Московский Государственный Университет

Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Курсовая Работа

Поиск скрытых поведенческих паттернов

Вишневский Валерий
группа 317
научный руководитель: Ветров Дмитрий Петрович

Москва, 2009

Аннотация

В работе рассматривается алгоритм поиска скрытых поведенческих закономерностей (hidden T-Patterns), которые сложно обнаружить визуально, или с помощью стандартных статистических методов. Проводится анализ параметров алгоритма. Предложенный метод поиска паттернов основывается на определении взаимосвязи между парами событий, названной критическим отношением(critical interval relation). Поиск производится снизу вверх: алгоритм сначала находит простые закономерности, потом, путем соединения простых, образуются более сложные паттерны. На каждом шаге проводится отбор самых существенных и полных паттернов. Данный алгоритм был реализован на языке Си в виде консольного приложения и модуля Matlab.

Содержание

1 Теоретическое обоснование. Определения **3**

1.1 Наблюдаемый временной ряд 3

1.2 Понятие критического интервала 3

1.3 Определение Т-Паттерна 4

2 Алгоритм поиска Т-Паттернов **4**

2.1 Формальное описание 5

2.2 О параметрах α и N_{min} 6

2.3 Случайные паттерны 6

3 Анализ результатов работы алгоритма **7**

3.1 Стратегия выбора критического интервала 7

3.2 Запрет паттернов с одинаковыми событиями 9

4 Документация модуля к Matlab **9**

4.1 Описание функций 9

1 Теоретическое обоснование. Определения

1.1 Наблюдаемый временной ряд

Пусть время наблюдения разбито на N_t интервалов. В каждый момент *периода наблюдения(observation period)* $[1, N_t]$ может произойти некоторое событие(*действие, event*) [1] из множества допустимых событий $\mathcal{E}$ (*event types*). Соответственно, каждому типу событий сопоставляется множество моментов времени $TS(\mathbf{A})$

$$TS(\mathbf{A}) = \{T_{\mathbf{A},1}, \ldots, T_{\mathbf{A},N_\mathbf{A}}\}, \quad \mathbf{A} \in \mathcal{E}, \quad 0 \leqslant T_{\mathbf{A},i} \leqslant N_t \;\; (i = 1, \ldots, N_\mathbf{A})$$

.

1.2 Понятие критического интервала

Во время поиска закономерностей в данных, нас интересуют отношения между распределениями отдельных событий. Если предположить, что в исходных не существует никаких закономерностей, то каждое событие должно появляться независимо, от других; то есть распределение компонентов должно быть независимым. Но паттерн характеризуется появлением своих компонентов в одинаковом порядке, более того, временные интервалы, разделяющие компоненты, должны быть примерно одинаковыми.

Будем говорить, что события $\mathbf{A}$ и $\mathbf{B}$ связаны отношением *критического интервала (Critical Interval, CI)*, если, после появления события $\mathbf{A}$ в момент времени t, существует интервал $[t + d_1, t + d_2]$, $(d2 \geqslant d1 \geqslant 0)$, содержащий $\mathbf{B}$, чаще, чем это ожидается из предположения о независимости событий. Данную взаимосвязь будем обозначать, как $\mathbf{A}[d_1, d_2]\mathbf{B}$, или, короче, $(\mathbf{AB})$.

Далее раскроем понятие «чаще чем это ожидается». Пусть $N_\mathbf{A}$ и $N_\mathbf{B}$ — количество возникновений $\mathbf{A}$ и $\mathbf{B}$, соответственно, в течение $[1, N_t]$. $P(\mathbf{A}) = N_\mathbf{A}/N_t$ — вероятность появления события $\mathbf{A}$ в некоторый момент времени; $P(\neg\mathbf{A}) = 1 - P(\mathbf{A})$. $P(\neg\mathbf{A})^d$ — вероятность, что $\mathbf{A}$ не появится в течение какого-либо интервала $[d_1, d_2]$, $(d = d_2 - d_1 + 1)$, длины d. Вероятность наблюдать $\mathbf{A}$ на интервале длины d один или более раз, равна $1 - P(\neg\mathbf{A})^d$.

Зафиксируем события $\mathbf{A}$, $\mathbf{B}$, и длину интервала d. Приняв гипотезу о независимом распределении событий, событие $\mathbf{B}$ содержится[2] в интервале длины d, после события $\mathbf{A}$, $N_\mathbf{A} * (1 - P(\neg\mathbf{B})^d)$ раз.

$$\rho = P(\geqslant N_\mathbf{AB}) = 1 - P(< N_\mathbf{AB})$$

[1] Чаще всего понимается, что в этот момент времени имеет место *начало* действия

[2] Один или более раз.

— априорная вероятность того, что $N_{\mathbf{AB}}$ из $N_{\mathbf{A}}$ интервалов содержат вхождения $\mathbf{B}$. Очевидно, что $P(< N_{\mathbf{AB}})$ распределено по биномиальному закону, где $N_{\mathbf{A}}$ — количество «испытаний», $1 - P(\neg\mathbf{B})^d$ — вероятность «успеха». Следовательно,

$$\rho = P(\geqslant N_{\mathbf{AB}}) = 1 - \sum_{i=0}^{N_{\mathbf{AB}}-1} C_{N_{\mathbf{A}}}^{i} (1 - P(\neg\mathbf{B})^d)^i P(\neg\mathbf{B})^{N_{\mathbf{A}}-i}$$

.

Полученная вероятность ρ сравнивается с пороговым значением α, являющимся структурным параметром поиска: если $\rho \leqslant \alpha$, то заданный интервал признается критическим. Заметим, что ρ зависит от $N_{\mathbf{A}}, N_{\mathbf{B}}, N_t, N_{\mathbf{AB}}, d$:

$$\rho = \rho(N_{\mathbf{A}}, N_{\mathbf{B}}, N_t, N_{\mathbf{AB}}, d)$$

1.3 Определение T-Паттерна

Дадим рекурсивное определение Т-Паттерна. Договоримся называть каждое допустимое событие *псевдопаттерном*. Тогда Т-Паттерн $\mathbf{Q}$ можно определить как:

$$\mathbf{Q} = \mathbf{X_1}[dL_1, dR_1]\mathbf{X_2}[dL_2, dR_2] \dots \mathbf{X_i}[dL_i, dR_i]\mathbf{X_{i+1}} \dots \mathbf{X_m},$$

$$Events(\mathbf{Q}) = \begin{cases} \{Events(\mathbf{X_1}), \dots Events(\mathbf{X_m})\}, & \text{если } \mathbf{Q} \text{ — Т-Паттерн} \\ Q, & \text{если } \mathbf{Q} \text{ — событие (псевдопаттерн)} \end{cases}$$

где $\mathbf{X_i}, (i = 1 \dots m)$ — Т-Паттерн, или псевдопаттерн. Для паттернов отношение критического интервала $\mathbf{Q_L}[dL, dR]\mathbf{Q_R}$ вводится, как и для событий, с учетом того, что интервал $[dL, dR]$ отсчитывается от последнего элемента $\mathbf{Q_L}$, и вхождение $\mathbf{Q_R}$ определяется его первым элементом. Будем называть *двойными сериями(double series, DS)* паттерна $\mathbf{Q}$, множество $\{\{Left_i, Right_i\}_{i=1\dots N_Q}\}$, где $N_{\mathbf{Q}}$ — количество появлений паттерна $\mathbf{Q}$, $Left_i, Right_i$ — индексы начального и конечного событий i-го появления паттерна $\mathbf{Q}$.

2 Алгоритм поиска T-Паттернов

Алгоритм поиска Т-Паттернов, описанный ниже, заключается в итеративном повторении двух стадий: конструирование новых паттернов и удаление неполных паттернов. На выходе алгоритм выдает множество Т-Паттернов с их сопутствующими характеристиками: критическим интервалом, уровнем значимости, частотой встречаемости.

Пусть $\mathcal{D}_i$ — множество паттернов, обнаруженных к i-ой итерации. Фактически, множество

$$\mathcal{D}_m \setminus \mathcal{E},$$

где m — номер последней итерации, и будет результатом работы алгоритма.

Стадия 1: Поиск и конструирование:
На данном шаге, для любой упорядоченной пары

$$(\mathbf{Q}', \mathbf{Q}'') : \mathbf{Q}', \mathbf{Q}'' \in \mathcal{D}_i,$$

проверяется существование критической связи. Если критическая связь $[dL, dR]$ была найдена, и паттерн $(\mathbf{Q}'\mathbf{Q}'')$ встречается чаще, чем N_{min} раз, то в множество $\mathcal{D}_{i+1}$ добавляется новый паттерн $\mathbf{Q}'[dL, dR]\mathbf{Q}''$.

Однако такие действия приводят к тому, что один и тот же паттерн **ABCD** может быть сконструирован разными способами (например как $(\mathbf{A}(\mathbf{BCD}))$ и $((\mathbf{AB})(\mathbf{CD}))$), что ведет к заполнению множества $\mathcal{D}$ лишними паттернами, и как следствие, к замедлению работы программы. Для избежания данной проблемы предлагается следующее решение: паттерн $(\mathbf{Q}'\mathbf{Q}'')$ добавляется в множество $\mathcal{D}$ тогда и только тогда, когда не существует паттерна $\mathbf{P}$ из $\mathcal{D}$ такого, что $Events(\mathbf{P}) = Events((\mathbf{Q}'\mathbf{Q}''))$ и $DS(\mathbf{P}) = DS((\mathbf{Q}'\mathbf{Q}''))$. Другими словами, найденный паттерн, перед добавлением, сравнивается с уже существующими паттернами и проверяется, не является ли он дубликатом.

Стадия 2: Удаление неполных паттернов:
На данной стадии алгоритм стремится удалить найденные паттерны, являющиеся меньшими частями, или неполными версиями других обнаруженных паттернов. Для индикации таких паттернов можно применять разные эвристики. Ниже опишем условие, взятое из статьи [**?**].

Итак, паттерн $\mathbf{Q_x}$ считается менее полным, чем $\mathbf{Q_y}$, если $\mathbf{Q_x}$ и $\mathbf{Q_y}$ появляются одинаково часто, и все события возникающие в $\mathbf{Q_x}$, также возникают в $\mathbf{Q_y}$.

2.1 Формальное описание

Require: $\mathcal{E}$ — допустимые события,

N_t — продолжительность наблюдения,

α — минимальный уровень значимости,

N_{min} — минимальное количество вхождений паттерна.

1: $\mathcal{D}_{-1} = \varnothing$
2: $\mathcal{D}_0 = \mathcal{E}$ ▷ Инициализируем множество паттернов
3: $t = 0$
4: **while** $\mathcal{D}_t \neq \mathcal{D}_{t-1}$ **do**
5: $t = t + 1$
6: $\mathcal{D}_t = \mathcal{D}_{t-1}$
7: **for** $P_L \in \mathcal{D}_{t-1}$ **do** ▷ Стадия 1
8: **for** $P_R \in \mathcal{D}_{t-1}$ **do**

```
9:          for all  d_L, d_R ∈ [1, N_t] таких, что d_L ⩽ d_R  do
10:             if  ρ(N_{P_L}, N_{P_R}, N_t, N_{P_L P_R}, d_R − d_L + 1) < α  then
11:                 if  isUnique(P_L ∪ P_R, D_t)  then
12:                     D_t = D_t ∪ {P_L ∪ P_R}
13:     for  P_L ∈ D_t \ E  do                                   ▷ Стадия 2
14:         for  P_R ∈ D_t  do
15:             if  P_L ⊂ P_R[3] and |DS(P_L)|=|DS(P_R)| and isIntersect(P_L, P_R) then
16:                 удалить P_L из D_t
17: function isUnique(Q, D)
18:     for P ∈ D do
19:         if  DS(Q)=DS(P) and Events(Q)=Events(P)  then
20:             return false
21:     return true
22: function isIntersect(P_L, P_R)
23:     for i = 1 . . . |DS(P_L)| do
24:         if  DS_{i,Left}(P_L)>DS_{i,Right}(P_R) or DS_{i,Right}(P_L) < DS_{i,Left}(P_R) then
25:             return false
26:         return true
27:     return true
```

2.2 О параметрах α и N_{min}

Так как для задачи поиска Т-Паттернов не вводится определение функционала качества найденных паттернов, то алгоритм требует ручной настройки параметров α и N_{min}. Выбор значений параметров должен основываться на специфике наблюдаемых процессов и ожидаемых результатов. Однако выбор значений $\alpha = 0.005$ и $N_{min} = 3$ обычно удовлетворителен [**?**, с. 99]. Для более тонкой настройки поиска, можно использовать разные α и N_{min} для паттернов разной длины. Например, при значениях $\alpha = 0.00001$, $N_{min} = 7$ для паттернов длины 2, и $\alpha = 0.005$, $N_{min} = 3$ для паттернов длины 3 и более, алгоритм найдет только несколько самых ярко выраженных патернов длины 2, и уже потом будет конструировать множество более длинных паттернов.

2.3 Случайные паттерны

Когда исследования проводятся на больших наборах данных, то Т-Паттерны могут возникать даже, если выборка была сгенерированна случайно. Поэтому, для найденного множества паттернов было бы полезно оценить, являются ли они случайными,

[3] Имеется в виду упорядоченная вложенность множеств. Т.е. $abd \subset abcd$, но $bdc \not\subset abcd$

или «структурными». Одним из подходов к решению данной задачи, является анализ рандомизированных данных:

По множеству $\mathcal{E}$ строится множество $\mathcal{E}'$:

$$\mathcal{E}' = \{rand(\mathbf{A}) | \mathbf{A} \in \mathcal{E}\}, \text{где}$$

$$rand(\mathbf{A}) = \mathbf{A}' : TS(\mathbf{A}') = \{\xi_1, \ldots, \xi_{N_\mathbf{A}}\}$$

$$\xi_1, \ldots, \xi_{N_\mathbf{A}} \sim \mathbf{U}[1, N_t].$$

Проще говоря, создаются данные с такой же протяженностью периода наблюдения и мощностью множества допустимых событий, но для каждого допустимого события $\mathbf{A}$, моменты времени его появления генерируются случайно с вероятностью появления $N_\mathbf{A}/N_t$.

Для полученного множества $\mathcal{E}'$ применяется процесс поиска паттернов с *теми же параметрами*, которые применялись на исходных данных. Описанные действия исполняются несколько раз, после чего, результаты поиска на рандомизированных данных сравниваются с исходными.

Считается, что поиск Т-Паттернов прошел успешно, если в исходных данных было выявлено значительно больше Т-Паттернов, чем в рандомизированных, или они оказались длиннее.

3 Анализ результатов работы алгоритма

3.1 Стратегия выбора критического интервала

Во время поиска связи критического интервала $\mathbf{Q_L}[dL, dR]\mathbf{Q_R}$ между двумя паттернами, вообще говоря, проверяются все возможные интервалы $[dL, dR]$. На практике, не является редкостью случай, когда для двух паттернов существует несколько пар dL и dR, удовлетворяющих отношению критического интервала. Для выбора конкретных значений dL и dR, предлагается использовать одну из нижеописанных стратегий. Для каждой стратегии представлен результат работы алгоритма на тестовых данных. Данные содержат паттерн длины 7, встречающийся 8 раз.

Выбор кратчайшего критического интервала: При использовании данной стратегии, на каждом шаге выбирается критический интервал, имеющий наименьшую длину d ($d = dR - dL + 1$). Такой подход позволяет уменьшить длину критических связей, тем самым выявляя более выраженные и «стройные» паттерны. Одним из недостатков данного метода является эффект «расщепления». Поясним данный эффект на примере: пусть в исходных данных существует критическая связь $\mathbf{A}[4, 20]\mathbf{B}$, наблюдаемая 15 раз. Данная связь настолько ярко

выражена, что алгоритму не требуется подбирать границы критического интервала, чтобы **B** появлялось после **A** все 15 раз. Алгоритм, скорее, выделит два(или даже больше) критических интервала: $\mathbf{A}[4, 12]\mathbf{B}$, наблюдаемый 6 раз, и $\mathbf{A}[13, 18]\mathbf{B}$, наблюдаемый 7 раз. Таким образом, один ярко выраженный паттерн будет распадаться на несколько более редких, что в свою очередь, может помешать дальнейшему выявлению закономерностей. Ниже описанный метод позволяет избавиться от такой проблемы.

Выбор самого длинного критического интервала: Как следует из названия, при использовании этой стратегии, среди всех значимых $[dL, dR]$, выбирается интервал, соответствующий наибольшему значению d. В результате некоторые паттерны «загрубляются», и алгоритм стремится найти максимальное количество вхождений каждого критического интервала.

Выбор самого значимого критического интервала: При этом подходе выбирается критический интервал, имеющий максимальный уровень значимости ρ.

В общем случае, нету каких-либо рекомендаций по выбору стратегии поиска критического интервала. Выбор должен основываться на априорных сведениях о наблюдаемом процессе и ожидаемых результатов эксперимента.

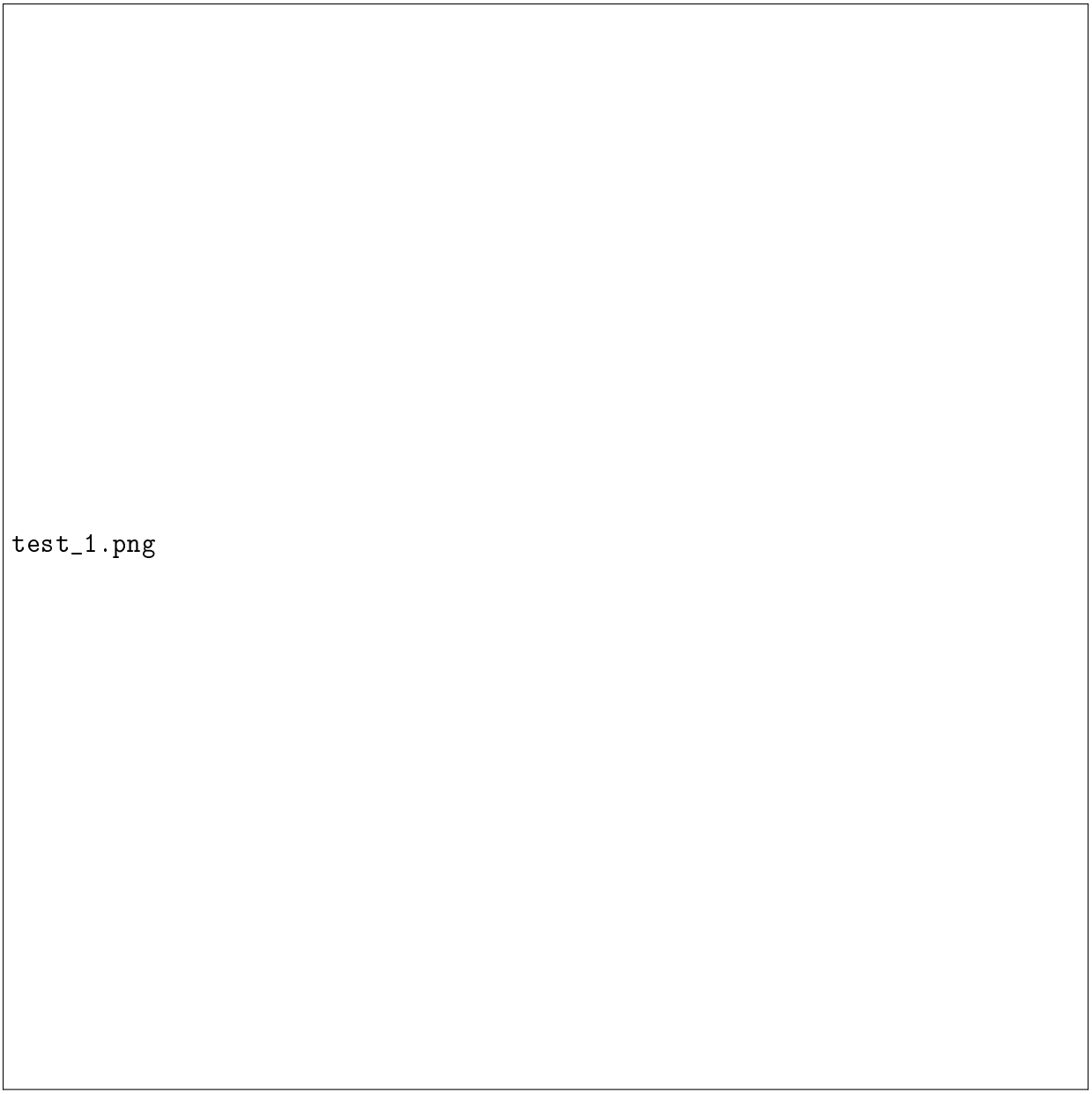

Рис. 1: Визуализация исходного паттерна `abcdef`.

Стратегия	**Найдено паттернов**	**Найдено вхождений целевого паттерна**
Короткий	40	Найдено 3 представления целевого паттерна. Для каждого представления 4 вхождения.
Длинный	5	8
Значимый	7	7

3.2 Запрет паттернов с одинаковыми событиями

Если при поиске поведенческих закономерностей известно, что в один и тот же паттерн не могут входить два одинаковых события, то существует возможность сообщить это алгоритму. Таким образом, если, $Events(\mathbf{Q_L}) \cap Events(\mathbf{Q_R}) \neq \varnothing$, то паттерн $(\mathbf{Q_L}\mathbf{Q_R})$, просто не будет создан. Это условие позволяет отбросить множество лишних паттернов, и найти более длинные и ценные закономерности.

4 Документация модуля к Matlab

Matlab-реализация алгоритма состоит из следующих файлов:

mexPattern.mex*: откомпилированный mex-файл, реализующий алгоритм поиска паттернов.

mexPattern.m: объявление mex-функции.

T_DRAW_PATTERNS.m: графический вывод найденных паттернов.

T_GENERATE_PATTERNS.m: создание искусственных паттернов во временных рядах.

T_LOAD_FILE.m: загрузка временного ряда из файла для дальнейшей работы с ним.

T_STAT_VALIDATE.m: процедура статистической валидации.

test.m: пример использования модуля.

4.1 Описание функций

```
function [events, Nt, ts] = T_GENERATE_PATTERN( pat_sym, noise_sym, CIs, Npat,
        dist_b_patterns, Pnoise1, Pnoise2 )
```

Создает временной ряд, содержащий один искусственный паттерн.

Параметр	Описание
Вход:	
`pat_sym`	Матрица `1x`N_p типа `char`, определяющая паттерн. Каждый символ — событие.
`noise_sym`	Матрица `1xN` типа `char`. Определяет события которые будут генерироваться случайно.

`CIs`	Матрица (`$N_p$-1)x2` типа `int`. В i-й строке которой, записан соответствующий критический интервал.
`Npat`	Количество паттернов, которые требуется сгенерировать.
`dist_b_patterns`	Максимальное расстояние между двумя появлениями паттерна.
`P_noise1`	Частота встречаемости шумовых символов.
`P_noise2`	Вероятность того, что символ из паттерна будет зашумлен.
Выход:	
`events`	Массив структур `1xN`, где `N` — количество событий. Каждая структура состоит из двух полей: `event_name` — строка названия события, и `indexes` — матрица `1xN` типа `int`. Определяет времеа появления событий.
`Nt`	Продолжительность получившегося периода наблюдений.
`ts`	Символьная матрица `1xNt`.

```
function patterns = mexPattern(events, Nt, levels, allow_same_events, ci_strategy);
```

Реализует поиск паттернов во временных рядах.

Параметр	**Описание**
Вход:	
textttevents, Nt	См. определение `T_GENERATE_PATTERN`.
`levels`	Vатрица `Nx3` уровней значимости. Каждая строка содержит: длину паттернов, к которым должны применяться следующие параметры; минимальный уровень значимости α; минимальное количество вхождений паттерна N_{min}.
`allow_same_events`	1 если, разрешается появление одинаковых событий в паттерне, 0 иначе.
`ci_strategy`	Стратегия выбора критического интервала. 1 — стратегия выбора длиннейшего интервала, 2 — кратчайшего, 3 — самого значимого.
Выход:	
`patterns`	Массив структур. Каждая структура описывает найденный паттерн. Поля структуры: `Events` — индексы событий, которые составляют паттерн. `CIs` — интервалы, соответствующие критическим связям между событиями в паттерне. `Sign` — уровень значимости найденного паттерна. `Nab` — количество появлений паттерна. `DS` — двойные серии(*double series*) паттерна. `String` — строка, описывающая паттерн в следующем формате: $Event_1[dL_1, dR_1]Event_2 \dots Event_m <$ уровень значимости $> \{N_{ab}\} : (DS_1)(DS_2)\dots(DS_{N_{ab}})$

`function T_DRAW_PATTERNS(patterns, events, Nt, np)`

Строит диаграмму найденных паттернов.

Параметр	**Описание**
Вход:	
`patterns, events, Nt`	См. предыдущие определения.
`np`	Номер паттерна, который нужно представить. Или -1 для последовательного вывода всех паттернов.

`function [events, Nt] = T_LOAD_FILE(fname)`

Загружает временной ряд из фалйа для последующей работы с ним. Формат входного файла:

```
Time    Event
0       :
time    event
...
time    event
time    &
```

`function [p] = T_STAT_VALIDATE( Nt, events, levels, nvalidations )`

Процедура статистической валидации.

Параметр	**Описание**
Вход:	
`Nt, events, levels`	См. предыдущие определения.
`nvalidations`	Количество повторений процедуры рандомизации.
Выход:	
`p`	Целочисленная матрица `1 x nvalidations`. В каждой ячейке — количество паттернов, найденных в рандомизированных данных.

Список литературы

[1] Magnus Magnusson

[2] Кнут Д. Всё про TEX. — Протвино, RDTEX, 1993.