BDA - Assignment 9

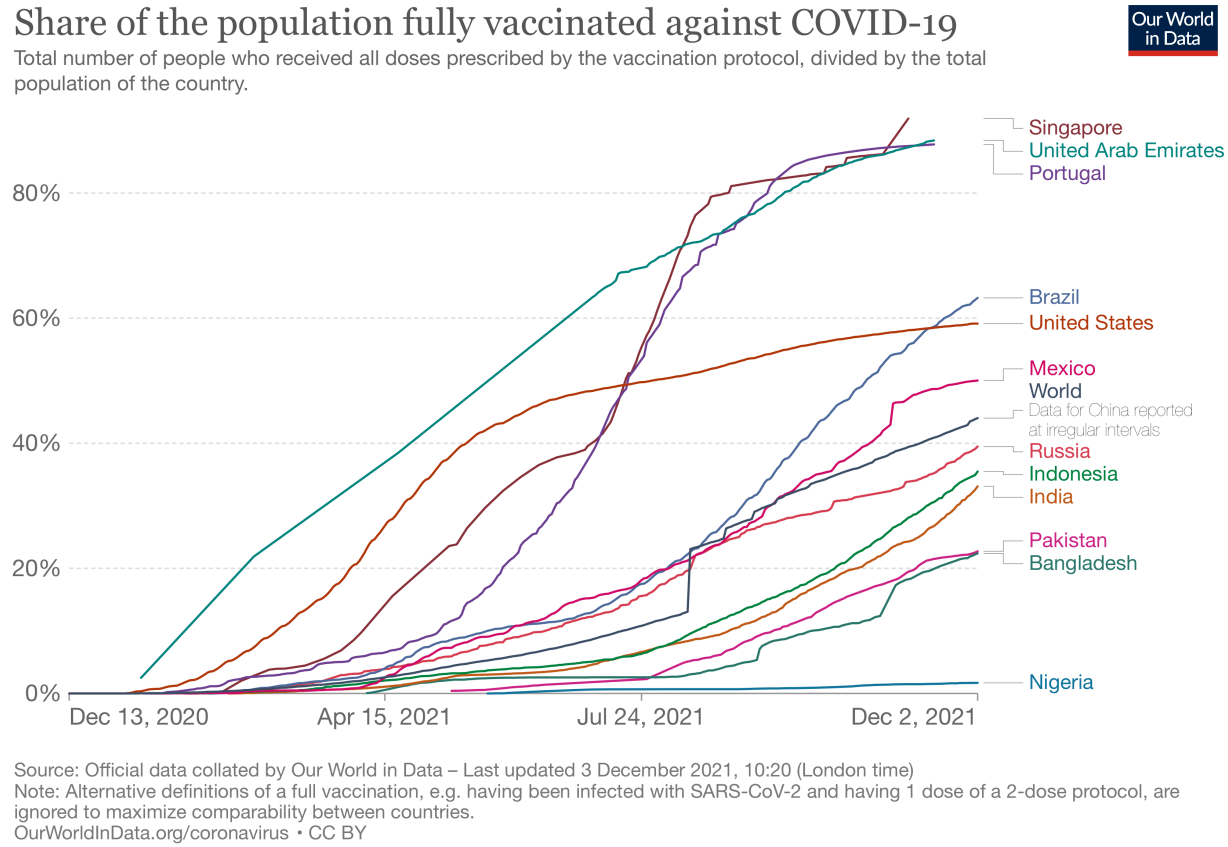
Anonymous

Contents

1. Introduction

It's been almost 2 years since the covid-19 started in December 2019, and it's been 1 year since world vaccination has been started December 2020. Even though most of the countries rushing into vaccination and some countries are picking over 80% rate in vaccination, the pandemic seems unstoppable. One of the popular idea to stop the pandemic is that to achieve a 'herd-immunity threshold, which occurs when a large portion of a community (the herd) becomes immune to a disease, making the spread of disease from person to person unlikely. Immune individuals are unlikely to contribute to disease transmission, disrupting chains of infection, which stops or slows the spread of disease (wiki citation). It is known that the herd-immunity threshold is achievable only with high vaccination rates, and many scientists had thought that once people started being immunized en masse, herd immunity would permit society to return to normal. Most estimates had placed the threshold at 60–70% of the population gaining immunity, either through vaccinations or past exposure to the virus(nature citation). Down below shows the graph of population fully vaccinated by countries.

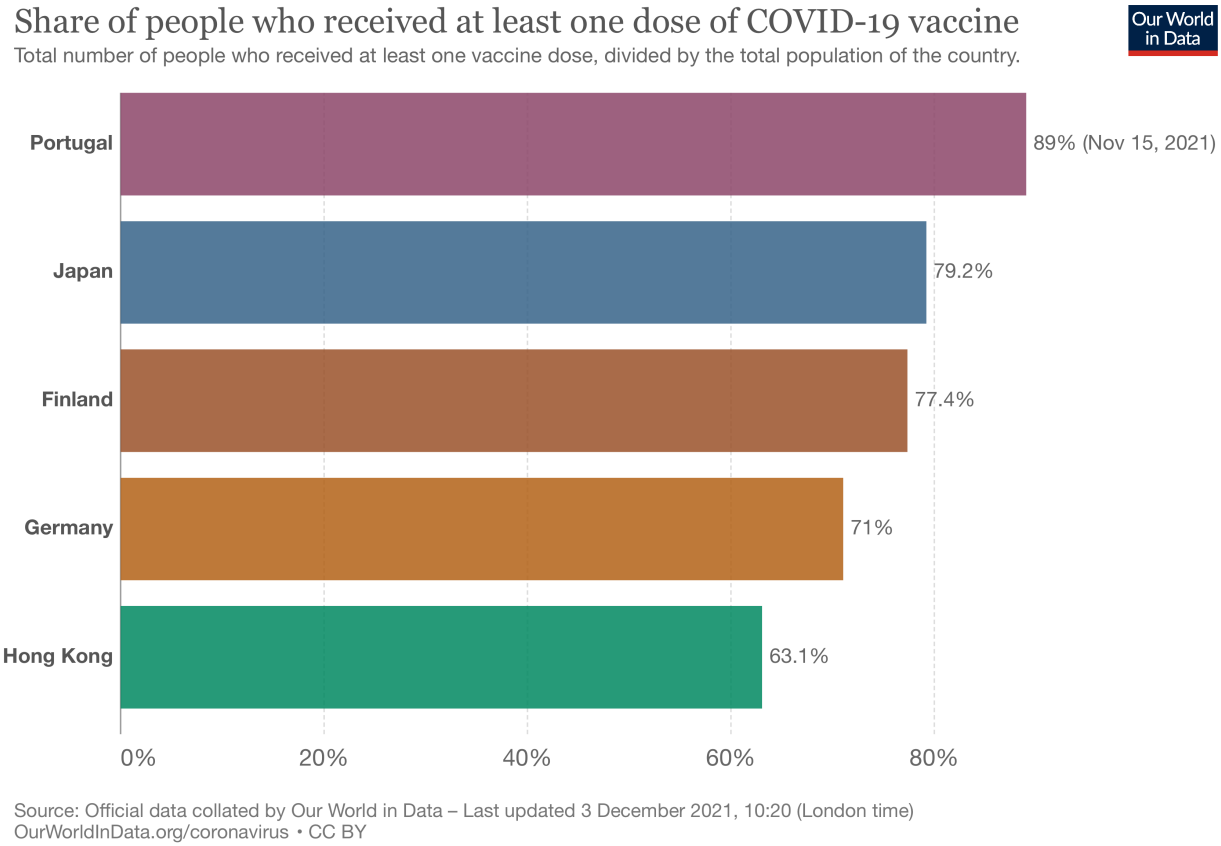
```
knitr::include_graphics("./image/share-people-fully-vaccinated-covid_world.png")
```



As the high vaccination rate is a big part of ending pandemic by achieving herd-immunity, our group was interested in predicting how much high vaccination rate that countries will achieve in future. As we expect the cumulative vaccination rate graph will follow logit function, we set our model as a logit function. We picked 5 countries,which are Finland, Portugal, Japan, Germany,and Hongkong for our data. The cumulative vaccination graphs in these countries roughly follow the logit function and our main modeling idea is to see find which model the data fits well (seperate model or hierarchical model) and to predict the future vaccination rate.

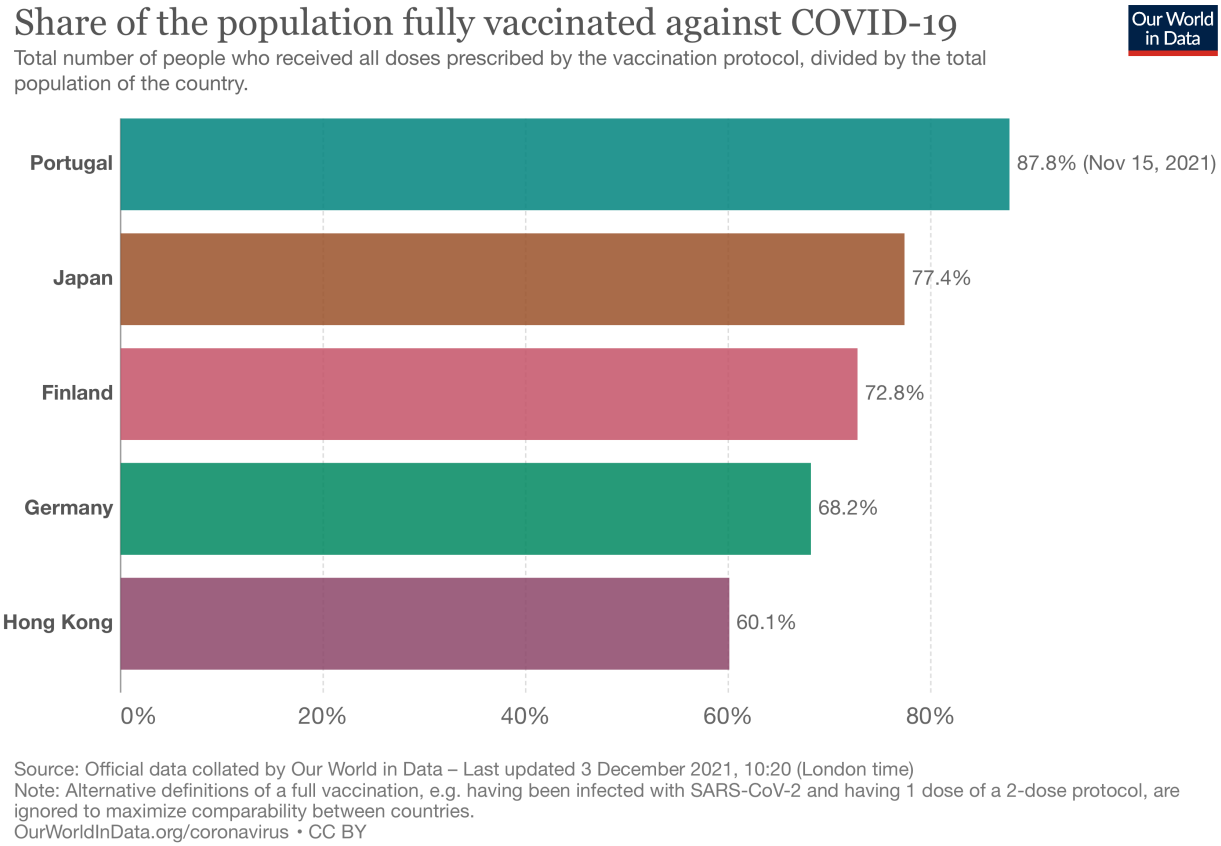
Down below shows the barplot of population rate who received at least one dose of covid19 vaccine by countries in our interest.

```
knitr::include_graphics("./image/share-people-vaccinated-covid.png")
```



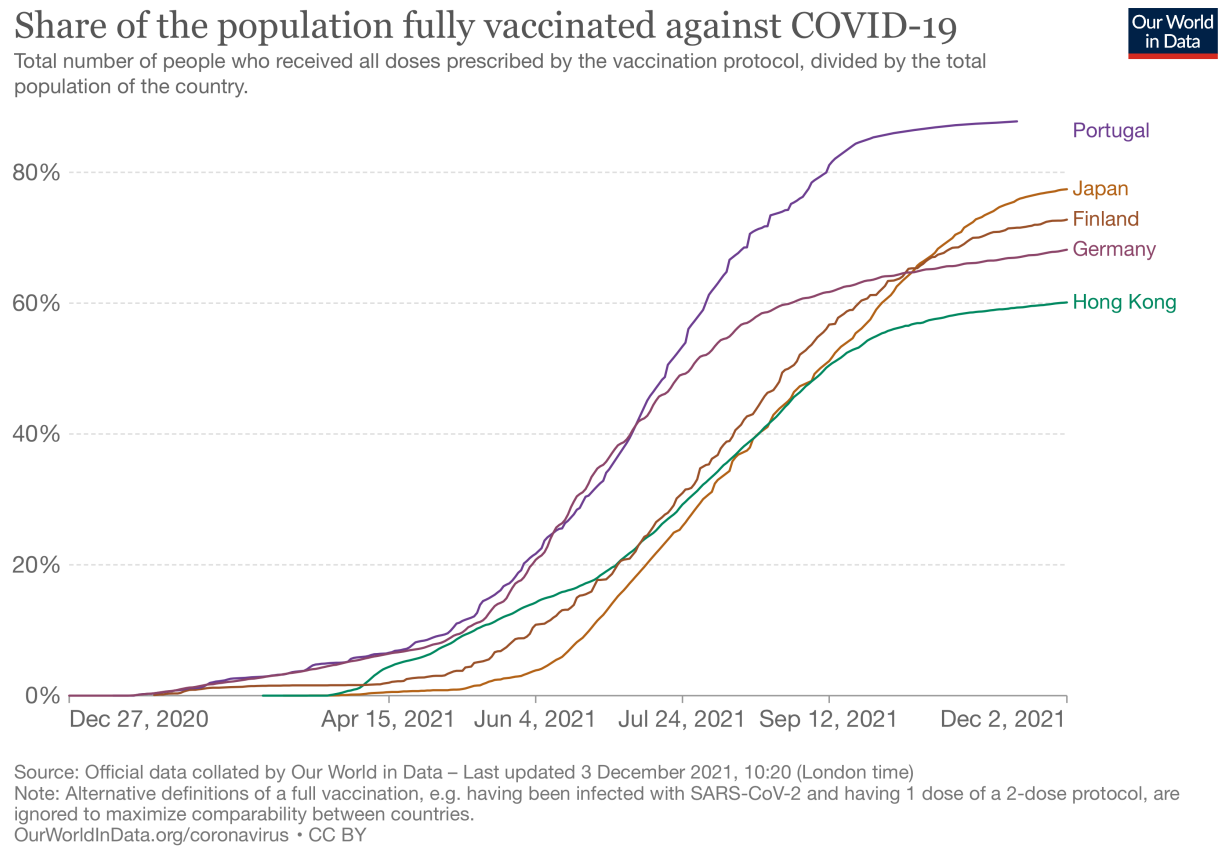
Down below shows the barplot of population rate of fully vaccinated against covid19 by countries in our interest.

```
knitr::include_graphics("./image/share-people-fully-vaccinated-covid_bar.png")
```



Down below shows the barplot of fully vaccinated population rate by countries in our interest.

```
knitr::include_graphics("./image/share-people-fully-vaccinated-covid_plot.png")
```



Relative work

2. Dataset

Datasets down below show cumulative covid vaccinations rate by countries(collected from linked phrase).Country column shows the country of this data. X column is originally from the date when the country started vaccination to the recent date of vaccination. We normalized the date column by giving index 0 to number of date and deviding it with the number of date. As the time length of the vaccination differs by country, we uniformly picked the 222 datapoints from the datasets before normalize the X column. Y column shows the covid vaccination rate in the country. With the cumulative covid vaccination number, we devide it by 2 times population because most of the vaccines require 2 doses to be fully vaccinated. Therefore, dimension of all datasets are (222,3).

```
#setwd('/Users/chuhyeongyeong/2021_period1/bayesian_data_analysis/bda_aalto_project')
finland <- read.csv("data/Finland_output.csv")
germany <- read.csv("data/Germany_output.csv")
hongkong <- read.csv('data/Hong Kong_output.csv')
portugal <- read.csv('data/Portugal_output.csv')
japan <- read.csv('data/Japan_output.csv')
```

```
cat('dimenstion of Finland dataset: ',dim(finland),'\n')
```

```
## dimenstion of Finland dataset:  222 3
```

```
cat('dimenstion of Germany dataset: ',dim(germany),'\n')
```

```
## dimenstion of Germany dataset:  222 3
```

```
cat('dimenstion of HongKong dataset: ',dim(hongkong),'\n')
```

```
## dimenstion of HongKong dataset:  222 3
```

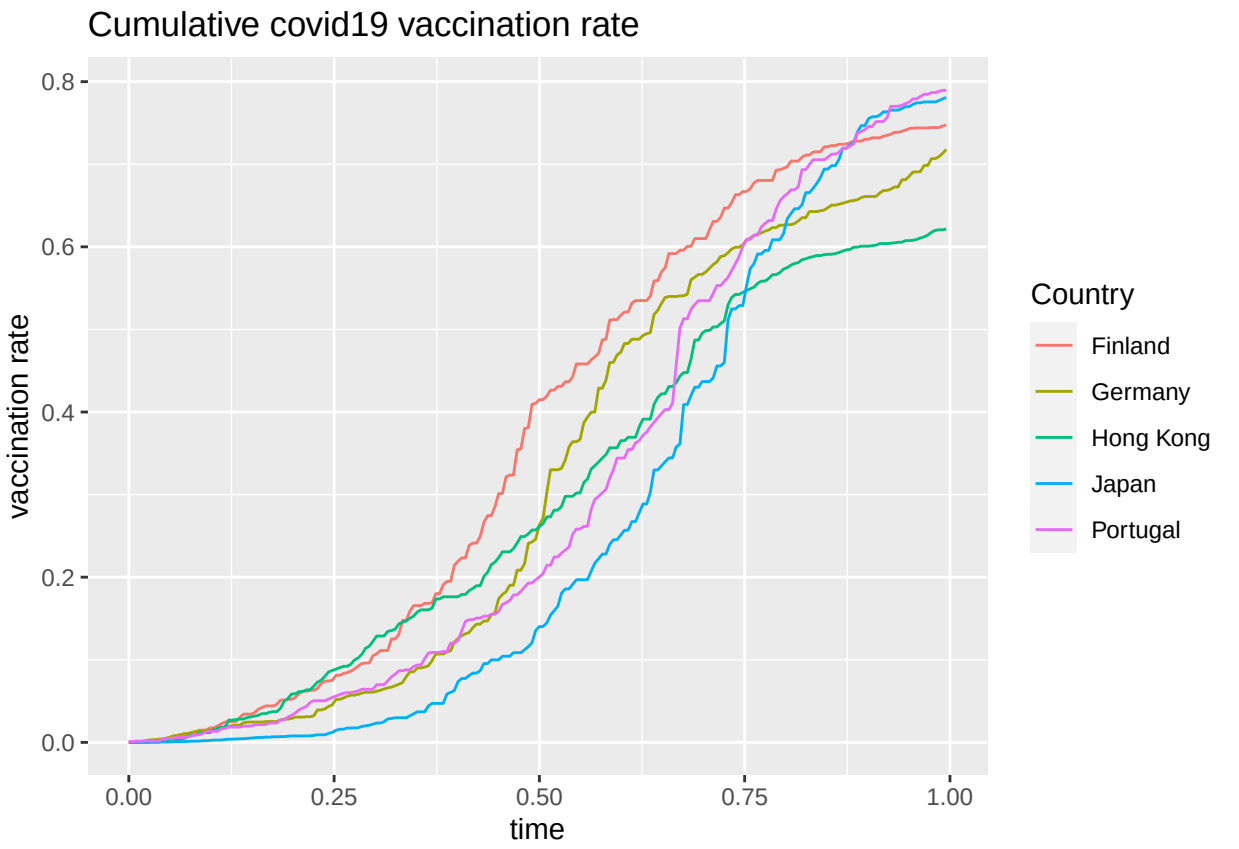
```
cat('dimenstion of Portugal dataset: ',dim(portugal),'\n')
```

```
## dimenstion of Portugal dataset:  222 3
```

```
cat('dimenstion of Japan dataset: ',dim(japan),'\n')
```

```
## dimenstion of Japan dataset:  222 3
```

```
total= rbind(finland,germany,hongkong,portugal,japan)
ggplot(data = total, aes(x = X, y = Y, color = Country)) +
    geom_line() +
    ggtitle("Cumulative covid19 vaccination rate") +
    xlab("time") + ylab("vaccination rate")
```



Mathematical Models and Stan code

Seperated Model

$$
\begin{aligned}
y_{ij} \mid \mu_i, \sigma &\sim \text{Normal}\left(\mu_i, \sigma\right) \\
\mu_i &\sim \text{logit}(\alpha_i, \beta_i) \\
\sigma &\sim \text{inv}\chi^2(1) \\
\\
\alpha_i &\sim \text{inv}\chi^2(1) \\
\beta_i &\sim N(0.5, 1)
\end{aligned}
$$

We will first try to build the seperated model in stan

```
seperated_model <- "
functions {
  real[] logit_transform(real[] x, real k, real x0) {
    int N = size(x);
    real xtemp[N];
    for (i in 1:N){
      xtemp[i] = 1 / (1 + exp(-k * (x[i] - x0)));
    }
     return xtemp;
  }
}

data {
    int<lower=0> J;
    int<lower=1> M;
    int<lower=0> N; // number of data points
    real x[M,N]; // observation year
    real y[M,N]; // observation number of drowned
    real xpred;  // prediction year
}

parameters {
  real alpha[M];
  real beta[M];
  real<lower = 0> sigma;
}

transformed parameters {
  real mu[M,N];
  for (i in 1:M){
    mu[i,] = logit_transform(x[i,],alpha[i], beta[i]);
  }
}

model {
  for (i in 1:M){
    // as prior we will change it to the same values
    alpha[i] ~ inv_chi_square(1);
    beta[i] ~ normal(0.5,1);
```

```
  }

  sigma ~ inv_chi_square(1);

  //likelihood
  for (i in 1:M){
    y[i,] ~ normal(mu[i,], sigma);
  }
}

generated quantities {
  real ypred[M];
  real t[1];
  real log_lik[M,N];
  t[1] = xpred;
  for (i in 1:M){
    ypred[i] = normal_rng(logit_transform(t,alpha[i], beta[i])[1], sigma);
  }

  //log-likelihood
  for (i in 1:M) {
    for (j in 1:N) {
    log_lik[i,j] = normal_lpdf(y[i,j] | mu[i,j], sigma);
    }
  }

}

"
```

Hirachical Model

$$
\begin{aligned}
y_{ij} \mid \mu_i, \sigma &\sim \text{Normal}\left(\mu_i, \sigma\right) \\
\mu_i &\sim \text{logit}(\alpha_i, \beta_i) \\
\sigma &\sim \text{Inv}-\chi^2(1)
\end{aligned}
$$

$$
\begin{aligned}
\beta_j \mid \mu_\beta, \sigma_\beta &\sim \text{Normal}(\mu_\beta, \sigma_\beta) \\
\alpha_j \mid \sigma_\alpha &\sim \text{Inv}-\chi^2\left(\sigma_\alpha\right)
\end{aligned}
$$

$$
\begin{aligned}
\mu_\beta &\sim \text{Normal}(0, 1) \\
\sigma_\beta &\sim \text{Inv}-\chi^2(1) \\
\sigma_\alpha &\sim \text{Inv}-\chi^2(1)
\end{aligned}
$$

Now we will build the hirachical model in stan:

```
hirachical_model <- "
functions {
  real[] logit_transform(real[] x, real k, real x0) {
    int N = size(x);
    real xtemp[N];
```

```
    for (i in 1:N){
      xtemp[i] = 1 / (1 + exp(-k * (x[i] - x0)));
    }
     return xtemp;
  }
}

data {
    int<lower=1> M; //number of country
    int<lower=0> N; // number of data points
    real x[M,N]; //
    real y[M,N]; //
    real xpred;  // prediction year
}

parameters {
  real alpha[M];
  real beta[M];
  real<lower = 0> sigma;
  real<lower = 0> hyper_sigma;
  real hyper_mu;
  real<lower = 0> hyper_alpha;
}

transformed parameters {
  real mu[M,N];
  for (i in 1:M){
    mu[i,] = logit_transform(x[i,],alpha[i], beta[i]);
  }
}

model {
    hyper_mu ~ normal(0, 1);
    hyper_sigma ~ inv_chi_square(10);
    hyper_alpha ~ inv_chi_square(10);

  for (i in 1:M){
    // as prior we will change it to the same values
    alpha[i] ~ inv_chi_square(hyper_alpha);
    beta[i] ~ normal(hyper_mu,hyper_sigma);
  }

  sigma ~ inv_chi_square(1);

  //likelihood
  for (i in 1:M){
    y[i,] ~ normal(mu[i,], sigma);
  }
}

generated quantities {
  real ypred[M];
  real log_lik[M,N];
```

```
  real t[1];
  t[1] = xpred;
  for (i in 1:M){
    ypred[i] = normal_rng(logit_transform(t,alpha[i], beta[i])[1], sigma);
  }

  //log-likelihood
  for (i in 1:M) {
    for (j in 1:N) {
    log_lik[i,j] = normal_lpdf(y[i,j] | mu[i,j], sigma);
    }
  }

}

"
```

Prior selection

For both models we choose to use weakly informative priors.

Prior choice of the seperated model

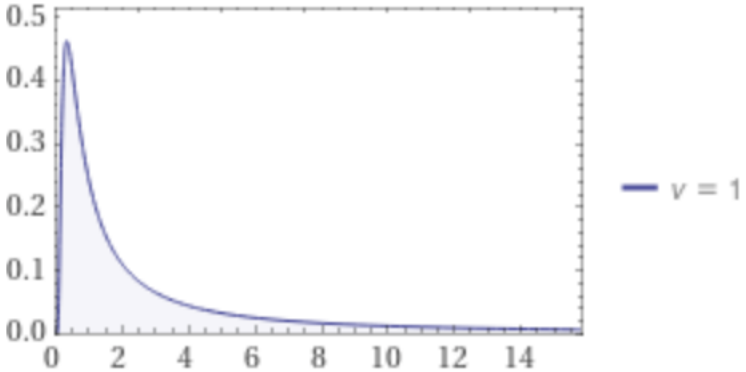
For the seperated model, the α prior needed to fullfill two criteria. First it needed to be a positive number, as we can expect from the vaccination, that the rate is rising and not dropping. Also, as the vaccination should follow roughly a logit function, as it can be observed from the data, we can expect to have a gradient, which is more likely around 1 than bigger than 50. Hence the prior of Inv $-\chi^2(1)$ was choosen.

For our β prior estimation we know, that it should be in the range of $[0, 1]$, as this is the range of our x-Data. Therefore we choose $N(0.5, 1)$.

The variance of our y-sampling should be a positive number. As we can also expect it to be quite small, as we want our model following the line quite tightly, we choose here as well Inv $-\chi^2(1)$.

Prior choices of the hirachical model

The hirachical model has the following priors. The α- prior still has the same distribution function: Inv $-\chi^2(\cdot)$, with the same reasoning as for the seperated model. Here however, the parameter for the function gets sampled as well. With the same reasoning about the order of magnitude of our parameter we choose



Inv $-\chi^2(1)$ as a suitable weakly informative prior.

As we can see here, the PDF of the probability distribution has nearly all amount of its mass in the interval of $[0, 10]$

The β prior is again, like in the seperated model a normal distribution $N(\cdot, \cdot)$. For the first argument, we choose a normal distribution of $N(0, 1)$, with the reasoning, that we expect it to be located somewhere in the interval of $[0, 1]$, as this is the range of the data. The prior of the variance is given by a Inv $-\chi^2(1)$, as here as well we want to have a variance, which is not much larger, than our expected interval.

```
# setwd("/Users/max/Documents/UniMac/Aalto/BDA/bda_aalto_project/data")
finland <- read.csv("data/Finland_output.csv")
germany <- read.csv("data/Germany_output.csv")
portugal <- read.csv("data/Portugal_output.csv")
hongkong <- read.csv("data/Hong Kong_output.csv")
japan <- read.csv("data/Japan_output.csv")
```

```
xData <- rbind(finland$X,germany$X,portugal$X,hongkong$X,japan$X)
#dim(xData) <- c(5,222)

yData <-rbind(finland$Y,germany$Y,portugal$Y,hongkong$Y,japan$Y)
#dim(yData) <- c(5,222)

sm <- rstan::stan_model(model_code = seperated_model)
```

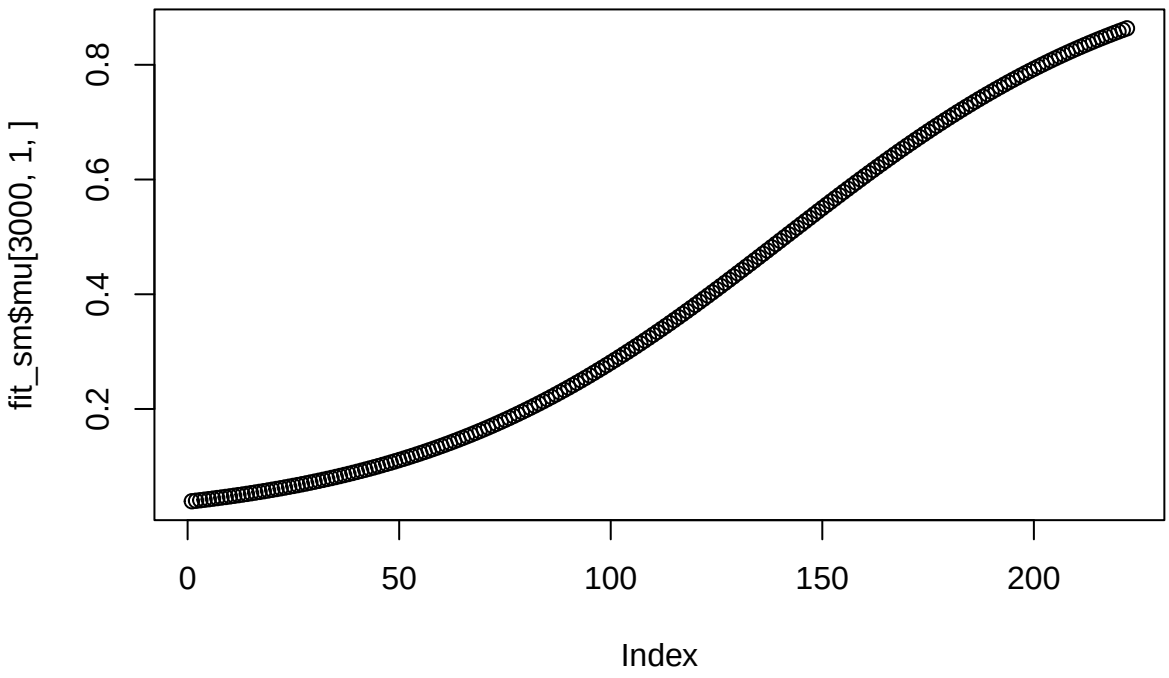
```
## Trying to compile a simple C file
```

```
stan_data <- list(
    J = 1,
    N = length(finland$Y),
    M = 5,
    y = yData,
    x = xData,
    xpred = 1.1
)
model_separated <- rstan::sampling(sm, data = stan_data, warmup=3000, iter=4000)
```

```
## Warning in .local(object, ...): some chains had errors; consider specifying
## chains = 1 to debug
```

```
## here are whatever error messages were returned
```

```
fit_sm <- extract(model_separated, permuted = TRUE, inc_warmup = FALSE)
plot(fit_sm$mu[3000,1,])
```




```
hm <- rstan::stan_model(model_code = hirachical_model)
```

```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

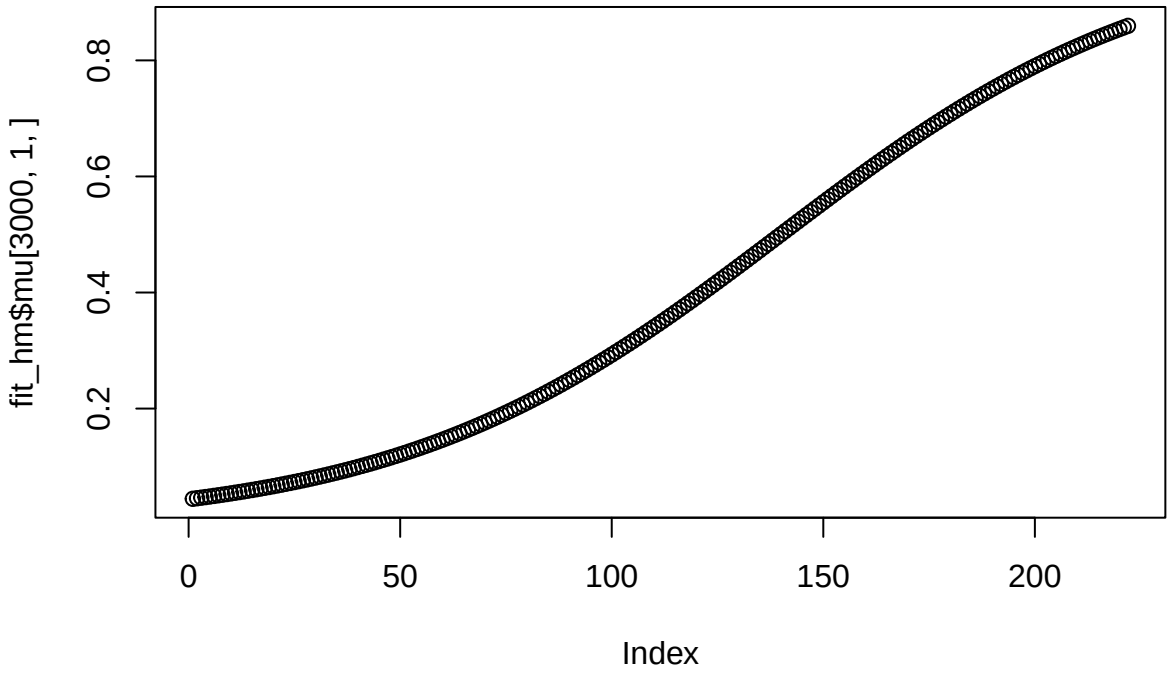
```
stan_data <- list(
    N = length(finland$Y),
    M = 5,
    y = yData,
    x = xData,
    xpred = 1.1
)
model_hirachical <- rstan::sampling(hm, data = stan_data, warmup=3000, iter=4000)
```

```
## Warning in .local(object, ...): some chains had errors; consider specifying
## chains = 1 to debug
```

```
## here are whatever error messages were returned
```

```
## [[1]]
## Stan model 'eaa9981d8948340015578e3e673d559f' does not contain samples.
```

```
fit_hm <- extract(model_hirachical, permuted = TRUE, inc_warmup = FALSE)
plot(fit_hm$mu[3000,1,])
```



6.How to the Stan model was run, that is, what options were used. This is also more clear as combination of textual explanation and the actual code line.

(explain)

7.Convergence diagnostics (Rhat, ESS, divergences) and what was done if the convergence was not good with the first try.

We will use the build in tool from the stan library to analyse the convergence of our chains.

```
sum_seperate <- summary(model_separated)
sum_hirachical <- summary(model_hirachical)
cat("Rhat of seperate model: \n")
```

```
## Rhat of seperate model:
```

```
sum_seperate$summary[1:11,10]
```

```
##  alpha[1]  alpha[2]  alpha[3]  alpha[4]  alpha[5]   beta[1]   beta[2]   beta[3]
## 0.9992340 0.9990659 0.9997385 0.9994141 0.9995752 0.9992858 0.9990901 0.9992662
##   beta[4]   beta[5]     sigma
## 0.9990690 0.9997857 0.9993120
```

```
cat("Rhat of hiracical model: \n")
```

```
## Rhat of hiracical model:
```

```
sum_hirachical$summary[1:11,10]
```

```
##  alpha[1]  alpha[2]  alpha[3]  alpha[4]  alpha[5]   beta[1]   beta[2]   beta[3]
## 0.9991955 0.9997524 0.9991918 0.9991751 0.9994445 0.9995333 0.9991676 0.9991643
##   beta[4]   beta[5]     sigma
## 0.9991183 0.9991346 0.9997293
```

As we can see here, the $\hat{R}$ values are close to 1, which means, that the chains converges. We will now run the ESS analysis, to further verify, that our $\hat{R}$ values are trustworthy. As mentioned in Verteri et al. 2019, a ESS value above 400 will indicate that the $\hat{R}$ value is reliable. We therefore now use the ESS method from Verteri et al. 2019:

```
cat("ESS of seperated model: \n")
```

```
## ESS of seperated model:
```

```
sum_seperate$summary[1:11,9]
```

```
## alpha[1] alpha[2] alpha[3] alpha[4] alpha[5]  beta[1]  beta[2]  beta[3]
## 6300.423 5575.610 4927.595 5030.168 4995.886 6833.536 6385.799 6680.663
##  beta[4]  beta[5]    sigma
## 4869.913 6558.545 6156.583
```

```
cat("ESS of hiracical model: \n")
```

```
## ESS of hiracical model:
```

```
sum_hirachical$summary[1:11,9]
```

```
## alpha[1] alpha[2] alpha[3] alpha[4] alpha[5]  beta[1]  beta[2]  beta[3]
## 5236.351 4941.697 5056.485 4033.396 4870.633 6300.853 5227.848 6367.452
##  beta[4]  beta[5]    sigma
## 4580.590 5457.098 4618.335
```

Here we can nicely see, that all ESS values are above 400, which indicates, that our $\hat{R}$ values are realiable. We can therefore conclude, that our chains converged.

8. Posterior predictive checks and what was done to improve the model.

9.Model comparison (e.g. with LOO-CV)

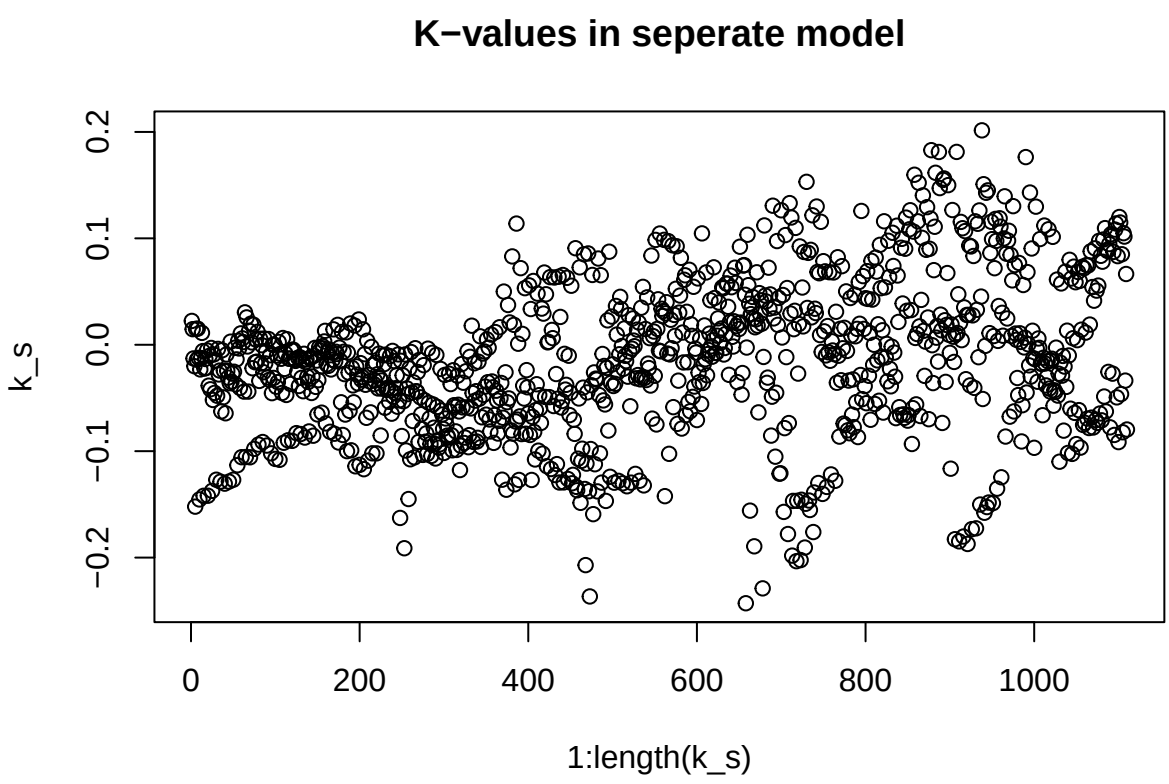
Seperate model

```
l_like_s <- extract_log_lik(model_separated)
loo_s <- loo(l_like_s)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
elpd_s <- loo_s$estimates[1,1]
print(elpd_s)
```

```
k_s <- loo_s$diagnostics$pareto_k
plot(1:length(k_s),k_s,main="K-values in seperate model")
```



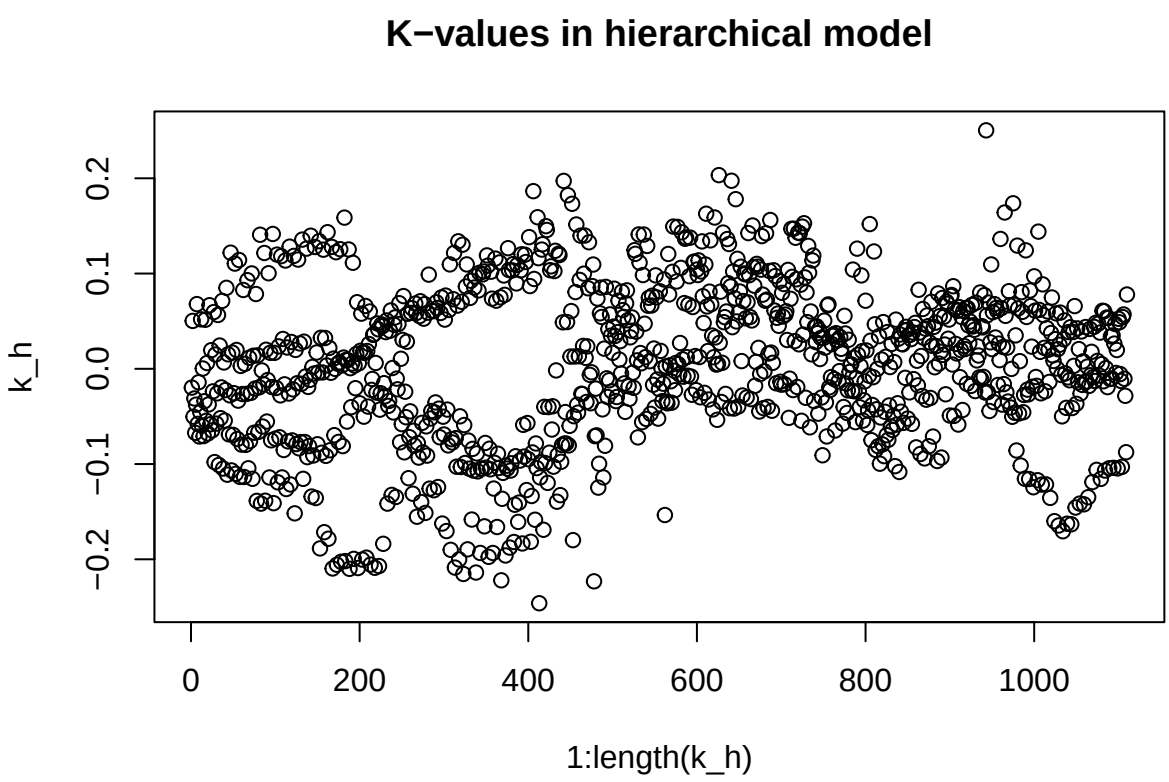
Hierarchical model

```
l_like_h <- extract_log_lik(model_hirachical)
loo_h <- loo(l_like_h)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
elpd_h <- loo_h$estimates[1,1]
print(elpd_h)
```

```
k_h <- loo_h$diagnostics$pareto_k
plot(1:length(k_h),k_h,main="K-values in hierarchical model")
```



11. Sensitivity analysis with respect to prior choices (i.e. checking whether the result changes a lot if prior is changed)