Sorting and restricting data

Suman Sahil, Steve Simon

Creation date: 2019-08-09

Description of the fat database

- The fat database
  - Single table, fat
  - Two measures of percentage body fat
  - Compare to various body dimensions
- You can find a description of this data set at
  - http://jse.amstat.org/datasets/body.txt

We will work with the database fat.sqlite. The name of the database is the same as the name of the only table in the database.

Selecting a limited number of records (1/2)

– SQL code (SQLite)

```
select *
  from fat
  limit 5
```

– SQL code (Oracle)

```
select *
  from fat
  where rownum <= 5
```

– Other databases

```
???
```

Selecting a limited number of records is one area where different databases use different approaches. SQLite uses a limit statement. Oracle uses a rownum variable in the where statement. Other databases use a keyword like top or first.

This is just one example of where the standards committees have not caught up with the database designers.

Selecting a limited number of records (2/2)

```
##   case_number age  bmi
## 1           1  23 23.7
## 2           2  22 23.4
## 3           3  22 24.7
## 4           4  26 24.9
## 5           5  24 25.6
```

Here's what the results looks like.

Warning!!!

– Order of records in a database is arbitrary
– You may prefer a random sample

Now the order in which the records are stored in a database is somewhat arbitrary and there is no guarantee that you will get the exact same rows, especially if the database is being regularly updated.

It does help to pick the first few records to get a rough idea of what your data table looks like, but there is a serious risk here. Often in a database, the weird or unusual records either float to the top of a table or sink to the bottom. You have no guarantee that there is any resemblance between the first few records and the parts of the table beyond the first few records.

You may, at times, prefer to select a random sample of records. You'll see how to do this in a future lecture.

Select a specific record (1/2)

– SQL code

```
select case_number, age, bmi
  from fat
  where case_number=3
```

The fat dataset has a primary key, case_number, that is a sequential number starting at 1 and going to the last record (252). This is a nice thing to have, but you may not have it in most databases. It makes it very easy to pick out a specific record like the third one.

Select a specific record (2/2)

```
##   case_number age  bmi
## 1           3  22 24.7
```

Here's what the results looks like.

Select a range of records (1/2)

– SQL code

```
select case_number, age, bmi
  from fat
  where case_number BETWEEN 110 and 113
```

You can select a range of values with the BETWEEN keyword.

Select a range of records (2/2)

```
##   case_number age  bmi
## 1         110  40 24.7
## 2         111  43 25.6
## 3         112  43 26.3
## 4         113  47 25.6
```

Here's what the results looks like.

Select a mix of records (1/2)

– SQL code

```
SELECT case_number, age, bmi
  FROM fat
  WHERE case_number IN (207, 209, 214)
```

You can use the IN keyword to select from a non-contiguous list of values.

Select a mix of records (2/2)

```
##   case_number age  bmi
## 1         207  44 27.2
## 2         209  47 22.9
## 3         214  50 27.4
```

Here's what the results looks like.

Select on other variables (1/2)

– SQL code

```
SELECT case_number, age, bmi
  FROM fat
  WHERE age >= 72
```

You can select on any variable, not just the case_number

Select on other variables (2/2)

```
##   case_number age  bmi
## 1          79  81 23.0
## 2          85  72 24.7
## 3          87  72 24.6
## 4         249  72 29.1
## 5         250  72 30.2
## 6         251  72 27.0
## 7         252  74 29.8
```

Here's what the results looks like.

Selections with Boolean logic (1/2)

– SQL code

```
SELECT case_number, age, bmi
  FROM fat
  WHERE
    age >= 72 AND
    bmi >= 30
```

You can use standard Boolean logic with keywords like AND, OR, and NOT

Selections with Boolean logic (2/2)

```
##   case_number age  bmi
## 1         250  72 30.2
```

Here's what the results looks like. Notice that there is only one record because the "and" requires both statements to be true before a record is selected.

Selections involving character data

- Re-introduce airlines data set
  - Airline is a variable width character data type.

The fat data set has all numeric data. Selection using character data has a few distinctions.

Remember what the full table looks like

```
##       Airline b2017 b2016 r2017 r2016
## 1       DELTA   679   912  0.07  0.09
## 2      VIRGIN   165    77  0.27  0.13
## 3     JETBLUE  1475  2140  0.54  0.82
## 4      UNITED  2067  2874  0.30  0.45
## 5    HAWAIIAN    92    30  0.11  0.04
## 6  EXPRESSJET   785  2541  0.67  1.58
## 7     SKYWEST   917  2177  0.37  0.96
## 8    AMERICAN  4517  6598  0.46  0.66
## 9      ALASKA   658   734  0.35  0.41
## 10  SOUTHWEST  6678 11907  0.58  1.06
## 11   FRONTIER   540   688  0.45  0.63
## 12     SPIRIT  1502  1418  0.88  0.93
```

This is a table small enough that you can view everything on a single slide.

Selecting a specific string (1/2)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline = 'UNITED'
```

Selecting a specific string is pretty easy. Just be sure to use quote marks. You can use single quotes or double quotes, but you can not use the so-called “Smart” quotes that slant differently at the beginning than at the end.

Selecting a specific string (2/2)

```
##   Airline r2016 r2017
## 1  UNITED  0.45   0.3
```

Here's what the results looks like.

Selecting a range of strings (1/2)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline between 'E' and 'J'
```

You can use the BETWEEN keyword for character strings. Note that ‘J’ excluded ‘JETBLUE’ which comes after ‘J’ in alphabetical order.

Selecting a range of strings (2/2)

```
##      Airline r2016 r2017
## 1   HAWAIIAN  0.04  0.11
## 2 EXPRESSJET  1.58  0.67
## 3   FRONTIER  0.63  0.45
```

Here's what the results looks like.

Sorting the results (1/2)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline between 'E' and 'J'
  order by Airline
```

If you'd like the records listed in order, use the ORDER BY statement.

Sorting the results (2/2)

```
##      Airline r2016 r2017
## 1 EXPRESSJET  1.58  0.67
## 2   FRONTIER  0.63  0.45
## 3   HAWAIIAN  0.04  0.11
```

Here are the results.

Sorting the results in reverse order (1/2)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  order by r2017 desc
```

If you'd like the records listed in order, add the DESC keyword. This particular code arranges the data so that the airline with the highest bump rate in 2017 appears at the top.

Sorting the results in reverse order (2/2)

```
##       Airline r2016 r2017
## 1      SPIRIT  0.93  0.88
## 2  EXPRESSJET  1.58  0.67
## 3   SOUTHWEST  1.06  0.58
## 4     JETBLUE  0.82  0.54
## 5    AMERICAN  0.66  0.46
## 6    FRONTIER  0.63  0.45
## 7     SKYWEST  0.96  0.37
## 8      ALASKA  0.41  0.35
## 9      UNITED  0.45  0.30
## 10     VIRGIN  0.13  0.27
## 11   HAWAIIAN  0.04  0.11
## 12      DELTA  0.09  0.07
```

Here are the results.

Sorting on multiple fields

– SQL code

```
order by sex, age
```

You can sort by mutliple values. The data is sorted by the first field, and among those values that are tied on the first field, you break those ties using the values in the second field. So the example shown above places the females first and then the males. Among the females the data is sorted from young to old and among the males the data is sorted from young to old.

You could reverse this and you'd get a different pattern. Everyone is sorted by age with the youngest at the beginning. If two or more records are tied on age, then the females are listed first followed by the males.

There may be settings where you might extend this to a third or fourth field, and it works pretty much the same way. The third field only becomes important for those records which are tied on both the first and second fields.

Warning

– What is between may not always be intuitive.
  - Is “w” between “U” and “Z”?
  - Where do numbers fit in?
  - How do special codes fit in (e.g, tabs versus spaces)?
  - How do accented characters fit in?

Ranges can be tricky. If you use this approach, run a few tests first so you know what you are getting. Do lower case letters fit in right next to their upper case counterparts? Where do the numbers fit in? There are certain things like tabs that can foul you up because they look just like spaces, but they have a different location in the order of things. If you have accented characters, which can occur easily for foreign names, make sure you know where they fit in as well. This is something that might depend on what country your computer resides in. European computers might produce different results than American computers for some of these queries.

Selecting a random sample (1/2)

– SQL code (Oracle) NOTE: an approximate sample proportion

```
SELECT select Airline, r2016, r2017
  from airlines_table
  sample(25)
```

– SQL code (SQLite)

```
select Airline, r2016, r2017
  from airlines_table
  order by random()
  limit 4
```

– SQL code (others)

```
???
```

Every database has a different way to select a random sample. There are also multiple ways to select random samples.

The Oracle approach gives a sample that is approximately 25%. What Oracle does is to generate for every record a random number between 0 and 1. If that random number is less than or equal to 0.25, Oracle includes the record and if the random number is greater than 0.25, Oracle excludes that record. This means that in a database of 100 records, the SAMPLE(25) could easily produce a sample as small as 19 records or as large as 31 records.

It turns out that trying to get exactly 25% of the records takes a bit more work, and for very large databases, the extra work could mean the difference between waiting a few seconds and waiting a few hours.

The SQLITE approach gives an exact number of records, but it requires a sort of the entire database.

Selecting a random sample (2/2)

```
##    Airline r2016 r2017
## 1 FRONTIER  0.63  0.45
## 2    DELTA  0.09  0.07
## 3  SKYWEST  0.96  0.37
## 4 AMERICAN  0.66  0.46
```

Here's what the results looks like.

Warnings about sampling

- No data on performance of the sampling
  - Potential for bias
  - Potential for lack of security
- Okay for informal uses
- Consider sampling within R or SAS

You should be cautious about the sampling algorithms used in these database packages. As far as I know, no one has done formal testing to see if these systems perform well. They are probably okay for informal applications. Where there may be concerns include settings where a biased sample might compromise the research, such as in a prospective clinical trial. It might also be a concern if there is a need to keep the actual random values securely hidden. Many random number generators allow you to easily predict what the next random number would be even after just reviewing a handful of the earlier random numbers.

I suspect that many of the intended uses of sampling in databases are fairly informal. In these settings, it's not worth worrying about. But if there is need for high quality and security, you may be better off doing the sampling within a statistical pacakge like R or SAS.

Selecting based on part of a string (1/6)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline like 'JET%'
```

If you are searching for just part of a string, use the keyword LIKE and include a percent sign for the portion of the string that you can leave open ended. The code shown above looks for an airline that has the letters J-E-T followed by anything (or nothing).

Selecting based on part of a string (2/6)

```
##   Airline r2016 r2017
## 1 JETBLUE  0.82  0.54
```

Here's what the results looks like.

Selecting based on part of a string (3/6)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline like '%JET'
```

The code here looks similar, but %JET means that you can start with anything, but you have to end with J-E-T

Selecting based on part of a string (4/6)

```
##      Airline r2016 r2017
## 1 EXPRESSJET  1.58  0.67
```

Here's what the results looks like.

Selecting based on part of a string (5/6)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline like '%A%'
```

Here's the code that looks for the single letter A either in the middle or in the beginning or the end.

Selecting based on part of a string (6/6)

```
##    Airline r2016 r2017
## 1    DELTA  0.09  0.07
## 2 HAWAIIAN  0.04  0.11
## 3 AMERICAN  0.66  0.46
## 4   ALASKA  0.41  0.35
```

Here's what the results looks like.

Selecting using NOT (1/2)

– SQL code

```
select Airline, r2016, r2017
  from airlines_table
  where Airline not like '%A%'
```

The NOT keyword is useful for finding things that fail a particular match.

Selecting using NOT (2/2)

```
##      Airline r2016 r2017
## 1     VIRGIN  0.13  0.27
## 2    JETBLUE  0.82  0.54
## 3     UNITED  0.45  0.30
## 4 EXPRESSJET  1.58  0.67
## 5    SKYWEST  0.96  0.37
## 6  SOUTHWEST  1.06  0.58
## 7   FRONTIER  0.63  0.45
## 8     SPIRIT  0.93  0.88
```

Here's what the results looks like.

Your homework

- The cigarettes database
  - Single table, cigarettes
  - Tar, nicotine, and carbon monoxide levels (mg)
  - Weight of cigarette in grams
- You can find a description of this data set at
  - http://jse.amstat.org/datasets/cigarettes.txt
- Find the records for cigarettes that have an ampersand (&) in their name
- Find the records for cigarettes that are a single word brand name (no spaces)
- Put your code and output in a single pdf file

You will have a different data set for your homework. This data set measures all the bad things that you can find in a cigarette: tar, nicotine, and carbon monoxide. Find and print the records of any cigarettes whose brand name includes an ampersand. Also, find and print cigarettes whose brand names include just a single word. You'll need the NOT keyword here. Put all your results in a single PDF file.