Analyze CMV transient infection stochastic model batch simulations

Contents

```
### The transient infection data - used for viral loads
load("../data-analysis/data/cmv_blip_data.RData")

### The transient infection duration data - used for consecutive swabs
load("../data-analysis/data/duration_data.RData")


### The batch simulation data stored on dataverse.

#This is a somewhat large data frame so this code is set up to download it to
#model-data/ if it is not already there.
if("stochastic_sim_I_latent.Rdata" %in% list.files("model-data/")){
  load("model-data/stochastic_sim_I_latent.Rdata")
} else{
  load(url("https://dataverse.harvard.edu/api/access/datafile/3008959"))
  save(stochastic_sim_latency_I, file = "model-data/stochastic_sim_I_latent.Rdata")

}
```

Summary of primary infection model fits

These are the results from fitting the CMV infection model (deterministic ode model) to the primary infection data on just the exponential viral expansion portion of the data. That analysis and data is available on github. This results file shows the fitted parameter for infection start time (relative to first positive), beta x initial susceptible cells (4e8) x 1000, and the subsequent calculation of R0.

```
#these are the results from fitting the exponential growth portion of the model (deterministic ode) to
base_model_fits <- read.csv(text = getURL("https://raw.githubusercontent.com/bryanmayer/CMV-Primary-Inf

r0_tab = summary(base_model_fits$R0) %>%
  tidy() %>%
  xtable(caption = "Summary of R0 fits from primary infection model (JID paper)")

#rescaling the parameter in this results data to beta
beta_tab = summary(1e12 * base_model_fits$beta/4e8/1e3) %>%
  tidy() %>%
  xtable(caption = "Summary of beta fits (times 1e-12) from primary infection model (JID paper)")


print(r0_tab)
```

minimum	q1	median	mean	q3	maximum
1.08	1.53	1.63	1.72	1.80	3.11

Table 1: Summary of R0 fits from primary infection model (JID paper)

```r
print(beta_tab)
```

minimum	q1	median	mean	q3	maximum
3.19	4.49	4.80	5.07	5.31	9.13

Table 2: Summary of beta fits (times 1e-12) from primary infection model (JID paper)

Summary of simulated transient infections

```r
sim_blips_lag = filter(stochastic_sim_latency_I, last_V == 0)
sim_blips_lag_obs = filter(stochastic_sim_latency_I, last_V == 0 & consecutive_blips > 0)
sim_blips_lag_obs$infection = sim_blips_lag_obs$max_I > 1

#log10(range(sim_blips_lag$max_I))

sim_blip_rate_lag = stochastic_sim_latency_I %>% group_by(R0, initI) %>%
  summarize(
    total_sim = n(),
    total_blip = sum(last_V == 0),
    blip_prob = total_blip/total_sim,
    total_trans_infection = sum(last_V == 0 & max_I > 1),
    trans_infection_prob = total_trans_infection/total_sim
  )

#among the observable viral loads
sim_blip_rate_lag_obs = subset(stochastic_sim_latency_I, consecutive_blips > 0) %>% group_by(R0, initI)
  summarize(
    total_sim = n(),
    total_blip = sum(last_V == 0),
    blip_prob = sum(last_V == 0)/n()
  )

features_blips_lag = sim_blips_lag %>% group_by(R0, initI) %>%
  dplyr::summarize(
    total_blips = n(),
    minDuration = min(max_time),
    maxDuration = max(max_time),
    medianDuration = median(max_time),
    medianMax = median(max_V),
    medianSampleV = median(random_blip_sample, na.rm = T)
  )

## Save data for the python code

write_csv(sim_blip_rate_lag_obs, "model-data/contour_plot_data.csv")
```

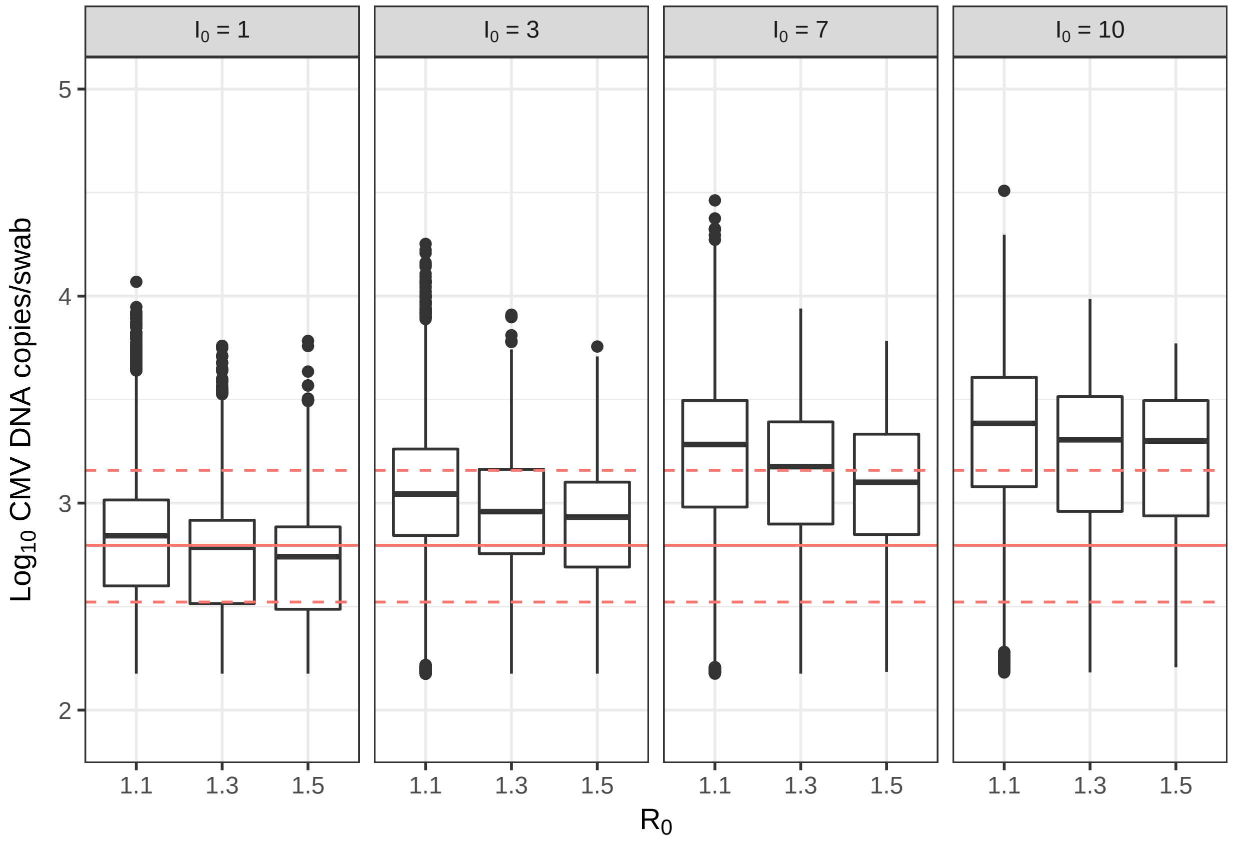
Transient infection viral loads

Comparing the randomly sampled viral concentrations from the model simulations to the values that are actually observed in the data.

```r
blip_size_data = select(subset(CMVblipdata, count > 0), count)
```

```
## Adding missing grouping variables: `PatientID2`
```

```r
# subset the simulation datafor plotting
plot_data = subset(sim_blips_lag_obs,
                   round(R0, 2) %in% r0_plot_range &
                     (initI %in% initI_plot_range))

plot_data$R0 = round(plot_data$R0, 2)

# this is the observed data reorganized with common variables names for with against simulation data
bind_data = data.frame(
  R0 = "data",
  initI = rep(initI_plot_range, each = length(blip_size_data$count)),
  max_V = 10^blip_size_data$count,
  random_blip_sample = 10^blip_size_data$count,
  max_V = 10^blip_size_data$count
)

data_size_summary = bind_data %>% group_by(R0, initI) %>%
  summarize(
    median_conc = median(log10(random_blip_sample)),
    lowerQR = quantile(log10(random_blip_sample), 0.25),
    upperQR = quantile(log10(random_blip_sample), 0.75)
  )

#label the initI
plot_data$icat = factor(plot_data$initI, levels = c(1,3,7,10),
                        labels =Ilabels)
data_size_summary$icat = factor(data_size_summary$initI, levels = c(1,3,7,10),
                                labels =Ilabels)

## make plot
line_col = "#F8766D"
ggplot(data = plot_data,
                      aes(x = factor(R0), y = log10(random_blip_sample))) +
  geom_boxplot(fill = "white") +
  geom_hline(data = data_size_summary, aes(yintercept = median_conc), colour= line_col) +
  geom_hline(data = data_size_summary, aes(yintercept = lowerQR), linetype = "dashed", colour= line_col
  geom_hline(data = data_size_summary, aes(yintercept = upperQR), linetype = "dashed", colour= line_col
  scale_x_discrete(expression(paste(R[0])), drop = F) +
  scale_y_continuous(cmv_title, limits = c(1.9, 5), breaks = 1:6) +
  facet_wrap(~icat, nrow = 1, labeller = label_parsed)
```



```
data_size_summary[1,] %>% ungroup() %>%
  select(median_conc, lowerQR, upperQR) %>%
  xtable(caption = "Summary of observed transient infection viral load data.") %>% print()
```

median_conc	lowerQR	upperQR
2.80	2.52	3.16

Table 3: Summary of observed transient infection viral load data.

Comparing duration between simulations and observed.

Visually comparing distributions of observed consecutive positive swabs

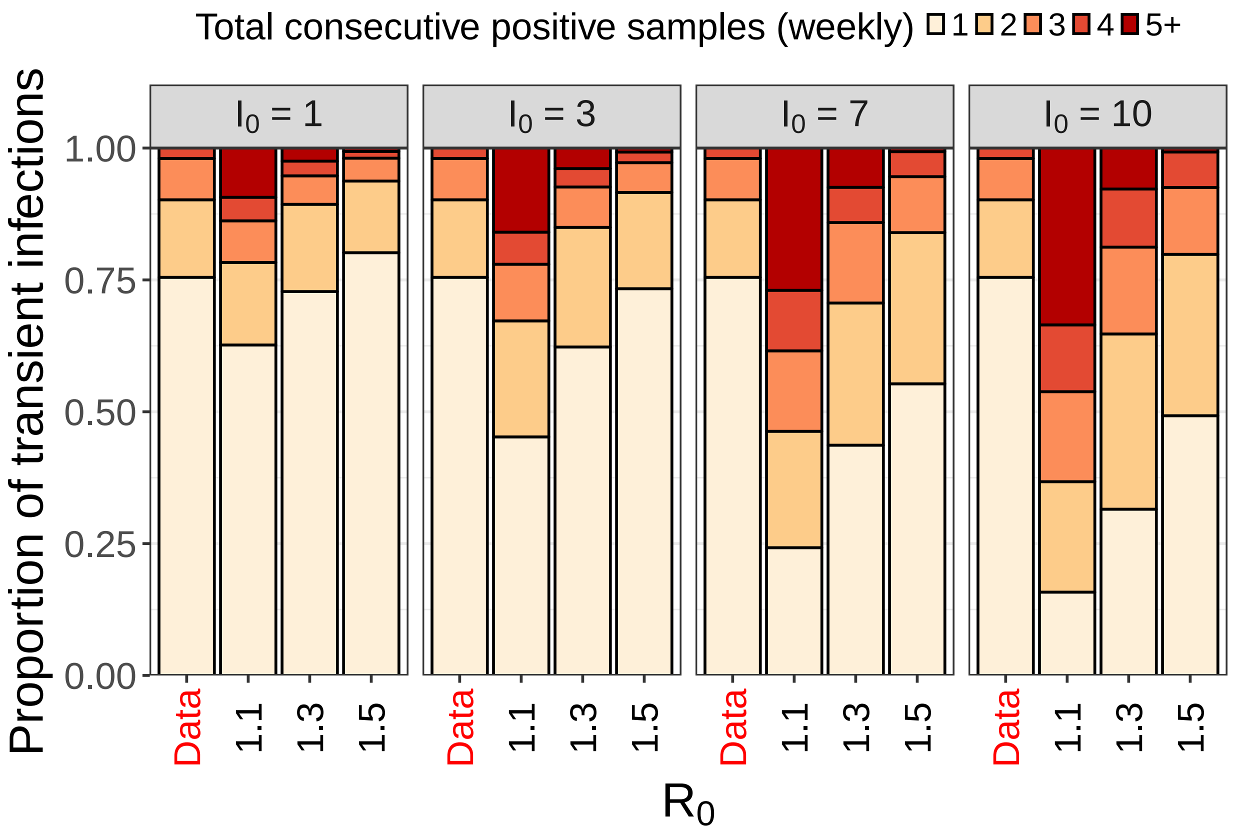
To compare duration, we looked at how many consecutive swabs would be sampled with a random weekly swabbing routine. While this is a coarse proxy of duration, it matches how the observed data was generated where the beginning and end of infection episodes were not observed.

```
## Duration (consecutive positives swabs) categories for duration plot
blip_breaks = c(0,1,2,3,4, Inf)
blip_labels = c(1,2,3,4, "5+")

#### To include the observed data on the plots, initI and R0 variables need to be assigned to the obser

## processing the duration calculations from the observed infections
duration_data = blip_data_duration_complete %>% group_by(total_blips) %>%
  summarize(
    total = n(),
```

```r
    prop = n()/dim(blip_data_duration_complete)[1])

# for combining with simulated duration
bind_duration_data = data.frame(
  R0 = "Data",
  total_blips = as.character(duration_data$total_blips),
  initI = rep(initI_plot_range, each = length(duration_data$prop)),
  prop = duration_data$prop
)

bind_duration_data$total_blips = factor(bind_duration_data$total_blips, levels = 1:5,
                                        labels = blip_labels)


# calculation proportions of consecutive swab distributions from simulated data
sim_duration_data = filter(sim_blips_lag_obs,
                           round(R0, 2) %in% r0_plot_range &
                             (initI %in% initI_plot_range)) %>%
  group_by(R0, initI) %>%
  mutate(
    total = n(),
    total_blips_raw = consecutive_blips, #as a check
    total_blips = cut(consecutive_blips, breaks = blip_breaks, labels = blip_labels)) %>%
  ungroup() %>%
  mutate(
    R0 = factor(R0, levels = c("Data", as.character(r0_plot_range)), ordered = T)
    ) %>%
  group_by(R0, initI, total_blips) %>%
  summarize(prop = n()/unique(total))

# combined simulations and observed data
sim_duration_plot_data = bind_rows(sim_duration_data, bind_duration_data)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```r
# processing for plot
sim_duration_plot_data$R0 = factor(sim_duration_plot_data$R0,
                                   levels = c("Data", as.character(r0_plot_range)),
                                   ordered = T)

#Ilabels made above
sim_duration_plot_data$icat = factor(sim_duration_plot_data$initI, levels = initI_plot_range,
                               labels =c( Ilabels))

#this manipulation is to make the order how I want it (1->5+ from bottom up)
sim_duration_plot_data$total_blips = factor(sim_duration_plot_data$total_blips,
                                levels = rev(sort(unique(sim_duration_plot_data$total_blips))))

ggplot(data = sim_duration_plot_data,
                              aes(x = R0, y = prop, fill = total_blips)) +
  geom_bar(stat = "identity", colour = "black") +
  scale_x_discrete(expression(paste(R[0])), drop = F) +
  facet_wrap(~icat, nrow = 1, labeller = label_parsed) +
  scale_fill_brewer("Total consecutive positive samples (weekly)", palette="OrRd",
                    breaks = rev(levels(sim_duration_plot_data$total_blips)),
                    direction = -1) +
  scale_y_continuous("Proportion of transient infections",  breaks = 0:4/4, expand = c(0,0)) +
  theme(
    legend.position = "top",
    legend.text = element_text(size = 12),
    legend.title = element_text(size = 14),
    legend.key.size = unit(0.5, "lines"),
    text = element_text(size = 18),
    axis.text.x = element_text(angle = 90, vjust = 0.5, colour = c("red", rep("black",12)))
    )
```

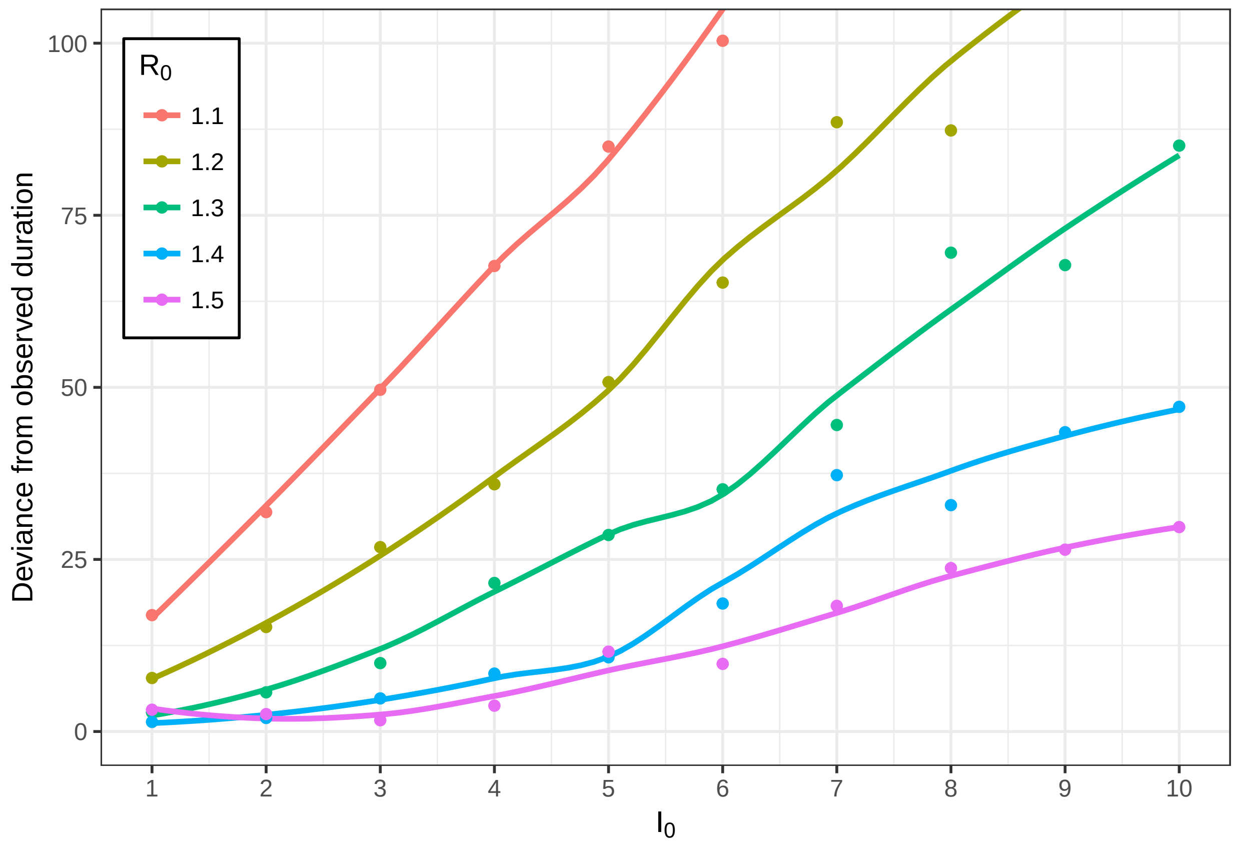


Statistical comparisons between distributions of consecutive positive swabs

Because this is a multinomial distribution, it is not necessarily easy to visually compare each individual category between the observed data and the simulated data. We used the G-test deviance statistics as an aggregate measure of comparison. Loess curves through the deviance across R0 and initial I depict the general trend: generally the least deviance was observed at lower initial infected cells.

```
########### Goodness of fit duration ###############
g_test = function(obs, e_prob){
  obs[obs == 0] <- 1
  expected = e_prob * sum(obs)
  expected[expected == 0] <- 1

  e_prob[e_prob == 0]<-0.001
  2 * sum(obs * log(obs/(sum(obs) * e_prob)))
```

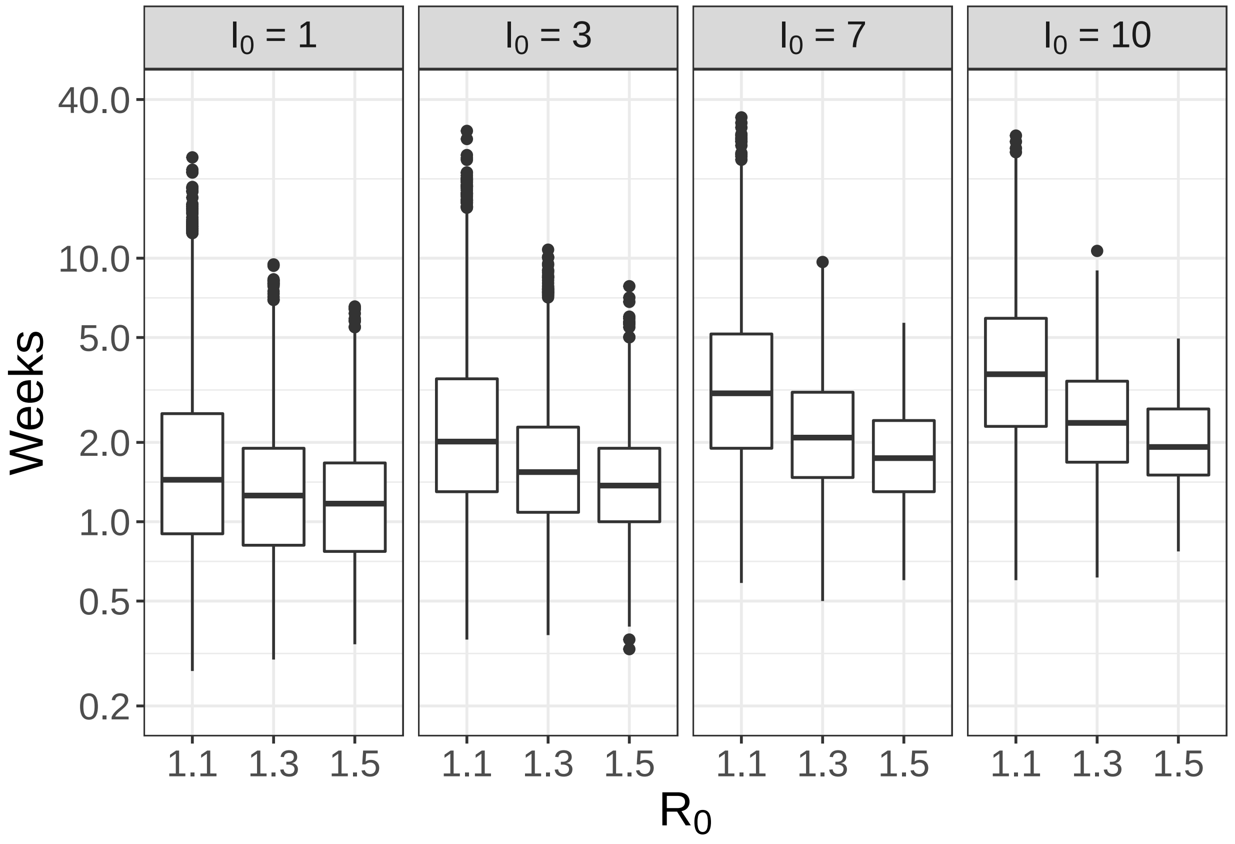
```r
  #sum((obs - expected)^2/expected)
}

#across all of the R0 values now, not just 3
full_sim_duration_data = subset(sim_blips_lag_obs, initI <= 10) %>%
  group_by(R0, initI) %>%
  mutate(
    total = n(), total_blips_raw = consecutive_blips, #as a check
    total_blips = cut(consecutive_blips, breaks = blip_breaks, labels = blip_labels)) %>%
  ungroup() %>%
  group_by(R0, initI, total_blips) %>%
  summarize(prop = n()/unique(total)) %>%
  group_by(R0, initI) %>%
  tidyr::complete(total_blips, fill = list(prop = 0)) %>%
  full_join(
    duration_data %>% rename(prop_data = prop) %>%
      mutate(total_blips = factor(total_blips, levels = 1:5, labels = blip_labels))
    , by = c("total_blips")
  )

#this assigns the total observed transient infections from NA to 0 (there were none observed)
full_sim_duration_data$total[full_sim_duration_data$total_blips == "5+"] = 0

#calculate the gstat deviance for each level
fitted_duration = full_sim_duration_data %>%
  group_by(R0, initI) %>%
  summarize(g_stat = g_test(total, prop))

ggplot(data = subset(fitted_duration,
                     round(R0, 2) %in% c(1.1, 1.2, 1.3, 1.4, 1.5)),
       aes(x = initI, y = g_stat, colour = factor(R0))) +
  geom_smooth(method = "loess", se= F) +
  ylab("Deviance from observed duration") +
  coord_cartesian(ylim = c(0, 100)) +
  geom_point() +
  scale_x_continuous(expression(paste(I[0])), breaks = 1:10) +
  scale_color_discrete(expression(paste(R[0]))) +
    theme(legend.position = c(0,1),
          legend.justification = c(-0.2,1.1),
          legend.background = element_rect(colour = "black"))
```



Some additional figures

Boxplots of simulated duration.

```r
#plot_data was created in the viral load chunk

plot_data$wks =  plot_data$max_time/7

ggplot(data = plot_data, aes(x = factor(R0), y = wks)) +
  geom_boxplot(fill = "white") +
  scale_x_discrete(expression(paste(R[0])), drop = F) +
  scale_y_log10("Weeks", breaks = c(0.2, 0.5, 1, 2, 5, 10, 40), limits = c(0.2, 40)) +
  facet_wrap(~icat, nrow = 1, labeller = label_parsed) +
  theme(text = element_text(size = 18))
```



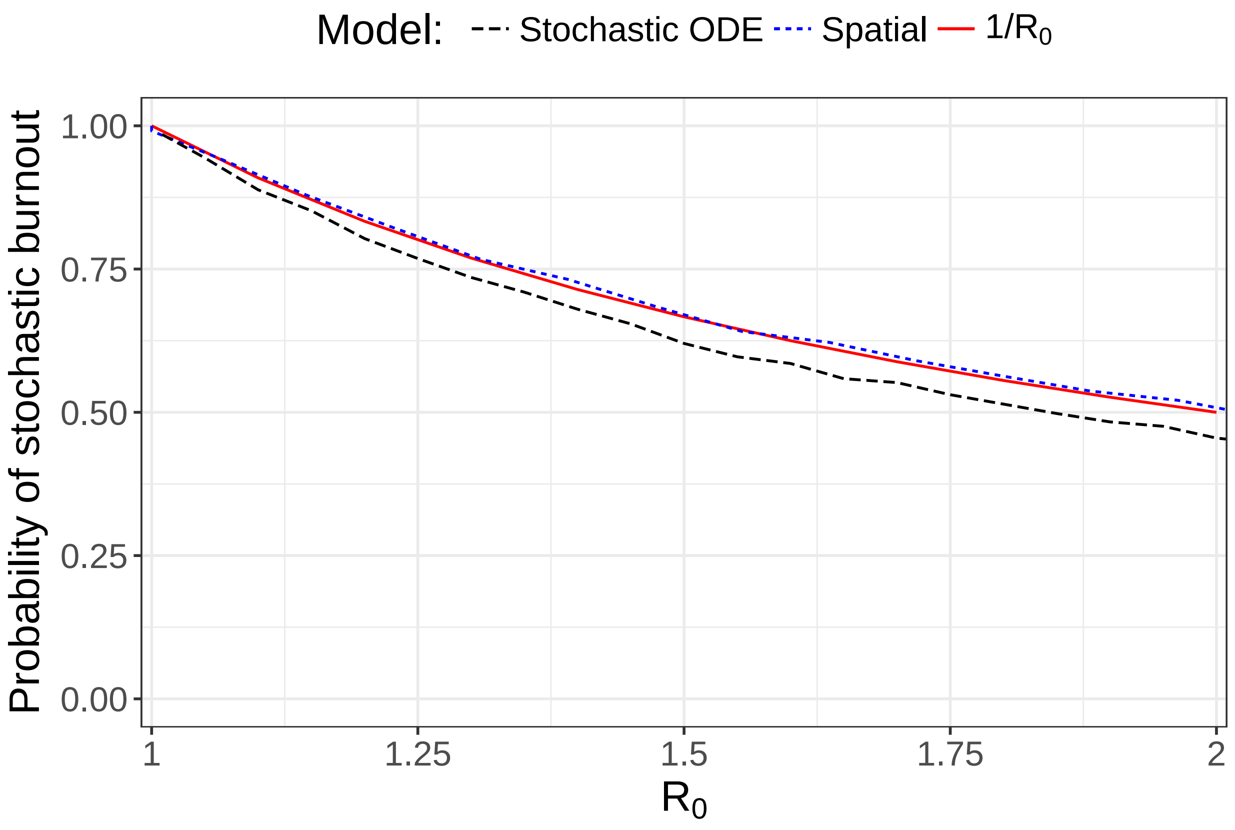
Comparing the stochastic burnout probability between the stochastic ODE to the agent based spatial model. The estimated probability is calculated as 1-1/R0. The ODE shows some deviance because it assumes there is no depletion of the susceptible cell population.

```
abm_data = readr::read_csv("model-data/abm_data.csv") %>%
  select(R0, `percent runs ongoing`) %>%
  rename(blip_prob = `percent runs ongoing`) %>%
  mutate(blip_prob = 100 - blip_prob, model = "Spatial")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Trial = col_integer(),
##   delta = col_integer(),
##   lambda = col_integer(),
##   `5%` = col_integer(),
##   `25%` = col_integer(),
##   `50%` = col_integer(),
##   `75%` = col_integer(),
##   `95%` = col_integer(),
##   `tot at 200` = col_integer(),
##   `inf at 250` = col_integer(),
##   `tot at 250` = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

```
#combining the two simulated data sets, all using initial I = 1
prob_data = subset(sim_blip_rate_lag, initI == 1) %>%
  select(R0, blip_prob) %>%
  mutate(blip_prob = blip_prob * 100, model = "Stochastic ODE") %>%
  bind_rows(abm_data)

#calculated burnout rate
est_data = data.frame(R0 = seq(1, 2, 0.1), blip_prob = 100 * 1/seq(1, 2, 0.1), model = "calc")

#plot labels
prob_data$R02 = ifelse(prob_data$model == "Spatial", prob_data$R0 * (4.5/5.5), prob_data$R0)
est_data$R02 = est_data$R0

ggplot(data = prob_data, aes(x = R02, y = blip_prob/100, colour = model, linetype = model)) +
  geom_line(data = est_data) +
  geom_line() +
  scale_x_continuous(expression(paste(R[0])), breaks = 1 + 0:4/4, limits = c(1, 2.05),
                     labels = as.character(1 + 0:4/4),
                     oob = squish, expand = c(0, 0.01)) +
  scale_y_continuous("Probability of stochastic burnout", limits = c(0, 1), breaks = 0:4/4) +
  coord_cartesian(xlim =  c(1, 2.0))  +
  scale_colour_manual("Model: ", values = c("Red", "Blue", "Black"),
                      breaks = c(unique(prob_data$model), "calc"),
                      labels = c(unique(prob_data$model), expression(paste("1/R"[0])))) +
  scale_linetype_discrete("Model: ", breaks = c(unique(prob_data$model), "calc"),
                          labels = c(unique(prob_data$model), expression(paste("1/R"[0])))) +
  theme(text = element_text(size = 16), legend.position = "top")
```


Model:
Stochastic ODE
Spatial
1/R0
Probability of stochastic burnout
R0