In [56]:
```
import pandas
import json
import os
import datetime
```

In [57]:
```
#the following is the path to your client json files
data_path="../../data/json_archive"
has_json_ext = lambda x: True if os.path.splitext(x)[-1] == ".json" else False
data_list=[]
for path, deeper_dirs, filenames in os.walk(data_path):
    for filename in filter(has_json_ext, filenames):
        f = open(os.path.join(path, filename), 'r')
        json_data = json.load(f)
        f.close()
        data_list.append(json_data)
df = pandas.DataFrame(data_list)
```

High level stats

In [58]:
```
uniq_devs = df["bluetoothAddress"].nunique()
total_entries = len(df)
print "%-25s %-25s" % ("Uniq Devs:", uniq_devs)
print "%-25s %-25s" % ("Total Sightings:", total_entries)
```

```
Uniq Devs:                2489
Total Sightings:          9362
```

Epoc to datetime

In [60]:
```
epoc_to_datetime = lambda x:datetime.datetime.fromtimestamp(float(x))
df["timestamp"] = df.timestamp.apply(epoc_to_datetime)
```

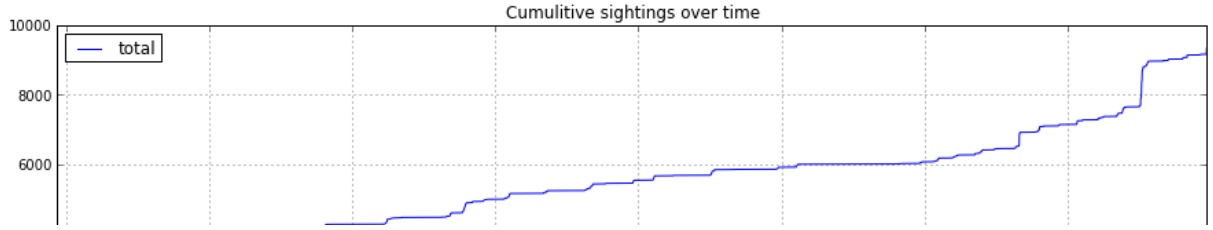
In [61]:
```
df.head()
```

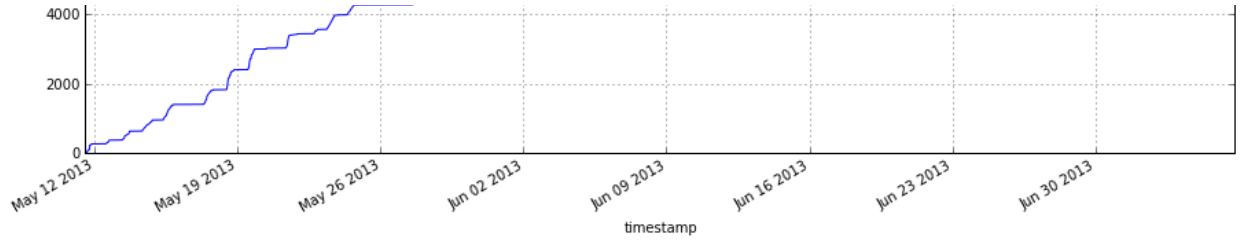
Out[61]:
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5 entries, 0 to 4
Data columns (total 10 columns):
bluetoothAddress    5  non-null values
bluetoothName       5  non-null values
clientType          0  non-null values
clientVersion       0  non-null values
deviceMajor         0  non-null values
deviceMinor         0  non-null values
latatude            5  non-null values
longitude           5  non-null values
timestamp           5  non-null values
type                0  non-null values
dtypes: datetime64[ns](1), float64(3), object(6)
```

Device discovery over time

In [62]:
```
all_devs = df[["timestamp"]]
all_devs.sort("timestamp")
all_devs = all_devs.set_index("timestamp")
all_devs["total"] = 1
all_devs["total"] = all_devs.total.cumsum()
fig = all_devs.plot(figsize=(15,5))
fig.set_title("Cumulitive sightings over time")
```

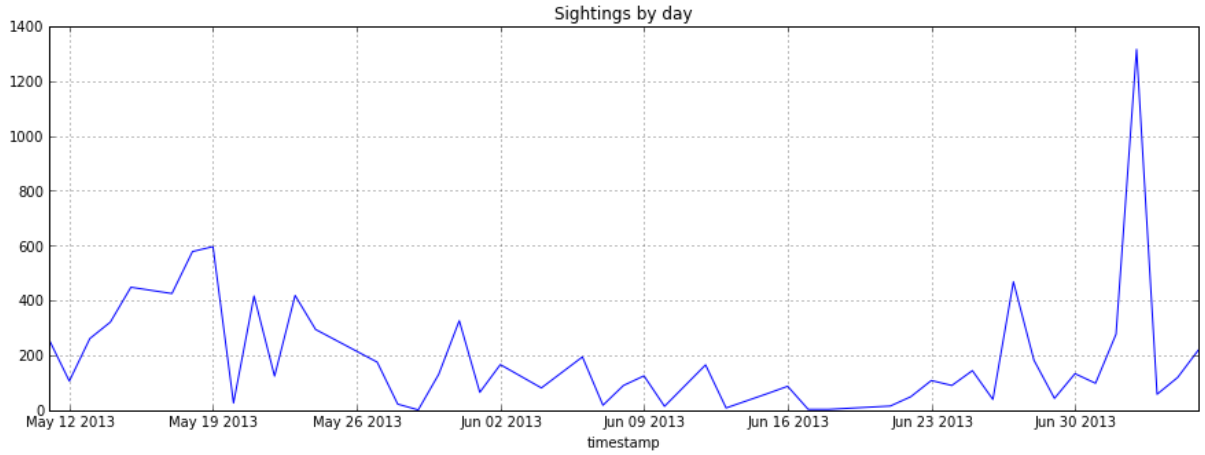
Out[62]: <matplotlib.text.Text at 0x1087b4a50>





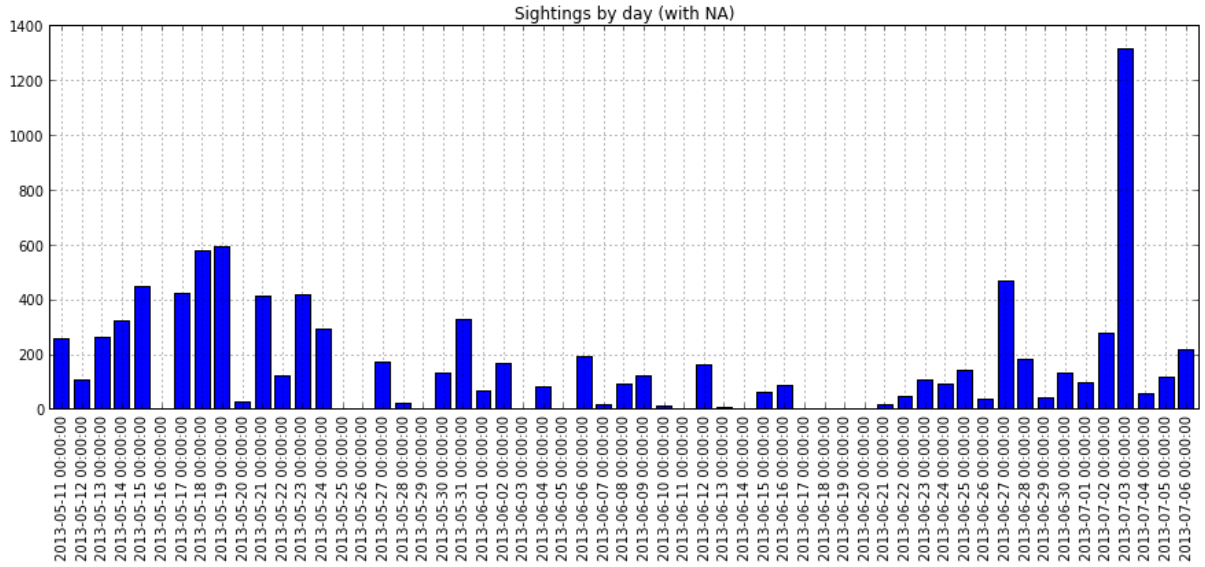
In [63]:
```
all_devs = df[["timestamp"]]
all_devs["timestamp"] = df.timestamp.apply(lambda x: x.date())
grouped = all_devs.groupby("timestamp")
all_by_day = grouped.size()
fig = all_by_day.plot(figsize=(15,5))
fig.set_title("Sightings by day")
```

Out[63]: <matplotlib.text.Text at 0x1097705d0>



In [64]:
```
all_devs = df["timestamp"]
ts = pandas.Series(0, all_devs)
all_by_day += ts.resample('D')
fig = all_by_day.plot(kind="bar", figsize=(15,5))
fig.set_title("Sightings by day (with NA)")
```

Out[64]: <matplotlib.text.Text at 0x1097441d0>

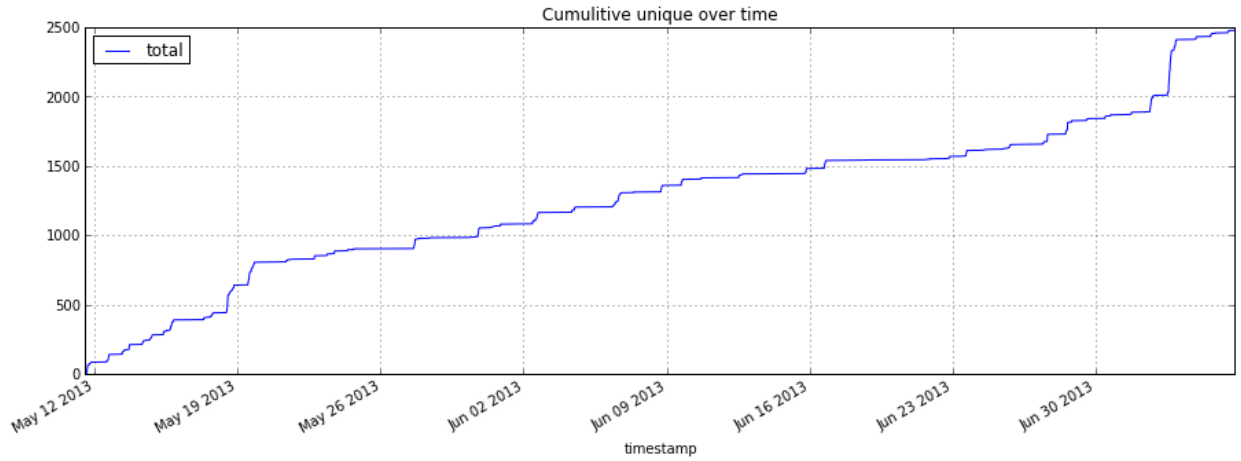


Unique discovery over time

In [65]:
```
uniq_devs = df.groupby("bluetoothAddress")
uniq_devs = uniq_devs.timestamp.min()
uniq_devs = uniq_devs.reset_index()
uniq_devs = uniq_devs[["timestamp"]]
```

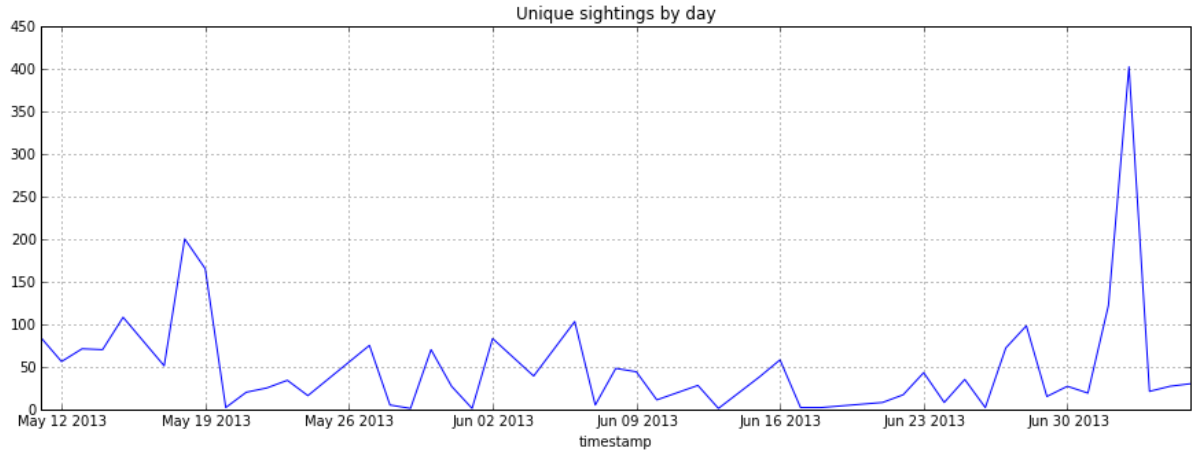
```
uniq_devs.sort("timestamp")
uniq_devs = uniq_devs.set_index("timestamp")
uniq_devs["total"] = 1
uniq_devs["total"] = uniq_devs.total.cumsum()
fig = uniq_devs.plot(figsize=(15,5))
fig.set_title("Cumulitive unique over time")
```

Out[65]: <matplotlib.text.Text at 0x1098a7dd0>



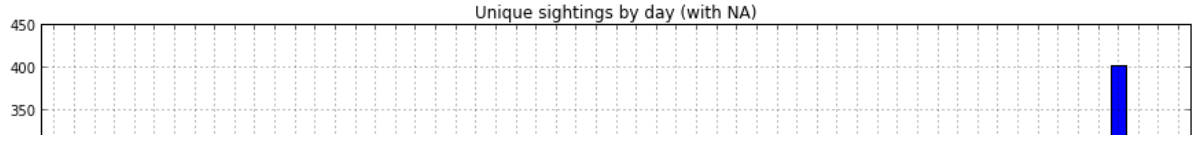
In [66]:
```
uniq_devs = df.groupby("bluetoothAddress")
uniq_devs = uniq_devs.timestamp.min()
uniq_devs = uniq_devs.reset_index()
uniq_devs = uniq_devs[["timestamp"]]
uniq_devs["timestamp"] = uniq_devs.timestamp.apply(lambda x: x.date())
grouped = uniq_devs.groupby("timestamp")
uniq_by_day = grouped.size()
fig = uniq_by_day.plot(figsize=(15,5))
fig.set_title("Unique sightings by day")
```

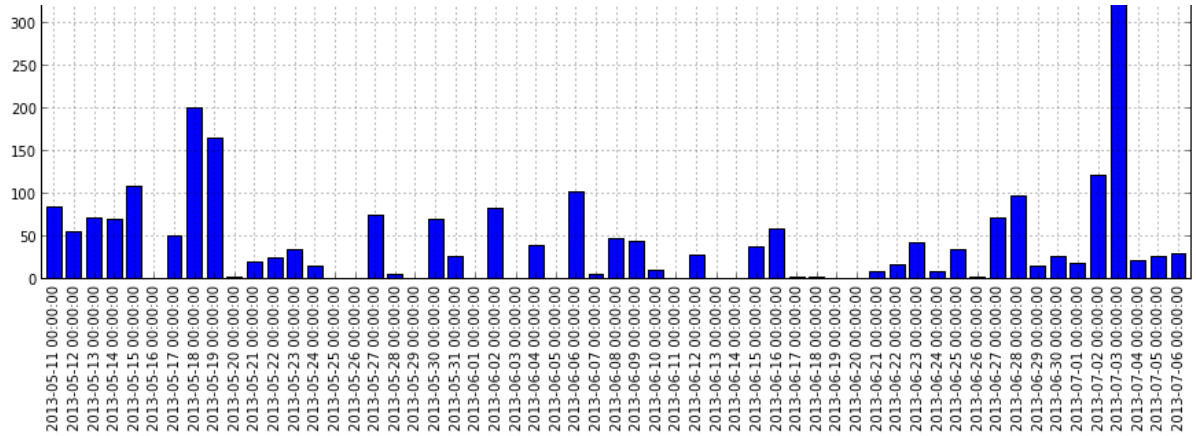
Out[66]: <matplotlib.text.Text at 0x109a486d0>



In [67]:
```
uniq_devs = df.groupby("bluetoothAddress")
uniq_devs = uniq_devs.timestamp.min()
uniq_devs = uniq_devs.reset_index()
uniq_devs = uniq_devs[["timestamp"]]
ts = pandas.Series(0, uniq_devs["timestamp"])
ts = ts.resample('D')
uniq_devs["timestamp"] = uniq_devs.timestamp.apply(lambda x: x.date())
grouped = uniq_devs.groupby("timestamp")
uniq_by_day = grouped.size()
uniq_by_day += ts
fig = uniq_by_day.plot(figsize=(15,5), kind='bar')
fig.set_title("Unique sightings by day (with NA)")
```

Out[67]: <matplotlib.text.Text at 0x109d23f10>





Device sighting prev

In [68]:
```
prev = df.groupby("bluetoothAddress")
prev = prev.size()
prev = prev.reset_index()
prev.columns = ["bluetoothAddress", "prev"]
df_w_prev = pandas.merge(df, prev, on="bluetoothAddress")
df_w_prev.head()
```

Out[68]:
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5 entries, 0 to 4
Data columns (total 11 columns):
bluetoothAddress    5  non-null values
bluetoothName       5  non-null values
clientType          0  non-null values
clientVersion       0  non-null values
deviceMajor         0  non-null values
deviceMinor         0  non-null values
latatude            5  non-null values
longitude           5  non-null values
timestamp           5  non-null values
type                0  non-null values
prev                5  non-null values
dtypes: datetime64[ns](1), float64(3), int64(1), object(6)
```

Device movement

In [81]:
```
import math
def distance(origin, destination):
    lat1, lon1 = origin
    lat2, lon2 = destination
    radius = 6371 # km
    dlat = math.radians(lat2-lat1)
    dlon = math.radians(lon2-lon1)
    a = math.sin(dlat/2) * math.sin(dlat/2) + math.cos(math.radians(lat1)) \
        * math.cos(math.radians(lat2)) * math.sin(dlon/2) * math.sin(dlon/2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = radius * c
    return d
```

In [82]:
```
#TOP MOVERS...
multi_sightings = df_w_prev[df_w_prev["prev"] > 1]
multi_sightings["longitude"] = multi_sightings.longitude.apply(lambda x: abs(float(x)))
multi_sightings["latatude"] = multi_sightings.latatude.apply(lambda x: abs(float(x)))
ms_grouped = multi_sightings.groupby("bluetoothAddress")
movement = ms_grouped.agg({"latatude":lambda x: x.max() - x.min(),
    "longitude":lambda x: x.max() - x.min(),
    "bluetoothName":lambda x: set(i for i in x),
    "prev":lambda x: x.max()})
movement["moved"] = movement["latatude"] + movement["longitude"]
movement.sort("moved", ascending=False).head()
```

Out[82]:
```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, 00:05:4F:7A:9B:59 to 40:5F:BE:B2:40:F6
Data columns (total 5 columns):
```

```
latatude         5  non-null values
prev             5  non-null values
bluetoothName    5  non-null values
longitude        5  non-null values
moved            5  non-null values
dtypes: float64(3), int64(1), object(1)
```

In [83]:
```
#Max Distance Traveled (also add in origin, dest lat/long in result)
multi_sightings = df_w_prev[df_w_prev["prev"] > 1]
multi_sightings["longitude"] = multi_sightings.longitude.apply(lambda x: float(x))
multi_sightings["latatude"] = multi_sightings.latatude.apply(lambda x: float(x))
ms_grouped = multi_sightings.groupby("bluetoothAddress")
traveled_data = []
for device, group in ms_grouped:
    max_dist = 0
    for i in range(len(group)):
        origin = (group.irow(i)["latatude"], group.irow(i)["longitude"])
        for j in range(len(group)):
            destination = (group.irow(j)["latatude"], group.irow(j)["longitude"])
            tmp = distance(origin, destination)
            if tmp > max_dist:
                max_dist = tmp
    traveled_data.append({"bluetoothAddress":device, "km_traveled":max_dist})
```

In [ ]:
```
df2 = pandas.DataFrame(traveled_data)
df2.sort("km_traveled", ascending=False).head()
```

UAP stats

most common uap

In [73]:
```
uap_df = pandas.DataFrame(df["bluetoothAddress"].drop_duplicates())
get_uap = lambda x: x.split(':')[2]
uap_df["UAP"] = uap_df["bluetoothAddress"].apply(get_uap)
uap_stats = uap_df.groupby("UAP").size()
uap_stats = uap_stats.reset_index()
uap_stats.columns = ["UAP","count"]
uap_stats.sort("count", ascending=False).head()
```

Out[73]:

	UAP	count
73	4F	192
226	FC	85
162	B2	72
117	7E	62
145	A0	61

NAP stats

most common nap

In [74]:
```
nap_df = pandas.DataFrame(df["bluetoothAddress"].drop_duplicates())
get_uap = lambda x: x.split(':')[2]
get_nap = lambda x: ':'.join(x.split(':')[0:2])
nap_df["NAP"] = uap_df["bluetoothAddress"].apply(get_nap)
nap_df["UAP"] = uap_df["bluetoothAddress"].apply(get_uap)

nap_stats = nap_df.groupby("NAP").size()
nap_stats = nap_stats.reset_index()
nap_stats.columns = ["NAP","count"]
nap_stats.sort("count", ascending=False).head()
```

Out[74]:

	NAP	count
5	00:05	182
32	00:22	97
71	10:C6	85
36	00:26	80

29	00:1E	75

most common nap by uap

In [75]:
```
nap_df = pandas.DataFrame(df["bluetoothAddress"].drop_duplicates())
get_uap = lambda x: x.split(':')[2]
get_nap = lambda x: ':'.join(x.split(':')[0:2])
nap_df["NAP"] = uap_df["bluetoothAddress"].apply(get_nap)
nap_df["UAP"] = uap_df["bluetoothAddress"].apply(get_uap)
nap_df.groupby(["UAP","NAP"]).size()
```

Out[75]:
```
UAP  NAP
00   00:17    1
     00:25    8
     A8:06    2
01   00:24    2
     2C:44    4
02   6C:9B    2
03   94:51    1
     9C:DF    2
04   00:07    2
     00:13    1
     70:1A    3
     78:CA    1
     A0:4E    2
05   70:81    4
06   00:17    1
...
F8   8C:71     2
F9   38:59     1
     C0:38     5
     E4:7C     2
FA   88:9F     2
FB   00:21     2
     00:25     1
FC   10:C6    83
     18:9E     2
FD   00:1D     1
     14:89     2
FE   00:16     1
     00:21     3
     3C:8B     6
FF   00:26     1
Length: 590, dtype: int64
```

Vendor stats

Anomalies

same address diff names

In [76]:
```
#do all bluetooth APIs return generic names on bad reception?
name_anom = df.groupby("bluetoothAddress")
exclude = ["Handsfree", "Misc", "Computer", "Mobile Phone", "Laptop", "Headset", "PDA", "Peripheral", "Mouse", "Keyboard"]
set_agg = lambda x: set(i for i in x if i not in exclude)
name_anom = name_anom.bluetoothName.apply(set_agg)
name_anom = name_anom.reset_index()
name_anom.columns = ["bluetoothAddress", "names"]
name_anom["name_count"] = name_anom.names.apply(len)
name_anom = name_anom[name_anom["name_count"] > 1]
name_anom.sort("name_count", ascending=False)
```

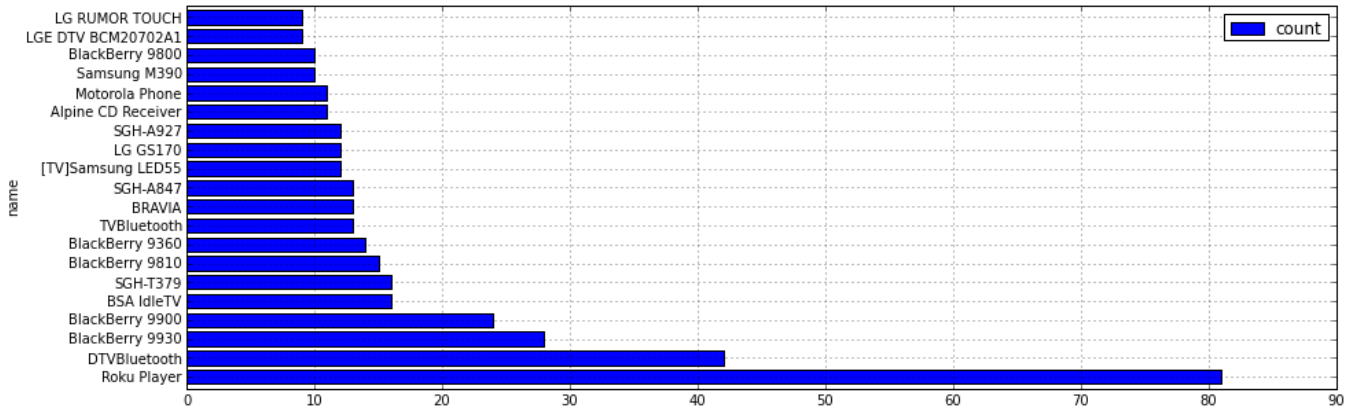
Out[76]:
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 2407 to 16
Data columns (total 3 columns):
bluetoothAddress    16  non-null values
names               16  non-null values
name_count          16  non-null values
dtypes: int64(1), object(2)
```

Generic stats

most common names

In [77]:
```
name_anom = df.groupby("bluetoothAddress")
set_agg = lambda x: set(i for i in x)
name_anom = name_anom.bluetoothName.apply(set_agg)
name_anom = name_anom.reset_index()
name_anom.columns = ["bluetoothAddress", "names"]
name_dict = {}
for i in name_anom["names"]:
    for j in i:
        if j in name_dict:
            name_dict[j] += 1
        else:
            name_dict[j] = 1
formated_dict = {"name":[], "count":[]}
excude = ["Handsfree", "Misc", "Computer", "Mobile Phone", "Laptop", "Headset", "PDA", "Peripheral"]
for k,v in name_dict.iteritems():
    if k in excude:
        continue
    formated_dict["name"].append(k)
    formated_dict["count"].append(v)
name_df = pandas.DataFrame(formated_dict)
name_df = name_df.set_index("name")
name_df.sort("count", ascending=False)[:20].plot(kind='barh', figsize=(15,5))
```

Out[77]: <matplotlib.axes.AxesSubplot at 0x109faf9d0>



Map example

In [ ]:
```
# Get latest sighting for each device
grouped = df.groupby("bluetoothAddress")
latest_times = grouped.timestamp.max()
latest_times = latest_times.reset_index()
latest_times.columns = ["bluetoothAddress","timestamp"]
latest_times["latest"]= True
latest_times = latest_times.set_index(["bluetoothAddress", "timestamp"])

df["timestamp"] = df.timestamp.apply(pandas.tslib.Timestamp)
df2 = df.set_index(["bluetoothAddress", "timestamp"])
latest_times = pandas.merge(latest_times, df2, left_index=True, right_index=True)
latest_times[:20]
```