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

Loading and visualizing the data

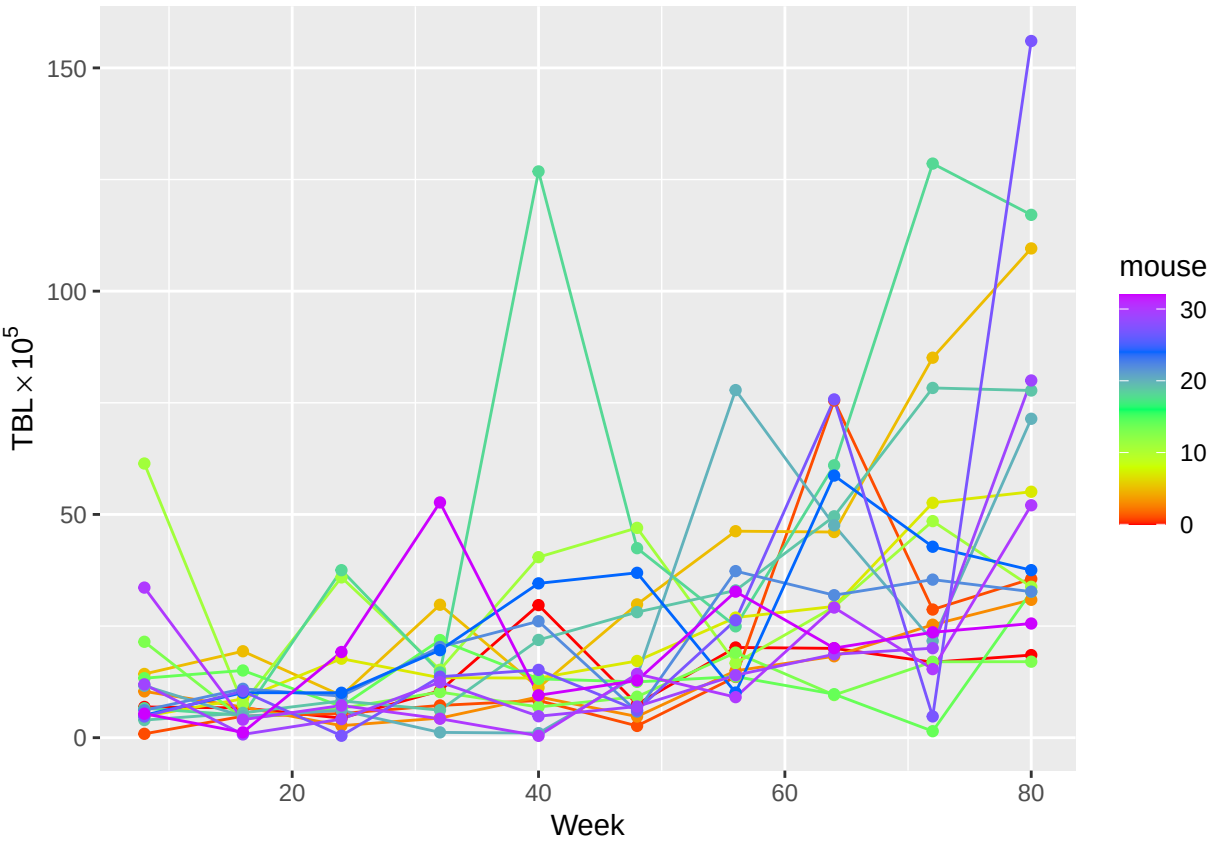
As our first step, we load the data we'll use to fit the 16 models described in Karin et al (TODO: actual reference). To do so, we load the raw data from their paper (used for their fig. 2) and transform it into a 3-column dataframe indexed by mouse number and week of measurement taken.

```
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$week, function (week) { return(as.integer(substr(week, 2, str_len
max(snc_counts$mouse)
```

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

Now we're going to reproduce Karin et al's fig. 2a, since it doesn't require any model-fitting, although we won't do the nice grey shading they did for now...

```
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()
```



Next we're going to replicate the first part of Karin et al's fig 2c, which shows the mean and standard deviation of normalized senescent cell counts at each week time point.

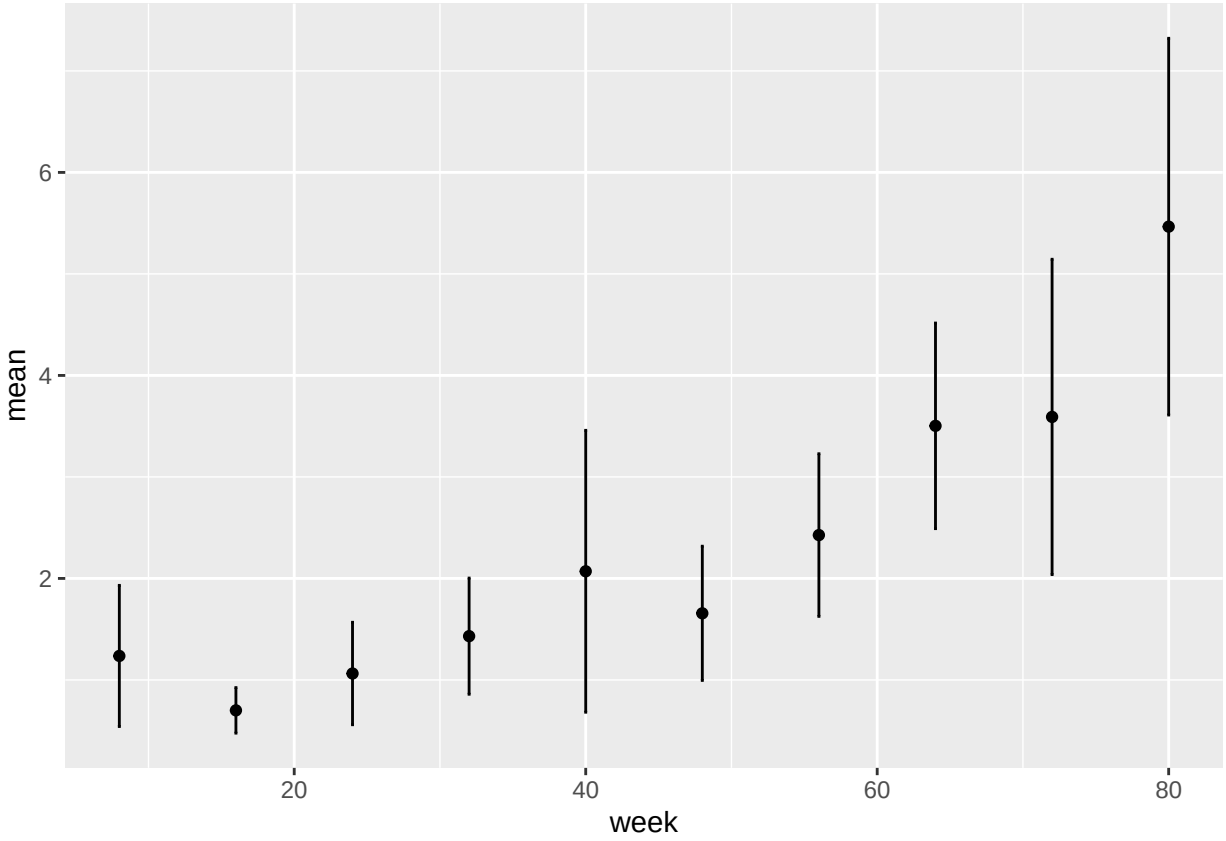
To normalize senescent cell counts, we adjust all counts to make the mean abundance across the first three measured time points – 8, 16, 24 – equal 1. To do so, we set

$$\beta = \frac{1}{3n} \sum_{i=1}^{n} \sum_{j \in 8,16,24} c_{ij} = 1$$

and divide all counts by it. In other words, normalize by the mean abundance from the first three weeks of data.

```
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)
```



Notably our standard deviation bars seem larger than whatever Karin et al show in their full version of this figure (fig. 2c), but we seem to reproduce their standard error values based on their display of them in fig. 2d.

Fitting the SR model

As a way to check we haven't missed something integral to Karin et al's work, we're going to fit a Bayesian version of their SR (Saturated-Removal) model and verify that we (hopefully) approximately reproduce their predictive distribution.

Since they did maximum likelihood estimation (with a grid search I'd add) whereas we're fitting a full posterior distribution with close-to-noninformative prior, we don't expect exact matching. Because the likelihood includes an ODE, analytically showing the data dominates the prior goes beyond my abilities, but intuitively I expect the posterior predictive distribution to have approximately the same mean as theirs with wider error bars.

```
# Time series of SnC counts
mice <- unique(snc_counts$mouse)
weeks <- unique(snc_counts$week)
n_mice <- length(mice)
n_weeks <- length(weeks)
print(n_mice)
```

```
## [1] 17
```

```
snc_abundance <- matrix(nrow = n_mice, ncol=n_weeks)
for (m in 1:n_mice) {
  for (t in 1:n_weeks) {
    week <- weeks[t]
    mouse <- mice[m]
    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 <- seq(0, n_weeks * 8, by = 8) # 8 hard-coded to match data
t0 = 0
t <- t[-1]

# 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)

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

```
model <- stan_model("snc_sr_model.stan")
fit_snc_sr_model <- sampling(
  model,
  data = data_snc,
  iter = niter,
  chains = 4
)
```

Diagnosing model fit

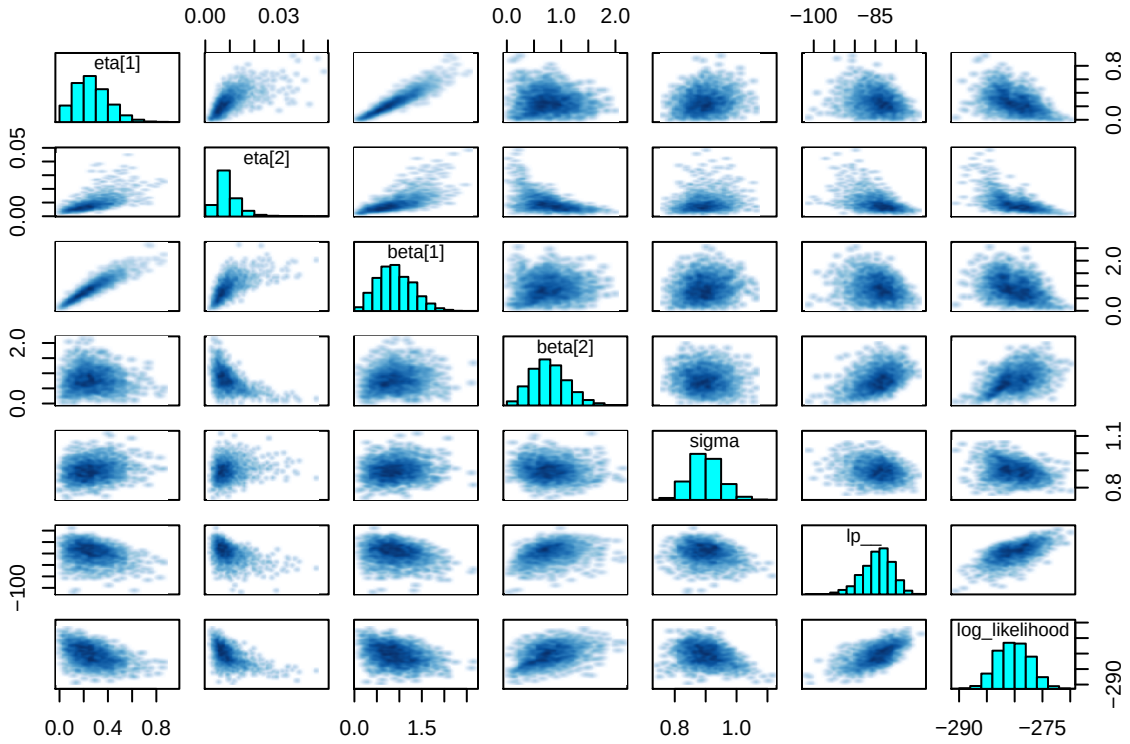
Inspired by this Stan case study and general best practices for checking the results of HMC, let's do some posterior checks to verify HMC ended up sampling a reasonable approximation of the true posterior. First, we can look at $\hat{R}$ and n_{eff}. $\hat{R}$ measures the level of agreement between our four independent Markov chains. An $\hat{R} \leq 1.01$ for a parameter implies close agreement between the chains, so it's great to see that all of our parameters of interest have $\hat{R}$s of 1. $ n_{eff} $, is a proxy measure of **effective** sample size, which we can think of as sample size adjusted down as a function of how correlated different samples from the posterior are. Again, our n_{eff}s all seem pretty good given the number of samples we draw.

```
pars=c('eta', 'beta', 'sigma', 'lp__', 'log_likelihood')
print(fit_snc_sr_model, pars = pars, digits_summary = 6)
```

```
## 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%
## eta[1]          0.267796 0.003797 0.139540    0.041412    0.164186
## eta[2]          0.009294 0.000146 0.005043    0.003374    0.005980
## beta[1]         0.908833 0.010657 0.395677    0.236166    0.616563
## beta[2]         0.783532 0.007368 0.333366    0.211888    0.546602
```

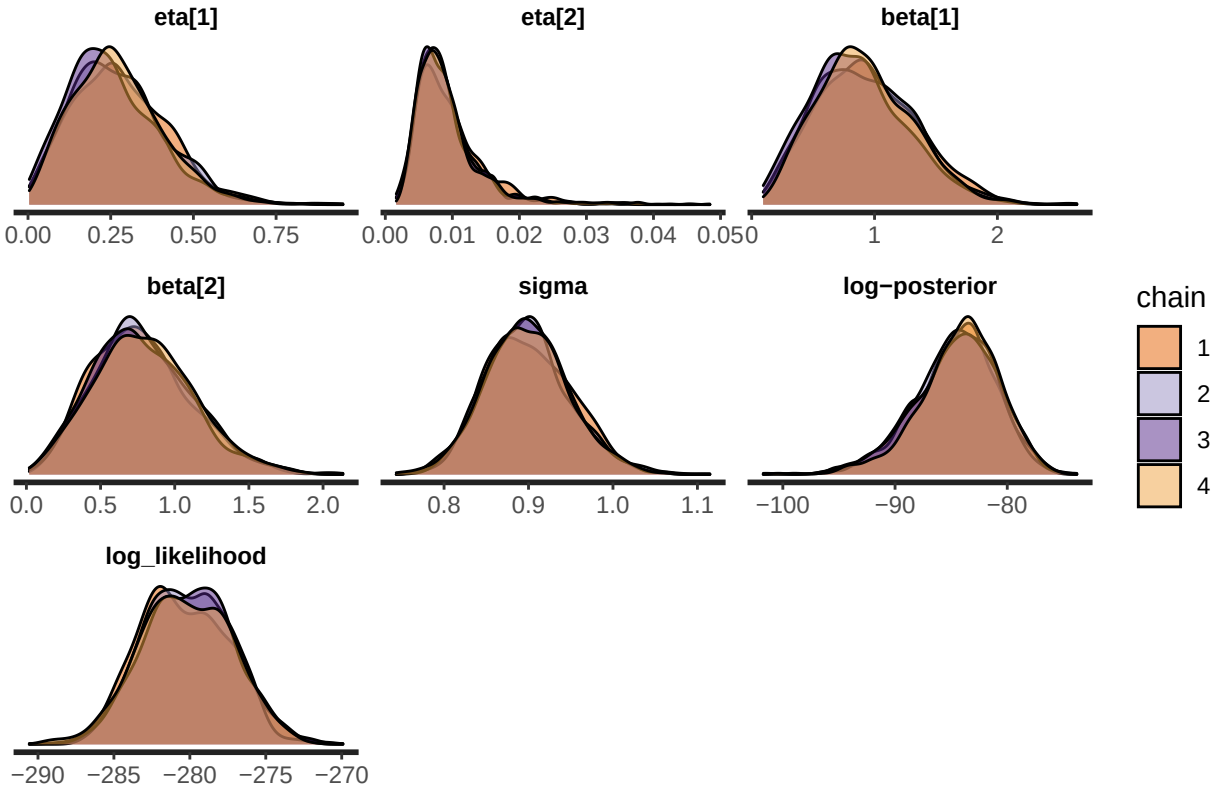
```
## sigma              0.899133 0.000901 0.049789    0.808415    0.864285
## lp__             -84.528699 0.097015 3.664896  -92.541482  -86.655464
## log_likelihood -280.205770 0.093672 2.922093 -285.616499 -282.330330
##                         50%         75%       97.5% n_eff     Rhat
## eta[1]             0.253170    0.353459    0.577214  1351 1.002581
## eta[2]             0.008115    0.011030    0.022925  1189 1.002460
## beta[1]            0.877521    1.174995    1.758455  1378 1.002306
## beta[2]            0.752258    0.994429    1.520031  2047 1.000784
## sigma              0.896932    0.930671    1.002227  3053 0.999787
## lp__             -84.173919  -81.981270  -78.180705  1427 1.001410
## log_likelihood -280.231502 -278.082878 -274.604763   973 1.000387
##
## Samples were drawn using NUTS(diag_e) at Sun Dec  6 20:04:28 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=pars)
```



Next, we can look at density plots for our posterior distribution to verify that the posterior distributions look uni-modal (assuming that's what we want) and relatively similar across chains. In this case, they do!

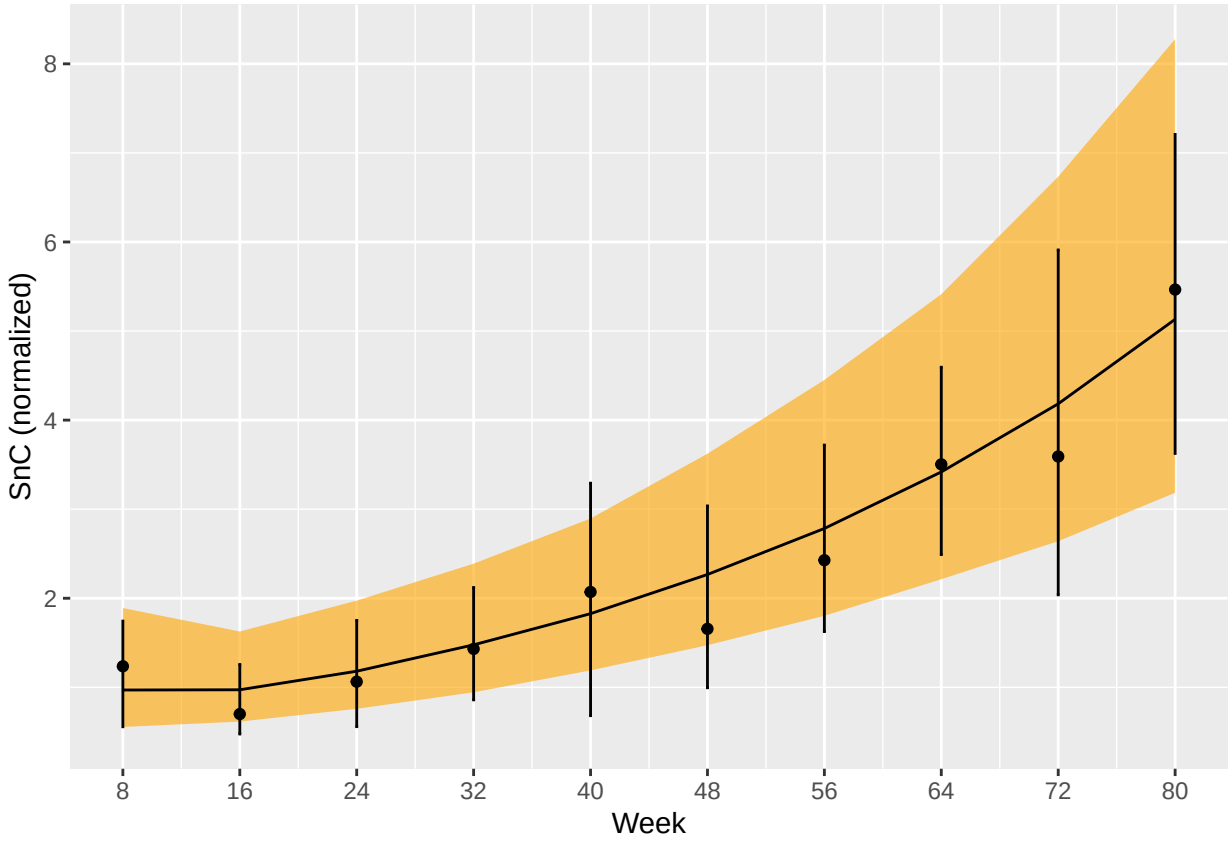
```
stan_dens(fit_snc_sr_model, pars = pars, separate_chains = TRUE)
```



Now we can actually evaluate our SR model with respect to data. As a first-pass evaluation, we'll basically replicate Karin et al's fig. 2c with only predictions from the SR model included for now. As the graph shows, our model seems to capture the progression of SnC abundance growth as well or, in my opinion, better than the max. likelihood equivalent in Karin et al. Our posterior predictive means – as we'd hope – look almost identical to Karin et al's but our 95% predictive intervals capture the variance (as measured by standard deviation) of the individual mouse estimates better than their SR model does.

```
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
)
colnames(snc_sr_counts_pred) <- make.names(colnames(snc_sr_counts_pred)) # to remove % in the col names
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 + stddev_abundances/2
    ),
    width=.2,
```

```
    position=position_dodge(0.05)
  ) +
  scale_x_continuous(name = "Week", breaks=t, labels=t, limits = c(8, 80)) +
  labs(x = "Week", y = "SnC (normalized)")
```



Fitting the USR model

Now, to enable us to validate that our comparison produces similar qualitative results to the original paper's, we're going to fit a second model. The biggest difference between this model and the first model is that lack of a saturated removal term. The saturated removal term in the prior model is intended to capture the phenomenon in which senescent cells become so abundant that the immune system becomes unable to efficiently remove them.

```
model <- stan_model("snc_usr_model.stan")
fit_usr_model <- sampling(
  model,
  data = data_snc,
  iter = niter,
  chains = 4
)
```

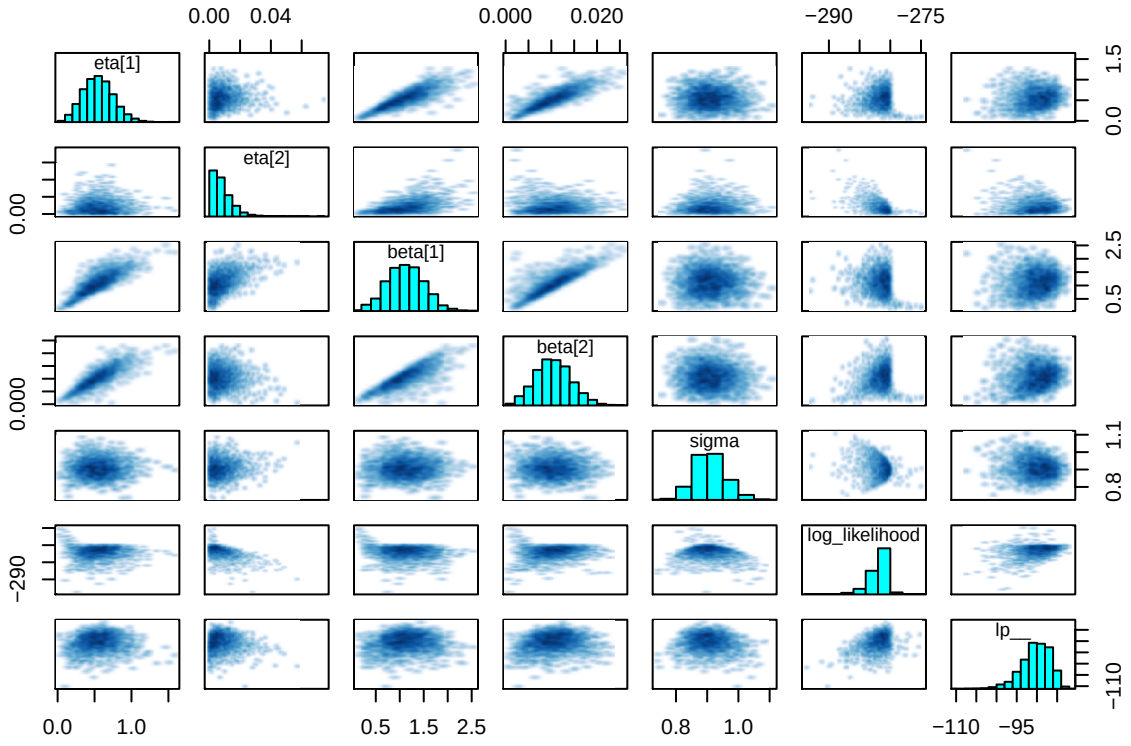
Diagnosing model fit

We look at the same diagnostics as previously and again find that our chains have mixed and our n_{eff}s seem reasonable. Our posterior densities also again look (mostly) unimodal.

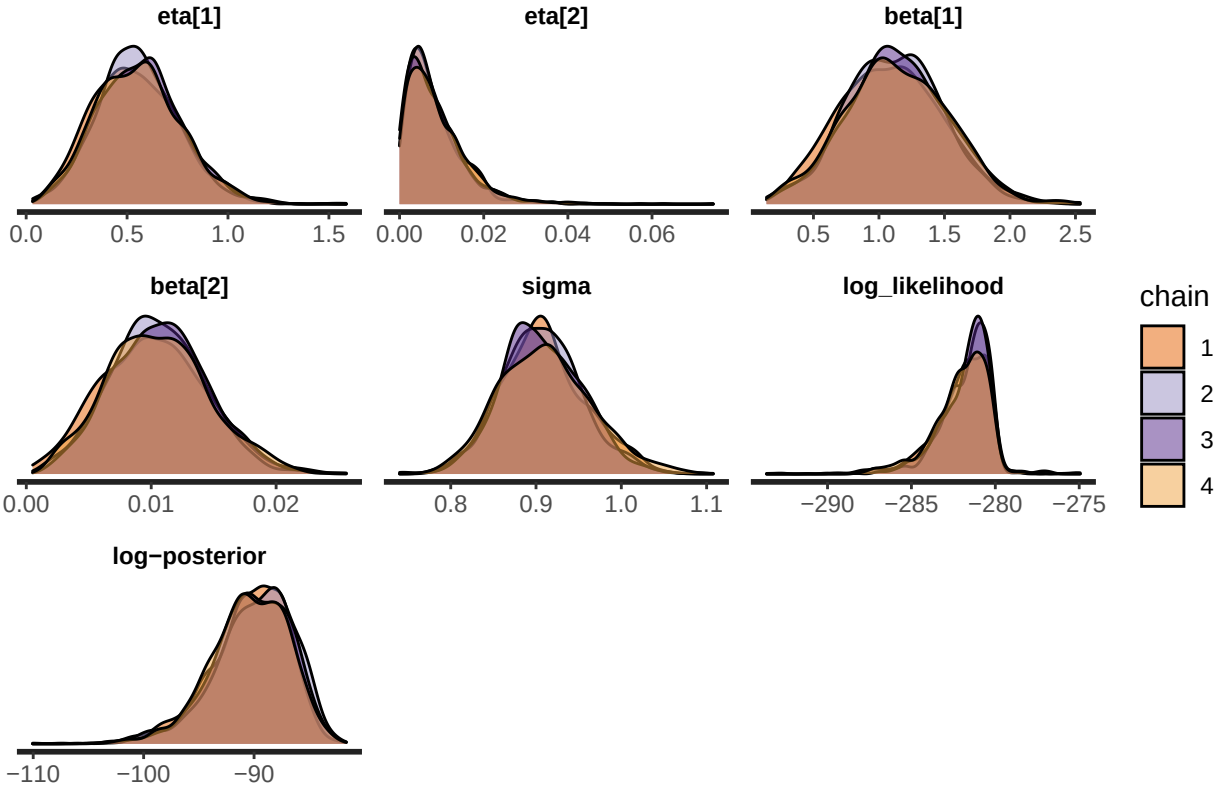
```
pars=c('eta', 'beta', 'sigma', 'log_likelihood', 'lp__')
print(fit_usr_model, pars = pars)
```

```
## 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%   97.5%
## eta[1]            0.55    0.01 0.21    0.17    0.41    0.55    0.69    0.98
## eta[2]            0.01    0.00 0.01    0.00    0.00    0.01    0.01    0.03
## beta[1]           1.12    0.01 0.38    0.37    0.86    1.11    1.38    1.88
## beta[2]           0.01    0.00 0.00    0.00    0.01    0.01    0.01    0.02
## sigma             0.91    0.00 0.05    0.82    0.87    0.91    0.94    1.01
## log_likelihood -281.89    0.04 1.61 -285.85 -282.68 -281.56 -280.77 -279.98
## lp__            -90.27    0.11 3.54  -98.50  -92.34  -89.98  -87.66  -84.60
##                 n_eff Rhat
## eta[1]           1467    1
## eta[2]           2097    1
## beta[1]          1324    1
## beta[2]          1370    1
## sigma            4787    1
## log_likelihood   1279    1
## lp__             1040    1
##
## Samples were drawn using NUTS(diag_e) at Sun Dec  6 20:20:36 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_usr_model, pars = pars)
```



```
stan_dens(fit_usr_model, pars = pars, separate_chains = TRUE)
```

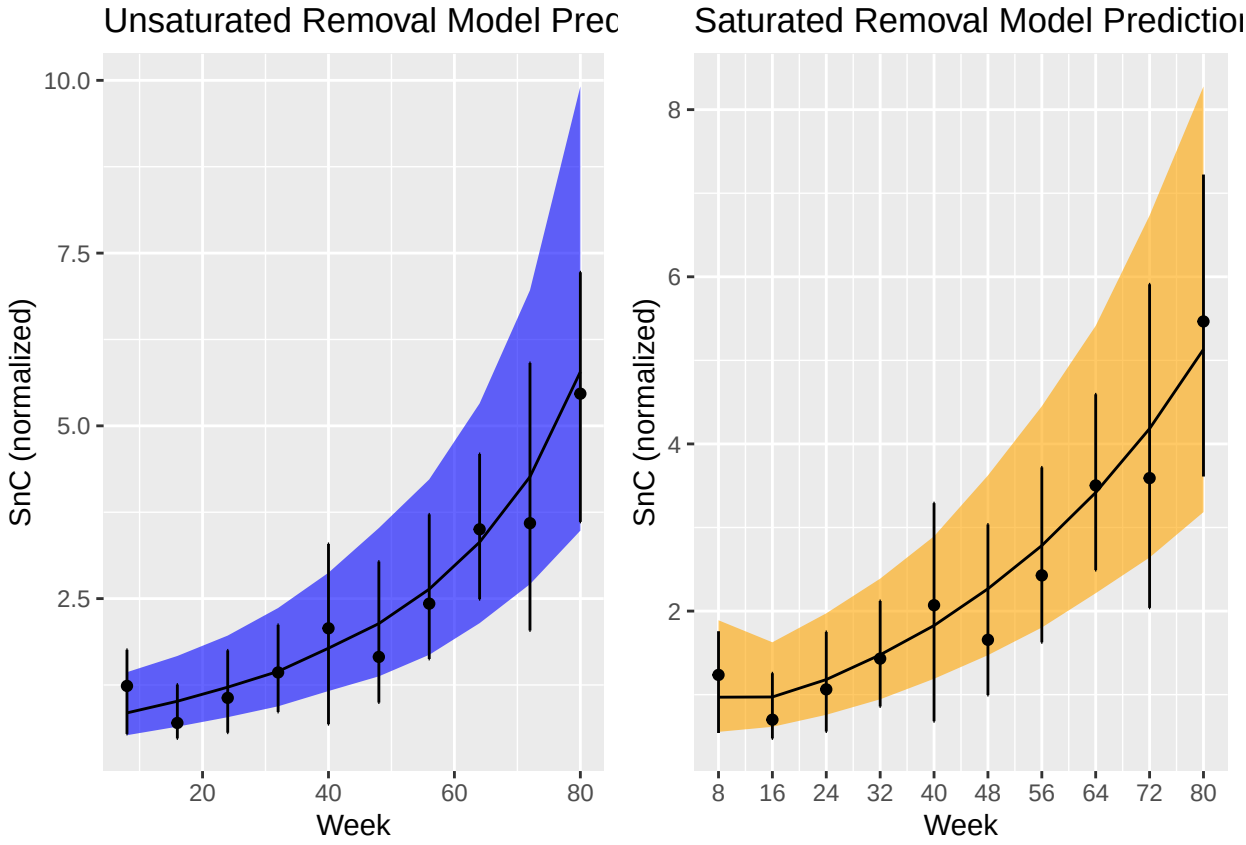


Finally, we can again look at how well our posterior intervals capture the true data points. To make qualitative comparison with the prior model easier we plot both model's fits on the graph.

```
snc_usr_counts_pred <- cbind(
  as.data.frame(
    summary(
      fit_usr_model, pars = "pred_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
colnames(snc_usr_counts_pred) <- make.names(colnames(snc_usr_counts_pred)) # to remove % in the col nam
p_usr <- ggplot(snc_usr_counts_pred, mapping = aes(x = t)) +
  geom_ribbon(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 + stddev_abundances/2,
    ),
    width=.2,
    position=position_dodge(0.05),
```

```
  ) +
  labs(title = "Unsaturated Removal Model Predictions", x = "Week", y = "SnC (normalized)", color = "nam

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 + 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")
ggarrange(p_usr, p_sr)
```



Interestingly, our USR graph looks much better than the one in the Alon paper with respect to matching the observations.

Fortunately, since we're being Bayesian about all this, we can improve over the Alon paper's original model comparison by using Bayes Factors to compute the odds ratio of our two models' marginal likelihoods (assuming equal prior probability between the two).

As we expected based on Karin et al, the SNC SR model is deemed much more probable than the USR model.

Note that the Bayes factor measures, assuming our two models are denoted $\mathcal{M}_1$, $\mathcal{M}_2$:

$$\frac{P(\mathcal{M}_1 \mid \text{data})}{P(\mathcal{M}_2 \mid \text{data})},$$

so this posterior odds ratio accounts for the relative sizes of the parameter spaces used by the two models.

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

```
## Estimated Bayes factor in favor of x1 over x2: 47.04496
```

```
post_prob(fit_snc_sr_model.bridge, fit_usr_model.bridge)
```

```
## fit_snc_sr_model.bridge    fit_usr_model.bridge
##              0.97918616              0.02081384
```