Lecture 18:

Last time

We discussed the data model underlying relational databases and compared it to the relatively flat model R uses for data frames.

We introduced SQL and described how to query databases; we illustrated SQL queries with the RMySQL package and pulled the resulting tables back to R.

In the process we started to explore a large data set accumulated by the Reality Mining project at MIT’s Media Lab.

Today

We continue with our treatment of relational databases and expand the kinds of queries we make to the Reality Mining data.

We will review the SQL commands we’ve seen so far and move on to deal with multiple tables; we will try to reconstruct the results from the Reality Mining group.

But first (since we are near the end of the quarter), a comment from last time on databases and media art...

Databases and culture

Over the last decade, artists have been exploring the idea of databases, of archives, of what it means to store and organize data.

Some of these works can be playful, some let you experience data in new ways; many depend on novel methods for “visualization.”
In this piece, the computer is used to make a catalog, or database of every shot of many shows. Each shot is then indexed according to the categories seen on the shelf of Video CDs. In some ways, the result is scrambled, in other ways it is highly ordered according to the logic of the database.

Every shot/every episode, Jennifer and Kevin McCoy, 2001

20 episodes of “Starsky and Hutch” divided by scene into 300 categories

“Every Anvil”, Jennifer and Kevin McCoy, 2001

“448 is enough”, Jennifer and Kevin McCoy, 2002
Dark Source is an artwork that shows the inner workings of a commercial electronic voting machine, the Diebold AccuVote-TS™ touch-screen voting terminal that has recently been adopted in many U.S. states. The artwork presents over 2,000 pages of software code, a printout of 49,609 lines of C++ that constitute version 4.3.1 of the AccuVote-TS™ source code...

Calling its source code a trade secret, Diebold has asserted its proprietary interest in protecting its intellectual property. Therefore in Dark Source the code, which had been obtained freely over the internet following a 2002 security failure at Diebold, has been blacked out in its entirety in order to comply with trade secrecy laws.
The authors conducted an exhaustive empirical study, with the aid of custom software, public search engines and powerful statistical techniques, in order to determine the relative popularity of every integer between 0 and one million. The resulting information exhibits an extraordinary variety of patterns which reflect and refract our culture, our minds, and our bodies...

For example, certain numbers, such as 212, 486, 911, 1040, 1492, 1776, 68040, or 90210, occur more frequently than their neighbors because they are used to denominate the phone numbers, tax forms, computer chips, famous dates, or television programs that figure prominently in our culture. Regular periodicities in the data, located at multiples and powers of ten, mirror our cognitive preference for round numbers in our biologically-driven base-10 numbering system. Certain numbers, such as 12345 or 8888, appear to be more popular simply because they are easier to remember.

President Bush’s recent assertion that North Korea, Iraq and Iran form an “Axis of Evil” was more than a calculated political act — it was also an imaginatively formal, geometric one, which had the effect of erecting a monumental, virtual, globe-spanning triangle.

Axis is an online tool intended to broaden opportunities for similar kinds of Axis creation. It allows its participants to connect any three points in space (countries) into a new Axis of their own design. With the help of multidimensional statistical metrics culled from international public databases[3], the commonalities amongst the user’s choices are revealed. In this manner, Axis presents an inversion of Bush’s praxis, obtaining lexico-political meaning from the formal act of spatial selection.

CARNIVORE.WEBCAM.SNIFFING combines both optical-(webcam) and data-surveillance (carnivore) tools into a realtime augmented-reality installation. The supervised person is removed out of the video-image and replaced by his digital personality, generated out of his own digital activity.
Database and culture

This is a very brief and extremely biased introduction to some interesting works that engage the concept of databases and programming.

Notice that many call for new forms of representation, and that in some cases the presentation is not purely virtual or screen-based.

But back to what we’re really after today...

Relational databases

We saw last time that a relational database can be thought of broadly as a series of tables or relations.

In our Reality Mining Example, these tables referred to users, devices, cell towers, and various kinds of activities like calls made and received, time spent near other users or time spent under the shadow of a particular cell tower.
Reality Mining

There are several tables related to the objects under study; in some cases data have been removed to protect the privacy of the participants

\begin{itemize}
  \item \textbf{cellname}: oid, name, person_oid, celltower_oid
  \item \textbf{celltower}: oid, name
  \item \textbf{device}: oid, macaddr, person_oid, name
  \item \textbf{person}: name, password, email, phonenumber_oid,...
  \item \textbf{coverspan}: oid, number
\end{itemize}

Reality Mining

In addition, there are several tables related to events that were captured during the monitoring period

\begin{itemize}
  \item \textbf{activityspan}: oid, endtime, starttime, person_oid
  \item \textbf{callspan}: oid, endtime, starttime, person_oid, phonenumber_oid, callid, contact, description, direction, duration, number, status, remote
  \item \textbf{cellspan}: oid, endtime, starttime, person_oid, celltower_oid
  \item \textbf{coverspan}: oid, endtime, starttime, person_oid
  \item \textbf{devicespan}: oid, endtime, starttime, person_oid, device_oid
\end{itemize}

SQL Review

Last time we introduced the Structured Query Language (SQL) and, in particular, discussed the \texttt{SELECT} statement that is used to retrieve information from relational databases

The general form of the \texttt{SELECT} statement is

\begin{verbatim}
SELECT column(s)
FROM relation(s)
WHERE constraints;
\end{verbatim}

and we presented a number of different constructions for setting constraints; we compared these specifically to the subsetting rules we are familiar with in R
SQL Review

SQL is primarily designed for data retrieval; it is not a computational language nor is it a statistical language. It does, however, contain certain summarizing capabilities in the form of functions that can be applied over rows in a table; the GROUP BY can be used to identify subsets to apply these over also.

- **COUNT**: returns the number of tuples (rows)
- **SUM**: the total for the attribute
- **AVG**: the average for the attribute
- **MIN**: the minimum across the attribute
- **MAX**: the maximum across the attribute
Relational databases

This example illustrates that it is often better to perform the summarization and aggregation inside the database rather than copying and manipulating potentially large objects in R.

When we bring data into R from a DMBS, we convert data from the database into formats that R understands; while we know about the kinds of data R handles, here is a sample of what our MySQL database can store:

- **BOOLEAN**
- **DATE**
- **INT**
- **FLOAT**
- **DOUBLE**
- **CHAR**
- **DATETIME**
- **TIMESTAMP**
- **TEXT**
- **TIME**
- **YEAR**

An example

Here is the structure or schema for the table `callspan`; we can see the names of the columns, the type of data they hold; what do you notice?

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>callid</td>
<td>int</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>timestamp</td>
<td>timestamp</td>
<td>Yes</td>
<td>0000-00-00 00:00:00</td>
</tr>
<tr>
<td>duration</td>
<td>int</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>handover</td>
<td>bit</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>location</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>callstate</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>terminated</td>
<td>bit</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>ended</td>
<td>date</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>start</td>
<td>date</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>duration</td>
<td>int</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>channel</td>
<td>int</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>remote</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>orig</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>dest</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>origdata</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>destdata</td>
<td>varchar</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>origtime</td>
<td>datetime</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>desttime</td>
<td>datetime</td>
<td>No</td>
<td>0</td>
</tr>
</tbody>
</table>

Relational databases

The analog to an NA in a database is the value NULL; note that aggregation functions remove all the NULLS before operating and operating on columns with NULLS produces NULLS.

Here is another example of how we can use these functions to provide something like `summary` in R.

Note again that SQL allows for simple mathematical operations; we can do a lot to format the data before it is returned to R.
Multiple relations

Consider the following questions we might ask of the data:

Find the amount of time people spend near other people in the study

Find the amount of time they are associated with cell towers they have identified as “Work”

Notice that in these cases, we have to combine data contained in multiple tables; this is an important component of the relational model

Recall that queries like those above are presented to the database in a special language called SQL

Joining relations

Recall that keys uniquely identify the rows (tuples) in a table (relation); we use these keys to merge (join) data in different tables (relations)

In the case of the Reality Mining database, each table has a column labeled `oid` that represents the key; in this case it’s nearly trivial in the sense that the `oid` is basically a row number

Suppose we want to get a table of when each person in the study was near another according based on their mobile phone’s Bluetooth record...
Joining relations

In this join, we basically replicated the data from device and added it to devicespan using device_oid as the key and returned the larger table.

Suppose we are not after all that information, but instead just the columns associated with when users were close to each other.

Remember that LIMIT restricts output to just the number of lines you request; this is a handy way to "test the waters" on queries that might take a lot of time to answer.

Sub-queries

If we only wanted information from devicespan that corresponded to devices owned by other users in the study, we could use a sub-query.

This construction allows us to assemble information from another table temporarily; these kinds of constructions can be nested beyond the simple application in the next example.
From the Reality Mining paper

We can now do something simple from the Reality Mining paper; we have extracted each time a person in the study is nearby another person in the study.

Let's tabulate these encounters by day of the week and look for trends; first, a brute force approach, knowing what we know from R.

But...

If there's one thing that databases are good at, it's dates.

In fact, many databases are constantly updated as transactions occur; in these settings, it can be crucial to be able to manipulate timestamps easily.

Here is another example that generates the same data as before, but only brings over to R a small table, not 114K records.
Temporary tables and views

Sometimes, our processing is easier if we construct temporary tables in the database to hold intermediate constructions.

In many databases, this would be accomplished with the SQL statement `CREATE VIEW`; our MySQL server supports something similar `CREATE TEMPORARY TABLE`.

Of course, if these tables are small enough, we could decide to bring them into R and merge them there; here is a simple example comparing incoming and outgoing average call durations.
Closure

Again, this has only been a very brief introduction to relational databases and SQL

Hopefully, you have now seen enough to perform basic manipulations in a database

Our particular DBMS, MySQL has a lot of online help so you can learn more about different functions that can help you handle as many big computations in the database as possible