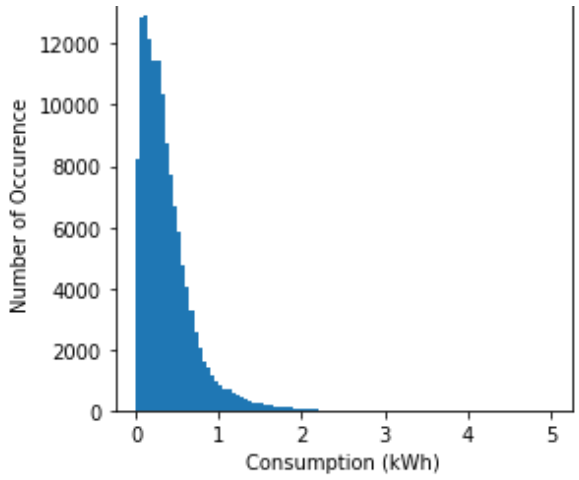
Time: 17:00 (Cold) - Medium Consumption

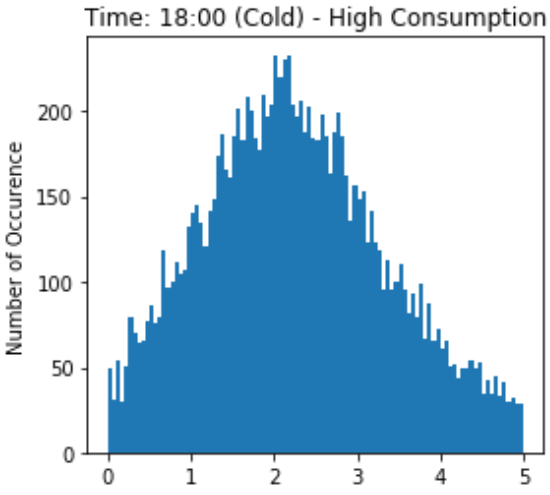


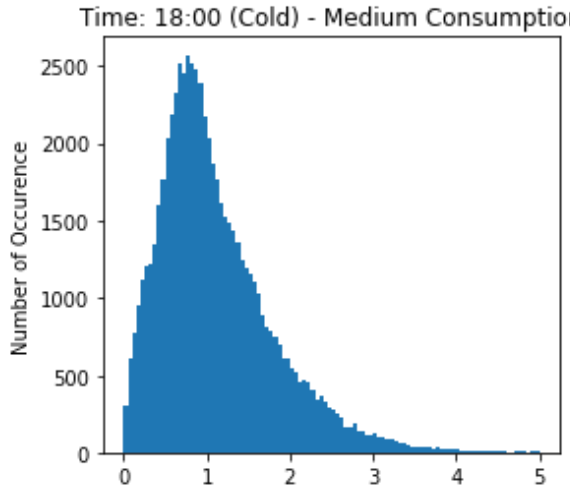
Time: 17:00 (Cold) - Low Consumption

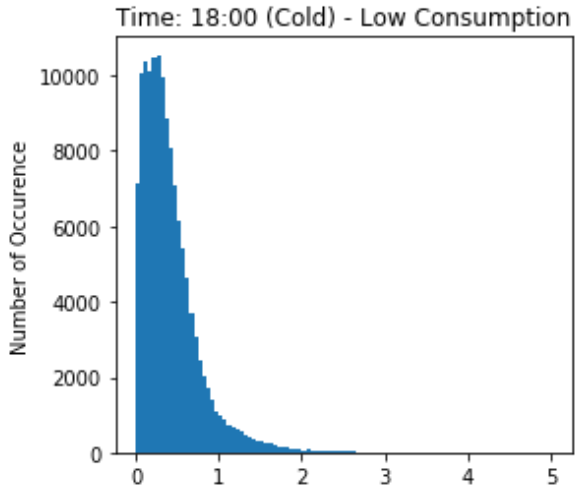

Number of Occurence
Consumption (kWh)

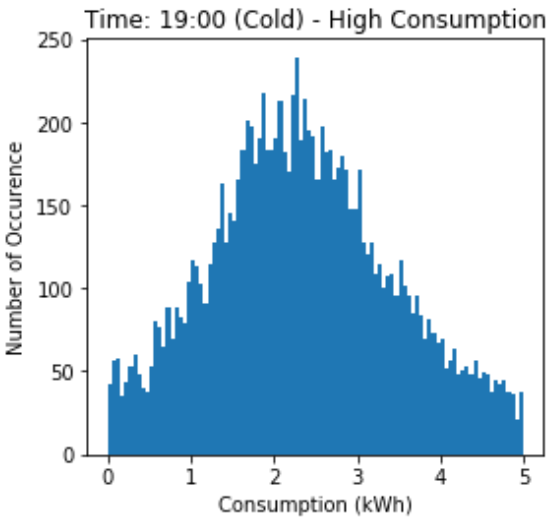

Number of Occurence
Consumption (kWh)

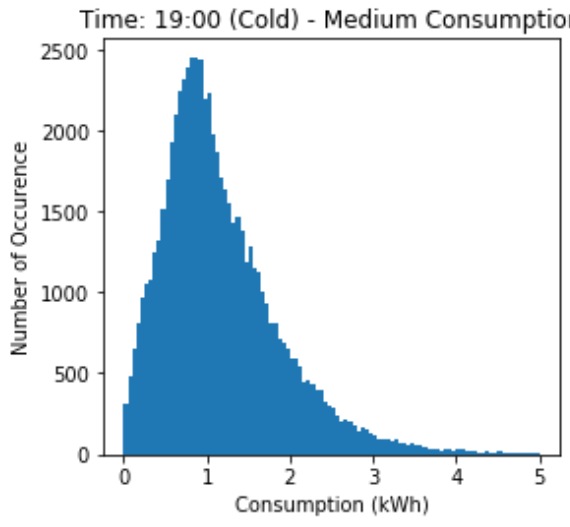

Number of Occurence
Consumption (kWh)

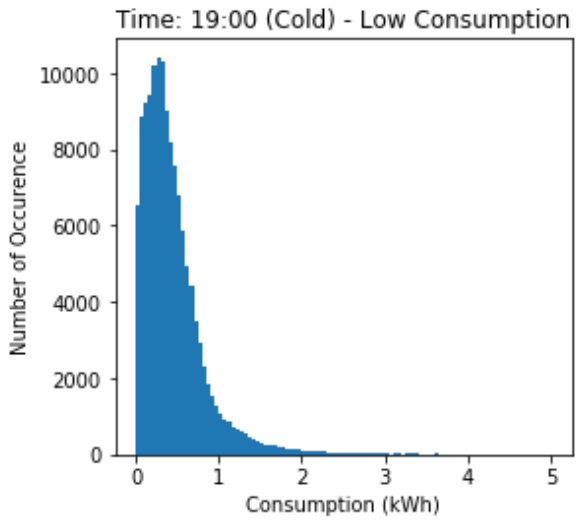

Time: 18:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

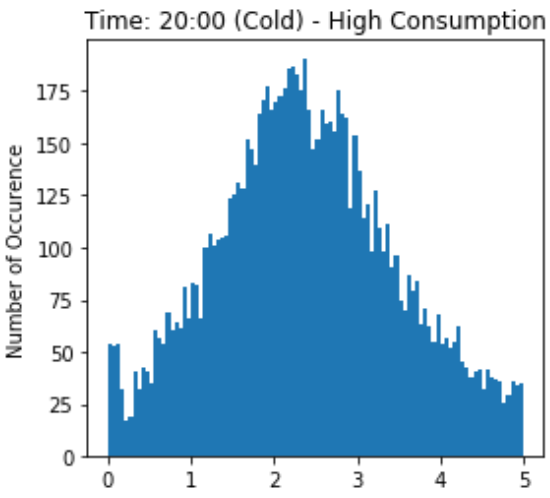

Time: 18:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

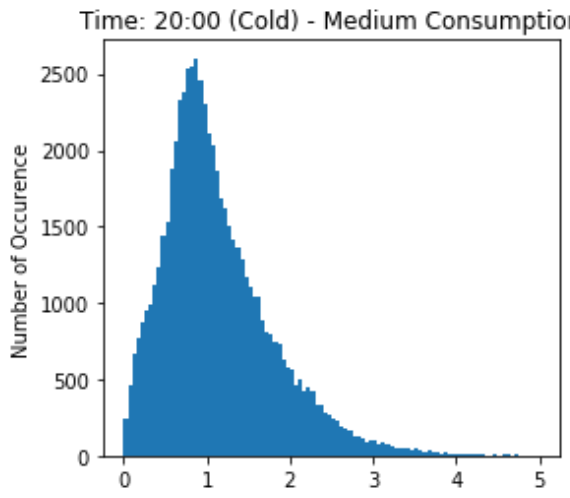

Time: 18:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

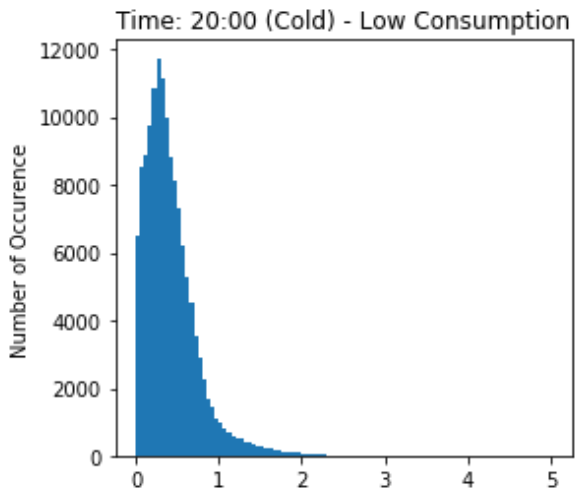

Time: 19:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

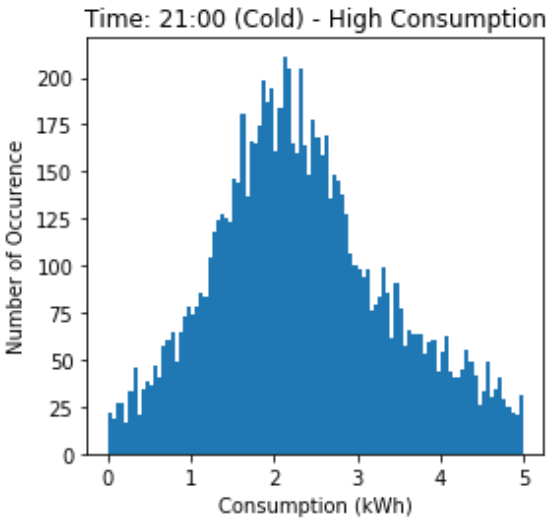

Time: 19:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

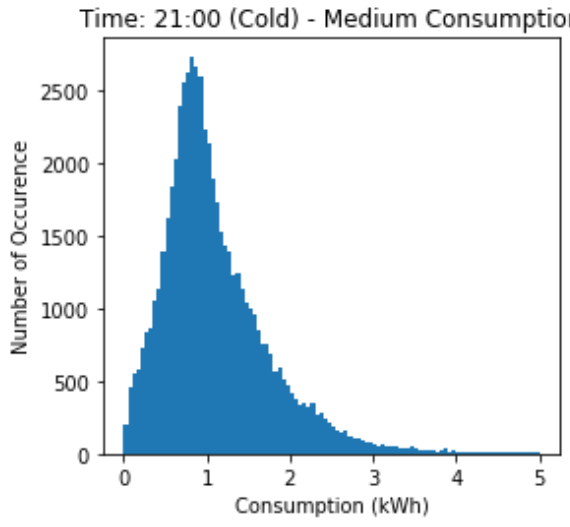

Time: 19:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

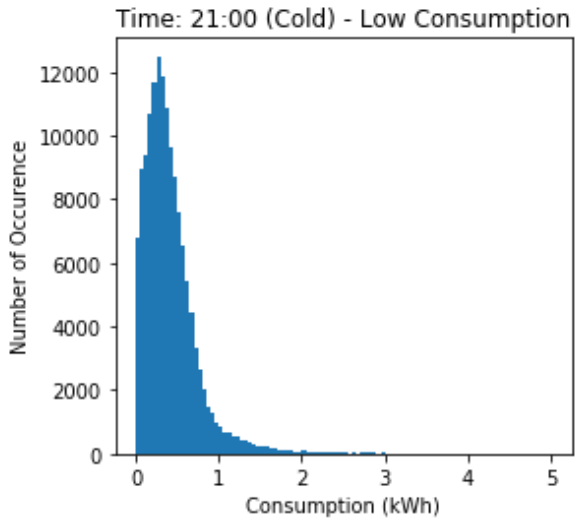

Time: 20:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

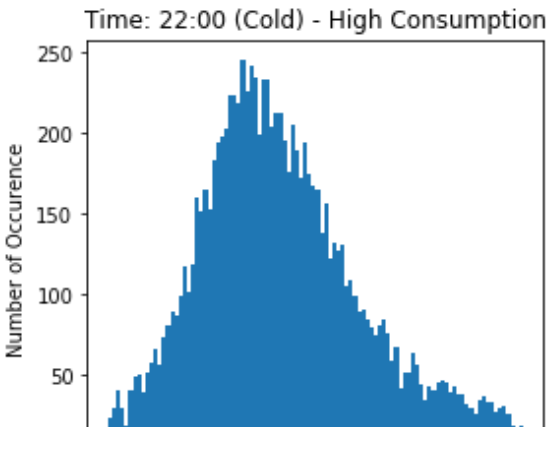

Time: 20:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)

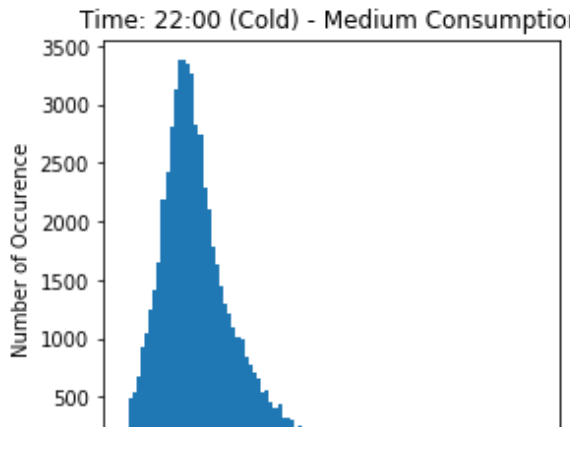

Time: 20:00 (Cold) - Low Consumption
Number of Occurence
Consumption (kWh)

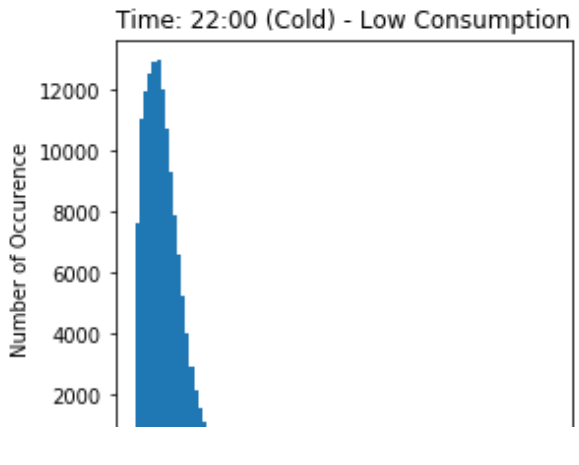

Time: 21:00 (Cold) - High Consumption
Number of Occurence
Consumption (kWh)

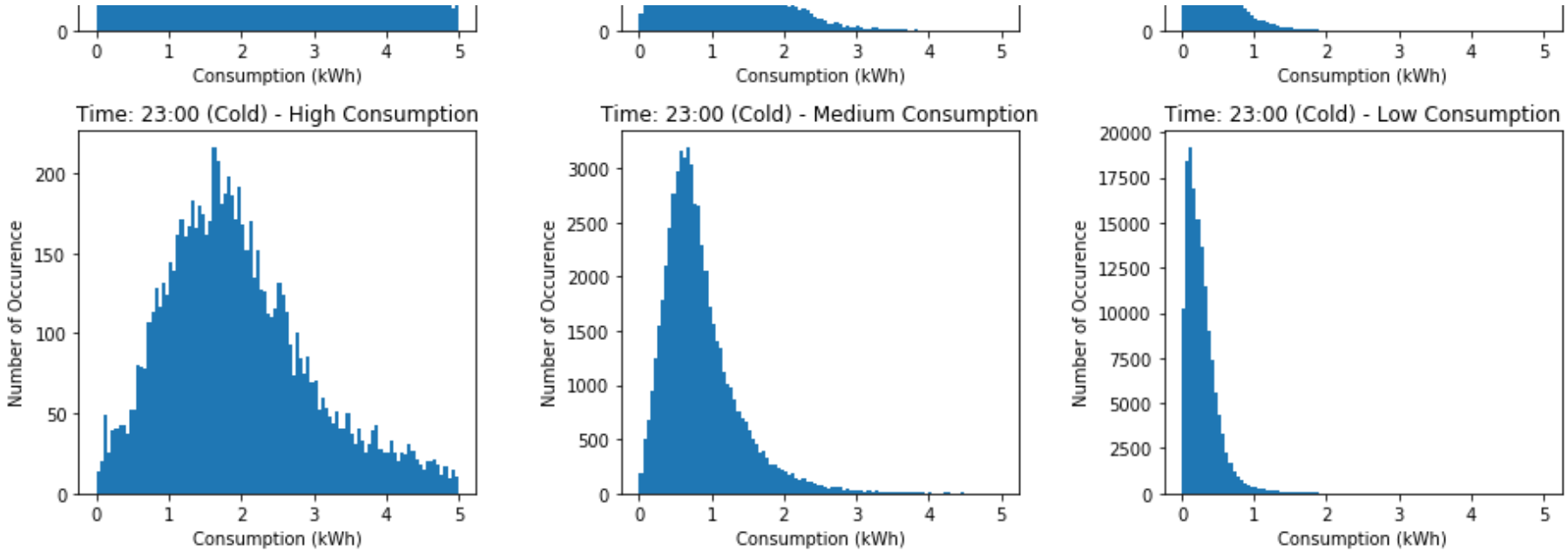

Time: 21:00 (Cold) - Medium Consumption
Number of Occurence
Consumption (kWh)


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


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

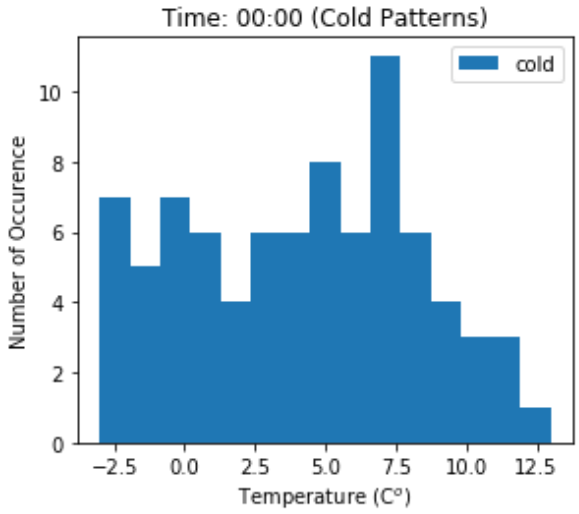

Time: 22:00 (Cold) - Medium Consumption
Number of Occurence

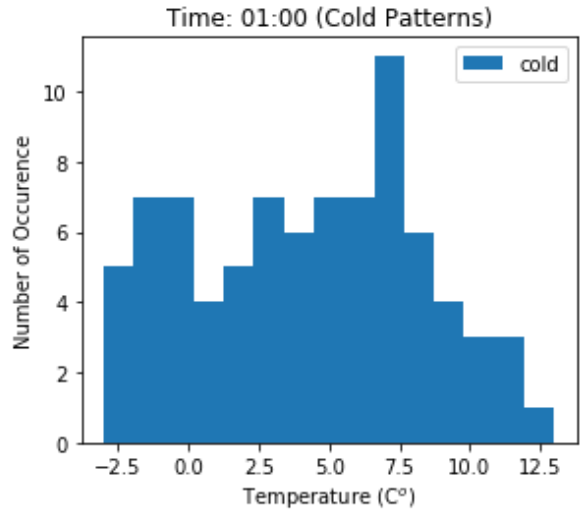

Time: 22:00 (Cold) - Low Consumption
Number of Occurence

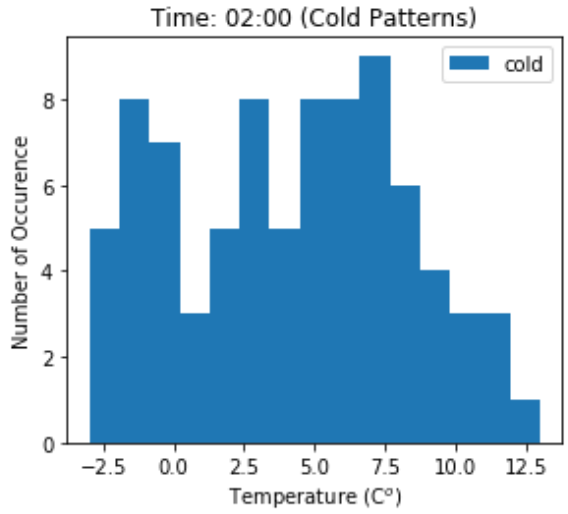


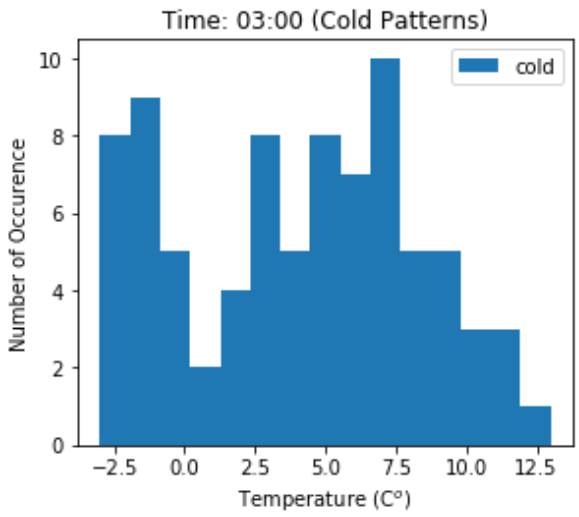
General days of a week distribution property of each cluster at each hour

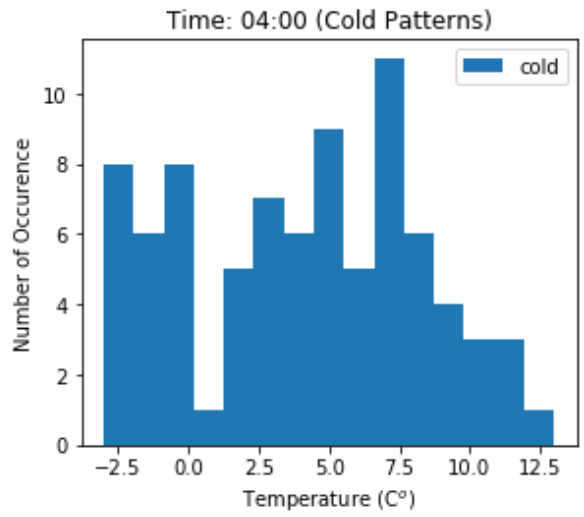
In [34]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (13,30))
for i in range(24):
    ax = fig_all.add_subplot(8, 3, i + 1)
    if i <= 9:
        ax.hist(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]['TempC'], bins=
15, label = 'cold')
        ax.set_title('Time: 0' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
    else:
        ax.hist(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['TempC'], bins=15, la
bel = 'cold')
        ax.set_title('Time: ' + str(i) + ':00' + ' (Cold Patterns)')
        ax.legend()
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Number of Occurence')
plt.tight_layout()
```

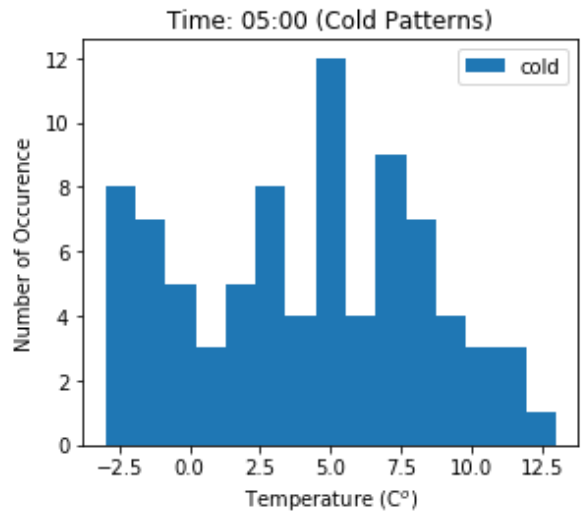

Time: 00:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

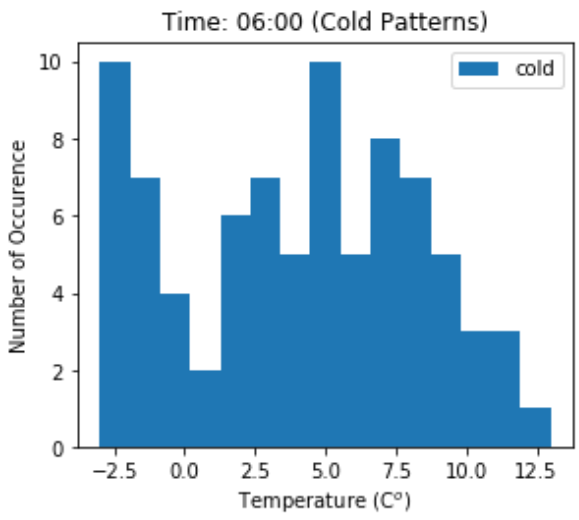

Time: 01:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

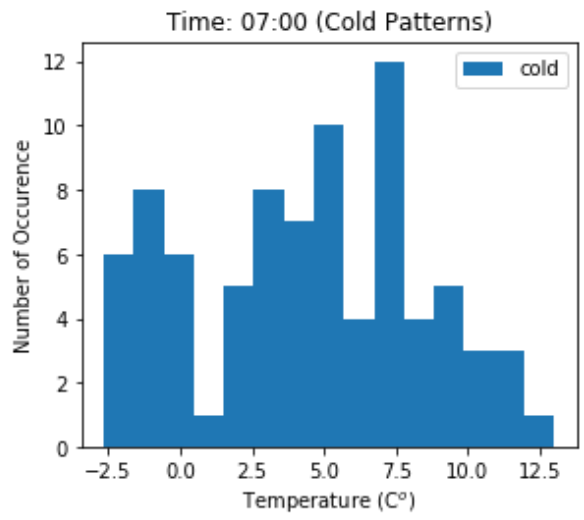

Time: 02:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

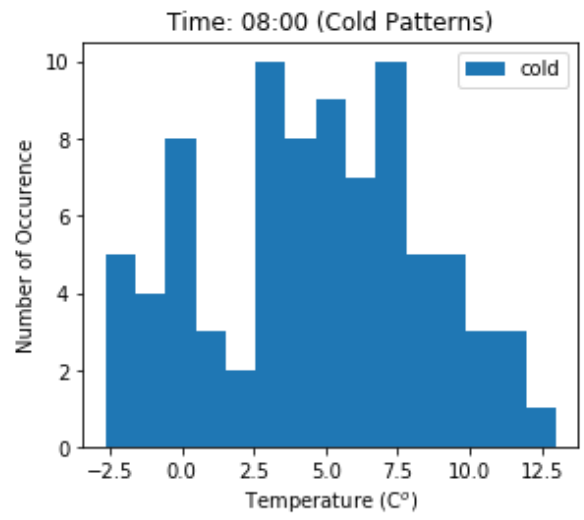

Time: 03:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

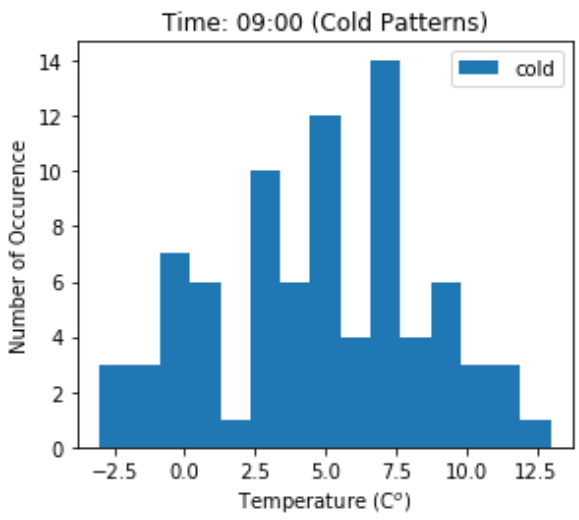

Time: 04:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

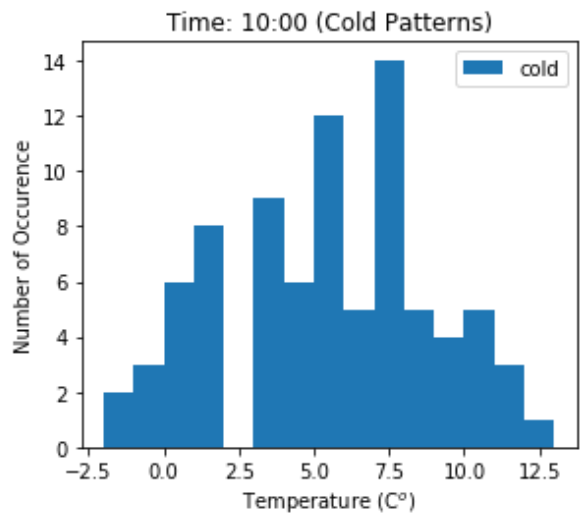

Time: 05:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

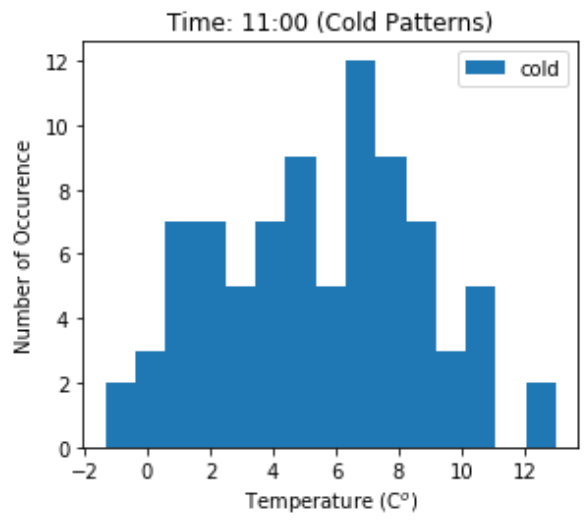

Time: 06:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

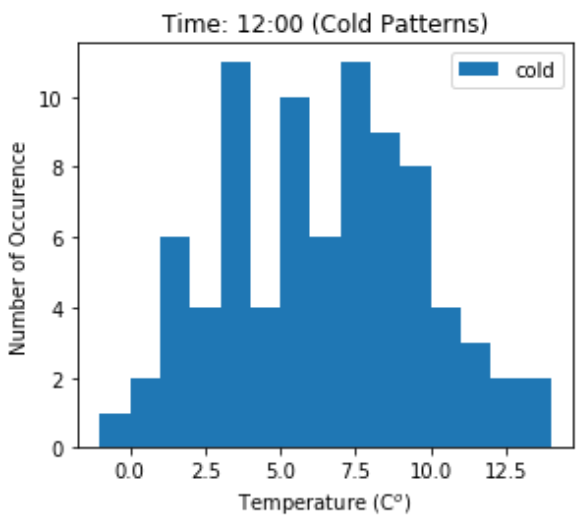

Time: 07:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

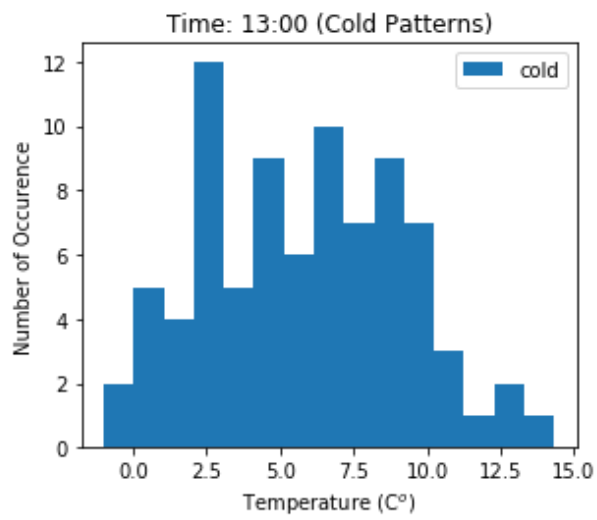

Time: 08:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

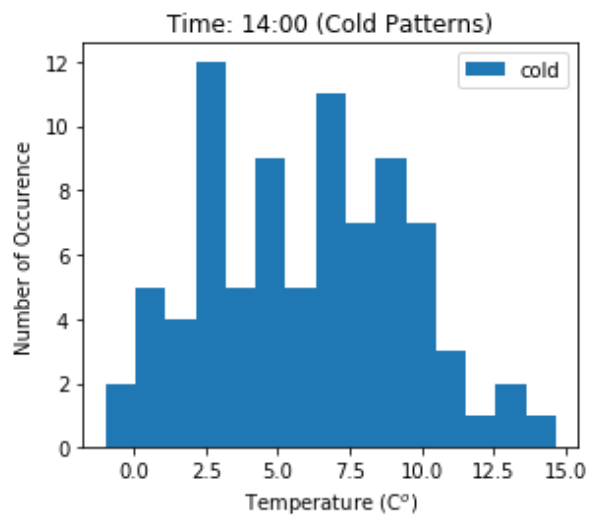

Time: 09:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

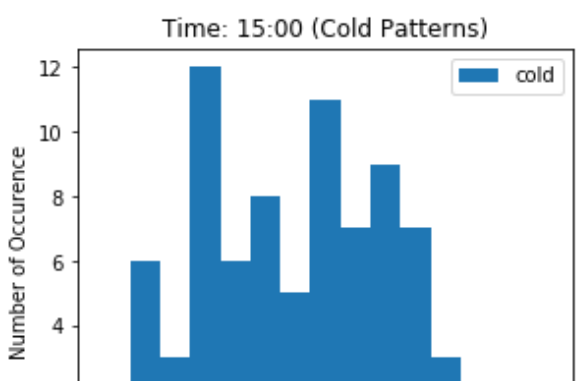

Time: 10:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

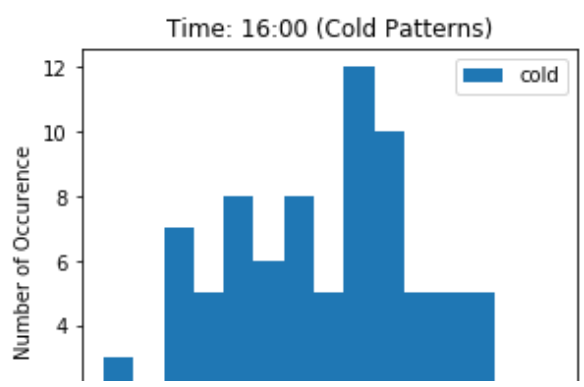

Time: 11:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

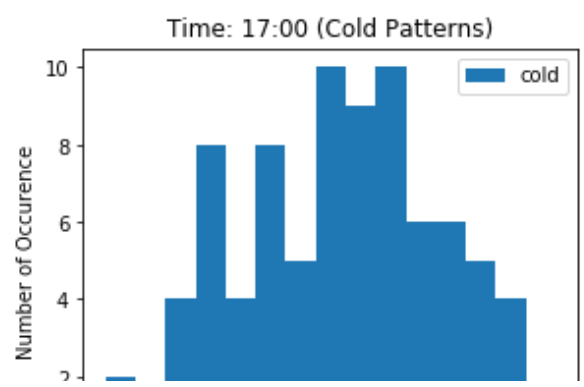

Time: 12:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)

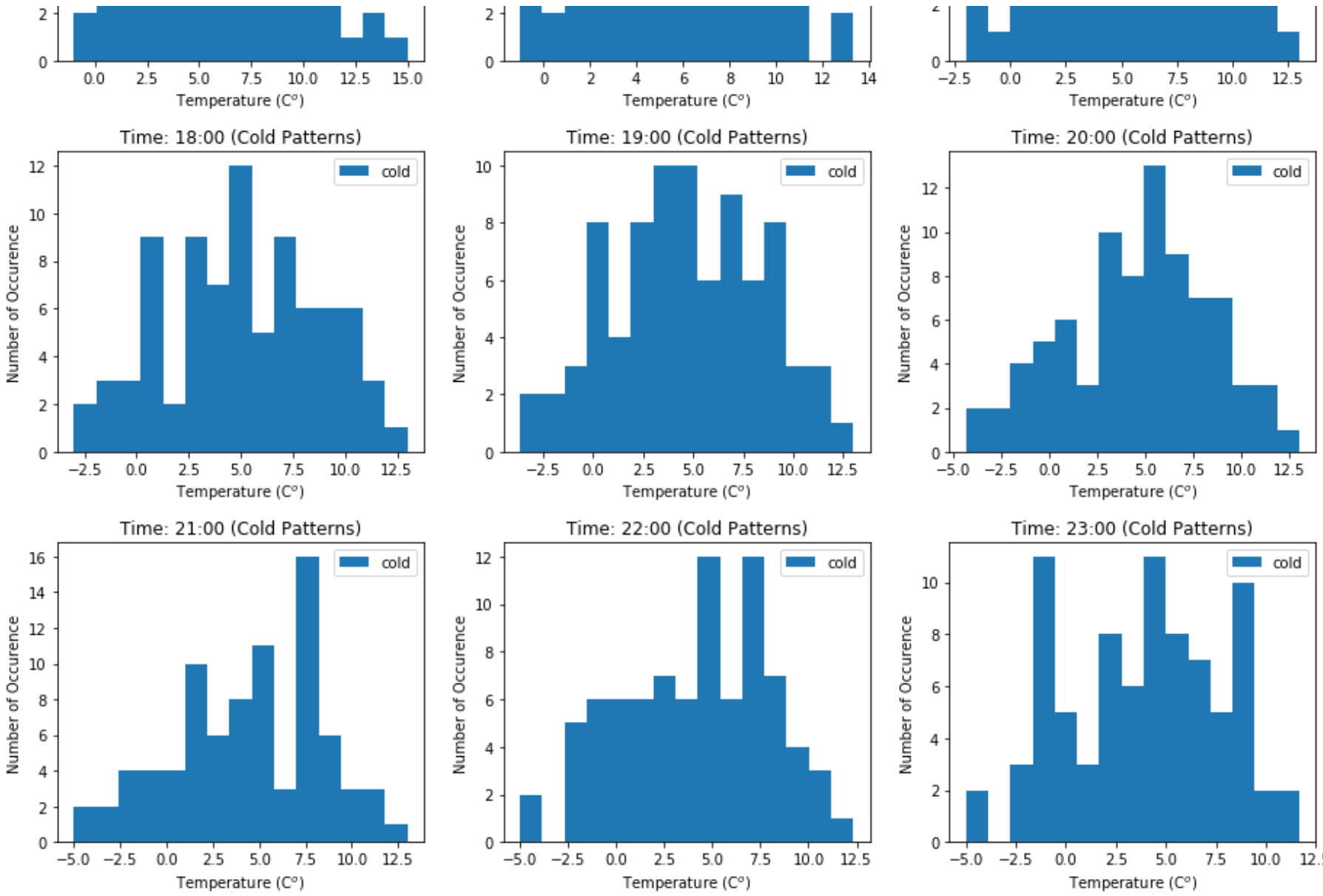

Time: 13:00 (Cold Patterns)
cold
Number of Occurence
Temperature (C°)


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


Time: 15:00 (Cold Patterns)
cold
Number of Occurence


Time: 16:00 (Cold Patterns)
cold
Number of Occurence


Time: 17:00 (Cold Patterns)
cold
Number of Occurence

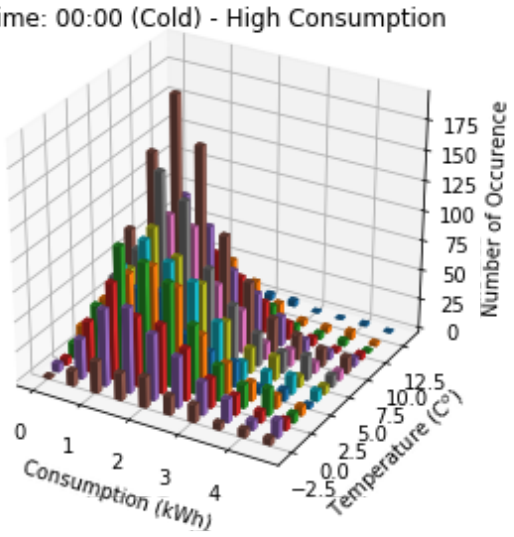


The distribution of days of a week (since it's non-event days, so it's worth checking this property)

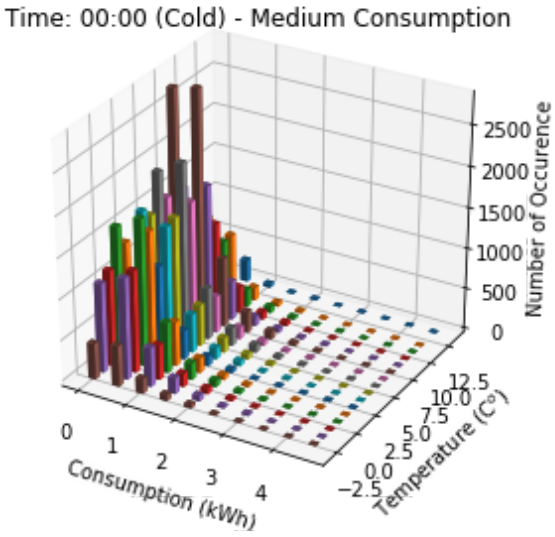
In [35]:
```
# temperature distribution of different hours of a day in a year
fig_all = plt.figure(figsize = (6,4))
ax = fig_all.add_subplot(1, 1, 1)
ax.hist([pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('00:00:00')]['GMT']).dt.dayofweek
+ 1,
         pd.to_datetime(df_wea1h_nf_warm[df_Ntou1h_nf_warm.GMT.str.contains('00:00:00')]['GMT']).dt.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 [36]:
```
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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:0
0')]['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' + ' (Cold) - ' + 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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['T
empC'].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' + ' (Cold) - ' + 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()
```

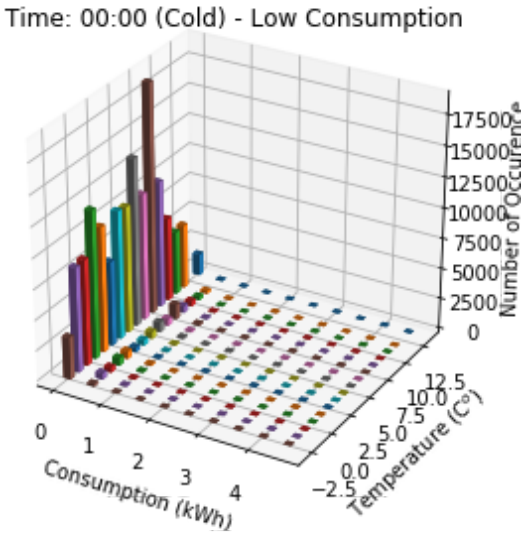
Time: 00:00 (Cold) - High Consumption



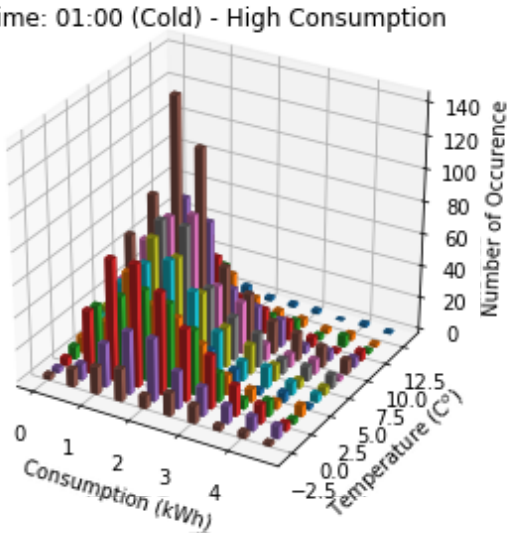
Time: 00:00 (Cold) - Medium Consumption



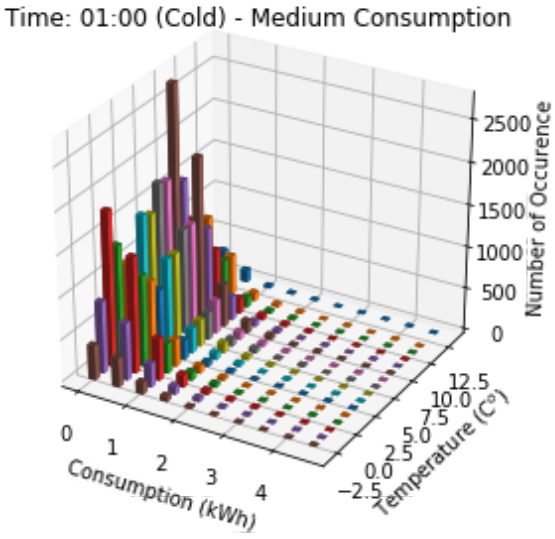
Time: 00:00 (Cold) - Low Consumption



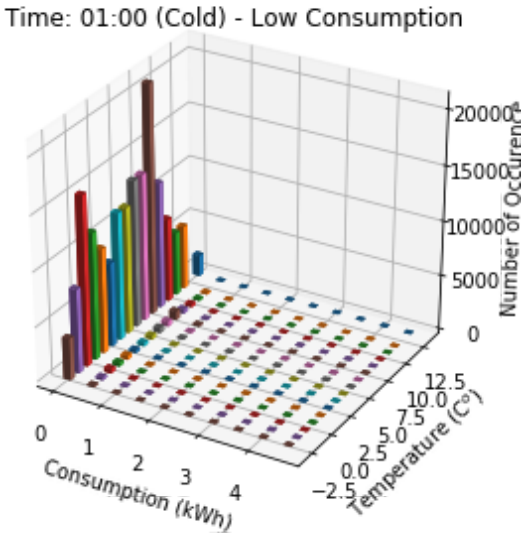
Time: 01:00 (Cold) - High Consumption



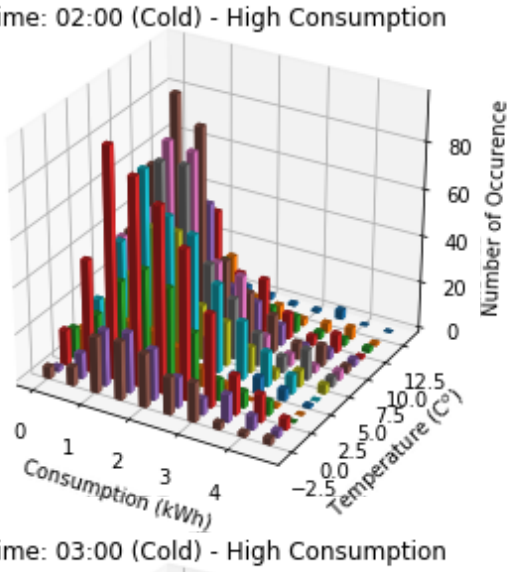
Time: 01:00 (Cold) - Medium Consumption



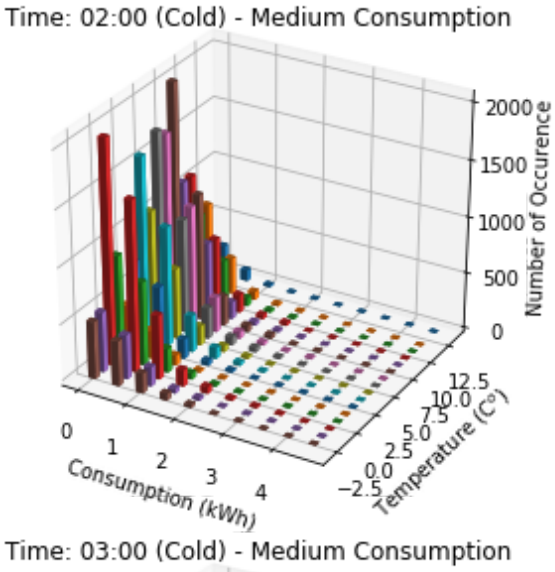
Time: 01:00 (Cold) - Low Consumption



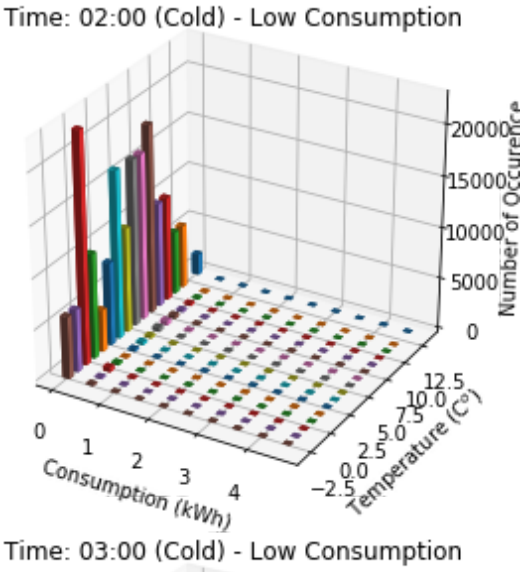
Time: 02:00 (Cold) - High Consumption



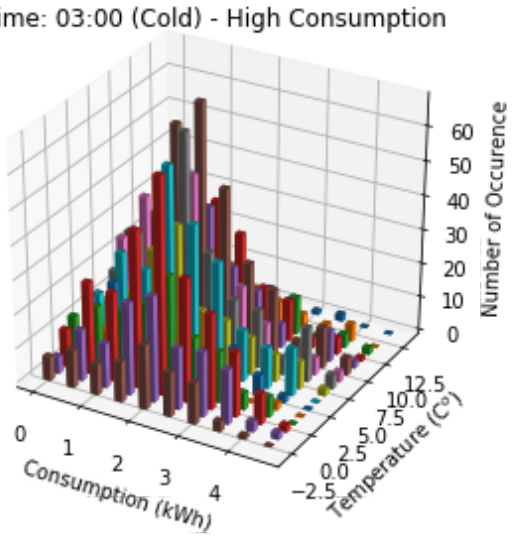
Time: 02:00 (Cold) - Medium Consumption



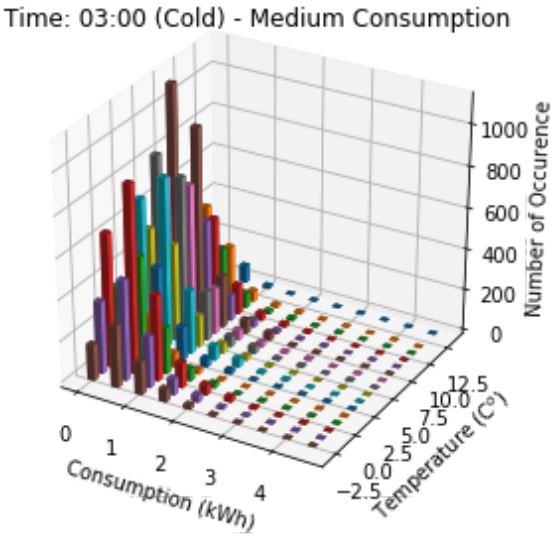
Time: 02:00 (Cold) - Low Consumption



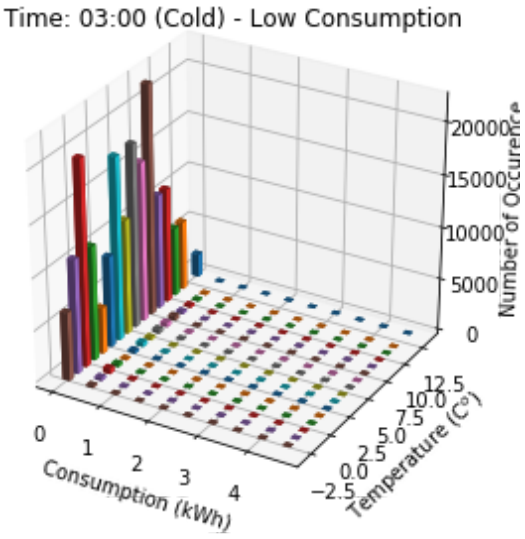
Time: 03:00 (Cold) - High Consumption



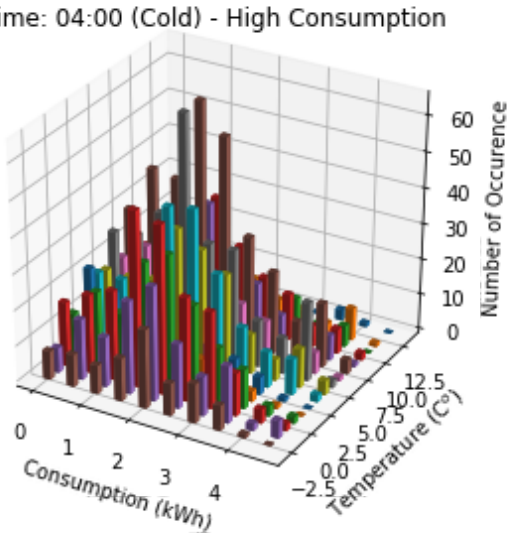
Time: 03:00 (Cold) - Medium Consumption



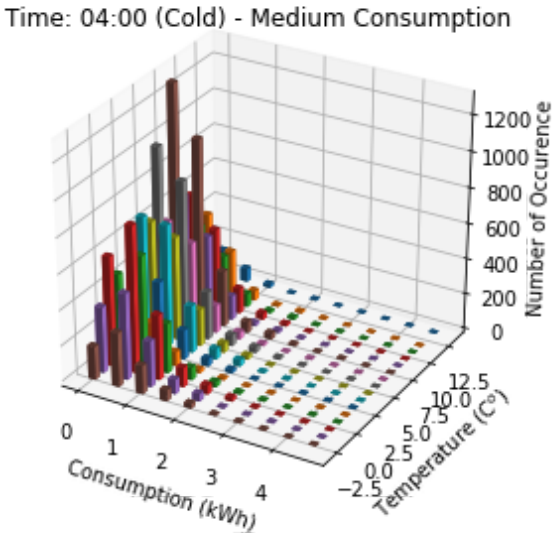
Time: 03:00 (Cold) - Low Consumption



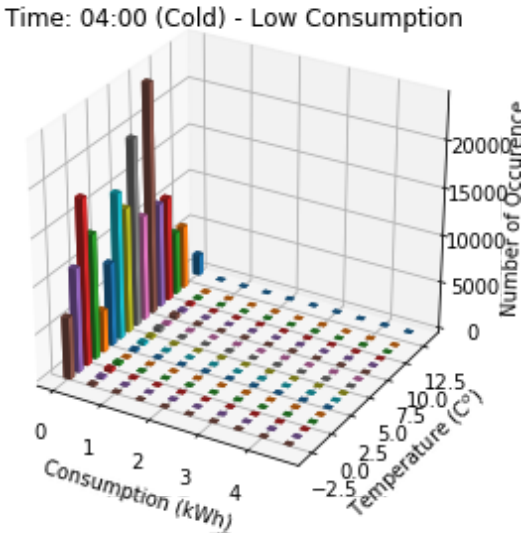
Time: 04:00 (Cold) - High Consumption



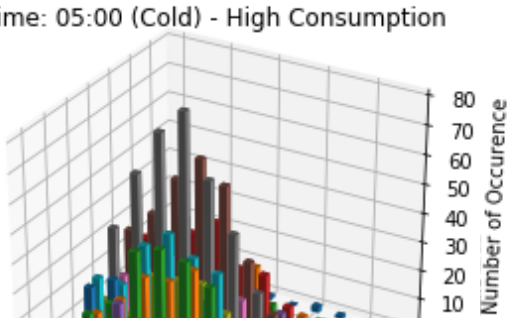
Time: 04:00 (Cold) - Medium Consumption



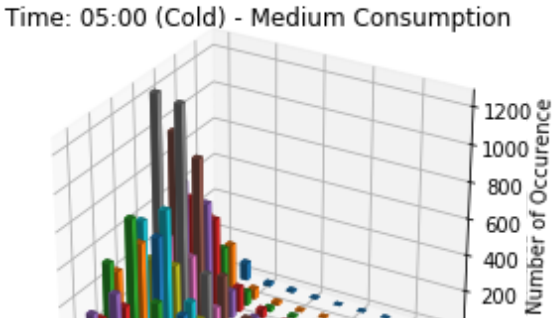
Time: 04:00 (Cold) - Low Consumption



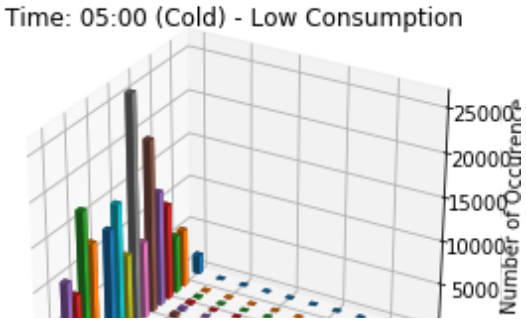
Time: 05:00 (Cold) - High Consumption

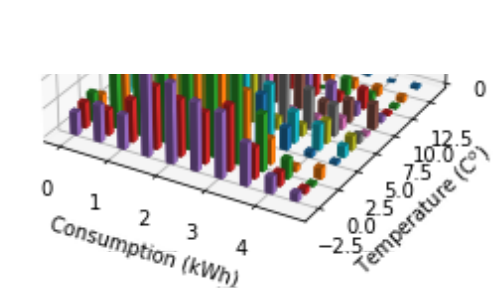


Time: 05:00 (Cold) - Medium Consumption

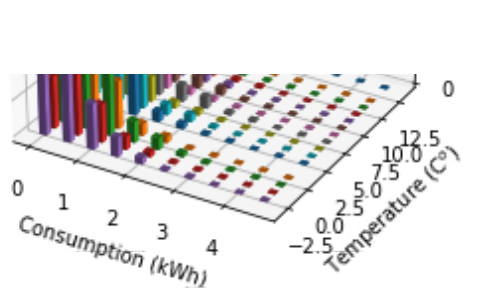


Time: 05:00 (Cold) - Low Consumption

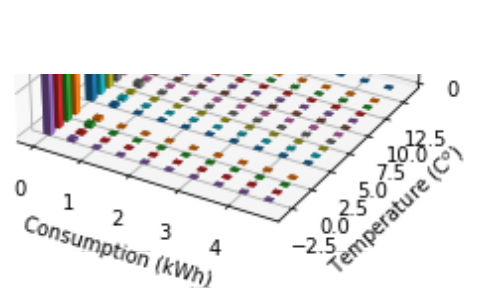




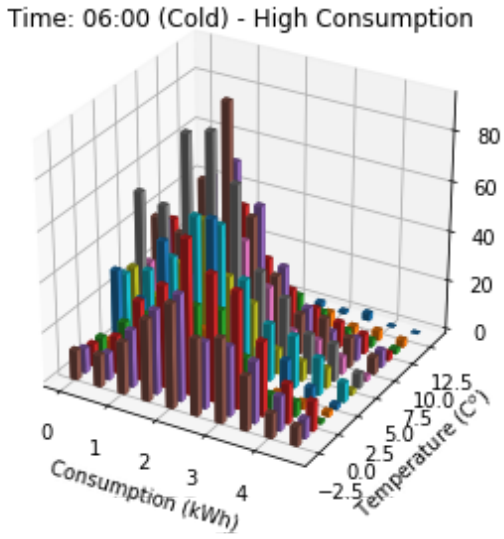
Time: 06:00 (Cold) - High Consumption



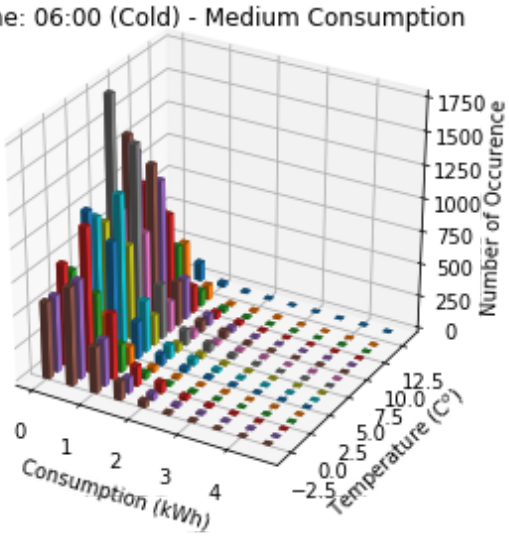
Time: 06:00 (Cold) - Medium Consumption



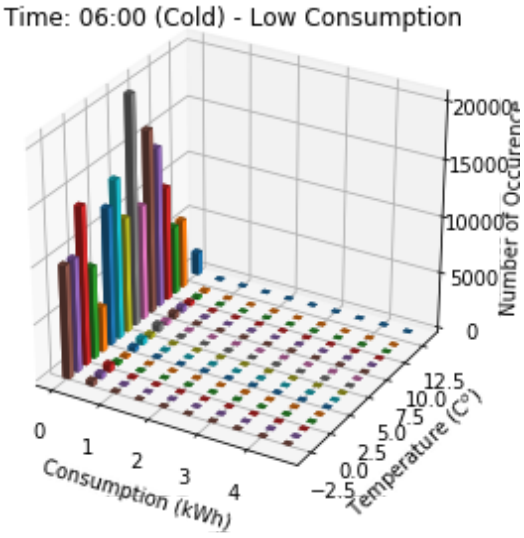
Time: 06:00 (Cold) - Low Consumption



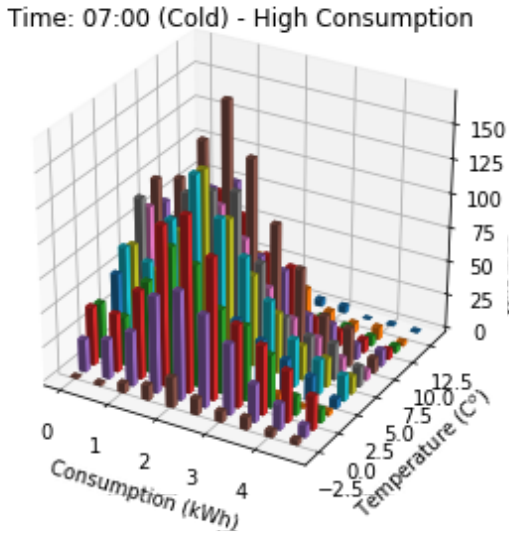
Time: 07:00 (Cold) - High Consumption



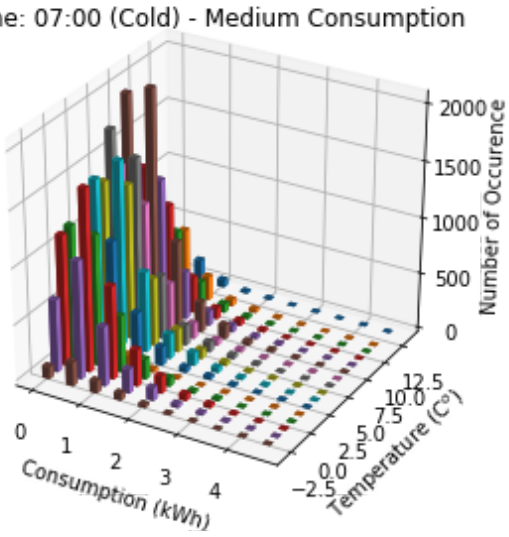
Time: 07:00 (Cold) - Medium Consumption



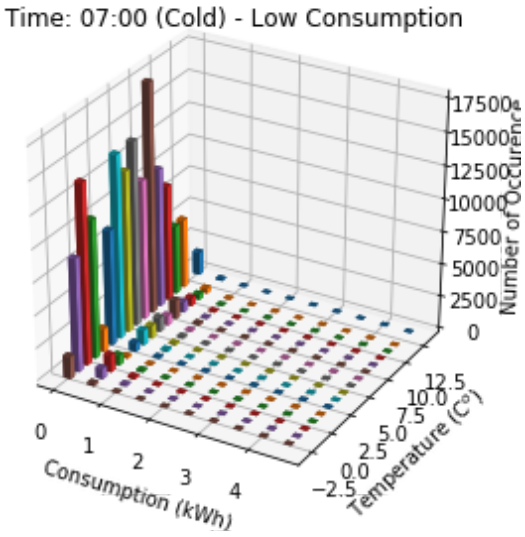
Time: 07:00 (Cold) - Low Consumption



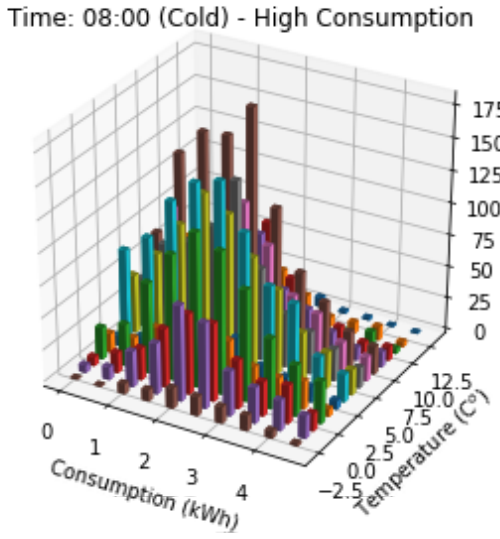
Time: 08:00 (Cold) - High Consumption



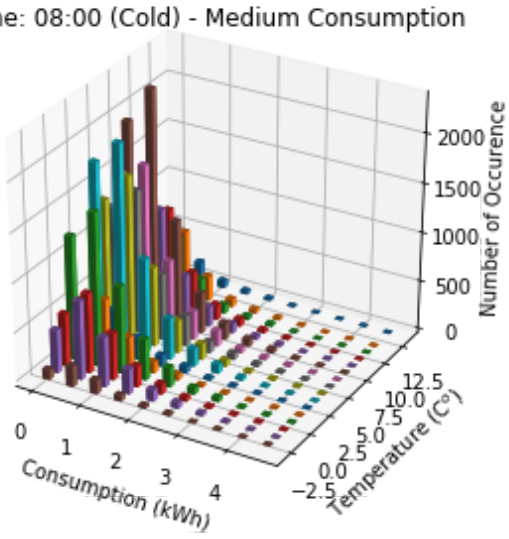
Time: 08:00 (Cold) - Medium Consumption



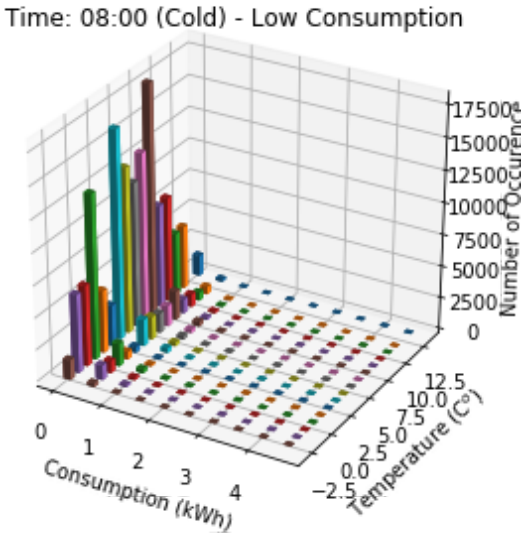
Time: 08:00 (Cold) - Low Consumption



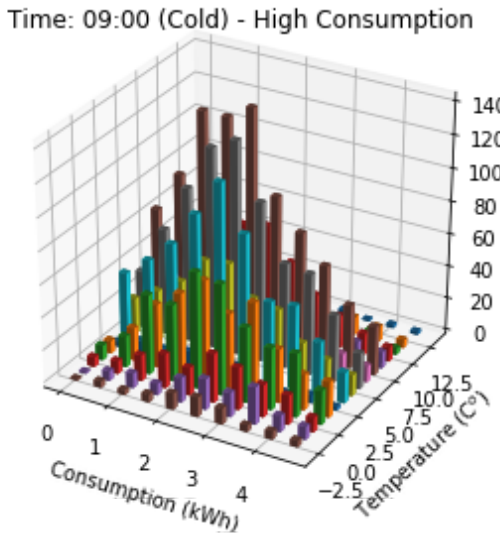
Time: 09:00 (Cold) - High Consumption



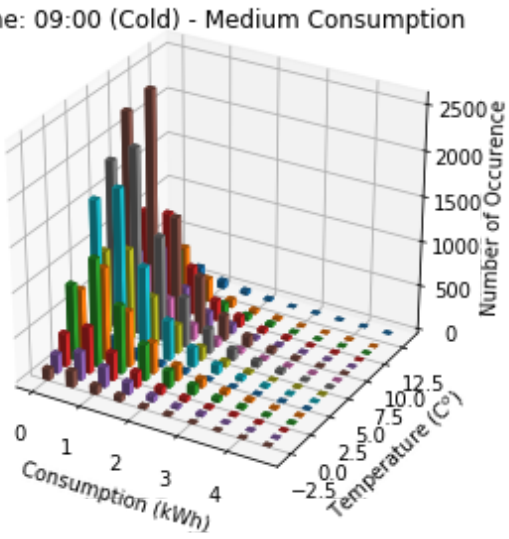
Time: 09:00 (Cold) - Medium Consumption



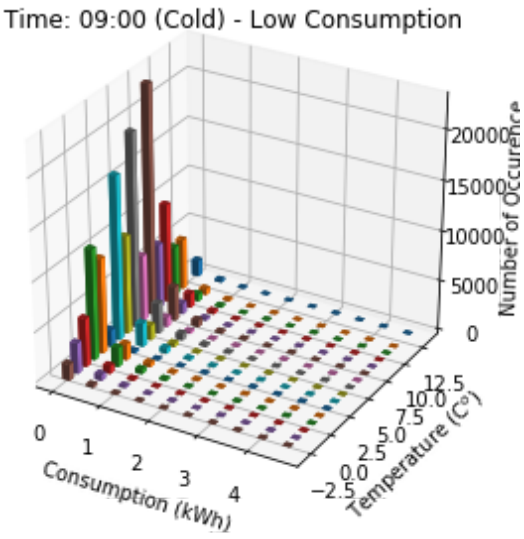
Time: 09:00 (Cold) - Low Consumption



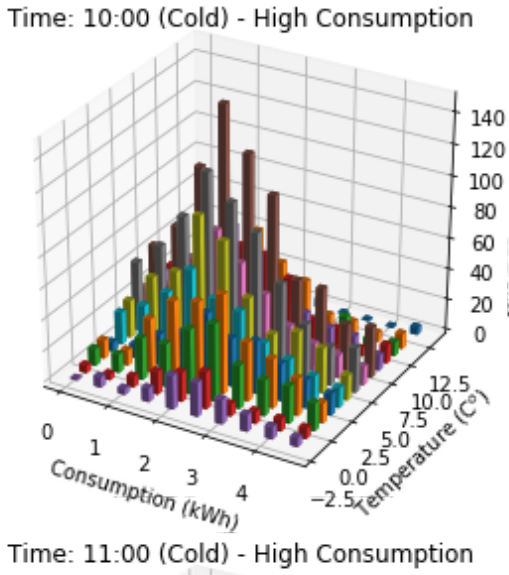
Time: 10:00 (Cold) - High Consumption



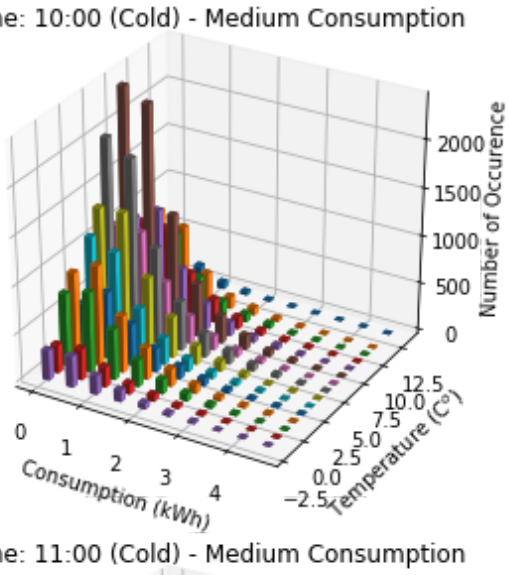
Time: 10:00 (Cold) - Medium Consumption



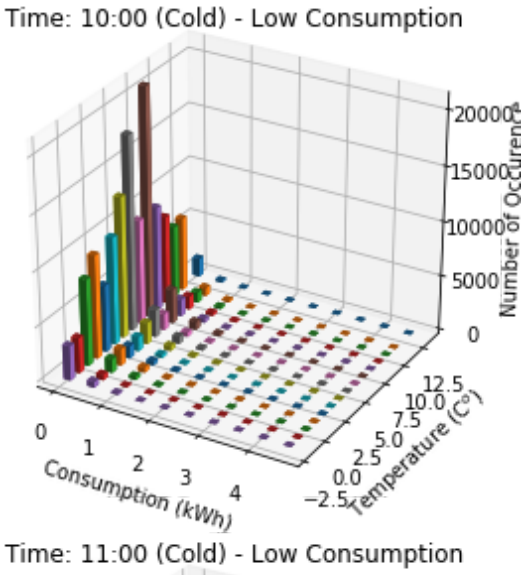
Time: 10:00 (Cold) - Low Consumption



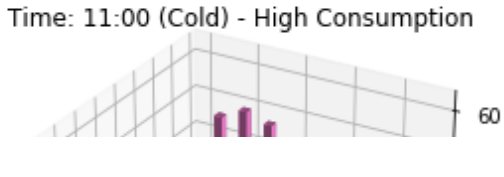
Time: 11:00 (Cold) - High Consumption

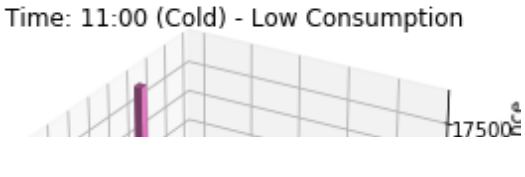


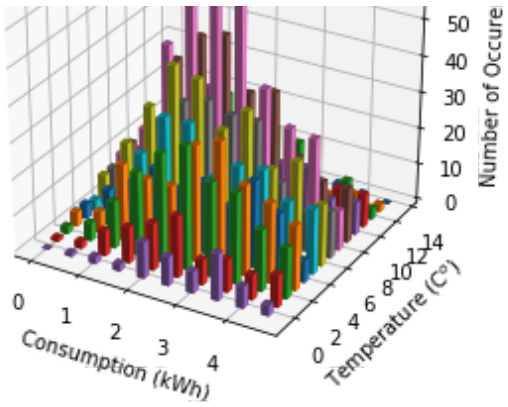
Time: 11:00 (Cold) - Medium Consumption



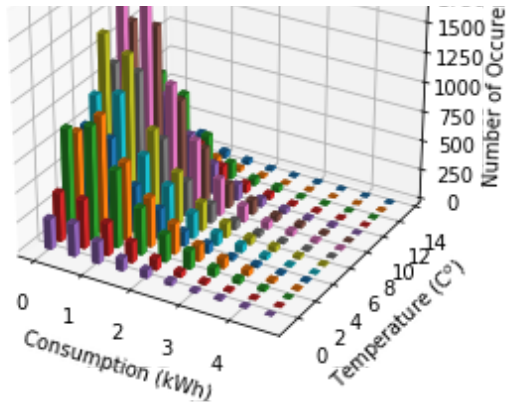
Time: 11:00 (Cold) - Low Consumption



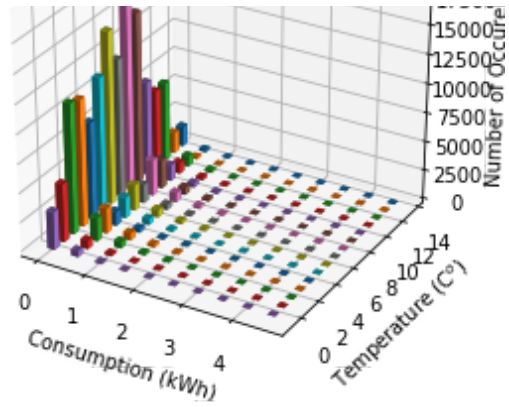




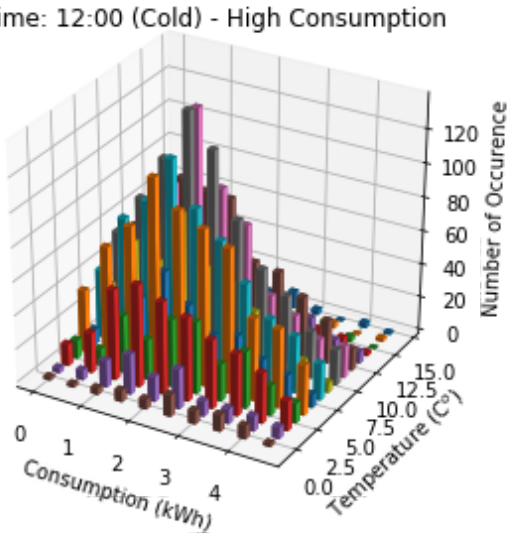
Time: 12:00 (Cold) - High Consumption



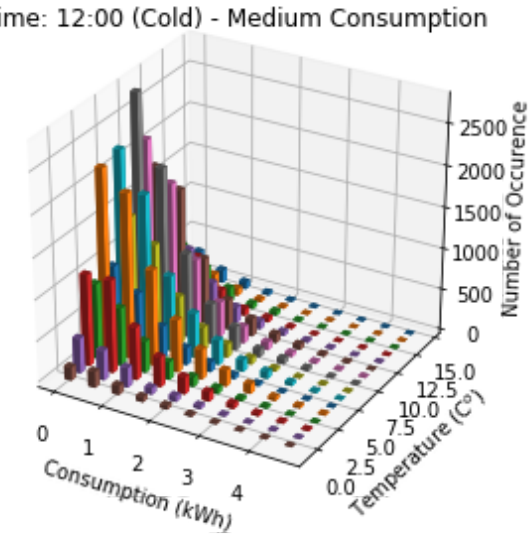
Time: 12:00 (Cold) - Medium Consumption



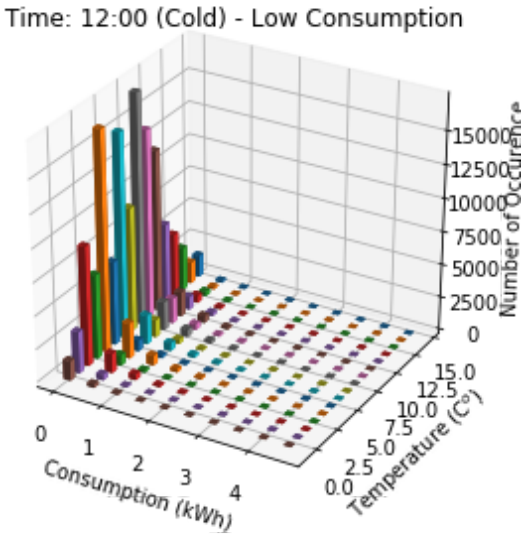
Time: 12:00 (Cold) - Low Consumption



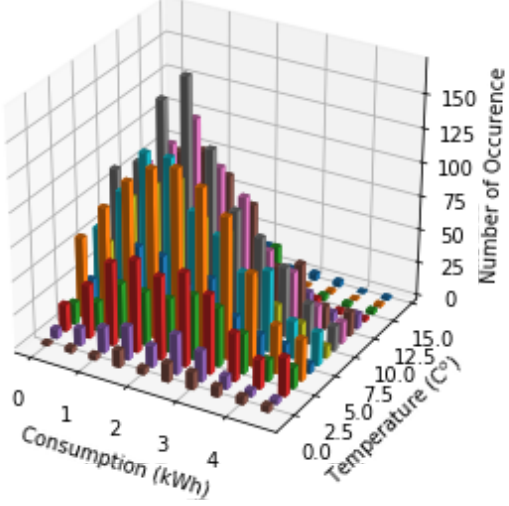
Time: 13:00 (Cold) - High Consumption



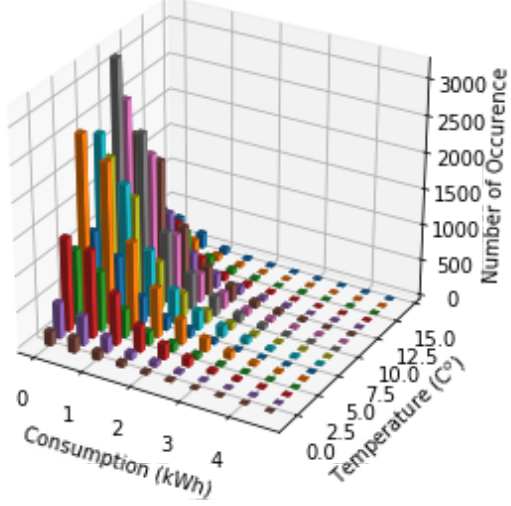
Time: 13:00 (Cold) - Medium Consumption



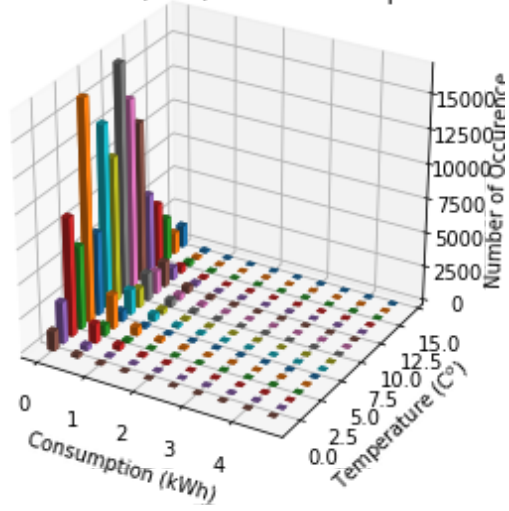
Time: 13:00 (Cold) - Low Consumption



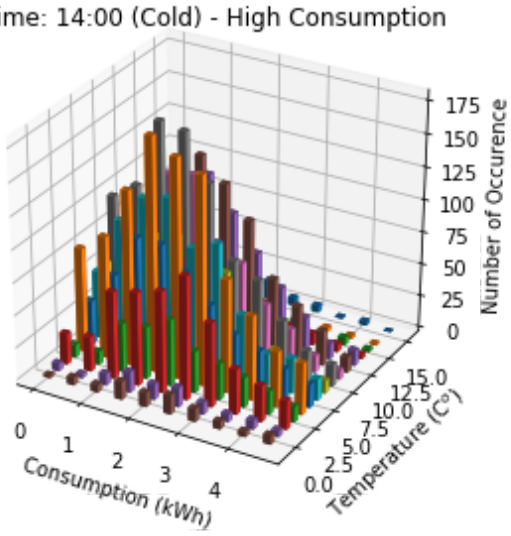
Time: 14:00 (Cold) - High Consumption



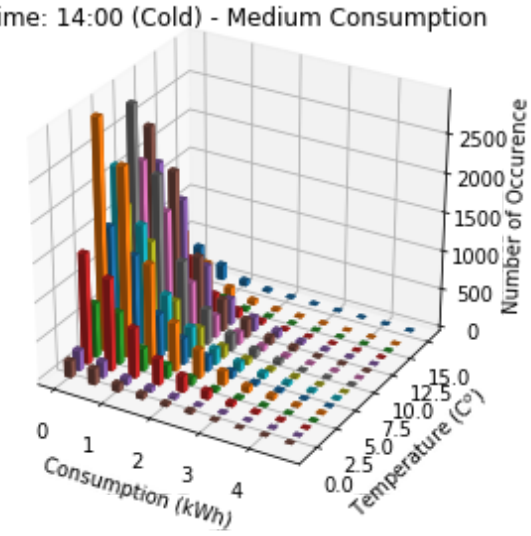
Time: 14:00 (Cold) - Medium Consumption



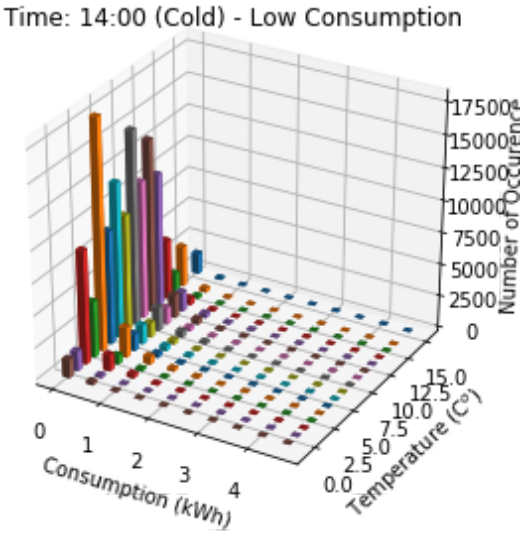
Time: 14:00 (Cold) - Low Consumption



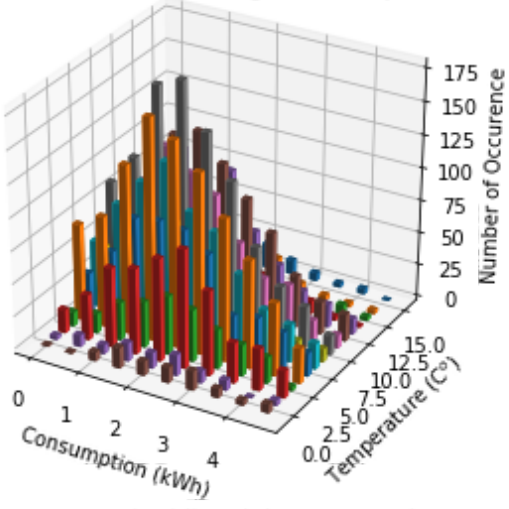
Time: 15:00 (Cold) - High Consumption



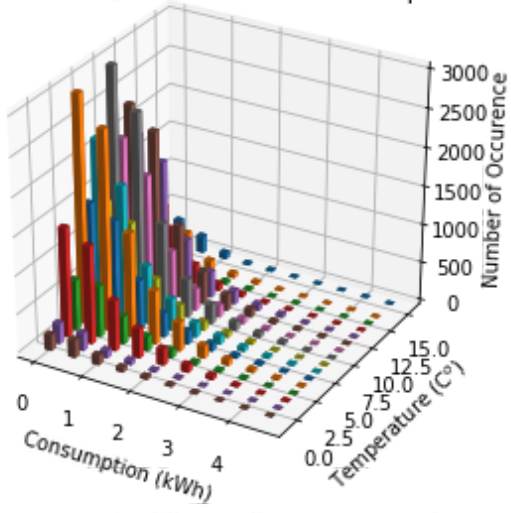
Time: 15:00 (Cold) - Medium Consumption



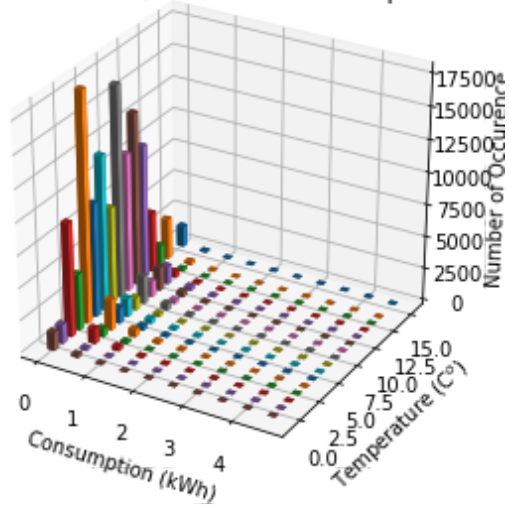
Time: 15:00 (Cold) - Low Consumption



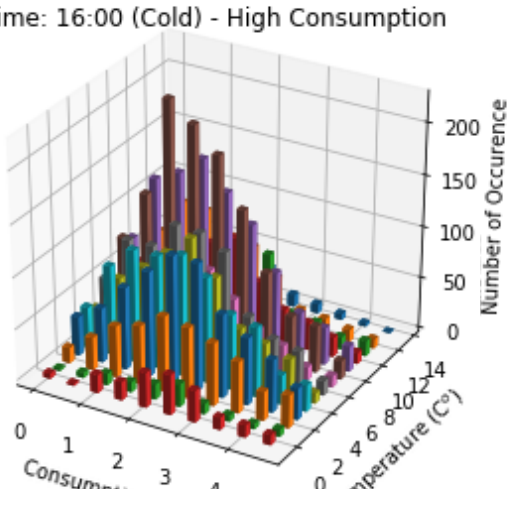
Time: 16:00 (Cold) - High Consumption

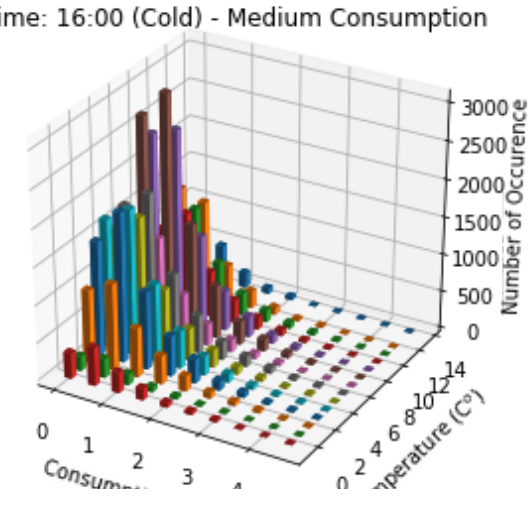


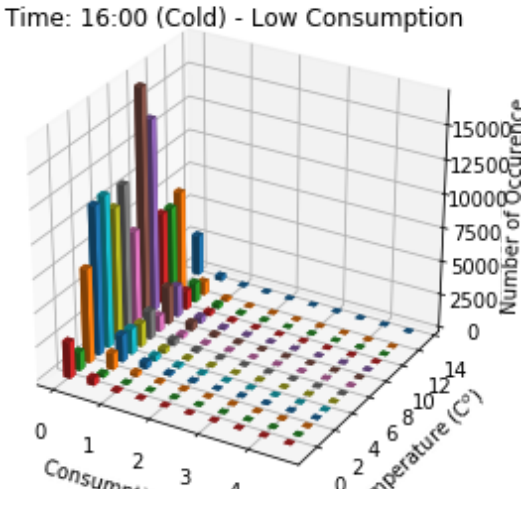
Time: 16:00 (Cold) - Medium Consumption

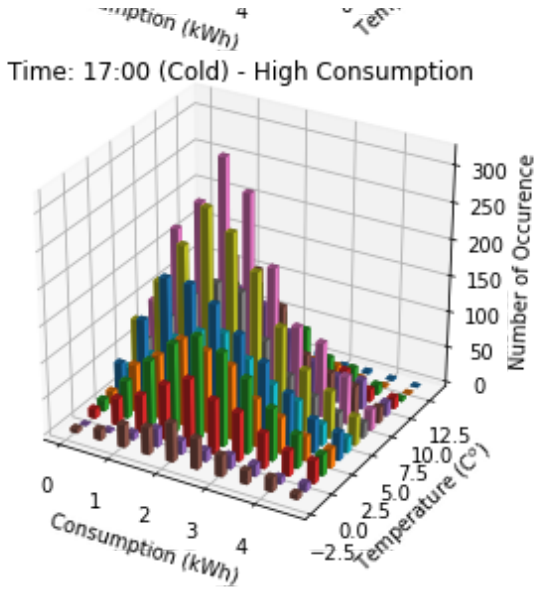


Time: 16:00 (Cold) - Low Consumption

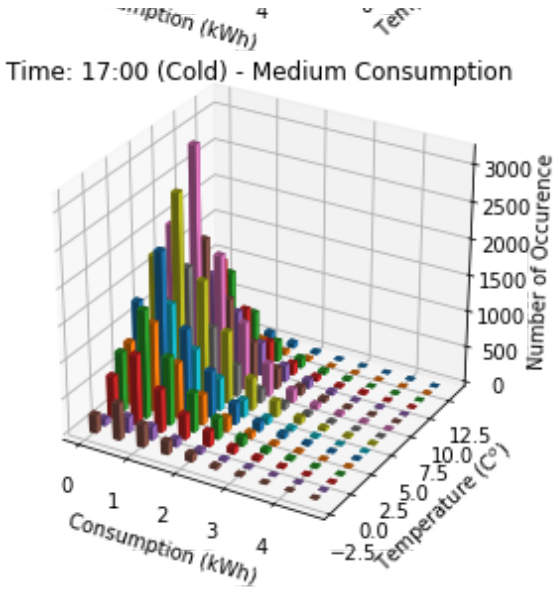




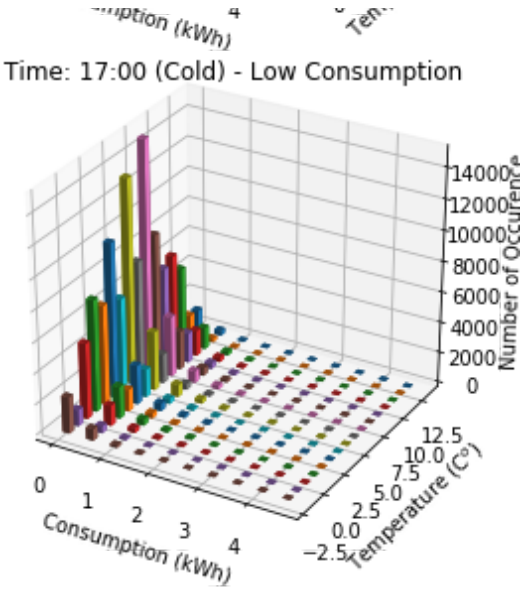




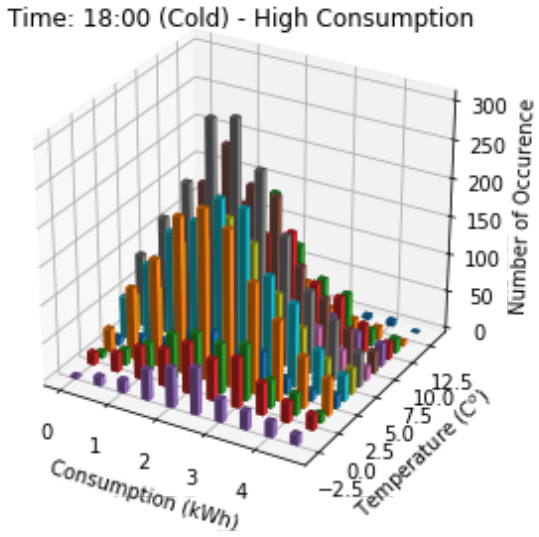
Time: 17:00 (Cold) - High Consumption



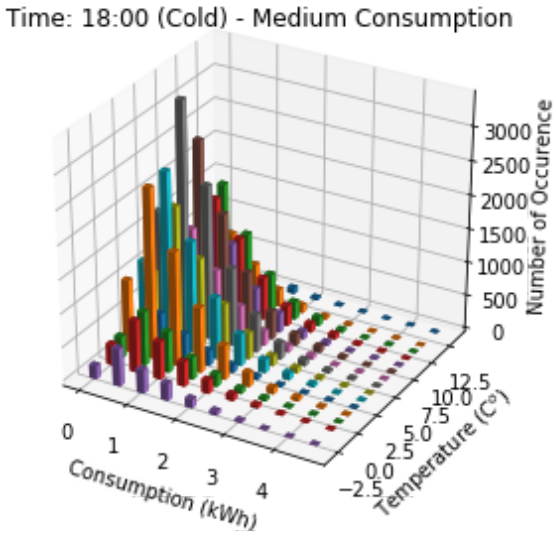
Time: 17:00 (Cold) - Medium Consumption



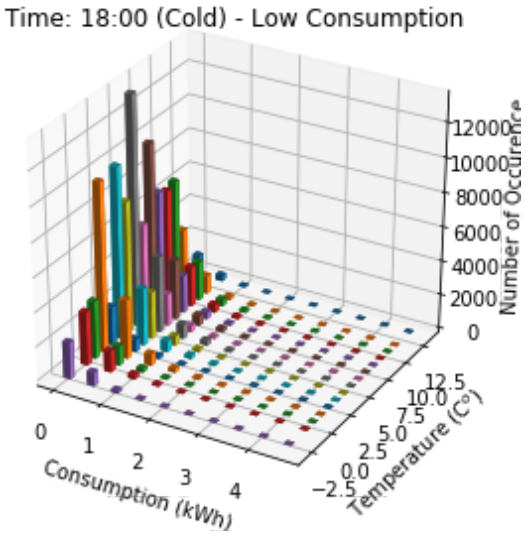
Time: 17:00 (Cold) - Low Consumption



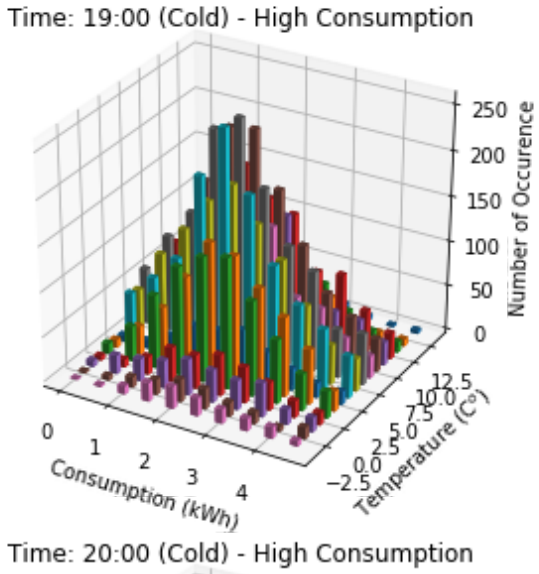
Time: 18:00 (Cold) - High Consumption



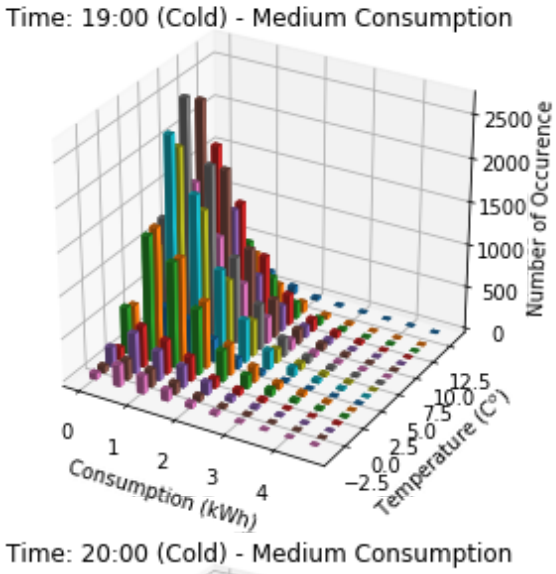
Time: 18:00 (Cold) - Medium Consumption



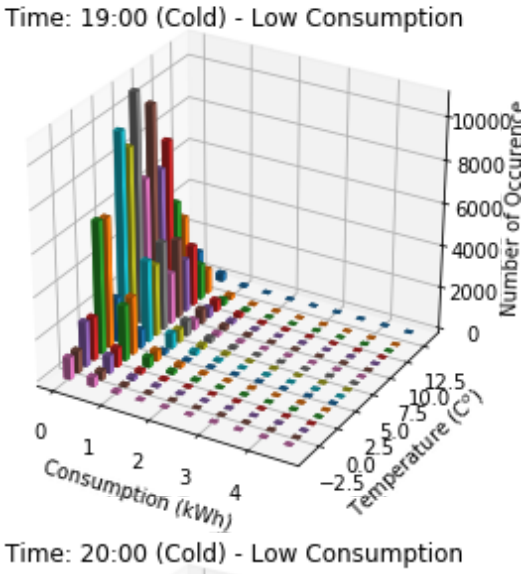
Time: 18:00 (Cold) - Low Consumption



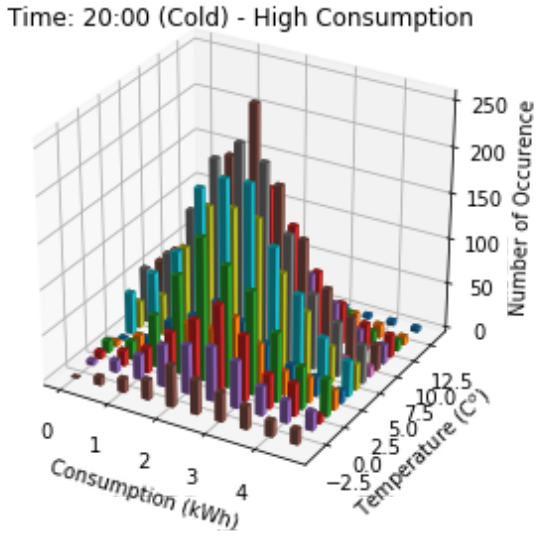
Time: 19:00 (Cold) - High Consumption



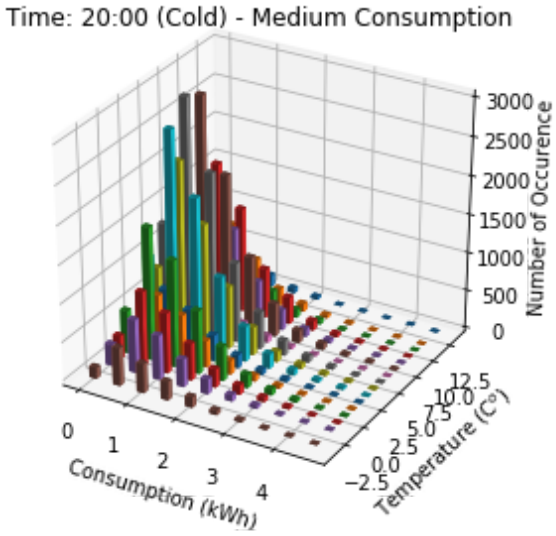
Time: 19:00 (Cold) - Medium Consumption



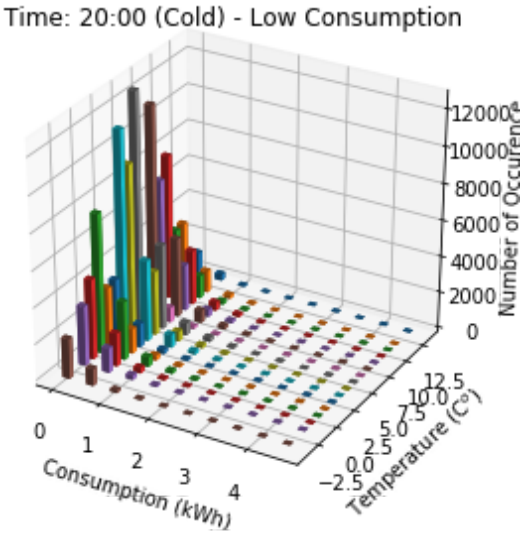
Time: 19:00 (Cold) - Low Consumption



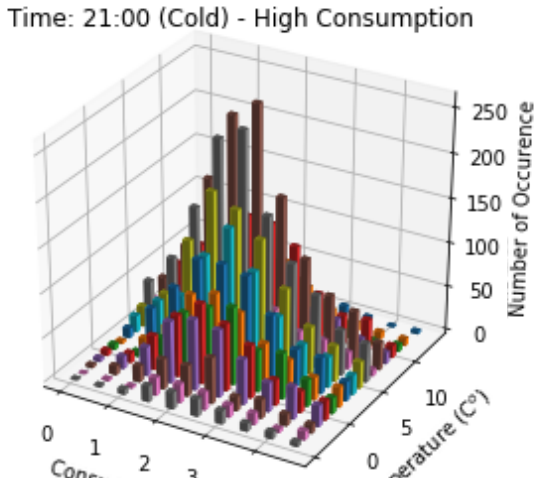
Time: 20:00 (Cold) - High Consumption



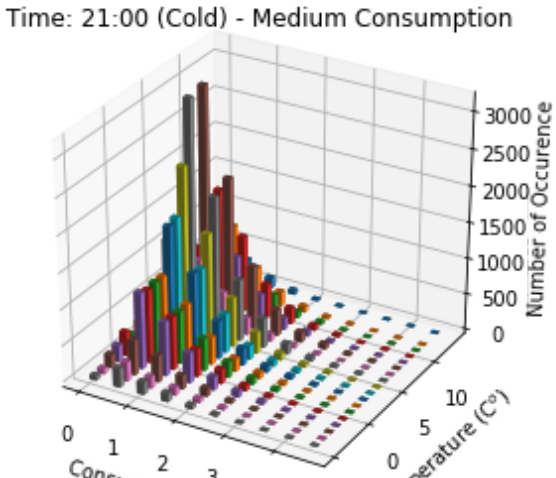
Time: 20:00 (Cold) - Medium Consumption



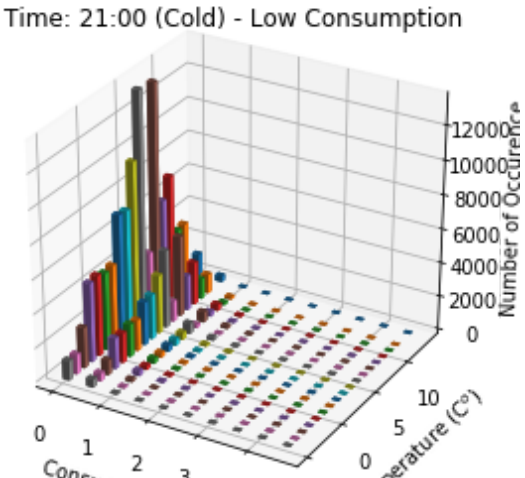
Time: 20:00 (Cold) - Low Consumption



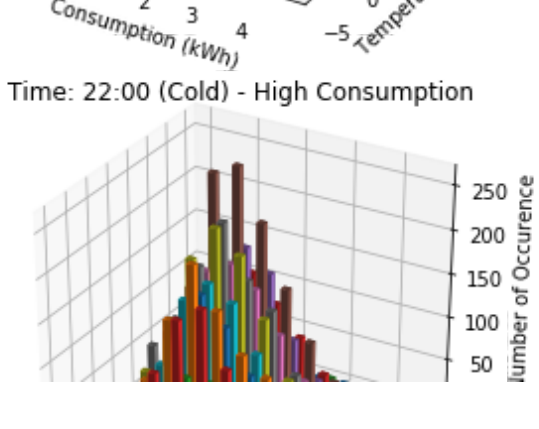
Time: 21:00 (Cold) - High Consumption



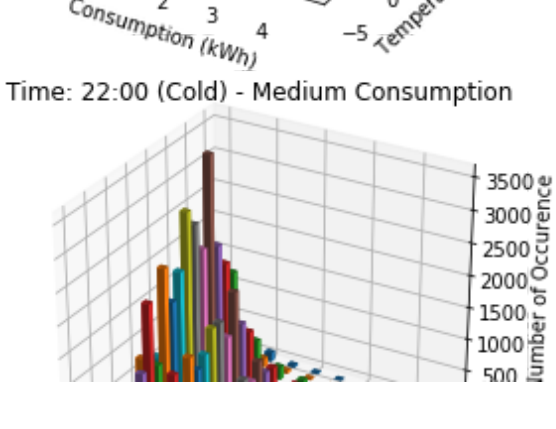
Time: 21:00 (Cold) - Medium Consumption



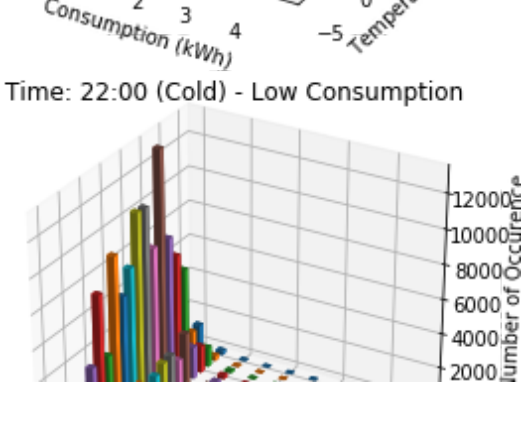
Time: 21:00 (Cold) - Low Consumption



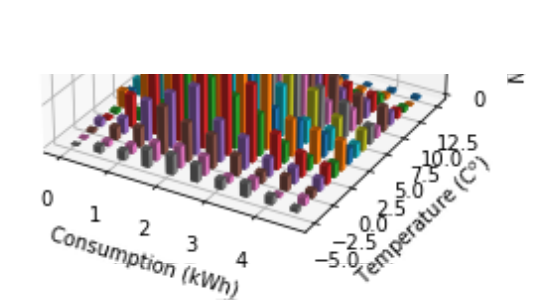
Time: 22:00 (Cold) - High Consumption



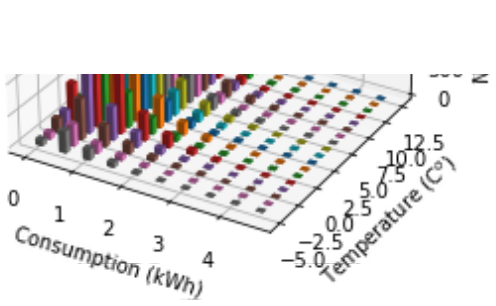
Time: 22:00 (Cold) - Medium Consumption



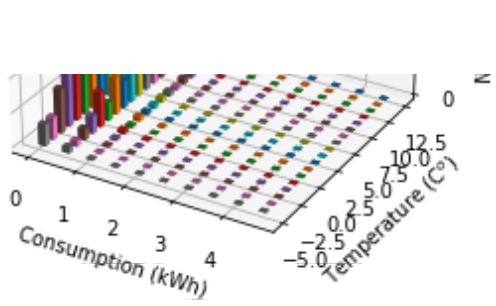
Time: 22:00 (Cold) - Low Consumption



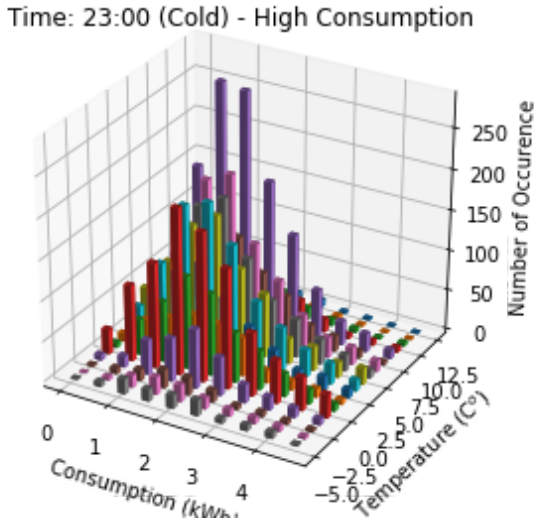
Time: 23:00 (Cold) - High Consumption

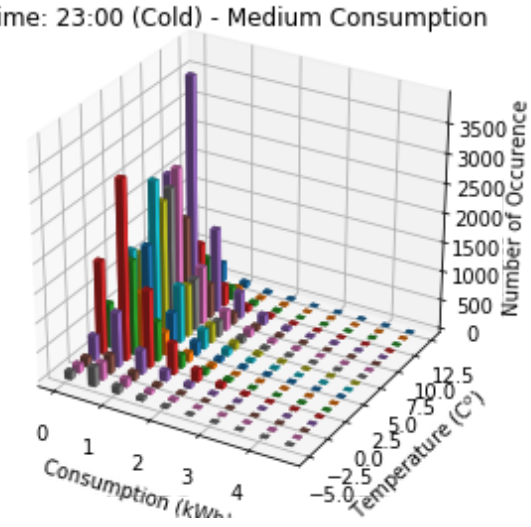


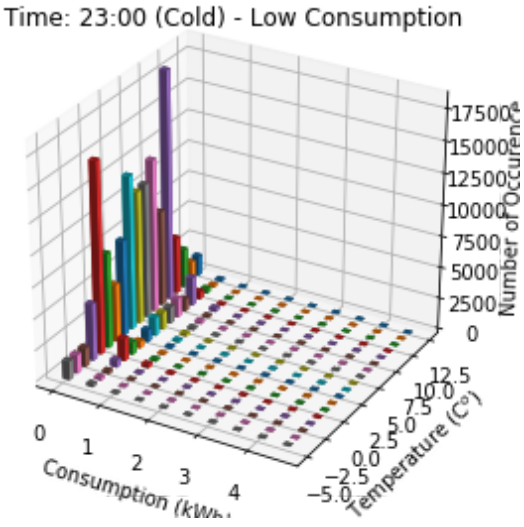
Time: 23:00 (Cold) - Medium Consumption



Time: 23:00 (Cold) - Low Consumption







Most of the above plots show that the highest demand of each group happens around 8-10 celcius degree, except for 2 pm and 3 pm, which have the highest load around 5 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_cold[df_Ntou1h_nf_cold.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_cold[df_Ntou1h_nf_cold.GMT.str.contains('23:00:00')]['G
MT']).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

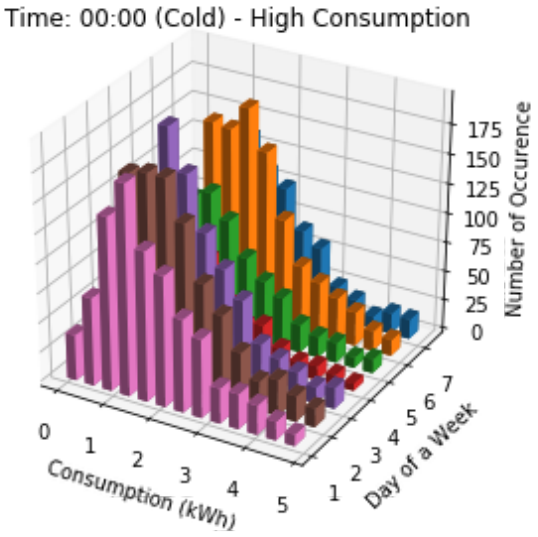
	GMT	D0000	D0001	D0003	D0004	D0005	D0007	D0008	D0009	D0010	...	D1015	D1016	D1017	D1018	D1020	D1021	D102
21	2013-02-13 23:00:00	0.119	0.196	0.154	0.115	0.354	0.292	0.238	0.284	0.025	...	0.030	0.728	0.479	0.117	0.183	0.214	0.355
22	2013-02-14 23:00:00	0.586	0.248	0.154	0.061	0.147	0.263	0.395	0.375	0.026	...	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

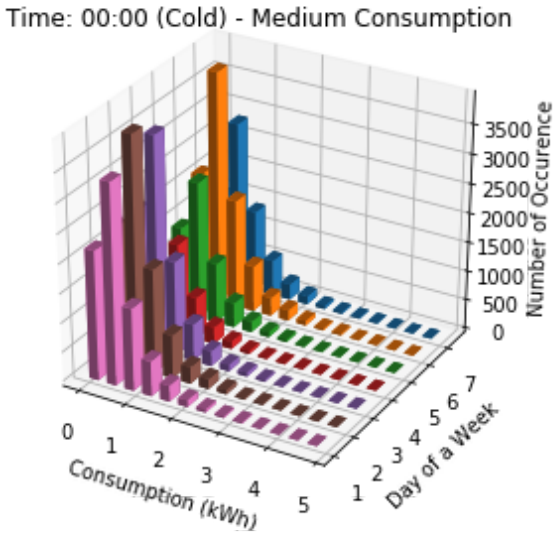
83 rows × 720 columns

In [37]:
```
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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.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' + ' (Cold) - ' + 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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.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' + ' (Cold) - ' + 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()
```

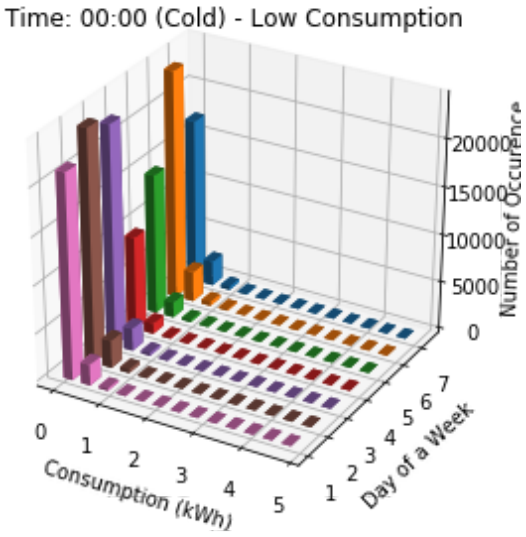
Time: 00:00 (Cold) - High Consumption



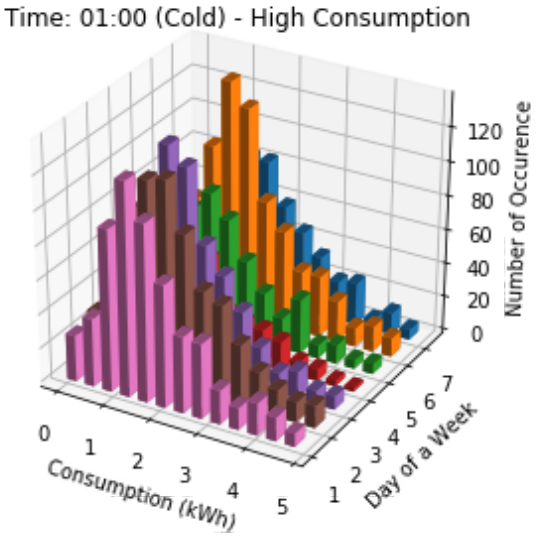
Time: 00:00 (Cold) - Medium Consumption



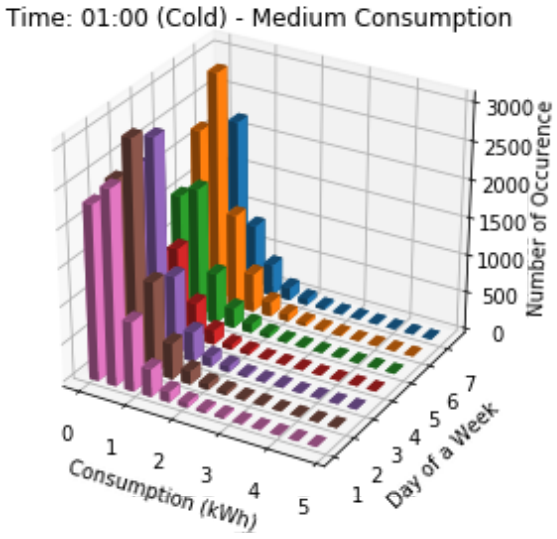
Time: 00:00 (Cold) - Low Consumption



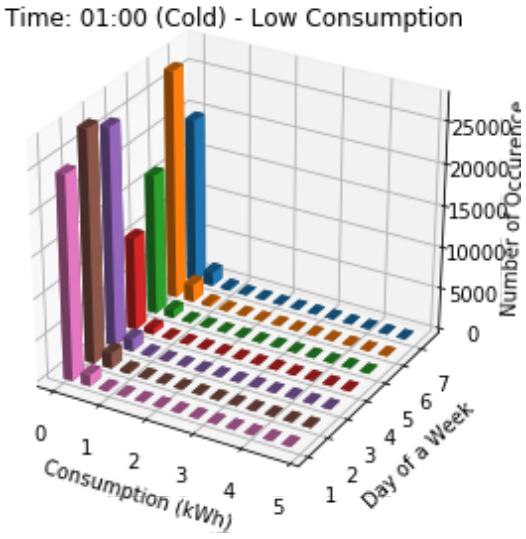
Time: 01:00 (Cold) - High Consumption



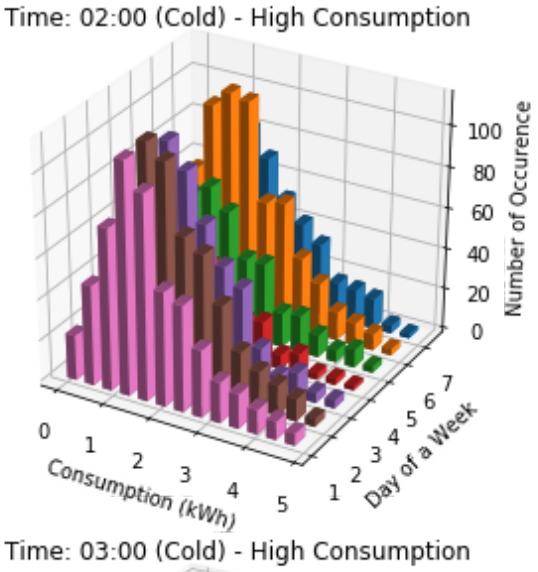
Time: 01:00 (Cold) - Medium Consumption



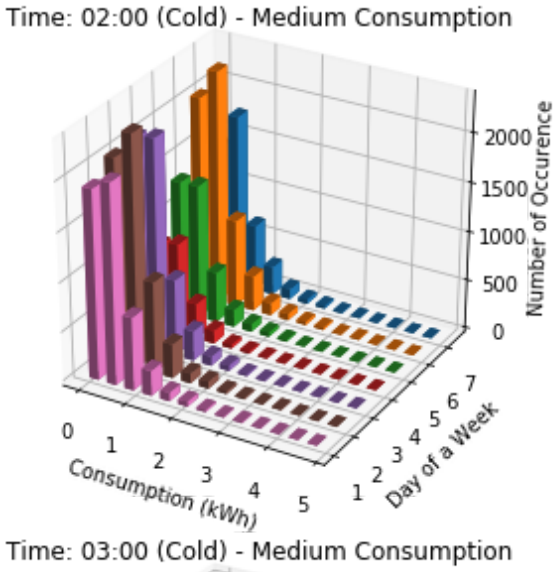
Time: 01:00 (Cold) - Low Consumption



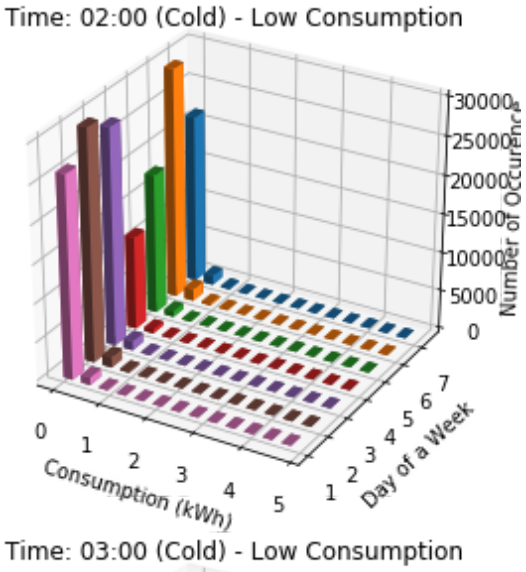
Time: 02:00 (Cold) - High Consumption



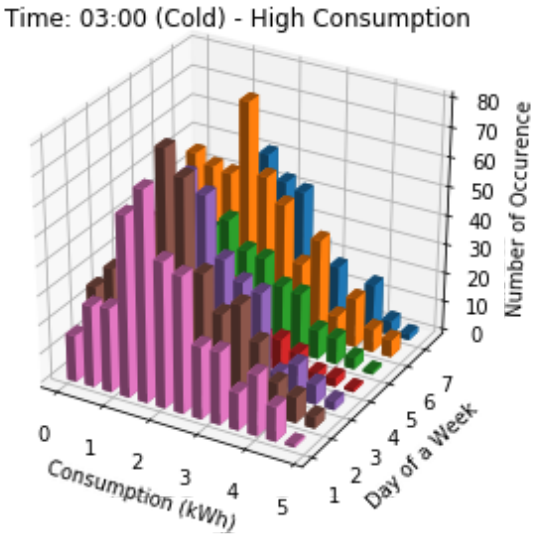
Time: 02:00 (Cold) - Medium Consumption



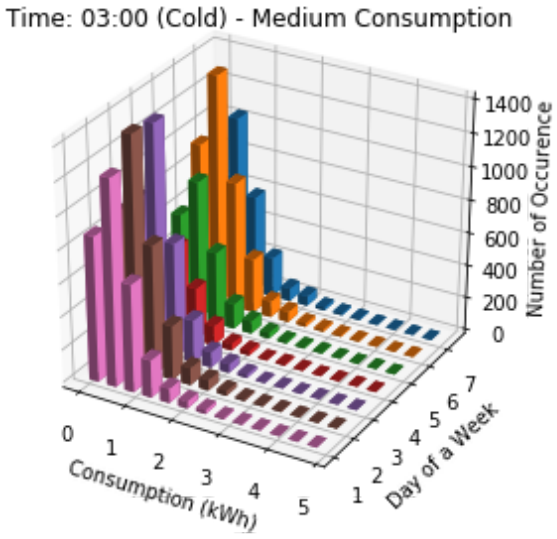
Time: 02:00 (Cold) - Low Consumption



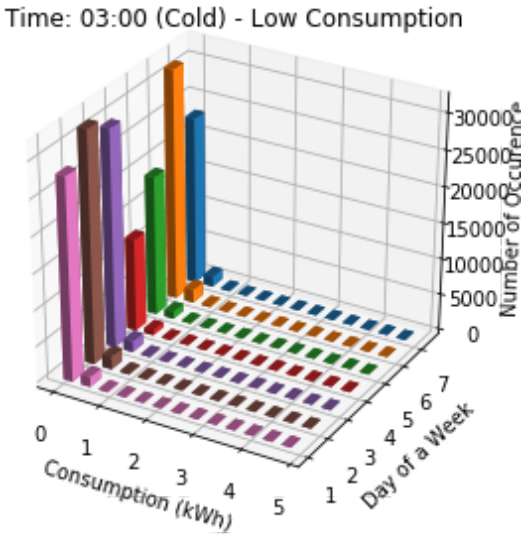
Time: 03:00 (Cold) - High Consumption



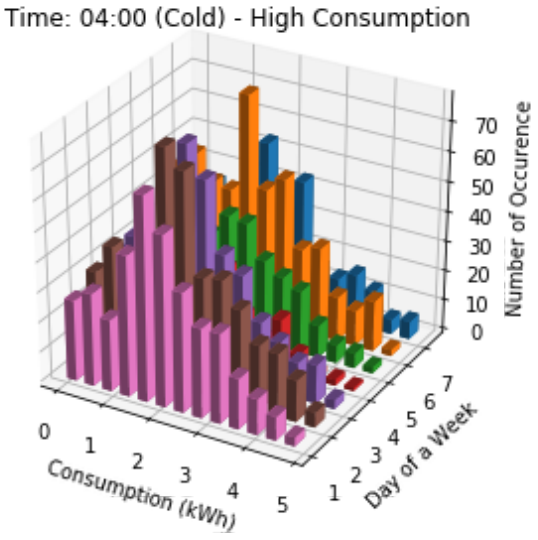
Time: 03:00 (Cold) - Medium Consumption



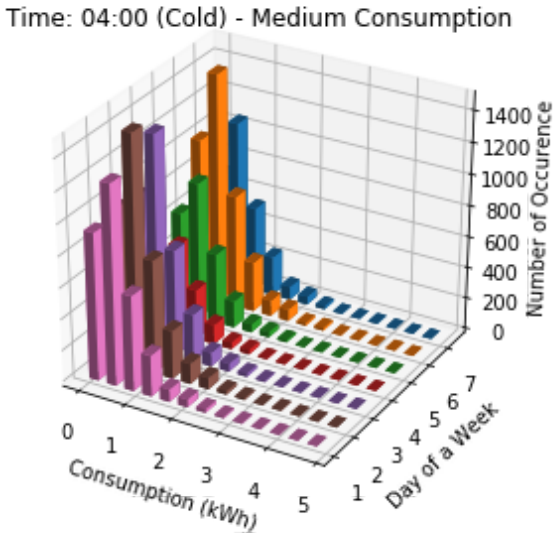
Time: 03:00 (Cold) - Low Consumption



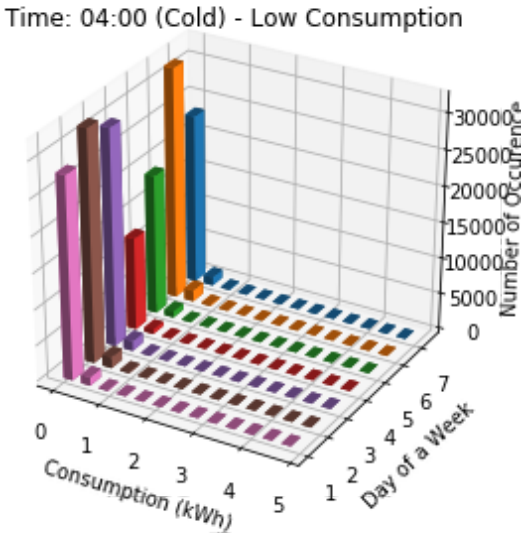
Time: 04:00 (Cold) - High Consumption



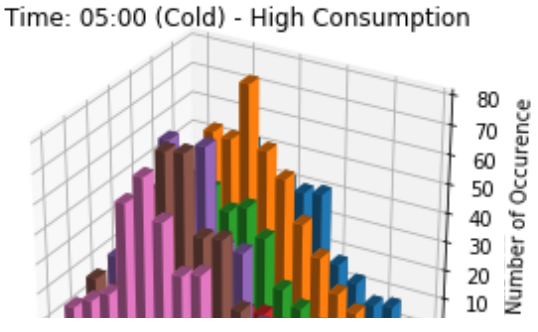
Time: 04:00 (Cold) - Medium Consumption



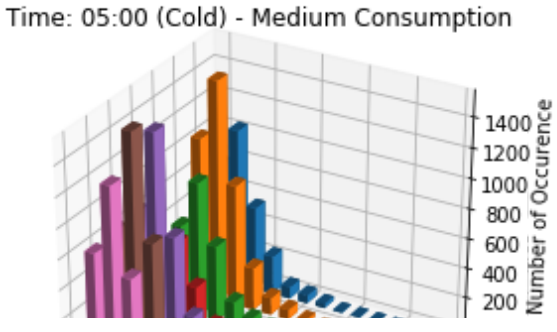
Time: 04:00 (Cold) - Low Consumption



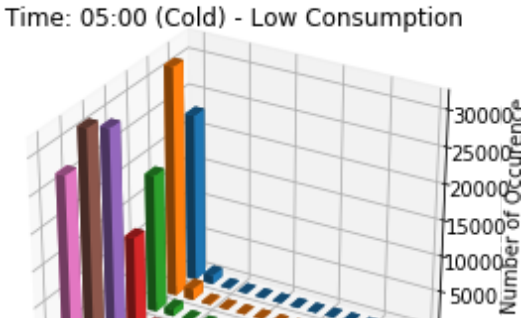
Time: 05:00 (Cold) - High Consumption

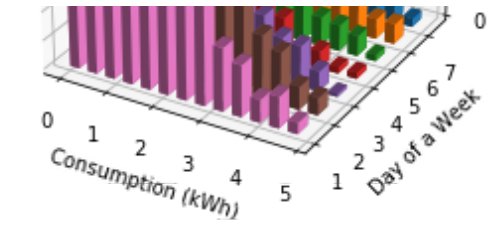


Time: 05:00 (Cold) - Medium Consumption

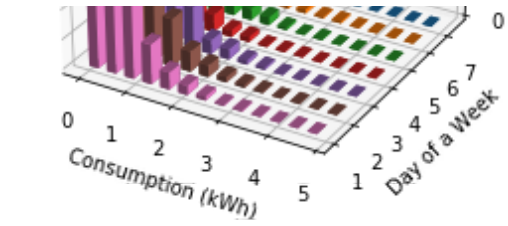


Time: 05:00 (Cold) - Low Consumption

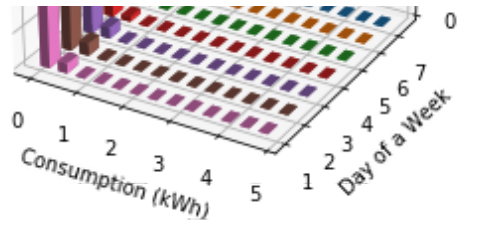




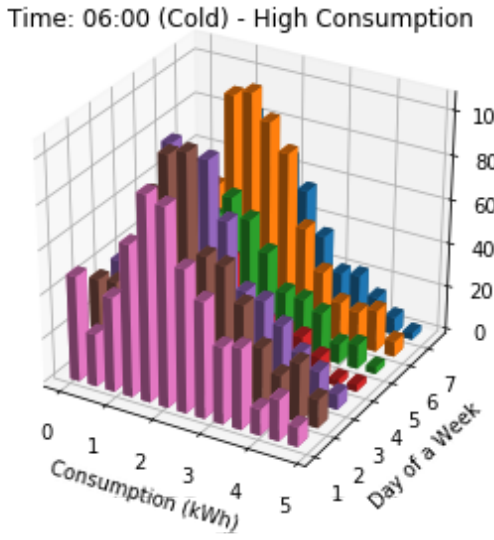
Time: 06:00 (Cold) - High Consumption



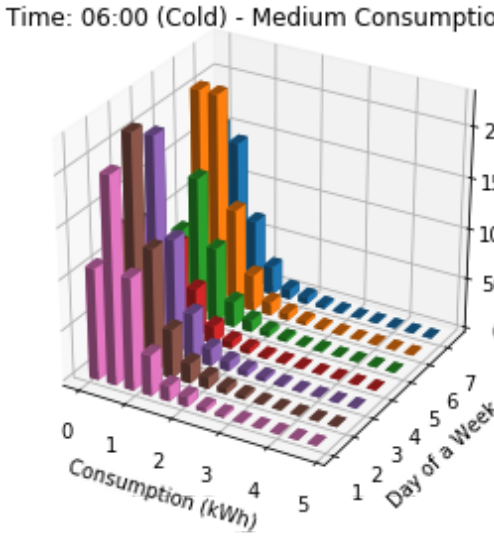
Time: 06:00 (Cold) - Medium Consumption



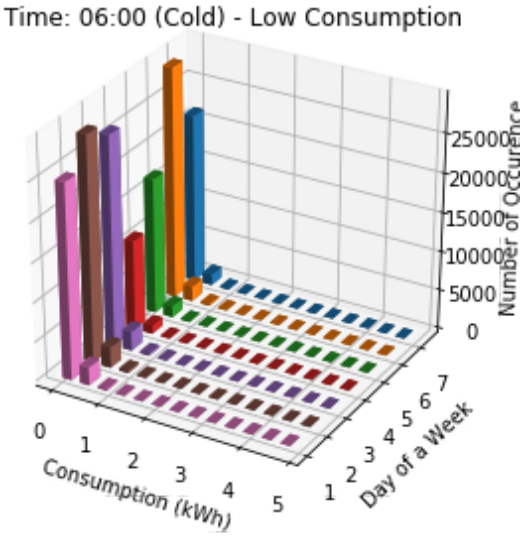
Time: 06:00 (Cold) - Low Consumption



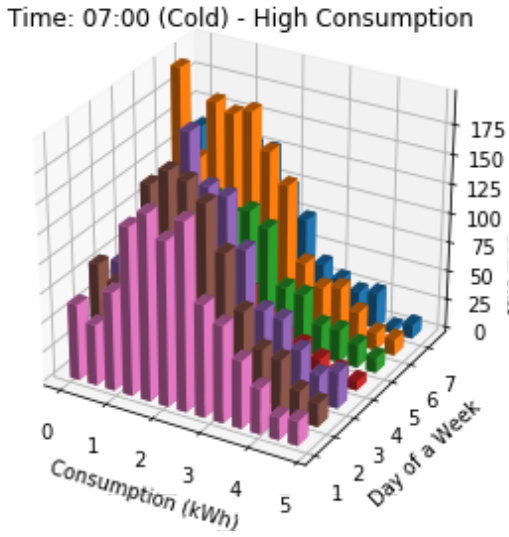
Time: 07:00 (Cold) - High Consumption



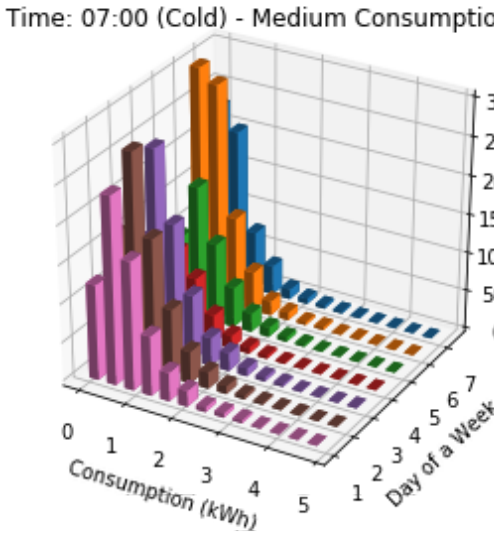
Time: 07:00 (Cold) - Medium Consumption



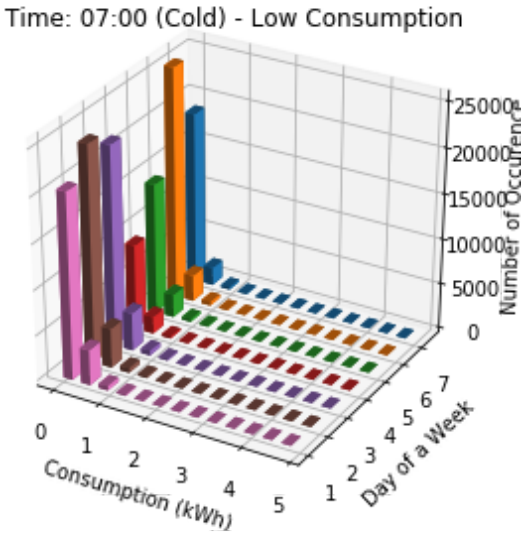
Time: 07:00 (Cold) - Low Consumption



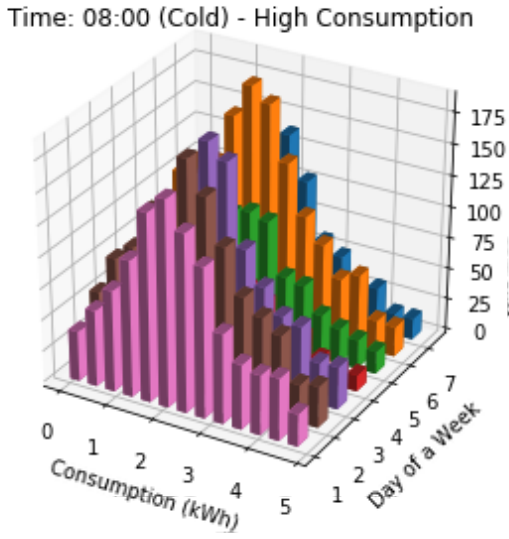
Time: 08:00 (Cold) - High Consumption



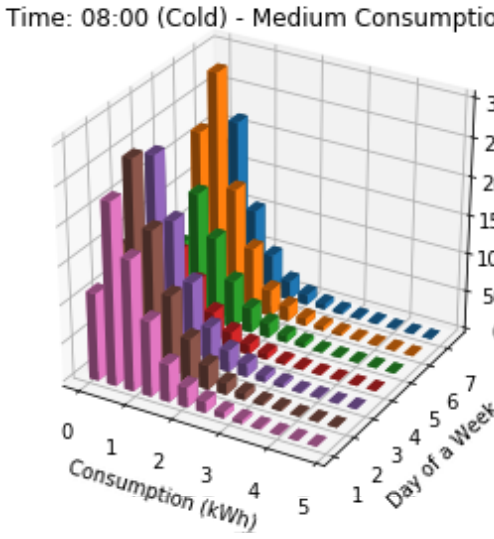
Time: 08:00 (Cold) - Medium Consumption



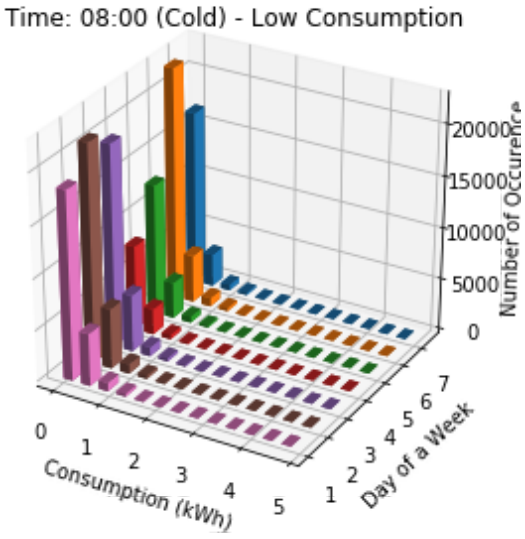
Time: 08:00 (Cold) - Low Consumption



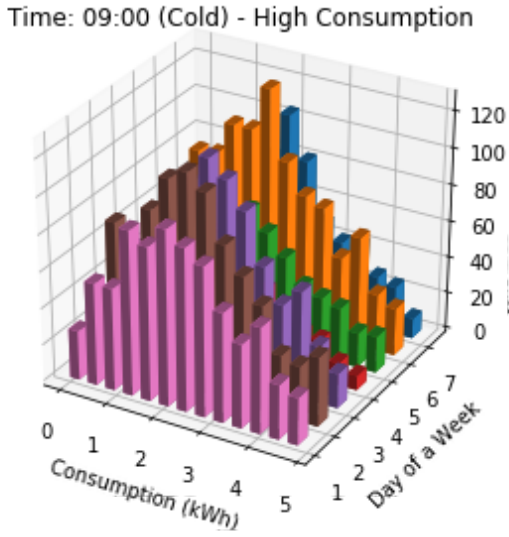
Time: 09:00 (Cold) - High Consumption



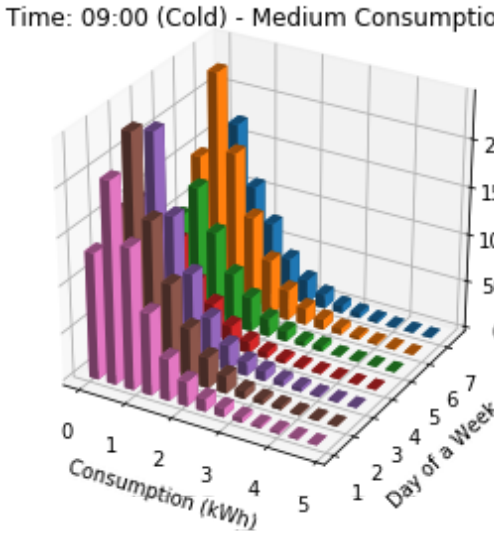
Time: 09:00 (Cold) - Medium Consumption



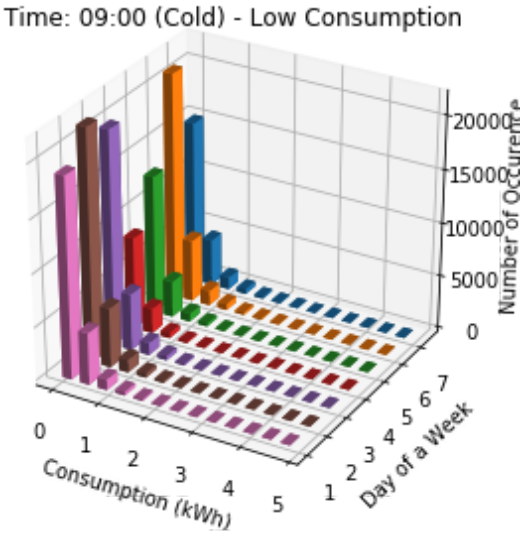
Time: 09:00 (Cold) - Low Consumption



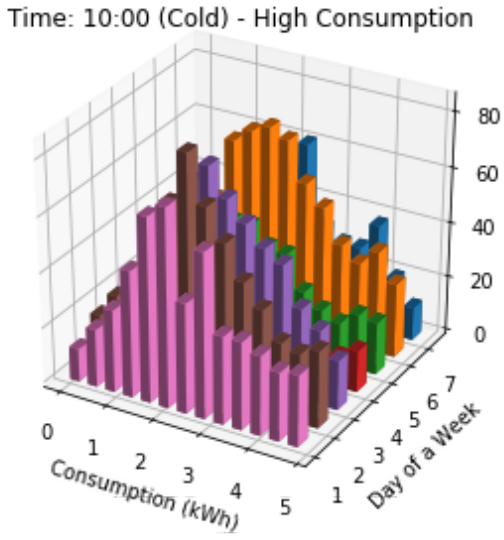
Time: 10:00 (Cold) - High Consumption



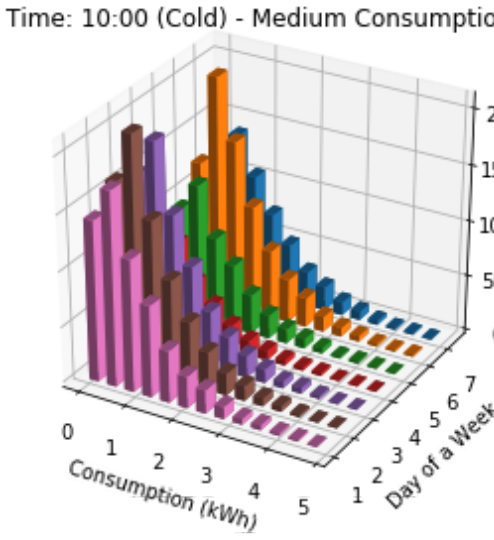
Time: 10:00 (Cold) - Medium Consumption



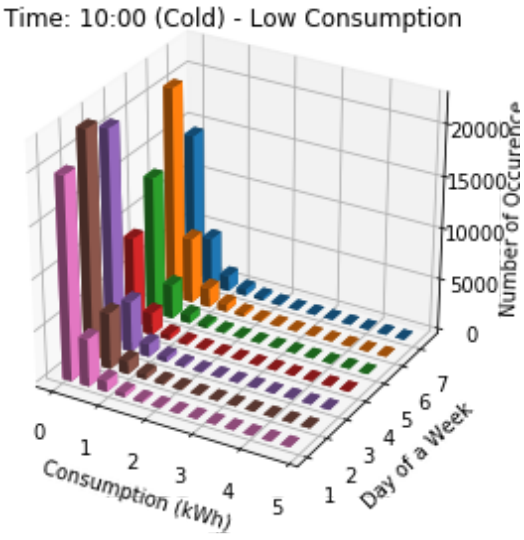
Time: 10:00 (Cold) - Low Consumption



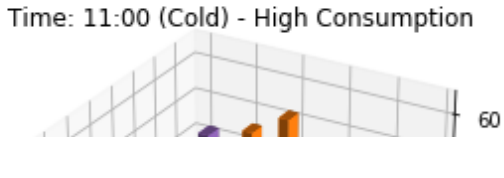
Time: 11:00 (Cold) - High Consumption

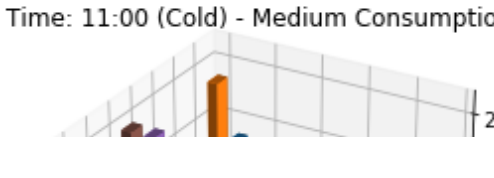


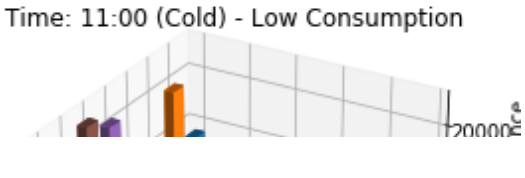
Time: 11:00 (Cold) - Medium Consumption

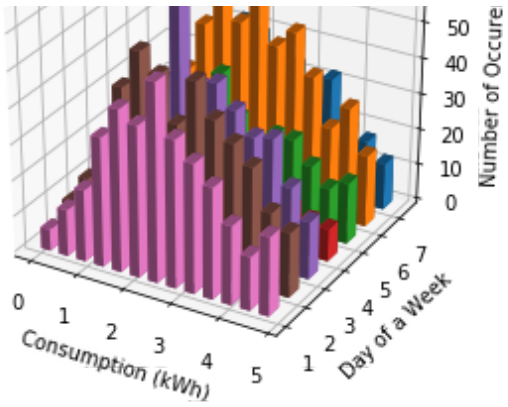


Time: 11:00 (Cold) - Low Consumption

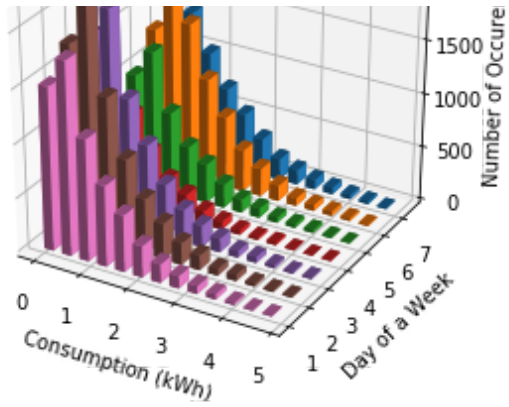




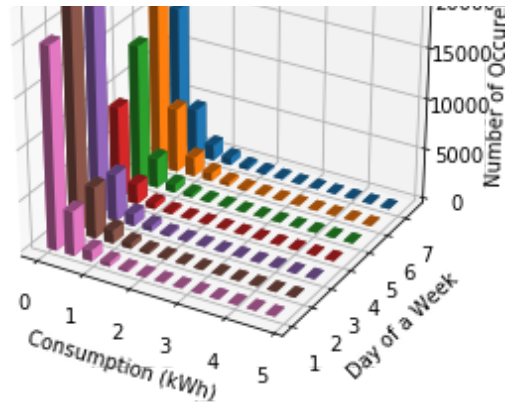




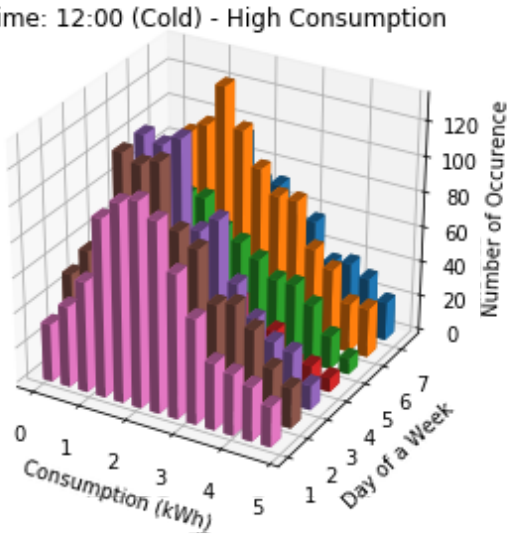
Time: 12:00 (Cold) - High Consumption



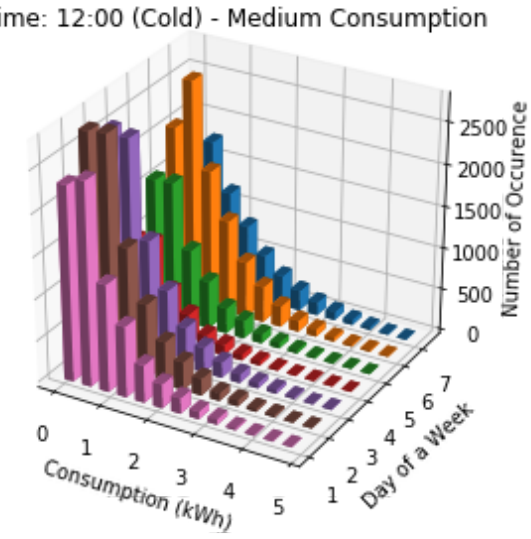
Time: 12:00 (Cold) - Medium Consumption



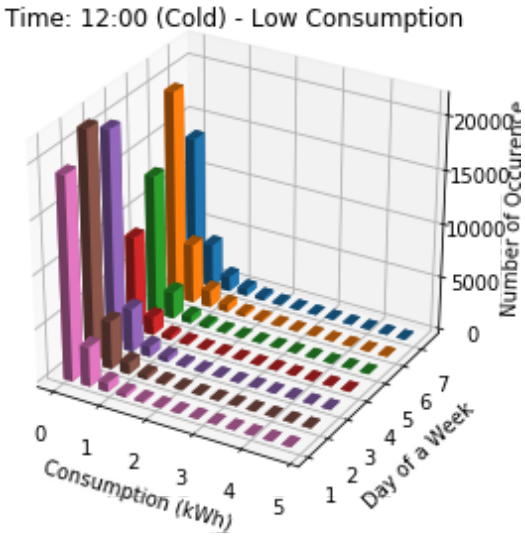
Time: 12:00 (Cold) - Low Consumption



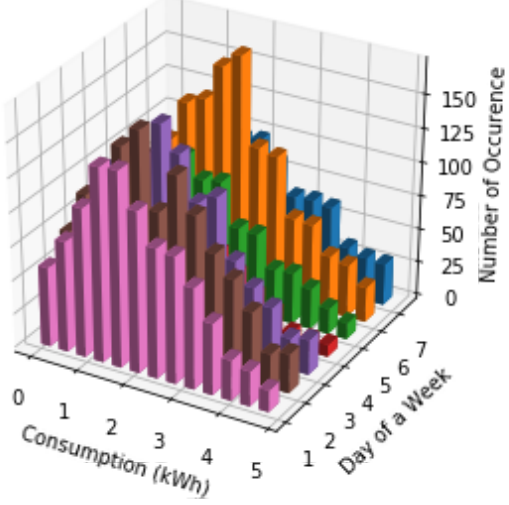
Time: 13:00 (Cold) - High Consumption



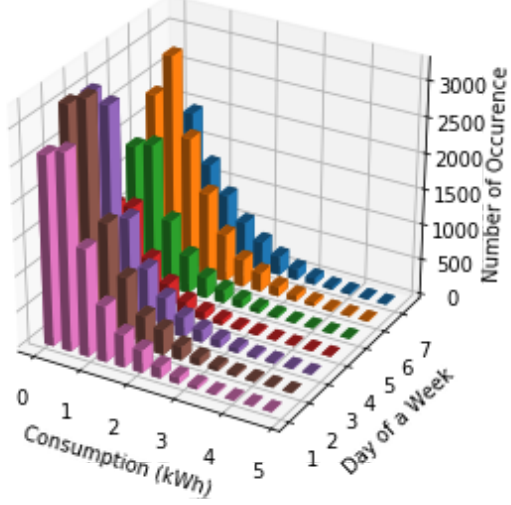
Time: 13:00 (Cold) - Medium Consumption



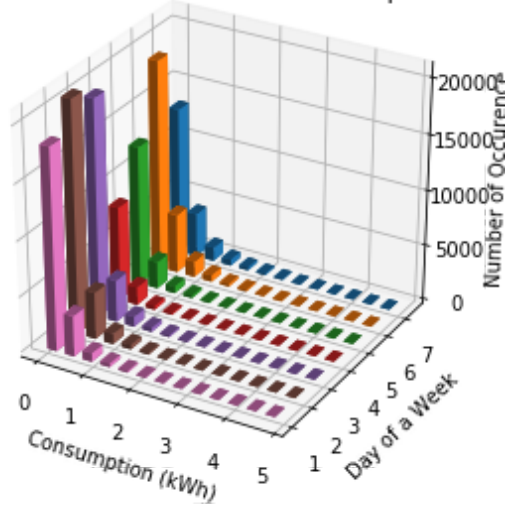
Time: 13:00 (Cold) - Low Consumption



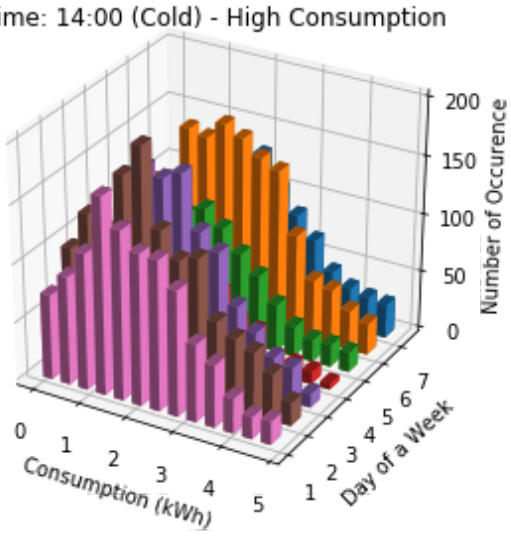
Time: 14:00 (Cold) - High Consumption



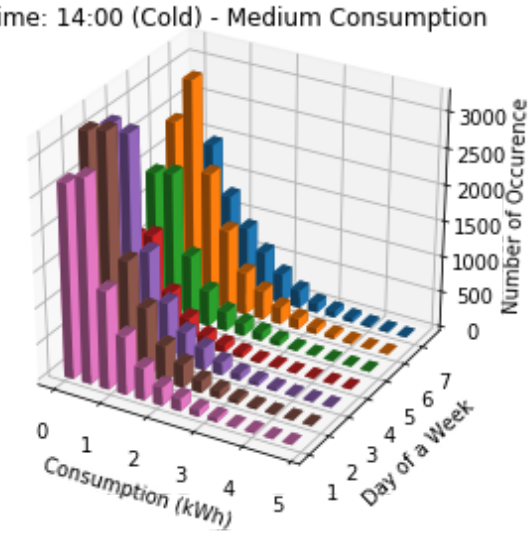
Time: 14:00 (Cold) - Medium Consumption



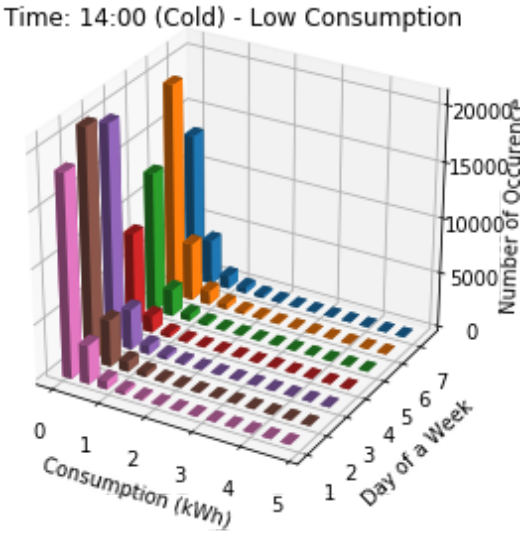
Time: 14:00 (Cold) - Low Consumption



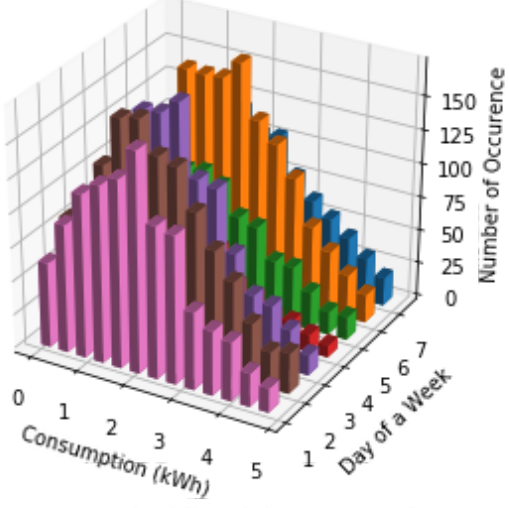
Time: 15:00 (Cold) - High Consumption



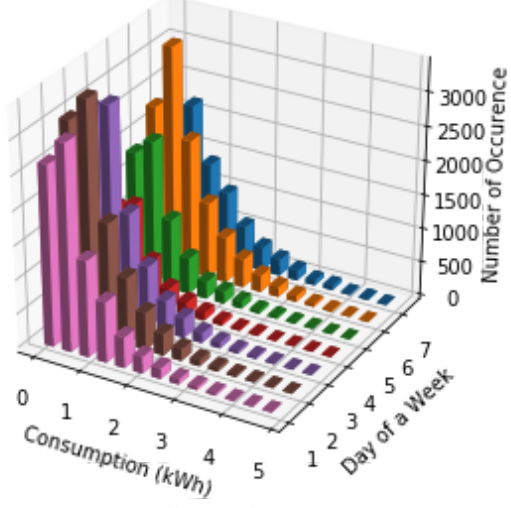
Time: 15:00 (Cold) - Medium Consumption



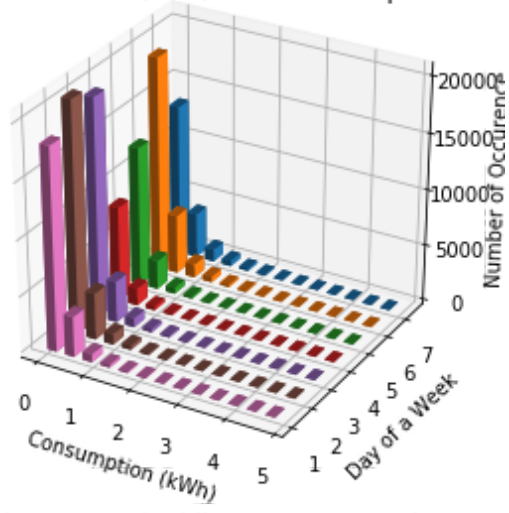
Time: 15:00 (Cold) - Low Consumption



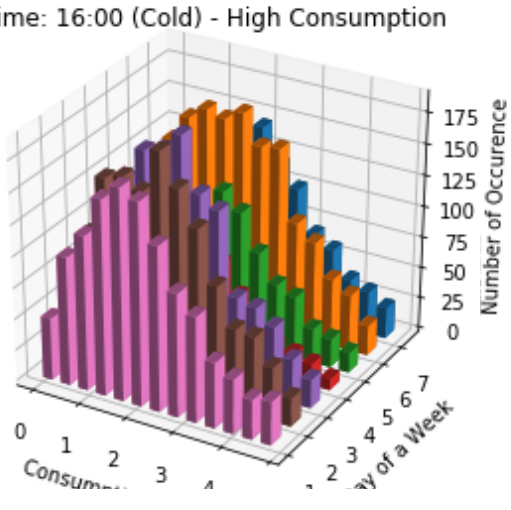
Time: 16:00 (Cold) - High Consumption

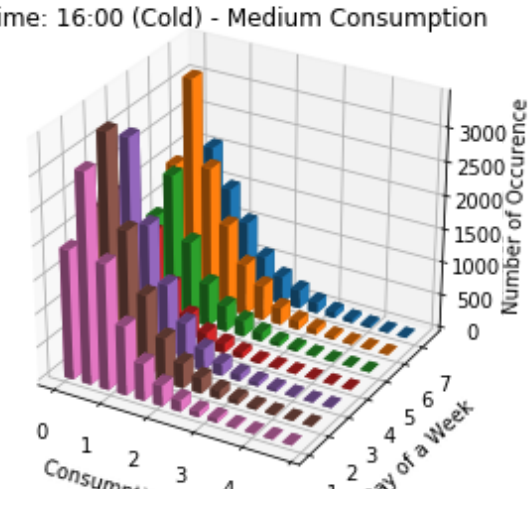


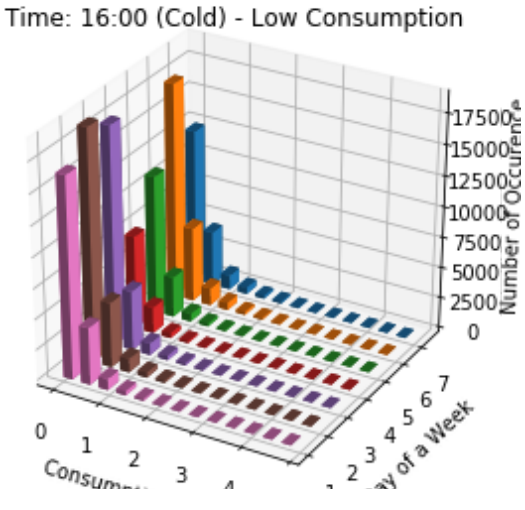
Time: 16:00 (Cold) - Medium Consumption

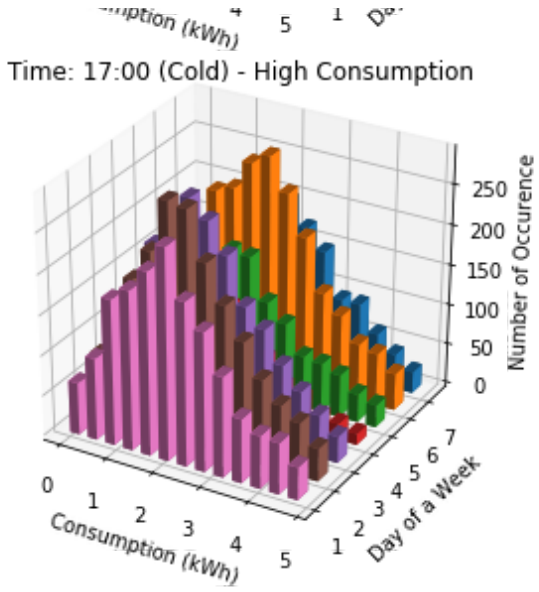


Time: 16:00 (Cold) - Low Consumption

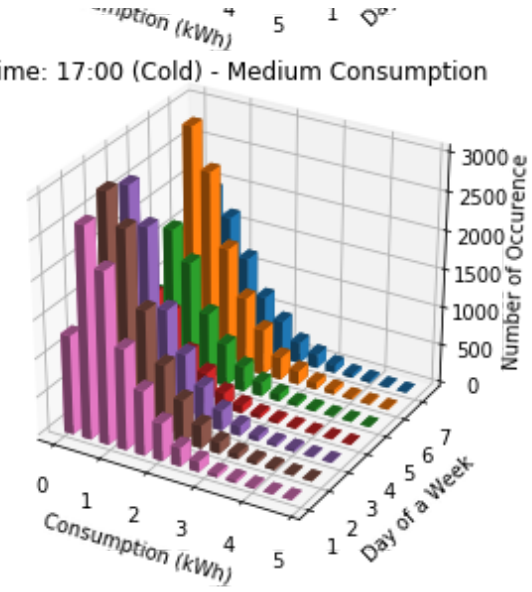




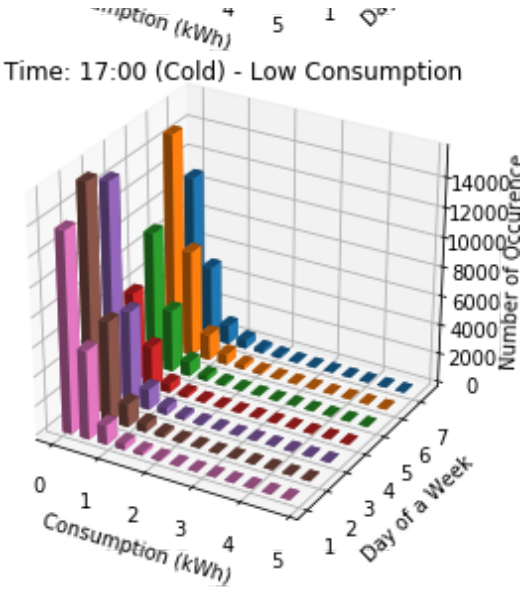




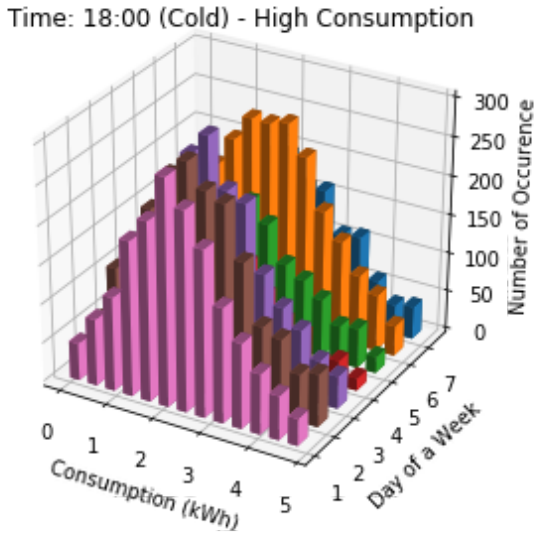
Time: 17:00 (Cold) - High Consumption



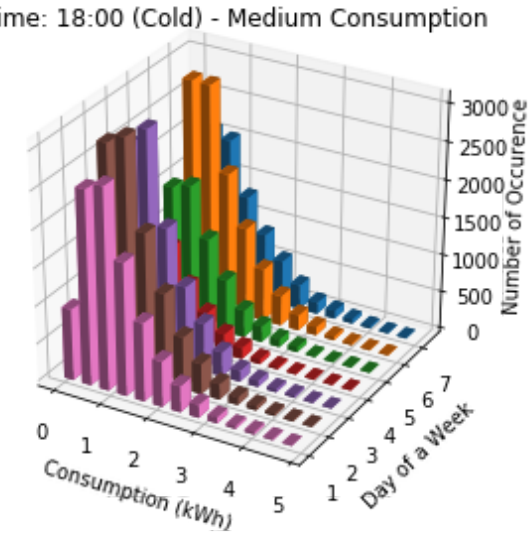
Time: 17:00 (Cold) - Medium Consumption



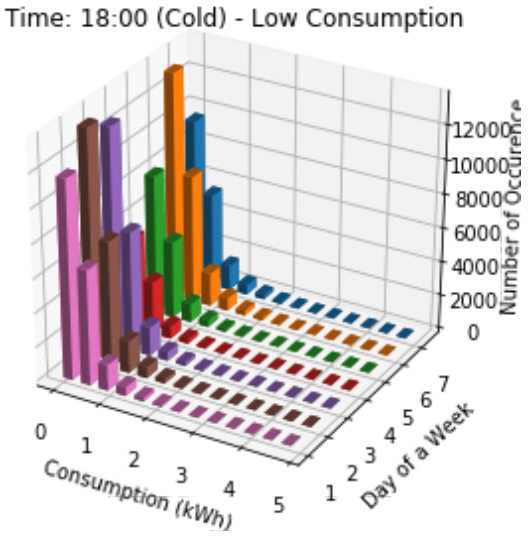
Time: 17:00 (Cold) - Low Consumption



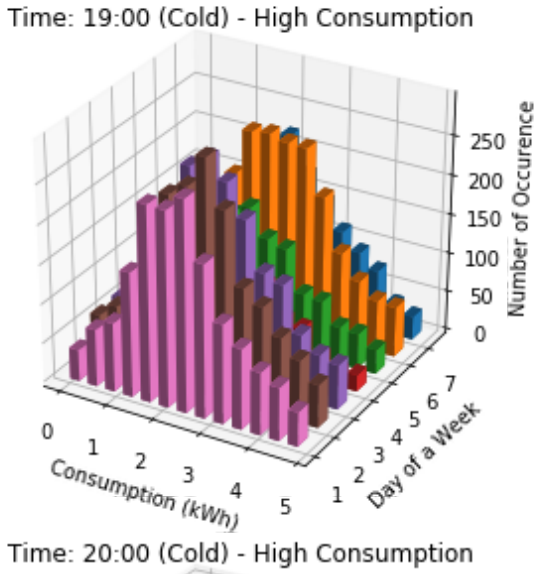
Time: 18:00 (Cold) - High Consumption



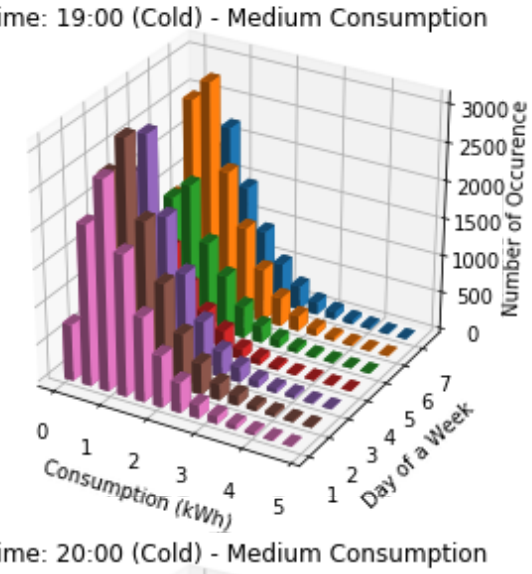
Time: 18:00 (Cold) - Medium Consumption



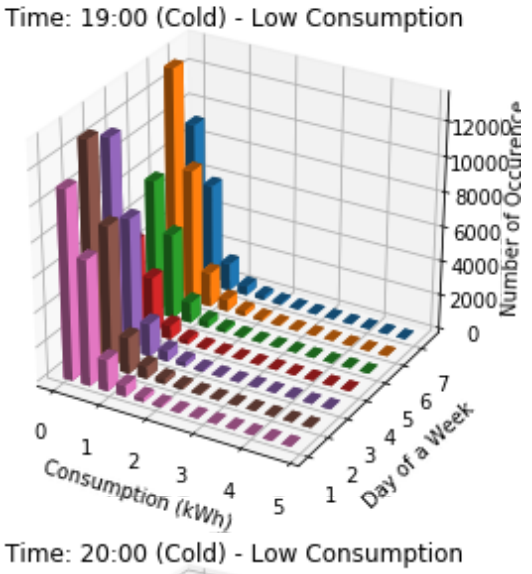
Time: 18:00 (Cold) - Low Consumption



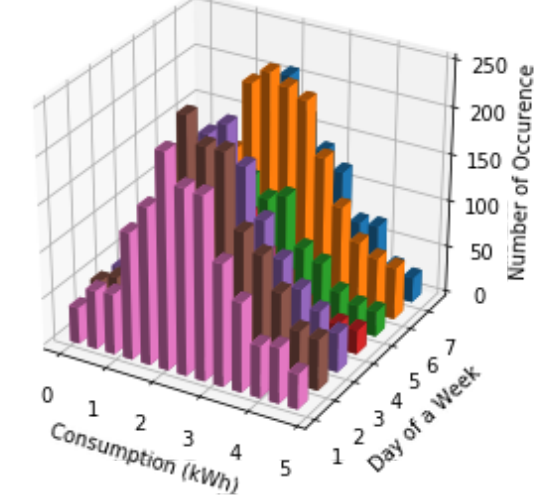
Time: 19:00 (Cold) - High Consumption



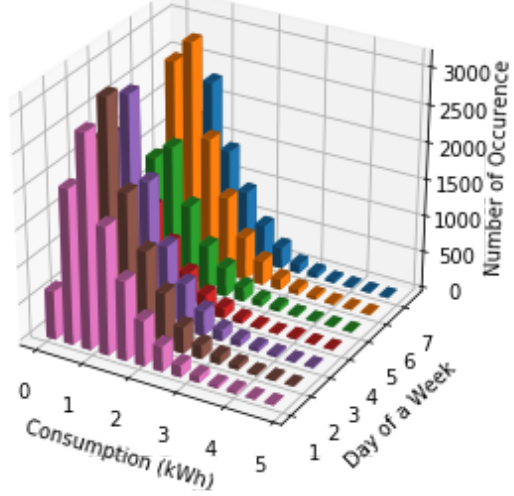
Time: 19:00 (Cold) - Medium Consumption



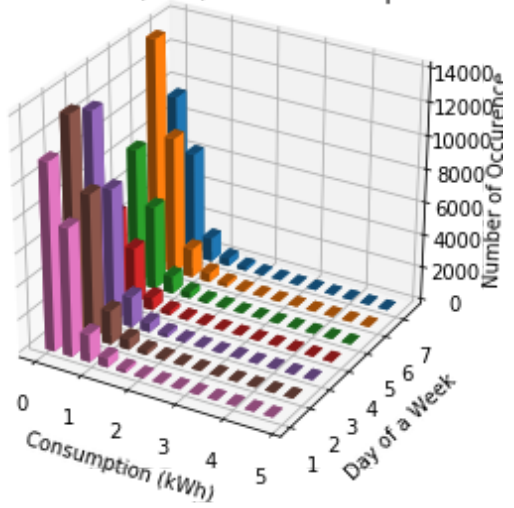
Time: 19:00 (Cold) - Low Consumption



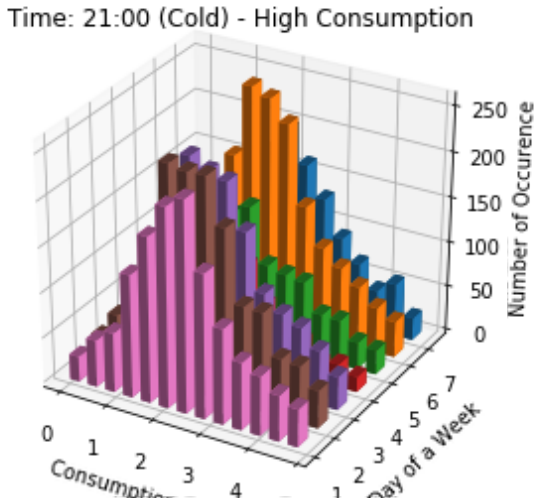
Time: 20:00 (Cold) - High Consumption



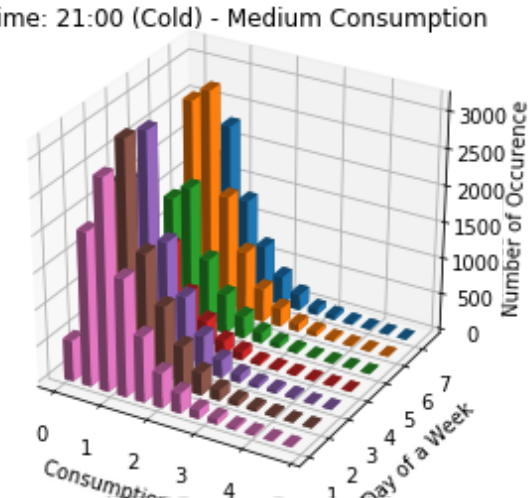
Time: 20:00 (Cold) - Medium Consumption



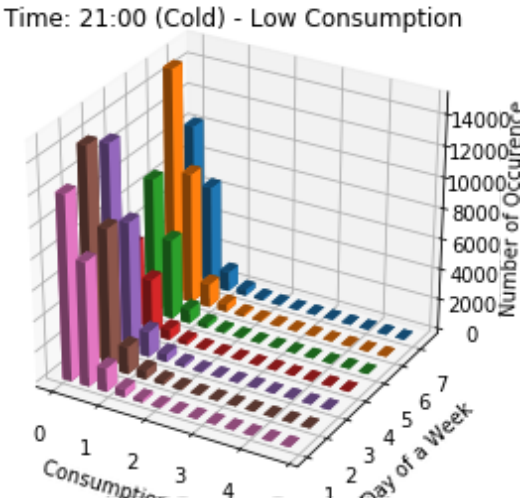
Time: 20:00 (Cold) - Low Consumption



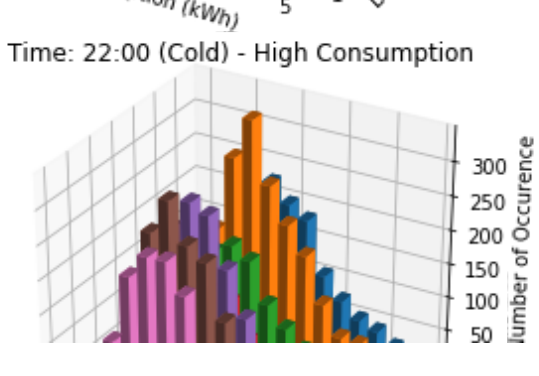
Time: 21:00 (Cold) - High Consumption



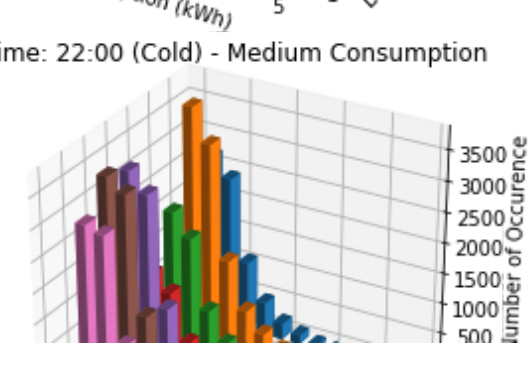
Time: 21:00 (Cold) - Medium Consumption



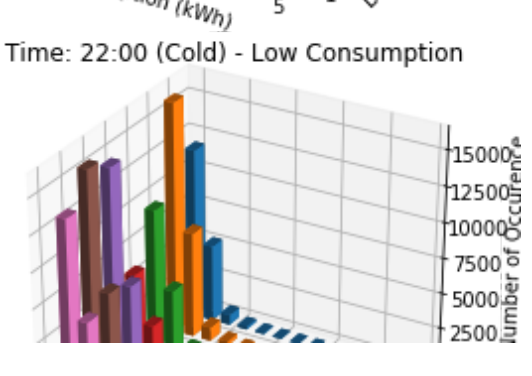
Time: 21:00 (Cold) - Low Consumption



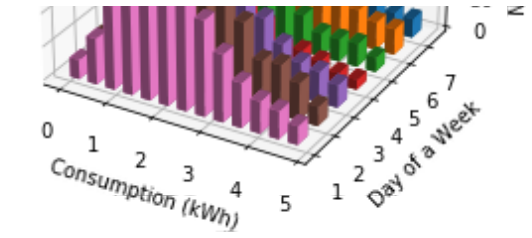
Time: 22:00 (Cold) - High Consumption



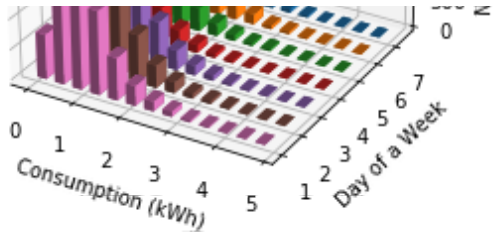
Time: 22:00 (Cold) - Medium Consumption



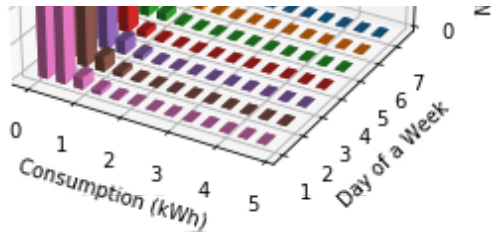
Time: 22:00 (Cold) - Low Consumption



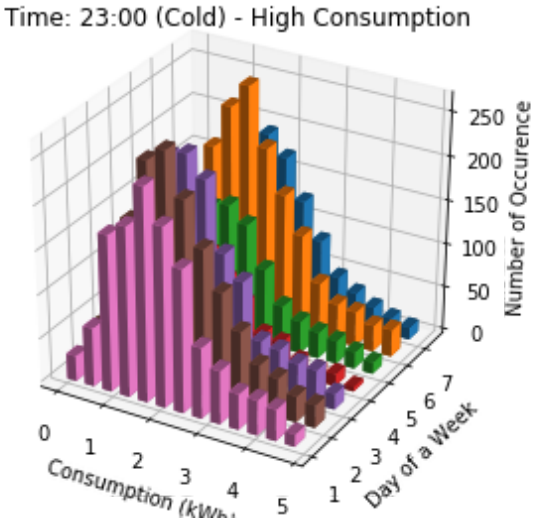
Time: 23:00 (Cold) - High Consumption

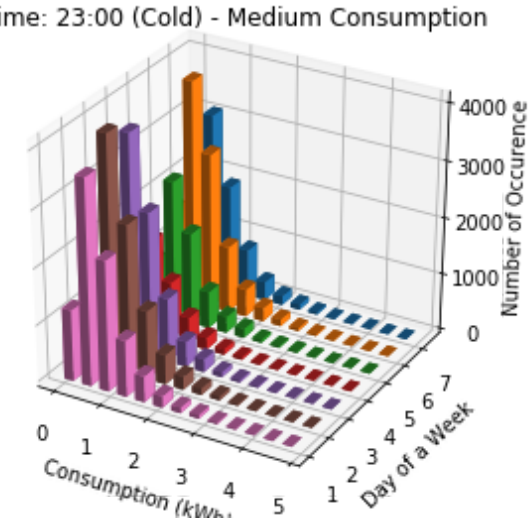


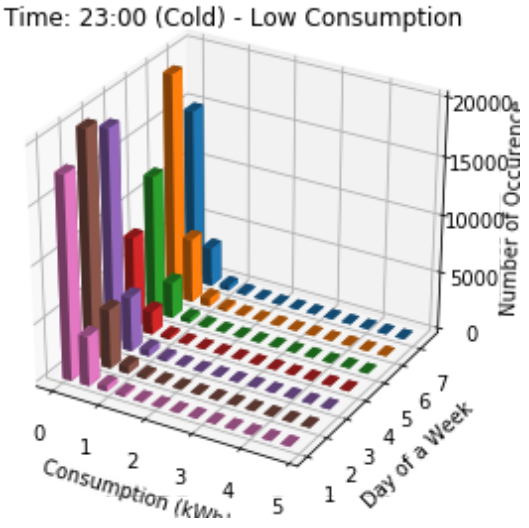
Time: 23:00 (Cold) - Medium Consumption



Time: 23:00 (Cold) - 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 [ ]:
```
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):
    print(i)
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:0
0')]['TempC'].values) # add temperature column to dataframe
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Day of a Week')
        ax.set_zlabel('Consumption (kWh)')
    else:
        x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        t = x.dropna(axis=1) # 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_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]['T
empC'].values) # add temperature column to dataframe
            t2['DayofWeek'] = np.array(pd.to_datetime(df_wea1h_nf_cold[df_Ntou1h_nf_cold.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' + ' (Cold)')
        ax.set_xlabel(r'Temperature (C$^o$)')
        ax.set_ylabel('Day of a Week')
        ax.set_zlabel('Consumption (kWh)')
plt.tight_layout()
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

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

In [8]:
```
df_Ntou1h_nf_cold.dropna(axis=1)
```

Out[8]:

	GMT	N0000	N0001	N0002	N0003	N0004	N0006	N0007	N0008	N0010	...	N4150	N4152	N4154	N4157	N4159	N4163	N4
0	2013-01-01 00:00:00	1.038	0.232	0.052	0.256	0.181	0.000	0.163	0.261	2.114	...	2.486	0.311	0.660	0.183	0.280	0.187	0.0
1	2013-01-01 01:00:00	0.484	0.267	0.101	0.192	0.185	0.000	0.152	0.090	2.132	...	2.066	0.031	0.307	0.158	0.201	0.304	0.0
2	2013-01-01 02:00:00	0.449	0.266	0.092	0.200	0.484	0.000	0.170	0.156	2.035	...	2.139	0.017	0.233	0.180	0.193	0.396	0.0
3	2013-01-01 03:00:00	0.390	0.230	0.034	0.262	0.132	0.000	0.152	0.094	1.914	...	1.706	0.014	0.211	0.197	0.191	0.355	0.1
4	2013-01-01 04:00:00	0.936	0.268	0.052	0.205	0.183	0.000	0.165	0.155	2.269	...	1.675	0.012	0.117	0.261	0.193	0.346	0.1
5	2013-01-01 05:00:00	0.388	0.265	0.100	0.199	0.112	0.000	0.159	0.085	4.034	...	1.415	0.012	0.099	0.252	0.190	0.343	0.0
6	2013-01-01 06:00:00	0.384	0.227	0.068	0.183	0.115	0.000	0.154	0.157	1.108	...	1.537	0.012	0.098	0.329	0.201	0.355	0.0
7	2013-01-01 07:00:00	0.406	0.264	0.033	0.253	0.117	0.000	0.169	0.185	0.590	...	1.558	0.012	0.098	0.277	0.190	0.324	0.2
8	2013-01-01 08:00:00	0.950	0.264	0.577	0.448	0.118	0.000	0.152	0.213	0.594	...	1.418	0.198	0.099	0.267	0.184	0.463	0.1
9	2013-01-01 09:00:00	0.448	0.228	0.238	0.598	0.305	0.000	0.167	0.151	0.576	...	1.658	0.164	0.169	0.215	0.498	0.337	0.1
10	2013-01-01 10:00:00	0.394	0.266	0.850	2.494	0.116	0.000	0.152	0.548	0.547	...	1.679	0.021	0.183	0.177	0.261	0.282	0.0
11	2013-01-01 11:00:00	0.478	0.265	0.362	1.514	0.269	0.000	0.156	0.201	0.583	...	1.882	0.012	1.179	0.165	0.279	0.385	0.1
12	2013-01-01 12:00:00	0.463	0.231	0.142	0.373	0.316	0.000	0.163	0.212	1.065	...	1.764	0.012	0.732	0.179	0.290	0.823	0.1
13	2013-01-01 13:00:00	0.647	0.267	0.212	0.488	1.016	0.000	0.148	0.440	1.471	...	1.719	0.011	3.297	0.154	0.282	0.382	0.1
14	2013-01-01 14:00:00	0.454	0.267	0.167	0.492	1.593	0.000	0.170	0.214	2.662	...	1.787	0.012	2.510	0.169	0.273	0.349	0.1
15	2013-01-01 15:00:00	0.411	0.231	0.289	0.407	2.116	0.000	0.150	0.298	4.211	...	2.600	1.064	1.136	0.166	0.270	0.538	0.1
16	2013-01-01 16:00:00	0.534	0.268	0.568	0.525	2.349	0.000	0.162	0.871	2.084	...	2.805	1.519	1.586	0.152	0.275	0.534	0.1
17	2013-01-01 17:00:00	0.502	0.256	0.763	0.689	1.281	0.000	0.154	1.186	1.893	...	2.526	1.296	1.940	0.175	0.299	0.576	0.2
18	2013-01-01 18:00:00	0.534	0.234	0.724	0.702	1.481	0.000	0.196	0.566	5.039	...	2.791	1.398	3.074	0.155	0.458	0.802	0.2
19	2013-01-01 19:00:00	0.709	0.265	0.663	1.357	1.412	0.000	0.223	1.676	4.480	...	2.643	1.482	1.967	0.151	0.360	0.768	0.2
20	2013-01-01 20:00:00	0.496	0.245	0.637	0.855	0.784	0.000	0.195	1.387	4.600	...	2.667	0.762	1.325	0.170	0.360	0.610	0.1

	GMT	N0000	N0001	N0002	N0003	N0004	N0006	N0007	N0008	N0010	...	N4150	N4152	N4154	N4157	N4159	N4163	N4
21	2013-01-01 21:00:00	0.341	0.245	0.456	0.549	0.682	0.000	0.227	0.470	4.591	...	2.108	0.315	0.372	0.455	0.361	0.610	0.2
22	2013-01-01 22:00:00	0.442	0.268	0.392	0.460	0.640	0.000	0.211	0.495	4.492	...	2.646	0.305	0.271	0.976	0.361	0.645	0.2
23	2013-01-01 23:00:00	0.297	0.227	0.401	0.231	0.705	0.000	0.194	0.270	4.460	...	3.247	0.336	0.272	0.450	0.357	0.610	0.0
24	2013-01-02 00:00:00	0.324	0.264	0.387	0.112	1.156	0.000	0.159	0.149	4.206	...	1.649	0.016	0.224	0.441	0.192	0.383	0.0
25	2013-01-02 01:00:00	0.322	0.265	0.244	0.101	0.794	0.000	0.153	0.081	1.992	...	2.934	0.012	0.147	0.440	0.191	0.365	0.1
26	2013-01-02 02:00:00	0.292	0.232	0.126	0.139	0.188	0.000	0.170	0.152	1.349	...	2.058	0.012	0.133	0.398	0.192	0.327	0.0
27	2013-01-02 03:00:00	0.387	0.267	0.098	0.062	0.120	0.000	0.152	0.093	1.047	...	1.509	0.012	0.136	0.384	0.203	0.457	0.1
28	2013-01-02 04:00:00	0.526	0.236	0.121	0.057	0.120	0.000	0.170	0.134	0.959	...	1.614	0.012	0.132	0.352	0.190	0.461	0.0
29	2013-01-02 05:00:00	0.841	0.276	0.053	0.058	0.109	0.000	0.150	0.117	0.622	...	1.247	0.012	0.131	0.368	0.184	0.375	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8682	2013-12-28 18:00:00	1.287	0.587	0.337	0.755	1.272	0.246	0.349	1.042	0.410	...	2.816	0.705	0.368	0.388	0.395	0.073	0.2
8683	2013-12-28 19:00:00	0.611	0.496	0.509	1.188	1.126	0.261	0.350	1.412	0.413	...	2.973	0.532	1.749	0.448	0.398	0.079	0.2
8684	2013-12-28 20:00:00	0.504	0.581	0.457	1.335	1.355	0.193	0.348	0.664	0.363	...	2.562	0.536	2.539	0.429	0.395	0.079	0.2
8685	2013-12-28 21:00:00	0.448	0.514	0.221	0.760	0.934	0.272	1.070	0.332	4.202	...	2.688	0.348	1.741	0.508	0.389	0.077	0.2
8686	2013-12-28 22:00:00	0.350	0.773	0.078	0.584	0.782	0.194	0.316	0.336	5.408	...	2.932	0.116	0.391	0.460	0.372	0.078	0.1
8687	2013-12-28 23:00:00	0.509	0.640	0.106	0.159	0.655	0.071	0.274	0.471	5.092	...	2.316	0.025	0.172	0.445	0.386	0.078	0.0
8736	2013-12-31 00:00:00	0.509	0.647	0.066	0.288	0.852	0.110	0.164	0.297	4.670	...	2.139	0.028	0.181	0.375	0.178	0.076	0.0
8737	2013-12-31 01:00:00	0.507	0.561	0.041	0.267	0.640	0.070	0.164	0.243	5.758	...	1.451	0.274	0.153	0.289	0.180	0.078	0.0
8738	2013-12-31 02:00:00	0.406	0.496	0.093	0.216	0.326	0.031	0.170	0.244	4.635	...	1.279	0.025	0.152	0.290	0.180	0.079	0.0
8739	2013-12-31 03:00:00	0.378	0.468	0.106	0.262	0.288	0.067	0.157	0.238	1.093	...	1.331	0.025	0.152	0.297	0.181	0.076	0.1
8740	2013-12-31 04:00:00	0.370	0.565	0.088	0.295	0.274	0.112	0.155	0.314	0.429	...	1.064	0.025	0.104	0.291	0.158	0.112	0.1
8741	2013-12-31 05:00:00	0.385	0.495	0.040	0.217	0.256	0.062	0.168	0.282	0.429	...	0.851	0.025	0.093	0.292	0.146	0.082	0.0

	GMT	N0000	N0001	N0002	N0003	N0004	N0006	N0007	N0008	N0010	...	N4150	N4152	N4154	N4157	N4159	N4163	N4
8742	2013-12-31 06:00:00	0.375	0.564	0.065	0.206	0.248	0.028	0.167	0.207	0.396	...	0.937	0.025	0.093	0.289	0.163	0.101	0.0
8743	2013-12-31 07:00:00	0.338	0.505	0.107	0.258	0.264	0.029	0.156	0.268	0.359	...	0.833	0.051	0.092	0.287	0.233	0.127	0.2
8744	2013-12-31 08:00:00	0.611	0.562	0.487	0.374	0.289	0.387	0.403	0.254	0.415	...	1.508	0.493	0.094	0.298	0.261	0.122	0.1
8745	2013-12-31 09:00:00	0.415	0.637	0.259	0.508	0.855	0.185	0.272	0.302	0.656	...	1.485	0.046	0.093	0.763	0.496	0.079	0.0
8746	2013-12-31 10:00:00	0.359	0.682	0.641	0.520	0.591	0.314	0.230	0.253	2.195	...	1.700	0.097	0.207	0.435	0.254	0.077	0.0
8747	2013-12-31 11:00:00	0.391	0.588	0.330	0.581	0.509	0.169	0.214	0.189	3.227	...	1.663	0.025	0.228	0.305	0.250	0.073	0.0
8748	2013-12-31 12:00:00	0.400	0.451	0.522	1.391	0.154	0.361	0.200	0.196	6.577	...	1.281	0.024	0.753	0.815	0.293	0.071	0.0
8749	2013-12-31 13:00:00	0.330	0.499	1.139	0.716	1.443	0.210	0.210	0.273	7.442	...	1.048	1.125	0.347	0.377	0.579	0.112	0.2
8750	2013-12-31 14:00:00	0.307	0.527	0.119	0.418	1.645	0.168	0.686	0.258	6.469	...	1.597	1.260	1.979	0.528	0.264	0.105	0.1
8751	2013-12-31 15:00:00	0.414	0.517	0.504	0.323	0.814	0.082	1.125	0.207	4.750	...	1.927	1.186	1.523	0.578	0.264	0.095	0.1
8752	2013-12-31 16:00:00	0.420	0.530	0.367	0.624	0.773	0.043	0.282	0.211	2.845	...	2.396	1.140	1.663	0.130	0.261	0.079	0.1
8753	2013-12-31 17:00:00	1.336	0.509	0.293	0.591	0.460	0.041	0.668	0.185	2.657	...	2.457	0.542	0.678	0.129	0.356	0.080	0.2
8754	2013-12-31 18:00:00	0.632	0.544	0.233	0.659	0.244	0.135	0.722	0.267	5.577	...	2.861	0.830	1.248	0.134	0.370	0.073	0.2
8755	2013-12-31 19:00:00	0.409	0.503	0.229	0.891	0.288	0.192	0.513	0.255	4.977	...	2.797	0.654	0.620	0.128	0.390	0.071	0.1
8756	2013-12-31 20:00:00	0.475	0.565	0.218	0.740	0.246	0.263	1.174	0.189	1.928	...	1.852	0.644	0.190	0.129	0.390	0.070	0.2
8757	2013-12-31 21:00:00	0.448	0.499	0.108	1.193	0.314	0.204	0.443	0.201	1.721	...	1.918	0.366	0.186	0.128	0.391	0.070	0.2
8758	2013-12-31 22:00:00	0.319	0.785	0.065	1.007	0.270	0.254	0.496	0.217	1.790	...	1.865	0.263	1.259	0.132	0.392	0.069	0.1
8759	2013-12-31 23:00:00	0.271	0.602	0.109	0.304	0.286	0.142	0.475	0.263	1.853	...	1.916	0.096	0.262	0.129	0.373	0.072	0.0

1992 rows × 2552 columns

In [14]:
```
len(kmeans.labels_)
```

Out[14]:
```
3720
```

In [8]:
```
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 = 2551 # total household numbers
houseLabel = [] # store the 24hour group labels for each hourshold
for i in range(2551):
    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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.dropna(axis=1).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_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(i) + ':00:00')]
        x.set_index('GMT', inplace = True)
        x = x.dropna(axis=1).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_Ntou1h_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[8]:

	Label	counts
140	00000200100002100220100 0	1088
853	11221021021120021102022 2	50
953	22112112212211212011211 1	16
357	00000201100002100220100 0	15
152	00000200100002100222100 0	14
521	10000200100002100220100 0	13
29	00000200000002100220100 0	13
142	00000200100002100220102 0	12
127	00000200100002100200100 0	12
119	00000200100002100100100 0	11
135	00000200100002100202100 0	10
387	00000220100002100220100 0	9
147	00000200100002100222000 0	9
126	00000200100002100120100 0	9
136	00000200100002100220000 0	8
243	00000200120002100220100 0	8
712	11200200021120021102022 2	8
57	00000200020002100220100 0	8
162	00000200100002101220100 0	7
630	11000200100002100220102 2	7
141	00000200100002100220100 2	7
557	10000200121120021102022 2	7
750	11200201021120021102022 2	6
137	00000200100002100220020 0	6
616	11000200021120021102022 2	6
499	10000200021120021102022 2	6
334	00000201021120021102022 2	6
124	00000200100002100102022 2	6
522	10000200100002100220100 2	6
76	00000200021120021102022 0	6
...	...	...
408	00000221020020021102022 2	1
409	00000221020020101102022 0	1
385	00000220100002100220022 0	1
383	00000220100002100102022 2	1
352	00000201100002100102022 0	1
382	00000220100002100100100 2	1
355	00000201100002100200100 2	1
358	00000201100002100222022 2	1
359	00000201100002100222100 0	1
360	00000201100002101100100 0	1
361	00000201100002101102022 0	1
362	00000201100002101120100 0	1
363	00000201100002121202100 0	1
364	00000201100020021220100 0	1
365	00000201100022101002102 0	1
366	00000201100022101220100 0	1
367	00000201120002121102100 0	1
368	00000201121002100220100 0	1
369	00000201121002101220100 0	1

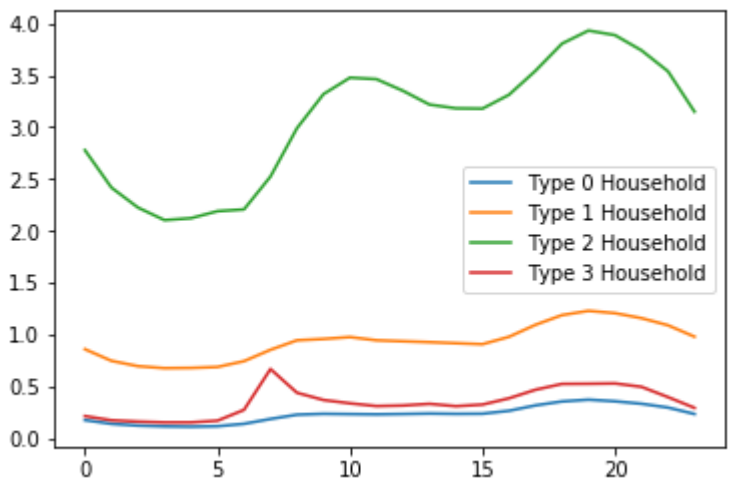
	Label	counts
370	00000201121020000102022 2	1
371	00000201121022121102022 2	1
373	00000202000002100220100 0	1
374	00000202021120021100100 0	1
375	00000202021120221102022 0	1
376	00000202100002100220100 0	1
377	00000220000120021102022 0	1
378	00000220021120021102022 2	1
379	00000220100000100222022 0	1
380	00000220100002000220000 0	1
973	22222112211120021011211 1	1

974 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 more than 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.

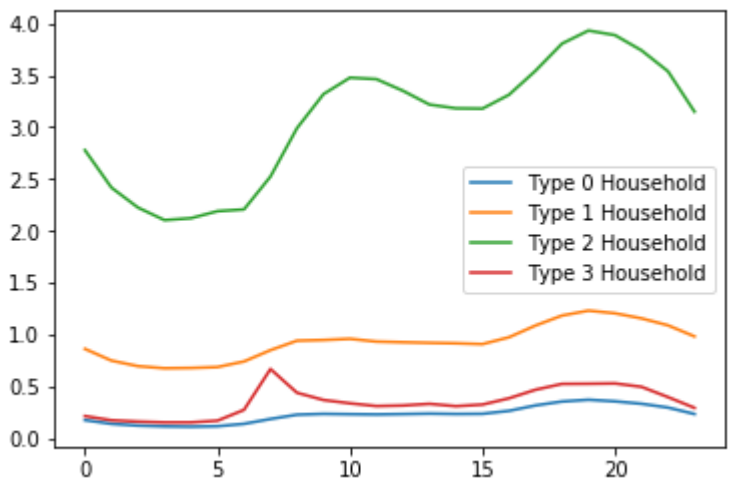
In [9]:
```
for i in range(4):
    load = []
    for j in range(24):
        if j <= 9:
            x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains('0' + str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.dropna(axis=1)
            x = x[list(houseLableDf[houseLableDf.Label == houseGroupDf.Label.iloc[i]].House)]
            load.append(x.mean().mean())
        else:
            x = df_Ntou1h_nf_cold[df_Ntou1h_nf_cold.GMT.str.contains(str(j) + ':00:00')]
            x.set_index('GMT', inplace = True)
            x = x.dropna(axis=1)
            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()
```



Note: the above grouping process are based on kmean-clustering, it's worth mentioning that, the kmean-clustering results highly realted to the initial value randomization. And it seems that this randomization affects the household 24 hour group significantly. So this means that it may not be an appropriate approach to use k-mean clustering algorithm here for the household grouping.

For example, if we specify the kmean to be randomized initilization, and see if the final main groups have a significant difference.

In [10]:
```
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 = 2551 # total household numbers
houseLabel = [] # store the 24hour group labels for each hourshold
for i in range(2551):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
for i in range(24):
    if i <= 9:
        x = df_Ntou1h_nf_cold[df_Ntou1h_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_Ntou1h_nf_cold[df_Ntou1h_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_Ntou1h_nf_cold.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_Ntou1h_nf_cold[df_Ntou1h_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_Ntou1h_nf_cold[df_Ntou1h_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()
```



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 19:00, 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 19:00 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 09:00 (maybe they work in the morning, at home with some works that consumes a lot of energy, maybe breakfast, maybe heating at every morning?)

For warm season

In [9]:
```
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 = 2018 # total household numbers
houseLabel = [] # store the 24hour group labels for each hoursehold
for i in range(2018):
    houseLabel.append('')
houseLabel = np.array(houseLabel)
cons_type = ['High Consumption', 'Medium Consumption', 'Low Consumption']
for i in range(24):
    if i <= 9:
        x = df_Ntou1h_nf_warm[df_Ntou1h_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_Ntou1h_nf_warm[df_Ntou1h_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_Ntou1h_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_Ntou1h_nf_warm[df_Ntou1h_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_Ntou1h_nf_warm[df_Ntou1h_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()
```