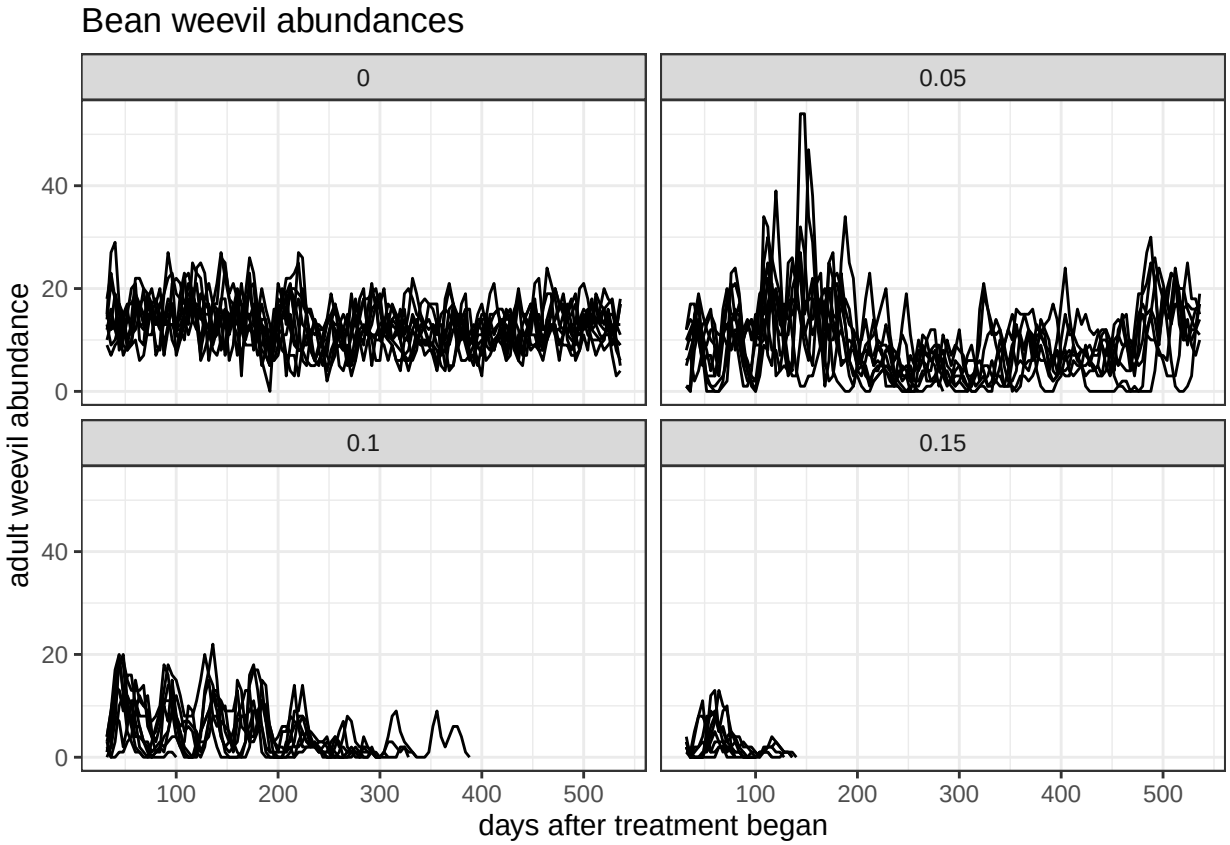
Deterministic and stochastic effects of food quality on extinction time

David Nguyen

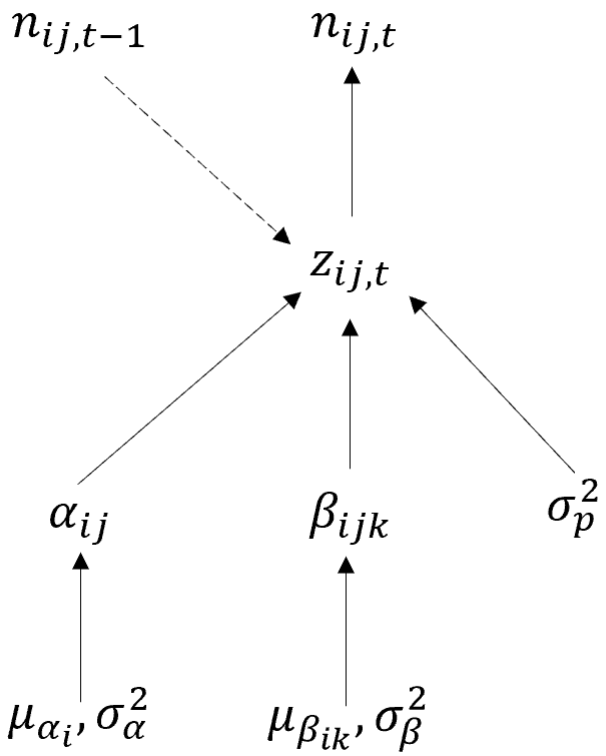
2021-04-15

Data



- Response: Census of adult weevils every 4 days
- Treatment: food quality (0, 5, 10 , 15 % peanut shell)
- Experimental design: 10 replicates per treatment level = 40 populationsx
  - Pure bean pellets and 4 adult weevils added every 4 days up to day 28
  - Day 32 treatment begins. Food added every 7 days.

Model



Definitions

- $n_{ij,t}$: the adult weevil count for treatment i, replicate j, and time t
- $n_{ij,t-1}$: the adult weevil count for treatment i, replicate j, and time $t-1$ lag
- $z_{ij,t}$: The expected number of adult weevils for treatment i, replicate j, and time t
- α_{ij} : treatment specific population growth rate in replicate j
- β_{ij} : effect of previous population size at $t-1$ on expected number of adult weevils at time t for treatment i in replicate j
- $\mu_{\alpha,i}$: the mean population growth rate for treatment i
- μ_{β_i} : the mean effect of previous population size at $t-1$ on expected number of adult weevils at time t for treatment i
- σ_α^2 : variance of distribution of population growth rates
- σ_β^2 : variance of distribution of density dependence magnitudes (maybe should make it specific to lag k?)

Gompertz process model

$$\begin{aligned} N_t &= N_{t-1} \exp(\alpha + \beta \ln(N_{t-1})) \\ &= g(\alpha, \beta, n_{t-1}) \end{aligned}$$

But, note that when $N_{t-1} = 0$, then all subsequent abundances will be zero.

The log-gompertz model (might?) be more useful for estimation since we no longer multiply by N_{t-1}.

$$\begin{aligned} X_t &= \alpha + \beta X_{t-1} \\ &= g(\alpha, \beta, n_{t-1}) \end{aligned}$$

Note that β_1 in the log-gompertz is technically $(1 + \beta_1)$ for the model on arithmetic (not-logged) scale.

Full model

$$\begin{aligned} [\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu_\alpha}, \boldsymbol{\mu_\beta}, Z, \sigma_p^2, \sigma_\alpha^2, \sigma_\beta^2 | \boldsymbol{N}] \propto \prod_{i=1}^{4} \prod_{j=1}^{J_{ij}} \prod_{t=1}^{T_{ij}} & \text{Poisson}[n_{ij,t} | \exp(Z_{ij,t})] \\ & \text{normal}(Z_{ij,t} | g(\alpha_i, \beta_{ik}, n_{ij,t-1}, \ldots, n_{ij,t-1}), \sigma_p^2) \\ & \text{N}[\alpha_{ij} | \mu_{\alpha,i}, \sigma_\alpha^2] \\ & \text{N}[\beta_{ij} | \mu_{\beta_i}, \sigma_\beta^2] \\ & \text{Unif}[\mu_{\alpha,i} | 0, 3] \\ & \text{Unif}[\mu_{\beta_{ik}} | -1.1, 1.1] \\ & \text{inv.gamma}[\sigma_\alpha^2 | 0.001, 0.001] \\ & \text{inv.gamma}[\sigma_\beta^2 | 0.001, 0.001] \\ & \text{inv.gamma}[\sigma_p^2 | 0.001, 0.001] \end{aligned}$$

Model fitting: Direct density dependence

This is the full model presented earlier with direct density dependence only ($k = 1$).

```
set.seed(123)
data <-
  list(
  n = as.numeric(weevil$N_lag0),
  lag1 = as.numeric(ifelse(weevil$N_lag1 == 0, 0.001, weevil$N_lag1)),
  lnlag1 = log(ifelse(weevil$N_lag1 == 0, 0.001, weevil$N_lag1)),
  replicate = as.numeric(weevil$replicate), # nx1 vector of replicate numbers for each observation
  sreplicate = as.numeric(1:10), # replicate numbers
  treatment = as.numeric(weevil$ftreatment), # nx1 vector of treatment levels for each observation
  streatment = as.numeric(1:4), # treatment levels
  start_index = as.numeric(trt_index$init_index), # 1st obs. number for trt 1:4
  end_index = as.numeric(trt_index$end_index) # last obs. number for trt 1:4
)

inits <-
  list(
    list(mu_a = rep(1, 4), mu_b = rep(0.5, 4), tau_a = 1/5, tau_b = 1/5, tau_p = 0.001),
    list(mu_a = rep(0.5, 4), mu_b = rep(0.2, 4), tau_a = 1/10, tau_b = 1/10, tau_p = 0.01),
    list(mu_a = rep(1.5, 4), mu_b = rep(0.9, 4), tau_a = 1/100, tau_b = 1/100, tau_p = 0.005)
```

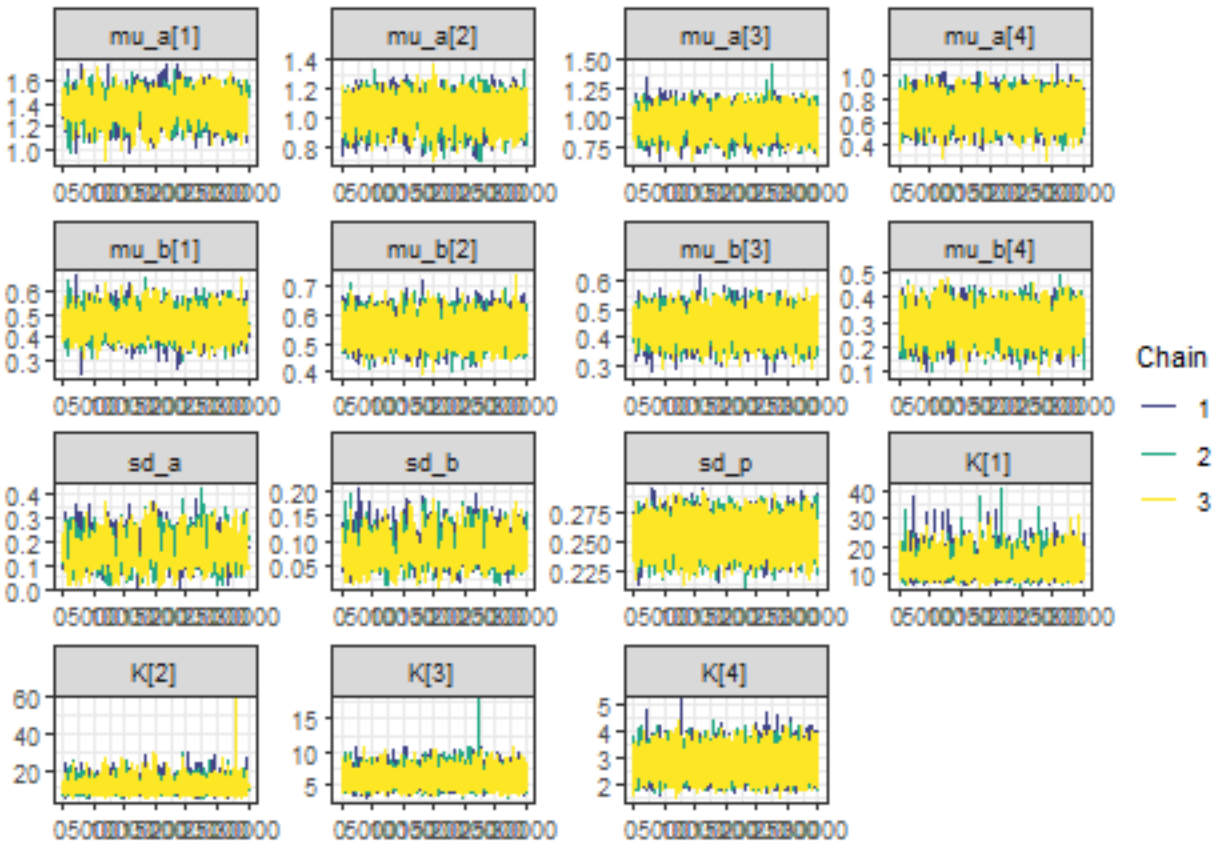
```
  )

variablenames <- c("mu_a","mu_b", "sd_a","sd_b", "sd_p", "K", #"n_new",
                   "ts_cv", "ts_new_cv", "indicator_cv",
                   "ts_max", "ts_new_max", "indicator_max")
modelstring="
model{
  #likelihood
  for (t in 1:length(n)){
    z[t] ~ dnorm(a[treatment[t], replicate[t]] + b[treatment[t], replicate[t]]*lnlag1[t], tau_p)
    n[t] ~ dpois(exp(z[t]))

    # posterior predictive dist. Uses new values for a and b for each replicate
    z_new[t] ~ dnorm(a_new[treatment[t], replicate[t]] + b_new[treatment[t], replicate[t]]*lnlag1[t], ta
    n_new[t] ~ dpois(exp(z_new[t]))
  }

  # draw a and b's
  # store in a 4 x 10 matrix (rows are treatment levels, cols are replicate number)
  for (i in 1:length(streatment)) {
  for (j in 1:length(sreplicate)) {
      a[i, j] ~ dnorm(mu_a[i], tau_a)
      b[i, j] ~ dnorm(mu_b[i], tau_b)

      # get new a and b for replicates for posterior predictive dist'n
      a_new[i, j] ~ dnorm(mu_a[i], tau_a)
      b_new[i, j] ~ dnorm(mu_b[i], tau_b)
  }
  }
  # Draw for mean of treatment specific a and b
  for (i in 1:length(streatment)) {
      mu_a[i] ~ dunif(0,3)
      mu_b[i] ~ dunif(-1.1, 1.1)
  }

  # Draw for precisions
  tau_a ~ dgamma(0.001, 0.001)
  tau_b ~ dgamma(0.001, 0.001)
  tau_p ~ dgamma(0.001, 0.001) # same process error across treatments

  # convert precision to sd
  sd_a <- 1/sqrt(tau_a)
  sd_b <- 1/sqrt(tau_b)
  sd_p <- 1/sqrt(tau_p)

  # Compute carrying capacity
  for (i in 1:length(streatment)) {
    K[i] <- exp(mu_a[i]/(1-mu_b[i]))
  }

  # compute test statistics for each treatment group
  for (i in 1:length(streatment)) {
    # coefficient of variation
```

```
    ts_cv[i] <- mean(n[start_index[i]:end_index[i]]) / sd(n[start_index[i]:end_index[i]])
    ts_new_cv[i] <- mean(n_new[start_index[i]:end_index[i]]) / sd(n_new[start_index[i]:end_index[i]])
    indicator_cv[i] <- ifelse(ts_new_cv[i] > ts_cv[i], 1, 0)

    # Max abundance
    ts_max[i] <- max(n[start_index[i]:end_index[i]])
    ts_new_max[i] <- max(n_new[start_index[i]:end_index[i]])
    indicator_max[i] <- ifelse(ts_new_max[i] > ts_max[i], 1, 0)
  }
}"

jm = jags.model(textConnection(modelstring),data,inits,n.chains=length(inits),n.adapt=1000) #create mod
update(jm, n.iter=10000) #burn-in
cs = coda.samples(jm,variable.names = variablenames,n.iter=30000) #generate chains
save(cs, file = "chains/weevil_random_trt_proc_error.RData")
```

Checking convergence

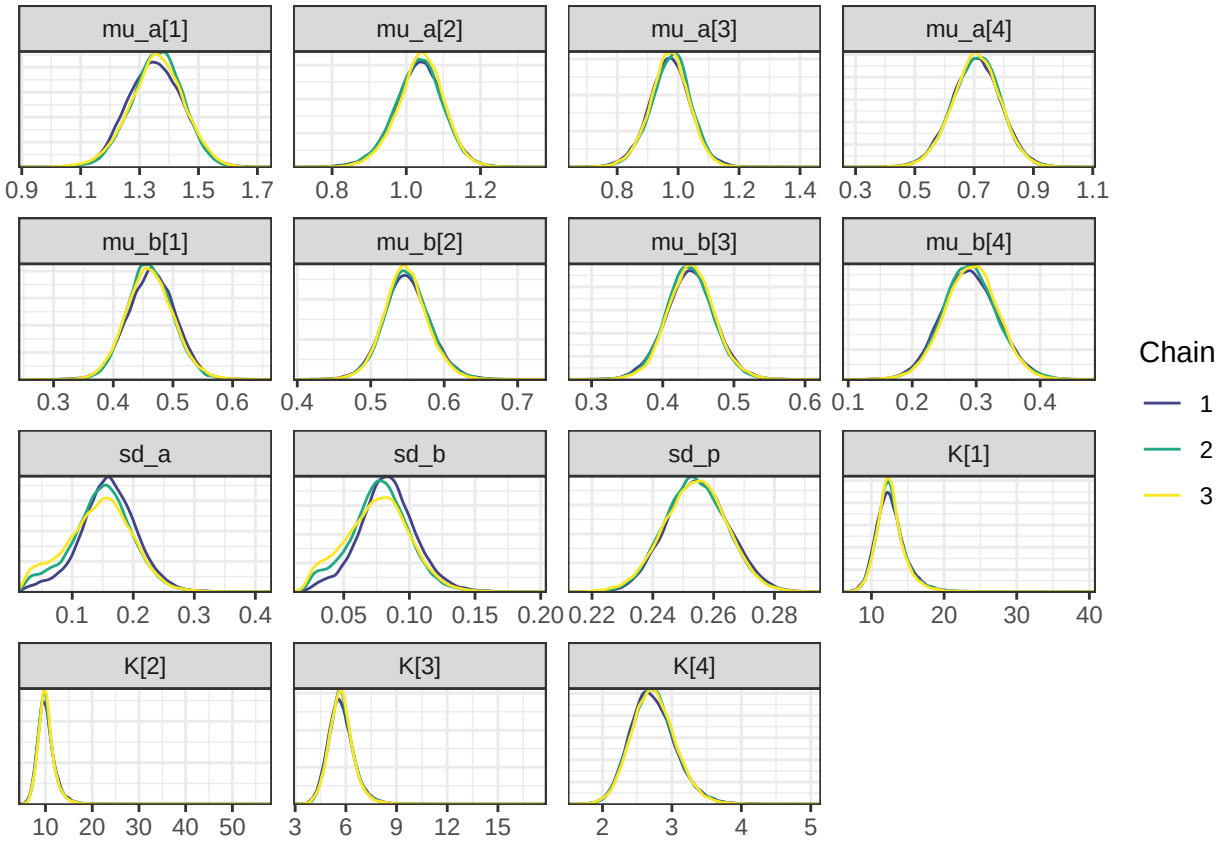
trace plots

```
load("chains/weevil_random_trt_proc_error.RData")
color_scheme_set("viridis")
bayesplot::mcmc_trace(cs, pars = vars(starts_with(c("mu", "sd", "K"))))
```



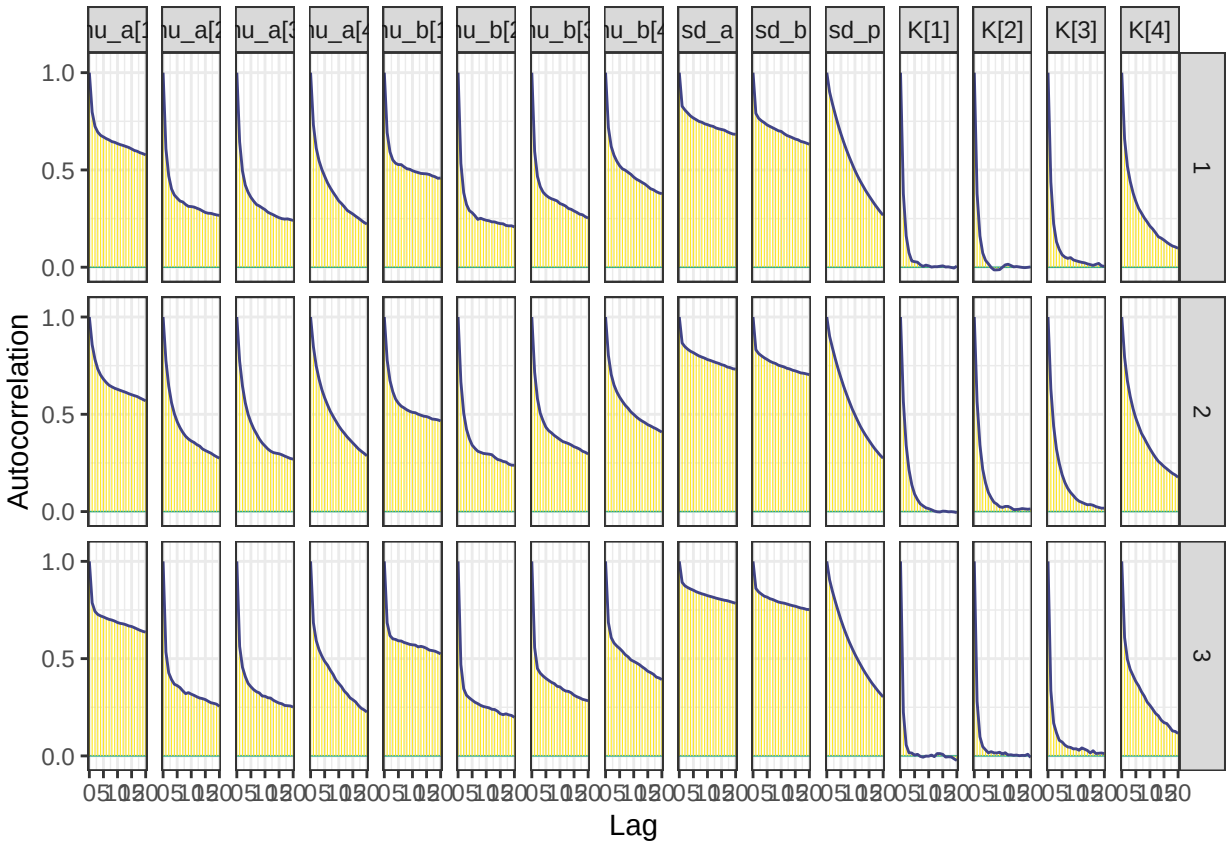
density plots

```
bayesplot::mcmc_dens_overlay(cs, pars = vars(starts_with(c("mu", "sd", "K"))))
```



autocorrelation of chains

```
bayesplot::mcmc_acf(cs, pars = vars(starts_with(c("mu", "sd", "K"))))
```



R-hat and effective sample size

```
MCMCvis::MCMCsummary(cs, params = c("mu_a","mu_b", "sd_a","sd_b", "sd_p", "K")) %>% knitr::kable(digits
```

	mean	sd	2.5%	50%	97.5%	Rhat	n.eff
mu_a[1]	1.36	0.09	1.18	1.36	1.53	1.00	541
mu_a[2]	1.04	0.07	0.90	1.04	1.16	1.00	2293
mu_a[3]	0.97	0.07	0.84	0.97	1.10	1.00	2383
mu_a[4]	0.71	0.08	0.54	0.71	0.86	1.00	3449
mu_b[1]	0.46	0.04	0.39	0.46	0.54	1.00	734
mu_b[2]	0.55	0.03	0.49	0.55	0.61	1.00	3205
mu_b[3]	0.44	0.03	0.37	0.44	0.51	1.00	2076
mu_b[4]	0.29	0.04	0.21	0.29	0.38	1.00	1496
sd_a	0.15	0.05	0.04	0.15	0.25	1.01	313
sd_b	0.08	0.02	0.03	0.08	0.12	1.01	353
sd_p	0.25	0.01	0.23	0.25	0.27	1.00	3038
K[1]	12.69	2.10	9.29	12.45	17.63	1.00	39381
K[2]	10.17	1.92	7.12	9.94	14.68	1.00	33446
K[3]	5.73	0.76	4.39	5.68	7.43	1.00	22130
K[4]	2.73	0.32	2.15	2.72	3.41	1.00	5934

```
#lapply(cs, coda::effectiveSize) %>% dplyr::bind_rows() #%>% summarise_all(funs(sum))# can also comput
```

- $\hat{R}$: measures ratio average variance within change compared to across chains.
- Effective sample size: measures how much autocorrelation in chains increases uncertainty in estimates of posterior
  - how large does my N_{eff} need to be? Not sure - see mcmcse

Model Checking

Coefficient of variation: $CV_i = \mu_i/\sigma_i$

Table 2: Posterior predictive check: coefficient of variation

treatment	ts_obs	ts_pred	p_value
0.00	2.9644458	2.055061	0.0000000
0.05	1.3182378	1.411516	0.8322889
0.10	1.0066157	1.180144	0.9872111
0.15	0.8217851	1.061684	0.9994667

Maximum: $\max(\boldsymbol{n_i})$

Table 3: Posterior predictive check: max abundance

treatment	ts_obs	ts_pred	p_value
0.00	29	48.06102	0.9974333
0.05	54	48.79607	0.2485556
0.10	22	24.34962	0.5663444
0.15	13	10.50621	0.0916111

Comparing models

$X_{ij,t} = a_i + b_i X_{ij,t-1} + N(0, \sigma_i^2)$, for treatment $i = 1, 2, 3, 4$, replicate $j = 1, \ldots, 10$.

- Both
  - Gompertz with direct density dependence
- Bayesian
  - response is poisson with lognormal process error
  - random effect of treatment (variation among replicates)
  - shared parameters: $\sigma_a, \sigma_b, \sigma_p$
- GLS
  - response is lognormal
  - fixed effect of treatments (no random effect of replicate)
  - treatment specific residual (process error)

Parameter estimates

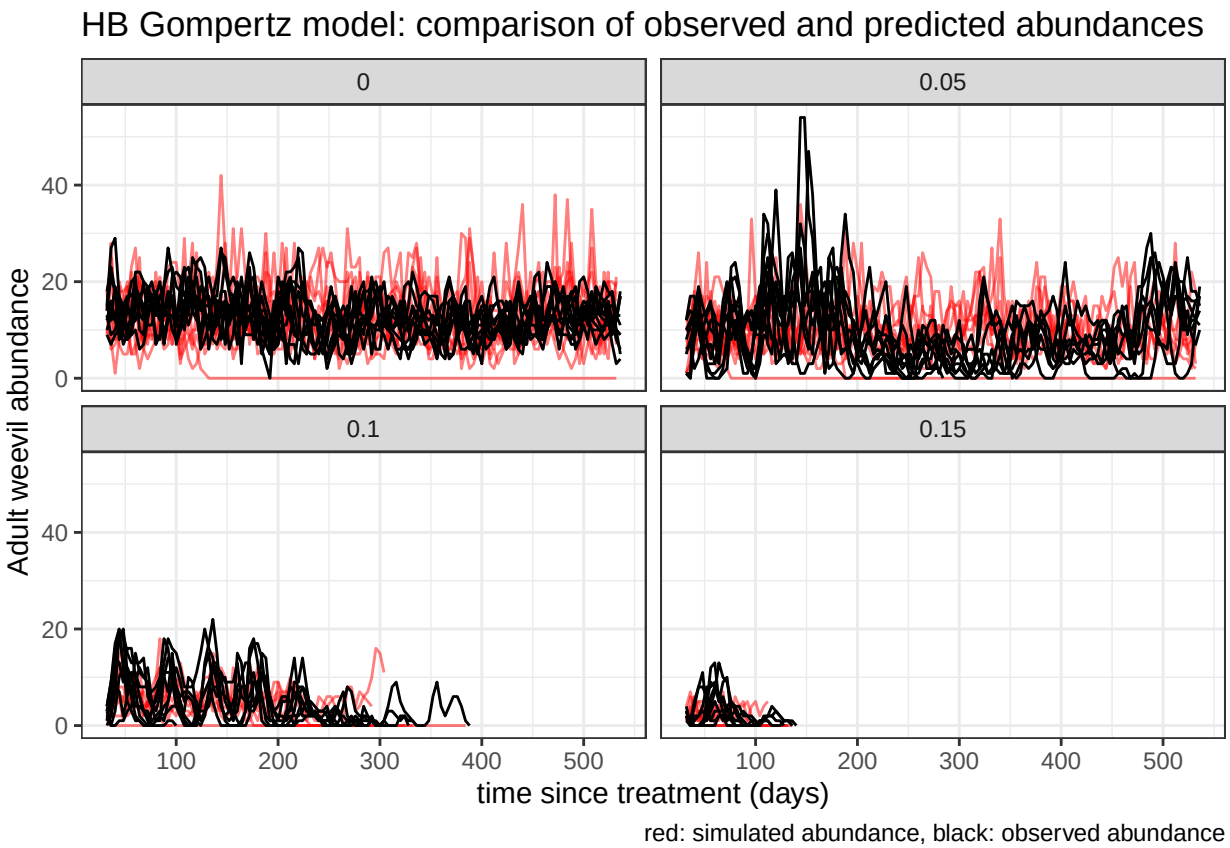
Table 4: Parameter estimates from hierarchical bayes model

treatment	density dependence (b)	population growth rate (a)	process noise (sd)
0	0.46 (0.39 , 0.54)	1.36 (1.18 , 1.53)	0.25 (0.23 , 0.27)
0.05	0.55 (0.49 , 0.61)	1.04 (0.9 , 1.16)	0.25 (0.23 , 0.27)
0.1	0.44 (0.37 , 0.51)	0.97 (0.84 , 1.1)	0.25 (0.23 , 0.27)
0.15	0.29 (0.21 , 0.38)	0.71 (0.54 , 0.86)	0.25 (0.23 , 0.27)

Table 5: Parameter estimates from previous model: treatment specific variance

treatment	density dependence (b)	population growth rate (a)	process noise (sd)
0	0.57 (0.52 , 0.61)	1.11 (0.99 , 1.23)	0.29 (0.27 , 0.3)
0.05	0.25 (0.19 , 0.31)	0.38 (0.3 , 0.45)	0.48 (0.44 , 0.53)
0.1	0.24 (0.17 , 0.31)	0.24 (0.16 , 0.32)	0.55 (0.49 , 0.61)
0.15	0.13 (0.03 , 0.23)	0.2 (0.08 , 0.32)	0.52 (0.45 , 0.6)

Comparison of model predictions



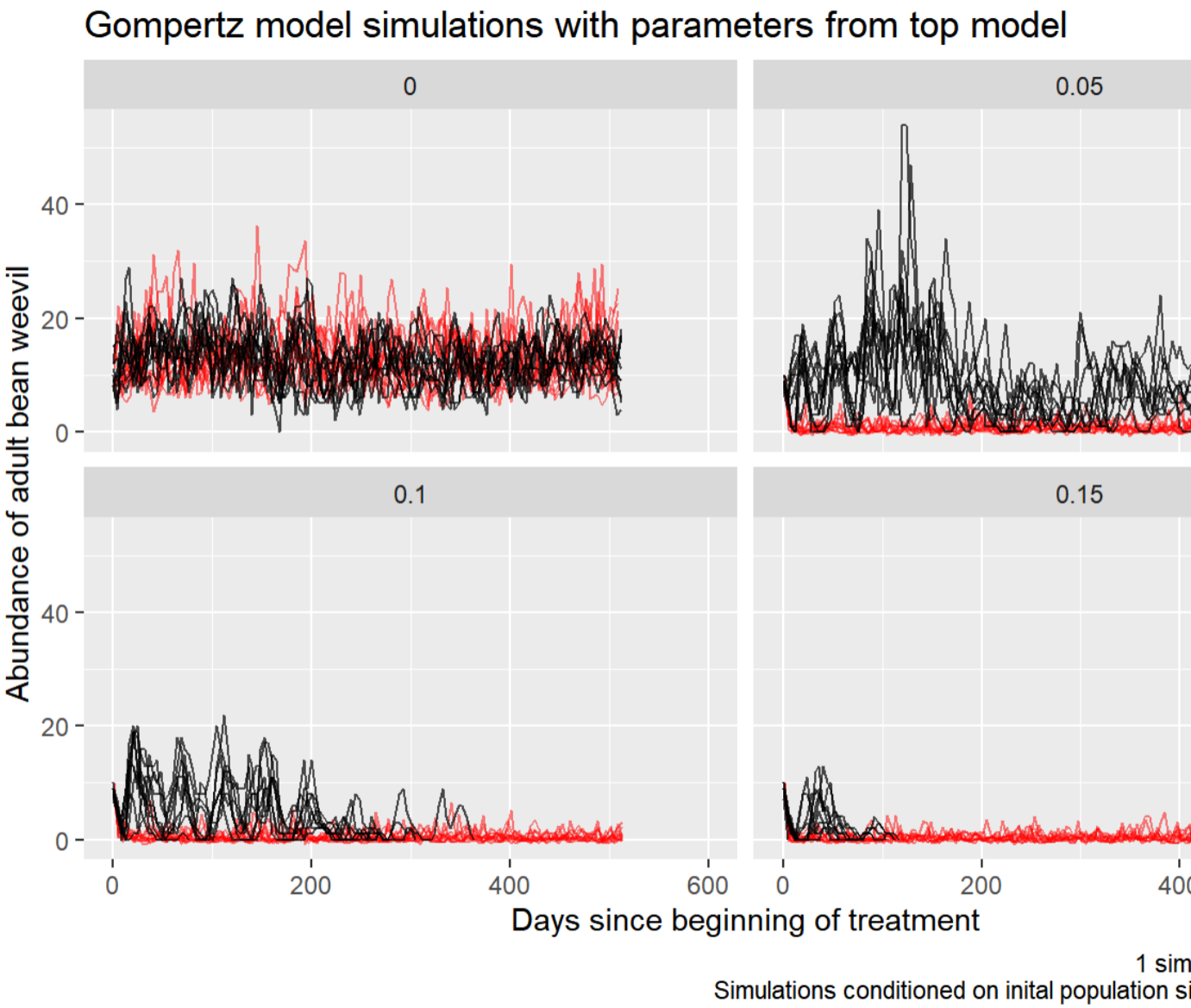
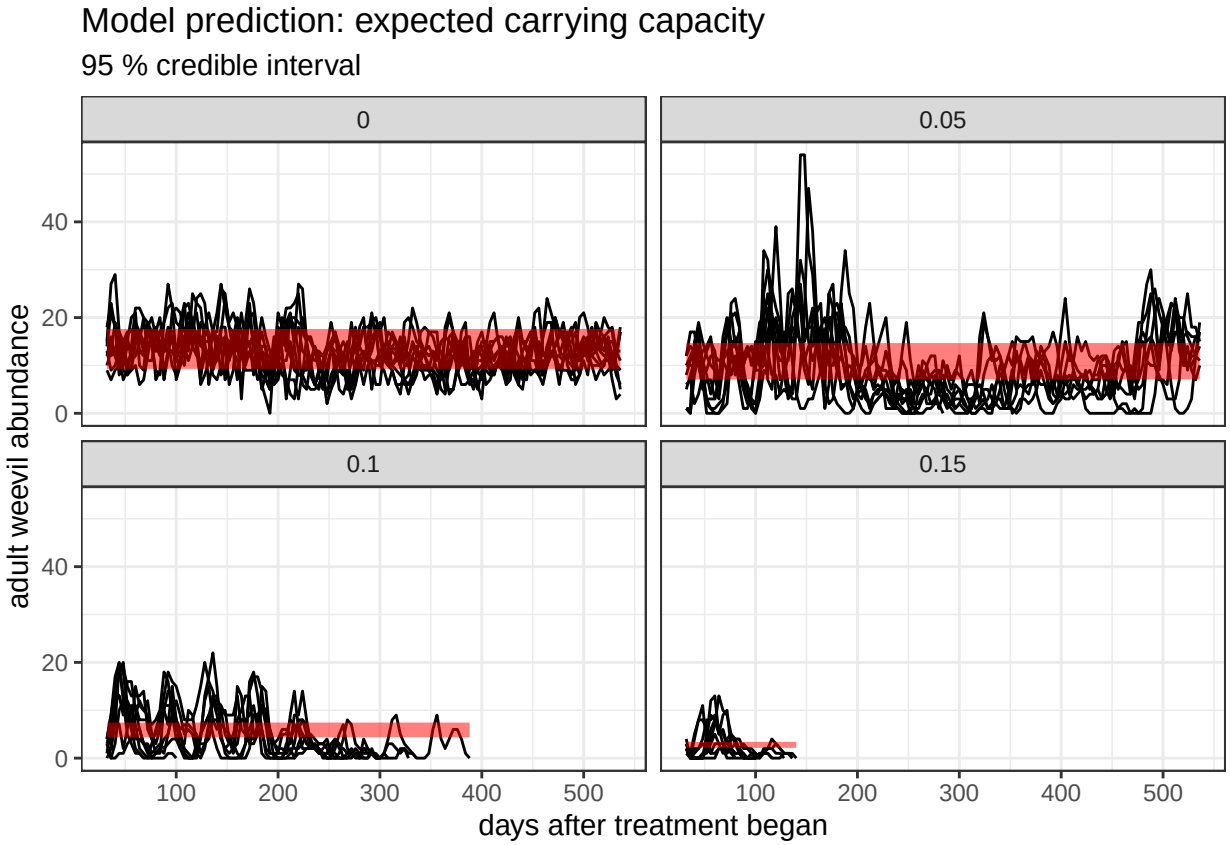
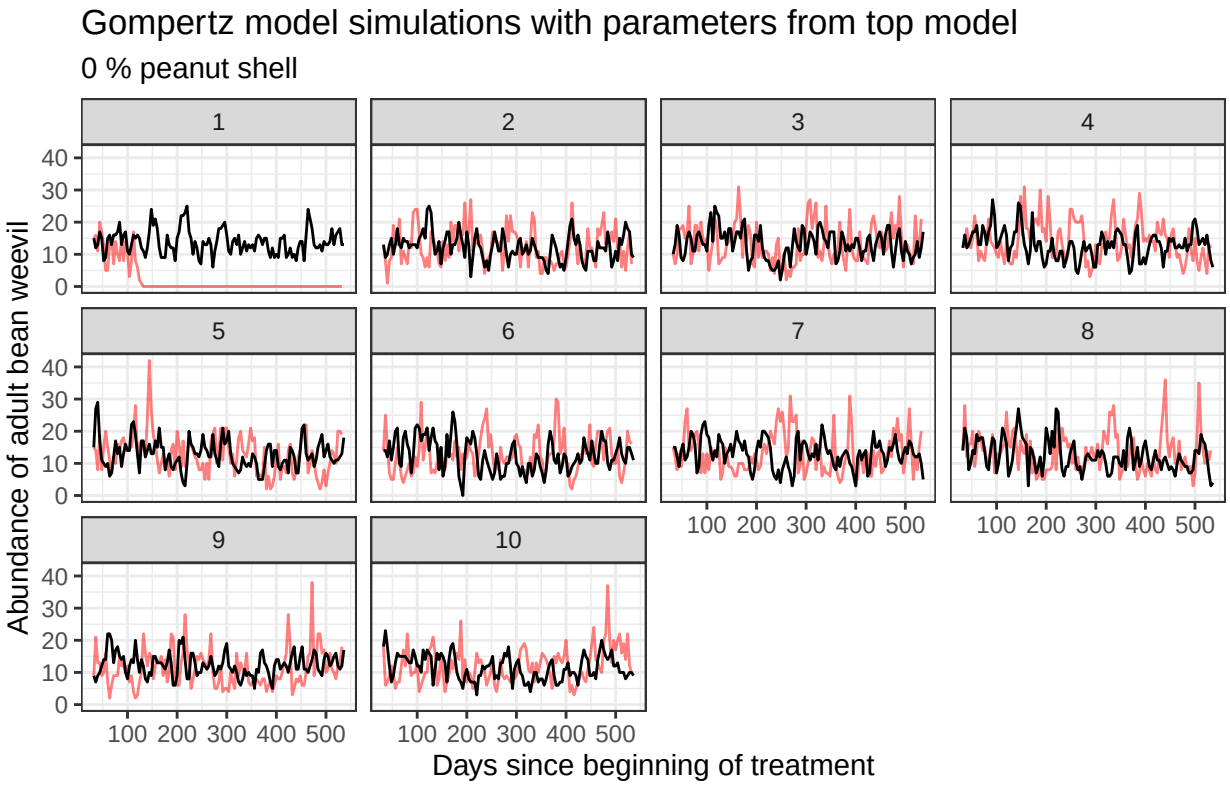


Figure 1: Log-Gompertz model estimated by GLS

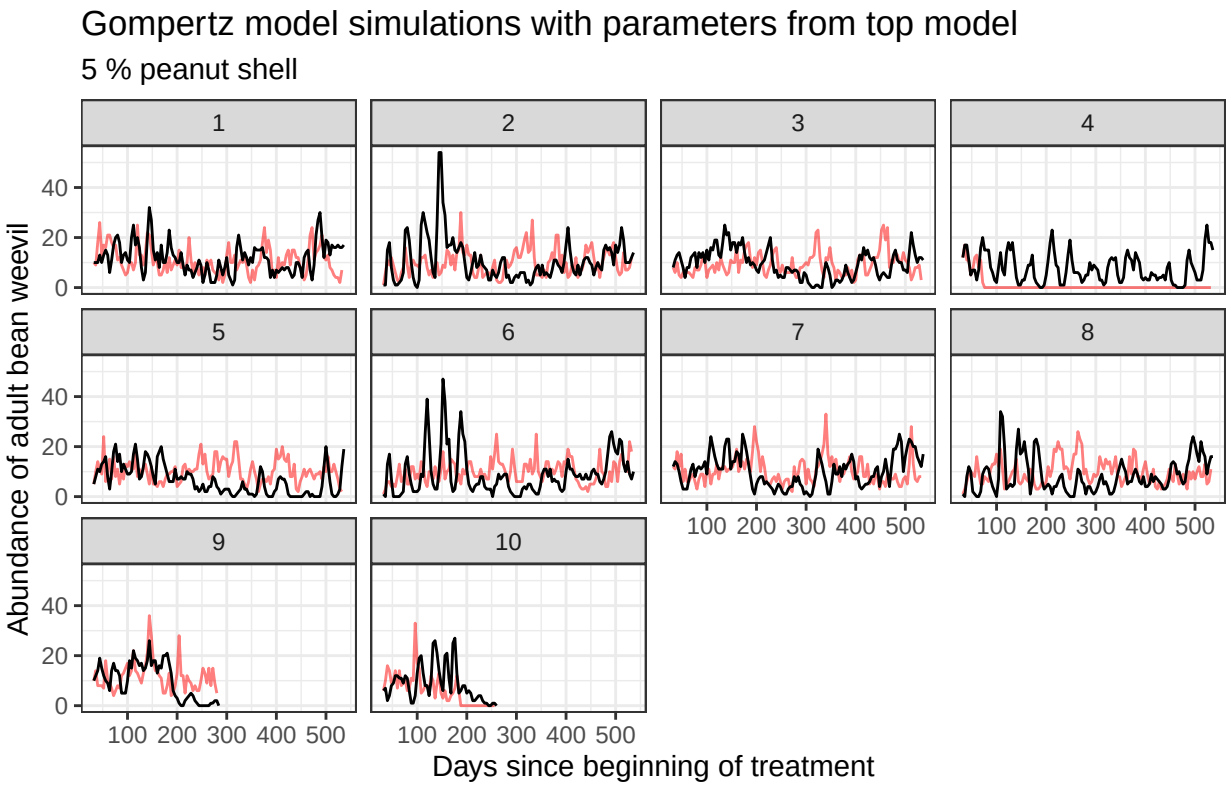
95% CrI of carrying capacity (derived quantity)



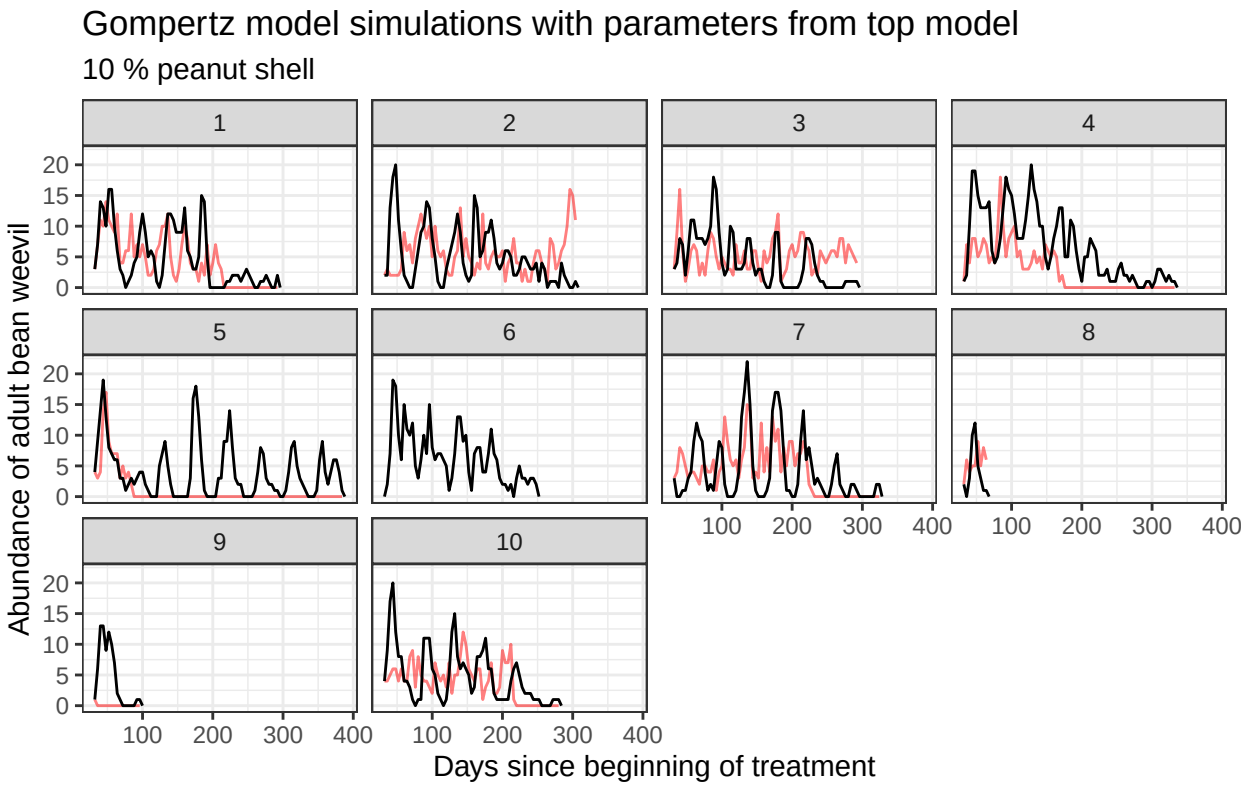
Predictions for individual replicates



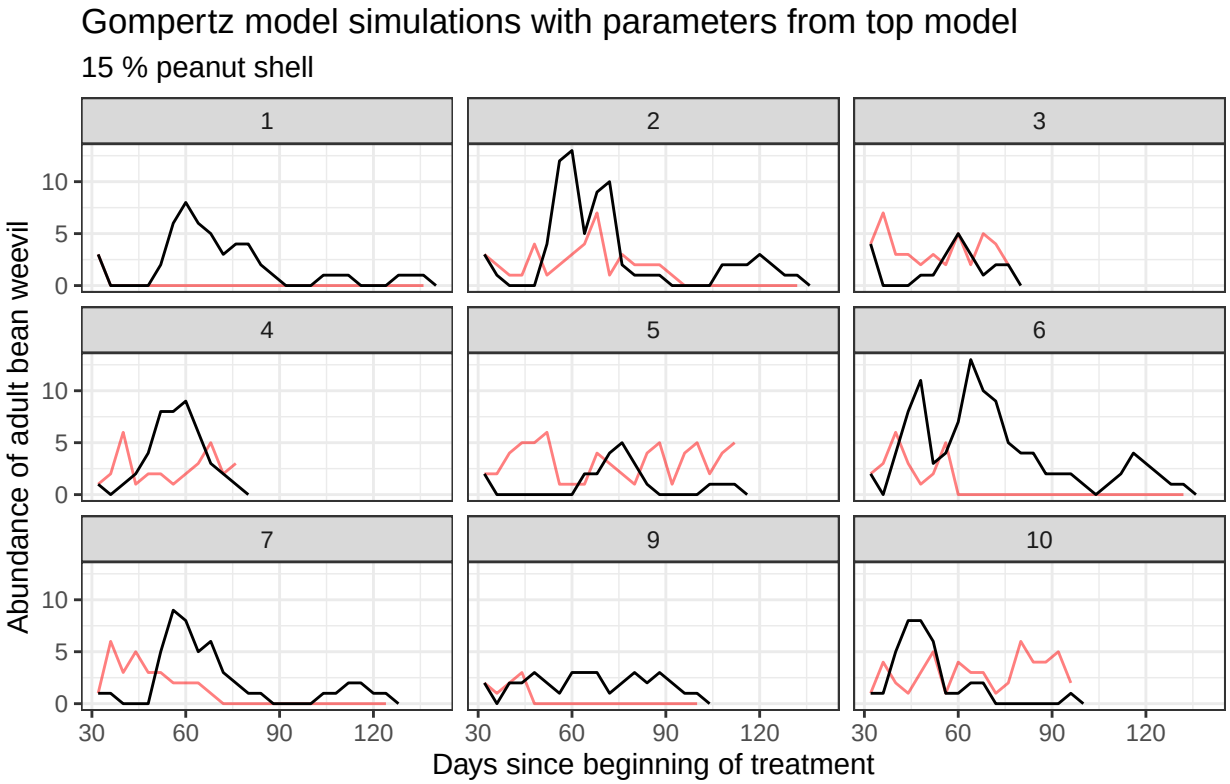
1 simulation per replicate.
Simulations conditioned on inital population size in each replicate.



1 simulation per replicate.
Simulations conditioned on inital population size in each replicate.



1 simulation per replicate.
Simulations conditioned on inital population size in each replicate.



1 simulation per replicate.
Simulations conditioned on inital population size in each replicate.

Final thoughts

- Modeling populations as discrete vs continuous
- Delayed density dependence
- Latent stages
- Look into effective sample size and monte carlo error