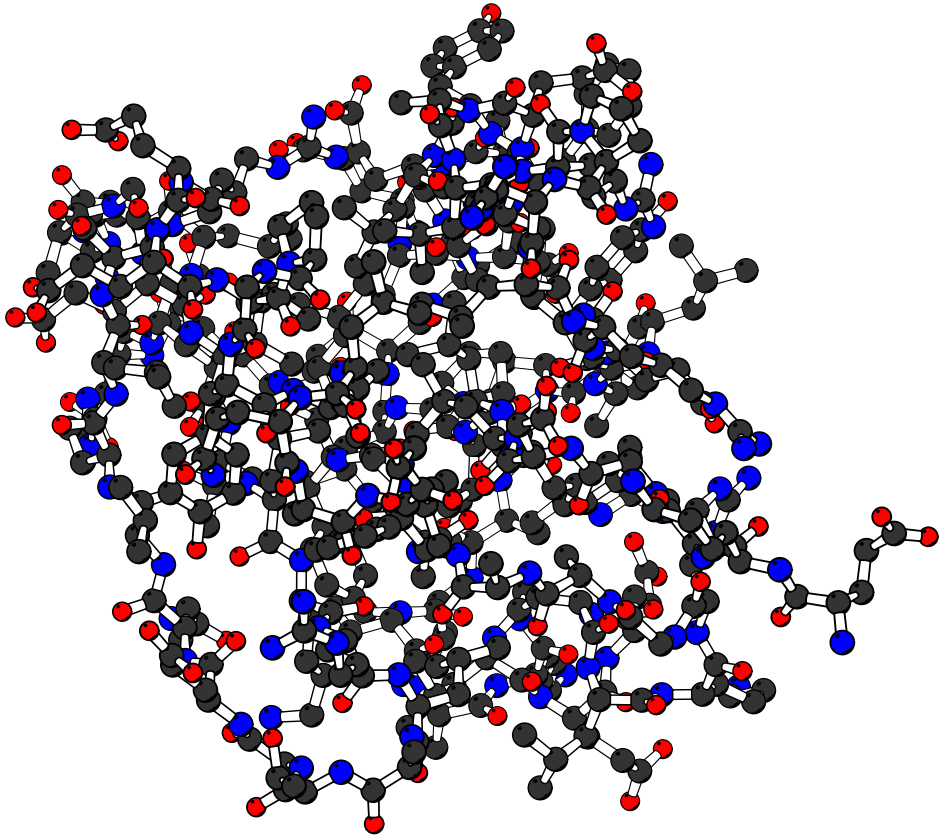
Evolving Perl code for protein secondary structure prediction

Bob MacCallum
Stockholm Bioinformatics Center
Stockholm
Sweden

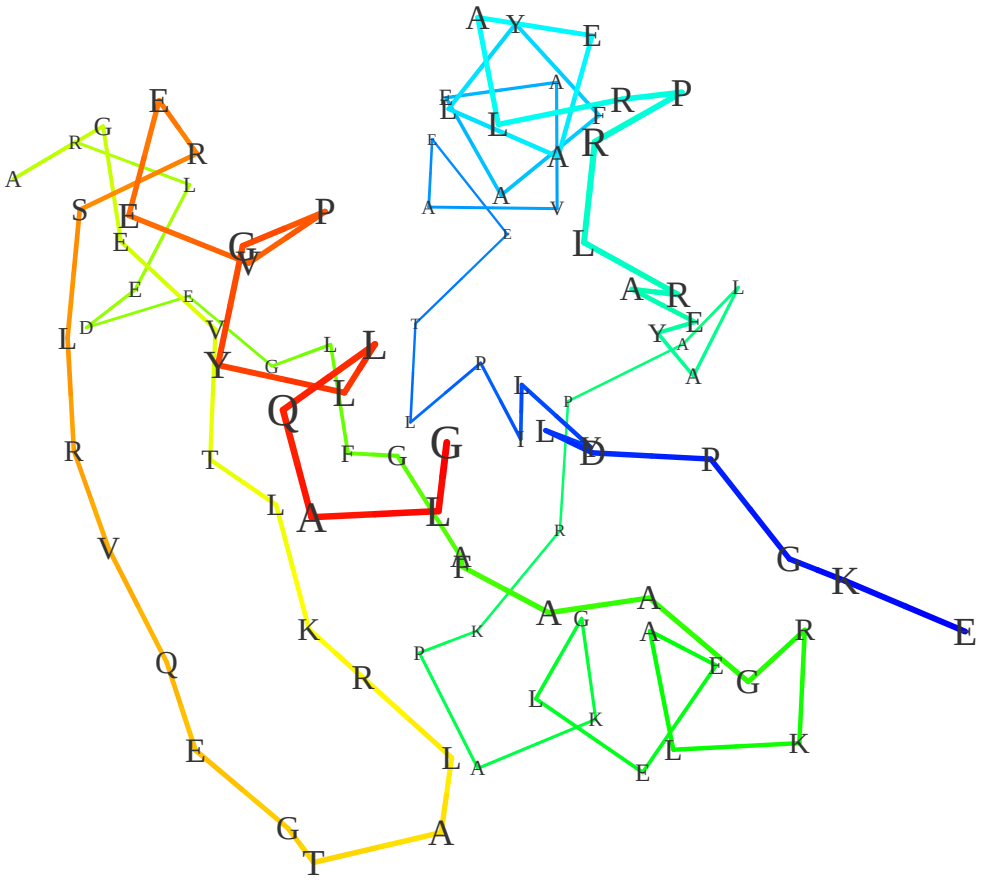
Protein structure

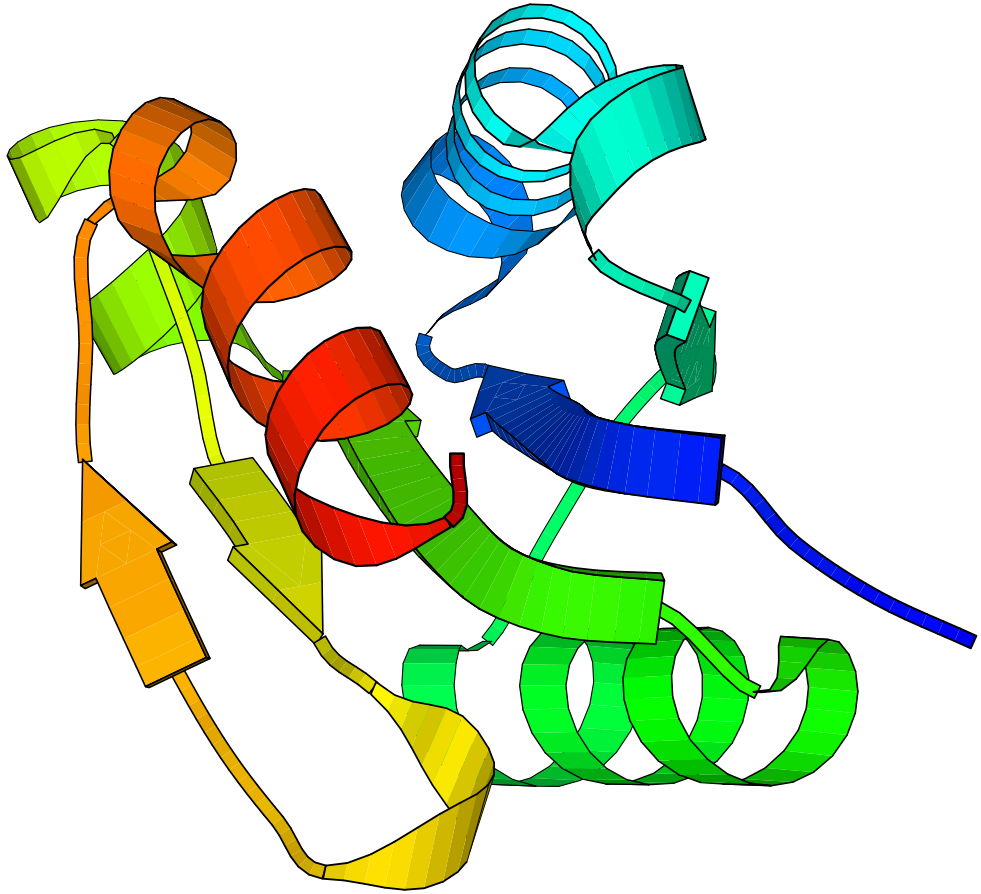
...AGCTGAAGCTCCTGTATTATCCCTTTTAGC...

EKGPDLYLIPLTEEAVAEAFYLAEALRPRLRAEYALAPRKPAKGLEEALKRGAAFAGFLGEDELRAGEVTLKRLATGEQVRLSREEVPGYLLQALG



Protein structure





Protein structure, function and evolution

Structure tells us a lot about molecular mechanisms:

- where other proteins and molecules bind or interact
- enzyme chemistry
- physical features (hinges, channels, rotors etc)

And evolutionary history:

> Evolution over long periods of time can change amino acid sequences beyond recognition.
>
> But structures remain basically the same.

Structure prediction

Physical techniques (X-ray crystallography, NMR) are expensive, slow, and results are not always guaranteed.

Quicker to predict the structures computationally...

the easy way - "copy" it from a related protein of known structure

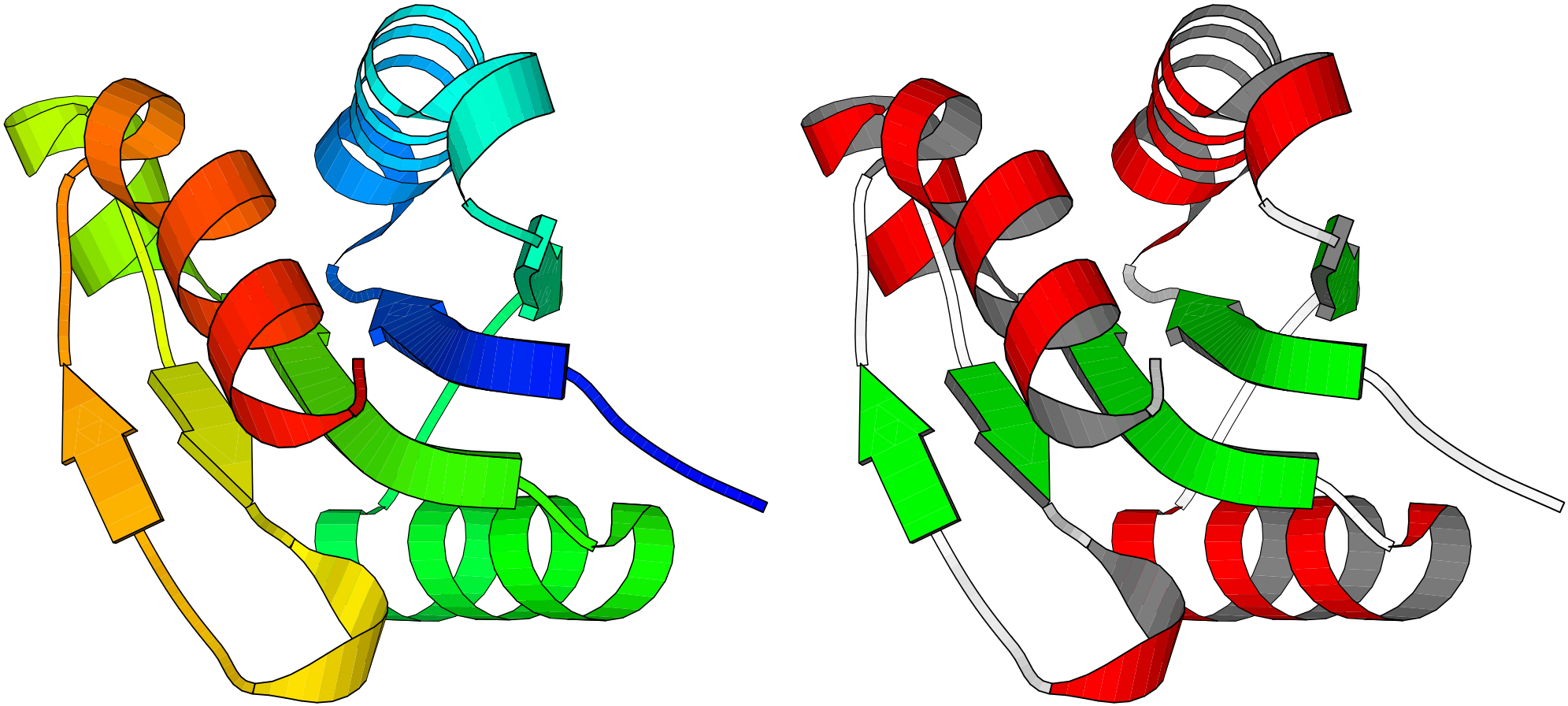
the hard way - build it from scratch with some physical or statistical model

Can't always take the easy route, but high-throughput structural genomics projects help

Intellectual satisfaction in solving the harder problem

Would be nice to design novel proteins

Secondary structure prediction (SSP)



EKGPDLYLIPLTEEAVAEAFYLAEALRPRLRAEYALAPRKPAKGLEEALKRGAAFAGFLGEDELRAGEVTLKRLATGEQVRLSREEVPGYLLQALG
CCCCCEEEEECCHHHHHHHHHHHHHHCCCCCEEECCCCCCHHHHHHHHHHCCCCEEEEECHHHHHHCEEEEEECCCCCEEEEECCHHHHHHHHHHC

H = Helix, E = strand (Extended conformation), C = Coil (or loop or nothing)

Current state-of-the-art in SSP

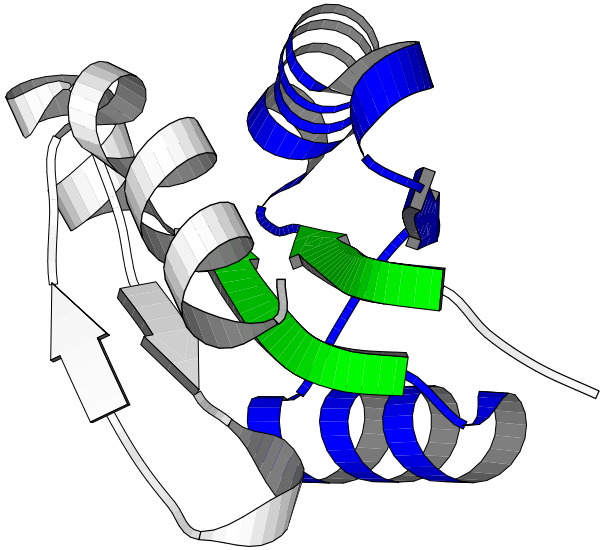
Mostly feed-forward neural-networks trained to predict H or E or C for each sequence position (residue) from windowed input:

$$Q_3 = \frac{\text{residues correct}}{\text{total residues}} \approx 76\%$$

(performance of predictors like PHD and PSIPRED)

Better SSP

Aiming for 100%? No, but about 90% would do nicely...



- folding brings distant residues close in 3D space
- want to use information far away in sequence
- local sequence information may be overridden by global context effects

long range interaction

EKGPDLYLIPLTEEAVAEAFYLAEALRPRLRAEYALAPRKPAKGLEEALKRGAAFAGFLGEDELRAGEVTLKRLATGEQVRLSREEVPGYLLQALG
CCCCCEEEEECCHHHHHHHHHHHHHHCCCCCEEECCCCCCHHHHHHHHHHCCCCEEEEECHHHHHHCEEEEEECCCCCEEEEECCHHHHHHHHHHC

Want to use: long range information and/or folding pathway

(of course 3D information would help...)

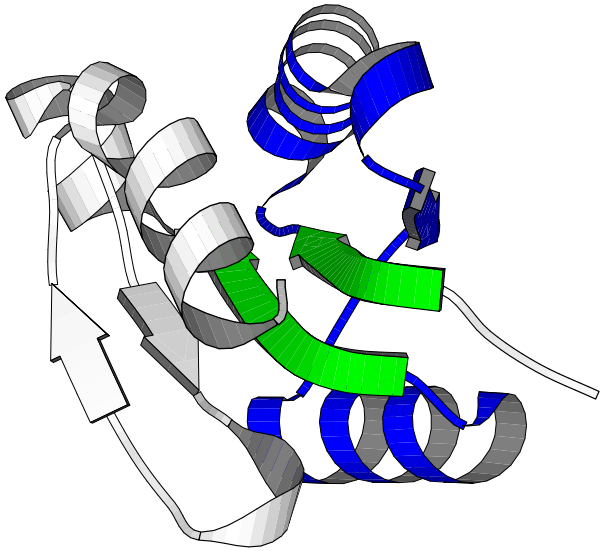
Tried and failed...

Bigger window - just adds more noise, also consider different length "inserts"

Recurrent NN's - Baldi *et al* have an interesting approach but $Q_3 \not> 76\%$

Global picture - Bystroff *et al* use an all-at-once HMM combining all known local sequence-structure patterns but $Q_3 \not> 76\%$

3D structure prediction - Baker's ROSETTA method?



My approach

Variable size solution

Self organised

Dynamic model/feedback

Regular expressions for long-range patterns

`[P][AT].{20,30}[TE][RN]`

Genetic programming... the tool of choice?

Pros

variable length/complexity solutions

can use high level constructs/grammar

we might be able to understand the solution

Cons

an Art more than a Science..?

GP the Perl way...

Powerful/easy string handling

Interpreted, eval()

Most bioinformatics people are Perl literate

Tree-as-hash representation/expansion...

Hash-to-disk synchronisation with tie()

Tree-as-hash expansion

```
$code = '{STAT0}';
$tree{STAT0} = 'if ({NUM1} > {NUM2}) { {STAT1} }';
$tree{NUM1} = '$x';
$tree{NUM2} = '7';
$tree{STAT1} = 'print "{STRING0}";';
$tree{STRING0} = '{STRING1} {STRING2}';
$tree{STRING1} = 'hello';
$tree{STRING2} = 'world';

while ($code =~ s/\{([A-Z]+[0-9]+)\}/$tree{$1}/g) {
  print "$code\n";
}
```

outputs:

```
if ({NUM1} > {NUM2}) { {STAT1} }
if ($x > 7) { print "{STRING0}"; }
if ($x > 7) { print "{STRING1} {STRING2}"; }
if ($x > 7) { print "hello world"; }
```

Grammar-as-hash definition

```
$functions{STAT}   = [ 'if ({NUM} > {NUM}) { {STAT} }',
                       'print "{STR}"' ];
$functions{NUM}    = [ '({NUM} + {NUM})', '{NUM} * {NUM}', '{NUMX}' ];
$functions{STR} = [ '{STR} {STR}', 'STRX' ];

$terminals{STAT}   = [ 'return;', 'exit;' ];
$terminals{NUM}    =
  $terminals{NUMX} = [ 1 .. 7, '$x', '$y' ];
$terminals{STR}    =
  $terminals{STRX} = [ 'hello', 'goodbye', 'world', 'mum' ];
```

Random individuals are generated by following a random path through the grammar.

Naturally terminated trees (no imposed depth limit) possible if you bias the grammar:

```
$functions{STR} = [ '{STR} {STR}', 'STRX', 'STRX', 'STRX' ];
```

Perl GP - main features

User-defined grammar

Strong typing

Population on disk

Parallel populations/migration

Tournament selection

Ageing

Per-node probability for crossover and mutation

x^6 size and time fitness penalties

No bloat

"Homologous" crossover

Same-size crossover

Macromutation: internal insert/delete, swap, copy, replace subtrees

Easy to evolve subroutines, object methods, GP parameters...

Regex based SSP

a randomly generated program

```
sub predict_secondary_structure {
  my $seq = shift;                    # input amino acid sequence
  my $predss = 'C' x length($seq); # initialise output with "Coil"

  scan($seq, "([HD][^KL][^F][^QFHIRVHKKM])([^P])([T])", \$predss, 'H');
  if ($seq =~ /(?:[^G]){1,}[KFY]/) {
    if ($predss =~ /.{1,}.{1,8}C{12,}/) {
      scan($seq, "([P][^A])(.{1,1}[^TC])()", \$predss, 'E');
    }
  }
  scan($seq, "()([^YN])()", \$predss, 'E');

  return $predss;
}
```

Input is **single, unaligned** sequence (c.f. PSIPRED, PHD).

The `scan()` function

Takes a regex of the form `(part1)(part2)(part3)` and (re)assigns secondary structure to subsequences matching `part2` wherever all three parts match.

```
scan($seq, "([P][^A])(.{1,1}[^TC])()", \$predss, 'E');
```

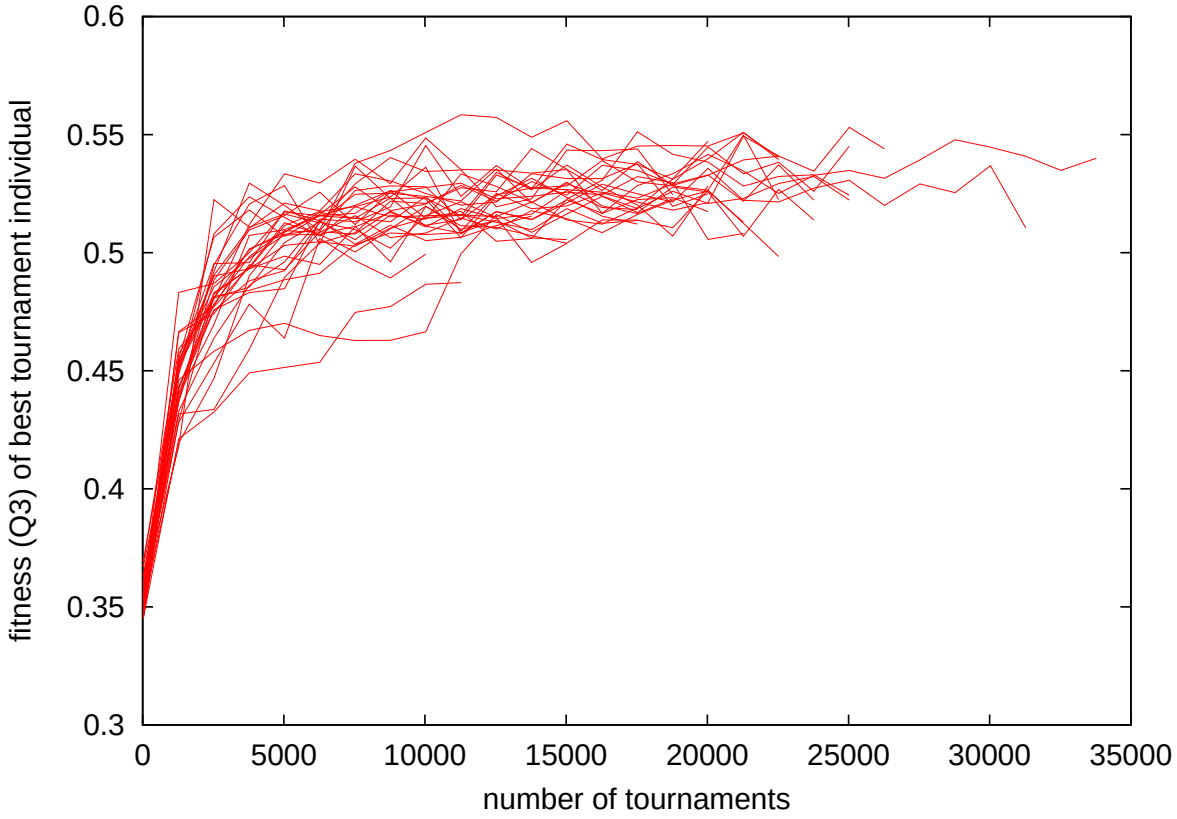
Should be read as "assign strand to all subsequences matching `.[^TC]` which are preceded by `[P][^A]`"

Input: `$seq = 'ALRPRLRAE';` `$predss = 'CCCCCCCCC';`

Output: `$predss = 'CCCCCEECC';`

Note that this function does not assign overlapping regions

Training protocol



30 populations $\times$ 60h

population size: 2000

tournament size: 50

fitness function: Q_3

best 20 reproduce

max age: 2 tournaments

trained on 100 sequences, sampled from 1000 every 1000 tournaments

test data 150 unrelated sequences (different SCOP superfamilies)

Best of tournament after 60h

One of 30 populations:

```
sub predict_secondary_structure {
  my $seq = shift;
  my $predss = 'C' x length($seq);

  scan($seq, "()((?:[LCYCVIF]){1,1}[VYFYVYLFVWICWL])()", \$predss, 'E');
  scan($seq, "([^I][^G])((?:[^GP]){6,})([^GP])", \$predss, 'H');
  scan($seq, "()((?:[^GP][VIFL]){2,12})()", \$predss, 'E');
  scan($seq, "()((?:[ILYWCYCVLIFFYFFFY][^P]){2,12})()", \$predss, 'E');

  return $predss;
}
```

Final subroutines have between 2 and 63 lines of evolved code, only two subroutines have $>$ 20 lines.

Mean Q_3 over 30 populations is 52.3%

Need to do better than 52%...

Search space is huge: `[amino_acid][amino_acid]` = 400 combinations

Unknown fitness landscape

Try to reduce the search space: amino acid alphabet, regex grammar...

The standard grammar is:

```
$functions{REGEX} = [ '{REGEX}{REGEX}', '[{REHAT}{AAMATCH}]',
                      '(?:{REGEX}){REMOD}', '.{REMOD}' ];
$functions{AAMATCH} = [ '{AAMATCH}{AAMATCH}', '{AAMATCHX}' ];

$terminals{AAMATCH} =
$terminals{AAMATCHX} = [ qw(A C D E F Y G H K R I L V M N Q P S T W) ];
$terminals{REHAT} = [ '^', '' ];
$terminals{REMOD} = [ '1,2', '1,3', '2,10' ... ];
```

Grouping amino acids

Generally agreed properties of amino acids: hydrophobicity, size, charge, polarity, backbone geometry etc...

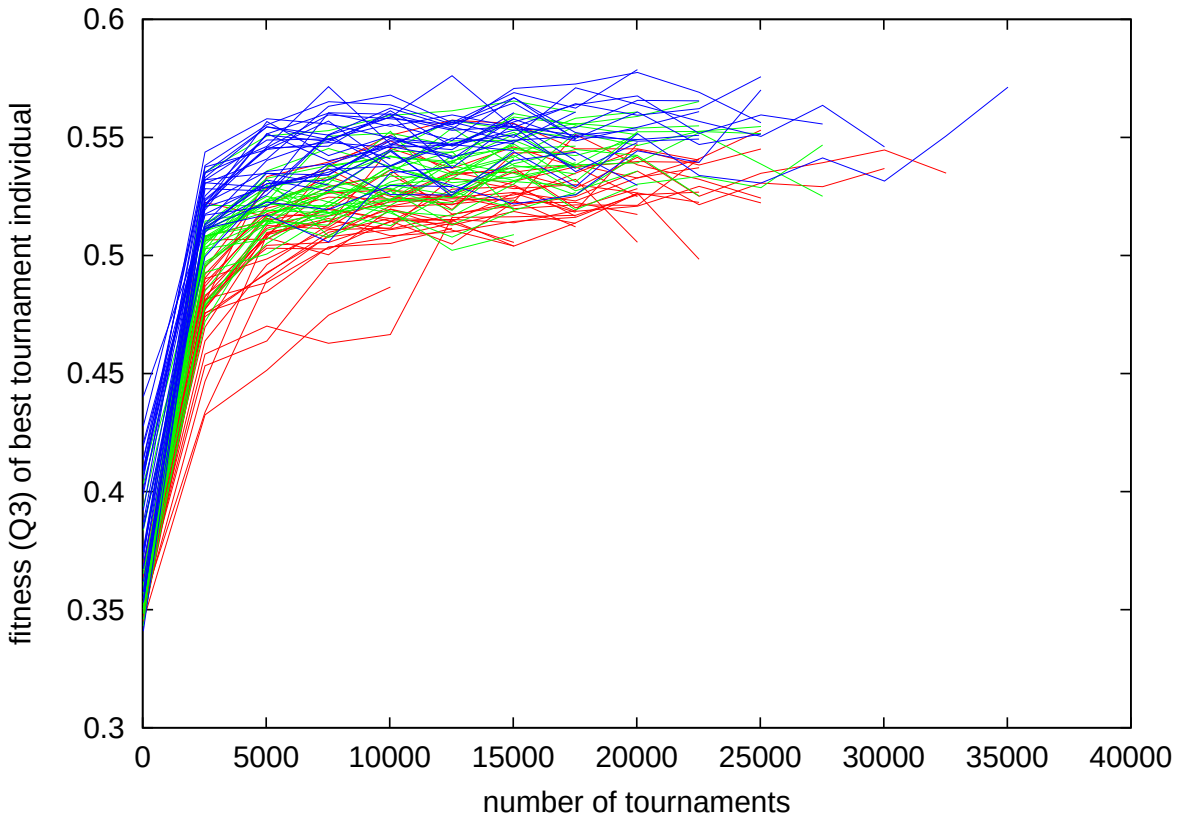
New hierarchical grammar based on physico-chemical properties:

```
$functions{AAMATCH} = [ '{HPHOB}', '{ALIPH}', '{CHRGD}', '{BRKR}',
                        '{AROM}', '{POSAA}', '{NEGAA}',
                        '{AAMATCH}{AAMATCH}','{AAMATCHX}' ];

$functions{HPHOB} = [ '{HPHOB}{HPHOB}', '{HPHOBX}' ];
$functions{ALIPH} = [ '{ALIPH}{ALIPH}', '{ALIPHX}' ];

$terminals{AAMATCH} = $terminals{AAMATCHX} = [ qw(A C D E .. T V W Y) ];
$terminals{HPHOB} = $terminals{HPHOBX} = [ qw(I L V M A F Y W H G) ];
$terminals{ALIPH} = $terminals{ALIPHX} = [ qw(I L V) ];
```

Training "speed" of three regex grammars



Standard grammar

Physico-chemical grouping

Fixed groups from previous green runs: `GP P G DGP DGNPS DGNP IV FIV CFIVWY FILVY CFILVY CFILMVWY ADEGHKNPQRS`

$Q_3 = 52.3\%$ $Q_3 = 53.9\%$ $Q_3 = 54.8\%$

on test set — differences are significant (t-test)

Reduced search space → quicker search and higher fitness

Consensus predictions

Simply take a bunch of independent predictors and see where they agree.

```
sequence  RSRLVQFQKNTDEPMGITLKMNELNHCIVARIMHGGMIHRQGTLHVGDEIREINGISVANQTVEQLQKMLRE...
52.27     ---heehhhhhh----eehhhhhhhhheeehee----ee------ee--ee-ee-eeeehhheehhhhhhhh...
52.27     --eeeeeeee------hhhhhhhhhhhhhhhhh--eeeee-eeee--eeeeee---hhhhhhhhhhhhhh...
52.27     -hheeeehhh-----eeeehhhhhhhhhhhhhh--eee-----eee-eeeeeeeeeee--ehhhhhhhhh...
54.55     -hhhhhhhhh------eehhhhhhhheeeeeeee--eeeee--eee--eeeee--eeee--ehhhhhhhhh...
55.68     -hhheehhhhhh----eeehhhhhhheeehhhh---------------hhhh--eehhhhhhhhhhhhhh...
55.68     hhhheeehhhhh----eeehhhhhhheeehhhh---eeee--eee---eeeee--eeehhhhhhhhhhhh...
56.82     hhhheeehh------eee-------heeehhehh---e-----eee---e--e--eee--hhhhhhhhhh...
60.23     eeeeeeeee-------eeehhhhhhhhhheee---eee---eeeee-eee----eeehhhhhhhhhhhh...
61.36     -hhheeeeh-------eehhhhhhhheeeehee----ee------ee-heehee-eeeehhheehhhhhhh...
all agree             ---          h        --     -     -                hhhhhhh...
consensus -hhheeehhh------ee-hhhhhhheeehhhh---eee----eee--eeeee--eee-hhhhhhhhhhh...
correct   --EEEEEEE-------EEEE------EEEEEE----HHHHH------EEEEEE--EEHHH--HHHHHHHHH...
psipred   --eeeeeee------eeeeee-----eeeeee----hhhhh------eeeee--ee-----hhhhhhhhh...
```

Consensus of around 20 randomly picked predictors from the three experiments reported here gives $Q_3 \approx 58\%$.

However PSIPRED would get $Q_3 \approx 65\%$ if given a single sequence input (it gets 76% with multiple alignments).

So it's **not yet a competitive approach** to SSP.

Dynamics in prediction model?

Not really, definitely **no feedback** from predicted secondary structure.

But the **order of prediction** is interesting:

```
ATLVGPHGPLASGQLAAFHIAAPLPVTATRWDFGDGSAEVDAAGPAASHRYVLPGRYHVTAVLALGAGSALLGTDVQVEA
C ()(.)()
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
E ()([^G][I].{1,1})([^Q])
CCCCCCCCCCCCCCCCCCEEECCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
H ([^F])((([^G]){1,2}[AEELQLAAALLL]){3,})([^Q])
CCCCCCCCCCCCCCCCCCCEEECCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHCCCCCCCCCCCCCCC
H ([^G])((([^G]){1,4}[MRAL]){5,})([^G])
CCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHCCCCCCCCCCCCCC
H ([^G])(((((^G]){1,3}[LA]){3,}){1,3}[MEH]){3,})()
CCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHCCCCCCCCCCCCCC
E ([^V])((((P]([^G]){1,3}){3,}[AEMAELHAYLQQE]){3,}){3,})([Q])
CCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHCCCCCCCCCCCCCC
E ()([V].{1,3})()
CCCEEEECCCCCCCHHHHHHHHHHHEEEEHCCCCCCCCCEEEECCCCCCCCEEEECCCEEEEHHHCCCCCCCCCCEEEEC
E ((([^G]){1,3}[LA])([^G])
EEEEEEECEEECCEEEEHEEEEHHEEEEEHCCCCCCEECEEEECEEECCEEEEEECCEEEEEEEHCCCEEECCCCEEEEC
H ([^T])(((^G]){1,2}[QEKALM]){3,})([^P])
EEEEEEECEEECCEEEEHEEEEHHEEEEEHCCCCCCEECEEEECEEECCEEEEEECCEHHHHHHHCCCEEECCCCEEEEC
```

Prediction histories

C→C→C→C→E→E→E→E→H ≡ C→E→H

108788	20.15%	C
83018	15.38%	C→H
60338	11.18%	C→E
45890	8.50%	H
40164	7.44%	C→H→E
32645	6.05%	H→E
23801	4.41%	E→C
21438	3.97%	C→E→H
21248	3.94%	H→C
20617	3.82%	E→H
18382	3.41%	E
12765	2.36%	H→E→C
6591	1.22%	E→C→H
4942	0.92%	E→C→E→H
4364	0.81%	H→C→E→C
4333	0.80%	E→H→C

we see more H→E transitions than E→H

suggests that helix may be a *default state*

in terms of our self-organised "model"

but helix sequence patterns are slightly more complex than strand patterns

many helix regex's are actually "non-coil" predictors

What next?

My conclusions:

Feedback mechanisms and long-range patterns didn't evolve because predicted SS is full of *noise*, and there's *less effective training data* for it.

SSP is better tackled with floating point ML methods (NNs/HMMs)

55 or 58% is probably the limit for this approach

Your comments...?

Other/future work includes forcing dynamics to happen (cellular automata), starting with PSIPRED predictions and try to improve them (with a higher level representation and global information).