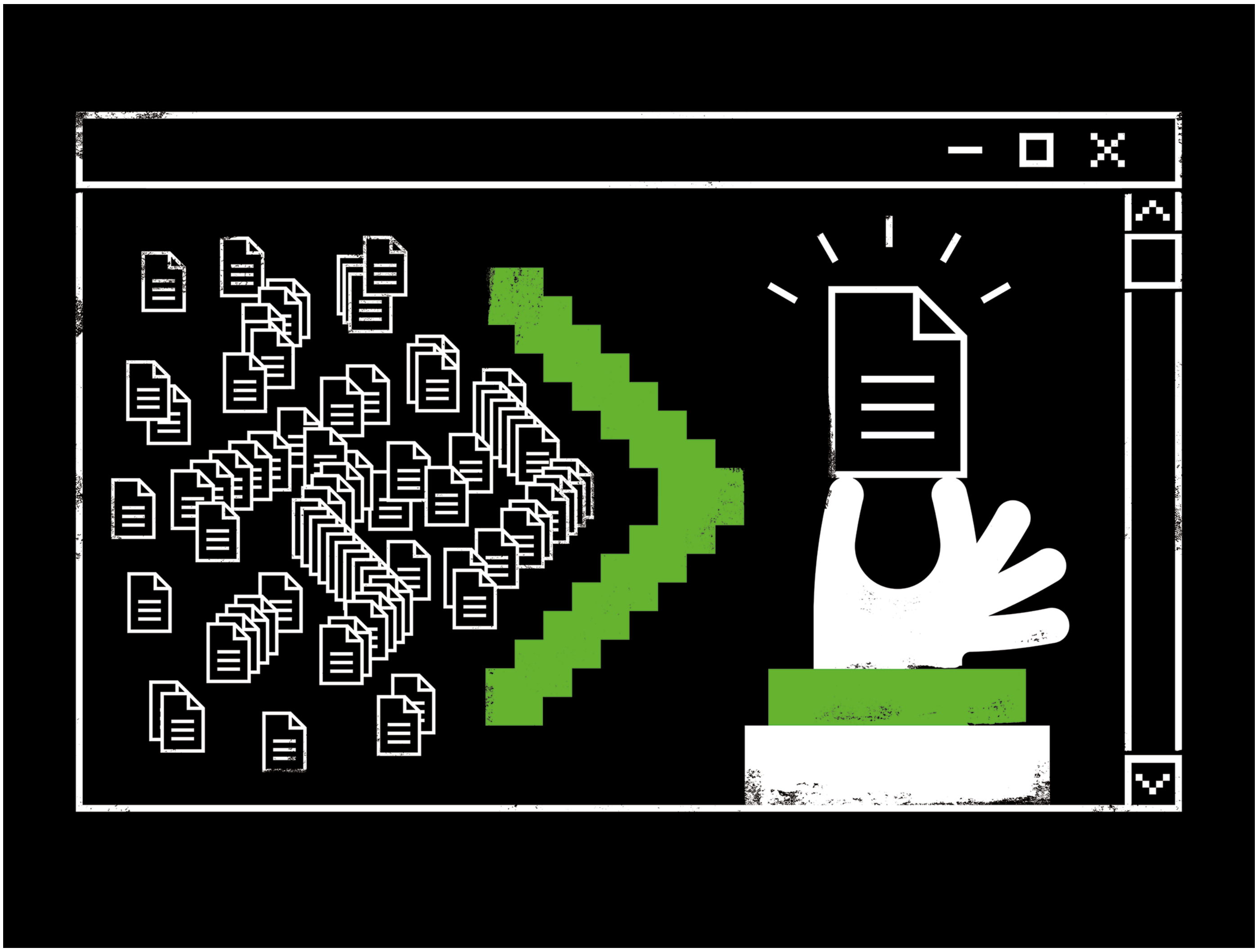

ILLUSTRATION BY THE PROJECT TWINS

FIVE REASONS 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. **By Jeffrey M. Perkel**

Jennifer Johnson's principal investigator had a simple request. Since 2018, their team had sequenced the DNA of some 1,300 silver fox (*Vulpes vulpes*) specimens, and the lab head wanted to know precisely how many bases it had collected, and how well those bases aligned to the reference genome.

Johnson, a bioinformatician at the University of Illinois at Urbana-Champaign, had the necessary data. But they were scattered across her hard drive. The most obvious solution was also the most painful: to open each file, find the required information, close and repeat — 1,300 times. So Johnson took another tack, using the command line.

Prebuilt into macOS and Unix systems, and available on Windows through such tools as the free 'Windows Subsystem for Linux' and MobaXterm, the command line (also called the shell) is a powerful text-based interface in which users issue terse instructions to create, find, sort and manipulate files, all without using the mouse. There are actually several distinct (although largely compatible) shell systems, among the most popular of which is Bash, an acronym for the 'Bourne again shell' (a reference to the Bourne shell, which it replaced in 1989).

Bash is both a collection of small utilities and a full-blown programming language, ranging from 'grep', a powerful text-search tool, to 'for loops', which allows users to repeat actions across multiple files. Johnson directed her terminal to scan her hard disk for sequencing data files, extract the needed information and compile them into a tidy spreadsheet. "That took me less than 10 minutes," she says; recomputing the data would have taken a full day.

Many computational disciplines, such as bioinformatics, rely heavily on the command line. But all researchers who use computers can benefit from it, says Jeroen Janssens, principal instructor of Data Science Workshops in Rotterdam, the Netherlands, and author of the 2014 book *Data Science at the Command Line*. "The mouse doesn't scale," Janssens explains. For instance, although it is certainly possible to rename a file by pointing and clicking, that task becomes tedious when it is scaled to hundreds or thousands of files.

Here, we highlight five ways the command line can ease your computational research.

Wrangle files

Perhaps the shell's most powerful feature is the ability to repeat simple tasks across multiple files. Researchers could, for instance, systematically rename their files to add a date stamp, or convert them from one format to another (see our example code at https://github.com/jperkel/nature_bash).

During his postdoctoral studies, Casey Greene's adviser insisted that all images of figures that were used in presentations had a black background. Greene got pretty good, he says, at opening figures in an image editor, inverting and colour-rotating them, and repeating. "But at some point, it turns out life is too short to continue importing and colour-rotating in even a free software program that is relatively easy to use," says Greene, who now directs the Center for Health Artificial Intelligence at the University of Colorado School of Medicine in Aurora. So, he turned to the command line – specifically, the free and open-source image-manipulation tool ImageMagick, using a for loop to repeat the operation across all his files:

```
for file in *.png; do convert $file -channel RGB -negate -modulate 100,100,200 out_$file; done;
```

Handle big data

Some data sets are simply too big to handle. For a project studying digital object identifier (DOI) metadata, Elizabeth Wickes, an information scientist at the University of Illinois at Urbana–Champaign, harvested millions of XML files. But committing those files to a version-control repository overloaded her system: "GitHub Desktop and my system indexing just barfed on it," she says. Using the 'git' command-line tool, however, she tackled the problem in an hour and a half.

Similarly, Lynette Strickland, an evolutionary biologist at Texas A&M University in Corpus Christi, has documented millions of genetic variants for her research on invasive lionfish (*Pterois* sp.). The data are too large for Microsoft Excel, which caps spreadsheets at about one million rows. So, Strickland used the shell to identify the records (spreadsheet rows) that exceeded a certain quality threshold, extract just the columns she needed, and save the data to a new file, using a command that looks something like this (assuming the quality score is in column 4, the cut-off threshold is 50, and the desired columns are 1–4):

```
awk -F, '{ if ($4 > 50) print $0 }' datafile.csv | cut -d, -f1-4 > newdatafile.csv
```

"By just taking the specific information that I need from it using [the shell], I can really, really condense it into something that I can actually work with and visualize," she says.

Manipulate spreadsheets

Shell commands perform seemingly simple operations. The 'cut' command, for instance, extracts one or more columns from a spreadsheet; 'wc' counts words, lines or characters; 'awk' filters files for lines that match a certain condition; and 'sed' manipulates text 'streams'. But these simple commands can be strung together using 'pipe' ('|'), a shell feature that funnels the output of one command into another, thus creating powerful bespoke workflows. "It's very good for what we call 'whip-it-up-itude' – getting things going quickly, prototyping," says Tom Ryder, a systems administrator based in Palmerston North, New Zealand, and author of the 2018 book *Bash Quick Start Guide*.

"Shell commands can be stored in text files called scripts, which can be saved, shared and version-controlled."

The following command, for instance, combines five utilities to count the unique gene names in a gene-expression data set:

```
cut -f1 GEOdataset.csv | sed -E 's/^>//' | sort | uniq | wc -l
```

These steps extract (cut) the first column (containing the name) from the spreadsheet; remove a leading greater-than symbol (sed); sort the list alphabetically; reduce that list to its unique values (uniq); and print the total number of lines (that is, gene names) to the screen (wc).

Parallelize your work

Christina Koch, a research computing facilitator at the University of Wisconsin–Madison, works at a computing centre that provides remote access to some 14,000 nodes and terabytes of memory. Suppose, Koch says, that a bioinformatician has a computational workflow for analysing gene-expression data sets. Each data set takes a day to process on their computer, and the researcher has 60 such data sets. "That's two months of non-stop running," she says. But, by sending the job to a computer cluster using the 'secure shell' command, 'ssh', which opens an encrypted portal to the remote system, the researcher can parallelize the computations across 60 computers. "Instead of two months, it takes one day."

Even without such power, 'ssh' provides the ability to work remotely – an especially useful feature during a pandemic. With COVID-19 lockdowns looming in 2020, Gabriel Devenyi, a systems administrator and programmer at the Douglas Mental Health University Institute in Montreal, Canada, worked to ensure that the researchers in his facility were set up to stay productive, wherever they happened to quarantine. "Without having this in place, none of the students would have been able to do anything," he says.

Automate

Shell commands can be stored in text files called scripts, which can be saved, shared and version-controlled, enhancing reproducibility. They can also be automated. Using the 'cron' command, users can schedule scripts to run when it's convenient for them. For instance, says Wickes, some websites ask that users who plan to 'scrape' content do so during off-peak hours – in the case of the PubMed literature database, between 9 p.m. and 5 a.m. Eastern time. "You can have [the script] run only at the times you are allowed," she says. Alternatively, users can download data to their primary computer daily, because many shared systems routinely delete older files.

Warning: no undo

The flip side of this power is the shell's intimidating interface, often just a dollar sign and a cursor. "That's scary for some people," says Janssens. The shell provides little in the way of help. And it can be "unforgiving", Janssens notes. An improperly positioned space can change a command's meaning, and few commands will by default ask if you know what you're doing. "If you have the right to do it within the system, [the shell] assumes that you meant to do it the way you said it," Devenyi says. "And so you can do lots of various dangerous things."

The classic example is rm -rf * – a command that deletes all files from the current directory and everything below. If executed from the wrong location, crucial work can be lost. "We've all done that," says Johnson. So proceed with caution.

One simple trick is to use the 'echo' command to ensure that you're specifying the files you intend. "'Echo before execute' is a very good rule of thumb," says Greene. Some commands provide a 'dry-run' mode, which reports what they intend to do, and/or an 'interactive' mode, which prompts the user before making changes. Users can also set variables to prevent the computer from overwriting files, or to exit when there is an error ('noclobber' and 'pipefail', respectively). And they should avoid running commands while they have administrative privileges.

"Life comes at you fast," says Wickes. "And science can come at you fast sometimes." The shell makes it possible to handle the unexpected. Besides, says Koch, it's fun. "It's so powerful, and I feel like such a cool nerd when I'm using it," she says. "You can feel very competent."

Jeffrey M. Perkel is technology editor at *Nature*.