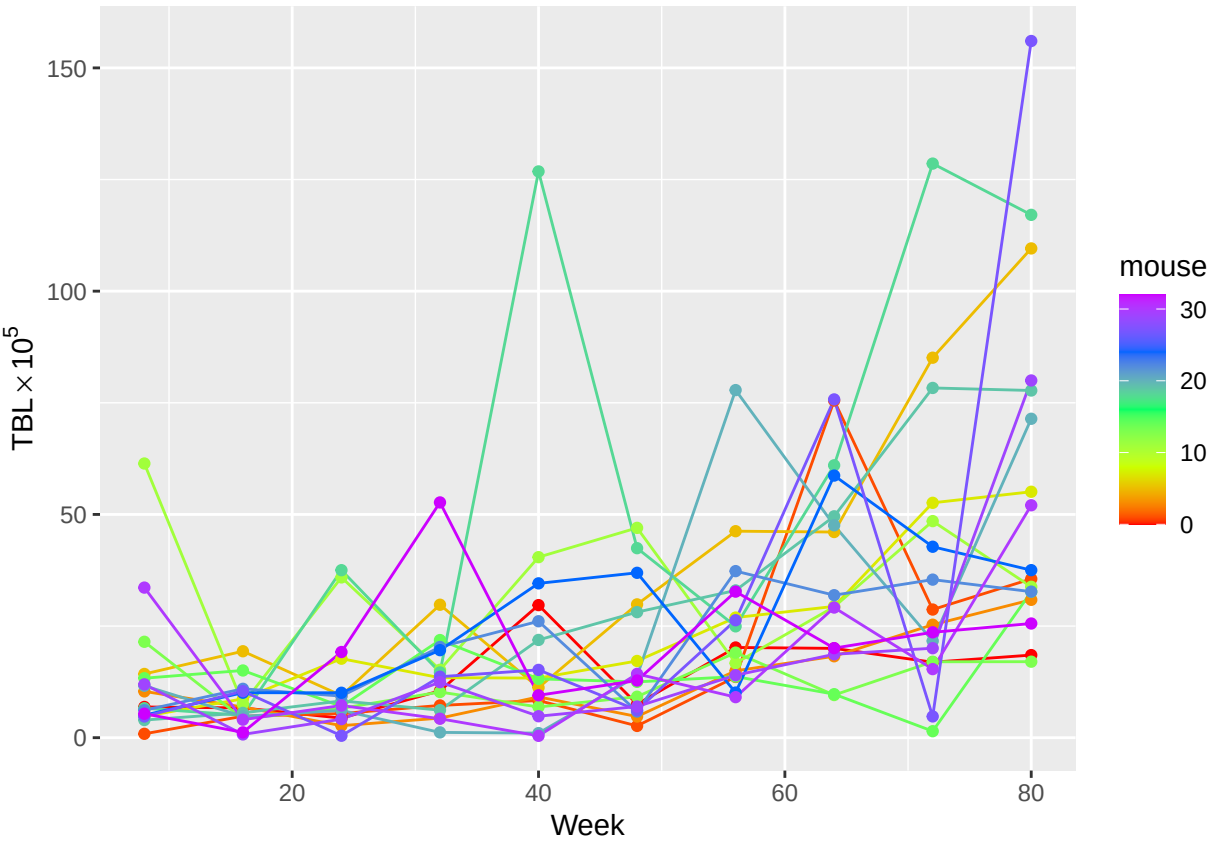
Reproducing Karin et al's longitudal model of senescent cell growth

Loading and visualizing the data

```
snc_counts_fname <- str_c(
  params$data_dir, "raw", "senescent_cells_and_the_dynamics_of_aging__fig_2.csv",
  sep="/"
)
raw_snc_counts <- read.csv(snc_counts_fname, header=TRUE) %>%
  drop_na()
snc_counts <- melt(
  data = raw_snc_counts,
  id.vars = "mouse.week",
  measure.vars = c(
    "X8", "X16", "X24", "X32", "X40", "X48", "X56", "X64", "X72", "X80"
  ),
  value.name = "TBL_x_10_5"
)
snc_counts <- snc_counts %>%
  rename(week = "variable", mouse = "mouse.week")
snc_counts$week <- map_int(snc_counts$week, function (week) { return(as.integer(substr(week, 2, str_leng
length(unique(snc_counts$week))
```

```
## [1] 10
```

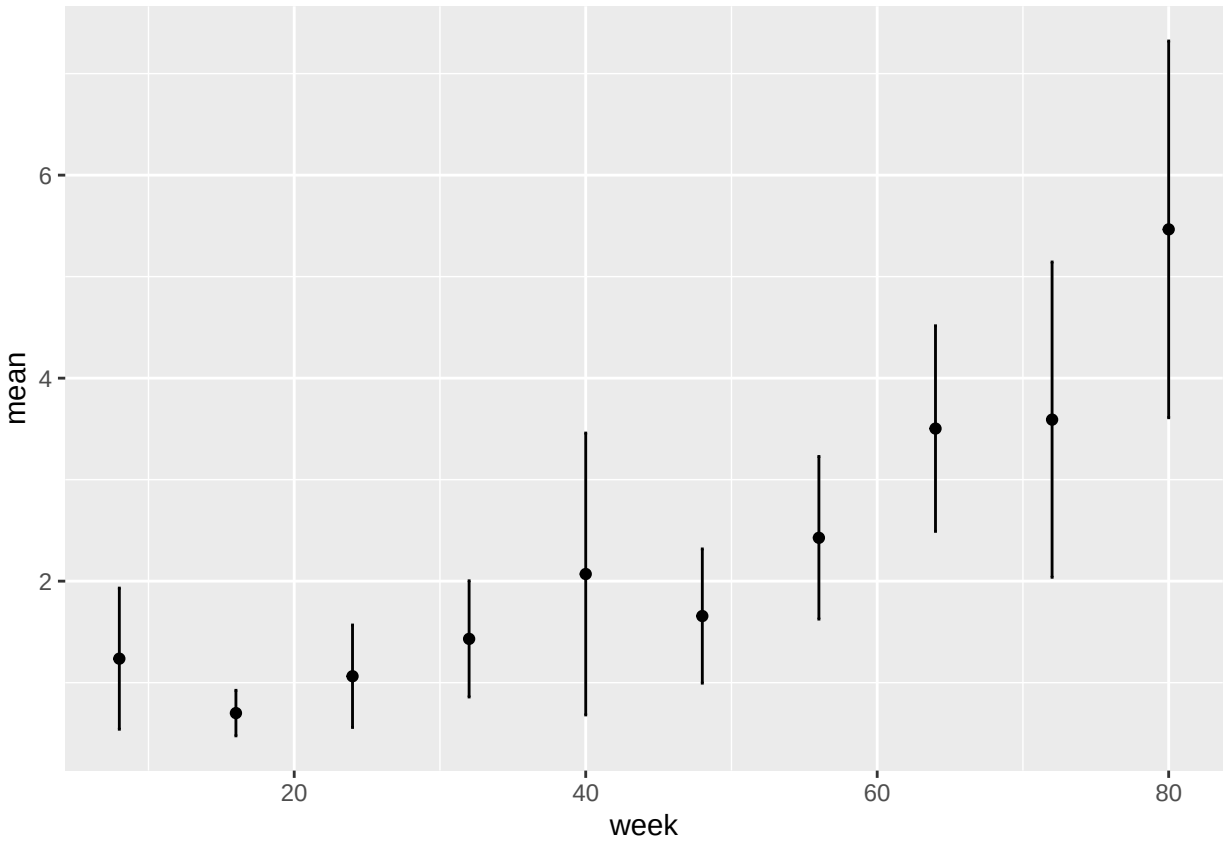
```
ggplot(
  data = snc_counts,
  aes(x = week, y = TBL_x_10_5, color = mouse, group = mouse)
) +
  xlab("Week") + ylab(TeX("$ TBL \\times 10^5 $")) +
  scale_color_gradientn(colours = rainbow(5)) + geom_line() + geom_point()
```




```
max(snc_counts$mouse)
```

```
## [1] 32
```

```
norm_const <- mean(
  snc_counts[snc_counts$week %in% c(8, 16, 24),]$TBL_x_10_5, na.rm = T
)
snc_counts$norm_abundance <- snc_counts$TBL_x_10_5 / norm_const
mean_abundances <- aggregate(
  snc_counts$norm_abundance, list(snc_counts$week), mean, na.rm = T
)$x
stddev_abundances <- aggregate(
  snc_counts$norm_abundance, list(snc_counts$week), sd, na.rm = T
)$x
mean_stddev_abundances <- data.frame(
  mean = mean_abundances,
  stddev = stddev_abundances,
  week = unique(snc_counts$week)
)
p <- ggplot(mean_stddev_abundances, aes(x=week, y=mean)) + geom_point() +
    geom_errorbar(aes(ymin=mean-stddev/2, ymax=mean+stddev/2), width=.2,
                  position=position_dodge(0.05))
print(p)
```



Fitting our three models on longitudinal data

Data processing

```
# Time series of SnC counts
mice <- unique(snc_counts$mouse)
weeks <- unique(snc_counts$week)
n_mice <- length(mice)
n_weeks <- length(weeks)
snc_abundance <- matrix(nrow = n_mice, ncol=n_weeks)
for (m in 1:n_mice) {
  for (t in 1:n_weeks) {
    mouse <- mice[m]
    week <- weeks[t]
    mouse_week_index <- which(snc_counts$mouse == mouse & snc_counts$week == week)[1]
    snc_abundance[m, t] <- snc_counts$norm_abundance[mouse_week_index]
  }
}

# times
t <- sort(weeks) # 8 hard-coded to match data
t0 = 0

# initial conditions
y0 = c(1)
dim(y0) <- (1)
```

```
# data for Stan
data_snc <- list(n_weeks = n_weeks, n_mice = n_mice, y0 = y0, t0 = t0, ts = t, cells = snc_abundance)

saveRDS(snc_counts, "saved_fit/snc_counts.RDS")
saveRDS(data_snc, "saved_fit/data_snc.RDS")

# number of MCMC steps
niter <- 2000
```

MCMC sampling

SR model

```
if (params$refit_models) {
  model <- stan_model("snc_sr_model.stan")
  fit_snc_sr_model <- sampling(
    model,
    data = data_snc,
    iter = niter,
    chains = 4
  )
  saveRDS(fit_snc_sr_model, "saved_fit/fit_snc_sr_model.RDS")
} else {
  fit_snc_sr_model <- readRDS("saved_fit/fit_snc_sr_model.RDS")
}
```

```
## Warning: There were 1 divergent transitions after warmup. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

USR model

```
if (params$refit_models) {
  model <- stan_model("snc_usr_model.stan")
  fit_snc_usr_model <- sampling(
    model,
    data = data_snc,
    iter = niter,
    chains = 4
  )
  saveRDS(fit_snc_usr_model, "saved_fit/fit_snc_usr_model.RDS")
} else {
  fit_snc_usr_model <- readRDS("saved_fit/fit_snc_usr_model.RDS")
}
```

SIS model

```
if (params$refit_models) {
  model <- stan_model("snc_sis_model.stan")
  fit_snc_sis_model <- sampling(
    model,
```

```
    data = data_snc,
    iter = niter,
    chains = 4,
  )
  saveRDS(fit_snc_sis_model, "saved_fit/snc_sis_model.RDS")
} else {
  fit_snc_sis_model <- readRDS("saved_fit/snc_sis_model.RDS")
}
```

```
## recompiling to avoid crashing R session
```

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

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -std=gnu99 -I"/usr/share/R/include" -DNDEBUG   -I"/home/stephen/R/x86_64-pc-linux-gnu-library/4.
## In file included from /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:88
##                  from /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1
##                  from /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/p
##                  from <command-line>:
## /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1
##   613 | namespace Eigen {
##       | ^~~~~~~~~
## /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17
##   613 | namespace Eigen {
##       |                 ^
## In file included from /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1
##                  from /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/p
##                  from <command-line>:
## /home/stephen/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:96:10: fatal error: comp
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:172: foo.o] Error 1
```

Fit diagnostics

```
sr_pars=c('eta', 'beta', 'sigma', 'lp__', 'log_likelihood')
print(fit_snc_sr_model, pars = sr_pars, digits_summary = 4)
```

```
## Inference for Stan model: snc_sr_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                    mean se_mean     sd      2.5%       25%       50%       75%
## eta[1]           0.2585  0.0053 0.1648    0.0262    0.1349    0.2270    0.3538
## eta[2]           0.0086  0.0002 0.0061    0.0025    0.0049    0.0069    0.0101
## beta[1]          0.9304  0.0154 0.5078    0.1844    0.5457    0.8463    1.2447
## beta[2]          0.9696  0.0137 0.5165    0.1908    0.5970    0.8848    1.2676
## sigma            0.8952  0.0010 0.0508    0.8019    0.8605    0.8934    0.9279
## lp__           -83.5138  0.1404 3.9630  -91.9111  -85.9252  -83.2224  -80.8047
## log_likelihood -279.5257  0.1371 3.2722 -285.9863 -281.8331 -279.4252 -277.2314
##                   97.5% n_eff   Rhat
## eta[1]           0.6486   959 1.0036
## eta[2]           0.0256   758 1.0036
## beta[1]          2.0907  1094 1.0036
```

```
## beta[2]           2.1818  1422 1.0004
## sigma             0.9997  2503 1.0015
## lp__            -76.6495   797 1.0055
## log_likelihood -273.4689   570 1.0053
##
## Samples were drawn using NUTS(diag_e) at Tue Dec  8 18:45:34 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
usr_pars=c('eta', 'beta', 'sigma', 'log_likelihood', 'lp__')
print(fit_snc_usr_model, pars = usr_pars, digits_summary = 4)
```

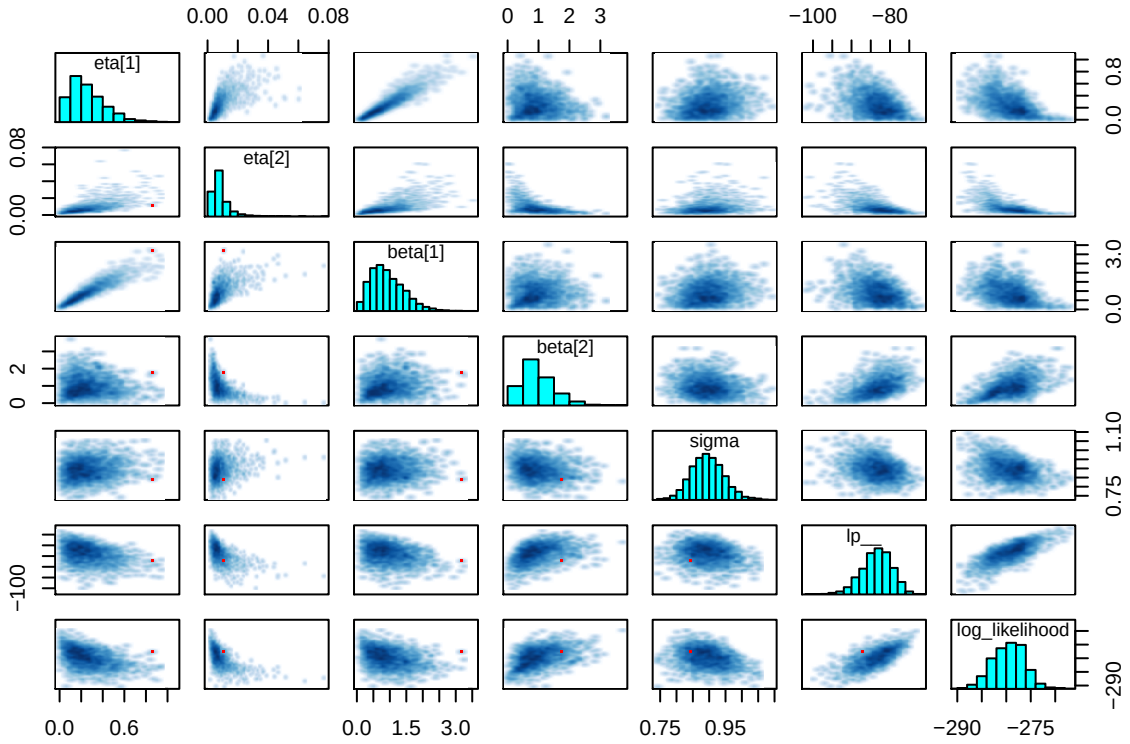
```
## Inference for Stan model: snc_usr_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                     mean se_mean     sd      2.5%       25%       50%       75%
## eta[1]            0.5997  0.0065 0.2617    0.1364    0.4164    0.5768    0.7642
## eta[2]            0.0099  0.0002 0.0085    0.0006    0.0038    0.0078    0.0134
## beta[1]           1.2414  0.0130 0.5045    0.3175    0.8925    1.2170    1.5686
## beta[2]           0.0115  0.0001 0.0050    0.0027    0.0079    0.0112    0.0146
## sigma             0.9085  0.0008 0.0501    0.8168    0.8733    0.9072    0.9410
## log_likelihood -281.9571  0.0447 1.7294 -286.0884 -282.8417 -281.5908 -280.8036
## lp__            -89.7420  0.0909 3.5274  -97.6533  -91.9021  -89.3319  -87.1946
##                    97.5% n_eff   Rhat
## eta[1]            1.1623  1623 1.0023
## eta[2]            0.0321  2852 1.0003
## beta[1]           2.3131  1505 1.0027
## beta[2]           0.0222  1499 1.0030
## sigma             1.0137  4234 0.9995
## log_likelihood -279.8799  1496 1.0009
## lp__            -83.9379  1505 1.0011
##
## Samples were drawn using NUTS(diag_e) at Tue Dec  8 19:04:24 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
sis_pars=c('eta', 'beta', 'sigma', 'alpha', 'll', 'log_likelihood', 'lp__')
print(fit_snc_sis_model, pars = sis_pars, digits_summary = 4)
```

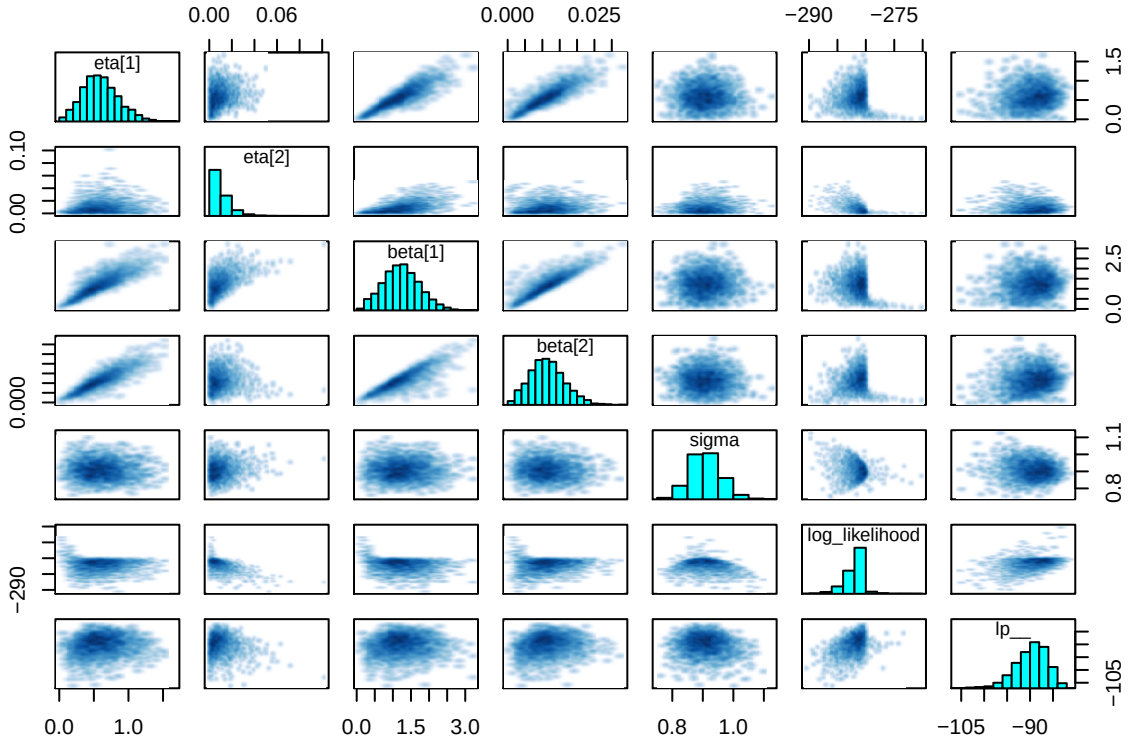
```
## Inference for Stan model: snc_sis_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                     mean se_mean     sd      2.5%       25%       50%       75%
## eta[1]            0.1777  0.0029 0.1309    0.0093    0.0786    0.1493    0.2481
## eta[2]            0.5748  0.0087 0.4166    0.1014    0.2687    0.4601    0.7738
## beta[1]           0.8952  0.0100 0.4457    0.2432    0.5536    0.8289    1.1581
## beta[2]           1.0580  0.0096 0.5018    0.2988    0.6944    0.9877    1.3383
## sigma             0.8945  0.0008 0.0513    0.8047    0.8589    0.8919    0.9265
## alpha             0.0180  0.0002 0.0119    0.0037    0.0090    0.0149    0.0238
## ll                0.1325  0.0015 0.1032    0.0049    0.0494    0.1101    0.1917
## log_likelihood -279.2234  0.0978 2.9544 -284.7900 -281.3095 -279.1236 -277.1771
```

```
## lp__             -86.9600  0.1174 3.7367  -94.7708  -89.2061  -86.7091  -84.4033
##                    97.5% n_eff   Rhat
## eta[1]             0.5093  2036 1.0032
## eta[2]             1.6586  2273 1.0008
## beta[1]            1.9141  1978 1.0031
## beta[2]            2.2018  2740 1.0016
## sigma              1.0030  3933 1.0005
## alpha              0.0479  3248 1.0007
## ll                 0.3821  4455 0.9997
## log_likelihood -273.3414   913 1.0084
## lp__             -80.2237  1014 1.0037
##
## Samples were drawn using NUTS(diag_e) at Tue Dec  8 19:16:59 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

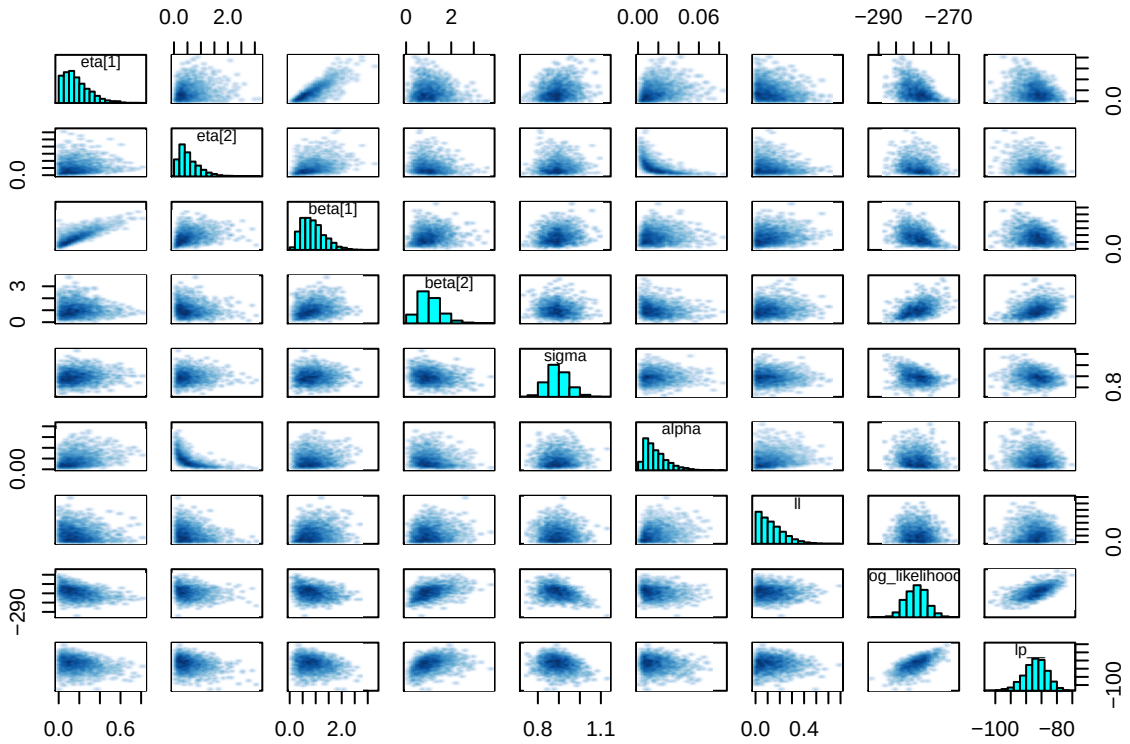
```
pairs(fit_snc_sr_model, pars=sr_pars)
```



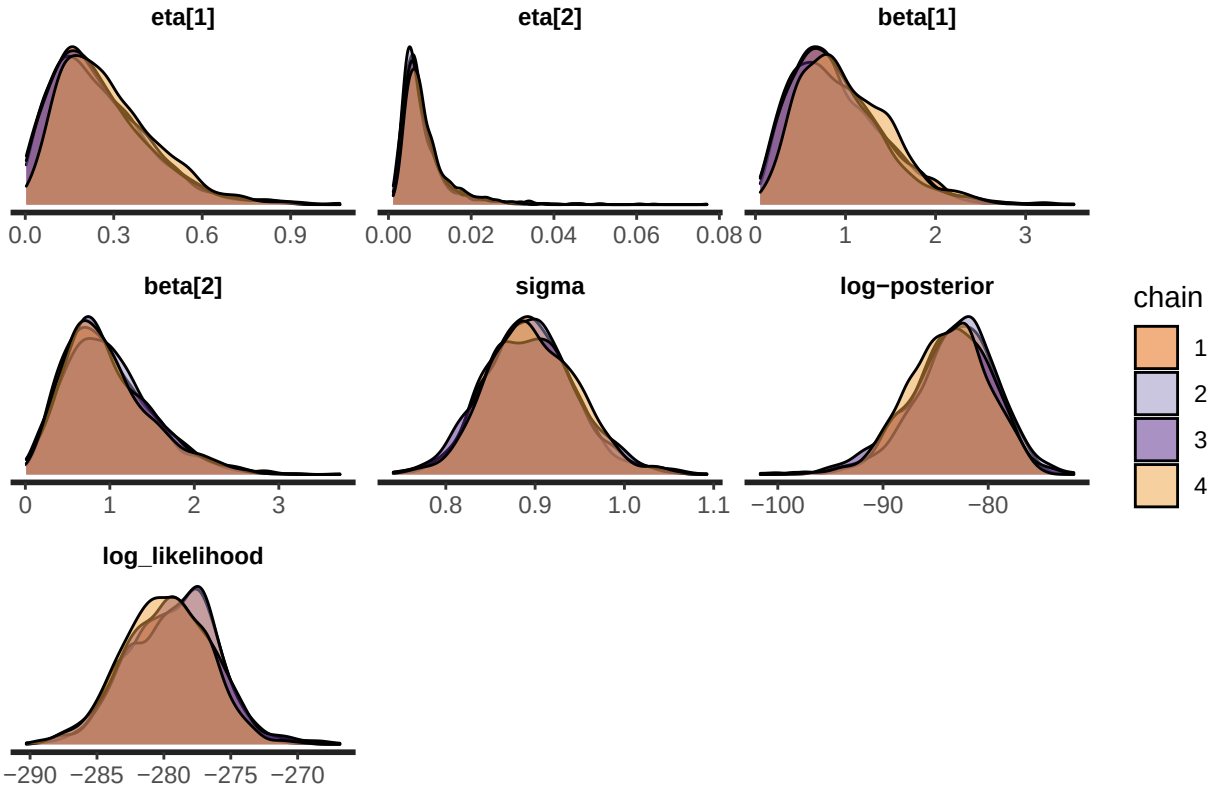

```
pairs(fit_snc_usr_model, pars = usr_pars)
```



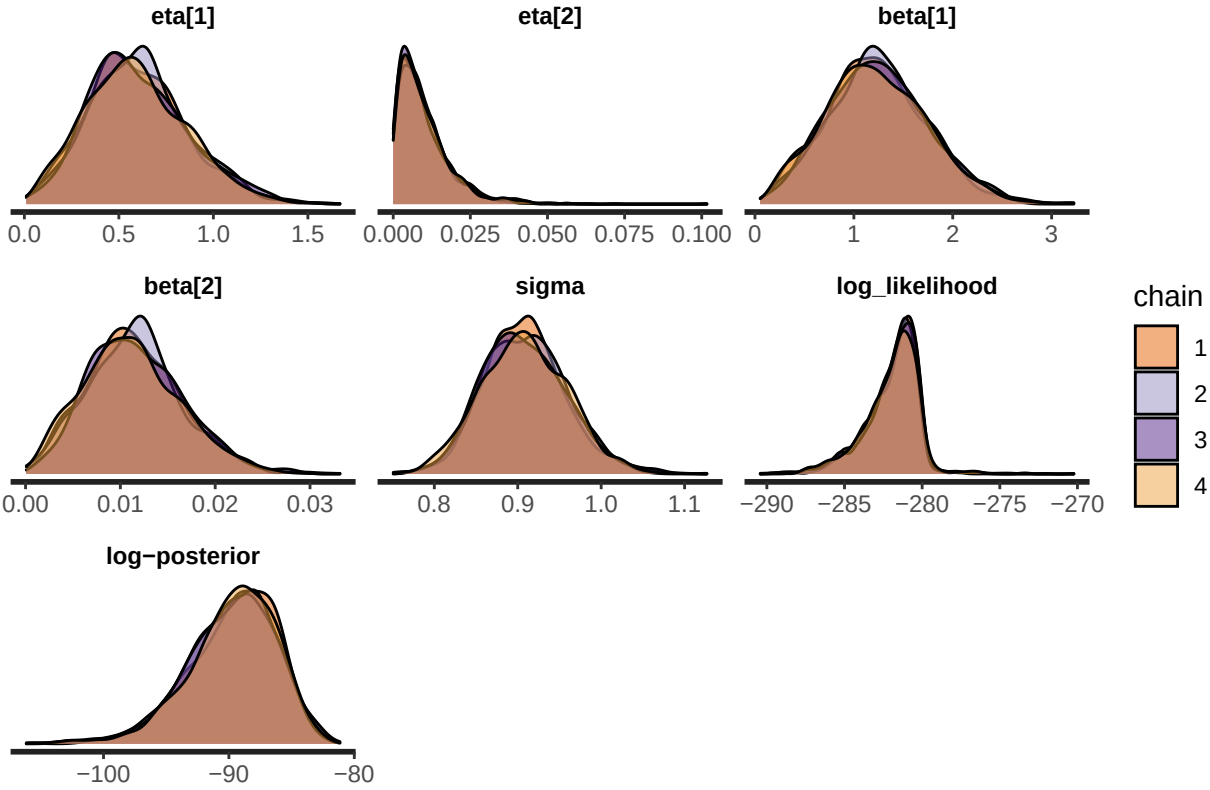

```
pairs(fit_snc_sis_model, pars = sis_pars)
```



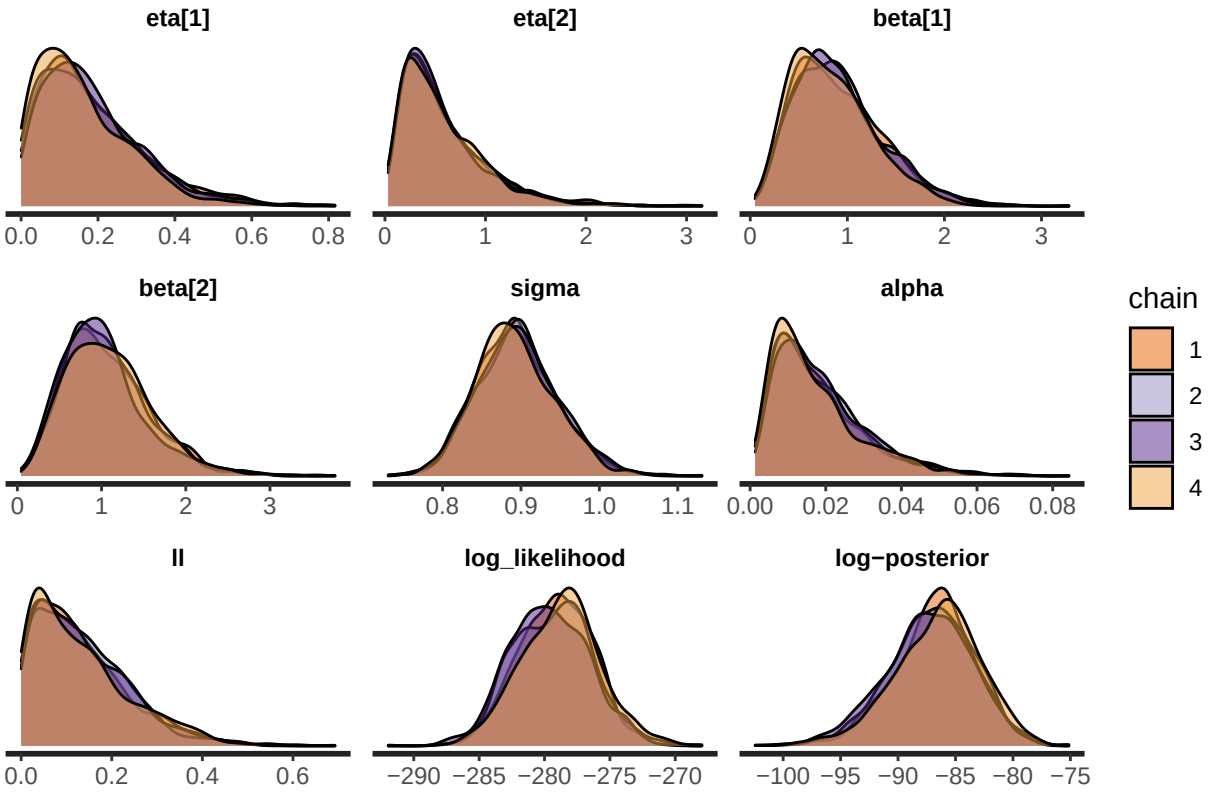

```
stan_dens(fit_snc_sr_model, pars = sr_pars, separate_chains = TRUE)
```




```
stan_dens(fit_snc_usr_model, pars = usr_pars, separate_chains = TRUE)
```



```
stan_dens(fit_snc_sis_model, pars = sis_pars, separate_chains = TRUE)
```



Trajectory plots

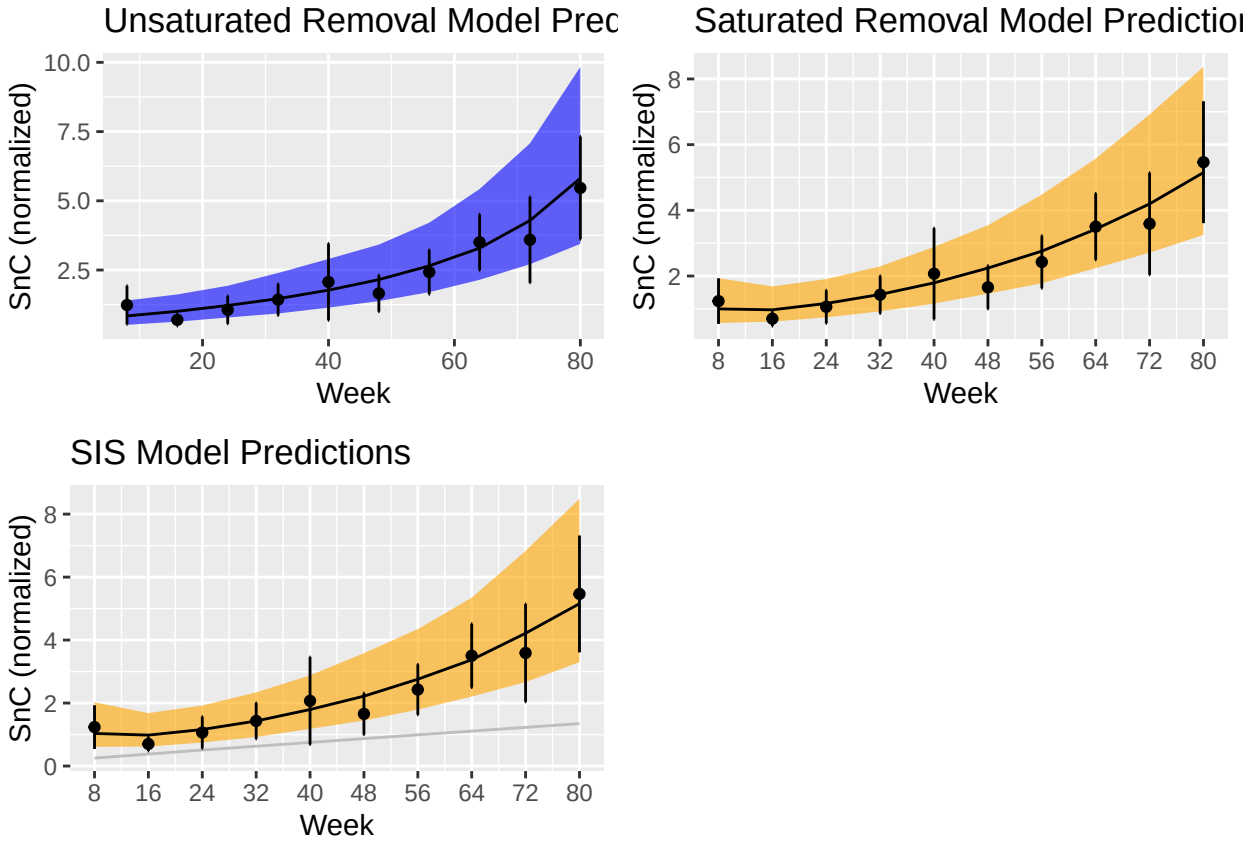
```
snc_sr_counts_pred <- cbind(
  as.data.frame(
    summary(
      fit_snc_sr_model, pars = "pred_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
snc_usr_counts_pred <- cbind(
  as.data.frame(
    summary(
      fit_snc_usr_model, pars = "pred_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
snc_sis_counts_pred <- cbind(
  as.data.frame(
```

```
    summary(
      fit_snc_sis_model, pars = "pred_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
snc_sis_ll_counts_pred <- cbind(
  as.data.frame(
    summary(
      fit_snc_sis_model, pars = "pred_ll_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
colnames(snc_sr_counts_pred) <- make.names(colnames(snc_usr_counts_pred)) # to remove % in the col name
colnames(snc_usr_counts_pred) <- make.names(colnames(snc_usr_counts_pred)) # to remove % in the col nam
colnames(snc_sis_counts_pred) <- make.names(colnames(snc_sis_counts_pred)) # to remove % in the col nam
colnames(snc_sis_ll_counts_pred) <- make.names(colnames(snc_sis_ll_counts_pred)) # to remove % in the c


p_usr <- ggplot(snc_usr_counts_pred, mapping = aes(x = t)) +
  geom_ribbon(data = snc_usr_counts_pred, aes(ymin = X5., ymax = X95.), fill = "blue", alpha = 0.6) +
  geom_line(mapping = aes(x = t, y = X50.)) +
  geom_point(mapping = aes(y = mean_abundances)) +
  geom_errorbar(
    data = snc_sr_counts_pred,
    aes(
      ymin=mean_abundances - stddev_abundances/2,
      ymax=mean_abundances + stddev_abundances/2,
    ),
    width=.2,
    position=position_dodge(0.05),
  ) +
  labs(title = "Unsaturated Removal Model Predictions", x = "Week", y = "SnC (normalized)", color = "na

p_sr <- ggplot(snc_sr_counts_pred, mapping = aes(x = t)) +
  geom_ribbon(aes( ymin = X5., ymax = X95.), fill = "orange", alpha = 0.6) +
  geom_line(mapping = aes(x = t, y = X50.)) +
  geom_point(mapping = aes(y = mean_abundances)) +
  geom_errorbar(
    aes(
      ymin=mean_abundances - stddev_abundances/2,
      ymax=mean_abundances + stddev_abundances/2
    ),
    width=.2,
    position=position_dodge(0.05)
  ) +
```

```
  scale_x_continuous(name = "Week", breaks=t, labels=t, limits = c(8, 80)) +
  labs(title = "Saturated Removal Model Predictions", x = "Week", y = "SnC (normalized)", color = "name
p_sis <- ggplot(snc_sis_counts_pred, mapping = aes(x = t)) +
  geom_ribbon(aes( ymin = X5., ymax = X95.), fill = "orange", alpha = 0.6) +
  geom_line(mapping = aes(x = t, y = X50.)) +
  geom_line(data = snc_sis_ll_counts_pred, aes(x = t, y = X50.), col = "grey") +
  geom_point(mapping = aes(y = mean_abundances)) +
  geom_errorbar(
    aes(
      ymin=mean_abundances - stddev_abundances/2,
      ymax=mean_abundances + stddev_abundances/2
    ),
    width=.2,
    position=position_dodge(0.05)
  ) +
  scale_x_continuous(name = "Week", breaks=t, labels=t, limits = c(8, 80)) +
  labs(title = "SIS Model Predictions", x = "Week", y = "SnC (normalized)", color = "name")
ggarrange(p_usr, p_sr, p_sis)
```



Model comparison

```
fit_snc_sr_model.bridge <- bridge_sampler(fit_snc_sr_model, silent = T)
fit_snc_sis_model.bridge <- bridge_sampler(fit_snc_sis_model, silent = T)
fit_snc_usr_model.bridge <- bridge_sampler(fit_snc_usr_model, silent = T)
bayes_factor(fit_snc_sis_model.bridge, fit_snc_sr_model.bridge)
```

```
## Estimated Bayes factor in favor of x1 over x2: 10.84554
```

```
bayes_factor(fit_snc_sis_model.bridge, fit_snc_usr_model.bridge)
```

```
## Estimated Bayes factor in favor of x1 over x2: 397.68057
```

```
bayes_factor(fit_snc_sr_model.bridge, fit_snc_usr_model.bridge)
```

```
## Estimated Bayes factor in favor of x1 over x2: 36.66764
```

```
post_prob(fit_snc_sis_model.bridge, fit_snc_sr_model.bridge, fit_snc_usr_model.bridge)
```

```
## fit_snc_sis_model.bridge  fit_snc_sr_model.bridge fit_snc_usr_model.bridge
##              0.913476975              0.084226013              0.002297012
```