Supporting Information: Implementation of Stochastic SIR Cosine Model

Rahul Subramanian

April 20, 2020

Set up baseline model

Load Non-POMP Libraries and Files

```
rm(list = ls())
source("load_libraries_essential.R")
source("rahul_theme.R")
```

Load POMP2

```
library(pomp)
```

```
## Warning: package 'pomp' was built under R version 3.5.2
```

```
## Welcome to pomp version 2!
## For information on upgrading your pomp version < 2 code, see the
## 'pomp version 2 upgrade guide' at https://kingaa.github.io/pomp/.
```

We consider fitting a simple SEIR spline model to monthly case counts of DENV1 incidence in the municipality of Rio de Janeiro from April 1,1986 to December 31, 1987. The data consist of monthly case counts that are reported each week and then aggregated by month. The dates correspond to notification dates, not date of disease onset. For example, if 535 cases were reported for April 1986, it means that 535 cases were observed between April 1st,1986-April 30th,1986.

Declare model name

```
full_model_name =
  "DENV1_SIR_Cosine_Model"
model_name = "A_7"
rds_index = 0
```

Load dengue case data

```
load(file ="../Down_Data/denguerj1986-1996.RData")
#head(dengue.ts)
```

Clean up data into correct time scale for POMP object

```
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 3.5.2
```

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:pomp':
##
##     time<-
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(pomp)
Rio_city_DENV1_clean = data.frame(Y = as.matrix(dengue.ts),
                                  Date = as.Date(as.yearmon(time(dengue.ts))))
Rio_city_DENV1_clean = filter(Rio_city_DENV1_clean, Date >= "1986-05-01")

head(Rio_city_DENV1_clean)
```

```
##      Y       Date
## 1 4927 1986-05-01
## 2 3781 1986-06-01
## 3 1378 1986-07-01
## 4  406 1986-08-01
## 5  163 1986-09-01
## 6   41 1986-10-01
```

```
add_a_month = Rio_city_DENV1_clean$Date %m+% months(1)
#add_a_week = Rio_city_DENV1_clean$Date %m+% weeks(1)
#last_day_of_month = add_a_month - 1
correct_date_in_days_since_Jan_1_1986 = add_a_month - as.Date("1986/01/01")


Rio_data_clean = data.frame(times = as.numeric(correct_date_in_days_since_Jan_1_1986),
                            Y = Rio_city_DENV1_clean$Y)
#Only use first two years of data (April  1, 1986 - December 1, 1987)
Rio_data_clean = filter(Rio_data_clean, times <= 365*2.50)
write.csv(Rio_data_clean,
     file = "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv", row.names = FALSE)
#head(Rio_data_clean)
```

Set $t0$

```
t0 = as.numeric(as.Date("1986/05/01") - as.Date("1986/01/01"))
```

Source Csnippets

```
knitr::read_chunk('Csnippet_SIR_cosine_model.R')
```

Define co-variate time range

```
#Start covariates 1 month before start of data
covar_start = 0

#End covariates 1 month after end of data
covar_end = max(Rio_data_clean$times) + 31

#Set covariate time step (Default is 1 hour, want it smaller than dt to be safe)
covar_dt = 1/24

covar_times = seq(from = covar_start,
                  to = covar_end,
                  by = covar_dt)
```

The SIR model has three states, Susceptible, Infected, and Recovered:

```
statenames = c("S", "I", "R" , "C", "N")
acumvarnames = c("C")
obsnames = c("Y")
```

Table 1: State Variables and Covariates

Term	Definition	Type
$S(t)$	Susceptible humans in city i	State Variable
$I(t)$	Infected humans in city i	State Variable
$R(t)$	Recovered humans in city i	State Variable
$C(t)$	Reported Human Cases	State Variable
$N(t)$	Human Population	State Variable

Parameters

The force of infection $\lambda(t)$ is a function of the infected immigration rate ϵ and overall transmission rate $\beta(t)$ which in turn is assumed to be a cosine function of time t with mean β_0, amplitude δ, frequency ω and phase ϕ which will be fit along with a gamma-distributed white noise parameter $\frac{d\Gamma}{dt}$. ω is fixed at an annual frequency ($\omega = \frac{2\pi}{365}$).

The white noise $\frac{d\Gamma}{dt}$ is drawn from a gamma distribution with intensity $\sigma = \sigma_{\mathrm{P}}$ and duration of Euler step $dt = \Delta$, where Δ is the simulation time step of two hours (or $\frac{1}{12}$ in units of days). The intensity parameter σ_{P} will be fit to the data.

Environmental Noise Intensity

The discretization of the Gamma-distributed environmental noise in the model has the form:

$$\Delta\Gamma \sim rgammawn(\mu = dt, \sigma = \sigma_{\mathrm{P}}) \tag{1}$$

Formally, this is equivalent to a draw from a Gamma-distribution with shape parameter $\alpha = \frac{\delta}{\sigma_{\mathrm{P}}^2}$ and scale parameter $\beta = \frac{1}{\sigma_{\mathrm{P}}^2}$. (Note that $\delta = \Delta t$, and for all of this sub-section β refers to the Gamma distribution scale parameter rather than the transmission rate function, which is referred to as $\beta(t)$).

$$\Delta\Gamma \sim \Gamma(\frac{\delta}{\sigma_{\mathrm{P}}^2}, \frac{1}{\sigma_{\mathrm{P}}^2}) \tag{2}$$

Population and Reporting

We started the model with the estimated resident population of the municipality of Rio de Janeiro in 1991 according to the 1991 census. This estimated population is $N = 5480768$. The estimate was obtained from the IBGE's "Censo Demographico- 1991-Rio de Janeiro". The full description of the document in the IBGE catalog is "Censo demográfico : 1991 : resultados do universo relativos as características da população e dos domicílios"

The document can be accessed at the following site on the IBGE catalog: https://biblioteca.ibge.gov.br/biblioteca-catalogo?id=782&view=detalhes

At that site, the name of the file (which can be downloaded) is:

cd_1991_n20_caracteristicas_populacao_domicilios_rj.pdf

In this document, the population estimate was found under Table 1.4: "População residente, por grupos de idade, segundo tU lolesorregiàes, as Microrregiões, os Municípios,os Distritos e o sexo"

The sub-section of the table (the sub-heading can be found on page 27 of the document (page 32 using the document's internal pagination)) was "Municipios e Distritos"

The population estimate for the municipality of Rio de Janeiro can be found on page 36 of that document (page 41 using internal pagination) under the row "Rio de Janeiro" and column heading "Total".

The population estimate again was $N = 5480768$.

We next obtained the estimated resident pouplation of the municipality of Rio de Janeiro in 2000 using the 2000 census from the IBGE website.

We obtained estimates of the resident population of the municipality of Rio de Janeiro in 2000 from the 200 Brazil census (specific table page: https://ww2.ibge.gov.br/home/estatistica/populacao/censo2000/universo.php?tipo=31o/tabela13_1.shtm&paginaatual=1&uf=33&letra=R).

Census website: https://ww2.ibge.gov.br/english/estatistica/populacao/censo2000/default.shtm

Heading Type: População residente, sexo e situação do domicílio; Total column

Estimated Population of Rio de Janeiro in 2000: 5,857,904

Estimated Population of Rio de Janeiro in 2010 (for reference): 6,320,446 (based on the 2010 population estimate of the municipality of Rio de Janeiro from Table 1378 of the 2010 Brazilian census (accessed at https://sidra.ibge.gov.br/tabela/1378 ; original website https://sidra.ibge.gov.br/pesquisa/censo-demografico/demografico-2010/universo-caracteristicas-da-populacao-e-dos-domicilios))

We calculate the rate of human population growth from 1991 to 2000 assuming exponential growth. We will then use this rate to back-calculate an estimate of the municipal resident population size in 1986 (again assuming exponential population growth).

Pop growth rate calculation

```
Population_Rio_2000 = 5857904 #Census
Population_Rio_1991 = 5480768# Census:
Two_hour_segments_in_year = 365 * 12
time_between_census_dates = 2000*365 - 1991*365
human_pop_growth_rate = (1 / time_between_census_dates) *
  log(Population_Rio_2000 / Population_Rio_1991)
human_pop_growth_rate
```

```
## [1] 2.025772e-05
```

Back-calculation of 1986 Population

```
time_before_1991_census_dates = 1991*365 - 1986*365
Population_Rio_1986 = Population_Rio_1991/
  (exp(human_pop_growth_rate*time_before_1991_census_dates))
```

Thus, the estimated pouplation of Rio de Janeiro is approximately $N_0 = 5281842$

This version of the model assumes a constant population size with demographic turnover μ given by the inverse of the life expectancy of Brazil in 2012 (74.49 years https://censo2010.ibge.gov.br/en/noticias-censo.html?busca=1&id=1&idnoticia=2528&t=life-expectancy-at-birth-was-74-6-years-in-2012&view=noticia).

A fraction ρ of newly infected cases are reported and enter the reported case category C. We will be fitting the reporting rate.

The observed monthly cases are assumed to have a negative binomial distribution with mean equal to the true number of monthly cases and size parameter equal to $\frac{1}{\sigma_M^2}$, where σ_M will be fit to the data.

We assume a duration of infection ($\frac{1}{\gamma}$) of 10.25 days. Dengue is believed to have a symptomatic period of 2-7 days following an incubation period of 4-7 days, which we have combined in our model into a single infectious period. (http://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue)

The model framework contains the parameterization necessary to incorporate population growth.

Let r represent the per capita rate at which new individuals enter the population, while μ_{H} is the death rate. Let h represent the per capita growth rate of the population (i.e. $h = r - \mu_{\mathrm{H}}$). We assume that the net population growth rate is exponential:

$$\frac{dN}{dt} = hN(t) \tag{3}$$

Let r represent the overall growth rate of the susceptible population taking into account both population growth and population turnover, where

$$r = h + \mu_{\mathrm{H}} \tag{4}$$

at rate h.

Population growth would then occur at rate:

In this instance, we assume that $h = 0$.

Process Model

ODE Equations with only Environmental Noise

$$\beta(t) = \beta_0(1 + \delta sin(\omega t + \phi)); \tag{5}$$

$$\frac{d\Gamma}{dt} \sim \ rgammawn(\sigma_{\mathrm{P}}, \Delta t) \tag{6}$$

$$\lambda(t) = \beta(t)(\frac{I(t) + \epsilon}{N})\frac{d\gamma}{dt} \tag{7}$$

$$\frac{dS}{dt} = rN + -\lambda(t)S(t) - \mu_{\mathrm{H}}S(t) \tag{8}$$

$$\frac{dI}{dt} = \lambda(t)S(t) - \gamma I(t) - \mu_{\mathrm{H}}I(t) \tag{9}$$

$$\frac{dR}{dt} = \gamma I(t) - \mu_{\mathrm{H}}R(t) \tag{10}$$

Cases C are summed over each month.

The expression for $\frac{dS}{dt}$ can be further specified into separate terms for the net population growth and replacement of deaths.

$$\frac{dS}{dt} = hN + \mu_{\mathrm{H}}N + -\lambda(t)S(t) - \mu_{\mathrm{H}}S(t) \tag{11}$$

Equations for model with demographic and environmental noise

Rates in continuous time:

$$\mu_{SI}(t) = \beta(\frac{I(t) + \epsilon}{N(t)}) \tag{12}$$

$$\mu_{IR}(t) = \gamma \tag{13}$$

Let $\mu_{\cdot N}$ represent the rate of net population growth.

$$\mu_{\cdot N} = h \tag{14}$$

Let $\mu_{\cdot S}$ represent the rate at which individuals who die are replaced by susceptible individuals. We assume that this replacement rate is equivalent to the death rate.

$$\mu_{\cdot S} = \mu_H \tag{15}$$

$$\mu_{S\cdot} = \mu_{I\cdot} = \mu_{R\cdot} = \mu_H \tag{16}$$

Discretizations

Discretization of population growth

$$\Delta\tilde{N}_{\cdot N} \sim Binomial(\tilde{N}(t), 1 - e^{-\tilde{\mu}_{\cdot N}\Delta t}) \tag{17}$$

Discretization of Gamma white noise from time t to $t + \Delta t$

$$\Delta\Gamma \sim \ rgammawn(\sigma_{\mathrm{P}}, \Delta t) \tag{18}$$

Discretization of force of infection from time t to $t + \Delta t$

$$\tilde{\lambda}(t) = \mu_{SI}(t)\Delta\Gamma \tag{19}$$

Discretization of compartment flows from time t to time $t + \Delta t$

$$\Delta\tilde{N}_{SI} \sim Binomial(\tilde{S}(t), 1 - e^{-\tilde{\lambda}(t)}) \tag{20}$$

$$\Delta\tilde{N}_{IR} \sim Binomial(\tilde{I}(t), 1 - e^{-\tilde{\mu}_{IR}(t)\Delta t}) \tag{21}$$

$$\Delta\tilde{N}_{\cdot S} \sim Binomial(\tilde{N}(t), 1 - e^{-\tilde{\mu}_{\cdot S}(t)\Delta t}) \tag{22}$$

$$\Delta\tilde{N}_{S\cdot} \sim Binomial(\tilde{S}(t), 1 - e^{-\tilde{\mu}_{S\cdot}(t)\Delta t}) \tag{23}$$

$$\Delta\tilde{N}_{I\cdot} \sim Binomial(\tilde{I}(t), 1 - e^{-\tilde{\mu}_{I\cdot}(t)\Delta t}) \tag{24}$$

$$\Delta\tilde{N}_{R\cdot} \sim Binomial(\tilde{R}(t), 1 - e^{-\tilde{\mu}_{R\cdot}(t)\Delta t}) \tag{25}$$

$$\Delta\tilde{S} = \Delta\tilde{N}_{\cdot N} + \Delta\tilde{N}_{\cdot S} - \Delta\tilde{N}_{SI} - \Delta\tilde{N}_{S\cdot} \tag{26}$$

$$\Delta\tilde{I} = \Delta\tilde{N}_{SI} - \Delta\tilde{N}_{IR} - \Delta\tilde{N}_{I\cdot} \tag{27}$$

$$\Delta\tilde{R} = \Delta\tilde{N}_{IR} - \Delta\tilde{N}_{R\cdot} \tag{28}$$

$$\Delta\tilde{N} = \Delta\tilde{N}_{\cdot N} + \Delta\tilde{N}_{\cdot S} - \Delta\tilde{N}_{S\cdot} - \Delta\tilde{N}_{I\cdot} - \Delta\tilde{N}_{R\cdot} \tag{29}$$

We note several notation differences between the written equations and the R implementation. First, the variable described as h in the write-up is instead denoted by r. The variable r in the write-up is not explicitly referred to in the code in this document. Secondly, the variable $\Delta\tilde{N}_{\cdot N}$ is written as dBS_N in the code while the variable $\Delta\tilde{N}_{\cdot S}$ in the write-up is written as dBS. Finally, in the written implementation, the reporting rate is multiplied by the true cases C in the measurement model. In the R Code, ρ is multiplied by C as C is calculated in the process model Csnippet, instead of being multiplied in the measurement model. This does not change the results of the calculation.

```
#Process model Csnippet
rproc <- Csnippet("


                  if(R < 0 || I < 0 ||  N < 0){
                    Rprintf(\"Neg state var det at  t = %lg \\n\", t);
                    Rprintf(\"I = %lg \\n\", I);
                    Rprintf(\"R = %lg \\n\", R);
                    Rprintf(\"N = %lg \\n\", N);
                    Rprintf(\"S = %lg \\n\", S);

                  }

                  if(isnan(R) || isnan(I) || isnan(N) || isnan(S)){
                    Rprintf(\"nan state var det at top of proc model t = %lg \\n\", t);
                    Rprintf(\"I = %lg \\n\", I);
                    Rprintf(\"R = %lg \\n\", R);
                    Rprintf(\"N = %lg \\n\", N);
                    Rprintf(\"S = %lg \\n\", S);

                  }

                  double beta = Beta_0*(1 + delta*sin(omega*t + phi));
                  double dW = rgammawn(sigma_P,dt);
                  double lambda = beta*((I+ epsilon)/N)*dW;
                  //Rprintf(\"start proc t = %lg \\n\", t);
                  //Rprintf(\"beta = %lg \\n\", beta);
                  //Rprintf(\"dW = %lg \\n\", dW);

                  //Rprintf(\"lambda = %lg \\n\", lambda);

                  //Rprintf(\"S = %lg \\n\", S);
```

```
// Rprintf(\"I = %lg \\n\", I);
//Rprintf(\"C = %lg \\n\", C);
// Rprintf(\"rho = %lg \\n\", rho);

double dSI = rbinom(S, 1 - exp(-lambda));

double dIR = rbinom(I, 1 - exp(-gamma*dt));
double dBS = rbinom(N, 1 - exp(-mu_H*dt));

//Add population growth
double dBS_N = rbinom(N, 1 - exp(-r*dt));
//if(t < 10){
//Rprintf(\"r = %lg \\n\", r);
//Rprintf(\"N = %lg \\n\", N);
//Rprintf(\"t = %lg \\n\", t);
//Rprintf(\"dBS_N = %lg \\n\", dBS_N);
//}


//Transition increments
S += dBS  + dBS_N - dSI;
I += dSI - dIR;
R += dIR;
N += dBS + dBS_N;

double dSM = rbinom(S, 1 - exp(-mu_H*dt));
double dIM = rbinom(I, 1 - exp(-mu_H*dt));
double dRM = rbinom(R, 1 - exp(-mu_H*dt));
S += - dSM;
I +=  - dIM;
R += - dRM;
N +=  - dSM - dIM - dRM;


//Rprintf(\"dSI = %lg \\n\", dSI);

//Rprintf(\"dIR = %lg \\n\", dIR);



C += rho*dSI;
if(C < 0 || S < 0 || I < 0 || R < 0 ){
Rprintf(\"Neg value at t = %lg \\n\", t);
//    Rprintf(\"beta = %lg \\n\", beta);
//    Rprintf(\"dSI = %lg \\n\", dSI);
Rprintf(\"S = %lg \\n\", S);
Rprintf(\"I = %lg \\n\", I);

Rprintf(\"I = %lg \\n\", I);
//    Rprintf(\"dSI = %lg \\n\", dSI);
//    Rprintf(\"dIR = %lg \\n\", dIR);
//    Rprintf(\"C = %lg \\n\", C);
//    Rprintf(\"rho = %lg \\n\", rho);
```

```
                  I = 0;

                  }

                  if(isnan(R) || isnan(I) ||isnan(N) || isnan(S)){
                    Rprintf(\"nan state var det at bot of proc model t = %lg \\n\", t);
                    Rprintf(\"I = %lg \\n\", I);
                    Rprintf(\"R = %lg \\n\", R);
                    Rprintf(\"N = %lg \\n\", N);
                    Rprintf(\"S = %lg \\n\", S);
                    Rprintf(\"lambda = %lg \\n\", lambda);
                    Rprintf(\"Beta_0 = %lg \\n\", Beta_0);
                    Rprintf(\"delta = %lg \\n\", delta);
                    Rprintf(\"phi = %lg \\n\", phi);
                    Rprintf(\"rho = %lg \\n\", rho);
                    Rprintf(\"I_0 = %lg \\n\", I_0);
                    Rprintf(\"R_0 = %lg \\n\", R_0);

                                    }

                  ")
```

Measurement Model

$$Y \sim \ NBinom(size = 1/(\sigma_{\mathrm{M}})^2, \mu = \rho C) \tag{30}$$

```
rmeas <- Csnippet("
                  double size = 1.0/sigma_M/sigma_M;
                  Y = rnbinom_mu(size,C);
                  ")
```

```
dmeas <- Csnippet("
                  double size = 1.0/sigma_M/sigma_M;
                  static double tol = 0.1;
                  lik = dnbinom_mu(Y,size,C+tol,1);

                  double total_0 = round(I_0) + round(R_0);
                  if(total_0 > round(N_0)){
                    lik = -40;

                  }

                 if(R_0 < 0 || I_0 < 0 || N_0 < 0){
                     lik = -40;

                  }

                  //Debugging Print Code
                  //Rprintf(\"t = %lg \\n\", t);
                  //Rprintf(\"I = %lg \\n\", I);

                  //Rprintf(\"Lik = %lg \\n\", lik);
```

Term	Definition	Value	Units
C_0	Monthly reported cases at start of human invasion	0 (Ignored)	person
I_0	Infected people at start of human invasion	Fit	person
S_0	Susceptible people at start of human invasion in city i	$N - I_{\text{Init}}$	person
R_0	Recovered people at start of human sim in city i	Fit	person

Table 2: Initial Conditions.

```
                  //Rprintf(\"Y = %lg \\n\", Y);
                  //Rprintf(\"C = %lg \\n\", C);
                  //Rprintf(\"tol = %lg \\n\", tol);
                  //Rprintf(\"size = %lg \\n\", size);


                  if (!give_log) lik = exp(lik);
                  ")
```

Initial Conditions

We assume that a small fraction of the population I_0 starts out infected, but that everyone else is susceptible at the start of the DENV1 invasion.

```
init <- Csnippet("
                 //Rprintf(\"At init N_0 = %lg \\n\", N_0);
                 //Rprintf(\"At init rho = %lg \\n\", rho);
                 double total_0 = round(I_0) + round(R_0);
                 if(total_0 > round(N_0)){
                  S = 0;
                  I = 0;
                  R = round(N_0);

                 }
                 if(I_0 > N_0){
                 I = round(N_0);
                 S = 0;
                 N = round(N_0);
                 R = 0;
                 }else{

                 if(R_0 > N_0){
                 I = 0;
                 S = 0;
                 R = round(N_0);
                 }else{
                    if(R_0 < 0 || I_0 < 0 || N_0 < 0){
                      I = 0;
                      N = 1;
                      R = 0;
                      S = 1;
                    } else{
                        I = round(I_0);
                        N = round(N_0);
                        R = round(R_0);
```

```
                        S = round(N_0)-round(I_0) - round(R_0);
                    }
                }

                }



                C = C_0;
                //Rprintf(\"At init N = %lg \\n\", N);
                //Rprintf(\"At init I = %lg \\n\", I);
                //Rprintf(\"At init C = %lg \\n\", C);

                ")
```

Parameter Transforms

```
par_trans = parameter_trans(log = c("mu_H", "Beta_0",
                                    "gamma", "sigma_P",
                                    "sigma_M", "r",
                                    "epsilon"),
                            logit = c("rho", "delta"))
```

Covariates

```
covar=covariate_table(
  t=covar_times,
  s=periodic.bspline.basis(t,nbasis=3,degree=3,period=365, name='%d'),
  times="t"
)
```

MIF Function Call from Parallelized Midway Script

Function to run single MIF run for given number of iterations followed by 10 Pfilter runs from final MIF values

```
get_MIF_final_params_and_pfilter_LL = function(data,
                                               times,
                                               t0,
                                               rproc,
                                               params,
                                               paramnames,
                                               statenames,
                                               obsnames,
                                               dmeas,
                                               accumvars,
                                               init,
                                               rmeas,
                                               par_trans,
                                               Np,
```

```
                                                Nmif ,
                                                cooling.fraction.50,
                                                rw.sd ,
                                                delta_time,
                                                param_index,
                                                i,
                                                detail_log = FALSE,
                                                covar) {

log_str = ""
if(detail_log == TRUE){
  log_str = paste0(log_str,
                   "subset:", param_index,
                   " comb: ", i,
                   " starting_param_guess: ", names(params)," = " ,params,"\n")
}


seed <- round(runif(1,min=1,max=2^30))
#Compute MIF calculation
mf <- tryCatch(
  mif2(
    data = data,
    times = times,
    t0 = t0,
    seed = seed,
    rprocess = pomp2::euler(rproc, delta.t = delta_time),
    params = params,
    paramnames = paramnames,
    statenames = statenames,
    obsnames = obsnames,
    dmeas = dmeas,
    accumvars = accumvars,
    covar=covar,
    rinit = init,
    rmeas = rmeas,
    partrans = par_trans,
    start = params,
    Np = Np,
    Nmif = Nmif,
    cooling.fraction.50 = cooling.fraction.50,
    rw.sd = rw.sd
  ),
  error = function(e) e
)
MIF_single_param_output = params
MIF_single_param_output$LL = NA
if(detail_log == TRUE){
  log_str = paste0(log_str, "mif warnings: \n ",
                   warnings(),
                   " \n Done with warnings \n")
}
```

```
if(!inherits(mf, "error")){
  if(length(coef(mf)) > 0){
    print(mf)
    if(detail_log == TRUE){
      log_str = paste0(log_str, "subset:", param_index,
                       " comb: ", i,
                       " mif_end_guess: ", names(params)," = " ,coef(mf),"\n")

      log_str = paste0(log_str, "subset:", param_index,
                       " comb: ", i,
                       " mif_nfail: ", mf@nfail," mif_ess: " ,
                       eff.sample.size(mf),
                       " MIF Log Lik: ", logLik(mf),"\n")
    }
    MIF_single_param_output = as.data.frame(t(coef(mf)))
    ll <- tryCatch(
      replicate(n=10,logLik(pfilter(
        data = data,
        times = times,
        t0 = t0,
        rprocess = pomp2::euler(rproc,delta.t = delta_time),
        paramnames = paramnames,
        statenames = statenames,
        obsnames = obsnames,
        dmeas = dmeas,
        accumvars = accumvars,
        covar = covar,
        rinit = init,
        rmeas = rmeas,
        partrans = par_trans,
        format = "data.frame",
        Np=Np,
        params=coef(mf)))),
      error = function(e) e
    )
    if(is(ll,"error")) {}else{
      ll <- logmeanexp(ll)
      if(detail_log == TRUE){
        log_str = paste0(log_str, "pfilter_warnings: \n ",
                         warnings(),
                         " \n Done with warnings \n")
      }

    }
    if(is.na(ll)) {}else{
      MIF_single_param_output$LL = ll
    }
  }
}
#return_list = list(MIF_single_param_output, mf)
#return(return_list)
if(detail_log == TRUE){
  log_str = paste0(log_str, "subset:", param_index,
```

```
                      " comb: ", i,
                      " end_param_guess: ", colnames(MIF_single_param_output)," = " ,
                      MIF_single_param_output,"\n")
  }
  log_str_collapsed = paste0(log_str, collapse = " ")

  MIF_single_param_output$seed = seed
  MIF_single_param_output$Log = log_str_collapsed
  return(MIF_single_param_output)
}
```

Generate profiles

I generated profiles for eleven model parameters: β_0, δ, E_0, I_0, ρ, μ_{EI}, γ ϕ, N_0, σ_{M}, and σ_{P}.

Generate set of parameter combinations for profiles

The profileDesign function was used to generate a set of starting points at 30 different evenly spaced values for the parameter being profiled. For each profile parameter value, the function created 40 different initial sampling points drawing from a box given by the boundaries of the original parameter range defined in the beginning. For example, for the I_0 profile, a set of 30 starting points evenly spaced between 1 and 10,000 was generated. For each of those 30 starting points, the profileDesign function created 40 different initial sampling points with the same value of I_0 but different values for the other parameters being fitted (β_0, δ, E_0, μ_{EI}, γ ϕ, , σ_{M}, and σ_{P}) where the different values were uniformly drawn from the boundaries for those parameters in the original box. This yielded a total of 1200 starting points for each parameter profile.

```
knitr::read_chunk('generate_profile_combinations_SIR_Cosine.R')
```

```
# Header ----------------------------------------------------------------------
## Name: generate_profile_combinations_SIR_Cosine.R
## Author: Rahul Subramanian
## Description: Creates 30*40-combination list for given by
## profile_var as 1st command line argument
rm(list = ls())

ptm <- proc.time()

#Load Libraries
source("load_libraries_essential.R")
source("rahul_theme.R")
library(pomp2)

#profile_var = "I_0"
#model_name = "SEIR_Spline_2_Year"

args = commandArgs(trailingOnly=TRUE)

profile_var = as.character(args[1])
print(profile_var)

model_name = as.character(args[2])
print(model_name)
```

```
city_name = as.character(args[3])

serotype_name = as.character(args[4])

R_Init_status = as.character(args[5])

Immigration_status = as.character(args[6])

Duration_status = as.character(args[7])

city_specific_param_boundaries = data.frame(City = c("Rio", "Rio", "Fortaleza",
                                                     "Rio","Rio", "Rio",
                                                     "Rio", "Rio","Rio"),
                                            Serotype = c("DENV1", "DENV4",
                                                         "DENV4", "DENV1",
                                                         "DENV1",  "DENV1",
                                                         "DENV1", "DENV1",
                                                         "DENV1"),
                                            R_Init_Status = c("Fix_R_Init",
                                                              "Fix_R_Init",
                                                              "Fix_R_Init",
                                                              "Fit_R_Init",
                                                              "Fit_R_Init",
                                                              "Fit_R_Init",
                                                              "Fix_R_Init",
                                                              "Fix_R_Init",
                                                              "Fix_R_Init"),
                                            Immigration = c("No_Immigration",
                                                            "No_Immigration",
                                                            "No_Immigration",
                                                            "No_Immigration",
                                                            "Immigration",
                                                            "Immigration",
                                                            "Immigration",
                                                            "No_Immigration",
                                                            "No_Immigration"),
                                            Duration_Params = c("Fit_Duration",
                                                                "Fit_Duration",
                                                                "Fit_Duration",
                                                                "Fit_Duration",
                                                                "Fit_Duration",
                                                                "Fix_Duration",
                                                                "Fix_Duration",
                                                                "Fix_Duration",
                                                                "Profile_Duration"),
                                            rho_upper = c(0.001, 0.15, 0.001,
                                                          0.001,0.001,  0.001,
                                                          0.001,  0.001,0.001),
                                            rho_lower = c(0.15, 0.15, 0.15, 0.15,
                                                          0.15, 0.15, 0.15, 0.15,
                                                          0.15),
                                            N_0_upper = c(5.301405e+06,
                                                          6.320446e+06,
```

```
                2.452185e+06,
                5.301405e+06,
                5.301405e+06,
                5.301405e+06,
                5.301405e+06,
                5.281842e+06,
                5.281842e+06),
N_0_lower = c(5.301405e+06,
                6.320446e+06,
                2.452185e+06,
                5.301405e+06,
                5.301405e+06,
                5.301405e+06,
                5.301405e+06,
                5.281842e+06,
                5.281842e+06),
R_0_lower = c(0, 0, 0, 0,
                0, 0, 0, 0,
                0),
R_0_upper = c(0, 0, 0, 5.101405e+06,
                5.101405e+06,
                5.101405e+06, 0, 0,
                0),
Beta_0_lower = c(-3, -2, -4, -3,
                  -5.5, -3,-3,0,
                  0),
Beta_0_upper = c(5, 1.75, 2, 7.5,
                  7.5, 7.5, 7.5, 0.25,
                  0.25),
delta_lower = c(-6, -4.5, -3.5, -8,
                  -7.5, -8, -8, 0,
                  0),
delta_upper = c(3, 0, 1.5, 3,
                  5, 5, 5, 1,
                  1),
phi_lower = c(-15, -8.0, -7.5, -18,
                -15, -18, -18, 0,
                0),
phi_upper = c(6, 0.5, 1.25, 6,
                6, 6, 6, pi,
                pi),
omega_lower = c(0, 0, 0, 0,
                  -4, 0, 0, (2*pi)/365,
                  (2*pi)/365),
omega_upper = c(0, 0, 0, 0,
                  4, 0, 0, (2*pi)/365,
                  (2*pi)/365),
epsilon_lower = c(0, 0, 0, 0,
                    0, 0, 0, 0,
                    0),
epsilon_upper = c(0, 0, 0, 0,
                    0.2, 0.2, 0.2, 0,
                    0),
```

```
                                        I_0_lower = c(1, 1, 1, 1,
                                                      1, 1, 1, 1,
                                                      1),
                                        I_0_upper = c(1.000000e+07,
                                                      2.000000e+05,
                                                      2.000000e+05,
                                                      1.000000e+06,
                                                      1.000000e+06,
                                                      6.000000e+05,
                                                      6.000000e+05,
                                                      6.000000e+05,
                                                      6.000000e+05),
                                        sigma_M_lower= c(.001, .001,
                                                         .001, .001,
                                                        0, .0001,
                                                        .0001, .0001,
                                                        .0001),
                                        sigma_M_upper = c(1, .25, .25, 1,
                                                          1, 1, 1, 1,
                                                          1),
                                        gamma_lower = c(1/17, 1/17,
                                                        1/17, 1/17,
                                                        1/17, 1/17,
                                                        1/17, 1/17,
                                                        1/22),
                                        gamma_upper = c(1/4, 1/4,
                                                        1/4, 1/4,
                                                        1/4, 1/17,
                                                        1/17, 1/17,
                                                        1/2),
                                        sigma_P_lower = c(1.9e-4, 1.9e-4,
                                                          1.9e-4, 1.9e-4,
                                                          1.9e-4, 1.9e-4,
                                                          1.9e-4, 1.9e-4,
                                                          1.9e-4),
                                        sigma_P_upper = c(3.8e1, 3.8e1,
                                                          3.8e1, 3.8e1,
                                                          3.8e1, 3.8e1,
                                                          1, 1,
                                                          1))

city_specific_param_boundaries = filter(city_specific_param_boundaries,
                                        City == city_name)
city_specific_param_boundaries = filter(city_specific_param_boundaries,
                                        Serotype == serotype_name)
city_specific_param_boundaries = filter(city_specific_param_boundaries,
                                        R_Init_Status == R_Init_status)
city_specific_param_boundaries = filter(city_specific_param_boundaries,
                                        Immigration == Immigration_status)
city_specific_param_boundaries = filter(city_specific_param_boundaries,
                                        Duration_Params == Duration_status)

rho_upper = city_specific_param_boundaries$rho_upper
```

```
rho_lower = city_specific_param_boundaries$rho_lower
N_0_upper = city_specific_param_boundaries$N_0_upper
N_0_lower = city_specific_param_boundaries$N_0_lower
R_0_lower = city_specific_param_boundaries$R_0_lower
R_0_upper = city_specific_param_boundaries$R_0_upper

Beta_0_lower = city_specific_param_boundaries$Beta_0_lower
Beta_0_upper = city_specific_param_boundaries$Beta_0_upper
delta_lower = city_specific_param_boundaries$delta_lower
delta_upper = city_specific_param_boundaries$delta_upper
phi_lower = city_specific_param_boundaries$phi_lower
phi_upper = city_specific_param_boundaries$phi_upper
omega_lower = city_specific_param_boundaries$omega_lower
omega_upper = city_specific_param_boundaries$omega_upper
epsilon_lower = city_specific_param_boundaries$epsilon_lower
epsilon_upper = city_specific_param_boundaries$epsilon_upper

I_0_lower = city_specific_param_boundaries$I_0_lower
I_0_upper = city_specific_param_boundaries$I_0_upper

sigma_M_lower = city_specific_param_boundaries$sigma_M_lower
sigma_M_upper = city_specific_param_boundaries$sigma_M_upper

gamma_lower = city_specific_param_boundaries$gamma_lower
gamma_upper = city_specific_param_boundaries$gamma_upper

sigma_P_lower = city_specific_param_boundaries$sigma_P_lower
sigma_P_upper = city_specific_param_boundaries$sigma_P_upper



par_box_boundaries = rbind(
  c(gamma_lower, gamma_upper), # gamma
  c(phi_lower,phi_upper), # phi
  c(sigma_P_lower, sigma_P_upper), # sigma_P
  c(sigma_M_lower,sigma_M_upper), # sigma_M
  c(rho_lower,rho_upper), # rho
  c(Beta_0_lower,Beta_0_upper), # Beta_0
  c(delta_lower, delta_upper), # delta
  c(3.680000e-05,3.680000e-05), # mu_H
  c(N_0_lower,N_0_upper), # N_0
  c(I_0_lower,I_0_upper), # I_0
  c(R_0_lower,R_0_upper), # R_0
  c(0,0), #C_0
  c(0,0), #r
  c(omega_lower, omega_upper), #omega
  c(epsilon_lower, epsilon_upper) #epsilon
)

par_box_boundaries = t(par_box_boundaries)
names <- c("gamma","phi","sigma_P","sigma_M","rho","Beta_0","delta",
           "mu_H","N_0","I_0","R_0","C_0", "r", "omega", "epsilon")
colnames(par_box_boundaries) = names
```

```
par_box_boundaries = as.data.frame(par_box_boundaries)
par_box_boundaries_clean = dplyr::select(par_box_boundaries,
                                         -one_of(profile_var) )
theta.t.lo = as.numeric(as.vector(par_box_boundaries_clean[1,]))
theta.t.hi = as.numeric(as.vector(par_box_boundaries_clean[2,]))
names(theta.t.lo) = colnames(par_box_boundaries_clean)
names(theta.t.hi) = colnames(par_box_boundaries_clean)

prof_var_boundaries = dplyr::select(par_box_boundaries, one_of(profile_var))
profileDesign(
  prof_var=seq(from=prof_var_boundaries[1,],
               to=prof_var_boundaries[2,],length=30),
  lower=theta.t.lo,upper=theta.t.hi,nprof=40
) -> pd
pd_col = colnames(pd)
colnames(pd) = c(profile_var, pd_col[2:length(pd_col)])


write.csv(pd, file = paste0("../Generated_Data/Profile_Combination_Lists/",
                            model_name,"_Model/", profile_var,"_",
                            model_name,
                            "_profile_combination_list.csv"),
     append = FALSE, row.names = FALSE)
proc.time() - ptm
```

Midway code for running MIF on each subset of parameter combinations

For each of those 1200 starting points, MIF was run 10 times.

Since this is a large number of iterations, two different parallelization strategies were employed at once on the University of Chicago's Research Computing Center's Midway cluster. First, multiple cores (28) were requested per job and a foreach loop was used to parallelize a single job between multiple cores on the cluster. However, if a large amount of cores are requested for a job, the Midway scheduler will wait until sufficient resources are available on the cluster, which can create a long lag time. To remedy this, the overall job was split into 50 array jobs (the maximum number of jobs that can be submitted to or running on the Midway cluster at any point in time). Each of those 50 array jobs in turn was parallelized over 28 cores.

```
knitr::read_chunk('MIF_run_Model_A_7.R')
```

The R code below was run on Midway for each of 50 array jobs.

```
# Header ----------------------------------------------------------------
## Name: MIF_run_Model_A_7.R
## Author: Rahul Subramanian
## Description: Runs parameter combinations
## on midway for profile from original param grid
## for SIR model with cosine function (Model A_7)

rm(list = ls())
ptm <- proc.time()

#Load Libraries
source("load_libraries_essential.R")
source("rahul_theme.R")
```

```
library(pomp2)

args = commandArgs(trailingOnly = TRUE)
#param_index = as.numeric(args[1]) +
# as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID"))

profile_var = as.character(args[1])
print(profile_var)

model_name = as.character(args[2])
print(model_name)


#Load dengue case data
Rio_data_clean = read.csv(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")
head(Rio_data_clean)
t0 = as.numeric(as.Date("1986/05/01") - as.Date("1986/01/01"))


#Declare Csnippets and data
source("Csnippet_SIR_cosine_model.R")


require(foreach)
require(doParallel)
require(deSolve)

#Core management
no_cores <- detectCores()
cat("no_cores = ", no_cores, "\n")
cl <- makeCluster(no_cores)
registerDoParallel(cl)


param_index = as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID"))
print("param_index")
print(param_index)


##load(param_grid)
pd = read.csv(
  file = paste0(
    "../Generated_Data/Profile_Combination_Lists/",
    model_name,
    "_Model/",
    profile_var,
    "_",
    model_name,
    "_profile_combination_list.csv"
  ),
  header = TRUE
)
```

```
head(pd)

midway_max_jobs = 50
group_size = nrow(pd) / midway_max_jobs
start_index = (param_index - 1) * group_size + 1
end_index = param_index * group_size
Num_mif_runs_per_start = 5
param_data_subset_act = pd[start_index:end_index, ]
param_data_subset =
  param_data_subset_act[rep(seq_len(nrow(param_data_subset_act)),
                            each = Num_mif_runs_per_start), ]

rw_sd_list_default = rw.sd(
  Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
  delta = ifelse(time <= 365 * 2.50, 0.02, 0),
  phi = ifelse(time <= 365 * 2.50, 0.02, 0),
  sigma_P = 0,
  sigma_M = 0.02,
  I_0 = ivp(0.2),
  R_0 = 0,
  epsilon = 0)




get_rwsd = function(profile_var) {
  if (profile_var == "I_0") {
    rw.sd = rw.sd(
      Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
      delta = ifelse(time <= 365 * 2.50, 0.02, 0),
      phi = ifelse(time <= 365 * 2.50, 0.02, 0),
      rho = 0.02,
      sigma_P = 0,
      sigma_M = 0.02,
      I_0 = ivp(0),
      R_0 = 0,
      epsilon = 0
    )
  } else{
    if (profile_var == "Beta_0") {
      rw.sd = rw.sd(
        Beta_0 = 0,
        delta = ifelse(time <= 365 * 2.50, 0.02, 0),
        phi = ifelse(time <= 365 * 2.50, 0.02, 0),
        rho = 0.02,
        sigma_P = 0,
        sigma_M = 0.02,
        I_0 = ivp(0.2),
        R_0 = 0,
        epsilon = 0
      )
    } else{
```

```
    if (profile_var == "delta") {
      rw.sd = rw.sd(
        Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
        delta = 0,
        phi = ifelse(time <= 365 * 2.50, 0.02, 0),
        rho = 0.02,
        sigma_P = 0,
        sigma_M = 0.02,
        I_0 = ivp(0.2),
        R_0 = 0,
        epsilon = 0
      )
    } else{
      if (profile_var == "phi") {
        rw.sd = rw.sd(
          Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
          delta = ifelse(time <= 365 * 2.50, 0.02, 0),
          phi = 0,
          rho = 0.02,
          sigma_P = 0,
          sigma_M = 0.02,
          I_0 = ivp(0.2),
          R_0 = 0,
          epsilon = 0
        )
      } else{
        if (profile_var == "rho") {
          rw.sd = rw.sd(
            Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
            delta = ifelse(time <= 365 * 2.50, 0.02, 0),
            phi = ifelse(time <= 365 * 2.50, 0.02, 0),
            rho = 0,
            sigma_P = 0,
            sigma_M = 0.02,
            I_0 = ivp(0.2),
            R_0 = 0,
            epsilon = 0
          )
        } else{
            if (profile_var == "sigma_P") {
              rw.sd = rw.sd(
                Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
                delta = ifelse(time <= 365 * 2.50, 0.02, 0),
                phi = ifelse(time <= 365 * 2.50, 0.02, 0),
                rho = 0.02,
                sigma_P = 0,
                sigma_M = 0.02,
                I_0 = ivp(0.2),
                R_0 = 0,
                epsilon = 0
              )
            } else{
              if (profile_var == "sigma_M") {
```

```
    rw.sd = rw.sd(
      Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
      delta = ifelse(time <= 365 * 2.50, 0.02, 0),
      phi = ifelse(time <= 365 * 2.50, 0.02, 0),
      rho = 0.02,
      sigma_P = 0,
      sigma_M = 0,
      I_0 = ivp(0.2),
      R_0 = 0,
      epsilon = 0
    )
  } else{
    if (profile_var == "R_0") {
      rw.sd = rw.sd(
        Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
        delta = ifelse(time <= 365 * 2.50, 0.02, 0),
        phi = ifelse(time <= 365 * 2.50, 0.02, 0),
        rho = 0.02,
        sigma_P = 0,
        sigma_M = 0.02,
        I_0 = ivp(0.2),
        R_0 = 0,
        epsilon = 0
      )
    } else{
      if (profile_var == "epsilon") {
        rw.sd = rw.sd(
          Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
          delta = ifelse(time <= 365 * 2.50, 0.02, 0),
          phi = ifelse(time <= 365 * 2.50, 0.02, 0),
          rho = 0.02,
          sigma_P = 0,
          sigma_M = 0.02,
          I_0 = ivp(0.2),
          R_0 = 0,
          epsilon = 0
        )
      } else{
        if (profile_var == "gamma") {
          rw.sd = rw.sd(
            Beta_0 = ifelse(time <= 365 * 2.50, 0.02, 0),
            delta = ifelse(time <= 365 * 2.50, 0.02, 0),
            phi = 0.02,
            rho = 0.02,
            sigma_P = 0,
            sigma_M = 0.02,
            I_0 = ivp(0.2),
            R_0 = 0,
            epsilon = 0,
            gamma = 0
          )
          }else{
            stop(
```

```
                            "Profile var not specified in rwsd wrapper function")

                        }

                      }

                    }

                  }
                }


              }




    }
  }
}
}


return(rw.sd)
}



rw.sd = get_rwsd(profile_var = profile_var)

detail_log = FALSE

if (detail_log == TRUE) {
  detailed_log_file_name = paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    profile_var,
    "_Profile/Detailed_Log/log_file_subset_",
    param_index,
    ".txt"
  )
  write(file = detailed_log_file_name,
        paste0("Log generated on ", Sys.time(), " \n"),
        append = FALSE)
}


mif_single_subset_data <-
  foreach(
    i = 1:nrow(param_data_subset),
    .combine = rbind,
    .packages = 'pomp2',
```

```
    .export = c(
      "rproc",
      "rmeas",
      "dmeas",
      "init",
      "paramnames",
      "statenames",
      "obsnames",
      "param_data_subset",
      "par_trans",
      "acumvarnames",
      "covar"
    )
  )  %dopar%
  {
    mif_single_param_output <-
      get_MIF_final_params_and_pfilter_LL(
        data = Rio_data_clean,
        times = Rio_data_clean$times,
        t0 = t0,
        rproc = rproc,
        params = param_data_subset[i, ],
        paramnames = paramnames,
        statenames = statenames,
        obsnames = obsnames,
        dmeas = dmeas,
        accumvars = acumvarnames,
        init = init,
        rmeas = rmeas,
        par_trans = par_trans,
        Np = 10000,
        Nmif = 100,
        cooling.fraction.50 = 0.5,
        rw.sd = rw.sd,
        delta_time = 1,
        param_index = param_index,
        i = i,
        detail_log = detail_log,
        covar = covar
      )

  }

mif_single_subset_data <- as.data.frame(mif_single_subset_data)
stopCluster(cl)

last_col = ncol(mif_single_subset_data)
mif_single_subset_rel_data = mif_single_subset_data[, -last_col]
log_output = mif_single_subset_data[, last_col]
write.csv(
  mif_single_subset_rel_data,
  file = paste(
    "../Generated_Data/Profiles/",
```

```
    model_name,
    "_Model/",
    profile_var,
    "_Profile/Subset_Outputs/",
    profile_var,
    "_",
    model_name,
    "_Profile_subset_",
    param_index,
    ".csv",
    sep = ""
  ),
  row.names = FALSE,
  na = ""
)
if (detail_log == TRUE) {
  write(file = detailed_log_file_name, log_output, append = TRUE)
}


proc.time() - ptm
```

Midway script code

I_0 Profile script

```
cat Midway_script_Model_A_7_I_0_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=I_0_Profile_A_7
#SBATCH --output=I_0_Profile_A_7_%A_%a.out
#SBATCH --error=error_I_0_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args I_0 A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

β_0 Profile script

```
cat Midway_script_Model_A_7_Beta_0_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=Beta_0_Profile_A_7
#SBATCH --output=Beta_0_Profile_A_7_%A_%a.out
```

```
#SBATCH --error=error_Beta_0_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args Beta_0 A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

σ_{P} Profile script

```
cat Midway_script_Model_A_7_sigma_P_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=sigma_P_Profile_A_7
#SBATCH --output=sigma_P_Profile_A_7_%A_%a.out
#SBATCH --error=error_sigma_P_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args sigma_P A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

σ_{M} Profile script

```
cat Midway_script_Model_A_7_sigma_M_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=sigma_M_Profile_A_7
#SBATCH --output=sigma_M_Profile_A_7_%A_%a.out
#SBATCH --error=error_sigma_M_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000
```

```
echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args sigma_M A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

ρ Profile script

```
cat Midway_script_Model_A_7_rho_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=rho_Profile_A_7
#SBATCH --output=rho_Profile_A_7_%A_%a.out
#SBATCH --error=error_rho_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args rho A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

ϕ Profile script

```
cat Midway_script_Model_A_7_phi_Profile.sbatch
```

```
#!/bin/bash
#SBATCH --job-name=phi_Profile_A_7
#SBATCH --output=phi_Profile_A_7_%A_%a.out
#SBATCH --error=error_phi_Profile_A_7_%A_%a.err
#SBATCH --array=1-50
#SBATCH --partition=broadwl
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=28
#SBATCH --mem-per-cpu=2000
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

echo $SLURM_ARRAY_TASK_ID

module load gcc
module load R/3.5.1
R CMD BATCH --vanilla '--args phi A_7' MIF_run_Model_A_7.R O/out.$SLURM_ARRAY_TASK_ID
```

Combine Midway output subsets

Once all of the 50 array jobs submitted to Midway for a particular profile have finished running on the cluster, the output from each of those 50 jobs is combined into one data frame with combinations and likelihoods for a particular profile.

```
  # ---- combine_profile_output ----


# Header ------------------------------------------------------------------
## Name: combine_profile_output.R
## Author: Rahul Subramanian
## Description: Combine MIF real profile output data into one big data frame

  combine_profile_output = function(profile_var, model_name){


ptm = proc.time()


#profile_var = "I_0"
#args = commandArgs(trailingOnly=TRUE)

#profile_var = as.character(args[1])
print(profile_var)

###Load parameter list
pd = read.csv(file = paste0(
  "../Generated_Data/Profile_Combination_Lists/",
                            model_name,"_Model/",profile_var,"_",
                            model_name,
                            "_profile_combination_list.csv"),
              header = TRUE)
#head(pd)
if(profile_var == "rho"){
  midway_max_jobs = 48
}else{
  midway_max_jobs = 50

}

mif_sim_combined_output_df = data.frame(
  matrix(nrow = 0, ncol = ncol(pd) + 1)
)
colnames(mif_sim_combined_output_df) = c(colnames(pd), "LL")

for(param_index in seq(1:midway_max_jobs)){
        #print(param_index)

  input_file_name = paste0(
    "../Generated_Data/Profiles/",
    model_name, "_Model/",
    profile_var,"_Profile/Subset_Outputs/",
    profile_var, "_", model_name,
```

```
    "_Profile_subset_",param_index,".csv")

  if(file.exists(input_file_name) == TRUE){
    mif_output_df_single_subset = read.csv(
      file = input_file_name)
  }else{
    group_size = nrow(pd)/midway_max_jobs
    start_index = (param_index-1)*group_size + 1
    end_index = param_index*group_size
    Num_mif_runs_per_start = 10
    param_data_subset_act = pd[start_index:end_index,]
    param_data_subset = param_data_subset_act[
      rep(seq_len(nrow(param_data_subset_act)),
          each = Num_mif_runs_per_start),]
    param_data_subset$seed = NA;
    param_data_subset$LL = NA;
    mif_output_df_single_subset = param_data_subset
  }

  #head(mif_output_df_single_subset)
  mif_sim_combined_output_df = rbind(
    mif_sim_combined_output_df,
    mif_output_df_single_subset)
}

output_file_name = paste0("../Generated_Data/Profiles/",
                          model_name,"_Model/",
                          profile_var, "_Profile/",
                          profile_var, "_", model_name, "_profile_combined_data.csv")
write.csv(mif_sim_combined_output_df, file = output_file_name, row.names=FALSE,na="")
}
combine_profile_output(profile_var = "sigma_P",
                       model_name = model_name)
```

```
## [1] "sigma_P"
```

```
combine_profile_output(profile_var = "sigma_M",
                       model_name = model_name)
```

```
## [1] "sigma_M"
```

```
combine_profile_output(profile_var = "I_0",
                       model_name = model_name)
```

```
## [1] "I_0"
```

```
combine_profile_output(profile_var = "Beta_0",
                       model_name = model_name)
```

```
## [1] "Beta_0"
```

```
combine_profile_output(profile_var = "delta",
                       model_name = model_name)
```

```
## [1] "delta"
```

```
combine_profile_output(profile_var = "rho",
                       model_name = model_name)
```

```
## [1] "rho"
```

```
combine_profile_output(profile_var = "phi",
                       model_name = model_name)
```

```
## [1] "phi"
```

Plot profiles

Each profile shows combinations within 20 log-likelihood units of the MLE. Blue horizontal lines denote likelihood values 2 log-likelihood units below the MLE.

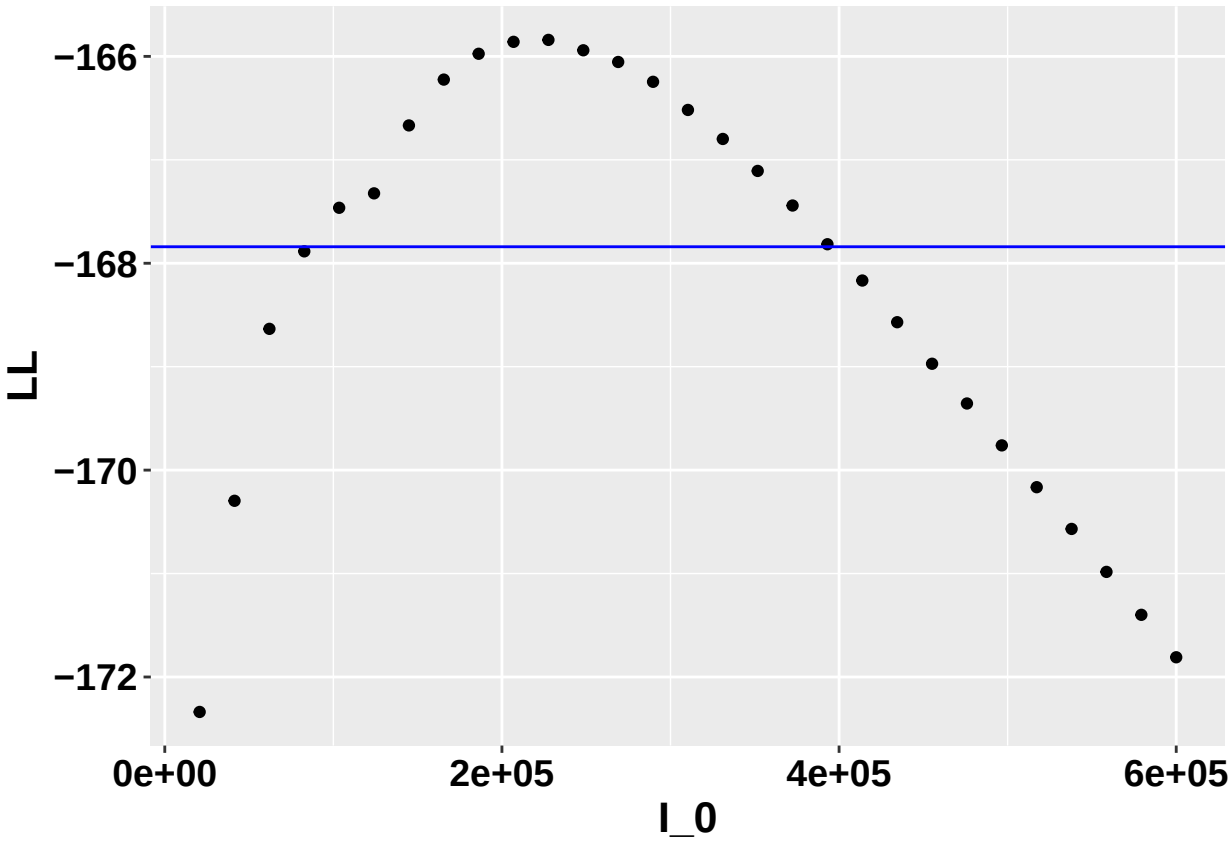
Plotting function

```
plot_profiles = function(profile_var, model_name){

#Load results
profile_data = read.csv(file = paste0(
  "../Generated_Data/Profiles/",
  model_name, "_Model/",
  profile_var, "_Profile/",
  profile_var, "_", model_name,
  "_profile_combined_data.csv"))
#head(profile_data)


profile_data_clean = na.omit(profile_data)


ML = max(profile_data_clean$LL)
cutoff_thres_20_LL_from_ML = ML - 20







cutoff_thres_2_LL_from_ML = ML - 2




### Take trace of profile
### (max at each value of profile variable)
profile_var_profile = aggregate(
  formula(paste0("LL ~ ",
                 eval(profile_var))),
  profile_data_clean, max)
```

```
top_20_LL_units = filter(
  profile_var_profile,
  LL > cutoff_thres_20_LL_from_ML)

p = ggplot(data = top_20_LL_units,
           aes_string(x = eval(profile_var),
                      y = "LL")) +
  geom_point() +
  geom_hline(yintercept = cutoff_thres_2_LL_from_ML,
             color = 'blue') +
  rahul_theme
print(p)

png(paste0("../Figures/Profiles/",
           model_name, "_Model/",
           profile_var, "_Profile/20_LL_from_ML_",
           profile_var, "_", model_name, "_profile.png"))
print(p)
dev.off()
}
```

I_0 Profile

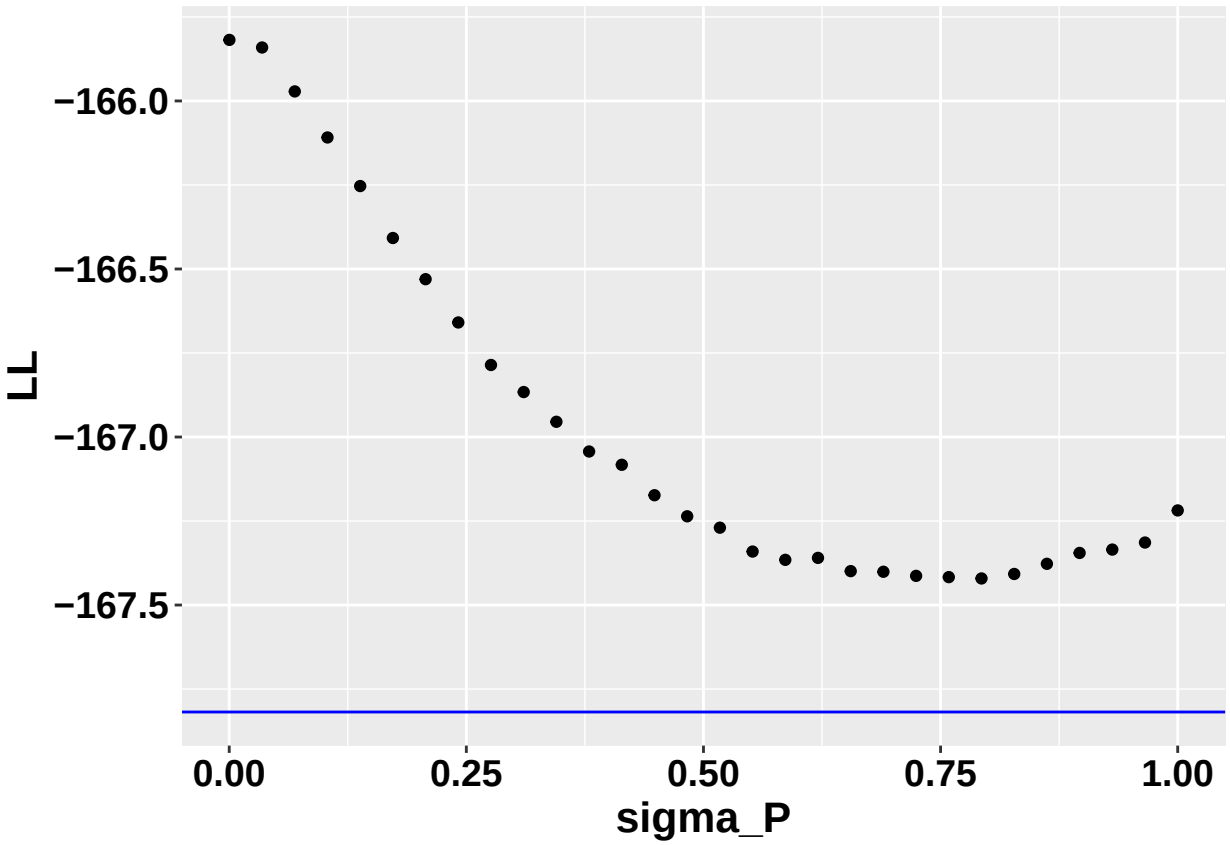
```
plot_profiles(profile_var = "I_0",
              model_name = model_name)
```




```
## pdf
##   2
```

$\sigma_{\mathbf{P}}$ Profile

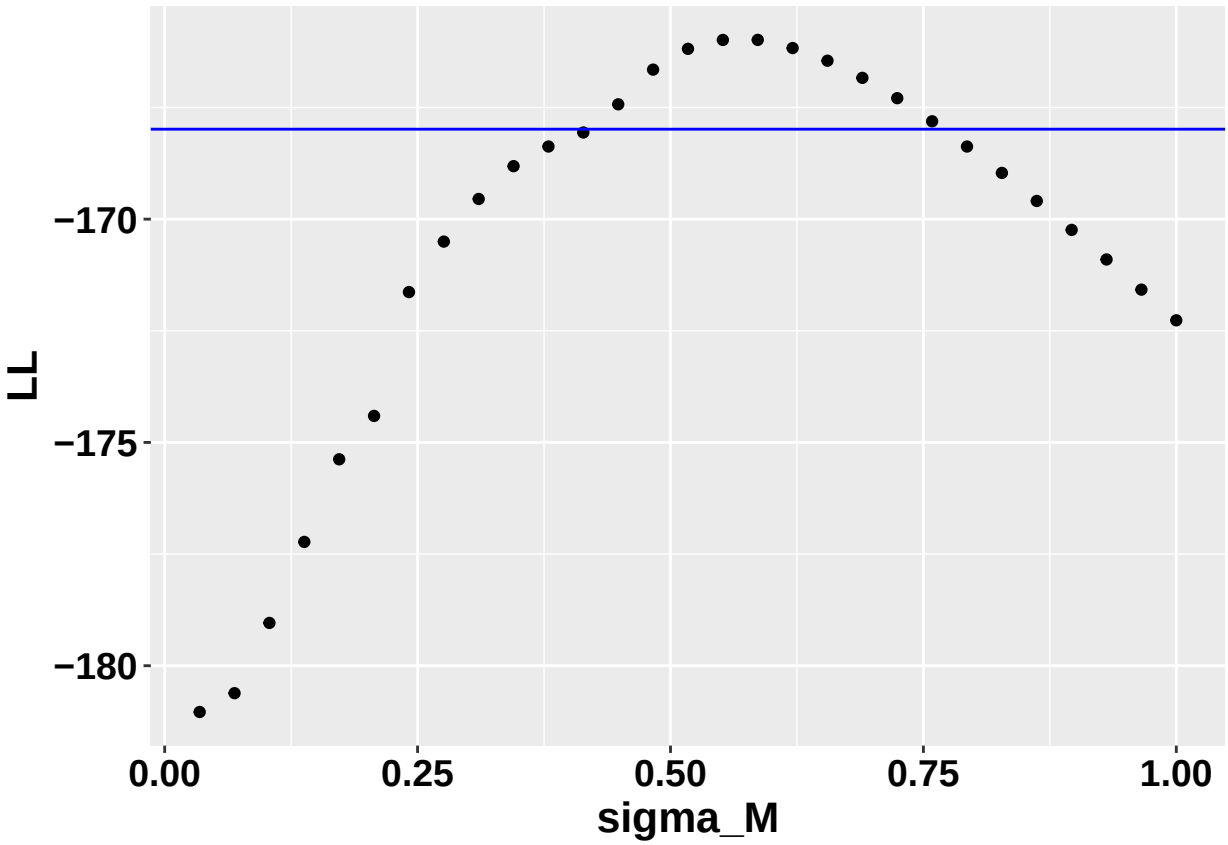
```
plot_profiles(profile_var = "sigma_P",
              model_name = model_name)
```




```
## pdf
##   2
```

σ_{M} Profile

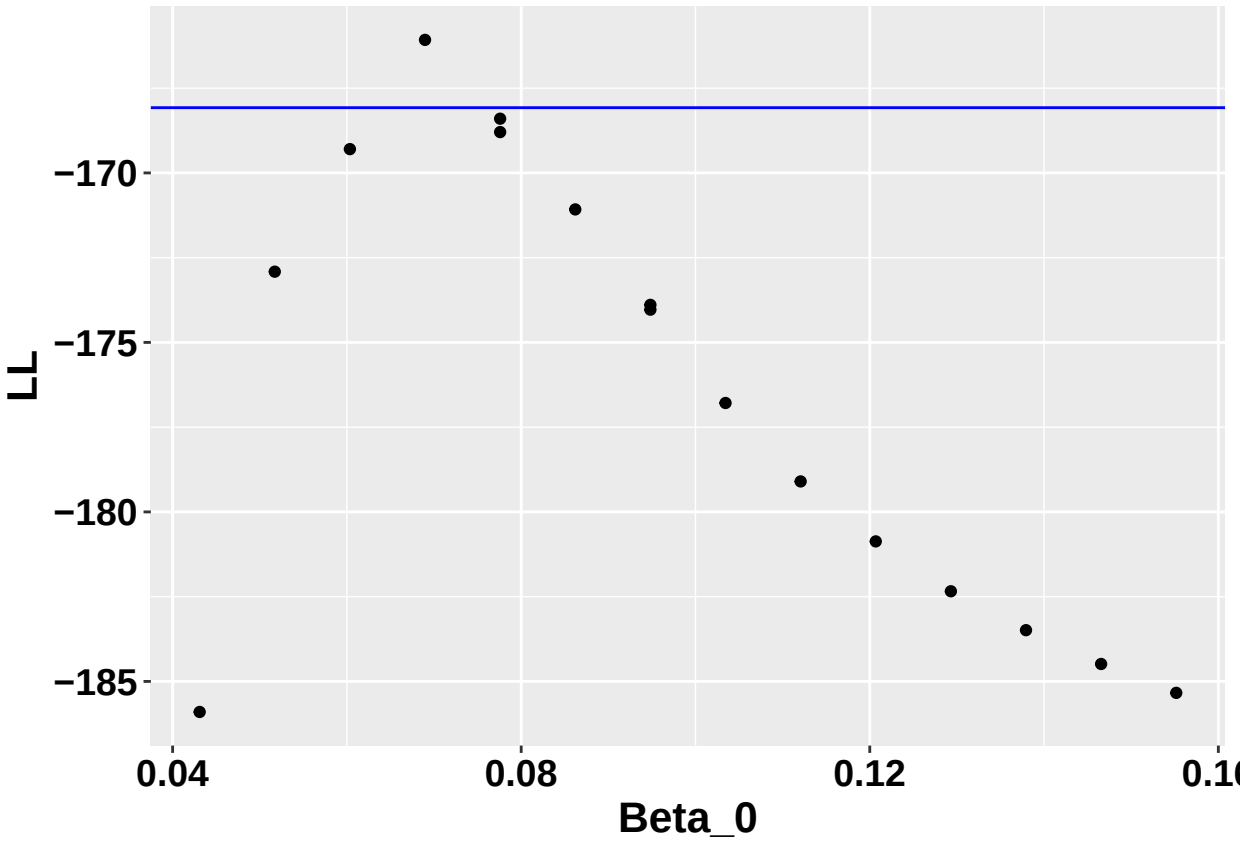
```
plot_profiles(profile_var = "sigma_M",
              model_name = model_name)
```




```
## pdf
##   2
```

β_0 Profile

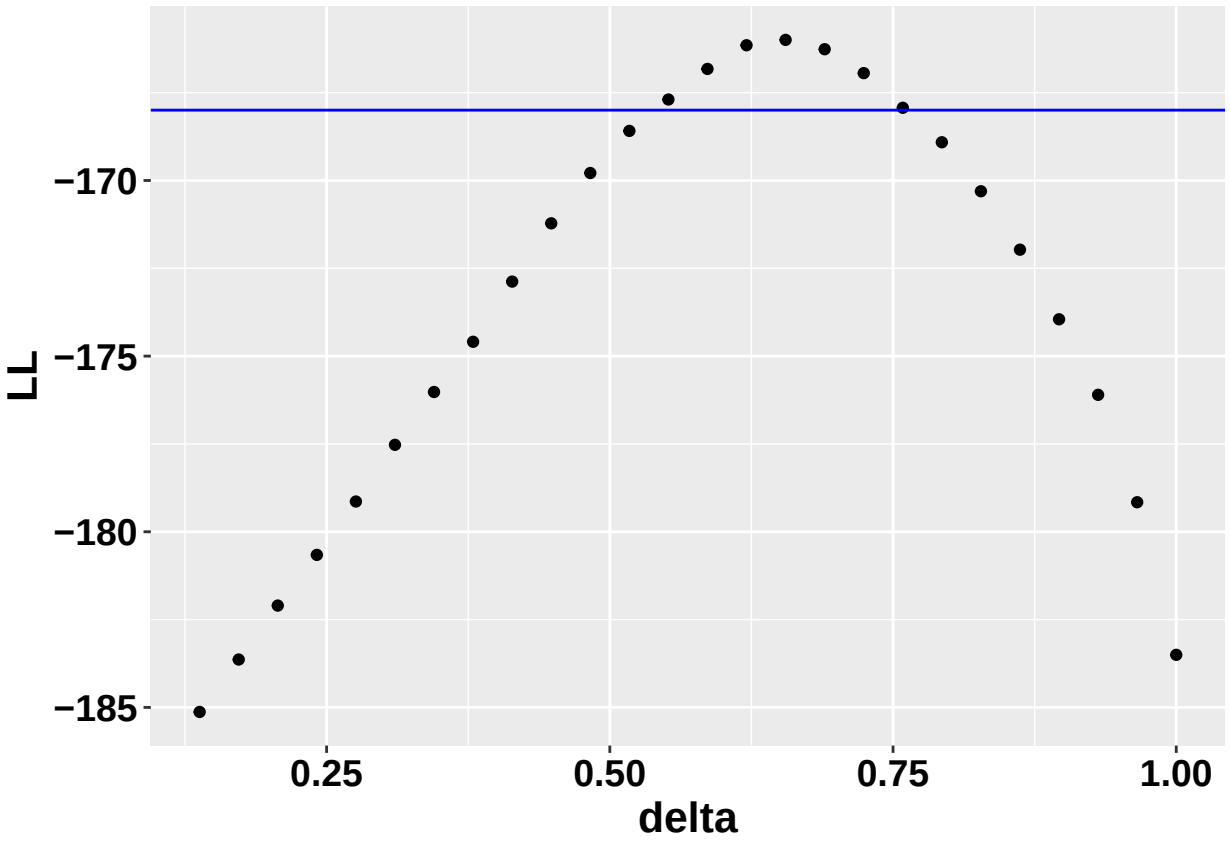
```
plot_profiles(profile_var = "Beta_0", model_name = model_name)
```




```
## pdf
##   2
```

delta **Profile**

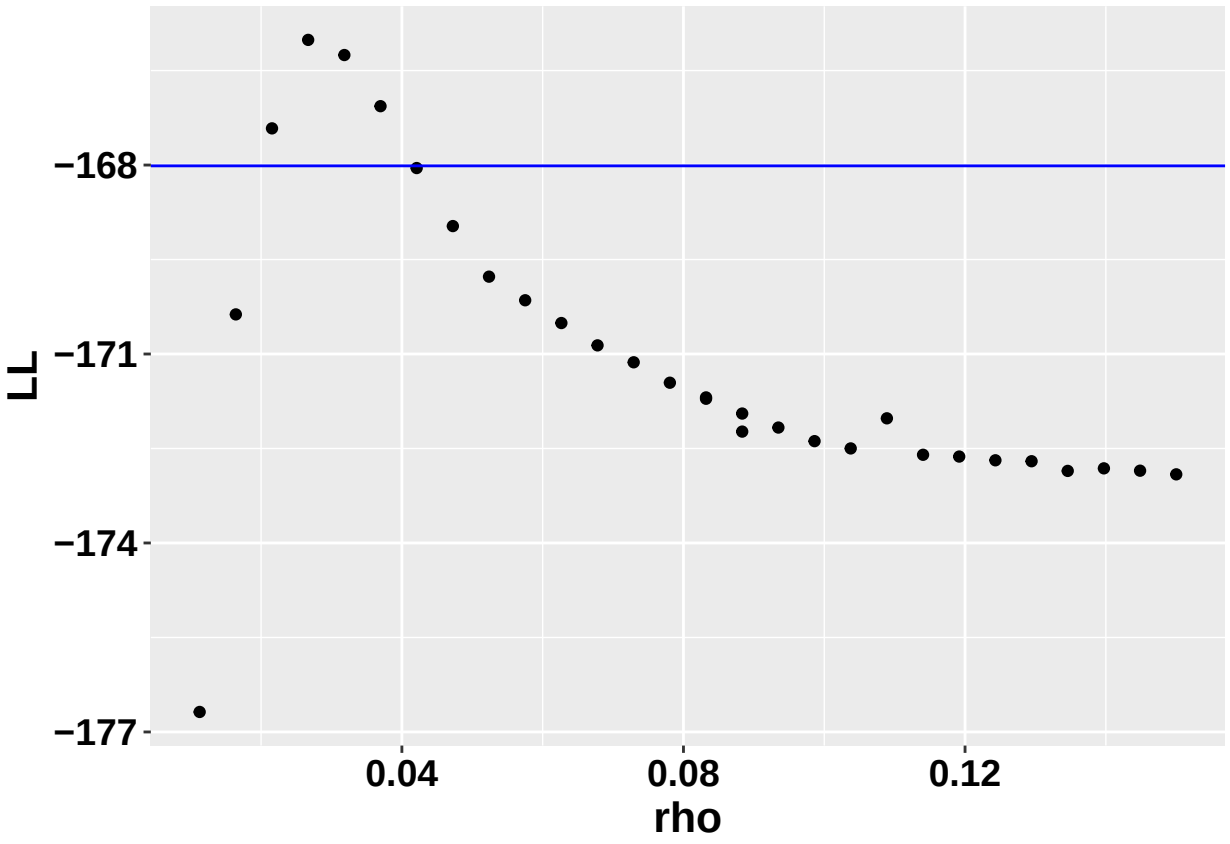
```
plot_profiles(profile_var = "delta", model_name = model_name)
```




```
## pdf
##   2
```

ρ profile

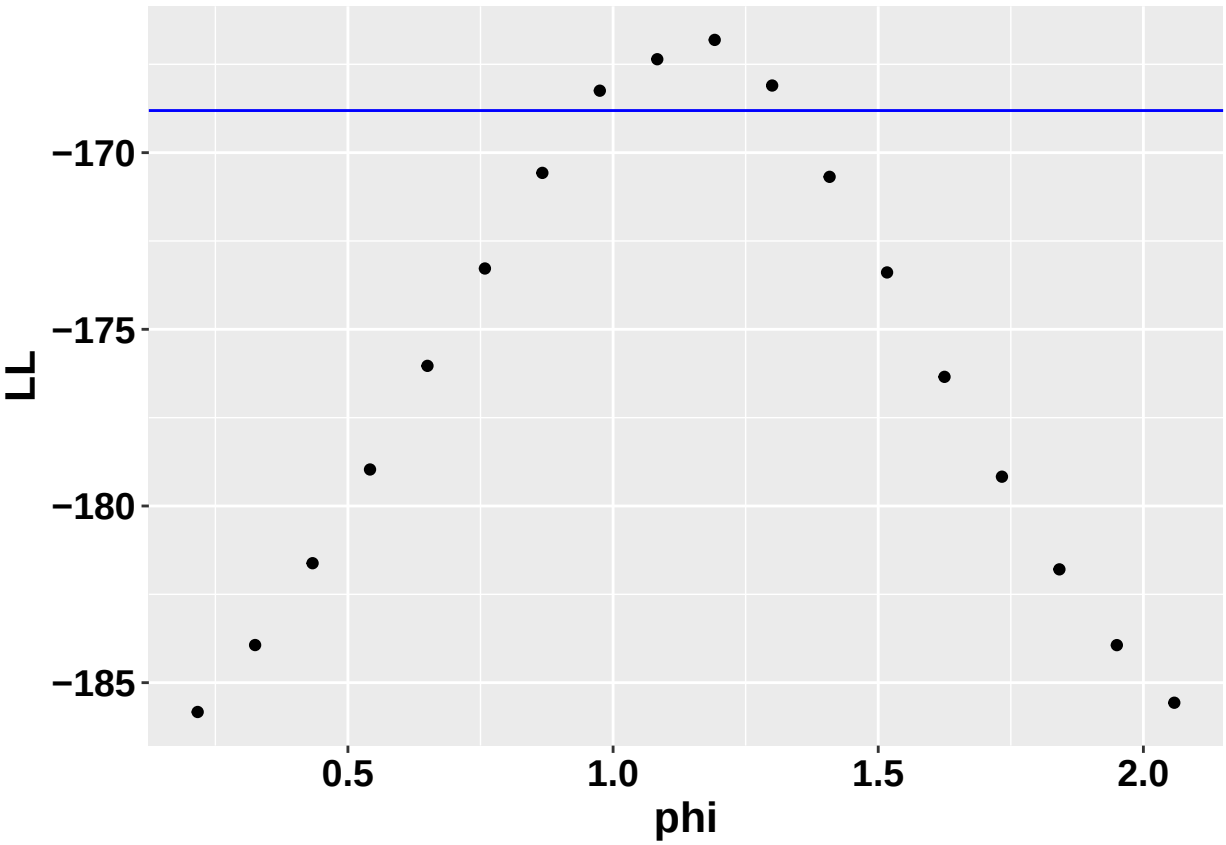
```
plot_profiles(profile_var = "rho", model_name = model_name)
```




```
## pdf
##   2
```

phi Profile

```
plot_profiles(profile_var = "phi", model_name = model_name)
```




```
## pdf
##   2
```

Combine profiles into one data frame

```
# ---- combine_likelihoods_across_profiles ----
# Header ------------------------------------------------------------------
## Name: compare_likelihoods_across_profiles.R
## Author: Rahul Subramanian
## Description: Combine likelihoods across
## all expanded profiles



I_0_profile_data = read.csv(paste0(
  "../Generated_Data/Profiles/", model_name,
                          "_Model/I_0_Profile/I_0_",
  model_name, "_profile_combined_data.csv"))
I_0_profile_data$Profile_Type = "I_0"

sigma_P_profile_data = read.csv(paste0(
  "../Generated_Data/Profiles/", model_name,
  "_Model/sigma_P_Profile/sigma_P_", model_name,
  "_profile_combined_data.csv"))
sigma_P_profile_data$Profile_Type = "sigma_P"
sigma_M_profile_data = read.csv(paste0(
```

```
  "../Generated_Data/Profiles/", model_name,
  "_Model/sigma_M_Profile/sigma_M_", model_name,
  "_profile_combined_data.csv"))
sigma_M_profile_data$Profile_Type = "sigma_M"

 combined_profile_data = rbind(sigma_P_profile_data,
                               sigma_M_profile_data)
combined_profile_data = rbind(combined_profile_data,
                              I_0_profile_data)

 Beta_0_profile_data = read.csv(paste0(
   "../Generated_Data/Profiles/", model_name,
   "_Model/Beta_0_Profile/Beta_0_", model_name,
   "_profile_combined_data.csv"))
Beta_0_profile_data$Profile_Type = "Beta_0"


combined_profile_data = rbind(combined_profile_data,
                              Beta_0_profile_data)

delta_profile_data = read.csv(paste0(
  "../Generated_Data/Profiles/", model_name,
  "_Model/delta_Profile/delta_", model_name,
  "_profile_combined_data.csv"))
  delta_profile_data$Profile_Type = "delta"

combined_profile_data = rbind(combined_profile_data,
                              delta_profile_data)
 rho_profile_data = read.csv(paste0(
   "../Generated_Data/Profiles/", model_name,
   "_Model/rho_Profile/rho_", model_name,
   "_profile_combined_data.csv"))
 rho_profile_data$Profile_Type = "rho"

 combined_profile_data = rbind(combined_profile_data,
                               rho_profile_data)

phi_profile_data = read.csv(paste0(
  "../Generated_Data/Profiles/", model_name,
  "_Model/phi_Profile/phi_", model_name,
  "_profile_combined_data.csv"))
phi_profile_data$Profile_Type = "phi"

combined_profile_data = rbind(combined_profile_data,
                              phi_profile_data)




write.csv(combined_profile_data, file = paste0(
  "../Generated_Data/Profiles/", model_name,
          "_Model/combined_", model_name,
```

```
  "_profile_data_directory.csv"),
          append = FALSE, row.names = FALSE)
```

```
## Warning in write.csv(combined_profile_data, file = paste0("../
## Generated_Data/Profiles/", : attempt to set 'append' ignored
```

Figure 1

```
rm(list =ls())
source("load_libraries_essential.R")
source("rahul_theme.R")
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.5.2
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(zoo)
library(pomp)
load(
  "../Generated_Data/Skip_Data/nCritics.Rdata")

skip_raw_data = as.numeric(as.character(nCritics))
skip_data = as.data.frame(as.matrix(nCritics))

#dim(nCritics) #18 (Reporitng rate) x 11 (delta) x 491 (R_0 value)
# Row name: Reporting rate (18 from 1% to 50%)


rep_rate_header = str_split(row.names(nCritics),
                            pattern = "%", n = Inf,
              simplify = TRUE)
delta_col_header = str_split(colnames(nCritics),
                             pattern = "_", n = Inf,
                             simplify = TRUE)
reporting_rate = as.numeric(rep_rate_header[,2])/100
delta_val = as.numeric(delta_col_header[,2])

# Reporitng Rates corresponding to S_0 values of 70%, 85%, and 90%
nine_percent_rho_index = which(reporting_rate == .09)
six_percent_rho_index = which(reporting_rate == .06)
three_percent_rho_index = which(reporting_rate == .03)



R_0_col_header = str_split(names(nCritics[1,1,]), pattern = "_", n = Inf,
```

```
                              simplify = TRUE)
R_0_skip_val = as.numeric(R_0_col_header[,2])

#Get 9% rho skip data
skip_data_rho_nine_percent =
  nCritics[nine_percent_rho_index,
           c(3,5,7,8,9),
           c(91:181)]
skip_df_rho_9 = as.data.frame(skip_data_rho_nine_percent)
skip_df_rho_9 <- tibble::rownames_to_column(skip_df_rho_9, "delta")

library(stringr)
skip_df_rho_9$delta = str_split(skip_df_rho_9$delta,
                                pattern = "_", simplify = TRUE)[,2]

skip_df_rho_9 = melt(skip_df_rho_9, id.vars = c("delta"))
skip_df_rho_9 = dplyr::select(skip_df_rho_9, delta, r0 = variable,
                              Num_Skips = value)
skip_df_rho_9$r0 = str_split(skip_df_rho_9$r0,
                             pattern = "_", simplify = TRUE)[,2]
skip_df_rho_9$rho = 0.09
skip_df_rho_9$rho_lab = "\rho~=~0.09"


#Get 6% rho skip data
skip_data_rho_six_percent =
  nCritics[six_percent_rho_index,
           c(3,5,7,8,9),
           c(91:181)]
skip_df_rho_6 = as.data.frame(skip_data_rho_six_percent)
skip_df_rho_6 <- tibble::rownames_to_column(skip_df_rho_6, "delta")

library(stringr)
skip_df_rho_6$delta = str_split(skip_df_rho_6$delta,
                          pattern = "_", simplify = TRUE)[,2]

skip_df_rho_6 = melt(skip_df_rho_6, id.vars = c("delta"))
skip_df_rho_6 = dplyr::select(skip_df_rho_6, delta, r0 = variable,
                        Num_Skips = value)
skip_df_rho_6$r0 = str_split(skip_df_rho_6$r0,
                          pattern = "_", simplify = TRUE)[,2]
skip_df_rho_6$rho = 0.06
skip_df_rho_6$rho_lab = "\rho~=~0.06"


#Get 3% rho skip data
skip_data_rho_three_percent =
  nCritics[three_percent_rho_index,
           c(3,5,7,8,9),
           c(91:181)]
skip_df_rho_3 = as.data.frame(skip_data_rho_three_percent)
skip_df_rho_3 <- tibble::rownames_to_column(skip_df_rho_3, "delta")
```

```
library(stringr)
skip_df_rho_3$delta = str_split(skip_df_rho_3$delta,
                                pattern = "_", simplify = TRUE)[,2]

skip_df_rho_3 = melt(skip_df_rho_3, id.vars = c("delta"))
skip_df_rho_3 = dplyr::select(skip_df_rho_3, delta, r0 = variable,
                              Num_Skips = value)
skip_df_rho_3$r0 = str_split(skip_df_rho_3$r0,
                             pattern = "_", simplify = TRUE)[,2]
skip_df_rho_3$rho = 0.03
skip_df_rho_3$rho_lab = "\rho~=~0.03"

skip_df = rbind(skip_df_rho_3,
                skip_df_rho_6)
skip_df = rbind(skip_df,
                skip_df_rho_9)
skip_df$r0 = as.numeric(as.character(skip_df$r0))
skip_df$Num_Skips = as.numeric(as.character(skip_df$Num_Skips))


library(latex2exp)




only_delta_07 = filter(skip_df, delta == 0.7)
only_delta_07$rho = as.factor(as.character(only_delta_07$rho))


# TIFF Figure 1 ------------------------------------------------------------
source("TIFF_Man_Fig_1.R")
```

```
## Joining by: rho
```

```
## Warning: Removed 231 rows containing missing values (geom_point).
```

```
## Joining by: rho
```

```
## Warning: Removed 180 rows containing missing values (geom_point).
```

```
## Warning: Removed 231 rows containing missing values (geom_point).
```

```
## Warning: Removed 180 rows containing missing values (geom_point).
```

TIFF Plotting Code for Figure 1

```
knitr::read_chunk('TIFF_Man_Fig_1.R')
```

```
knitr::read_chunk('TIFF_Man_Fig_1_Panel_A.R')
```

Figure 2

```
rm(list =ls())
source("load_libraries_essential.R")
source("rahul_theme.R")
library(stringr)
library(gridExtra)
library(zoo)

library(pomp)
load("../Generated_Data/Skip_Data/nCritics.Rdata")
#head(nCritics)

skip_raw_data = as.numeric(as.character(nCritics))
skip_data = as.data.frame(as.matrix(nCritics))

#dim(nCritics) #18 (Reporitng rate) x 11 (delta) x 491 (R_0 value)
# Row name: Reporting rate (18 from 1% to 50%)


rep_rate_header = str_split(row.names(nCritics),
                            pattern = "%", n = Inf,
              simplify = TRUE)
delta_col_header = str_split(colnames(nCritics),
                             pattern = "_", n = Inf,
                             simplify = TRUE)
reporting_rate = as.numeric(rep_rate_header[,2])/100
delta_val = as.numeric(delta_col_header[,2])

#10% Reporitng Rates
ten_percent_rho_index = which(reporting_rate == .10)
three_percent_rho_index = which(reporting_rate == .03)



R_0_col_header = str_split(names(nCritics[1,1,]),
                           pattern = "_", n = Inf,
                             simplify = TRUE)
R_0_skip_val = as.numeric(R_0_col_header[,2])

#Get 10% rho skip data
skip_data_rho_ten_percent =
  nCritics[ten_percent_rho_index,
           c(3,5,7,8,9),
           c(91:181)]
skip_df_rho_10 = as.data.frame(skip_data_rho_ten_percent)
skip_df_rho_10 <- tibble::rownames_to_column(skip_df_rho_10, "delta")

library(stringr)
skip_df_rho_10$delta = str_split(skip_df_rho_10$delta,
                                 pattern = "_", simplify = TRUE)[,2]

skip_df_rho_10 = melt(skip_df_rho_10, id.vars = c("delta"))
```

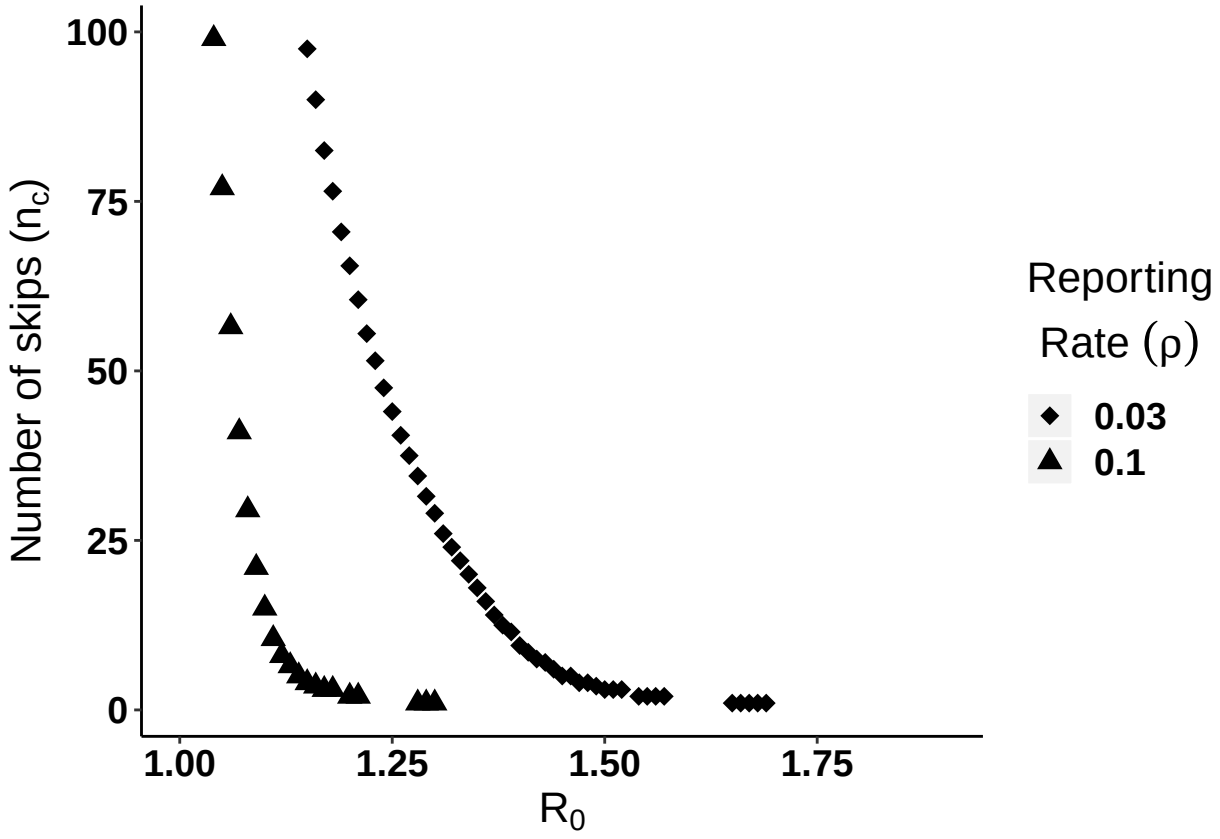
```
skip_df_rho_10 = dplyr::select(skip_df_rho_10, delta, r0 = variable,
                               Num_Skips = value)
skip_df_rho_10$r0 = str_split(skip_df_rho_10$r0,
                              pattern = "_", simplify = TRUE)[,2]
skip_df_rho_10$rho = 0.10
skip_df_rho_10$rho_lab = "\rho~=~0.10"

#Get 3% rho skip data
skip_data_rho_three_percent =
  nCritics[three_percent_rho_index,
           c(3,5,7,8,9),
           c(91:181)]
skip_df_rho_3 = as.data.frame(skip_data_rho_three_percent)
skip_df_rho_3 <- tibble::rownames_to_column(skip_df_rho_3, "delta")

library(stringr)
skip_df_rho_3$delta = str_split(skip_df_rho_3$delta,
                                pattern = "_", simplify = TRUE)[,2]
skip_df_rho_3 = melt(skip_df_rho_3, id.vars = c("delta"))
skip_df_rho_3 = dplyr::select(skip_df_rho_3, delta, r0 = variable,
                              Num_Skips = value)
skip_df_rho_3$r0 = str_split(skip_df_rho_3$r0,
                             pattern = "_", simplify = TRUE)[,2]
skip_df_rho_3$rho = 0.03
skip_df_rho_3$rho_lab = "\rho~=~0.03"
skip_df = rbind(skip_df_rho_3,
                skip_df_rho_10)
save(skip_df_rho_3,
     file =
       "../Generated_Data/Data_for_Manuscript_Figures/skip_data_rho_3.RData"
     )
skip_df$r0 = as.numeric(as.character(skip_df$r0))
skip_df$Num_Skips = as.numeric(as.character(skip_df$Num_Skips))

only_delta_07 = filter(skip_df, delta == 0.7)
only_delta_07$rho = as.factor(as.character(only_delta_07$rho))



library(latex2exp)

Fig_2_B = ggplot(data = only_delta_07) + geom_point(data = only_delta_07,
                          aes(x = r0, y = Num_Skips,
                              shape = rho),size = 3 )+
  labs(shape = expression(rho)) +
  rahul_theme +
  theme_white_background +
  scale_shape_manual(values = c(18,17),
                     name = expression(
                       atop("Reporting",
                            paste("Rate ",
                                  (rho)))))+
  labs(x = expression(R[0])) +
```

```
  labs(y = expression(n[c])) +
  labs(y = expression(
    paste("Number of skips (",
          n[c], ")")))
  labs(shape = expression(rho)) +
  rahul_man_figure_theme
```

```
## NULL
```

```
Fig_2_B
```

```
## Warning: Removed 115 rows containing missing values (geom_point).
```




```
tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig2B.tiff"),
  height = 5, width = 10, res = 300, units = "in")
print(Fig_2_B)
```

```
## Warning: Removed 115 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

```
#Load bio_good LL

model_name = "A_7"
```

```
## R_naught_act_data
profile_data_with_R_naught_act = read.csv(
  file = paste0("../Generated_Data/Profiles/",
                model_name, "_Model/combined_", model_name,
                "_profile_data_directory_with_mean_R_0.csv"))

MLE_with_R_naught_act = filter(profile_data_with_R_naught_act,
                               LL == max(LL))
bio_good_2_LL_with_R_naught = read.csv(
  file = paste0("../Generated_Data/Profiles/",
                model_name, "_Model/combined_",
                model_name, "_bio_good_2_LL_param_list.csv"))
A_7_MLE_R_naught_act = MLE_with_R_naught_act$R_naught
A_7_min_R_naught_act = min(
  bio_good_2_LL_with_R_naught$R_naught)
A_7_max_R_naught_act = max(
  bio_good_2_LL_with_R_naught$R_naught)

A_7_bio_good_2_LL = read.csv(paste0(
  "../Generated_Data/Profiles/", model_name,
  "_Model/",model_name,
  "_Model_BP_top_2_LL_all_params_bio_good_2_LL.csv"))

A_7_bio_good_2_LL$R_naught_theo =
  A_7_bio_good_2_LL$Beta_0/(
    A_7_bio_good_2_LL$gamma +
      A_7_bio_good_2_LL$mu_H)

A_7_bio_good_2_LL$nearest_skip_rho = 0
A_7_bio_good_2_LL$nearest_skip_R_naught = 0
A_7_bio_good_2_LL$nearest_skip_delta = 0
A_7_bio_good_2_LL$skips = -1
#single_param_data$nearest_skip_R_naught_index = NA
A_7_bio_good_2_LL$nearest_skip_delta_index = NA
A_7_bio_good_2_LL$nearest_skip_rho_index = NA
```

TIFF Figure 2_A Revised

```
# TIFF Figure 2_A ------------------------------------------------------------




Rio_data_clean = read.csv(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")
Rio_clean_data = Rio_data_clean
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))
Rio_data_first_three_years_only = filter(Rio_data_clean,
                                         times < 365*4)
```

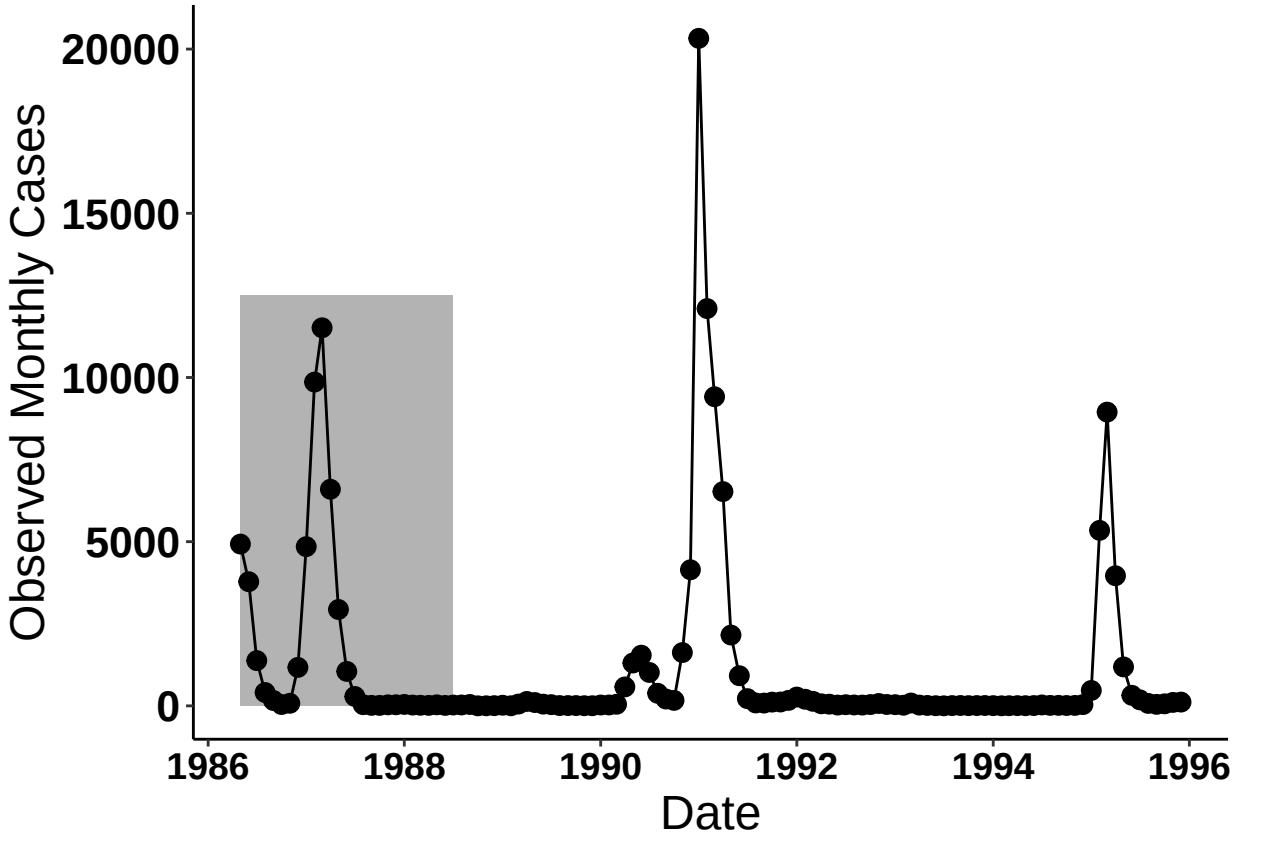
```r
# Data plot
load(file = "../Down_Data/denguerj1986-1996.RData")

Rio_city_DENV1_clean = data.frame(
  Y = as.matrix(dengue.ts),
  Date = as.Date(
    as.yearmon(time(dengue.ts))))
Rio_city_DENV1_clean = filter(Rio_city_DENV1_clean,
                              Date >= "1986-05-01")
Rio_city_DENV1_clean = filter(Rio_city_DENV1_clean,
                              Date <= "1995-12-31")

Rio_city_DENV1_clean$Year = year(Rio_city_DENV1_clean$Date)

serotype_year_map = data.frame(
  Serotype = factor(c(rep("DENV1", 5),
                      rep("DENV1 or \n DENV2",6)),
                    levels = c("DENV1",
                               "DENV2",
                               "DENV1 or \n DENV2")),
  Year = seq(from = 1986, to = 1996, by = 1))
Rio_city_dengue_86_to_96 = filter(Rio_city_DENV1_clean,
                                  Date < "1997-01-01")
dengue_data_with_serotype = join(Rio_city_dengue_86_to_96,
                                 serotype_year_map)
```

```
## Joining by: Year
```

```r
dengue_data_with_serotype$Scale =
  rep("Observed Monthly Cases",
      nrow(dengue_data_with_serotype))
dengue_data_with_serotype$Scale_index =
  rep(1, nrow(dengue_data_with_serotype))
dengue_data_with_serotype$Serotype =
  as.factor(dengue_data_with_serotype$Serotype)
dengue_data_with_serotype$Serotype =
  ordered(dengue_data_with_serotype$Serotype,
          levels = c( "DENV1","DENV1 or \n DENV2"
          ))

dengue_data_with_serotype$Serotype <- factor(
  dengue_data_with_serotype$Serotype,
  levels = c("DENV1 or \n DENV2", "DENV1", "DENV2"))


dengue_data_with_serotype$Rect_max = 12500

dengue_data_with_serotype$Rect_min = 0
dengue_data_with_serotype$Spark_Rect_max = 8000
dengue_data_with_serotype$Spark_Rect_min = 4000

log_dengue_data_with_serotype = data.frame(
  Date = dengue_data_with_serotype$Date,
  Year = dengue_data_with_serotype$Year,
```

```
  Serotype = dengue_data_with_serotype$Serotype,
  Y = log(dengue_data_with_serotype$Y),
  Scale = rep("log(Observed Monthly Cases)",
              nrow(dengue_data_with_serotype)),
  Scale_index = rep(2, nrow(dengue_data_with_serotype))
)
log_dengue_data_with_serotype$Rect_max =  10
log_dengue_data_with_serotype$Rect_min = 0.0
log_dengue_data_with_serotype$Spark_Rect_max = 7.5
log_dengue_data_with_serotype$Spark_Rect_min = 2.5

dengue_data_both_scales = rbind(dengue_data_with_serotype,
                                log_dengue_data_with_serotype)
s_0_calc_point = as.Date("1987-9-01")
dengue_data_with_serotype$Rect_max_x = as.Date("1988-07-01")
dengue_data_with_serotype$Rect_min_x = as.Date("1986-05-01")

dengue_data_both_scales$spark_start_date = as.Date("1990-01-01")
dengue_data_both_scales$spark_end_date = as.Date("1990-01-31")

Fig_2_A = ggplot(
  data = dengue_data_with_serotype,
  aes(x = Date, y = Y))+
  geom_rect(aes(xmin = as.Date(Rect_max_x),
                xmax = as.Date(Rect_min_x),
                ymin = Rect_min,
                ymax = Rect_max),
            fill = 'grey70', alpha = 0.9) +
  geom_line() +
  geom_point(size = 3)  +
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank())+
  theme(legend.position = c(.75,.87)) + xlab("Date") +
  ylab("Observed Monthly Cases") +
  theme(axis.title.y = element_text(size = 18,
                                    color = "black",
                                    face = "plain"),
        axis.text.x = element_text(size = 14,
                                   face = "bold",
                                   color = "black"),
        legend.text = element_text(size = 18,
                                   face = "bold",
                                   color = "black"),
        legend.title = element_text(size = 21,
                                    face = "bold",
                                    color = "black"),
        axis.title.x = element_text(size = 18,
                                    face = "plain"),
        legend.background = element_blank(),
        strip.background = element_blank(),
```

```
        strip.text.x = element_blank()) +
  rahul_man_figure_theme +
  theme(axis.title.x = element_text(face = "plain")) +
    theme(axis.title.y = element_text(face = "plain")) +
  theme(axis.text.x = element_text(size = 14,
                                   face = "bold",
                                   color = "black"))
Fig_2_A
```




```
tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig2_A.tiff"),
  height = 5, width = 10, res = 300, units = "in")
print(Fig_2_A)
dev.off()
```

```
## pdf
##   2
```

```
Fig_2_B_mod = Fig_2_B +
  theme(legend.position = c(.50, .75))


tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig2_raw.tiff"),
  height = 5, width = 10, res = 500, units = "in")
print(grid.arrange(Fig_2_A, Fig_2_B_mod, ncol = 2))
```

```
## Warning: Removed 115 rows containing missing values (geom_point).
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```
dev.off()
```

```
## pdf
##   2
```

Figure 3

Source plot function

```
source("Man_Figure_profile_facet_plots_plot_functions_simplified.R")
```

```
knitr::read_chunk('Man_Figure_profile_facet_plots_plot_functions_simplified.R')
```

```
## Function to get profile data
## for plotting ( Figure 3 Plot Code)
get_profile_df = function(profile_var,
                          model_name, model_label, MLE){
  #Load results
  profile_data = read.csv(
    file = paste0("../Generated_Data/Profiles/",
                  model_name, "_Model/",
                  profile_var, "_Profile/",
                  profile_var, "_",
                  model_name, "_profile_combined_data.csv"))

  na_data = filter(profile_data,
                   is.na(LL) == TRUE)
  print(paste("There are ", nrow(na_data),
              " entries with NA likelihoods"))

  profile_data_clean = na.omit(profile_data)


  profile_var_profile = aggregate(
    formula(paste0("LL ~ ",eval(profile_var))),
    profile_data_clean, max)
  profile_all_params =profile_var_profile
  MLE_prof_threshold  = MLE$LL - 2
  prof_peak_threshold = max(profile_all_params$LL) - 2

  profile_all_params$Model = model_name
  profile_all_params$Model_Name = model_label
  profile_all_params$Profile_Var = profile_var

  single_model_prof_peak_treshold_df =
    data.frame(Profile_threshold = prof_peak_threshold,
               Model = model_name,
```

```
                Model_Name = model_label)
  MLE_value_for_prof_var = dplyr::select(MLE,
                                         eval(profile_var))
  MLE_value_for_prof_var_df = data.frame(
    MLE_value_for_prof_var = as.numeric(MLE_value_for_prof_var),
    Model = model_name, Model_Name = model_label)
  prof_peak_value_for_prof_var = dplyr::select(
    filter(profile_all_params, LL == max(LL)),
    eval(profile_var) )
  prof_peak_value_for_prof_var_df = data.frame(
    prof_peak_value_for_prof_var = as.numeric(
      prof_peak_value_for_prof_var),
    Model = model_name, Model_Name = model_label)
  output_list = list(profile_all_params,
                     single_model_prof_peak_treshold_df,
                     MLE_value_for_prof_var_df,
                     prof_peak_value_for_prof_var_df)

}
```

Fig_3_Panel_A_B_C

```
# Fig_3_Panel_A_B_C ------------------------------------------------------


rahul_poster_theme = theme(
  axis.title.x = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 21,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 21,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 23,
                             face = "bold",
                             color = "black"),
  axis.text.y = element_text(size = 21,
                             face = "bold",
                             color = "black")
)
model_name_list = c("A_7")
model_label_list = factor(
  c("SIR Cosine No Immigration"))

model_label_list = factor(
  model_label_list,
  levels = c("SIR Cosine No Immigration"))
```

```
Csnippet_file_path_list = c(
  "Csnippet_SIR_cosine_model.R")
Num_est_parameters_list = c(7)
data_file_path_list = c(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")

num_years_list = c(2.50)

model_ref_df = data.frame(
  model_name = model_name_list,
  model_label = model_label_list,
  Csnippet_file_path = Csnippet_file_path_list,
  Num_est_parameters = Num_est_parameters_list,
  data_file_path = data_file_path_list,
  num_years = num_years_list,
  stringsAsFactors = FALSE)
Sup_Fig_3A_df_colnames = c("Beta_0","LL", "Model",
                           "Model_Name", "Profile_Var")
Sup_Fig_3A_prof_peak_treshold_df_colnames = c(
  "Profile_threshold", "Model", "Model_Name")
Sup_Fig_3A_MLE_value_for_prof_var_df_colnames = c(
  "MLE_value_for_prof_var", "Model", "Model_Name")
Sup_Fig_3A_prof_peak_value_for_prof_var_df_colnames = c(
  "prof_peak_value_for_prof_var", "Model", "Model_Name")


Sup_Fig_3B_df_colnames = c("rho","LL" ,"Model",
                           "Model_Name", "Profile_Var")
Sup_Fig_3B_prof_peak_treshold_df_colnames = c(
  "Profile_threshold", "Model", "Model_Name")
Sup_Fig_3B_MLE_value_for_prof_var_df_colnames = c(
  "MLE_value_for_prof_var", "Model", "Model_Name")
Sup_Fig_3B_prof_peak_value_for_prof_var_df_colnames =
  c("prof_peak_value_for_prof_var",
    "Model", "Model_Name")

Sup_Fig_3C_df_colnames = c("delta","LL" ,"Model",
                           "Model_Name", "Profile_Var")
Sup_Fig_3C_prof_peak_treshold_df_colnames = c(
  "Profile_threshold", "Model",
  "Model_Name")

Sup_Fig_3C_MLE_value_for_prof_var_df_colnames = c(
  "MLE_value_for_prof_var", "Model",
  "Model_Name")

Sup_Fig_3C_prof_peak_value_for_prof_var_df_colnames =
  c("prof_peak_value_for_prof_var",
    "Model", "Model_Name")

ML_df_colnames = c("ML", "Model",
                   "Model_Name")
```

```
Sup_Fig_3A_df = data.frame(
  matrix(nrow = 0,
         ncol = length(Sup_Fig_3A_df_colnames)))


Sup_Fig_3A_prof_peak_treshold_df = data.frame(
  matrix(nrow = 0,
         ncol = length(
           Sup_Fig_3A_prof_peak_treshold_df_colnames)))
Sup_Fig_3A_MLE_value_for_prof_var_df =
  data.frame(
    matrix(nrow = 0,
           ncol = length(
             Sup_Fig_3A_MLE_value_for_prof_var_df_colnames)))
Sup_Fig_3A_prof_peak_value_for_prof_var_df =
  data.frame(
    matrix(
      nrow = 0,
      ncol = length(
        Sup_Fig_3A_prof_peak_value_for_prof_var_df_colnames)))

Sup_Fig_3B_df = data.frame(
  matrix(
    nrow = 0,
    ncol = length(Sup_Fig_3B_df_colnames)))
Sup_Fig_3B_prof_peak_treshold_df = data.frame(
  matrix(nrow = 0,
         ncol = length(
           Sup_Fig_3B_prof_peak_treshold_df_colnames)))
Sup_Fig_3B_MLE_value_for_prof_var_df =
  data.frame(
    matrix(
      nrow = 0,
      ncol = length(
        Sup_Fig_3B_MLE_value_for_prof_var_df_colnames)))
Sup_Fig_3B_prof_peak_value_for_prof_var_df =
  data.frame(
    matrix(nrow = 0,
           ncol = length(
             Sup_Fig_3B_prof_peak_value_for_prof_var_df_colnames)))

Sup_Fig_3C_df = data.frame(
  matrix(nrow = 0,
         ncol = length(
           Sup_Fig_3C_df_colnames)))
Sup_Fig_3C_prof_peak_treshold_df = data.frame(
  matrix(nrow = 0,
         ncol = length(
           Sup_Fig_3C_prof_peak_treshold_df_colnames)))
Sup_Fig_3C_MLE_value_for_prof_var_df =
  data.frame(
    matrix(
      nrow = 0,
```

```
      ncol = length(
        Sup_Fig_3C_MLE_value_for_prof_var_df_colnames)))
Sup_Fig_3C_prof_peak_value_for_prof_var_df =
  data.frame(
    matrix(
      nrow = 0,
      ncol = length(
        Sup_Fig_3C_prof_peak_value_for_prof_var_df_colnames)))

ML_df = data.frame(
  matrix(nrow = 0,
         ncol = length(ML_df_colnames)))

colnames(Sup_Fig_3A_df) =
  Sup_Fig_3A_df_colnames

colnames(Sup_Fig_3A_prof_peak_treshold_df) =
  Sup_Fig_3A_prof_peak_treshold_df_colnames

colnames(Sup_Fig_3A_MLE_value_for_prof_var_df) =
  Sup_Fig_3A_MLE_value_for_prof_var_df_colnames

colnames(Sup_Fig_3A_prof_peak_value_for_prof_var_df) =
  Sup_Fig_3A_prof_peak_value_for_prof_var_df_colnames

colnames(Sup_Fig_3B_df) = Sup_Fig_3B_df_colnames
colnames(Sup_Fig_3B_prof_peak_treshold_df) =
  Sup_Fig_3B_prof_peak_treshold_df_colnames

colnames(Sup_Fig_3B_MLE_value_for_prof_var_df) =
  Sup_Fig_3B_MLE_value_for_prof_var_df_colnames

colnames(Sup_Fig_3B_prof_peak_value_for_prof_var_df) =
  Sup_Fig_3B_prof_peak_value_for_prof_var_df_colnames

colnames(Sup_Fig_3C_df) = Sup_Fig_3C_df_colnames
colnames(Sup_Fig_3C_prof_peak_treshold_df) =
  Sup_Fig_3C_prof_peak_treshold_df_colnames

colnames(Sup_Fig_3C_MLE_value_for_prof_var_df) =
  Sup_Fig_3C_MLE_value_for_prof_var_df_colnames

colnames(Sup_Fig_3C_prof_peak_value_for_prof_var_df) =
  Sup_Fig_3C_prof_peak_value_for_prof_var_df_colnames

colnames(ML_df) = ML_df_colnames

for(model_index in seq(1:length(model_name_list))){
  model_name = as.character(
    model_name_list[model_index])
  single_model_ref_data = filter(
    model_ref_df, model_name == !!model_name)
  model_label = single_model_ref_data$model_label
```

```
Csnippet_file_path =
  single_model_ref_data$Csnippet_file_path
Num_est_parameters =
  single_model_ref_data$Num_est_parameters
data_file_path =
  single_model_ref_data$data_file_path
num_years = single_model_ref_data$num_years

Rio_data_clean = read.csv(
  file = data_file_path)

Rio_clean_data = Rio_data_clean

source(Csnippet_file_path,
       local = TRUE)

#Set t0
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))

#Load param combination directory
combined_profile_data = read.csv(
  file = paste0(
    "../Generated_Data/Profiles/",
    model_name, "_Model/combined_",
    model_name,"_profile_data_directory.csv"))

ML = max(combined_profile_data$LL,
         na.rm = TRUE)
MLE = filter(combined_profile_data,
             LL >= ML)

ML_params = dplyr::select(MLE,
                          -one_of(
                            "seed", "LL",
                            "Profile_Type"))

MLE
single_model_ML_df = data.frame(
  ML = ML, Model = model_name,
  Model_Name = model_label)
ML_df = rbind(ML_df,
              single_model_ML_df)
#Get data for Sup Figure 3A
profile_var = "Beta_0"
single_model_output_list = get_profile_df(
  profile_var = profile_var,
  model_name = model_name,
  model_label = model_label,
  MLE = MLE)

Sup_Fig_3A_df = rbind(Sup_Fig_3A_df,
                      single_model_output_list[[1]])
```

```
Sup_Fig_3A_prof_peak_treshold_df = rbind(
  Sup_Fig_3A_prof_peak_treshold_df,
  single_model_output_list[[2]])

Sup_Fig_3A_prof_peak_treshold_df$Profile_Var = profile_var
Sup_Fig_3A_MLE_value_for_prof_var_df =
  rbind(
    Sup_Fig_3A_MLE_value_for_prof_var_df,
    single_model_output_list[[3]])

Sup_Fig_3A_MLE_value_for_prof_var_df$Profile_Var = profile_var

Sup_Fig_3A_prof_peak_value_for_prof_var_df =
  rbind(
    Sup_Fig_3A_prof_peak_value_for_prof_var_df,
    single_model_output_list[[4]])

Sup_Fig_3A_prof_peak_value_for_prof_var_df$Profile_Var =
  profile_var

#Get data for Sup Figure 3B
profile_var = "rho"
single_model_output_list = get_profile_df(
  profile_var = profile_var,
  model_name = model_name,
  model_label = model_label,
  MLE = MLE)

Sup_Fig_3B_df = rbind(Sup_Fig_3B_df,
                      single_model_output_list[[1]])

Sup_Fig_3B_prof_peak_treshold_df = rbind(
  Sup_Fig_3B_prof_peak_treshold_df,
  single_model_output_list[[2]])

Sup_Fig_3B_prof_peak_treshold_df$Profile_Var =
  profile_var

Sup_Fig_3B_MLE_value_for_prof_var_df =
  rbind(
    Sup_Fig_3B_MLE_value_for_prof_var_df,
    single_model_output_list[[3]])

Sup_Fig_3B_MLE_value_for_prof_var_df$Profile_Var =
  profile_var

Sup_Fig_3B_prof_peak_value_for_prof_var_df =
  rbind(
    Sup_Fig_3B_prof_peak_value_for_prof_var_df,
    single_model_output_list[[4]])

Sup_Fig_3B_prof_peak_value_for_prof_var_df$Profile_Var =
  profile_var
```

```
  #Get data for Sup Figure 3C
  profile_var = "delta"
  single_model_output_list = get_profile_df(
    profile_var = profile_var,
    model_name = model_name,
    model_label = model_label,
    MLE = MLE)

  Sup_Fig_3C_df = rbind(
    Sup_Fig_3C_df,
    single_model_output_list[[1]])

  Sup_Fig_3C_prof_peak_treshold_df = rbind(
    Sup_Fig_3C_prof_peak_treshold_df,
    single_model_output_list[[2]])

  Sup_Fig_3C_prof_peak_treshold_df$Profile_Var = profile_var

  Sup_Fig_3C_MLE_value_for_prof_var_df =
    rbind(
      Sup_Fig_3C_MLE_value_for_prof_var_df,
      single_model_output_list[[3]])

  Sup_Fig_3C_MLE_value_for_prof_var_df$Profile_Var = profile_var

  Sup_Fig_3C_prof_peak_value_for_prof_var_df =
    rbind(Sup_Fig_3C_prof_peak_value_for_prof_var_df,
          single_model_output_list[[4]])
  Sup_Fig_3C_prof_peak_value_for_prof_var_df$Profile_Var =
    profile_var

}
```

```
## [1] "There are  0  entries with NA likelihoods"
## [1] "There are  0  entries with NA likelihoods"
## [1] "There are  0  entries with NA likelihoods"
```

```
Sup_Fig_3A_df  =  Sup_Fig_3A_df %>%
  mutate(var_value = Beta_0) %>%
  dplyr::select(-Beta_0)
Sup_Fig_3B_df = Sup_Fig_3B_df %>%
  mutate(var_value = rho) %>%
  dplyr::select(-rho)
Sup_Fig_3C_df = Sup_Fig_3C_df %>%
  mutate(var_value = delta) %>%
  dplyr::select(-delta)

combined_Sup_Fig_3_df = rbind(Sup_Fig_3A_df,
                              Sup_Fig_3B_df)

combined_Sup_Fig_3_df = rbind(combined_Sup_Fig_3_df,
                              Sup_Fig_3C_df)

combined_profile_Sup_Fig_3_MLE_value_for_prof_var_df =
```

```
  rbind(
    Sup_Fig_3A_MLE_value_for_prof_var_df,
    Sup_Fig_3B_MLE_value_for_prof_var_df)

combined_profile_Sup_Fig_3_MLE_value_for_prof_var_df =
  rbind(
    combined_profile_Sup_Fig_3_MLE_value_for_prof_var_df,
    Sup_Fig_3C_MLE_value_for_prof_var_df)

combined_profile_Sup_Fig_3_prof_peak_treshold_df =
  rbind(
    Sup_Fig_3A_prof_peak_treshold_df,
    Sup_Fig_3B_prof_peak_treshold_df)

combined_profile_Sup_Fig_3_prof_peak_treshold_df =
  rbind(
    combined_profile_Sup_Fig_3_prof_peak_treshold_df,
    Sup_Fig_3C_prof_peak_treshold_df)

combined_profile_Sup_Fig_3_prof_peak_value_for_prof_var_df =
  rbind(
    Sup_Fig_3A_prof_peak_value_for_prof_var_df,
    Sup_Fig_3B_prof_peak_value_for_prof_var_df)

combined_profile_Sup_Fig_3_prof_peak_value_for_prof_var_df =
  rbind(
    combined_profile_Sup_Fig_3_prof_peak_value_for_prof_var_df,
    Sup_Fig_3C_prof_peak_value_for_prof_var_df)

ymin =
  combined_profile_Sup_Fig_3_prof_peak_treshold_df %>%
  group_by(Profile_Var) %>%
  summarize(ymin = Profile_threshold-10) %>%
  as.data.frame()

min_prof_value = combined_Sup_Fig_3_df %>%
  group_by(Profile_Var) %>%
  summarize(prof_min = min(LL)) %>%
  as.data.frame()

ymin = join(
  ymin, min_prof_value)
```

```
## Joining by: Profile_Var
```

```
y_thres_df = ymin %>%
  group_by(Profile_Var) %>%
  summarize(y_thres = max(ymin, prof_min)) %>%
  as.data.frame()
```

```
y_lim_min = min(y_thres_df$y_thres)
combined_Sup_Fig_3_df_clean = filter(
  combined_Sup_Fig_3_df, LL > y_lim_min)
```

```
plot_label_df = data.frame(
  Profile_Var =
    combined_profile_Sup_Fig_3_MLE_value_for_prof_var_df$Profile_Var,
  plot_var_label = c("beta[0]","rho", "delta" ))

combined_Sup_Fig_3_df = join(
  combined_Sup_Fig_3_df, plot_label_df)
```

```
## Joining by: Profile_Var
```

```
Fig_3_ABC_plot_data = join(
  combined_Sup_Fig_3_df,
  combined_profile_Sup_Fig_3_MLE_value_for_prof_var_df)
```

```
## Joining by: Model, Model_Name, Profile_Var
```

Load panel plot theme

```
# Combined Plot ------------------------------------------------------------

library(gridExtra)
library(grid)
library(lattice)

rahul_panel_theme = theme(
  axis.title.x = element_text(size = 10,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 10,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 10,
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 10,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 9,
                             face = "bold",
                             color = "black"),
  axis.text.y = element_text(size = 8,
                             face = "bold",
                             color = "black")
)

rahul_big_panel_theme = theme(
  axis.title.x = element_text(size = 14,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 14,
```

```
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 14,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  axis.text.y = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  plot.margin = unit(c(.5,.5,.5,.5), "cm"),
  legend.background = element_rect(
    fill = "transparent"),
  legend.box.margin = unit(c(.5,.5,.5,.5), "cm")
)
```

Add Polynomial Fit Curves to Profiles for Figure 3

```
Fig_3_ABC_plot_data = join(
  Fig_3_ABC_plot_data, ML_df)
```

```
## Joining by: Model, Model_Name
```

```
cutoff_value  = -174
Fig_3_ABC_plot_data = filter(
  Fig_3_ABC_plot_data, LL > ML - 10 )

Fig_3_ABC_plot_data$Metric = "LL"
Fig_3_ABC_plot_data$low_bound = ML-11
Fig_3_ABC_plot_data$Line_Color = "Show_Line"

Fig_3_combined_data = Fig_3_ABC_plot_data

Fig_3_combined_data$plot_var_label =
  factor(
    Fig_3_combined_data$plot_var_label,
    levels = c("beta[0]", "delta","rho"))

## Calculate polynomial fit

#### Beta_0 Profile
beta_0_poly_data = Fig_3_ABC_plot_data %>%
  filter(Profile_Var == "Beta_0") %>%
  dplyr::select(Profile_Var, var_value, LL)
beta_0_poly_fit_model <-
  lm(beta_0_poly_data$LL ~
       poly(beta_0_poly_data$var_value,
            2, raw = TRUE))

beta_0_poly_data$Poly_Fit =
  beta_0_poly_fit_model$fitted.values
```

```
small_breaks_beta_0 = seq(
  from = min(beta_0_poly_data$var_value),
  to = max(beta_0_poly_data$var_value),
  length = 10^3)

beta_0_poly_intercept =summary(
  beta_0_poly_fit_model)$coefficients[1,1]

beta_0_poly_order_1 = summary(
  beta_0_poly_fit_model)$coefficients[2,1]

beta_0_poly_order_2 = summary(
  beta_0_poly_fit_model)$coefficients[3,1]

beta_0_poly_curve = beta_0_poly_intercept +
  beta_0_poly_order_1*small_breaks_beta_0 +
  beta_0_poly_order_2*I(small_breaks_beta_0^2)

beta_0_poly_curve_df = data.frame(
  small_breaks = small_breaks_beta_0,
  poly_curve = beta_0_poly_curve,
  plot_var_label = "beta[0]")

#### rho Profile
rho_poly_data = Fig_3_ABC_plot_data %>%
  filter(Profile_Var == "rho") %>%
  dplyr::select(Profile_Var,
                var_value, LL)
rho_poly_fit_model <- lm(
  rho_poly_data$LL ~ poly(
    rho_poly_data$var_value,4, raw = TRUE))

rho_poly_data$Poly_Fit =
  rho_poly_fit_model$fitted.values

small_breaks_rho = seq(
  from= min(rho_poly_data$var_value),
  to = max(rho_poly_data$var_value),
  length = 10^3)

rho_poly_intercept =summary(
  rho_poly_fit_model)$coefficients[1,1]

rho_poly_order_1 = summary(
  rho_poly_fit_model)$coefficients[2,1]

rho_poly_order_2 = summary(
  rho_poly_fit_model)$coefficients[3,1]

rho_poly_order_3 = summary(
  rho_poly_fit_model)$coefficients[4,1]

rho_poly_order_4 = summary(
```

```
  rho_poly_fit_model)$coefficients[5,1]

rho_poly_curve = rho_poly_intercept +
  rho_poly_order_1*small_breaks_rho +
  rho_poly_order_2*I(small_breaks_rho^2) +
  rho_poly_order_3*I(small_breaks_rho^3) +
  rho_poly_order_4*I(small_breaks_rho^4)

rho_poly_curve_df = data.frame(
  small_breaks = small_breaks_rho,
  poly_curve = rho_poly_curve,
  plot_var_label = "rho")

#### delta Profile
delta_poly_data = Fig_3_ABC_plot_data %>%
  filter(Profile_Var == "delta") %>%
  dplyr::select(Profile_Var,
                var_value, LL)

delta_poly_fit_model <- lm(
  delta_poly_data$LL ~ poly(
    delta_poly_data$var_value,2,
    raw = TRUE))

delta_poly_data$Poly_Fit =
  delta_poly_fit_model$fitted.values

small_breaks_delta = seq(
  from= min(delta_poly_data$var_value),
  to = max(delta_poly_data$var_value),
  length = 10^3)

delta_poly_intercept =summary(
  delta_poly_fit_model)$coefficients[1,1]

delta_poly_order_1 = summary(
  delta_poly_fit_model)$coefficients[2,1]

delta_poly_order_2 = summary(
  delta_poly_fit_model)$coefficients[3,1]

delta_poly_curve = delta_poly_intercept +
  delta_poly_order_1*small_breaks_delta +
  delta_poly_order_2*I(small_breaks_delta^2)

delta_poly_curve_df = data.frame(
  small_breaks = small_breaks_delta,
  poly_curve = delta_poly_curve,
  plot_var_label = "delta")

combined_poly_data = rbind(
  beta_0_poly_curve_df, rho_poly_curve_df)
combined_poly_data = rbind(
```

```
  combined_poly_data, delta_poly_curve_df)
```

Make Combined Plot for Figure 3

```
Fig_3_comb_plot = ggplot()  +
  geom_point(data = Fig_3_combined_data,
             aes(x = var_value, y = LL,
                 color = Metric, shape = Metric)) +
  scale_linetype_manual(values = c("blank", "solid")) +
  rahul_man_figure_theme +
  rahul_big_panel_theme  +
  theme_white_background  +
  geom_hline(data = Fig_3_combined_data,
             aes(yintercept = ML -2),
             size = 1.0, linetype = "dashed",
             color = 'grey70') +
  geom_vline(data = Fig_3_combined_data,
             aes(xintercept = MLE_value_for_prof_var),
             size = 1.0, linetype = "twodash",
             show.legend= F, color = 'grey70') +
  facet_wrap(~plot_var_label,
             scales = "free",
             strip.position = "bottom",
             labeller=label_parsed,
             nrow = 1) +
  geom_hline(data = Fig_3_combined_data,
             aes(yintercept = low_bound),
             color = 'white', linetype = 'blank') +
  geom_line(data = combined_poly_data,
            aes(x = small_breaks, y = poly_curve),
            color = 'red', show.legend = F) +
  scale_x_continuous(
    breaks = scales::pretty_breaks(n = 3)) +
  theme(
    aspect.ratio = 1,
    strip.background = element_blank(),
    strip.placement = "outside"
  ) +
  theme(legend.position = "None") +
  scale_color_manual(values = c("black", "red",
                                "black", "white"),
                     limits = c("LL", "Skips",
                                "Show_Line", "No_Line")) +
  scale_shape_manual(values = c(16, 1)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 4))  +
  ylab(expression(paste(" Log Likelihood "))) +
  theme(axis.text.y = element_text(size = 16),
        axis.text.x = element_text(size = 16),
        axis.title.x = element_text(face = "plain")) +
  theme(panel.spacing = unit(1.75, "lines")) +
  xlab(
    paste0(
```

```
      "(Transmission Rate",
      ")                              (",
      "Seasonality Amplitude",
      ")                         (",
      "Reporting Rate)      ")
    )

#Fig_3_comb_plot

tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig3_raw.tiff"),
  height = 5, width = 10, res = 500, units = "in")
Fig_3_comb_plot
dev.off()
```

```
## pdf
##   2
```

Figure 4

Fig 4 Panel A

```
# Fig_3_Panel_D ----------------------------------------------------------------


model_name  = "A_7"

bio_good_2_LL = read.csv(paste0(
  "../Generated_Data/Profiles/",
  model_name,
  "_Model/",
  model_name,
  "_Model_BP_top_2_LL_all_params_bio_good_2_LL.csv"))
ML_combo_num = which(
  bio_good_2_LL$LL == max(bio_good_2_LL$LL))
all_R0_data =
  read.csv(
    paste0("../Generated_Data/Profiles/",
           model_name, "_Model/",
           model_name,
           "_Model_BP_top_2_LL_all_params_sim_R0_data.csv"))

R0_min = aggregate(R_0 ~ time,
                   all_R0_data, FUN = min)
R0_min = dplyr::select(R0_min, time = time,
                       R_0_min = R_0)
R0_max = aggregate(R_0 ~ time,
                   all_R0_data, FUN = max)
R0_max = dplyr::select(R0_max,
                       time = time, R_0_max = R_0)
```

```
ML_R0_df = filter(all_R0_data,
                  combo_num == ML_combo_num)
ML_R0_df = dplyr::select(ML_R0_df,
                         time = time, R_0_MLE = R_0)
R_0_ribbon_df = join(R0_min, R0_max)
```

```
## Joining by: time
```

```
R_0_ribbon_df = join(R_0_ribbon_df,
                     ML_R0_df)
```

```
## Joining by: time
```

```
R_0_ribbon_df_melt = melt(
  R_0_ribbon_df,
  id.vars = c("time", "R_0_min", "R_0_max" ))

ribbon_label = "R_0 range \n  (All  2 LL Combinations)"
R_0_ribbon_df_melt$Ribbon_label = ribbon_label

## R_0 upper and lower bounds
## for on and off-season peak and trough
min(R_0_ribbon_df_melt$R_0_min)
```

```
## [1] 0.3061775
```

```
min(R_0_ribbon_df_melt$R_0_max)
```

```
## [1] 0.523106
```

```
max(R_0_ribbon_df_melt$R_0_min)
```

```
## [1] 1.78973
```

```
max(R_0_ribbon_df_melt$R_0_max)
```

```
## [1] 2.092432
```

```
fill_vec = c("grey70", "NA")
names(fill_vec) = ribbon_label


### Plot 1 year only

all_R0_data$Year = all_R0_data$time/365
all_R0_data$Days_in_Year = (
  all_R0_data$time%%365)
all_R0_data$Month = round(
  (all_R0_data$Days_in_Year/365)*12) + 1

single_year_R_0_data= filter(
  all_R0_data, Year <=  2 & Year >= 1 )

month_lookup_table = data.frame(
  Month = seq(1:12), Month_Name = month.abb)

all_R0_data = join(all_R0_data,
                   month_lookup_table)
```

```
## Joining by: Month
```

```
all_R0_min = aggregate(R_0 ~ time, all_R0_data,
                               FUN = min)
all_R0_min = dplyr::select(
  all_R0_min, time = time,
  R_0_min = R_0)

all_R0_max = aggregate(
  R_0 ~ time, all_R0_data,FUN = max)

all_R0_max = dplyr::select(
  all_R0_max, time = time, R_0_max = R_0)
all_ML_R0_df = filter(all_R0_data,
                      combo_num == ML_combo_num)
all_ML_R0_df = dplyr::select(
  all_ML_R0_df, time = time, R_0_MLE = R_0)
all_R_0_ribbon_df = join(all_R0_min, all_R0_max)
```

```
## Joining by: time
```

```
all_R_0_ribbon_df = join(
  all_R_0_ribbon_df, all_ML_R0_df)
```

```
## Joining by: time
```

```
all_R_0_ribbon_df_melt =
  melt(
    all_R_0_ribbon_df,
    id.vars = c("time", "R_0_min", "R_0_max" ))
ribbon_label =
  "R_0 range \n  (All  2 LL Combinations)"

all_R_0_ribbon_df_melt$Ribbon_label =
  ribbon_label

fill_vec = c("grey70")
names(fill_vec) = ribbon_label
plot_label_months =seq(
  from = 1, to = length(unique(all_R0_data$time)),
  by = 2)

plot_label_month_names =
  all_R0_data$Month_Name[plot_label_months]

plot_label_times =
  all_R0_data$time[plot_label_months]

ribbon_label = "2 LL from \n MLE"
all_R_0_ribbon_df_melt$Ribbon_label = ribbon_label
fill_vec = c("grey70")
names(fill_vec) = ribbon_label

Fig_4_Panel_A = ggplot(data = all_R_0_ribbon_df_melt) +
  geom_ribbon(aes(x = time, ymin = R_0_min,
                  ymax =  R_0_max,
```

```
                  fill = Ribbon_label)) +
  geom_line(aes(x = time, y = value,
                color = variable)) +
  geom_point(aes(x = time, y = value,
                 color = variable),
             size = 3) +
  rahul_theme +
  theme(
    legend.text = element_text(size = 12,
                               face = "bold",
                               color = "black")) +
  theme_white_background +
  scale_color_manual(
    name = "",
    values = c("red"),
    labels = c(
      "MLE Trajectory \n (Shaded Region: \n 95% Quantiles)",
      "Observed"))  +
  scale_fill_manual(
    name = "", values = fill_vec,
    labels = c(
      "MLE Trajectory \n (Shaded Region: \n 95% Quantiles)",
      "Observed")) +
  xlab("Month")+
  scale_x_continuous(
    breaks = as.numeric(plot_label_times),
    labels = plot_label_month_names) +
  ylab(expression(paste(R[0]))) +
  rahul_man_figure_theme +
  theme(legend.margin = margin(t = 0, unit='cm'))
#Fig_4_Panel_A
```

Figure 4 Panel B

```
rahul_poster_theme = theme(
  axis.title.x = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 21,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 21,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 23,
                             face = "bold",
                             color = "black"),
  axis.text.y = element_text(size = 21,
                             face = "bold",
```

```
                              color = "black")
)

model_name_list = c("A_7")
model_label_list = factor(
  c("SIR Cosine No Immigration"))
model_label_list = factor(
  model_label_list,
  levels = c("SIR Cosine No Immigration"))

Csnippet_file_path_list = c(
  "Csnippet_SIR_cosine_model.R")
Num_est_parameters_list = c(7)
data_file_path_list = c(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")

num_years_list = c(2.50)

model_ref_df = data.frame(
  model_name = model_name_list,
  model_label = model_label_list,
  Csnippet_file_path = Csnippet_file_path_list,
  Num_est_parameters = Num_est_parameters_list,
  data_file_path = data_file_path_list,
  num_years = num_years_list,
  stringsAsFactors = FALSE
)


model_index = 1
print(model_index)
```

```
## [1] 1
```

```
model_name = as.character(
  model_name_list[model_index])

single_model_ref_data = filter(
  model_ref_df, model_name == !!model_name)

model_label =
  single_model_ref_data$model_label

Csnippet_file_path =
  single_model_ref_data$Csnippet_file_path

Num_est_parameters =
  single_model_ref_data$Num_est_parameters

data_file_path =
  single_model_ref_data$data_file_path

num_years =
  single_model_ref_data$num_years
```

```
Rio_data_clean = read.csv(file = data_file_path)

Rio_clean_data = Rio_data_clean

source(Csnippet_file_path,
       local = TRUE)

#Set t0
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))

all_combo_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_cases_data.csv"
  )
)

all_R0_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_R0_data.csv"
  )
)

all_combo_S_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_S_over_N_data.csv"
  )
)

all_R_eff_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_Reff_data.csv"
  )
)

bio_good_2_LL = read.csv(
  paste0(
```

```
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_bio_good_2_LL.csv"
  )
)


ML_combo_num = which(
  bio_good_2_LL$LL ==
    max(bio_good_2_LL$LL))

ML_output = filter(
  all_combo_data, combo_num == ML_combo_num)
ML_output = dplyr::select(
  ML_output,
  time = time,
  ML_median = sim_data_median,
  ML_high_Q = sim_data_high_Q,
  ML_low_Q = sim_data_low_Q
)
true_data = dplyr::select(Rio_clean_data,
                          time = times,
                          Observed_Data = Y)
comp_data = join(ML_output, true_data)
```

```
## Joining by: time
```

```
comp_data_melt = melt(
  comp_data,
  id.vars = c("time", "ML_high_Q", "ML_low_Q"))


label_df  =
  data.frame(
    Label_name =
      c(
        "Simulation Median \n (Shaded Region: \n 95% Quantiles)",
        "Observed"),
    variable = c("ML_median",
                 "Observed_Data"))

comp_data_melt_with_label =
  join(comp_data_melt, label_df)
```

```
## Joining by: variable
```

```
Fig_4_Panel_B =
  ggplot(data =
           comp_data_melt_with_label) +
  geom_ribbon(aes(
    x = time / 365,
    ymin = log(ML_low_Q),
    ymax = log(ML_high_Q),
```

```
    fill = Label_name
  )) +
  geom_line(aes(
    x = time / 365,
    y = log(value),
    color = Label_name
  )) +
  geom_point(aes(
    x = time / 365,
    y = log(value),
    color = Label_name),
    size = 3) +
  rahul_theme +
  theme_white_background +
   rahul_man_figure_theme +
  xlab("Years since Jan 1 1986") +
  ylab("log(Monthly \n Reported Cases)")

Fig_4_Panel_B = Fig_4_Panel_B +
  scale_color_manual(
    name = "",
    values = c("red",
               "blue"),
    labels = c(
      "Simulation Median \n (Shaded Region: \n 95% Quantiles)",
      "Observed")) +
  scale_fill_manual(
    name = "",
    values = c("grey70",
               "NA"),
    labels = c(
      "Simulation Median \n (Shaded Region: \n 95% Quantiles)",
      "Observed"))
```

Figure 4 Panel C

```
rahul_poster_theme = theme(
  axis.title.x = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 21,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 23,
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 21,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 23,
                             face = "bold",
                             color = "black"),
```

```
  axis.text.y = element_text(size = 21,
                             face = "bold",
                             color = "black")
)

model_name_list = c("A_7")
model_label_list = factor(
  c("SIR Cosine No Immigration"))
model_label_list = factor(
  model_label_list,
  levels = c("SIR Cosine No Immigration"))

Csnippet_file_path_list = c(
  "Csnippet_SIR_cosine_model.R")
Num_est_parameters_list = c(7)
data_file_path_list = c(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")

num_years_list = c(2.50)

model_ref_df = data.frame(
  model_name = model_name_list,
  model_label = model_label_list,
  Csnippet_file_path = Csnippet_file_path_list,
  Num_est_parameters = Num_est_parameters_list,
  data_file_path = data_file_path_list,
  num_years = num_years_list,
  stringsAsFactors = FALSE
)


model_index = 1
print(model_index)
```

```
## [1] 1
```

```
model_name = as.character(
  model_name_list[model_index])

single_model_ref_data = filter(
  model_ref_df,
  model_name == !!model_name)

model_label = single_model_ref_data$model_label
Csnippet_file_path =
  single_model_ref_data$Csnippet_file_path

Num_est_parameters =
  single_model_ref_data$Num_est_parameters

data_file_path =
  single_model_ref_data$data_file_path

num_years = single_model_ref_data$num_years
```

```
Rio_data_clean = read.csv(file = data_file_path)

Rio_clean_data = Rio_data_clean
#head(Rio_data_clean)

source(Csnippet_file_path, local = TRUE)

#Set t0
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))

all_combo_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_cases_data.csv"
  )
)

all_R0_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_R0_data.csv"
  )
)

all_combo_S_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_S_over_N_data.csv"
  )
)

all_R_eff_data = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_sim_Reff_data.csv"
  )
)

bio_good_2_LL = read.csv(
  paste0(
```

```
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/",
    model_name,
    "_Model_BP_top_2_LL_all_params_bio_good_2_LL.csv"
  )
)


ML_combo_num = which(
  bio_good_2_LL$LL ==
    max(bio_good_2_LL$LL))

ML_output = filter(
  all_combo_data,
  combo_num == ML_combo_num)

ML_output = dplyr::select(
  ML_output,
  time = time,
  ML_median = sim_data_median,
  ML_high_Q = sim_data_high_Q,
  ML_low_Q = sim_data_low_Q
)
true_data = dplyr::select(
  Rio_clean_data, time = times,
  Observed_Data = Y)

comp_data = join(ML_output, true_data)
```

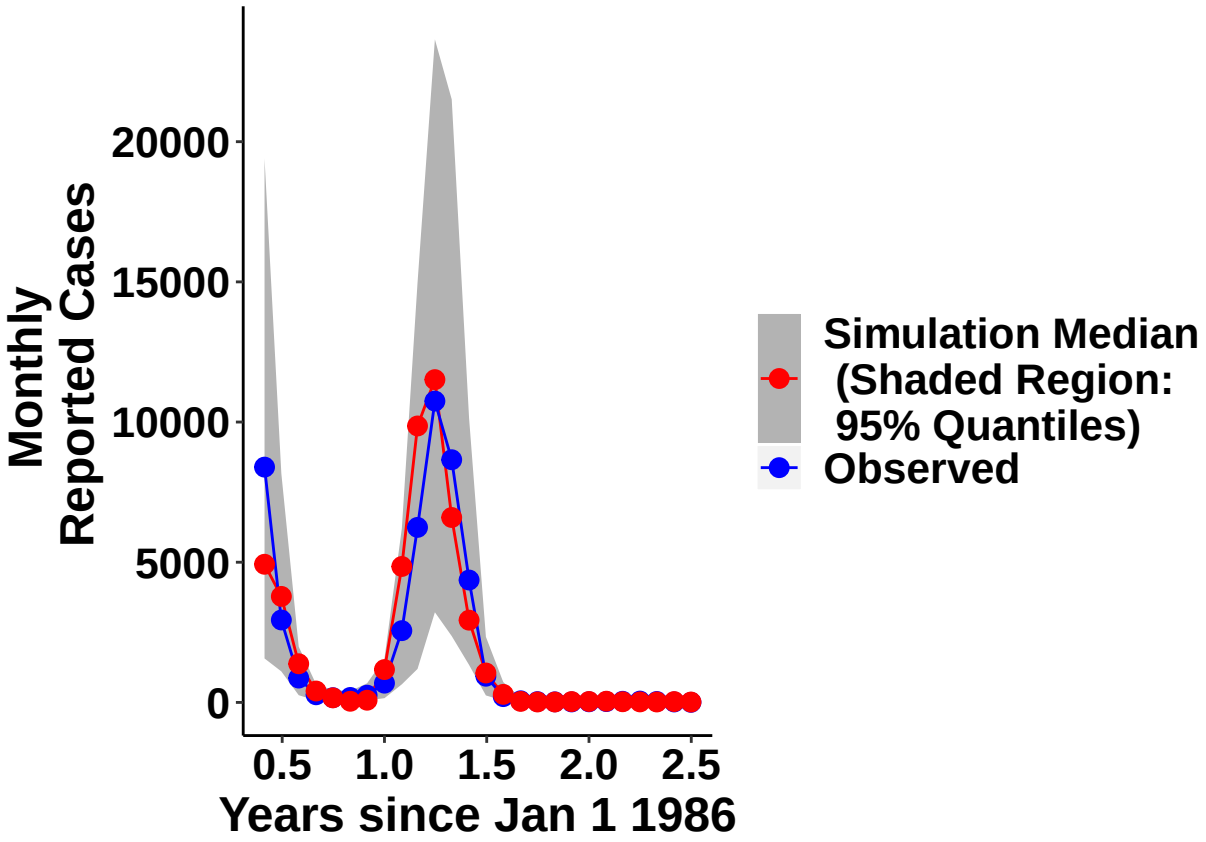
```
## Joining by: time
```

```
comp_data_melt = melt(
  comp_data, id.vars = c(
    "time", "ML_high_Q", "ML_low_Q"))

label_df  =
  data.frame(
    Label_name =
      c("Simulation Median \n (Shaded Region: \n 95% Quantiles)",
        "Observed"),
    variable = c("ML_median",
                 "Observed_Data"))

comp_data_melt_with_label =
  join(comp_data_melt, label_df)
```

```
## Joining by: variable
```

```
Fig_4_Panel_C =
  ggplot(data = comp_data_melt_with_label) +
  geom_ribbon(aes(
    x = time / 365,
    ymin = ML_low_Q,
    ymax = ML_high_Q,
```

```
    fill = Label_name
  )) +
  geom_line(aes(
    x = time / 365,
    y = value,
    color = Label_name
  )) +
  geom_point(aes(
    x = time / 365,
    y = value,
    color = Label_name),
    size = 3) +
  rahul_theme +
  theme_white_background +
   rahul_man_figure_theme +
  xlab("Years since Jan 1 1986") +
  ylab("Monthly \n Reported Cases")

Fig_4_Panel_C = Fig_4_Panel_C +
  scale_color_manual(
    name = "",
    values = c("red",
               "blue"),
    labels = c(
      "Simulation Median \n (Shaded Region: \n 95% Quantiles)",
      "Observed")) +
  scale_fill_manual(
    name = "",
    values = c("grey70",
               "NA"),
    labels = c(
      "Simulation Median \n (Shaded Region: \n 95% Quantiles)",
      "Observed"))

Fig_4_Panel_C
```



Combine panels for Figure 4

```
Panel_A_df = all_R_0_ribbon_df_melt %>%
  dplyr::select(time, ribbon_min = R_0_min,
                ribbon_max = R_0_max,
                value = value,
                Ribbon_label = Ribbon_label,
                variable = variable)

Panel_B_df = comp_data_melt_with_label %>%
  dplyr::select(time = time,
                ribbon_min = ML_low_Q,
                ribbon_max = ML_high_Q,
                value = value,
                variable = variable,
                Ribbon_label = Label_name)
Panel_B_df = Panel_B_df %>%
  mutate(time = time,
         ribbon_min = log(ribbon_min),
         ribbon_max = log(ribbon_max),
         value = log(value))
```

```
Panel_C_df = comp_data_melt_with_label %>%
  dplyr::select(time = time,
                ribbon_min = ML_low_Q,
                ribbon_max = ML_high_Q,
                value = value,
                variable = variable,
                Ribbon_label = Label_name)

Panel_A_df = Panel_A_df %>%
  mutate(panel = "R[0]")

Panel_B_df = Panel_B_df %>%
  mutate(panel = "log (Monthly Cases)")

Panel_C_df = Panel_C_df %>%
  mutate(panel = "Monthly Cases")

Panel_A_df = Panel_A_df %>%
  mutate(Colored_Var = NA,
         Line_Color_Var = "ML_median",
         R_0_var = value)

Panel_B_df = Panel_B_df %>%
  mutate(Colored_Var = value,
         Line_Color_Var = variable,
         R_0_var = NA)

Panel_C_df = Panel_C_df %>%
  mutate(Colored_Var = value,
         Line_Color_Var = variable,
         R_0_var = NA)




Figure_4_combined_df = rbind(
  Panel_A_df, Panel_B_df)
Figure_4_combined_df = rbind(
  Figure_4_combined_df, Panel_C_df)
```

```
Fig_4_combined =
  ggplot(data =
           Figure_4_combined_df) +
  geom_ribbon(aes(
    x = time / 365,
    ymin = ribbon_min,
    ymax = ribbon_max,
    fill = Ribbon_label,
    alpha = 0.75 )) +
  geom_line(aes(
    x = time / 365,
    y = Colored_Var,
```

```
    color = Line_Color_Var
  )) +
  geom_line(aes(
    x = time / 365,
    y = R_0_var
  ), color = "black") +
  geom_point(aes(
    x = time / 365,
    y = Colored_Var,
    color = Line_Color_Var),
    size = 3) +
  geom_point(aes(
    x = time / 365,
    y = R_0_var),
    color = "black",
    size = 3) +
  rahul_theme +
  theme_white_background +
   rahul_man_figure_theme +
  xlab(expression(paste("Years since Jan 1 1986"))) +
  ylab(
    expression(
      paste(
        "                  ",
        R[0],
        "                          Monthly Cases               log(Monthly Cases)"
    )))  +
  facet_wrap(~panel, ncol = 1,
             scales = "free",
             strip.position = "left")

Fig_4_combined = Fig_4_combined +
  scale_color_manual(name = "",
                     values = c("red",
                                "blue"),
                     labels = c("Simulated",
                                "Observed")) +
  scale_fill_manual(name = "",
                     values = c("grey70",
                                 "NA",
                                "red"
                                ),
                     labels = c(" R_0 ",
                                "Observed",
                                "Simulated"))
Fig_4_combined = Fig_4_combined +
  theme(
    aspect.ratio = 1,
    strip.background = element_blank(),
    strip.placement = "outside"
  )
Fig_4_combined = Fig_4_combined +
  rahul_big_panel_theme +
```

```
  theme(strip.text = element_blank()) +
  theme(legend.position = c(.75, .55)) +
  guides(fill=FALSE, alpha = FALSE) +
  theme(legend.key=element_blank()) +
  theme(axis.text.y = element_text(size = 18)) +
  theme(axis.text.x = element_text(size = 18),
        axis.title.x = element_text(size = 20),
        axis.title.y = element_text(size = 20)) +
  theme(legend.text = element_text(size = 15.5,
                                   face = "plain"))
#Fig_4_combined
```

```
tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig4_raw.tiff"
    ),
  height = 10,
  width = 5, res = 700,
  units = "in")#
Fig_4_combined
```

```
## Warning: Removed 26 rows containing missing values (geom_path).
```

```
## Warning: Removed 104 rows containing missing values (geom_path).
```

```
## Warning: Removed 26 rows containing missing values (geom_point).
```

```
## Warning: Removed 104 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

Re-emergence analysis via forward simulation

Simulate re-emergence outbreak in 1991

The following code was run in parallel on a computer cluster:

```
knitr::read_chunk('Man_Fig_5_gardner_code.R')
```

```
#Make one line of heat map for plot of
## re-emergence probability given spark size and time
#Here we give it a single time and iterate
# over multiple spark sizes, param combinatinos (all
# BP params within 2LL of BP MLE), and simulations (N_sim = 100)
rm(list = ls())
source("load_libraries_essential.R")
library(zoo)
library(pomp)
source("rahul_theme.R")
args = commandArgs(trailingOnly = TRUE)

model_name = as.character(args[1])
```

```r
#model_name = "A_3"
print(model_name)

Rio_data_clean = read.csv(
  "../Generated_Data/Rio_DENV1_Data_3_75_years_clean.csv")
head(Rio_data_clean)
Rio_clean_data = Rio_data_clean
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))
load(file = "../Down_Data/denguerj1986-2017.RData")
Rio_city_DENV1_clean = data.frame(
  Y = as.matrix(dengue.ts),
  Date = as.Date(as.yearmon(time(dengue.ts))))
Rio_city_DENV1_clean = filter(
  Rio_city_DENV1_clean,
  Date >= "1986-05-01")

head(Rio_city_DENV1_clean)
Rio_city_DENV1_clean$Date =
  Rio_city_DENV1_clean$Date %m+% months(1)

head(Rio_city_DENV1_clean)
##Calculate Re-Emergence Probability

Population_Rio_2000 = 5857904 #Census
Population_Rio_1991 = 5480768# Census:
Two_hour_segments_in_year = 365 * 12
time_between_census_dates = 2000 * 365 - 1991 * 365
human_pop_growth_rate =
  (1 / time_between_census_dates) *
  log(Population_Rio_2000 / Population_Rio_1991)
human_pop_growth_rate

#Source Csnippets
source(file = "Csnippet_SIR_cosine_model.R")


all_combos = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/combined_",
    model_name,
    "_profile_data_directory_with_mean_R_0.csv"
  )
)

MLE_params = filter(all_combos, LL == max(LL))

bio_good_2_LL = filter(
  all_combos, LL > max(all_combos$LL) - 2)

within_20_LL = filter(
```

```
  all_combos, LL > max(all_combos$LL) - 20)

param_index = as.numeric(
  Sys.getenv("MOAB_JOBARRAYINDEX"))
print("param_index")
print(param_index)
gardner_max_jobs = 500
group_size = ceiling(nrow(
  bio_good_2_LL) / gardner_max_jobs)
start_index = (param_index - 1) * group_size + 1
end_index = param_index * group_size
Num_mif_runs_per_start = 5
param_data_subset =
  bio_good_2_LL[start_index:end_index,]

start_date = as.Date("1986-01-01") +
  min(Rio_clean_data$times)
long_re_emergence_time_series_2 =
  seq.Date(from = start_date, by = "month", length = 600)

years_in_long_re_emergence_time_series_2 =
  year(long_re_emergence_time_series_2)
long_re_emergence_time_series_2 =
  as.numeric(long_re_emergence_time_series_2 -  as.Date("1986-01-01"))
years_in_long_re_emergence_time_series_2 =
  years_in_long_re_emergence_time_series_2 - 1986

year_table = data.frame(time =
                          long_re_emergence_time_series_2,
                        year =
                          years_in_long_re_emergence_time_series_2)

long_covar_start = min(long_re_emergence_time_series_2) -
  3 * 30

#End covariates 1 week after end of data
log_covar_end = max(long_re_emergence_time_series_2) +
  3 * 30

# Set covariate time step (Default is 1 hour,
# want it smaller than dt to be safe)
long_covar_dt = 1 / 24
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))

long_covar_times = seq(from = long_covar_start,
                       to = log_covar_end,
                       by = long_covar_dt)

long_covar = covariate_table(
  t = long_covar_times,
  s = periodic.bspline.basis(
    t,
```

```
    nbasis = 3,
    degree = 3,
    period = 365,
    name = '%d'
  ),
  times = "t"
)

time_seq = long_covar@times
covar_table = as.data.frame(t(long_covar@table))
head(covar_table)
covar_table_with_time = mutate(covar_table,
                               time = long_covar_times)
covar_at_obs_times = filter(
  covar_table_with_time,
  time %in% long_re_emergence_time_series_2)


table(years_in_long_re_emergence_time_series_2)


re_emergence_threshold = 1


a = Rio_city_DENV1_clean
a$Year = year(a$Date)
total_dengue_cases_by_year = aggregate(
  Y ~ Year, a, FUN = sum)
Re_emergnce_95_observed_cases = filter(
  total_dengue_cases_by_year,
  Year == 1995)$Y
order_of_magnitude_epi_threshold = exp(
  trunc(log(Re_emergnce_95_observed_cases)))

ptm = proc.time()


spark_size_list = c(20)
spark_year_list = c(1990)

relevant_column_data = dplyr::select(
  all_combos, -one_of("seed", "LL",
                      "Profile_Type","R_naught"))

relevant_colnames = colnames(relevant_column_data)

re_emergence_prob_data_all_combos =
  data.frame(matrix(nrow = 0, ncol = 27))

colnames(re_emergence_prob_data_all_combos) = c(
  "spark_size",
  "total_re_emergence_prob_1_year",
  "total_re_emergence_prob_2_year",
```

```
  "total_re_emergence_prob_3_year",
  "spark_year",
  "R_naught",
  relevant_colnames,
  "t_stop_immigration",
  "spark",
  "spark_time_start",
  "spark_time_end"
)
for (combo_index in seq(
  1:nrow(param_data_subset))) {
  print("combo_index")
  print(combo_index)
  combo_params = param_data_subset[combo_index,]
  combo_LL = combo_params$LL
  R_naught = combo_params$R_naught
  combo_params = dplyr::select(combo_params,
                               -one_of("seed", "LL",
                                       "Profile_Type",
                                       "R_naught"))
  combo_params$r = human_pop_growth_rate
  combo_params$r = human_pop_growth_rate
  combo_params$t_stop_immigration =
    max(Rio_data_clean$times)

  re_emergence_prob_data_all_years_single_combo =
    data.frame(matrix(nrow = 0, ncol = 5))
  colnames(re_emergence_prob_data_all_years_single_combo) =
    c(
    "spark_size ",
    "total_re_emergence_prob_1_year",
    "total_re_emergence_prob_2_year",
    "total_re_emergence_prob_3_year",
    "spark_year"
  )
  total_lik = sum(exp(combo_LL))


  for (spark_year_index in seq(
    1:length(spark_year_list))) {
    print(spark_year_index)
    spark_year =
      spark_year_list[spark_year_index]

    total_re_emergence_prob_across_all_comb_and_sim_1_year =
      as.numeric(vector(length = length(spark_size_list)))
    total_re_emergence_prob_across_all_comb_and_sim_2_year =
      as.numeric(vector(length = length(spark_size_list)))
    total_re_emergence_prob_across_all_comb_and_sim_3_year =
      as.numeric(vector(length = length(spark_size_list)))
    for (spark_size_index in seq(1:length(spark_size_list))) {
      spark_size = spark_size_list[spark_size_index]
      print("spark_size_index = ")
```

```
print(spark_size_index)


combo_params$spark = spark_size

spark_time = as.numeric(
  as.Date(paste0(spark_year, "-01-01")) -
    as.Date("1986-01-01"))
spark_time_end = as.numeric(
  as.Date(paste0(spark_year, "-02-01")) -
    as.Date("1986-01-01"))

combo_params$spark_time_start =
  spark_time
combo_params$spark_time_end =
  spark_time_end

sim_data_sample_param =
  simulate(
    nsim = 100,
    seed = 12345,
    times = long_re_emergence_time_series_2,
    t0 = t0,
    rprocess = euler(
      rproc_re_emerge_spark_month_stop_immigration,
      delta.t = 1
  ),
  params = combo_params,
  paramnames =
    paramnames_spark_month_stop_immigration,
  statenames =
    statenames_spark_month,
  obsnames = obsnames,
  accumvars = acumvarnames,
  covar = long_covar,
  rinit = init_spark_month,
  rmeas = rmeas,
  partrans = par_trans,
  format = "data.frame",
  cdir = '/scratch/rsubramanian/tempdir'
)
#head(sim_data)

Beta_t =
  combo_params$Beta_0*(
    1 +
      combo_params$delta*sin(
        combo_params$omega*covar_at_obs_times$time +
          combo_params$phi));

R_0 = (
  Beta_t / (
    combo_params$gamma + combo_params$mu_H
```

```
    )) * (combo_params$mu_EI /(
      combo_params$mu_EI + combo_params$mu_H))

sim_data_sample_param =
  join(sim_data_sample_param,
       year_table, by = "time")

year_of_nearest_obs_greater_than_spark_time =
  min(year_table[year_table$time >
                   spark_time, ]$year)

second_year_of_re_emgergence_epi =
  year_of_nearest_obs_greater_than_spark_time + 1

third_year_of_re_emgergence_epi =
  year_of_nearest_obs_greater_than_spark_time + 2

all_sim_probs_unweighted_1_year =
  as.numeric(vector(length = 1))
all_sim_probs_unweighted_2_year =
  as.numeric(vector(length = 1))
all_sim_probs_unweighted_3_year =
  as.numeric(vector(length = 1))

for (s in seq(
  1:length(
    unique(
      sim_data_sample_param$.id)))) {
  single_sim_data =
    filter(sim_data_sample_param, .id == s)

  sim_data_year_of_re_emergence =
    filter(single_sim_data,
           year ==
             year_of_nearest_obs_greater_than_spark_time)
  sim_data_second_year_of_re_emergence =
    filter(single_sim_data,
           year ==
             second_year_of_re_emgergence_epi)
  sim_data_third_year_of_re_emergence =
    filter(single_sim_data,
           year == third_year_of_re_emgergence_epi)

  epi_start_time_1_year =
    min(sim_data_year_of_re_emergence$time)
  epi_start_time_2_year =
    min(sim_data_second_year_of_re_emergence$time)
  epi_start_time_3_year =
    min(sim_data_third_year_of_re_emergence$time)

  sim_data_at_epi_start_1_year =
    filter(sim_data_year_of_re_emergence,
           time == epi_start_time_1_year)
```

```
    sim_data_at_epi_start_2_year = filter(
      sim_data_second_year_of_re_emergence,
      time == epi_start_time_2_year)
    sim_data_at_epi_start_3_year = filter(
      sim_data_third_year_of_re_emergence,
      time == epi_start_time_3_year)

    epi_end_time_1_year = max(
      sim_data_year_of_re_emergence$time)
    epi_end_time_2_years = max(
      sim_data_second_year_of_re_emergence$time)
    epi_end_time_3_years = max(
      sim_data_third_year_of_re_emergence$time)

    sim_data_at_epi_end_1_year =
      filter(
        sim_data_year_of_re_emergence,
        time == epi_end_time_1_year)
    sim_data_at_epi_end_2_years =
      filter(sim_data_second_year_of_re_emergence,
             time == epi_end_time_2_years)
    sim_data_at_epi_end_3_years =
      filter(sim_data_third_year_of_re_emergence,
             time == epi_end_time_3_years)

    dS_over_epidemic_1_year =
      sim_data_at_epi_end_1_year$S -
      sim_data_at_epi_start_1_year$S
    dS_over_epidemic_2_year =
      sim_data_at_epi_end_2_years$S -
      sim_data_at_epi_start_2_year$S
    dS_over_epidemic_3_year =
      sim_data_at_epi_end_3_years$S -
      sim_data_at_epi_start_3_year$S


    single_sim_spark_time_spark_size_status_1_year =
      as.numeric(dS_over_epidemic_1_year < 0)
    single_sim_spark_time_spark_size_status_2_year =
      as.numeric(dS_over_epidemic_2_year < 0)
    single_sim_spark_time_spark_size_status_3_year =
      as.numeric(dS_over_epidemic_3_year < 0)

    all_sim_probs_unweighted_1_year =
      all_sim_probs_unweighted_1_year +
      single_sim_spark_time_spark_size_status_1_year
    all_sim_probs_unweighted_2_year =
      all_sim_probs_unweighted_2_year +
      single_sim_spark_time_spark_size_status_2_year
    all_sim_probs_unweighted_3_year =
      all_sim_probs_unweighted_3_year +
      single_sim_spark_time_spark_size_status_3_year
  }
```

```
    all_sim_probs_unweighted_1_year =
      all_sim_probs_unweighted_1_year /
      length(unique(sim_data_sample_param$.id))
    all_sim_probs_unweighted_2_year =
      all_sim_probs_unweighted_2_year /
      length(unique(sim_data_sample_param$.id))
    all_sim_probs_unweighted_3_year =
      all_sim_probs_unweighted_3_year /
      length(unique(sim_data_sample_param$.id))

    #Multiply by weight (function of L of parameters)
    all_sim_probs_weighted_1_year =
      all_sim_probs_unweighted_1_year
    all_sim_probs_weighted_2_year =
      all_sim_probs_unweighted_2_year
    all_sim_probs_weighted_3_year =
      all_sim_probs_unweighted_3_year


    ## Add to total probability
    ## accross all combinations and simulations
    total_re_emergence_prob_across_all_comb_and_sim_1_year[spark_size_index] =
      total_re_emergence_prob_across_all_comb_and_sim_1_year[spark_size_index] +
      all_sim_probs_weighted_1_year
    total_re_emergence_prob_across_all_comb_and_sim_2_year[spark_size_index] =
      total_re_emergence_prob_across_all_comb_and_sim_2_year[spark_size_index] +
      all_sim_probs_weighted_2_year
    total_re_emergence_prob_across_all_comb_and_sim_3_year[spark_size_index] =
      total_re_emergence_prob_across_all_comb_and_sim_3_year[spark_size_index] +
      all_sim_probs_weighted_3_year


  }
  re_emergence_prob_data = data.frame(
    spark_size = spark_size_list,
    total_re_emergence_prob_1_year =
      total_re_emergence_prob_across_all_comb_and_sim_1_year,
    total_re_emergence_prob_2_year =
      total_re_emergence_prob_across_all_comb_and_sim_2_year,
    total_re_emergence_prob_3_year =
      total_re_emergence_prob_across_all_comb_and_sim_3_year,
    spark_year = rep(
      spark_year,
      length = length(
        total_re_emergence_prob_across_all_comb_and_sim_1_year)
    )
  )
  re_emergence_prob_data_all_years_single_combo =
    rbind(
      re_emergence_prob_data_all_years_single_combo,
      re_emergence_prob_data)

}
```

```
  re_emergence_prob_data_all_years_single_combo$R_naught =
    R_naught
  combo_params$R_naught =
    R_naught
  re_emergence_prob_data_all_years_single_combo =
    join(
      re_emergence_prob_data_all_years_single_combo,
      combo_params)
  re_emergence_prob_data_all_combos =
    rbind(
      re_emergence_prob_data_all_combos,
      re_emergence_prob_data_all_years_single_combo
  )
}
proc.time() - ptm




write.csv(
  re_emergence_prob_data_all_combos,
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/stoch_re_emerge_test/",
    model_name,
    "_re_mergence_spark_probability_data_subset_",
    param_index,
    ".csv"
  ),
  row.names = FALSE
)
```

Script for running code on computer cluster (Uchicago BSD Gardner)

```
#knitr::read_chunk('A_7_Man_Fig_5_re_emerge_calc.pbs')
cat A_7_Man_Fig_5_re_emerge_calc.pbs
```

```
#MSUB -N arrayJob
#MSUB -l nodes=1:ppn=1,mem=2gb,walltime=48:00:00
#MSUB -t [1-457]

cd /scratch/rsubramanian/Spring_2019/riodengue/Rio_State_Data_Fitting/Code
echo $MOAB_JOBARRAYINDEX

module load gcc/6.2.0
module load R/3.5.0

R CMD BATCH --vanilla '--args A_7' Man_Fig_5_gardner_code.R O/out.$MOAB_JOBARRAYINDEX
```

Collect output

```
rm(list = ls())
source("load_libraries_essential.R")
library(zoo)
library(pomp)
source("rahul_theme.R")
args = commandArgs(trailingOnly = TRUE)

#model_name = as.character(args[1])
model_name = "A_7"
print(model_name)

Rio_data_clean = read.csv(
  "../Generated_Data/Rio_DENV1_Data_2_25_years_clean.csv")
Rio_clean_data = Rio_data_clean
t0 = as.numeric(as.Date("1986/05/01") -
                  as.Date("1986/01/01"))
load(
  file = "../Down_Data/denguerj1986-1996.RData")

Rio_city_DENV1_clean = data.frame(
  Y = as.matrix(dengue.ts),
  Date = as.Date(as.yearmon(time(dengue.ts))))

Rio_city_DENV1_clean = filter(
  Rio_city_DENV1_clean,
  Date >= "1986-05-01")

head(Rio_city_DENV1_clean)
Rio_city_DENV1_clean$Date =
  Rio_city_DENV1_clean$Date %m+% months(1)

Population_Rio_2000 = 5857904 #Census
Population_Rio_1991 = 5480768# Census:
Two_hour_segments_in_year = 365 * 12
time_between_census_dates = 2000 * 365 - 1991 * 365
human_pop_growth_rate = (1 / time_between_census_dates) *
  log(Population_Rio_2000 / Population_Rio_1991)
human_pop_growth_rate

#Source Csnippets
source(file = "Csnippet_SIR_cosine_model.R")


all_combos = read.csv(
  paste0("../Generated_Data/Profiles/", model_name,
         "_Model/combined_",model_name,
         "_profile_data_directory_with_mean_R_0.csv"))

MLE_params = filter(all_combos, LL == max(LL))

bio_good_2_LL = filter(all_combos, LL >
```

```
                        max(all_combos$LL) - 2 )

within_20_LL = filter(all_combos, LL >
                        max(all_combos$LL) - 20 )
test_param_index = 1
single_test_subset_output = read.csv(
  paste0(
    "../Generated_Data/Profiles/",
    model_name,
    "_Model/stoch_re_emerge_test/",
    model_name,
    "_re_mergence_spark_probability_data_subset_",
    test_param_index,
    ".csv"
  ))
all_param_spark_data = data.frame(
  matrix(nrow = 0,
         ncol = ncol(single_test_subset_output)))

colnames(all_param_spark_data) =
  colnames(single_test_subset_output)
num_param_combinations = 457
for(param_index in c(seq(1:23),
                     seq(from = 25,to = 168),
                     seq(from = 170,to = 450),
                     seq(from = 452,
                         to = num_param_combinations))){

  gardner_max_jobs = 500
  group_size =
    ceiling(nrow(bio_good_2_LL) / gardner_max_jobs)
  start_index = (param_index - 1) * group_size + 1
  end_index = param_index * group_size
  Num_mif_runs_per_start = 5
  param_data_subset =
    bio_good_2_LL[start_index:end_index, ]

  single_subset_output = read.csv(
    paste0(
      "../Generated_Data/Profiles/",
      model_name,
      "_Model/stoch_re_emerge_test/",
      model_name,
      "_re_mergence_spark_probability_data_subset_",
      param_index,
      ".csv"
    ))
  if(
    sum(
      is.na(
        single_subset_output$total_re_emergence_prob_1_year)) >
    0) {
    print(paste0(
```

```
      "Param set fail at ",
      param_index))
  }

  all_param_spark_data =
    rbind(all_param_spark_data,
          single_subset_output)

}

## Save data (FILE IS LARGE SO COMMENTED OUT)
# write.csv(all_param_spark_data, file =
#  paste0("../Generated_Data/Profiles/",
#           model_name,
#           "_Model/stoch_re_emerge_test/",
#           model_name,
#           "_re_mergence_spark_prob_all_params.csv"
# ))
```

Figure 5

Figure 5 Panel A

```
source("load_libraries_essential.R")
source("rahul_theme.R")
library(stringr)
library(gridExtra)
library(zoo)
load("../Generated_Data/Skip_Data/nCritics.Rdata")
#head(nCritics)

skip_raw_data =
  as.numeric(as.character(nCritics))
skip_data = as.data.frame(as.matrix(nCritics))

#dim(nCritics) #18 (Reporitng rate) x 11 (delta)
# x 491 (R_0 value)
# Row name: Reporting rate (18 from 1% to 50%)


#reporting_rates = strsplit(reporting_rate, "%")
rep_rate_header = str_split(row.names(nCritics),
                            pattern = "%", n = Inf,
                            simplify = TRUE)
delta_col_header = str_split(colnames(nCritics),
                             pattern = "_", n = Inf,
                             simplify = TRUE)
reporting_rate = as.numeric(rep_rate_header[,2])/100
delta_val = as.numeric(delta_col_header[,2])
```

```
R_0_col_header = str_split(names(nCritics[1,1,]),
                           pattern = "_", n = Inf,
                             simplify = TRUE)
R_0_skip_val = as.numeric(R_0_col_header[,2])

model_name = "A_7"

## R_naught_act_data
profile_data_with_R_naught_act = read.csv(
  file = paste0(
    "../Generated_Data/Profiles/",
    model_name,  "_Model/combined_",
    model_name,
    "_profile_data_directory_with_mean_R_0.csv"))

MLE_with_R_naught_act = filter(
  profile_data_with_R_naught_act,
  LL == max(LL))

bio_good_2_LL_with_R_naught = read.csv(
  file = paste0(
    "../Generated_Data/Profiles/", model_name,
    "_Model/combined_", model_name,
    "_bio_good_2_LL_param_list.csv"))

A_7_MLE_R_naught_act = MLE_with_R_naught_act$R_naught
A_7_min_R_naught_act = min(
  bio_good_2_LL_with_R_naught$R_naught)

A_7_max_R_naught_act = max(
  bio_good_2_LL_with_R_naught$R_naught)

A_7_bio_good_2_LL = read.csv(
  paste0("../Generated_Data/Profiles/",
         model_name, "_Model/",model_name,
         "_Model_BP_top_2_LL_all_params_bio_good_2_LL.csv"))

A_7_bio_good_2_LL$R_naught_theo =
  A_7_bio_good_2_LL$Beta_0/(
    A_7_bio_good_2_LL$gamma + A_7_bio_good_2_LL$mu_H)

A_7_bio_good_2_LL$nearest_skip_rho = 0
A_7_bio_good_2_LL$nearest_skip_R_naught = 0
A_7_bio_good_2_LL$nearest_skip_delta = 0
A_7_bio_good_2_LL$skips = -1
A_7_bio_good_2_LL$nearest_skip_delta_index = NA
A_7_bio_good_2_LL$nearest_skip_rho_index = NA
for(param_index in seq(1,
                       nrow(A_7_bio_good_2_LL))){
  load(
    "../Generated_Data/Skip_Data/nCritics_detailedRepRate_From2to5.Rdata")
```

```
#head(nCritics_detailedRepRate_From2to5)


#dim(nCritics) #18 (Reporitng rate) x
# 11 (delta) x 491 (R_0 value)
# Row name: Reporting rate (18 from 1% to 50%)


rep_rate_header_det = str_split(row.names(
  nCritics_detailedRepRate_From2to5),
  pattern = "%", n = Inf,
  simplify = TRUE)
delta_col_header_det = str_split(colnames(
  nCritics_detailedRepRate_From2to5),
  pattern = "_", n = Inf,
  simplify = TRUE)

reporting_rate_det = as.numeric(
  rep_rate_header_det[,2])/100
delta_val_det = as.numeric(
  delta_col_header_det[,2])


R_0_col_header_det = str_split(names(
  nCritics_detailedRepRate_From2to5[1,1,]),
  pattern = "_", n = Inf,
  simplify = TRUE)

R_0_skip_val_det = as.numeric(
  R_0_col_header_det[,2])

#Get R_naught ref on skip plot
A_7_bio_good_2_LL$nearest_skip_R_naught_index[param_index] =
  which.min(
    abs(
      R_0_skip_val_det -
        A_7_bio_good_2_LL$R_naught_theo[param_index] ))
A_7_bio_good_2_LL$nearest_skip_R_naught[param_index] =
  R_0_skip_val_det[
    A_7_bio_good_2_LL$nearest_skip_R_naught_index[param_index]]

#Get rho ref on skip plot
A_7_bio_good_2_LL$nearest_skip_rho_index[param_index] =
  which.min(
    abs(
      reporting_rate_det -
        A_7_bio_good_2_LL$rho[param_index] ))
A_7_bio_good_2_LL$nearest_skip_rho[param_index] =
  reporting_rate_det[
    A_7_bio_good_2_LL$nearest_skip_rho_index[param_index]]

#Get delta ref on skip plot
A_7_bio_good_2_LL$nearest_skip_delta_index[param_index] =
```

```
    which.min(
      abs(
        delta_val_det -
          A_7_bio_good_2_LL$delta[param_index] ))
  A_7_bio_good_2_LL$nearest_skip_delta[param_index] =
    delta_val[
      A_7_bio_good_2_LL$nearest_skip_delta_index[param_index]]
  A_7_bio_good_2_LL$skips[param_index] =
    nCritics_detailedRepRate_From2to5[
      A_7_bio_good_2_LL$nearest_skip_rho_index[param_index],
      A_7_bio_good_2_LL$nearest_skip_delta_index[param_index],
      A_7_bio_good_2_LL$nearest_skip_R_naught_index[param_index]]
}

relevant_skip_plot_data = dplyr::select(
  A_7_bio_good_2_LL,
  "R[0]" = nearest_skip_R_naught,
  skips,
  rho = nearest_skip_rho,
  delta = nearest_skip_delta)
min(A_7_bio_good_2_LL$skips, na.rm = TRUE)
```

```
## [1] 32.5
```

```
relevant_skip_plot_data_melt = melt(
  relevant_skip_plot_data, id.vars = c("skips"))
relevant_skip_plot_data_melt$value = signif(
  relevant_skip_plot_data_melt$value,
  digits = 3)

relevant_skip_plot_data_melt$skip_category = cut(
  relevant_skip_plot_data_melt$skips,
  breaks = c(0,1,100,101),
  include.lowest = TRUE)

relevant_skip_plot_data$r0 =
  relevant_skip_plot_data$`R[0]`


relevant_skip_plot_data$rho = as.factor(
  as.character(relevant_skip_plot_data$rho))

relevant_skip_plot_data_delta_07 = filter(
  relevant_skip_plot_data, delta == 0.7)

Fig_5_A_plot_data = relevant_skip_plot_data_delta_07

Fig_5_A_plot_data$plot_var = ""
test = ggplot(data = Fig_5_A_plot_data,
              aes(x = r0, y = skips)) +
  geom_point(color = 'red', size = 3,
             shape =  "circle open") +
  facet_wrap(~plot_var) +
  theme_white_background +
```

```
  labs(x = expression(R[0])) +
  labs(y = expression(n[c])) +
  labs(y = expression(paste("Number of skips (", n[c], ")"))) +
  theme(strip.background = element_rect(colour="white", fill="white"))


### Add line
load(
  "../Generated_Data/Data_for_Manuscript_Figures/skip_data_rho_3.RData"
  )

skip_df_rho_3$rho_lab = "\rho~=~0.03"
rho_3_line_df = skip_df_rho_3

rho_3_line_df$r0 = as.numeric(as.character(
  rho_3_line_df$r0))

rho_3_line_df$Num_Skips = as.numeric(as.character(
  rho_3_line_df$Num_Skips))

rho_3_line_df = filter(rho_3_line_df, delta == 0.7)
rho_3_line_df$rho = as.factor(as.character(
  rho_3_line_df$rho))
rho_3_line_df$'R[0]' = rho_3_line_df$r0
rho_3_line_df = rho_3_line_df %>%
  dplyr::select('R[0]' ='R[0]',
                skips = Num_Skips,
                rho = rho, delta = delta,
                r0 = r0)
rho_3_line_df$plot_var = ""
rho_3_line_df$Line_Color = "Show_Line"

Fig_5_A_plot_data$Line_Color = "No_Line"


Fig_5_A_plot_data_subset = Fig_5_A_plot_data %>%
  dplyr::select(var_value =  'R[0]',
                LL = skips)
Fig_5_A_plot_data_subset$Profile_Var =
  'R[0]'
Fig_5_A_plot_data_subset$plot_var_label =
  ' R[0]'
Fig_5_A_plot_data_subset$Metric = "Skips"
Fig_5_A_plot_data_subset$low_bound = -1
Fig_5_A_combined_data = Fig_5_A_plot_data_subset

Fig_5_A_combined_data$plot_var_label = factor(
  Fig_5_A_combined_data$plot_var_label,
  levels = c(" R[0]"))

rho_3_line_df = rho_3_line_df %>%
  dplyr::select(var_value =  'R[0]',
                LL = skips) %>%
```

```
  mutate(Profile_Var = 'R[0]',
         plot_var_label = ' R[0]',
         )
rho_3_line_df = na.omit(rho_3_line_df)
```

Plot Figure 5 Panel A

```
rahul_big_panel_theme = theme(
  axis.title.x = element_text(size = 14,
                              face = "bold",
                              color = "black"),
  axis.text.x = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  axis.title.y = element_text(size = 14,
                              face = "bold",
                              color = "black"),
  legend.title = element_text(size = 14,
                              face = "bold",
                              color = "black"),
  legend.text = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  axis.text.y = element_text(size = 12,
                             face = "bold",
                             color = "black"),
  plot.margin = unit(c(.5,.5,.5,.5), "cm"),
  legend.background = element_rect(
    fill = "transparent"),
  legend.box.margin = unit(c(.5,.5,.5,.5), "cm")
)

Fig_5_A_plot = ggplot()  +
  geom_point(
    data = Fig_5_A_combined_data,
    aes(x = var_value,
        y = LL, color = Metric,
        shape = Metric)) +
  scale_linetype_manual(
    values = c("blank", "solid")) +
  rahul_man_figure_theme +
  rahul_big_panel_theme  +
  theme_white_background  +
  geom_hline(
    data = Fig_5_A_combined_data,
    aes(yintercept = low_bound),
    color = 'white', linetype = 'blank') +
  geom_line(
    data = rho_3_line_df,
    aes(x = var_value, y = LL),
    color = 'black', size = 1.0) +
  scale_x_continuous(
    breaks = scales::pretty_breaks(n = 3)) +
  theme(
```

```
    aspect.ratio = 1,
    strip.background = element_blank(),
  ) +
  theme(legend.position = "None") +
  scale_color_manual(
    values = c("black", "red", "black", "white"),
    limits = c("LL", "Skips", "Show_Line", "No_Line")) +
  scale_shape_manual(values = c(16, 1)) +
  scale_y_continuous(
    breaks = scales::pretty_breaks(n = 4))  +
  ylab(
    expression(paste(
      " Number of skips ", (n[c])))) +
  theme(axis.text.y = element_text(size = 16),
        axis.text.x = element_text(size = 16),
        axis.title.x = element_text(size = 15),
        strip.text = element_blank()
        ) +
    xlab(expression(
      paste(
        " Reproductive Number ", (R[0]))))
Fig_5_A_plot
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```

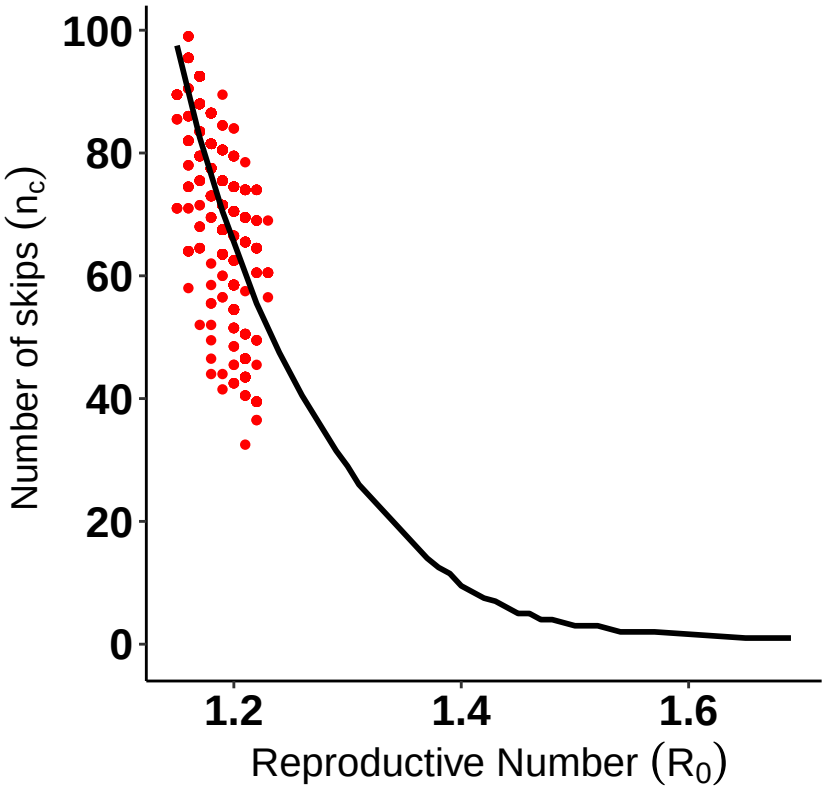


Figure 5 Panel B

```
spark_data_90_only = filter(
  all_param_spark_data,
  spark_year == 1990)

spark_90_size_20 = filter(
  spark_data_90_only,
  spark_size == 20)
no_na_20 = na.omit(spark_90_size_20)

ML_df = MLE_params
ML_df$r = unique(no_na_20$r)
ML_with_re_emerge_prob =
  join(ML_df, no_na_20,
       by = c("sigma_P", "gamma",
              "phi", "sigma_M",
              "rho", "Beta_0",
              "delta", "mu_H",
              "N_0", "I_0",
              "R_0", "C_0",
              "r", "omega",
              "epsilon", "R_naught"))
Fig_5_B_plot = ggplot(data = no_na_20,
           aes(x =sigma_P,
               y = total_re_emergence_prob_1_year)) +
  geom_point(size = 3) +
  xlab(expression(paste(" Process Noise ",
                        (sigma[P])))) +
  ylab(expression(paste("Re-Emergence Probability in ",
                        1990))) +
  geom_point(data = ML_with_re_emerge_prob,
             aes(x = sigma_P,
                 y = total_re_emergence_prob_1_year),
             color = 'red', fill = "NA",
             size = 5, shape = 21, stroke = 3) +
  rahul_man_figure_theme +
  rahul_big_panel_theme  +
  theme_white_background  +
  theme(aspect.ratio = 1,
        axis.text.y = element_text(size = 16),
        axis.text.x = element_text(size = 16),
        axis.title.y = element_text(
          face = "plain"),
        strip.text = element_blank()
        )
```

Make Combined Plot of Figure 5

```
tiff(
  paste0(
    "../Figures/Manuscript_Figures/TIFF_Files/Fig5_raw.tiff"),
```

```
  height = 5, width = 10,
  res = 500, units = "in")
print(grid.arrange(
  Fig_5_A_plot, Fig_5_B_plot, ncol = 2))
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```
dev.off()
```

```
## pdf
##   2
```