

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное образовательное учреждение высшего
профессионального образования «Санкт-Петербургский государственный
университет»
Математико-механический факультет
Кафедра статистического моделирования

Дипломная работа

Векличева Мария Константиновна

«ИССЛЕДОВАНИЕ ПЕРИОДИЧНОСТЕЙ В
ПРОСТРАНСТВЕННО-ВРЕМЕННЫХ ДАННЫХ»

Допущена к защите

Заведующий кафедрой:

д. ф.-м. н., профессор С. М. Ермаков

Научный руководитель:

к. ф.-м. н. А. И. Коробейников

Рецензент:

к. ф.-м. н., доцент Н. Э. Голяндина

Санкт-Петербург

2012

Saint Petersburg State University
Faculty of Mathematics and Mechanics
Department of Statistical Modelling

Graduation Thesis

Veklicheva Mariia Konstantinovna

INVESTIGATION OF PERIODICITIES IN SPATIAL-TEMPORAL DATA

Admitted for defence

Head of Department:

Professor S. M. Ermakov

Scientific Supervisor:

Assistant Professor A. Korobeynikov

Reviewer:

Associate Professor N. Golyandina

Saint Petersburg

2012

Оглавление

Введение	5
Глава 1. Базовый алгоритм	7
1.1. Кластеризация пространственных данных	8
1.2. Выделение периодов	8
1.3. Определение периодического поведения	8
Глава 2. Модификации	12
2.1. Кластеризация	12
2.1.1. Метод ядерной оценки плотности	13
2.1.2. Динамическое подрезание дендрограммы	14
2.2. Выделение периодов	18
2.2.1. Перестановочная процедура	18
2.2.2. Точный g -тест Фишера	19
2.2.3. Критерий распределения максимума периодограммы	20
2.3. Определение периодического поведения	21
2.3.1. Кластеризация дискретных распределений	22
Глава 3. Моделирование	25
3.1. Моделирование данных	26
3.1.1. Моделирование распределений	26
3.1.2. Моделирование последовательности передвижений и временных меток	30
3.1.3. Моделирование географических координат	34
3.2. Вариации алгоритма	35
3.3. Результаты моделирования	36
3.3.1. Выводы	43
Глава 4. Применение алгоритма к реальным данным	51
4.1. Кластеризация	51
4.2. Выделение периодов	51
4.3. Определение периодических поведений	52

4.3.1. Итоги	58
Заключение	59
Список литературы	60
Приложение А. Обозначения для параметров	61
Приложение Б. Реализация алгоритма	62
Приложение В. Моделирование данных	76
Приложение Г. Кластеризация распределений методом k -средних . . .	80
Приложение Д. Вариации алгоритма	82

Введение

Существуют задачи, когда имея последовательность наблюдений, состоящих из пары географических координат и временной метки (дата и время), необходимо выделить некую модель периодического поведения. Периодическое поведение упрощенно можно понимать как повторяющиеся появления наблюдений в определенных географических областях через регулярный промежуток времени (период). Например, перелетные птицы совершают сезонные перемещения между местами гнездовий и местами зимовок, и в этом случае мы можем говорить о периодическом поведении с периодом в один год. При этом, рассматривая только область гнездовий (или зимовок) можно говорить и суточной периодичности, когда основную часть времени птицы проводят в гнезде, но в течение дня улетають за пропитанием в пределах некоторой окрестности около гнезда. Идея модели периодического поведения основывается на том, что для каждого периода существует как минимум одна пространственная область, в которой наблюдения появляются через время, равное этому периоду.

Самые распространенные примеры задач, для решения которых необходимо построения модели периодического поведения, это: анализ путей миграций животных и предотвращения взломов в некие системы, например, интернет-банковские, которыми пользуется человек.

В работе будет рассмотрен алгоритм обнаружения периодических поведений в данных, предложенный в статье [1]. Был взят именно этот алгоритм потому, что он широко используется ресурсом <http://movebank.org/> для определения периодических поведений при исследовании путей миграций животных. Этот ресурс является самым большим хранилищем данных по перемещениям животных, полученных более чем за 20 лет наблюдений исследователями со всего мира, и координируется Институтом Орнитологии Макса Планка (Германия), Музеем Естественных Наук (Северная Каролина) и Констанцким Институтом (Германия). Однако при применении алгоритма к реальным данным другого происхождения (история авторизаций одного индивида в интернет-банковскую систему) не были получены ожидаемые результаты. В связи с этим возникла необходимость исследовать алгоритм, предложить возможные модификации и исследовать сами модификации.

Данная работа организована следующим образом: в Главе 1 вкратце будут описаны отдельные шаги алгоритма; в Главе 2 эти шаги будут рассмотрены более подробно, а также будут предложены некоторые их модификации; в Главе 3 будут описаны результаты моделирования; в Главе 4 будет подробно рассмотрен пример применения алгоритма к реальным данным.

Глава 1

Базовый алгоритм

Опишем алгоритм, предложенный в статье [1], который можно разделить на три почти самостоятельных этапа:

1. Кластеризация пространственных данных.
2. Определение периодов.
3. Выделение периодического поведения.

Этапы почти самостоятельны, потому что каждый следующий этап зависит только от результатов предыдущего, а не от способа получения этих результатов. Именно эта независимость позволила в дальнейшем рассматривать варианты модификации алгоритма. Так как основные идеи и определения понадобятся нам далее в работе, то остановимся на каждом этапе, предложенном в статье [1], но для начала сделаем небольшое замечание.

Начальные данные по передвижениям имеют вид:

$$D = \{(x_i^0, y_i^0, t_i)\}, i = 1, \dots, N,$$

где (x_i^0, y_i^0, t_i) — географические координаты и время наблюдения соответственно, а N — общее количество наблюдений. Алгоритм применим только для равноотстоящих по времени данных, поэтому прежде всего начальную последовательность перемещений D приводят к нужному виду с помощью линейной интерполяции с постоянным временным интервалом. В статье [1] в качестве единичного временного интервала берется один астрономический час¹, т.е. новая последовательность содержит ежечасные наблюдения. Не умоляя общности будем обозначать полученную последовательность как и раньше $D = \{(x_i, y_i)\}, i = 1, \dots, n$, где пара (x_i, y_i) — географические координаты, n — общее количество регуляризованных наблюдений. Параметр t_i больше не имеет важного значения, поэтому опускается, необходимо лишь помнить значение единичного временного интервала, т.к. он понадобится позже. Именно к новой последовательности наблюдений далее применяется алгоритм.

¹ Эффекты, связанные с переходом на зимнее/летнее время, временными зонами, игнорируются.

1.1. Кластеризация пространственных данных

Интуитивно понятно, что объект, имеющий периодическое поведение, будет появляться в некоторых пространственных областях чаще, чем в других. Для выделения таких часто посещаемых областей вычисляется ядерная оценка плотности на решетке, построенной на всей области, где есть географические наблюдения. Самые «плотные» области объявляются кластерами, а наблюдениям присваивается индекс того кластера, к которому данное наблюдение принадлежит, т.е. последовательность наблюдений D приобретает вид:

$$D = \{(x_i, y_i, c_i)\}, i = 1, \dots, n,$$

где (x_i, y_i) — как и раньше, географические координаты, c_i — индекс кластера, к которому принадлежит i -ое наблюдение. Более подробно этот метод кластеризации будет описан далее в Главе 2.

1.2. Выделение периодов

Отметим сразу, что выделение периодов для каждого из кластеров, полученных на предыдущем этапе, происходит отдельно. Поэтому для пояснений основного принципа этого этапа ограничимся рассмотрением работы шага для одного кластера.

Для кластера строится бинарная последовательность $B = b_1 b_2 \dots b_n$, где $b_i = 1$, если i -тое наблюдение последовательности D находится в рассматриваемом кластере и $b_i = 0$ иначе, n — общее число наблюдений в последовательности перемещений. Далее для последовательности B вычисляется периодограмма, и на её основе с помощью перестановочной процедуры, которая будет описана позже в Главе 2, определяются значения возможных периодов, заложенных в рассматриваемый кластер. Чтобы уточнить полученные значения периодов, дополнительно рассматривается автокорреляционная функция.

1.3. Определение периодического поведения

Процедура получения всех периодических поведений состоит из шага выделения периодического поведения для одного из периодов, полученных на предыдущем этапе, и

повторения этого шага для всех периодов. Поэтому опишем этот шаг только для одного периода T .

Сначала все кластеры, для которых период T возможен, объединяются в одну совокупность $O = \{o_1, \dots, o_d\}$, где o_i — индекс кластера, d — количество кластеров, для которых период T на предыдущем шаге был определен как возможный. К этой совокупности добавляется еще один кластер o_0 , обозначающий любой другой кластер, для которого период T не был обнаружен. Из уже имеющейся последовательности наблюдений D для полученной совокупности кластеров строится последовательность перемещений $S = s_1 \dots s_n$, где $s_i = j$, если i -е наблюдение находится в кластере с индексом o_j . Далее последовательность S делится на $m = \lfloor \frac{n}{T} \rfloor$ сегментов, обозначаемых I^j для $j = 1, \dots, m$. Через t_k ($1 \leq k \leq T$) обозначается k -тый относительный единичный момент времени в периоде. Запись $I_k^j = i$ будет означать, что наблюдение находится в кластере o_i в относительный момент времени t_k периода T в j -том сегменте. Отметим, что именно потому, что в статье [1] в качестве единичного временного интервала при линейной интерполяции берется один астрономический час, количество относительных моментов времени t_k равно T .

В статье [1] предлагается рассматривать модель одного периодического поведения для периода T в виде многомерной случайной величины:

$$\mu = (\mu_1, \dots, \mu_T),$$

где все μ_i независимы и μ_i принимает значения $\{o_0, o_1, \dots, o_d\}$, $i = 1, \dots, T$. Распределение случайной величины μ , в свою очередь, однозначно можно задать при помощи матрицы $\mathbf{P}$ следующего вида:

Определение 1. Пусть $\mathcal{T} = \{t_1, t_2, \dots, t_T\}$ — множество относительных моментов времени периода T . Пусть матрица $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_T]$ построена так, что столбец $\mathbf{p}_k = [P(\mu_k = o_0), P(\mu_k = o_1), \dots, P(\mu_k = o_d)]^T$ задает распределение кластеров в момент времени t_k , т.е. $\sum_{i=0}^d P(\mu_k = o_i) = 1$ и $0 \leq P(\mu_k = o_i) \leq 1$, $k = 1, \dots, T$. Тогда матрица $\mathbf{P}$ однозначно задаёт периодическое поведение μ , т.е. каждое μ_i принимает значения $\{o_0, \dots, o_d\}$ с вероятностями $\mathbf{p}_i$.

Рассмотрим набор сегментов $\mathcal{I} = \bigcup_{j=1}^l I^j$, тогда вероятность того, что этот набор порожден неким периодическим поведением μ с распределением $\mathbf{P}$, равна

$$P(\mathcal{I} | \mathbf{P}) = \prod_{I^j \in \mathcal{I}} \prod_{k=1}^T P(\mu_k = I_k^j).$$

Оценку матрицы $\mathbf{P}$ можно найти при помощи метода максимального правдоподобия, а именно:

$$\hat{\mathbf{P}} = \arg \max_{\mathbf{P}} \{L(\mathbf{P} \mid \mathcal{I}) = \log \mathbf{P}(\mathcal{I} \mid \mathbf{P}) = \sum_{I^j \in \mathcal{I}} \sum_{k=1}^T \mathbf{P}(\mu_k = I_k^j)\}$$

Несложно видеть, что:

$$\hat{\mathbf{P}}(\mu_k = o_i) = \frac{\sum_{I^j \in \mathcal{I}} \mathbf{1}_{I_k^j = i}}{|\mathcal{I}|}.$$

Ясно, что $\hat{\mathbf{P}}(\mu_k = o_i)$ — это относительная частота кластера o_i в относительный момент времени t_k по всем сегментам из набора $\mathcal{I}$.

Мы описали модель одного периодического поведения μ для периода T , но таких поведений для одного периода может быть несколько, т.е. мы потенциально можем иметь дело со смесью и тогда все сегменты I^j представляют собой реализации случайной величины, распределение которой является смесью ρ распределений периодических поведений μ^i , причем значение ρ неизвестно.

Отметим, что для одного фиксированного периода разные поведения сменяются последовательно, т.е. сначала некий промежуток времени последовательность наблюдений порождена распределением одного периодического поведения, потом — другого и т.д., причем сами распределения поведений могут повторяться через некий промежуток времени, который необязательно должен быть постоянным.

Таким образом задача сводится к задаче оценки смеси. Задача оценки смеси с неизвестным числом ρ компонент с точки зрения machine learning, как это и происходит в статье [1], рассматривается как задача кластеризации распределений, в результате которой выделяется ρ вероятностных распределений из m распределений, которые получаются как оценки распределений по каждому из сегментов $I^j, j = 1, \dots, m$. В статье [1] в качестве метода кластеризации авторы выбирают агломеративный иерархический (см., например, [2]), а в качестве расстояния — расстояние Кульбака-Лейблера, которое для двух поведений μ и η с распределениями $\mathbf{P}$ и $\mathbf{Q}$ соответственно, удовлетворяющих Определению 1, имеет вид:

$$KL(\mathbf{P}, \mathbf{Q}) = \sum_{k=1}^T \sum_{i=0}^d \mathbf{P}(\mu_k = o_i) \log \frac{\mathbf{P}(\mu_k = o_i)}{\mathbf{P}(\eta_k = o_i)}.$$

$KL(\mathbf{P}, \mathbf{Q})$ становится равным ∞ , когда $\mathbf{P}(\mu_k = o_i) = 0$ или $\mathbf{P}(\eta_k = o_i) = 0$. Чтобы этого избежать, предлагается регуляризовать вероятности, а именно положить:

$$\tilde{\mathbf{P}}(\mu_k = o_i) = (1 - \lambda) \mathbf{P}(\mu_k = o_i) - \lambda u,$$

где $0 < \lambda < 1$ — небольшой сглаживающий параметр, а $u = \frac{1}{d+1}$.

Количество ρ скрытых поведений в данных неизвестно, поэтому параллельно с каждой итерацией иерархической кластеризации выполняется оценка качества текущей итерации при помощи ошибки представления.

Определение 2 (Ошибка представления). Пусть есть набор сегментов $C = \{I^1, I^2, \dots, I^l\}$ и его периодическое поведение μ с распределением $\mathbf{P}$, тогда ошибка представления $\mathcal{E}(C)$ имеет вид:

$$\mathcal{E}(C) = \frac{\sum_{I^j \in C} \sum_{i=1}^T \mathbf{1}_{I_i^j \neq 0} (1 - \mathbf{P}(\mu_i = I_i^j))}{\sum_{I^j \in C} \sum_{i=1}^T \mathbf{1}_{I_i^j \neq 0}}$$

На каждой итерации все распределения делятся на k (k меняется от m до 1) кластеров $\{C_1, C_2, \dots, C_k\}$. Ошибка представления текущей итерации вычисляется как среднее по всем кластерам:

$$\varepsilon_k = \frac{1}{k} \sum_{i=1}^k \mathcal{E}(C_i)$$

Ошибка представления позволяет отслеживать изменения качества кластеризации: если при переходе от k к $k - 1$ кластеру ε_k резко возросла, то это означает, что при переходе к $k - 1$ кластеру объединились два разных распределения, соответствующие разным поведением, т.е. k было правильным количеством поведений.

Таким образом, после применения иерархической кластеризации с параллельной оценкой качества кластеризации, авторы выделяют нужные периодические поведения. Напомним, что это шаг только для одного периода, поэтому, повторив этот шаг для всех обнаруженных периодов, можно найти все поведения.

Глава 2

Модификации

Как уже отмечалось ранее, отдельные этапы алгоритма почти независимы, что позволяет на каждом отдельном этапе применять другие методы. Существует достаточно большое разнообразие альтернативных методов для каждого из этапов, но мы ограничимся рассмотрением только некоторых из них, которые и опишем далее.

2.1. Кластеризация

Напомним, что в статье [1] перед тем, как применять алгоритм, данные модифицируются с помощью линейной интерполяции так, чтобы они стали равноотстоящими по времени. Понятно, что результаты кластеризации модифицированных данных не могут отражать скрытое деление по кластерам начальных данных лучше, чем результаты кластеризации самих начальных данных до модификации, так как процесс модификации добавляет в данные наблюдения, которых в действительности не существует и которые могут достаточно сильно влиять на результаты кластеризации, например, достаточно сильно перетягивать на себя центры получившихся кластеров (особенно, если наблюдений мало). В связи с этим возникла идея на этапе кластеризации работать именно с начальными еще не модифицированными данными, и уже после этого трансформировать их так, чтобы они стали равноотстоящими по времени.

Существуют достаточно большое разнообразие методов кластеризации (см., например, [2]), но всем им необходимо или изначально задавать то количество кластеров, на которое требуется разделить данные, или вводить некий метод-критерий, который, работая параллельно с процессом кластеризации, будет определять оптимальное или количество кластеров, или непосредственно деление по кластерам. Куда как меньше существует методов, которые позволяют обнаруживать выбросы в данных, которые в действительности могут иметь место. Одним из таких наиболее известных методов является иерархическая кластеризация, но, так как его результатом является дендрограмма, то для него также необходим способ определения кластеров. Такой способ был предложен в статье [3]. Его главная идея заключается в анализе формы ветвей денд-

программы, на основе которого и происходит выделение кластеров. Этот способ будет описан далее.

Если представить, что у нас уже есть результаты кластеризации, то вполне естественным выглядит следующий способ модификации данных так, чтобы они стали равноотстоящими по времени:

1. добавить к уже обнаруженным кластерам новый кластер, который будет указывать на то, что в действительности неизвестно в каком из обнаруженных кластеров находится данное наблюдение;
2. все наблюдения, которые добавились в процессе модификации данных, отнести к этому новому кластеру.

После модификации данные будут иметь вид $D = \{c_i\}, i = 1, \dots, n$, где c_i — индекс кластера, которому принадлежит соответствующее наблюдение.

Далее будет описан метод кластеризации на основе ядерной оценки плотности, который используется в статье [1], и способ определения кластеров для иерархической кластеризации.

2.1.1. Метод ядерной оценки плотности

Напомним, что метод применяется к уже модифицированным данным.

Для последовательности наблюдений $D = \{(x_i, y_i)\}, i = 1, \dots, n$, где пара (x_i, y_i) — географические координаты, строится решетка на всей области, где имеются географические наблюдения. Для каждого узла решетки c оценка плотности имеет вид:

$$f(c) = \frac{1}{n\gamma^2} \sum_{i=1}^n \frac{1}{2\pi} \exp\left(-\frac{\|c - d_i\|^2}{2\gamma^2}\right),$$

где $d_i = (x_i, y_i)$ — координаты i -того наблюдения, $\|c - d_i\|$ — расстояние между узлом решетки c и i -ым наблюдением последовательности D , γ — сглаживающий параметр вида:

$$\gamma = \frac{1}{2}(\sigma_x^2 + \sigma_y^2)^{\frac{1}{2}} n^{-\frac{1}{6}},$$

где σ_x и σ_y — стандартные отклонения всей последовательности перемещений D вдоль x и y координат соответственно.

Вычислив значения плотности в каждом узле решетки, внутренняя область кластера может быть определена, как область, ограниченная линией, соединяющей узлы

решетки с равными между собой значениями плотности, но большими, чем граничное значение. Граничное значение задается как квантиль уровня p , где p определяется для разных случаев по-разному в зависимости от масштаба области, где есть наблюдения. Самые плотные области объявляются кластерами, а наблюдениям присваивается индекс того кластера, к которому данное наблюдение принадлежит, т.е. последовательность наблюдений D приобретает вид:

$$D = \{(x_i, y_i, c_i)\}, i = 1, \dots, n,$$

где (x_i, y_i) — как и раньше, географические координаты, c_i — индекс кластера, к которому принадлежит i -ое наблюдение, причем если наблюдение не принадлежит ни одному из найденных кластеров, то будем обозначать $c_i = 0$.

В действительности для дальнейшего выполнения алгоритма географические координаты не нужны, поэтому последовательность наблюдений можно свести к виду $D = \{c_i\}, i = 1, \dots, n$.

2.1.2. Динамическое подрезание дендрограммы

Одним из наиболее известных методов кластеризации, который позволяет одновременно с процессом кластеризации обнаруживать выбросы, является иерархическая кластеризация. На основе матрицы расстояний между кластеризуемыми наблюдениями и дендрограммы — результата применения иерархической кластеризации, в статье [3] был предложен способ определения кластеров, основанный на анализе формы ветвей дендрограммы. Отметим еще, что так как обрабатываются географические координаты, то вполне разумно в качестве расстояния использовать расстояние на Земном шаре, которое, согласно статье [4], для двух точек $\mathbf{x}$ и $\mathbf{y}$ с координатами (x_1, x_2) и (y_1, y_2) соответственно, измеряемыми не в градусах, а в радианах, может быть приближенно вычислено следующим образом:

$$d(\mathbf{x}, \mathbf{y}) = \frac{R}{\pi} \cdot \arctan \frac{\sqrt{1-a}}{\sqrt{a}},$$

где $R = 40041.47$ и $a = \left(\sin \frac{y_1 - x_1}{2} \right)^2 + \cos x_1 \cdot \cos y_1 \cdot \left(\sin \frac{y_2 - x_2}{2} \right)^2$.

В качестве входных данных алгоритму требуются матрица расстояний между наблюдениями и дендрограмма. В алгоритме можно выделить два основных шага:

1. обнаружение кластеров;

2. обработка оставшихся наблюдений, которые не попали в кластеры, обнаруженные на предыдущем шаге.

Чтобы описать алгоритм, необходимо ввести некоторые понятия.

Определение 3 (Ядро кластера). Ядром кластера называется совокупность n_c элементов, принадлежащих данному кластеру, которые были объединены на самых первых итерациях иерархической кластеризации.

Количество n_c элементов в ядре кластера определяется следующим образом:

$$n_c = \min\{\lfloor N_0/2 + \sqrt{N - N_0/2} \rfloor, N\},$$

где N_0 — минимальный размер кластера, N — число всех элементов всей рассматриваемой ветви или кластера.

Определение 4 (Разброс ядра). Разброс ядра $\bar{d}$ вычисляется как среднее попарных расстояний между элементами, принадлежащих ядру кластера.

Определение 5 (Кластерный интервал). Кластерный интервал g определяется как расстояние от $\bar{d}$ до высоты дендрограммы, на которой кластер присоединяется к оставшейся части дендрограммы.

Определение 6 (Радиус кластера). Радиус r кластера, содержащего элементы $\mathbf{e}_1, \dots, \mathbf{e}_n$, определяется следующим образом:

$$r = \max_{1 \leq i \leq n} \frac{1}{n-1} \sum_{j \neq i, j=1}^n d(\mathbf{e}_i, \mathbf{e}_j),$$

где $d(\mathbf{e}_i, \mathbf{e}_j)$ — расстояние между точками $\mathbf{e}_i$ и $\mathbf{e}_j$ на Земном шаре.

В алгоритме необходимы следующие параметры, которые задаются пользователем:

1. N_0 — минимальное количество элементов в кластере;
2. h_{max} — максимально допустимая высота, на которой могут объединяться элементы ветви;
3. g_{min} — минимальная длина кластерного интервала;
4. d_{max} — максимально допустимый разброс кластера.

Теперь опишем шаги алгоритма.

Шаг 1: Обнаружение кластеров

Ветви дендрограммы, удовлетворяющие всем нижеследующим условиям, объявляются кластерами:

- С1: ветвь имеет не меньше, чем N_0 элементов;
- С2: все высоты, на которых объединяются элементы рассматриваемой ветви, не больше h_{max} ;
- С3: кластерный интервал g не меньше g_{min} ;
- С4: разброс ядра $\bar{d}$ не больше d_{max} .

Алгоритм обрабатывает дендрограмму, начиная с самых первых объединений, которые произошли на стадии иерархической кластеризации. Объединения в дендрограмме могут быть трех типов и, соответственно, существует три возможных варианта действий алгоритма:

1. если объединяются два элемента, то создается новая ветвь, состоящая из этих элементов;
2. если объединяются один элемент и ветвь, то элемент добавляется к ветви;
3. если объединяются две ветви, то:
 - а. если обе ветви удовлетворяют всем условиям С1-С4, то они объявляются закрытыми кластерами, т.е. ни один элемент не присоединяется к ним на этом шаге, и все элементы, которые объединяются сразу после объединения двух новообразовавшихся кластеров, остаются без назначенного им кластера;
 - б. в противном случае ветви объединяются в одну ветвь.

Далее выполняется второй шаг алгоритма.

Шаг 2: Обработка элементов, которые не попали ни в один из кластеров

В данном шаге последовательно анализируются все элементы, которые не попали в кластеры на предыдущем шаге, исключая те ветви дендрограммы, которые из условий С1-С4 не удовлетворяют только условию С1. Такие ветви рассматриваются как

неделимые кластеры с небольшим числом наблюдений, то есть они либо полностью присоединяются к какому-либо кластеру, обнаруженному на первом шаге, либо полностью остаются как выбросы. Для всех элементов, которые остались без назначенного им кластера, и неделимых кластеров вычисляются расстояния до всех кластеров, обнаруженных на первом шаге, и если наименьшее из этих расстояний удовлетворяет хотя бы одному из нижеследующих условий, то элемент или неделимый кластер присоединяются к соответствующему кластеру:

1. расстояние меньше максимально допустимого d_{max} ;
2. расстояние меньше радиуса кластера.

Такой способ определения кластеров по результатам иерархической кластеризации оказывается достаточно эффективным.

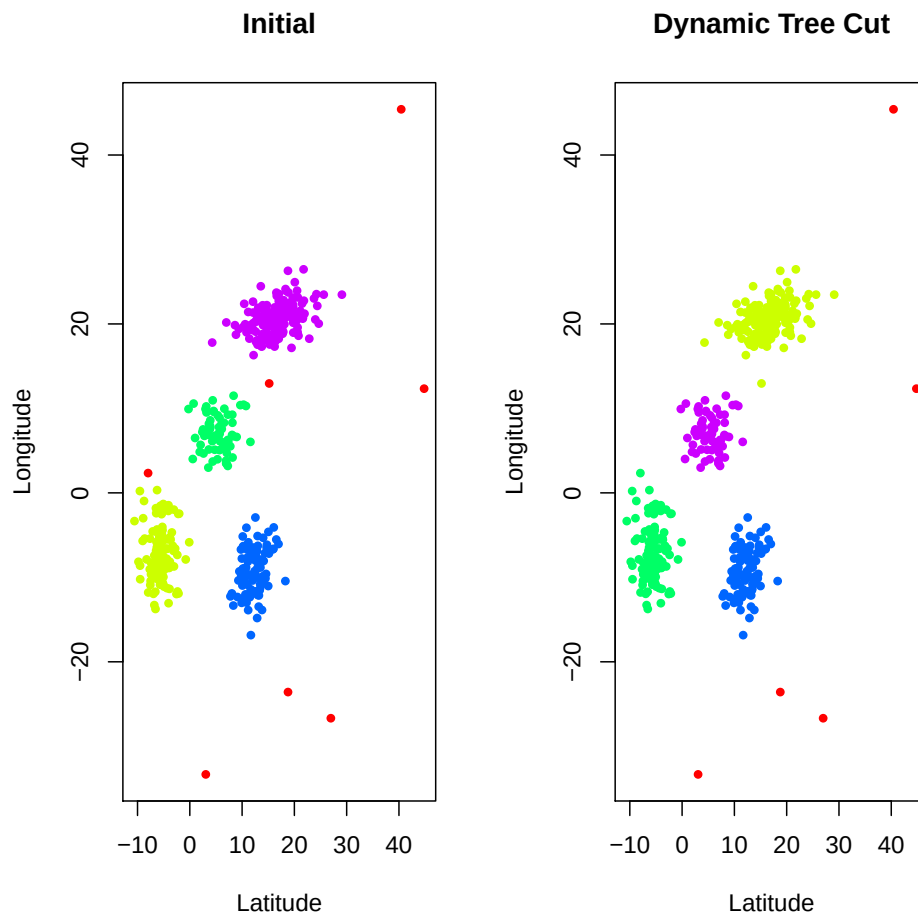


Рис. 2.1. Кластеризация: слева — заложенная в данные при их генерации, справа — полученная с помощью динамического подрезания дендрограммы.

2.2. Выделение периодов

Напомним, что выделение периодов происходит для каждого обнаруженного кластера отдельно.

Как уже описывалось в разделе 1.2, для рассматриваемого кластера строится бинарная последовательность B , вычисляется периодограмма и на её основе выделяются периоды. Для выявления наличия периодичности в данных существуют такие известные методы, как:

1. перестановочная процедура;
2. точный g -тест Фишера;
3. критерий распределения максимума периодограммы.

Прежде, чем более подробно описывать методы исследования периодичности в последовательности, введем понятие периодограммы.

Определение 7 (Периодограмма). Пусть дана последовательность $X = x_1 \dots x_l$ длины l , тогда периодограммой называется функция $I(\lambda)$ от частоты $\lambda \in [0, \pi]$ вида:

$$I(\lambda) = \frac{1}{l} \left| \sum_{t=1}^l x_t e^{-i\lambda t} \right|^2.$$

Пусть значение периодограммы посчитано в точках $\omega_1, \dots, \omega_q$, где $\omega_j = 2\pi j/l$ и $q = \lfloor (l-1)/2 \rfloor$.

Теперь опишем методы более подробно.

2.2.1. Перестановочная процедура

Пусть есть периодограмма, соответствующая бинарной последовательности B . Чтобы выделить значимые периоды, необходимо найти пороговое значение периодограммы и все те периоды, у которых соответствующее значение периодограммы больше порогового, считать значимыми. Пороговое значение вычисляется следующим образом:

1. посчитать максимум периодограммы перемешанной случайным образом бинарной последовательности;
2. повторить предыдущий шаг r раз;

3. взять α -квантиль необходимого уровня из полученной последовательности максимумов.

Перемешивание последовательности необходимо для того, чтобы избавиться от периодичности в ней, то есть если в начальной последовательности есть скрытая периодичность, то в перемешанной случайным образом последовательности её уже не будет. В статье [1] авторы утверждают, что хорошую оценку порогового значения можно получить в случае $r = 100$ и $\alpha = 0.99$, т.е. в качестве порогового значения берется 99-й по величине максимум. При таких значениях r и α определение порогового значения неустойчиво, поэтому мы используем $r = 200$ и $\alpha = 0.95$. В качестве возможных для рассматриваемого кластера периодов берутся только те, у которых соответствующее значение периодограммы больше порогового.

2.2.2. Точный g -тест Фишера

Точный g -тест Фишера (см., например, Глава 10 книги [5]) для некоторой последовательности $X = x_1 \dots x_l$ проверяет гипотезу о том, что она является Гауссовым белым шумом, т.е. все x_i независимы и $x_i \sim N(0, 1), i = 1, \dots, l$.

Пусть для последовательности $X = x_1 \dots x_l$ посчитана периодограмма $I(\lambda)$ в точках $\{\omega_j\}, j = 1, \dots, q$. Предположим, что последовательность X является Гауссовым белым шумом, тогда случайные величины Y_i вида:

$$Y_i = \frac{\sum_{k=1}^i I(\omega_k)}{\sum_{k=1}^q I(\omega_k)}, i = 1, \dots, q - 1$$

независимы и равномерно распределены на интервале $(0, 1)$.

Положим $Y_0 = 0$ и $Y_q = 1$ и определим случайную величину M_q :

$$M_q = \max_{1 \leq i \leq q} \frac{I(\omega_i)}{\sum_{i=1}^q I(\omega_i)},$$

тогда её функция распределения будет иметь вид:

$$F_{M_q}(x) = \sum_{j=0}^q (-1)^j C_q^j (1 - jx)_+^{q-1},$$

где $x_+ = \max(x, 0)$.

Основная идея критерия заключается в том, чтобы отвергнуть нулевую гипотезу о том, что данные являются белым шумом, если максимум периодограммы будет значительно больше, чем среднее периодограммы. В качестве статистики критерия рассматривается величина ξ_q , смысл которой выразить отношение максимума периодограммы

к среднему:

$$\xi_q = \frac{\max_{1 \leq i \leq q} I(\omega_i)}{q^{-1} \sum_{i=1}^q I(\omega_i)} = qM_q,$$

Функция распределения статистики критерия ξ_q выглядит следующим образом:

$$F_{\xi_q}(x) = F_{M_q}(x/q).$$

Чтобы применить тест, необходимо вычислить реализацию x случайной величины ξ_q для бинарной последовательности B и вычислить:

$$P(\xi_q \geq x) = 1 - \sum_{j=0}^q (-1)^j C_q^j (1 - jx/q)_+^{q-1}.$$

Если полученная вероятность окажется меньше заданного уровня значимости α , то нулевая гипотеза отвергается, то есть считаем, что в последовательности B есть периодичность.

Стоит заметить, что в действительности при использовании теста Фишера для определения наличия периодичности в данных периодичность данных приравнивается к тому, что данные не являются Гауссовым белым шумом.

2.2.3. Критерий распределения максимума периодограммы

Рассмотрим периодограмму $I(\lambda)$ последовательности X длины l и вычислим для нее величину:

$$M_l(X) = \max_{1 \leq i \leq q} I(\omega_i).$$

В работе [6] была доказана теорема:

Теорема 1. Пусть дана последовательность $X = x_1 \dots x_l$ из независимых, одинаково распределенных величин, удовлетворяющих условиям:

1. $Ex_i = 0$
2. $Ex_i^2 = 1$
3. $\exists s > 2 : E|x_i|^s < \infty$.

Тогда

$$M_l(X) - \ln q \xrightarrow{\mathcal{D}} Y$$

при $l \rightarrow +\infty$, где $\xrightarrow{\mathcal{D}}$ означает сходимость по распределению, а Y — стандартное распределение Гумбеля с функцией распределения $\Lambda(x) = e^{-e^{-x}}$, $x \in \mathbb{R}$.

Заметим, что в более распространенном критерии максимума периодограммы нулевая гипотеза представляет собой предположение о том, что последовательность является Гауссовым белым шумом. Последовательность $B = b_1, \dots, b_n$, которая исследуется на периодичность в рассматриваемом алгоритме, является бинарной, поэтому никак не может быть Гауссовым белым шумом и, соответственно, классическая версия критерия максимума периодограммы не будет работать должным образом. Именно Теорема 1 позволяет нам проверять последовательность B на наличие в ней периодичности.

2.3. Определение периодического поведения

Метод определения периодического поведения тем способом, который был предложен в статье [1] и описан в разделе 1.3, на практике оказался не так хорош, как хотелось бы. Поясним почему. Проверая метод на данных, в которых было заложено несколько периодических поведений, при кластеризации вероятностных распределений было логично ожидать, что распределения, порождающие разные поведения, будут объединяться на самых последних итерациях. Но этого не наблюдалось, т.к. начальная оценка поведения только по одному сегменту заведомо не будет хорошей. Если же оценивать поведение по набору нескольких сегментов, то полученная оценка будет заведомо лучше.

Модификации данного этапа имеют два направления:

1. применение иного расстояния между распределениями;
2. применение иного метода кластеризации распределений.

Для простоты будем предполагать, что все дискретные распределения, которые будут упоминаться далее, имеют такой же вид, как описывалось в разделе 1.3, в частности представлены в виде стохастической матрицы размером $(d + 1) \times T$.

В качестве альтернативных расстояний между дискретными распределениями, задаваемых матрицами $\mathbf{P}$ и $\mathbf{Q}$, можно рассмотреть следующие:

1. Расстояние Йенсена-Шеннона (см., например, [7]):

$$JSD^2(\mathbf{P}, \mathbf{Q}) = \sum_{i=0}^d \sum_{k=1}^T \left(p_{ik} \log \frac{p_{ik}}{r_{ik}} + q_{ik} \log \frac{q_{ik}}{r_{ik}} \right),$$

где $\mathbf{R} = \frac{1}{2}(\mathbf{P} + \mathbf{Q})$.

2. Расстояние Хеллинджера:

$$He^2(\mathbf{P}, \mathbf{Q}) = \sum_{i=0}^d \sum_{k=1}^T \left(\sqrt{p_{ik}} - \sqrt{q_{ik}} \right)^2$$

3. Условное:

$$C_D(\mathbf{P}, \mathbf{Q}) = D(\hat{\mathbf{P}}, \hat{\mathbf{Q}}),$$

где D — какое-либо расстояние между распределениями с соответствующими матрицами $\mathbf{P}$ и $\mathbf{Q}$, а $\hat{\mathbf{A}}$ означает матрицу размера $d \times T$, которая получается из матрицы $\mathbf{A}$ и имеет вид $[\hat{\mathbf{a}}_1, \dots, \hat{\mathbf{a}}_T]$, где $\hat{\mathbf{a}}_k = (a_{1k}, \dots, a_{dk})^T / \sum_{i=1}^d a_{ik}$, $k = 1, \dots, T$. Смысл этого преобразования состоит в том, чтобы «проигнорировать» наличие 0-го кластера в распределении.

Далее опишем альтернативный метод кластеризации.

2.3.1. Кластеризация дискретных распределений

Достаточно часто, когда имеется большое количество объектов, которые нужно кластеризовать, перед тем, как выполнять агломеративную иерархическую кластеризацию, сначала осуществляют кластеризацию каким-либо из хорошо известных способов, а потом уже, рассматривая полученные кластеры как новые объекты, выполняют иерархическую кластеризацию. Предварительная кластеризация также позволит улучшить оценки поведений. В связи с этим, в качестве иного способа кластеризации вероятностных распределений была рассмотрена комбинация метода k -средних с расстоянием Хеллинджера и иерархической агломеративной кластеризации с различными расстояниями. Обоснуем выбор метода k -средних именно с использованием расстояния Хеллинджера.

Пусть есть m дискретных T -мерных распределений $\{\mathbf{p}^1, \dots, \mathbf{p}^m\}$, тогда задачу нахождения ρ кластеров можно рассматривать как задачу нахождения таких распределений $\{\mathbf{c}^1, \dots, \mathbf{c}^\rho\}$ на их выпуклой оболочке Δ , чтобы сумма расстояний от каждого распределения $\mathbf{p}^j$ до ближайшего $\mathbf{c}^i$ была минимальной:

$$\{\mathbf{c}^1, \dots, \mathbf{c}^\rho\} = \arg \min_{\mathbf{c}^{i_1}, \dots, \mathbf{c}^{i_\rho} \in \Delta} \left(\sum_{j=1}^m \min_{1 \leq s \leq \rho} KL(\mathbf{p}^j, \mathbf{c}^{i_s}) \right) \quad (*)$$

Алгоритм нахождения таких распределений $\{\mathbf{c}^1, \dots, \mathbf{c}^\rho\}$ не может быть выполнен меньше, чем за экспоненциальное время, поэтому в статье [8] предлагается рассмотреть

алгоритм α -приближения, то есть алгоритм, который будет находить распределения $\{\tilde{\mathbf{c}}^1, \dots, \tilde{\mathbf{c}}^\rho\}$, удовлетворяющие следующему условию:

$$\frac{\sum_{j=1}^m \min_{1 \leq i \leq \rho} KL(\mathbf{p}^j, \tilde{\mathbf{c}}^i)}{\min_{\mathbf{c}^1, \dots, \mathbf{c}^\rho} \sum_{j=1}^m \min_{1 \leq i \leq \rho} KL(\mathbf{p}^j, \mathbf{c}^i)} \leq \alpha,$$

для некоторого $\alpha \geq 1$, и работать за полиномиальное время. Ясно, что чем меньше α , тем лучше приближение.

В этой же статье было показано, что для расстояний Хеллинджера и Кульбака-Лейблера выполняется следующее неравенство:

$$\frac{1}{2} \leq \frac{\min_{\mathbf{c} \in \Delta} \sum_{j=1}^m KL(\mathbf{p}^j, \mathbf{c})}{\min_{\mathbf{c} \in \Delta} \sum_{j=1}^m He(\mathbf{p}^j, \mathbf{c})} \leq \lceil \lg m \rceil \ln 16, \quad (**)$$

где $\mathbf{p}^1, \dots, \mathbf{p}^m$ — некоторые дискретные распределения.

Обозначим через $s^i, i = 1, \dots, \rho$ — множество тех индексов $j = 1, \dots, m$, для которых к соответствующему распределению $\mathbf{p}^j$ распределение $\mathbf{c}^i$ является ближайшим из всех распределений $\mathbf{c}^1, \dots, \mathbf{c}^\rho$. Ясно, что $\bigcap_{i=1}^\rho s^i = \emptyset$. Вспомним переформулировку задачи кластеризации согласно формуле (*) и применим к ней левую часть неравенства (**):

$$\begin{aligned} \min_{\mathbf{c}^1, \dots, \mathbf{c}^\rho} \sum_{j=1}^m \min KL(\mathbf{p}^j, \mathbf{c}^i) &= \min_{\mathbf{c}^1, \dots, \mathbf{c}^\rho} \left(\sum_{j \in s^1} KL(\mathbf{p}^j, \mathbf{c}^1) + \dots + \sum_{j \in s^\rho} KL(\mathbf{p}^j, \mathbf{c}^\rho) \right) = \\ &= \min_{\mathbf{c}^1} \sum_{j \in s^1} KL(\mathbf{p}^j, \mathbf{c}^1) + \dots + \min_{\mathbf{c}^\rho} \sum_{j \in s^\rho} KL(\mathbf{p}^j, \mathbf{c}^\rho) \geq \\ &\geq \frac{1}{2} \min_{\mathbf{c}^1} \sum_{j \in s^1} He(\mathbf{p}^j, \mathbf{c}^1) + \dots + \frac{1}{2} \min_{\mathbf{c}^\rho} \sum_{j \in s^\rho} He(\mathbf{p}^j, \mathbf{c}^\rho) = \\ &= \frac{1}{2} \min_{\mathbf{c}^1, \dots, \mathbf{c}^\rho} \sum_{j=1}^m \min He(\mathbf{p}^j, \mathbf{c}^i). \end{aligned}$$

Из этого вытекает, что алгоритм кластеризации с использованием расстояния Хеллинджера является α -приближенным алгоритмом для кластеризации распределений с расстоянием Кульбака-Лейблера при $\alpha = 2$. При этом расстояние Хеллинджера между дискретными распределениями $\mathbf{p}$ и $\mathbf{q}$ можно выразить следующим образом:

$$He(\mathbf{p}, \mathbf{q}) = l_2^2(\sqrt{\mathbf{p}} - \sqrt{\mathbf{q}}),$$

где $\sqrt{\mathbf{x}} = (\sqrt{x_1}, \dots, \sqrt{x_T})$ и $l_2^2(\mathbf{x}) = \sum_{i=1}^T x_i^2$.

В связи с этим для кластеризации дискретных распределений с использованием расстояния Хеллинджера был предложен следующий метод:

1. для каждого распределения $\{\mathbf{p}^j\}, j = 1, \dots, m$ вычислить $\sqrt{\mathbf{p}^j}$;
2. применить метод k -средних для векторов $\{\sqrt{\mathbf{p}^1}, \dots, \sqrt{\mathbf{p}^m}\}$ и найти центры кластеров $\{\mathbf{c}^1, \dots, \mathbf{c}^\rho\}$;
3. для каждого полученного центра $\mathbf{c}^j$:
 - а. найти множество начальных распределений $\{\mathbf{p}^{j_1}, \dots, \mathbf{p}^{j_t}\}$, для которых центр $\mathbf{c}^j$ самый близкий;
 - б. в качестве итогового центра для кластера, содержащего множество распределений $\{\mathbf{p}^{j_1}, \dots, \mathbf{p}^{j_t}\}$ положить:

$$\tilde{\mathbf{c}}^j = \frac{1}{t} \sum_{i=1}^t \mathbf{p}^{j_i}.$$

Корректировать центры кластеров необходимо потому, что, вообще говоря, не гарантируется, что они будут лежать на единичном шаре и, соответственно, описывать некое вероятностное распределение.

Глава 3

Моделирование

Как уже было сказано ранее, отдельные этапы алгоритма почти независимы, что позволяет комбинировать на каждом этапе алгоритма различные методы, описанные в Главе 2. Благодаря такой вариативности алгоритма можно оценить, какие из сочетаний методов для отдельных этапов приводят к обнаружению периодических поведений, наиболее близких в некотором смысле к тем, что заложены в обрабатываемых данных. Чтобы избежать ошибочного определения периодических поведений из-за некорректной кластеризации пространственных данных, было решено оставить то деление по кластерам, которое закладывается в данные при их моделировании.

В связи с вышесказанным, проверяется работа различных методов для двух последних этапов алгоритма:

1. Определение периодов:

- перестановочная процедура;
- точный g -тест Фишера;
- критерий максимума периодограммы.

2. Выделение периодических поведений:

а. всевозможные сочетания метод–расстояние:

i. метод:

- иерархическая кластеризация;
- предварительная кластеризация с помощью метода k -средних и последующая иерархическая кластеризация;

ii. расстояние:

- расстояние Кульбака-Лейблера;
- расстояние Йенсона-Шеннона;
- расстояние Хеллинджера;
- условное с расстоянием Кульбака-Лейблера;
- условное с расстоянием Йенсона-Шеннона;

б. метод k -средних для распределений с расстоянием Хеллингера.

Данные, на которых можно экспериментировать с различными комбинациями методов алгоритма, сами могут быть качественно разными. Например, данные могут быть хорошо периодичны, но при этом быть мало регулярными, т.е. будет наблюдаться явная периодичность в данных в некоторые относительные моменты времени для всех периодов, но при этом таких моментов будет мало. Соответственно, сами данные тоже можно моделировать по-разному, варьируя степень выраженности периодичности и регулярность.

Основная идея моделирования состоит в том, чтобы сравнить работу различных методов последнего этапа алгоритма для качественно разных данных.

Весь процесс моделирования состоит из двух частей:

1. моделирование данных разного качества;
2. выполнение процедуры оценивания различных методов для смоделированных данных.

3.1. Моделирование данных

Моделирование данных происходит в несколько этапов:

1. моделирование распределений для каждого из поведений;
2. моделирование последовательности передвижений и соответствующих временных меток для каждого поведения согласно его распределению;
3. моделирование географических координат.

Опишем каждый этап подробнее.

3.1.1. Моделирование распределений

Для моделирования данных мы будем рассматривать совокупность групп поведений, каждая из которых состоит из нескольких поведений с различными периодами, и наблюдения, принадлежащие одной группе поведений, происходят в пределах одного и того же промежутка времени. Такая конфигурация позволяет усложнить структуру

данных, а также проверять работу методов определения периодичности. Так как одной группе принадлежат поведения с различными периодами, то для одного периода разные поведения будут принадлежать разным группам.

Моделирование распределения некоторого поведения сводится к моделированию матрицы, которая удовлетворяет Определению 1, поэтому далее будем говорить именно о моделировании матриц.

Одна группа поведений с различными периодами задается совокупностью пар вида (T_j, O^j) , где T_j — это период, измеряемый в астрономических часах, а O^j — набор индексов тех кластеров, в которых наблюдения будут появляться с периодом T_j для текущей группы поведений. Если говорить в терминах раздела 1.3, то T_j — это период T , а O^j — совокупность индексов кластеров $O = \{o_1, \dots, o_d\}$. Отметим, что в действительности это не та же самая совокупность из раздела 1.3 для того же периода T , т.к. там совокупности принадлежат индексы всех кластеров с периодом T , здесь же — только тех кластеров с периодом T , которые принадлежат рассматриваемой группе поведений. В действительности это различие не существенно для моделирования матрицы, но такая аналогия позволит лучше понимать сам процесс. Отметим, что в разделе 1.3 к совокупности кластеров O добавлялся кластер o_0 , обозначающий любой другой кластер, далее он тоже будет фигурировать, но относить его к совокупности O мы не будем. Уточним, что среди всего разнообразия периодов $\{T_j\}$ существует особенный период, обозначаемый NA , который означает, что для соответствующих ему кластеров нет никакой периодичности, и никакое распределение периодического поведения для него не моделируется.

Перед тем, как изложить сам процесс моделирования, расскажем о параметрах, в зависимости от значений которых генерируются качественно разные данные. Для упрощения изложения индекс j у периода T_j и набора O^j соответствующих кластеров будем опускать.

Регулярность данных определяется параметром r . Чтобы описать смысл этого параметра, напомним, что одной реализацией поведения с периодом T является сегмент длины T , каждый элемент которого принимает значения из $\{o_0, \dots, o_d\}$. Параметр r для каждого периода T задает математическое ожидание количества ненулевых элементов сегмента. Отметим, что хорошо задавать значение параметра регулярности для каждого периода так, чтобы оно было не меньше, чем наибольшее количество соот-

ветствующих периоду кластеров среди всевозможных поведений, которые участвуют в моделировании данных, иначе наблюдения из каких-то соответствующих периоду кластеров не будут периодическими с периодом T . Это станет более очевидным чуть позже в этом же подразделе.

Степень выраженности периодичности задается параметром $I_{per}(a, b)$, где a и b — минимальное и максимальное значения соответственно, которые могут принимать вероятности нахождения или в известном кластере (одном из совокупности O) в некоторый относительный момент времени периода t_k ($k = 1, \dots, T$), или в кластере o_o , если в этот момент времени нет наблюдения в известном кластере. Отметим, что чем ближе к 1 эти значения, тем больше вектор $\mathbf{p}_k$, задающий распределение индексов кластеров в относительный момент времени t_k , будет похож на бинарный и, как следствие, при дальнейшем моделировании последовательности перемещений в момент времени t_k наиболее вероятным результатом моделирования будет один и тот же индекс кластера, у которого соответствующее значение вероятности близко к 1.

Процесс моделирования матрицы распределения периодического поведения для каждого поведения из всех имеющихся групп поведений происходит отдельно.

Матрица распределения имеет размеры $(d + 1) \times T$, и нумерация строк матрицы начинается с 0.

Опишем алгоритм моделирования матрицы распределения формально, но сначала введем обозначения:

- $\alpha \leftarrow U[a, b]$ — получение реализации случайной величины с равномерным распределением на $[a, b]$;
- $\alpha \leftarrow U\{a_1, \dots, a_k\}$ — получение реализации случайной величины с равномерным распределением на множестве $\{a_1, \dots, a_k\}$;
- $\alpha_1, \dots, \alpha_n \leftarrow U\{a_1, \dots, a_k\}$ — получение n реализаций случайной величины с распределением на множестве $\{a_1, \dots, a_k\}$ без повторений, т.е. α_1 равномерно распределено на множестве $\{a_1, \dots, a_k\}$, α_2 равномерно распределено на множестве $\{a_1, \dots, a_k\} / \{\alpha_1\}$ и т.д.;
- $\alpha \leftarrow N(a, \sigma^2)$ — получение реализации случайной величины с нормальным распределением с соответствующими параметрами.

Алгоритм 1: Моделирование матрицы распределения периодического поведения

Вход: T — период;

$O = \{o_1, \dots, o_d\}$ — индексы кластеров, соответствующих периоду;

r — параметр регулярности;

$I_{per}(a, b)$ — степень выраженности периодичности;

Результат: $\mathbf{P}$ — матрица распределения.

```

1  $j_1, \dots, j_r \leftarrow U\{1, \dots, T\};$ 
2 if  $r \leq d$  then
3    $i_1, \dots, i_r \leftarrow U\{o_1, \dots, o_d\};$ 
4 else
5    $i_1 \leftarrow o_1, \dots, i_d \leftarrow o_d;$ 
6    $i_{d+1} \leftarrow U\{o_1, \dots, o_d\}, \dots, i_r \leftarrow U\{o_1, \dots, o_d\};$ 
7 end
8 for  $j = 1, \dots, T$  do
9   if  $j \in \{j_1, \dots, j_r\}$  then  $i \leftarrow i_j;$ 
10  else  $i \leftarrow 0;$ 
11  ;
12   $p_{ij} \leftarrow U[a, b];$ 
13  for  $l = 0, l \neq i$  to  $d$  do  $p'_{lj} \leftarrow U[0, 1];$ 
14  ;
15   $p_{lj} \leftarrow p'_{lj}(1 - p_{ij}) / \sum_{k=0, k \neq i}^d p'_{kj};$ 
16 end

```

Дадим некоторые комментарии к вышеописанному алгоритму.

В строке 1 случайным образом выбираются r относительных моментов времени периода, в которые будут появляться наблюдения, принадлежащие совокупности O и формирующие периодичность. Далее для каждой выбранной метки из $\{j_1, \dots, j_r\}$ в строках 3, 5 и 6 произвольным образом назначается индекс кластера из совокупности $\{o_1, \dots, o_d\}$, в котором и будет находиться наблюдение. Отметим, что в зависимости от соотношения параметра регулярности r и количества соответствующих периоду кластеров d , индексы кластеров назначаются по-разному. Именно для того, чтобы все кластеры из совокупности соответствующих периоду кластеров были выбраны и, как

следствие, не осталось кластеров, которые соответствуют периоду, но не представлены в распределении, чуть выше было сделано замечание о хорошем выборе параметра регулярности для периода. Это особенно важно при дальнейшей проверке работы алгоритма, т.к. в противном случае полученные результаты могут сильно расходиться с ожидаемыми именно из-за некорректно смоделированных данных, а не из-за некачественной работы алгоритма. Далее в строках 11-15 описывается моделирование отдельного столбца матрицы, которое повторяется для всех столбцов.

Итак, повторив построение матрицы для каждого поведения из всех групп поведений, мы получим все необходимые распределения.

Рассмотрим небольшой пример. Пусть есть:

1. период $T = 12$ и совокупность соответствующих кластеров $O = \{3, 5\}$;
2. параметр регулярности $r = 3$;
3. параметр $I_{per}(a, b)$ при $a = 0.95$ и $b = 1$.

Для периода $T = 12$ часов совокупностью соответствующих кластеров будет совокупность $O = \{o_1, o_2\}$, где $o_1 = 3, o_2 = 5$ и, соответственно, $d = 2$. Матрица $\mathbf{P}$ будет иметь размер 3×12 и может иметь следующий вид:

	1	2	3	4	5	6	7	8	9	10	11	12
0	0.020	0.992	0.976	0.994	0.040	0.988	0.004	0.973	0.975	0.997	0.984	0.998
3	0.009	0.006	0.009	0.003	0.007	0.001	0.988	0.014	0.018	0.002	0.015	0.002
5	0.970	0.002	0.015	0.002	0.953	0.011	0.008	0.013	0.007	0.001	0.001	0.001

Таблица 3.1. Матрица распределения поведения.

3.1.2. Моделирование последовательности передвижений и временных меток

Напомним, что группы поведений сменяются поочерёдно друг за другом, и наблюдения, порожденные одной группой поведений, лежат в одном временном интервале. В связи с этим, массив данных, состоящий из последовательности индексов кластеров и временных меток, для разных групп моделируется не параллельно, а последовательно.

Параметр b_{time} определяет для каждой текущей рассматриваемой группы поведений начальный момент времени, т.е. тот момент времени, с которого начинаются наблюдения. Для самой первой группы его значение равно 0 и меняется в зависимости от смоделированных данных для текущей группы поведений после того, как сами данные уже полностью смоделированы. Как именно меняется параметр, будет описано ниже. Отметим, что временные метки приобретают вид именно даты и времени только после того, как смоделированы данные для всех поведений. До этой трансформации временные метки принимают числовое значение и интерпретируется как количество часов, прошедших с момента начала наблюдений.

Параметры, которые участвуют в моделировании:

- N_{obs} — длительность наблюдений одного поведения, измеряемая в месяцах;
- τ — доля наблюдений, которые будут принадлежать кластерам, соответствующим рассматриваемому поведению, но не будут порождены периодическим поведением (по сути, это доля временных выбросов для рассматриваемого поведения);
- σ_t — дисперсия шума, который будет добавлен к последовательности временных меток;
- N_{na} — общее количество элементов во всех кластерах, которые соответствуют периоду $\mathbf{NA}$, для одного поведения;
- N_{out} — количество географических выбросов в данных.

Процедура моделирования данных последовательно повторяется для каждой группы поведений, но, рассматривая только одну группу, процедура также повторяется для всех поведений, принадлежащих группе. В связи с этим опишем процедуру только для одного поведения, но введем еще обозначения:

- $\mathbf{v} \leftarrow \mathbf{P}$ — моделирование одного сегмента по распределению $\mathbf{P}$;
- $c(v_1, \dots, v_k)$ — конкатенация векторов или скаляров $v_1, \dots, v_k$;
- $a : b$ — последовательность чисел от a до b с шагом 1.

Алгоритм 2: Моделирование последовательности перемещений и временных меток

Вход: T — период;

$O = \{o_1, \dots, o_d\}$ — индексы соответствующих периоду кластеров;

$\mathbf{P}$ — матрица распределения;

параметры: N_{obs} , τ , σ_t , N_{na} , b_{time} , N_{out} , r , $I_{per}(a, b)$.

Результат: Последовательности перемещений $\mathbf{s}$ и временных меток $\mathbf{t}$

```

1  if  $T \neq NA$  then
2       $n_{rep} \leftarrow \lfloor 24 \cdot 30 \cdot N_{obs} / T \rfloor$ ,  $N_{per} \leftarrow n_{rep} \cdot T$ ;
3      for  $i = 1$  to  $n_{rep}$  do  $\mathbf{s}_{(i-1)T+1}^{per}, \dots, \mathbf{s}_{iT}^{per} \leftarrow \mathbf{P}$ ;
4      ;
5       $k \leftarrow \#\{\mathbf{s}_i^{per} \neq 0\}, i = 1, \dots, N_{per}$ ;
6       $N_{junk} \leftarrow \lfloor (k \cdot \tau) / (1 - \tau) \rfloor$ ;
7      for  $i = 1$  to  $N_{junk}$  do  $\mathbf{s}_i^{junk} \leftarrow U\{o_1, \dots, o_d\}$ ;
8      ;
9       $\mathbf{t}^{per} \leftarrow 1 : N_{per}$ ;
10      $s_{min} \leftarrow \arg \min(\mathbf{s}_i^{per} \neq 0)$ ,  $s_{max} \leftarrow \arg \min(\mathbf{s}_i^{per} \neq 0)$ ;
11     for  $i = 1$  to  $N_{junk}$  do  $\mathbf{t}_i^{junk} \leftarrow U[s_{min}, s_{max}]$ ;
12     ;
13      $\mathbf{s} \leftarrow c(\mathbf{s}^{per}, \mathbf{s}^{junk})$ ,  $\mathbf{t} \leftarrow c(\mathbf{t}^{per}, \mathbf{t}^{junk})$ ;
14     for  $i = 1$  to  $(N_{per} + N_{junk})$  do
15          $\varepsilon \leftarrow N(0, \sigma_t)$ ;
16          $\mathbf{t}_i \leftarrow \mathbf{t}_i + b_{time} + \varepsilon$ ;
17     end
18 else
19     for  $i = 1$  to  $N_{na}$  do  $\mathbf{s}_i \leftarrow U\{o_1, \dots, o_d\}$ ;
20     ;
21      $e_{time} \leftarrow b_{time} + \lfloor 30 \cdot 24 \cdot N_{obs} / T \rfloor$ ;
22     for  $i = 1$  to  $N_{na}$  do  $\mathbf{t}_i \leftarrow U[b_{time}, e_{time}]$ ;
23     ;
24 end
```

Прокомментируем некоторые шаги алгоритма.

Вся процедура разбивается на два случая в зависимости от значения периода. Если период не равен NA:

Сначала вычисляется количество сегментов n_{rep} (строка 2), которое нужно смоделировать согласно распределению $\mathbf{P}$. Это количество считается как наибольшее целое число сегментов длины T , общая продолжительность которых не превосходит длительности наблюдений N_{obs} . Для простоты считаем, что один календарный месяц состоит из 30 дней. Здесь же (строка 2) вычисляется N_{per} — количество регулярных наблюдений, порожденных рассматриваемым периодическим поведением. Далее (строка 4) повторяем n_{rep} раз моделирование сегмента по распределению $\mathbf{P}$, получая таким образом регулярную последовательность перемещений $\mathbf{s}^{per}$. В строке 5 вычисляется количество элементов последовательности $\mathbf{s}^{per}$, которые не равны 0. Это необходимо для того, чтобы зная долю τ «шумовых» наблюдений, которые будут принадлежать кластерам с индексами из совокупности O , но не будут принадлежать периодическому поведению, посчитать их количество (строка 6) и смоделировать (строка 8). Далее для регулярной последовательности $\mathbf{s}^{per}$ создается регулярная последовательность временных меток $\mathbf{t}^{per}$ (строка 9). Последовательность временных меток $\mathbf{t}^{junk}$ для последовательности перемещений $\mathbf{s}^{junk}$ моделируется (строки 10-12) случайным образом. Далее (строки 15-16) последовательности временных меток зашумляются и сдвигаются относительно начала наблюдений b_{time} .

Если период равен NA, то последовательности перемещений и временных меток моделируются случайным образом (строки 20-23).

После этого необходимо исключить наблюдения с индексом кластера, равным 0, и смоделированные любым из выше описанных способом наблюдения добавить к уже ранее смоделированным.

Если был рассмотрен последний период в текущей группы поведения и далее уже необходимо моделировать данные для следующей группы, то в качестве нового значения параметра b_{time} берется самое близкое (в арифметическом смысле) к последнему моменту времени в уже смоделированных данных начало нового сегмента, длина которого равна значению наименьшего периода среди всех групп поведения. Нельзя брать в качестве нового начала наблюдений следующую за последней временной меткой в данных потому, что почти наверняка получится сдвиг во относительных временных метках, т.е. периодичность будет, но она будет наблюдаться в отличных от ожидаемых времен-

ных метках, и, как следствие, результаты всего алгоритма будут сильно расходиться с ожидаемыми.

После того, как смоделированы данные для всех поведений всех групп, моделируется еще блок данных, если $N_{out} \neq 0$:

1. В качестве последовательности перемещений берется последовательность из нулей длины N_{out} .
2. Последовательность временных меток получается как N_{out} реализаций равномерного распределения, с минимальным и максимальным значениями равными времени самого раннего и самого позднего возможных наблюдений соответственно.

Вновь смоделированный блок данных присоединяется к основному массиву данных.

Последовательность перемещений берется именно из нулей только из-за того, что процедура моделирования географических координат наблюдениям-выбросам по умолчанию приписывает принадлежность кластеру с индексом 0.

На этом этап моделирования последовательностей индексов кластеров и временных меток завершается. Отметим только, что последовательность перемещений станет последовательностью перемещений в смысле раздела 1.3 только после того, как будет упорядочена по времени.

3.1.3. Моделирование географических координат

Моделирование географических координат происходит с использованием функции `genRandomClust` из пакета `clusterGeneration` (см. [9]). У функции достаточно большое разнообразие параметров, но нам потребуются задавать только следующие:

- `numClust` — количество кластеров;
- `clustSizes` — размеры кластеров;
- `sepVal` — коэффициент разделимости кластеров;
- `numOutlier` — количество выбросов.

Количество кластеров `numClust` вычисляется как то количество различных индексов кластеров, исключая нулевой, которые встречаются в данных, смоделированных этапом выше, а размеры кластеров `clustSizes` — это частоты соответствующих

кластеров, опять же исключая нулевой. Коэффициент разделимости кластеров `sepVal` принимает значения из интервала $[-1, 1)$. Чем ближе значение параметра к 1, тем получаемые кластеры будут больше разделимы. Согласно документации к пакету, хорошо разделимыми кластеры будут уже при `sepVal = 0.34`. Значение параметра `numOutlier` равно N_{out} .

Результатом функции `genRandomClust` являются наблюдения, состоящие из пары географических координат и индекса соответствующего кластера. Эти наблюдения и данные, полученные на предыдущем этапе, объединяются в один большой массив данных. Далее географические выбросы, т.е. наблюдения, у которых индекс кластера равен 0, произвольным образом приписываются к остальным кластерам, а временные метки приводятся к формату дата и время, и весь массив данных сортируется по времени — от самых ранних до самых поздних наблюдений.

3.2. Вариации алгоритма

Для определения значений периодов, характерных для каждого кластера, используется перестановочная процедура, описанная в разделе 2.2.1, при этом параллельно выполняются точный g -тест Фишера и критерий максимума периодограммы, описанные там же, но для следующего этапа определения периодических поведений используются результаты именно перестановочной процедуры, т.к. только она обнаруживает непосредственно значения периодов, а не только наличие/отсутствие периодичности, как это происходит в последних двух тестах

Как уже было сказано в начале главы, для последнего этапа определения периодических поведений используются различные сочетания методов и расстояний:

1. иерархическая кластеризация;
2. сочетание метода k -средних и иерархической кластеризации.
3. метод k -средних;

Иерархическая кластеризация в первых двух методах выполняется ровно так, как описывалось в разделе 1.3, с той только разницей, что первом случае в качестве начальных распределений берутся оценки распределений по единичным сегментам, а во втором случае — распределения, полученные в результате предварительного применения метода k -средних.

Метод k -средних выполняется точно так, как описывалось в разделе 2.3.1. Единственное различие между вторым и третьем методом состоит в количестве итоговых распределений: во втором случае это некоторое разумное промежуточное количество, чтобы далее имелся смысл выполнять иерархическую кластеризацию, а в третьем — количество истинных поведений,

Для иерархической кластеризации используются следующие расстояния:

- расстояние Кульбака-Лейблера;
- расстояние Йенсена-Шеннона;
- расстояние Хеллинджера;
- условное с расстоянием Кульбака-Лейблера;
- условное с расстоянием Йенсена-Шеннона.

Комбинируя таким образом, мы можем сравнить работу различных вариантов сочетаний методов для данных, смоделированных с различными параметрами.

3.3. Результаты моделирования

При моделировании данных некоторые параметры оставались фиксированными:

- Совокупность поведений состояла из двух групп, каждая из которых содержала один период в 24 часа и период, равный ΔA . Последний период необходим для того, чтобы иметь возможность проверять работу методов определения наличия периодичности.
- Значение параметра делимости кластеров `sepVal` установился равным 0.4.
- Продолжительность наблюдений N_{obs} была установлена равной 10 для большей устойчивости обнаружения периодов.
- Степень выраженности периодичности $I_{per}(a, b)$ зафиксирована равной $I_{per}(0.95, 1)$.

Параметры, которые варьировались при моделировании данных:

- Доля наблюдений τ , которые принадлежат кластерам, соответствующим одному периоду, но не принадлежащих периодическому поведению, принимает значения от 0 до 0.5 с шагом в 0.1.

- Дисперсия шума σ_t , добавляемого к временным меткам принимает значения от 0 до 2 с шагом 0.25, т.е. от 0 до 2 часов с шагом 15 минут.
- Параметр регулярности r для периода в 24 часа изменяется от 2 до 24 с шагом 2.

Весь алгоритм был реализован в математической среде R версии 2.13.2 (см. [10]) с использованием некоторых дополнительных пакетов. Процесс моделирования весьма трудоемок: время, необходимое для повторения итерации, состоящей из моделирования данных и выполнения алгоритма со всевозможными комбинациями метод–расстояние, для всех комбинаций вышеперечисленных варьируемых параметров, составляет примерно 15 часов. Само моделирование выполняется на компьютере с 8 процессорами, и отдельные итерации перебора комбинаций параметров выполняются параллельно при помощи пакетов `foreach` (см. [11]) и `doMC` (см. [12]).

Результатом одной итерации моделирования является:

- список значений параметров, используемых в алгоритме.
- начальные распределения, по которым моделировались данные;
- результат проверки данных на периодичность;
- набор итоговых распределений, полученных при применении каждой комбинации метод–расстояние;

Чтобы оценить качество определения периодических поведений, для каждого полученного поведения вычисляется ошибка, т.е. расстояние до ближайшего из начальных поведений, относительно которых моделировались данные. Ясно, что чем меньше полученное расстояние, тем лучше сработал алгоритм. В качестве расстояния рассматривали среднее модуля разности между соответствующими вероятностями распределений, т.е. для двух распределений с задающими их матрицами $\mathbf{P}$ и $\mathbf{Q}$ расстояние имеет вид:

$$\frac{1}{(d+1) \cdot T} \sum_{i=0}^d \sum_{k=1}^T |p_{ik} - q_{ik}|.$$

Заметим, что результаты моделирования весьма громоздки (более 25 тысяч значений), поэтому весь анализ результатов моделирования проводится на элементарном иллюстративном уровне. Так как моделирование проводилось не так много раз, то при сравнении различных методов можно говорить только об общих тенденциях.

При моделировании данных параметры перебирались так, что параметр τ менялся медленнее всех, а параметр r — быстрее всех. При первичном анализе результатов было замечено, что независимо от применяемых методов при увеличении значений параметров величина ошибки увеличивается, причем более быстро меняющийся параметр оказывает большее влияние на значение ошибки, чем параметр, меняющийся более медленно, т.е. по оказываемому влиянию на итоговый результат алгоритма параметры, используемые при моделировании данных, можно расположить в порядке увеличения влияния следующим образом: τ , σ_t , r . Тенденция к постепенному увеличению ошибки при увеличении значений отдельных параметров, безусловно, весьма ожидаема, интересна именно закономерность влияния комбинаций различных параметров на итоговый результат алгоритма.

Напомним, что модельные данные независимо от параметров, с которыми они моделировались, содержат два поведения для периода в 24 часа, первое из которых с соответствующими двумя кластерами, второе — с одним кластером. Также напомним, что из-за вычислительной сложности было проведено только 5 моделирований, и поэтому мы не можем при анализе результатов моделирования делать статистически значимых выводов, а можем лишь говорить о наблюдаемых тенденциях.

Если посмотреть на влияние отдельных параметров, взяв при этом в качестве значения ошибки среднее ошибок по всем остальным параметрам, то можно увидеть следующее:

- Для параметра r значения ошибки постепенно возрастает при увеличении значения параметра, но резкий скачок происходит при переходе значения параметра от 22 к 24 (см. Рисунок 3.1).
- Для параметра σ_t резкий скачок ошибки происходит сразу при переходе значений от 0 к 0.25, а далее для оставшихся значений параметра выделяется тенденция к постепенному увеличению ошибки, но без строгого монотонного возрастания (см. Рисунок 3.2).
- Для параметра τ наблюдается постепенное увеличение значения ошибки без каких-то резких скачков (см. Рисунок 3.3).

При изучении своего рода проекций на отдельные параметры наблюдался тот факт, что для иерархической кластеризации с использованием расстояния Йенсона-Шеннона,

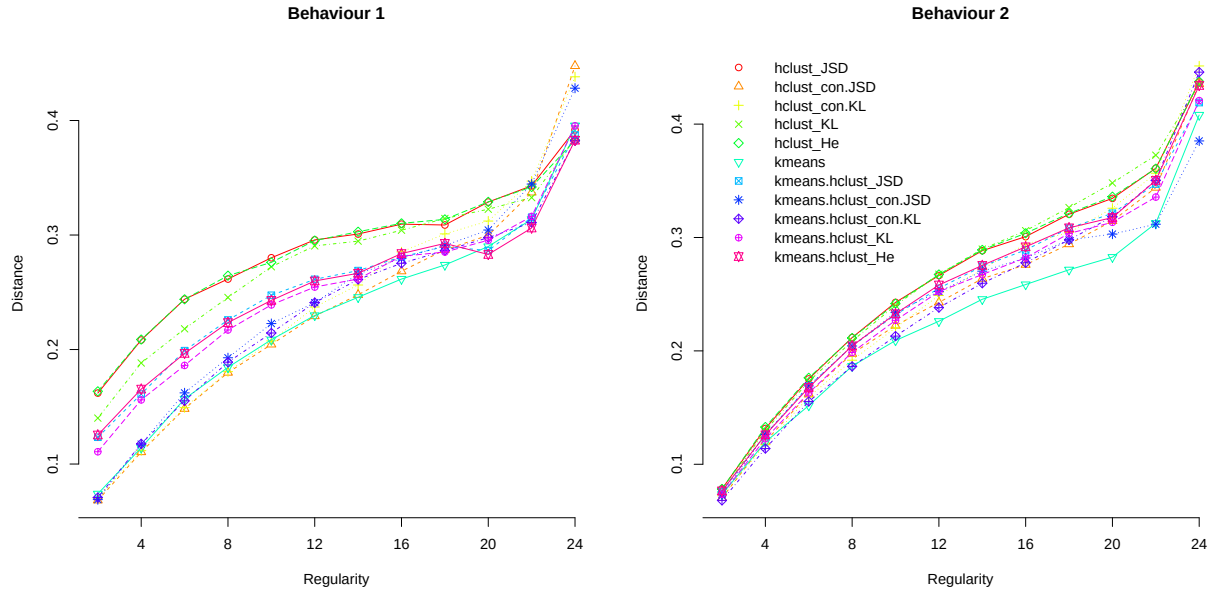


Рис. 3.1. Изменение ошибки при изменении значения параметра регулярности r .

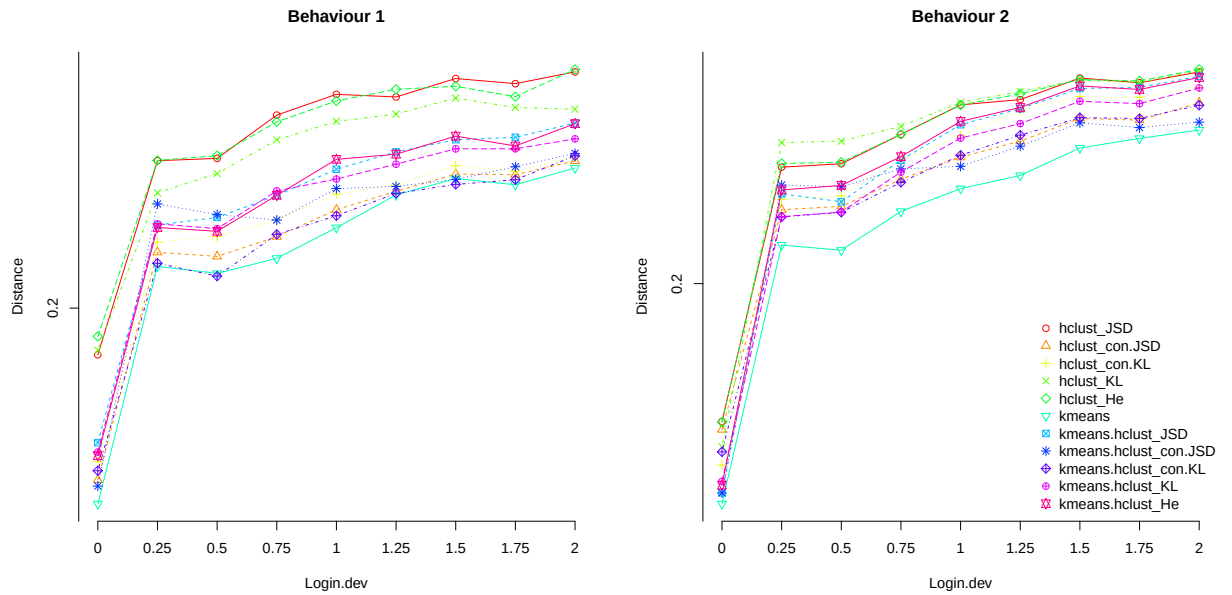


Рис. 3.2. Изменение ошибки при изменении дисперсии временного шума σ_t .

Кульбака-Лейблера и Хеллинджера значение ошибки больше, чем для остальных сочетаний метод–расстояние. При анализе влияния более сложных комбинаций параметров на значение ошибки итогового определения периодических поведений, подобная картина уже не наблюдалась столь явно, поэтому мы рассмотрели разницу ошибок между

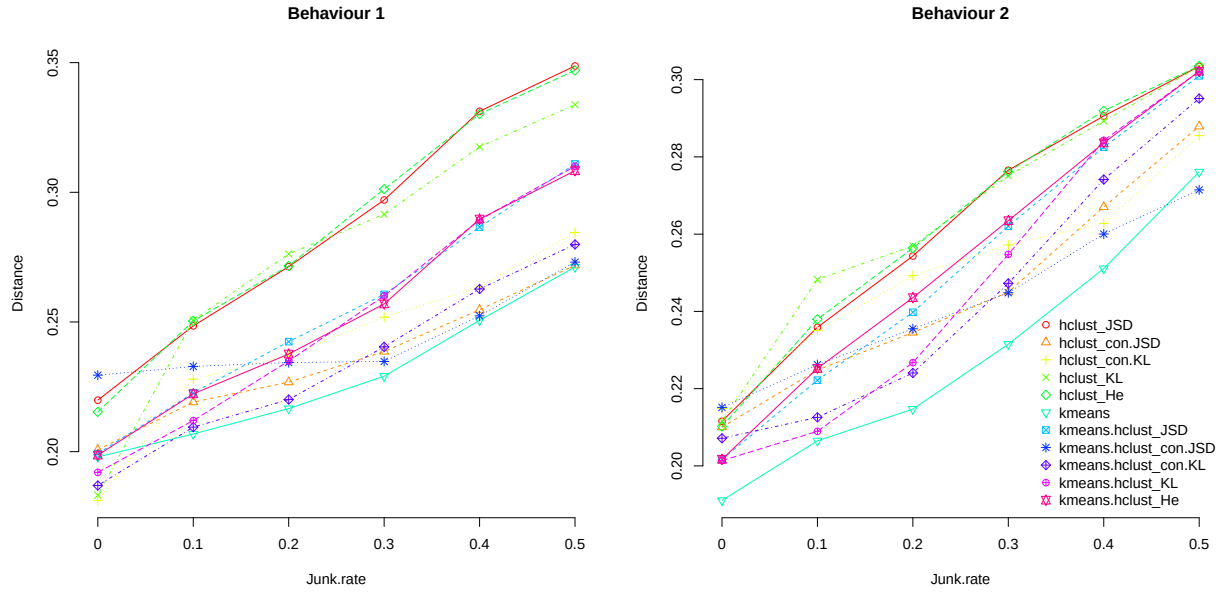


Рис. 3.3. Изменение ошибки при изменении доли неперiodических наблюдений τ .

различными методами с использованием одного расстояния, т.е. рассмотрели разность между ошибкой для метода иерархической кластеризации с расстоянием Хеллингера и ошибкой сочетания метода k -средних и иерархической кластеризации с тем же расстоянием и т.д. Соответственно, если полученная разность ошибок отрицательна, то метод иерархической кластеризации привел к результату, ошибка которого меньше, чем ошибка результата, полученного с помощью сочетания метода k -средних и иерархической кластеризации, а значит, метод иерархической кластеризации привел к лучшему результату. Разность между ошибками методов с одним и тем же расстоянием далее для простоты будем называть разницей. Также далее часто будет упоминаться слово «заметно», что будет означать что по сравнению со значениями разницы для других комбинаций всех параметров, величина разницы при некоторых конкретных условиях имеет одно из самых больших по модулю значений, уточнение о том, положительно или отрицательно само значение, также будет приводиться. Теперь опишем полученные результаты сравнения для каждого из возможных расстояний.

Для расстояния Йенсона-Шеннона:

1. Поведение 1 (см. Рисунок 3.4):

- Разница заметно меньше 0, когда τ равен 0.1 или 0.2 и при этом $\sigma_t \leq 0.5$.

- В целом видно, что при $r \geq 22$ разница меньше 0, но то, насколько велико отличие от 0, зависит от τ .
- В остальных случаях значение разницы в большинстве своем положительно.

2. Поведение 2 (см. Рисунок 3.5):

- Разница заметно больше 0, когда одновременно $\sigma_t = 0$ и τ не равен 0 или 0.5.
- При значении параметра $\tau \geq 0.4$ разница отлична от 0 только при значении $\sigma_t \leq 0.5$.
- При увеличении σ_t разница все менее заметно отличается от 0, при этом для каждого значения τ этот спад происходит со своей интенсивностью.
- Видна тенденция к тому, что при увеличении τ для всё меньших значений параметра σ_t разница хоть сколько-нибудь отлична от 0.
- Для подавляющего большинства всех комбинаций параметров значения разницы неотрицательны.

Для расстояния Хеллинджера:

1. Поведение 1 (см. Рисунок 3.6):

- В целом видно, что разница меньше 0 при $r \geq 18$, но при некоторых сочетаниях значений σ_t и r разница остается положительной и при большом значении регулярности.
- Разница заметно больше 0, когда τ равен 0.3 или 0.4 и при этом параметр $\sigma_t = 0$.
- Большая часть значений разницы неотрицательна, но сочетаний параметров, для которых разница отрицательна, не так мало.

2. Поведение 2 (см. Рисунок 3.7):

- Разница меньше 0, когда $0.25 \leq \sigma_t \leq 0.75$ при $\tau \leq 0.2$ и $r \geq 20$. Это отличие от 0 не столь существенно, но оно есть.
- Разница заметно больше 0, когда τ равен 0.2 при $0 \leq \sigma_t \leq 1$ и при средней регулярности.

- Разница заметно больше 0, когда τ равен 0.3 при $0 \leq \sigma_t \leq 0.75$ и большой регулярности.
- Разница почти везде равна 0 при $\tau \geq 0.4$ при любых сочетаниях параметров σ_t и r , исключая случай $\tau = 0.4$ при $\sigma_t = 0$.
- Для подавляющего большинства всех комбинаций параметров значения разницы неотрицательны.

Для расстояния Кульбака-Лейблера:

1. Поведение 1 (см. Рисунок 3.8):

- Разница заметно меньше при $r \geq 20$, но для $0 \leq \tau \leq 0.2$ при $0.25 \leq \sigma_t \leq 0.75$, а для $\tau \geq 0.3$ при $\sigma_t \geq 1.75$.
- Разница заметно больше 0, когда $0.2 \leq \tau \leq 0.4$ при $\sigma_t = 0$.
- В большинстве своем значения разницы положительны, но примерно треть всех значений отрицательна.

2. Поведение 2 (см. Рисунок 3.9):

- Разница заметно меньше 0, когда $0.1 \leq \tau \leq 0.4$ при $\sigma_t = 0$ и $r \geq 22$.
- Разница заметно больше 0, когда $\tau \leq 0.3$ при $0.25 \leq \sigma_t \leq 0.75$ и $r \geq 22$.
- Когда $\tau \geq 0.4$ и $\sigma_t \neq 0$, разница почти равна 0.
- Подавляющее большинство значений разницы неотрицательно.

Для условного расстояния Йенсона-Шеннона:

1. Поведение 1 (см. Рисунок 3.10):

- Фактически для всех сочетаний параметров разница отрицательна, но с увеличение τ наблюдается тенденция к её увеличению, но есть и некоторые заметные отличия от 0.
- Разница заметно больше при разных значениях σ_t в зависимости от значения τ , но при большой регулярности.
- Чуть меньше половины значений разницы положительна.

2. Поведение 2 (см. Рисунок 3.11):

- Разница также заметно больше 0 при различных значениях σ_t в зависимости от значения τ , но при $r \geq 20$.
- Разница заметно меньше 0, когда $\tau \leq 0.2$ при $\sigma_t \leq 0.5$ и $12 \leq r \leq 16$.
- Доля положительных значений разницы не превышает 40%.

Для условного расстояния Кульбака-Лейблера:

1. Поведение 1 (см. Рисунок 3.12):

- При увеличении значения τ наблюдается тенденция к уменьшению значения разницы, но для каждого значения $\tau \geq 0.1$ есть свои значения σ_t , для которых разница значительно больше 0, причем во всех случаях разница достигается при $r \geq 22$.
- Для $\tau = 0$ нет сочетаний параметров, при которых разница была бы заметно больше 0.
- Примерно половина значений разницы положительна.

2. Поведение 2 (см. Рисунок 3.13):

- Разница почти для всех сочетаний параметров неотрицательна при $0.1 \leq \tau \leq 0.3$, за исключением случаев, когда $\sigma_t = 0$.
- Значение разницы при $\tau = 0$ с увеличением σ_t уменьшается, но в большинстве случаев остается неотрицательной.
- Разница для большинства сочетаний параметров при $\tau \geq 0.4$ неположительна и не имеет значений, заметно больших 0.

3.3.1. Выводы

Из всего вышесказанного видно, что существуют достаточно сложные зависимости между применяемым расстоянием и параметрами, влияющими на качество данных, определяющие какой из методов (иерархическая кластеризация или сочетание метода k -средних и иерархической кластеризации) приведет к лучшему результату.

Один из первых вопросов, который возникает после ознакомления, это то, почему в некоторых случаях так ощутимо отличаются результаты для первого и второго поведения. Судя по всему, так происходит именно из-за различия в количестве кластеров,

соответствующих каждому из поведений: добавление шума к наблюдениям в процессе моделирования приводит к изменению итогового распределения периодического поведения в том смысле, что происходит своего рода перераспределение вероятностей между всеми кластерами, участвующими в распределении. Соответственно, для второго поведения с единственным кластером такое перераспределение происходит между соответствующим поведением кластером и нулевым кластером, который, в действительности нас не интересует. В первом же поведении подобное перераспределение затрагивает уже большее количество кластеров и, как следствие, приводит к более ощутимым отличиям оценки распределения от начального распределения.

Если абстрагироваться от применяемых методов и расстояний, то видно, что параметры тесно взаимосвязаны друг с другом и некоторые их сочетания почти мгновенно ведут к резкому увеличению ошибки. Например, казалось бы, большая регулярность (количество известных наблюдений на протяжении временного интервала, равного периоду) в данных должна приводить к меньшей ошибке, но это наблюдается только если дисперсия временного шума σ_t равна 0 и доля τ непериодических наблюдений мала (см. Рисунок 3.14). Причина такой закономерности кроется в том, что чем больше регулярность, тем вероятнее, что при ненулевой дисперсии временного шума зашумленные наблюдения или наблюдения, не принадлежащие периодическому поведению, попадут в некоторый относительный момент времени периода, для которого согласно начальному распределению, более вероятен другой кластер, и все подобные наблюдения «перетянут» на свой кластер часть этой вероятности. В итоге получится, что вероятности, заложенные в поведения изначально, распределились между всеми кластерами.

Подводя итог всем рассуждениям о результатах моделирования, приведем основные выводы:

- Условные расстояния не дают лучших результатов, чем традиционные расстояния.
- Применение предварительной кластеризации с помощью метода k -средних в большинстве случаев приводит к улучшению результата алгоритма.
- Использование расстояний Хеллингера и Йенсона-Шеннона приводят к весьма схожим результатам, которые лучше, чем результаты, полученные с использованием расстояния Кульбака-Лейблера.

- Выбор оптимального сочетания метод–расстояние напрямую зависит от параметров, используемых при моделировании данных.

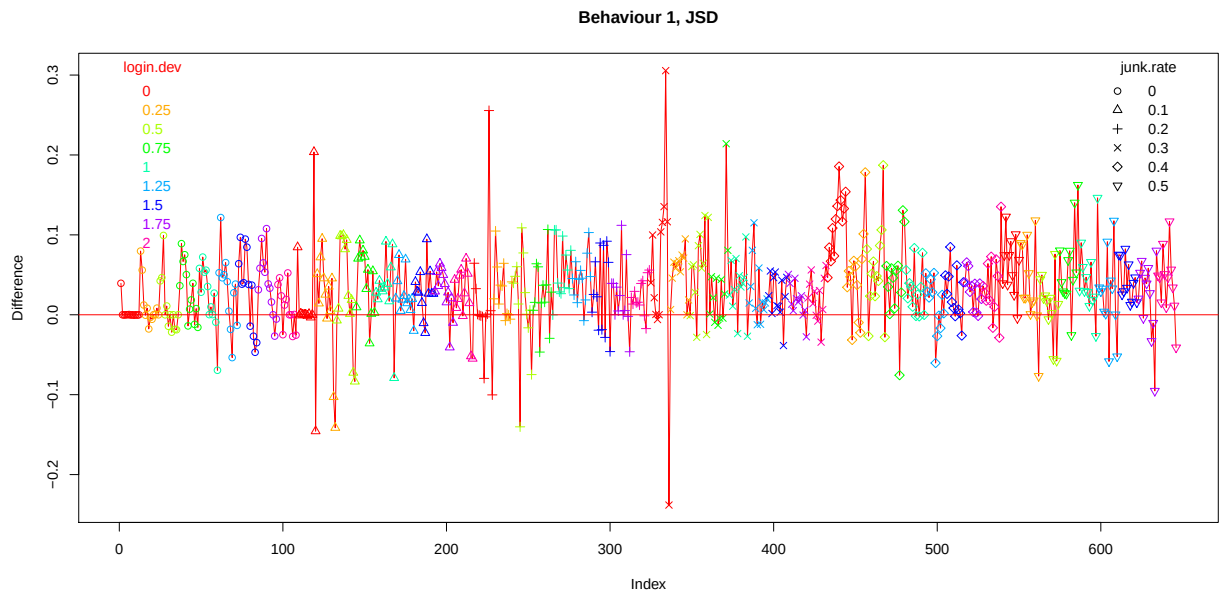


Рис. 3.4. Изменение разницы для расстояния Йенсона-Шеннона для поведения 1.

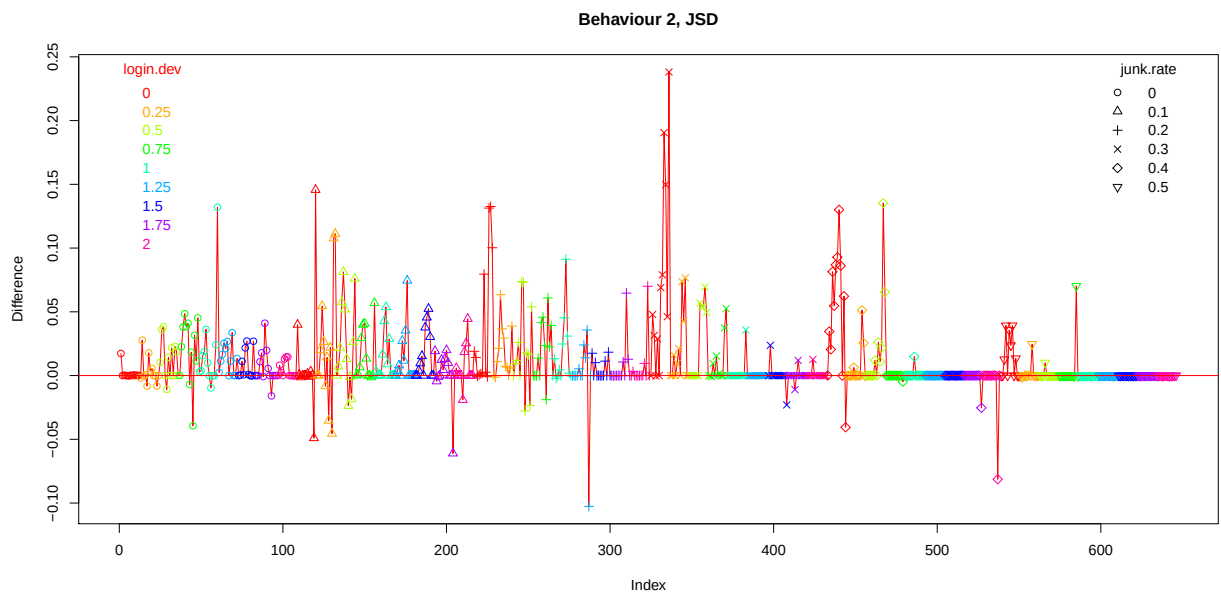


Рис. 3.5. Изменение разницы для расстояния Йенсона-Шеннона для поведения 2.

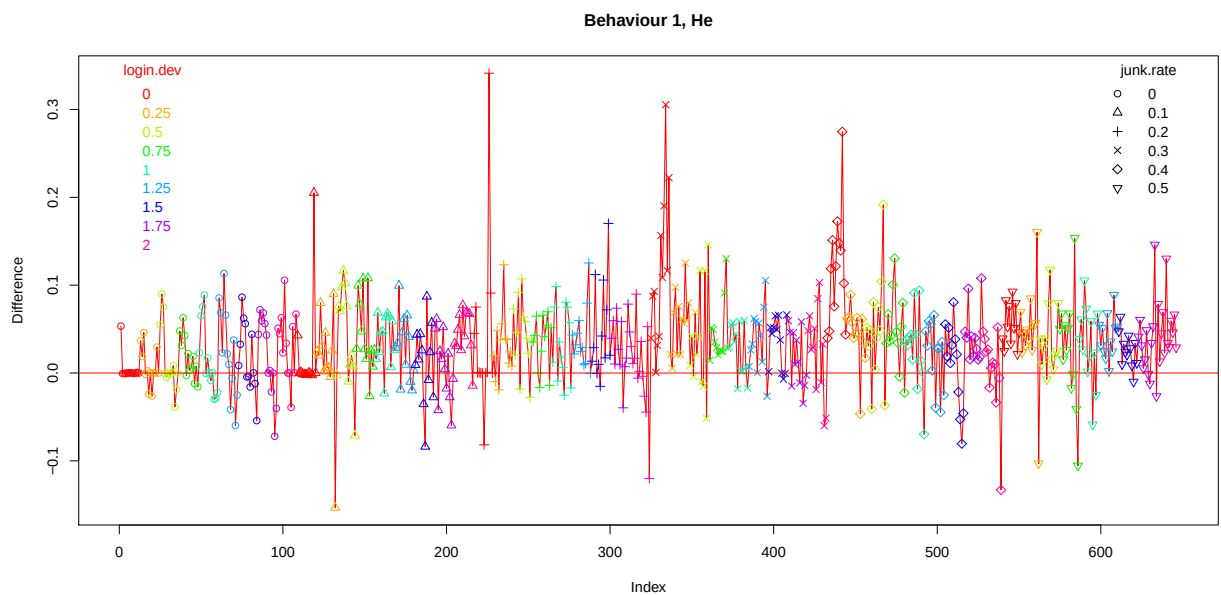


Рис. 3.6. Изменение разницы для расстояния Хеллинджера для поведения 1.

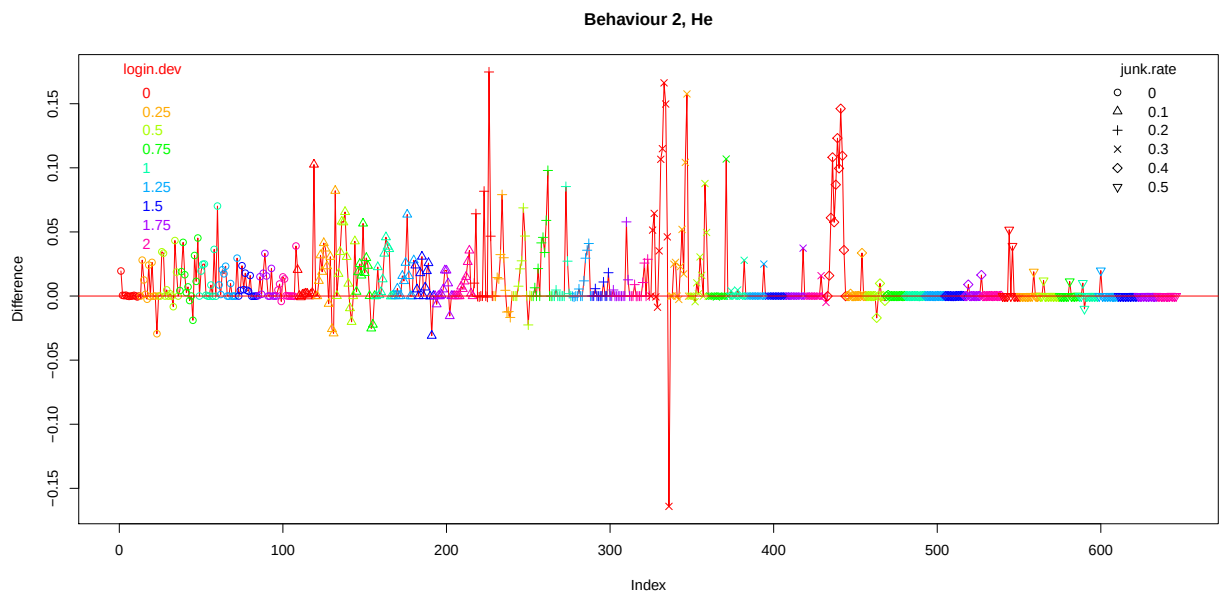


Рис. 3.7. Изменение разницы для расстояния Хеллинджера для поведения 2.

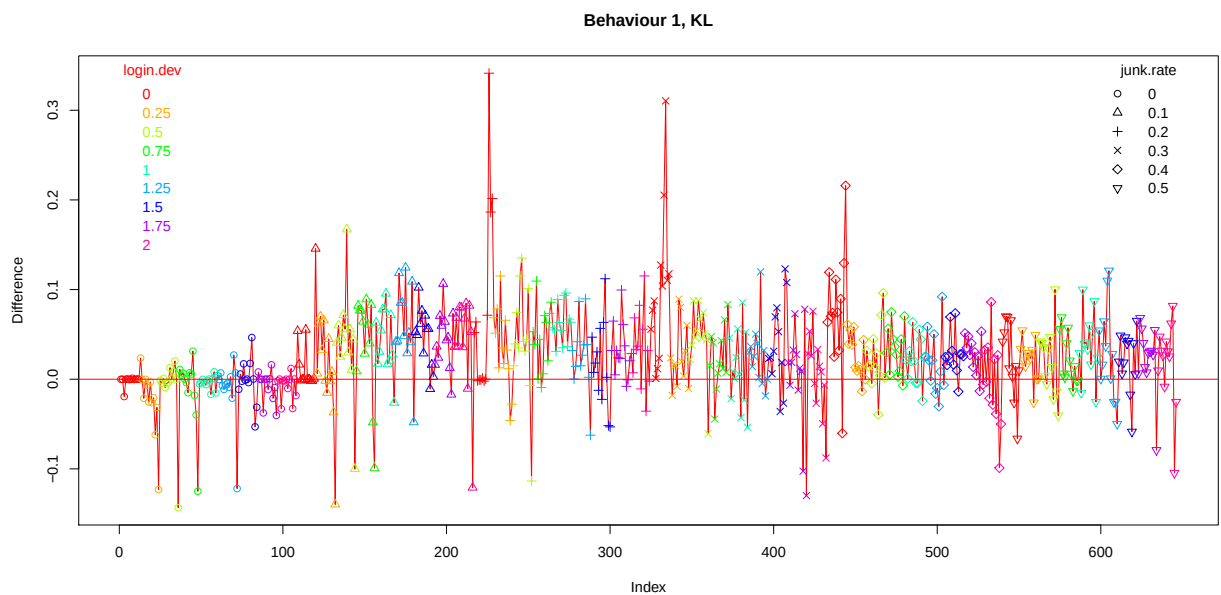


Рис. 3.8. Изменение разницы для расстояния Кульбака-Лейблера для поведения 1.

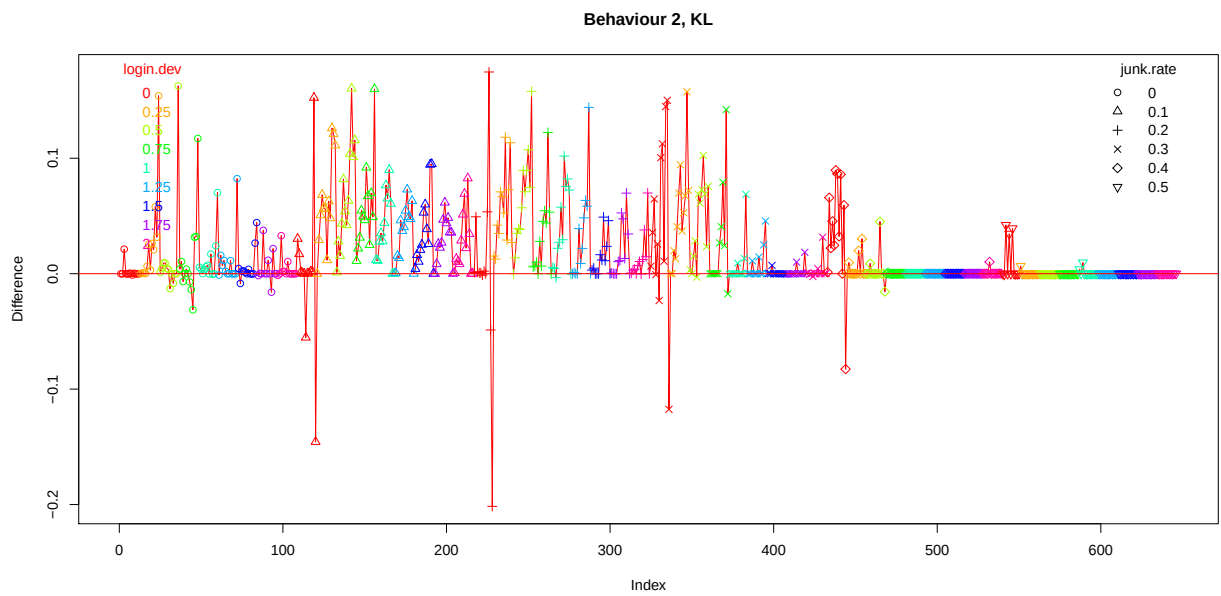


Рис. 3.9. Изменение разницы для расстояния Кульбака-Лейблера для поведения 2.

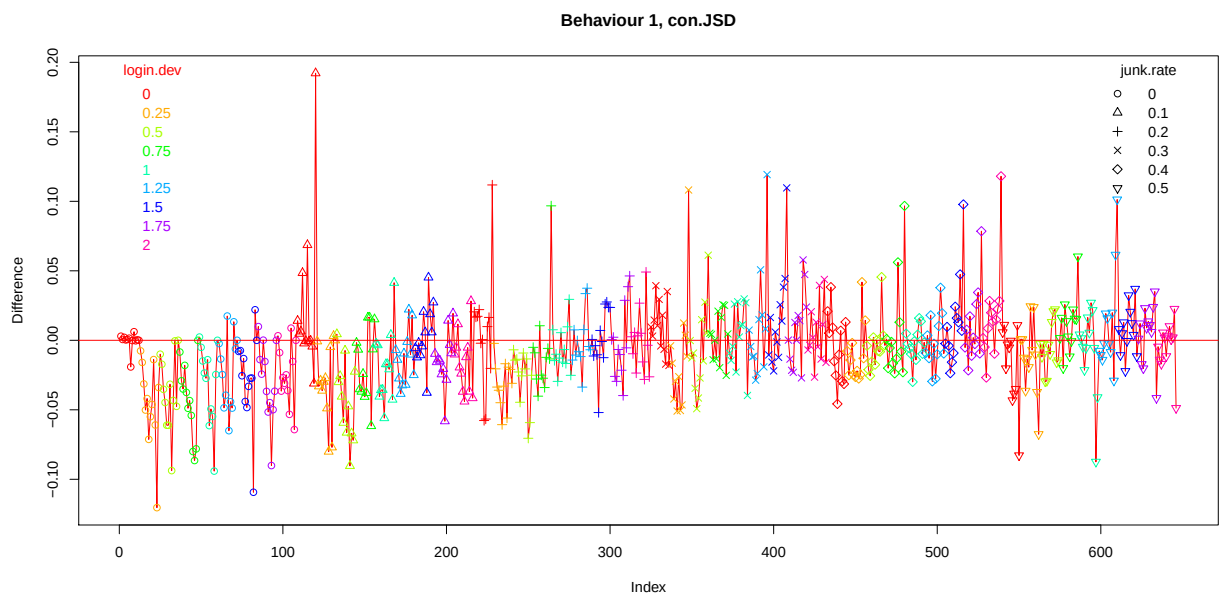


Рис. 3.10. Изменение разницы для условного расстояния Йенсона-Шеннона для поведения 1.

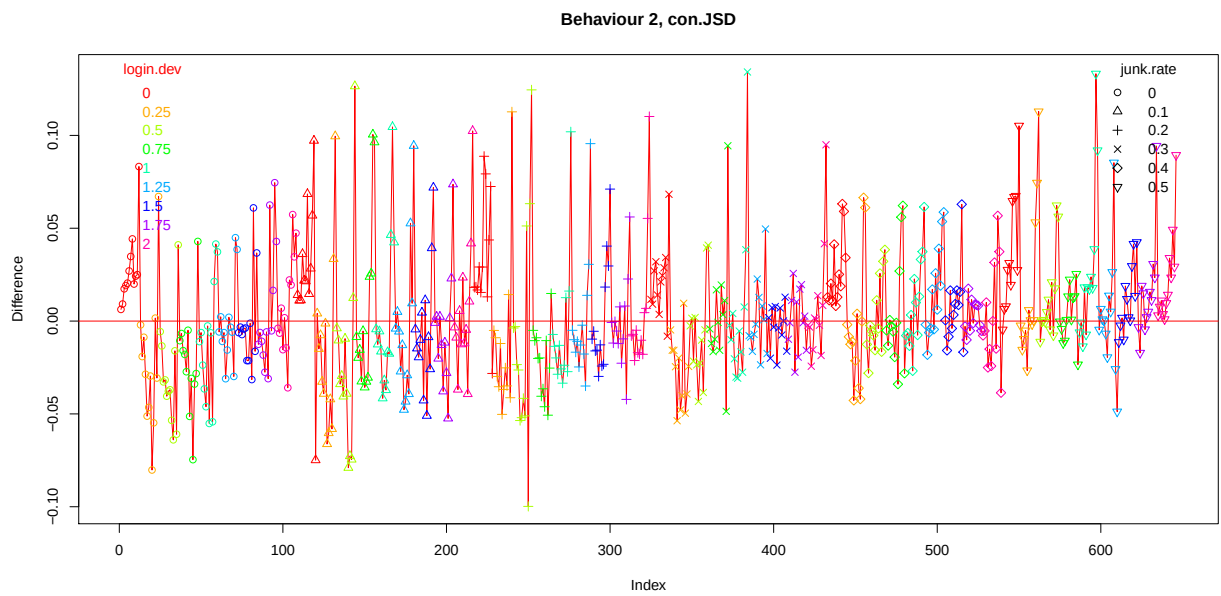


Рис. 3.11. Изменение разницы для условного расстояния Йенсона-Шеннона для поведения 2.

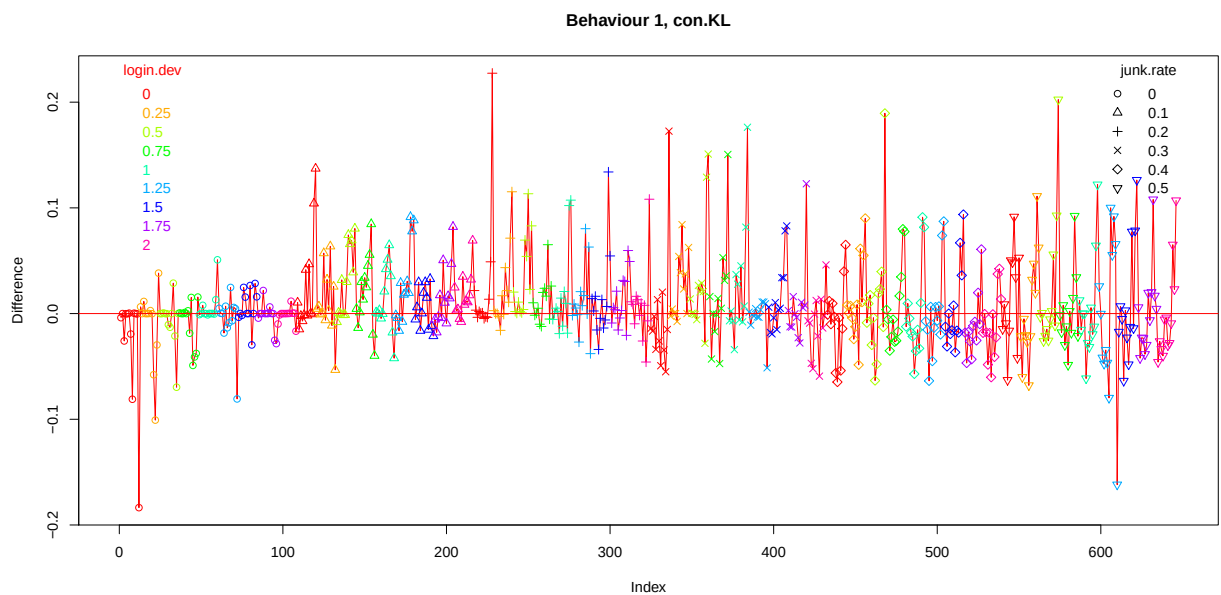


Рис. 3.12. Изменение разницы для условного расстояния Кульбака-Лейблера для поведения 1.

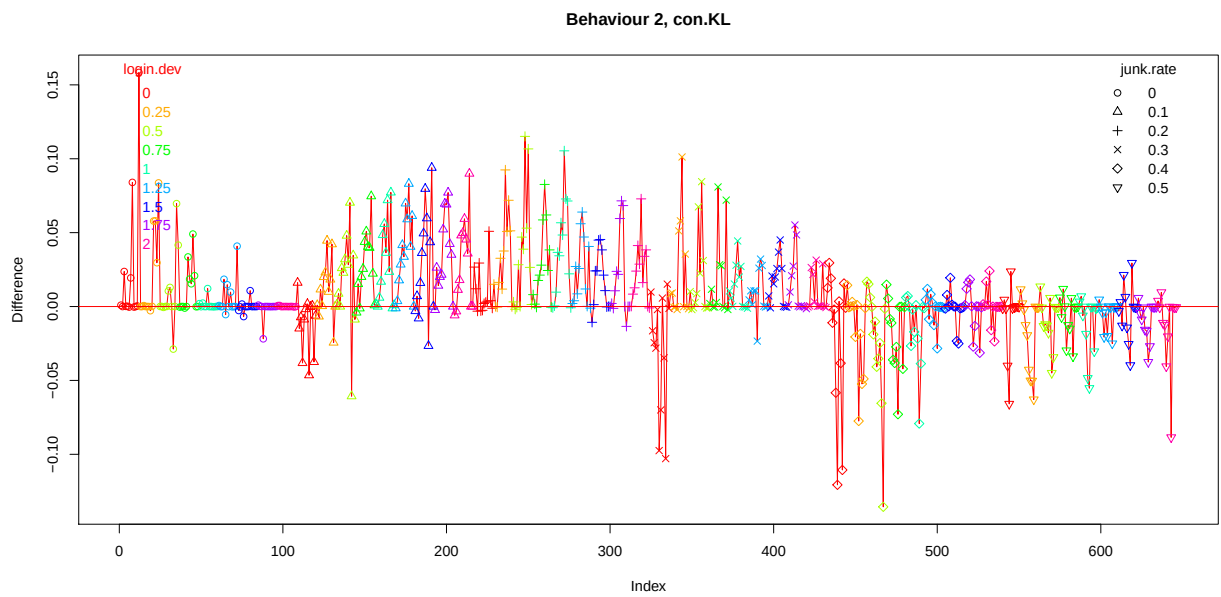


Рис. 3.13. Изменение разницы для условного расстояния Кульбака-Лейблера для поведения 2.

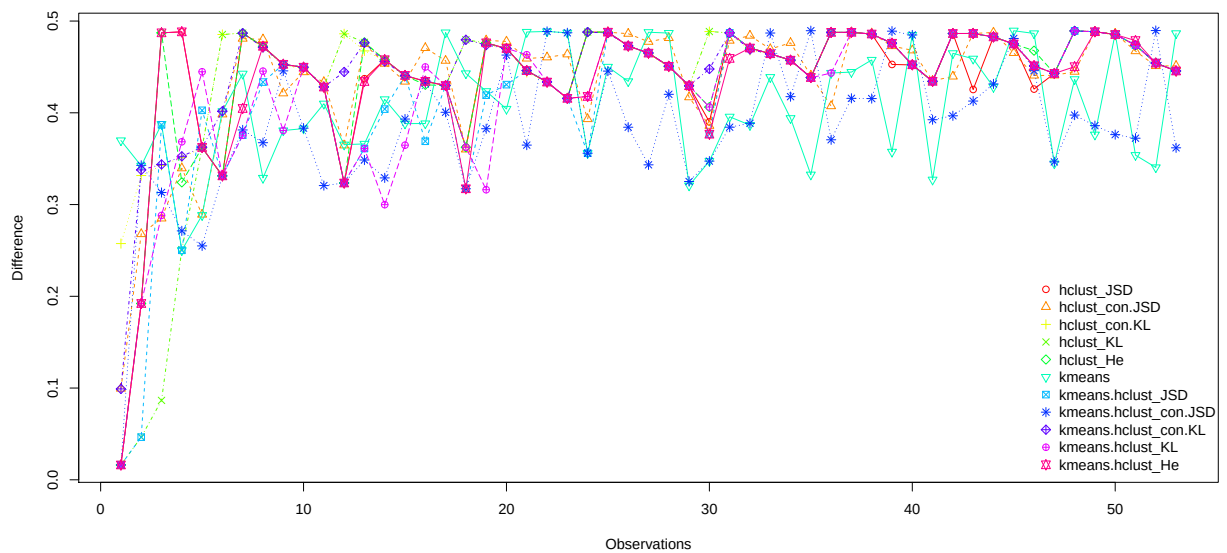


Рис. 3.14. Изменение ошибки в зависимости от параметров login.dev и junk.rate при регулярности 24.

Глава 4

Применение алгоритма к реальным данным

Рассмотрим как работает алгоритм на реальных данных, в качестве которых были взята история авторизаций при заходе в интернет-банковскую систему для одного индивида. Соответственно, имеется массив наблюдений, каждое из которых состоит из пары географических координат и даты со временем. Отметим, что географические координаты получаются из IP адресов, поэтому могут иметь некую неточность. Максимальная разница по времени между наблюдениями в данных составляет 16 дней, но для большей части наблюдений (примерно 70%) эта разница не превышает 20 часов, для половины наблюдений — меньше 11 часов, т.е. данные по своей структуре обладают хорошей регулярностью.

4.1. Кластеризация

Кластеризацию наблюдений проведем с помощью иерархической агломеративной кластеризации с расстоянием на Земном шаре и дальнейшего динамического подрезания дендрограммы, как это описывалось в разделе 2.1.2. Отметим, что результаты алгоритма динамического подрезания существенно зависят от собственных параметров. Сам алгоритм умеет по матрице расстояний между объектами высчитывать так называемые значения параметров по умолчанию, но они могут оказаться не оптимальными.

Если изобразить наблюдения на карте (см. Рисунок 4.1), то видно, что имеется 5 хорошо разделимых областей. Единственное, что пока остается не совсем очевидным, так это сколько наблюдений принадлежат каждой области. Результаты кластеризации можно увидеть на Рисунке 4.2. Поясним, что в итоге получили 4 кластера, каждый из которых состоит из 15, 589, 92 и 35 наблюдений соответственно.

4.2. Выделение периодов

Выделение периодов происходит с использованием перестановочного теста с уровнем значимости, равным 0.05, и при перемешивании последовательности 200 раз. В итоге получаем, что период 168 часов, т.е. одна неделя, обнаруживается в кластере 2,



Рис. 4.1. Наблюдения до кластеризации.

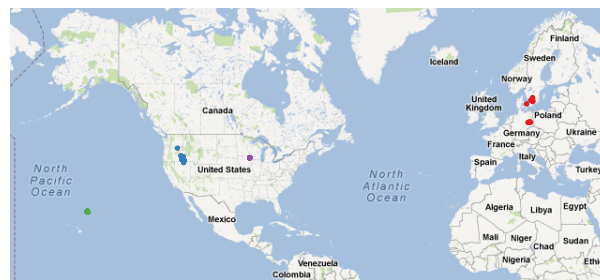


Рис. 4.2. Наблюдения после кластеризации.

период 24 часа — в кластерах 2 и 3. Изображения периодограмм для второго и третьего кластеров можно увидеть на Рисунках 4.3 и 4.4 соответственно.

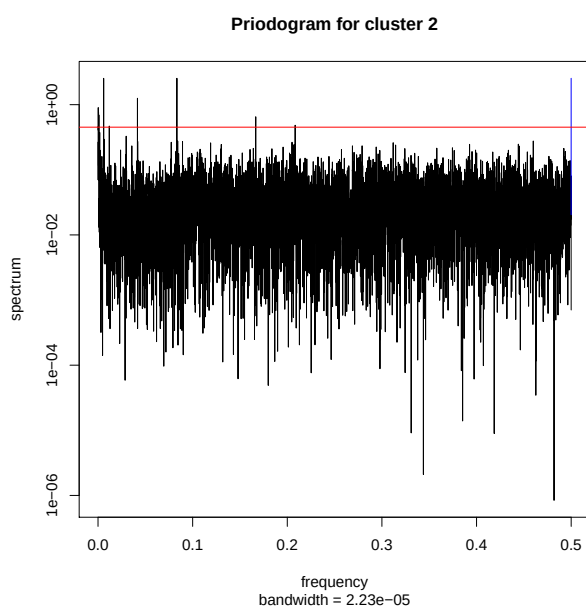


Рис. 4.3. Периодограмма с пороговым значением для второго кластера.

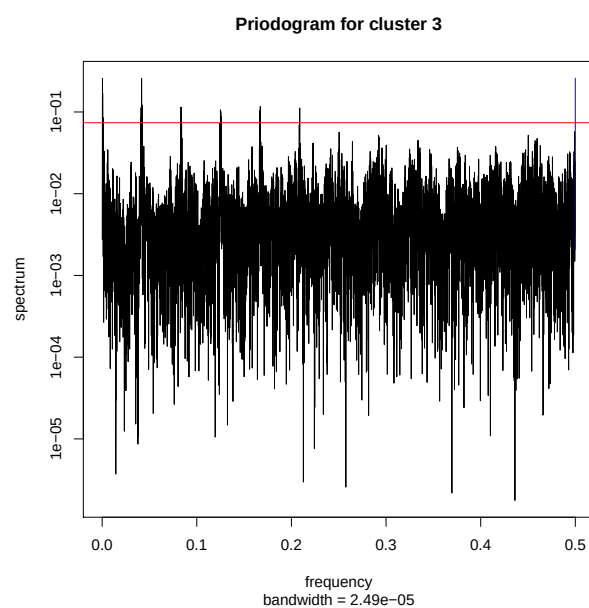


Рис. 4.4. Периодограмма с пороговым значением для второго кластера.

4.3. Определение периодических поведения

Перед тем как браться непосредственно выделять периодические поведения, попробуем проанализировать сами данные: посмотрим на последовательность перемещений (см. Рисунок 4.5) и сами наблюдения, соответствующие кластерам.

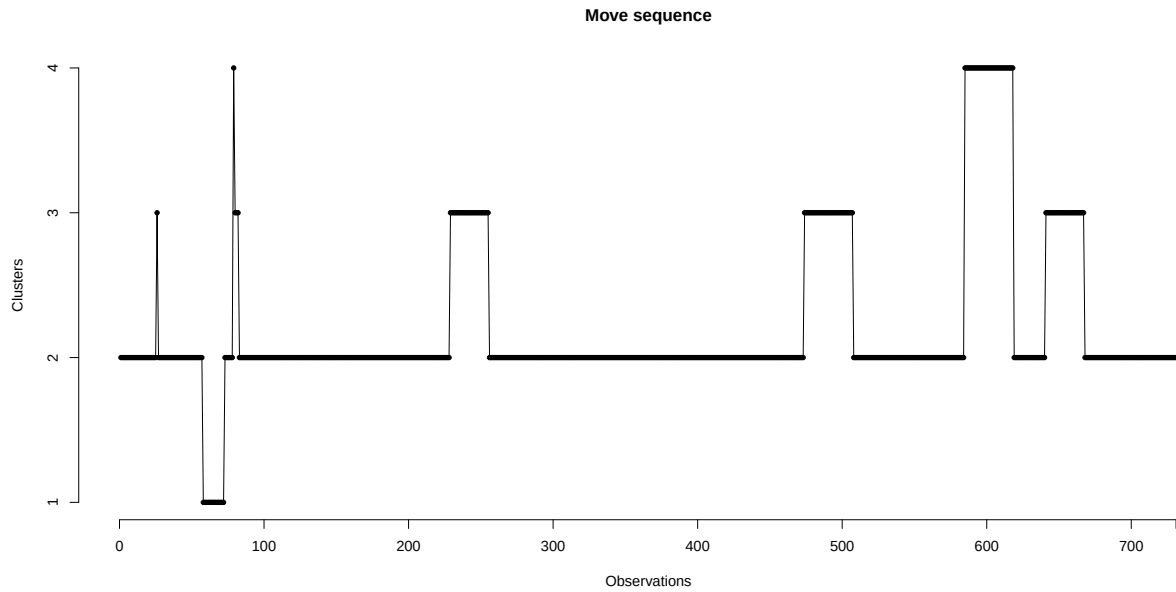


Рис. 4.5. Последовательность перемещений.

Наблюдения, принадлежащие первому кластеру (область Германии и Швеции), происходят в течение трех недель, причем наблюдения весьма нерегулярны — средний промежуток времени между составляют почти 40 часов. Можно предположить, что это или разовая командировка, или отпуск.

Наблюдения, принадлежащие второму кластеру, преобладают, поэтому само собой напрашивается предположение, что это основное место пребывания индивида. Максимальное расстояние между наблюдениями, принадлежащими этому кластеру, составляет 330 км, что означает, что кластер охватывает обширную область, и при такой пространственной кластеризации не получится обнаружить какие-то различные поведения для более мелких локаций как, например, место работы и дом. Именно поэтому этап пространственной кластеризации является, пожалуй, самым важным среди всех этапов алгоритма.

Наблюдения третьего кластера (область Гавайев) особо выделяются в последовательности перемещений трижды и каждый раз происходят в течение примерно одного месяца. Отметим, что регулярность наблюдений меняется от случая к случаю, а именно если смотреть на среднее по промежуткам времени между наблюдениями, то для каждого периода времени, когда наблюдения находятся в этом кластере, они составляют 31 час, 20 и 25 часов соответственно. Наблюдения перемещаются в этот кластер через

3.5 и 2 месяца соответственно, поэтому, скорее всего, имеет место некая командировка индивида, а не отпуск.

Наблюдения, принадлежащие четвертому кластеру, происходят в течение двух недель, что вполне можно интерпретировать как стандартный двухнедельный отпуск, при этом сами наблюдения весьма регулярны — среднее по промежуткам времени между наблюдениями составляет 9 часов.

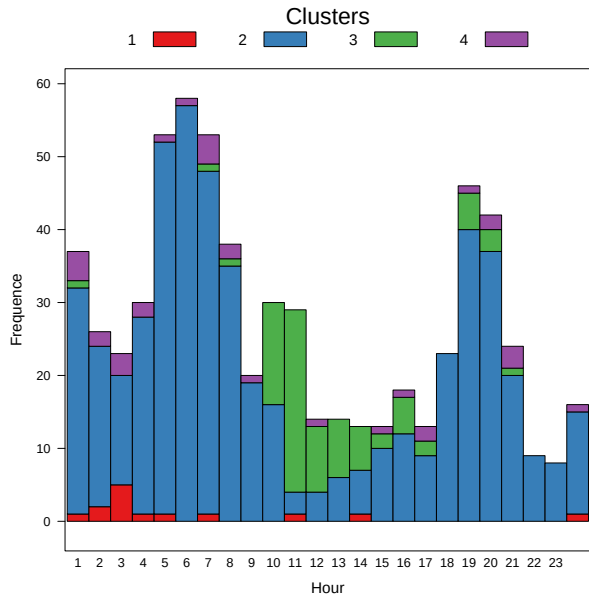


Рис. 4.6. Гистограмма наблюдений по всем кластерам для периода 24 часа.

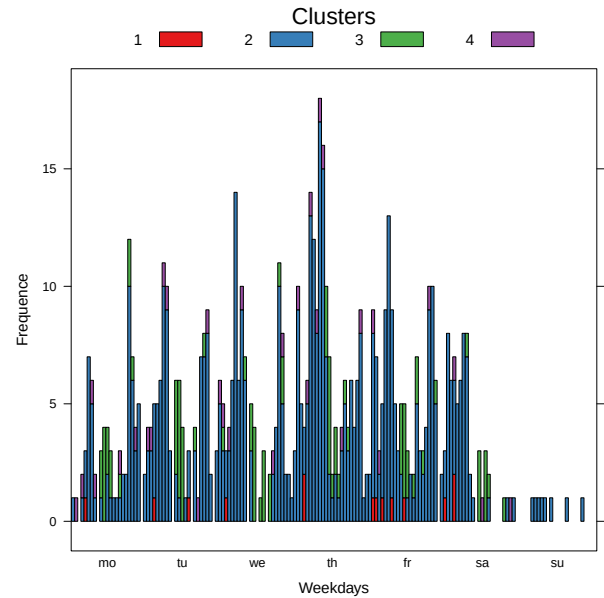


Рис. 4.7. Гистограмма наблюдений по всем кластерам для периода в одну неделю.

Рассмотрим гистограмму наблюдений для периода в 24 часа для всех кластеров (см. Рисунок 4.6). По гистограмме видно, что можно выделить два основных поведения — поведения для второго и третьего кластеров соответственно. Наблюдения первого и четвертого кластеров ближе к наблюдениям второго кластера, поэтому, скорее всего, если говорить о двух поведении, будут принадлежать поведению, соответствующему второму кластеру. Заметим, что так как наблюдения третьего кластера происходят каждый раз только в течение одного месяца с весьма небольшой регулярностью, то они вполне могут «затеряться» среди наблюдений второго кластера при кластеризации распределений. По гистограмме для недельного периода (см. Рисунок 4.7) можно предположить, что есть только одно значимое поведение, которому будут принадлежать все наблюдения. Отметим только два важных момента. Во-первых, поведение для одного

и того же кластера с одним периодом может меняться, и поэтому вполне может получиться так, что поведений для вышеобозначенных периодов окажется больше, чем только что было предположено. Во-вторых, может оказаться так, что в процессе иерархической кластеризации на последних итерациях основные поведения окажутся ближе друг к другу, нежели основное поведения и распределение-выброс, и, как следствие, количество итоговых поведений изменится.

Рассмотрим сначала кластеризацию распределений для периода 24 часа. Не будем описывать результаты всех комбинаций методов и расстояний, а отметим только некоторые из них.

При проверке сочетания метода k -средних и иерархической кластеризации со всеми возможными расстояниями наилучшим оказался метод при использовании расстояния Хеллинджера. Результаты каждого сочетания метода и расстояния сравнивались по тому, как сильно они похожи на то, что ожидалось получить. Если посмотреть на дендрограмму кластеризации распределений (см. Рисунок 4.8), полученных с помощью предварительной кластеризации методом k -средних, то можно видеть, что хорошо выделяются пять кластеров-распределений, два из которых являются своего рода выбросами, т.к. оценены только по трём и двум сегментам. Посмотрим снова на последовательность перемещений, только уже зная принадлежность наблюдений поведениям (см. Рисунок 4.9).

По рисунку видно, что поведения 3 и 5 характерны для третьего кластера, исключая первую группу наблюдений этого кластера. Отметим, что подобная ситуация наблюдалась и при использовании остальных расстояний. Чтобы понять почему так происходит, обратимся к гистограмме частот наблюдений третьего кластера, различая при этом принадлежность наблюдений разным группам (см. Рисунок 4.10). Видно, что в действительности существует различие между первой группой наблюдений и остальными, которое, по всей видимости, и вызывает наблюдаемое явление. О поведении 1 и 2 не будем говорить, т.к. они являются выбросами. Из важного отметим еще то, что получившиеся поведения весьма четко последовательно сменяют друг друга, и это получилось только при использовании расстояния Хеллинджера.

Рассмотрим теперь результаты использования иерархической кластеризации с различными расстояниями без предварительной кластеризации с помощью метода k -средних. Наилучший результат получился при использовании условного расстояния Кульбака-

Cluster Dendrogram with Hellinger

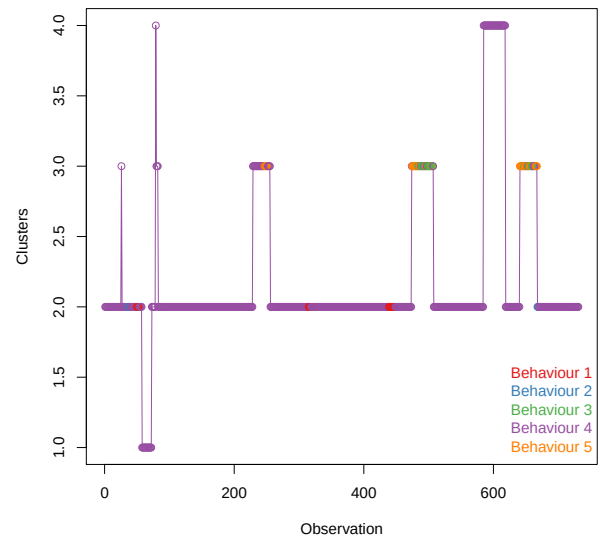
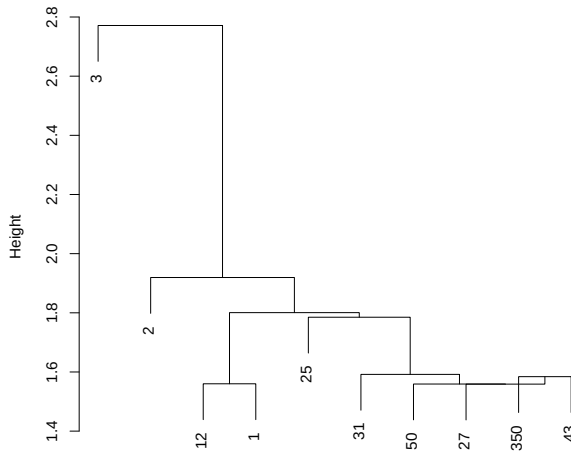


Рис. 4.8. Дендрограмма иерархической кластеризации с использованием расстояния Хеллингера после выполнения предварительной кластеризации методом k -средних.

Рис. 4.9. Последовательность перемещений после выделения периодических поведений.

Лейблера. Из дендрограммы кластеризации (см. Рисунок 4.11) видно, что хорошо выделяются три основных кластера-распределения. Отметим сразу, что первое поведение содержит только 5 сегментов, поэтому является выбросом, и его в расчет брать не будем. Если сразу взглянуть на последовательность перемещений (см. Рисунок 4.12), то видно, что поведения получились ровно такими, как и ожидалось. Важно отметить, что при использовании остальных расстояний результаты получались вовсе неприемлемыми, т.к. по дендрограмме невозможно было увидеть хоть какую-то разумную кластеризацию распределений, потому что на многих последних итерациях происходило объединение некоторого большого кластера (распределения, полученного как оценка по многим сегментам) и кластера, распределение которого было оценено по единственному сегменту.

Осталось рассмотреть работу метода k -средних. Так как методу необходимо задавать то количество поведений, на которое необходимо разделить все распределения, а двумя вышеописанными способами обнаружили 3 и 5 поведений с учетом выбросов, то решено было попробовать применить метод k -средних с количеством итоговых поведе-

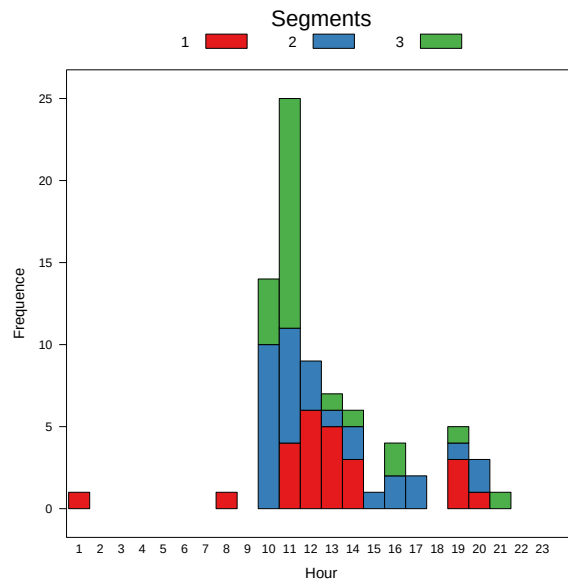


Рис. 4.10. Гистограмма частот наблюдений для кластера 3.

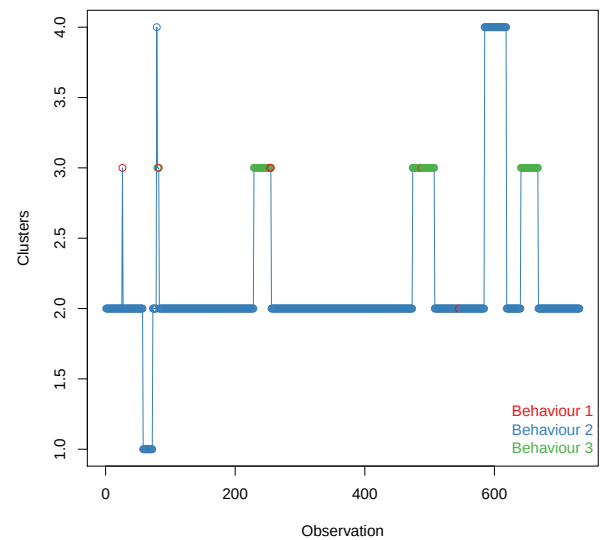
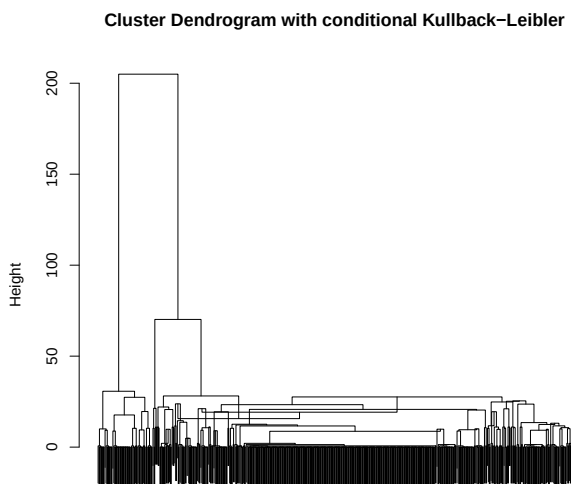


Рис. 4.11. Дендрограмма иерархической кластеризации с использованием условного расстояния Кульбака-Лейблера.

Рис. 4.12. Последовательность перемещений после определения периодических поведений.

ний от 2 до 5. К сожалению, все эксперименты не дали результатов, сколько-нибудь похожих на ожидаемые.

Попробуем сравнить результаты, полученные иерархической кластеризацией с использованием условного расстояния Кульбака-Лейблера и сочетания метода k -средних

и иерархической кластеризации с использованием расстояния Хеллинджера. Основное различие между полученными разными способами результатами состоит в том, к какому поведению относить первую группу наблюдений кластера 3. Обратимся к гистограмме частот наблюдений по всем кластерам для периода 24 часа (см. 4.6) Видно, что главное отличие между наблюдениями второго и третьего кластера в промежуток времени от 10 до 14 часов (тогда, когда есть наблюдения для обоих кластеров) происходит в 11 часов. Если обратиться к гистограмме частот наблюдений для третьего кластера (см. 4.10), то видно, что основная часть наблюдений, происходящих в 11 часов, — это наблюдения из второй и третьей групп. Скорее всего, именно из-за этого наблюдения первой группы были отнесены к другому поведению при использовании сочетания метода k -средних и иерархической кластеризации с расстоянием Хеллинджера. Уверенно утверждать, какой из результатов лучше, нельзя, потому что наблюдений для третьего кластера мало.

Для периода в 168 часов никакие из комбинаций метод-расстояние не показали, что существует больше одного поведения.

4.3.1. Итоги

Подведем итоги применения различных методов к реальным данным:

1. Базовый алгоритм без модификаций не дал результатов, хоть сколько-нибудь похожих на ожидаемые.
2. Для иерархической кластеризации приемлемые результаты получились только при использовании условного расстояния Кульбака-Лейблера.
3. Для сочетания метода k -средних и иерархической кластеризации наилучший результат получен при использовании расстояния Хеллинджера, но использование расстояний Кульбака-Лейблера и Йенсона-Шеннона также приводит к разумным результатам.
4. Метод k -средних так же, как и базовый алгоритм, не привел к приемлемым результатам.

Заключение

В работе был исследован алгоритм из статьи [1], который несмотря на ожидания, дал не совсем хорошие результаты при применении его к реальным данным. В связи с этим был предложен альтернативный метод выполнения этапа пространственной кластеризации и некоторые модификации этапа определения периодических поведений.

Также было выполнено исследование качества работы начального алгоритма, т.е. алгоритма из статьи [1], и алгоритма с предложенными модификациями на модельных данных разного качества. Было установлено, что предложенные модификации, исключая применение условных расстояний, в подавляющем большинстве комбинаций параметров, влияющих на качество данных, для которых выполняется алгоритм, приводят к результатам, которые лучше, чем результаты полученные начальным алгоритмом.

В математической среде R были реализованы:

1. начальный алгоритм;
2. предложенные модификации;
3. моделирование данных;
4. параллельное выполнение всех модификаций и начального алгоритма для одного массива данных;
5. визуализация результатов моделирования.

Так же был рассмотрен пример выполнения алгоритма с модификациями для реальных данных.

В дальнейшем планируется предложить иные методы определения периодических поведений без использования кластеризации распределений, а также модификации метода, не требующие регуляризации данных.

Список литературы

1. Mining periodic behaviors for moving objects / Zhenhui Li, Bolin Ding, Jiawei Han et al. // Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. — KDD '10. — New York, NY, USA : ACM, 2010. — P. 1099–1108.
2. Kaufman L., Rousseeuw P. J. Finding Groups in Data. An Introduction to Cluster Analysis. — Wiley-Interscience, 1995.
3. Langfelder P., Zhang B., Horvath S. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r // [Bioinformatics](#). — 2008. — Vol. 24, no. 5. — P. 719–720.
4. Sinnott R. W. Virtues of the haversine // *Sky and Telescope*. — 1984. — Vol. 68, no. 2. — P. 159.
5. Brockwell P. J., Davis R. A. Time Series: Theory and Methods. — USA : Springer, 2006.
6. Davis R. A., Mikosch T. The maximum of the periodogram of a non-gaussian sequence // *The Annals of Probability*. — 1999. — Vol. 27, no. 1. — P. 522–536.
7. Lin J. Divergence measures based on the shannon entropy // [IEEE Transactions on Information Theory](#). — 1991. — Vol. 37, no. 1. — P. 145–151.
8. Chaudhuri K., McGregor A. Finding metric structure in information theoretic clustering // *COLT*. — 2008. — P. 391–402.
9. Qiu W., Joe. H. — clusterGeneration: random cluster generation (with specified degree of separation), 2012. — R package version 1.2.9. URL: <http://CRAN.R-project.org/package=clusterGeneration>.
10. R Development Core Team. — R: A Language and Environment for Statistical Computing. — R Foundation for Statistical Computing, Vienna, Austria, 2011. — ISBN 3-900051-07-0. URL: <http://www.R-project.org/>.
11. Revolution Analytics. — foreach: Foreach looping construct for R, 2011. — R package version 1.3.2. URL: <http://CRAN.R-project.org/package=foreach>.
12. Revolution Analytics. — doMC: Foreach parallel adaptor for the multicore package, 2012. — R package version 1.2.5. URL: <http://CRAN.R-project.org/package=doMC>.

Приложение А

Обозначения для параметров

В таблице [A.1](#) приведены соотношения между параметрами, используемыми в Главе [3](#) и параметрами, используемыми в программе.

Глава 3	Программа
r	<code>regularity</code>
τ	<code>junk.rate</code>
σ_t	<code>login.dev</code>
N_{na}	<code>na.size</code>
N_{out}	<code>num.outlier</code>
N_{obs}	<code>observ.period</code>
$I_{per}(a, b)$	<code>range.known.obs</code>

Таблица А.1. Обозначения для параметров, используемых в программе.

Приложение Б

Реализация алгоритма

Модуль “Global_Source.r”

```

1 spatial.clusterify.kmeans <- function(data, centers = 7) {
2   kmeans(data[, c("latitude", "longitude")],
3           centers = centers)$cluster
4 }
5
6 spatial.clusterify.pam <- function(data, k = 11) {
7   require(cluster)
8   pam(data[, c("latitude", "longitude")], k = k)$clustering
9 }
10
11 spatial.clusterify.clara <- function(data, k = 11) {
12   require(cluster)
13   clara(data[, c("latitude", "longitude")], k = k,
14          samples = nrow(data) / 11)$clustering
15 }
16
17 earth.dist2 <- function (lats, dist = TRUE){
18   require("fossil")
19   if (class(lats) == "SpatialPoints")
20     lats <- coordinates(lats)
21   name <- list(rownames(lats), rownames(lats))
22   n <- nrow(lats)
23   z <- outer(1:n, 1:n, function(i, j)
24     deg.dist(long1 = lats[i, 1],
25              lat1 = lats[i, 2], long2 = lats[j, 1],
26              lat2 = lats[j, 2]))
27   if (dist == TRUE)
28     z <- as.dist(z)
29   return(z)
30 }
31
32 spatial.clusterify.earth.hclust <- function(data){
33   require("dynamicTreeCut")

```



```

73                                     fast = fast)$spec))
74   spc.data <- spec.pgram(cluster.data, plot = FALSE, taper = taper,
75                           fast = fast,
76                           main = paste("Priodogram_ for _ cluster", cluster,
77                                         sep = "_"))
78   quan <- quantile(spc.tmp, probs = (1-alpha), names = FALSE)
79   idx <- which(spc.data$spec > quan)
80   r <- (1/spc.data$freq[idx])[order(spc.data$spec[idx],
81                                   decreasing = TRUE)]
82   if(rounds) r <- round(r)
83   return(list(periodogram = spc.data, periods = unique(r)))
84 }
85
86 # Периодограммы возможные периоды для всех кластеров
87 all.periods.periodogram <- function(movement,
88                                     periods.fun = periods.spec.pgram,
89                                     nrep = 200, alpha = 0.05){
90   clusters <- sort(unique(movement[!is.na(movement)]))
91   res <- lapply(clusters, periods.fun, movement = movement, nrep = nrep,
92               alpha = alpha)
93   names(res) <- clusters
94   res
95 }
96
97 # В каких кластерах период
98 in.which.clusters <- function(period, list.periods){
99   period.in.clusters <- sapply(list.periods,
100                               function(x) period %in% x)
101   names(period.in.clusters)[which(period.in.clusters)]
102 }
103
104 # Последовательность передвижений
105 move.seq.for.period <- function(period, list.periods, movement){
106   clusters <- in.which.clusters(period, list.periods)
107   idx <- movement %in% clusters
108   res <- numeric(length(movement))
109   res[idx] <- movement[idx]
110   res
111 }

```

```

112
113 # Матрица для j-го сегмента последовательности S и периода T
114 prob.matrix <- function(j,S,T){
115   names <- sort(unique(S))
116   d <- length(names)
117   segment <- S[((j-1)*T+1):(j*T)]
118   rez <- matrix(0, nrow = d, ncol = T)
119   rownames(rez) <- names
120   colnames(rez) <- 1:T
121   rez[cbind(as.character(segment), as.character(1:T))] <- 1
122   attributes(rez)$num.of.obs <- 1
123   attributes(rez)$consist <- j
124   rez
125 }
126
127 # Количество наблюдений в матрице
128 nobs <- function(M) attributes(M)$num.of.obs
129
130 # Содержание матрицы
131 consist <- function(M) attributes(M)$consist
132
133 # Стадия матрицы
134 stage <- function(M) attributes(M)$stage
135
136 # Высота
137 height <- function(M) attributes(M)$height
138
139 mtrx.merge <- function(M) attributes(M)$merge
140
141 # Для merge
142 for.merge <- function(M){
143   ifelse(is.null(attr(M, "stage")), -consist(M), stage(M))
144 }
145
146 # Список всех матриц
147 list.matrix.segments <- function(S,T){
148   m <- length(S)/%T
149   lapply(1:m, prob.matrix, S, T)
150 }

```

```

151
152 # "Нормирование" вероятностной матрицы с параметром lambda
153 lambda.norm.prob <- function(M, lambda = 1e-9){
154   (1-lambda*nrow(M))*M + lambda
155 }
156
157 # "Нормирование" частотной матрицы с параметром lambda
158 lambda.norm.freq <- function(M, lambda = 1e-9){
159   (1-lambda*nrow(M))*M + lambda*nobs(M)
160 }
161
162 # Расстояние Кульбака-Лейблера через вероятности
163 prob.KL <- function(P, Q, lambda.norm = lambda.norm.prob, lambda = 1e-9){
164   P1 <- lambda.norm(P, lambda = lambda)
165   Q1 <- lambda.norm(Q, lambda = lambda)
166   sum(P1*log(P1/Q1))
167 }
168
169 # Расстояние Кульбака-Лейблера через частоты
170 freq.KL <- function(P, Q, lambda = 1e-9){
171   prob.KL(P, Q, lambda.norm.freq, lambda = lambda)/nobs(P) -
172     ncol(P)*log(nobs(P)/nobs(Q))
173 }
174
175 # Расстояние Йёнсона-Шеннона
176 JSD <- function(P, Q, lambda = 1e-9){
177   n <- nobs(P)
178   m <- nobs(Q)
179   R <- lambda.norm.prob((P*m + Q*n)/(2*n*m), lambda)
180   P1 <- lambda.norm.prob(P/nobs(P), lambda)
181   Q1 <- lambda.norm.prob(Q/nobs(Q), lambda)
182   (sum(P1*log(P1/R) + Q1*log(Q1/R)))/2
183 }
184
185 freq.JSD <- function(P, Q, lambda = 1e-9){
186   n <- nobs(P)
187   m <- nobs(Q)
188   R <- m*P + n*Q
189   (round(ncol(P)*log(4*n*m) + (sum(P*log(P/R), na.rm = TRUE))/n +

```

```

190     (sum(Q*log(Q/R), na.rm = TRUE))/m, 10))/2
191 }
192
193 # Расстояние Хеллинджера
194 hellinger.dist <- function(P, Q, lambda = 1e-9){
195     sqrt(sum((sqrt(P/nobs(P)) - sqrt(Q/nobs(Q)))^2))
196 }
197
198 # Условное с KL
199 freq.KAI.KL <- function(P, Q, lambda = 1e-9){
200     P1 <- lambda.norm.freq(P, lambda)
201     P1 <- t(t(P1[-1,])/(nobs(P1) - P1[1,]))
202     Q1 <- lambda.norm.freq(Q, lambda)
203     Q1 <- t(t(Q1[-1,])/(nobs(Q1) - Q1[1,]))
204     sum(P1*log(P1/Q1))
205 }
206
207 # Условное с JSD
208 freq.KAI.JSD <- function(P, Q, lambda = 1e-9){
209     n <- nobs(P)
210     m <- nobs(Q)
211     P1 <- lambda.norm.freq(P, lambda)
212     P1 <- t(t(P1[-1,])/(n - P1[1,]))
213     Q1 <- lambda.norm.freq(Q, lambda)
214     Q1 <- t(t(Q1[-1,])/(m - Q1[1,]))
215     R1 <- m*P1 + n*Q1
216     sqrt(round(ncol(P1)*log(4*n*m) + (sum(P1*log(P1/R1), na.rm = TRUE))/n +
217         (sum(Q1*log(Q1/R1), na.rm = TRUE))/m, 10))
218 }
219
220 # Попарные расстояния
221 pairwise.dist <- function(list.matrix, dist.func = freq.KL,
222                             lambda = 1e-9){
223     m <- length(list.matrix)
224     res <- diag(+Inf, nrow = m)
225     for (i in 1:m){
226         for (j in (1:m)[-i]){
227             res[i,j] <- dist.func(list.matrix[[i]],list.matrix[[j]],
228                                   lambda = lambda)

```

```

229     }
230 }
231 res
232 }
233
234 # Объединение списка матриц через частоты
235 merge.list.matrix.freq <- function(list.matrix){
236   temp <- simplify2array(list.matrix)
237   res <- apply(temp, c(1,2), sum)
238   attributes(res)$num.of.obs <- sum(sapply(list.matrix,
239                                           function(m) nobs(m)))
240   attributes(res)$consist <- unlist(lapply(list.matrix,
241                                           function(m) consist(m)))
242   if (length(list.matrix) == 2){
243     attributes(res)$merge <- c(for.merge(list.matrix[[1]]),
244                               for.merge(list.matrix[[2]]))
245   } else attributes(res)$merge <- c(0,0)
246   res
247 }
248
249 # Объединение двух матриц через вероятности
250 merge.matrix.prob <- function(P, Q){
251   n <- nobs(P)
252   m <- nobs(Q)
253   res <- (n*P + m*Q)/(n+m)
254   attributes(res)$num.of.obs <- n + m
255   attributes(res)$consist <- c(consist(P),consist(Q))
256   res
257 }
258
259 # Добавление в список матриц
260 add.list.matrix <- function(list.matrix, indx.segments, height = 0,
261                             merge.matrix = merge.list.matrix.freq){
262   n <- length(list.matrix) + 1
263   list.matrix[[n]] <- merge.matrix(list.matrix[indx.segments])
264   attributes(list.matrix[[n]])$height <- height
265   list.matrix[indx.segments] <- NULL
266   list.matrix
267 }

```

```

268
269 # Пересчет матрицы расстояний
270 recalc.dist.matrix <- function(matrix.dist, indx.segments, list.matrix,
271                                dist.func = freq.KL, lambda = 10^(-9)){
272   matrix.dist <- matrix.dist[-indx.segments,-indx.segments, drop = FALSE]
273   n <- length(list.matrix)
274   if (n > 1) {
275     matrix.dist <- rbind(matrix.dist,
276                           sapply(1:(n-1), function(j)
277                                dist.func(list.matrix[[n]], list.matrix[[j]],
278                                           lambda = lambda)))
279     matrix.dist <- cbind(matrix.dist,
280                           c(sapply(1:(n-1), function(j)
281                                dist.func(list.matrix[[j]], list.matrix[[n]],
282                                           lambda = lambda)), +Inf))
283   }
284   else matrix.dist <- +Inf
285   matrix.dist
286 }
287
288 # Объединение сегментов
289 merge.segments <- function(matrix.dist, list.matrix, dist.func = freq.KL,
290                             merge.matrix = merge.list.matrix.freq,
291                             lambda = 1e-9){
292   m <- nrow(matrix.dist)
293   min.dist <- min(matrix.dist)
294   vec.idx <- which.min(matrix.dist)[1]
295   idx <- vec.idx %% m
296   if (idx == 0) idx <- m
297   group.idx <- c(idx, which(equal(matrix.dist[idx,], min.dist)))
298   merge.idx <- group.idx[1:2]
299   list.matrix <- add.list.matrix(list.matrix, merge.idx,
300                                 height = min.dist, merge.matrix)
301   matrix.dist <- recalc.dist.matrix(matrix.dist, merge.idx, list.matrix,
302                                     dist.func, lambda)
303   list(dist = matrix.dist, lmx = list.matrix)
304 }
305
306 # Representation Error

```

```

307 # Ошибка по одному сегменту
308 rep.err.ones <- function(segment, pmatrix, type.error = "mean"){
309   d <- nrow(pmatrix)
310   cols <- which(segment != 0)
311   rows <- match(segment[cols], rownames(pmatrix))
312   idx <- d*(cols - 1) + rows
313   if (type.error %in% c("mean", "max")){
314     return (c(sum(1 - pmatrix[idx]/nobs(pmatrix)), length(cols)))
315   } else if (identical(type.error, "likelihood")){
316     return (sum(log(pmatrix[idx]/nobs(pmatrix))))
317   } else stop(paste("Invalid_type_error:", type.error))
318 }
319
320 # Ошибка по одному кластеру
321 rep.err.clus <- function(pmatrix, S, T, type.error = "mean"){
322   rez <- sapply(consist(pmatrix), function(j){
323     segment <- S[((j-1)*T+1):(j*T)]
324     rep.err.ones(segment, pmatrix, type.error = type.error)
325   })
326   if (type.error %in% c("mean", "max")){
327     rez <- ifelse(all(rez[2,] == 0), 0, sum(rez[1,])/sum(rez[2,]))
328     return(rez)
329   } else if (identical(type.error, "likelihood")){
330     return (sum(rez))
331   } else stop("Invalid_type_error!")
332 }
333
334 # Ошибка кластеризации
335 rep.err <- function(list.matrix, S, T, type.error = "mean"){
336   rez <- sapply(list.matrix, rep.err.clus, S = S, T = T,
337     type.error = type.error)
338   if (identical(type.error, "mean"))
339     return(mean(rez))
340   if (identical(type.error, "max"))
341     return(max(rez))
342   else if (identical(type.error, "likelihood"))
343     return(sum(rez))
344   else stop("Invalid_type_error!")
345 }

```

```

346
347 # Объединение всех сегментов последовательно и подсчёт ошибки
348 merge.and.error <- function(list.matrix, T, S, dist.func = freq.KL,
349                               merge.matrix = merge.list.matrix.freq,
350                               lambda = 1e-9, type.error = "mean"){
351   matrix.dist <- pairwise.dist(list.matrix, dist.func = dist.func,
352                                 lambda = lambda)
353   rez <- list()
354   i <- 1
355   while(length(list.matrix) > 1){
356     temp <- merge.segments(matrix.dist, list.matrix,
357                             dist.func = dist.func,
358                             merge.matrix = merge.matrix, lambda = lambda)
359     matrix.dist <- temp$dist
360     attributes(temp$lmx[[length(temp$lmx)]])$stage <- i
361     list.matrix <- temp$lmx
362     rez[[i]] <- list(lmx = list.matrix,
363                      error = rep.err(list.matrix, S, T, type.error))
364     i <- i + 1
365   }
366   rez
367 }
368
369 # Только объединение без ошибки
370 merge.clusters <- function(list.matrix, dist.func = freq.KL,
371                              merge.matrix = merge.list.matrix.freq,
372                              lambda = 1e-9){
373   matrix.dist <- pairwise.dist(list.matrix, dist.func = dist.func,
374                                 lambda = lambda)
375   rez <- list()
376   i <- 1
377   while(length(list.matrix) > 1){
378     temp <- merge.segments(matrix.dist, list.matrix,
379                             dist.func = dist.func,
380                             merge.matrix = merge.matrix,
381                             lambda = lambda)
382     matrix.dist <- temp$dist
383     attributes(temp$lmx[[length(temp$lmx)]])$stage <- i
384     list.matrix <- temp$lmx

```

```

385     rez[[i]] <- list(lmx = list.matrix)
386     i <- i + 1
387   }
388   rez
389 }
390
391 # Data frame для barchart'a из матрицы поведения
392 behbar <- function(behmatrix, freq = TRUE){
393   tm <- t(behmatrix)
394   if (freq) prob <- as.vector(tm)/nobs(behmatrix)
395   else prob <- as.vector(tm)
396   time <- rep(rownames(tm), ncol(tm))
397   time <- as.factor(as.numeric(time))
398   cluster <- numeric()
399   for(i in 1:ncol(tm)) cluster <- c(cluster,
400                                     rep(as.numeric(colnames(tm))[i],
401                                     nrow(tm)))
402   cluster <- as.factor(cluster)
403   data.frame(prob = prob, cluster = cluster, time = time)
404 }
405
406 # Representation error us behaviour'a
407 rep.error <- function(beh){
408   res <- lapply(beh$history, function (x) {
409     sapply(x, function(x) x$error)
410   })
411   names(res) <- names(beh$history)
412   res
413 }
414
415 time.identity <- function(tdata, period){
416   if(period == "week") {
417     res <- as.numeric(format(tdata, "%w"))
418     if(any(res == 0)) res[res == 0] <- 7
419     res <- (res - 1)*24 + as.numeric(format(tdata, "%H"))
420   }
421   if(period == "day") res <- as.numeric(format(tdata, "%H"))
422   if(period == "12") res <- as.numeric(format(tdata, "%I"))
423   res

```

```

424 }
425
426 # Гистограмма частот для наблюдений, принадлежащих cluster
427 plot.periodic <- function(data, cluster, period = c("week", "day", "12")){
428   period <- match.arg(period)
429   if(period == "week") {
430     time.format <- "%Y-%m-%d_%w%H"
431     breaks <- 0:168
432     nt <- 168
433     at <- seq(from = 0, by = 12, length = 14)
434     labels <- c("", "mo", "", "tu", "", "we", "", "th", "", "fr", "", "sa", "", "su")
435     xlab <- "Weekdays"
436   }
437   if(period == "day") {
438     time.format <- "%Y-%m-%d_%H"
439     breaks <- 0:24
440     at <- 0:23
441     labels <- 0:23
442     nt <- 24
443     xlab <- "Hours"
444   }
445   if(period == "12") {
446     time.format <- "%Y-%m-%d_%I"
447     breaks <- 0:12
448     at <- 0:12
449     labels <- 0:12
450     nt <- 12
451     xlab <- "Hours"
452   }
453   n <- length(cluster)
454   if (n == 1) { main <- paste("Cluster", cluster, sep = "_")
455   } else { main <- paste("Clusters", cluster, sep = "_")
456   }
457   df <- data.frame(time = numeric(0), cluster = numeric(0),
458                   freq = numeric(0))
459   for (cl in cluster){
460     times <- data$AuthenticationDate[data$clind == cl]
461     gluing <- strptime(unique(format(times, time.format)),
462                       time.format, tz = "GMT")

```

```

463     identity <- table(time.identity(gluing, period))
464     if(length(identity) != nt){
465         time <- 0:(nt-1)
466         cluster <- rep(cl, nt)
467         freq <- numeric(nt)
468         freq[as.numeric(names(identity)) + 1] <- as.numeric(identity)
469     } else {
470         time <- as.numeric(names(identity))
471         cluster <- rep(cl, length(identity))
472         freq <- as.numeric(identity)
473     }
474     tdf <- data.frame(time = time, cluster = cluster, freq = freq)
475     df <- merge(df, tdf, all = TRUE)
476     df <- df[order(df$time),]
477 }
478 df$cluster <- as.factor(df$cluster)
479 df$time <- as.factor(df$time)
480 require(RColorBrewer)
481 plot(barchart(freq ~ time, data = df, groups = cluster, stack = TRUE,
482             ylab = "Frequence", xlab = xlab, box.ratio = 1000,
483             par.settings = list(
484                 superpose.polygon = list(col = brewer.pal(n,"Set1"))),
485             auto.key = list(title = "Clusters", columns = n),
486             scales = list(x = list(at = at, labels = labels))))
487 }
488
489 # Рисование матрицы поведения
490 plot_beh <- function(behmatrix, sub, clust.method, freq = TRUE,...){
491     require('RColorBrewer')
492     data <- behbar(behmatrix, freq)
493     data <- data[data$cluster != 0,]
494     data$cluster <- droplevels(data$cluster)
495     n <- length(levels(data$cluster))
496     if (!missing(sub)) {
497         sub <- paste("Behavior", sub, sep = "_")
498         if (!missing(clust.method))
499             sub <- paste(sub, clust.method, sep = "_")
500     }
501     else sub <- NULL

```

```
502 dev.new()
503 plot(barchart(prob ~ time, data = data, groups = cluster, stack = TRUE,
504             ylab = "Probability", xlab = "Hour", box.ratio = 1000,
505             par.settings = list(superpose.polygon =
506                               list(col = brewer.pal(n,"Set1"))),
507             auto.key = list(title = "Clusters", columns = n),
508             sub = sub))
509 }
```

Приложение В

Моделирование данных

Модуль “Data_Simulation.r”

```

1 # Распределение поведения для одного периода
2 period.prob.matrix <- function(period, clusters, reg,
3                               range.known.obs = c(.95,1)){
4   nrow <- length(clusters)
5   ncol <- period
6   idx.col <- sample(ncol, size = reg, rep = FALSE)
7   if (nrow <= reg) {
8     idx.row <- c(sample(1:nrow),
9                  sample(nrow, reg-nrow, rep = TRUE))
10  } else idx.row <- sample(nrow, reg, rep = FALSE)
11  prob.matrix <- sapply(1:ncol, function (i) {
12    res <- numeric(nrow+1)
13    indx <- which(i == idx.col)
14    indx <- ifelse(length(indx) != 0, idx.row[indx], 0) + 1
15    res[indx] <- runif(1, min = range.known.obs[1],
16                     max = range.known.obs[2])
17    res[-indx] <- runif(nrow)
18    res[-indx] <- (1-res[indx])*(res[-indx])/sum(res[-indx])
19    res
20  })
21  rownames(prob.matrix) <- c(0, clusters)
22  colnames(prob.matrix) <- 1:period
23  prob.matrix
24 }
25
26 # Последовательность перемещений одного сегмента,
27 # порожденная распределением поведения
28 move.from.matrix <- function(prob.matrix, clusters){
29   sapply(1:ncol(prob.matrix), function(i){
30     sample(clusters, size = 1, prob = prob.matrix[,i])
31   })
32 }
33

```

34 *# Генерирование распределений*

```

35 gen.beh.matrix <- function(behaviours = list(list("12" = c(3,5),
36                                               "NA" = c(8,9)),
37                                               list("24" = c(2),
38                                               "NA" = c(6,7))),
39                             regularity = list("12" = 3,"24" = 3,
40                                               "168" = 20),
41                             range.known.obs = c(.95,1)){
42   require(plyr)
43   llply(behaviours, function(beh){
44     periods <- names(beh)[names(beh) != "NA"]
45     res <- lapply(periods, function(period){
46       period.prob.matrix(period = as.numeric(period),
47                           clusters = beh[[period]],
48                           reg = regularity[[period]],
49                           range.known.obs = range.known.obs)
50     })
51   })
52   names(res) <- periods
53   res
54 })
55 }
56

```

57 *# Генерирование данных по смоделированным распределениям*

```

58 gen.data <- function(sep.val = 0.4, num.outlier = 10, junk.rate = 0.1,
59                       login.dev = 0.5, observ.period = 6, na.size = 150,
60                       behaviours = list(list("24" = c(1,2),"NA" = c(4,5)),
61                                           list("24" = c(3), "NA" = c(6,7))),
62                       list.beh.matrix){
63   require(clusterGeneration)
64   num.observs <- 24*30*observ.period
65   data <- data.frame(clind = numeric(0), time = numeric(0))
66   start.time <- 0
67   for(i in seq_along(behaviours)){
68     beh <- behaviours[[i]]
69     periods <- names(beh)
70     for(cur.period in periods){
71       cur.clusters <- beh[[cur.period]]
72       if(cur.period != "NA"){

```

```

73     prob.matrix <- list.beh.matrix[[i]][[cur.period]]
74     nrep <- round(num.observs/as.numeric(cur.period))
75     move <- as.vector(replicate(nrep,
76                               move.from.matrix(prob.matrix,
77                                                c(0, cur.clusters))))
78     length.move <- length(move)
79     idx <- which(move != 0)
80     n.junk <- round(length(idx)*junk.rate/(1-junk.rate))
81     r.idx <- range(idx)
82     junk.move <- sample(cur.clusters, n.junk, replace = TRUE)
83     move <- c(move, junk.move)
84     junk.time <- runif(n.junk, min = r.idx[1], max = r.idx[2])
85     time <- start.time + c(1:length.move, junk.time)
86     time <- time + rnorm(length.move + n.junk, sd = login.dev)
87     df <- data.frame(clind = move, time = time)
88     df <- df[df$clind != 0, ]
89     data <- merge(data, df, all = TRUE)
90   } else {
91     move <- sample(cur.clusters, na.size, rep = TRUE)
92     time <- runif(na.size, min = 1, max = num.observs) + start.time
93     data <- merge(data, data.frame(clind = move, time = time),
94                  all = TRUE)
95   }
96 }
97 start.time <- (max(data$time)%/%24 + 1)*24
98 }
99 if(num.outlier > 0){
100   move <- rep(0, num.outlier)
101   time <- runif(num.outlier, min = 1,
102                max = num.observs*length(behaviours))
103   data <- merge(data, data.frame(clind = move, time = time),
104                all = TRUE)
105 }
106 table.clust <- table(data$clind)
107 if("0" %in% names(table.clust)) {
108   clust.sizes <- table.clust[-1]
109 } else clust.sizes <- table.clust
110 num.clust <- length(clust.sizes)
111 cldata <- genRandomClust(numClust = num.clust, sepVal = sep.val,

```

[illegible]

Приложение Г

Кластеризация распределений методом k -средних

Модуль "Matrix_Clustering.r"

```

1 source("Global_Source.r")
2
3 # Преобразование матрицы для дальнейшего применения kmeans
4 matrix.for.kmeans <- function(list.matrix){
5   res <- t(sapply(list.matrix, function(x) sqrt(as.vector(x))))
6   rownames(res) <- 1:nrow(res)
7   res
8 }
9
10 # Кластеризация
11 matrix.kmeans <- function(list.matrix, centers, iter.max = 1000,
12                           nstart = 1000,
13                           algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
14                                         "MacQueen"),
15                           km.method){
16   require(flexclust)
17   x <- matrix.for.kmeans(list.matrix)
18   if (identical(km.method, "kmeans")){
19     res <- kmeans(x, centers, iter.max, nstart, algorithm)
20   }
21   if (identical(km.method, "cclust")){
22     res <- cclust(x, k = centers, method = "hardcl", control =
23                 as(list(iter.max = iter.max), "cclustControl"))
24   }
25   res
26 }
27
28 # Список матриц - кластеров после кластеризации
29 list.matrix.after.clustering <- function(list.matrix, cluster){
30   mm <- matrix.for.kmeans(list.matrix)
31   aa <- by(mm, cluster, colSums)
32   lapply(seq_along(aa), function(i) {
33     m <- matrix(aa[[i]], nrow = nrow(list.matrix[[1]]),

```

```

34             ncol = ncol(list.matrix[[1]]))
35     stopifnot(length(unique(colSums(m))) == 1)
36     rownames(m) <- rownames(list.matrix[[1]])
37     colnames(m) <- colnames(list.matrix[[1]])
38     attributes(m)$num.of.obs <- sum(m[,1])
39     attributes(m)$consist <- as.numeric(rownames(mm))[cluster == i]
40     attributes(m)$stage <- -i
41     m
42   })
43 }

```

Приложение Д

Вариации алгоритма

Модуль "Simulation.r"

```

1 source("Global_Source.r")
2 source("Data_Simulation.r")
3
4 # Максимум периодограммы
5 max.periodogram <- function(data, periodogram.fun = spec.pgram,
6                               plot = FALSE, fast = FALSE, ...){
7   spc <- periodogram.fun(scale(data), plot = plot, detrend = FALSE,
8                           demean = TRUE, fast = fast, ...)
9   max(spc$spec) - log(length(spc$spec))
10 }
11
12 # Вычисление периодов через перестановочный тест
13 permutation.test <- function(data, periodogram.fun = spec.pgram,
14                               plot = FALSE, nrep = 100,
15                               alpha = .01, fast = FALSE, ...){
16   dots <- list(...)
17   spc.tmp <- replicate(nrep,
18                       max(do.call(periodogram.fun,
19                                   args = c(list(x = sample(data),
20                                               plot = plot,
21                                               fast = fast),
22                                             unlist(dots)))$spec))
23
24   spc.data <- periodogram.fun(data, plot = plot, fast = fast, ...)
25   res <- max(spc.data$spec) >= quantile(spc.tmp,
26                                         probs = (1-alpha), names = FALSE)
27   res
28 }
29
30 # Проверка данных (один кластер) на периодичность:
31 # 0 - нет периодичности
32 # 1 - есть периодичность
33 periodicity.test <- function(data, periodogram.fun = spec.pgram,

```

```

34                                     plot = FALSE, nrep = 100, alpha = .01,
35                                     fast = FALSE,...){
36   require("evd")
37   require("GeneCycle")
38   f <- (fisher.g.test(scale(data)) < alpha)
39   g <- (pgumbel(max.periodogram(data, periodogram.fun, fast = fast,...),
40               lower.tail = FALSE) < alpha)
41   pt <- permutation.test(data, periodogram.fun, alpha = alpha,
42                          nrep = nrep, fast = fast, ...)
43   res <- as.numeric(c(f, g, pt))
44   names(res) <- c("fisher","max.periodogram", "permutation")
45   return (res)
46 }
47
48 # Проверка данных на периодичность по всем кластерам
49 test <- function(data, periodogram.fun = spec.pgram, alpha = .01,
50                  numClust = 7, power = 168,
51                  true.periods = NULL,
52                  nrep = 100, fast = FALSE, ... ){
53   cluster <- 1:numClust
54   expdata <- expand.data(data)
55   expdata[is.na(expdata)] <- FALSE
56   res <- sapply(cluster, function(cl){
57     idx <- expdata == cl
58     r <- range(which(idx))
59     new.idx <- seq(from = r[1], to = r[2], by = 1)
60     n <- (length(new.idx )%% power) * power
61     cldata <- as.numeric(idx[new.idx[1:n]])
62     periodicity.test(cldata, periodogram.fun = periodogram.fun,
63                     alpha = alpha, fast = fast, nrep = nrep,...)
64   })
65   res <- t(res)
66   if(!is.null(true.periods)) res <- cbind(res, true.periods)
67   cbind(cluster, res)
68 }
69
70 # Определение по поведению правильное наличие периодичности в кластерах
71 # 0 - нет периодичности
72 # 1 - есть периодичность

```

```

73 periodicity.in.clusters <- function(behaviours){
74   per <- numeric()
75   non.per <- numeric()
76   for(x in behaviours){
77     idx <- names(x) == "NA"
78     per <- c(per, unlist(x[!idx]))
79     non.per <- c(non.per, unlist(x[idx]))
80   }
81   res <- numeric(length(unique(per)) + length(unique(non.per)))
82   res[per] <- 1
83   res
84 }
85
86 # Дополнение матрицы p до матрицы q
87 extension.matrix <- function(p,q){
88   rp <- rownames(p)
89   rq <- rownames(q)
90   idx <- which(!(rq %in% rp))
91   if(length(idx) != 0){
92     add.mtrx <- matrix(0, nrow = length(idx), ncol = ncol(p),
93                       dimnames = list(rownames(q)[idx]))
94     p1 <- rbind(p, add.mtrx)
95     p1 <- p1[order(rownames(p1)),]
96   } else {
97     p1 <- p
98   }
99   p1
100 }
101
102 # Сравнение стохастических матриц
103 compare.matrix <- function(x,y, method = c("max","mean","JSD","con")){
104   method <- match.arg(method)
105   x1 <- extension.matrix(x,y)
106   y1 <- extension.matrix(y,x)
107   if (identical(method, "max")) res <- max(abs(x1-y1))
108   if (identical(method, "mean")) res <- mean(abs(x1-y1))
109   if (identical(method, "con")) res <- mean(abs(x1[-1,] - y1[-1,]))
110   if (identical(method, "JSD")) res <- prob.JSD(x1,y1)
111   res

```

```

112 }
113
114 # Кластеризация распределений, результат - список матриц.
115 kmeans.clustering <- function(list.matrix, centers = 2,
116                               iter.max = 100, nstart = 1000, n_beh,
117                               km.method,...){
118   if(!missing(n_beh)) centers = n_beh
119   km <- matrix.kmeans(list.matrix, centers = centers,
120                       iter.max = iter.max, nstart = nstart,
121                       km.method = km.method)
122   if (identical(km.method, "kmeans")){
123     cluster <- km$cluster
124   }
125   if (identical(km.method, "cclust")){
126     cluster <- km@cluster
127   }
128   list.matrix.after.clustering(list.matrix, cluster)
129 }
130
131 hclust.clustering <- function(list.matrix, period, move.seq, dist.func,
132                               n_beh, ...){
133   mc <- merge.and.error(list.matrix = list.matrix, T = period,
134                         S = move.seq, dist.func = dist.func)
135   idx <- length(mc) - n_beh + 1
136   mc[[idx]]$lmx
137 }
138
139 kmeans.hclust.clustering <- function(list.matrix, centers, iter.max = 100,
140                                       nstart = 1000,
141                                       period, move.seq, dist.func, n_beh,
142                                       km.method, ...){
143   km <- kmeans.clustering(list.matrix, centers = centers,
144                           iter.max = iter.max, nstart = nstart,
145                           km.method = km.method)
146   hclust.clustering(list.matrix = km, period = period,
147                     move.seq = move.seq,
148                     dist.func = dist.func, n_beh = n_beh)
149 }
150

```

```

151 prob.clustering <- function(list.matrix, method,...){
152   method(list.matrix, ...)
153 }
154
155 rest.algorithm <- function(sep.val, num.outlier, junk.rate,
156                             login.dev, observ.period, na.size, behaviours,
157                             list.beh.matrix, periodogram.fun, alpha, nrep,
158                             fast, real.periods, true.periods, all.methods,
159                             all.dist, n_beh, centers, iter.max, nstart,
160                             km.method = "cclust"){
161   data <- gen.data(sep.val = sep.val, num.outlier = num.outlier,
162                   junk.rate = junk.rate, login.dev = login.dev,
163                   observ.period = observ.period, na.size = na.size,
164                   behaviours = behaviours,
165                   list.beh.matrix = list.beh.matrix)
166   exp.data <- expand.data(data)
167   allperiods <- lapply(all.periods.periodogram(exp.data),
168                       function(x) x$periods)
169   test.periodicity <- test(data = data, alpha = alpha,
170                           periodogram.fun = periodogram.fun,
171                           numClust = length(table(exp.data)),
172                           true.periods = true.periods,
173                           nrep = nrep, fast = fast)
174   res <- lapply(real.periods, function(period){
175     stopifnot(length(in.which.clusters(period, allperiods))!= 0)
176     move.seq <- move.seq.for.period(period, allperiods, exp.data)
177     list.matrix <- list.matrix.segments(move.seq, period)
178     require(plyr)
179     list.res.beh <- llply(all.methods, function(cur.method){
180       llply(all.dist, function(cur.dist){
181         prob.clustering(list.matrix,
182                         method = cur.method, period = period,
183                         move.seq = move.seq, dist.func = cur.dist,
184                         n_beh = n_beh, centers = centers,
185                         iter.max = iter.max, nstart = nstart,
186                         km.method = km.method)
187       })
188     })
189     list.res.beh[[length(list.res.beh) + 1]] <- list()

```



```

268         range.known.obs = range.known.obs,
269         periodogram.fun = periodogram.fun,
270         alpha = alpha, nrep = nrep,
271         fast = fast, centers = centers,
272         iter.max = iter.max, nstart = nstart,
273         km.method = km.method))
274   arg.list <- list(N = N, Nrep = Nrep, sep.val = sep.val,
275                 num.outlier = num.outlier, junk.rate = junk.rate,
276                 login.dev = login.dev, observ.period = observ.period,
277                 na.size = na.size, behaviours = behaviours,
278                 regularity = regularity,
279                 range.known.obs = range.known.obs,
280                 alpha = alpha, nrep = nrep, fast = fast,
281                 centers = centers, iter.max = iter.max,
282                 nstart = nstart, km.method = km.method)
283   list(res.sim = res.sim, arg.list = arg.list)
284 }
285
286 fixed.regularity <- function(N = 1, Nrep = 1, sep.val = 0.4,
287                             num.outlier = 10, observ.period = 10,
288                             na.size = 150,
289                             behaviours=list(list("24" = 1:2,"NA" = 4:5),
290                                             list("24" = 3, "NA" = 6:7)),
291                             regularity = list("24" = 3, "168" = 10),
292                             range.known.obs = c(.95,1),
293                             periodogram.fun = spec.pgram,
294                             alpha = .05, nrep = 100, fast = FALSE,
295                             centers = 10, iter.max = 1000, nstart = 1000,
296                             km.method = "cclust",
297                             file.name = "good_reg"){
298   require("foreach")
299   range.junk.rate <- seq(0, .5, by = .1)
300   range.login.dev <- seq(0, 2, by = .25)
301   range.regularity <- lapply(seq(from = 2, to = 24, by = 2),
302                             function(x) list("24" = x, "168" = 10))
303   foreach(cur.junk.rate = range.junk.rate) %:%
304     foreach(cur.login.dev = range.login.dev) %:%
305       foreach(cur.regularity = range.regularity) %do% {
306         sim <- nrep.simulation(N = N, Nrep = Nrep, sep.val = sep.val,

```

```

307         num.outlier = num.outlier,
308         junk.rate = cur.junk.rate,
309         login.dev = cur.login.dev,
310         observ.period = observ.period,
311         na.size = na.size, behaviours = behaviours,
312         regularity = cur.regularity,
313         range.known.obs = range.known.obs,
314         periodogram.fun = periodogram.fun,
315         alpha = alpha, nrep = nrep, fast = fast,
316         centers = centers, iter.max = iter.max,
317         nstart = nstart, km.method = km.method)
318     cur.names = paste(file.name, cur.junk.rate,
319                       cur.login.dev, cur.regularity[["24"]], sep = "_")
320     save(sim, file = cur.names)
321 }
322 return(":)")
323 }

```