R and SQL

Kylie Ariel Bemis

2/4/2019

R and databases

Often, relational data is stored in a database managed by an RDBMS, so we need to know how to work with data in a database.

A database can also be a good solution for data which you do not want to load all into memory at once.

R and `dplyr` can interface with databases through the `DBI` package and a suitable backend:

- `RSQLite` works with SQLite databases
- `RMySQL` works with MySQL databases
- `RPostgreSQL` works with PostgreSQL databases

etc.

We can work with these with `dplyr` using the `dbplyr` package.

R and databases with dbplyr

Suppose we wish to work with an SQLite database.

First we need to install the necessary packages.

```
install.packages(c("DBI", "RSQLite", "dbplyr"))
```

```
library(dbplyr)
```

```
##
## Attaching package: 'dbplyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     ident, sql
```

```
library(RSQLite)
```

Connecting to a database in R

The DBLP is a database containing bibliographic data on major computer science journals and proceedings. A subset of it has been loaded into the SQLite database file “dblp.db”.

First, we need to open a *connection* to the database using `dbConnect()`.

The first argument of `dbConnect()` is the backend to use (provided by the `RSQLite` package in this case), and the second is the filepath to the database.

```r
dbpath <- paste0("~/Documents/Northeastern/",
                 "Courses/DS5110-Spring2019/data/",
                 "DBLP-CSR-sqlite/dblp.db")
con <- dbConnect(SQLite(), dbname=dbpath)
```

Creating a `tbl` of the connected database

A `tbl` is the tidyverse's generalized notion of tabular data.

A tibble is a type of `tbl`. We can also create a `tbl` from a database.

```
dblp_main <- tbl(con, "general")
dblp_authors <- tbl(con, "authors")
```

The first argument is the data source (our database connection).

The second argument is the name of the table within the database.

Our SQLite database contains the table "general", which has information about papers published in computer science journals and proceedings.

```
dblp_main
```

```
## # Source:   table<general> [?? x 10]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##    k         year conf  crossref     cs    de    se    th publi
##    <chr>    <int> <chr> <chr>     <int> <int> <int> <int> <chr>
##  1 conf/a~  2013 AAAI  conf/aa~      1     0     0     0 AAAI
##  2 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  3 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
##  4 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  5 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  6 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
##  7 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  8 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  9 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
## 10 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
## # ... with more rows
```

```
dblp_authors
```

```
## # Source:   table<authors> [?? x 6]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##       id k                          pos name               gende
##    <int> <chr>                    <int> <chr>              <chr>
##  1     1 conf/aaai/0001M13            0 Chang Wang 0001    M
##  2     2 conf/aaai/0001M13            1 Sridhar Mahadevan  M
##  3     3 conf/aaai/0001T15            0 Claudia Schulz 0001 F
##  4     4 conf/aaai/0001T15            1 Francesca Toni     F
##  5     5 conf/aaai/0001TZLL14         0 Jing Zhang 0001    F
##  6     6 conf/aaai/0001TZLL14         1 Jie Tang           F
##  7     7 conf/aaai/0001TZLL14         2 Honglei Zhuang     -
##  8     8 conf/aaai/0001TZLL14         3 Cane Wing-ki Leung F
##  9     9 conf/aaai/0001TZLL14         4 Juan-Zi Li         -
## 10    10 conf/aaai/0001VD15           0 Bart Bogaerts 0001 M
## # ... with more rows
```

We can perform `dplyr` operations on the database using `dplyr` verbs.

```r
dblp_main %>% summarise(cs_papers=sum(cs))
```

```
## Warning: Missing values are always removed in SQL.
## Use `SUM(x, na.rm = TRUE)` to silence this warning
```

```
## # Source:   lazy query [?? x 1]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##   cs_papers
##       <int>
## 1     96823
```

```r
dblp_main %>%
  filter(year > 2010) %>%
  group_by(year) %>%
  summarise(cs_papers=sum(cs))
```

```
## Warning: Missing values are always removed in SQL.
## Use `SUM(x, na.rm = TRUE)` to silence this warning
```

```
## # Source:   lazy query [?? x 2]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##    year cs_papers
##   <int>     <int>
## 1  2011      6060
## 2  2012      5772
## 3  2013      7391
## 4  2014      6538
## 5  2015      5910
```

When working with databases, `dbplyr` tries to offload as much work as possible to the database itself, and delay execution until we need the result.

We can see the actual SQL commands generated using `show_query()`

```
query <- dblp_main %>%
  filter(year > 2010) %>%
  group_by(year) %>%
  summarise(cs_papers=sum(cs))
show_query(query)
```

```
## Warning: Missing values are always removed in SQL.
## Use `SUM(x, na.rm = TRUE)` to silence this warning
```

```
## <SQL>
## SELECT `year`, SUM(`cs`) AS `cs_papers`
## FROM `general`
## WHERE (`year` > 2010.0)
## GROUP BY `year`
```

Collecting a query into R

The query is not executed until the result is needed. You can use `collect()` to execute the query and pull the result into R.

```
query %>% collect()
```

```
## Warning: Missing values are always removed in SQL.
## Use `SUM(x, na.rm = TRUE)` to silence this warning
```

```
## # A tibble: 5 x 2
##    year cs_papers
##   <int>     <int>
## 1  2011      6060
## 2  2012      5772
## 3  2013      7391
## 4  2014      6538
## 5  2015      5910
```

Pulling database data into R

Not all databases support all `dplyr` data manipulation verbs, so if necessary (and the data is small enough), we can always pull the data into R as a tibble and work that way.

```r
dblp_main %>% collect()
```

```
## # A tibble: 148,521 x 10
##    k         year conf  crossref     cs    de    se    th publi
##    <chr>    <int> <chr> <chr>     <int> <int> <int> <int> <chr>
##  1 conf/a~  2013 AAAI  conf/aa~      1     0     0     0 AAAI
##  2 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  3 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
##  4 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  5 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  6 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
##  7 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  8 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
##  9 conf/a~  2014 AAAI  conf/aa~      1     0     0     0 AAAI
## 10 conf/a~  2015 AAAI  conf/aa~      1     0     0     0 AAAI
## # ... with 148,511 more rows
```

SQL equivalents to `dplyr` functions

Most functions in `dplyr` are simple and flexible versions of statements in SQL.

- `select()` and SELECT are used to select columns from a table
- `filter()` and WHERE are used to subset a table by on rows based on given conditions
- `arrange()` and ORDER BY are used to order the rows of a table
- `mutate()` does not have a direct equivalent, but AS performs a similar function of returning columns (potentially transformed) with a new name
- `summarise()` does not have a direct equivalent, but SQL provides several summary functions such as SUM, AVG, COUNT, etc.

Let's see some ways to perform the same operations using `dplyr` and SQL.

select() and SELECT

```r
dblp_authors %>% select(k, name, gender)
```

```r
query <- con %>%
  dbSendQuery("SELECT k, name, gender
               FROM authors")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##    k                    name                gender
##    <chr>                <chr>               <chr>
##  1 conf/aaai/0001M13    Chang Wang 0001     M
##  2 conf/aaai/0001M13    Sridhar Mahadevan   M
##  3 conf/aaai/0001T15    Claudia Schulz 0001 F
##  4 conf/aaai/0001T15    Francesca Toni      F
##  5 conf/aaai/0001TZLL14 Jing Zhang 0001     F
##  6 conf/aaai/0001TZLL14 Jie Tang            F
##  7 conf/aaai/0001TZLL14 Honglei Zhuang      -
##  8 conf/aaai/0001TZLL14 Cane Wing-ki Leung  F
##  9 conf/aaai/0001TZLL14 Juan-Zi Li          -
## 10 conf/aaai/0001VD15   Bart Bogaerts 0001  M
## # ... with more rows
```

filter() and WHERE

```r
dblp_authors %>% filter(gender == "F")
```

```r
query <- con %>%
  dbSendQuery("SELECT *
              FROM authors
              WHERE gender = 'F'")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:   lazy query [?? x 6]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##       id k                               pos name               gend
##    <int> <chr>                         <int> <chr>              <chr
##  1     3 conf/aaai/0001T15                 0 Claudia Schulz 0001 F
##  2     4 conf/aaai/0001T15                 1 Francesca Toni     F
##  3     5 conf/aaai/0001TZLL14              0 Jing Zhang 0001    F
##  4     6 conf/aaai/0001TZLL14              1 Jie Tang           F
##  5     8 conf/aaai/0001TZLL14              3 Cane Wing-ki Leung F
##  6    17 conf/aaai/0002GYSZL14             1 Hua Guo            F
##  7    18 conf/aaai/0002GYSZL14             2 Yi Yang            F
##  8    27 conf/aaai/0003MGF14               0 Wei Li 0002        F
##  9    29 conf/aaai/0003MGF14               2 Tovi Grossman      F
## 10    33 conf/aaai/0005YJZ15               2 Rong Jin           F
## # ... with more rows
```

filter() and WHERE (cont'd)

```
dblp_authors %>% filter(gender == "F" & prob > 0.95)
```

```
query <- con %>%
  dbSendQuery("SELECT *
              FROM authors
              WHERE gender = 'F' AND prob > 0.95")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:   lazy query [?? x 6]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##       id k                              pos name             g
##    <int> <chr>                        <int> <chr>            <
##  1     3 conf/aaai/0001T15                0 Claudia Schulz 0001 F
##  2     4 conf/aaai/0001T15                1 Francesca Toni   F
##  3    29 conf/aaai/0003MGF14              2 Tovi Grossman    F
##  4    39 conf/aaai/Abbott88               0 Kathy H. Abbott  F
##  5    52 conf/aaai/AbellaK93              0 Alicia Abella    F
##  6    54 conf/aaai/AbergALS06             0 Cécile Aberg     F
##  7    58 conf/aaai/AbernethySB08          0 Jennifer Abernethy F
##  8    60 conf/aaai/AbernethySB08          2 Elizabeth Bradley F
##  9    66 conf/aaai/AbtahiF11              0 Farnaz Abtahi    F
## 10    70 conf/aaai/Abu-HakimaHP91         2 Sieu Phan        F
## # ... with more rows
```

arrange() and ORDER BY

```
dblp_authors %>% arrange(name)
```

```
query <- con %>%
  dbSendQuery("SELECT *
              FROM authors
              ORDER BY name")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:     table<authors> [?? x 6]
## # Database:   sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
## # Ordered by: name
##        id k                              pos name          gender
##     <int> <chr>                        <int> <chr>         <chr>
##  1 134173 conf/dolap/MangisengiT98         1 A Min Tjoa    -
##  2 134238 conf/dolap/NguyenST05            2 A Min Tjoa    -
##  3 140374 conf/er/EderK86                  2 A Min Tjoa    -
##  4 142860 conf/er/PernulT91                1 A Min Tjoa    -
##  5 142863 conf/er/PernulWT93               2 A Min Tjoa    -
##  6 143978 conf/er/TjoaB93                  0 A Min Tjoa    -
##  7 143980 conf/er/TjoaW79                  0 A Min Tjoa    -
##  8 181589 conf/icde/EsmayrKPPT96           4 A Min Tjoa    -
##  9 187808 conf/icde/SchreflTW84            1 A Min Tjoa    -
## 10   9495 conf/aaai/KuanPH86               2 A-Chuan Hsueh -
## # ... with more rows
```

mutate() and AS

```
dblp_authors %>%
  mutate(pos1 = pos + 1) %>%
  select(k, pos1, name)
```

```
query <- con %>%
  dbSendQuery("SELECT k, pos + 1 AS pos1, name
               FROM authors")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##    k                         pos1 name
##    <chr>                    <dbl> <chr>
##  1 conf/aaai/0001M13            1 Chang Wang 0001
##  2 conf/aaai/0001M13            2 Sridhar Mahadevan
##  3 conf/aaai/0001T15            1 Claudia Schulz 0001
##  4 conf/aaai/0001T15            2 Francesca Toni
##  5 conf/aaai/0001TZLL14         1 Jing Zhang 0001
##  6 conf/aaai/0001TZLL14         2 Jie Tang
##  7 conf/aaai/0001TZLL14         3 Honglei Zhuang
##  8 conf/aaai/0001TZLL14         4 Cane Wing-ki Leung
##  9 conf/aaai/0001TZLL14         5 Juan-Zi Li
## 10 conf/aaai/0001VD15           1 Bart Bogaerts 0001
## # ... with more rows
```

summarise() and COUNT, AVG, SUM, etc.

```
dblp_authors %>% summarize(mean(prob))
```

```
query <- con %>%
  dbSendQuery("SELECT AVG(prob)
              FROM authors")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## Warning: Missing values are always removed in SQL.
## Use `AVG(x, na.rm = TRUE)` to silence this warning

## # Source:   lazy query [?? x 1]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##   `mean(prob)`
##          <dbl>
## 1         1.12
```

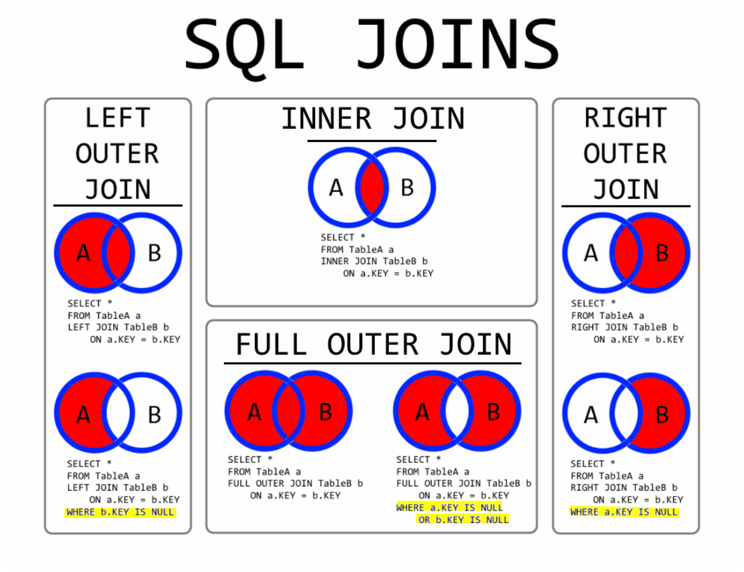
group_by() and COUNT, AVG, SUM, etc.

```r
dblp_authors %>%
  group_by(gender) %>%
  summarize(n())
```

```r
query <- con %>%
  dbSendQuery("SELECT gender, COUNT() AS n
              FROM authors
              GROUP BY gender")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```
## # Source:   lazy query [?? x 2]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##   gender  `n()`
##   <chr>   <int>
## 1 -       59009
## 2 F       69500
## 3 M      287936
```

More joins in SQL and R



```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM TableA a
INNER JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM TableA a
RIGHT JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
WHERE b.KEY IS NULL
```

```
SELECT *
FROM TableA a
FULL OUTER JOIN TableB b
    ON a.KEY = b.KEY
```

```
SELECT *
FROM TableA a
FULL OUTER JOIN TableB b
    ON a.KEY = b.KEY
WHERE a.KEY IS NULL
    OR b.KEY IS NULL
```

```
SELECT *
FROM TableA a
RIGHT JOIN TableB b
    ON a.KEY = b.KEY
WHERE a.KEY IS NULL
```

Figure 1:

```r
x <- tribble(
  ~key, ~val_x,
     1, "x1",
     2, "x2",
     3, "x3"
)
y <- tribble(
  ~key, ~val_y,
     1, "y1",
     2, "y2",
     4, "y3"
)
```

```r
con2 <- dbConnect(SQLite(), ":memory:")
dbWriteTable(con2, "x", x)
dbWriteTable(con2, "y", y)
x <- tbl(con2, "x")
y <- tbl(con2, "y")
```

Inner joins

```
inner_join(x, y, by="key")

query <- con2 %>%
  dbSendQuery("SELECT x.key AS key, val_x, val_y
              FROM x
              INNER JOIN y USING(key)")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```r
inner_join(x, y)
```

```
## Joining, by = "key"
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0 [:memory:]
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

Inner joins (cont'd)

```r
inner_join(x, y, by=c("key", "key"))
```

```r
query <- con2 %>%
  dbSendQuery("SELECT x.key AS key, val_x, val_y
              FROM x
              INNER JOIN y ON x.key = y.key")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```r
inner_join(x, y, by=c("key", "key"))
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0 [:memory:]
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

Left joins

```r
left_join(x, y)
```

```r
query <- con2 %>%
  dbSendQuery("SELECT x.key AS key, val_x, val_y
              FROM x
              LEFT JOIN y ON x.key = y.key")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```r
left_join(x, y)
```

```
## Joining, by = "key"
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0 [:memory:]
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
```

Left excluding joins

```
x %>% left_join(y) %>% anti_join(y)
```

```
query <- con2 %>%
  dbSendQuery("SELECT x.key AS key, val_x, val_y
              FROM x
              LEFT JOIN y ON x.key = y.key
              WHERE y.key IS NULL")
result <- as_tibble(dbFetch(query))
dbClearResult(query)
result
```

```r
x %>% left_join(y) %>% anti_join(y)
```

```
## Joining, by = "key"
```

```
## Joining, by = c("key", "val_y")
```

```
## # Source:   lazy query [?? x 3]
## # Database: sqlite 3.22.0 [:memory:]
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     3 x3    <NA>
```

Right joins

```
# not supported by SQLite
right_join(x, y)

# not supported by SQLite
query <- con2 %>%
  dbSendQuery("SELECT x.key AS key, val_x, val_y
              FROM x
              RIGHT JOIN y ON x.key = y.key")
```

```r
right_join(x, y)
```

```
## Joining, by = "key"
```

```
## Error in result_create(conn@ptr, statement): RIGHT and FULL O
```

Thoughts on SQL vs `dplyr`

Some basic SQL rules:

- SQL statements that retrieve data from the database begin with SELECT; use `*` to select all columns
  - You don't need to use `dplyr::select()` unless you're subsetting columns
- In SQL statements, you always need to specify the table(s) you are manipulating
  - In `dplyr`, each table will usually be a separate variable
- SQL expect clauses following SELECT to be in a particular order
  - You can apply `dplyr` functions in any order (though results may differ)

Thoughts on SQL vs `dplyr` (cont'd)

Advantages and disadvantages:

- It is easy to chain functions seperately in `dplyr` and see intermediate results; this may be more difficult in SQL
- Reasoning about tables and operations may be easier when you can see intermediate results like this
- Some complex SQL operations are not supported by `dplyr` directly
- If the data fits in memory, you can use R programming to perform more complex operations
- If the data does not fit in memory, you may need to use SQL to perform the complex operation on the database
- Use whichever is easier for you, then pull data into R for visualization and analysis.

It is always good to be familiar with all available tools. Most data science workflows will involve incorporating multiple tools

Relational data examples with DBLP

Let's go back to the DBLP dataset for more practice with `dbplyr`.

Calculate the total number of distinct authors in the dataset.

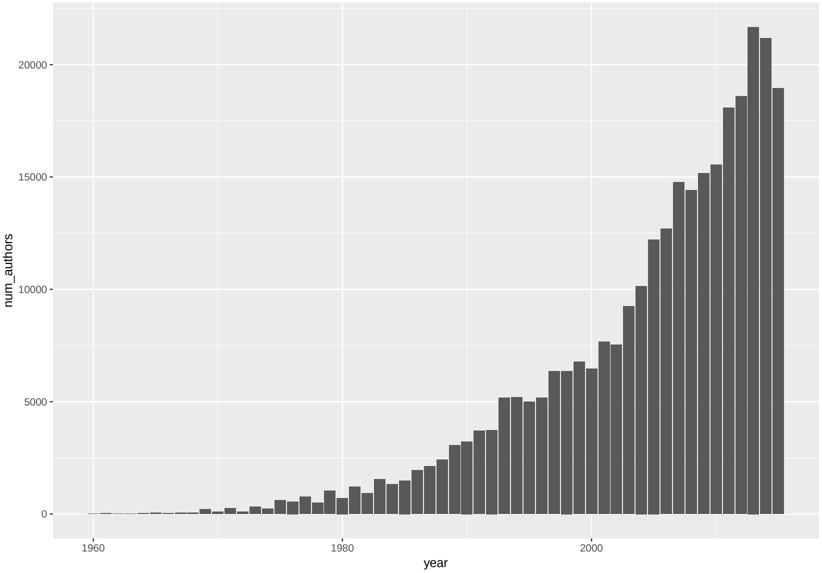
```r
dblp_authors %>%
  summarise(num_papers = n(),
            num_authors = n_distinct(name))
```

```
## # Source:   lazy query [?? x 2]
## # Database: sqlite 3.22.0
## #   [/Users/kuwisdelu/Dropbox/Northeastern/Courses/DS5110-Spr
##   num_papers num_authors
##        <int>       <int>
## 1     416445      126094
```

Calculate and plot the number of distinct authors published each year.

```r
dblp_authors %>%
  left_join(dblp_main) %>%
  group_by(year) %>%
  summarise(num_authors = n_distinct(name)) %>%
  collect() %>%
  ggplot() +
  geom_col(aes(x=year, y=num_authors))
```

```
## Joining, by = "k"
```

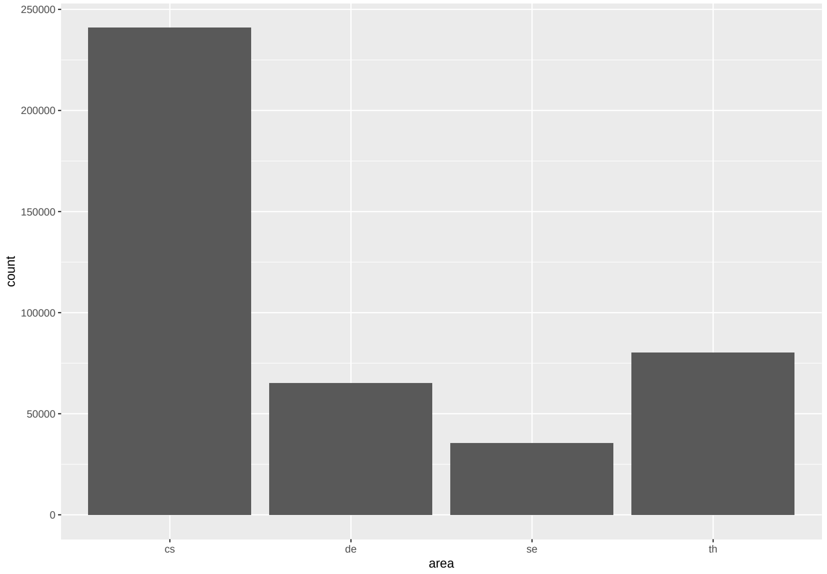


Calculate and plot the number of papers published in CS, DE, SE, and TH.

```
dblp_tidy <- dblp_authors %>%
  left_join(dblp_main) %>%
  select(k, year, conf, name, gender, prob, cs, de, se, th) %>%
  collect() %>%
  gather(key="area", value="indicator", cs, de, se, th) %>%
  filter(indicator == 1) %>%
  select(-indicator) %>%
  filter(gender %in% c('M', 'F'))
```

Some papers may belong to multiple areas, so we will count such a paper multiple times, once for each area. Since we are primarily interested in comparing between different areas of computer science, this is fine. However, if we wanted to count each paper only once, a different approach (adding an additional column for “interdisciplinary”?) would be required.

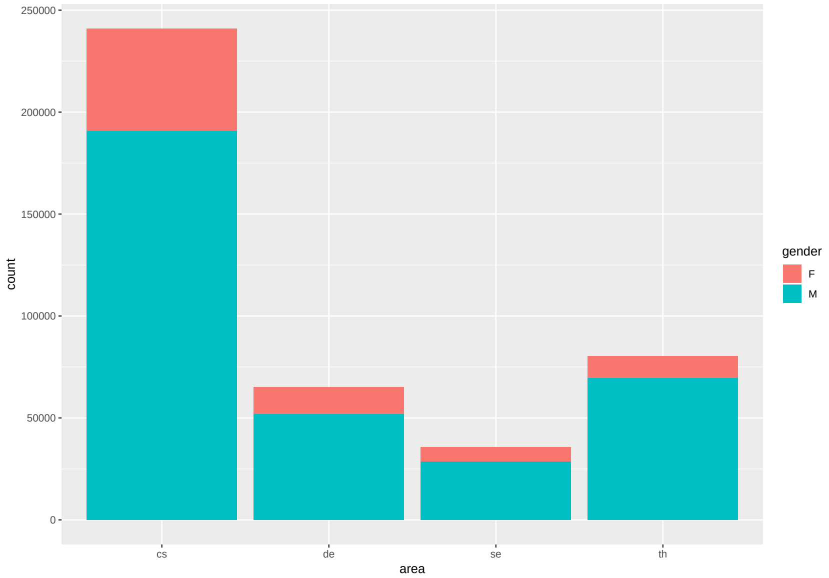
```
ggplot(dblp_tidy) + geom_bar(aes(x=area))
```



250000
200000
150000
100000
50000
0
count
cs
de
se
th
area

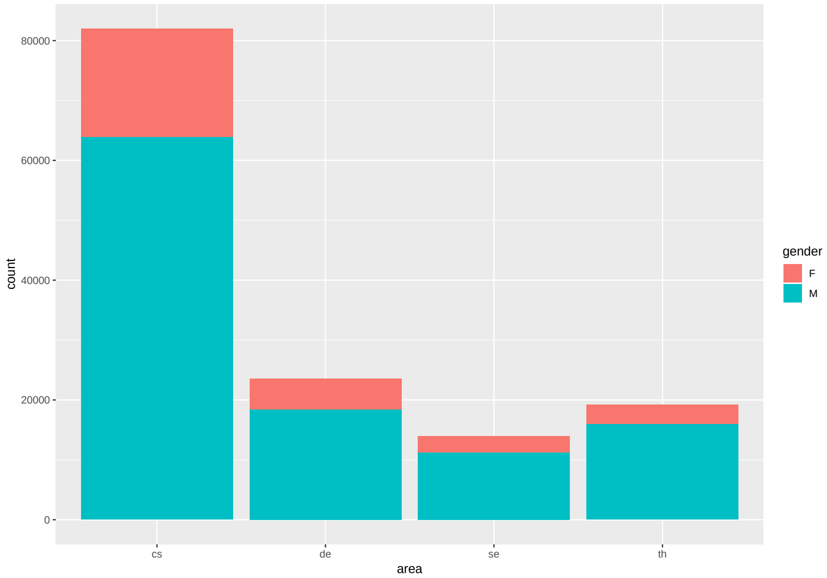
Plot the number of papers published in CS, DE, SE, and TH by gender.

```
ggplot(dblp_tidy) + geom_bar(aes(x=area, fill=gender))
```



Plot the number of distinct men and women published in CS, DE, SE, and TH.

```r
dblp_tidy %>%
  group_by(gender, area) %>%
  summarise(count = n_distinct(name)) %>%
  collect() %>%
  ggplot() +
  geom_col(aes(x=area, y=count, fill=gender))
```



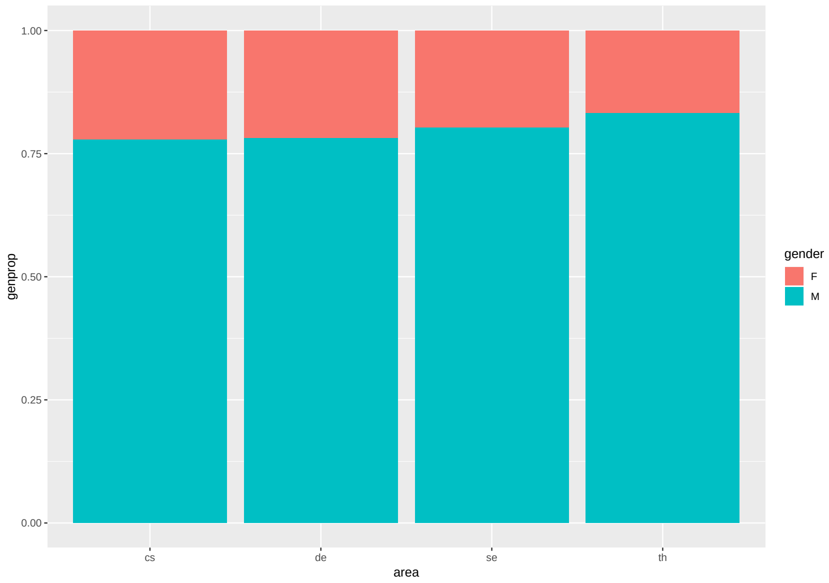
80000
60000
40000
20000
0
count
cs
de
se
th
area
gender
F
M

Plot the proportion of distinct men and women published in CS, DE, SE, and TH.

```r
auth_area <- dblp_tidy %>%
  group_by(area) %>%
  summarise(total = n_distinct(name))

genauth_area <- dblp_tidy %>%
  group_by(gender, area) %>%
  summarise(gencount = n_distinct(name))

left_join(genauth_area, auth_area) %>%
  mutate(genprop = gencount / total) %>%
  ggplot() +
  geom_col(aes(x=area,
               y=genprop,
               fill=gender))
```



1.00
0.75
0.50
0.25
0.00
genprop
cs
de
se
th
area
gender
F
M