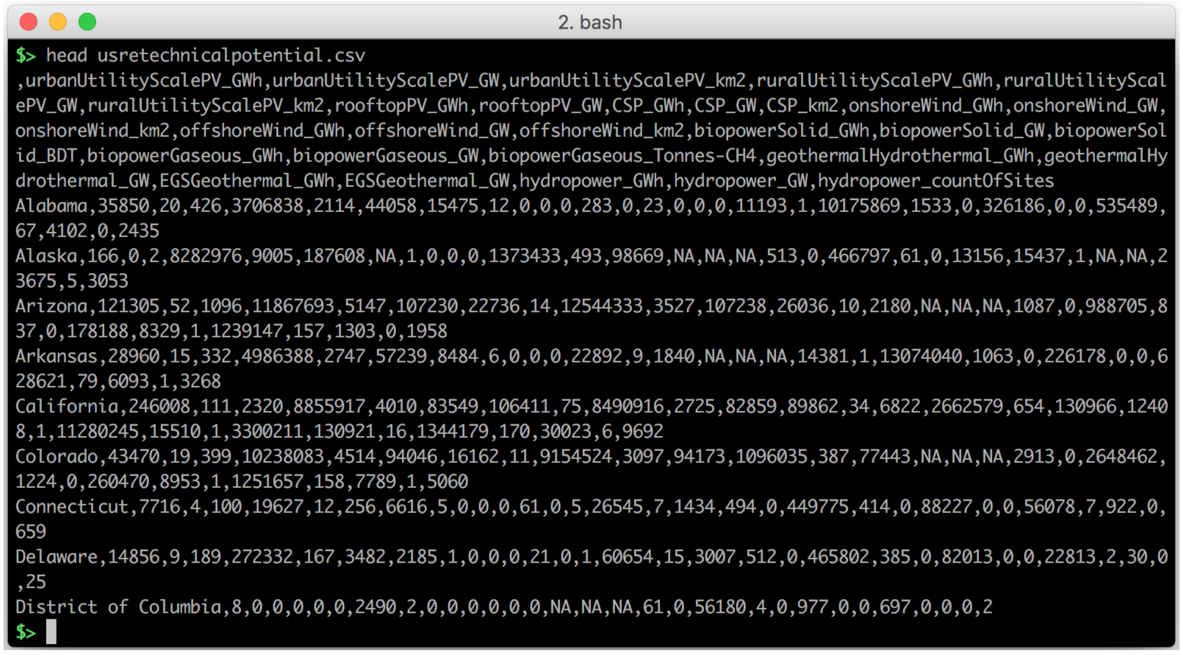
Pretty CSV viewing on the Command Line

BY STEFAAN LIPPENS ON 2016/10/21

TAGGED: CSV, BASH

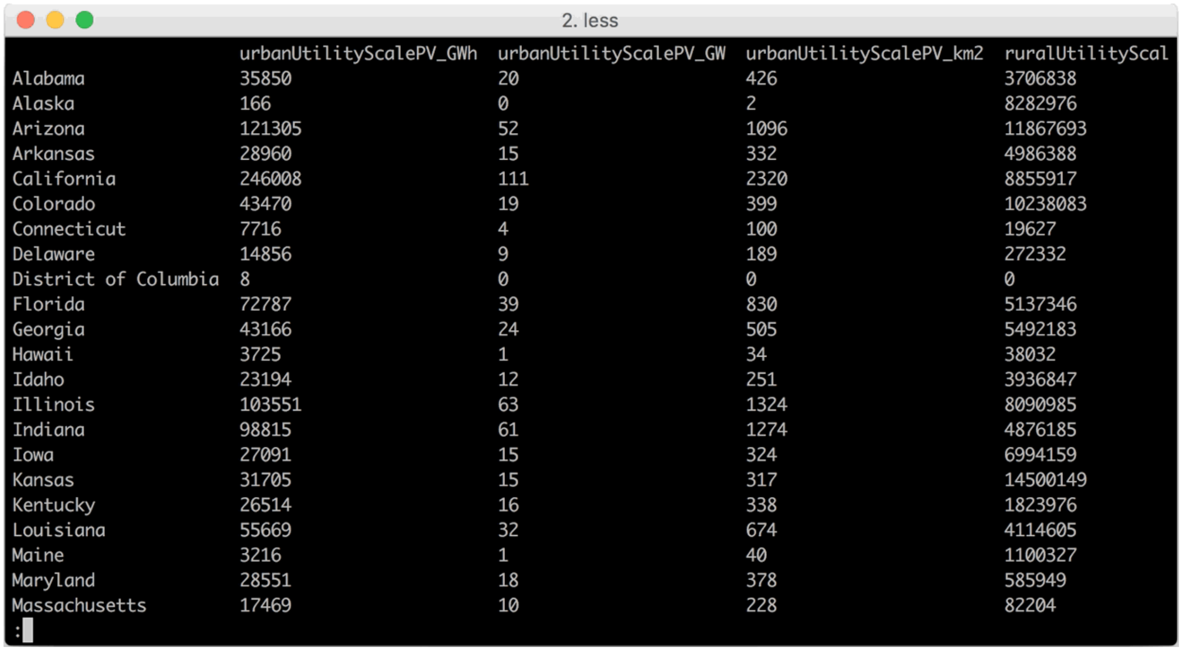
CSV (comma separated values) files are to data formats what FAT32 is to file systems: everybody loves to hate them, but you can't find a more widely supported alternative.

For example, viewing CSV files in a command line environment is typically pretty annoying. You can't make much from this, right?



```
$> head usretechnicalpotential.csv
,urbanUtilityScalePV_GWh,urbanUtilityScalePV_GW,urbanUtilityScalePV_km2,ruralUtilityScalePV_GWh,ruralUtilityScal
ePV_GW,ruralUtilityScalePV_km2,rooftopPV_GWh,rooftopPV_GW,CSP_GWh,CSP_GW,CSP_km2,onshoreWind_GWh,onshoreWind_GW,
onshoreWind_km2,offshoreWind_GWh,offshoreWind_GW,offshoreWind_km2,biopowerSolid_GWh,biopowerSolid_GW,biopowerSol
id_BDT,biopowerGaseous_GWh,biopowerGaseous_GW,biopowerGaseous_Tonnes-CH4,geothermalHydrothermal_GWh,geothermalHy
drothermal_GW,EGSGeothermal_GWh,EGSGeothermal_GW,hydropower_GWh,hydropower_GW,hydropower_countOfSites
Alabama,35850,20,426,3706838,2114,44058,15475,12,0,0,0,283,0,23,0,0,0,11193,1,10175869,1533,0,326186,0,0,535489,
67,4102,0,2435
Alaska,166,0,2,8282976,9005,187608,NA,1,0,0,0,1373433,493,98669,NA,NA,NA,513,0,466797,61,0,13156,15437,1,NA,NA,2
3675,5,3053
Arizona,121305,52,1096,11867693,5147,107230,22736,14,12544333,3527,107238,26036,10,2180,NA,NA,NA,1087,0,988705,8
37,0,178188,8329,1,1239147,157,1303,0,1958
Arkansas,28960,15,332,4986388,2747,57239,8484,6,0,0,0,22892,9,1840,NA,NA,NA,14381,1,13074040,1063,0,226178,0,0,6
28621,79,6093,1,3268
California,246008,111,2320,8855917,4010,83549,106411,75,8490916,2725,82859,89862,34,6822,2662579,654,130966,1240
8,1,11280245,15510,1,3300211,130921,16,1344179,170,30023,6,9692
Colorado,43470,19,399,10238083,4514,94046,16162,11,9154524,3097,94173,1096035,387,77443,NA,NA,NA,2913,0,2648462,
1224,0,260470,8953,1,1251657,158,7789,1,5060
Connecticut,7716,4,100,19627,12,256,6616,5,0,0,0,61,0,5,26545,7,1434,494,0,449775,414,0,88227,0,0,56078,7,922,0,
659
Delaware,14856,9,189,272332,167,3482,2185,1,0,0,0,21,0,1,60654,15,3007,512,0,465802,385,0,82013,0,0,22813,2,30,0
,25
District of Columbia,8,0,0,0,0,0,2490,2,0,0,0,0,0,0,NA,NA,NA,61,0,56180,4,0,977,0,0,697,0,0,0,2
$> 
```

However, if you weld a couple of common command line tools together, you can create a handy viewer that make the data look like this:



```
                      urbanUtilityScalePV_GWh  urbanUtilityScalePV_GW  urbanUtilityScalePV_km2  ruralUtilityScal
Alabama               35850                    20                      426                      3706838
Alaska                166                      0                       2                        8282976
Arizona               121305                   52                      1096                     11867693
Arkansas              28960                    15                      332                      4986388
California            246008                   111                     2320                     8855917
Colorado              43470                    19                      399                      10238083
Connecticut           7716                     4                       100                      19627
Delaware              14856                    9                       189                      272332
District of Columbia  8                        0                       0                        0
Florida               72787                    39                      830                      5137346
Georgia               43166                    24                      505                      5492183
Hawaii                3725                     1                       34                       38032
Idaho                 23194                    12                      251                      3936847
Illinois              103551                   63                      1324                     8090985
Indiana               98815                    61                      1274                     4876185
Iowa                  27091                    15                      324                      6994159
Kansas                31705                    15                      317                      14500149
Kentucky              26514                    16                      338                      1823976
Louisiana             55669                    32                      674                      4114605
Maine                 3216                     1                       40                       1100327
Maryland              28551                    18                      378                      585949
Massachusetts         17469                    10                      228                      82204
:
```

The basics

There is this well hidden command line tool called "`column`" that allows you to align the data nicely in properly sized columns. Combine this with a pager like `less` and we have a nice prototype already

```
cat data.csv | column -t -s, | less -S
```

One problem with this is that `column` ignores/merges empty cells in your data, which ruins the whole point of aligning all together. On Debian/Ubuntu, `column` provides an option `-n` to disable this behavior, but for other platforms (like with the BSD flavor of `column` on the Mac), we need some additional trickery. A simple solution is just adding a space before each comma:

```
cat data.csv | sed 's/,/ ,/g' | column -t -s, | less -S
```

Or, if you want to avoid wasting too much horizontal space, you can add a space only to the empty cells as follows:

```
cat data.csv | perl -pe 's/((?<=,)|(?<=^)),/ ,/g;' | column -t -s, | less -S
```

Shortcuts

Time to create some shortcuts and put this in, for example, your `.bashrc`, `.bash_aliases` or whatever other customization options your favorite shell provides. I'll just cover bash here, because that's the shell I currently use most.

In the end we'll have a tool `pretty_csv` which can be used in different ways:

- `pretty_csv data.csv`
- `pretty_csv < data.csv`
- `sort data.csv | pretty_csv` (to illustrate that the input doesn't necessary have to be a file, you can also pipe the output of another process to it)

For Debian/Ubuntu

On Debian/Ubuntu systems I just put this in my `.bashrc` (note some additional `less` options, roughly based on how `git log` works):

```
function pretty_csv {
    column -t -s, -n "$@" | less -F -S -X -K
}
```

For other platforms

For non-Debian systems we have to add preprocessing of empty cells:

```
function pretty_csv {
    perl -pe 's/((?<=,)|(?<=^)),/ ,/g;' "$@" | column -t -s, | less  -F -S -X -K
}
```

Conflict with iTerm2 on ~~Mac OS X~~ macOS

On my Mac I use iTerm2 and I noticed that its shell integration conflicts in some weird ways with `less` in the above bash function if I apply it through a pipe (`cat data.csv | pretty_csv`).

As workaround I use a bash script instead of a bash function. For example, create a file `~/.bash.d/pretty_csv.sh`, containing:

```
#!/bin/bash
perl -pe 's/((?<=,)|(?<=^)),/ ,/g;' "$@" | column -t -s, | exec less  -F -S -X -K
```

make it executable (`chmod u+x ~/.bash.d/pretty_csv.sh`) and create a bash alias for it (e.g in `.bashrc` or `.bash_aliases`)

```
alias pretty_csv='~/.bash.d/pretty_csv.sh'
```

TSV: tab separated values

I regularly also have to work with TSV files, where the columns are separated by the *tab* character. The tricky part here is passing this special character correctly to the parts of the pipeline.

For Debian/Ubuntu:

```
function pretty_tsv {
    column -t -s $'\t' -n "$@" | less -F -S -X -K
}
```

For non-Debian systems:

```
function pretty_tsv {
    perl -pe 's/((?<=\t)|(?<=^))\t/ \t/g;' "$@" | column -t -s $'\t' | less  -F -S -X -K
}
```

As a bash script (`pretty_tsv.sh`):

```
#!/bin/bash
perl -pe 's/((?<=\t)|(?<=^))\t/ \t/g;' "$@" | column -t -s $'\t' | exec less  -F -S -X -K
```

Bye

Now you can enjoy the warm cosy feeling of browsing pretty, shiny CSV and TSV files in your terminal.

I've put the code and scripts on github too.