SCABIO – a framework for bioinformatics algorithms in Scala

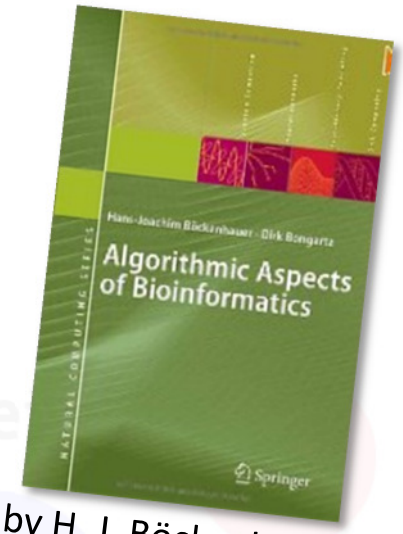
Markus Gumbel

Institute for
Medical Informatics

Mannheim University
of Applied Sciences / Germany

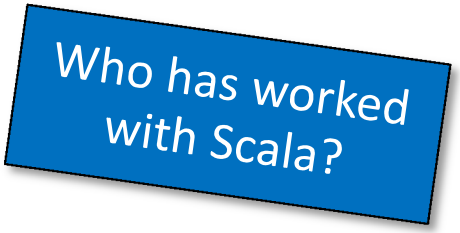
SCABIO (Scala algorithms for bioinformatics) is...

- a framework for bioinformatics algorithms written in Scala
- used primarily for education (since 2010)
  - introduced for a bioinformatics lecture & lab
  - contains mainly standard algorithms
- open source but not yet *released* (i.e. no announcements etc.)
  - premiere today!
  - `https://github.com/markusgumbel/scalabioalg`
  - Apache License Version 2.0



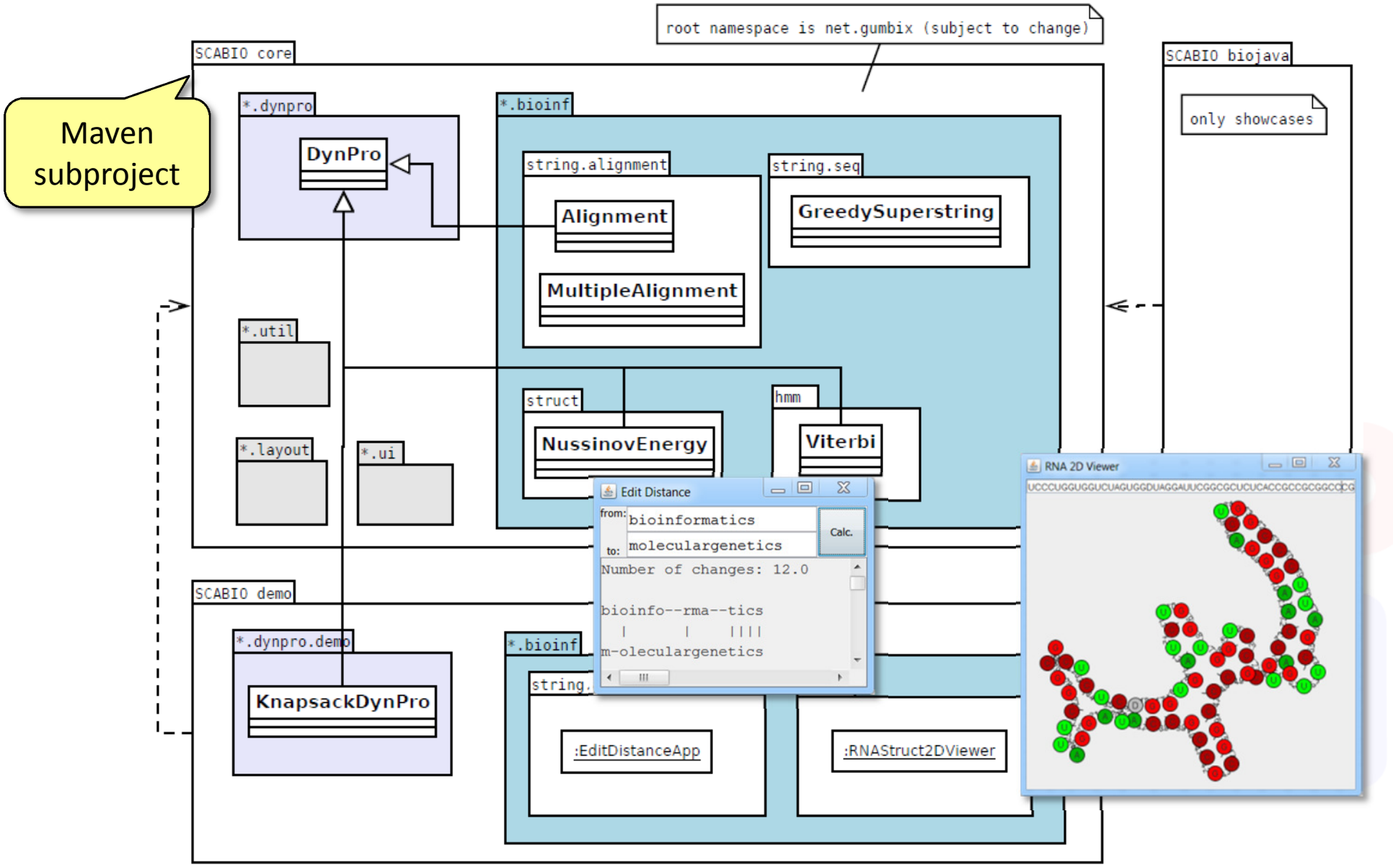
by H. J. Böckenhauer, D. Bongartz

Why a new framework, why Scala?



- For education
  - generic dynamic programming package to demonstrate optimization principle of many bioinformatics algorithms
  - it's fun!
- Scala
  - seamless integration with other JVM-based frameworks like BioJava or bioscala
  - combines functional and object-oriented paradigms
  - suitable for concurrent algorithms (e.g. AKKA)
  - built-in support for domain specific languages etc.

Modules, Packages and Core-Classes



Prerequisites / Installation

- Little dependencies
  - Java 5 + Scala 2.9 for all modules
  - Apache commons-math library (`*.stat.Frequency` used just one time) in module `core`
  - Java Universal Network/Graph Framework (JUNG) in module `demo` (RNA viewer)
  - BioJava 3 in module `bioscala`
- Right now no real installation procedure for *users*
- For developers: Maven 2

SCABIO's dynamic programming (DP) package

- Algorithm paradigm for optimization (min/max)

> Bellman's Principle of Optimality
>
> "If I have an already optimal solution at step *n* - 1 and I choose a decision for step *n* which is optimal relative to *n* - 1 then I get the optimal solution for step *n*."

- (intermediate) results of the steps are stored in a table
- Solution (=decisions) can be derived from the table

- A DP-based algorithm requires the specification of the following information for each state
  - What decisions can result in being in this state?
  - What are the previous states depending on a decision?
  - What is the value (cost) to get to this state when a specific decision was made?

Hochschule Mannheim

Example: Pairwise alignment (global, semi-global, local)

- Table size $n \times m$
- decision(*idx*)
- previousStates(*idx*, *d*)
- value(*idx*, *d*)

```
class Alignment(val s1: String, val s2: String, val mode: AlignmentMode,
... extends DynPro[AlignmentStep] ... {

 def n = s1.length + 1
 def m = s2.length + 1

 def decisions(idx: Idx) = {
   if (idx.i == 0 && idx.j == 0) Array(NOTHING)
   else if (idx.i == 0) Array(INSERT)
   else if (idx.j == 0) Array(DELETE)
   else Array(INSERT, DELETE, BOTH)
 }

 def prevStates(idx: Idx, d: AlignmentStep) = d match {
   case NOTHING => Array()
   case _ => Array(idx + deltaIdx(d))
 }

 def value(idx: Idx, d: AlignmentStep) = d match {
   case NOTHING => 0
   case DELETE => if (idx.j == 0 && allowLeftGapsString2) 0 else values(d)
   case INSERT => if (idx.i == 0 && allowLeftGapsString1) 0 else values(d)
   case _ => score(s1(idx.i - 1), s2(idx.j - 1))
 }
...
}
```

Just to give an idea of the LOCs

Example: Pairwise alignment (global, semi-global, local)

- Table size $n \times m$

```
sim = 0.0

ACG-T.
|||  |
ACGATC

    |  0    1    2    3    4     5     6
    |  -    A    C    G    A     T     C
- -|---- ---- ---- ---- ---- ----- -----
0 -| 0.* -2.  -4.  -6.  -8.  -10.  -12.
1 A|-2.   1.* -1.  -3.  -5.   -7.   -9.
2 C|-4.  -1.   2.*  0.  -2.   -4.   -6.
3 G|-6.  -3.   0.   3.*  1.*  -1.   -3.
4 T|-8.  -5.  -2.   1.   2.    2.*   0.*

BBBIBI
```

Decisions for optimal solution (B = both, I =Insert)

```
class Alignment(val s1: String, val s2: String, val mode: AlignmentMode,
... extends DynPro[AlignmentStep] ... {

 def n = s1.length + 1
 def m = s2.length + 1

 def decisions(idx: Idx) = {
   if (idx.i == 0 && idx.j == 0) Array(NOTHING)
   else if (idx.i == 0) Array(INSERT)
   else if (idx.j == 0) Array(DELETE)
   else Array(INSERT, DELETE, BOTH)
 }

 def prevStates(idx: Idx, d: AlignmentStep) = d match {
   case NOTHING => Array()
   case _ => Array(idx + deltaIdx(d))
 }

 def value(idx: Idx, d: AlignmentStep) = d match {
   case NOTHING => 0
   case DELETE => if (idx.j == 0 && allowLeftGapsString2) 0 else values(d)
   case INSERT => if (idx.i == 0 && allowLeftGapsString1) 0 else values(d)
   case _ => score(s1(idx.i - 1), s2(idx.j - 1))
 }
...
}
```

Just to give an idea of the LOCs

Example: Viterbi-algorithm for pattern recognition (e.g. CpG islands)

- Table size $n \times m$
- decision(*idx*)
- previousStates(*idx*, *d*)
- value(*idx*, *d*)

```
class Viterbi(val s: Array[Char], val alphabet: Array[Char],
 val states: Array[Char],  val transP: Array[Array[Double]],
 val emmP: Array[Array[Double]])  extends DynPro[Int]
 with Backpropagation[Int] with MatrixPrinter[Int] {

  def n = s.length + 1
  def m = states.length

  def decisions(idx: Idx) = {
    if (idx.i == 0) (0 to 0).toArray
    else (1 to states.length).toArray
  }

  def prevStates(idx: Idx, d: Int) =
    if (idx.i > 0) Array(Idx(idx.i - 1, d - 1)) else Array()

  def value(idx: Idx, dState: Int) = (idx.i, idx.j) match {
    case (0, jState) => log(transP(0)(jState + 1))
    case (iChar, jState) => {
      val idxS = alphabet.indexOf(s(iChar - 1))
      val e = emmP(jState + 1 - 1)(idxS) // state, char
      val t = transP(dState)(jState + 1)
      log(e * t)
    }
  }
  ...
}
```

Example: Nussinov-algorithm for secondary RNA-structures

```
class NussinovEnergy(val s: String)
  extends DynPro[NussinovDecision] {

  def n = s.length
  def m = n

  def decisions(idx: Idx) = {
    if (idx.j - idx.i < 1) Array()
    else {
      val splitList = if (idx.j - idx.i > 1) {
        val h = for (k <- (idx.i + 1) to (idx.j - 1)) yield {
          new NussinovDecision(SPLIT, idx, k, ' ', ' ')
        }
        h.toList
      } else Nil
      val otherList = for (state <- NussinovState.values;
                           if (state != SPLIT)) yield {
        new NussinovDecision(state, idx, 0, s(idx.i), s(idx.j))
      }
      val u = splitList.toList ::: otherList.toList ::: Nil
      u.toArray
    }
  }
```

Again: Just to give an idea of the LOCs

```
  def prevStates(idx: Idx, d: NussinovDecision) = d.move match {
    case EXTEND_BEGINNING => Array(Idx(idx.i + 1, idx.j))
    case EXTEND_END => Array(Idx(idx.i, idx.j - 1))
    case PAIR => Array(Idx(idx.i + 1, idx.j - 1))
    case SPLIT => {
      Array(Idx(idx.i, d.k), Idx(d.k + 1, idx.j))
    }
  }

  def value(idx: Idx, d: NussinovDecision) = d.move match {
    case PAIR =>
      val e = (s(idx.i), s(idx.j)) match {
        case ('A', 'U') => -2.0
        case ('U', 'A') => -2.0
        case ('C', 'G') => -3.0
        case ('G', 'C') => -3.0
        case ('G', 'U') => -1.0
        case ('U', 'G') => -1.0
        case _ => 0.0
      }
      if (idx.j - idx.i > hairPinLoopSize) e else 0
    case _ => 0
  }

  override def extremeFunction = MIN
}
```

Example for BioJava 3 Integration

cast to BioJava Class

```
object DNASeq {
  implicit def seq2string(seq: DNASequence) = seq.toString

  def main(args: Array[String]) {
    // AAI44185: Homo sapiens (human) NR1H4 protein
    // AAL57619: Gallus gallus (chicken) part. orphan nucl. receptor FXR
    val fxr1 = getEMBLBank("AAI44185").values.iterator.next
    val fxr2 = getEMBLBank("AAL57619").values.iterator.next
    val alignFxr = new Alignment(fxr1, fxr2, GLOBAL)
    println("sim = " + alignFxr.similarity)
    println(alignFxr.makeAlignmentString(alignFxr.solution))
  }

  def getEMBLBank(accId: String) = {
    val url = new URL("http://www.ebi.ac.uk/ena/data/view/" +
            accId + "&display=fasta")
    val ios = url.openStream()
    val map = FastaReaderHelper.readFastaDNASequence(ios)
    ios.close()
    map
  }
}
```

How can a `DNASequence` be a string?

Discussion & Outlook

- Source-code very concise
- How will SCABIO compete in real life bioinformatics applications?
- Future work
  - Make project *more* open source
  - More test cases
  - More documentation
    - manual in work (> 12 pages)
    - scaladoc on-line
  - Implement concurrent algorithms

Hochschule Mannheim

Thank you! Questions?

- SCABIO
  - `https://github.com/markusgumbel/scalabioalg`
  - Apache License Version 2.0
- Contact
  - `m.gumbel@hs-mannheim.de`
  - `http://www.mi.hs-mannheim.de/gumbel`
- I am really looking forward for discussions with the BioJava, bioscala, ... folks
- There's also a poster