Clojure for Data Science

Stéphane Labruyère - @_slabruyere
stephane@slabruyere.net
April 23rd, 2019

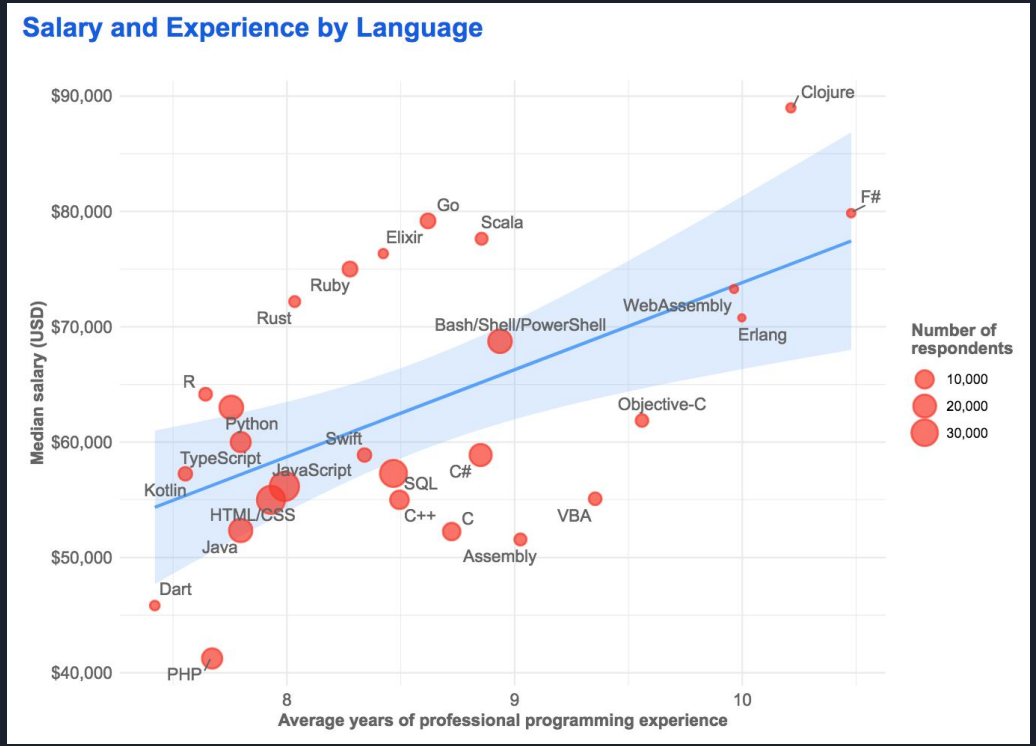
Clojure

Clojure is a **modern**, **dynamic**, and **functional** dialect of the **Lisp** programming language on the Java platform.

Like other Lisps, Clojure treats **code as data** and has a Lisp **macro** system.

Source: https://en.wikipedia.org/wiki/Clojure

Clojure



Source: https://insights.stackoverflow.com/survey/2019?utm_source=Iterable&utm_medium=email&utm_campaign=dev-survey-2019#work-_-salary-and-experience-by-language

Clojure

« Yes, that was the big revelation to me when I was in graduate school—when I finally understood that the half page of code on the bottom of page 13 of the Lisp 1.5 manual was Lisp in itself. These were “Maxwell’s Equations of Software!” This is the whole world of programming in a few lines that I can put my hand over. »

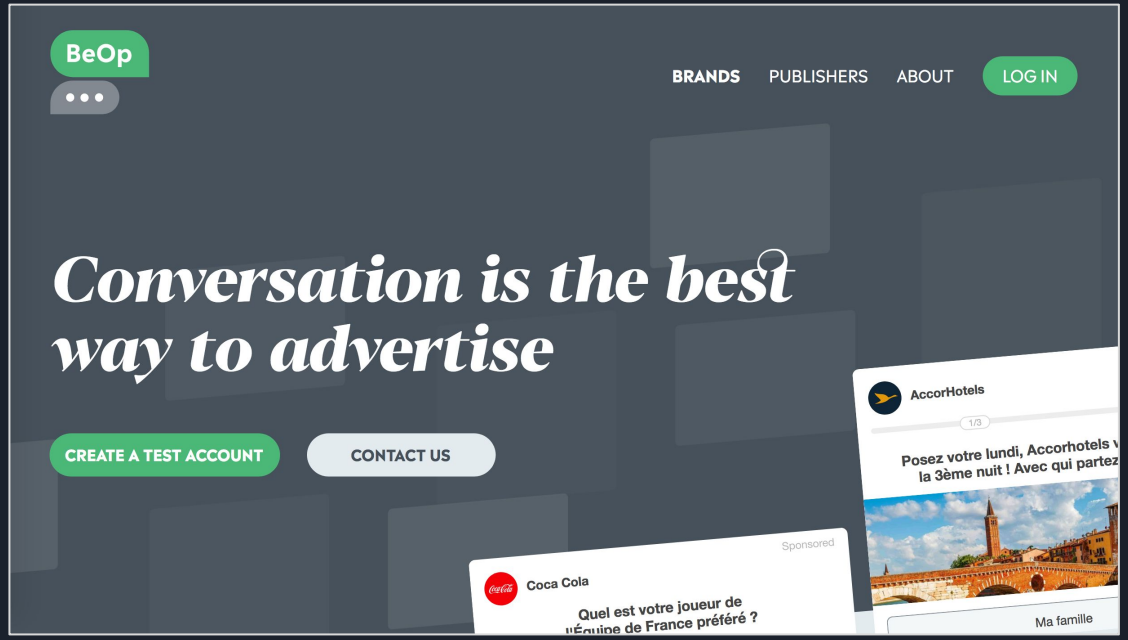
Alan Kay

Source: https://queue.acm.org/detail.cfm?id=1039523

Clojure

Source: https://twitter.com/paulg/status/1118808090466574336

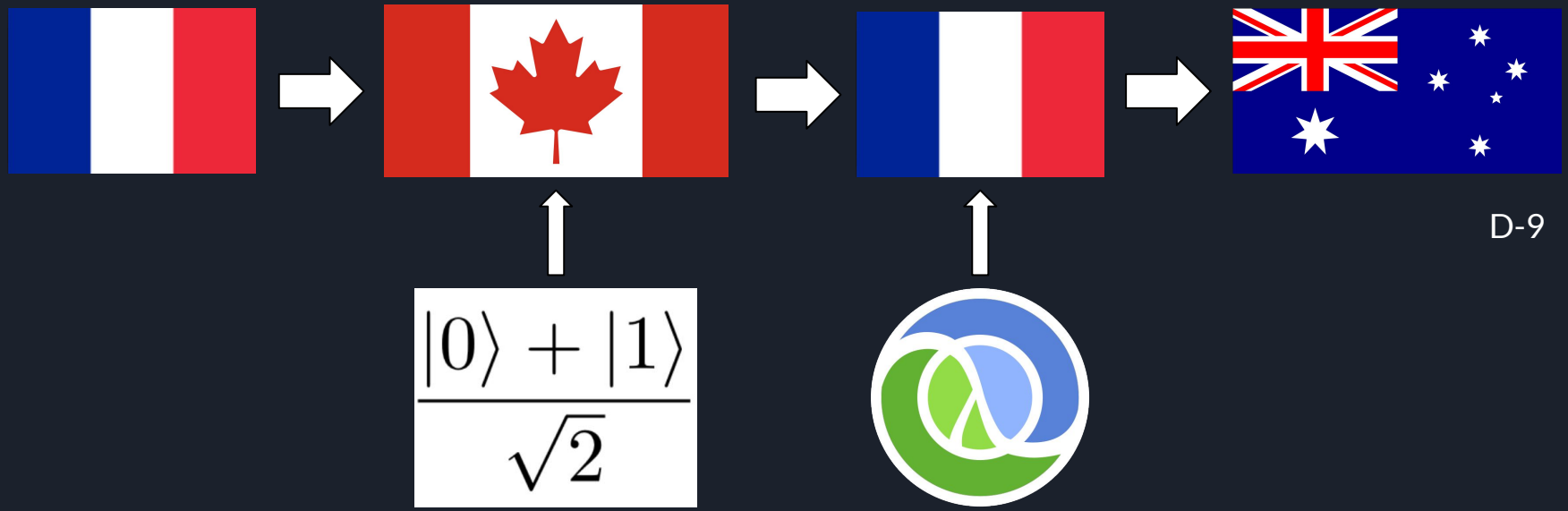
A little about me



Source: https://www.beop.io

A little about me

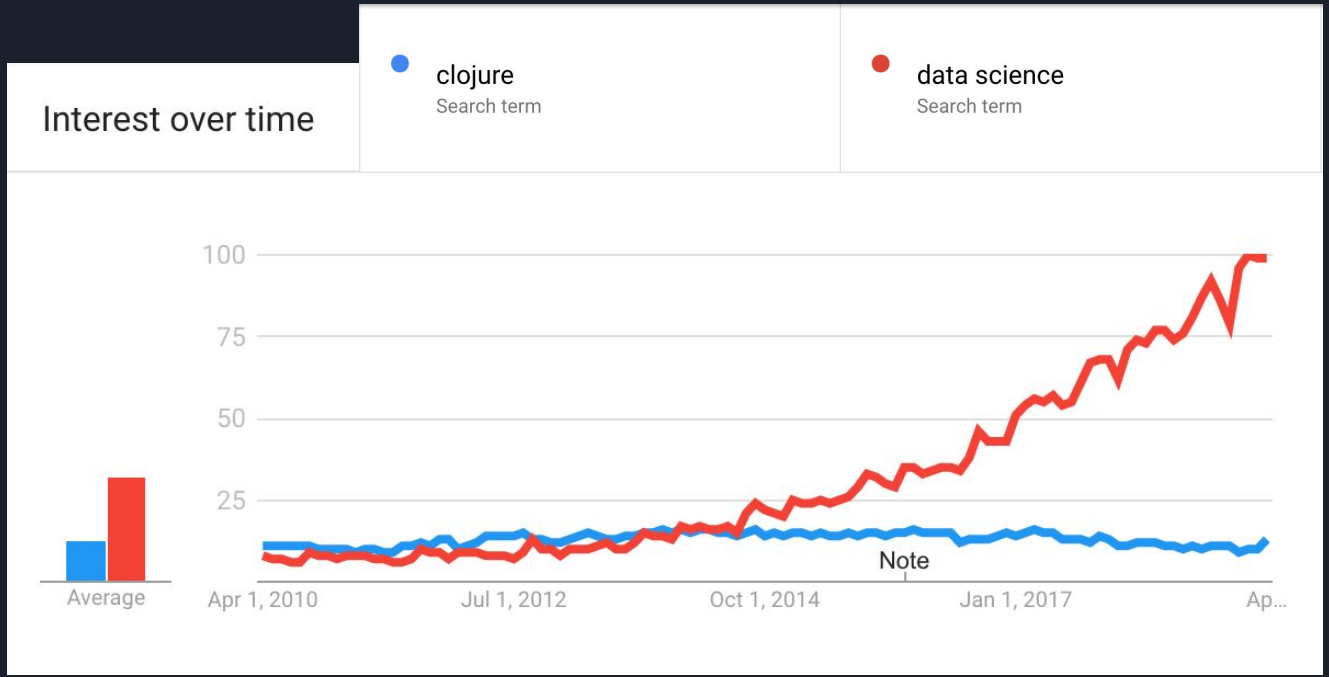
I'm a husband, a father, an engineer. I play the guitar. I love startups, and Clojure of course!



A little about me

Source: https://twitter.com/josh_wills/status/198093512149958656

Comparing trends



So what happened ?

Rise of Python

- Used a lot by researchers
- More portable than Matlab
- No need to be a software engineer to learn it

  => lots of good libraries, books, trainings

Clojure basics

```clojure
; everything is a list
(def l
  (list 1 2 3))

; vectors also exist
(def v
  [1 2 3])

; sets can be super useful
(def s
  #{1 2 3})

; we can map, filter, reduce...
(map inc v)
(filter even? l)
(reduce + s)
```

```clojure
; function call
(f arg1 arg2)

; function definition
(defn my-function
  [arg1 arg2 & arg3]
  (do-something arg1))

; maps usually use keywords
(def m
  {:a 1
   :b 2
   :c {:d 3}})

; keywords can be used as accessor functions
(:a m)
;; => 1

; accessors are handy
(get-in m [:y :z])
;; => nil
```

Data manipulation in Clojure

```clojure
; map-filter-reduce
(->> my-collection
  (map do-some-transormation)
  (filter do-some-filtering)
  (map do-something-else)
  (reduce do-some-operation))

; grouping
(group-by :a [{:a 1, :b 2}
              {:a 1, :b 3}])
;; => {1 [{:a 1, :b 2}
;;        {:a 1, :b 3}]}

; associating headers with values, for csv for instance
(zipmap ["last name" "first name" "dob"] ["doe" "john" "01/01/1970"])
;; => {"last name" "doe"
;;     "first name" "john"
;;     "dob" "01/01/1970"}
```

Data manipulation in Clojure

```clojure
(def x-form
  (comp
    (map inc)
    (filter even?)
    (dedupe)
    (mapcat range)
    (partition-all 3)
    (partition-by #(< (apply + %) 7))
    (mapcat flatten)
    (random-sample 1.0)
    (take-nth 1)
    (keep #(when (odd? %) (* % %)))
    (keep-indexed #(when (even? %1) (* %1 %2)))
    (replace {2 "two" 6 "six" 18 "eighteen"})
    (take 11)
    (take-while #(not= 300 %))
    (drop 1)
    (drop-while string?)
    (remove string?)))

(transduce x-form + (vec (interleave (range 18) (range 20))))
;; 246
```

Source: https://gist.github.com/richhickey/b5aefa622180681e1c81

Data validation in Clojure

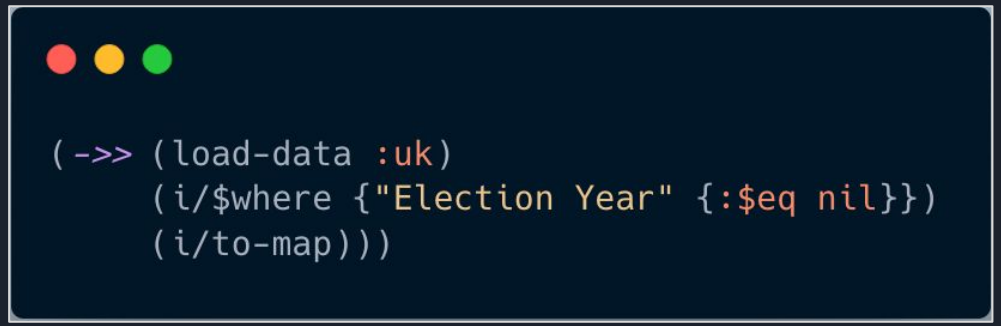
```clojure
; spec examples
(s/def ::first-name (s/and string? #(<= (count %) 20)))
(s/def ::last-name (s/and string? #(<= (count %) 30)))
(s/def ::email (s/and string? valid-email?))
(s/def ::phone (s/and number? valid-phone?))

; define what is expected from a user
(def user-spec
  (s/keys :req [::first-name ::last-name ::email]
          :opt [::phone]))
```

A few years back

Incanter

- Does everything
- Not really fast
- Not actively maintained anymore
- Nice ideas that are slowly reused in other libraries



A wind of change

Kixi.stats

- Statistics with transducers
- Still beta but very complete

Cortex

- Easily work with neural networks
- Regression and feature learning

Koala

- Equivalent of Pandas in Clojure
- Work in progress

Flare

- Dynamic neural networks

Heavy-lifting

Apache MXNet

- Open source deep learning
- Very active maintainers

Integration with NextJournal

- Shared notebooks
- A lot of cool examples

Heavy-lifting

Neanderthal

- Based on native libraries
- Vectors and matrices on GPU and CPU

ClojureCL

- Taking control of the GPU

QUICK COMPARISON

n × n	Neanderthal	Optimized Java	×
2 × 2	118 ns	57 ns	0.49
4 × 4	143 ns	132 ns	0.93
16 × 16	1.1 µs	3.8 µs	3.44
64 × 64	47 µs	211 µs	4.46
128 × 128	191 µs	1.6 ms	8.14
256 × 256	639 µs	12 ms	18.85
1024 × 1024	38 ms	751 ms	19.77
2048 × 2048	288 ms	6 sec	20.91
8192 × 8192	18 sec	6 min	20.62

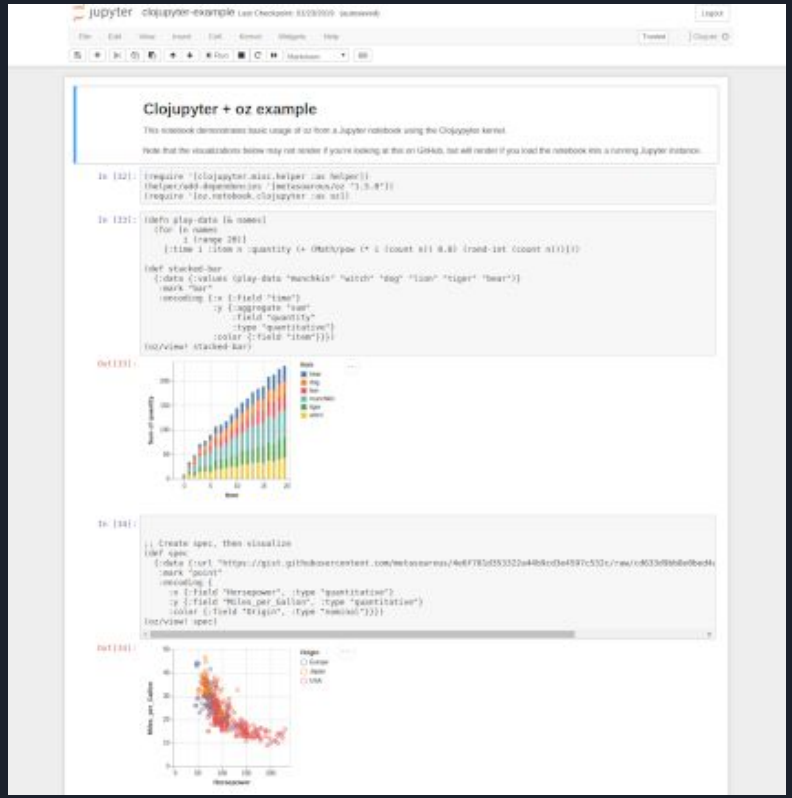
Matrix multiplication benchmark

Source: https://www.youtube.com/watch?v=bEOOYbscyTs


Elements Network Sources Timeline Profiles
<top frame>
Figwheel: reloading javascript file ...
/js/out/flappy_bird_demo/core.js client.cljs:28
Figwheel: reloading javascript file ...
/js/out/flappy_bird_demo/core.js client.cljs:28
Console Search Emulation Rendering
core.cljs 418 (105,34) Git:master (Clojure -2 cljs

The figwheel of data science

- Becoming a sort of challenge that the community wants to tackle
- Oz
  - Using Vega / Vega-lite for data viz
  - Clojupyter notebook integration



Why now ?

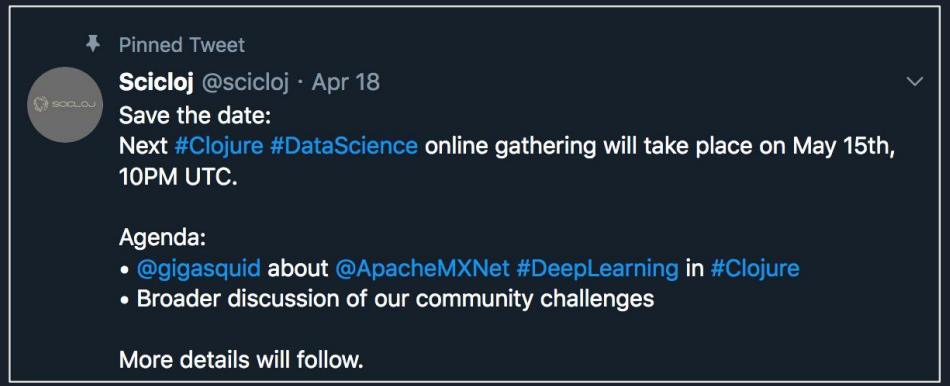
- A more mature ecosystem
- A lot of effort in JVM efficiency / Graal VM
- Functional Programming is a hot topic
- Clojure Spec !
- ClojureScript
- Scala interop
- Companies supporting Clojure
- Incredible (and growing) community, Open Source work

Who to follow ?

- People
  - Carin Meier @gigasquid
  - Dragan Djuric @draganrocks
  - Aria Haghighi @aria42
  - Simon Belak @sbelak
  - Henry Garner @henrygarner
- Orgs
  - Uncomplicate
  - Scicloj

More info

- Clojurians Slack #data-science
- Clojurians Zulip
- Online Clojure data science meetup



In a nutshell

Try Clojure, it’s fun !

Thank you !