1 Delivery

Download `jan2017articles.csv` and `example.bed` files to solve the next questions.

Question 1 (1p)

Take a look at the last 10 lines of the `jan2017articles.csv` file. Which command are you going to use? Modify the command to show just the last line of the file.

Answer

This can be easily accomplished using `tail` and its `-n` option.

```
# Show last 10 lines
tail jan2017articles.csv
```

```
04 Jan 2017,Article,Seth Kenlon,By Jove! It's a lightweight alternative to
    Vim,14,/article/17/1/jove-lightweight-alternative-vim,Text editors,1346
04 Jan 2017,Article,Preston Ward,DronePan: An app that captures panorama views
    with your aircraft,2,/article/17/1/dronepan,Hardware,533
03 Jan 2017,Article,Shawn Powers,4 hot skills for Linux pros in
    2017,10,/article/17/1/yearbook-4-hot-skills-linux-pros-2017,"2016 Open Source
    Yearbook, Yearbook, Careers",708
03 Jan 2017,Article,Anna Ossowski,What does cross stitch have to do with
    programming? More than you
    think,5,/article/17/11/traditional-arts-crafts-code-programming,"JavaScript,
    Programming",1997
03 Jan 2017,Article,Mihai Raulea,Tapitoo OpenCart: An open source e-commerce
    mobile app,1,/article/17/1/tapitoo-opencart,Mobile,1127
03 Jan 2017,Article,Sam Knuth,Avoid echo chambers and make open
    decisions,1,/open-organization/17/1/avoid-echo-chambers-make-open-decisions,The
    Open Organization,723
02 Jan 2017,Article,Richard Fontana,7 notable legal developments in open source
    in 2016,3,/article/17/1/yearbook-7-notable-legal-developments-2016,"2016 Open
    Source Yearbook, Yearbook, Licensing",2359
02 Jan 2017,Article,Scott Nesbitt,3 tips for effectively using wikis for
    documentation,1,/article/17/1/tips-using-wiki-documentation,"Documentation,
    Wiki",710
02 Jan 2017,Article,Jen Wike Huger,The Opensource.com preview for
    January,0,/article/17/1/editorial-preview-january,,358
02 Jan 2017,Poll,Jason Baker,What is your open source New Year's
    resolution?,1,/poll/17/1/what-your-open-source-new-years-resolution,,186

02 Jan 2017,Poll,Jason Baker,What is your open source New Year's
    resolution?,1,/poll/17/1/what-your-open-source-new-years-resolution,,186
```

```
# Show last line
tail -n1 jan2017articles.csv
```

```
02 Jan 2017,Poll,Jason Baker,What is your open source New Year's
    resolution?,1,/poll/17/1/what-your-open-source-new-years-resolution,,186
```

Question 2 (1p)

Extract all lines that belong to January 6th from the file and store them in a new file named `reyes.csv`. Check that the first line of the new file has the expected values.

Answer

This can be done using any tool to search plain-text data, such as **`grep`** or **`awk`**. For the sake of practising and learning, we'll use **`awk`** as much as possible.

```
# First method (could cause some problems)
awk '/06 Jan/ { print $0 }' jan2017articles.csv > reyes1.csv
# Second method (more reliable)
awk '{ if($1 == "06␣Jan␣2017") print $0 }' FS=","
    jan2017articles.csv > reyes2.csv
```

We'll see however, that in this case both methods are equally valid:

```
# Empty output (both files are the same)
diff reyes1.csv reyes2.csv
```

```

```

Let's check the content of the first line to check if it has the expected values:

```
head -n1 reyes1.csv
```

```
06 Jan 2017,Article,Jen Wike Huger,"Top 5: Hot programming trends, How Linux got
    to be Linux, and more",0,/article/17/1/top-5-january-6,Top 5,241
```

Question 3 (1p)

Use the original `csv` to find which entries have `0` at the comment count only for those entries from January 25th.

Answer

The advantages of using **`awk`** is that we can do more than one operation in a robust and easy to understand way.

```
awk '{ if($1 == "25␣Jan␣2017" && $5 == 0) print $0 }' FS=","
    jan2017articles.csv
```

```
25 Jan 2017,Article,Ben Cotton,24 Pull Requests challenge encourages fruitful
    contributions,0,/article/17/1/24-Pull-Requests-challenge,Programming,429
25 Jan 2017,Article,Rikki Endsley,Announcing the 2016 Open Source Yearbook:
    Download now,0,/article/17/1/announcing-2016-open-source-yearbook,2016 Open
    Source Yearbook,58
```

We could've done this by `grep`ping both patterns, but with **`awk`** we not only search for the patterns, but we specify where to find them, which reduces the possibility of committing any filtering error.

Question 4 (1p)

Now count the number of entries of Question 3 and compare with the total number of entries.

Answer

We'll simply use `wc` to handle the word-counting and store the number in two different variables.

```
# We count the lines excluding the header
counttot=$(cat jan2017articles.csv | tail -n +2 | wc -l)
# We count the lines of question 3's output
count3=$(awk '{ if($1 == "25␣Jan␣2017" && $5 == 0) print $0 }'
    FS="," jan2017articles.csv | wc -l)

# Calculation using a bc (byte code) pipe
echo $counttot - $count3 | bc
```

```
90
```

Let's notice that for comparison we used the difference between both values. We could have just printed both values, but that's probably just a matter of opinion.

Question 5 (1p)

Now use `example.bed` file. In this file, we are interested in the exon sizes of each entry. They are located in field number 11. Now you have to get the exon sizes of the first 10 entries of the file.

Answer

Being unfamiliar with the format of `.bed` files, we first of all check the data structure [1]:

> 11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.

Knowing this, we proceed as instructed.

```
awk '{ print $11 }' example.bed | head
```

```
541,322,429,
385,143,144,186,125,573,
258,19,143,144,186,125,573,
370,107,97,101,57,77,163,98,80,263,
315,113,97,101,57,77,163,98,101,
370,113,97,101,57,77,163,98,80,257,
370,104,97,101,57,77,163,98,80,263,
370,113,97,101,57,77,163,98,80,263,
293,93,81,72,132,87,72,86,133,189,275,
203,96,81,72,132,87,72,86,133,189,275,
```

Question 6 (1p)

How would you remove the last comma?

Answer

We'll compare different methods using **awk** and gsub to substitute patterns/characters in the data.

```
# Using awk and substring (removes last character)
awk '{print substr($11, 1, length($11)-1)}' example.bed | head -n2
# Using awk and gsub (removes last character)
echo
awk '{ gsub(/.$/,"",$11) ; print $11 }' example.bed | head -n2
# Using awk and gsub (removes last comma; probably the best)
echo
awk '{ gsub(/,$/,"",$11) ; print $11 }' example.bed | head -n2
```

```
541,322,429
385,143,144,186,125,573

541,322,429
385,143,144,186,125,573

541,322,429
385,143,144,186,125,573
```

Question 7 (1p)

How would get the smallest size from each of the records? The result should provide a number for each line of the input.

Answer

This can be done with a classical algorithm to find a minimal value in a list of numbers using **awk**.

```
awk '{ gsub(/,$/,"",$11) ; print $11 }' example.bed | awk '{m=$1;
    for(i=1;i<=NF;i++) if($i<m) m=$i; print "min␣of␣line", NR":␣",
    m}' FS="," | head -n5
```

```
min of line 1: 322
min of line 2: 125
min of line 3: 19
min of line 4: 57
min of line 5: 57
```

Question 8 (1p)

How would you now sort the records so that the first number shown is the smallest exon size? Again, the answer must provide a sorted list of numbers for each line of the input.

Answer

Although the commands we used in Question 7 are valid for the purpose we wanted to achieve, it involved working exclusively with field 11 and forgetting about the rest of the data file.

We want to work with field 11 within the whole file, so we'll have to rewrite the command a little bit to divide field 11 into an array (`a`), although it does essentially the same. By printing the minimum values in a new field (at the beginning, for instance), we can easily sort the output.

```
awk '{ gsub(/,$/,"",$11); split($11, a, ","); m=$11;
    for(i=1;i<=length(a);i++) if(a[i]<m) m=a[i]; print m, $0}'
    OFS="\t" example.bed | sort -n 2>/dev/null | head -n5
```

```
1 chr3 3628592 3630410 AT3G11530.2 0 - 3628800 3630324 0 5 105,1,52,125,392
   1713,1417,1260,471,0,
1 chr4 15669218 15671194 AT4G32470.2 0 - 15669704 15671095 0 5 193,158,48,1,491
   1783,1514,852,801,0,
2 chr1 10274047 10275539 AT1G29355.1 0 + 10274047 10275539 0 3 2,697,225
   0,346,1267,
2 chr2 14807448 14810164 AT2G35130.1 0 - 14807588 14810164 0 8
   2,185,233,250,158,206,125,757 2714,2245,1930,1590,1333,1048,839,0,
2 chr5 1716870 1719541 AT5G05720.1 0 + 1716870 1719541 0 11
   2,111,115,33,66,282,66,196,83,220,182
   0,117,295,492,690,848,1302,1496,1876,2163,2489,
```

Notice that we use `2>/dev/null` in **sed** to redirect the following error:

```
sort: write failed: 'standard output': Broken pipe
sort: write error
```

This may have something to do with `bash`; `zsh` (the shell I've been using for years) doesn't seem to give this error.

Question 9 (1p)

Now get the 10 largest exons of `chr1` stored in `example.bed`.

Answer

This is very similar to the previous exercise, but we calculate the maximum of the exons (field 11) and sort using this new field. Notice that we declare `m` as `m=asort(a)` instead of `m=$11` as we did for the minimum calculation for the algorithm to work (not sure why, though; it surely has something to do with the internal structure of the constructed array).

```
awk '/chr1/ { gsub(/,$/,"",$11); split($11, a, ","); m=asort(a);
    for(i=1;i<=length(a);i++) if(a[i]>m) m=a[i];  print m, $0}'
    OFS="\t" example.bed | sort -r -n 2>/dev/null | head
```

```
7713 chr1 26488521 26501281 AT1G70320.1 0 - 26488744 26501281 0 15
    33,96,207,7713,318,202,245,251,1108,147,81,226,140,107,326
    12727,12423,12128,4271,3877,3522,3194,2850,1584,1355,1163,848,609,397,0,
5616 chr1 28816640 28822256 AT1G76780.1 0 + 28816640 28822256 0 1 5616 0,
5239 chr1 7560564 7565803 AT1G21580.1 0 - 7560564 7565655 0 1 5239 0,
4755 chr1 7773062 7780586 AT1G22060.1 0 - 7773372 7780586 0 9
    78,201,123,165,4755,156,102,143,587 7446,7092,6884,6609,1641,1178,904,680,0,
4154 chr1 731703 737332 AT1G03080.1 0 - 731793 737332 0 3 100,4154,1038
    5529,1224,0,
4075 chr1 24149542 24154274 AT1G65010.1 0 + 24149542 24154024 0 3 17,196,4075
    0,361,657,
3897 chr1 3333594 3337491 AT1G10170.1 0 - 3333924 3337491 0 1 3897 0,
3882 chr1 20879465 20895393 AT1G55860.1 0 - 20879899 20895393 0 19
    100,68,612,96,207,3762,3882,321,202,245,269,1108,147,81,226,140,107,188,256
    15828,15203,14477,12857,12558,8588,4589,4192,3835,3507,3149,1871,1642,1454,1135,
    916,617,349,0,
3875 chr1 4788558 4794654 AT1G13980.1 0 + 4789586 4794397 0 3 96,864,3875
    0,902,2221,
3757 chr1 28075073 28078830 AT1G74720.1 0 + 28075172 28078418 0 1 3757 0,
```

Question 10 (1p)

Now modify Question 9 script to receive as a parameter the number of exons to search for.

Answer

To read N as a command line argument, we just need to modify the script following the typical bash syntax:

Script 1: Contents of `largest-exons.sh`

```
#!/bin/bash

N=$1

awk '/chr1/ { gsub(/,$/,"",$11); split($11, a, ","); m=asort(a);
    for(i=1;i<=length(a);i++) if(a[i]>m) m=a[i];  print m, $0}'
    OFS="\t" example.bed | sort -r -n 2>/dev/null | head -n$N
```

Notice

The `Title` field in `jan2017articles.csv` is not formatted well and may have commas inside it, making really difficult to work with columns/fields. Thankfully, the field separator is just a single comma (`,`), whilst the commas in the titles are accompanied by a space (`,␣`). Knowing this, we just need to systematically perform a `gsub(",␣", "␣")` substitution before working with this file.

We should have done this as well in questions 3 and 4, but the titles of the matched lines in those questions are properly formatted and don't induce any error.

Question 11 (1p)

Get the first 10 records of `jan2017articles.csv` with largest number of comments from the original `csv` file.

Answer

```
awk '{ gsub(",␣", "␣"); print $0}' FS="," jan2017articles.csv | sort
    -t',' -k5 -r 2>/dev/null | head
```

```
Post date,Content type,Author,Title,Comment count,Path,Tags,Word count
18 Jan 2017,Article,Jeffrey Robert Kaufman,Do I need to provide access to source
    code under the AGPLv3 license?,9,
    /article/17/1/providing-corresponding-source-agplv3-license,Licensing,348
09 Jan 2017,Article,Seth Kenlon,How to get started as an open source
    programmer,8,/article/17/1/how-get-started-open-source-programmer,Getting
    started,1885
23 Jan 2017,Article,Greg Pittman,Python and successive
    approximation,7,/article/17/1/python-and-successive-approximation,"Python
    Programming",347
```

```
13 Jan 2017,Article,Chris Hermansen,"3 open source music players: Aqualung
    Lollypop and GogglesMM",6,/article/17/1/open-source-music-players,Open Music
    column,1606
16 Jan 2017,Article,Seth Kenlon,Getting started with shell
    scripting,6,/article/17/1/getting-started-shell-scripting,Getting started,1990
26 Jan 2017,Article,Ron McFarland,A 5-step plan to encourage your team to make
    changes on your project,5,/open-organization/17/1/escape-the-cave,The Open
    Organization,1160
05 Jan 2017,Article,Jono Bacon,10 steps to innersource in your organization in
    2017,5,/article/17/1/yearbook-10-steps-innersource-your-organization,"2016
    Open Source Yearbook Yearbook Innersource",1753
16 Jan 2017,Article,Don Watkins,Can academic faculty members teach with
    Wikipedia?,5,/article/17/1/Wiki-Education-Foundation,"Education Wikipedia",1032
12 Jan 2017,Article,Scott Nesbitt,3 to-do list managers for the Linux command
    line,5,/article/17/1/task-managers-linux-command-line,"Linux Command line
    Business",780
```

Question 12 (1p)

Modify your previous script to receive a number as a parameter N and then show the top N entries with more comments.

As we said in Question 10, we just need to use an argument for N:

Script 2: Contents of `largest-comments.sh`

```bash
#!/bin/bash

N=$1

awk '{ gsub(", ", " "); print $0}' FS="," jan2017articles.csv | sort
    -t',' -k5 -r 2>/dev/null | head -n$N
```

Question 13 (1p)

Now we are going to create a new `articles.csv` where we get a different output data layout using **awk** tool:

```
INPUT: Post date,Content type,Author,Title,Comment count,Path,Tags,Word count
OUTPUT: Title;Comment count;Word count;Post date
```

Answer

```bash
awk '{ gsub(", ", " "); print $4, $5, $8, $1}' FS="," OFS=";"
    jan2017articles.csv > articles.csv
head -n5 articles.csv
```

```
Title;Comment count;Word count;Post date
Book review: Ours to Hack and to Own;0;660;31 Jan 2017
5 new guides for working with OpenStack;2;419;31 Jan 2017
Be the open source supply chain;1;1668;31 Jan 2017
Developing open leaders;1;768;31 Jan 2017
```

Question 14 (1p)

Now create a new `article2.csv` format where we cut the `Title` text to 10 characters and we get only the last level of the `Path`.

Answer

```
awk '{ gsub(",␣", "␣"); a = substr($4, 1, 10) ; split($6, b, "/");
    print $1, $2, $3, a, $5, b[length(b)], $7, $8 }' FS="," OFS=";"
    jan2017articles.csv > article2.csv
head -n5 article2.csv
```

```
Post date;Content type;Author;Title;Comment count;Path;Tags;Word count
31 Jan 2017;Article;Scott Nesbitt;Book
    revie;0;review-book-ours-to-hack-and-own;Books;660
31 Jan 2017;Article;Jason Baker;5 new guid;2;openstack-tutorials;"OpenStack
    How-tos and tutorials";419
31 Jan 2017;Article;John Mark Walker;Be the
    ope;1;be-open-source-supply-chain;Business;1668
31 Jan 2017;Article;DeLisa Alexander;Developing;1;developing-open-leaders;The
    Open Organization;768
```

References

[1] Genome Browser FAQ: BED format. URL: `https://genome.ucsc.edu/FAQ/FAQformat.html#format1`.