Database Practise for Data Science

Sebastian Maneth
University of Edinburgh

October 6, 2015

Agenda

1. Data Conversion
2. SQL Queries
3. Display Results

Relational Databases

Why?

→ fast access to data
→ majority of data stored in relational DBs

Relational DBs

→ very mature pieces of software
[IBM, Oracle, Microsoft] [PostgreSQL, MySQL, SQLite, etc]

→ SQL *standard query language*

→ query optimization

1. Data Conversion

→ data typically has to be converted from one format to another before it can be loaded into a Database

→ often you have to write your own code to perform this task

1. Data Conversion

→ data typically has to be converted from one format to another
before it can be loaded into a Database

→ often you have to write your own code to perform this task

Example Data

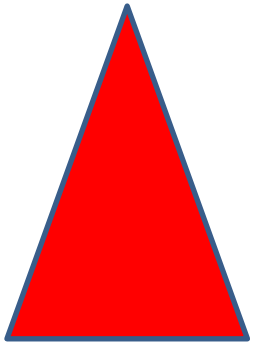
dblp.xml -- Bibliography data (computer science)

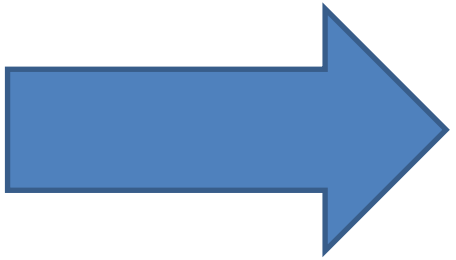
Size: 1.6GB (from dblp.uni-trier.de on 01.08.2015)

→ 3.3m entries
→ 1.6m authors

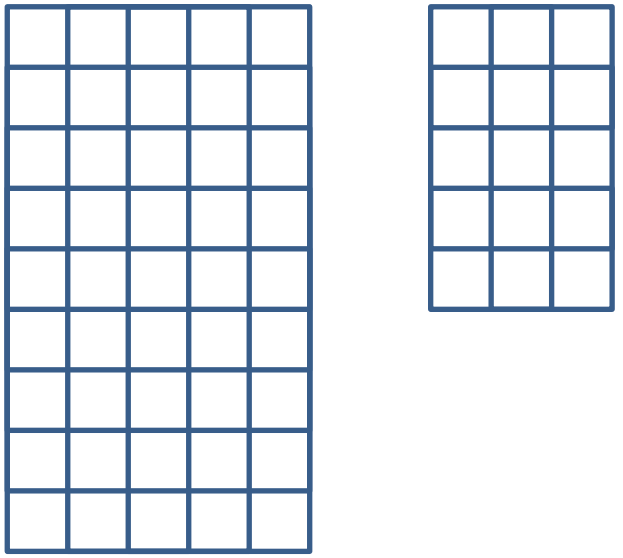
Data Conversion Example

XML





.csv files

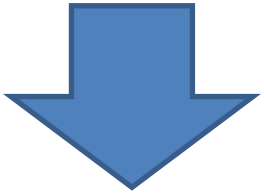


XML

- → eXtensible Markup Language (W3C Standard)
- → like HTML, but lets you define your own tags
- → lingua franca for data on the web
- → uses Unicode

```
<article mdate="2011-01-11" key="journals/acta/Milner96">
<author>Robin Milner</author>
<title>Calculi for Interaction.</title>
<year>1996</year>
<pages>707-737</pages>
<volume>33</volume>
<journal>Acta Inf.</journal>
<number>8</number>
<url>db/journals/acta/acta33.html#Milner96</url>
<ee>http://dx.doi.org/10.1007/BF03036472</ee>
</article>
```

```
<article mdate="2011-01-11" key="journals/acta/Milner96">
<author>Robin Milner</author>
<title>Calculi for Interaction.</title>
<year>1996</year>
.
.
</article>
```



AID	NAME
.	
.	
7	Robin Milner
8	
.	
.	

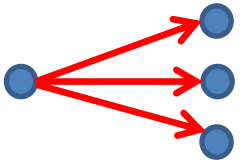
author.csv

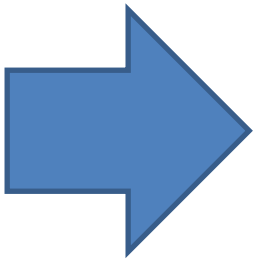
PID	NAME
.	
.	
13	Calculi for Interaction
14	
.	
.	

paper.csv

PID	AID
.	
.	
13	7
8	
.	
.	

writtenBy.csv





www.google.de/search

Google

DBLP to csv converter python

Web Videos Bilder Shopping Maps Mehr Suchoptionen

Ungefähr 340.000 Ergebnisse (0,57 Sekunden)

Ajeo/dblp-to-csv · GitHub
https://github.com/Ajeo/**dblp-to-csv** ▾ Diese Seite übersetzen
01.11.2013 - **dblp-to-csv** - **Python** parser for **dblp** xml to **csv**, with all articles titles and years since 1936.

dblp-to-csv - Python parser for dblp xml to csv, with all ...
java-hackers.com/p/Ajeo/**dblp-to-csv**/watchers ▾ Diese Seite übersetzen
Python parser for **dblp** xml to **csv**, with all articles titles and years since 1936.

xmlutils 1.1 : Python Package Index
https://pypi.**python**.org/pypi/xmlutils ▾ Diese Seite übersetzen
A set of utilities for processing XML documents and **converting** to other formats. ... xml files serially, namely **converting** them to other formats (SQL, **CSV**, JSON).

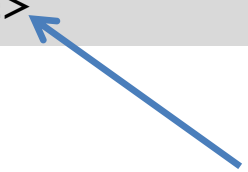
```
import xml.sax
from unidecode import unidecode

def endElement(self, tag):
    if tag == "title":
        self.article['title'] = self.title
    if tag == "year":
        self.article['year'] = self.year
    if len(self.article['title']) > 0 and len(self.article['year']) > 0:
        data = unidecode(self.article['year']+','+self.article['title']+'\n')
            self.file.write(data)
            self.article['title'] = ""
            self.article['year'] = ""
        elif self.CurrentData == "dblp":
            self.file.close()
            sys.exit("stop")

def characters(self, content):
    if self.CurrentData == "title":
        self.title = content.strip().rstrip('\n').replace('"','')
    elif self.CurrentData == "year":
        self.year = content.strip().rstrip('\n')
```

Python code, using SAX-Parser for XML

Issues with this Code:

→ Writes out title/year content whenever another closing tag is encountered

```
<article>
<title>Calculi for Interaction.</title>
<year></year>
</article>

<article>
<title></title>
<year>2014</year>
</article>
```



Issues with this Code:

→ converts Unicode characters to ASCI

```
<article>
<author>Muller</author>
</article>

<article>
<author>Müller</author>
</article>
```

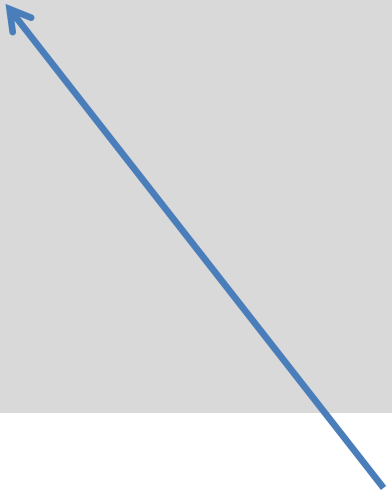
```
Muller
```

→ sqlite3 does support Unicode

Issues with this Code:

→ “characters” is called several times for one text content (by the parser)

```
<article>
<title>Calculi for Interaction.</title>
<year></year>
</article>

<article>
<title></title>
<year>2014</year>
</article>
```



“Cal”
“c”
“uli for In”
“teraction”
“.”

1. Data Conversion

LESSON 1

→ **always check output manually!**
(before loading it into DB)

→ **understand the logic of the converter code!**

Example Solution

→ open three files: **`author.csv`**, **`paper.csv`**, **`writtenBy.csv`**

→ hash author-names by their AID

→ if new author encountered, hash it & write it to **`author.csv`**

→ collect authors of one item

→ *at end of an item*, write title/year to **`paper.csv`**
write (PID, AID) to **`writtenBy.csv`** (for all authors of item)

```
1;Parallel Integer Sorting and Simulation Amongst CRCW Models.;1996
2;Pattern Matching in Trees and Nets.;1983
3;NP-complete Problems Simplified on Tree Schemas.;1983
4;On the Power of Chain Rules in Context Free Grammars.;1982
5;Schnelle Multiplikation von Polynomenueber Korpern der Charakteristik 2.;1977
6;A characterization of rational DOL power series.;2011
7;The Derivation of Systolic Implementations of Programs.;1987
8;Fifo Nets Without Order Deadlock.;1988
9;On the Complementation Rule for Multivalued Dependencies in Database Relations.
10;Equational weighted tree transformations.;2012
11;Merged processes: a new condensed representation of Petri net behaviour.;2006
12;Verifying a simplification of mutual exclusion by Lycklama-Hadzilacos.;2013
13;A Three-Stage Construction for Multiconnection Networks.;1983
14;The Expressive Power of Delay Operators in SCCS.;1991
15;Calculi for Interaction.;1996
.
```

Finally …

```
1;Sanjeev Saxena
2;Hans-Ulrich Simon
3;Nathan Goodman
4;Oded Shmueli
5;Norbert Blum
6;Arnold Schonhage
7;Juha Honkala
8;Chua-Huang Huang
9;Christian Lengauer
10;Alain Finkel
11;Annie Choquet
12;Joachim Biskup
13;Symeon Bozapalidis
14;Zoltan Fulop 0001
15;George Rahonis
.
```

```
1;1
2;2
3;3
3;4
4;5
5;6
6;7
7;8
7;9
8;10
8;11
9;12
10;13
10;14
10;15
```

2. SQL

→ fire up your favourite Relational Database
→ here: **sqlite3**

→ we will use only very simple SQL to express interesing queries

Step 1: create tables & indexes

aka “attribute”

```
create table table_name (column_name type [primary key], ... )
```

```
int,char,float,real,date,time,
numeric(precision, scale), …
```

```
create index index_name on table_name (column_name, ...);
```

Create Tables & Indexes

creates indexes

```
$ sqlite3 test.sq3
SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
sqlite> create table author (aid int primary key,
                             name text);
sqlite> create table paper (pid int primary key,
                            title text,
                            year int);
sqlite> create table writtenBy (pid int,
                                aid int);
sqlite> create index ai on author (name);
sqlite> create index wi on writtenBy (pid, aid);


sqlite> .separator ";"
sqlite> .import author.csv author
sqlite> .import paper.csv paper
sqlite> .import writtenBy.csv writtenBy
sqlite> .timer on
```

sqlite3 specific syntax

2. SQL Queries

```
select    list, of, attributes
from      list of tables
where     conditions
group by  list of attributes
order by  attribute asc|desc
```

← optionally with aggregates

Aggregates: `count, sum, avg, min, max`

Conditions: `and, or, not, in, <, =, >, like`

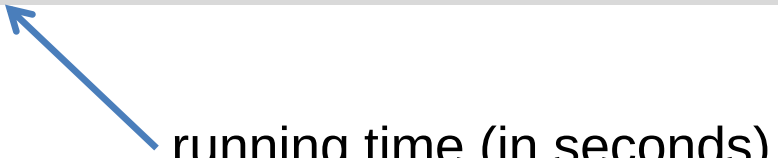
Combine tables (using set-semantics): `union, intersect, except`

→ query returns a table

→ where a table is allowed,
you can place a *nested* query: `(select * from ... )`

Text Queries

```
sqlite> select *
sqlite> from author
sqlite> where name="Robin Milner";
24;Robin Milner
Run Time: real 1.174 user 0.672000 sys 0.500000
```



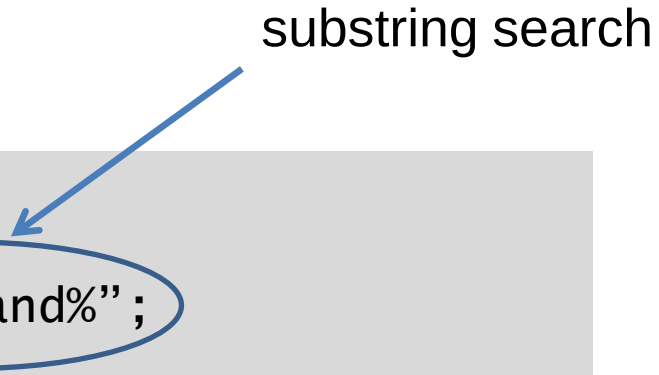
running time (in seconds)

Text Queries

```
sqlite> select *
sqlite> from author
sqlite> where name="Robin Milner";
24;Robin Milner
Run Time: real 1.174 user 0.672000 sys 0.500000
```


```
sqlite> select *
sqlite> from author
sqlite> where name="Robin Milner";
24;Robin Milner
Run Time: real 0.002 user 0.000000 sys 0.000000
```

Text Queries




```
sqlite> select count(*)
sqlite> from paper
sqlite> where title like “%and%”;
832133
Run Time: real 6.128 user 4.172000 sys 1.875000
```

Text Queries

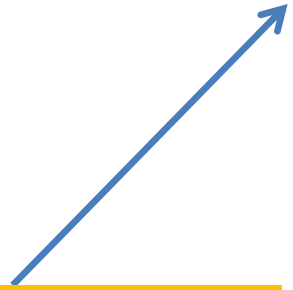
```
sqlite> select count(*)
sqlite> from author a
sqlite> where a.name like “%Milner%”;
45
Run Time: real 1.516 user 1.187000 sys 0.250000
```


```
sqlite> select count (*)
sqlite> from author a
sqlite> where a.name like “%Milner%”;
45
Run Time: real 1.525 user 1.203000 sys 0.313000
```

☹ → use specialized Text Index (e.g. Apache Lucene)

SQL: per author queries

```
-- title and year of papers by Robin Milner

select p.title, p.year
from   paper p, author a, writtenBy w
where  a.name = “Robin Milner”
and    a.aid = w.aid and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
An Interview with Robin Milner.;1993
.
.
.
Run Time: real 430.591 user 178.375000 sys 240.688000
```



without any index

SQL: per author queries

```
-- title and year of papers by Robin Milner

select p.title, p.year
from   paper p, author a, writtenBy w
where  a.name = “Robin Milner”
and    a.aid = w.aid and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
An Interview with Robin Milner.;1993
.
.
.
Run Time: real 430.591 user 178.375000 sys 240.688000
```

```
Run Time: real 18.378 user 13.562000 sys 3.485000
```

without any index

with indexes

SQL: per author queries

```
-- title and year of papers by Robin Milner

select p.title, p.year
from   paper p, author a, writtenBy w
where  a.name = “Robin Milner”
and    a.aid = w.aid and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
..
Run Time: real 18.378 user 13.562000 sys 3.485000
```

with indexes

```
select p.title, p.year
from   paper p, writtenBy w
where  w.aid=24 and w.pid = p.pid;
Run Time: real 3.579 user 2.484000 sys 1.047000
```

no join with author

SQL: per author queries

```
-- title and year of papers by Robin Milner

select p.title, p.year
from   paper p, author a, writtenBy w
where  a.name = "Robin Milner"
and    a.aid = w.aid and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
..
Run Time: real 18.378 user 13.562000 sys 3.485000
```

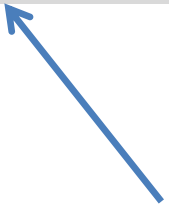
with indexes

```
select pid
from writtenBy
where aid=24;
Run Time: real 3.659 user 2.484000 sys 1.047000
```

Oops. – single scan –
Right index **must** be missing.

SQL: per author queries

```
create index wi on writtenBy (aid);

select p.title, p.year
from   paper p, writtenBy w
where  w.aid=24 and w.pid = p.pid;
Calculi for Interaction.;1996
Elements of Interaction - Turing Award Lecture.;1993
..
Run Time: real 0.015 user 0.000000 sys 0.016000
```



almost 30,000-times faster than without any index

2. SQL Queries

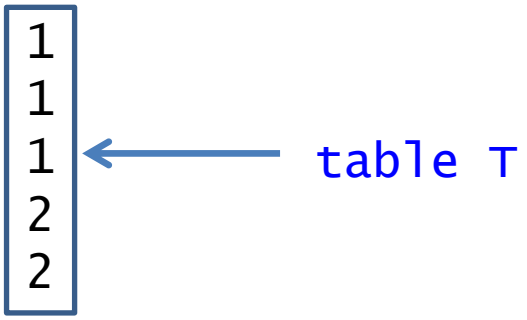
LESSON 2

→ **create indexes / primary keys!**

→ **avoid joins / use IDs!**

2. SQL Queries

How to use “group by”




```
select n,count(n) from T group by n;
```

1	3
2	2

→ histogram of T

→ without “group by”, semantics of this query is undefined
(sqlite3 goes wild, psql reports error)

SQL: per author queries

```
-- number of papers per year by aid=314 (Jeffrey D. Ullman)

select y.year,y.count
from (
      select p.year, count(p.year) as count
      from (paper p join writtenBy w on (p.pid = w.pid))
      where w.aid=314
      group by year
) y;
1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12
.
.
Run Time: real 0.020 user 0.000000 sys 0.031000
```

SQL: per author queries

```
-- number of papers per year by aid=314 (Jeffrey D. Ullman)
1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12
```

3. Display Results

```
-- number of papers per year by aid=314 (Jeffrey D. Ullman)
1966|1
1967|4
1968|8
1969|7
1970|6
1971|4
1972|12
```

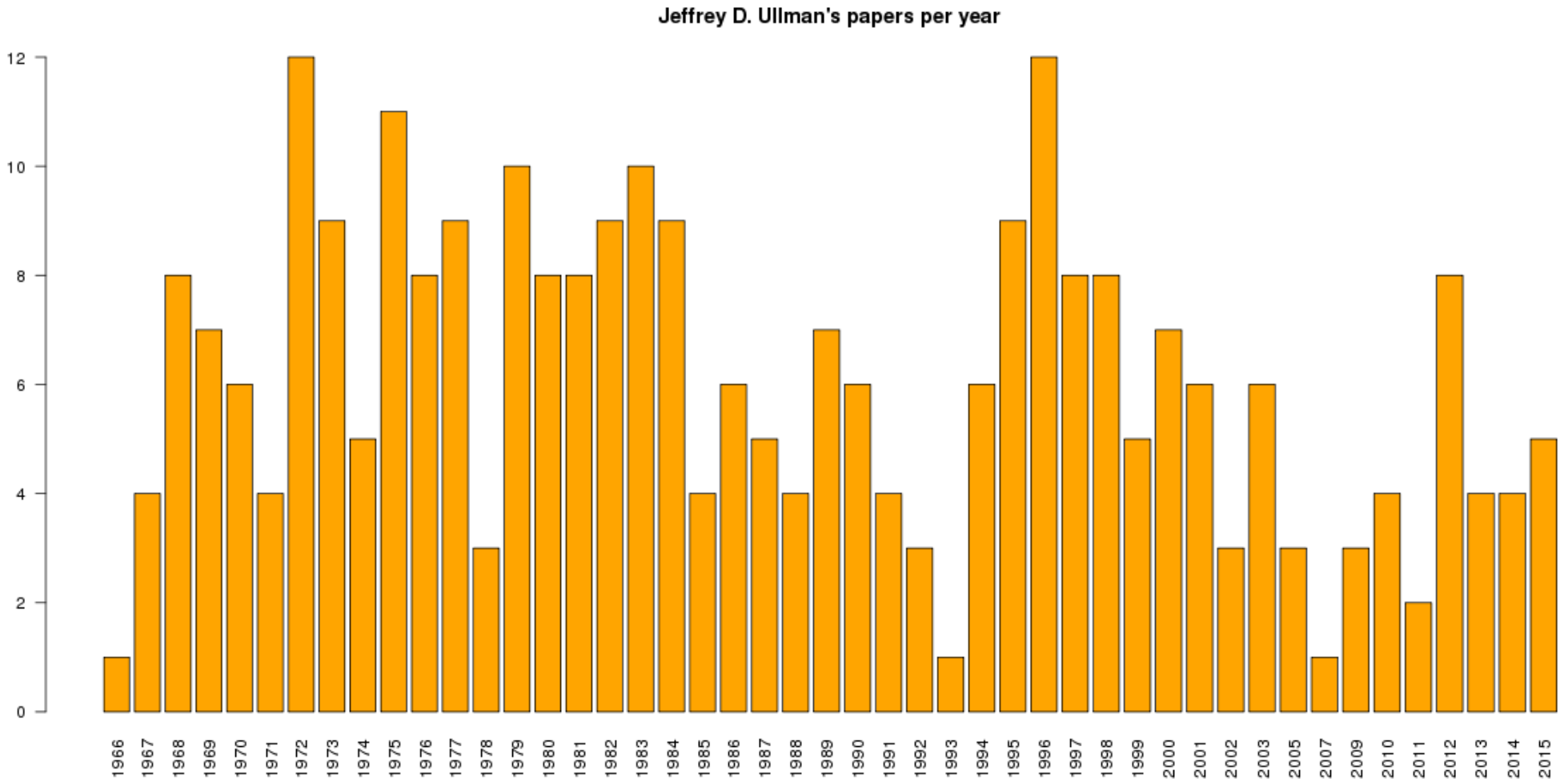
→ **plot** histogram (using R):

```
.output Ullman.csv
select y.year, y.count ....
.output stdout
```

sqlite

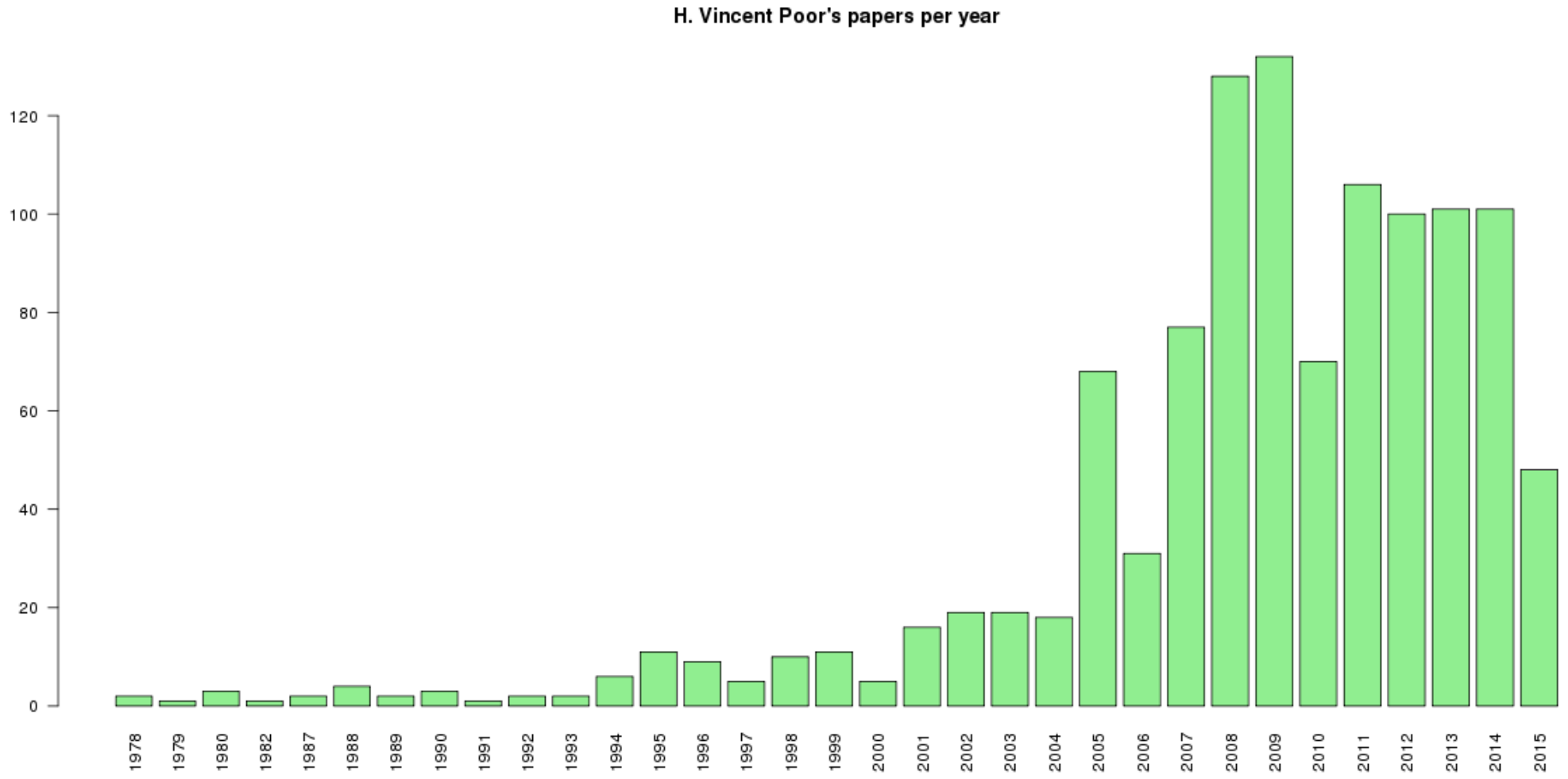
```
$ R
u <- read.csv( 'Ullman.csv', head=FALSE, sep="|" )
barplot( u[,2],u[,1],names.arg=u[,1],col="orange",
         main="Jeffrey Ullman's papers per year",las=2 )
```

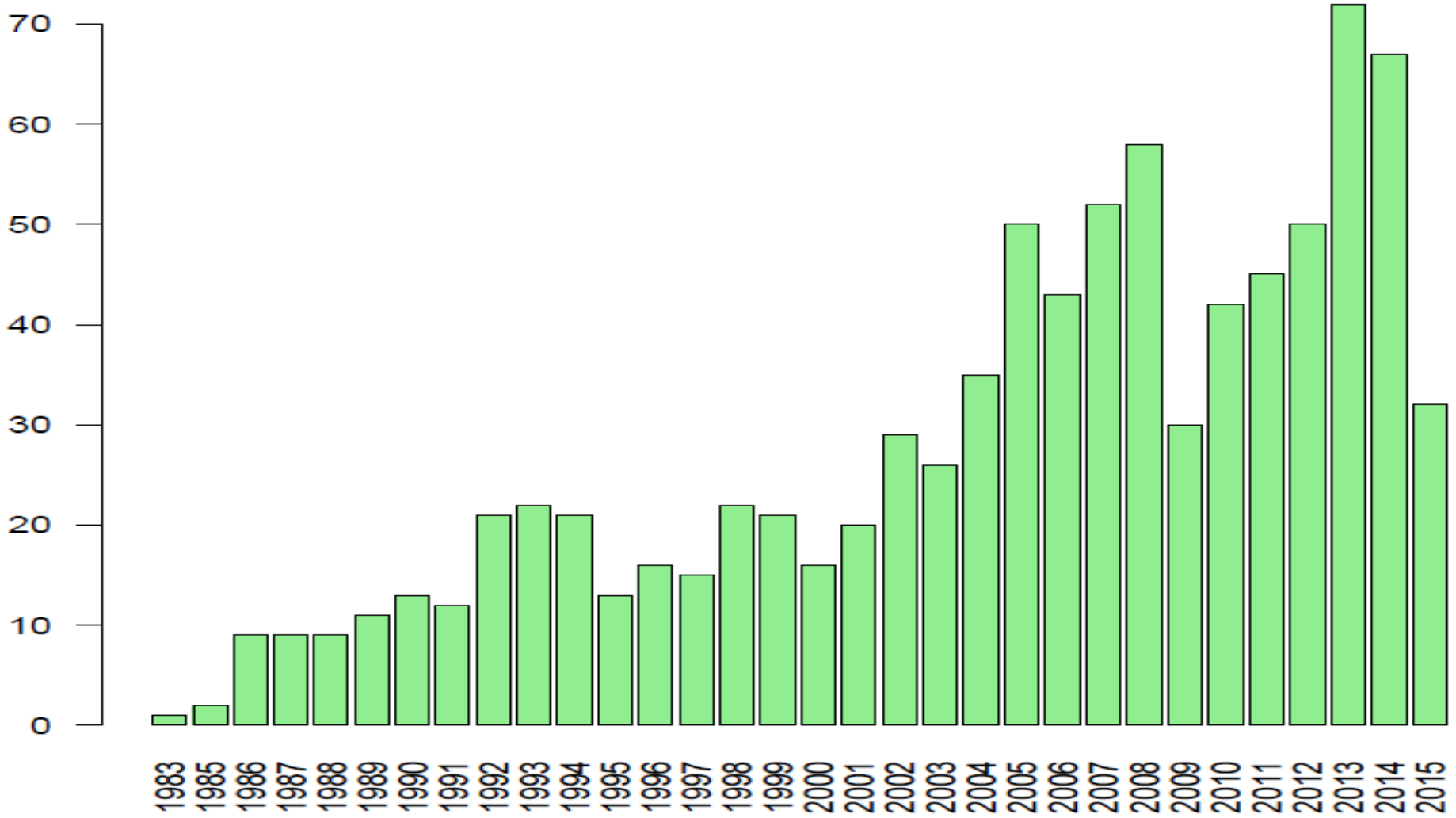
3. Display Results



3. Display Results

```
-- most prolific authors??

select   a.name, count(w.pid) as count
from     author a, writtenBy w
Where    a.aid = w.aid
group by w.aid
order by count desc limit 40;
H. Vincent Poor|1114
Wei Wang|1064
Yan Zhang|999
Wei Liu|981
Wen Gao|926
Philip S. Yu|885
Thomas S. Huang|838
Chin-Chen Chang|795
Lajos Hanzo|790
Elisa Bertino|782
Wei Zhang|779
...
Run Time: real 101.823 user 38.312000 sys 52.125000
```


H. Vincent Poor's papers per year
120
100
80
60
40
20
0
1978
1979
1980
1982
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

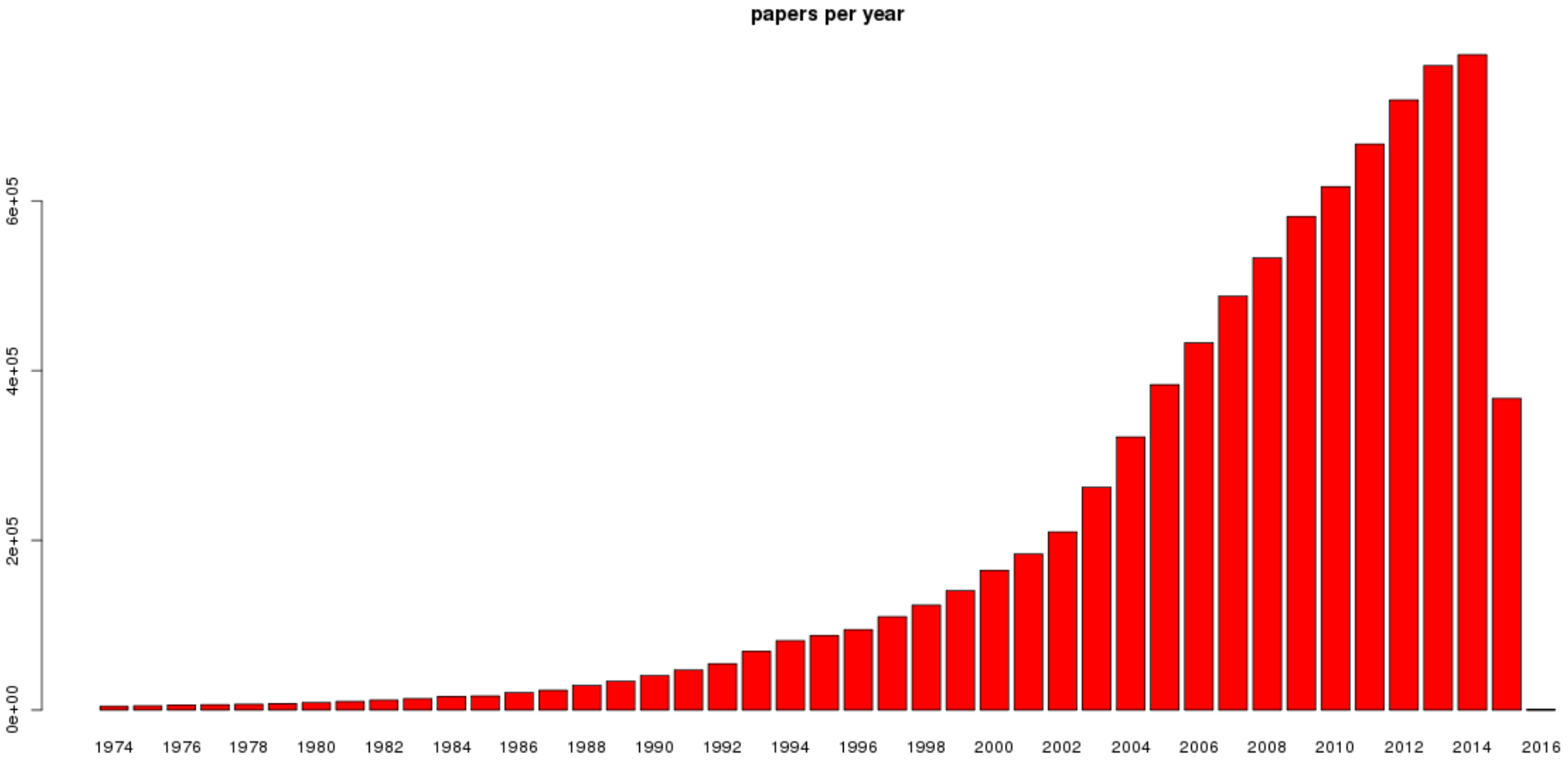

Philip S. Yu's papers per year
70
60
50
40
30
20
10
0
1983
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

Global queries

```
-- number of papers per year
select y.year, y.count
from (select p.year, count(p.year) as count
      from (paper p join writtenBy w on (p.pid = w.pid))
      group by year) y;
```

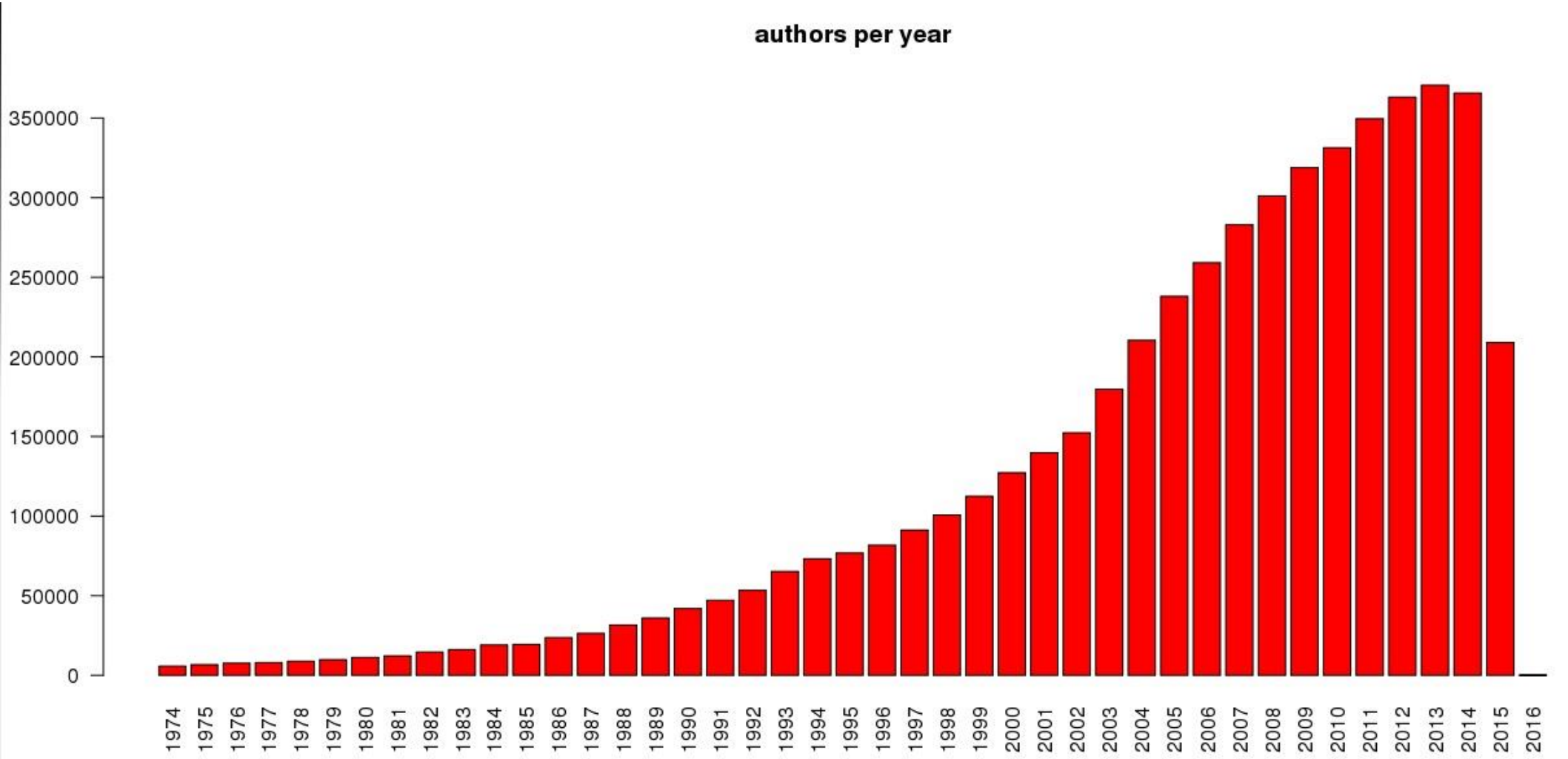
Global queries

```
-- number of papers per year
select y.year, y.count
from (select p.year, count(p.year) as count
      from (paper p join writtenBy w on (p.pid = w.pid))
      group by year) y;
```



SQL: global queries

```
-- number of authors per year
select y.year, y.count
from (select p.year, count(ww.aid) as count
      from (paper p join writtenBy w on (p.pid=w.pid)),writtenBy ww
      where ww.pid=w.pid group by year) y;
```

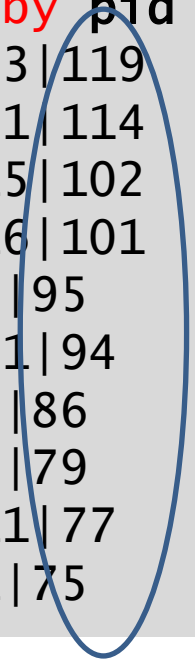


SQL: per paper queries

```
select avg(na) from
(select count(aid) as na from writtenBy group by pid);
2.8366622915623

select max(na) from
(select count(aid) as na from writtenBy group by pid);
119

select pid,count(aid) as na from writtenBy
group by pid order by na desc limit 10;
1277503|119
1496761|114
2283215|102
1077916|101
867649|95
2729731|94
587668|86
853657|79
2183911|77
679621|75
```

SQL: per paper queries

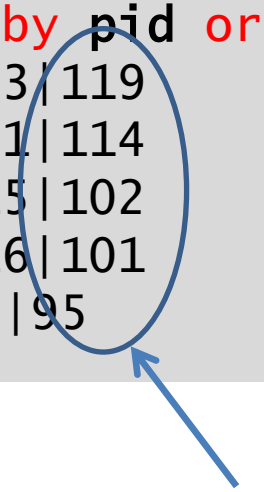
```
select avg(na) from
(select count(aid) as na from writtenBy group by pid);
2.8366622915623

select max(na) from
(select count(aid) as na from writtenBy group by pid);
119

select pid,count(aid) as na from writtenBy
group by pid order by na desc limit 10;
1277503|119
1496761|114
2283215|102
1077916|101
867649|95
2729731|94
587668|86
853657|79
2183911|77
679621|75
```



How to get a histogram of this?

SQL: per paper queries

```
select pid,count(aid) as na from writtenBy
group by pid order by na desc limit 10;
1277503|119
1496761|114
2283215|102
1077916|101
867649|95
```



CAnum

```
 1 |  554,173
 2 |  906,003
 3 |  763,152
 4 |  432,813
 5 |  193,211
 6 |   83,906
 7 |   35,814
 8 |   17,910
 9 |    9,534
10 |    5,665
```

```
select i,100.0*count(i)/
                (select count(*) from paper)
from CAnum group by i;
1;18.3 ┐
2;30.0 │
3;25.3 ├ 94.3% of all papers
4;14.3 │
5; 6.4 ┘
6; 2.7
7; 1.1
8; 0.5
9; 0.3
10; 0.1
```

Who has written the most solo-papers?

```
-- solo-papers

select y.pid from
(select pid,count(aid) as nu from writtenBy group by pid) y
where y.nu=1;
```

Who has written the most solo-papers?

```
create table SoloPapers (pid int);
create table solo (aid int, pid int);

insert into SoloPapers select y.pid from
(select pid,count(aid) as nu from writtenBy group by pid) y
where y.nu=1;

insert into solo select w.aid, s.pid from writtenBy w,
SoloPapers s where s.pid=w.pid;

select aid,count(pid) as nu from solo group by aid order by nu
desc limit 40;
19121;289     552121;166
132395;261    96052;161
192460;258    107288;150
367982;249    487337;148
94760;201     15807;139
584760;186    15644;138
773443;184    ...
419479;176
```

Who has written only solo-papers?

?

- End Lecture 1 -