# Übung 03: Pandemieausbruch

Tobias Blesgen und Leonardo Thome

09.06.2021

Im Folgenden wollen wir ein Simulierte Pandemieausbreitung von SARS-CoV-2 für verschiedene Parameter im SIR Modell betrachten. Das SIR Modell beschreibt den zeitlichen Verlauf mit folgendem Differentialgleichungssystem.

$$S'(t) = -\frac{\beta}{N} S(t) I(t) - \Gamma(t) + \delta V(t) \tag{1}$$

$$I'(t) = \frac{\beta}{N} S(t) I(t) - \alpha I(t) \tag{2}$$

$$R'(t) = \alpha I(t) \tag{3}$$

$$V'(t) = \Gamma(t) - \delta V(t) \tag{4}$$

```cpp
#include<Rcpp.h>
#include<stdlib.h>
#include<stdio.h>
#include <vector>

using namespace Rcpp;

typedef struct
{
    double S, I, R, V;
} Status;

typedef struct
{
    double alpha, beta, delta, N, Gamma;
} Parameter;

template<typename T>
std::vector<T> slice(std::vector<T> const &v, int m, int n)
{
    auto first = v.cbegin() + m;
    auto last = v.cbegin() + n + 1;

    std::vector<T> vec(first, last);
    return vec;
}
```

```
void f(Status alterStatus, Parameter parameter, Status& neuerStatus){
    neuerStatus.V = parameter.Gamma - parameter.delta * alterStatus.V;
    neuerStatus.R = parameter.alpha * alterStatus.I;
    neuerStatus.I = parameter.beta * alterStatus.S * alterStatus.I/parameter.N
      - parameter.alpha * alterStatus.I;
    neuerStatus.S = -parameter.beta * alterStatus.S * alterStatus.I/parameter.N
      - parameter.Gamma + parameter.delta * alterStatus.V;
}

void rkSchritt(Status& status, Parameter parameter, double h){
    Status fStatus;
    f(status, parameter, fStatus);
    Status f2Status;
    Status gemischt = {.S = status.S + h*fStatus.S, .I = status.I
      + h*fStatus.I, .R = status.R + h*fStatus.R, .V = status.V + h*fStatus.V};
    f(gemischt, parameter, f2Status);
    status.V = status.V + h/2*(fStatus.V + f2Status.V);
    status.R = status.R + h/2*(fStatus.R + f2Status.R);
    status.S = status.S + h/2*(fStatus.S + f2Status.S);
    status.I = status.I + h/2*(fStatus.I + f2Status.I);
}

//[[Rcpp::export]]
Rcpp::List durchlauf(const int maxSchritte, const double h,
                           const double S, const double I, const double R,
                           const double V, const double alpha,
                           const double beta, const double delta,
                           const double N, const double Gamma){
  // Array der Werte:
    std::vector<double> xValue(maxSchritte);
    std::vector<double> SValue(maxSchritte);
    std::vector<double> IValue(maxSchritte);
    std::vector<double> RValue(maxSchritte);
    std::vector<double> VValue(maxSchritte);
  // Quelltext
  Status status = {.S = S, .I = I, .R = R, .V = V};
  Parameter parameter = {.alpha = alpha, .beta = beta, .delta = delta, .N = N, .Gamma = Gamma};

    for (int i = 0; i < maxSchritte; i++){
      xValue[i] = i*h;
      SValue[i] = status.S;
      IValue[i] = status.I;
      RValue[i] = status.R;
      VValue[i] = status.V;
      rkSchritt(status, parameter, h);

      if(status.I < 1 && i>10){
          // Rückgabe für eine grafische Wiedergabe
          return List::create(Named("x") = slice(xValue,0,i),
                              Named("S") = slice(SValue,0,i),
                              Named("I") = slice(IValue,0,i),
                              Named("R") = slice(RValue,0,i),
                              Named("V") = slice(VValue,0,i));
```

```
      }
    }
  // Rückgabe für eine grafische Wiedergabe
    return List::create(Named("x") = xValue, Named("S") = SValue, Named("I") = IValue, Named("R") = RVal
}
```