Benchmarking

```
m <- matrix(rnorm(1e6), ncol=100)
f1 <- function(x, t = 0.3) {
  xx <- 0
  for (i in 1:nrow(x)) {
    xx <- c(xx, sum(m[i, ]))
  }
  mean(xx, trim = t)
}
f2 <- function(x, t = 0.3) mean(rowSums(x), trim = t)

library(rbenchmark)
benchmark(f1(m), f2(m),
          columns=c("test", "replications",
            "elapsed", "relative"),
          order = "relative", replications = 10)
```

Outline

Lotka–Volterra equations

Describe simple models of populations dynamics of species competing for some common resource. When two species are not interacting, their population evolve according to the logistic equations and the rate of reproduction is proportional to both the existing population and the amount of available resources

$$\frac{\partial x}{\partial t} = r_1 x \; (1 - \frac{x}{k_1})$$
$$\frac{\partial y}{\partial t} = r_2 y \; (1 - \frac{y}{k_2})$$

where the constant r_i defines the growth rate and k_i is the carrying capacity of the environment.

Competitive Lotka–Volterra equations

When competing for the same resource, the animals have a negative influence on their competitors growth.

$$\frac{\partial x}{\partial t} = r_1 x \; (1 - \frac{x}{k_1}) - axy$$
$$\frac{\partial y}{\partial t} = r_2 y \; (1 - \frac{y}{k_2}) - bxy$$

Rabbits vs sheep (Strogatz, p155)

Here is an example with $r_1 = 3$, $k_1 = 3$, $a = 2$, $r_2 = 2$, $k_2 = 2$, $b = 1$, which simplifies to

$$\frac{\partial r}{\partial t} = r(3 - r - 2s)$$
$$\frac{\partial s}{\partial t} = s(2 - r - s)$$

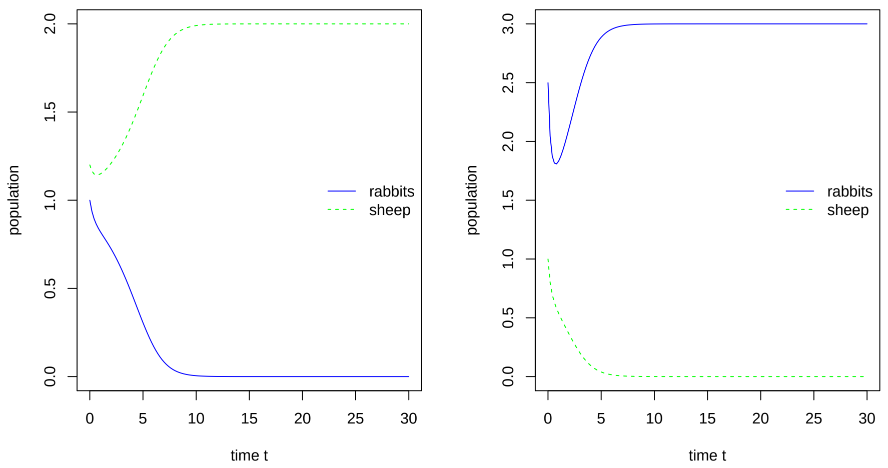
Computing a trajectory over time

i.e. use numerical integration, with $r_0 = 1$ and $s_0 = 1.2$

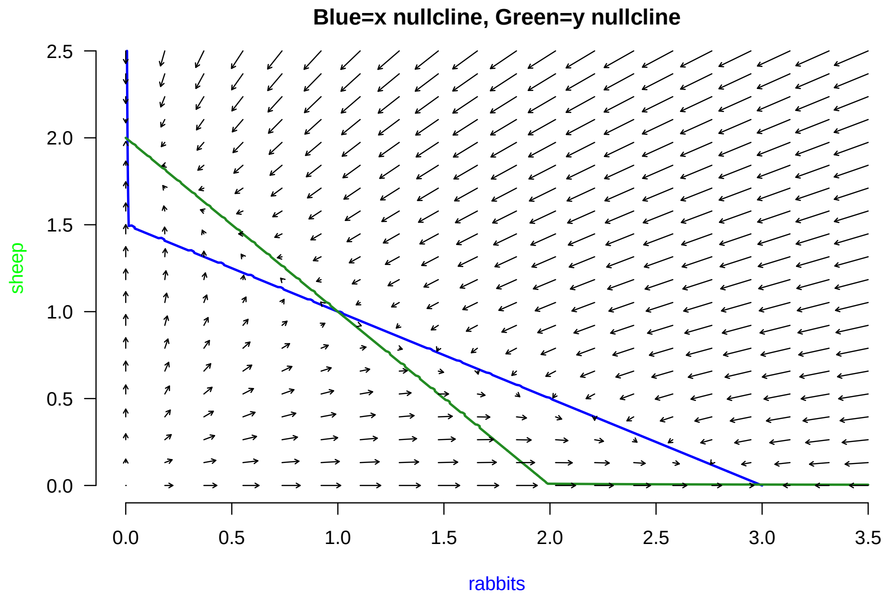
```
library(deSolve)
Sheep <- function(t, y, parms) {
  r=y[1]; s=y[2]
  drdt = r * (3 - r - (2*s))
  dsdt = s * (2 - r - s)
  list(c(drdt, dsdt))
}

x0 <- c(1, 1.2)
times <- seq(0, 30, by=0.2)
parms <- 0
out <- rk4(x0, times, Sheep, parms)
head(out)
```

Plotting population growth



Phase plane analysis



Starting points

- `deSolve` package
- phase planes and nullclines (`DMBpplane.r` from DMB site, modified from Daniel Kaplan)
- `integrate()` – quadrature
- `D()` – symbolic differentiation
- `optimize()` (1d) and `optim()` (n-d)
- Steven Strogatz. Nonlinear dynamics and chaos.
- NR: William Press et al. Numerical Recipes in C/C++
- More slides about DE and phase plane – `de.pdf`

Outline

Conclusions

- Looking for packages
  - CRAN Task Views http://cran.r-project.org/web/views/
  - Bioconductor biocViews http://bioconductor.org/packages/release/BiocViews.html
- Reproducibilty is crutial
- Have several tools at hand
  - editor, programming languages, shell, ...
- Practice to keep learning
- Have fun! ☺