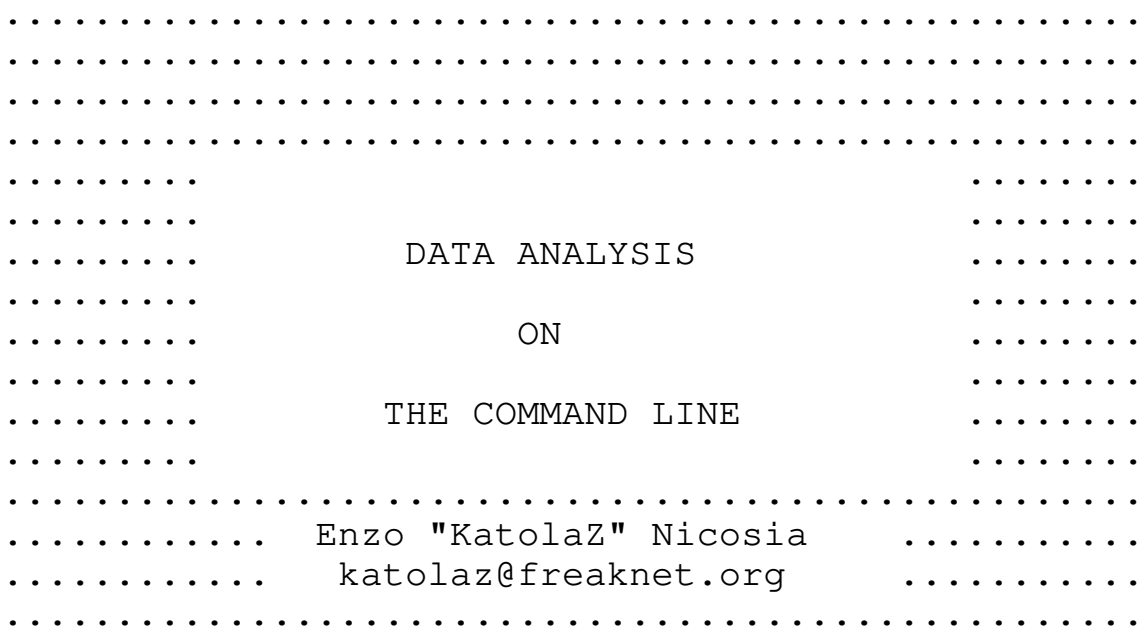
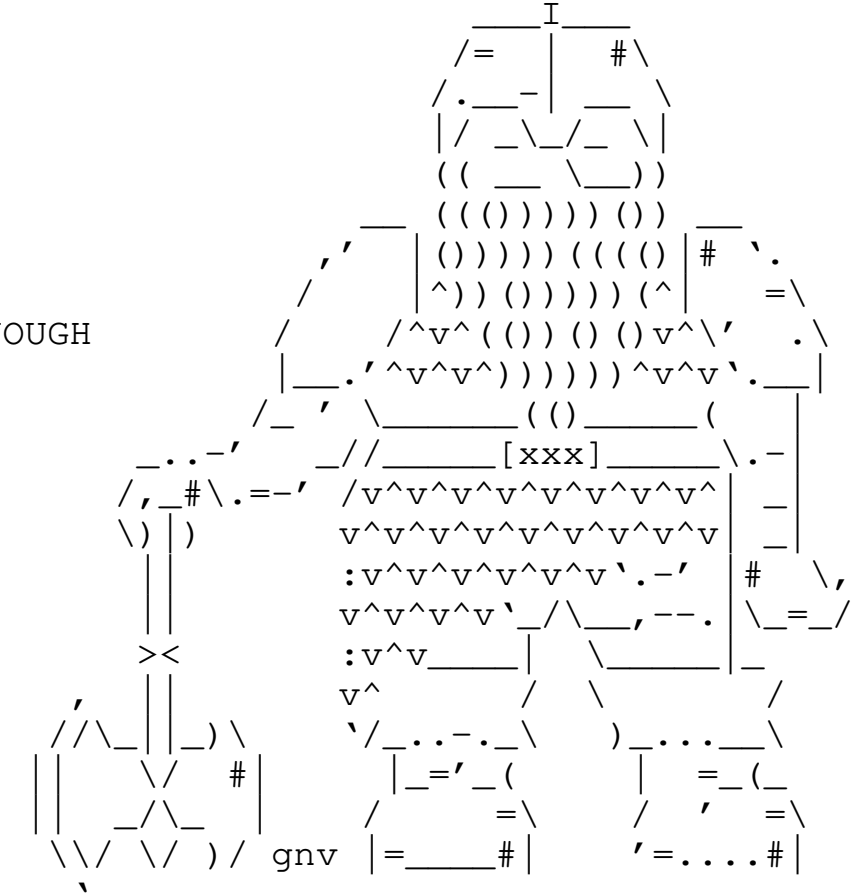
=


DATA ANALYSIS
ON
THE COMMAND LINE
Enzo "KatolaZ" Nicosia
katolaz@freaknet.org

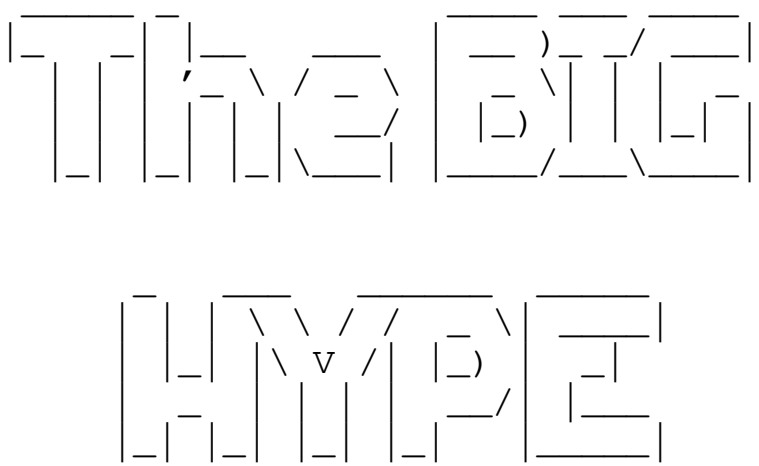
=

NO DATA IS BIG ENOUGH

IF YOUR BEARD IS GREY ENOUGH



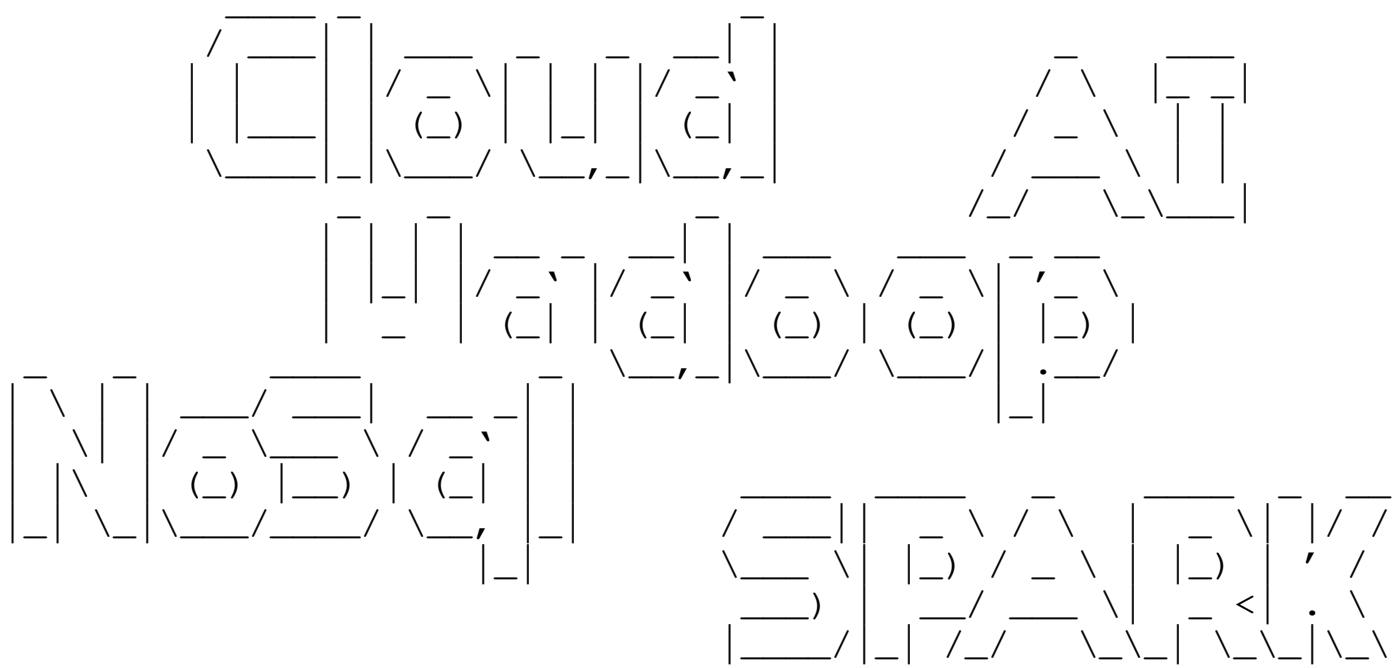
=



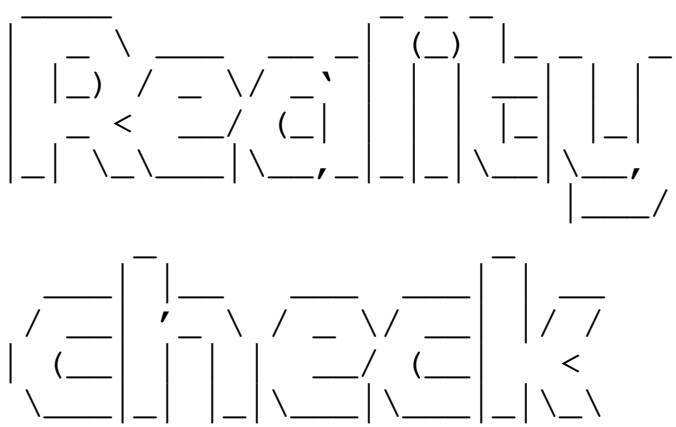
=


OH NO! BIG DATA IS HERE!
WE WILL ALL DIE!!!!
Jym Dyer
jgs

=

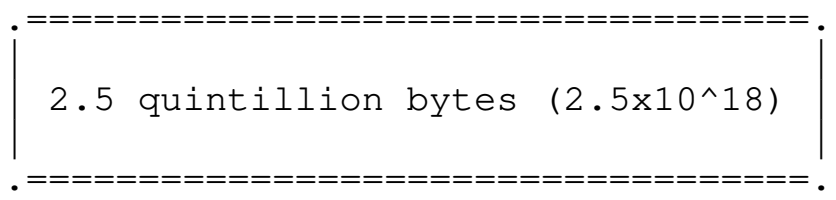

Claude API
Node.js
SDK

=



=

- Data produced each day:



which is about one Sextillion bytes per year (0.9x10^21)

=

BUT AT THE SAME TIME:
==========

- Not all this data is accessible/usable/USEFUL

- Computational power has increased at a similar pace

- We waste 90% of CPU time on layers upon layers upon layers upon layer of "abstraction" (in the end just to watch facebook feeds...)

=

Example 0: Shell-script vs Hadoop (235-to-1)

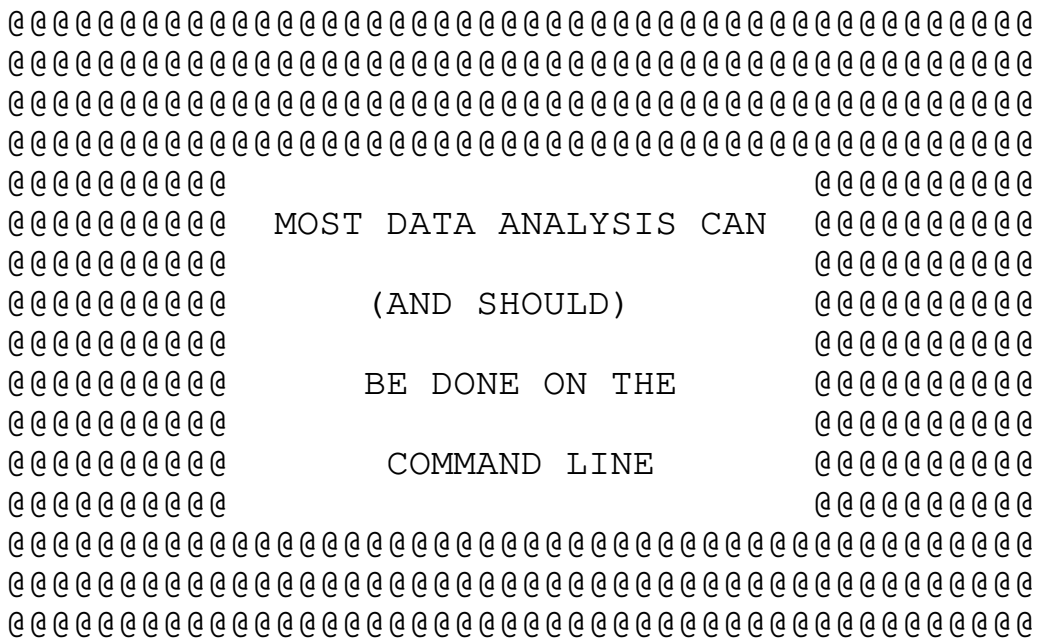
- 1.75 GB of data (2 million chess games)
- Task: compute some statistics
- Solution 1 ==> use hadoop map/reduce

  several machines ==> 26 MINUTES
- Solution 2 ==> `find ... | xargs ... | awk ...`

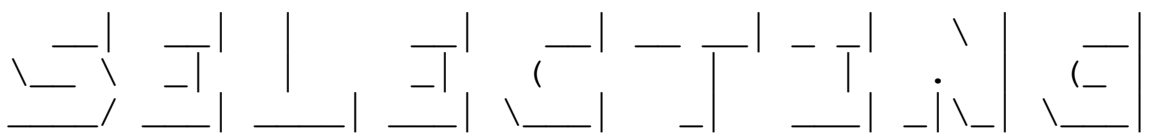
  a single CPU ==> 12 SECONDS

https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html

=


MOST DATA ANALYSIS CAN
(AND SHOULD)
BE DONE ON THE
COMMAND LINE

=



(decide what you need)

=

Example 1: How many packages available?

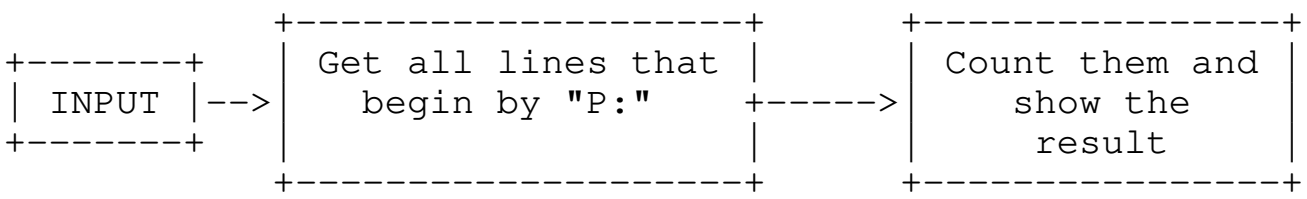
Alpine Linux keeps an index of available packages in APKINDEX

```
     +-------------------------------------------------------------------------+
     | C:Q1EPp2wfY2W6JurBN2CuY+6fS1fGI=                                        |
---> | P:tftp-hpa                                                              |
     | V:5.2-r2                                                                |
     | A:x86_64                                                                |
     | S:30831                                                                 |
     | I:114688                                                                |
     | T:Official tftp server                                                  |
     | U:https://www.kernel.org/pub/software/network/tftp/                     |
     | L:BSD                                                                   |
     | o:tftp-hpa                                                              |
     | m:Natanael Copa <ncopa@alpinelinux.org>                                 |
     | t:1557154288                                                            |
     | c:730cdcef6901750f4029d4c3b8639ce02ee3ead1                              |
     | D:so:libc.musl-x86_64.so.1                                              |
     | p:cmd:in.tftpd cmd:tftp                                                 |
     |                                                                         |
     +-------------------------------------------------------------------------+
```

=

Example 1: How many packages available?

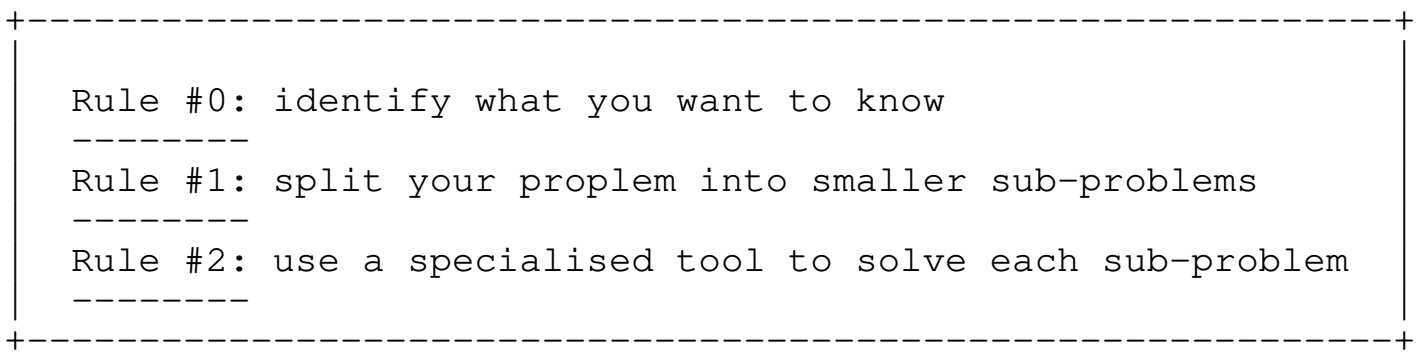
Data flow:



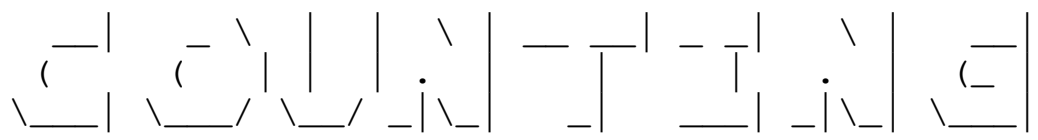
Which translates to:

```
$ egrep "^P:" APKINDEX  |   wc -l
10335
```

=


Rule #0: identify what you want to know
Rule #1: split your proplem into smaller sub-problems
Rule #2: use a specialised tool to solve each sub-problem

=



=

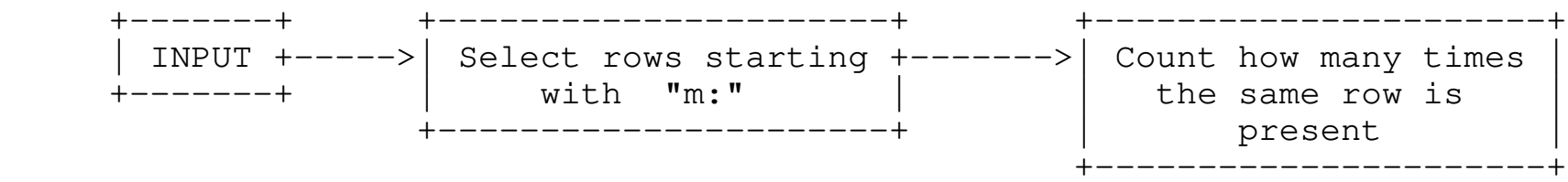
Example 2: How many packagers in Alpine?

```
    +-------------------------------------------------------------------+
    | C:Q1EPp2wfY2W6JurBN2CuY+6fS1fGI=                                  |
    | P:tftp-hpa                                                        |
    | V:5.2-r2                                                          |
    | A:x86_64                                                          |
    | S:30831                                                           |
    | I:114688                                                          |
    | T:Official tftp server                                            |
    | U:https://www.kernel.org/pub/software/network/tftp/               |
    | L:BSD                                                             |
    | o:tftp-hpa                                                        |
--->| m:Natanael Copa <ncopa@alpinelinux.org>                           |
    | t:1557154288                                                      |
    | c:730cdcef6901750f4029d4c3b8639ce02ee3ead1                        |
    | D:so:libc.musl-x86_64.so.1                                        |
    | p:cmd:in.tftpd cmd:tftp                                           |
    |                                                                   |
    +-------------------------------------------------------------------+
```

=

Example 2: How many packagers in Alpine?

Data Flow:



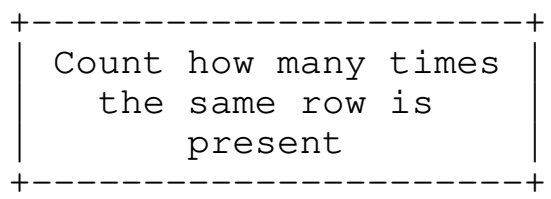
Which translates to:

```
egrep "^m:" APKINDEX    |   ???
```

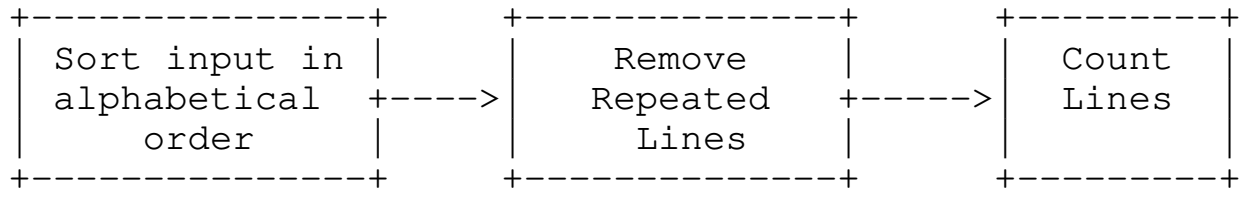
=

Example 2: How many packagers in Alpine?

At a closer look:



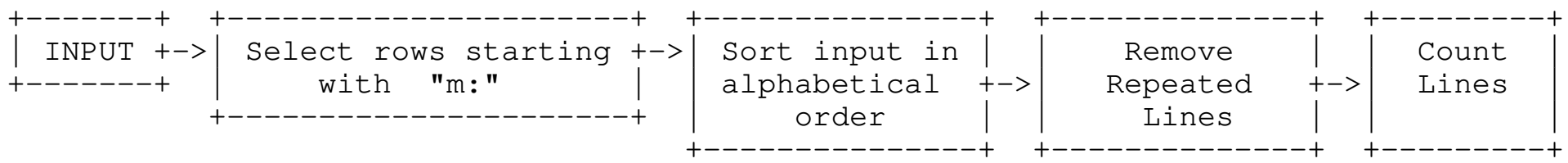
Becomes:



=

Example 2: How many packagers in Alpine?

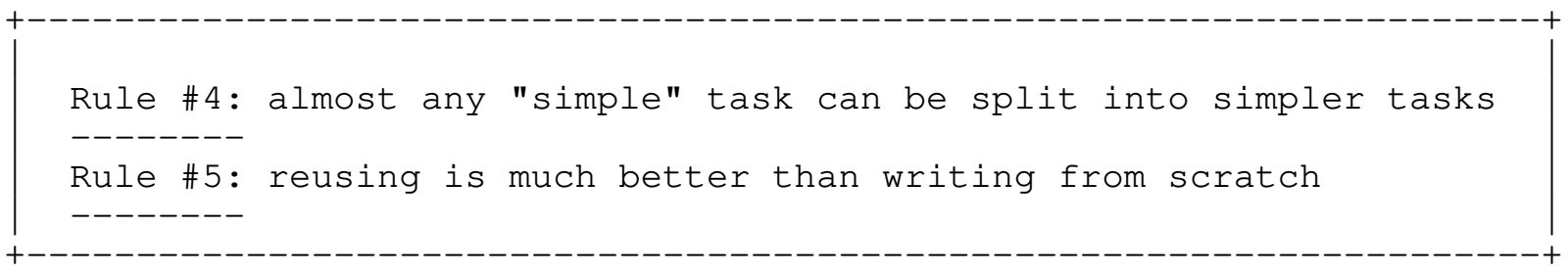
Amended data flow:



Which translates to:

```
$ egrep "^m:" APKINDEX  |       sort        |     uniq       |    wc -l
181
```

=


Rule #4: almost any "simple" task can be split into simpler tasks
Rule #5: reusing is much better than writing from scratch

=

(focus on important stuff)

=

Are you sure you are counting it right?
==========

```
$ egrep "^m:" APKINDEX | sort | uniq | tail -n +25 | head -15
+-------------------------------------------------+
| m:Breno Leitao <breno.leitao@gmail.com>         |
| m:Cameron Banta <cbanta@gmail.com>              |
| m:Camille Scholtz <onodera@openmailbox.org>     |
!!! | m:Carlo Landmeter <clandmeter@alpinelinux.org>  |
!!! | m:Carlo Landmeter <clandmeter@gmail.com>        |
| m:Chloe Kudryavtsev <toast@toastin.space>       |
| m:Christian Kampka <christian@kampka.net>       |
| m:Christine Dodrill <me@christine.website>      |
| m:Clayton Craft <clayton@craftyguy.net>         |
%%% | m:Corey Oliver <corey.jon.oliver@gmail.com>      |
%%% | m:Corey Oliver <coreyjonoliver@gmail.com>        |
| m:CÃ¡g <ca6c@bitmessage.ch>                    |
| m:Dan Theisen <djt@hxx.in>                      |
| m:Daniel Isaksen <d@duniel.no>                  |
| m:Daniel Sabogal <dsabogalcc@gmail.com>         |
+-------------------------------------------------+
```

=

Are you sure you are counting it right?

We actually would like either of the following:

    m:Carlo Landmeter <clandmeter@alpinelinux.org>
    m:Carlo Landmeter <clandmeter@gmail.com>

to be counted as the single user "Carlo Landmeter". The same for:

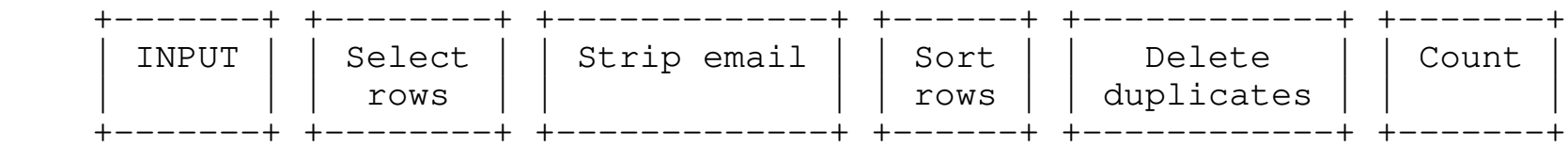
    m:Corey Oliver <corey.jon.oliver@gmail.com>
    m:Corey Oliver <coreyjonoliver@gmail.com>

which should be counted as the single user "Corey Oliver".

=

Are you sure you are counting it right?
==========

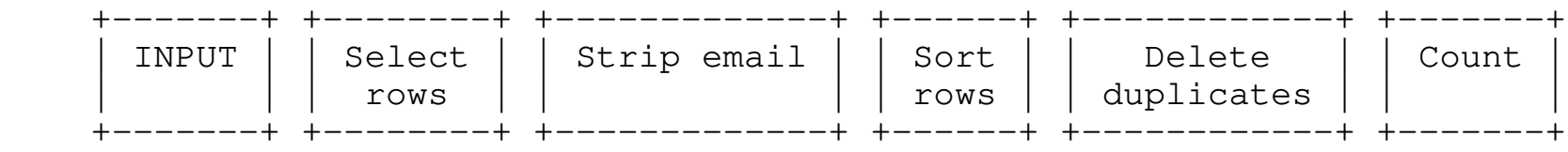
Data flow:



=

Are you sure you are counting it right?

Data flow:



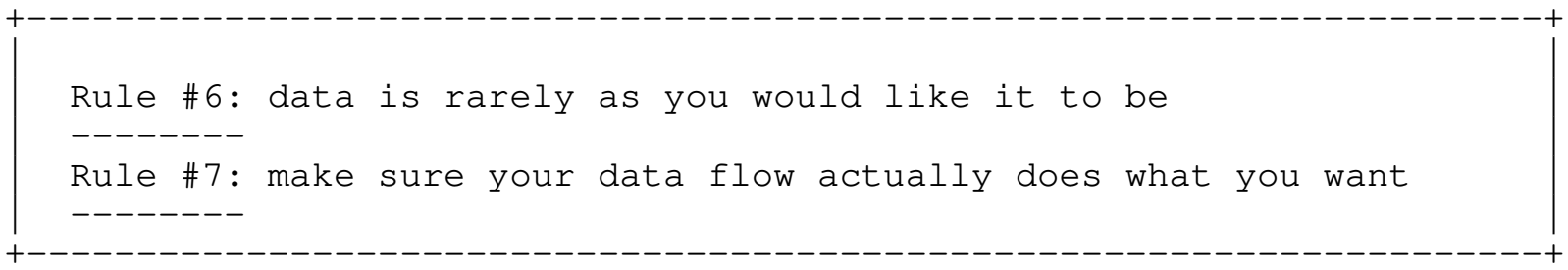
Which becomes:

```
$ egrep "^m:" APKINDEX | cut -d"<" -f1 | sort | uniq | wc -l
152
```

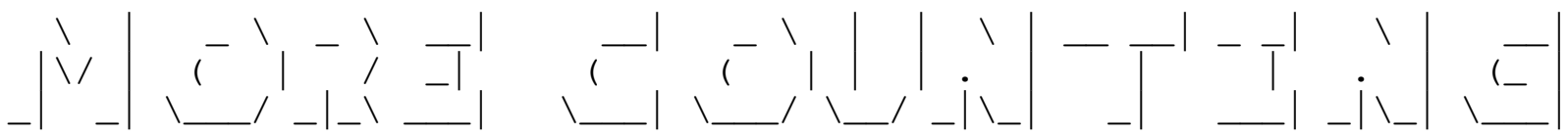
and is different from:

```
$ egrep "^m:" APKINDEX | sort | uniq | wc -l
181
```

=


Rule #6: data is rarely as you would like it to be
Rule #7: make sure your data flow actually does what you want

=



=

Example 3: Who is the most prolific packager?

```
$ egrep "^m:" APKINDEX | cut -d"<" -f1 |  sort   | head -15
m:7heo
m:7heo
m:7heo
m:7heo
m:7heo
m:7heo
m:7heo
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
m:A. Wilcox
```

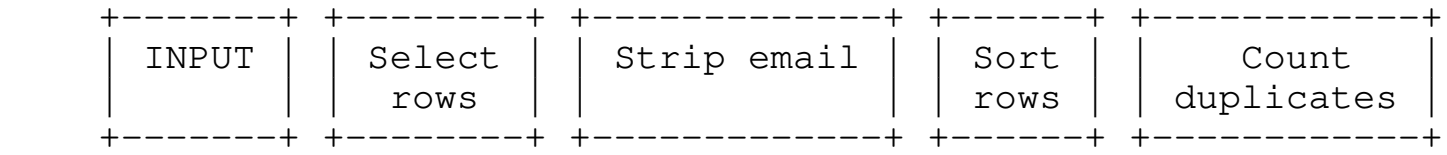
=

Example 3: Who is the most prolific packager?

Counting repeated (sorted) lines is easy:

```
... | sort | uniq -c
```

New data flow:

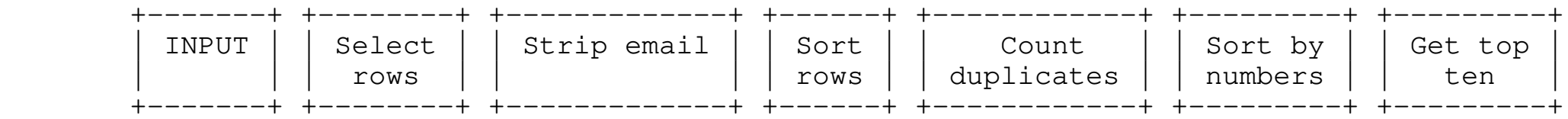



```
$ egrep "^m:" APKINDEX | cut -d"<" -f1 |  sort  |   uniq -c
      2 m:lemon
     32 m:prspkt
      5 m:stef
     57 m:tcely
      2 m:wener
      .........
```

=

Example 3: Who is the most prolific packager?

We can easily rank the packagers by total number of packages!




```
$ egrep "^m:" APKINDEX | cut -d"<" -f1 |  sort  |    uniq -c   | sort -rn  | head -10
  4081 m:Natanael Copa
  1104 m:Francesco Colista
   484 m:Jakub Jirutka
   478 m:Leonardo Arena
   392 m:Timo TerÃ¤s
   271 m:Carlo Landmeter
   262 m:Stuart Cardall
   253 m:Fabian Affolter
   242 m:Valery Kartel
   145 m:Rasmus Thomsen
```

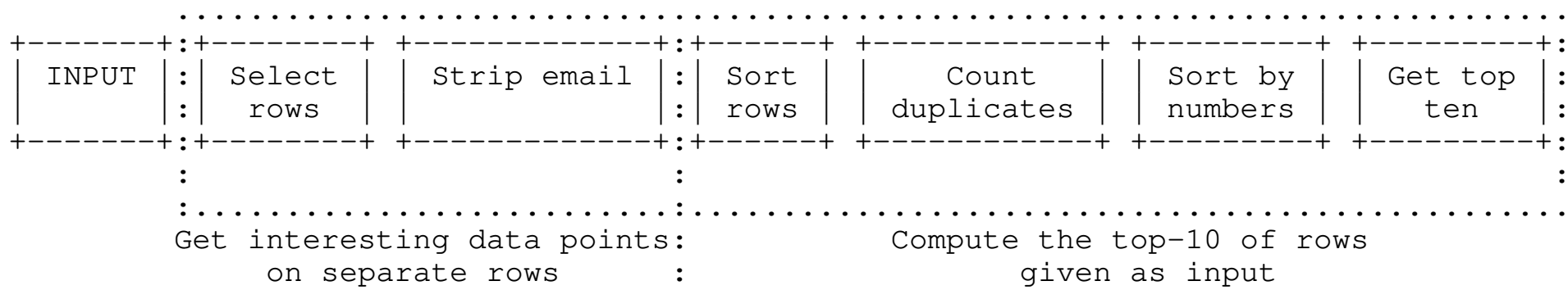
=


ZOOMING OUT

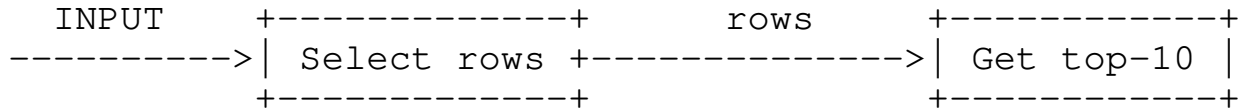
=

Making new tools out of simple ones

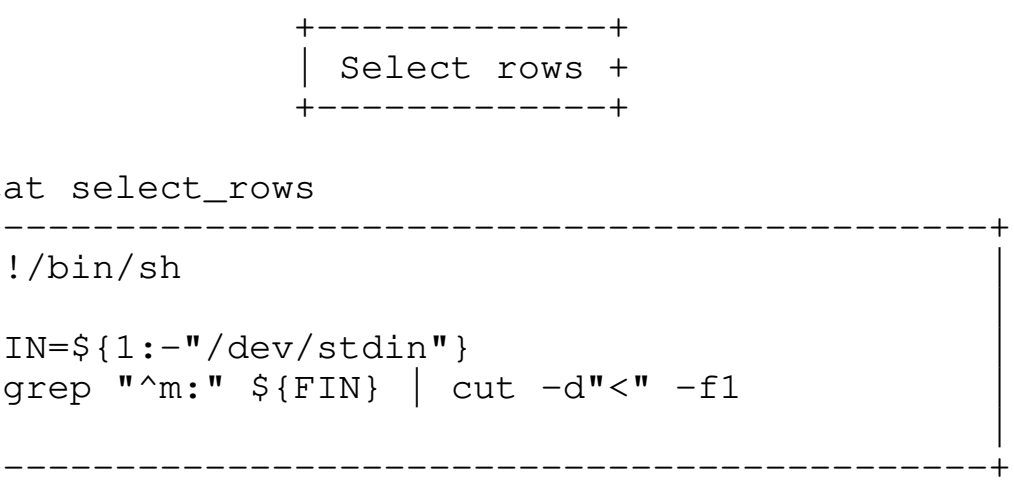
This is the pipeline we used to get the top-10 of Alpine Linux contributors:



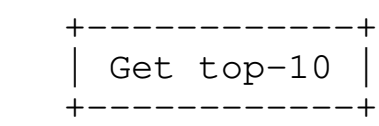
TWO LOGICAL BLOCKS



=

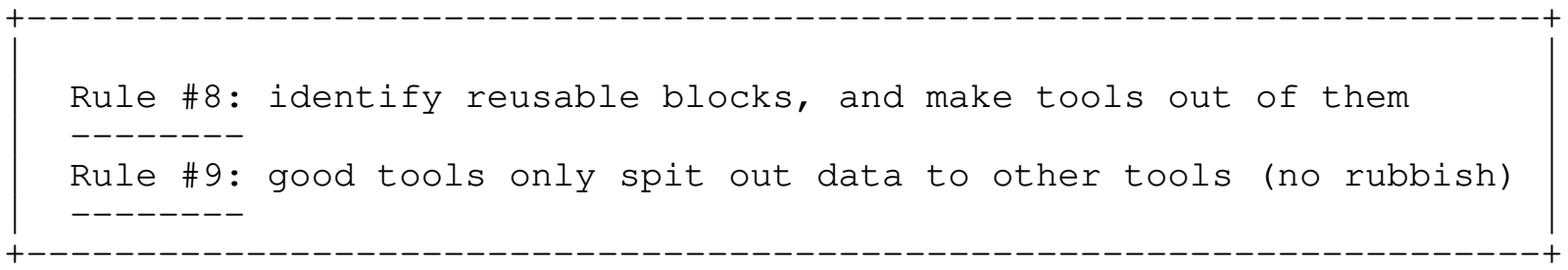

Select rows
$ cat select_rows
#!/bin/sh
FIN=${1:-"/dev/stdin"}
egrep "^m:" ${FIN} | cut -d"<" -f1

=

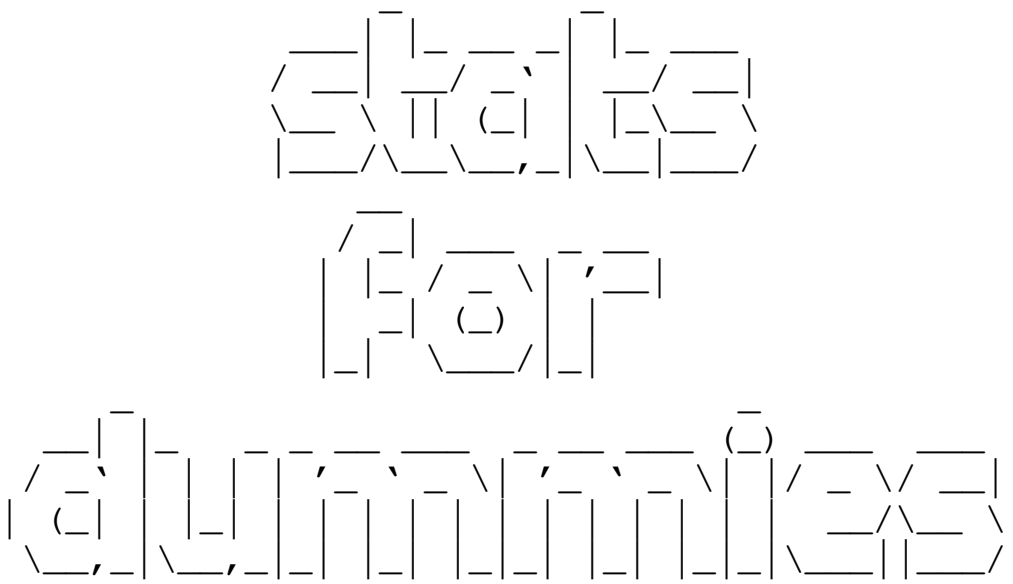



```
$ cat top10
+-----------------------------------------------------+
| #!/bin/sh                                           |
|                                                     |
| FIN=${1:-"/dev/stdin"}                              |
| sort $FIN  |    uniq -c   | sort -rn  | head -10     |
|                                                     |
+-----------------------------------------------------+
```

=


Rule #8: identify reusable blocks, and make tools out of them
Rule #9: good tools only spit out data to other tools (no rubbish)

=



=

The typical (unmotivated) complaint

THE UNIX SHELL IS NOT GOOD FOR COMPUTATIONS!!!

but...

THERE ARE MANY UNIX TOOLS WHICH ARE
VERY GOOD AT THAT!

=

Example 4: Learn from the champions

TWIC: The Week In Chess -- 2.4+ Million Chess Games in official tournaments

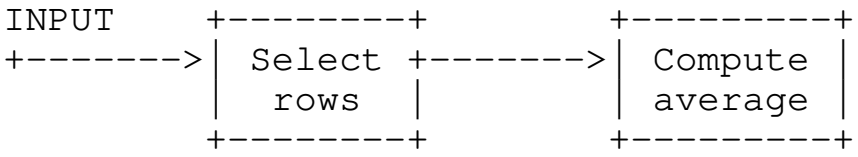
```
[Event "Sparkassen Gp 1"]
[Site "Dortmund GER"]
[Date "2002.07.06"]
[Round "1"]
[White "Topalov, V"]
[Black "Lutz, C"]
[Result "1-0"]
[WhiteElo "2745"]
[BlackElo "2650"]
[ECO "B48"]
[EventDate "2002.07.06"]

1.e4 c5 2.Nf3 e6 3.d4 cxd4 4.Nxd4 Nc6 5.Nc3 Qc7 6.Be3 a6 7.Qd2 Nf6 8.O-O-O
Bb4 9.f3 Ne5 10.Nb3 b5 11.Kb1 Nc4 12.Bxc4 bxc4 13.Nc1 Qb7 14.N1e2 Rb8 15.
b3 O-O 16.Bf4 Ra8 17.Bd6 Bxd6 18.Qxd6 cxb3 19.axb3 a5 20.Rd4 Ra6 21.Qa3 d5
22.exd5 exd5 23.Nf4 Be6 24.Rhd1 h6 25.Ncxd5 Nxd5 26.Nxd5 Rb8 27.Nf6+ gxf6
28.Rd8+ Rxd8 29.Rxd8+ Kh7 30.Qf8 Kg6 31.Qg8+ Kh5 32.Qg7 f5 33.Rd4 Bc8 34.
g3 1-0
```

=

Example 4: Learn from the champions

We want to find the average of the Elo score across all the games.

Data Flow:



Which translates to:

```
egrep "^\[.....Elo " twic.pgn  |  ???
```

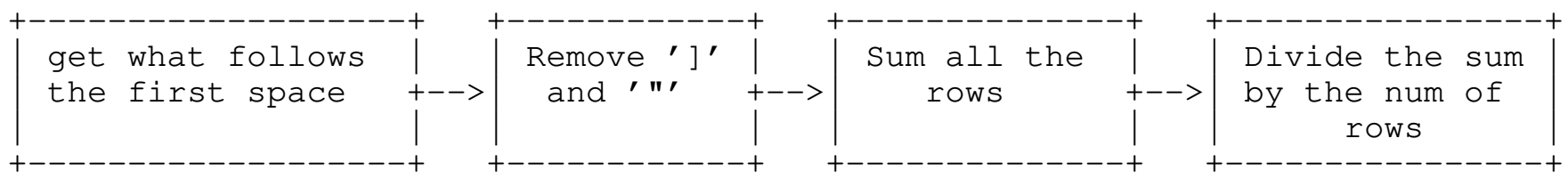
=

Example 4: Learn from the champions

Let’s have a look at what comes out of:

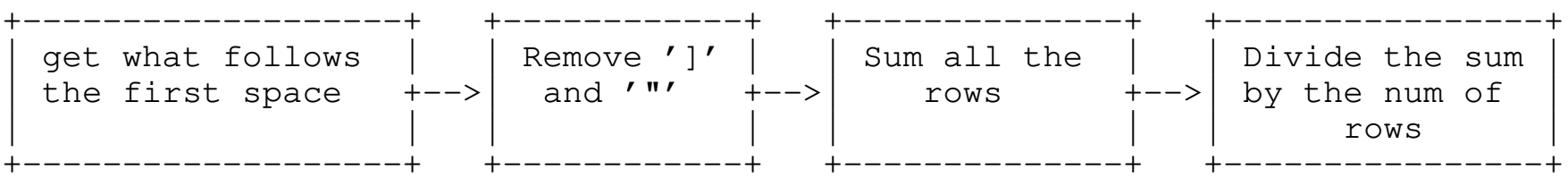
```
egrep "^\[.....Elo " twic.pgn  |  head -5
[WhiteElo "2745"]
[BlackElo "2650"]
[WhiteElo "2710"]
[BlackElo "2697"]
[WhiteElo "2745"]
```

So we need to:



=

Example 4: Learn from the champions



Translates to:

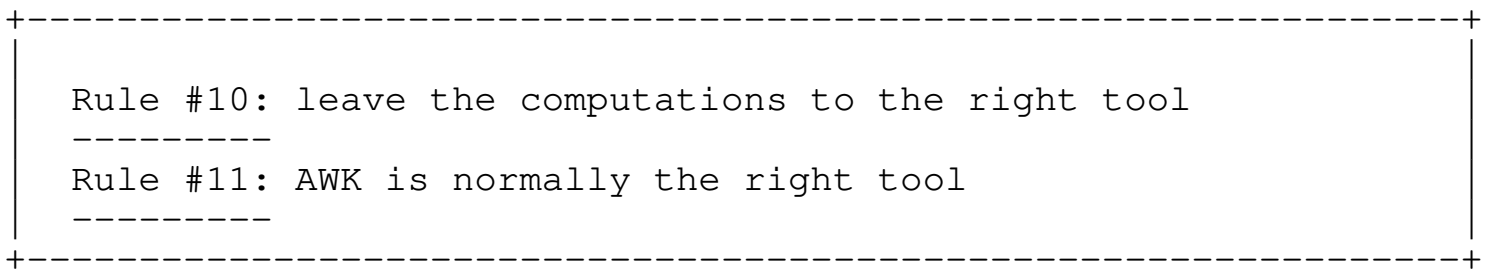
```
cut -d" " -f2  | sed -r 's/\]//g;s/\"//g' | awk '{t += $1}END{print t/NR}'
```

So in the end we get:

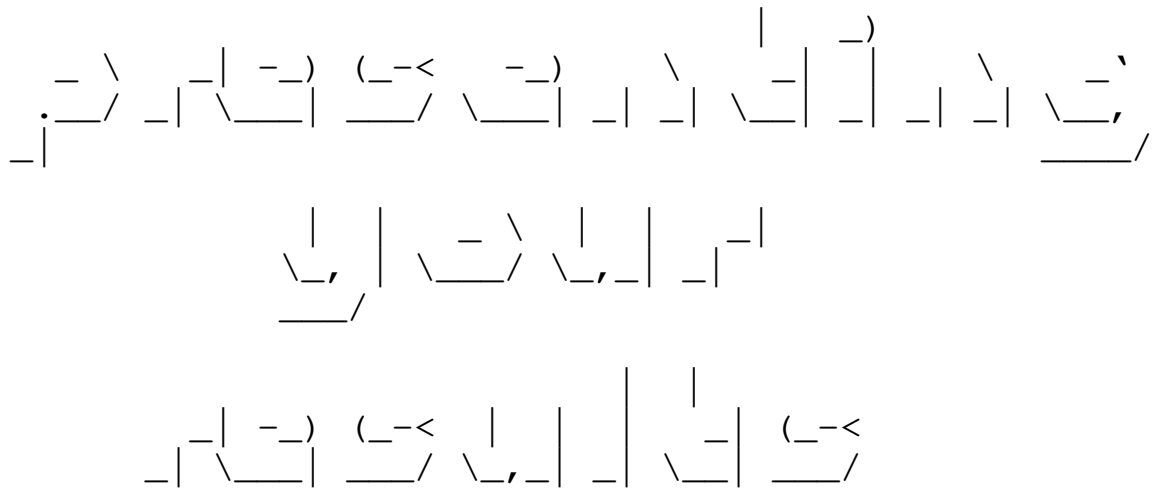
```
$ egrep "^\[.....Elo " twic.pgn  | cut -d" " -f 2 | \
      sed -r 's/\]//g;s/\"//g' | awk '{t += $1}END {print t/NR}'
2263.95
```

So the average player in there is at least a Candidate Master!

=


Rule #10: leave the computations to the right tool
Rule #11: AWK is normally the right tool

=



=

Example 5: How many "good" players in the TWIC DB?

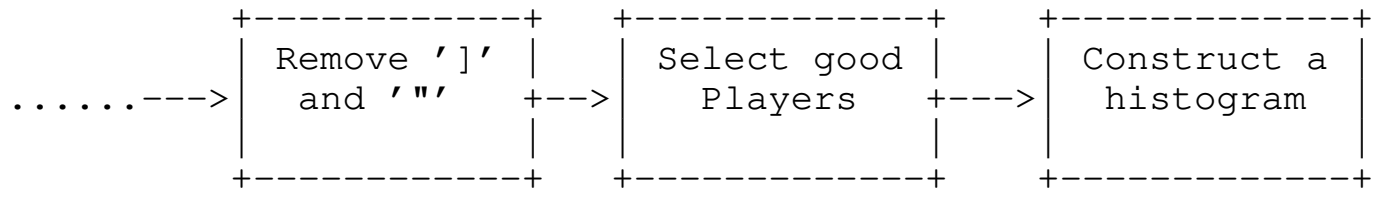
How many games by players with a score higher than 2000 (1st class)?

And how many by players with a score higher than 2200 (Candidate Master)?

And how many by players with a score higher than 2300 (FIDE Master)?

....

Reusing the data last data flow:



=

Example 5: How many "good" players in the TWIC DB?

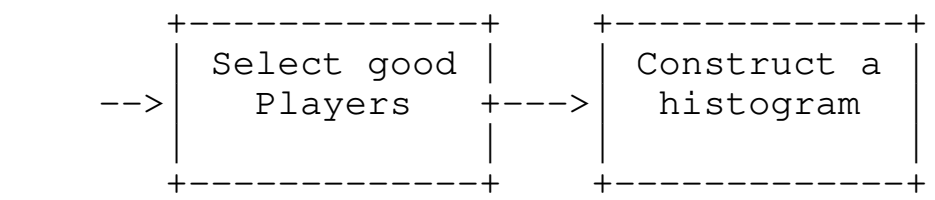
We use a trick: Maintain only the two largest digits of the Elo score!

```
2456  -->  24
2786  -->  27
....
```

And then we count how many 20, 21, 22, etc we have!

=

Example 5: How many "good" players in the TWIC DB?




```
   ...  | egrep "2..."    | sed -E 's/(2.).*/\1/g' | sort | uniq -c
 404184 20
 548579 21
 670332 22
 714168 23
 717261 24
 534279 25
 246087 26
  75139 27
   6114 28
```

So we have:

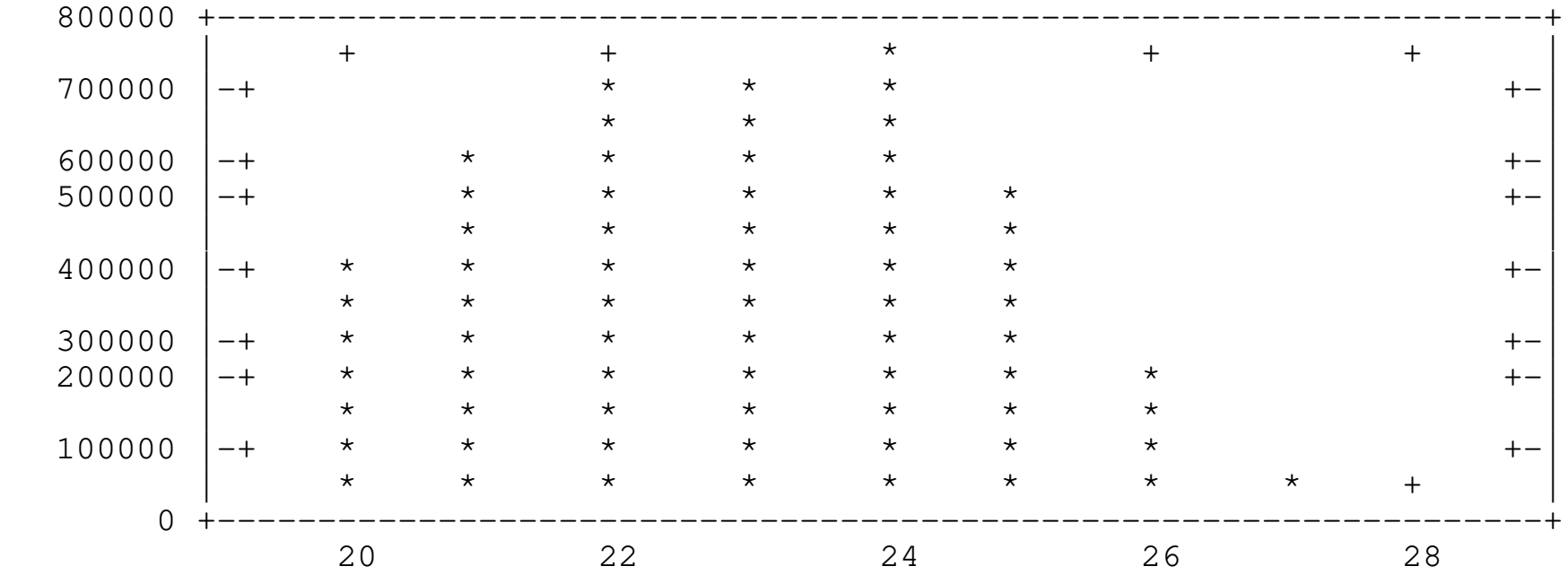
404184 players whose score is between 2000 and 2099,
548579 players whose score is between 2100 and 2199,
.....

Let's produce a histogram now!

=

Example 5: How many "good" players in the TWIC DB?

```
egrep "^\[.....Elo " twic.pgn  | cut -d" " -f 2 | \
    sed -r 's/\]//g;s/\"//g' | egrep "2..." | sed -E 's/(2.).*/\1/g' | \
    sort | uniq -c | gnuplot dumbplot
```



=

Example 5: How many "good" players in the TWIC DB?

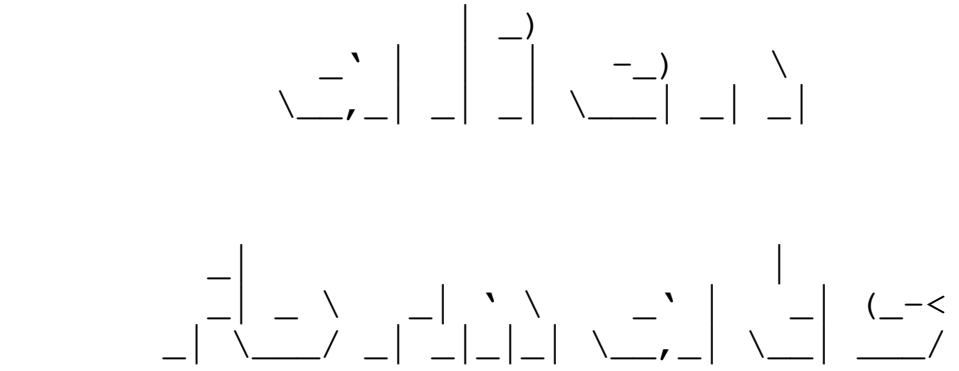
The magic is done by gnuplot:

```
$ cat dumbplot
+-------------------------------------------------------------+
| set term dumb size 80,18 ;                                  |
| set xrange [19:29]                                          |
| plot "< cat -" u 2:1 w impulses title "";                   |
|                                                             |
+-------------------------------------------------------------+
```

=

```
+-------------------------------------------------------------+
|                                                             |
|  Rule #12: using a shell does not preclude good visualisations |
|  ---------                                                  |
|  Rule #13: learn plot/gnuplot                               |
|  ---------                                                  |
+-------------------------------------------------------------+
```

=



=

Example 6: a geographic data set

Geographic data sets (points, lines, polygons) are often available in CSV format (Comma-Separated Values).

Points of the perimeter of Wards across the UK (45+Million points - 30GB+):

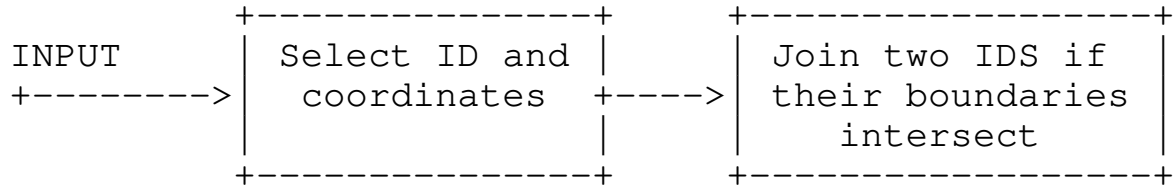
```
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441163.8963,236323.0026
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441173.4029,236325.102
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441185.6962,236353.9038
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441199.2017,236366.9
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441223.4998,236279.3951
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441257.9972,236236.2974
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441266.704,236429.3022
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441271.2965,236222.7013
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441299.7007,236234.7978
1538,"Adderbury, Bloxham & Bodicote Ward",E05011348,441330.8011,236493.3038
```

=

Example 6: a geographic data set

Objective: construct a graph where each node is a ward and two wards are connected by an edge if they share a point on their perimeter

Data flow:

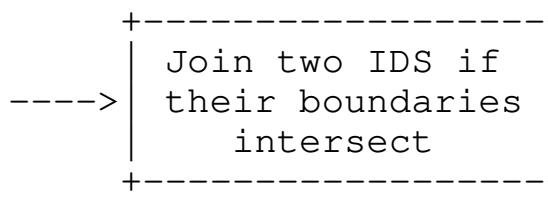


Which translates to:

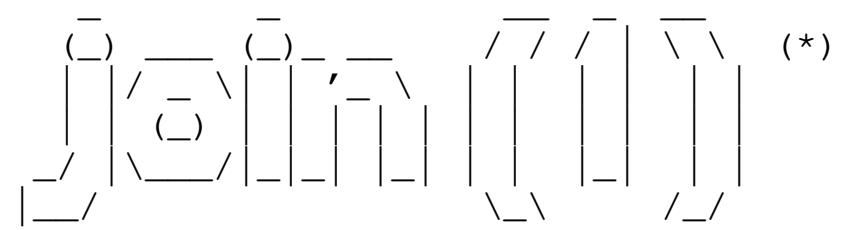
```
csvtool cols 1,4,5  wards.csv | sed -r 's/,/ /' | sort -k2 | ?????
```

=

Example 6: a geographic data set



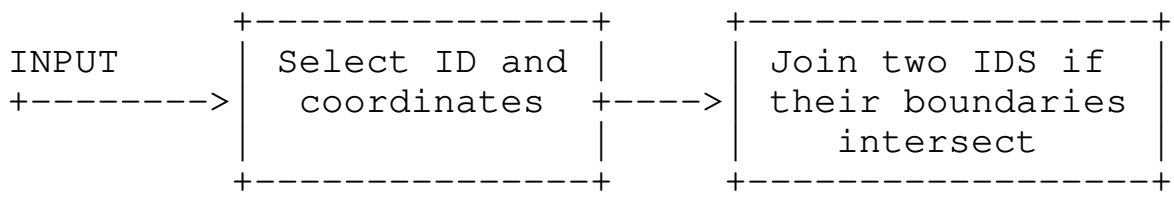
To perform this operation, we will use a tool as old as myself:



(*) join(1) appeared in Unix V7 (1979)

=

Example 6: a geographic data set
==========



Translates to:

```
csvtool cols 1,4,5  wards.csv | sed -r 's/,/ /' | sort -k2 > tmpfile
```

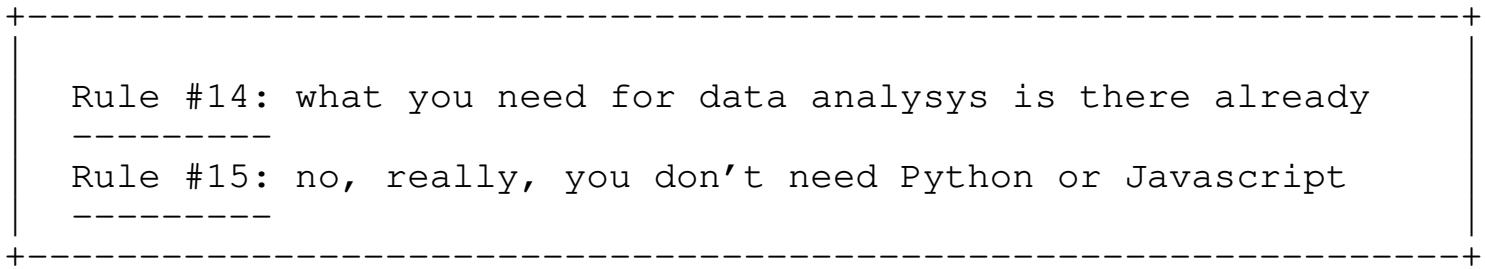
followed by:

```
join -1 2 -2 2 tmpfile tmpfile | cut -d " " -f 2,3  | sort | uniq
```

where the last " ... | sort | uniq " removes duplicated edges.

(*) Each edge still appears twice in the list though... use AWK to solve this ;P

=


Rule #14: what you need for data analysys is there already
Rule #15: no, really, you don’t need Python or Javascript

=

Other tools useful for data analysis

- xargs, parallel (run commands in parallel on many input chunks)
- jq (parse JSON structures)
- bc/dc (arbitrary precision calculators)
- tbl/eqn/pic (troff packages to create tables, equations, graphs)
- split/csplit (split files)
- comm (get common lines between files)
- paste (paste two files side-by-side)
- seq (create sequences of numbers -- useful in for loops...)

=


Thanks !

=

Credits

- This presentation was created with vim(1)

      https://www.vim.org

- Box-and-arrow plots were made with gramscii(1)

      git clone git://kalos.mine.nu/gramscii

- Headings were created with figlet(6) and toilet(1)

      http://figlet.org
      http://caca.zoy.org/wiki/toilet

- This presentation was delivered with catpoint

      git clone git://r-36.net/catpoint

DISCLAIMER: most ASCII arts are (c) by their original
            authors. If no author is indicated, it is
            either unknown or myself. In the latter
            cases the ASCII arts are free for all.

=

Contacts

- email: katolaz [at] freaknet [dot] org [dot] remove [dot] dots [dot]
- gopher://kalos.mine.nu
- gopher://republic.circumlunar.space/1/~katolaz
- gopher://medialab.freaknet.org/1/katolaz/
- gopher://cgit.mine.nu
- https://cgit.mine.nu

=

Other References

csvtool - manage CSV anc TSV files
http://forge.ocamlcore.org/projects/csv/

jq - manage and convert JSON files
https://github.com/stedolan/jq

TWIC The Week In Chess -- The oldest e-zine about chess
https://theweekinchess.com/

twic_script: keep an updated Chess games databases from TWIC
git clone git://cgit.mine.nu/twic_script

These slides were prepared for the talk "Data Analysis on the command line" I gave at the Bitreich brcon 2019,

The slides are meant to be viewed using `catpoint`:

```
$ git clone git://r-36.net/catpoint
$ cd catpoint && make && make install
$ catpoint 0*.txt
```

The master file is in markdown format (slides.md). The single .txt files were obtained using `md2point`, available in `pointtools`:

```
$ git clone git://r-36.net/pointtools
```
