**Задание 1. Анализ Web-логов средствами R.**

Выполнила студентка 2 курса

группы 09-715(1)

Анисимова Татьяна.

**Текст задания:**

Пусть имеется книжный Интернет-магазин с простой структурой:

index.phtml - главная страница

|--catalog.phtml - сценарий просмотра каталога

|--search.phtml - сценарий поиска

|--addbasket.phtml?id\_book=NN - сценарий добавления в корзину книги с номером NN

|--order.phtml - сценарий оформления заказа

Пусть имеется лог-файл web-сервера Apache с данными за месяц работы интернет-магазина. Формат данных соответствует следующим параметрам в файле конфигурации Apache:

LogFormat "%h %l %u %t \"%r\" %>s %b %{id\_bask}C" common

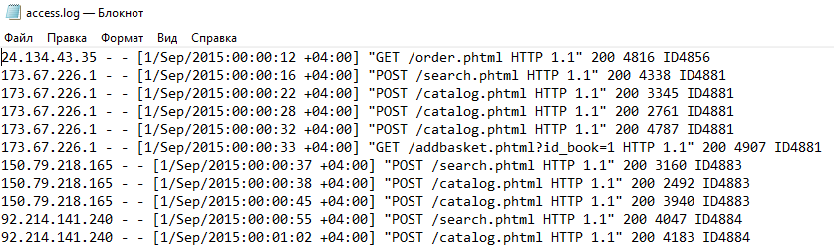
id\_bask - это cookie для идентификатора корзины, корзины хранятся в течение месяца. Полагаем, что идентификатор корзины совпадает с идентификатором посетителя.

Анализируя данные лог-файла, ответьте на следующие вопросы:

* Сколько посетителей было на сайте за месяц?
* Сколько в среднем посетителей бывает за час?
* Сколько посетителей сделало заказы?
* Сколько страниц просмотрел посетитель в среднем, максимум, минимум?
* Сколько времени прошло с момента входа на сайт до оформления заказа, в среднем, максимум, минимум?
* Сколько в среднем заказов оформляется за день?
* Имеются ли какие-то взаимозависимые товары, которые покупатель кладет в корзину (например, с уровнем поддержки выше 0.01 и достоверностью выше 0.6) ? (Задача анализа покупательских корзин - поиск ассоциативных правил, пакет **Arules**).

**Описание выполненной работы:**

Данные находятся в файле "access.log". Всего 233117 записей:



Прочитаем данные из файла в переменную dataFrame:

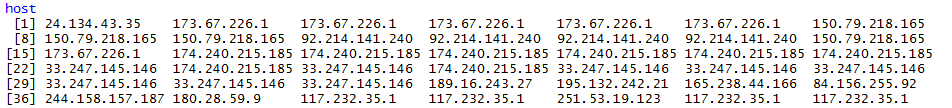
|  |
| --- |
| dataFrame = read.table("access.log", sep = ' ') |

Подключим пакет **stringr**, чтобы использовать регулярные выражения:

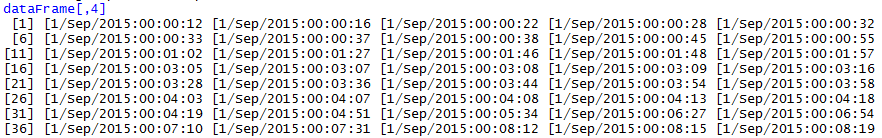
|  |
| --- |
| library(stringr) |

Хост сохраним в переменную host:

|  |
| --- |
| host = dataFrame[,1] |



Преобразуем дату к нужному виду. dataFrame[,4] имеет следующий вид:

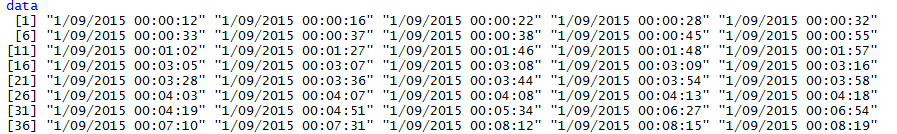


Так как во всех записях указан месяц сентябрь, заменим все буквы в дате на 09 (девятый месяц) с помощью регулярного выражения:

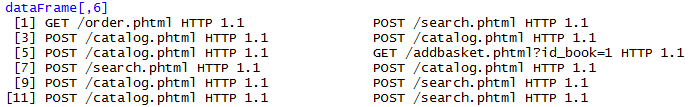
str\_replace(dataFrame[,4], "[:alpha:]+", "09")

Также удалим из записей символ "[" и заменим ":" на пробел " ". Преобразованную дату запишем в переменную data:

|  |
| --- |
| data = str\_replace(str\_replace(str\_replace(dataFrame[,4], "[:alpha:]+", "09"), "\\[", ""), ":", " ") |



dataFrame[,6] имеет следующий вид:



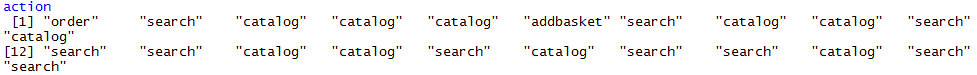
Выделим метод запроса (GET/POST). Функция str\_extract() из библиотеки stringr извлекает первую строку, которая удовлетворяет регулярному выражению. Запишем регулярное выражение, которое соответствует всем заглавным буквам. Сохраним результат в переменную p\_g:

|  |
| --- |
| p\_g = str\_extract(dataFrame[,6], "[A-Z]+") |



Выделим действие пользователя (order/catalog/search/addbasket) с помощью функции str\_extract(dataFrame[,6], "/[a-z]+"), удалим из строки символ "/", результат запишем в переменную action:

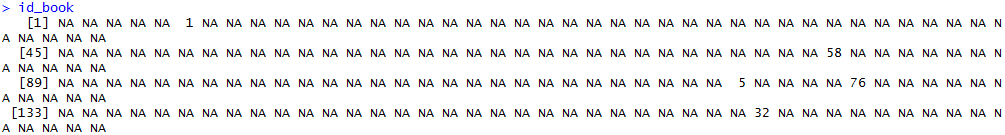
|  |
| --- |
| action = str\_replace(str\_extract(dataFrame[,6], "/[a-z]+"), "/", "") |



Выделим номер книги, которую пользователь добавляет в корзину, результат сохраним как число в переменную id\_book:

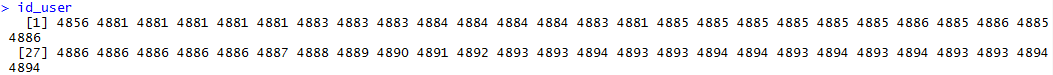
|  |
| --- |
| id\_book = as.numeric(str\_extract(str\_extract(dataFrame[,6], "=[:digit:]+"), "[:digit:]+")) |

Если запись хранит информацию о добавленной книги, то в id\_book хранится номер этой книги, иначе – NA:



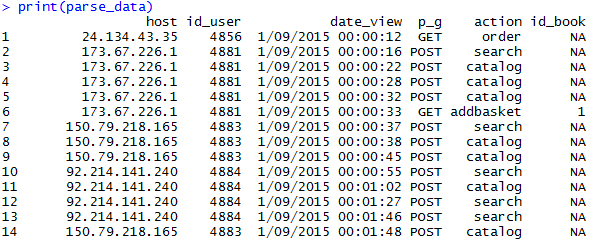
Выделим идентификатор посетителя (идентификатор корзины). Результат сохраним в переменную id\_user:

|  |
| --- |
| id\_user = as.numeric(str\_extract(dataFrame[,9], "[:digit:]+")) |



Сохраним преобразованные поля как data.frame в parse\_data:

|  |
| --- |
| parse\_data = data.frame("host" = host, "id\_user" = id\_user, "date\_view" = data, "p\_g" = p\_g,  "action" = action, "id\_book" = id\_book); |

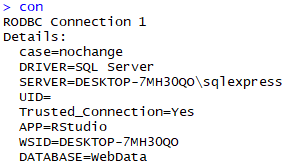


Загрузим данные в СУБД SQL SERVER. Для этого подключим библиотеку RODBC:

|  |
| --- |
| library(RODBC) |

В SQL Server Management Studio создадим базу данных “WebData”. С помощью функции odbcDriverConnect библиотеки RODBC установим соединение с БД “WebData”:

|  |
| --- |
| con<-odbcDriverConnect('driver={SQL Server};  server=DESKTOP-7MH30QO\\sqlexpress;database=WebData;trusted\_connection=true') |



Создадим таблицу “WebLog”, в которой будем хранить преобразованные данные. Для этого выполним запрос с помощью функции sqlQuery:

|  |
| --- |
| sqlQuery(con,  "create table WebLog (  host nchar(20) not null,  id\_user int not null,  date datetime not null,  p\_g nchar(5) not null,  action nchar(20) not null,  id\_book int)") |

Сохраним данные parse\_data в созданную таблицу “WebLog” с помощью функции sqlSave:

|  |
| --- |
| sqlSave(con, parse\_data, tablename = "WebLog", rownames=TRUE,safer=FALSE,  varTypes = c(host = "nchar(20)", id\_user = "int", date\_view = "datetime2", p\_g = "nchar(5)", action = "nchar(20)", id\_book = "int")); |

**Написание запросов:**

1. Сколько посетителей было на сайте за месяц?

|  |
| --- |
| SELECT COUNT(DISTINCT wl.id\_user) AS USER\_NUMB FROM WebLog wl |



То есть, за месяц сайт посетило 49337 пользователей.

1. Сколько в среднем посетителей бывает за час?

|  |
| --- |
| SELECT AVG(COUNT\_ORDER) AVG\_COUNT\_ORDER FROM  (  SELECT DATEPART(DAY, wl.date\_view) DAY, DATEPART(HOUR, wl.date\_view) HOUR, COUNT(DISTINCT wl.id\_user) COUNT\_ORDER  FROM WebLog wl  GROUP BY DATEPART(DAY, wl.date\_view), DATEPART(HOUR, wl.date\_view)  )  tmpUser; |



То есть, в среднем за час на сайт заходит 76 посетителей.

1. Сколько посетителей сделало заказы?

|  |
| --- |
| SELECT COUNT(DISTINCT wl.id\_user) AS USER\_NUMB  FROM WebLog wl  WHERE wl.action = 'order' |



То есть, заказы сделали 1790 посетителей.

1. Сколько страниц просмотрел посетитель в среднем, максимум, минимум?

|  |
| --- |
| SELECT AVG(VIEW\_NUMB) AVG\_VIEW\_COUNT, MAX(VIEW\_NUMB) MAX\_VIEW\_COUNT, MIN(VIEW\_NUMB) MIN\_VIEW\_COUNT  FROM(  SELECT wl.id\_user, COUNT(wl.id\_user) VIEW\_NUMB  FROM WebLog wl  GROUP BY wl.id\_user  )  tmpTable; |



Посетитель в среднем просмотрел 4 страницы, максимум – 40 страниц, минимум – 1 страницу.

1. Сколько времени прошло с момента входа на сайт до оформления заказа, в среднем, максимум, минимум?

|  |
| --- |
| SELECT AVG(SUB) AS AVG\_TIME, MAX(SUB) AS MAX\_TIME, MIN(SUB) AS MIN\_TIME FROM(  SELECT Table\_1.ID\_USER, DATEDIFF(MINUTE, TIME\_MIN, TIME\_MIN\_ORDER) AS SUB  FROM  (  SELECT wl.id\_user ID\_USER, MIN(wl.date\_view) TIME\_MIN\_ORDER  FROM WebLog wl  WHERE wl.action = 'order'  GROUP BY wl.id\_user  )Table\_1,  (SELECT wl.id\_user ID\_USER, MIN(wl.date\_view) TIME\_MIN  FROM WebLog wl  GROUP BY wl.id\_user  )Table\_2  WHERE Table\_1.ID\_USER = Table\_2.ID\_USER  )table\_3; |



С момента входа на сайт до оформления заказа в среднем прошло 35 минут, максимум 1816 минут, минимум 0 минут.

1. Сколько в среднем заказов оформляется за день?

|  |
| --- |
| SELECT AVG(COUNT\_ORDER) AVG\_COUNT\_ORDER FROM  (  SELECT DATEPART(DAY, wl.date\_view) DAY, COUNT(wl.id\_user) COUNT\_ORDER  FROM WebLog wl  WHERE wl.action = 'order'  GROUP BY DATEPART(DAY, wl.date\_view)  )  tmpOrder; |



В среднем оформляется 59 заказов в день.

**Поиск ассоциативных правил:**

Для работы с ассоциативными правилами подключим библиотеку arules:

|  |
| --- |
| library(arules) |

Напишем запрос, чтобы выбрать тех пользователей, у которых в корзине больше одной книги, сохраним результат в переменную books:

|  |
| --- |
| books = sqlQuery (con, "SELECT wl.id\_user, wl.id\_book FROM  (SELECT wl.id\_user USER\_ID, COUNT(wl.id\_book) COUNT\_BOOK  FROM WebLog wl  WHERE wl.id\_book IS NOT NULL  GROUP BY wl.id\_user) tmpTable, WebLog wl  WHERE COUNT\_BOOK > 1  AND USER\_ID = wl.id\_user  AND wl.id\_book IS NOT NULL  ORDER BY USER\_ID"); |

Сформируем файл trans.txt, в котором каждая строка содержит сведения о корзине, номера книг разделены запятыми:

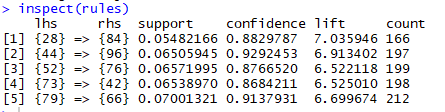
Прочтем данные из файла с помощью метода read.transactions:

|  |
| --- |
| trans = read.transactions("trans.txt", sep= ','); |

Используем априорный алгоритм. Будем учитывать только те книги, которые встречаются с частотой не ниже 1% и те ассоциативные правила, у которых confidence выше 60%.

|  |
| --- |
| rules = apriori(data=trans, parameter=list(support=0.01,confidence=0.6)) |

Распечатаем ассоциативные правила:



Видим, что

* с вероятностью больше 88% если посетитель кладет в корзину книгу номер 28, то он положит и книгу номер 84.
* с вероятностью больше 92% если посетитель кладет в корзину книгу номер 44, то он положит и книгу номер 96.
* с вероятностью больше 87% если посетитель кладет в корзину книгу номер 52, то он положит и книгу номер 76.

и т.д.

**Код программы:**

# назначим рабочий директорий

setwd("C:/Users/TatyanaAnisimova/Documents/Rprojects")

# прочитаем данные из файла в переменную dataFrame

dataFrame = read.table("access.log", sep = ' ')

print(dataFrame)

# число записей

nrow(dataFrame)

# для работы с регулярными выражениями

library(stringr)

# хост сохраним в переменную host

host = dataFrame[,1];

host

# преобразуем дату к нужному виду

data = str\_replace(str\_replace(str\_replace(dataFrame[,4], "[:alpha:]+", "09"), "\\[", ""), ":", " ");

data

# выделим метод запроса (GET/POST)

p\_g = str\_extract(dataFrame[,6], "[A-Z]+")

p\_g

# выделим действие пользователя (order/catalog/search/addbasket)

action = str\_replace(str\_extract(dataFrame[,6], "/[a-z]+"), "/", "");

action

# выделим номер книги, которую пользователь добавляет в корзину

id\_book = as.numeric(str\_extract(str\_extract(dataFrame[,6], "=[:digit:]+"), "[:digit:]+"));

id\_book

# выделим идентификатор посетителя (идентификатор корзины)

id\_user = as.numeric(str\_extract(dataFrame[,9], "[:digit:]+"));

id\_user

# сохраним преобразованные поля как data.frame

parse\_data = data.frame("host" = host, "id\_user" = id\_user, "date\_view" = data, "p\_g" = p\_g,

"action" = action, "id\_book" = id\_book);

print(parse\_data)

library(RODBC)

# установим соединение с БД “WebData”

con<-odbcDriverConnect('driver={SQL Server}; server=DESKTOP-7MH30QO\\sqlexpress;database=WebData;trusted\_connection=true')

con

# cоздадим таблицу “WebLog”, в которой будем хранить преобразованные данные

sqlQuery(con, "create table WebLog (host nchar(20) not null, id\_user int not null, date datetime not null, p\_g nchar(5) not null, action nchar(20) not null, id\_book int)")

# сохраним данные parse\_data в созданную таблицу “WebLog”

sqlSave(con, parse\_data, tablename = "WebLog", rownames=TRUE,safer=FALSE, varTypes = c(host = "nchar(20)", id\_user = "int", date\_view = "datetime2", p\_g = "nchar(5)", action = "nchar(20)", id\_book = "int"));

# запрос - 1) Сколько посетителей было на сайте за месяц?

sqlQuery(con, "SELECT COUNT(DISTINCT wl.id\_user) AS USER\_NUMB FROM WebLog wl");

# запрос - 2) Сколько в среднем посетителей бывает за час?

sqlQuery(con, "SELECT AVG(COUNT\_ORDER) AVG\_COUNT\_ORDER FROM

(

SELECT DATEPART(DAY, wl.date\_view) DAY, DATEPART(HOUR, wl.date\_view) HOUR, COUNT(DISTINCT wl.id\_user) COUNT\_ORDER

FROM WebLog wl

GROUP BY DATEPART(DAY, wl.date\_view), DATEPART(HOUR, wl.date\_view)

)

tmpUser;");

# запрос - 3) Сколько посетителей сделало заказы?

sqlQuery(con, "SELECT COUNT(DISTINCT wl.id\_user) AS USER\_NUMB FROM WebLog wl WHERE wl.action = 'order'");

# запрос - 4) Сколько страниц просмотрел посетитель в среднем, максимум, минимум?

sqlQuery(con,

"SELECT AVG(VIEW\_NUMB) AVG\_VIEW\_COUNT, MAX(VIEW\_NUMB) MAX\_VIEW\_COUNT, MIN(VIEW\_NUMB) MIN\_VIEW\_COUNT

FROM(

SELECT wl.id\_user, COUNT(wl.id\_user) VIEW\_NUMB

FROM WebLog wl

GROUP BY wl.id\_user

)

tmpTable;");

# запрос - 5) Сколько времени прошло с момента входа на сайт до оформления заказа, в среднем, максимум, минимум?

sqlQuery(con, "SELECT AVG(SUB) AS AVG\_TIME, MAX(SUB) AS MAX\_TIME, MIN(SUB) AS MIN\_TIME FROM(

SELECT Table\_1.ID\_USER, DATEDIFF(MINUTE, TIME\_MIN, TIME\_MIN\_ORDER) AS SUB

FROM

(

SELECT wl.id\_user ID\_USER, MIN(wl.date\_view) TIME\_MIN\_ORDER

FROM WebLog wl

WHERE wl.action = 'order'

GROUP BY wl.id\_user

)Table\_1,

(SELECT wl.id\_user ID\_USER, MIN(wl.date\_view) TIME\_MIN

FROM WebLog wl

GROUP BY wl.id\_user

)Table\_2

WHERE Table\_1.ID\_USER = Table\_2.ID\_USER

)table\_3;");

# запрос - 6) Сколько в среднем заказов оформляется за день?

sqlQuery(con, "SELECT AVG(COUNT\_ORDER) AVG\_COUNT\_ORDER FROM

(

SELECT DATEPART(DAY, wl.date\_view) DAY, COUNT(wl.id\_user) COUNT\_ORDER

FROM WebLog wl

WHERE wl.action = 'order'

GROUP BY DATEPART(DAY, wl.date\_view)

)

tmpOrder;");

library(arules)

# Напишем запрос, чтобы выбрать тех пользователей, у которых в корзине больше одной книги

# сохраним результат в переменную books

books = sqlQuery (con, "SELECT wl.id\_user, wl.id\_book FROM

(SELECT wl.id\_user USER\_ID, COUNT(wl.id\_book) COUNT\_BOOK

FROM WebLog wl

WHERE wl.id\_book IS NOT NULL

GROUP BY wl.id\_user) tmpTable, WebLog wl

WHERE COUNT\_BOOK > 1

AND USER\_ID = wl.id\_user

AND wl.id\_book IS NOT NULL

ORDER BY USER\_ID");

tmp = books[1,1];

check = as.character(books[1,2]);

j = 0;

basket = data.frame();

for(i in 2:nrow(books))

{

if(books[i, 1] == tmp)

{

check = paste0(check, ",", as.character(books[i, 2]));

}

else

{

j = j + 1;

basket[j, 1] = tmp;

basket[j,2] = check;

tmp = books[i,1];

check = as.character(books[i, 2]);

}

}

j = j + 1;

basket[j, 1] = tmp;

basket[j,2] = check;

# формируем файл, где каждая строка содержит сведения о корзине, номера книг разделены запятыми

write.table(basket[,2], "trans.txt", quote=FALSE, row.names=FALSE, col.names=FALSE);

# прочтем данные из файла

trans = read.transactions("trans.txt", sep= ',');

library(arules)

# используем априорный алгоритм

rules = apriori(data=trans, parameter=list(support=0.01,confidence=0.6))

# распечатаем ассоциативные правила

inspect(rules)