Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет к шестому заданию практикума на ЭВМ: Использование Cython для решения задачи кластеризации

Студент 3 курса ВМК (317 группа):
Оспанов А.М.

Москва, 2015

Содержание

1 Введение

Данный отчет написан к шестому заданию практикума на ЭВМ 317 группы. Тема задания: Использование Cython для решения задачи кластеризации. Отчет написан студентом 317 группы – Оспановым Аятом.

В данной работе были реализованы алгоритмы кластеризаций k-Means и k-Medoids на языках Python, Cython и C++. Были проведены исследования по производительности кодов на разных языках и их точности на городах Мира. Также результаты были визуализированы.

2 Основная часть

2.1 Математические выкладки

Для кластеризации городов на сфере, нужно найти расстояние по сфере. Это расстояние высчитывается через следующую формулу:

$$L = R * arccos(sin(\theta_1) * sin(\theta_2) + cos(\theta_1) * cos(\theta_2) * cos(\varphi_1 - \varphi_2))$$

где θ_1, θ_2 - широты городов, φ_1, φ_2 - долготы городов.

Для нахождения центров масс использовался поворот относительно векторного произведения. Сперва точки переводились в трехмерное пространство через следующие формулы:

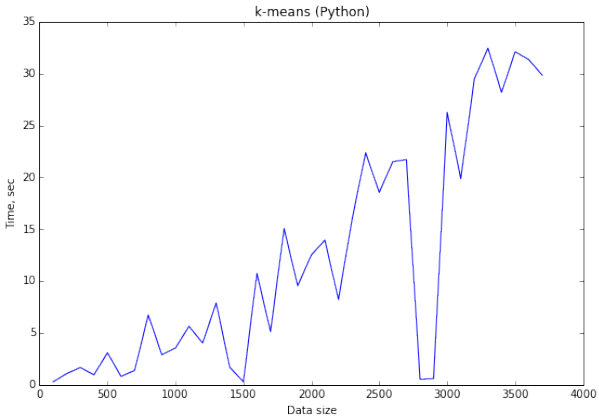
$$x = r * cos(\theta) * cos(\varphi)$$
$$y = r * cos(\theta) * sin(\varphi)$$
$$z = r * sin(\theta)$$

Далее находилось их векторное произведение и нормировалось. Затем относительно этой оси первый вектор поварачивался на угол умноженный на его вес, поделенный на сумму весов двух векторов. Угол высчитывался с помощью скалярного произведения. Далее найденная точка переводилась в сферическую систему.

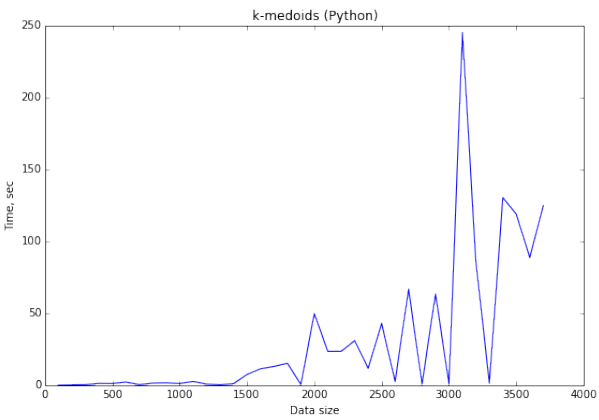
2.2 Замер производительности

Для замера производительности ставилось точное количество итераций, равное 50. Т.е. алгоритм не выходил раньше 50 итераций, даже если сходился. После замеров, получились следующие результаты:

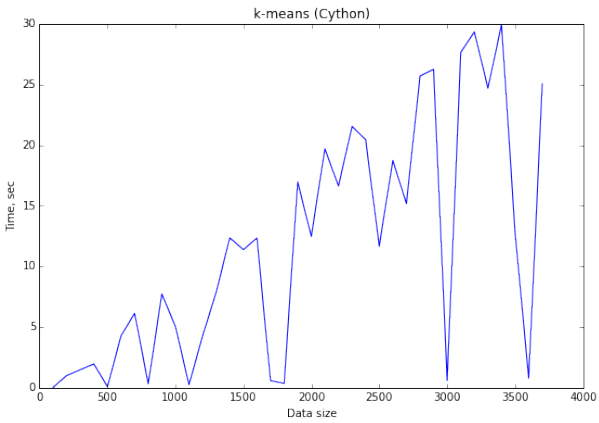
k-Means на Python



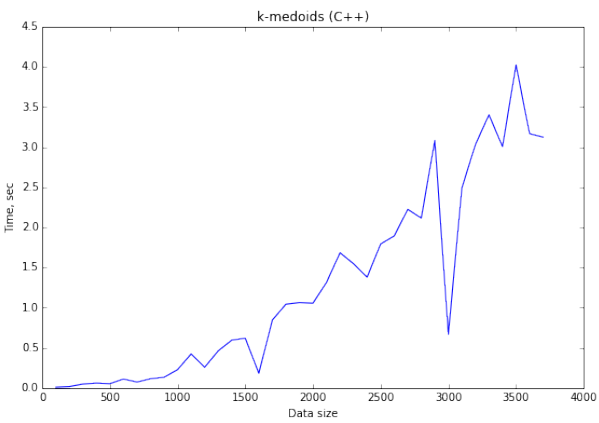
k-Medoids на Python



k-Means на Cython

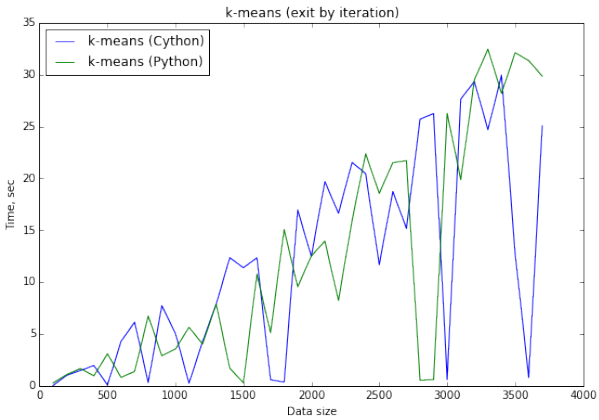


k-Medoids на C++

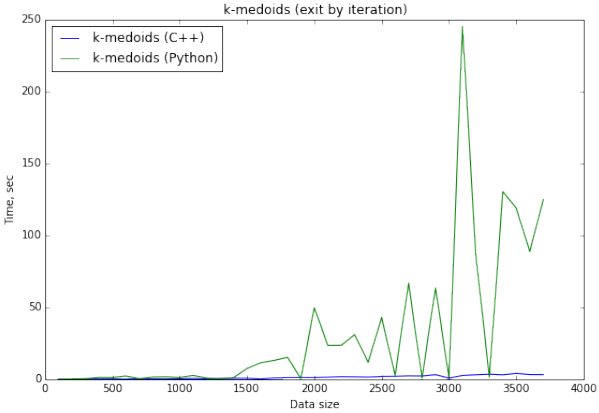


Из таблиц видно, что k-Means работает за время $O(N)$, а k-Medoids за $O(N^2)$, где N - количество данных. Это подтверждается теорией, т.к. k-Means вычисляет расстояние до центров масс, а k-Medoids вычисляет расстояния от каждой точки до каждой точки в кластере.

Сравнение скоростей k-Means



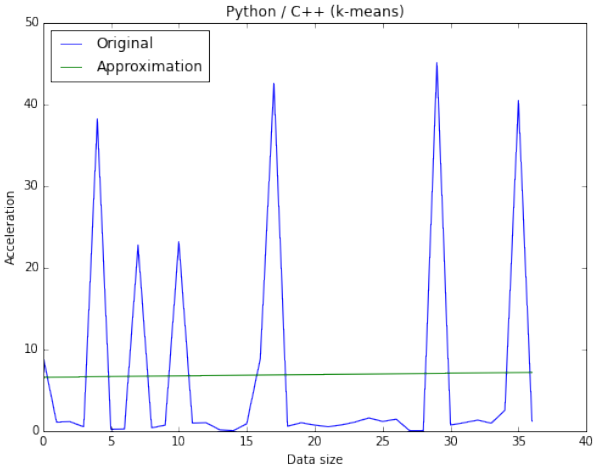
Сравнение скоростей k-Medoids



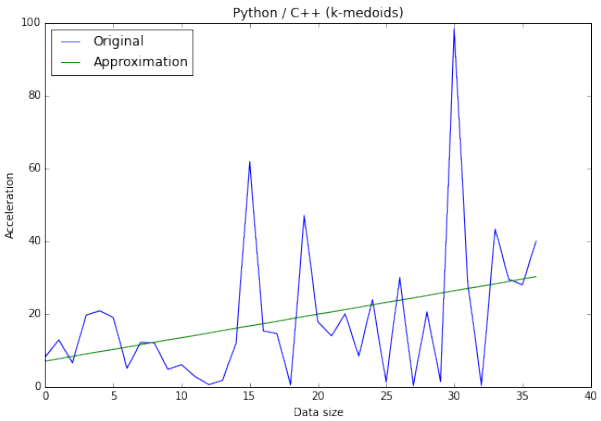
Как видно из таблицы, реализации k-Means на Python и Cython почти одинаковые, но в следующей таблице (по аппроксимации) можно увидеть, что реализация на Cython уско-

ряется по сравнению с реализацией на Python, следовательно для дальнейших тестов выберем именно реализацию на Cython. Реализации k-Medoids на Python и C++ отличаются в несколько десятков раз. Из следующей таблицы можно увидеть, что реализация на C++ ускоряется очень быстро, по сранению с реализацией на Python. Тут без сомнений выбирается реализация на C++.

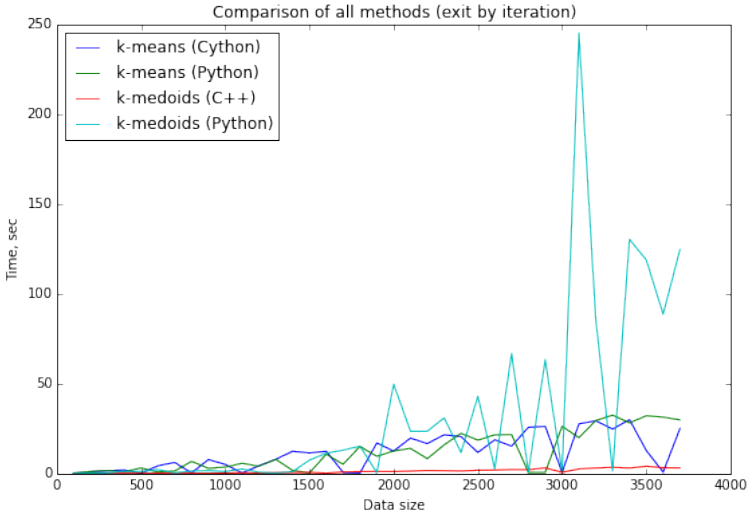
Относительная скорость
Python к Cython в k-Means



Относительная скорость
Python к C++ в k-Medoids



Далее для интереса приведем сранительную таблицу всех методов



2.3 Тестирование алгоритмов

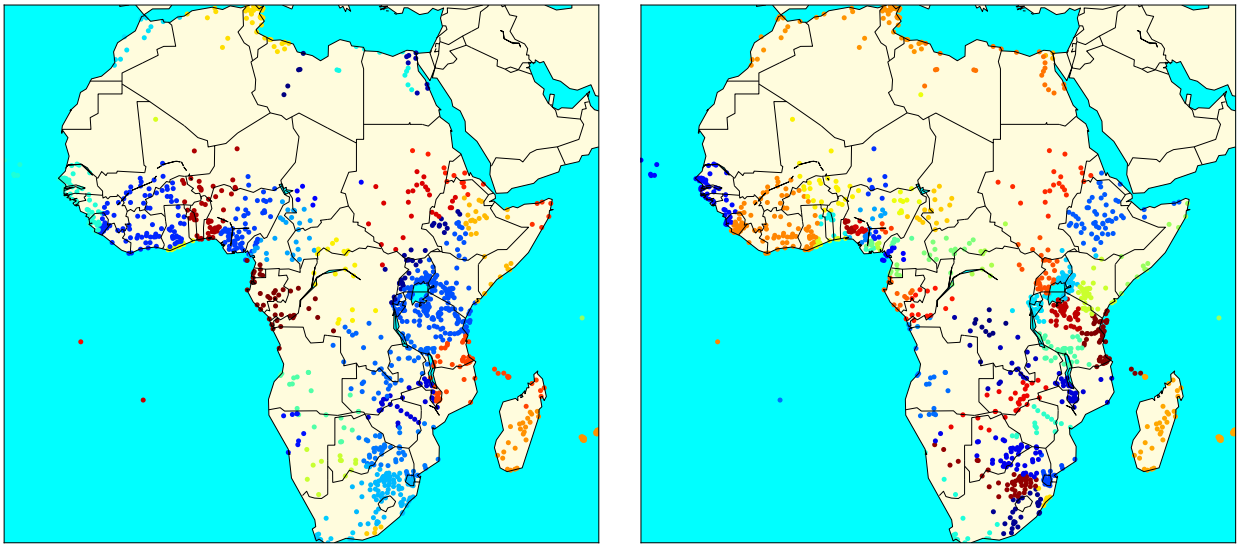
Для дальнейших исследований алгоритм был настроен на выход по схождению. Приведем таблицу точностей для методов во всех регионах:

	Мир	Африка	Азия	Австралия	Европа	С. Америка	Ю. Америка
k-Means	0.4115	0.4055	0.5484	0.5517	0.2607	0.3652	0.4403
k-Medoids	0.4928	0.3947	0.5760	0.2675	0.4062	0.0970	0.3524

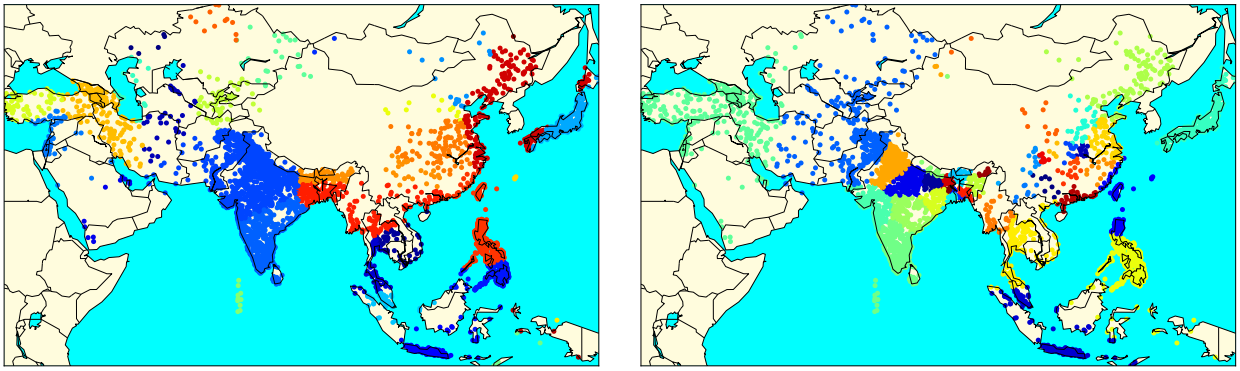
Далее приведем результаты работ алгоритмов (изображения векторные):

k-Means k-Medoids

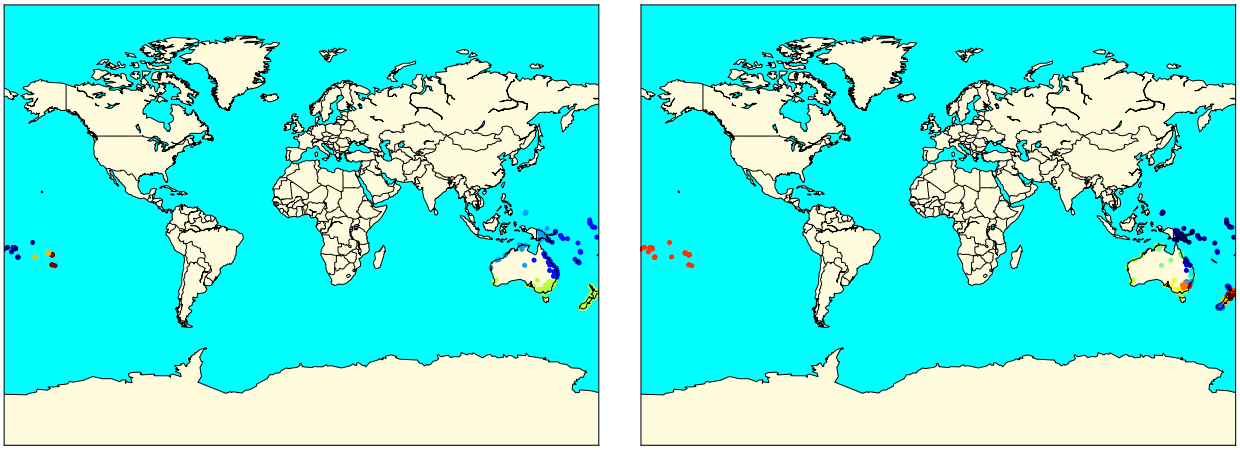
Африка



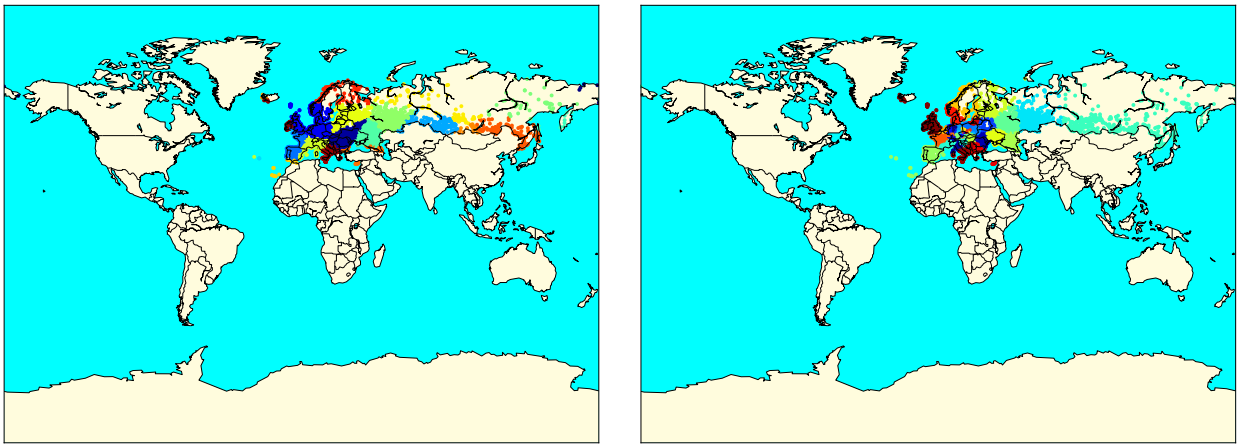
Азия



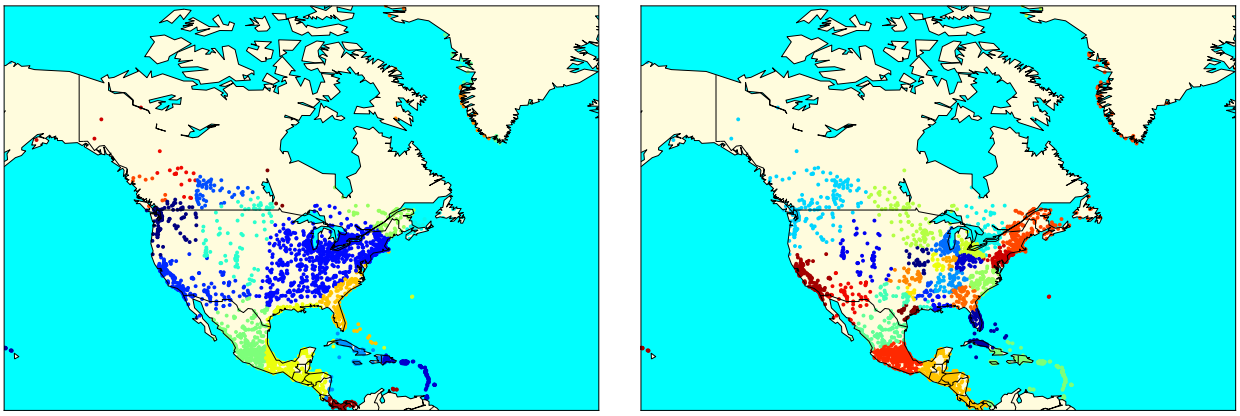
Австралия



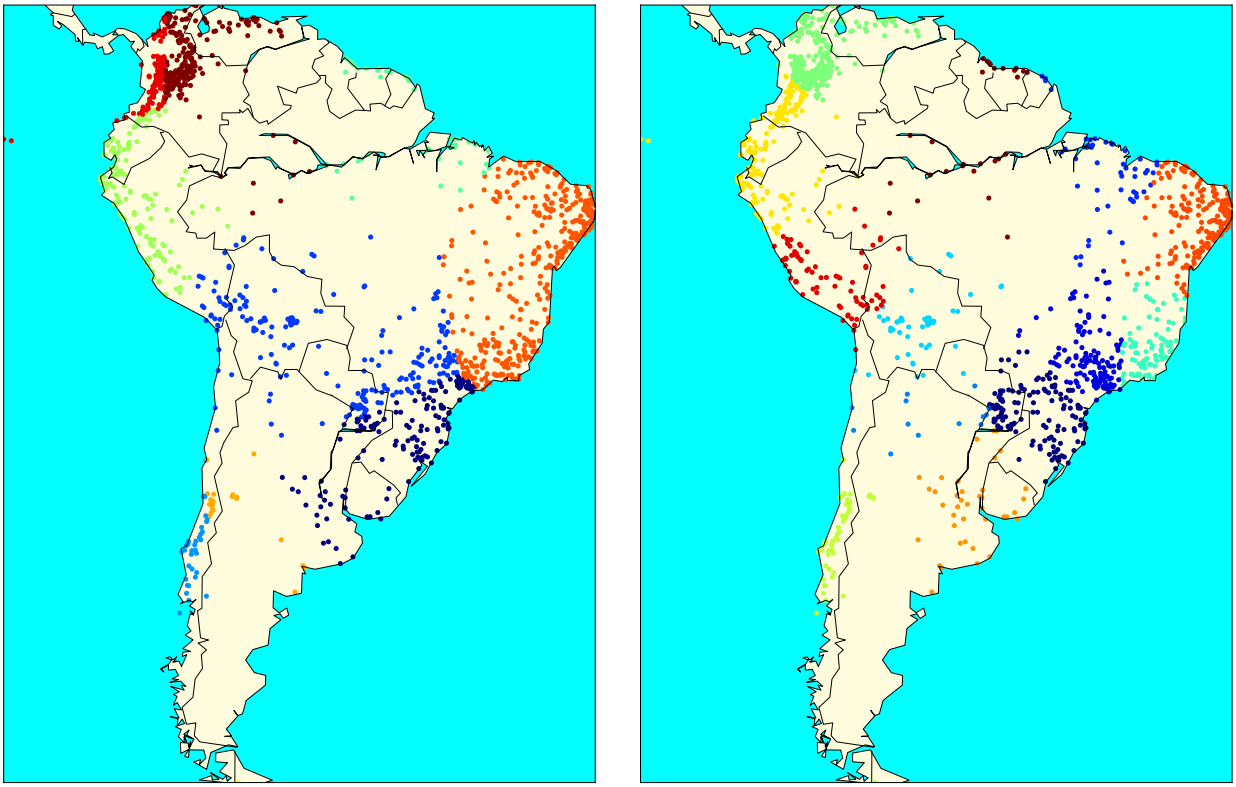
Европа



Северная Америка

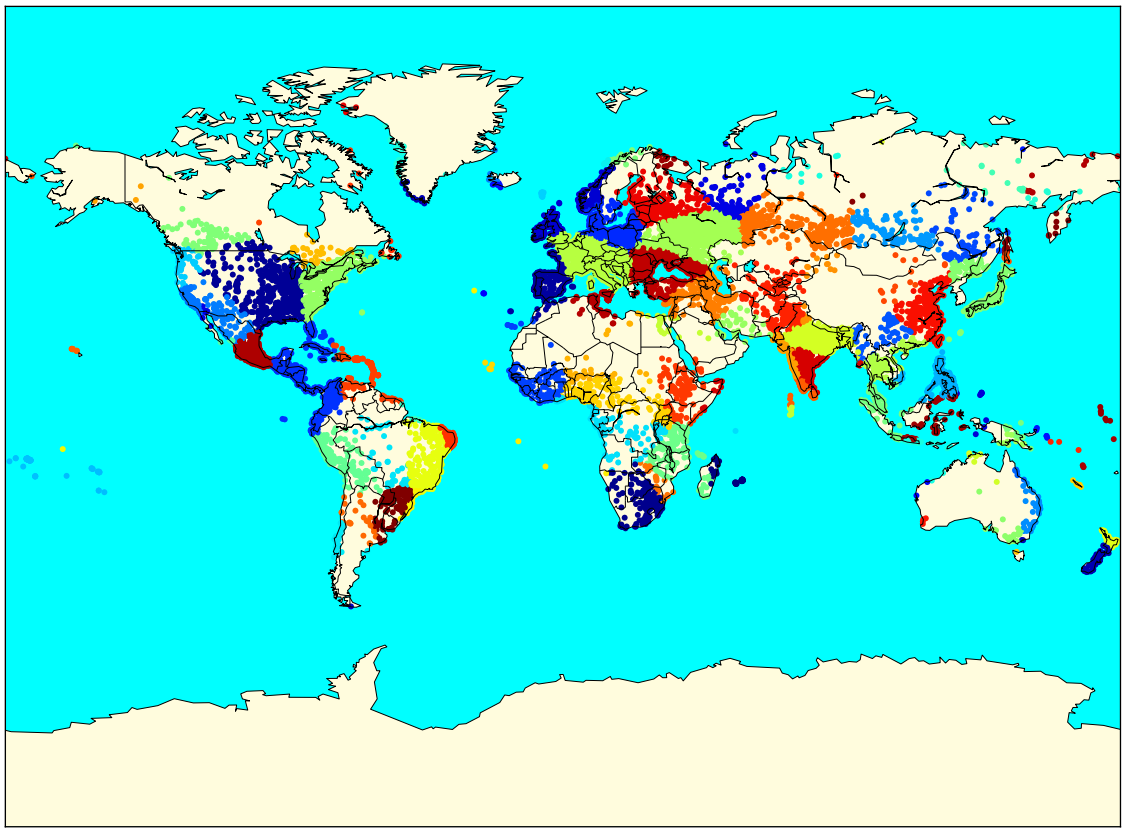


Южная Америка

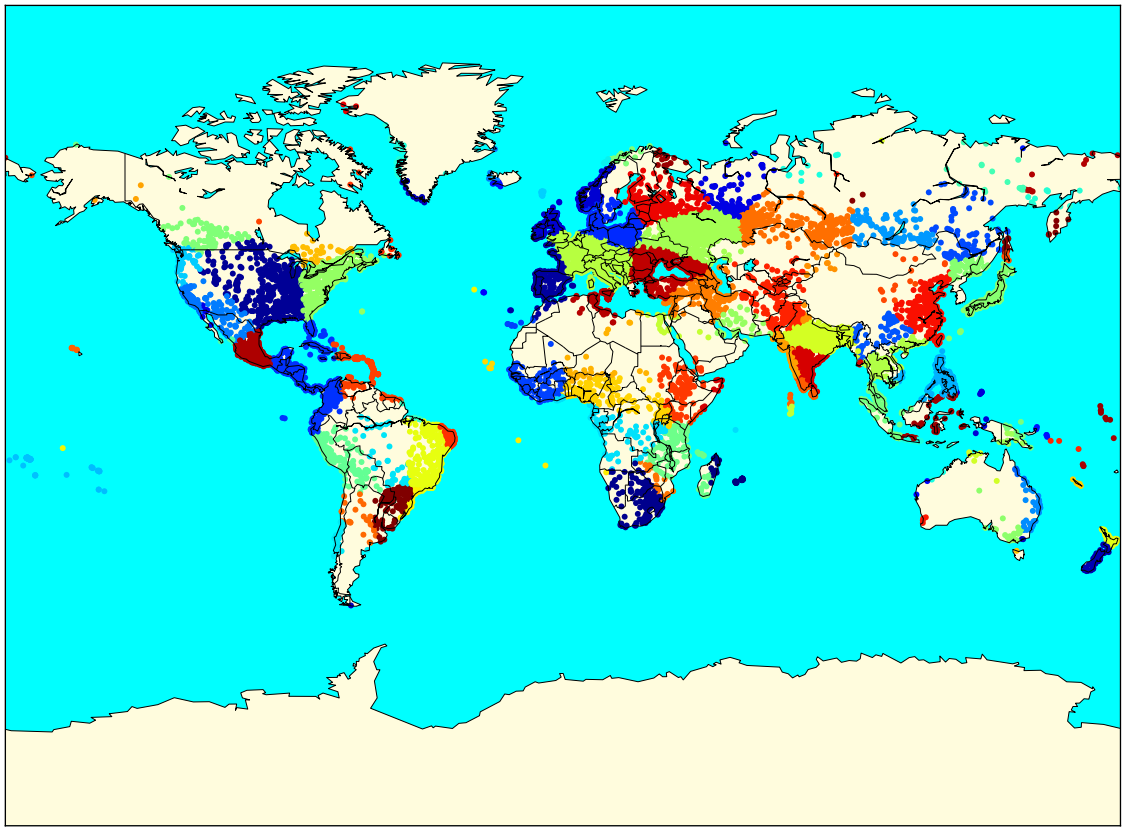


И наконец на всем Мире:

k-Means



k-Medoids



3 Заключение

В результате исследовательской работы выяснилось, что один и тот же код написанный на Python работает хуже, чем код напсанный на Cython и тем более на C++. Но при этом точность не пострадала. Все результаты можно видеть в пунктах 2.2 и 2.3.

Для ускорения методов не применялось ничего заоблачного и в основном код не ускорялся. Для уменьшения вычислительных расходов радиус сферы брался равной единице. А для передачи данных в функцию на C++ использовался передача по адресу.