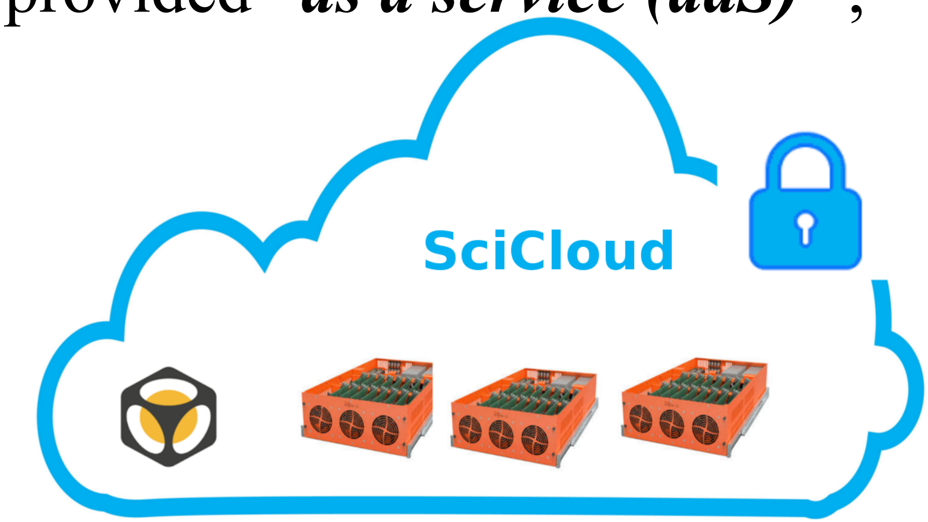
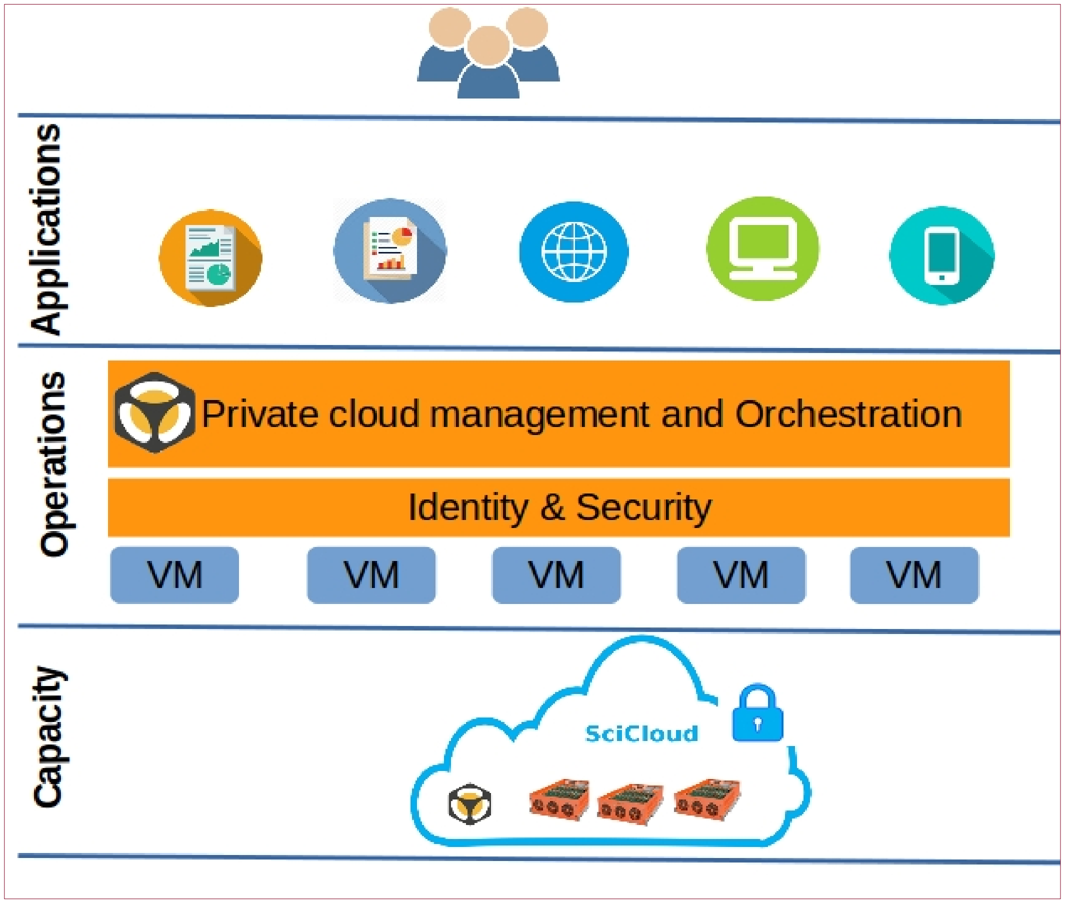
What is Scientific Cloud (SciCloud)?

- **SciCloud** is a private cloud for scientific computing for researchers
- It is running:
  - 18 physical servers
  - 80 cores
  - 1/5 TB of memory (RAM)
  - 12 TB storage (including backup storage)
- Provides customized images
- Real-time scalable resources provided "***as a service (aaS)***" , e.g.:
  - Data (DaaS)
  - Analytics *(AaaS)*

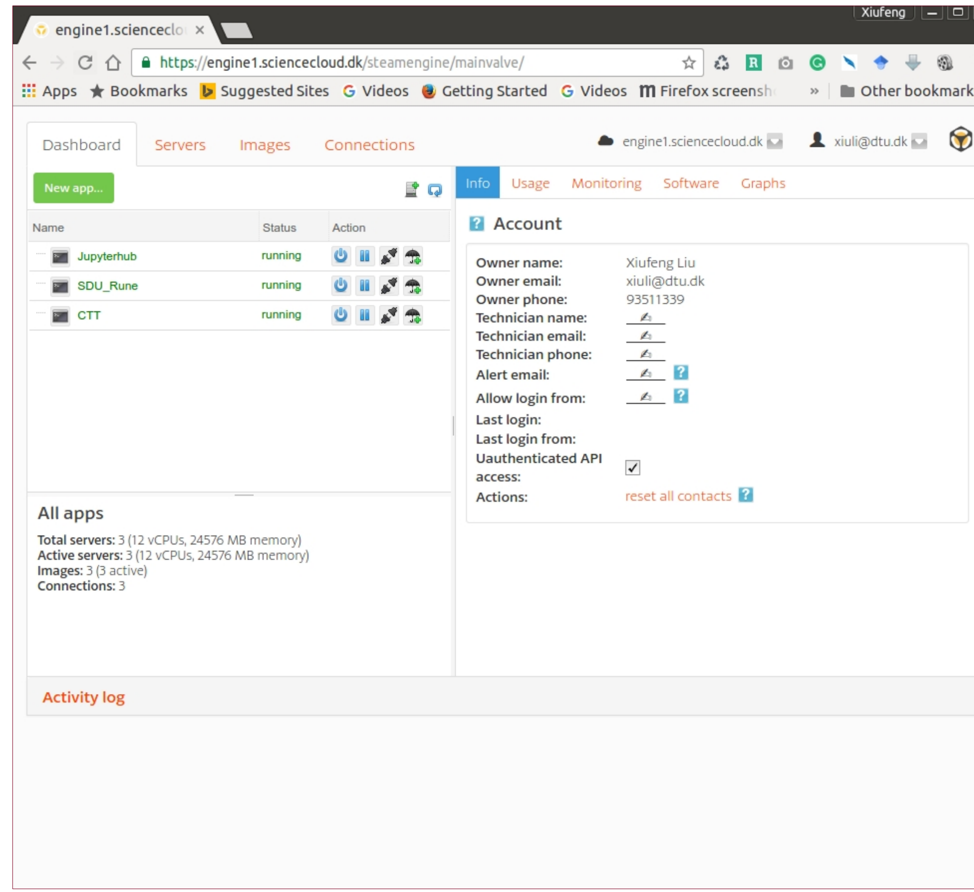


SciCloud Overview

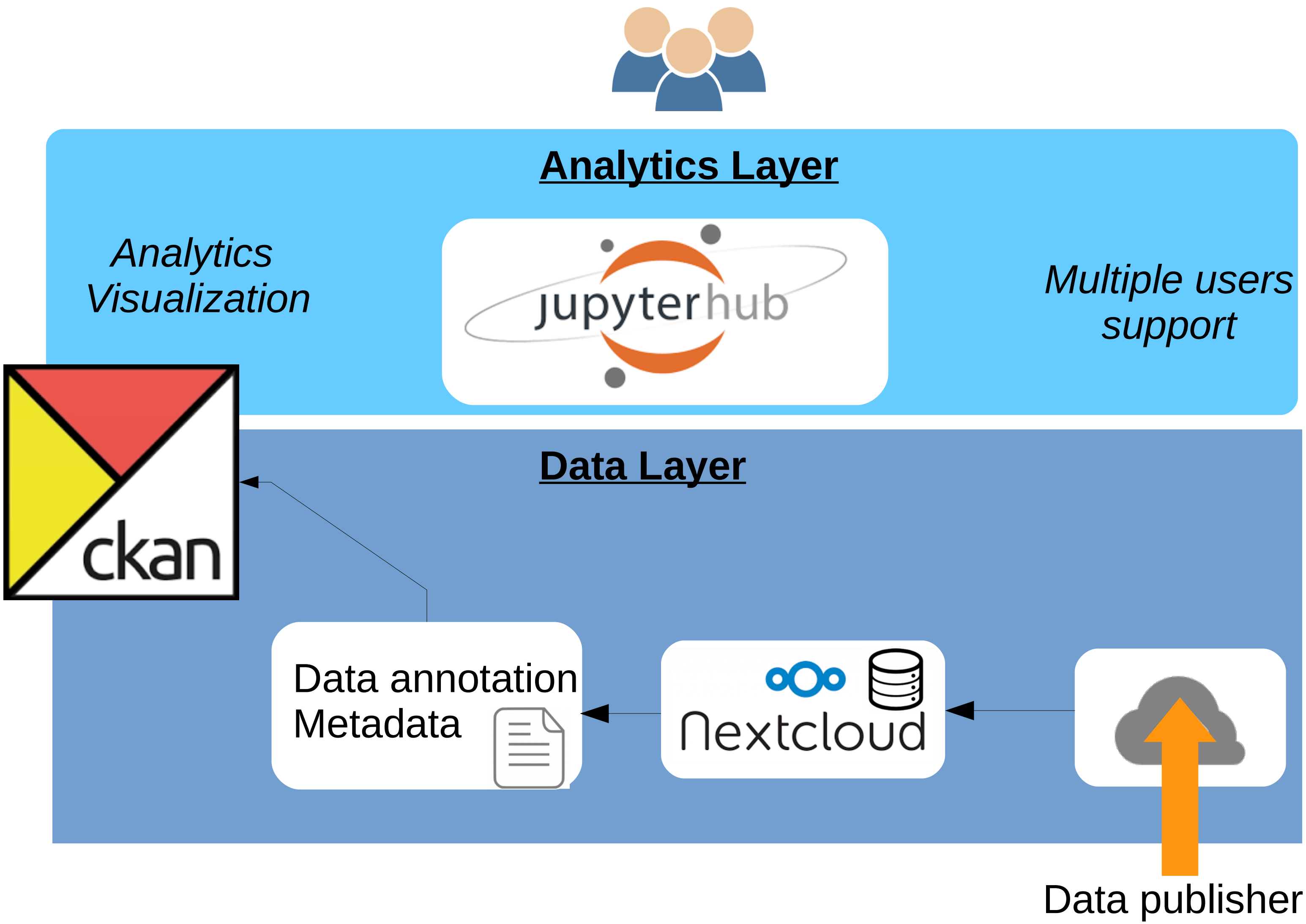
Architecture



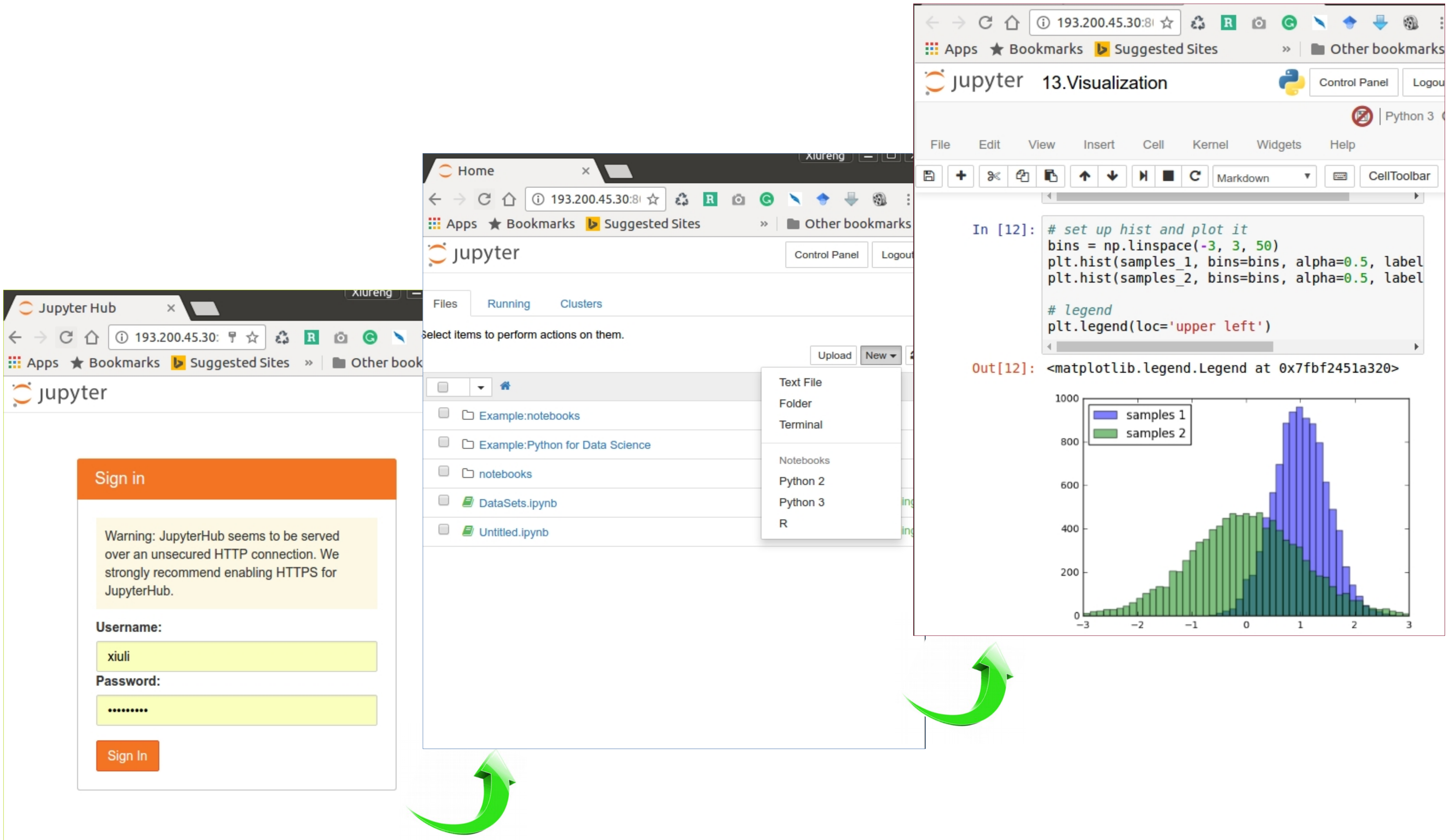
Web-based admin console



SciCloud – *Analytics as a Service (Jupyterhub)*



SciCloud – *Analytics as a Service (Jupyterhub)*

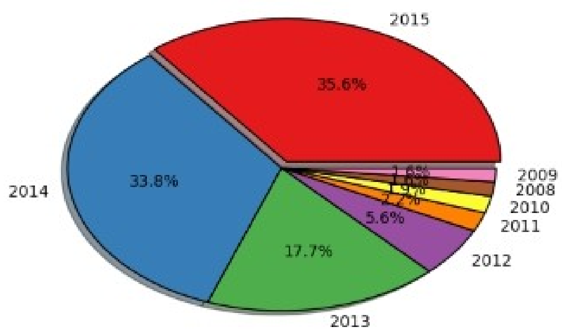


0. Description of the data

The data used is the district heating consumption of single family houses in Aarhus. The data contains the consumption of customers at 1 hour intervals. There are 15,065 installations and the total number of 24 hour load profiles is 9,051,636. The data ranges from December 2008 to November 2015. However, the data range is different depending on the installations. For example, there are a small number of installations having the data before 2013. There are 7,507 installations with data ranging in 2013, and 14,333 in 2014, and 15061 in 2015. We use the data of 2015 in this experiment.

In [98]:
```
dailyProfiles = [15061,14333, 7507, 2384, 920, 805, 681, 663]
Years = [2015, 2014, 2013, 2012, 2011, 2010, 2008, 2009]

fracs = [100.0*x/sum(dailyProfiles) for x in dailyProfiles]
explode=(0.05,0,0, 0, 0, 0, 0,0)
pie(fracs, explode=explode, labels=Years, colors=cm.Set1(np.arange(8)/8.0), autopct='%1.1f%%', shadow=True);
```



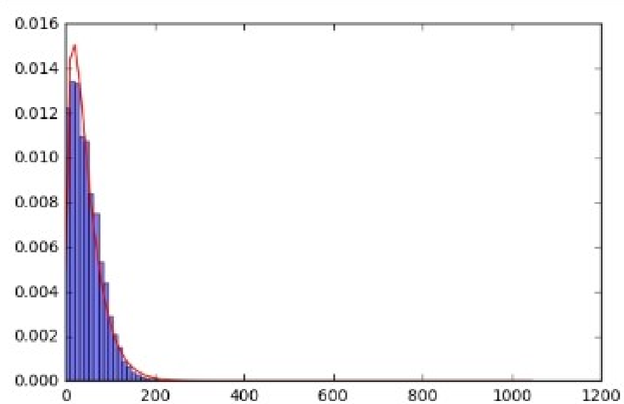
In the following, we use histogram to plot the distribution of daily profiles of 2015, and we see that the gamma distribution can fix the best.

In [73]:
```
X = np.load('HourlyReadings_2015.npy')
```

In [99]:
```
dailySum = X.sum(axis=1)
x = np.linspace(dailySum.min(), dailySum.max(), 100)

# fit
param = stats.gamma.fit(dailySum)
pdf_fitted = stats.gamma.pdf(x, *param)
plt.plot(x, pdf_fitted, color='r')

# plot the histogram
plt.hist(dailySum, normed=True,  bins=100, alpha=0.5);
```



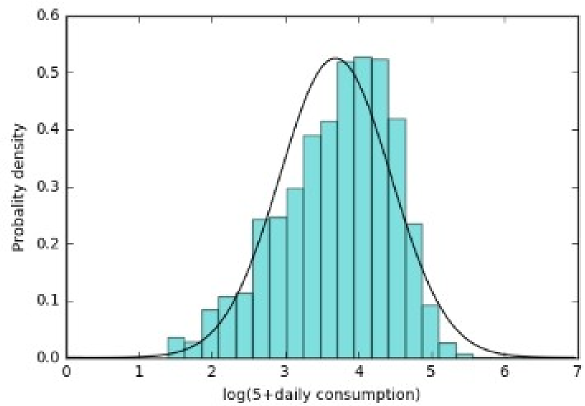
The mixture of log normal distributions fits best the actual data.

$f(a) = \frac{1}{\sqrt{2\pi\delta^2}} e^{\frac{-(\log(n+a)-\mu)^2}{2\delta^2}}$ where $n = 0, 1, \ldots$

In [139]:

```
logdata = np.log(5+dailySum)
logdata = logdata[~np.isneginf(logdata)]
estimated_mu, estimated_sigma = stats.norm.fit(logdata)
xmin = logdata.min()
xmax = logdata.max()

plt.hist(logdata, bins=30, normed=True, color='c', alpha=0.5, range=(0, xmax))
x = np.linspace(0, xmax, 100)
pdf = stats.norm.pdf(x, loc=estimated_mu, scale=estimated_sigma)
plt.plot(x, pdf, 'k')
plt.xlabel("log(5+daily consumption)")
plt.ylabel("Probality density");
```



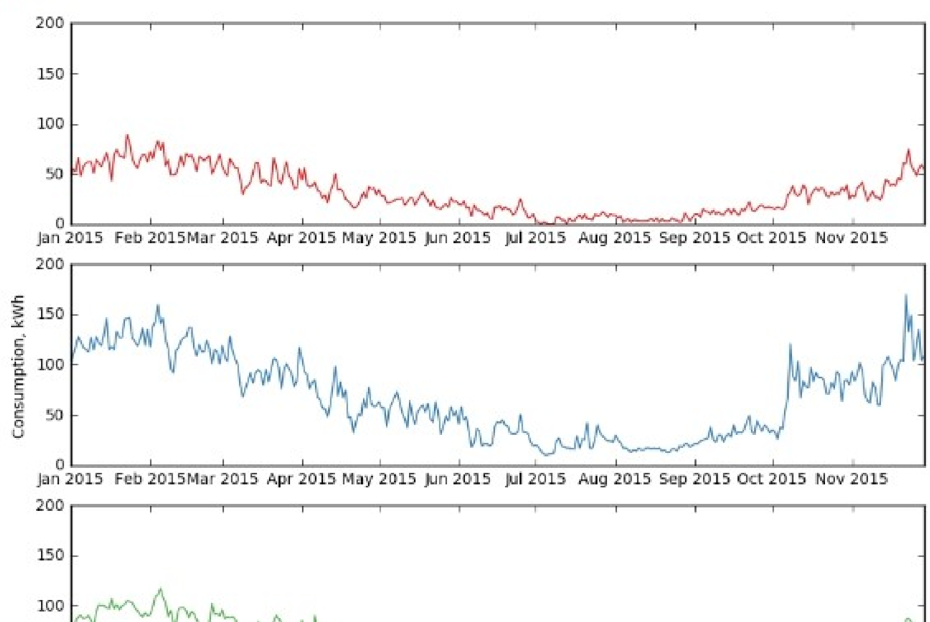
In [146]:

```
df = pd.read_csv('ava_dailydata_2015.csv', header=0, sep='|')
```

In [175]:

```
fig, axs = plt.subplots(nrows=3, ncols=1, figsize=(10, 8))
nos = [100015, 100020, 100166]
colors = cm.Set1(np.arange(8)/8.0)

for i in range(3):
    ts = df[df.installnr==nos[i]]
    ts.index = pd.to_datetime(ts['readdate'])
    axs[i].plot(ts.index, ts.reading, color=colors[i])
    axs[i].set_ylim([0, 200])
#axs[2].set_xlabel('Month')
axs[1].set_ylabel('Consumption, kWh')
plt.plot();
```

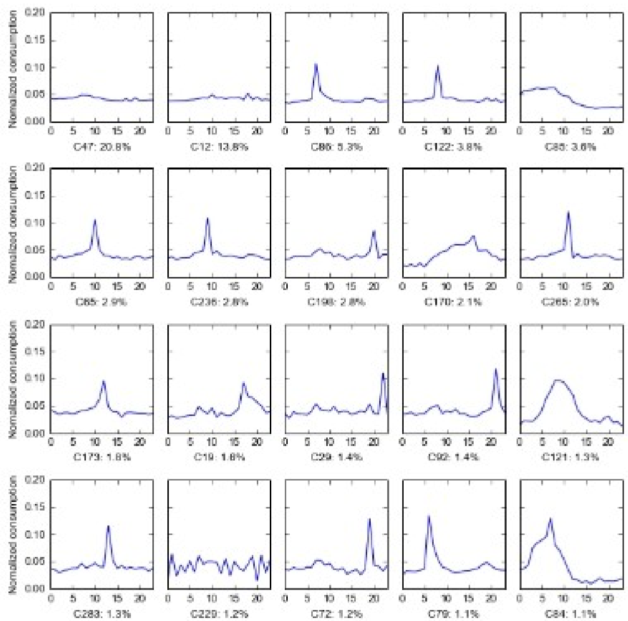


3. Clustering

We adopt the two-stage clustering method proposed in [1], first by adaptive clustering following the merging small clusters.

In [ ]:
```
km = adaptive_clustering(sampleX, 10, 1000, 0.3)
labels, centroids = hierarchical_clustering(np.copy(km.labels_), np.copy(km.cluster_centers_))
```

In [9]:
```
plot_centroids(labels, centroids, 4, 5, 0.2)
```



In [8]:
```
plot_points(sampleX, labels, centroids, 4, 5, 0.2)
```