dplyr Example 2 - Sessionize Web Events

Jim Porzak

2016-02-17

When attempting to understand customers from their observed behavior, recorded as events, it is typically useful to sessionize the event stream to correspond to a single customer engagement. Also these individual sessions can be aggregated to give a high level characterization of each customer.

I have been talking about these issues for a few years. See my achives here and here. Most recently, I presented Structuring Data for Self-serve Customer Insights at ODSC West in November 2015.

For this example, we use the infamous 3.5 million row AOL search data set which was used in a recent Stanford class. See also the AOL Readme.

The goal of this exercise is to roll-up the individual search events to

1. aol_sessions: one record for each series of contiguous searches summarizing the search activity in the sesison. The record is identified by the AnonID and a Session Sequence Number [1, 2, 3, â€¦].
2. aol_visitors: one record for each visitor summarizing the visitors over-all search activity.

Load the raw AOL search data

```
library(readr)
library(dplyr)
library(ggplot2)
library(knitr)
library(dplyrExamples)
fn <- "http://sing.stanford.edu/cs303-sp10/assignments/user-ct-test-collection-01.zip"
download.file(fn, "aolzip")
aol.tsv <- unzip("aolzip")
t0 <- Sys.time()
# fn <- system.file("extdata", "user-ct-test-collection-01.zip", package = "dplyrExamples")
aol <- read_tsv(aol.tsv)
(Elapsed <- Sys.time() - t0)
```

```
## Time difference of 8.96502 secs
```

```
glimpse(aol)
```

```
## Observations: 3,558,411
## Variables: 5
## $ AnonID    (int) 142, 142, 142, 142, 142, 142, 142, 142, 142, 142, 14...
## $ Query     (chr) "rentdirect.com", "www.prescriptionfortime.com", "st...
## $ QueryTime (time) 2006-03-01 07:17:12, 2006-03-12 12:31:06, 2006-03-1...
## $ ItemRank  (int) NA, NA, NA, NA, NA, NA, 1, NA, NA, NA, NA, NA, NA, N...
## $ ClickURL  (chr) NA, NA, NA, NA, NA, NA, "http://www.westchestergov.c...
```

Data set description from the AOL readme.

The data set includes {AnonID, Query, QueryTime, ItemRank, ClickURL}.

- AnonID - an anonymous user ID number.
- Query - the query issued by the user, case shifted with most punctuation removed.
- QueryTime - the time at which the query was submitted for search.
- ItemRank - if the user clicked on a search result, the rank of the item on which they clicked is listed.
- ClickURL - if the user clicked on a search result, the domain portion of the URL in the clicked result is listed.

Each line in the data represents one of two types of events:

1. A query that was NOT followed by the user clicking on a result item.
2. A click through on an item in the result list returned from a query.

In the first case (query only) there is data in only the first three columns/fields â€“ namely AnonID, Query, and QueryTime (see above).

In the second case (click through), there is data in all five columns. For click through events, the query that preceded the click through is included. Note that if a user clicked on more than one result in the list returned from a single query, there will be TWO lines in the data to represent the two events. Also note that if the user requested the next â€œpageâ€ or results for some query, this appears as a subsequent identical query with a later time stamp.

Sessionize into visitor sessions

The big idea is to model a visitorâ€™s search session. IOW, a visitor comes to the site and does one or more searches (with posible click-throughs) and then leaves the site. The goal is to summarize that activity in one record. Industry convention is a session ends if there is more that a 30 minute gap to the next event for a visitor. We will use that convention here.

For the purpose of this example, we are going to ignore data issues around multiple clicks per search and next page records.

This is the dplyr sequence you would use in production. The next section breaks down the process step-by-step.

```r
t0 <- Sys.time()
aol_sessions <- aol %>%
  arrange(AnonID, QueryTime) %>%
  group_by(AnonID) %>%
  mutate(Minutes_After_Last = difftime(QueryTime, lag(QueryTime), units = "mins"),
         New_Session_Flag = is.na(lag(AnonID)) | Minutes_After_Last > 30,
         Session_Seq_Num = cumsum(New_Session_Flag)
         ) %>%
  group_by(AnonID, Session_Seq_Num) %>%
  summarize(Session_Start_At = first(QueryTime),
            Number_Searches = n(),
            Number_Terms = n_distinct(Query),
            Session_Duration_Minutes = as.numeric(difftime(last(QueryTime), first(QueryTime),
                                                           units = "mins")),
            Number_Clicks = sum(!is.na(ClickURL))
            )
(Elapsed <- Sys.time() - t0)
```

```
## Time difference of 3.557838 mins
```

```
glimpse(aol_sessions)
```

```
## Observations: 1,069,200
## Variables: 7
## $ AnonID                   (int) 142, 142, 142, 142, 142, 142, 142, 14...
## $ Session_Seq_Num          (int) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12...
## $ Session_Start_At         (time) 2006-03-01 07:17:12, 2006-03-12 12:3...
## $ Number_Searches          (int) 1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 4, 2, 1...
## $ Number_Terms             (int) 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1...
## $ Session_Duration_Minutes (dbl) 0.0000000, 0.0000000, 0.2666667, 0.18...
## $ Number_Clicks            (int) 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0...
```

Step-by-step dplyr

Breaking down the above block of dplyr codeâ€¦

Ensure sorted by visitor ID and then query timestamp. Set up group_by() for following mutate().

It turns out the raw data file is already in this sort order, so the only visible change is the setting of the group AnonID.

```
glimpse(aol)
```

```
## Observations: 3,558,411
## Variables: 5
## $ AnonID    (int) 142, 142, 142, 142, 142, 142, 142, 142, 142, 142, 14...
## $ Query     (chr) "rentdirect.com", "www.prescriptionfortime.com", "st...
## $ QueryTime (time) 2006-03-01 07:17:12, 2006-03-12 12:31:06, 2006-03-1...
## $ ItemRank  (int) NA, NA, NA, NA, NA, NA, 1, NA, NA, NA, NA, NA, NA, N...
## $ ClickURL  (chr) NA, NA, NA, NA, NA, NA, "http://www.westchestergov.c...
```

```
groups(aol)
```

```
## NULL
```

```
aols <- aol %>%
  arrange(AnonID, QueryTime) %>%
  group_by(AnonID)
glimpse(aols)
```

```
## Observations: 3,558,411
## Variables: 5
## $ AnonID    (int) 142, 142, 142, 142, 142, 142, 142, 142, 142, 142, 14...
## $ Query     (chr) "rentdirect.com", "www.prescriptionfortime.com", "st...
## $ QueryTime (time) 2006-03-01 07:17:12, 2006-03-12 12:31:06, 2006-03-1...
## $ ItemRank  (int) NA, NA, NA, NA, NA, NA, 1, NA, NA, NA, NA, NA, NA, N...
## $ ClickURL  (chr) NA, NA, NA, NA, NA, NA, "http://www.westchestergov.c...
```

```
groups(aols)
```

```
## [[1]]
## AnonID
```

Append the session sequence number to each record

The session sequence number starts at 1 for each visitor and is incremented whenever the time interval from the last record is greater than 30 minutes. Figuring this out is done in three steps:

1. Compute the time lag, in minutes, from the prior record
2. Set a new session flag TRUE when
    - there is no prior record for the visitor (IOW, a new visitor ID), or
    - the lag to the prior record is greater than 30 minutes
3. Do a cumulative sum of the session flags to get the session sequence number

```
aols <- aols %>%
  mutate(Minutes_After_Last = difftime(QueryTime, lag(QueryTime), units = "mins"),
         New_Session_Flag = is.na(lag(AnonID)) | Minutes_After_Last > 30,
         Session_Seq_Num = cumsum(New_Session_Flag)
         )
glimpse(aols)
```

```
## Observations: 3,558,411
## Variables: 8
## $ AnonID             (int) 142, 142, 142, 142, 142, 142, 142, 142, 142...
## $ Query              (chr) "rentdirect.com", "www.prescriptionfortime....
## $ QueryTime          (time) 2006-03-01 07:17:12, 2006-03-12 12:31:06, ...
## $ ItemRank           (int) NA, NA, NA, NA, NA, NA, 1, NA, NA, NA, NA, ...
## $ ClickURL           (chr) NA, NA, NA, NA, NA, NA, "http://www.westche...
## $ Minutes_After_Last (dfft) NA mins, 1.615390e+04 mins, 7.728383e+03 m...
## $ New_Session_Flag   (lgl) TRUE, TRUE, TRUE, FALSE, TRUE, FALSE, TRUE,...
## $ Session_Seq_Num    (int) 1, 2, 3, 3, 4, 4, 5, 6, 7, 7, 8, 9, 9, 10, ...
```

```
kable(aols[14:30, -c(2, 4, 5)], caption = "Look at some interesting rows:")
```

Look at some interesting rows:

AnonID	QueryTime	Minutes_After_Last	New_Session_Flag	Session_Seq_Num
142	2006-04-08 01:31:04	6.005733e+03 mins	TRUE	10
142	2006-04-08 01:31:14	1.666667e-01 mins	FALSE	10
142	2006-04-08 08:38:23	4.271500e+02 mins	TRUE	11
142	2006-04-08 08:38:31	1.333333e-01 mins	FALSE	11
142	2006-04-08 08:38:42	1.833333e-01 mins	FALSE	11
142	2006-04-08 08:39:30	8.000000e-01 mins	FALSE	11
142	2006-04-09 02:19:24	1.059900e+03 mins	TRUE	12
142	2006-04-09 02:20:44	1.333333e+00 mins	FALSE	12
142	2006-04-13 00:25:27	5.644717e+03 mins	TRUE	13
142	2006-04-22 23:51:18	1.436585e+04 mins	TRUE	14
142	2006-05-06 08:49:34	1.925827e+04 mins	TRUE	15
142	2006-05-12 22:43:36	9.474033e+03 mins	TRUE	16
142	2006-05-18 09:21:57	7.838350e+03 mins	TRUE	17
142	2006-05-19 19:36:31	2.054567e+03 mins	TRUE	18
217	2006-03-01 11:58:51	NA	TRUE	1
217	2006-03-01 11:58:51	0.000000e+00 mins	FALSE	1
217	2006-03-01 14:06:23	1.275333e+02 mins	TRUE	2

Summarize by session sequence number within visitor

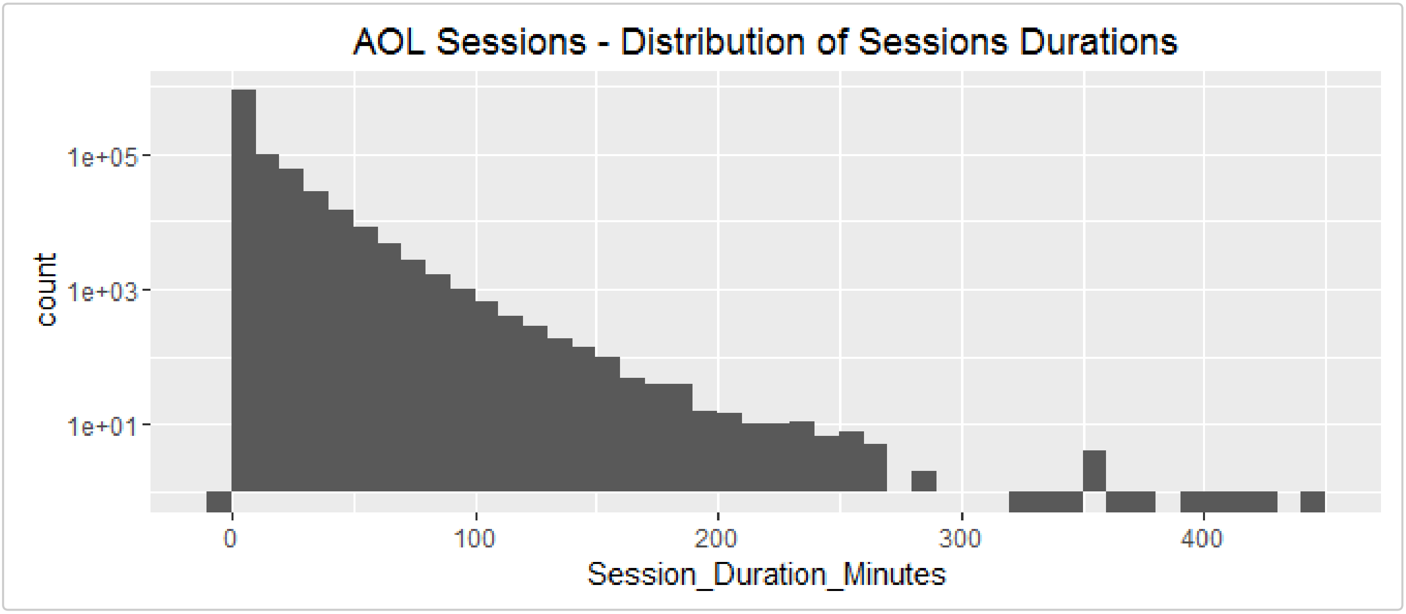
This final step is straightforward.

```
aols <- aols %>%
  group_by(AnonID, Session_Seq_Num) %>%
  summarize(Session_Start_At = first(QueryTime),
            Number_Searches = n(),
            Number_Terms = n_distinct(Query),
            Session_Duration_Minutes = difftime(last(QueryTime), first(QueryTime), units = "mins"),
            Number_Clicks = sum(!is.na(ClickURL))
            )
glimpse(aols)
```

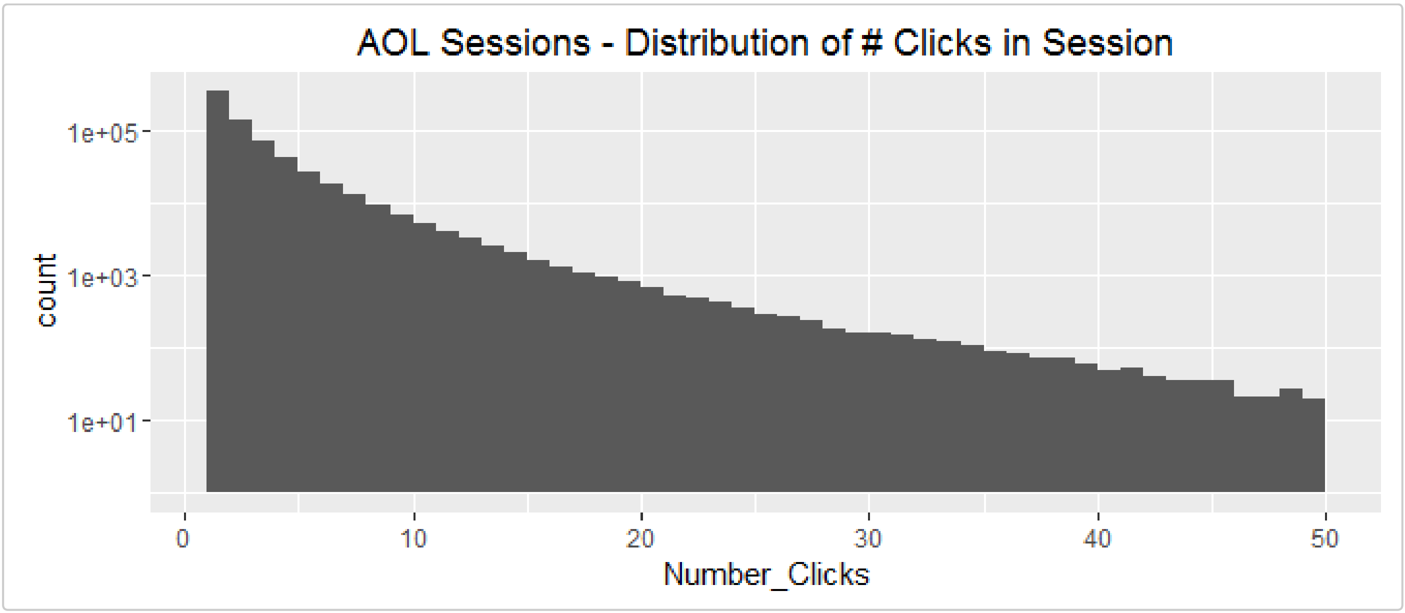
```
## Observations: 1,069,200
## Variables: 7
## $ AnonID                   (int) 142, 142, 142, 142, 142, 142, 142, 14...
## $ Session_Seq_Num          (int) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12...
## $ Session_Start_At         (time) 2006-03-01 07:17:12, 2006-03-12 12:3...
## $ Number_Searches          (int) 1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 4, 2, 1...
## $ Number_Terms             (int) 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1...
## $ Session_Duration_Minutes (dfft) 0.0000000 mins, 0.0000000 mins, 0.26...
## $ Number_Clicks            (int) 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0...
```

EDA of AOL Sessions

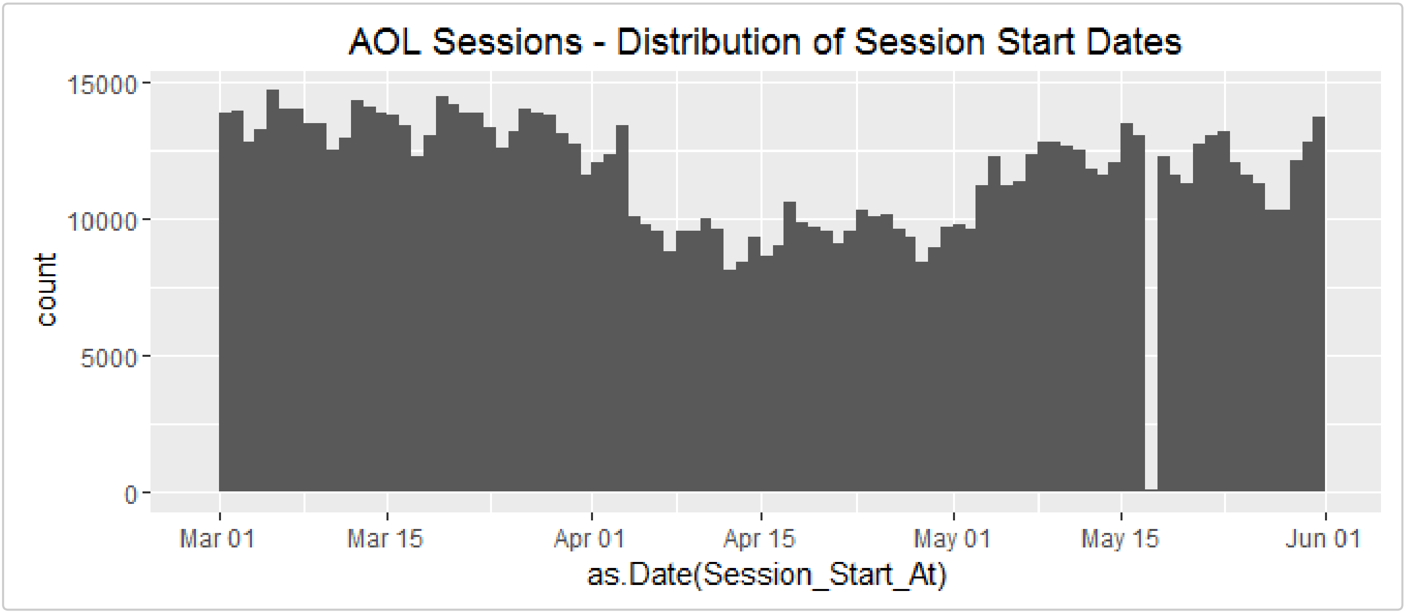
```
ggplot(aol_sessions, aes(Session_Duration_Minutes)) +
  geom_histogram(binwidth = 10) +
  ggtitle("AOL Sessions - Distribution of Sessions Durations") +
  scale_y_log10()
```



```
ggplot(aol_sessions, aes(Number_Clicks)) +
  geom_histogram(binwidth = 1) +
  ggtitle("AOL Sessions - Distribution of # Clicks in Session") +
  scale_y_log10() +
  xlim(1, 50)
```



```
ggplot(aol_sessions, aes(as.Date(Session_Start_At))) +
  geom_histogram(binwidth = 1) +
  ggtitle("AOL Sessions - Distribution of Session Start Dates")
```



Summarizing to Visitor Level

This is trivial and fast. We just group_by the visitor unique identifier and then use `summarize()` to create the visitor metrics we are interested in.

```
t0 <- Sys.time()
aol_visitors <- aol_sessions %>%
  group_by(AnonID) %>%
  summarize(Number_Sessions = n(),
            First_Session_At = min(Session_Start_At),
            Last_Session_At = max(Session_Start_At),
            Total_Duration_Minutes = as.numeric(sum(Session_Duration_Minutes)),
            Avg_Duration_Minutes = as.numeric(mean(Session_Duration_Minutes)),
            Median_Duration_Minutes = as.numeric(median(Session_Duration_Minutes)),
            Avg_Num_Searches = mean(Number_Searches),
            Median_Num_Searches = median(Number_Searches),
            Avg_Num_Clicks = mean(Number_Clicks),
            Median_Num_Clicks = median(Number_Clicks)
            )
(Elapsed <- Sys.time() - t0)
```

```
## Time difference of 27.48149 secs
```

```
glimpse(aol_visitors)
```

```
## Observations: 65,516
## Variables: 11
## $ AnonID                  (int) 142, 217, 993, 1268, 1326, 1337, 1410,...
## $ Number_Sessions         (int) 18, 22, 3, 18, 25, 14, 9, 17, 50, 29, ...
## $ First_Session_At        (time) 2006-03-01 07:17:12, 2006-03-01 11:58...
## $ Last_Session_At         (time) 2006-05-19 19:36:31, 2006-05-23 15:41...
## $ Total_Duration_Minutes  (dbl) 1.628333e+01, 1.988333e+01, 8.333333e-...
## $ Avg_Duration_Minutes    (dbl) 0.90462963, 0.90378788, 0.02777778, 0....
## $ Median_Duration_Minutes (dbl) 0.0000000, 0.0000000, 0.0000000, 0.000...
## $ Avg_Num_Searches        (dbl) 1.500000, 1.318182, 1.333333, 1.333333...
## $ Median_Num_Searches     (dbl) 1, 1, 1, 1, 1, 2, 1, 1, 1, 3, 1, 2, 4,...
## $ Avg_Num_Clicks          (dbl) 0.1666667, 0.4545455, 0.3333333, 0.222...
## $ Median_Num_Clicks       (dbl) 0.0, 0.0, 0.0, 0.0, 0.0, 1.5, 0.0, 1.0...
```

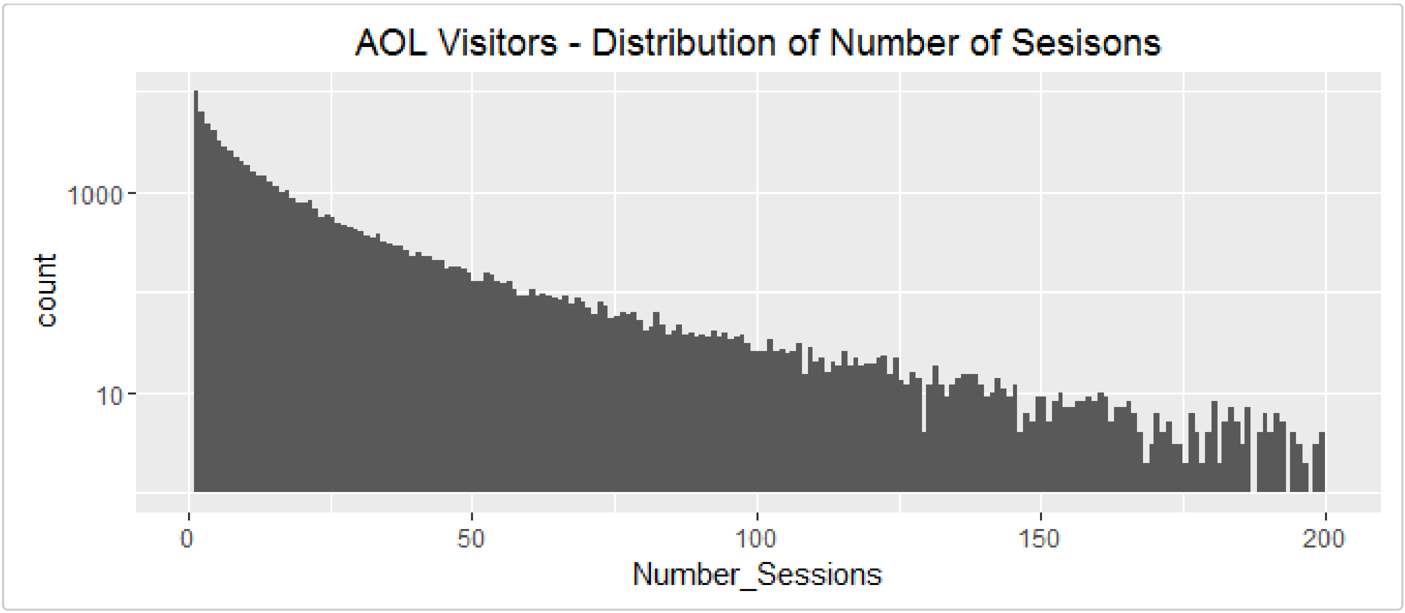
EDA of AOL Visitors

Basic metrics for the AOL Search data set:

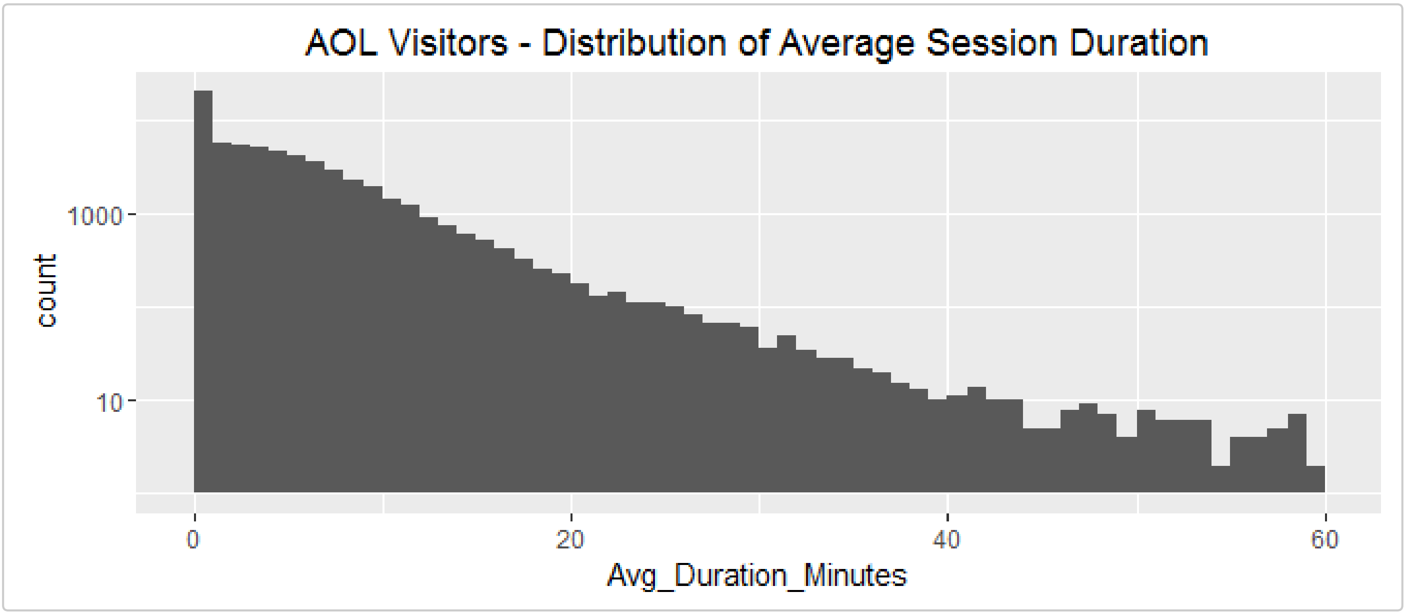
- Number of visitors: 65516
- Number of sessions: 1069200
- Number of records: 3558411
- First search at: 2006-03-01 00:01:53
- Last search at: 2006-05-31 23:59:58

Distributions of visitor properties:

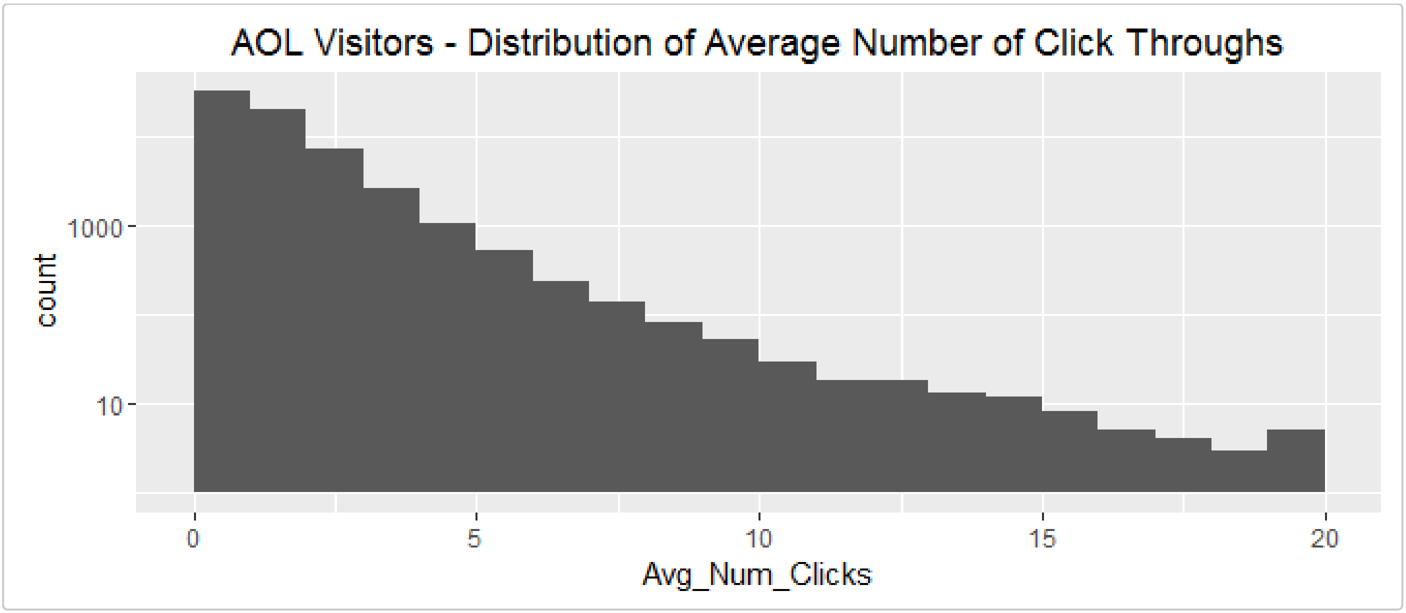
```
ggplot(aol_visitors, aes(Number_Sessions)) +
  geom_histogram(binwidth = 1) +
  ggtitle("AOL Visitors - Distribution of Number of Sesisons") +
  scale_y_log10() +
  xlim(1, 200)
```



```r
ggplot(aol_visitors, aes(Avg_Duration_Minutes)) +
  geom_histogram(binwidth = 1) +
  ggtitle("AOL Visitors - Distribution of Average Session Duration") +
  scale_y_log10() +
  xlim(0, 60)
```



```r
ggplot(aol_visitors, aes(Avg_Num_Clicks)) +
  geom_histogram(binwidth = 1) +
  ggtitle("AOL Visitors - Distribution of Average Number of Click Throughs") +
  scale_y_log10() +
  xlim(0, 20)
```



Learning More

The place to start, of course, is Hadleyâ€™s vignettes in the dplyr package. Especially Introduction to dplyr and Window functions and grouped mutate/filter.

Now that Hadley is with RStudio, search their blog for dplyr; get the Data Wrangling Cheat Sheet; watch Data Wrangling with R & RStudio. To understand Hadleyâ€™s current thinking about data analysis watch Pipelines for Data Analysis in R and The Grammar and Graphics of Data Science - the latter with Winston Chang.

Lastly, see Garrett & Hadleyâ€™s chapter on data transform in their upcoming R for Data Science

To compare the dplyr windowing method with how it works in SQL see this simple example or Google â€˜â€œpartition byâ€ sqlâ€™, perhaps replacing â€˜sqlâ€™ with your favorite DBMS; e.g. â€˜postgresqlâ€™, â€˜redshiftâ€™, etc.

Conclusion

This example did a full refresh of the aol_session and aol_visitor. In a production environment where new data come in, say, nightly we would only sessionize the new records and append them to existing records, if any, for the visitors in the nightly set.

(The brute force method is to let sessions end at the end of each batch load. Generally this is acceptable. You can check this on the fully processed sessions by seeing how many actually span the cut off time, eg midnight.)

Then update just those visitor level summaries for visitor IDs which were in the incremental batch.

I hope this dplyr example inspired you to add the library to your regular took set.

Please send comments and suggestions to Jim at DS4CI.org or leave an issue or pull request at my github.

Thanks! Jim

Remember to clean house!

```r
file.remove("user-ct-test-collection-01.txt")
```

```
## [1] TRUE
```

```r
file.remove("aolzip")
```

```
## [1] TRUE
```