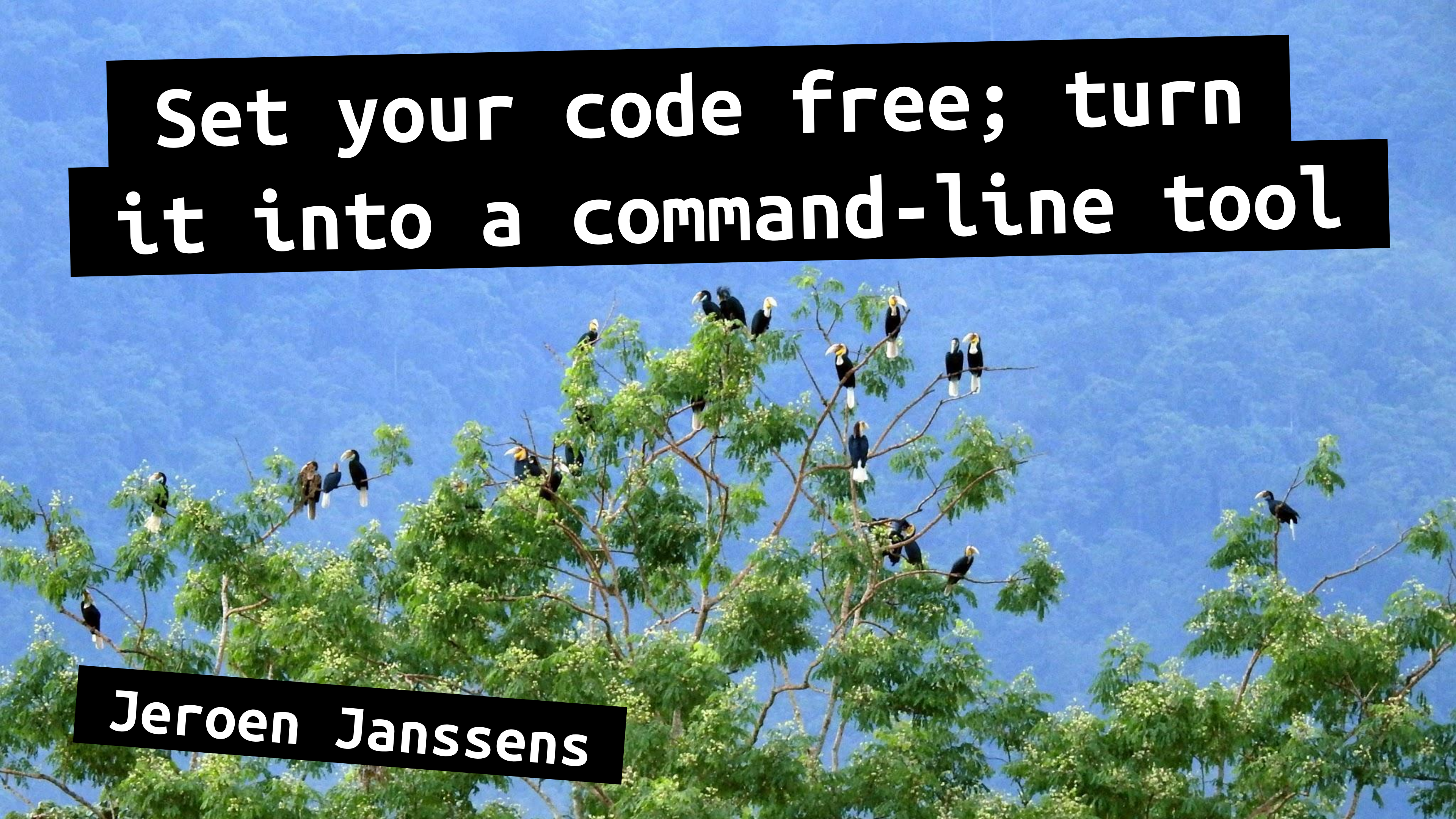
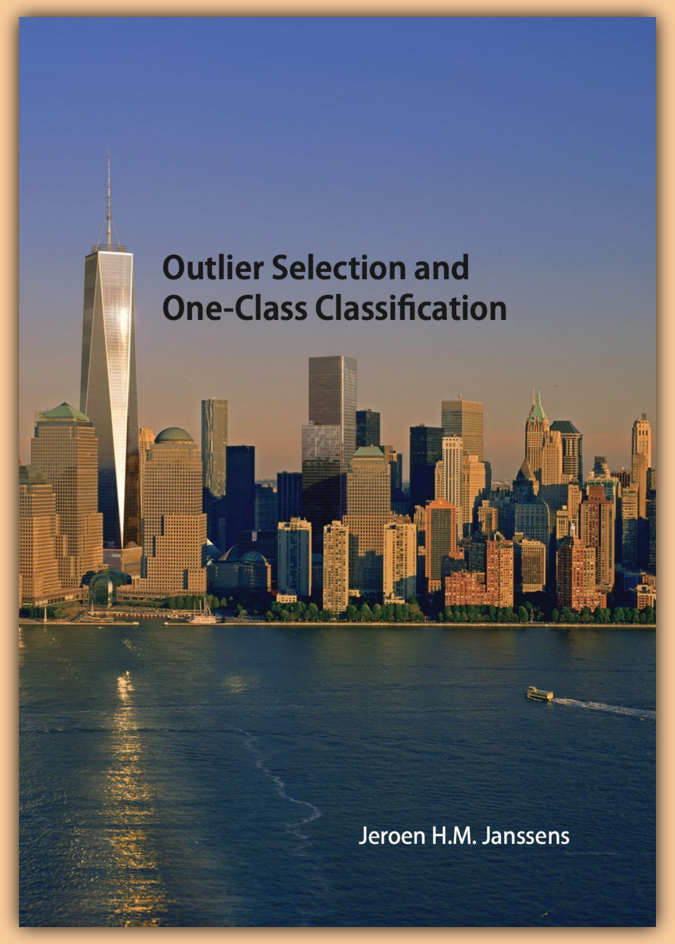
Set your code free; turn it into a command-line tool

Jeroen Janssens

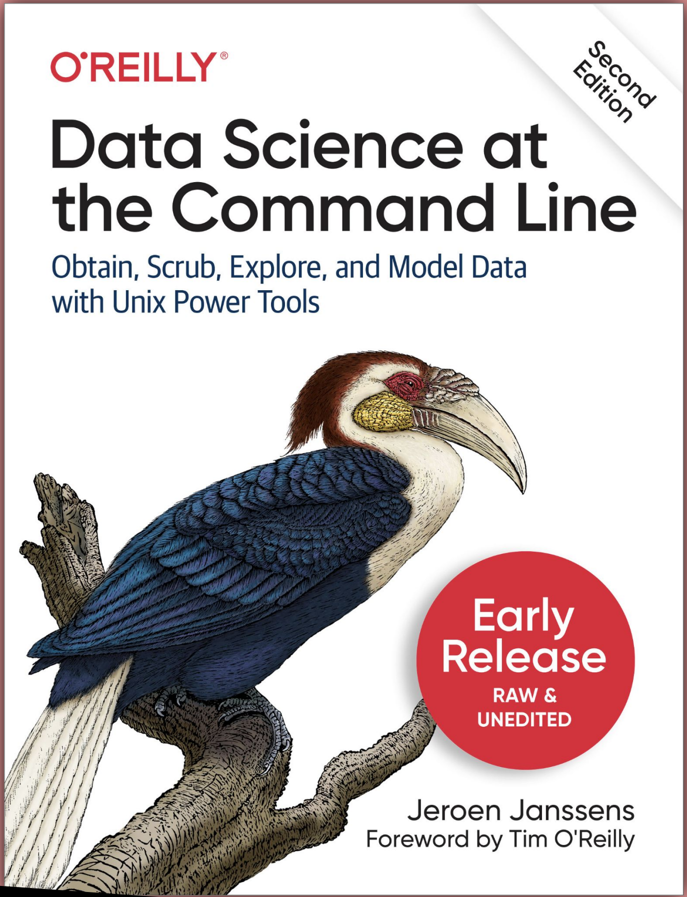


1. whoami
2. which
3. whereis
4. make
5. exit



Outlier Selection and One-Class Classification

Jeroen H.M. Janssens



O'REILLY

Second Edition

Data Science at the Command Line

Obtain, Scrub, Explore, and Model Data with Unix Power Tools

Early Release
RAW & UNEDITED

Jeroen Janssens
Foreword by Tim O'Reilly

datascienceatthecommandline.com

Jeroen Janssens, PhD

Founder & Trainer

☎ +31 619 628

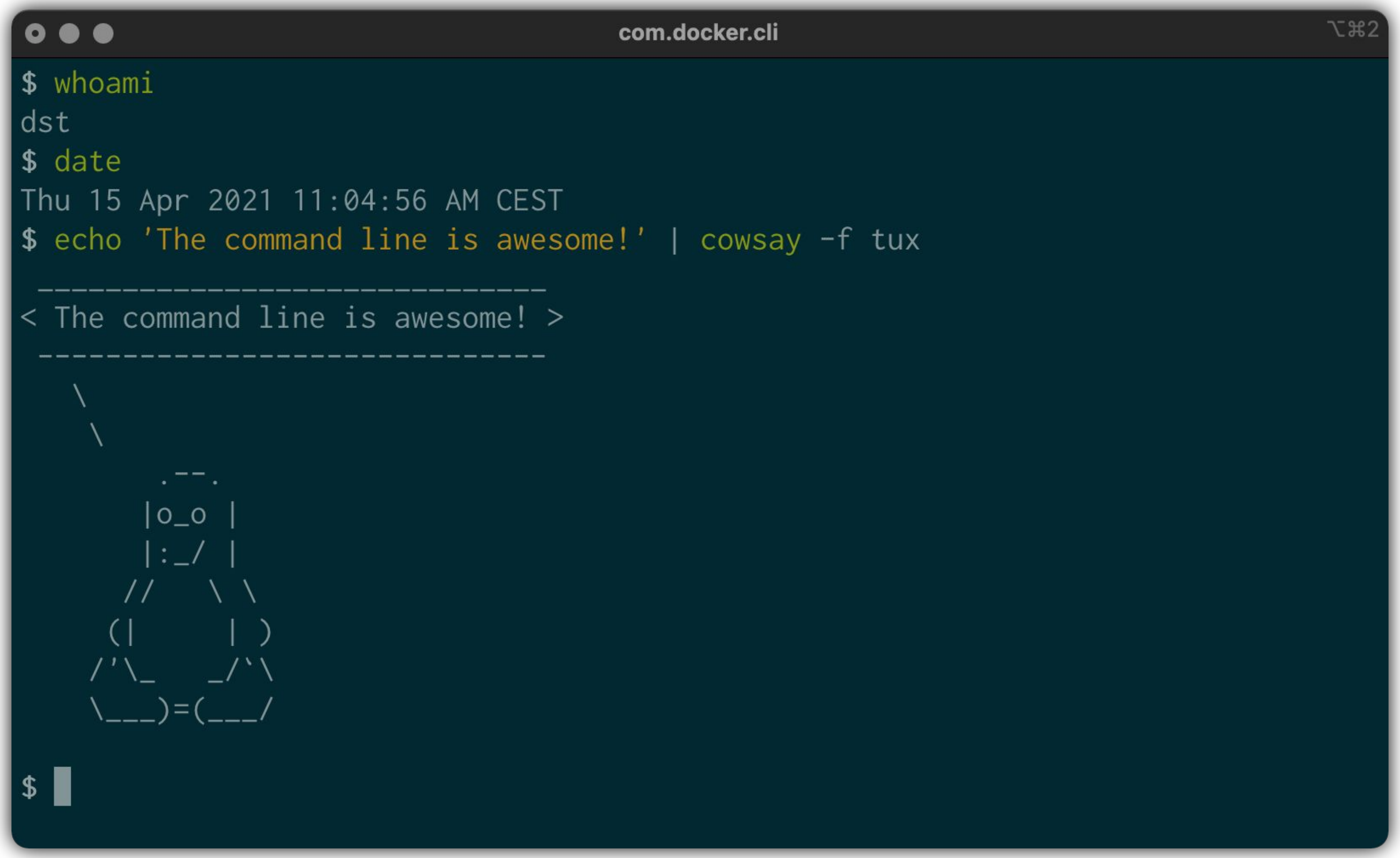
✉ jeroen@d

⌂ www.dat

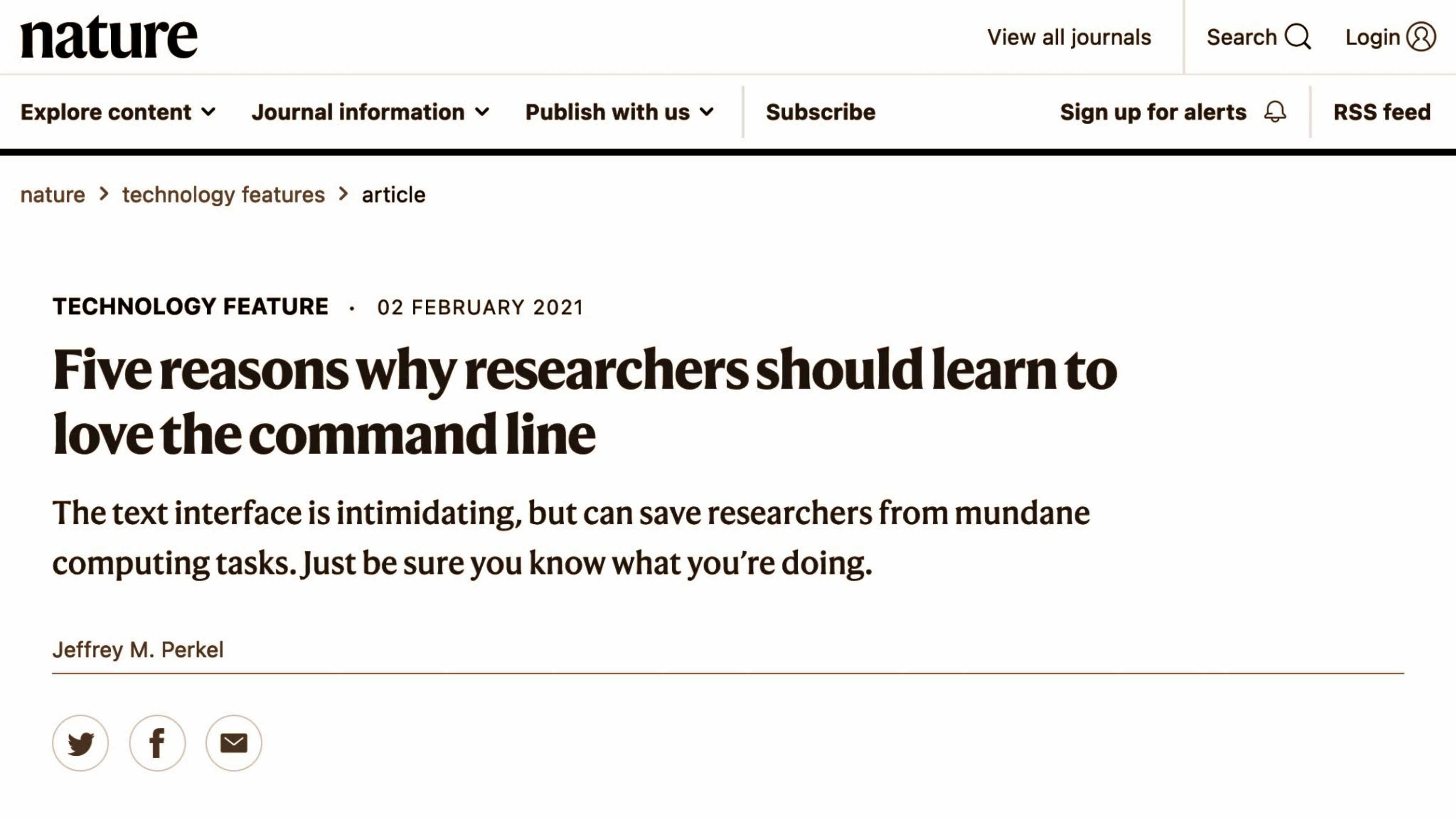
data science
workshops.com

Set your code free; turn it into a command-line tool

Free as in “bird”



```
$ whoami
dst
$ date
Thu 15 Apr 2021 11:04:56 AM CEST
$ echo 'The command line is awesome!' | cowsay -f tux
 ______________________________
< The command line is awesome! >
 ------------------------------
   \
    \
        .--.
       |o_o |
       |:_/ |
      //   \ \
     (|     | )
    /'\_   _/`\
    \___)=(___/

$ 
```



TECHNOLOGY FEATURE · 02 FEBRUARY 2021

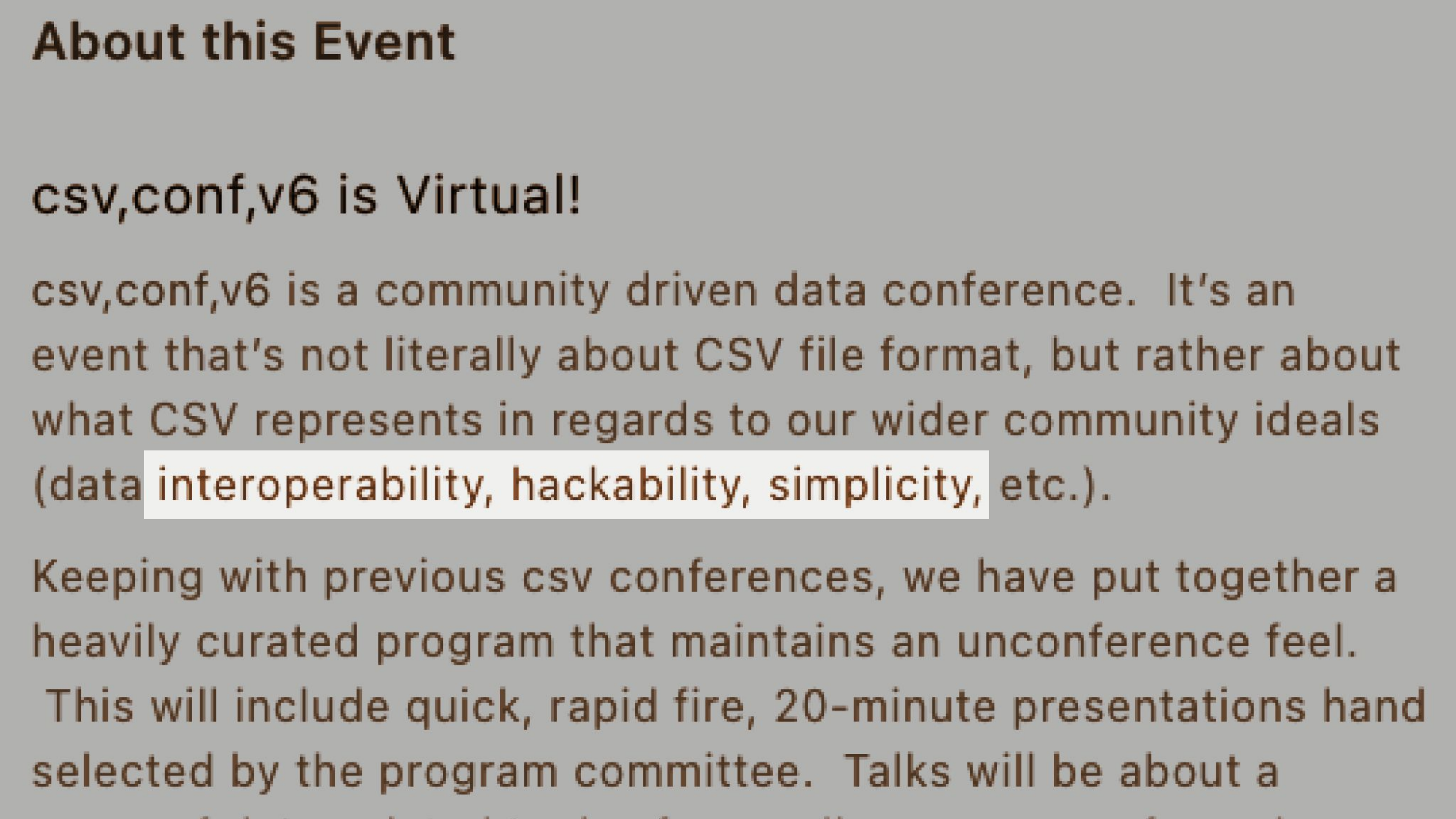
Five reasons why researchers should learn to love the command line

The text interface is intimidating, but can save researchers from mundane computing tasks. Just be sure you know what you're doing.

Jeffrey M. Perkel

Clever girl!

It’s a UNIX system!



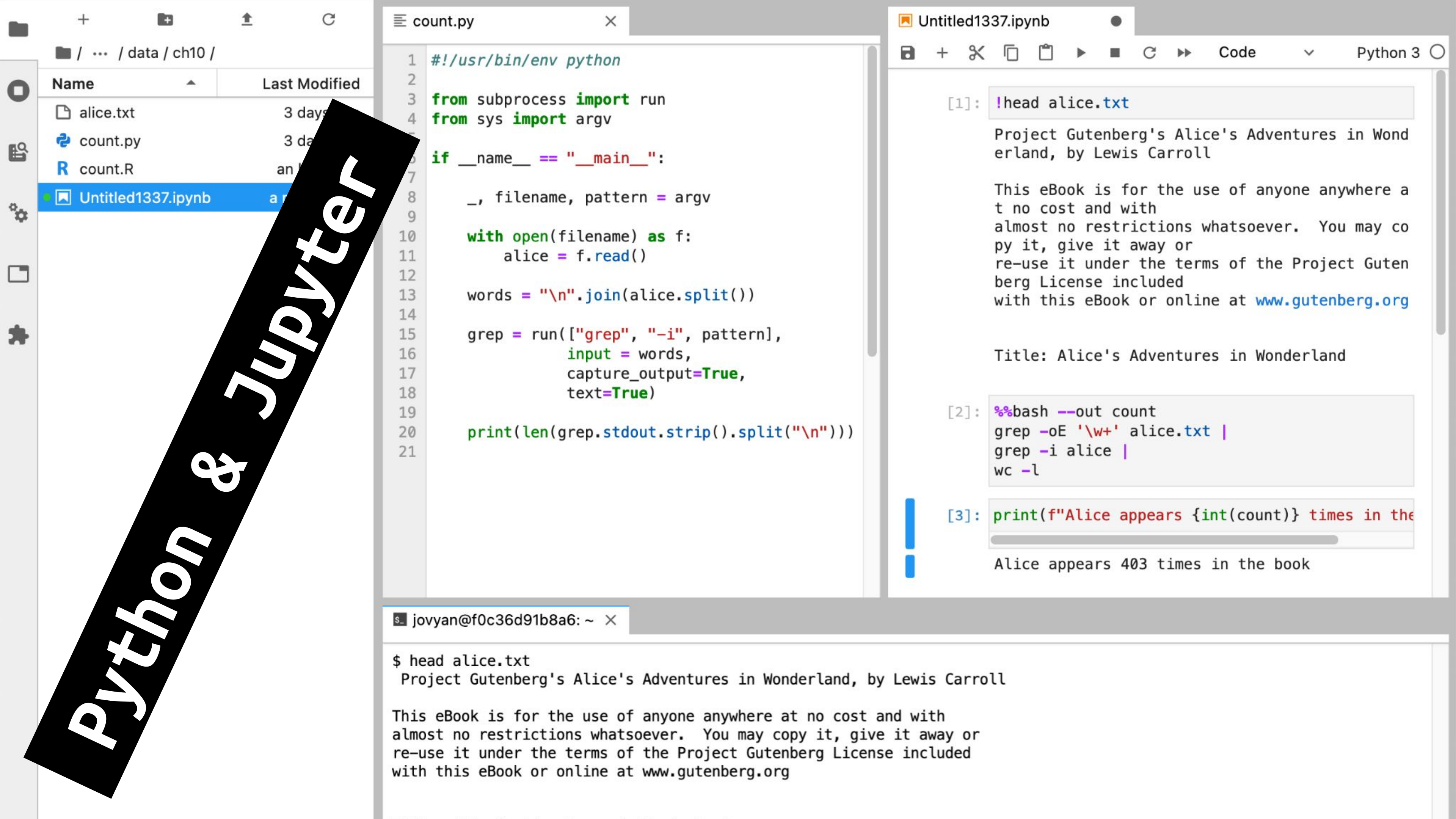
About this Event

csv,conf,v6 is Virtual!

csv,conf,v6 is a community driven data conference. It's an event that's not literally about CSV file format, but rather about what CSV represents in regards to our wider community ideals (data interoperability, hackability, simplicity, etc.).

Keeping with previous csv conferences, we have put together a heavily curated program that maintains an unconference feel. This will include quick, rapid fire, 20-minute presentations hand selected by the program committee. Talks will be about a

Python & Jupyter



```python
#!/usr/bin/env python

from subprocess import run
from sys import argv

if __name__ == "__main__":

    _, filename, pattern = argv

    with open(filename) as f:
        alice = f.read()

    words = "\n".join(alice.split())

    grep = run(["grep", "-i", pattern],
               input = words,
               capture_output=True,
               text=True)

    print(len(grep.stdout.strip().split("\n")))
```

```
[1]: !head alice.txt
```

```
Project Gutenberg's Alice's Adventures in Wond
erland, by Lewis Carroll

This eBook is for the use of anyone anywhere a
t no cost and with
almost no restrictions whatsoever.  You may co
py it, give it away or
re-use it under the terms of the Project Guten
berg License included
with this eBook or online at www.gutenberg.org


Title: Alice's Adventures in Wonderland
```

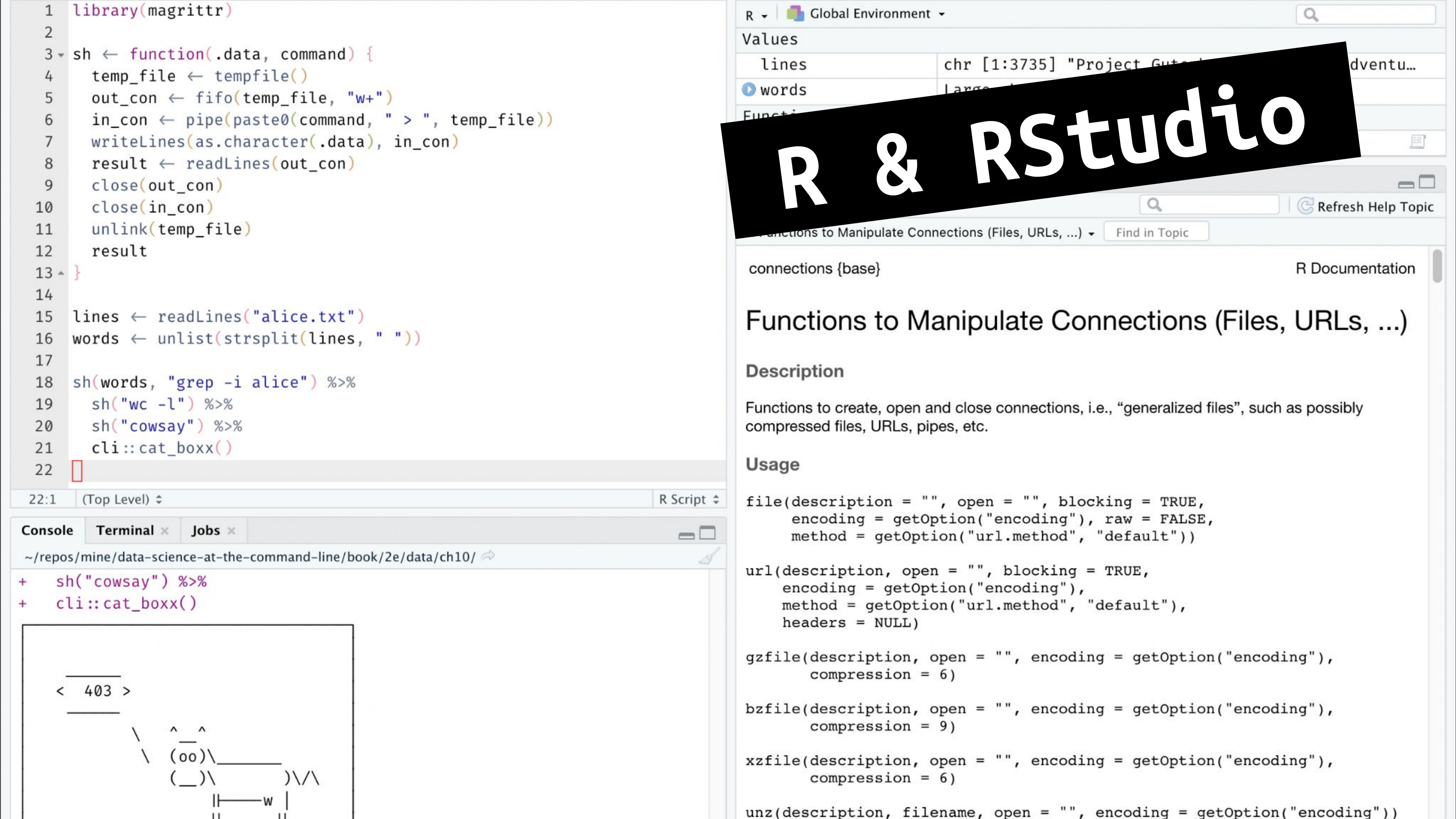
```
[2]: %%bash --out count
grep -oE '\w+' alice.txt |
grep -i alice |
wc -l
```

```
[3]: print(f"Alice appears {int(count)} times in the
```

```
Alice appears 403 times in the book
```

```
$ head alice.txt
 Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org
```

R & RStudio



```r
library(magrittr)

sh <- function(.data, command) {
  temp_file <- tempfile()
  out_con <- fifo(temp_file, "w+")
  in_con <- pipe(paste0(command, " > ", temp_file))
  writeLines(as.character(.data), in_con)
  result <- readLines(out_con)
  close(out_con)
  close(in_con)
  unlink(temp_file)
  result
}

lines <- readLines("alice.txt")
words <- unlist(strsplit(lines, " "))

sh(words, "grep -i alice") %>%
  sh("wc -l") %>%
  sh("cowsay") %>%
  cli::cat_boxx()

```

~/repos/mine/data-science-at-the-command-line/book/2e/data/ch10/

```
+   sh("cowsay") %>%
+   cli::cat_boxx()

   _____
  < 403 >
   -----
          \   ^__^
           \  (oo)\_______
              (__)\       )\/\
                  ||----w |
                  ||     ||
```

```
lines   chr [1:3735] "Project Gut… …dventu…
words   Large …
Functi…
```

connections {base} R Documentation

Functions to Manipulate Connections (Files, URLs, ...)

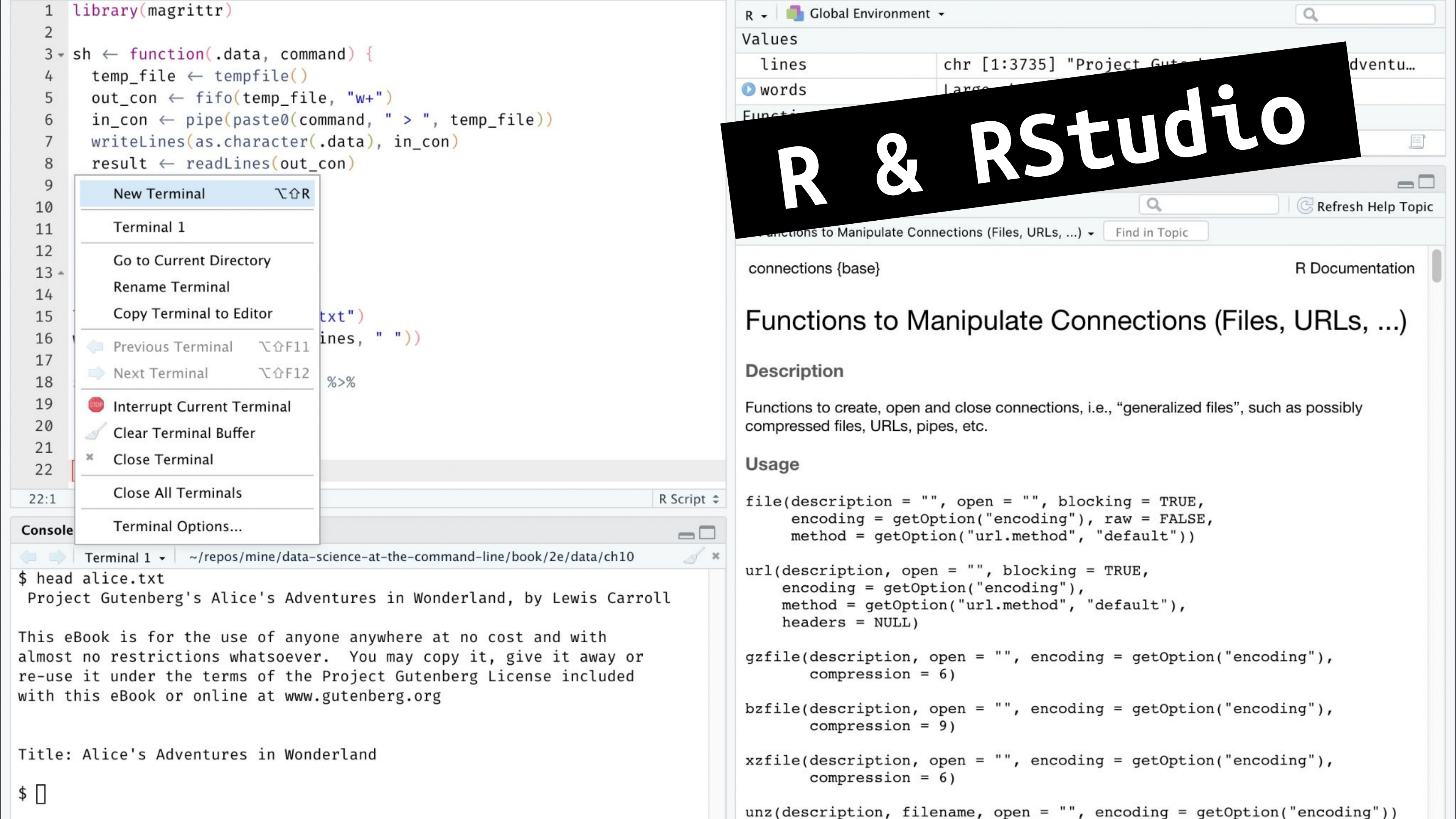
Description

Functions to create, open and close connections, i.e., “generalized files”, such as possibly compressed files, URLs, pipes, etc.

Usage

```
file(description = "", open = "", blocking = TRUE,
     encoding = getOption("encoding"), raw = FALSE,
     method = getOption("url.method", "default"))

url(description, open = "", blocking = TRUE,
    encoding = getOption("encoding"),
    method = getOption("url.method", "default"),
    headers = NULL)

gzfile(description, open = "", encoding = getOption("encoding"),
       compression = 6)

bzfile(description, open = "", encoding = getOption("encoding"),
       compression = 9)

xzfile(description, open = "", encoding = getOption("encoding"),
       compression = 6)

unz(description, filename, open = "", encoding = getOption("encoding"))
```

R & RStudio



```
library(magrittr)

sh ← function(.data, command) {
  temp_file ← tempfile()
  out_con ← fifo(temp_file, "w+")
  in_con ← pipe(paste0(command, " > ", temp_file))
  writeLines(as.character(.data), in_con)
  result ← readLines(out_con)
```

```
txt")
ines, " "))
%>%
```

```
$ head alice.txt
 Project Gutenberg's Alice's Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org


Title: Alice's Adventures in Wonderland

$ 
```

connections {base} R Documentation

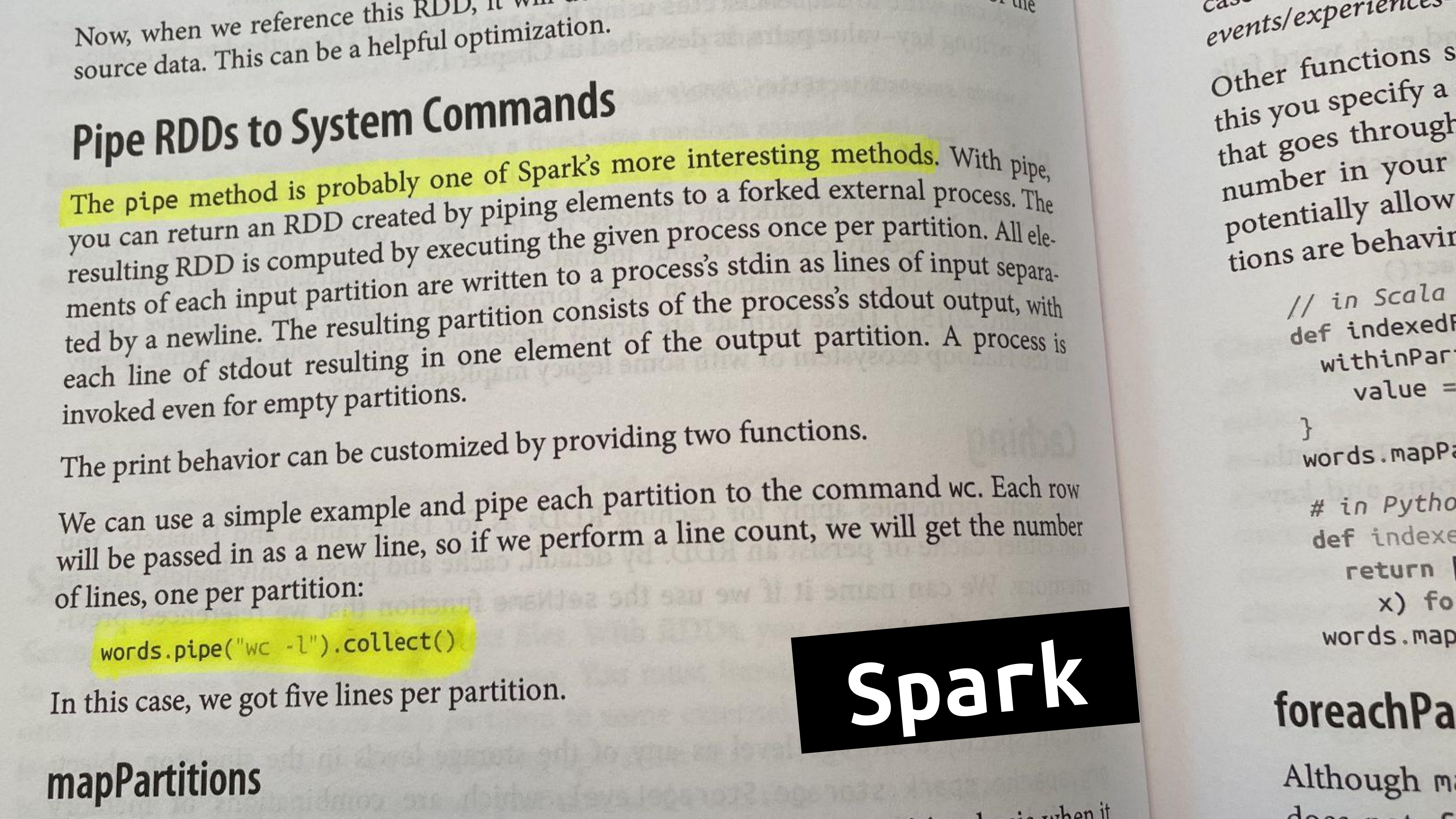
Functions to Manipulate Connections (Files, URLs, ...)

Description

Functions to create, open and close connections, i.e., “generalized files”, such as possibly compressed files, URLs, pipes, etc.

Usage

```
file(description = "", open = "", blocking = TRUE,
     encoding = getOption("encoding"), raw = FALSE,
     method = getOption("url.method", "default"))

url(description, open = "", blocking = TRUE,
    encoding = getOption("encoding"),
    method = getOption("url.method", "default"),
    headers = NULL)

gzfile(description, open = "", encoding = getOption("encoding"),
       compression = 6)

bzfile(description, open = "", encoding = getOption("encoding"),
       compression = 9)

xzfile(description, open = "", encoding = getOption("encoding"),
       compression = 6)

unz(description, filename, open = "", encoding = getOption("encoding"))
```



Now, when we reference this RDD, it will
source data. This can be a helpful optimization.

Pipe RDDs to System Commands

The pipe method is probably one of Spark's more interesting methods. With pipe, you can return an RDD created by piping elements to a forked external process. The resulting RDD is computed by executing the given process once per partition. All elements of each input partition are written to a process's stdin as lines of input separated by a newline. The resulting partition consists of the process's stdout output, with each line of stdout resulting in one element of the output partition. A process is invoked even for empty partitions.

The print behavior can be customized by providing two functions.

We can use a simple example and pipe each partition to the command wc. Each row will be passed in as a new line, so if we perform a line count, we will get the number of lines, one per partition:

```
words.pipe("wc -l").collect()
```

In this case, we got five lines per partition.

mapPartitions

Spark

events/experiences

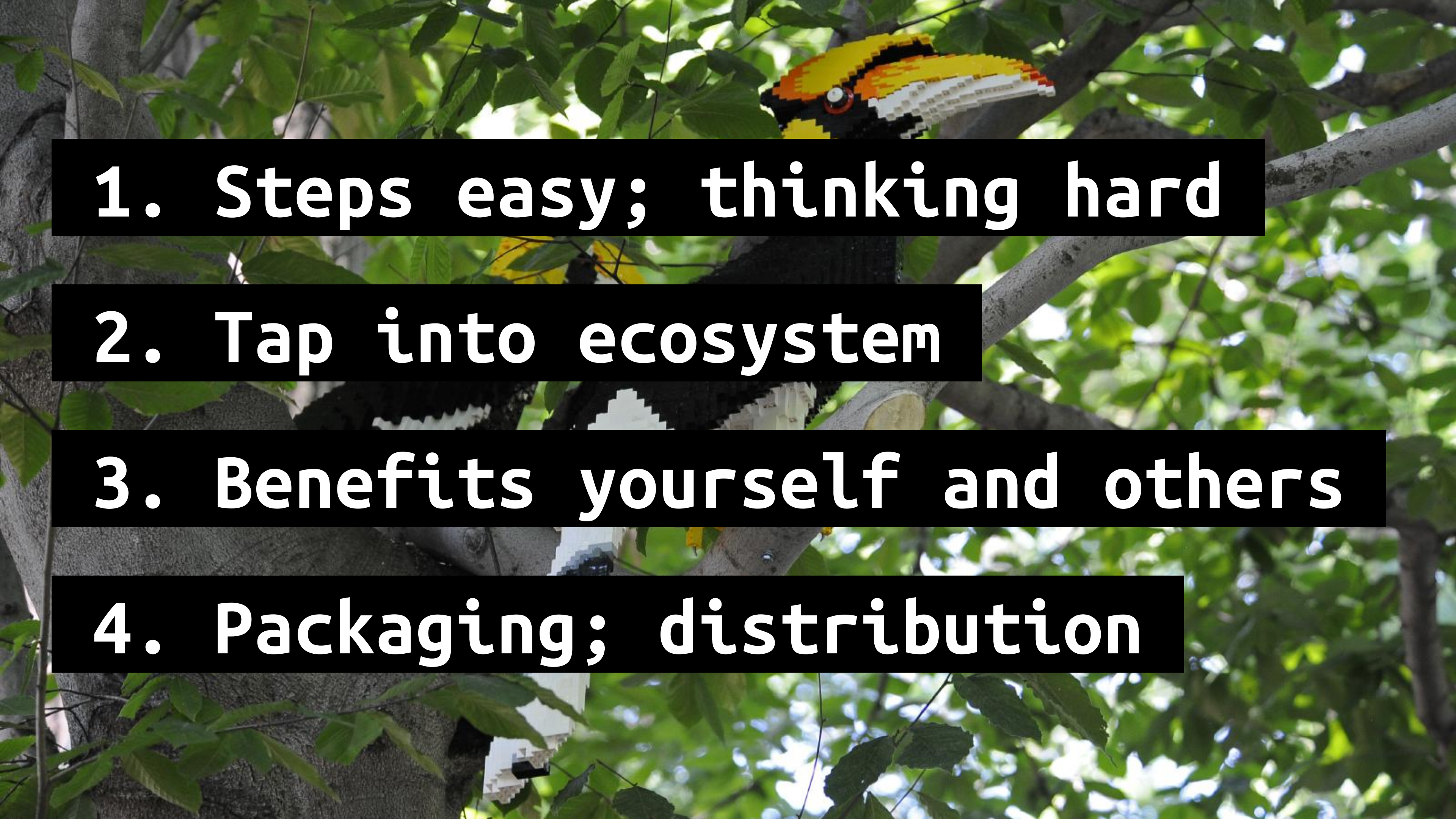
Other functions s
this you specify a
that goes through
number in your
potentially allow
tions are behavir

```
// in Scala
def indexedF
  withinPar
    value =
}
words.mapPa

# in Pytho
def indexe
  return
    x) fo
words.map
```

foreachPa

Although ma

1. cp!
2. argv!
3. stdin!
4. shebang#!
5. chmod!
6. PATH!



1. Steps easy; thinking hard
2. Tap into ecosystem
3. Benefits yourself and others
4. Packaging; distribution

Thanks!

@jeroenhjanssens