

Linux y HPC para big data

Alex Di Genova

09/04/2024

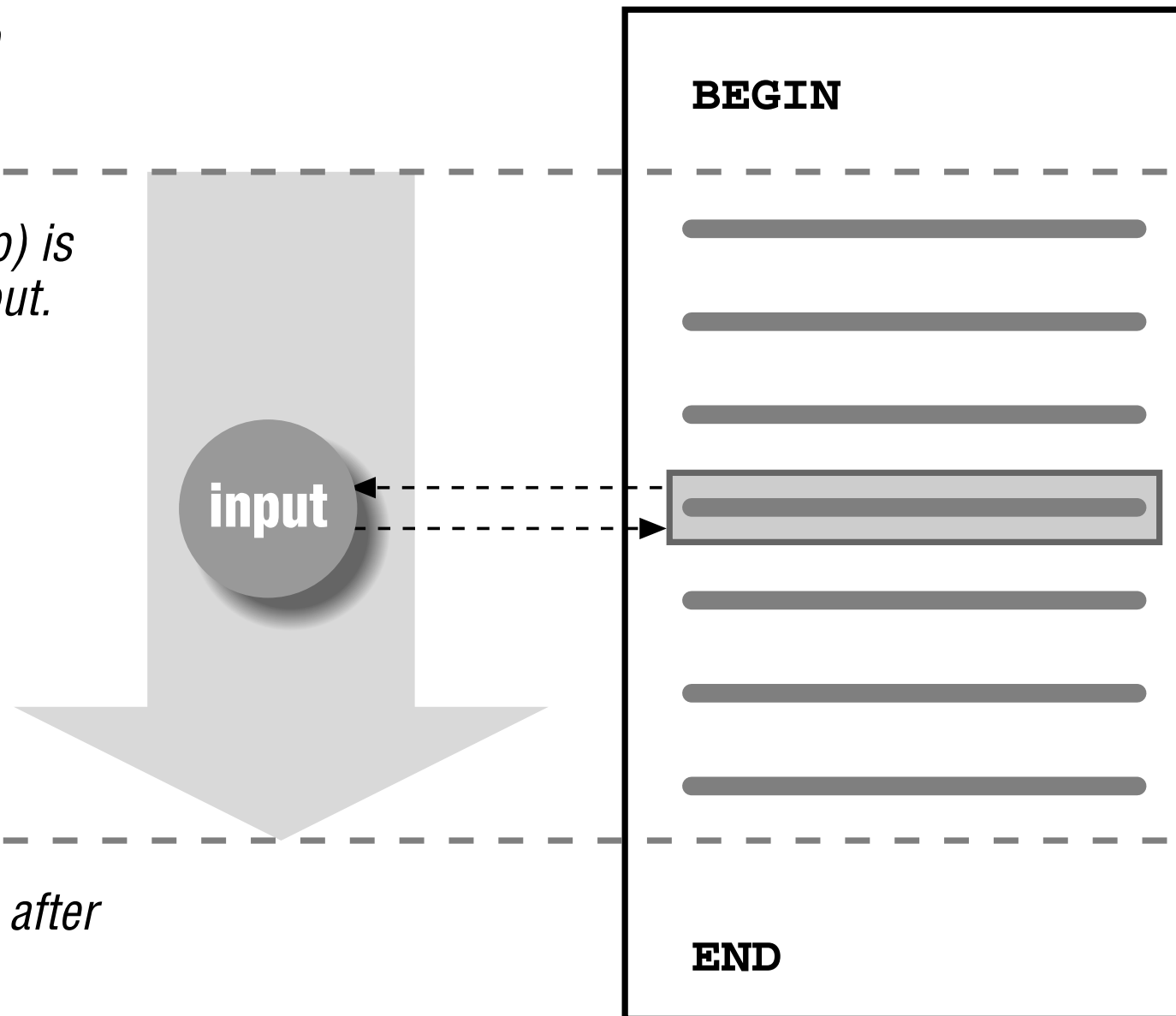
Linux

Awk

1 *1st routine is executed once before any input is read.*

2 *2nd routine (main input loop) is executed for each line of input.*

3 *3rd routine is executed once after all input is read.*



Linux

Awk

- *if (expression) action1; [else action2]*
expr ? action1 : action2
grade = (avg >= 65) ? "Pass" : "Fail"
- array[subscript] = value*
for (variable in array)
do something with array[variable]

while (condition)

action

for (set_counter ; test_counter ; increment_counter)

action

Awk Function	Description
<code>cos(<i>x</i>)</code>	Returns cosine of <i>x</i> (<i>x</i> is in radians).
<code>exp(<i>x</i>)</code>	Returns <i>e</i> to the power <i>x</i> .
<code>int(<i>x</i>)</code>	Returns truncated value of <i>x</i> .
<code>log(<i>x</i>)</code>	Returns natural logarithm (base- <i>e</i>) of <i>x</i> .
<code>sin(<i>x</i>)</code>	Returns sine of <i>x</i> (<i>x</i> is in radians).
<code>sqrt(<i>x</i>)</code>	Returns square root of <i>x</i> .
<code>atan2(<i>y</i>,<i>x</i>)</code>	Returns arctangent of <i>y/x</i> in the range $-\pi$ to π .
<code>rand()</code>	Returns pseudo-random number <i>r</i> , where $0 \leq r < 1$.
<code>srand(<i>x</i>)</code>	Establishes new seed for <code>rand()</code> . If no seed is specified, uses time of day. Returns the old seed.

Linux

Awk

Awk Function	Description
<code>gsub(<i>r,s,t</i>)</code>	Globally substitutes <i>s</i> for each match of the regular expression <i>r</i> in the string <i>t</i> . Returns the number of substitutions. If <i>t</i> is not supplied, defaults to <code>\$0</code> .
<code>index(<i>s,t</i>)</code>	Returns position of substring <i>t</i> in string <i>s</i> or zero if not present.
<code>length(<i>s</i>)</code>	Returns length of string <i>s</i> or length of <code>\$0</code> if no string is supplied.
<code>match(<i>s,r</i>)</code>	Returns either the position in <i>s</i> where the regular expression <i>r</i> begins, or 0 if no occurrences are found. Sets the values of RSTART and RLENGTH .
<code>split(<i>s,a,sep</i>)</code>	Parses string <i>s</i> into elements of array <i>a</i> using field separator <i>sep</i> ; returns number of elements. If <i>sep</i> is not supplied, FS is used. Array splitting works the same way as field splitting.
<code>sprintf("<i>fmt</i>",<i>expr</i>)</code>	Uses printf format specification for expr .
<code>sub(<i>r,s,t</i>)</code>	Substitutes <i>s</i> for first match of the regular expression <i>r</i> in the string <i>t</i> . Returns 1 if successful; 0 otherwise. If <i>t</i> is not supplied, defaults to <code>\$0</code> .
<code>substr(<i>s,p,n</i>)</code>	Returns substring of string <i>s</i> at beginning position <i>p</i> up to a maximum length of <i>n</i> . If <i>n</i> is not supplied, the rest of the string from <i>p</i> is used.
<code>tolower(<i>s</i>)</code>	Translates all uppercase characters in string <i>s</i> to lowercase and returns the new string.
<code>toupper(<i>s</i>)</code>	Translates all lowercase characters in string <i>s</i> to uppercase and returns the new string.

Linux

Sed

- Substitutions
 - /Pattern/replacement/flags
 - Flags : n <replace the n-matchin pattern>,g<global>,l <insensitive case>,p <print pattern>
- Transform
 - s/abc/xyz/
 - y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/
- Regular expressions with Sed
 - sed 's/^(.*\):\(.*\)-\(.*\)^3:\2:\1/' test.txt

1:101-201	201:101:1
2:102-202	202:102:2
3:103-203	203:103:3
4:104-204	204:104:4
5:105-205	205:105:5
6:106-206	206:106:6

Linux

Regular expressions

Basic Syntax

- `/.../`: Start and end
- `|`: Alternation
- `()`: Grouping

Groups and Ranges

- `.`: Any character except newline (`\n`)
- `(alb)`: a or b
- `(...)`: Group
- `(?:...)`: Passive (non-capturing) group
- `[abc]`: a, b or c
- `[^abc]`: Not a, b or c
- `[a-z]`: Letters from a to z
- `[A-Z]`: Uppercase letters from A to Z
- `[0-9]`: Digits from 0 to 9

Position Matching

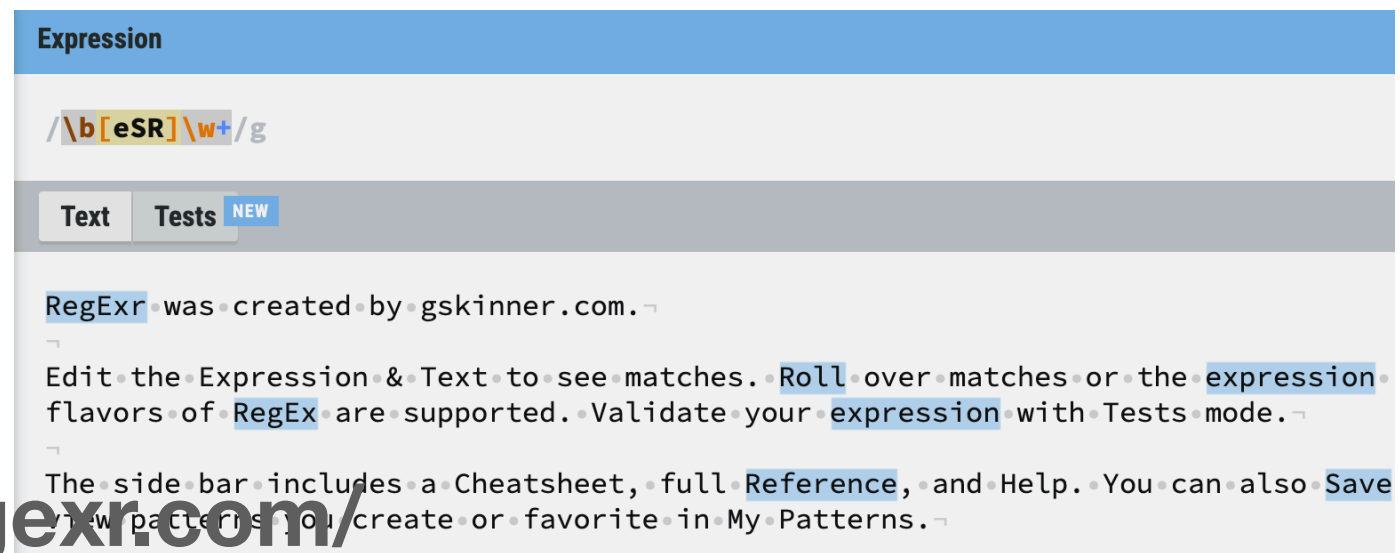
- `^`: Start of string or start of line
- `$`: End of string or end of line
- `\b`: Word boundary
- `\B`: Not word boundary

Character Classes

- `\s`: Whitespace
- `\S`: Not whitespace
- `\w`: Word
- `\W`: Not word
- `\d`: Digit
- `\D`: Not digit
- `\x`: Hexadecimal digit
- `\O`: Octal digit

Quantifiers

- `*`: 0 or more
- `+`: 1 or more
- `?`: 0 or 1
- `{3}`: Exactly 3
- `{3,}`: 3 or more
- `{3,5}`: 3, 4 or 5



<https://regexr.com/>

Awk

Operators

Operator	Effect
<i>lvalue</i> += <i>increment</i>	Add <i>increment</i> to the value of <i>lvalue</i> .
<i>lvalue</i> -= <i>decrement</i>	Subtract <i>decrement</i> from the value of <i>lvalue</i> .
<i>lvalue</i> *= <i>coefficient</i>	Multiply the value of <i>lvalue</i> by <i>coefficient</i> .
<i>lvalue</i> /= <i>divisor</i>	Divide the value of <i>lvalue</i> by <i>divisor</i> .
<i>lvalue</i> %= <i>modulus</i>	Set <i>lvalue</i> to its remainder by <i>modulus</i> .
<i>lvalue</i> ^= <i>power</i>	Raise <i>lvalue</i> to the power <i>power</i> .
<i>lvalue</i> **= <i>power</i>	Raise <i>lvalue</i> to the power <i>power</i> . (c.e.)

Awk

Operators

Expression	Result
$x < y$	True if x is less than y
$x \leq y$	True if x is less than or equal to y
$x > y$	True if x is greater than y
$x \geq y$	True if x is greater than or equal to y
$x == y$	True if x is equal to y
$x != y$	True if x is not equal to y
$x \sim y$	True if the string x matches the regexp denoted by y
$x !\sim y$	True if the string x does not match the regexp denoted by y
$subscript \text{ in } array$	True if the array $array$ has an element with the subscript $subscript$

Awk & sed

Problems

- 1.Count the frequency of words in a text file.
- 2.Change a hello for bye in the 10-20 lines of a file.
- 3.Merge two files by a common field.
- 4.Create a table combining one or more fields from several files.
- 5.Count 100 most frequent 15-mer of a fasta sequence file.
- 6.Transpose of a numeric matrix.

Awk and Sed

Solutions

1. **awk** '{for(i=1;i<NF;i++){a[\$i]+=1;}}END{for(i in a){print i" "a[i]}}' matrix.txt | sort -nr -k2,2

2. seq 1 100 | sed -n '20,30 p'

3. todo

4. todo

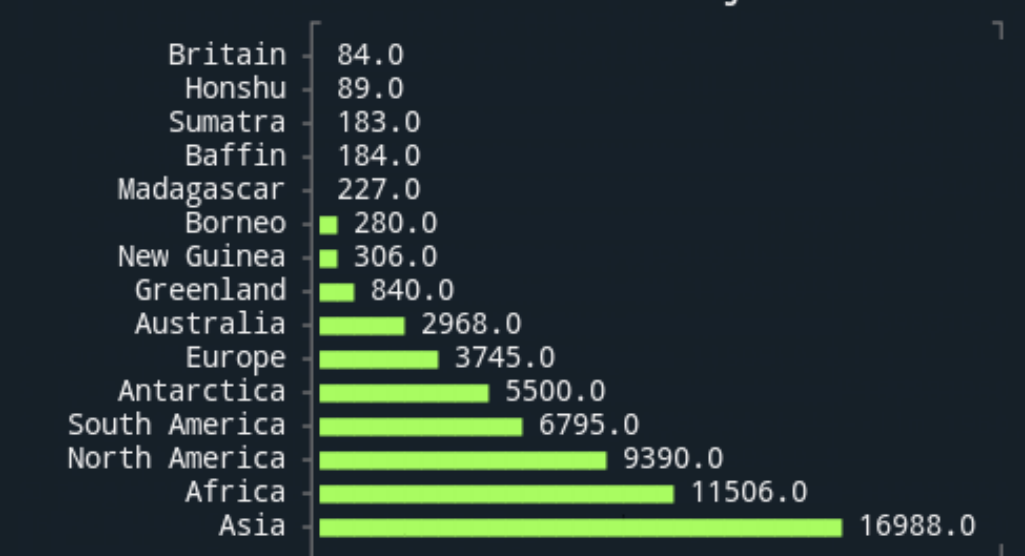
5. **sed** 's/.\{15\}/&\n/g' | **awk** '{if(\$1 ~/[ACTG]/){a[\$1]++}}END{for(i in a){print i" "a[i]}}' | sort -nr -k2,2 | head -n 30

6. **awk** 'NR == 1 {n=NF; for(l=0;i<NF;i++){row[i]=\$i} next;}{if(NF > n){n=NF} for(i=1; i<NF; i++){row[i]=row[i]" "\$i}}END{for(i=1;i<=n; i++){print row[i]}}' matrix.txt

XSV, Youplot and csvtk

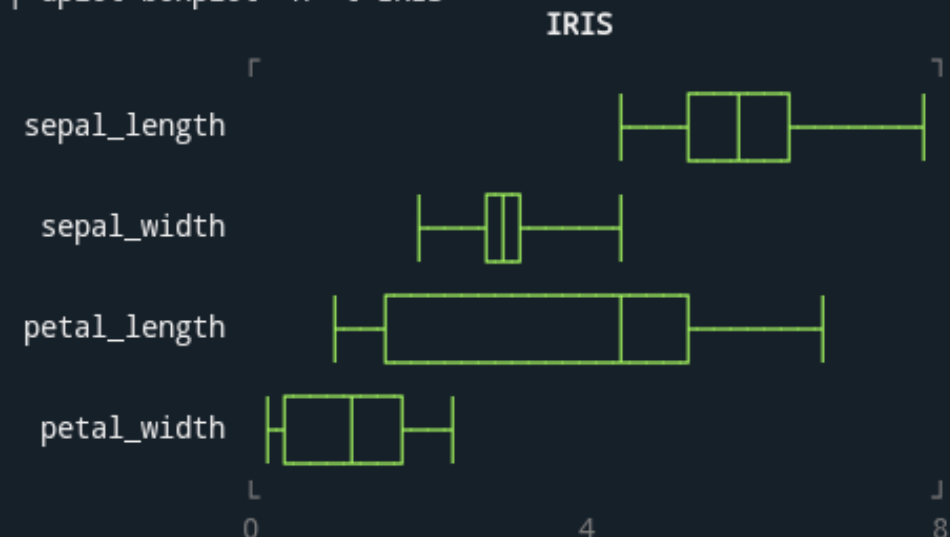
- **xsv** is a command line program for indexing, slicing, analyzing, splitting and joining CSV files.
 - <https://github.com/BurntSushi/xsv>
- **YouPlot**
 - command line tool that draws plots on the terminal.
- **Csvtk** is convenient for rapid data investigation and easily integrated into analysis pipelines.
 - <https://github.com/shenwei356/csvtk>

```
$ curl -sL https://git.io/ISLANDScsv \  
> | sort -nk2 -t, \  
> | tail -n15 \  
> | uplot bar -d, -t "Areas of the World's Major Landmasses"  
Areas of the World's Major Landmasses
```



Landmass	Area (km²)
Britain	84.0
Honshu	89.0
Sumatra	183.0
Baffin	184.0
Madagascar	227.0
Borneo	280.0
New Guinea	306.0
Greenland	840.0
Australia	2968.0
Europe	3745.0
Antarctica	5500.0
South America	6795.0
North America	9390.0
Africa	11506.0
Asia	16988.0

```
$ curl -sL https://git.io/IRISrstsv \  
> | cut -f1-4 \  
> | uplot boxplot -H -t IRIS
```



```
$
```