Bandicoot notebook

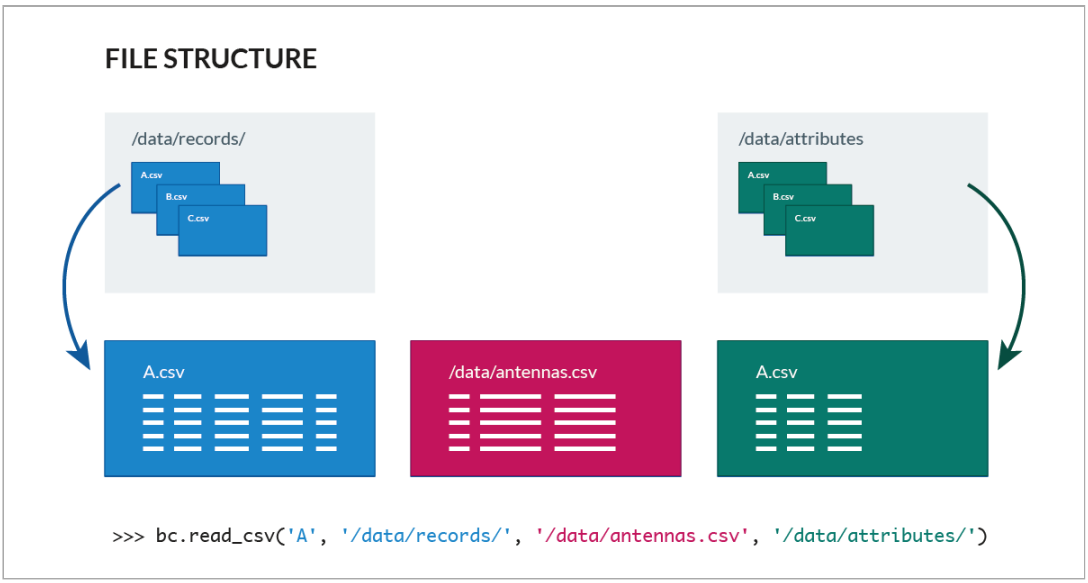
bandicoot is an open-source python toolbox to analyze mobile phone metadata.

For more information, see: http://bandicoot.mit.edu/ (http://bandicoot.mit.edu/) You can download this notebook on Github from our repository at github.com/yvesalexandre/bandicoot (https://github.com/yvesalexandre/bandicoot/blob/master/demo/demo.ipynb).

Try bandicoot on your phone!

If you want to try bandicoot with your own data, download our Android app at bandicoot.mit.edu/android (http://bandicoot.mit.edu/android)

Input files



In [1]:

```
# Records for the user 'ego'
!head -n 5 data/ego.csv
```

```
interaction,direction,correspondent_id,datetime,call_duration,antenna_id
text,in,A,2014-03-02 07:13:30,,1
text,in,E,2014-03-02 07:53:30,,1
text,in,E,2014-03-02 08:22:30,,2
text,out,D,2014-03-02 08:34:30,,3
```

In [2]:

```
# GPS locations of cell towers
!head -n 5 data/antennas.csv
```

```
antenna_id,latitude,longitude
1,42.366944,-71.083611
2,42.386722,-71.138778
3,42.3604,-71.087374
4,42.353917,-71.105
```

Loading a user

In [3]:

```
import bandicoot as bc

U = bc.read_csv('ego', 'data/', 'data/antennas.csv')
```

```
[x] 314 records from 2014-03-02 07:13:30 to 2014-04-14 12:04:37
[x] 7 contacts
[ ] No attribute stored
[x] 27 antennas
[ ] No recharges
[x] Has home
[x] Has texts
[x] Has calls
[ ] No network
```

Visualization

Export and serve an interactive visualization using:

```
bc.visualization.run(U)
```

or export only using:

```
bc.visualization.export(U, 'my-viz-path')
```

In [4]:

```
import os
viz_path = os.path.dirname(os.path.realpath(__name__)) + '/viz'

bc.visualization.export(U, viz_path)
```

```
Successfully exported the visualization to /Volumes/Data/projects/bandicoo
t/demo/viz
```

Out[4]:

```
'/Volumes/Data/projects/bandicoot/demo/viz'
```

In [5]:

```
from IPython.display import IFrame
IFrame("/files/viz/index.html", "100%", 700)
```

Out[5]:

Top 3 users

1. E (55)
2. B (53)
3. C (40)

Ego network

Indicators / number of interactions

Individual and spatial indicators

Using bandicoot, compute aggregated indicators from `bc.individual` and `bc.spatial`:

In [6]:

```
bc.individual.percent_initiated_conversations(U)
```

Out[6]:


```
{
    "allweek": {
        "allday": {
            "callandtext": {
                "mean": 0.3244815064488733,
                "std": 0.09659866096759165
            }
        }
    }
}
```


In [7]:

```
bc.spatial.number_of_antennas(U)
```

Out[7]:


```
{
    "allweek": {
        "allday": {
            "mean": 5.375,
            "std": 1.8666480653835098
        }
    }
}
```


In [8]:

```
bc.spatial.radius_of_gyration(U)
```

Out[8]:


```
{
    "allweek": {
        "allday": {
            "mean": 1.4503807789208683,
            "std": 0.8575480642906887
        }
    }
}
```

Let's play with indicators

The signature of the `active_days` indicators is:

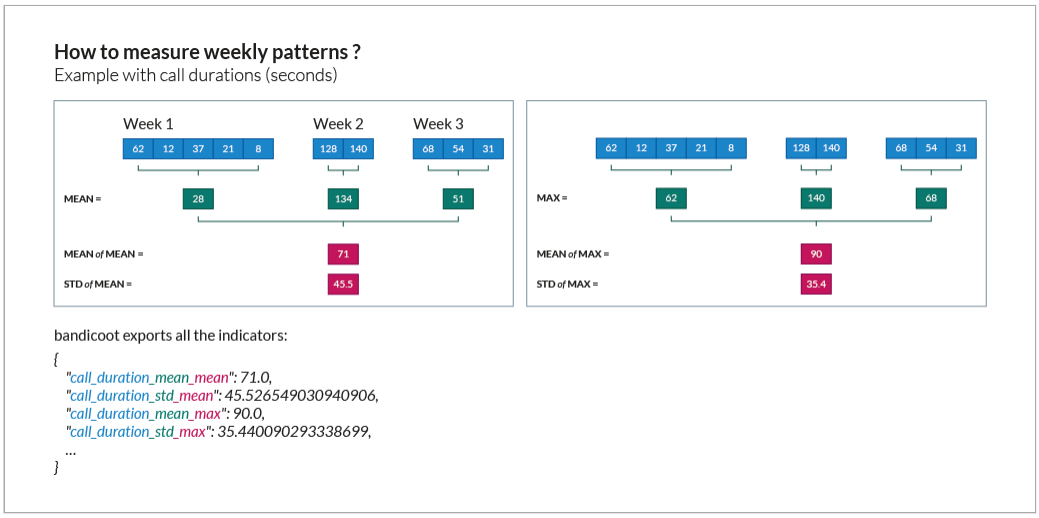
```
bc.individual.active_days(user, groupby='week', interaction='callandtext', summa
ry='default', split_week=False, split_day=False, filter_empty=True, datatype=Non
e)
```

What does that mean?

The *'groupby'* keyword

Weekly aggregation

By default, _bandicoot_ computes the indicators on a weekly basis and returns the average (mean) over all the weeks available and its standard deviation (std) in a nested dictionary.



In [9]:

```
bc.individual.active_days(U)
```

Out[9]:


```
{
    "allweek": {
        "allday": {
            "callandtext": {
                "mean": 5.5,
                "std": 2.598076211353316
            }
        }
    }
}
```

The `groupby` keyword controls the aggregation:

- `groupby='week'` to divide by week (by default),
- `groupby='month'` to divide by month,
- `groupby=None` to aggregate all values.

In [10]:

```
bc.individual.active_days(U, groupby='week')
```

Out[10]:


```
{
    "allweek": {
        "allday": {
            "callandtext": {
                "mean": 5.5,
                "std": 2.598076211353316
            }
        }
    }
}
```


In [11]:

```
bc.individual.active_days(U, groupby='month')
```

Out[11]:


```
{
    "allweek": {
        "allday": {
            "callandtext": {
                "mean": 22.0,
                "std": 8.0
            }
        }
    }
}
```


In [12]:

```
bc.individual.active_days(U, groupby=None)
```

Out[12]:


```
{
    "allweek": {
        "allday": {
            "callandtext": 44
        }
    }
}
```

The *'summary'* keyword

Some indicators such as *active_days* returns one number. Others, such as *duration_of_calls* returns a distribution.

The summary keyword can take three values:

- `summary='default'` to return mean and standard deviation,
- `summary='extended'` for the second type of indicators, to return mean, sem, median, skewness and std of the distribution,
- `summary=None` to return the full distribution.

In [13]:

```
bc.individual.call_duration(U)
```

Out[13]:


```
{
    "allweek": {
        "allday": {
            "call": {
                "mean": {
                    "mean": 3776.7093501036775,
                    "std": 1404.827412706482
                },
                "std": {
                    "mean": 1633.3931770157765,
                    "std": 689.2035500056488
                }
            }
        }
    }
}
```

In [14]:

```
bc.individual.call_duration(U, summary='extended')
```

Out[14]:


```
{
    "allweek": {
        "allday": {
            "call": {
                "mean": {
                    "mean": 3776.7093501036775,
                    "std": 1404.827412706482
                },
                "std": {
                    "mean": 1633.3931770157765,
                    "std": 689.2035500056488
                },
                "median": {
                    "mean": 3714.714285714286,
                    "std": 1532.9148671064283
                },
                "skewness": {
                    "mean": 0.12925073170191398,
                    "std": 0.48628300355189896
                },
                "kurtosis": {
                    "mean": 1.8063876957023484,
                    "std": 0.8998073161683097
                },
                "min": {
                    "mean": 1330.857142857143,
                    "std": 2200.2680634459994
                },
                "max": {
                    "mean": 6468.857142857143,
                    "std": 519.0475972040188
                }
            }
        }
    }
}
```

In [15]:

```
bc.individual.call_duration(U, summary=None)
```

Out[15]:


```
{
    "allweek": {
        "allday": {
            "call": [
                [
                    374,
                    1086,
                    1099,
                    1330,
                    2456,
                    3404,
                    4472,
                    5359,
                    5413,
                    6233
                ],
                [
                    594,
                    1927,
                    2072,
                    2258,
                    2854,
                    3286,
                    3552,
                    4202,
                    4689,
                    5142,
                    5689,
                    5752,
                    6429,
                    6891,
                    7082,
                    7123
                ],
                [
                    403,
                    539,
                    2109,
                    2726,
                    2871,
                    3609,
                    3782,
                    4154,
                    4240,
                    4666,
                    5658,
                    6392,
                    6541,
                    6674
                ],
                [
                    154,
                    267,
                    706,
                    1273,
                    1890,
                    3435,
                    3454,
                    3503,
```

```
                    3531,
                    3668,
                    4832,
                    4877,
                    4929,
                    5096,
                    5184,
                    6539,
                    7038
                ],
                [
                    140,
                    231,
                    620,
                    1853,
                    1937,
                    5728
                ],
                [
                    969,
                    1309,
                    1355,
                    1999,
                    2626,
                    3210,
                    4002,
                    4146,
                    4227,
                    4451,
                    5804
                ],
                [
                    6682
                ]
            ]
        }
    }
}
```


Splitting days and weeks

- **`split_week`** divide records by 'all week', 'weekday', and 'weekend'.
- **`split_day`** divide records by 'all day', 'day', and 'night'.

In [16]:

```
bc.individual.active_days(U, split_week=True, split_day=True)
```

Out[16]:


```
{
    "allweek": {
        "allday": {
            "callandtext": {
                "mean": 5.5,
                "std": 2.598076211353316
            }
        },
        "day": {
            "callandtext": {
                "mean": 5.5,
                "std": 2.598076211353316
            }
        },
        "night": {
            "callandtext": {
                "mean": 5.375,
                "std": 2.54644359843292
            }
        }
    },
    "weekday": {
        "allday": {
            "callandtext": {
                "mean": 4.428571428571429,
                "std": 1.3997084244475304
            }
        },
        "day": {
            "callandtext": {
                "mean": 4.428571428571429,
                "std": 1.3997084244475304
            }
        },
        "night": {
            "callandtext": {
                "mean": 4.428571428571429,
                "std": 1.3997084244475304
            }
        }
    },
    "weekend": {
        "allday": {
            "callandtext": {
                "mean": 1.8571428571428572,
                "std": 0.34992710611188266
            }
        },
        "day": {
            "callandtext": {
                "mean": 1.8571428571428572,
                "std": 0.34992710611188266
            }
        },
        "night": {
            "callandtext": {
                "mean": 1.7142857142857142,
                "std": 0.45175395145262565
            }
```

```
        }
    }
}
```

Exporting indicators

The function `bc.utils.all` computes automatically all indicators for a single user.

You can use the same keywords to group by week/month/all time range, or return extended statistics.

In [17]:

```
features = bc.utils.all(U, groupby=None)
```

In [18]:

```
features
```

Out[18]:


```
{
    "name": "ego",
    "reporting": {
        "antennas_path": "data/antennas.csv",
        "attributes_path": None,
        "recharges_path": None,
        "version": "0.5.0",
        "code_signature": "92baf56749980c1cda5cb4ae7cc533683c311b9c",
        "groupby": None,
        "split_week": false,
        "split_day": false,
        "start_time": "2014-03-02 07:13:30",
        "end_time": "2014-04-14 12:04:37",
        "night_start": "19:00:00",
        "night_end": "07:00:00",
        "weekend": [
            6,
            7
        ],
        "number_of_records": 314,
        "number_of_antennas": 27,
        "number_of_recharges": 0,
        "bins": 1,
        "bins_with_data": 1,
        "bins_without_data": 0,
        "has_call": true,
        "has_text": true,
        "has_home": true,
        "has_recharges": false,
        "has_attributes": false,
        "has_network": false,
        "percent_records_missing_location": 0.0,
        "antennas_missing_locations": 0,
        "percent_outofnetwork_calls": 0,
        "percent_outofnetwork_texts": 0,
        "percent_outofnetwork_contacts": 0,
        "percent_outofnetwork_call_durations": 0,
        "ignored_records": {
            "all": 0,
            "interaction": 0,
            "location": 0,
            "correspondent_id": 0,
            "call_duration": 0,
            "direction": 0,
            "datetime": 0
        }
    },
    "active_days": {
        "allweek": {
            "allday": {
                "callandtext": 44
            }
        }
    },
    "number_of_contacts": {
        "allweek": {
            "allday": {
                "call": 7,
                "text": 7
```

```
            }
        }
    },
    "call_duration": {
        "allweek": {
            "allday": {
                "call": {
                    "mean": 3557.2933333333335,
                    "std": 2067.863501448348
                }
            }
        }
    },
    "percent_nocturnal": {
        "allweek": {
            "allday": {
                "call": 0.4266666666666667,
                "text": 0.4393305439330544
            }
        }
    },
    "percent_initiated_conversations": {
        "allweek": {
            "allday": {
                "callandtext": 0.3080357142857143
            }
        }
    },
    "percent_initiated_interactions": {
        "allweek": {
            "allday": {
                "call": 0.41333333333333333
            }
        }
    },
    "response_delay_text": {
        "allweek": {
            "allday": {
                "callandtext": {
                    "mean": 2310.0,
                    "std": 747.5961476626268
                }
            }
        }
    },
    "response_rate_text": {
        "allweek": {
            "allday": {
                "callandtext": 0.025806451612903226
            }
        }
    },
    "entropy_of_contacts": {
        "allweek": {
            "allday": {
                "call": 1.8626732085630935,
                "text": 1.8462551114653172
            }
        }
    },
    "balance_of_contacts": {
```

```
            "allweek": {
                "allday": {
                    "call": {
                        "mean": 0.05904761904761905,
                        "std": 0.018662778992633737
                    },
                    "text": {
                        "mean": 0.04363419007770473,
                        "std": 0.01828697972597532
                    }
                }
            }
        },
        "interactions_per_contact": {
            "allweek": {
                "allday": {
                    "call": {
                        "mean": 10.714285714285714,
                        "std": 4.025429372458677
                    },
                    "text": {
                        "mean": 34.142857142857146,
                        "std": 15.027865274012724
                    }
                }
            }
        },
        "interevent_time": {
            "allweek": {
                "allday": {
                    "call": {
                        "mean": 49024.86486486487,
                        "std": 49455.92699110296
                    },
                    "text": {
                        "mean": 15683.474789915967,
                        "std": 16816.128561460955
                    }
                }
            }
        },
        "percent_pareto_interactions": {
            "allweek": {
                "allday": {
                    "call": 0.06666666666666667,
                    "text": 0.02092050209205021
                }
            }
        },
        "percent_pareto_durations": {
            "allweek": {
                "allday": {
                    "call": 0.06666666666666667
                }
            }
        },
        "number_of_interactions": {
            "allweek": {
                "allday": {
                    "call": 75,
                    "text": 239
```

```
            }
        }
    },
    "number_of_interaction_in": {
        "allweek": {
            "allday": {
                "call": 44,
                "text": 166
            }
        }
    },
    "number_of_interaction_out": {
        "allweek": {
            "allday": {
                "call": 31,
                "text": 73
            }
        }
    },
    "number_of_antennas": {
        "allweek": {
            "allday": 10
        }
    },
    "entropy_of_antennas": {
        "allweek": {
            "allday": 1.5002835799997023
        }
    },
    "percent_at_home": {
        "allweek": {
            "allday": 0.4738562091503268
        }
    },
    "radius_of_gyration": {
        "allweek": {
            "allday": 1.8065370474407052
        }
    },
    "frequent_antennas": {
        "allweek": {
            "allday": 3
        }
    },
    "churn_rate": {
        "mean": 0.07391866292877013,
        "std": 0.05195576174609364
    }
}
```


Exporting in CSV and JSON

bandicoot supports exports in CSV and JSON format. Both `to_csv` and `to_json` functions require either a single feature dictionnary, or a list of dictionnaries (for multiple users).

In [19]:

```
bc.to_csv(features, 'demo_export_user.csv')
bc.to_json(features, 'demo_export_user.json')
```

```
Successfully exported 1 object(s) to demo_export_user.csv
Successfully exported 1 object(s) to demo_export_user.json
```

In [20]:

```
!head demo_export_user.csv
```

In [21]:

```
!head -n 15 demo_export_user.json
```


```
{
    "ego": {
        "name": "ego",
        "reporting": {
            "antennas_path": "data/antennas.csv",
            "attributes_path": null,
            "recharges_path": null,
            "version": "0.5.0",
            "code_signature": "92baf56749980c1cda5cb4ae7cc533683c311b9c",
            "groupby": null,
            "split_week": false,
            "split_day": false,
            "start_time": "2014-03-02 07:13:30",
            "end_time": "2014-04-14 12:04:37",
            "night_start": "19:00:00",
```


Extending bandicoot

You can easily develop your indicator using the `@grouping` decorator. You only need to write a function taking as input a list of records and returning an integer or a list of integers (for a distribution). The `@grouping` decorator wraps the function and call it for each group of weeks.

In [22]:

```
from bandicoot.helper.group import grouping

@grouping(interaction='call')
def shortest_call(records):
    in_durations = (r.call_duration for r in records)
    return min(in_durations)
```

In [23]:

```
shortest_call(U)
```

Out[23]:


```
{
    "allweek": {
        "allday": {
            "call": {
                "mean": 1330.857142857143,
                "std": 2200.2680634459994
            }
        }
    }
}
```


In [24]:

```
shortest_call(U, split_day=True)
```

Out[24]:


```
{
    "allweek": {
        "allday": {
            "call": {
                "mean": 1330.857142857143,
                "std": 2200.2680634459994
            }
        },
        "day": {
            "call": {
                "mean": 1395.5714285714287,
                "std": 2186.820187078088
            }
        },
        "night": {
            "call": {
                "mean": 1182.1666666666667,
                "std": 908.7290422464895
            }
        }
    }
}
```