Lecture 19 - grep

DSE 511

Drew Schmidt
2022-11-01

Announcements

- New homework
  - Due Monday Nov 7
  - Problem 1: read the instructions carefully!
  - Problem 5: Freebie - the answer is NOT $O(n)$
- Questions?

Content

- Background
- Basic grep
- Examples

Background

grep

- CLI tool
- What does it do?
  - Searches for patterns in text
  - Can be files, outputs from commands, ...
- Very powerful!

The Name

- Comes from `ed`
- Also explains some patterns in `vim`
- `g/re/p` - **g**lobally search for **re**gular expression and **p**rint matches

Common Uses for grep

- Software engineering
  - Find file containing function definition
  - Finding data source
  - Making sure some value is actually where you think it is
- Data science
  - Quick summaries (how many observations are from such-and-such class)
  - Software engineering!

Regular Expressions

- String-based pattern matching.
- Often looks like your cat just stomped on your keyboard.
- Concise (not necessarily readable!) descriptions of a set of strings.
- Find/Replace cranked up to 11.
- Regular Expression Language - Quick Reference
  https://learn.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference

Regular Expressions: Some Important Examples

- `^A` - is `A` at the beginning of the string?
- `A$` - is `A` at the end of the string?
- `^A$` - is the string *exactly* `A`?
- `(^A|A$)` - is `A` at the beginning or at the end of the string?

A Very Powerful Tool

`grep` your way to success!

Basic grep

Basic grep

- `grep <flags> pattern file(s)`
- `grep -R` - recursive
- `grep -F` - "fixed" strings
- `grep -i` - ignore (upper/lower) case in text
- `grep -v` - invert match

Basic grep

```
echo "hello world" > /tmp/example.txt
cat /tmp/example.txt
```

```
## hello world
```

```
grep hello /tmp/example.txt
```

```
## hello world
```

```
grep ello /tmp/example.txt
```

```
## hello world
```

Basic grep

```
grep goodbye /tmp/example.txt
```

```
grep Hello /tmp/example.txt
```

```
grep -i Hello /tmp/example.txt
```

```
## hello world
```

```
grep -v Hello /tmp/example.txt
```

```
## hello world
```

Examples

Contrived Example

- We're going to randomly generate a bunch of files
- Structure
  - Root path (arbitrary)
  - Two directories (single capital letter)
  - Some number of files (single lower case letter)
  - File is text of random "words"

Setup

```
num_rand_files = function(range = 5:15) sample(range, size = 1)
gen_dirs = function(sample_space, root_paths) {
  lapply(1:length(root_paths), function(i) {
    n_dirs = num_rand_files()
    root_path = root_paths[i]
    dirs = sample(sample_space, size = n_dirs)
    full_paths = file.path(root_path, dirs)
    sapply(full_paths, dir.create, recursive = TRUE)
    return(full_paths)
  })
}
```

Setup

```
gen_word = function(..., length = 1:10) paste0(sample(LETTERS, sample(length, 1), replace
gen_files = function(sample_space, root_paths) {
  lapply(1:length(root_paths), function(i) {
    n_files = num_rand_files()
    root_path = root_paths[i]
    for (j in 1:n_files) {
      n_words = num_rand_files(range = 1:1000)
      body = paste(sapply(1:n_words, gen_word), collapse = " ")
      file = file.path(root_path, sample(sample_space, size=1))
      cat(glue::glue("Writing to {file}"), "\n")
      writeLines(body, con = file)
    }
  })
}
```

Setup

```
library(magrittr)
set.seed(1234)
root = "~/tmp/example"
unlink(root, recursive = TRUE)
dirs = gen_dirs(LETTERS, root = root) %>% unlist()
subdirs = gen_dirs(LETTERS, root = dirs)
ret = parallel::mclapply(unlist(subdirs), gen_files, sample_space = letters)
```

What Was Generated?

```
find /tmp/example -name "[a-z]" | wc -l
```

```
## 1238
```

```
du -h /tmp/example/ | tail -n 1
```

```
## 7.3M    /tmp/example/
```

```
find . -type f -exec wc -w {} + | tail -n 1
```

```
##  1121638 total
```

Finding Files

```
find /tmp/example -name x | grep B
```

```
## /tmp/example/B/N/x
## /tmp/example/B/G/x
## /tmp/example/B/U/x
## /tmp/example/B/S/x
## /tmp/example/B/W/x
## /tmp/example/B/I/x
## /tmp/example/O/B/x
## /tmp/example/F/B/x
```

```
find /tmp/example -name x | grep ^/tmp/example/B
```

```
## /tmp/example/B/N/x
## /tmp/example/B/G/x
## /tmp/example/B/U/x
## /tmp/example/B/S/x
## /tmp/example/B/W/x
## /tmp/example/B/I/x
```

Finding Text In Files

```
grep -R ABC /tmp/example | wc -l
```

```
## 113
```

```
grep -R " ABC " /tmp/example | wc -l
```

```
## 3
```

```
grep -R -l " ABC " /tmp/example | sort
```

```
## /tmp/example/B/U/t
## /tmp/example/G/S/q
## /tmp/example/W/W/i
```

Finding Text In Files

```
grep -R AB /tmp/example | wc -l
```

```
## 1034
```

```
grep -R "^AB" /tmp/example | wc -l
```

```
## 2
```

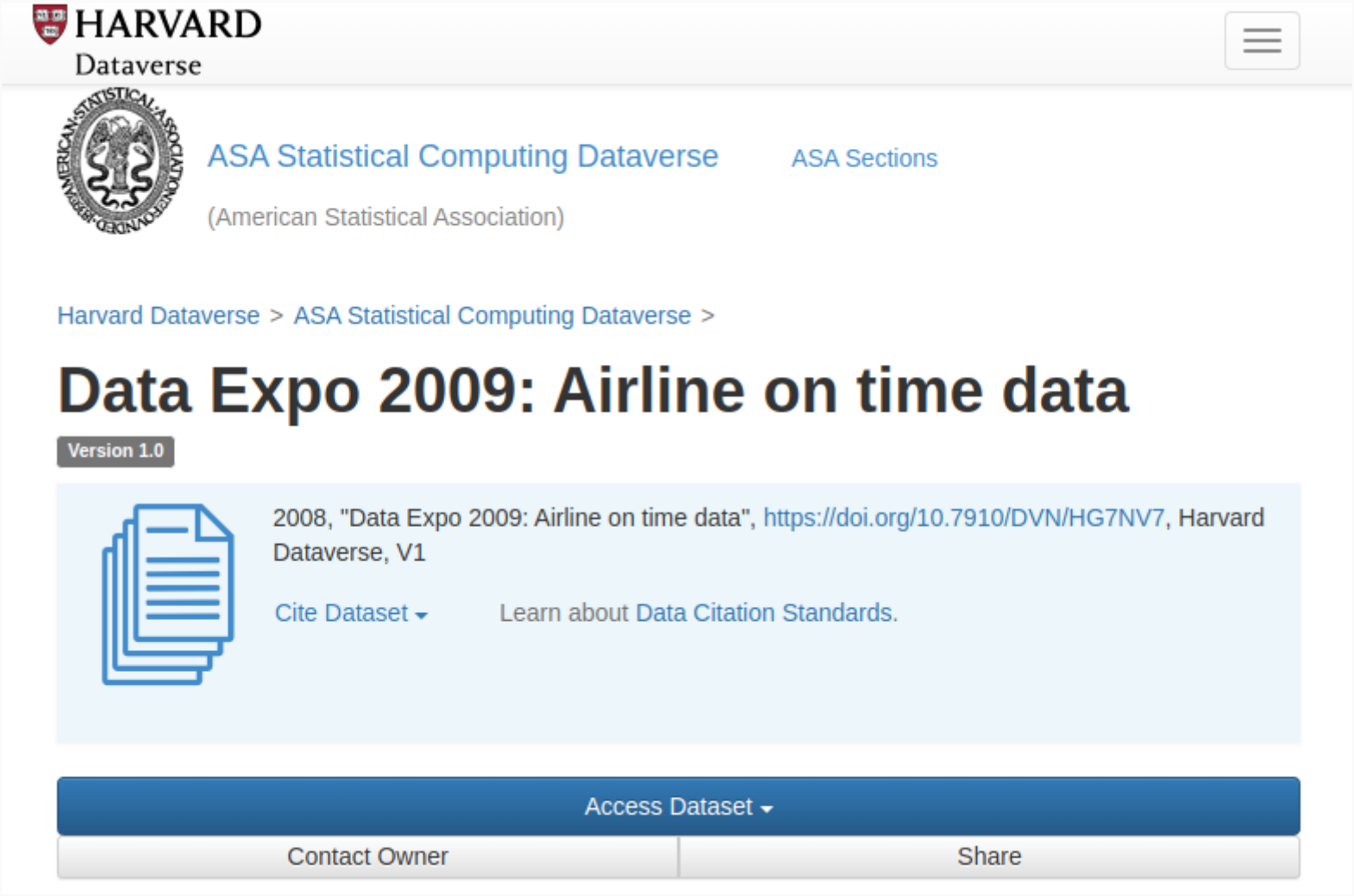
```
grep -R -l "^AB" /tmp/example
```

```
## /tmp/example/T/B/j
## /tmp/example/D/H/w
```

```
grep -o '^.\{3\}' /tmp/example/D/H/w
```

```
## ABK
```

Example: The Airlines Dataset



Example: The Airlines Dataset

```
ls ~/sw/data/airlines/csv
```

```
## 1987.csv
## 1988.csv
## 1989.csv
## 1990.csv
## 1991.csv
## 1992.csv
## 1993.csv
## 1994.csv
## 1995.csv
## 1996.csv
## 1997.csv
## 1998.csv
## 1999.csv
## 2000.csv
## 2001.csv
## 2002.csv
## 2003.csv
## 2004.csv
```

Example: The Airlines Dataset

Ungraded (for now?) homework: download and update the file names for "the airlines dataset".

- 22 csv files
- Years: 1987-2008
- File name: `${YEAR}.csv`
- Do it programmatically!

Example: The Airlines Dataset

```
du -h ~/sw/data/airlines/csv/1987.csv
```

```
## 122M    /home/iao/sw/data/airlines/csv/1987.csv
```

```
wc -l ~/sw/data/airlines/csv/1987.csv
```

```
## 1311827 /home/iao/sw/data/airlines/csv/1987.csv
```

Example: The Airlines Dataset

```
head -n 3 ~/sw/data/airlines/csv/1987.csv
```

```
## Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrier,FlightNum,Tai
## 1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,NA
## 1987,10,15,4,729,730,903,849,PS,1451,NA,94,79,NA,14,-1,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,NA
```

Example: The Airlines Dataset

Example: The Airlines Dataset

```
grep TYS ~/sw/data/airlines/csv/1987.csv | head -n 3
```

```
## 1987,10,1,4,614,615,559,552,UA,282,NA,45,37,NA,7,-1,TYS,BNA,152,NA,NA,0,NA,0,NA,NA,NA,NA,NA
## 1987,10,2,5,618,615,559,552,UA,282,NA,41,37,NA,7,3,TYS,BNA,152,NA,NA,0,NA,0,NA,NA,NA,NA,NA
## 1987,10,3,6,621,615,602,552,UA,282,NA,41,37,NA,10,6,TYS,BNA,152,NA,NA,0,NA,0,NA,NA,NA,NA,NA
```

```
head -n 3 ~/sw/data/airlines/csv/1987.csv | tr ',' ' '
```

```
## Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime UniqueCarrier FlightNum Tai
## 1987 10 14 3 741 730 912 849 PS 1451 NA 91 79 NA 23 11 SAN SFO 447 NA NA 0 NA 0 NA NA NA NA NA
## 1987 10 15 4 729 730 903 849 PS 1451 NA 94 79 NA 14 -1 SAN SFO 447 NA NA 0 NA 0 NA NA NA NA NA
```

```
grep TYS ~/sw/data/airlines/csv/1987.csv | wc -l
```

```
## 4795
```

Example: The Airlines Dataset

```
time grep TYS 1987.csv | wc -l
```

```
4795

real    0m0.066s
user    0m0.051s
sys     0m0.018s
```

```
system.time(nrow(read.csv("1987.csv")))
```

```
   user  system elapsed
  7.034   0.370   7.405
```

```
system.time(nrow(data.table::fread("1987.csv")))
```

```
   user  system elapsed
  0.886   0.075   0.235
```

Strictly Speaking...

```
echo -e "a\nb\nc\nd\ne" > /tmp/test.txt
```

R

```
con <- file("/tmp/test.txt", "r")
readLines(con = con, n = 1)
```

```
## [1] "a"
```

```
readLines(con = con, n = 1)
```

```
## [1] "b"
```

```
close(con)
```

Python

```
con = open("/tmp/test.txt", "r")
con.readline()
```

```
'a\n'
```

```
con.readline()
```

```
'b\n'
```

```
con.close()
```

Example: The Airlines Dataset

```
grep ,TYS, ~/sw/data/airlines/csv | wc -l
```

```
grep: 2001.csv: binary file matches
grep: 2002.csv: binary file matches
394298
```

```
ls ~/sw/data/airlines/csv | wc -w
```

```
22
```

```
echo "scale=4; 394298/22" | bc
```

```
17922.6363
```

Example: The Airlines Dataset

```
head -n 3 ~/sw/data/airlines/csv/1987.csv
```

```
## Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrier,FlightNum,Tai
## 1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,NA
## 1987,10,15,4,729,730,903,849,PS,1451,NA,94,79,NA,14,-1,SAN,SFO,447,NA,NA,0,NA,0,NA,NA,NA,NA,NA
```

```
grep [0-9],TYS, * | wc -l
```

```
grep: 2001.csv: binary file matches
grep: 2002.csv: binary file matches
193641
```

```
echo "scale=4; 193641/22" | bc
```

```
8801.8636
```

Wrapup

Wrapup

- `grep` your way to success!
- More on regular expressions next time with `sed`

Questions?