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

HELMHOLTZ
Zentrum für Umweltforschung

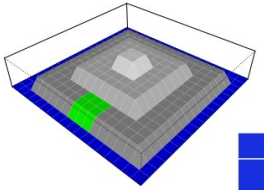
Why I like Julia

Performance, flexibility, and elegance

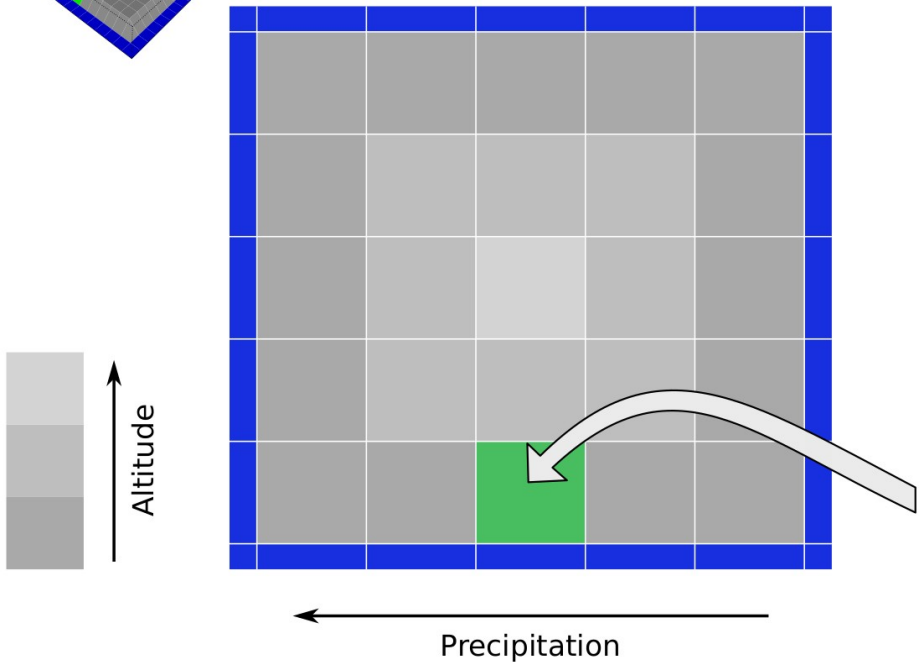
Daniel Vedder

iDiv is a research centre of the
DFG Deutsche Forschungsgemeinschaft

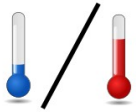
First steps: island plants



Simulation arena



Scenarios



15°C vs. 35°C
lowland temperature

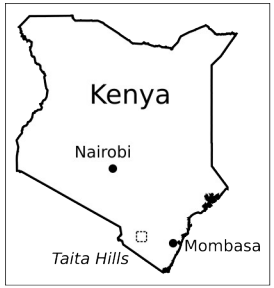
1% vs. 10%
disturbance mortality

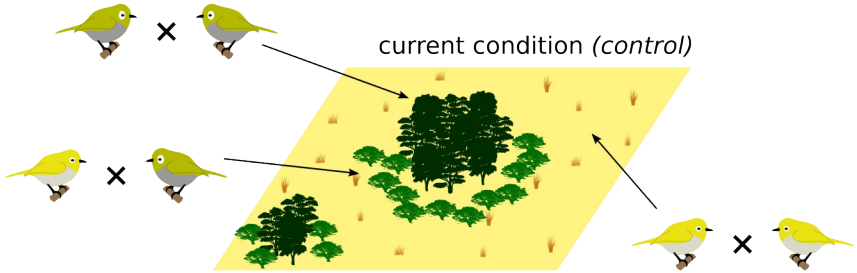
1 vs. 10 plants per year
propagule pressure

Vedder, D., Leidinger, L., & Sarmento Cabral, J. (2021). Propagule pressure and an invasion syndrome determine invasion success in a plant community model. Ecology and Evolution, 11(23), 17106–17116. https://doi.org/10.1002/ece3.8348

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

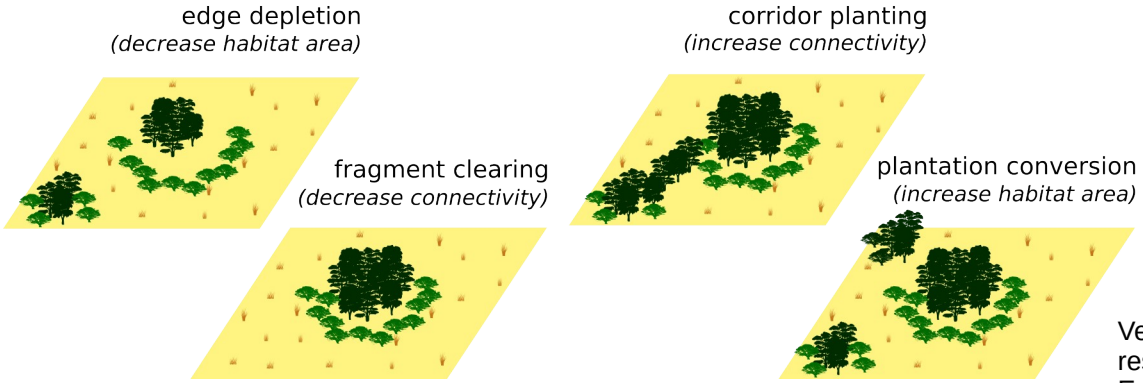
Next steps: sky-island birds





© Lars Peterssen, used by permission

Management scenarios:

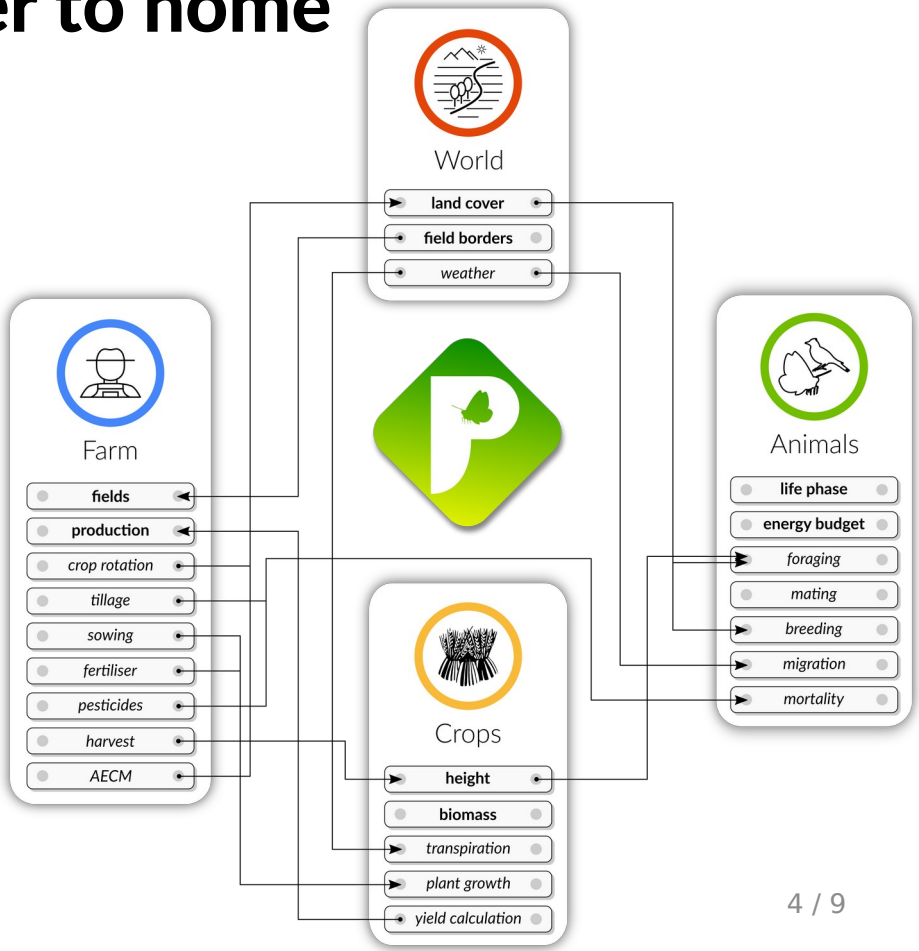


Vedder et al. (2022). Hybridization may aid evolutionary rescue of an endangered East African passerine. Evolutionary Applications, 15(7), 1177–1188. https://doi.org/10.1111/eva.13440

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

Now: more birds, but closer to home

PERSEFONE.jl
agriculture and ecosystems



https://persefone-model.eu

iDiv UFZ HELMHOLTZ Zentrum für Umweltforschung

“After having used all of these languages at various times, our working group has settled on Julia as our current ‘ideal candidate’. […] Overall, our experiences with the language have been almost entirely positive.”

Vedder, D., Ankenbrand, M., & Cabral, J. S. (2021).
Dealing with software complexity in individual-based models.
Methods in Ecology and Evolution, 12(12), 2324–2333.
https://doi.org/10.1111/2041-210X.13716

What I like about Julia

- **Performance:** complicated simulations of 100.000s of individuals over 100s of updates (with dozens of replicates) are feasible
- **Flexibility:** the language provides just the right tools for the job – and when necessary, you can adapt it to your needs
- **Elegance:** the language doesn’t just get out of your way, it speeds you on your way

Extending the language: AnnualDates

```julia
mutable struct AnnualDate
    month::Int64
    day::Int64
end

# allow creating AnnualDates from a string of the format "8 August"
AnnualDate(ad::String) = AnnualDate(Date(ad, dateformat"d U"))
Base.convert(::Type{AnnualDate}, ad::String) = AnnualDate(ad)
Base.tryparse(::Type{AnnualDate}, ad::String) = AnnualDate(ad)

# Interface with Dates
AnnualDate(date::Date) = AnnualDate(month(date), day(date))
Base.convert(::Type{AnnualDate}, ad::Date) = AnnualDate(ad)
Dates.month(ad::AnnualDate) = ad.month
Dates.day(ad::AnnualDate) = ad.day
Dates.monthday(ad::AnnualDate) = (ad.month, ad
Date(year::Int64, ad::AnnualDate) = Date(year,

# Addition and subtraction of date periods
Base.:(+)(ad::AnnualDate, time::DatePeriod) =
Base.:(-)(ad::AnnualDate, time::DatePeriod) =
Base.:(-)(ad1::AnnualDate, ad2::AnnualDate) =
    Date(2022, ad1) - Date(2022, ad2) :
    Date(2022, ad1) - Date(2021, ad2)

# Taking ranges
Base.:(:)(start::AnnualDate, stop::AnnualDate)
    if start < stop # normal case, e.g. Easter
        AnnualDate.(Date(2022, start):Date(202
    else # handle wrap-around, e.g. Christmas:
        AnnualDate.(Date(2021, start):Date(202
    end
```

```julia
julia> christmas = AnnualDate(12, 24)

julia> today() < christmas
true

julia> birthday::AnnualDate = “21 August”

julia> christmas - birthday
125 days
```

Molding the language: species macros

```
11 @species Mermaid begin
12     ageofmaturity = 2
13     pesticidemortality = 1.0
14 end
15
16 @create Mermaid begin
17     @debug "Created $(animalid(self))."
18 end
19
20 @phase Mermaid life begin
21     @debug "$(animalid(self)) is swimming happily in its pond."
22     @respond pesticide @kill(self.pesticidemortality, "poisoning")
23     @respond harvesting @setphase(drought)
24     if self.sex == female && length(@neighbours()) < 3 &&
25         self.age >= self.ageofmaturity && @landcover() == water
26         @reproduce()
27     end
28 end
29
30 @phase Mermaid drought begin
31     n = sum(1 for a in @neighbours())
32     @debug "$(animalid(self)) is experiencing d
33     @respond sowing @setphase(life)
34 end
35
36 @populate Mermaid begin
37     birthphase = life
38     initphase = life
39     habitat = @habitat(@landcover() == water)
40     pairs=true
41 end
```

```
45 function life(self::Mermaid, model::SimulationModel)
46     pos = self.pos
47     @debug "$(animalid(self)) is swimming happily in its pond."
48     if pesticide in model.landscape[pos...].events
49         kill!(self, model, self.pesticidemortality, "poisoning")
50     end
51     if harvesting in model.landscape[pos...].events
52         self.phase = drought
53     end
54     if self.sex == female && length(neighbours(self, model)) < 3 &&
55         self.age >= self.ageofmaturity && landcover(pos, model) == water
56         reproduce(self, model)
57     end
58 end
```

Happy hacking!