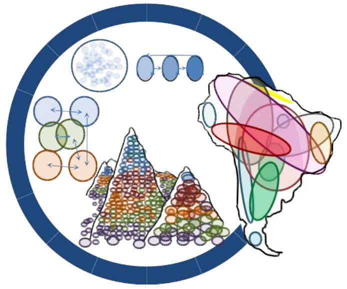
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

:::::: Faculty of Biology
Center for Computational and
Theoretical Biology

GeMM:

a genetically-explicit metacommunity model

Ecomods group meeting,
14/01/2021



Daniel Vedder, Ecosystem Modeling Group
Center for Computational and Theoretical Biology

Overview

1) What was GeMM designed for?

2) What biological components and processes does GeMM simulate?

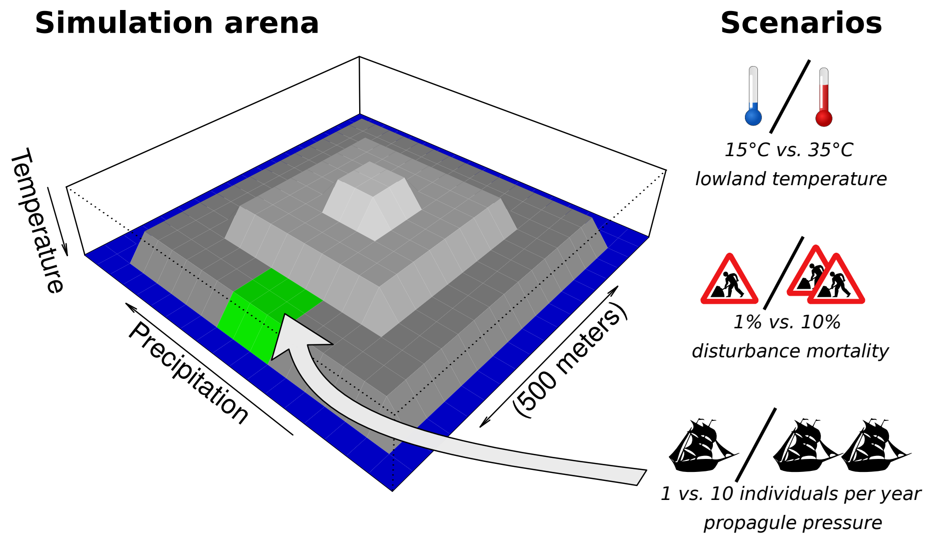
3) How is the source code organised?

4) What are some notable implementation details?

Studies using GeMM

The GeMM plant community model has been used to investigate:

- Effects & evolution of genetic linkage
- Effects of temporal environmental variation
- Factors affecting success of species invasions
- Evolutionary rescue and extinction debt



Vedder et al., *under review*

A new challenge: *Zosterops*

Zosterops: a “great speciator” (Diamond, Gilpin & Mayr, 1976)

- Numerous species all around the Indian Ocean
- Well-studied, though phylogeny is still in flux
- Species delineation difficult; known to hybridise

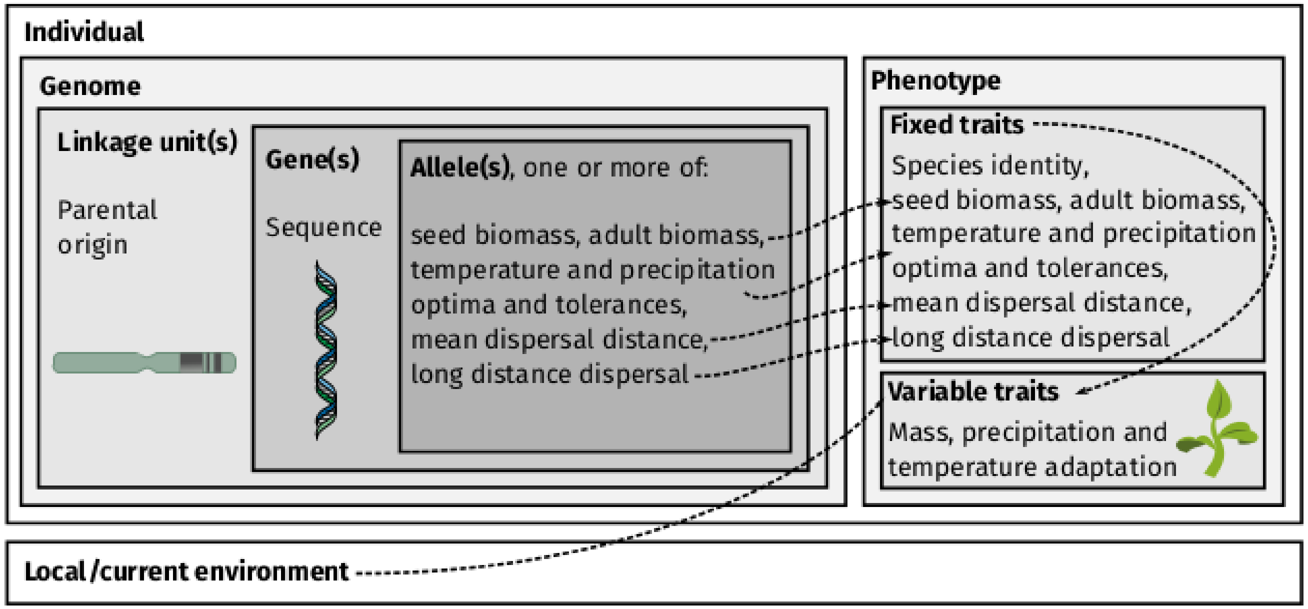
Zosterops abyssinicus – Lip Kee, Wikimedia Commons

Study question

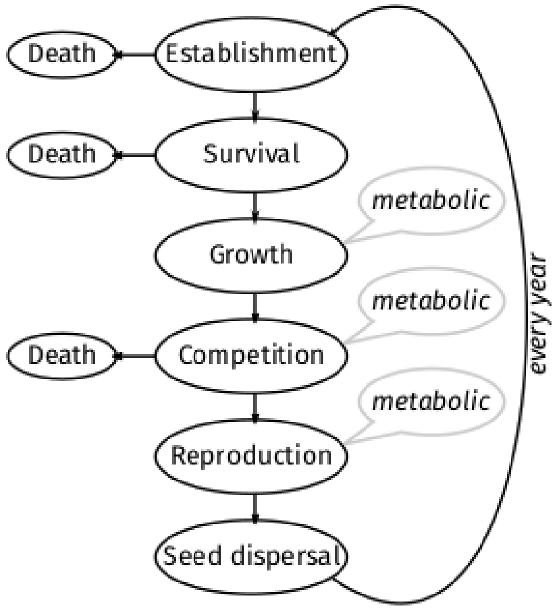
How likely is an evolutionary rescue of *Z. silvanus* through introgressive hybridisation with *Z. jubaensis*?

View of the Taita Hills – Islanti, Wikimedia Commons

GeMM: Genes & Communities



Individual
Genome
Linkage unit(s)
Parental origin
Gene(s)
Sequence
Allele(s), one or more of:
seed biomass, adult biomass, temperature and precipitation optima and tolerances, mean dispersal distance, long distance dispersal
Phenotype
Fixed traits
Species identity, seed biomass, adult biomass, temperature and precipitation optima and tolerances, mean dispersal distance, long distance dispersal
Variable traits
Mass, precipitation and temperature adaptation
Local/current environment



Leidinger et al., *under review*

Strengths of GeMM

- Multidimensional biology: Combines genetic, physiological & life-history processes
- Ecological and evolutionary patterns as emergent properties
- Abstract & generalised, but empirical foundation through use of the MTE
- Good code quality, decent performance

Quick stats:

- ~2500 lines Julia
- ~1100 lines R & Python
- ~1500 commits
- 3.5 years development time

Project organisation

Github repository
https://github.com/lleiding/gemm

README.md
LICENSE.txt

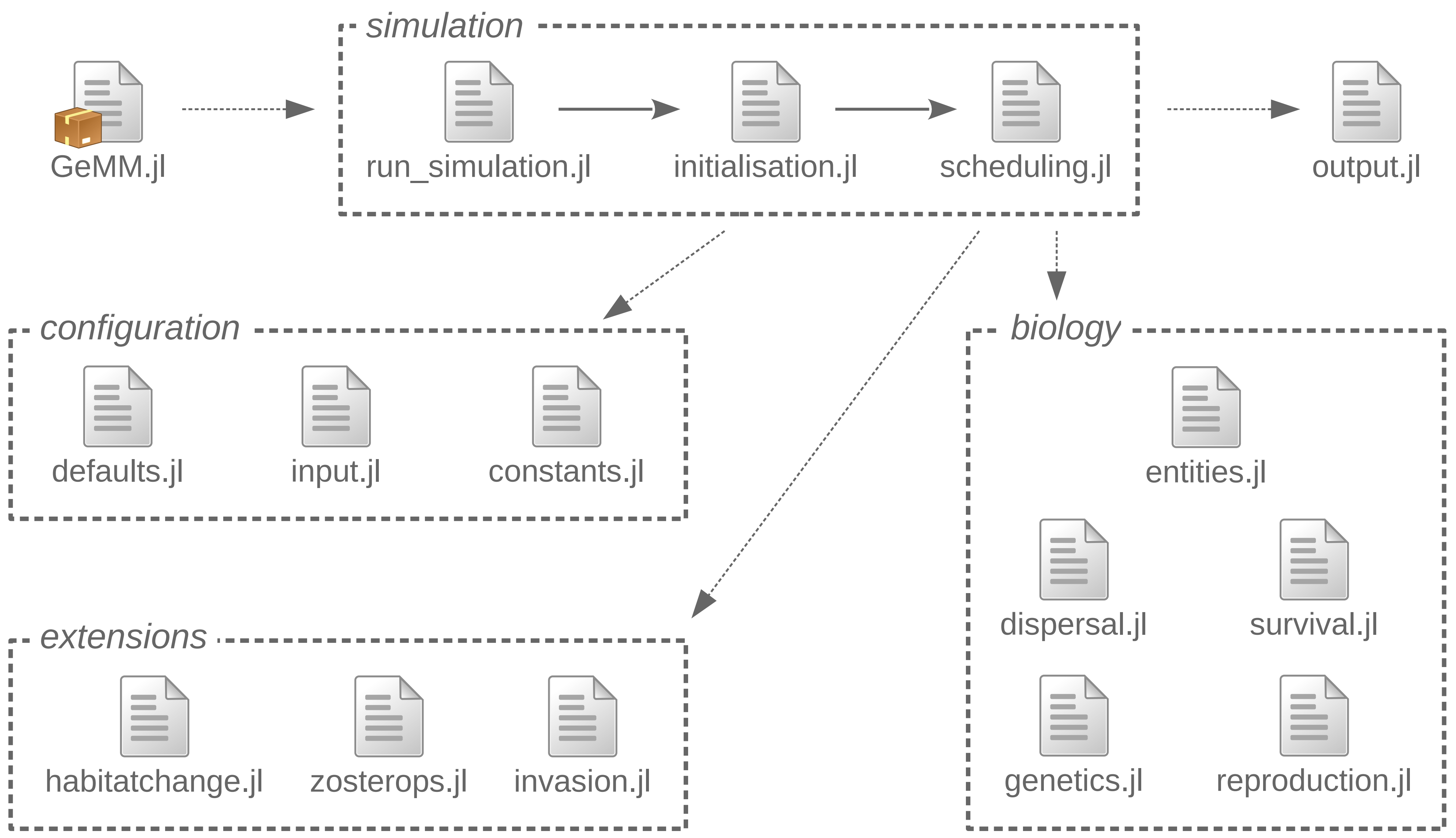
rungemm.jl
rungemmparallel.jl

docs
- ↳ ODD.md
- ↳ architecture.pdf
- ↳ index.html
- ↳ build.sh

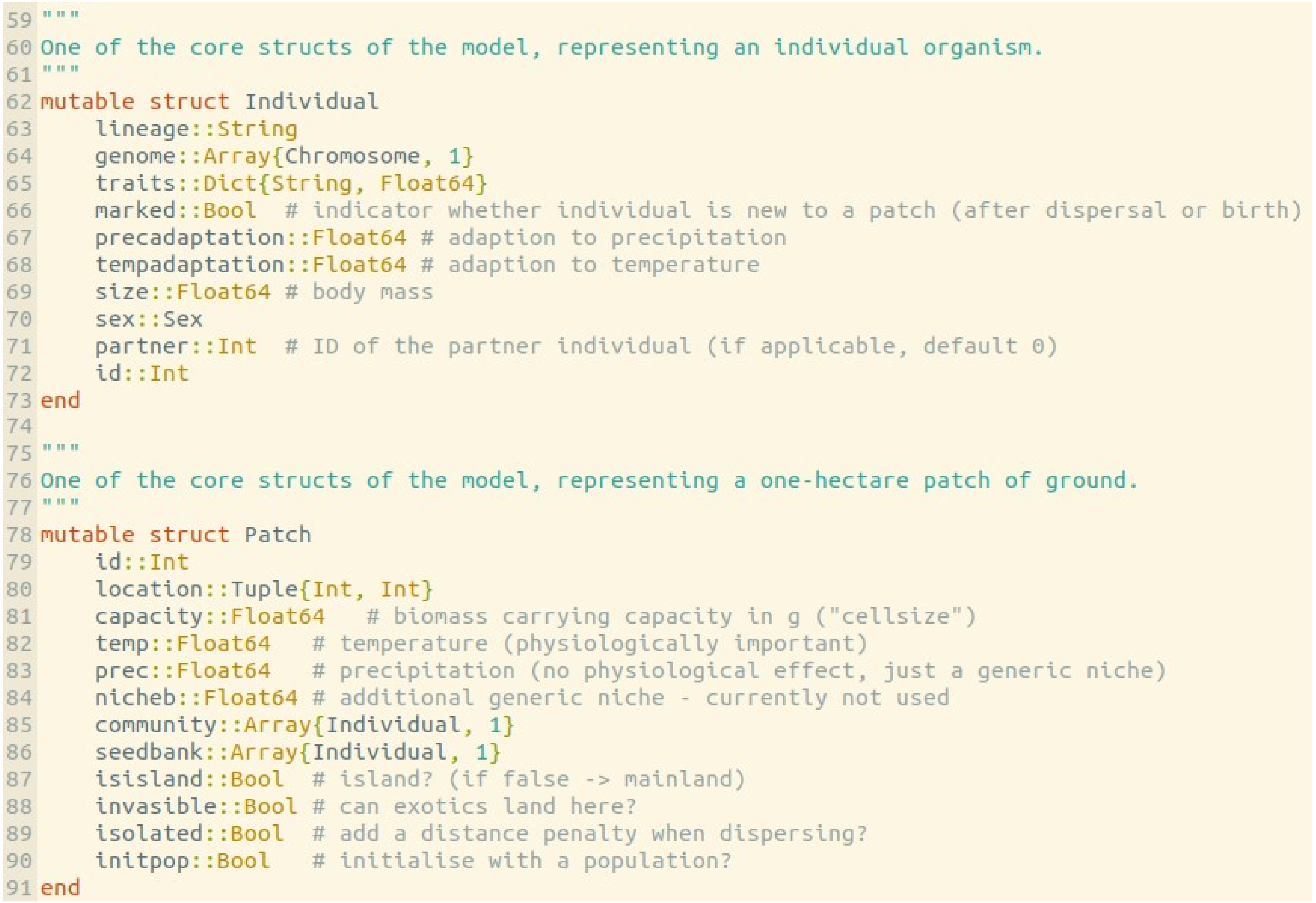
examples
- ↳ gradient
- ↳ invasions
- ↳ islandradiations
- ↳ zosterops

src
- ↳ GeMM.jl
- ↳ defaults.jl
- ↳ genetics.jl
- ↳ …

Source code architecture



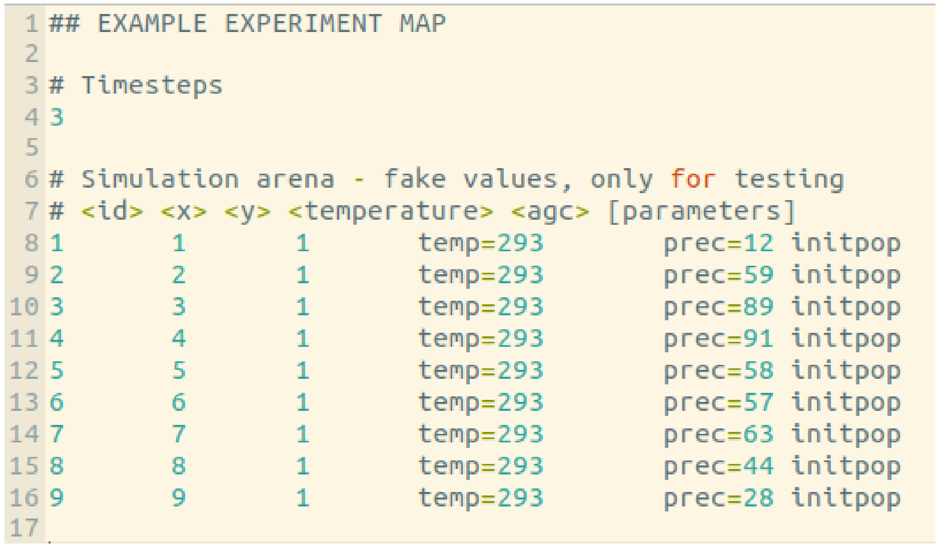
Entities



```julia
"""
One of the core structs of the model, representing an individual organism.
"""
mutable struct Individual
    lineage::String
    genome::Array{Chromosome, 1}
    traits::Dict{String, Float64}
    marked::Bool  # indicator whether individual is new to a patch (after dispersal or birth)
    precadaptation::Float64 # adaption to precipitation
    tempadaptation::Float64 # adaption to temperature
    size::Float64 # body mass
    sex::Sex
    partner::Int  # ID of the partner individual (if applicable, default 0)
    id::Int
end

"""
One of the core structs of the model, representing a one-hectare patch of ground.
"""
mutable struct Patch
    id::Int
    location::Tuple{Int, Int}
    capacity::Float64   # biomass carrying capacity in g ("cellsize")
    temp::Float64   # temperature (physiologically important)
    prec::Float64   # precipitation (no physiological effect, just a generic niche)
    nicheb::Float64 # additional generic niche - currently not used
    community::Array{Individual, 1}
    seedbank::Array{Individual, 1}
    isisland::Bool  # island? (if false -> mainland)
    invasible::Bool # can exotics land here?
    isolated::Bool  # add a distance penalty when dispersing?
    initpop::Bool   # initialise with a population?
end
```

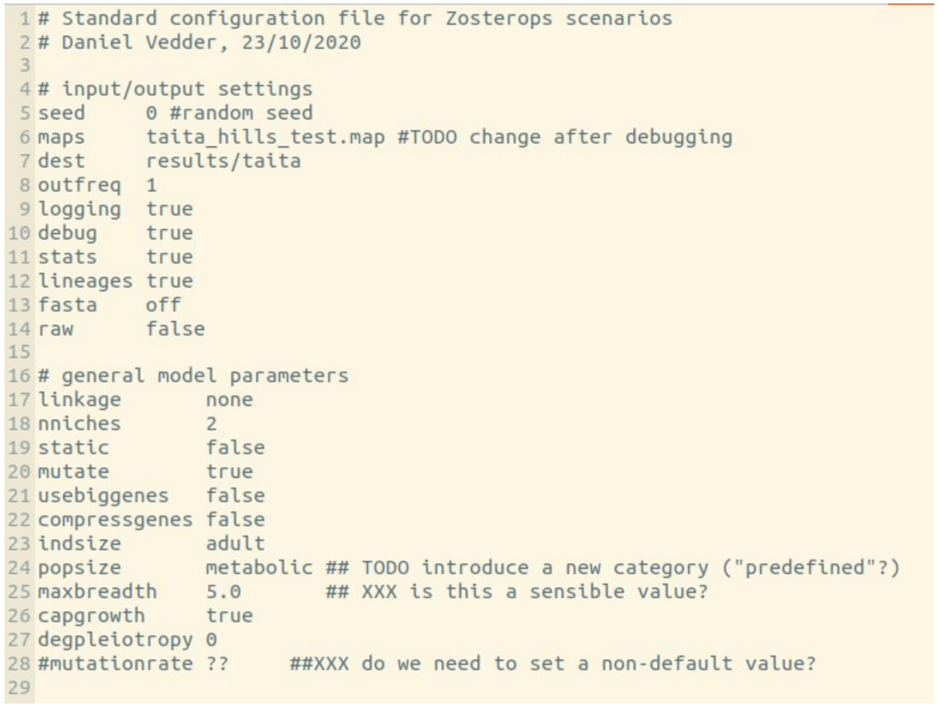
(also: traits, genes, chromosomes)

Configuration



```
## EXAMPLE EXPERIMENT MAP

# Timesteps
3

# Simulation arena - fake values, only for testing
# <id> <x> <y> <temperature> <agc> [parameters]
1       1       1       temp=293        prec=12 initpop
2       2       1       temp=293        prec=59 initpop
3       3       1       temp=293        prec=89 initpop
4       4       1       temp=293        prec=91 initpop
5       5       1       temp=293        prec=58 initpop
6       6       1       temp=293        prec=57 initpop
7       7       1       temp=293        prec=63 initpop
8       8       1       temp=293        prec=44 initpop
9       9       1       temp=293        prec=28 initpop

```

map file



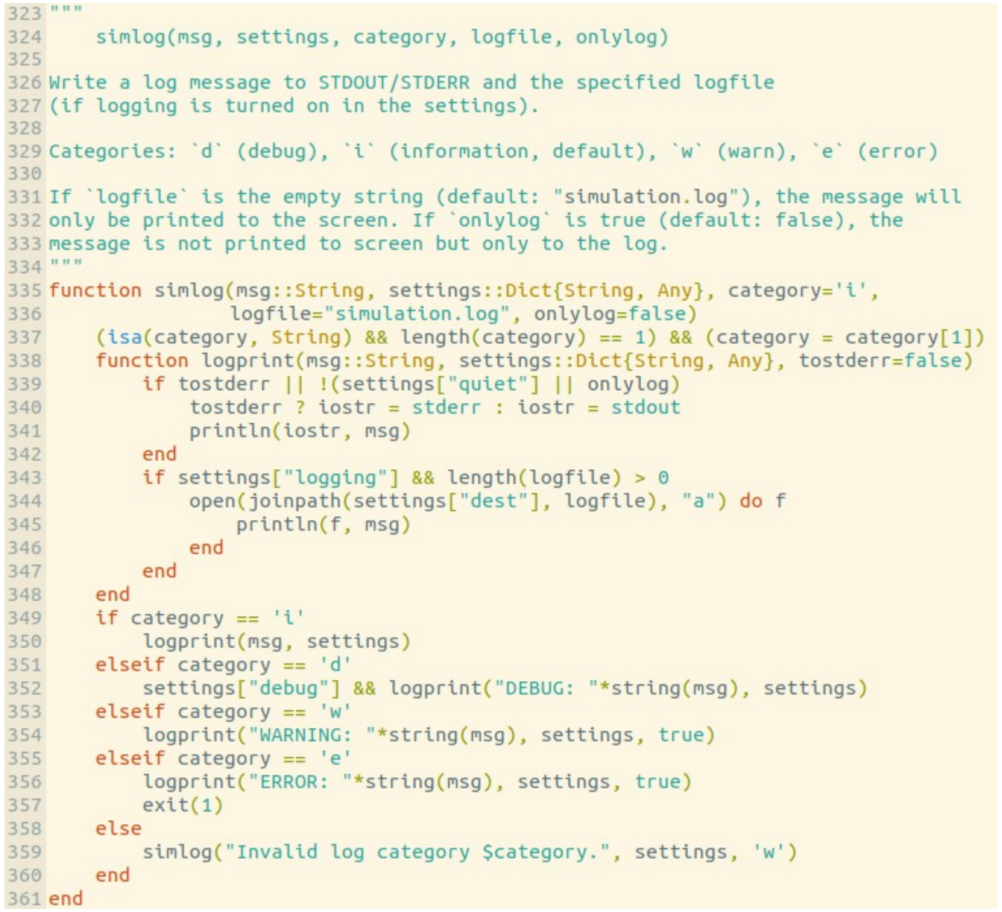
```
# Standard configuration file for Zosterops scenarios
# Daniel Vedder, 23/10/2020

# input/output settings
seed     0 #random seed
maps     taita_hills_test.map #TODO change after debugging
dest     results/taita
outfreq  1
logging  true
debug    true
stats    true
lineages true
fasta    off
raw      false

# general model parameters
linkage       none
nniches       2
static        false
mutate        true
usebiggenes   false
compressgenes false
indsize       adult
popsize       metabolic ## TODO introduce a new category ("predefined"?)
maxbreadth    5.0       ## XXX is this a sensible value?
capgrowth     true
degpleiotropy 0
#mutationrate ??     ##XXX do we need to set a non-default value?

```

configuration file

`getsettings()` order of precedence:

library calls → *commandline parameters* → *configuration file* → *default values*

Output

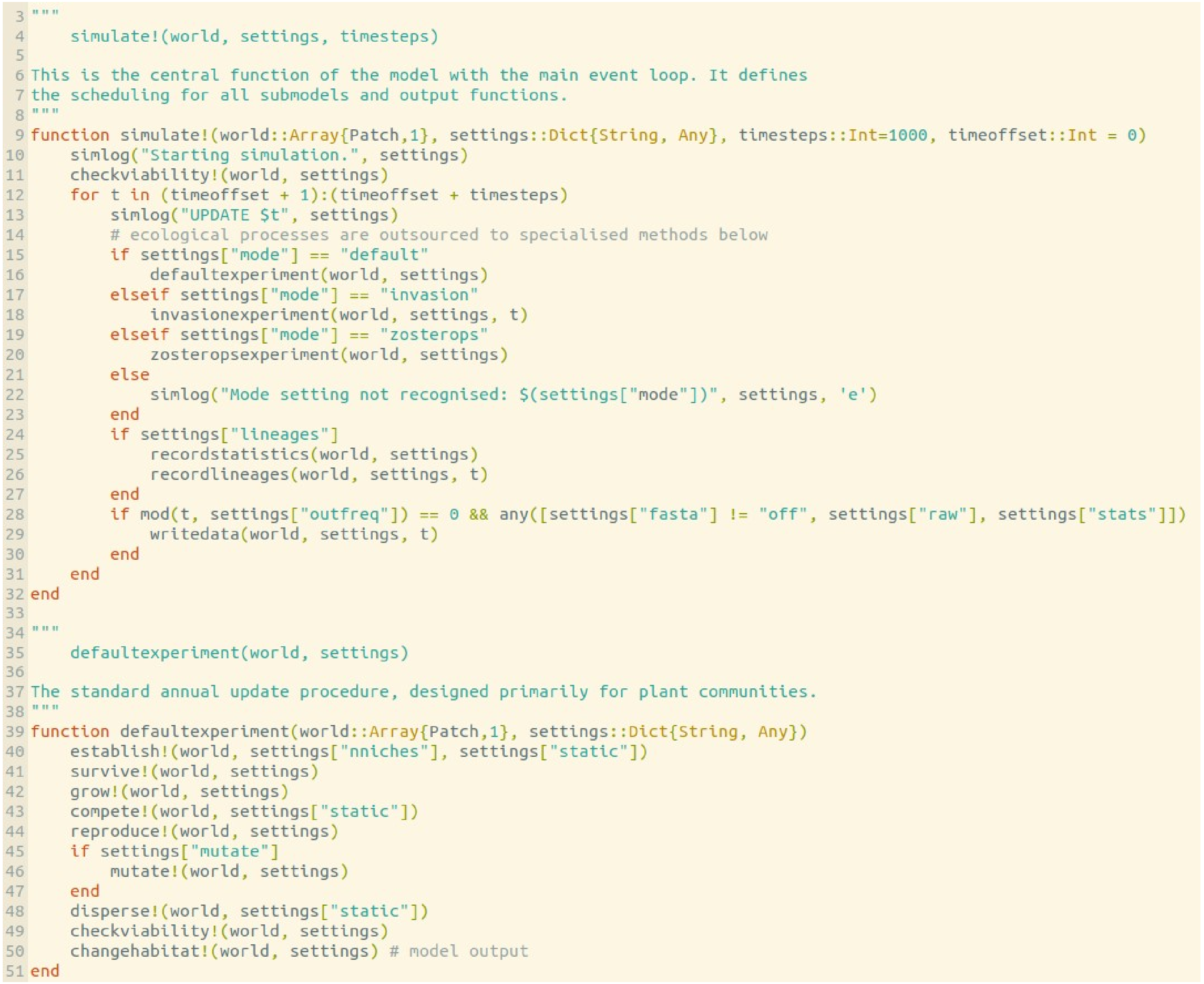


```julia
"""
    simlog(msg, settings, category, logfile, onlylog)

Write a log message to STDOUT/STDERR and the specified logfile
(if logging is turned on in the settings).

Categories: `d` (debug), `i` (information, default), `w` (warn), `e` (error)

If `logfile` is the empty string (default: "simulation.log"), the message will
only be printed to the screen. If `onlylog` is true (default: false), the
message is not printed to screen but only to the log.
"""
function simlog(msg::String, settings::Dict{String, Any}, category='i',
                logfile="simulation.log", onlylog=false)
    (isa(category, String) && length(category) == 1) && (category = category[1])
    function logprint(msg::String, settings::Dict{String, Any}, tostderr=false)
        if tostderr || !(settings["quiet"] || onlylog)
            tostderr ? iostr = stderr : iostr = stdout
            println(iostr, msg)
        end
        if settings["logging"] && length(logfile) > 0
            open(joinpath(settings["dest"], logfile), "a") do f
                println(f, msg)
            end
        end
    end
    if category == 'i'
        logprint(msg, settings)
    elseif category == 'd'
        settings["debug"] && logprint("DEBUG: "*string(msg), settings)
    elseif category == 'w'
        logprint("WARNING: "*string(msg), settings, true)
    elseif category == 'e'
        logprint("ERROR: "*string(msg), settings, true)
        exit(1)
    else
        simlog("Invalid log category $category.", settings, 'w')
    end
end
```

Notable functions:

- simlog()
- setupdatadir()
- writesettings()
- recordstatistics()
- recordlineages()
- dumpinds()
- makefasta()

Scheduling



```julia
"""
    simulate!(world, settings, timesteps)

This is the central function of the model with the main event loop. It defines
the scheduling for all submodels and output functions.
"""
function simulate!(world::Array{Patch,1}, settings::Dict{String, Any}, timesteps::Int=1000, timeoffset::Int = 0)
    simlog("Starting simulation.", settings)
    checkviability!(world, settings)
    for t in (timeoffset + 1):(timeoffset + timesteps)
        simlog("UPDATE $t", settings)
        # ecological processes are outsourced to specialised methods below
        if settings["mode"] == "default"
            defaultexperiment(world, settings)
        elseif settings["mode"] == "invasion"
            invasionexperiment(world, settings, t)
        elseif settings["mode"] == "zosterops"
            zosteropsexperiment(world, settings)
        else
            simlog("Mode setting not recognised: $(settings["mode"])", settings, 'e')
        end
        if settings["lineages"]
            recordstatistics(world, settings)
            recordlineages(world, settings, t)
        end
        if mod(t, settings["outfreq"]) == 0 && any([settings["fasta"] != "off", settings["raw"], settings["stats"]])
            writedata(world, settings, t)
        end
    end
end

"""
    defaultexperiment(world, settings)

The standard annual update procedure, designed primarily for plant communities.
"""
function defaultexperiment(world::Array{Patch,1}, settings::Dict{String, Any})
    establish!(world, settings["nniches"], settings["static"])
    survive!(world, settings)
    grow!(world, settings)
    compete!(world, settings["static"])
    reproduce!(world, settings)
    if settings["mutate"]
        mutate!(world, settings)
    end
    disperse!(world, settings["static"])
    checkviability!(world, settings)
    changehabitat!(world, settings) # model output
end
```

Thank you for your attention!

Any questions?