German Centre for Integrative Biodiversity Research (iDiv)
Halle-Jena-Leipzig

HELMHOLTZ
Zentrum für Umweltforschung

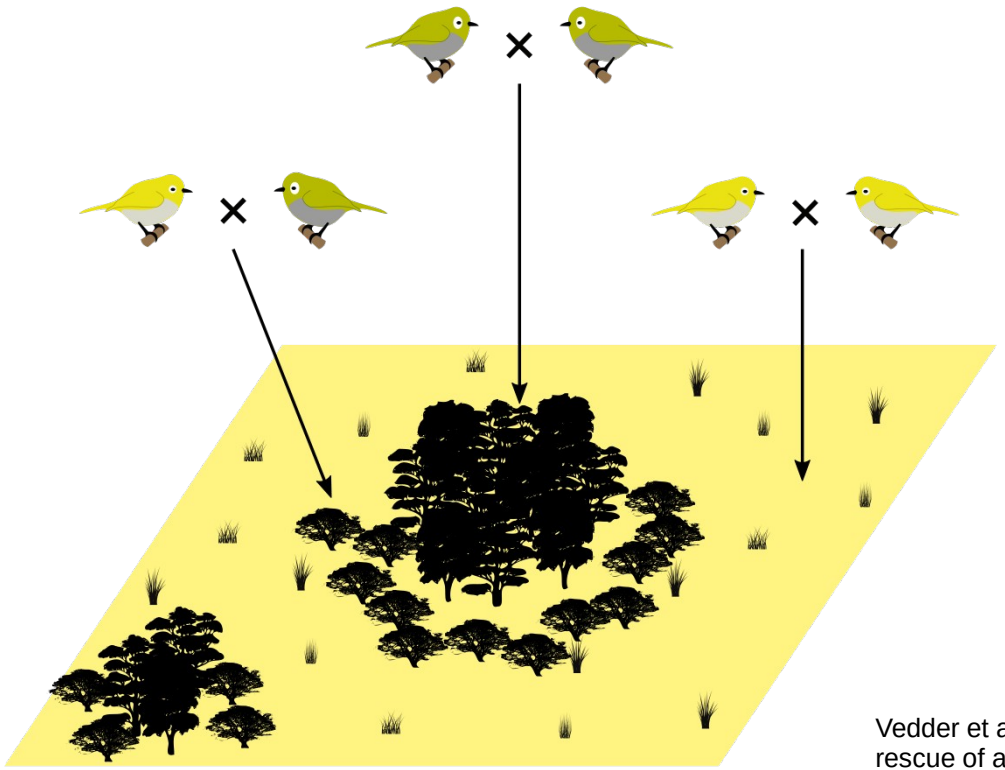
Fast code

Performance optimisation in Julia

Daniel Vedder

iDiv is a research centre of the
DFG Deutsche Forschungsgemeinschaft

Houston, we have a problem…



measured

350 000 individuals
x 500 years
= 17 hours
→ 10.3M IndYears/h

calculated

350 000 individuals
x 2M years
= 70 000 hours
→ 8 years!

Vedder et al. (2022). Hybridization may aid evolutionary rescue of an endangered East African passerine. Evolutionary Applications, 15(7), 1177–1188. https://doi.org/10.1111/eva.13440

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Before we start

"We should forget about small inefficiencies, say about 97% of the time; premature optimization is the root of all evil."

Donald Knuth

Before we start

Reasons to optimise:

- Make a problem computationally tractable
- Speed up the modelling cycle *(code change → simulation → data analysis → code change ...)*
- It's fun, and teaches you a lot

Reasons *not* to optimise:

- Optimisation takes time (lots!)
- Code readability suffers
- Outcome is uncertain, performance gains are hard to predict and sometimes counterintuitive

Measure twice, cut once

- Knuth (1971): 4% of the software consume >50% of the time
  Boehm (1987): 20% of the software consume 80% of the time
- Do you want to optimise CPU or RAM usage (i.e. speed or memory)?
- Therefore:
  - get the program working *first*
  - then, profile to find the relevant bottleneck
  - *then* (and only then), optimise
  - finally, profile again to verify it worked

Profiling in Julia

```julia
#!/usr/bin/env julia
# A very thin wrapper to start a GeMM simulation. If you need something fancier,
# have a look at `rungemmparallel.jl`, or import the GeMM module into your own
# wrapper script.

thisDir = joinpath(pwd(), "src")
any(path -> path == thisDir, LOAD_PATH) || push!(LOAD_PATH, thisDir)
using Pkg
Pkg.activate(".")
using GeMM

using Profile
Profile.clear()

rm("results/taita_test", recursive=true, force=true)

@profile rungemm("zosterops.config")

open("profile_flat.txt", "w") do s
    Profile.print(IOContext(s, :displaysize=>(300,145)), format=:flat, mincount=10, sortedby=:count)
end

open("profile_tree.txt", "w") do s
    Profile.print(IOContext(s, :displaysize=>(300,300)), mincount=10)
end

```

Profiling in Julia

```
Count  Overhead File                                          Line Function
=====  ======== ====                                          ==== ========
  417         0 .../Studium/Masterarbeit/model/src/scheduling.jl   13 simulate!(::Array{GeMM.Patch,1}, ::Int64, ::Int64)
  482         0 .../Studium/Masterarbeit/model/src/scheduling.jl   25 simulate!(::Array{GeMM.Patch,1}, ::Int64, ::Int64)
  539         1 ...beit/Studium/Masterarbeit/model/src/output.jl  357 (::GeMM.var"#logprint#44"{String,Bool})(::String, ::Bool)
  545         0 ...beit/Studium/Masterarbeit/model/src/output.jl  363 simlog(::String, ::Char, ::String, ::Bool)
  548         9 ...tudium/Masterarbeit/model/src/reproduction.jl  135 createoffspring(::Int64, ::GeMM.Individual, ::GeMM.Individual, ::Bool)
  552         0 ...beit/Studium/Masterarbeit/model/src/output.jl  350 logprint
  556         0 ...beit/Studium/Masterarbeit/model/src/output.jl  348 simlog
  791         3 ...t/Studium/Masterarbeit/model/src/zosterops.jl  251 zreproduce!(::GeMM.Patch)
  850         0 @Base/io.jl                                       323 open
  862         1 .../Studium/Masterarbeit/model/src/scheduling.jl   75 macro expansion
 1096         0 @Base/threadingconstructs.jl                       81 (::GeMM.var"#70#threadsfor_fun#117"{Array{GeMM.Patch,1}})(::Bool)
 1098         0 @Base/threadingconstructs.jl                       48 (::GeMM.var"#70#threadsfor_fun#117"{Array{GeMM.Patch,1}})()
 1129         0 ...dium/Masterarbeit/model/src/run_simulation.jl   22 runsim(::String, ::Int64)
 1256         0 @Base/timing.jl                                   174 macro expansion
 1257         0 ...dium/Masterarbeit/model/src/run_simulation.jl   36 rungemm(::String, ::Int64)
 1257         0 @Base/client.jl                                   457 include(::String)
 1257         0 @Base/boot.jl                                     331 eval(::Module, ::Any)
 1257         0 .../usr/share/julia/stdlib/v1.5/REPL/src/REPL.jl  134 eval_user_input(::Any, ::REPL.REPLBackend)
 1257         0 .../usr/share/julia/stdlib/v1.5/REPL/src/REPL.jl  195 repl_backend_loop(::REPL.REPLBackend)
 1257         0 .../usr/share/julia/stdlib/v1.5/REPL/src/REPL.jl  180 start_repl_backend(::REPL.REPLBackend, ::Any)
 1257         0 .../usr/share/julia/stdlib/v1.5/REPL/src/REPL.jl  292 run_repl(::REPL.AbstractREPL, ::Any; backend_on_current_task::Bool)
 1257         0 .../usr/share/julia/stdlib/v1.5/REPL/src/REPL.jl  288 run_repl(::REPL.AbstractREPL, ::Any)
 1257         0 @Base/client.jl                                   399 (::Base.var"#807#809"{Bool,Bool,Bool,Bool})(::Module)
 1257         0 @Base/essentials.jl                               710 #invokelatest#1
 1257         0 @Base/essentials.jl                               709 invokelatest
 1257         0 @Base/client.jl                                   383 run_main_repl(::Bool, ::Bool, ::Bool, ::Bool, ::Bool)
 1257         0 @Base/client.jl                                   313 exec_options(::Base.JLOptions)
 1257         0 @Base/client.jl                                   506 _start()
Total snapshots: 3491
```

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Profiling in Julia

```
Overhead ┊ [+additional indent] Count File:Line; Function
=========================================================
```

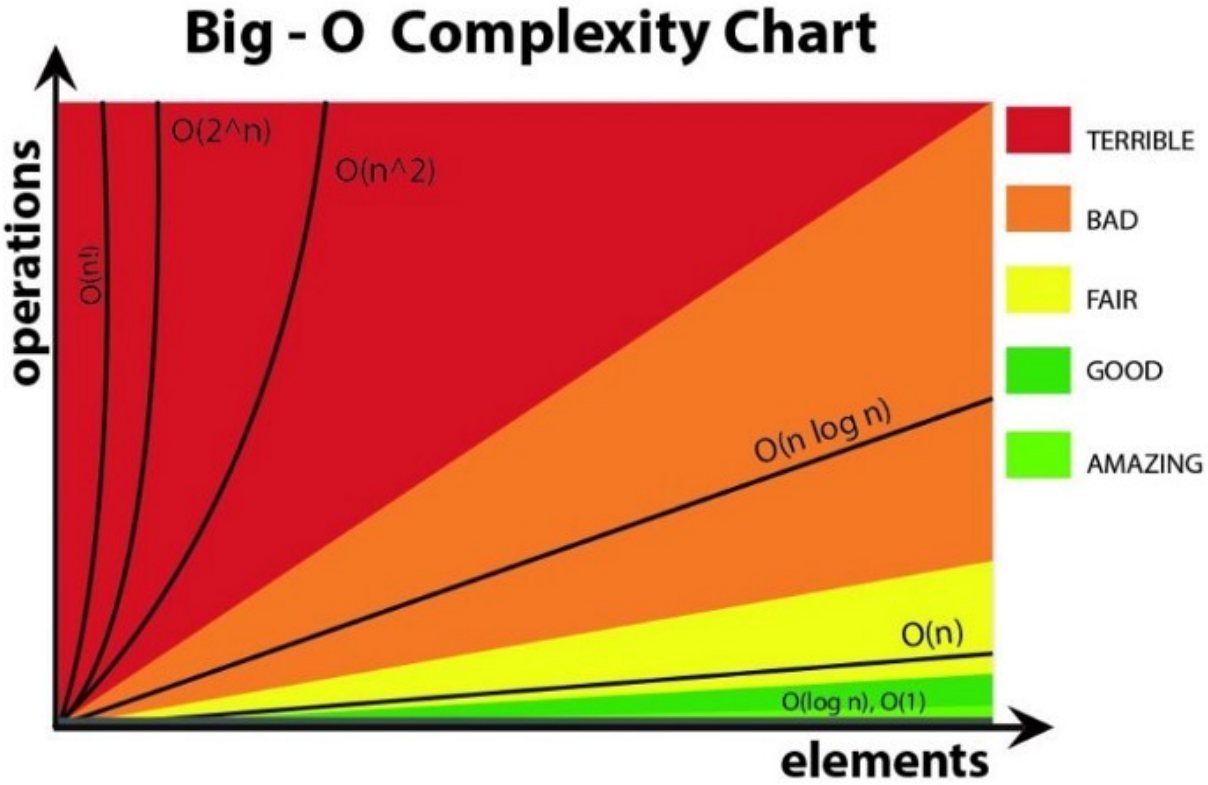
```
  37┊  47   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/zosterops.jl:247; zreproduce!(::GeMM.Patch)
   3┊  791  /media/DATA/Arbeit/Studium/Masterarbeit/model/src/zosterops.jl:251; zreproduce!(::GeMM.Patch)
   ┊   66   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/reproduction.jl:130; createoffspring(::Int64, ::GeMM.Individual, ::GeMM.Individual, ::Bool)
   ┊   12   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:17; meiosis(::Array{GeMM.Chromosome,1}, ::Bool, ::String, ::Bool)
  12┊   ┊ 12   @Base/boot.jl:406; Array
   8┊   43   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:21; meiosis(::Array{GeMM.Chromosome,1}, ::Bool, ::String, ::Bool)
  29┊   ┊ 29   @Base/Base.jl:33; getproperty
   2┊   55   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/reproduction.jl:131; createoffspring(::Int64, ::GeMM.Individual, ::GeMM.Individual, ::Bool)
   4┊   35   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:21; meiosis(::Array{GeMM.Chromosome,1}, ::Bool, ::String, ::Bool)
  17┊   ┊ 17   @Base/Base.jl:33; getproperty
   6┊   16   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/reproduction.jl:134; createoffspring(::Int64, ::GeMM.Individual, ::GeMM.Individual, ::Bool)
   9┊   548  /media/DATA/Arbeit/Studium/Masterarbeit/model/src/reproduction.jl:135; createoffspring(::Int64, ::GeMM.Individual, ::GeMM.Individual, ::Bool)
   1┊   44   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:67; gettraitdictfast(::Array{GeMM.Chromosome,1}, ::Array{String,1})
   ┊   ┊ 43   @Base/dict.jl:90; Dict{String,Float64}()
  36┊   ┊ 36   @Base/boot.jl:406; Array
   ┊   12   /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:69; gettraitdictfast(::Array{GeMM.Chromosome,1}, ::Array{String,1})
   ┊   ┊ 12   @Base/boot.jl:420; Array
  12┊   ┊ 12   @Base/boot.jl:406; Array
 148┊   341  /media/DATA/Arbeit/Studium/Masterarbeit/model/src/genetics.jl:71; gettraitdictfast(::Array{GeMM.Chromosome,1}, ::Array{String,1})
  19┊   ┊ 19   @Base/Base.jl:33; getproperty
 154┊   ┊ 155  @Base/Base.jl:33; getproperty(::GeMM.Trait, ::Symbol)
  12┊   ┊ 12   @Base/array.jl:809; getindex(::Array{GeMM.Trait,1}, ::Int64)
```

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Runtime classes

- In order to get the result of a computation (e.g. sorting), how often must an operation (e.g. comparing two numbers) be performed on each element?
- How does the runtime depend on the number of elements *n*?
  - O(n) = worst-case runtime
  - Ω(n) = best-case runtime
  - Θ(n) = average-case runtime

Runtime classes



https://joseiciano.medium.com/an-introduction-to-big-o-big-omega-and-big-theta-d10b5c7db002

Runtime classes

Examples from IBM development:

- Find a random individual in a patch: $O(1)$
- Find the biggest individual in a patch: $O(n)$
- Find a suitable mate for all individuals in the patch: $O(n^2)$

Dealing with bad runtimes:

- Reduce the runtime class (e.g.: finding the biggest individual in a patch is $O(n)$, but doing so in a size-sorted patch is $O(1)$)
  → algorithms and data structures
- Do it less often (reduce n)

Code tuning

Some common miscreants:

- disk I/O
- nested loops
- array allocation

Strategies:

- buffer output
- structure data
- reduce allocations
- cache calculations
- parallelise
- compress

Buffer output

```
252 """
253     recordlineages(w)
254
255 Save the abundance of each lineage per patch. (Low-detail data recording function.)
256 """
257 function recordlineages(world::Array{Patch,1}, timestep::Int)
258     #XXX despite being low-detail, calling this frequently still means a lot of I/O
259     if !isfile(joinpath(setting("dest"), "lineages.log"))
260         simlog("t,X,Y,lineage,abundance,temp,prec", 'i', "lineages.log", true)
261     end
262     datastring = ""
263     for p in world
264         for l in unique(map(x -> x.lineage, p.community))
265             datastring *= string(timestep)*","*string(p.location[1])*","*string(p.location[2])*
266             ","*string(l)*","*string(length(findall(x -> x.lineage == l, p.community)))*","*
267             string(p.temp)*","*string(p.prec)*"\n"
268         end
269     end
270     simlog(datastring, 'i', "lineages.log", true)
271 end
```

first construct a complete string, then write it out

Structure data

```
28 """
29     gettraitdict(chromosomes, traitnames)
30
31 Convert a genome (an array of chromosomes) into a dict of traits and their values.
32 """
33 function gettraitdict(chrms::Array{Chromosome, 1}, traitnames::Array{String, 1})
34     #TODO can this be made more efficient? It's called really often...
35     traitdict = Dict{String, Float64}()
36     traits = Array{Trait,1}()
37     nchrms = 0
38     ngenes = 0
39     for chrm in chrms
40         nchrms += 1
41         for gene in chrm.genes
42             ngenes += 1
43             append!(traits, gene.codes)
44         end
45     end
46     for traitidx in eachindex(traitnames)
47         wantedtraits = skipmissing(map(x -> x.value, filter(x -> x.nameindex == traitidx, traits)))
48         traitdict[traitnames[traitidx]] = mean(wantedtraits)
49         traitdict[traitnames[traitidx] * "sd"] = std(wantedtraits)
50     end
51     traitdict["ngenes"] = ngenes
52     traitdict["nlnkgunits"] = nchrms
53     traitdict
54 end
```

$O(n) = n^2$

Structure data

```
"""
    gettraitdictfast(chromosomes, traitnames)

Convert a genome (an array of chromosomes) into a dict of traits and their values.
This is an optimised version that can be run if `degpleiotropy` is 0 and `linkage` is "none".
"""
function gettraitdictfast(chrms::Array{Chromosome, 1}, traitnames::Array{String, 1})
    # Makes use of the fact that with `degpleiotropy == 1` and `linkage == "none"`,
    # there is exactly one trait per chromosome (one gene per chromosome and one trait per gene),
    # and the chromosomes are arranged in trait-order.
    genomesize = length(chrms)
    traitdict = Dict{String, Float64}()
    haploidlength = Int(genomesize/2)
    values = Array{Float64,}(undef,genomesize)
    for c in eachindex(chrms)
        @inbounds values[c] = chrms[c].genes[1].codes[1].value
    end
    for traitidx in eachindex(traitnames)
        wantedtraits = (values[traitidx], values[traitidx+haploidlength])
        traitdict[traitnames[traitidx]] = mean(wantedtraits)
        traitdict[traitnames[traitidx] * "sd"] = std(wantedtraits)
    end
    traitdict["ngenes"] = genomesize
    traitdict["nlnkgunits"] = genomesize
    traitdict
end
```

O(n) = n

Reduce allocations

```
"""
    zdisperse!(bird, world, location)

Dispersal of a single bird. Birds look patches with a suitable
habitat and a free territory or available mate. (Cf. Aben et al. 2016)
"""
function zdisperse!(bird::Individual, world::Array{Patch,1}, location::Tuple{Int, Int})
    # keep track of where we've been and calculate the max dispersal distance
    x, y = location
    route = [location]
    !(bird.traits["dispshape"] > 0) && @goto failure # Logistics() requires θ > zero(θ)
    maxdist = rand(Logistic(bird.traits["dispmean"], bird.traits["dispshape"]))
    while maxdist > 0
        # calculate the best habitat patch in the surroundings (i.e. the closest to AGC optimum)
        target = [(x-1, y-1), (x, y-1), (x+1, y-1),
                  (x-1, y),             (x+1, y),
                  (x-1, y+1), (x, y+1), (x+1, y+1)]
        bestdest = nothing
        bestfit = nothing
        for t in target
            (t in route) && continue
            possdest = coordinate(t[1], t[2], world)
            isnothing(possdest) && continue
            patchfit = abs(possdest.prec - bird.traits["precopt"])
            if isnothing(bestdest) || patchfit < bestfit
                bestdest, bestfit = possdest, patchfit
            end
        end
        (isnothing(bestdest)) && @goto failure
        x, y = bestdest.location
        # check if the patch is within the bird's AGC range and has free space
        if (bestfit <= bird.traits["prectol"] && length(bestdest.community) < setting("cellsize"))
            # can we settle here?
```

array allocation

Reduce allocations

```
"""
    zdisperse!(bird, world, location)

Dispersal of a single bird. Birds look patches with a suitable
habitat and a free territory or available mate. (Cf. Aben et al. 2016)
"""
function zdisperse!(bird::Individual, world::Array{Patch,1}, location::Tuple{Int, Int})
    # keep track of where we've been and calculate the max dispersal distance
    x, y = location
    route = [location]
    !(bird.traits["dispshape"] > 0) && @goto failure # Logistics() requires θ > zero(θ)
    maxdist = rand(Logistic(bird.traits["dispmean"], bird.traits["dispshape"]))
    while maxdist > 0
        # calculate the best habitat patch in the surroundings (i.e. the closest to AGC optimum)
        bestdest = nothing
        bestfit = nothing
        for xnew in (x-1):(x+1)
            for ynew in (y-1):(y+1)
                ((xnew, ynew) in route) && continue
                possdest = coordinate(xnew, ynew, world) #XXX expensive?
                isnothing(possdest) && continue
                patchfit = abs(possdest.prec - bird.traits["precopt"])
                if isnothing(bestdest) || patchfit < bestfit
                    bestdest, bestfit = possdest, patchfit
                end
            end
        end
        (isnothing(bestdest)) && @goto failure
        x, y = bestdest.location
        # check if the patch is within the bird's AGC range and has free space
        if (bestfit <= bird.traits["prectol"] && length(bestdest.community) < setting("cellsize"))
            # can we settle here?
```

no array allocation
(but deeply nested loops)

Cache calculations

```
"""
    zdisperse!(bird, world, location)

Dispersal of a single bird. Birds look patches with a suital
habitat and a free territory or available mate. (Cf. Aben e
"""
function zdisperse!(bird::Individual, world::Array{Patch,1}
    # keep track of where we've been and calculate the max
    x, y = location
    route = [location]
    !(bird.traits["dispshape"] > 0) && @goto failure # Logi
    maxdist = rand(Logistic(bird.traits["dispmean"], bird.t
    while maxdist > 0
        # calculate the best habitat patch in the surroundi
        bestdest = nothing
        bestfit = nothing
        for xnew in (x-1):(x+1)
            for ynew in (y-1):(y+1)
                ((xnew, ynew) in route) && continue
                possdest = coordinate(xnew, ynew, world) #XXX expensive?
                isnothing(possdest) && continue
                patchfit = abs(possdest.prec - bird.traits["precopt"])
                if isnothing(bestdest) || patchfit < bestfit
                    bestdest, bestfit = possdest, patchfit
                end
            end
        end
        (isnothing(bestdest)) && @goto failure
        x, y = bestdest.location
        # check if the patch is within the bird's AGC range and has free space
        if (bestfit <= bird.traits["prectol"] && length(bestdest.community) < setting("cellsize"))
            # can we settle here?
```

```
let width = 0, height = 0
    """
        coordinate(x, y, world)

    A utility function to perform a fast look-up for the patch at coordinate x/y.
    Important: this assumes a rectangular world with coordinates in row-major order!
    Returns the index of the desired patch.
    """
    global function coordinate(x::Int, y::Int, world::Array{Patch,1})
        if iszero(width)
            width = maximum(p -> p.location[1], world)
            height = maximum(p -> p.location[2], world)
        end
        (x <= 0 || y <= 0 || x > width || y > height) && return
        i = ((y-1) * width) + x
        return i
    end
end
```

Cache calculations

```julia
let width = 0, height = 0
    """
        coordinate(x, y, world)

    A utility function to perform a fast look-up for the patch at coordinate x/y.
    Important: this assumes a rectangular world with coordinates in row-major order!
    Returns the index of the desired patch.
    """
    global function coordinate(x::Int, y::Int, world::Array{Patch,1})
        if iszero(width)
            width = maximum(p -> p.location[1], world)
            height = maximum(p -> p.location[2], world)
        end
        (x <= 0 || y <= 0 || x > width || y > height) && return
        i = ((y-1) * width) + x
        return i
    end
end

"""
    findneighbours(world)

Construct a list of neighbours for each patch in the world, for faster lookup
later on. (Must be called during initialisation.)
"""
function findneighbours!(world::Array{Patch,1})
    for patch in world
        for x in (patch.location[1]-1):(patch.location[1]+1)
            for y in (patch.location[2]-1):(patch.location[2]+1)
                ((x,y) == patch.location) && continue
                neighbour = coordinate(x,y,world)
                (isnothing(neighbour)) && continue
                push!(patch.neighbours, neighbour)
            end
        end
    end
end
```

```julia
"""
    zdisperse!(bird, patch, world)

Dispersal of a single bird. Birds look patches with a suitable
habitat and a free territory or available mate. (Cf. Aben et al. 2016)
"""
function zdisperse!(bird::Individual, patch::Patch, world::Array{Patch,1})
    # keep track of where we've been and calculate the max dispersal distance
    route = [patch.location]
    !(bird.traits["dispshape"] > 0) && @goto failure # Logistics() requires θ > zero(θ)
    maxdist = rand(Logistic(bird.traits["dispmean"], bird.traits["dispshape"]))
    while maxdist > 0
        # calculate the best habitat patch in the surroundings (i.e. the closest to AGC optimum)
        bestdest = nothing
        bestfit = nothing
        for pid in patch.neighbours
            neighbour = world[pid]
            (neighbour.location in route) && continue
            neighbourfit = abs(neighbour.prec - bird.traits["precopt"])
            if isnothing(bestdest) || neighbourfit < bestfit
                bestdest, bestfit = neighbour, neighbourfit
            end
        end
        (isnothing(bestdest)) && @goto failure
        # check if the patch is within the bird's AGC range and has free space
        if (bestfit <= bird.traits["prectol"] && length(bestdest.community) < bestdest.capacity)
            # can we settle here?
            partner = zfindmate(bestdest.community, bird)
            if !isnothing(partner)
                # if we've found a partner
```

single loop

50% total runtime reduction!

Parallelise

```julia
"""
    zosteropsexperiment(world)

The annual update procedure for the Zosterops experiments, this time for bird populations.
"""
function zosteropsexperiment(world::Array{Patch,1})
    # The first four processes are patch-internal and can therefore be parallelised
    # Note: multithreading requires calling Julia with the -p parameter
    Threads.@threads for patch in world
        establish!(patch)
        survive!(patch)
        zreproduce!(patch)
        if setting("mutate")
            mutate!(patch)
        end
    end
    zdisperse!(world)
end
```

Compression

GeMM has a very high RAM usage – how do we reduce this?

→ idea: compress genetic sequences

20 bytes

4 bytes

sequence: `acgtaatgcccacatttgga`

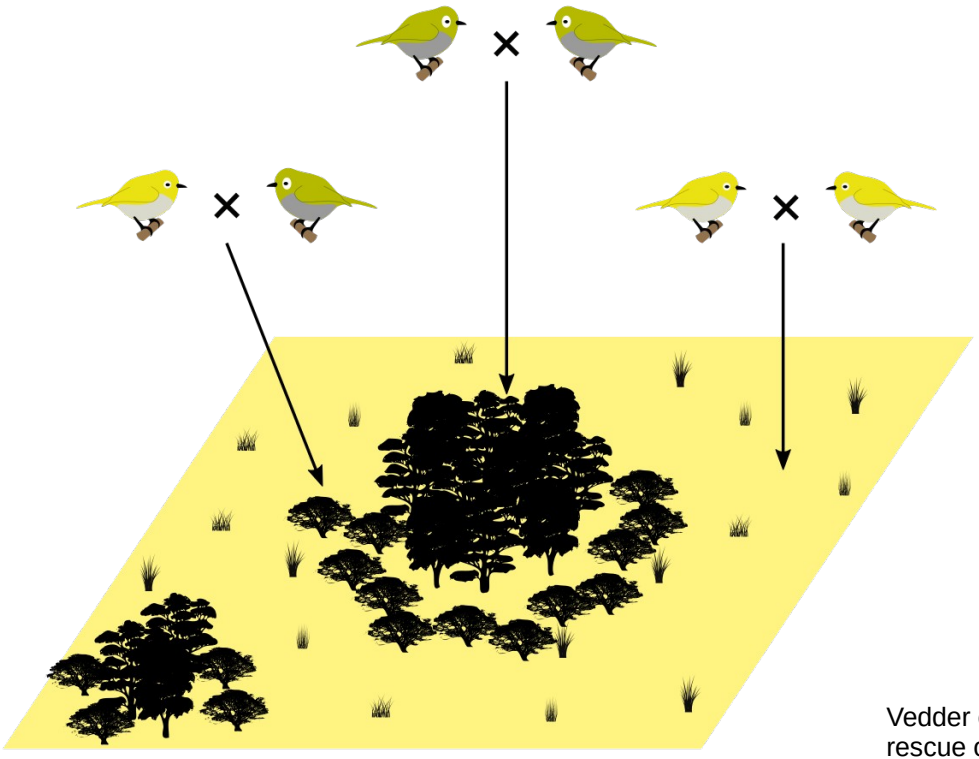
binary: `10010111011110010011 1`
`1101011011011001011 00`
`11111111111011010 0`

decimal: `6821765079202528 52`

```
"""
    seq2num(sequence)

Convert a DNA base sequence (a string) into binary and then into an integer.
This saves memory.
"""
function seq2num(sequence::String)
    num::Int64 = 0  # Int64 allows for max length of 21bp
    for b in eachindex(sequence)
        if sequence[end+1-b] == 'a'
            num += 2^(3*(b-1)) * 4 # b'100'
        elseif sequence[end+1-b] == 'c'
            num += 2^(3*(b-1)) * 5 # b'101'
        elseif sequence[end+1-b] == 'g'
            num += 2^(3*(b-1)) * 6 # b'110'
        elseif sequence[end+1-b] == 't'
            num += 2^(3*(b-1)) * 7 # b'111'
        end
    end
    num
end

"""
    num2seq(n)

Convert an integer into binary and then into a DNA base sequence string.
"""
function num2seq(n::Integer)
    bases = "acgt"
    binary = string(n, base = 2)
    sequence = ""
    for i in 1:3:(length(binary) - 2)
        sequence *= string(bases[parse(Int, binary[i:(i + 2)], base = 2) - 3])
    end
    sequence
end
```

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Houston, we've solved the problem!



measured

350 000 individuals
x 500 years
= 17 hours
→ 53M Ind/h

calculated

350 000 individuals
x 200 000 years
= 70 000 hours
→ 8 years!

Vedder et al. (2022). Hybridization may aid evolutionary rescue of an endangered East African passerine. Evolutionary Applications, 15(7), 1177–1188. https://doi.org/10.1111/eva.13440

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Thank you for your attention!

Any questions?