A Gaussian process emulator of complex HIV and non-communicable disease models. A tutorial using Kenyan data and reproducible R code

inMODELA team

2023-11-21

Emulator development

We developed an emulator for HIV, hypertension and depression using the gaussian process

Data

We used data from established simulators for HIV and the aforementioned NCDs which were built using data from Kenya.

Gaussian Process

A Gaussian process fits a model to a dataset, which gives a function that gives a prediction for the mean at any point along with a variance of this prediction. We implemented the gaussian process using the GauPro package.

GauPro package

This R package provides R code for fitting Gaussian process models to data.

A detailed guide of the GauPro package in R by Collin Erickson can be found here:

https://cran.r-project.org/web/packages/GauPro/vignettes/GauPro.html

Article information

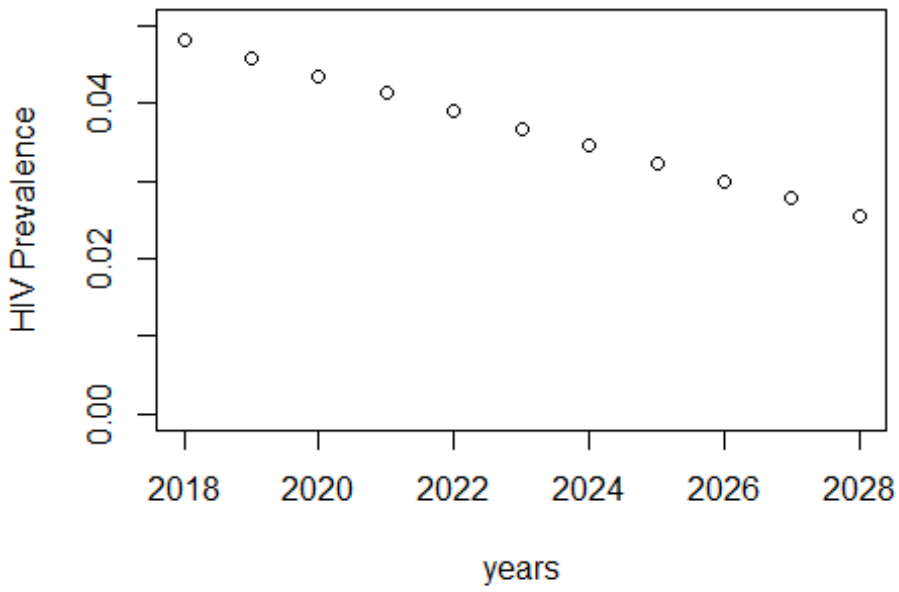
This code forms the basis for the emulator in the article: Sawe S, Mugo M, Wilson-Barthes M, Osetinsky B, Chrysanthopolou S, Yego F, Mwangi A, Galárraga O

"Gaussian process emulation to improve efficiency of computationally intensive multidisease models: A practical tutorial with adaptable R code" BMC.med

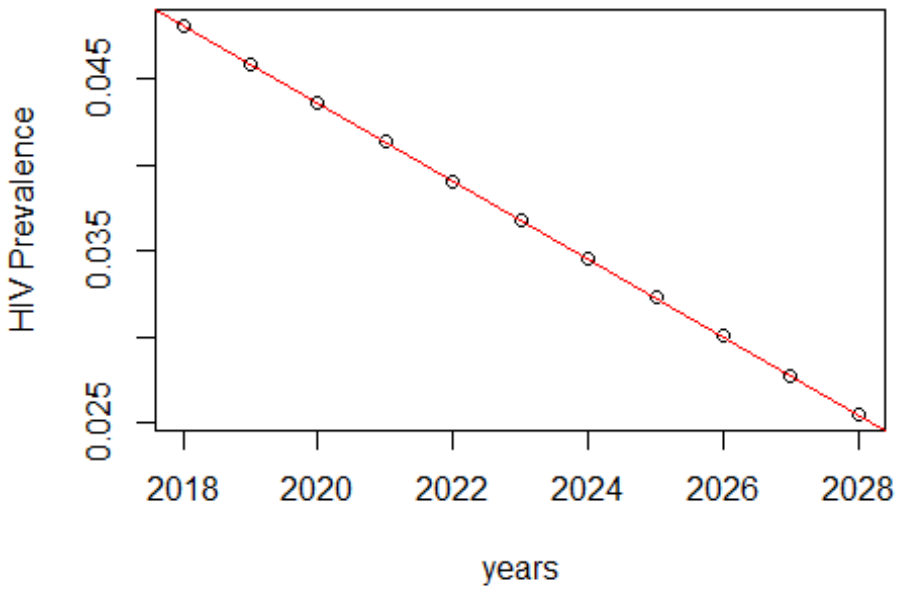
Please cite the article when using this code

1. HIV

```r
# HIV modelling

# Define the emulation period / number of years on the x-axis

# in this tutorial, start year for the inMODELA simulation model was 2018
# in this tutorial, end year for the inMODELA simulation model was 2028
# in this tutorial, number of years forecasted = 11 years

x <- seq(2018,2028,l=11)

# Define the prevalence estimates (based on simulation model(s) and/or the
literature) on the y-axis

# in this tutorial, HIV prevalence estimated from the inMODELA simulation
model in 2018 was 4.81%
# in this tutorial, HIV prevalence estimated from the inMODELA simulation
model in 2023 was 2.55%
# in this tutorial, number of years forecasted = 11 years

y <- seq(0.0481,0.0255, l=11)

# Look at the trend(s) in HIV prevalence over the forecast time period
plot(x, y,
     ylim=c(0,0.05), xlab="years", ylab= "HIV Prevalence")
```

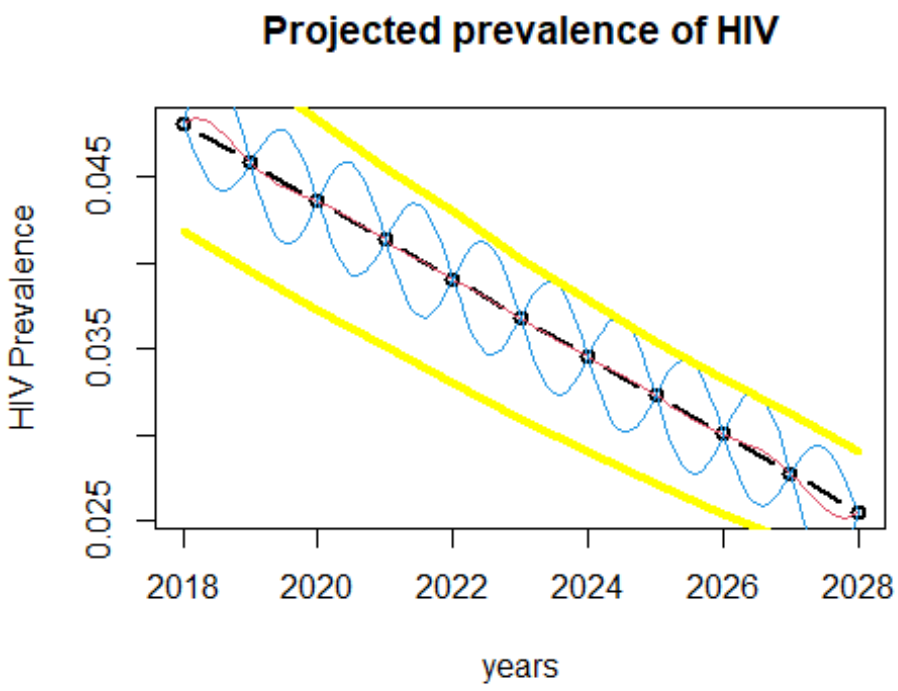


```r
plot(x, y,
     xlab="years", ylab= "HIV Prevalence")
# Look at how A linear model (LM) would fit the simulation data.
lm_mod <- lm(y ~ x)
plot(x, y, xlab="years", ylab= "HIV Prevalence")
abline(a=lm_mod$coef[1], b=lm_mod$coef[2], col='red')    #The LM is
insufficient for describing the underlying function producing the data.
```




```r
# Instead of a linear model, use a Guassian process (GP) model that assumes
that the distribution of points follows a multivariate distribution.
# This emulator uses the GauPro R Package: https://cran.r-
project.org/web/packages/GauPro/index.html
# A guide to the Gaupro package can be found here: https://cran.r-
project.org/web/packages/GauPro/vignettes/GauPro.html

# inMODELA simulator HIV Prevalence confidence intervals
l =
c(0.0419,0.0395,0.0373,0.0351,0.0330,0.0309,0.0290,0.0271,0.0254,0.0238,0.022
0)  #For each annual prevalence estimate, define the lower bound of the 95%
Credibility Interval from the simulation estimates
u =
c(0.0543,0.0512,0.0483,0.0455,0.0430,0.0402,0.0378,0.0354,0.0333,0.0312,0.029
0)  #For each annual prevalence estimate, define the upper bound of the 95%
Credibility Interval from the simulation estimates

# For example, the simulated prevalence of HIV in 2018 was mean=04.81, 95%
CI= 04.19, 05.43.
```

```r
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

gp

## GauPro object of GauPro_Gauss
##  D = 1, N = 11
##  Theta =  1
##  Nugget = 1e-06
##  Run update to add data and/or optimize again
##  Use pred to get predictions at new points

# Look at how the GP model fits the simulation data.
plot(x,y, xlab="years", ylab= "HIV Prevalence", main= "Projected prevalence
of HIV",type="b", lwd=2)
curve(gp$predict(x), add=T, col=2)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4) #Blue lines
visually depict the CIs for emulator's estimates.
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)                                    #Yellow lines
visually depict the CIs for simulator's estimates.
lines(x,u,col="yellow",lwd=4)
```



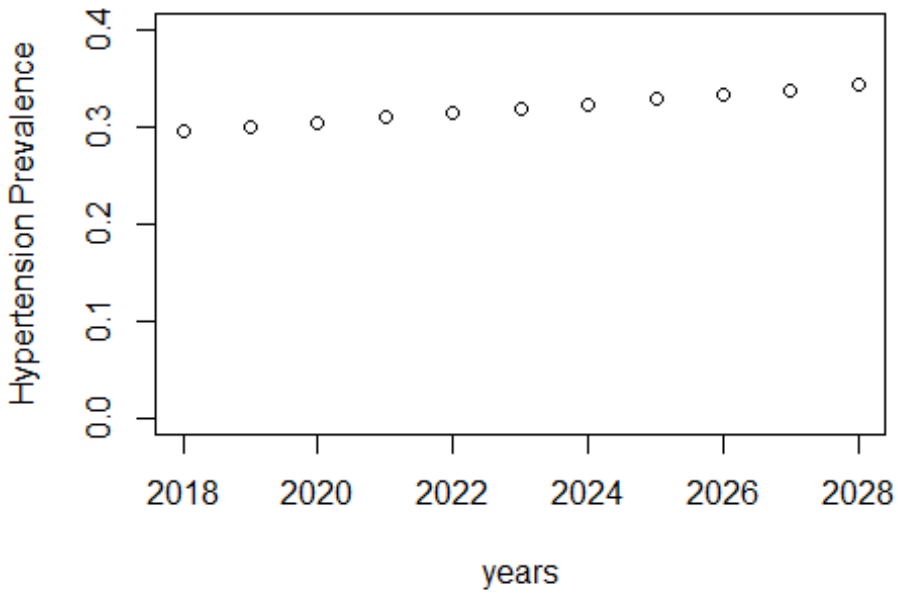

```r
# Abstract the confidence intervals for each annual prevalence estimate
forecasted by the emulator.
set.seed(1234)                                                  # Create a
```

```
dataframe using the emulator's annual prevalence predictions.
new_x <- seq(min(x), max(x), length.out = 11)                    # 11
observations = ll years in the forecast.
predict_df11       <- predict(gp, new_x, se.fit = TRUE)
predict_df11$x     <- new_x
predict_df11$y     <- predict_df11$mean
predict_df11$lower <- predict_df11$y - 1.96 * predict_df11$se
predict_df11$upper <- predict_df11$y + 1.96 * predict_df11$se
View(predict_df11)

# Abstract the confidence intervals for the mean annual prevalence between
each point estimate forecasted by the emulator.
set.seed(1234)                                                   # Create a
dataframe using the emulator's annual prevalence predictions.
new_x <- seq(min(x), max(x), length.out = 21)                    # 21
observations = ll annual point estimates + 10 mean estimates between each
year.
predict_df21       <- predict(gp, new_x, se.fit = TRUE)
predict_df21$x     <- new_x
predict_df21$y     <- predict_df21$mean
predict_df21$lower <- predict_df21$y - 1.96 * predict_df21$se
predict_df21$upper <- predict_df21$y + 1.96 * predict_df21$se
View(predict_df21)
```

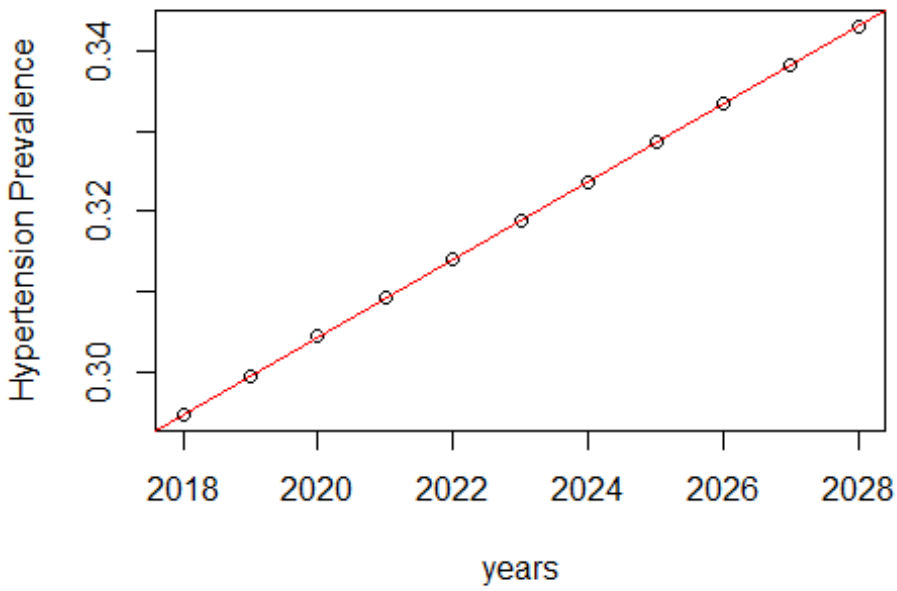
2. Hypertension

```
# Hypertension modelling

################## Enter prevalence estimates and forecast period for
Hypertension (HTN) ######################

# Define the emulation period / number of years on the x-axis
x <- seq(2018,2028,l=11)              # in this tutorial, start year for the
inMODELA simulation model was 2018
                                      # in this tutorial, end year for the
inMODELA simulation model was 2028
                                      # in this tutorial, number of years
forecasted = 11 years

# Define the prevalence estimates (based on simulation model(s) and/or the
literature) on the y-axis
y <- seq(0.2947,0.3430, l=11)         # in this tutorial, HTN prevalence
estimated from the inMODELA simulation model in 2018 was 29.47%
                                      # in this tutorial, HIV prevalence
estimated from the inMODELA simulation model in 2028 was 34.30%
                                      # in this tutorial, number of years
forecasted = 11 years

# Look at the trend(s) in HTN prevalence over the forecast time period
```

```r
plot(x, y,
     ylim=c(0,0.4), xlab="years", ylab= "Hypertension Prevalence")
```

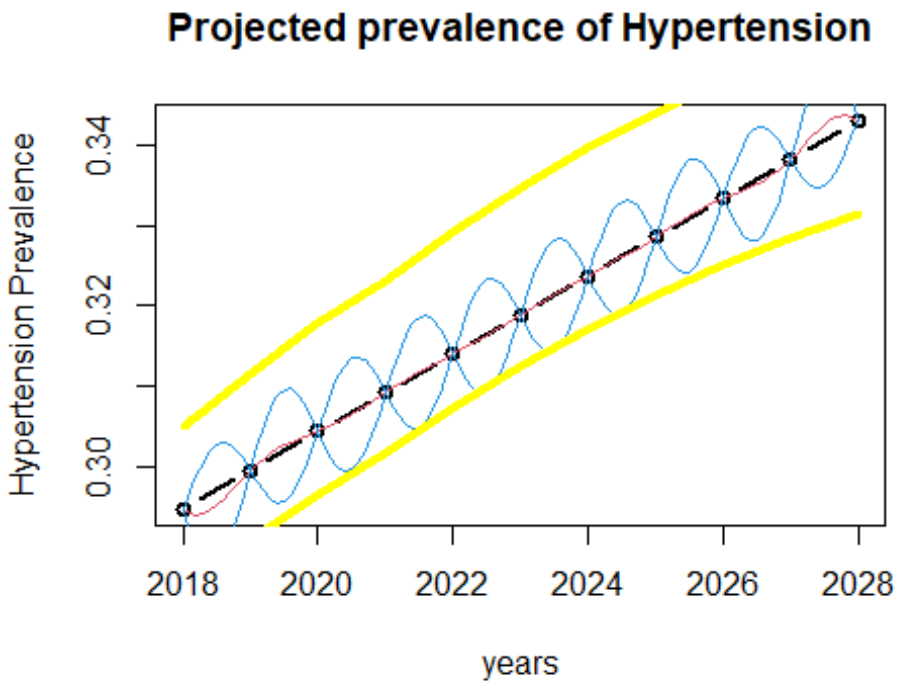



```r
# Look at how A linear model (LM) would fit the simulation data.
lm_mod <- lm(y ~ x)
plot(x, y, xlab="years", ylab= "Hypertension Prevalence")
abline(a=lm_mod$coef[1], b=lm_mod$coef[2], col='red')  #The LM is
insufficient for describing the underlying function producing the data.
```




```
# Instead of a linear model, use a Guassian process (GP) model that assumes
that the distribution of points follows a multivariate distribution.
# This emulator uses the GauPro R Package: https://cran.r-
project.org/web/packages/GauPro/index.html

# inMODELA simulator Hypertension Prevalence confidence intervals
l =
c(0.2845,0.2908,0.2965,0.3017,0.3073,0.3125,0.3171,0.3213,0.3250,0.3284,0.331
3)  #For each annual prevalence estimate, define the lower bound of the 95%
Credibility Interval from the simulation estimates
u =
c(0.3050,0.3116,0.3177,0.3232,0.3292,0.3347,0.3396,0.3441,0.3480,0.3516,0.354
7)  #For each annual prevalence estimate, define the upper bound of the 95%
Credibility Interval from the simulation estimates

library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

gp

## GauPro object of GauPro_Gauss
##  D = 1, N = 11
##  Theta =  1
##  Nugget = 1e-06
##  Run update to add data and/or optimize again
##  Use pred to get predictions at new points
```

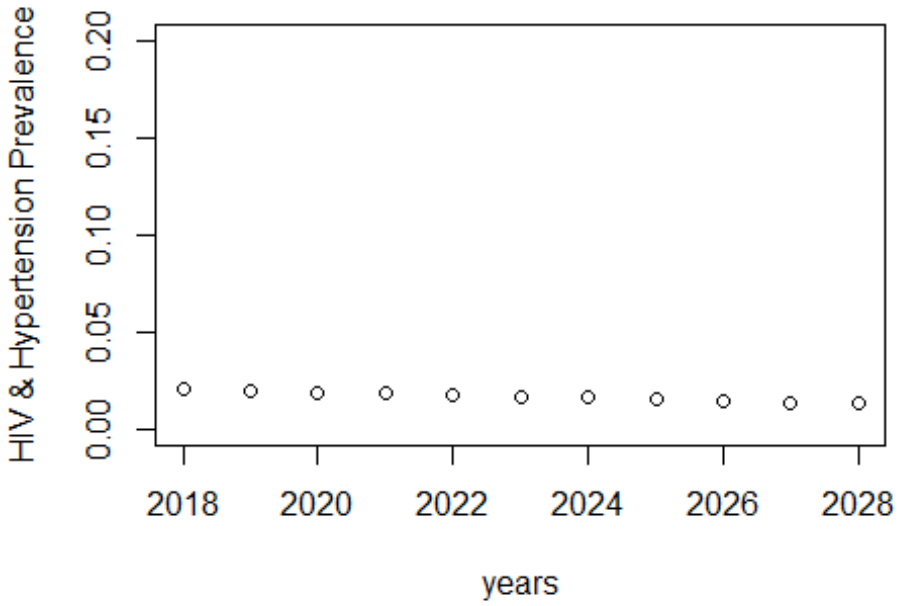
```
# Look at how the GP model fits the simulation data.
plot(x,y, xlab="years", ylab= "Hypertension Prevalence", main= "Projected
prevalence of Hypertension",type="b", lwd=2)
curve(gp$predict(x), add=T, col=2)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
#Blue lines visually depict the CIs for emulator's estimates.
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)
#Yellow lines visually depict the CIs for simulator's estimates.
lines(x,u,col="yellow",lwd=4)
```



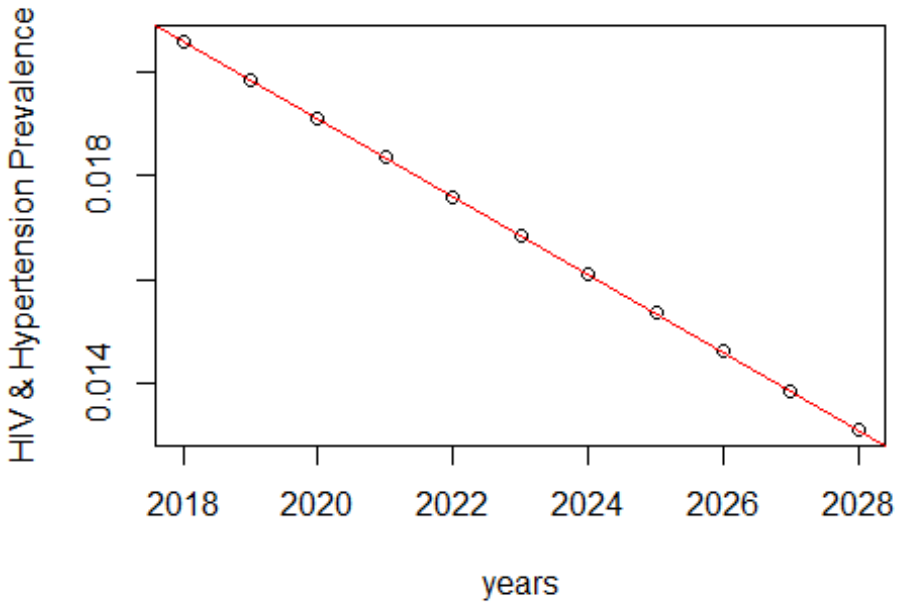
3. Comorbid HIV and Hypertension

```
# Comorbid HIV and Hypertension modelling
################## Enter prevalence estimates and forecast period for comorbid
HIV and Hypertension (HIV+HTN) ######################

# Define the emulation period / number of years on the x-axis
x <- seq(2018,2028,l=11)              # in this tutorial, start year for the
inMODELA simulation model was 2018
                                      # in this tutorial, end year for the
inMODELA simulation model was 2028
                                      # in this tutorial, number of years
forecasted = 11 years

# Define the prevalence estimates (based on simulation model(s) and/or the
literature) on the y-axis
```

```r
y <- seq(0.0206,0.0131, l=11)          # in this tutorial, HIV+HTN prevalence
estimated from the inMODELA simulation model in 2018 was 2.06%
                                       # in this tutorial, HIV prevalence
estimated from the inMODELA simulation model in 2028 was 1.31%
                                       # in this tutorial, number of years
forecasted = 11 years

# Look at the trend(s) in HIV+HTN prevalence over the forecast time period
plot(x, y,
     ylim=c(0,0.2), xlab="years", ylab= "HIV & Hypertension Prevalence")
```

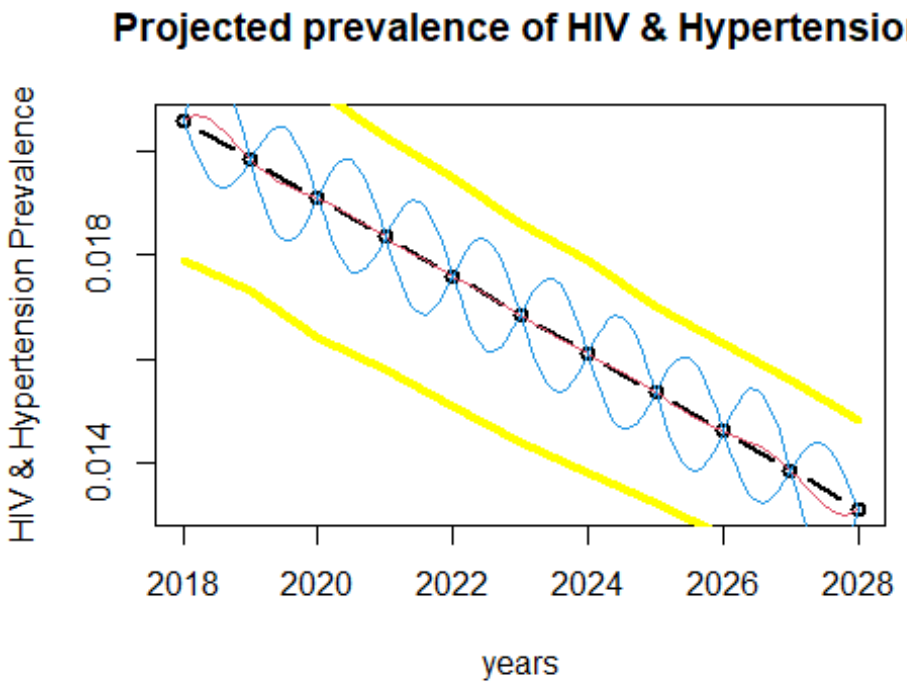



```r
# Look at how A linear model (LM) would fit the simulation data.
lm_mod <- lm(y ~ x)
plot(x, y, xlab="years", ylab= "HIV & Hypertension Prevalence")
abline(a=lm_mod$coef[1], b=lm_mod$coef[2], col='red')  #The LM is
insufficient for describing the underlying function producing the data.
```




```
# Instead of a linear model, use a Guassian process (GP) model that assumes
that the distribution of points follows a multivariate distribution.
# This emulator uses the GauPro R Package: https://cran.r-
project.org/web/packages/GauPro/index.html


# inMODELA simulator HIV+HTN Prevalence confidence intervals
l =
c(0.0179,0.0173,0.0164,0.0158,0.0151,0.0144,0.0138,0.0132,0.0126,0.0120,0.011
4)  #For each annual prevalence estimate, define the lower bound of the 95%
Credibility Interval from the simulation estimates
u =
c(0.0232,0.0224,0.0212,0.0203,0.0195,0.0186,0.0179,0.0170,0.0163,0.0156,0.014
8)  #For each annual prevalence estimate, define the upper bound of the 95%
Credibility Interval from the simulation estimates

library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

gp

## GauPro object of GauPro_Gauss
##  D = 1, N = 11
##  Theta =  1
##  Nugget = 1e-06
##  Run update to add data and/or optimize again
##  Use pred to get predictions at new points
```

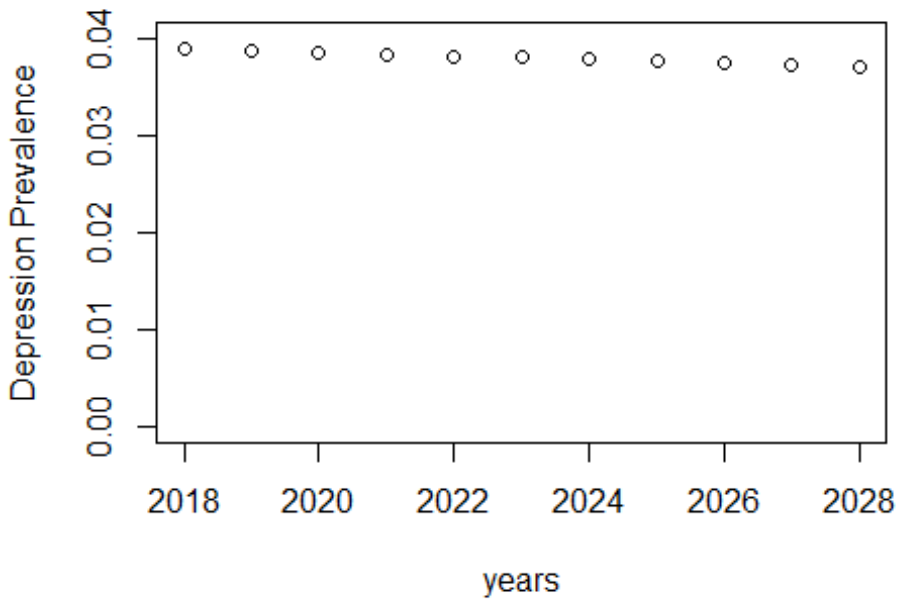
```r
# Look at how the GP model fits the simulation data.
plot(x,y, xlab="years", ylab= "HIV & Hypertension Prevalence", main=
"Projected prevalence of HIV & Hypertension",type="b", lwd=2)#,
curve(gp$predict(x), add=T, col=2)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
#Blue lines visually depict the CIs for emulator's estimates.
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)
#Yellow lines visually depict the CIs for simulator's estimates.
lines(x,u,col="yellow",lwd=4)
```



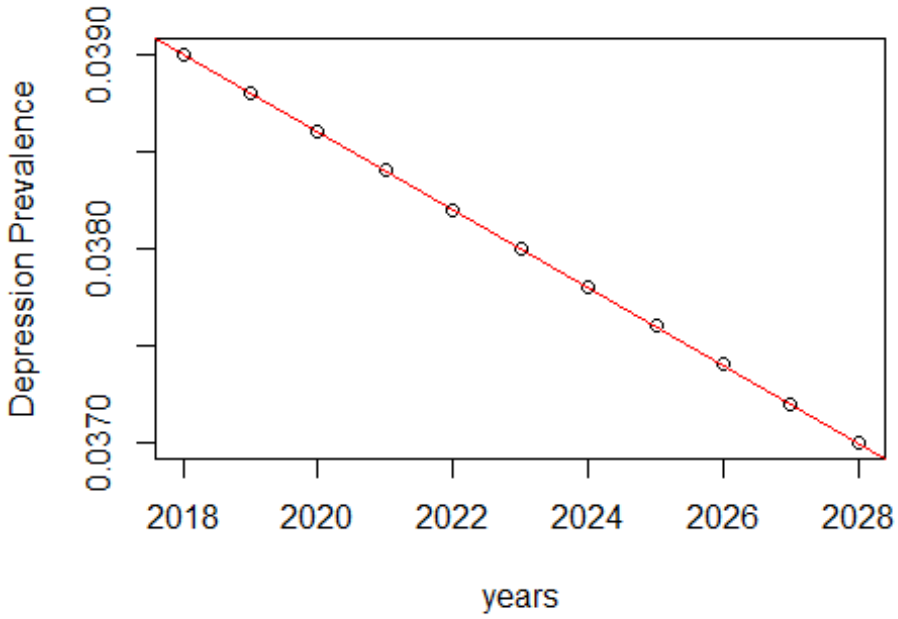
4. Depression

```r
# Depression modelling
################## Enter prevalence estimates and forecast period for
depression among people living with HIV ######################

# Define the emulation period / number of years on the x-axis
x <- seq(2018,2028,l=11)              # in this tutorial, start year for the
Smit et al. simulation model was 2018
                                      # in this tutorial, end year for theSmit
et al. simulation model was 2028
                                      # in this tutorial, number of years
forecasted = 11 years

# Define the prevalence estimates (based on simulation model(s) and/or the
literature) on the y-axis
y <- seq(0.039,0.037, l=11)           # in this tutorial, depression
```

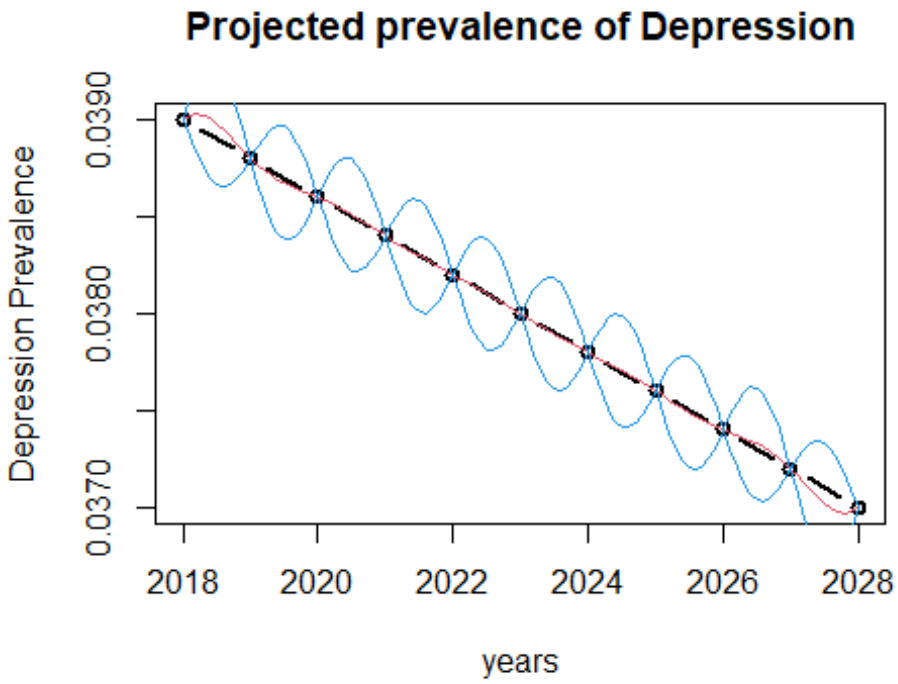
```
prevalence estimated from the simulation model in 2018 was 3.9%
                                    # in this tutorial, depression
prevalence estimated from the simulation model in 2028 was 3.7%. This was
calculated by averaging the prevalence in 2030 and 2025 to maintain
consistency with the inMODELA simulator.
                                    # in this tutorial, number of years
forecasted = 11 years

# Look at the trend(s) in depression prevalence among PLWH over the forecast
time period
plot(x, y,
     ylim=c(0,0.04), xlab="years", ylab= "Depression Prevalence")
```




```
# Look at how A linear model (LM) would fit the simulation data.
lm_mod <- lm(y ~ x)
plot(x, y, xlab="years", ylab= "Depression Prevalence")
abline(a=lm_mod$coef[1], b=lm_mod$coef[2], col='red')   #The LM is
insufficient for describing the underlying function producing the data.
```




```
# Instead of a linear model, use a Guassian process (GP) model that assumes
that the distribution of points follows a multivariate distribution.
# This emulator uses the GauPro R Package: https://cran.r-
project.org/web/packages/GauPro/index.html

library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Look at how the GP model fits the simulation data.
gp

## GauPro object of GauPro_Gauss
##  D = 1, N = 11
##  Theta =  1
##  Nugget = 1e-06
##  Run update to add data and/or optimize again
##  Use pred to get predictions at new points

plot(x,y, xlab="years", ylab= "Depression Prevalence", main= "Projected
prevalence of Depression",type="b", lwd=2)#,
curve(gp$predict(x), add=T, col=2)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
```

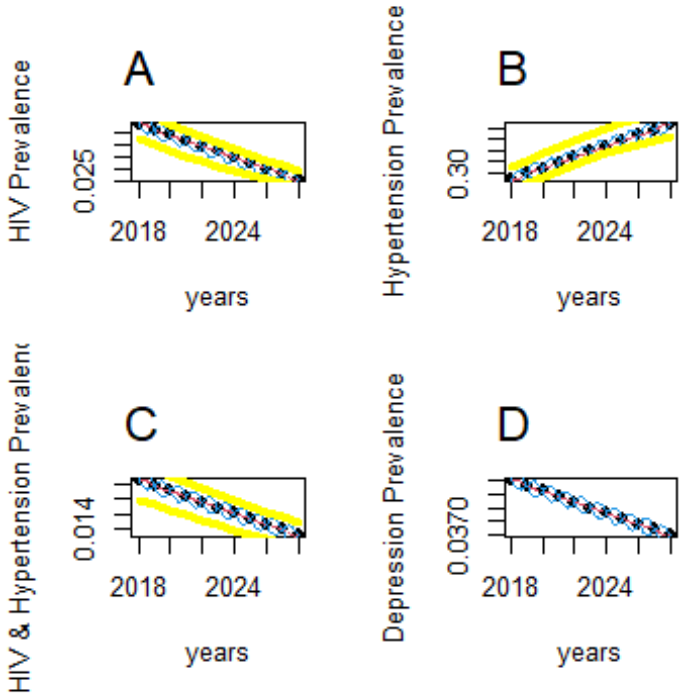


5. Plotting the above plots together

```
# COMBINING THE ABOVE PLOTS


################## Combine GP plots 1 - 4 ######################
line = 1
cex = 1
side = 3
adj=-0.05

# To save the comined plot as .tiff format

# tiff("D:/inMODELA TRAINING/CONSULTANCY 1/BMC medical journal manuscript
review/Plots/Figure2.tiff", width=12, height=10, units="in", res=300)  #
Adjust width, height, and resolution as needed

par(mfrow=c(2,2), oma=c(1,6,1,1))

# To include more options use the below line
# par(mfrow=c(2,2), oma=c(1,6,1,1), cex.lab=1.5, cex.axis=1.2, cex.sub=1.2,
font.lab=2)
### PLOT 1
# HIV
x <- seq(2018,2028, l=11)
y <- seq(0.0481,0.0255, l=11)
# The simulator's confidence intervals
```

```
l =
c(0.0419,0.0395,0.0373,0.0351,0.0330,0.0309,0.0290,0.0271,0.0254,0.0238,0.022
0)
u =
c(0.0543,0.0512,0.0483,0.0455,0.0430,0.0402,0.0378,0.0354,0.0333,0.0312,0.029
0
)
# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
# Run the following lines of code together
plot(x,y,xlab="years", ylab= "HIV Prevalence",type="b", lwd=2, pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)
lines(x,u,col="yellow",lwd=4)
mtext("A", side=side, line=line, cex=1.5, adj=adj)

# PLOT 2
# Hypertension
x <- seq(2018,2028, l=11)
y <- seq(0.2947,0.3430, l=11)
# Fitting the gaussian process
# The simulator's confidence intervals
l =
c(0.2845,0.2908,0.2965,0.3017,0.3073,0.3125,0.3171,0.3213,0.3250,0.3284,0.331
3)
u =
c(0.3050,0.3116,0.3177,0.3232,0.3292,0.3347,0.3396,0.3441,0.3480,0.3516,0.354
7)
# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "Hypertension Prevalence",type="b", lwd=2,
pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)
lines(x,u,col="yellow",lwd=4)
mtext("B", side=side, line=line, cex=1.5, adj=adj)
# PLOT 3
# Comorbid HIV and hypertension
x <- seq(2018,2028, l=11)
y <- seq(0.0206,0.0131, l=11)
```

```r
# Fitting the gaussian process
# The simulator's confidence intervals
l =
c(0.0179,0.0173,0.0164,0.0158,0.0151,0.0144,0.0138,0.0132,0.0126,0.0120,0.011
4)
u =
c(0.0232,0.0224,0.0212,0.0203,0.0195,0.0186,0.0179,0.0170,0.0163,0.0156,0.014
8)
# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "HIV & Hypertension Prevalence",type="b",
lwd=2, pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="yellow",lwd=4)
lines(x,u,col="yellow",lwd=4)
mtext("C", side=side, line=line, cex=1.5, adj=adj)
# PLOT 4
x <- seq(2018,2028, l=11)
y <- seq(0.039,0.037, l=11)
# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "Depression Prevalence",type="b", lwd=2,
pch=16)
curve(gp$predict(x), add=T, col = 2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col = 4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col = 4)
mtext("D", side=side, line=line, cex=1.5, adj=adj)
```




```
# dev.off()  # Close the TIFF device
```

BAYESIAN POSTERIOR PREDICTIVE ANALYSIS

- This step provides posterior predictive distributions that can be used to analyze the credibility of future observable data based on the posterior distribution.
- The procedure is implemented via the Bayesian inference rstanarm package
- A detailed guide of the posterior predictive analysis can be found here:

https://cran.r-project.org/web/packages/bayesplot/vignettes/graphical-ppcs.html

```
# Combination
side=3
line = 1
cex = 1
adj=-0.05

par(mfrow=c(2,2), oma=c(1,6,1,1))


set.seed(1234)


# PLOT 1
# HIV
```

```r
x <- c(2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028)
y <-
c(0.0481,0.0454,0.0428,0.0403,0.0380,0.0355,0.0334,0.0313,0.0294,0.0275,0.025
5)
# R stanarm
sm1   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 9.8e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.98 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.172 seconds (Warm-up)
## Chain 1:                0.293 seconds (Sampling)
## Chain 1:                1.465 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.726 seconds (Warm-up)
## Chain 2:                0.387 seconds (Sampling)
## Chain 2:                1.113 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.18 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.292 seconds (Warm-up)
## Chain 3:                0.332 seconds (Sampling)
## Chain 3:                1.624 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.763 seconds (Warm-up)
## Chain 4:                0.343 seconds (Sampling)
## Chain 4:                1.106 seconds (Total)
## Chain 4:
```

```
# Y predicted
ypred <- posterior_predict(sm1, draws = 40)
# density plot
plot(density(y),col="black",lwd=4, xlab="HIV prevalence", main="HIV")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]),col="grey",lty=3)}
#lines(density(colMeans(ypred)), col="red", lty=2, lwd=3)
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("A", side=side, line=line, cex=cex, adj=adj)
```

```
# PLOT 2
# Hypertension
x <- c(2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028)
y <-
c(0.2947,0.3012,0.3071,0.3125,0.3182,0.3236,0.3283,0.3327,0.3365,0.3400,0.343
0)
# R stanam
sm2   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.268 seconds (Warm-up)
## Chain 1:                0.125 seconds (Sampling)
## Chain 1:                0.393 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.369 seconds (Warm-up)
## Chain 2:                0.148 seconds (Sampling)
## Chain 2:                0.517 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.314 seconds (Warm-up)
## Chain 3:                0.17 seconds (Sampling)
## Chain 3:                0.484 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.22 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.369 seconds (Warm-up)
## Chain 4:                0.135 seconds (Sampling)
## Chain 4:                0.504 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm2, draws = 40)
# density plot
plot(density(y),col="black", lwd=4, xlab="Hypertension prevalence",
main="Hypertension")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]),col="grey",lty=3)}
#lines(density(colMeans(ypred)), col="red", lty=2, lwd=3)
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
```

```
mtext("B", side=side, line=line, cex=cex, adj=adj)



# PLOT 3
# Comorbid HIV and hypertension
x <- c(2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028)
y <-
c(0.0206,0.0198,0.0188,0.0181,0.0173,0.0165,0.0158,0.0151,0.0144,0.0138,0.013
1)
# R stanam
sm3   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.523 seconds (Warm-up)
## Chain 1:                1.122 seconds (Sampling)
## Chain 1:                3.645 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.19 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.085 seconds (Warm-up)
## Chain 2:                0.806 seconds (Sampling)
## Chain 2:                3.891 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.18 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.624 seconds (Warm-up)
## Chain 3:                0.984 seconds (Sampling)
## Chain 3:                3.608 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
```

```
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.021 seconds (Warm-up)
## Chain 4:                0.843 seconds (Sampling)
## Chain 4:                2.864 seconds (Total)
## Chain 4:
```

```
# Y predicted
ypred <- posterior_predict(sm3, draws = 40)
# density plot
plot(density(y), col="black", lwd=4, xlab="HIV & Hypertension prevalence",
main="HIV & Hypertension")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]), col="grey", lty=3)}
#lines(density(colMeans(ypred)), col="red", lty=2, lwd=3)
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("C", side=side, line=line, cex=cex, adj=adj)


# PLOT 4
# Depression
x <- c(2018,2020,2025,2030,2035)
y <- c(0.039,0.038,0.037,0.037,0.036)
# R stanam
sm4   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)
```

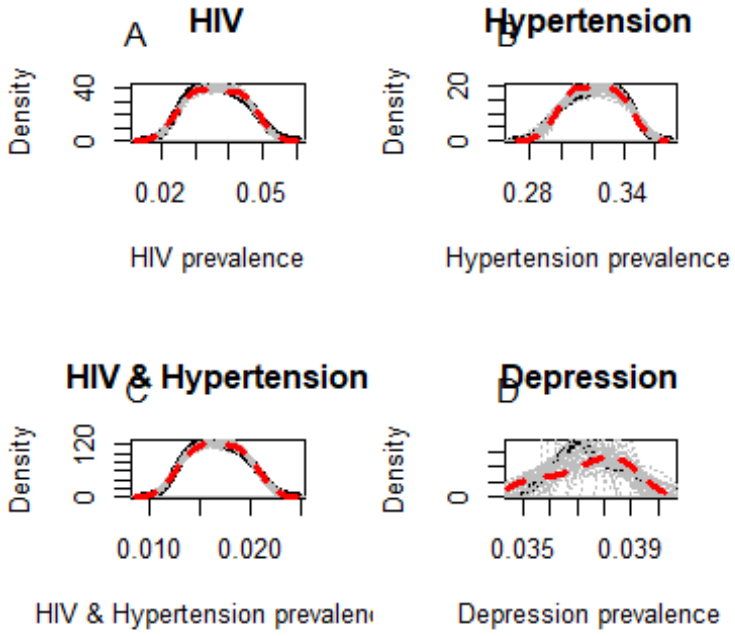
```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.509 seconds (Warm-up)
## Chain 1:                0.345 seconds (Sampling)
## Chain 1:                1.854 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.36 seconds (Warm-up)
## Chain 2:                0.176 seconds (Sampling)
## Chain 2:                1.536 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.538 seconds (Warm-up)
## Chain 3:                0.22 seconds (Sampling)
## Chain 3:                1.758 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.156 seconds (Warm-up)
## Chain 4:                0.292 seconds (Sampling)
## Chain 4:                1.448 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm4, draws = 40)
# density plot
plot(density(y),col="black", lwd=4, xlab="Depression prevalence",
main="Depression")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
```

```
{lines(density(ypred[i,]),col="grey",lty=3)}
#lines(density(colMeans(ypred)), col="red", lty=2, lwd=3)
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("D", side=side, line=line, cex=cex, adj=adj)
```



EMULATOR VALIDATION

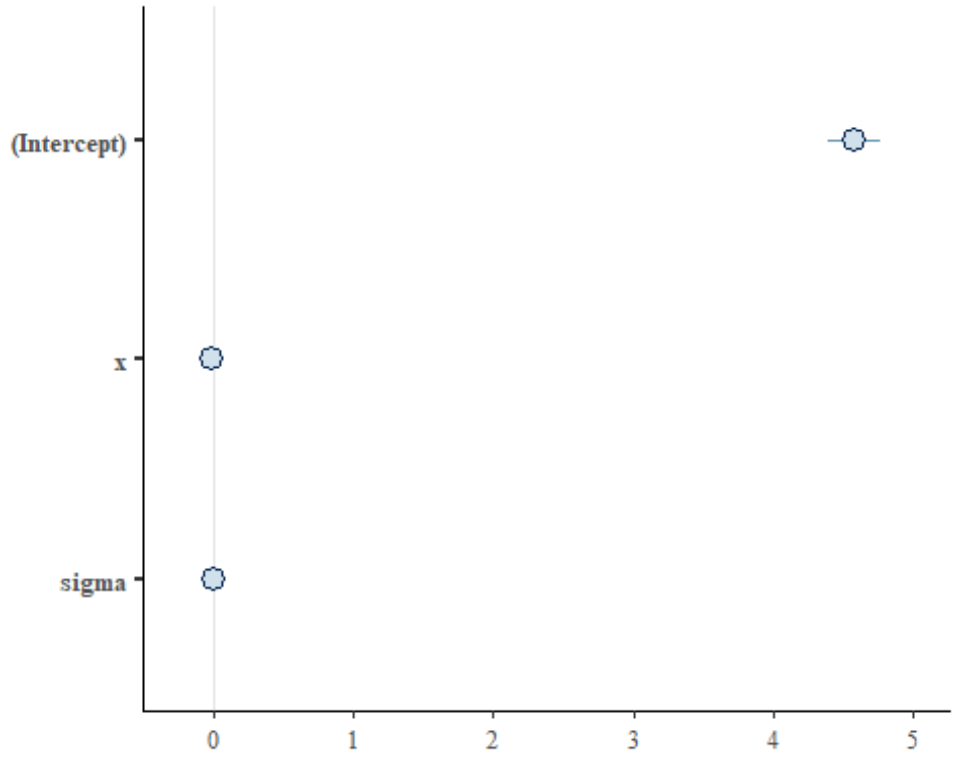
- We evaluate and validate our emulator using the leave-one-out cross validation (LOO) technique
- The technique is implemented using the loo R package whereby the computation is done using Pareto-smoothed importance sampling (PSIS) which regularizes importance weights
- A detailed guide can be found here: https://cran.r-project.org/web/packages/loo/vignettes/loo2-large-data.html

https://andrewproctor.github.io/rcourse/module6.html#model_testing

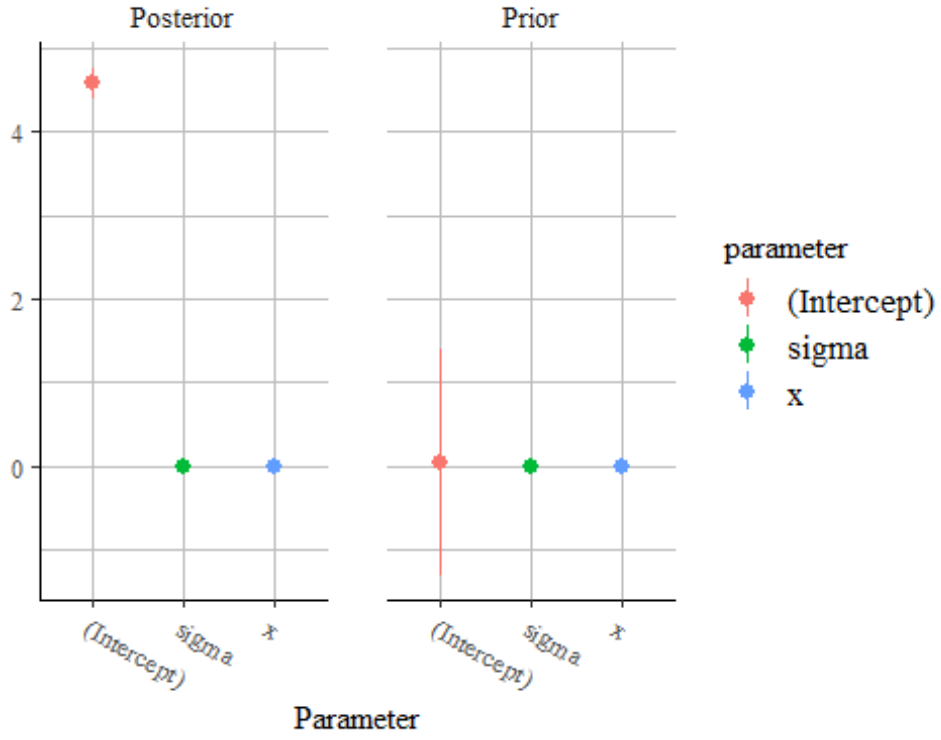
```
# 1. Validating the accuracy of the HIV emulation model compared to the
simulation model
# We call sm1 from the above posterior predictive analysis
# Getting the credible intervals
posterior_interval(sm1, prob=0.95) # Use sm1 from the above posterior
predictive analysis to obtain the 95% credible intervals

##                       2.5%         97.5%
## (Intercept)  4.3419850877  4.7978427215
```

```
## x           -0.0023537363 -0.0021286166
## sigma        0.0003401975  0.0009479938
```

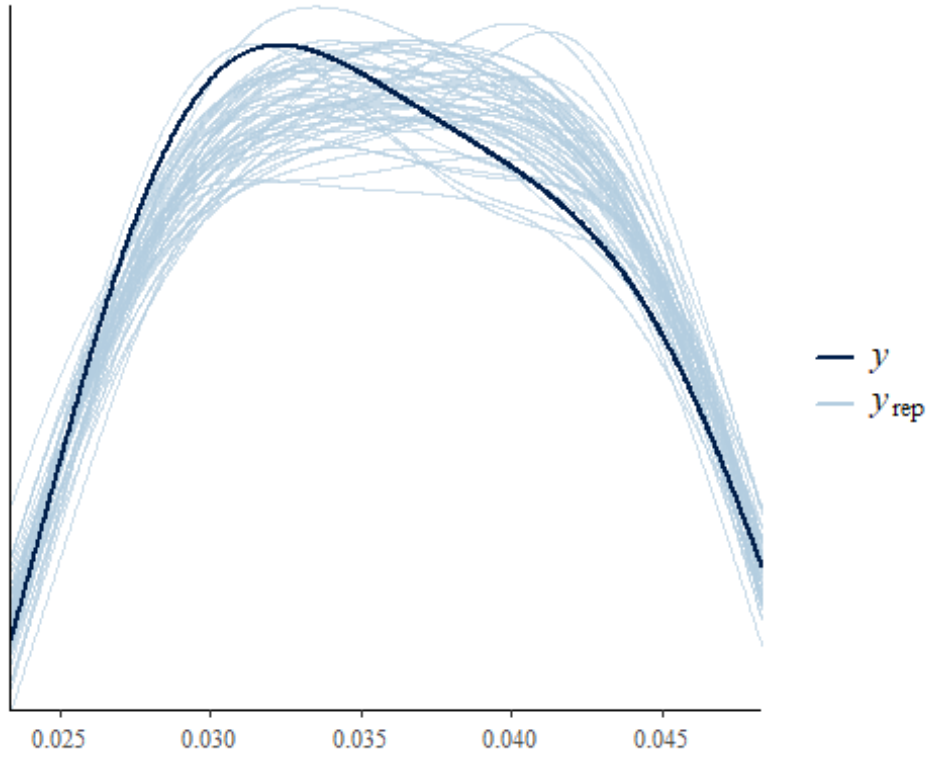
```
plot(sm1) # Graphically display the 95% credible intervals
```




```
posterior_vs_prior(sm1) # Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm1) # Graphically display the posterior predictive analysis
```




```
waic(sm1) # Watanabe-Akaike information criterion
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     66.1 1.7
## p_waic         2.4 0.8
## waic        -132.2 3.5
##
## 2 (18.2%) p_waic estimates greater than 0.4. We recommend trying loo
instead.
```

```
loo(sm1) # Leave One Out Cross-Validation
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     65.8 1.9
## p_loo         2.7 1.0
## looic      -131.6 3.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     9     81.8%   803
##  (0.5, 0.7]   (ok)       1      9.1%   204
##    (0.7, 1]   (bad)      1      9.1%   78
##    (1, Inf)   (very bad) 0      0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
# In this tutorial, the Pareto k diagnostic is set at 0.7.
# the Pareto K diagnostic estimates how far an individual leave-one-out
distribution is from the full distribution.
# If k<0.5, then the corresponding component is estimated with high accuracy.
If 0.5<k<0.7 the accuracy is lower, but still ok.

# If # This line gives a warning.
# It is recommended to call 'loo' again with argument 'k_threshold = 0.7'
# in order to calculate the ELPD without the assumption that these
observations are negligible.
# This will refit the model 1 times to compute the ELPDs for the problematic
observations directly.
loo(sm1, k_threshold = 0.7)
```

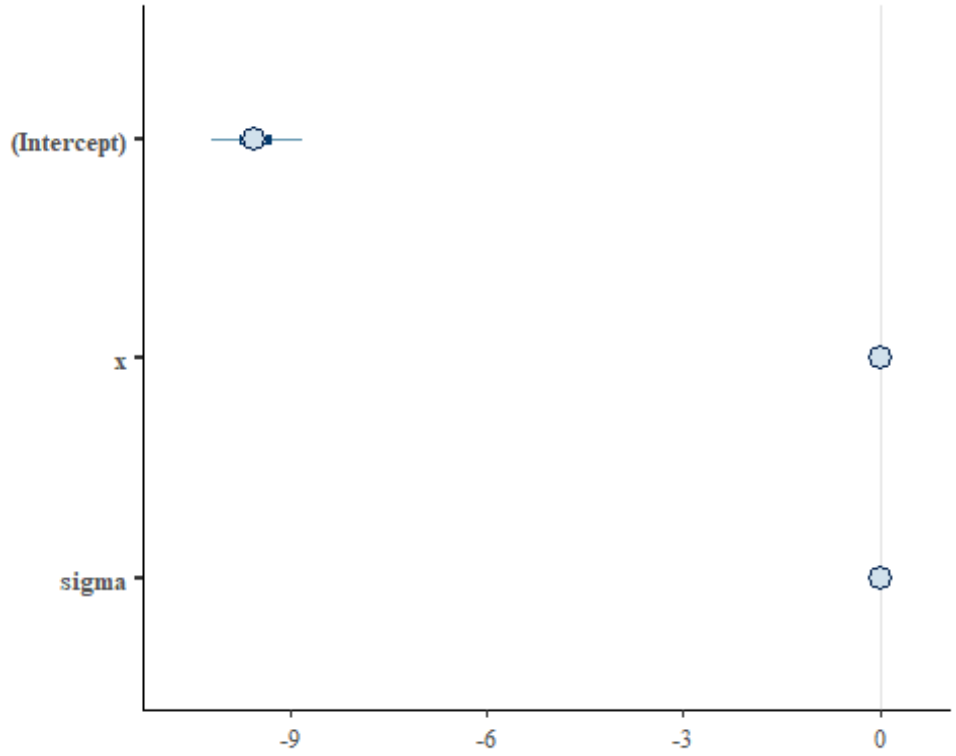
```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     65.8 1.9
## p_loo         2.7 1.0
```

```
## looic      -131.6 3.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     9     90.0%   803
##  (0.5, 0.7]   (ok)       1     10.0%   204
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

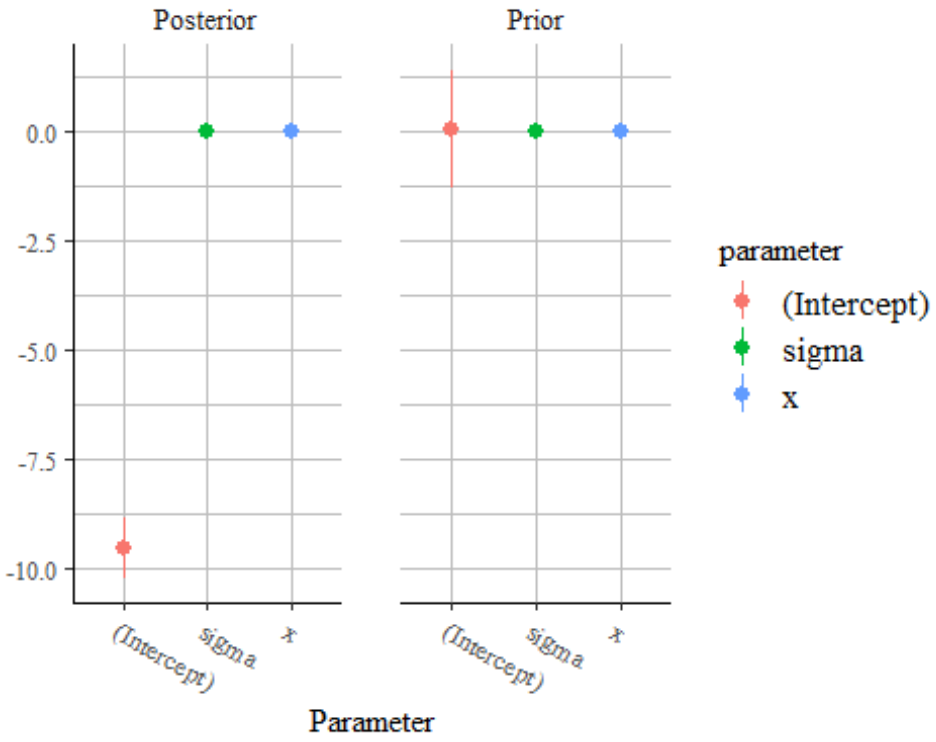
```
# 2. Validating the accuracy of the HTN emulation model compared to the simulation model

posterior_interval(sm2, prob=0.95) # Use sm2 from the above posterior predictive analysis to obtain the 95% credible intervals
```

```
##                       2.5%        97.5%
## (Intercept) -10.374898784 -8.649424454
## x             0.004433931  0.005287458
## sigma         0.001288169  0.003443498
```

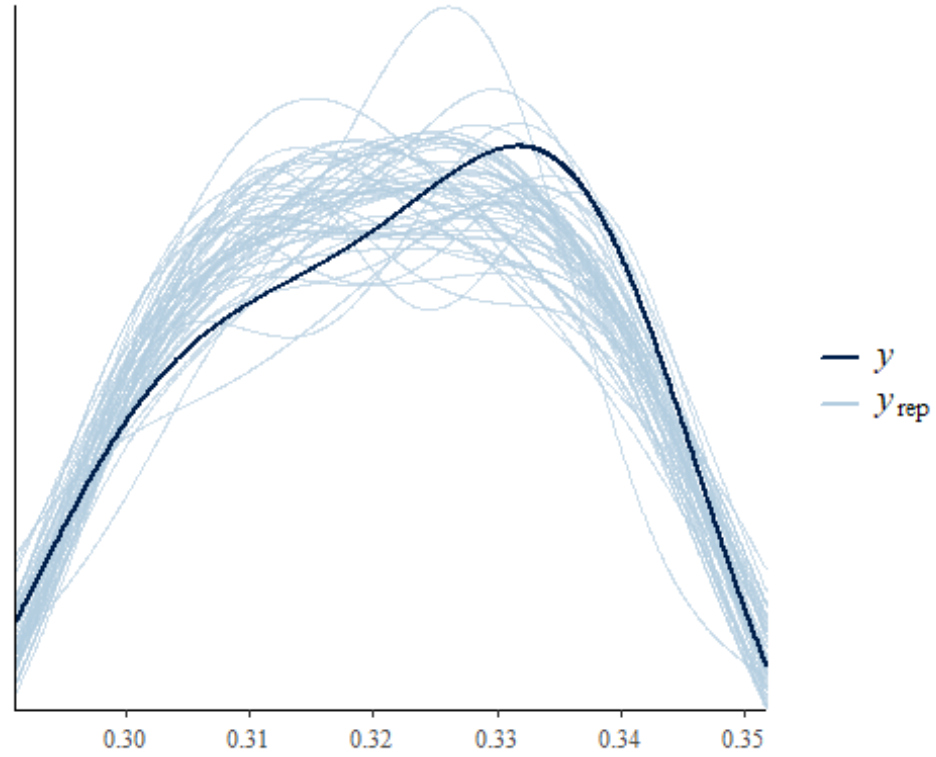
```
plot(sm2)# Graphically display the 95% credible intervals
```



```
posterior_vs_prior(sm2)# Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm2)# Graphically display the posterior predictive analysis
```



```r
waic(sm2) # Watanabe-Akaike information criterion
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     51.6 2.0
## p_waic         2.6 1.0
## waic        -103.1 4.1
##
## 2 (18.2%) p_waic estimates greater than 0.4. We recommend trying loo
instead.
```

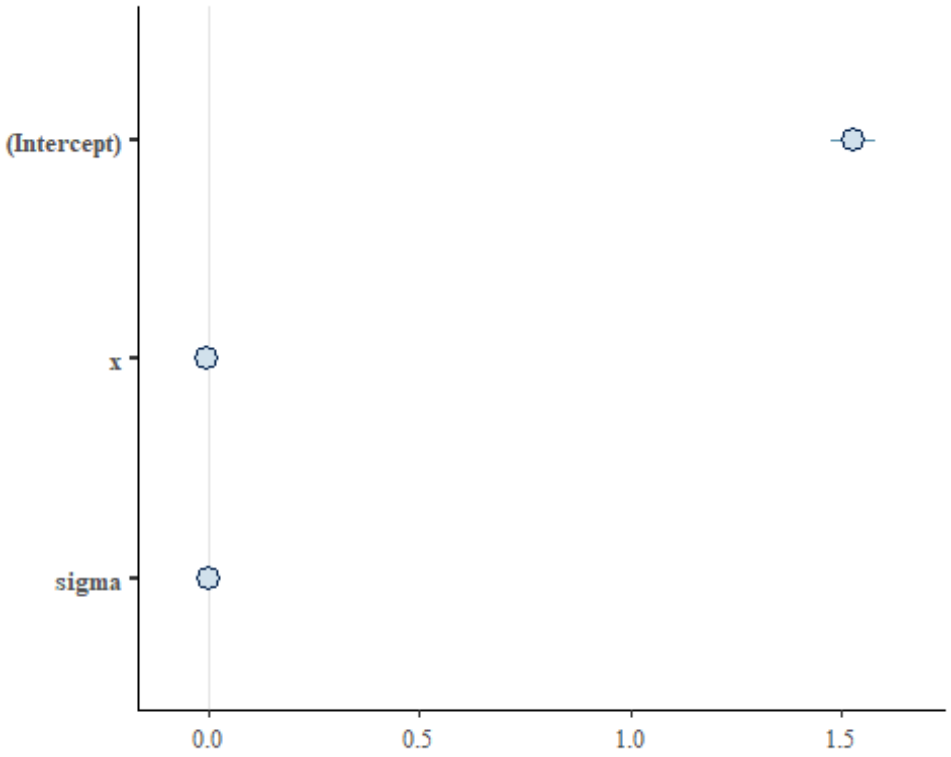
```r
# Leave One Out Cross-Validation
loo(sm3, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     79.6 1.4
## p_loo         2.3 0.7
## looic      -159.2 2.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     9     81.8%   1060
##  (0.5, 0.7]   (ok)       2     18.2%   347
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

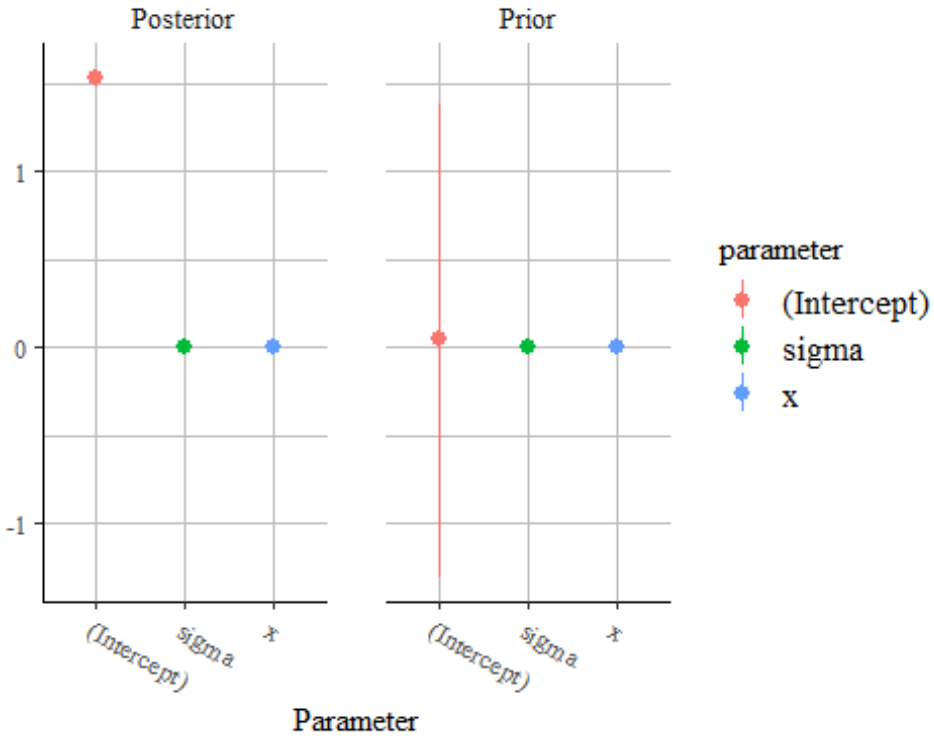
```r
# 3. Validating the accuracy of the HIV+HTN emulation model compared to the
simulation model

posterior_interval(sm3, prob=0.95) # Use sm3 from the above posterior
predictive analysis to obtain the 95% credible intervals
```

```
##                      2.5%         97.5%
## (Intercept)  1.4636864717  1.5922169602
## x           -0.0007788131 -0.0007152793
## sigma        0.0001021339  0.0002901978
```

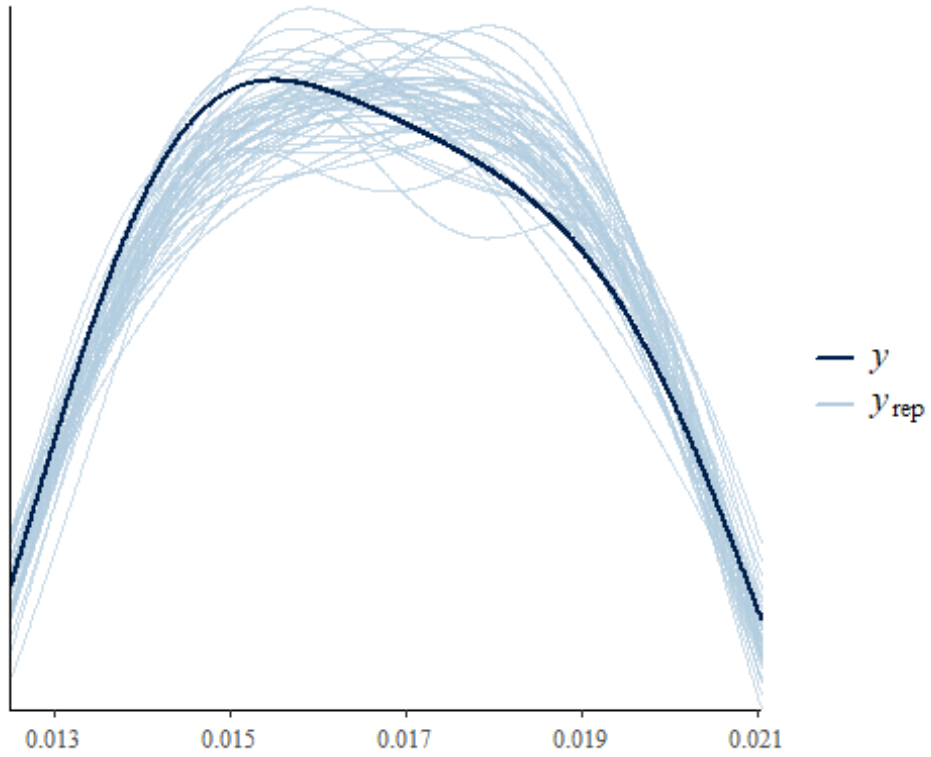
```r
plot(sm3) # Graphically display the 95% credible intervals
```




```
posterior_vs_prior(sm3) # Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm3) # Graphically display the posterior predictive analysis
```




```
waic(sm3) # Watanabe-Akaike information criterion

##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     79.8 1.3
## p_waic         2.2 0.6
## waic        -159.6 2.7
##
## 2 (18.2%) p_waic estimates greater than 0.4. We recommend trying loo
instead.

loo(sm3)   # Leave One Out Cross-Validation

##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     79.6 1.4
## p_loo         2.3 0.7
## looic      -159.2 2.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
```

```
## (-Inf, 0.5]   (good)     9     81.8%   1060
##  (0.5, 0.7]   (ok)       2     18.2%   347
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(sm3, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

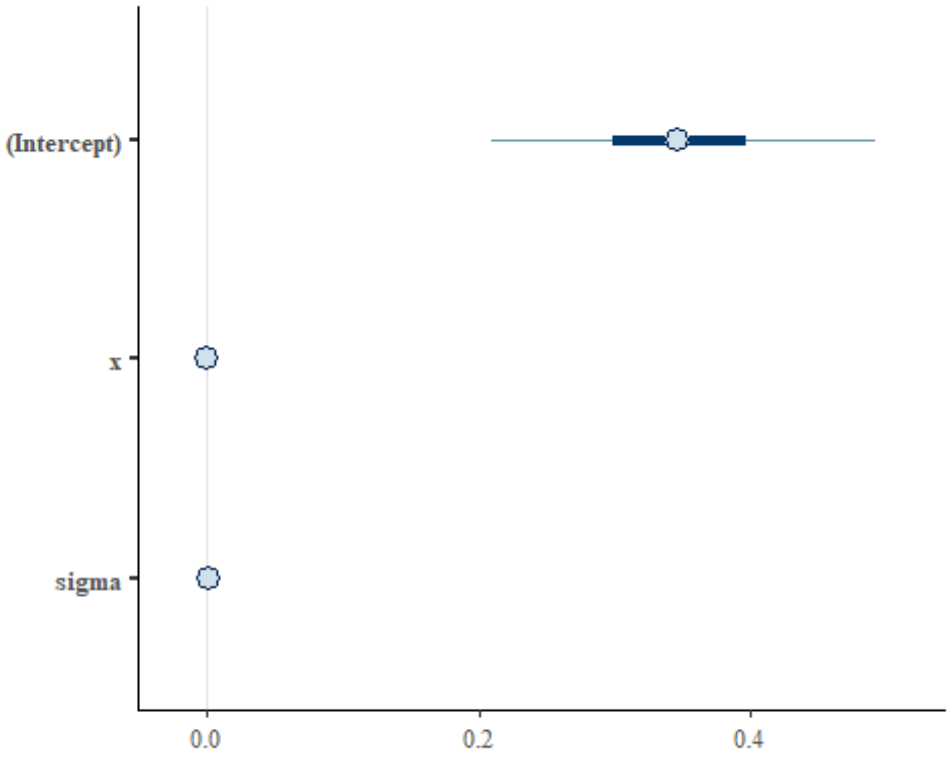
```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     79.6 1.4
## p_loo         2.3 0.7
## looic      -159.2 2.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     9     81.8%   1060
##  (0.5, 0.7]   (ok)       2     18.2%   347
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
# 4. Validating the accuracy of the depression emulation model compared to
the simulation model
```

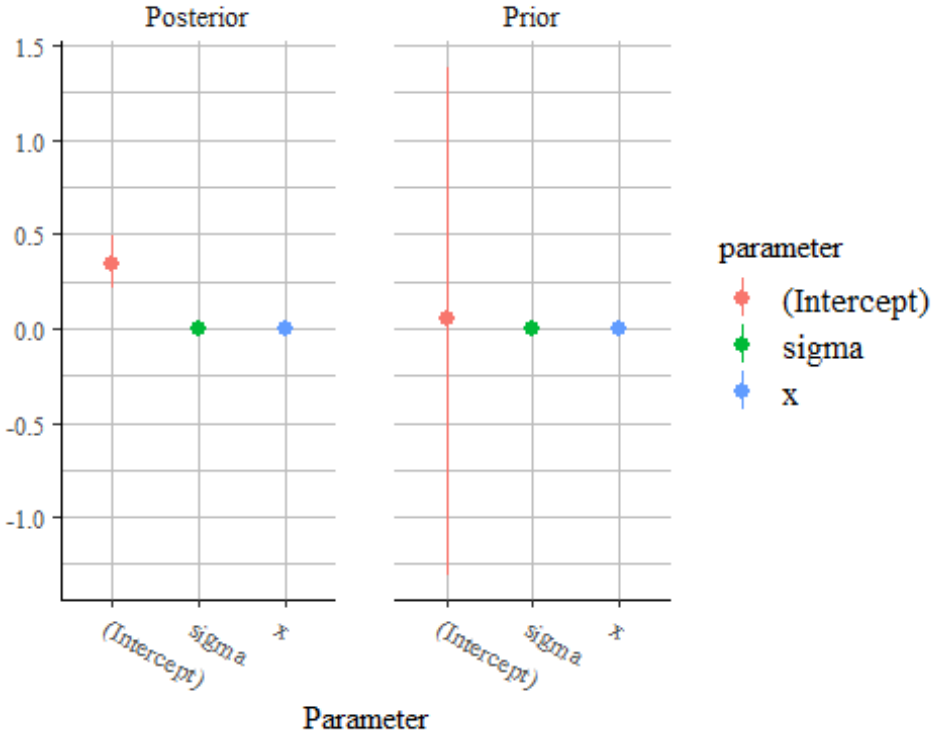
```
posterior_interval(sm4, prob=0.95) # Use sm4 from the above posterior
predictive analysis to obtain the 95% credible intervals
```

```
##                       2.5%         97.5%
## (Intercept)  0.1592922534  5.415858e-01
## x           -0.0002489235 -6.029133e-05
## sigma        0.0002577907  1.653513e-03
```

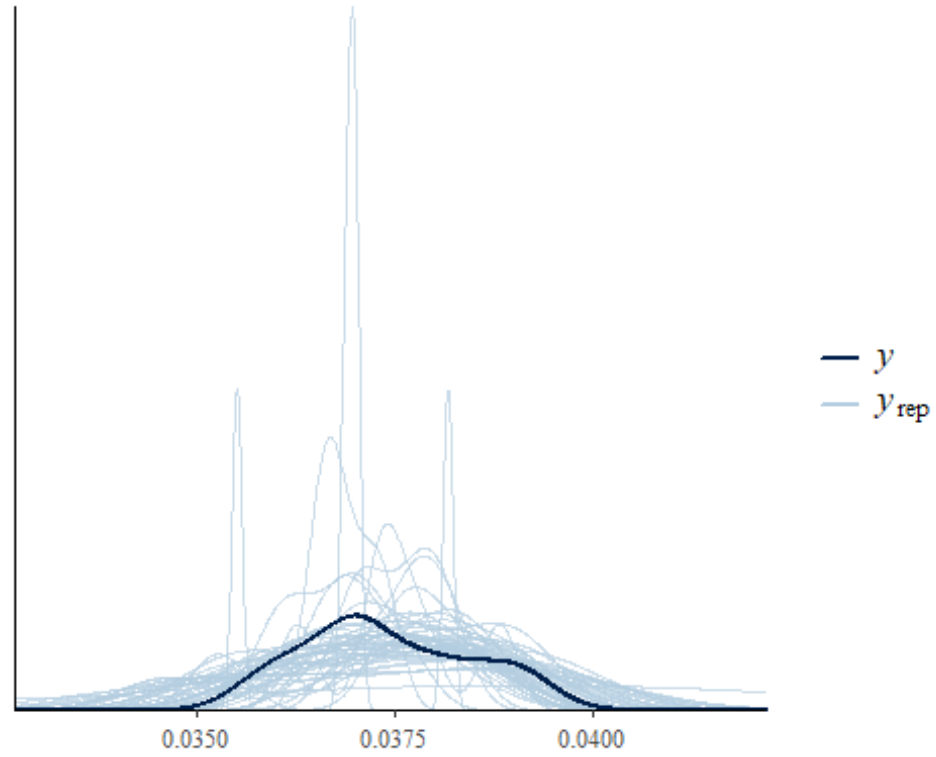
```
plot(sm4) # Graphically display the 95% credible intervals
```




```
posterior_vs_prior(sm4) # Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm4) # Graphically display the posterior predictive analysis
```




```
waic(sm4) # Watanabe-Akaike information criterion

##
## Computed from 4000 by 5 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic     29.5 0.5
## p_waic         1.8 0.2
## waic         -59.0 1.1
##
## 2 (40.0%) p_waic estimates greater than 0.4. We recommend trying loo
instead.

loo(sm4) # Leave One Out Cross-Validation

##
## Computed from 4000 by 5 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     29.3 0.6
## p_loo         2.1 0.3
## looic       -58.6 1.2
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
```

```
## (-Inf, 0.5]   (good)     2     40.0%   633
##  (0.5, 0.7]   (ok)       1     20.0%   174
##    (0.7, 1]   (bad)      2     40.0%   165
##    (1, Inf)   (very bad) 0      0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
loo(sm4, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

```
##
## Computed from 4000 by 5 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo     29.1 0.8
## p_loo         2.3 0.5
## looic       -58.1 1.6
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     2     66.7%   633
##  (0.5, 0.7]   (ok)       1     33.3%   174
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

Loo output explanation

- From the above loo cross validation outputs, we observe that all the four GP models have pareto k estimates with k-threshold values of k<0.7 which are okay. -We therefore validate that our emulator is a good representation of the simulators.

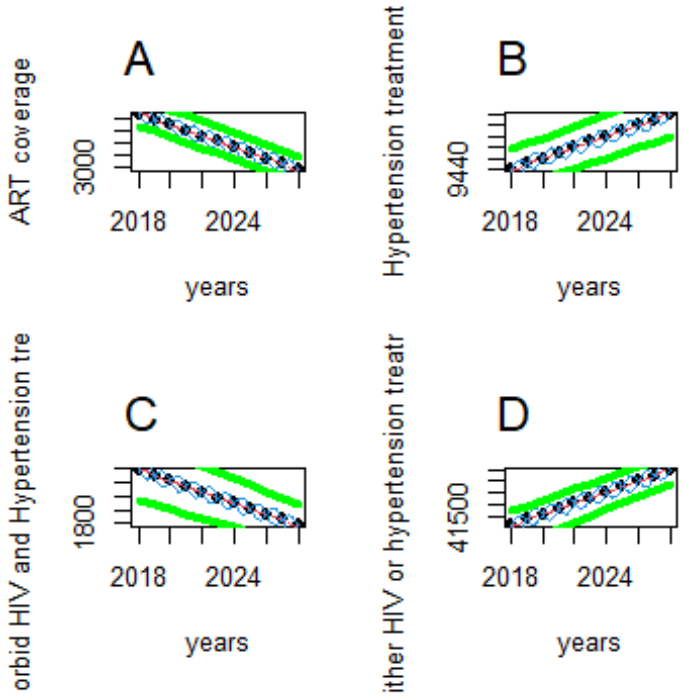
ADDITIONAL EMULATIONS TO APPROXIMATE HEALTH SYSTEM BURDEN

The following chunks of code represent the additional emulations performed to approximate simulations for different hypothetical intervention/disease control program options. We display the plots combined into 1

```
################## Combine GP plots 1 - 4 ######################
line = 1
cex = 1
side = 3
adj=-0.05

# To generate a .tiff plot use the below line of code
# tiff("D:/inMODELA TRAINING/CONSULTANCY 1/BMC medical journal manuscript
review/Plots/additional_emul1.tiff", width=12, height=10, units="in",
```

```r
res=300)  # Adjust width, height, and resolution as needed

par(mfrow=c(2,2), oma=c(1,6,1,1))

### PLOT 1
# ART coverage target
x <- seq(2018,2028, l=11) # Simulated years
y <- seq(5219,2935, l=11) # Target values
# The simulator's confidence intervals
l <- seq(4728,2434, length.out = 11)
u <- seq(5911,3436, length.out = 11)
# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
# Run the following lines of code together
plot(x,y,xlab="years", ylab= "ART coverage",type="b", lwd=2, pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="green",lwd=4)
lines(x,u,col="green",lwd=4)
mtext("A", side=side, line=line, cex=1.5, adj=adj)

# PLOT 2
# Hypertension treatment target
x <- seq(2018,2028, l=11) # Simulated years
y <- seq(9440,9540, l=11) # Target values
# Fitting the gaussian process
# The simulator's confidence intervals
l <- seq(9403, 9499, length.out=11)
u <- seq(9476, 9580, length.out=11)

# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "Hypertension treatment",type="b", lwd=2,
pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="green",lwd=4)
lines(x,u,col="green",lwd=4)
mtext("B", side=side, line=line, cex=1.5, adj=adj)

# PLOT 3
# Comorbid HIV and hypertension
```

```r
x <- seq(2018,2028, l=11) # Simulated years
y <- seq(2581,1774, l=11) # Target values
# Fitting the gaussian process
# The simulator's confidence intervals
l <- seq(2137, 1472, length.out=11)
u <- seq(3025, 2075, length.out=11)

# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "Comorbid HIV and Hypertension
treatment",type="b", lwd=2, pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="green",lwd=4)
lines(x,u,col="green",lwd=4)
mtext("C", side=side, line=line, cex=1.5, adj=adj)
# PLOT 4
# Either HIV or hypertension
x <- seq(2018,2028, l=11)
y <- seq(41108,43476, l=11)
# Fitting the gaussian process
# The simulator's confidence intervals
l <- seq(40171, 42858, length.out=11) # Simulated years
u <- seq(41700, 44093, length.out=11) # Target values

# Fitting the gaussian process
library(GauPro)
gp <- GauPro(x,y, parallel=FALSE)

# Plotting
plot(x,y, xlab="years", ylab= "Either HIV or hypertension
treatment",type="b", lwd=2, pch=16)#,
curve(gp$predict(x), add=T, col=2, lty=5)
curve(gp$predict(x)+1.96*gp$predict(x, se=T)$se, add=T, col=4)
curve(gp$predict(x)-1.96*gp$predict(x, se=T)$se, add=T, col=4)
lines(x,l,col="green",lwd=4)
lines(x,u,col="green",lwd=4)
mtext("D", side=side, line=line, cex=1.5, adj=adj)
```




```
# dev.off()  # Close the TIFF device
```

BAYESIAN POSTERIOR PREDICTIVE ANALYSIS FOR THE ADDITIONAL EMULATIONS

```
side=3
line = 1
cex = 1
adj=-0.05

# tiff("D:/inMODELA TRAINING/CONSULTANCY 1/BMC medical journal manuscript
review/Plots/additional_emul2.tiff", width=12, height=10, units="in",
res=300)  # Adjust width, height, and resolution as needed

par(mfrow=c(2,2), oma=c(1,6,1,1))

# To add more options to the plot, use the line below
# par(mfrow=c(2,2), oma=c(1,6,1,1), cex.lab=1.5, cex.axis=1.2, cex.sub=1.2,
font.lab=2)

set.seed(12345)

## PLOT 1 - Emulator's posterior predictions of ART coverage target
x <- seq(2018,2028, l=11)
y <- seq(5219,2935, l=11)
# Add some random noise to y to show variation
set.seed(123)  # Set seed for reproducibility
```

```r
y <- y + rnorm(length(y), mean = 0, sd = 50)
# using rstanarm's default priors. For details see the section on default
weakly informative priors at https://mc-
stan.org/rstanarm/articles/priors.html
sm1   <- stan_glm (y~x,
                   family=gaussian(),
# Specify distribution.
                   data = data.frame(x,y),
                   seed=1234)

##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.505 seconds (Warm-up)
## Chain 1:                0.066 seconds (Sampling)
## Chain 1:                0.571 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.5 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.864 seconds (Warm-up)
## Chain 2:                0.056 seconds (Sampling)
## Chain 2:                0.92 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.21 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.508 seconds (Warm-up)
## Chain 3:                0.056 seconds (Sampling)
## Chain 3:                0.564 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.445 seconds (Warm-up)
## Chain 4:                0.071 seconds (Sampling)
## Chain 4:                0.516 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm1, draws = 50)
# Specify the number of draws to return (50 at minimum).

# density plot
plot(density(y),col="black",lwd=3, xlab="ART coverage", main="")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]),col="lightblue",lty=3)}
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("A", side=side, line=line, cex=1.5, adj=adj)
```

```r
# PLOT 2 - Emulator's posterior predictions of Hypertension treatmenbt target
x <- seq(2018,2028, l=11)
y <- seq(9440,9540, l=11)
# Add some random noise to y to show variation
set.seed(123)  # Set seed for reproducibility
y <- y + rnorm(length(y), mean = 0, sd = 50)
# R stanam
sm2   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.136 seconds (Warm-up)
## Chain 1:                0.054 seconds (Sampling)
## Chain 1:                0.19 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.18 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.141 seconds (Warm-up)
## Chain 2:                0.053 seconds (Sampling)
## Chain 2:                0.194 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.166 seconds (Warm-up)
## Chain 3:                0.065 seconds (Sampling)
## Chain 3:                0.231 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.195 seconds (Warm-up)
## Chain 4:                0.055 seconds (Sampling)
## Chain 4:                0.25 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm2, draws = 50)
# density plot
plot(density(y),col="black", lwd=3, xlab="Hypertension treatment", main="")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]),col="lightblue",lty=3)}
```

```r
#lines(density(colMeans(ypred)), col="red", lty=2, lwd=3)
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("B", side=side, line=line, cex=1.5, adj=adj)



# PLOT 3 - Emulator's posterior predictions of Comorbid HIV and hypertension
treatment target
x <- seq(2018,2028, l=11)
y <- seq(2581,1774, l=11)
# Add some random noise to y to show variation
set.seed(123)  # Set seed for reproducibility
y <- y + rnorm(length(y), mean = 0, sd = 50)
# R stanam
sm3   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.444 seconds (Warm-up)
## Chain 1:                0.062 seconds (Sampling)
## Chain 1:                0.506 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
```

```
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.226 seconds (Warm-up)
## Chain 2:                0.068 seconds (Sampling)
## Chain 2:                0.294 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.7e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.302 seconds (Warm-up)
## Chain 3:                0.053 seconds (Sampling)
## Chain 3:                0.355 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
```

```
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.233 seconds (Warm-up)
## Chain 4:                0.055 seconds (Sampling)
## Chain 4:                0.288 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm3, draws = 50)
# density plot
plot(density(y), col="black", lwd=3, xlab="HIV & Hypertension treatment",
main="")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]), col="lightblue", lty=3)}
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("C", side=side, line=line, cex=1.5, adj=adj)
```

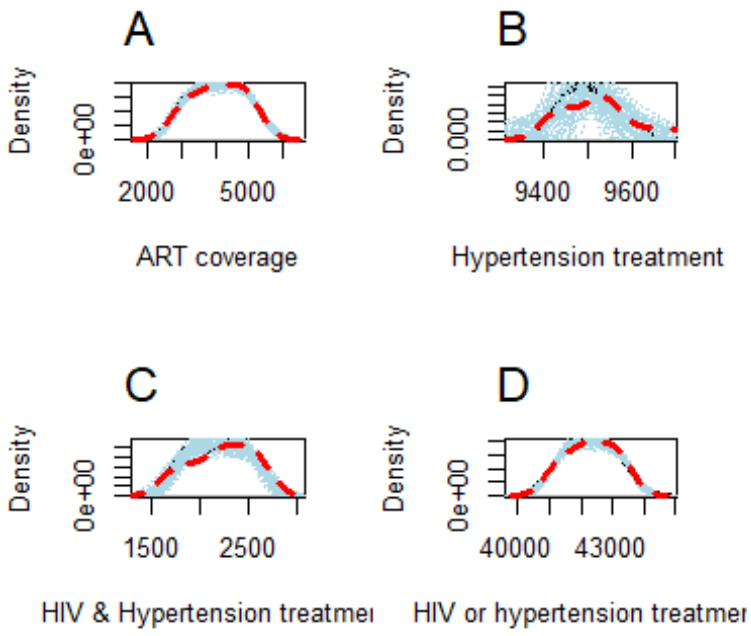
```r
# PLOT 4 - Emulator's posterior predictions of either HIV or hypertension
treatment target
x <- seq(2018,2028, l=11)
y <- seq(41108,43476, l=11)
# Add some random noise to y to show variation
set.seed(123)  # Set seed for reproducibility
y <- y + rnorm(length(y), mean = 0, sd = 50)
# R stanam
sm4   <- stan_glm(y~x, family=gaussian(), data = data.frame(x,y), seed=1234)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.767 seconds (Warm-up)
## Chain 1:                0.052 seconds (Sampling)
## Chain 1:                0.819 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.7e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.781 seconds (Warm-up)
## Chain 2:                0.062 seconds (Sampling)
## Chain 2:                0.843 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.7e-05 seconds
```

```
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 0.17 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.683 seconds (Warm-up)
## Chain 3:                0.054 seconds (Sampling)
## Chain 3:                0.737 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.18 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.072 seconds (Warm-up)
## Chain 4:                0.064 seconds (Sampling)
## Chain 4:                1.136 seconds (Total)
## Chain 4:
```

```r
# Y predicted
ypred <- posterior_predict(sm4, draws = 50)
# density plot
plot(density(y),col="black", lwd=3, xlab="HIV or hypertension treatment",
main = "")
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
for(i in 1:nrow(ypred))
{lines(density(ypred[i,]),col="lightblue",lty=3)}
lines(density(ypred[1,]), col="red", lty=2, lwd=3)
mtext("D", side=side, line=line, cex=1.5, adj=adj)
```




```r
# dev.off()  # Close the TIFF device
```

EMULATOR VALIDATION OF ADDITIONAL EMULATIONS

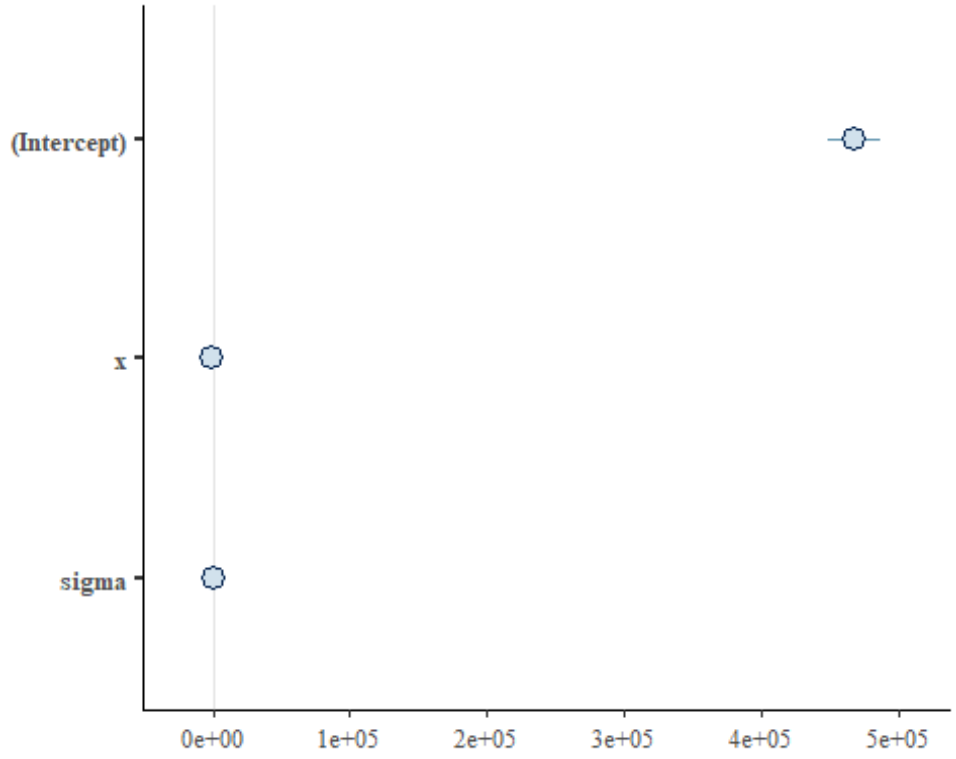
- We evaluate and validate our emulator using the leave-one-out cross validation (LOO) technique
- The technique is implemented using the loo R package whereby the computation is done using Pareto-smoothed importance sampling (PSIS) which regularizes importance weights
- A detailed guide can be found here: https://cran.r-project.org/web/packages/loo/vignettes/loo2-large-data.html

https://andrewproctor.github.io/rcourse/module6.html#model_testing

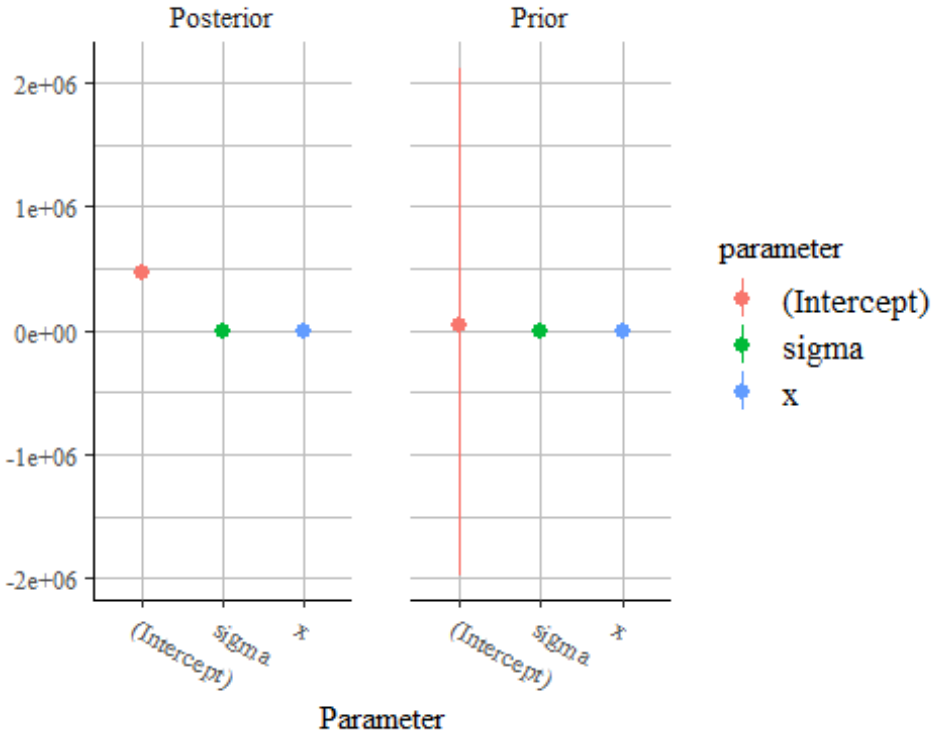
```r
# 1. Validating the accuracy of the ART coverage target emulation model
compared to the simulation model
# We call sm1 from the above posterior predictive analysis
# Getting the credible intervals
posterior_interval(sm1, prob=0.95) # Use sm1 from the above posterior
predictive analysis to obtain the 95% credible intervals
```

```
##                    2.5%       97.5%
## (Intercept) 443215.3060 490231.8999
## x             -240.2849   -217.0703
## sigma           35.9990    106.1454
```

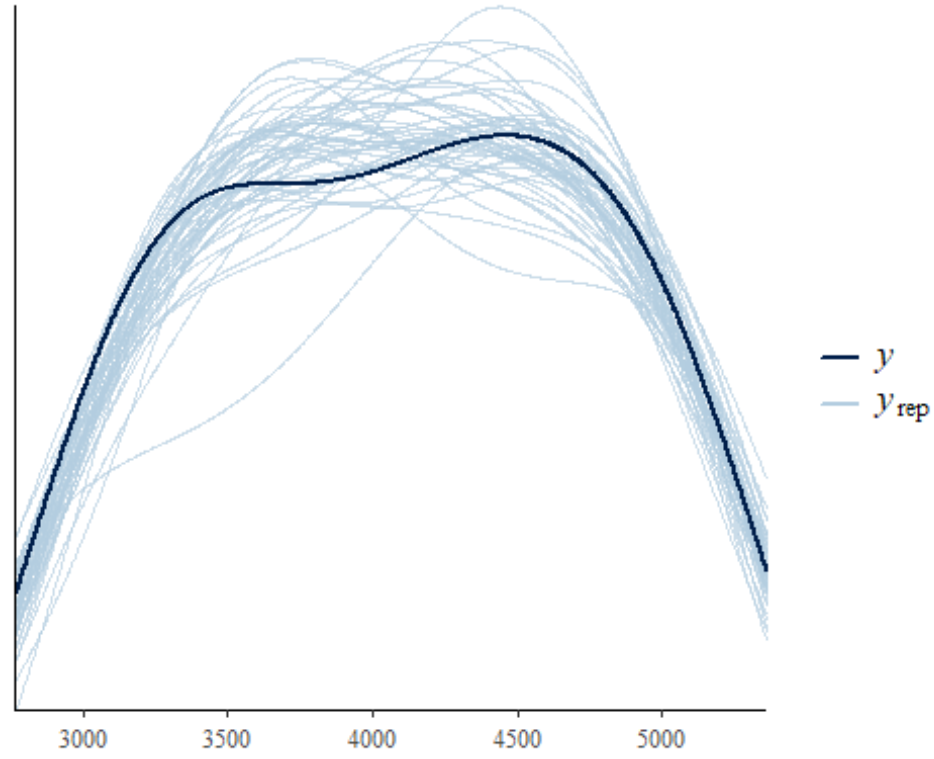
```r
plot(sm1) # Graphically display the 95% credible intervals
```




```r
posterior_vs_prior(sm1) # Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm1) # Graphically display the posterior predictive analysis
```




```
waic(sm1) # Watanabe-Akaike information criterion
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -61.0 1.4
## p_waic         2.2 0.4
## waic         122.0 2.9
```

```
loo(sm1) # Leave One Out Cross-Validation
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.1 1.5
## p_loo         2.3 0.4
## looic       122.2 2.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     10    90.9%   1214
##  (0.5, 0.7]   (ok)        1     9.1%   1371
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
# In this tutorial, the Pareto k diagnostic is set at 0.7.
# the Pareto K diagnostic estimates how far an individual leave-one-out
distribution is from the full distribution.
# If k<0.5, then the corresponding component is estimated with high accuracy.
If 0.5<k<0.7 the accuracy is lower, but still ok.

# If # This line gives a warning.
# It is recommended to call 'loo' again with argument 'k_threshold = 0.7'
# in order to calculate the ELPD without the assumption that these
observations are negligible.
# This will refit the model 1 times to compute the ELPDs for the problematic
observations directly.
loo(sm1, k_threshold = 0.7)
```

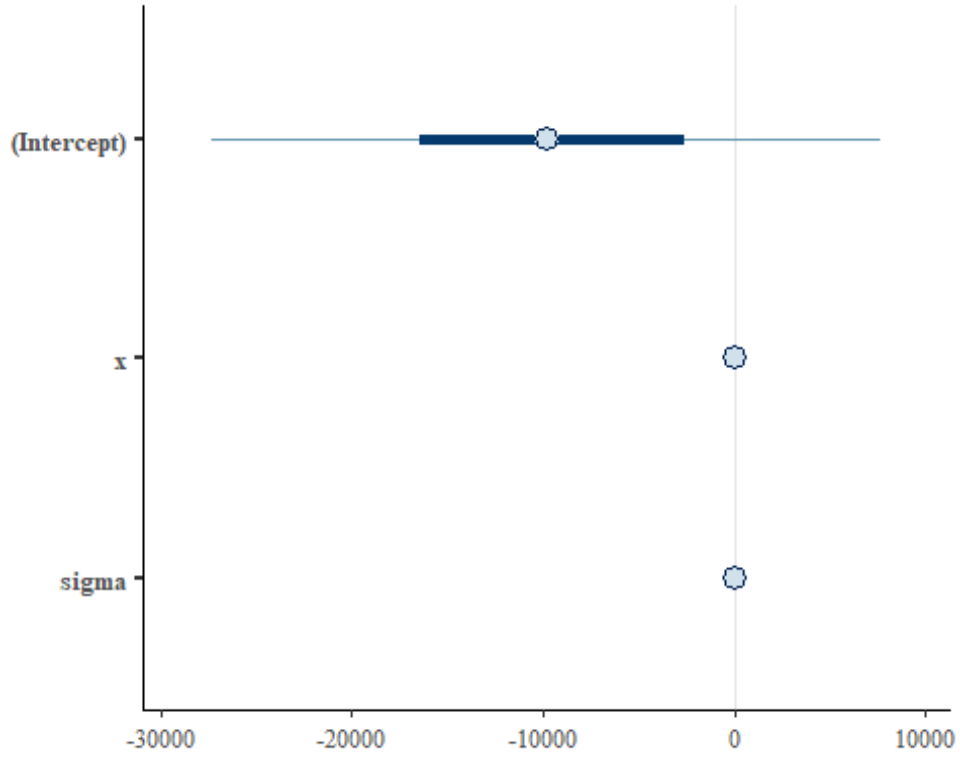
```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.1 1.5
## p_loo         2.3 0.4
## looic       122.2 2.9
```

```
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     10    90.9%   1214
##  (0.5, 0.7]   (ok)        1     9.1%   1371
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

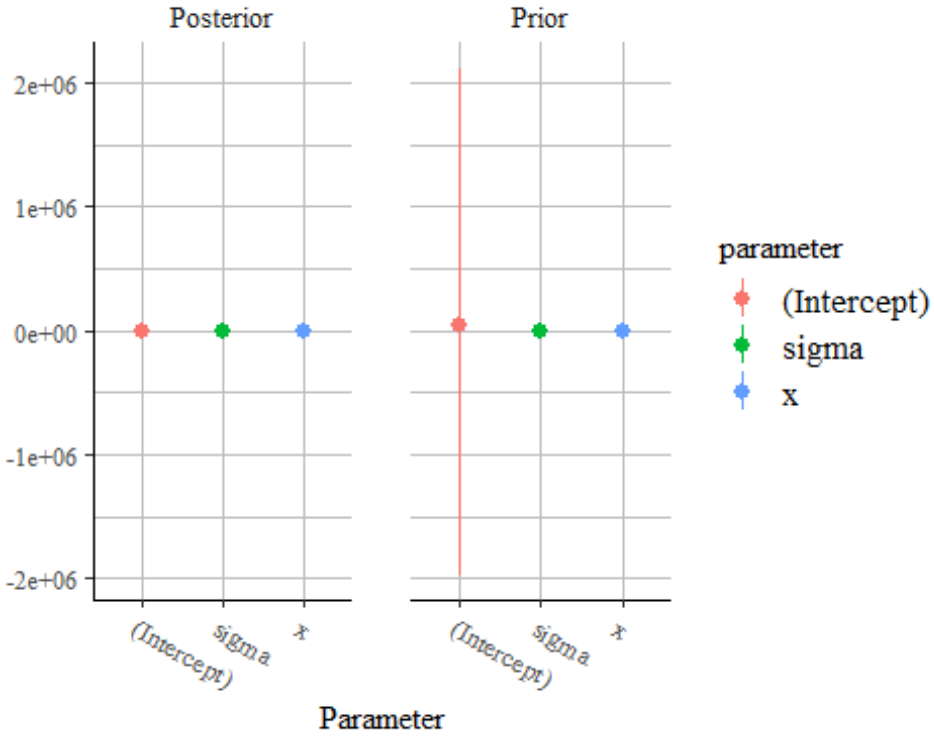
```
# 2. Validating the accuracy of the hypertension treatment target emulation model compared to the simulation model

posterior_interval(sm2, prob=0.95) # Use sm2 from the above posterior predictive analysis to obtain the 95% credible intervals
```

```
##                        2.5%       97.5%
## (Intercept) -3.099613e+04 11382.69256
## x           -9.559984e-01    20.00485
## sigma        3.552076e+01    90.45286
```

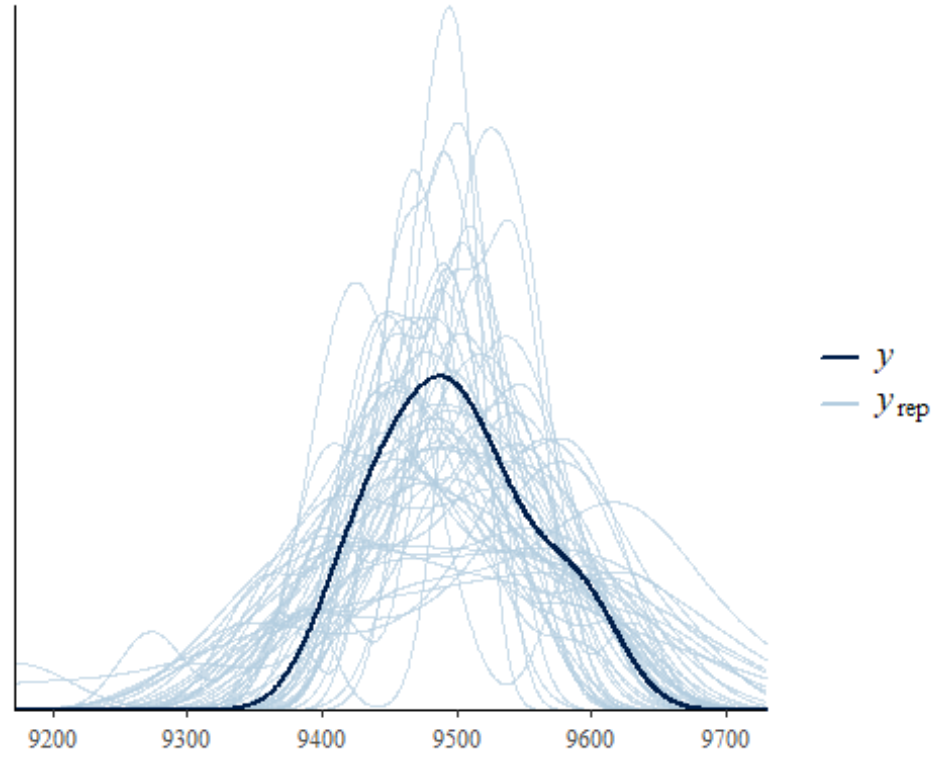
```
plot(sm2)# Graphically display the 95% credible intervals
```




```
posterior_vs_prior(sm2)# Juxtaposing the prior and the posterior estimates
```




```
pp_check(sm2)# Graphically display the posterior predictive analysis
```




```
waic(sm2) # Watanabe-Akaike information criterion
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -60.7 1.6
## p_waic         2.2 0.4
## waic         121.4 3.2
```

```
# Leave One Out Cross-Validation
loo(sm2)
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -60.8 1.6
## p_loo         2.3 0.4
## looic       121.6 3.2
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
loo(sm2, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

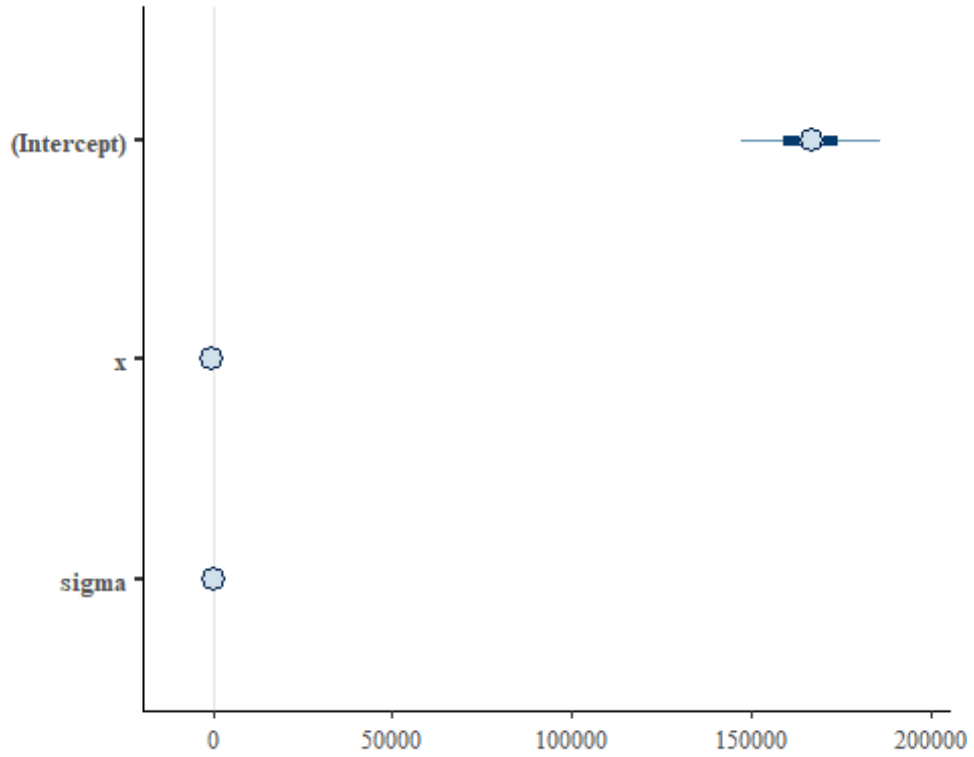
```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -60.8 1.6
## p_loo         2.3 0.4
## looic       121.6 3.2
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
# 3. Validating the accuracy of the comorbid HIV and hypertension treatment
target emulation model compared to the simulation model
```

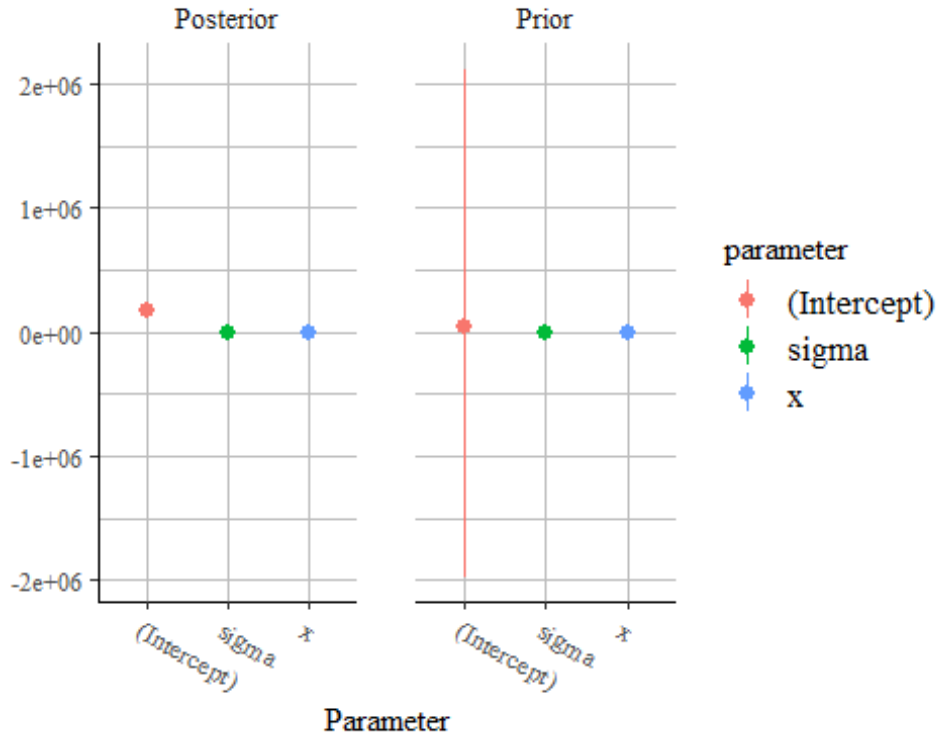
```
posterior_interval(sm3, prob=0.95) # Use sm3 from the above posterior
predictive analysis to obtain the 95% credible intervals
```

```
##                      2.5%        97.5%
## (Intercept) 142981.04874 190090.73022
## x              -92.90279    -69.60018
## sigma           36.43413     99.09866
```

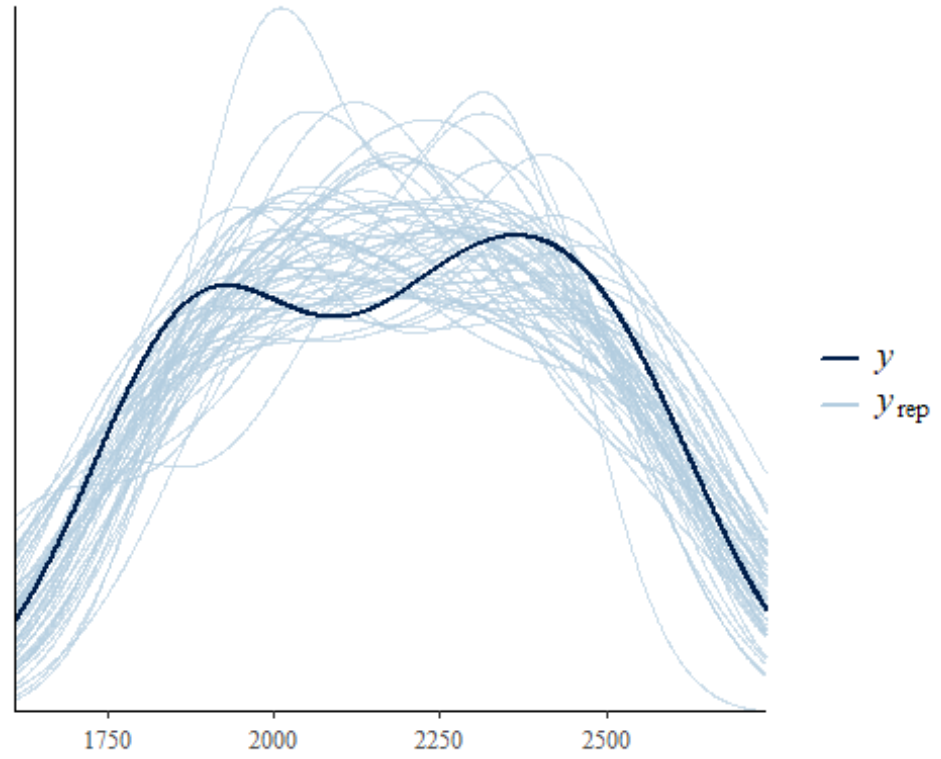
```
plot(sm3) # Graphically display the 95% credible intervals
```




```
posterior_vs_prior(sm3) # Juxtaposing the prior and the posterior estimates
```



```
pp_check(sm3) # Graphically display the posterior predictive analysis
```




```
waic(sm3) # Watanabe-Akaike information criterion

##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -60.9 1.4
## p_waic         2.1 0.4
## waic         121.8 2.9

loo(sm3)   # Leave One Out Cross-Validation

##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.0 1.5
## p_loo         2.2 0.4
## looic       122.1 2.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     10    90.9%   998
```

```
##  (0.5, 0.7]   (ok)        1     9.1%   1602
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(sm3, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

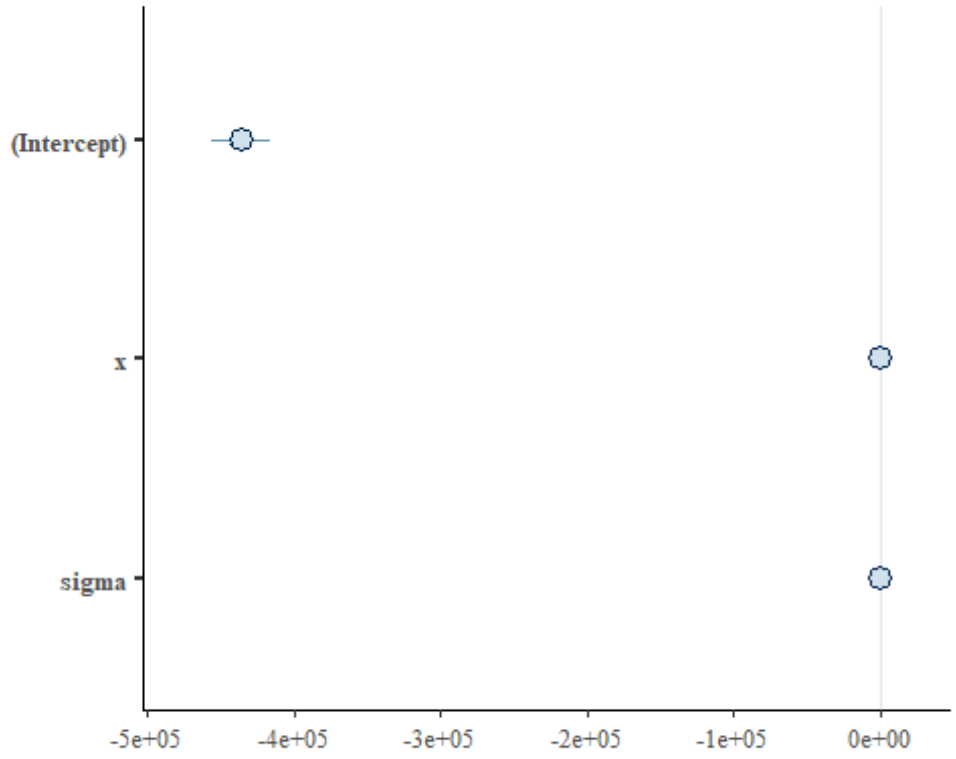
```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.0 1.5
## p_loo         2.2 0.4
## looic       122.1 2.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     10    90.9%   998
##  (0.5, 0.7]   (ok)        1     9.1%   1602
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
# 4. Validating the accuracy of the either HIV or hypertension treatment
target emulation model compared to the simulation model
```

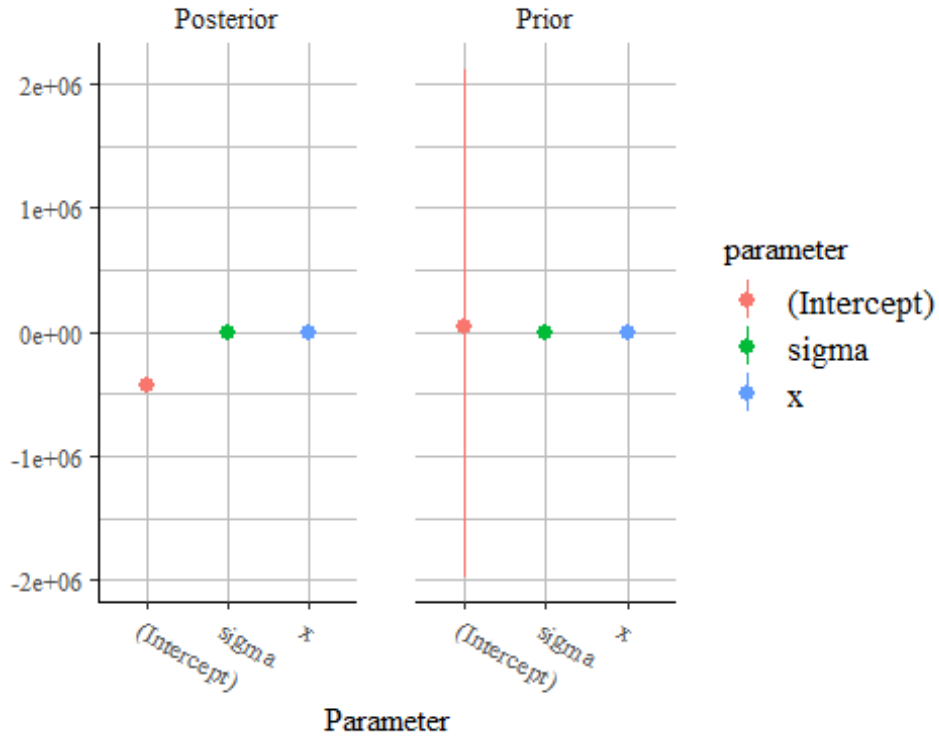
```
posterior_interval(sm4, prob=0.95) # Use sm4 from the above posterior
predictive analysis to obtain the 95% credible intervals
```

```
##                      2.5%        97.5%
## (Intercept) -460758.01397 -411604.0298
## x               224.37110     248.6686
## sigma            36.39897     104.1451
```

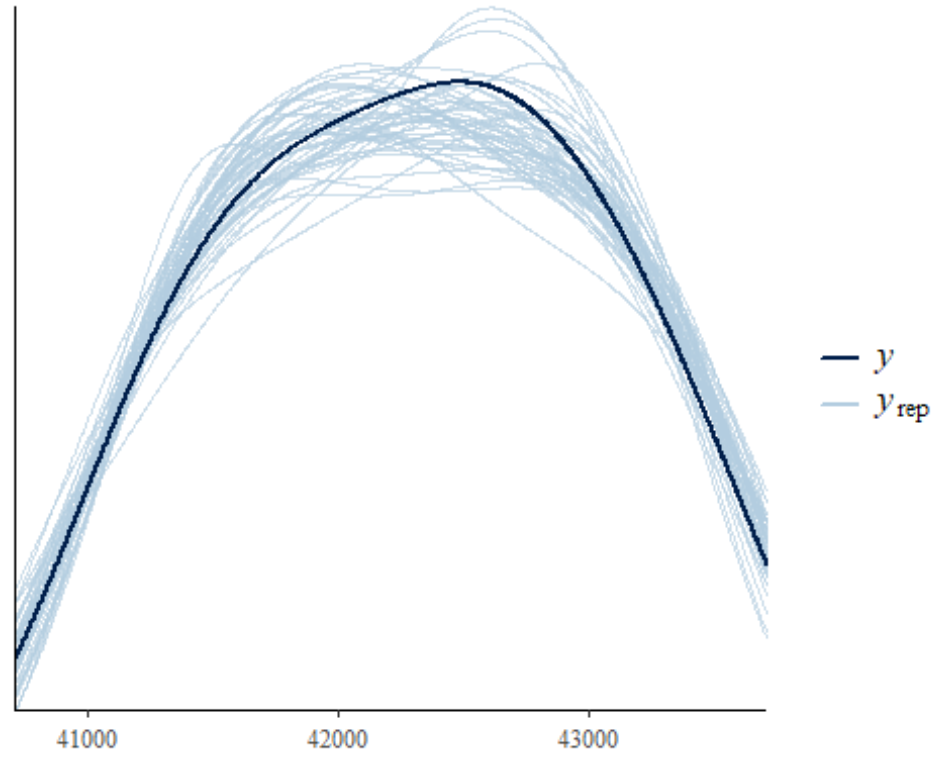
```
plot(sm4) # Graphically display the 95% credible intervals
```




```r
posterior_vs_prior(sm4) # Juxtaposing the prior and the posterior estimates
```




```r
pp_check(sm4) # Graphically display the posterior predictive analysis
```




```
waic(sm4) # Watanabe-Akaike information criterion

##
## Computed from 4000 by 11 log-likelihood matrix
##
##           Estimate  SE
## elpd_waic    -61.0 1.4
## p_waic         2.1 0.4
## waic         122.0 2.8
##
## 1 (9.1%) p_waic estimates greater than 0.4. We recommend trying loo
instead.

loo(sm4) # Leave One Out Cross-Validation

##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.1 1.5
## p_loo         2.3 0.4
## looic       122.2 2.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
```

```
## (-Inf, 0.5]   (good)     9     81.8%   1584
##  (0.5, 0.7]   (ok)       2     18.2%   844
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
loo(sm4, k_threshold = 0.7) # Use this alternative if you get a warning in
the above line
```

```
##
## Computed from 4000 by 11 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo    -61.1 1.5
## p_loo         2.3 0.4
## looic       122.2 2.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     9     81.8%   1584
##  (0.5, 0.7]   (ok)       2     18.2%   844
##    (0.7, 1]   (bad)      0      0.0%   <NA>
##    (1, Inf)   (very bad) 0      0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

Loo output explanation

- From the above loo cross validation outputs, we observe that all the four GP models have pareto k estimates with k-threshold values of k<0.7 which are okay. -We therefore validate that our emulator is a good representation of the simulators.