Statistical Computing (36-350)

Databases II

Cosma Shalizi and Vincent Vu

November 30, 2011

Presentations

- 10~12 minutes each group
- E-mail slides (PDF or Keynote) to vqv+teaching@cmu.edu (subject: 36-350 Presentation) by Sunday 11:59PM
- Be prepared to present Monday 10:30AM

Presentations

- Explain
  - what your project is about
  - how you solved/will solve problems
  - why you made certain choices
  - where you are now

Agenda

- SQL
  - More on aggregation
  - Joining tables
- Accessing databases from R with DBI

Note from last time

- **Q:** "Is it possible to select all columns except one?"
- **A:** Yes, but in general a bad practice
- Be explicit. Select only columns that you need – databases can be huge!

Aggregation

- Use GROUP BY clause to perform calculations on groups
- Just like split-apply-combine
  - Group rows in table according to some criterion
  - Perform some computation
  - Return a table with results

Aggregation

```
sqlite> SELECT playerID, SUM(salary) FROM Salaries GROUP
BY playerID
aardsda01|4259750.0
aasedo01|2300000.0
abadan01|327000.0
abbotje01|985000.0
abbotji01|12960500.0
abbotku01|4237000.0
abbotky01|259000.0
abbotpa01|6471000.0

...
```

Total salary by player

Aggregation

```
sqlite> SELECT playerID, SUM(salary) AS totalSalary FROM
Salaries GROUP BY playerID ORDER BY totalSalary DESC
LIMIT 10;
rodrial01|264416252.0
jeterde01|205430000.0
ramirma02|204807769.0
bondsba01|188245322.0
johnsra05|175550019.0
sheffga01|168008550.0
maddugr01|153845000.0
griffke02|151703682.0
delgaca01|146299000.0
martipe02|146259585.0
```

Top 10 total salaries

Selecting rows while aggregating

- **WHERE** clause is applied pre-aggregation.
- Select subsets of rows post-aggregation with **HAVING** clause

Use HAVING to subset results of aggregation

```
sqlite> SELECT playerID, SUM(salary) AS totalSalary FROM
Salaries GROUP BY playerID HAVING totalSalary >
200000000;
jeterde01|205430000.0
rodrial01|264416252.0
ramirma02|204807769.0
```

Return groups having “total Salary > $200 million”

Joining Tables

- Data often divided across tables
- Join tables to fetch data from multiple tables
  - Use INNER JOIN to merge tables based on a **key**
  - Other types of joins:
    - OUTER, OUTER LEFT, OUTER RIGHT

Inner Join

patients

last_name	first_name	physician_id
Doe	John	34
Brown	Charlie	55
Morgan	Dexter	55
Vu	Vince	

physicians

physician_id	last_name	first_name
34	Jekyll	Henry
55	House	Gregory

Inner Join

patients

last_name	first_name	physician_id
Doe	John	34
Brown	Charlie	55
Morgan	Dexter	55
Vu	Vince	

physicians

physician_id	last_name	first_name
34	Jekyll	Henry
55	House	Gregory

Inner Join

Joined table

patients. last_name	patients. first_name	patients. physician_id	physicians. last_name	physicians. first_name	physicians. physician_id
Doe	John	34	Jekyll	Henry	34
Brown	Charlie	55	House	Gregory	55
Morgan	Dexter	55	House	Gregory	55

Note that the row corresponding to
"Vince Vu" was dropped

Inner Join

- Cartesian Product:
  - Combine every row of Table A with every row of Table B
  - Select rows that satisfy the JOIN predicate
    - In previous example, JOIN predicate is "patients.physician_id == physicians.physician_id"

Example

- Biographic information in table ‘Master’
- Batting statistics in table ‘Batting’
- Salary information in table ‘Salaries
- **Problem**: Get player name, batting statistics, and salaries
- **Solution**: Join tables using `playerID` key

Batting stats and player name

```
SELECT nameLast,nameFirst,yearID,AB,H
FROM Master INNER JOIN Batting USING(playerID);
```

Batting stats and player name

```
SELECT nameLast,nameFirst,yearID,AB,H
FROM Master INNER JOIN Batting USING(playerID);
```

This part “creates” a table containing columns from Master and Batting where playerID matches both

Batting stats and player name

```
SELECT nameLast,nameFirst,yearID,AB,H
FROM Master INNER JOIN Batting USING(playerID);
```

```
sqlite> SELECT nameLast,nameFirst,yearID,AB,H FROM Master
INNER JOIN Batting USING(playerID) LIMIT 10;
Aaron|Hank|1954|468|131
Aaron|Hank|1955|602|189
Aaron|Hank|1956|609|200
Aaron|Hank|1957|615|198
Aaron|Hank|1958|601|196
Aaron|Hank|1959|629|223
Aaron|Hank|1960|590|172
Aaron|Hank|1961|603|197
Aaron|Hank|1962|592|191
Aaron|Hank|1963|631|201
...
```

Equivalently

```
SELECT nameLast,nameFirst,yearID,AB,H
FROM Master INNER JOIN Batting
ON Master.playerID == Batting.playerID;
```

This form is useful when the key column has a different name in each table

Natural Join

- When two tables have exactly one column in common, INNER JOIN can be replaced by NATURAL JOIN and USING() can be omitted
- *Resulting table willy have one copy of the common column

Natural Join

patients

last_name	first_name	physician_id
Doe	John	34
Brown	Charlie	55
Morgan	Dexter	55
Vu	Vince	

physicians

physician_id	last_name	first_name
34	Jekyll	Henry
55	House	Gregory

Natural Join

Joined table

patients. last_name	patients. first_name	physicians. last_name	physicians. first_name	physician_id
Doe	John	Jekyll	Henry	34
Brown	Charlie	House	Gregory	55
Morgan	Dexter	House	Gregory	55

Note that there is only one copy of "physician_id"

Batting stats and player name

```
SELECT nameLast,nameFirst,yearID,AB,H
FROM Master NATURAL JOIN Batting;
```

```
sqlite> SELECT nameLast,nameFirst,yearID,AB,H FROM Master
NATURAL JOIN Batting;
Aaron|Hank|1954|468|131
Aaron|Hank|1955|602|189
Aaron|Hank|1956|609|200
Aaron|Hank|1957|615|198
Aaron|Hank|1958|601|196
Aaron|Hank|1959|629|223
Aaron|Hank|1960|590|172
Aaron|Hank|1961|603|197
Aaron|Hank|1962|592|191
Aaron|Hank|1963|631|201
...
```

Multiple joins

```
SELECT nameLast,nameFirst,yearID,AB,H,salary
FROM Master NATURAL JOIN Batting NATURAL JOIN
Salaries ORDER BY salary DESC LIMIT 10;
```

Rodriguez	Alex	2009	444	127	33000000.0
Rodriguez	Alex	2010	522	141	33000000.0
Rodriguez	Alex	2008	510	154	28000000.0
Rodriguez	Alex	2005	605	194	26000000.0
Sabathia	C.C.	2010	5	1	24285714.0
Ramirez	Manny	2009	352	102	23854494.0
Giambi	Jason	2007	254	60	23428571.0
Giambi	Jason	2008	458	113	23428571.0
Rodriguez	Alex	2007	583	183	22708525.0
Jeter	Derek	2010	663	179	22600000.0

Dealing with name collisions

- When two tables have columns with same names, refer to each with **table.var** notation
- Rename variables using the **AS** operator

patients and doctors both have last_name

```
SELECT patients.last_name AS patname,
       physicians.last_name AS docname
FROM patients NATURAL JOIN physicians;
```

Use AS to make it easier to refer to multiple tables

```
SELECT p.last_name AS patname,
       d.last_name AS docname
FROM patients AS p NATURAL JOIN physicians AS d;
```

Accessing Databases with R and DBI

DBMS

- Database Management System
- Software package for operating a database – can include server
- **Problem**: Different systems, different software, different interfaces

Not necessary to learn every different system

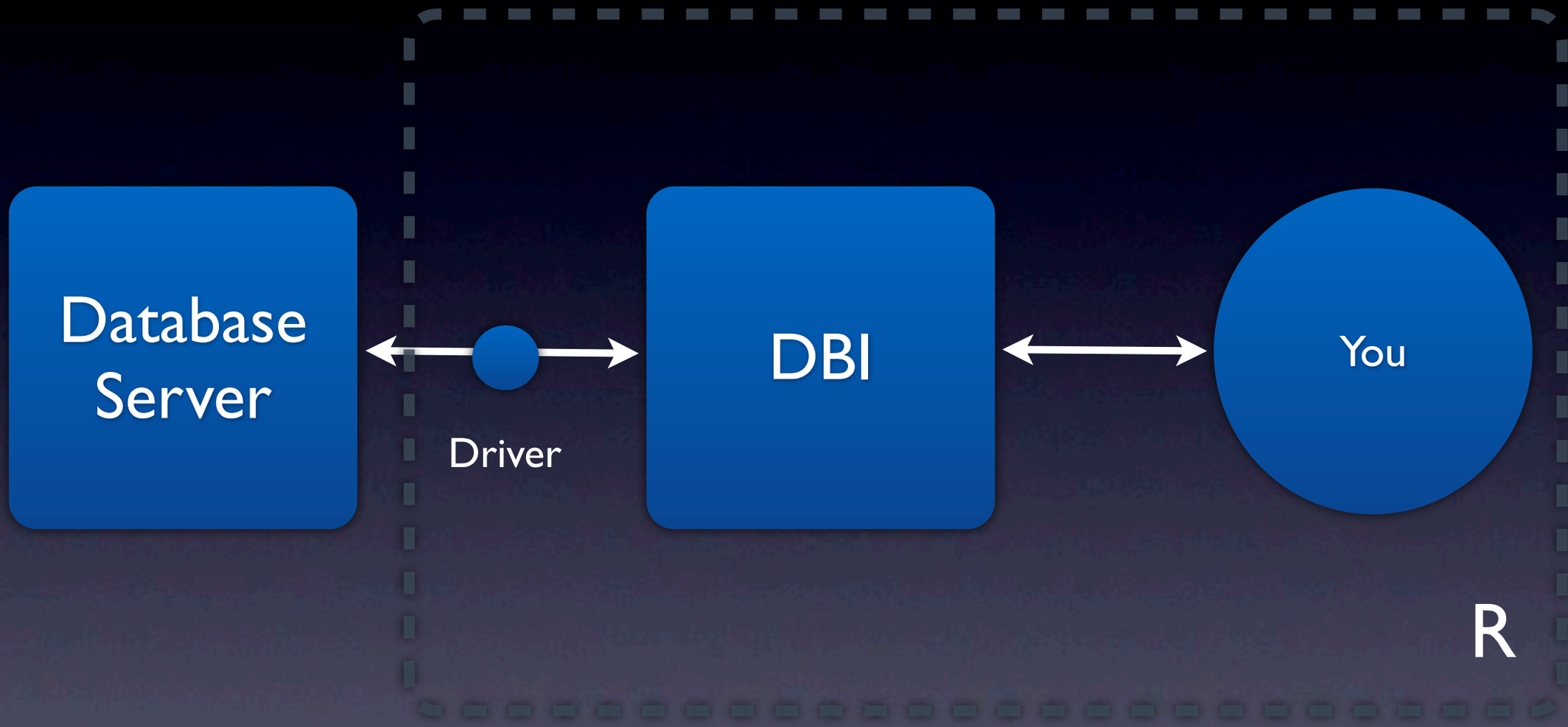
DBI

- **D**ata**b**ase **I**nterface for R
- Decouples front-end and back-end of connectivity to the DBMS
- Ensures that interface that you see, is same regardless of what specific DBMS you connect to.

DBI

```
> install.packages('DBI', dependencies = T)
```

Driver allows DBI to talk to specific DBMS



DBMS could be Oracle, MySQL, PostgreSQL, ...

Driver

- Handles back-end communication with specific Database Server.
- Drivers available for
  - Oracle, MySQL, PostgreSQL, SQLite, ...
- See http://cran.r-project.org/package=DBI

Example: SQLite

```
> install.packages('RSQLite', dependencies = T)
```

DBI Device Driver for SQLite

How to use DBI

1. Load proper DBMS driver
2. Establish a connection to the DBMS
3. *Interact with DBMS*
4. Disconnect from DBMS
5. Unload driver

Load driver and connect

```
library(RSQLite)

# Instantiate driver and open a connection to the db
drv <- dbDriver('SQLite')
con <- dbConnect(drv, dbname = 'baseball.db')
```

In this example we connect to an SQLite database

Load driver and connect

```
library(RMySQL)

# Instantiate driver and open a connection to the db
drv <- dbDriver('RMySQL')
con <- dbConnect(drv, dbname = 'mydb',
                 use = 'vqv', password = 'secret',
                 host = 'sql.vincevu.com')
```

In this example we connect to a MySQL database

Basic DBI Functions

- **dbListTables()** – List tables in the db
- **dbListFields()** – List columns in table
- **dbReadTable()** – Import table as data frame
- **dbGetQuery()** – Send SQL query and get data frame as result

List tables in the database

```
> dbListTables(con)
 [1] "AllstarFull"          "Appearances"          "AwardsManagers"
 [4] "AwardsPlayers"        "AwardsShareManagers"  "AwardsSharePlayers"
 [7] "Batting"              "BattingPost"          "Fielding"
[10] "FieldingOF"           "FieldingPost"         "HallOfFame"
[13] "Managers"             "ManagersHalf"         "Master"
[16] "Pitching"             "PitchingPost"         "Salaries"
[19] "Schools"              "SchoolsPlayers"       "SeriesPost"
[22] "Teams"                "TeamsFranchises"      "TeamsHalf"
[25] "sqlite_sequence"      "xref_stats"
```

List columns in a table

```
> dbListFields(con, 'Salaries')
[1] "yearID"   "teamID"   "lgID"   "playerID" "salary"
```

Read entire table as data frame

```
> df <- dbReadTable(con, 'Salaries')
> head(df)
  yearID teamID lgID  playerID salary
1   1980    TOR   AL stiebda01  55000
2   1981    NYA   AL jacksre01 588000
3   1981    TOR   AL stiebda01  85000
4   1982    TOR   AL stiebda01 250000
5   1983    TOR   AL stiebda01 450000
6   1984    TOR   AL stiebda01 650000
```

Get results of SQL query

```
> dbGetQuery(con, 'SELECT * FROM Salaries LIMIT 10')
yearID teamID lgID  playerID salary
1    1980    TOR   AL stiebda01  55000
2    1981    NYA   AL jacksre01 588000
3    1981    TOR   AL stiebda01  85000
4    1982    TOR   AL stiebda01 250000
5    1983    TOR   AL stiebda01 450000
6    1984    TOR   AL stiebda01 650000
7    1985    ATL   NL barkele01 870000
8    1985    ATL   NL bedrost01 550000
9    1985    ATL   NL benedbr01 545000
10   1985    ATL   NL  campri01 633333
```

Use paste() to construct complex queries

```
> df <- dbGetQuery(con, paste(
    'SELECT nameLast,nameFirst,yearID,salary ',
    'FROM Master NATURAL JOIN Salaries')
)
> head(df)
  nameLast nameFirst yearID salary
1     Aase       Don   1986 600000
2     Aase       Don   1987 625000
3     Aase       Don   1988 675000
4     Aase       Don   1989 400000
5     Abad      Andy   2006 327000
6   Abbott      Jeff   1998 175000
```

Disconnect and unload driver

```
# Close the connection and unload the driver
dbDisconnect(con)
dbUnloadDriver(drv)
```

Remember to clean-up!

Trade-offs

- **Q**: Why not just read all tables in the database into R and do the merging, selecting, etc... within R?
- **A**: Tables can be huge! More efficient to do simple computations, reductions on the server.

Trade-offs

- Best solutions are a mix of both R and SQL

Summary

- Join tables to get columns from different tables
- DBI provides a uniform interface within R for accessing databases**Next**: Subqueries in SQL (in Lab)