# README
## Population and disease modeling

March 28, 2021

# Contents



# 1 Brief introduction

The R code featured here reproduces the analyses carried out for the manuscript Colchero *et al.* (in prep) in which we modeled the potential impact of COVID-19 on the population dynamics of mountain gorillas.

# 2 Configuration

1. **Configurations:**

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave Version 10.14.6
Running time: 1 minute.
```

2. **Special hardware:** None.

# 3 Analysis R code

## 3.1 File structure

The project is ogranized into four main folders:

- `01docs`: with subfolder `README` containing this file;
- `02code`: containing two R files, `popDiseaseFunctions.R` which contains all the functions used for analysis, and `popDiseaseRuns.R` with the code to be run by the user for analysis.
- `03data`: with subfolders `rdata` and `tables`;
- `04results`: with subfolders `plots`, `rdata`, and `tables`. The directory `rdata` contains the file `testOutput.RData` with the test outputs featured in this file.

## 3.2 Setting up the analysis

To run analysis, it is enough to run and modify the code in file `popDiseaseFunctions.R`. Here are some general details on how to run it. If you are interested on running multiple simulations, it is recommended to install and load the package `snowfall` [1]. Also, the plotting function uses the package `RColorBrewer` [2] to assign colors to the different stages.

```
# Install libraries:
install.packages(c("snowfall", "RColorBrewer"))

# Load libraries:
library(snowfall)
library(RColorBrewer)
```

Next, point R to the directory where you have stored the `popDisease` R project:

```
# Working directory:
setwd("Path to main directory/popDisease/")
```

Load the functions for analysis:

```
# Sourced files:
source("02code/popDiseaseFunctions.R")
```

Next is to specify the age-specific mortality parameters for females and males. These are based on the Siler mortality function [3], where the hazards rate is given by

$$\mu(x) = \exp(a_0 - a_1\ x) + c + \exp(b_0 + b_1\ x) \tag{1}$$

where $a_0, b_0 \in \mathbb{R}$, $c, a_1, b_1 \geq 0$ are the mortality parameters.

The parameters for the age-specific fecundity are based on the model by Muller *et al.* [4] and Colchero *et al.* [5], where the expected value of the number of offspring produced by female per age is given by

$$g(x) = \exp\left[\beta_0 + \beta_1(x-\alpha) - \beta_2(x-\alpha)^2 + \frac{\beta_3}{x-\alpha+1}\right], \tag{2}$$

where $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_3]$ are parameters where $\beta_0, \beta_3 \in \mathbb{R}$ and $\beta_1, \beta_2 \geq 0$ and $\alpha$ is the age at sexual maturity.

As default, we use the estimated parameters for mountain gorillas:

```
# Mortality parameters:
theta <- cbind(F = c(a0 = -0.229, a1 = 1.882, c = 0.013,
                     b0 = -10.275, b1 = 0.223),
               M = c(a0 = 0.004, a1 = 2.382, c = 0.021,
                     b0 = -7.52, b1 = 0.189))

# Reproduction parameters:
beta <- c(r0 = -1.216, r1 = 0.022, r2 = 0.001, r3 = -0.972)
```

Next is to construct the demographic object that will serve as main input for the stochastic simulation function:

```
# Demographic object:
demog <- CreateDemoObj(theta = theta, beta = beta, N = c(F = 60, M = 50),
                       omega = 60, alpha = 8, propM = 0.5, yearIni = 2021,
                       R0 = 2, qMax = 0.3, maxImPr = 0.8, immDur = 3)
```

Argument `N` provides a general initial number for females and males, although the simulated numbers will be randomly drawn. Arguments $\omega$ and $\alpha$ are the maximum possible age and the age at female sexual maturity, respectively. Argument `propM` is the average proportion of males born per reproductive event (i.e. sex ratio), `yearIni` is the initial year (for plotting purposes). Next are the disease specific variables, where `R0` is the reproductive number (average number of infected individuals per new infection), `qMax` is the maximum infected mortality probability, `maxImPr` is the maximum immunity probability, and `immDur` is the immunity duration in months.

## 3.3 Single run

Next, define the initial time and the final number of years, as well as the number of initial infections:

```
# Start and end times and sequence of time steps:
t0 <- 0
```

```r
tFin <- 10

# Initial number of infected individuals:
infIni <- 1
```

To run a single simulation, use function `ProjPopSIRS()` as follows:

```r
# Run function:
disProj <- ProjPopSIRS(tFin = tFin, demog = demog, infIni = infIni)
```

The results can be visualized by using the buil-in function plot:

```r
# Plot results:
plotSIRS(disProj)
```

which produces the plot in Fig.1, which shows the time series of each one of the four stages: The plot can be modified to show the progression stacked per stage as (Fig. 2:

```r
# Plot results:
plotSIRS(disProj, type = "ab")
```

## 3.4 Multiple runs

In order to run multiple analyses and test different levels of the epidemiological variables, first decide which levels for each of these variables you want to test:

```r
# Vary R0:
R0vec <- c(0.5, 1, 2, 3)

# Vary maximum immunity:
maxImProb <- c(0.2, 0.4, 0.6, 0.8)

# Vary immunity duration:
immDurVec <- c(1, 3, 6, 12)

# Vary maximum mortality:
maxQxVec <- c(0.3, 0.6)
```

Next the code produces a matrix with all the combinations of these variables:

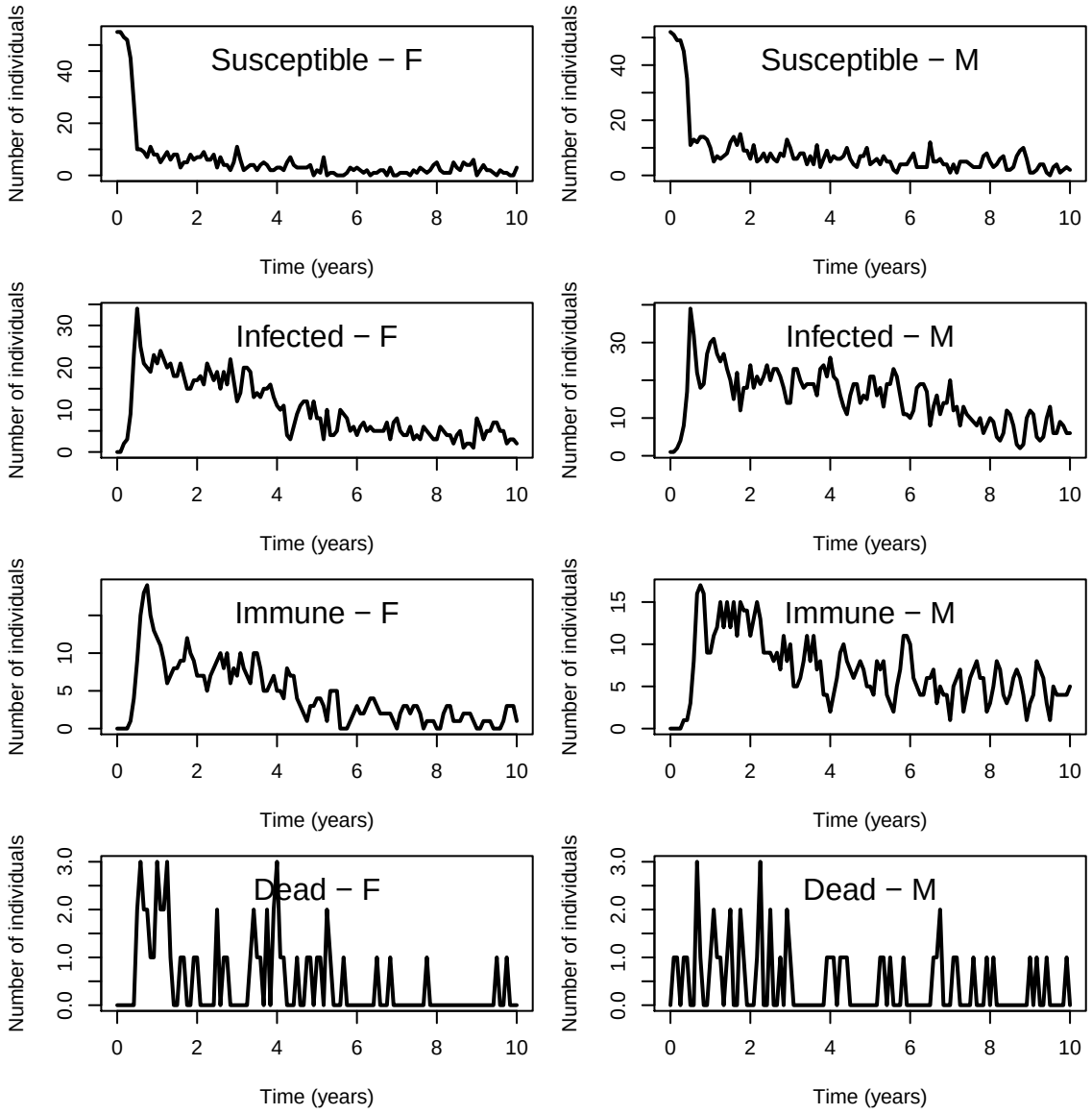


Figure 1: Time series for each of the four stages by sex.

```
# Grid of new values:
covidVars <- expand.grid(R0 = R0vec, MImProb = maxImProb,
                         immDur = immDurVec, maxQx = maxQxVec)
```

Note that with the values chosen here, there are 128 different combinations which, if you are to run 2000 simulations per combination (total of 256000 runs), the model can take quite a long time. For instance, on a MacBook Pro 2017, a single run as shown above takes 0.7 seconds, which implies that, following the parallel routine below with 4 CPUS, the model may take about 12 hours to complete.

for illustration purposes, here we provide an example with the following settings:

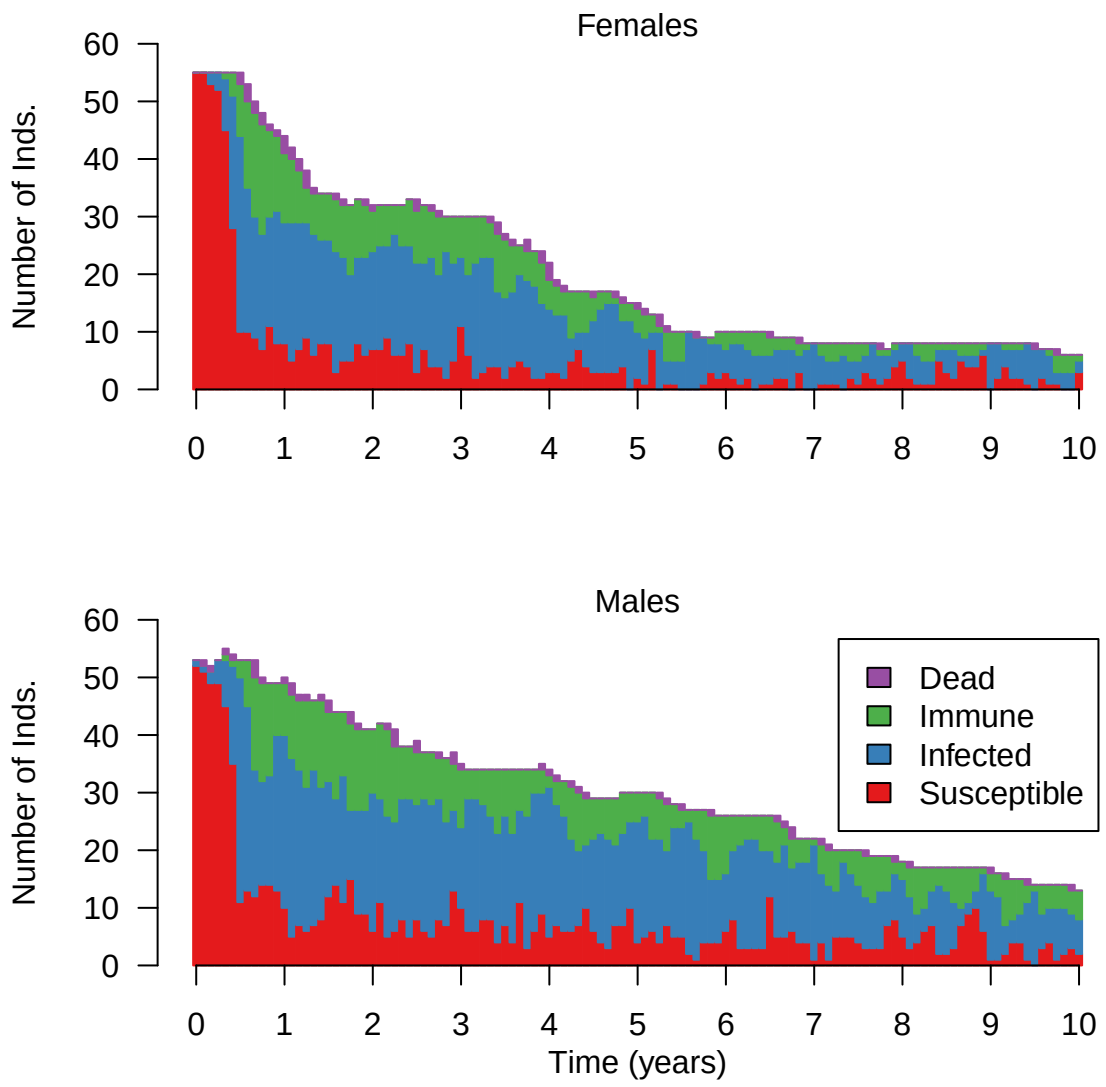


Figure 2: Stacked numbers of individuals per stage per sex.

```
# Vary R0:
R0vec <- c(1, 3)

# Vary maximum immunity:
maxImProb <- c(0.2)

# Vary immunity duration:
immDurVec <- c(3)
```

```
# Vary maximum mortality:
maxQxVec <- c(0.3)

# Grid of new values:
covidVars <- expand.grid(R0 = R0vec, MImProb = maxImProb, immDur = immDurVec,
                         maxQx = maxQxVec)
print(covidVars)

>   R0 MImProb immDur maxQx
> 1  1     0.2      3   0.3
> 2  3     0.2      3   0.3
```

Then, to run multiple simulations in parallel use the following code (in the `popDiseaseRuns.R` file):

```
# Output:
outMulti <- list()

for (vv in 1:nrow(covidVars)) {
  # Create new demog object:
  demogn <- demog

  # Replace reproductive value:
  demogn$disease$R0 <- covidVars$R0[vv]

  # Replace immunity probability
  demogn$disease$immProb <- covidVars$MImProb[vv] /
    (1 + exp(-0.1 * (demog$x - demog$settings["omega"] * 0.4)))

  # Replace immunity duration:
  demogn$disease$immDur <- covidVars$immDur[vv]

  # Mortality probability:
  demogn$disease$qx <- covidVars$maxQx[vv] /
    (1 + exp(-0.2 * (demog$x - demog$settings["omega"] * 0.4)))

  # Run multiple models in parallel:
  outParal <- ParalProjSIRS(nsim = 2000, ncpus = 4, tFin = 10,
                            demog = demogn, infIni = infIni)

  # Extract average values:
  meanOutParal <- ExtractParalSIRS(outParal = outParal)
```

```
  # Plot barplots:
  cat(sprintf("%s - R0 = %s; ImmPr = %s; ImmDur = %s\n", vv,
              covidVars$R0[vv],
              covidVars$MImProb[vv], covidVars$immDur[vv]))
  plotSIRS(meanOutParal, type = "bp")

  # Store results:
  outMulti[[vv]] <- meanOutParal
}
```

The output list `outMulti` has the same number of sub-lists as the number of scenarios (i.e. combinations of the epidemiological variables). For this example, it is only two. Each sub-list can be called as, for example for the first scenario:

```
outMulti[[1]]
```

The outputs of each scenario can be ploted as shown in Fig. 3:

Bear in mind that the sub-list contains several large matrices, so we recommend not to print them to the console. Each sublist includes the following variables:

- `settings`: A vector with the simulation settings, namely the number of simulations ("nsim"), the final year ("tFin"), the number of years ("nt"), and the initial number of infected individuals ("infIni");
- `demog`: The demographic object as used for the scenario;
- `statesMean`: a list with arrays per stage (Succeptible, Infected, Immune, Dead) of the average number of individuals per age, time step and sex. Each array has dimensions $n_x \times n_t \times n_s$ where $n_x$ is the number of ages in months, $n_t$ is the number of time steps in months, and $n_s$ is the number of sexes (i.e. "F" and "M");
- `statesVar`: a list with arrays per stage (Succeptible, Infected, Immune, Dead) of the variance in the number of individuals per age, time step and sex. The dimensions are the same as for `statesMean`;
- `nStates`: Numerical value with the number of states (i.e. 4);
- `stateName`: Character vector with the states names;
- `tseq`: Numerical vector with the sequence of time steps in years (increments are by month fractions);
- `Mt`: Matrix with summary values of the number of individuals in the population at each time step. The rows correspond to the mean, 2.5%, 33%, 50%, 67%, and 97.5%;

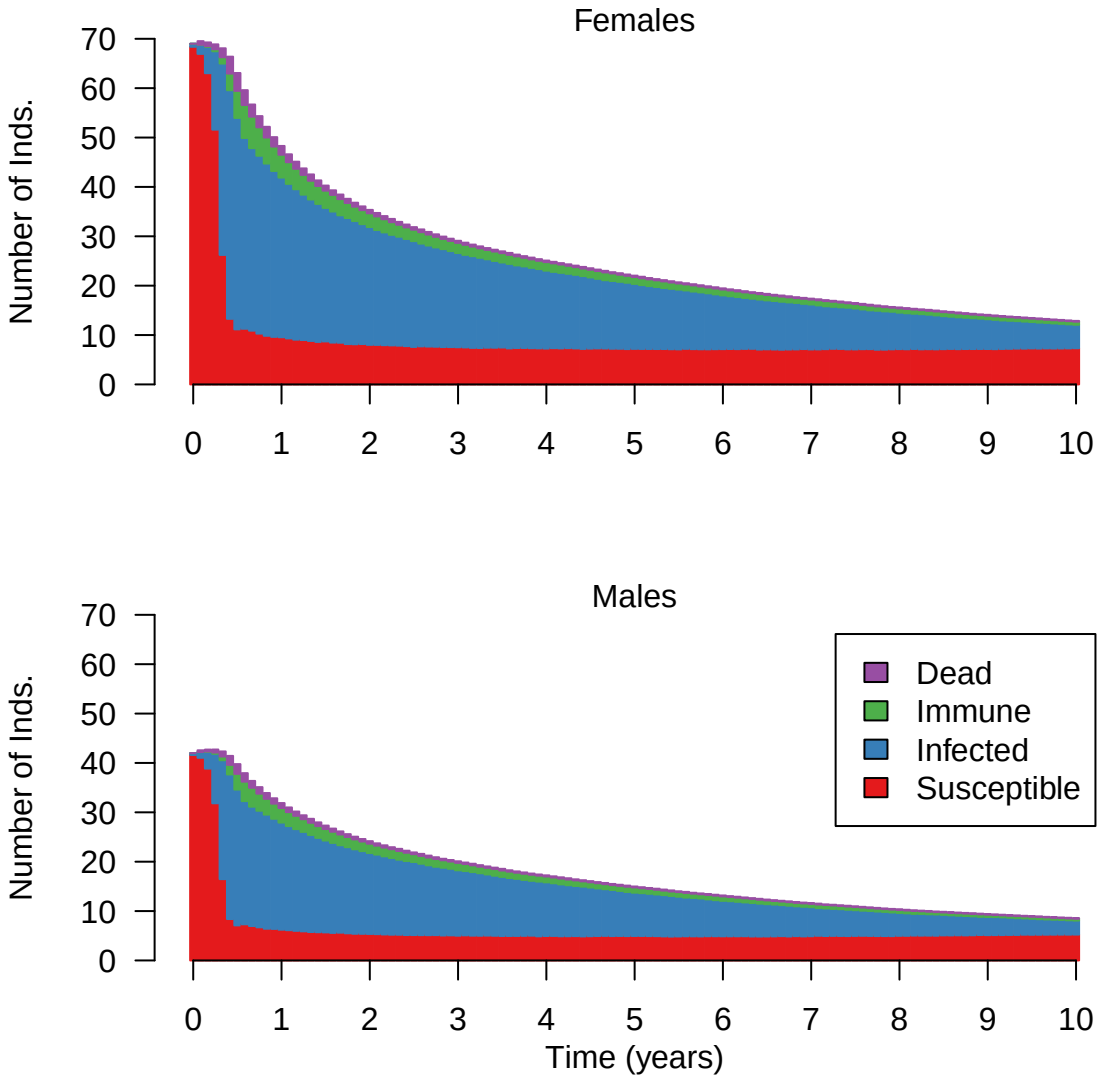


Figure 3: average number of individuals per stage from multiple runs.

- `Mtmat`: Raw matrix of number of individuals in the population at each time step for each run (rows);
- `nExt`: Numerical vector with the number of populations that went extinct per time step.

# References


[1] Knaus, J. *snowfall: Easier cluster computing (based on snow).* (2015). URL `https://CRAN.R-project.org/package=snowfall`. R package version 1.84-6.1.

[2] Neuwirth, E. *RColorBrewer: ColorBrewer Palettes* (2014). URL `https://CRAN.R-project.org/package=RColorBrewer`. R package version 1.1-2.

[3] Siler, W. A competing-risk model for animal mortality. *Ecology* **60**, 750 – 757 (1979). URL `http://www.jstor.org/stable/1936612`.

[4] Muller, M. N. *et al.* Sexual dimorphism in chimpanzee (Pan troglodytes schweinfurthii) and human age-specific fertility. *Journal of human evolution* **144**, 102795 (2020).

[5] Colchero, F., Eckardt, W. & Stoinski, T. Evidence of demographic buffering in an endangered great ape: social buffering on immature survival and the role of refined sex-age-classes on population growth rate. *Journal of Animal Ecology* (2021).