base_general.stan

W tym pliku mamy opis danych wejściowych, parametrów modelu i obliczenia przewidywanych śmierci i zakażeń.

base_general.stan

Check on Save

```stan
data {
  int <lower=1> M; // number of countries
  int <lower=1> P; // number of covariates
  int <lower=1> N0; // number of days for which to impute infections
  int<lower=1> N[M]; // days of observed data for country m. each entry must be <= N2
  int<lower=1> N2; // days of observed data + # of days to forecast
  int cases[N2,M]; // reported cases
  int deaths[N2, M]; // reported deaths -- the rows with i > N contain -1 and should be ignored
  matrix[N2, M] f; // h * s
  matrix[N2, P] X[M];
  int EpidemicStart[M];
  real pop[M];
  real SI[N2]; // fixed pre-calculated SI using emprical data from Neil
}

parameters {
  real<lower=0> mu[M]; // intercept for Rt
  real<lower=0> alpha_hier[P]; // sudo parameter for the hier term for alpha
  real<lower=0> kappa;
  real<lower=0> y[M];
  real<lower=0> phi;
  real<lower=0> tau;
  real <lower=0> ifr_noise[M];
}

transformed parameters {
    vector[P] alpha;
    matrix[N2, M] prediction = rep_matrix(0,N2,M);
    matrix[N2, M] E_deaths  = rep_matrix(0,N2,M);
    matrix[N2, M] Rt = rep_matrix(0,N2,M);
    matrix[N2, M] Rt_adj = Rt;

    {
      matrix[N2,M] cumm_sum = rep_matrix(0,N2,M);
      for(i in 1:P){
        alpha[i] = alpha_hier[i] - ( log(1.05) / 6.0 );
      }
      for (m in 1:M){
        for (i in 2:N0){
          cumm_sum[i,m] = cumm_sum[i-1,m] + y[m];
        }
        prediction[1:N0,m] = rep_vector(y[m],N0); // learn the number of cases in the first N0 days

        Rt[,m] = mu[m] * exp(-X[m] * alpha);
          Rt_adj[1:N0,m] = Rt[1:N0,m];
        for (i in (N0+1):N2) {
          real convolution=0;
          for(j in 1:(i-1)) {
            convolution += prediction[j, m] * SI[i-j];
          }
          cumm_sum[i,m] = cumm_sum[i-1,m] + prediction[i-1,m];
          Rt_adj[i,m] = ((pop[m]-cumm_sum[i,m]) / pop[m]) * Rt[i,m];
          prediction[i, m] = Rt_adj[i,m] * convolution;
        }

        E_deaths[1, m]= 1e-15 * prediction[1,m];
        for (i in 2:N2){
          for(j in 1:(i-1)){
            E_deaths[i,m] += prediction[j,m] * f[i-j,m] * ifr_noise[m];
          }
        }
      }
    }
}
model {
  tau ~ exponential(0.03);
  for (m in 1:M){
      y[m] ~ exponential(1/tau);
  }
  phi ~ normal(0,5);
  kappa ~ normal(0,0.5);
  mu ~ normal(3.28, kappa); // citation: https://academic.oup.com/jtm/article/27/2/taaa021/5735319
  alpha_hier ~ gamma(.1667,1);
  ifr_noise ~ normal(1,0.1);
  for(m in 1:M){
    deaths[EpidemicStart[m]:N[m], m] ~ neg_binomial_2(E_deaths[EpidemicStart[m]:N[m], m], phi);
  }
}

generated quantities {
    matrix[N2, M] prediction0 = rep_matrix(0,N2,M);
    matrix[N2, M] E_deaths0  = rep_matrix(0,N2,M);

    {
      matrix[N2,M] cumm_sum0 = rep_matrix(0,N2,M);
      for (m in 1:M){
         for (i in 2:N0){
          cumm_sum0[i,m] = cumm_sum0[i-1,m] + y[m];
        }
        prediction0[1:N0,m] = rep_vector(y[m],N0);
        for (i in (N0+1):N2) {
          real convolution0 = 0;
          for(j in 1:(i-1)) {
            convolution0 += prediction0[j, m] * SI[i-j];
          }
          cumm_sum0[i,m] = cumm_sum0[i-1,m] + prediction0[i-1,m];
          prediction0[i, m] =  ((pop[m]-cumm_sum0[i,m]) / pop[m]) * mu[m] * convolution0;
        }

        E_deaths0[1, m] = uniform_rng(1e-16, 1e-15);
        for (i in 2:N2){
          for(j in 1:(i-1)){
            E_deaths0[i,m] += prediction0[j,m] * f[i-j,m] * ifr_noise[m];
          }
        }
      }
    }
}
```

$$\alpha_k \sim \text{Gamma}(1/6,1) - \frac{\log(1.05)}{6},$$

$$R_{t,m} = R_{0,m} \exp\left(-\sum_{k=1}^{6} \alpha_k I_{k,t,m} - \beta_m I^*_{t,m}\right),$$

$$c_{t,m} = \left(1 - \frac{\sum_{i=1}^{t-1} c_{i,m}}{N_m}\right) R_{t,m} \sum_{\tau=0}^{t-1} c_{\tau,m}\, g_{t-\tau},$$

$$d_{t,m} = \text{ifr}^*_m \sum_{\tau=0}^{t-1} c_{\tau,m} \pi^*_{t-\tau,m},$$

$$c_{1,m}, \dots, c_{6,m} \sim \text{Exponential}(1/\tau), \text{ where } \tau \sim \text{Exponential}(0.03).$$

$$R_{0,m} \sim N^+(3.28, |\kappa|) \text{ with } \kappa \sim N(0,0.5),$$

$$D_{t,m} \sim \text{Negative Binomial}\left(d_{t,m}, d_{t,m} + \frac{d_{t,m}^2}{\psi}\right),$$

Counterfactual:
liczymy tylko na podstawie R0,m
(nie obliczamy Rt,m)

$$c_{t,m} = \left(1 - \frac{\sum_{i=1}^{t-1} c_{i,m}}{N_m}\right) R_{t,m} \sum_{\tau=0}^{t-1} c_{\tau,m}\, g_{t-\tau},$$

$$d_{t,m} = \text{ifr}^*_m \sum_{\tau=0}^{t-1} c_{\tau,m} \pi^*_{t-\tau,m},$$

base_general.r

W tym pliku podajemy parametry do modelu w base_general.stan, np. liczbę dni, które chcemy przewidzieć, czas od zakażenia do śmierci π

$$\pi \sim \mathrm{Gamma}(5.1, 0.86) + \mathrm{Gamma}(17.8, 0.45)$$

```
regions.csv  popt_ifr.csv  serial_interval.csv  base_general.r  preprocessing-subnation-brazil.r  process-covariates.r  compute-ifr-europ
Source on Save                                                                    Run    Source
covariates$schools_universities[covariates$schools_universities > covariates$lockdown] <- covariates$lockdown[covar
covariates$public_events[covariates$public_events > covariates$lockdown] <- covariates$lockdown[covariates$public_e
covariates$social_distancing_encouraged[covariates$social_distancing_encouraged > covariates$lockdown] <- covariate
covariates$self_isolating_if_ill[covariates$self_isolating_if_ill > covariates$lockdown] <- covariates$lockdown[cov

forecast = 0

N2 = 90 # increase if you need more forecast

dates = list()
reported_cases = list()
stan_data = list(M=length(countries),N=NULL,covariate1=NULL,covariate2=NULL,covariate3=NULL,covariate4=NULL,covaria
                 N0=6,cases=NULL,SI=serial.interval$fit[1:N2],
                 EpidemicStart = NULL, pop = NULL) # N0 = 6 to make it consistent with Rayleigh
deaths_by_country = list()

# various distributions required for modeling
mean1 = 5.1; cv1 = 0.86; # infection to onset
mean2 = 18.8; cv2 = 0.45 # onset to death
x1 = rgammaAlt(1e7,mean1,cv1) # infection-to-onset distribution
x2 = rgammaAlt(1e7,mean2,cv2) # onset-to-death distribution

ecdf.saved = ecdf(x1+x2)

for(Country in countries) {
  IFR=ifr.by.country$ifr[ifr.by.country$country == Country]

  covariates1 <- covariates[covariates$Country == Country, c(2,3,4,5,6)]

  d1_pop = ifr.by.country[ifr.by.country$country==Country,]
  d1=d[d$Countries.and.territories==Country,c(1,5,6,7)]
  d1$date = as.Date(d1$DateRep,format='%d/%m/%Y')
  d1$t = decimal_date(d1$date)
  d1=d1[order(d1$t),]

110:46   (Top Level)                                                                    R Script
```

IFR

Jest odwrotnie, niż myśleliśmy - dane o age-specific IFR są pobierane, a potem jest z nich liczona średnia IFR dla danego kraju. Średnie IFR są wpisywane do pliku **popt_ifr.csv**

```
"","country","popt","ifr"
"1","Denmark",5792203,0.0102074697882139
"2","Norway",5421242,0.00914956406203488
"3","Sweden",10099270,0.0103110432111081
"4","United Kingdom",67886004,0.0103504384028517
"5","Italy",60461828,0.0124496263593357
"6","Spain",46754783,0.0107838730158524
"7","Austria",9006400,0.0103882259256609
"8","Belgium",11589616,0.0109598776668092
"9","France",65273512,0.012556187488595
"10","Germany",83783945,0.0123324426034139
"11","Switzerland",8654618,0.0102134528252841
"12","Greece",10423056,0.0117992290838236
"13","Portugal",10196707,0.0117258680881815
"14","Netherlands",17134873,0.010289760240957
```

base_general.stan × | base_general.r × | compute-ifr-europe.r ×

Source on Save | Run | Source

ifrs | Next | Prev | All | Replace | Replace | All

In selection | Match case | Whole word | Regex | Wrap

```r
### This code was written by Patrick Walker and Charlie for the LMIC global report
### Edited by Ettie for European setting

# Load Required Packages and Source Functions
library(tidyverse)
library(readxl)
library(socialmixr)
library(dfoptim)

source("utils/get-ar.r")

# Loading in Population Data and Severity Parameters
demog_WPP <- readRDS('data/country-inputs.rds')
pop_columns <- grep("pop", names(demog_WPP))
severity_inputs <- readRDS('data/severity-inputs.rds')
IFRS <- severity_inputs$IFR_adj
```

pobieramy IFR adjusted
(czyli z podziałem na wiek)

```r
# Set Parameters for Optimisation
R <- 3.28
guess_hom <- 0.85 # guess homogeneous attack rate
guess_modifier <- 0.7 # guess a lower bound for extent age-specific matrix alters average attack rate
reiterates <- 3
iterates <- 10000
restarts <- 10
tol <- 1e-07
init <- 1

country_data <- read.csv("data/popt_ifr.csv", stringsAsFactors = FALSE)
countries <- unique(country_data$country)

IFR <- vector(length = length(countries))
total_pop <- vector(length = length(countries))
```

lista pojedynczych
IFR dla krajów

```r
for (i in 1:length(countries)){
  # Pick the state You want and Extract Population
  country <- countries[i]
  idx <- which(demog_WPP$Country_or_region == country)
  raw_country_pop <- unlist(demog_WPP[idx, pop_columns])
  country_pop <- c(raw_country_pop[1:15], sum(raw_country_pop[16:21])) * 1000

  # Pick the Country You want and Extract Relevant Contact Matrix
  #   Note: Only limited countries have contact matrices available.
  # Contact matrices are taken from PolyMod apart for France
  # Mossong J, Hens N, Jit M, Beutels P, Auranen K, Mikolajczyk R, Massari M, Salmaso S, Tomba GS, Wallinga J, Heijne J, Sadkowska-Todys M, Rosinska M, Edmunds WJ (2017). "POLYMOD
  contact_mat_list<-readRDS("data/contact-matrices.rds")
  contact_mat <- data.matrix((contact_mat_list[[demog_WPP$Matrix[idx]]]))

  # Processing the Contact Matrix to Generate a Probability Matrix
  # (i.e. Likelihood a person in each age group mixes with people in a different age group)
  MIJ <- t(sapply(seq(country_pop),function(x){
    contact_mat[x,]*country_pop[x]
```

base_general.stan × | base_general.r × | compute-ifr-europe.r ×

Source on Save | Run | Source

ifrs | Next | Prev | All | Replace | Replace | All

In selection | Match case | Whole word | Regex | Wrap

```r
  # Contact matrices are taken from PolyMod apart for France
  # Mossong J, Hens N, Jit M, Beutels P, Auranen K, Mikolajczyk R, Massari M, Salmaso S, Tomba GS, Wallinga J, Heijne J, Sadkowska-Todys M, Rosinska M, Edmunds WJ (2017). "POLYMOD
  contact_mat_list<-readRDS("data/contact-matrices.rds")
  contact_mat <- data.matrix((contact_mat_list[[demog_WPP$Matrix[idx]]]))

  # Processing the Contact Matrix to Generate a Probability Matrix
  # (i.e. Likelihood a person in each age group mixes with people in a different age group)
  MIJ <- t(sapply(seq(country_pop),function(x){
    contact_mat[x,]*country_pop[x]
```


```r
  }))
  adjust_mat<-(MIJ+t(MIJ))/2
  new_mix_mat<-t(sapply(seq(country_pop),function(x){
    adjust_mat[x,]/country_pop[x]
  }))
  c_mat<-t(sapply(seq(country_pop),function(x){
    new_mix_mat[x,]/sum(new_mix_mat[x,])
  }))

  ai <- rowSums(new_mix_mat)
  ng_eigen <- Re(eigen(new_mix_mat)$val[1])
  rmod <- R/ng_eigen*ai
  tot_pop <- sum(country_pop)
  total_pop[i] <- tot_pop

  # Running an Optimiser to Get the Number Infected by Age for Each Age Group
  x <- get_AR(R = R, rmod = rmod, c_mat = c_mat, demog = country_pop, init = init, guess_hom = guess_hom,
              guess_modifiers = guess_modifier, iterates = iterates, reiterates = reiterates,
              restarts = restarts, tol = tol)

  # Number infected and attack rate for 5 year age bands up to 75+
  number_inf_by_age <- x$par
  attack_rates_by_age <- x$par/country_pop
  #plot(attack_rates_by_age, ylim = c(0, 1))

  # Splitting up 75+ into 75-80 and 80+ to incorporate the age-specific IFRS for these two
  # groups that we have
  infs_75_80 <- number_inf_by_age[16] * raw_country_pop[16]/(sum(raw_country_pop[16:21]))
  infs_80_plus <- number_inf_by_age[16] * sum(raw_country_pop[17:21])/(sum(raw_country_pop[16:21]))
  number_inf_by_age[16] <- infs_75_80
  number_inf_by_age[17] <- infs_80_plus



  # Calculating the number of Deaths in Each Age Group
  deaths <- number_inf_by_age * IFRS
  IFR[i] <- sum(deaths)/sum(number_inf_by_age)
}
```

liczymy srednie IRF
dla kraju i

```r
ifrs <- data.frame("country" = countries, "popt" = total_pop, "ifr" = IFR)
```

wpisujemy do pliku

Szukałam źródła danych o age-specific IFR, w źródle z Supplementary information jest ta tabela:

Age-group (years)	% symptomatic cases requiring hospitalisation	% hospitalised cases requiring critical care	Infection Fatality Ratio
0 to 9	0.1%	5.0%	0.002%
10 to 19	0.3%	5.0%	0.006%
20 to 29	1.2%	5.0%	0.03%
30 to 39	3.2%	5.0%	0.08%
40 to 49	4.9%	6.3%	0.15%
50 to 59	10.2%	12.2%	0.60%
60 to 69	16.6%	27.4%	2.2%
70 to 79	24.3%	43.2%	5.1%
80+	27.3%	70.9%	9.3%

(https://www.imperial.ac.uk/media/imperial-college/medicine/mrc-gida/2020-03-16-COVID19-Report-9.pdf)
Są w niej dane o pacjentach z Chin. W Supplementary information tłumaczą, że dane Attack Rate w innych krajach pobrali z https://www.imperial.ac.uk/media/imperial-college/medicine/sph/ide/gida-fellowships/Imperial-College-COVID19-Global-Impact-26-03-2020v2.pdf - te dane są tam zbierane na podstawie ankiet o narażeniu na kontakt.