SCALA AS AN ENVIRONMENT FOR STATISTICAL PROGRAMMING(?)

Charles Francis
@agentcoops

Wednesday, October 23, 13

few archetypes of tech talks

mean to offer this as both an inquiry and an intervention---how does scala compare and what would be necessary to make it better, in someways a longwinder continuation of some of the questions after tom's talk

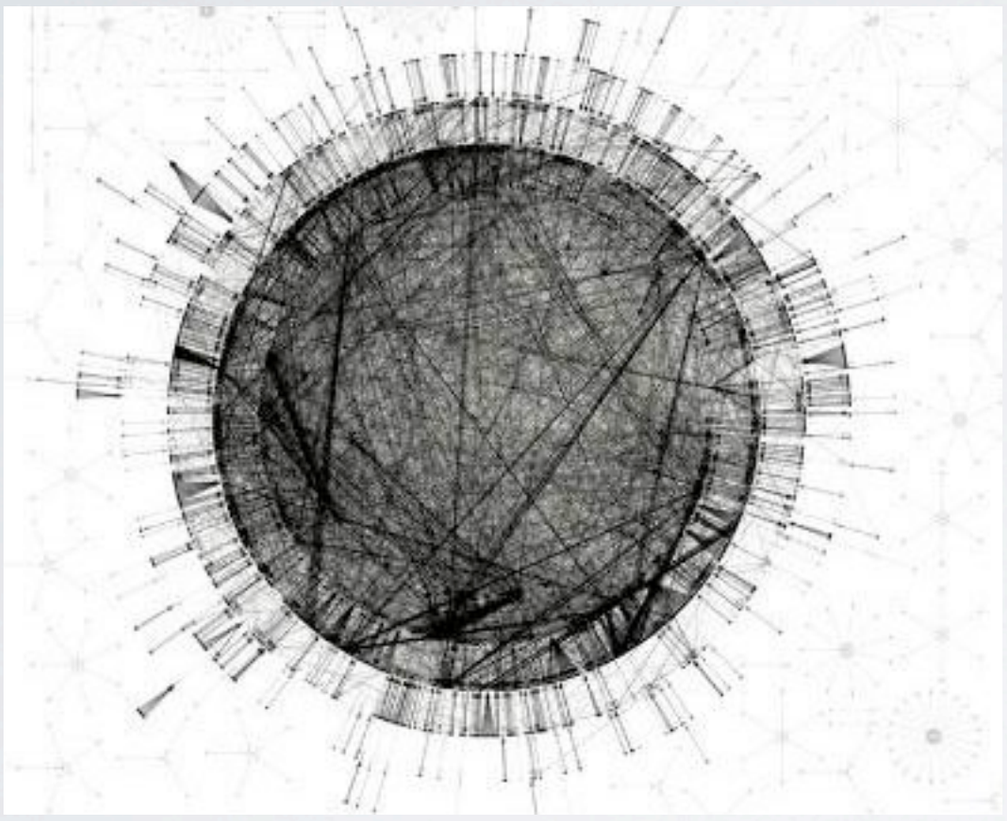
i feel a little bit of shame that this talk was a bit last minute, so some prototypes aren't quite ready, so this is an attempt at an honest inquiry, an intervention and, let's say, the announcement of a program to come

great talks today, glad i'm at the end, hopefully when you're all more susceptable to influence

BACKGROUND

Qmedtrix®

Berkman
The Berkman Center for Internet & Society
at Harvard University



Isabelle

PELLUCID

NewScientist Tech

Home News In-Depth Articles Opinion CultureLab Galleries Topic Gui

SPACE TECH ENVIRONMENT HEALTH LIFE PHYSICS&MAT

Home | Tech | News

Family tree for inventions shows joined-up thinking

› 27 August 2008 by **Sumit Paul-Choudhury**
› Magazine issue 2671. **Subscribe and save**

DOES culture evolve according to laws just like those that govern biological development?

This has been hotly debated ever since biologist Richard Dawkins posited the idea of memes - units of information that replicate, mutate and evolve - in his 1976 book *The Selfish Gene*. So might it be possible to create the equivalent of a genealogy or genome for cultural "organisms"?

A team lead by Mark Bedau, a philosophy researcher at Reed College in Portland, Oregon, has now investigated this for one particular cultural "life form": patents. "Technology is a window on culture, and patents are a window on technology," Bedau told delegates at the Artificial Life XI conference in Winchester, UK, earlier this month.

What also makes patents attractive is that they offer a large and readily accessible data set. Patented inventions are uniquely numbered, and have to be novel and useful, so they are distinct "organisms". Reproduction corresponds to citation by another patent, and since every invention has to cite the patents it borrows from, it is easy to map out its genealogy. Every

$$[\![\mu Z.\phi]\!]_i = \bigcap\{T \subseteq S \mid [\![\phi]\!]_{i[Z:=T]} \subseteq T\}$$

- portlander:
  - Used to come to PDX Scala (gave introduction to Scala 2.8 in 2010)

    Studied mathematics and CS here in Portland

been doing "data science", let's assume this refers to some real thing and we all know as we do, what it means, and functional programming for pretty much all my career researched formal logics for program verification

- Began working as data scientist on study of corpora containing US patent abstracts and citation networks
  - Also worked on Mediacloud (news analytics), Isabelle (visualization and sanitization of proofs), medical fraud in Portland, currently financial analytics in NYC
- Mostly R and Python for analytics to begin with, but heavy emphasis on FP in my studies, separate paths, until...

SCALA!

- Utterly sold on the language, but

- Reimplemented simple data munging from Python, horrified at memory use, how difficult just to figure out how to parse a file...

- (even with a REPL to explore always been allergic to Java apis...)

scala.io

class Source

[source: scala/io/Source.scala]

```
abstract class Source
    extends Iterator[Char]
```

The class `Source` implements an iterable representation of source files. Calling method `reset` returns an identical, resetted source.

Author
Burak Emir

Version
1.0

Value Summary

protected var	*ccol* : Int
var	*ch* : Char the last character returned by next. the value before the first call to next is undefined.
protected var	*cline* : Int
var	*descr* : String description of this source, default empty
protected abstract val	*iter* : Iterator[Char] the actual iterator
var	*nerrors* : Int
var	*nwarnings* : Int
var	*pos* : Int

was never really a java developer... so all the different file types horrified me (python: file("x.txt", r), etc) and then we get the data into some reasonable format and find that weka is...

as elegant as it’s namesake

and the performance is crazy (boxinggg) and we have all these insane numeric types (non-commutative addition, whatever) with varying levels of support (and even ways to do the same thing) for even core java (can we log it?)

BEAT A RETREAT...

- I loved Scala for web and systems/distributed development, but missed core of what's needed for the general workflow of a data scientist

- Stuck to data munging in Python

- Analytics/visualization in R

THIS WORKFLOW...

- What does this analyst workflow look like:
  - Get/**munge** data: apis, csvs, binary protocols, XML files, maybe already processed database
  - **Explore** when unfamiliar (arbitrary manipulations, analyses, visualizations on subsets)
  - Build **model** for problem (when understood) on structure/features of data (lately bound as possible)
  - **Production**???

remember, data scientist specialises in nothing, not a finance expert, a blah expert

we work in areas where we have few a priori assumptions about the distributions

IN SHORT...

- **Interactive** environment that promotes code re-use
- A good data structure for exploring, subsetting, transforming, **heterogeneously typed columnar data**
- A sensible **numeric abstraction** that promotes precision, performance and reusable code
- Access to a decent library for easily and variously **visualizing** this data structure
- A **standard statistical library** of well-tested and performant methods/models

THE SPREADSHEET

- The spreadsheet or frame is the fundamental way we have to work with data.
- This is not the matrix. Heterogeneously typed at this exploratory phase.
- Historically contingent (Excel...), but also incredibly natural for exploratory work.
- Great when one wants both composability (chaining together statistical transformations and visualization) and exploration (lately bound data model)

Wednesday, October 23, 13

i think point two is maybe the one I should clarify the most---everything else is pretty self-evident, i think. excel is awful, but it does some things right. for exploratory work...

TIDY DATA

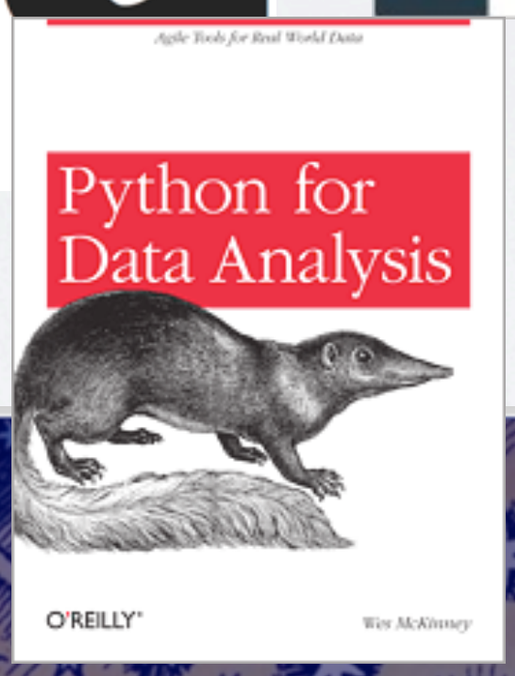
- **“Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: variables are stored in columns, observations in rows, and a single type of experimental unit per dataset.”**

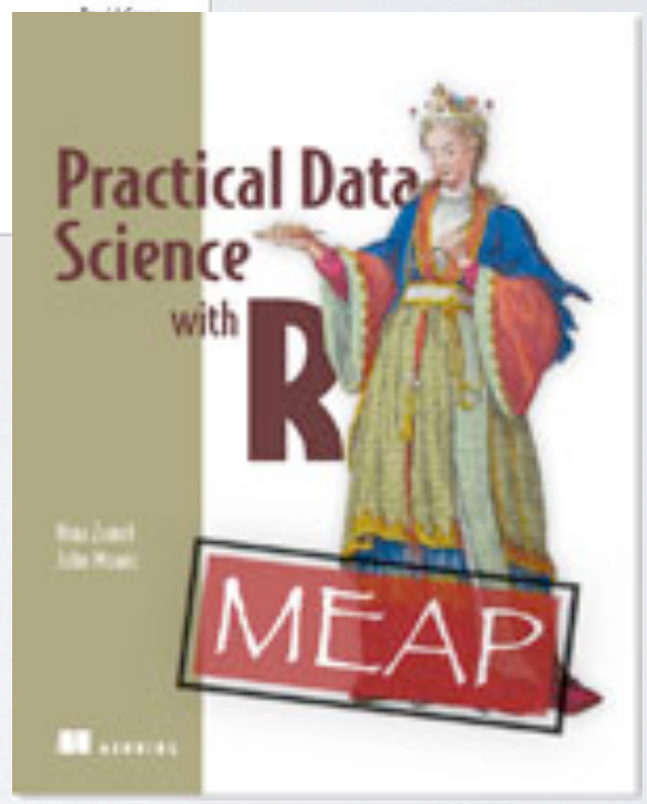
Hadley Wickham

- the point is that this makes for the cleanest and ideal way to build up such a consistent environment

- But obviously statistical/visualization libraries will ultimately need to be able to pull out to high performant matrix data structure

OUR CONTEMPORARIES





- Obviously these bare criteria are met by an ever-increasing number of languages today.

- Important that we be able to assess languages along this dangerous border between the objective (what tools exist, how mature, history of error) and subjective (experience thereof)

- Given the terrible proliferation of new languages and libraries every week, new libraries, it's important that we be able to assess languages along this dangerous border, talking about the experience, between, let's say, the objective (what tools exist, how mature, history of error) and subjective (structure of experience)

- Alex Payne's recent talk "Noone Got Fired for Using Java" is a beginning in this direction

- Must look at the way syntax, semantics and community interrelate

A REALLY HARD PROBLEM

- "[Computer scientists] fully realize that in our work, more than perhaps anywhere else, appropriate **notational conventions are crucial**, but also, we suffer more than anyone else from the general misunderstanding of their proper role. The problem is, of course, quite general: each experienced mathematician knows that **achievements depend critically on the availability of suitable notations**. Naïvely one would therefore expect that the design of suitable notations would be a central topic of mathematical methodology. Amazingly enough, the traditional mathematical world hardly addresses the topic. This amazing silence of the mathematicians on what is at the heart of their trade suggests that **it is a very tough job to say something sensible about it**. (Otherwise they would have done it.) And I can explain it only by the fact that the traditional mathematician has always the escape of natural language with which he can glue his formulae together. Here, the computing scientist is in a unique position, for **a program text is, by definition, one hundred percent a formal text. As a result we just cannot afford not to understand why a notational convention is appropriate or not.** Such is our predicament."

- "And this brings me to **my final hope**: before I die, I hope to understand by which virtues the one formal notational technique serves its purpose well and due to which shortcomings the other one is just a pain in the neck."

Obituary

Edsger Dijkstra

Pioneering computer programmer who made his subject intellectually respectable

Jack Schofield

The Guardian, Monday 19 August 2002 04.39 EDT

PHENOMENOLOGY OF THE DATA SCIENTIST

- That is, it is really important that we work to assess the **experience** of one doing their job in whatever environment, with whatever abstractions, we provide---such as I think we are more prone to do with APIs than languages
    - Largely because we're **often stuck** with a language depending on niche, legacy
    - But more important---languages, even formal, are more **mediums** than tools
- The question is what can make for the optimal experience of this unknown world of data that is our provenance, letting us **turn our archives into decisions**.

Wednesday, October 23, 13

- In short, we should work towards be something like a phenomenology of the data scientist
- somehow this is especially important, here, as there just isn't any other way to access these large numeric (or, at least, somehow quantifiable) archives. i don't need twitter to talk to a friend, but the interface and way we interact with data is really important... excel is obviously an important piece of software to keep in mind that has in fact changed the world given defficiencies of its model (this huge crazy unversion controlled, highly mutable, completely unsafe, but somehow beautiful and perfect and really powerful window)
- This is one part notational (how the syntax and semantics of a language can be leveraged to escape the arbitrary) and another tooling, what exists, and the infrastructure to support it.

SOME WITTGENSTEINIAN COROLLARIES

- "The limits of my language are the limits of my world."

SOME WITTGENSTEINIAN COROLLARIES

- "The limits of my language are the limits of my world."

- "The **world of the happy** is quite different from the **world** of the unhappy."

CRITERIA

- **Predictable** and **Concise** APIs
- **Consistency** of data structure inputs and outputs
- Easy to **reason** about
- **Performance**
- **Correctness**
- **Works** without error

Wednesday, October 23, 13

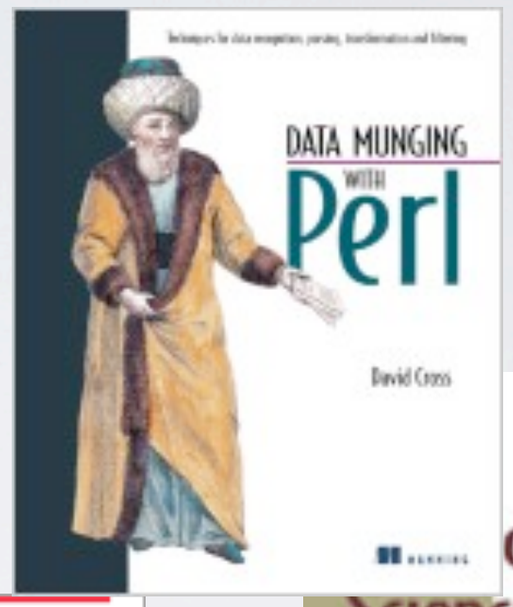
ok, so maybe we haven't solved this problem yet, "programming" is still 99% cargo cult, but what kind of criteria do we have...

and obviously we've stumbled down into the abyss, these aren't at all categories worth the name of djikstra

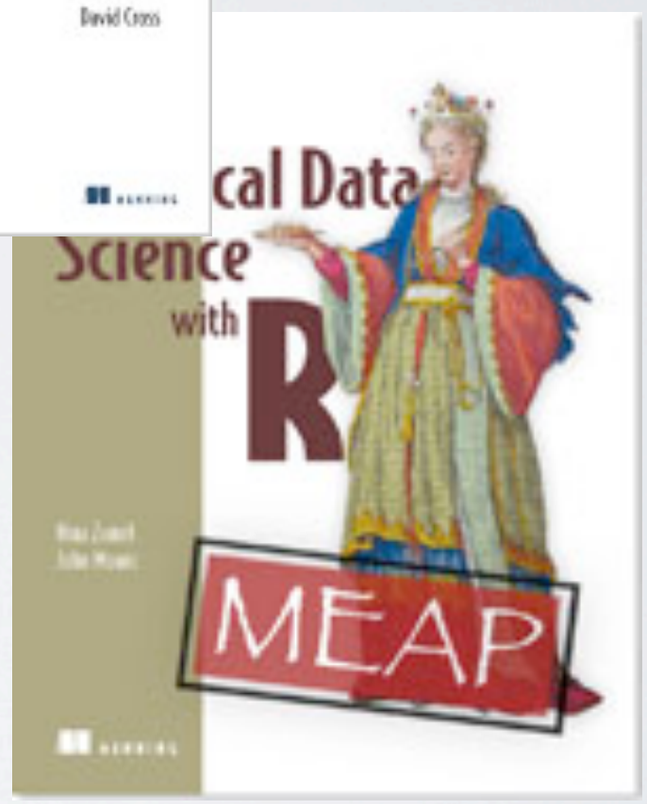
but this is a start

CONTENDERS


julia


DATA MUNGING
WITH
Perl
David Cross


Python for
Data Analysis


cal Data
Science
with
R
MEAP

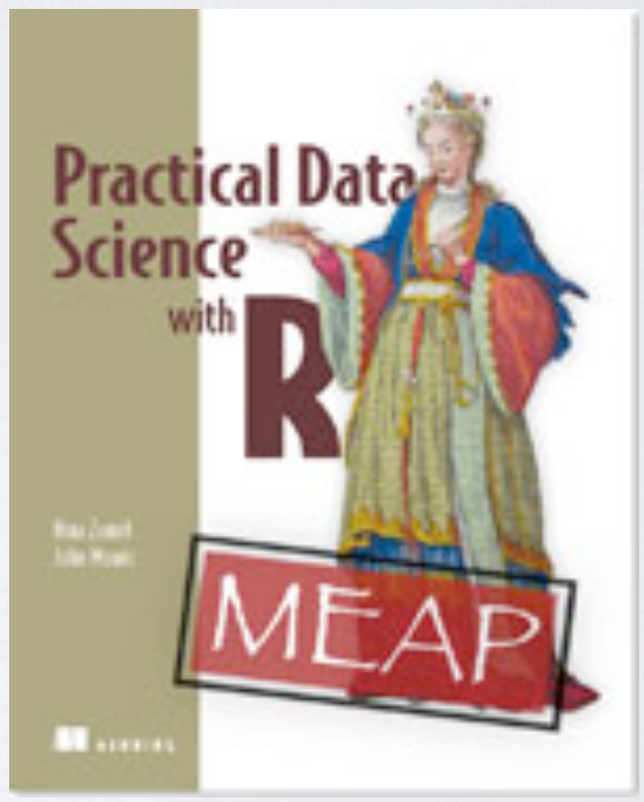

INCANTER
Data Sorcery

CONTENDERS


julia

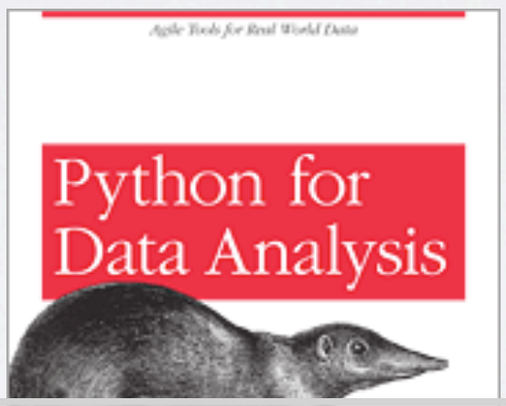

Python for
Data Analysis

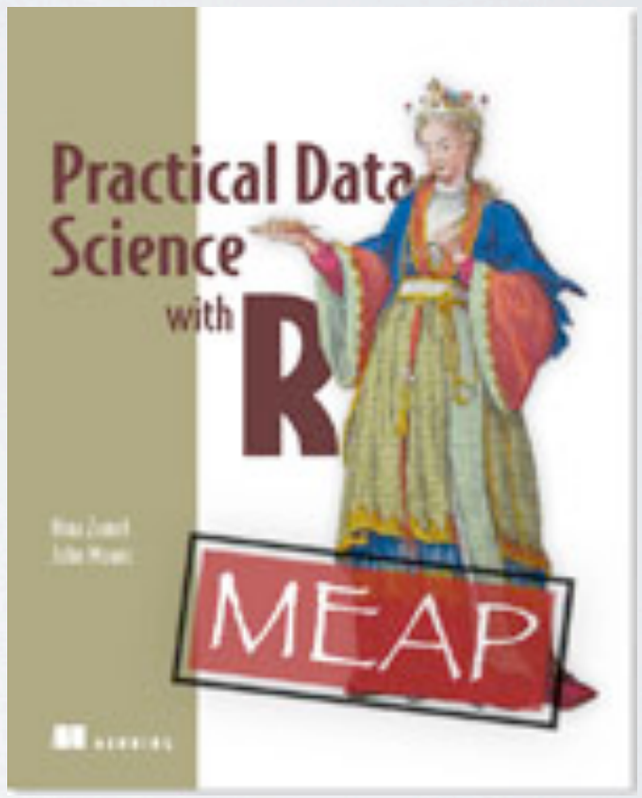

INCANTER
Data Sorcery


Practical Data
Science
with
R
MEAP

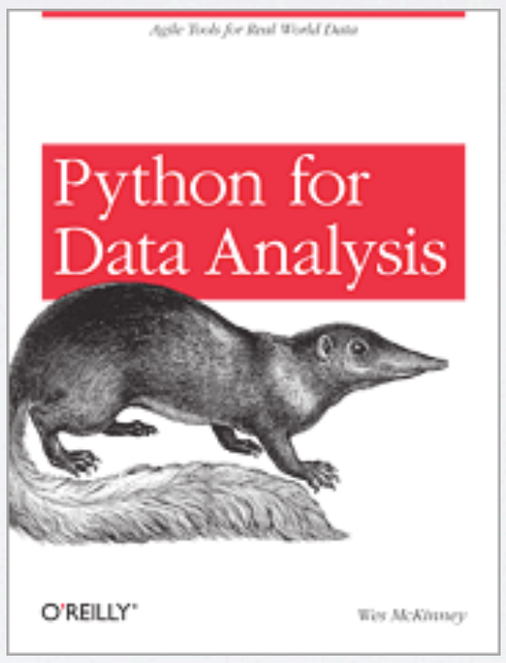
CONTENDERS

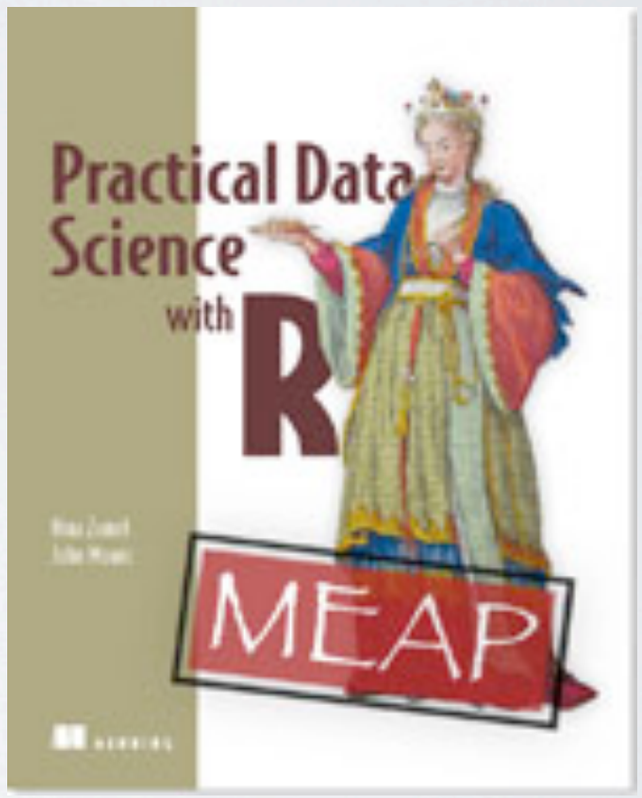

INCANTER
Data Sorcery


Agile Tools for Real World Data
Python for
Data Analysis

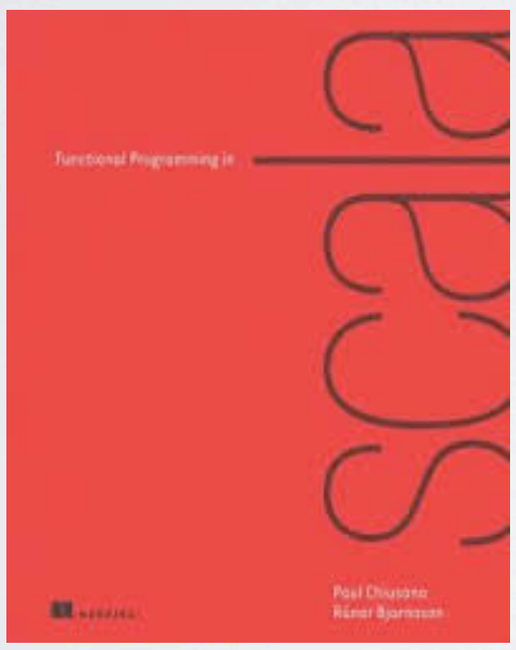

Practical Data
Science
with
R
MEAP

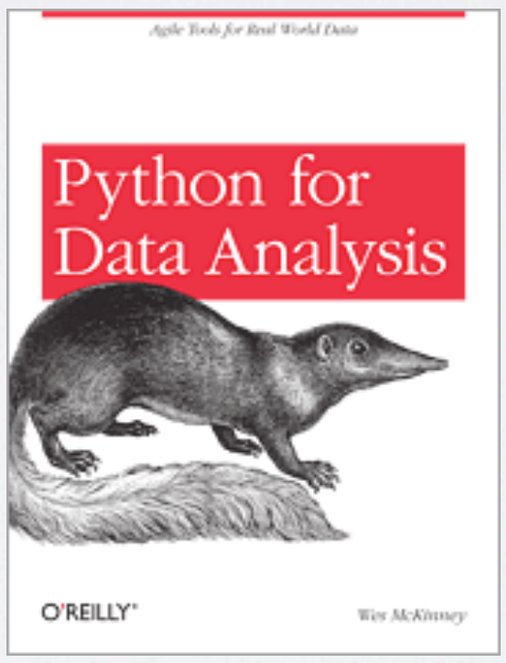
CONTENDERS

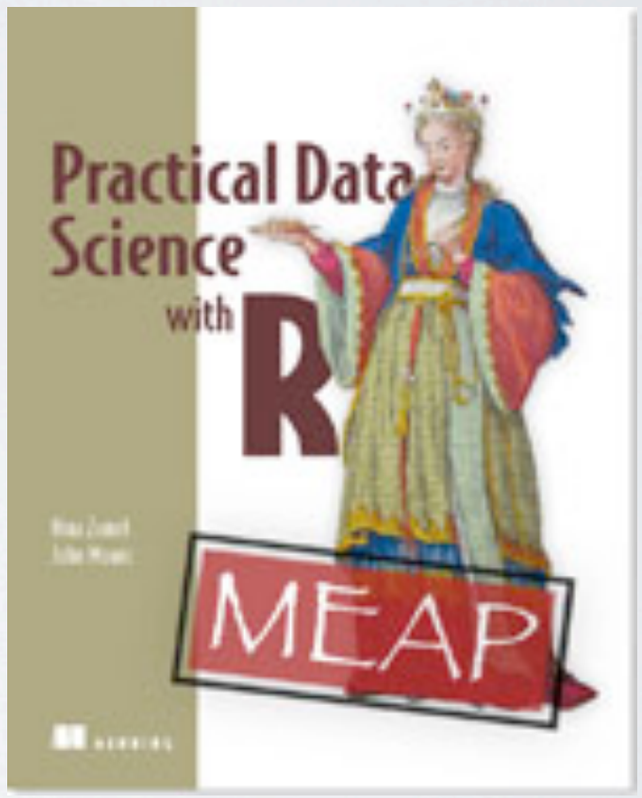

Agile Tools for Real World Data
Python for Data Analysis
O'REILLY
Wes McKinney


Practical Data Science with R
MEAP

CONTENDERS


Functional Programming in
Scala
Paul Chiusano
Rúnar Bjarnason


Agile Tools for Real World Data
Python for Data Analysis
O'REILLY
Wes McKinney


Practical Data Science with R
MEAP

PYTHON

- Dynamic language with nice REPL, interactive workflow via IPython
- Pandas now provides a nice frame structure
- Sane numeric types with pretty good performance via numpy
- Visualization picture pretty weak
- By now a pretty good collection of core ML/statistical tools

ASSESSMENT

- Thus internal to a library high standards of functionality, consistently named, straightforward APIs
- Obviously not structured from the ground up around analytics
- Panda data structure isn't the input to all the machine learning libraries
- As a consequence, never know the particular inputs to statistical methods across libraries

Wednesday, October 23, 13

but, it's this consistency question, correctness, reason about that starts to get so messy.

it's just obviously not a language designed around analytics, and we just don't have the power to build enforceable abstractions (types, etc) at a really good level to remedy this, and so left with a hodgepodge of libraries that fulfill numerous niches.

but composing inputs outputs, is /almost/ as bad as cross-language

i mean, i guess places really do build systems in dynamic languages with large teams of people, and isn't too outlandish...

R

- Similarly dynamic language, nice interactive environment
- Terrific frame data structure at root of language
- Similarly consistent numeric abstraction
- Beautiful and modern visualization tools like ggplot
- Every statistical method ever

ASSESSMENT

- Under the hood it is this strange (APL + Scheme?) and sometimes surprisingly nice dynamic, functional language (macros, lazy evaluation, first class functions, etc) --- used for things like nice regression DSLs:
  - `fit <- lm(y ~ x1 + x2 + x3, data=mydata)`
- All methods built around frame, lots of work, for example by Hadley Wickham, to even further consolidate community standards of inputs, outputs
- Grep? x <- 2, C(...), x[[[1]]][[[2], x[“]]”??!
- ggplot2 semantics unclear, uses ‘+’ to compose visualizations but definitely not commutative...
- So much academic abandonware... Tools that just don’t work...
- Integrating with production systems...

Even if it’s unified, it’s just such a free for all, not at all done by developers, no language tools for correctness, and, of course, no one really knows best practices for writing apl-scheme

so, whereas python has the problem that most libraries are well done, we just don’t have the consistency across projects nor really the language features to support this

here, we have a unified environment, but crazy language that makes it difficult to write good code

THE COMMON ERRORS

- Insanely inconsistent/unpredictable APIs
- Difficult to reason about
- Performance
- Building systems in the large

in both cases, the experience is so ad-hoc, so debilitating, so much time with cheat sheets, thousands of api doc windows open (or brute force memorization of special quirks of some library)

FUNCTIONAL PROGRAMMING!

- FP ought to be the eradication of the arbitrary, the introduction of the compile time error, beautiful compile time optimizations for various run time environments

- As we've seen today, all these wonderful numeric abstractions from higher mathematics that our languages can actually express/enforce.

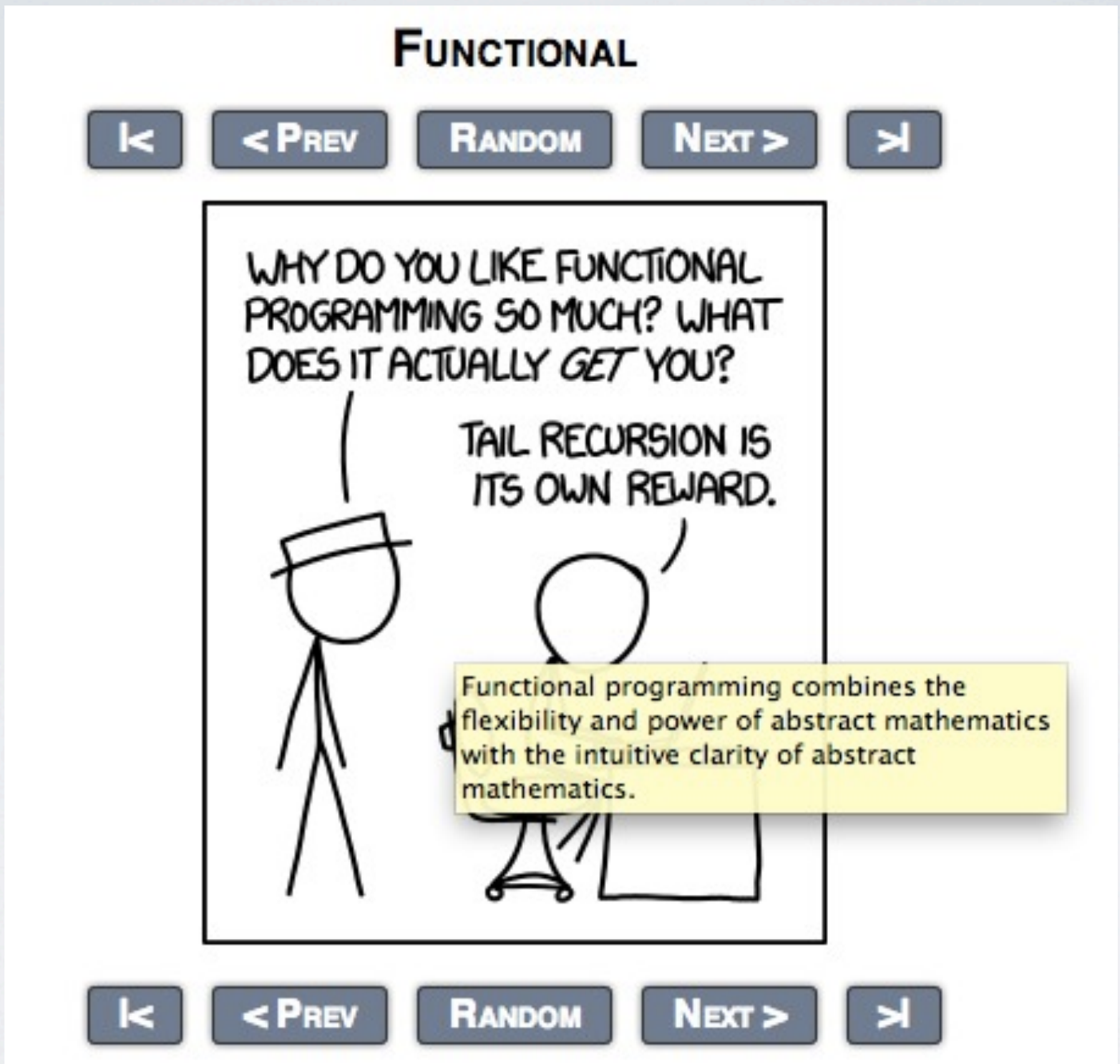
- We should be able to have the flexibility of R/Python/etc with the added bounty of better abstractions, better performance, and compile-time guarantees.

- We should be able to leverage types to provide more consistent, predictable APIs.

Wednesday, October 23, 13

ok, we all watched paul phillip's talk, this isn't quite the dream we actually live in, right

but my talk is meant to be motivational, right, fruitlessly trying to will a happy data science world into existence where we can really focus on understanding the domain we're trying to learn and produce insights about, rather than beating against bad APIs



but when you're actually trying to do abstract mathematics... you know, this is something we should get right. should obviously be able to beat java, but also little reason why we can't combine the flexibility people demand of R/python (i mean, at least the flexibility we "really need"---heterogeneous, frames, interactivity, etc---freedom is slavery per paul p's addage) with sensible, uniform abstractions and correctness,

THE CONTEMPORARY SCENE

- And we see this, fragmented:
  - Saddle
  - Spire
  - Breeze/Nak/ScalaNLP
  - Scala Notebook
  - Algebird

THAT IS...

- Frame data structure with full API modeled after Pandas
- Performant numeric abstractions for generic programming
- Great NLP and ML library, performant matrix/vector data structure and linear algebra library
- Improvements on REPL, reproducibility, visualization with D3
- Leveraging these abstractions for large scale distributed analytics

SPIRE

- Seen (and will see yet) many examples of this library's power.

- Outstanding, so many thumbs up.

- For example, currently working on a visualization library and Spire has been a great boon in terms of making it possible to develop statistical methods that operate on both big decimal and double. Even useful just for providing a uniform standard library of log, etc, atop all numeric types.

SADDLE

- Great first step towards a frame structure in Scala
- Sometimes inconsistent APIs (Ix vs Index...)
- Not so great in the heterogeneous case ([RowType, ColumnType, Any])
- Doesn't support big decimal out of the box

Wednesday, October 23, 13

I've harped on this heterogeneous case because I really do think this is essential. now, it isn't absurd that they took the approach, right, I mean, what should the type be, here? they do provide a method to extract a typed subset (all columns of type T in frame) but in general, lots of explicit matching, casting, etc.

we don't have, naturally, real record types (obviously a case class would be horrifying here)

so, one reason i harped on this so much in my preparation, is that we do now have a way---i think this is one of the most natural use cases for... shapeless! which might be a bit controversial, might think too much flexibility, but i think this a moment to really showcase the library in a way that could have good traction

SCALA NLP

- Former Scalala.
- Very fast tool for doing core linear algebraic tasks
- Beginnings of a core library for statistical methods
- Focused on the NLP side of things

MORE RESEARCH

- Obviously the main difficulty, the one so difficult to overcome with respect to R, is simply the quantity, especially of cutting edge, statistical libraries

- Also worth mentioning Scalabha (UT Austin), various great tools from University of Maryland and Topic Modeling Toolkit (Stanford)

- Surprising amount of research and tools coming out of major research universities done in Scala, especially NLP

SCALA NOTEBOOK

- Stepping up the REPL with modern web tools
- Especially great for collaboration
- Great presentation by Nathan Hamblen
- Much more work on usability (difficult to start up a new project)

ALL GOOD?

- No modern, sane visualization tools (especially that operate on any of these data structures)
- Almost everything is working with doubles which, in some cases, just can't provide the needed accuracy
- Improvements to heterogeneously typed Saddle frame (Shapeless?) and integration with Spire and Breeze
- Duplication of efforts: Spire working towards linear algebra library, Breeze duplicating numeric abstractions, Saddle reproducing matrix operations
  - Everyone reduplicates all of this specialization logic, especially, which is so difficult to get right, crucial to be able to rely on libraries where this is done, clean up code
- Places where types could be used to promote a better API, especially leveraging these existing abstractions

THE BAR IS REALLY REALLY REALLY LOW

Wednesday, October 23, 13

probably could fit some more reallys in there

because it is.

and i think most of the groundwork and momentum are there, that with a bit of a push we can raise it a bit (maybe even a lot)

(I mean, spark just got 15 millions dollars from anderson horrowitz the other day...), just one more effort, scala, if you want to attract data scientists

to be able to leverage these extractions at the level of small and twitter-size data would be huge.

IN SHORT

- The pieces are there for Scala to provide a really first class environment for statistical computing, from the exploratory to production
- Work on this division of labour, focus on one thing, outsource to this ecosystem
  - The art of coordinating multiple projects over twitter in the age of github
- It's in all of our interest, as Scala developers, to promote this state of affairs (hiring, our own lives, etc)
- This is chance to really get more widespread adoption for some of these abstractions

we really owe it to ourselves to put in the extra effort to take the responsibility and try and see how these tools fit together

i said this before, and obviously not a diss on github, it's great all this code is out there, really changed the landscape, but it's really important that we put effort into thinking how our projects relate to others.

anyway, as I said at the beginning, this talk was a bit rushed and I wasn't able to release any code with it, but these are some of the kinds of problems i'm thinking about in my job with pellucid and i hope to have some, at least proofs of concepts out, in the coming month

THANKS! QUESTIONS?

- **@agentcoops**
- Let's have some discussions and get hacking!
  - https://groups.google.com/forum/#!forum/scala-analytics
- And, yes, we're hiring at Pellucid, come talk to me

- i'm also trying, as a first step towards this, to set up a google group to discuss this problem
- if you're interested in the other coast