



Clojure for Data Science

Stéphane Labruyère - [@slabruyere](https://twitter.com/slabruyere)
stephane@slabruyere.net
April 23rd, 2019



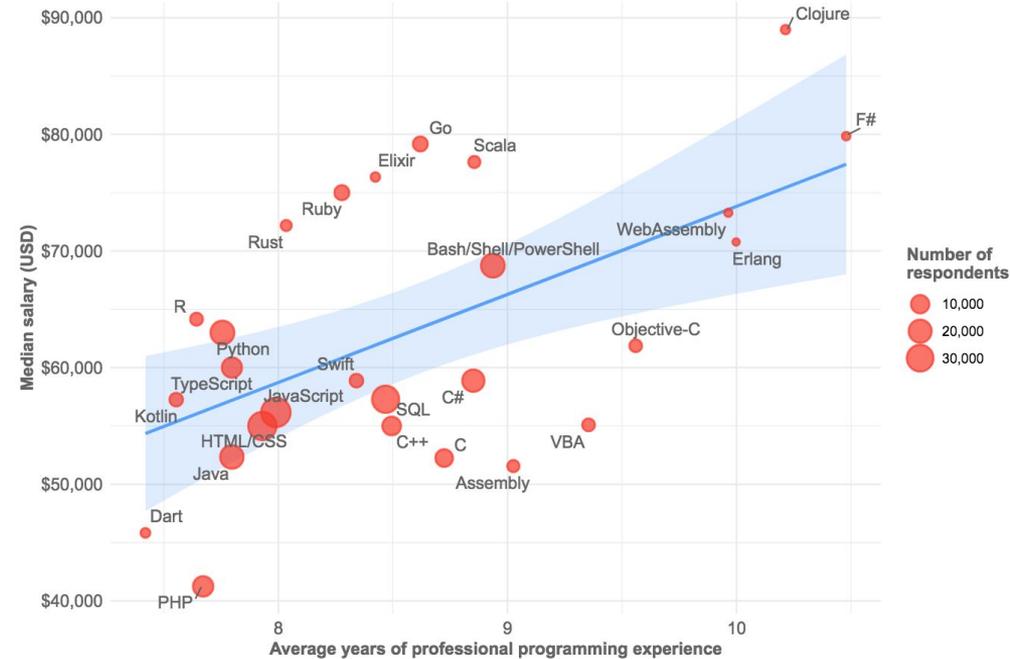
Clojure

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.

Clojure

Salary and Experience by Language



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



David Desimon @David_Desimon · Apr 18

Do you still think it's worth learning Lisp nowadays?



1



3



10



 You Retweeted



Paul Graham 

@paulg

Replying to [@David_Desimon](#)

Yes. Try Clojure.

11:26 AM · Apr 18, 2019 · [Twitter Web Client](#)

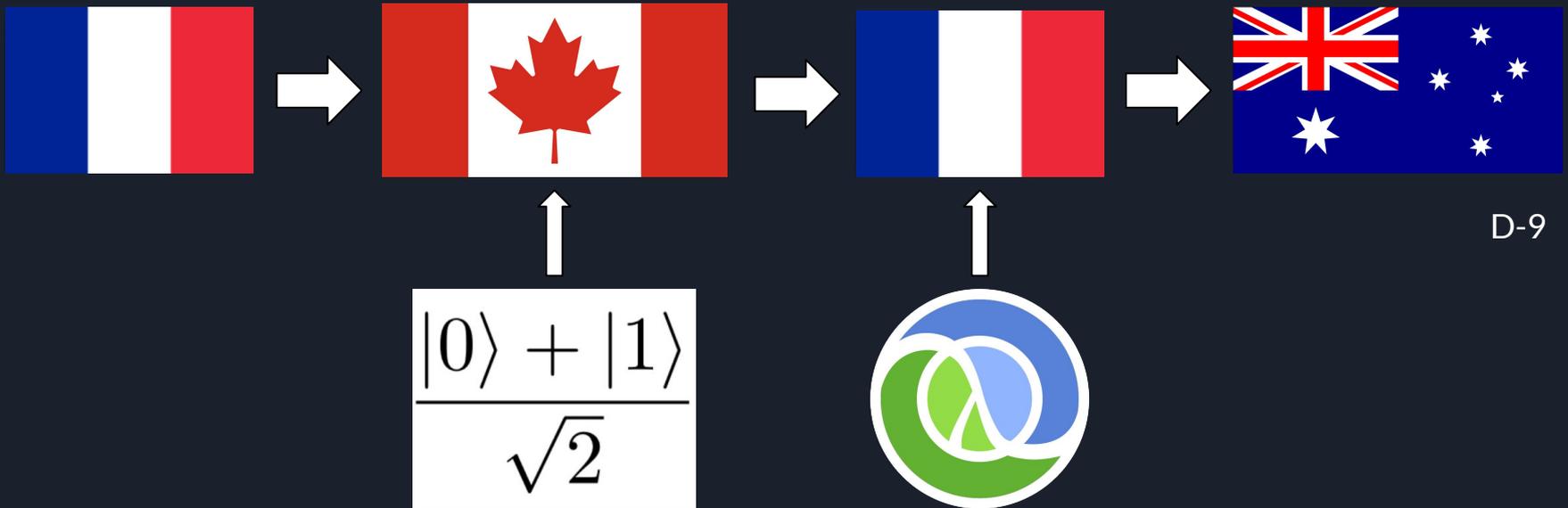
A little about me

The screenshot shows the BeOp website interface. At the top left is the BeOp logo in a green rounded rectangle, followed by a three-dot menu icon. The navigation bar includes the links 'BRANDS', 'PUBLISHERS', 'ABOUT', and a 'LOG IN' button in a green rounded rectangle. The main content area features the headline 'Conversation is the best way to advertise' in a white, italicized serif font. Below the headline are two buttons: 'CREATE A TEST ACCOUNT' in a green rounded rectangle and 'CONTACT US' in a white rounded rectangle. At the bottom, two advertisement cards are displayed. The first is a Coca-Cola ad with the text 'Quel est votre joueur de l'Équipe de France préféré ?' and a 'Sponsored' label. The second is an AccorHotels ad with the text 'Posez votre lundi, Accorhotels v la 3ème nuit ! Avec qui partez' and a 'Ma famille' label. Both ads feature images of buildings.

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



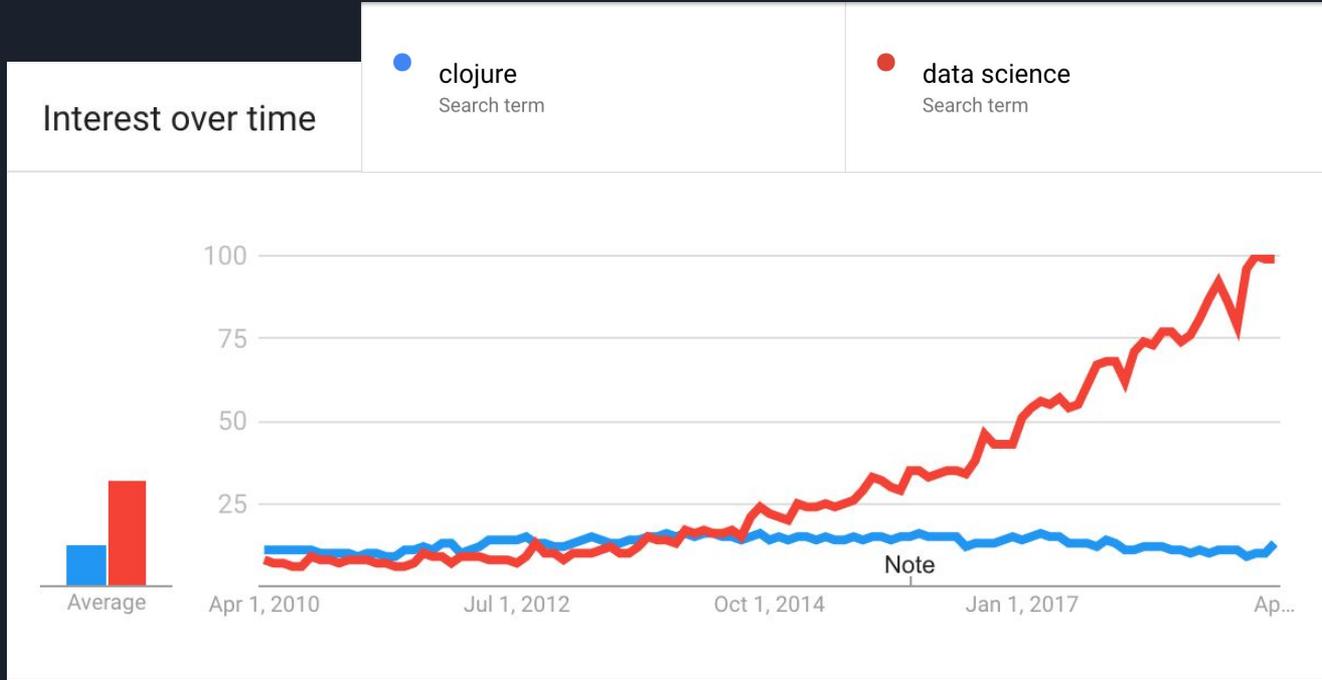
Josh Wills
@josh_wills



Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

6:55 PM · May 3, 2012 · [Twitter Web Client](#)

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



```
; 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)
```



```
; 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

```

; 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

```
(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"}))

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

Data validation in 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



```
(->> (load-data :uk)
      (i/$where {"Election Year" {:$eq nil}})
      (i/to-map)))
```



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			
$n \times n$	Neanderthal	Optimized Java	\times
2×2	118 ns	57 ns	0.49
4×4	143 ns	132 ns	0.93
16×16	1.1 μ s	3.8 μ s	3.44
64×64	47 μ s	211 μ s	4.46
128×128	191 μ s	1.6 ms	8.14
256×256	639 μ s	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



Elements Network Sources Timeline Profiler

stop frame

Filetree: reloading javascript file s	chrome://1114/28
Filetree: reloading javascript file s	chrome://1114/28
Filetree: reloading javascript file s	chrome://1114/28

Console Search Emulation Rendering

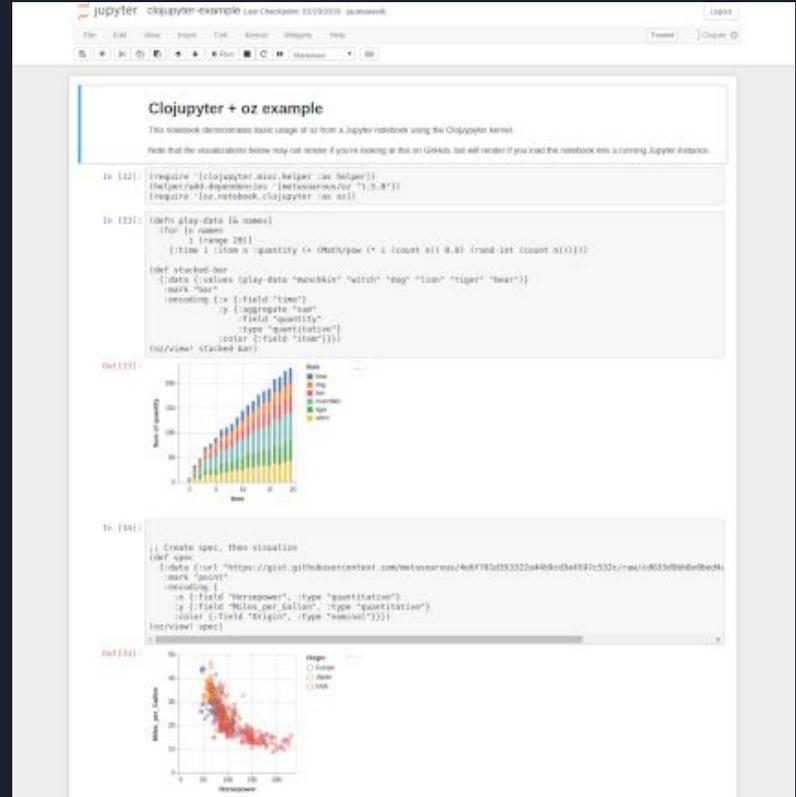
```

pillars-in-world())
(defn state-mov [[keys [time-delta] :as st])
  (cond
    (Flappy-y
     )
    (defn update-flappy [[keys [time-delta initial-vel flappy-y jump-count] :as st])
      (if (not? jump-count)
        (let [cur-vel (+ initial-vel (* time-delta gravity))
              new-y (- flappy-y cur-vel)
                  new-y (if (> new-y (- bottom-y flappy-height)
                             (- bottom-y flappy-height)
                             new-y))]
          (assoc st
                 :flappy new-y))
          st)
        (defn score [[keys [cur-time start-time] :as st])
          (let [score (- (.abs js/Math) (floor (/ (- (* (- cur-time start-time)
                                                         pillar-vel)
                                                         pillar-spacing)))
                        4)]
              (assoc st :score (if (not? score) 0 score)))
              score)
          (defmethod transform :time [[_ timestamp] state]
            (-> state
                (cond
                  (cur-time timestamp
                   time-delta (- timestamp (:flappy-start-time state)))
                  update-flappy
                  update-pillars
                  #.initial-vel
                  #.score))
                state)
            (defn jump [[keys [cur-time jump-count] :as state]]
              (-> state
                  (cond
                    (jump-count (inc jump-count)
                     :flappy-start-time cur-time
                     initial-vel jump-vel))
                    (defmethod transform :jump [_ state]
                      (jump state))
                      (defn start-game [event-chan]
                        (.requestAnimationFrame js/window (/ (time)
                                                                (pub) event-chan (:start-game time)))
                        (defmethod transform :start-game [[_ time] state]
                          (reset-state state time))
                          -- core.cljs 41% (185,34) Git-master: (Clojure -2.cld)

```

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

📌 Pinned Tweet

 **Scicloj** @scicloj · Apr 18

Save the date:
Next [#Clojure](#) [#DataScience](#) online gathering will take place on May 15th, 10PM UTC.

Agenda:

- [@gigasquid](#) about [@ApacheMXNet](#) [#DeepLearning](#) in [#Clojure](#)
- Broader discussion of our community challenges

More details will follow.



In a nutshell

Try Clojure, it's fun !

Thank you !

