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

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(foreach)
library(ggplot2)
library(latex2exp)
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following objects are masked from 'package:foreach':
##
##     accumulate, when
```

```r
library(reshape2)
library(stringr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.4     v readr   1.4.0
## v tidyr   1.1.2     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x purrr::when()       masks foreach::when()
```

```r
# Stan-specific stuff
library(rstan)
```

```
## Loading required package: StanHeaders

## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

##
```

```
## Attaching package: 'rstan'

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(gridExtra)
```

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

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

```r
rstan_options (auto_write = TRUE)
options (mc.cores = parallel::detectCores ())
set.seed(3) # for reproductibility
```

Loading and visualizing the data

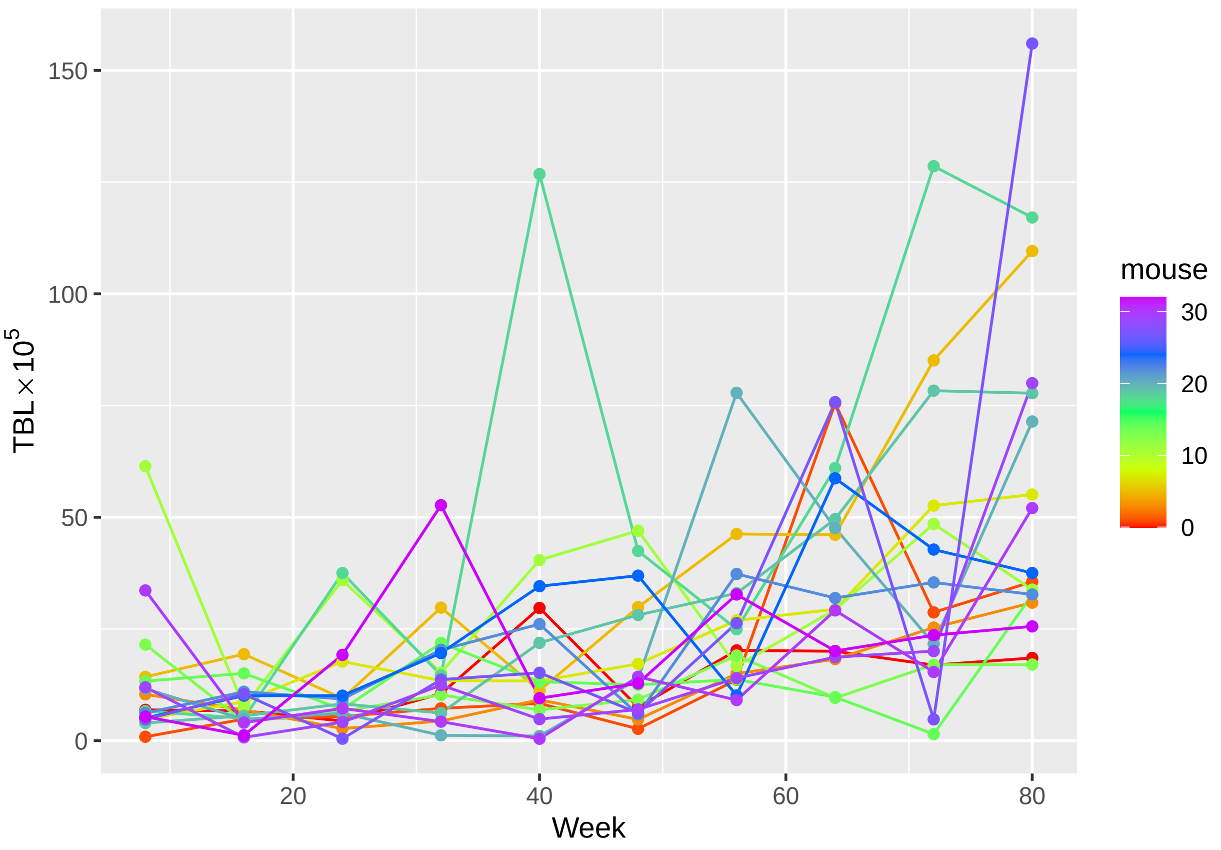
As our first step, we load the data we'll use to fit the 16 models described in Karin et al (TODO: actual reference). To do so, we load the raw data from their paper (used for their fig. 2) and transform it into a 3-column dataframe indexed by mouse number and week of measurement taken.

```r
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…

```r
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()
```



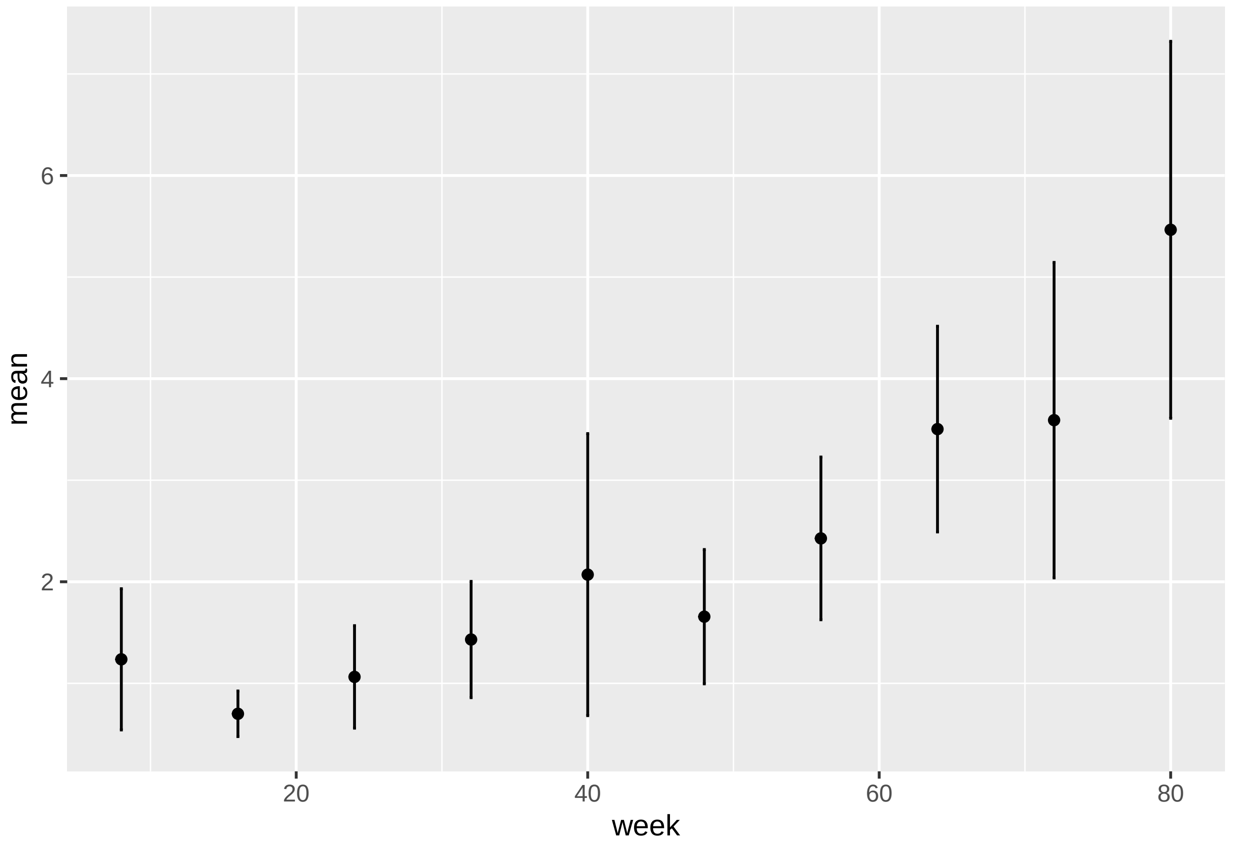
Next we're going to replicate the first part of Karin et al's fig 2c, which shows the mean and standard deviation of normalized senescent cell counts at each week time point.

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 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
```

```r
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
```

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

Diagnosing model fit

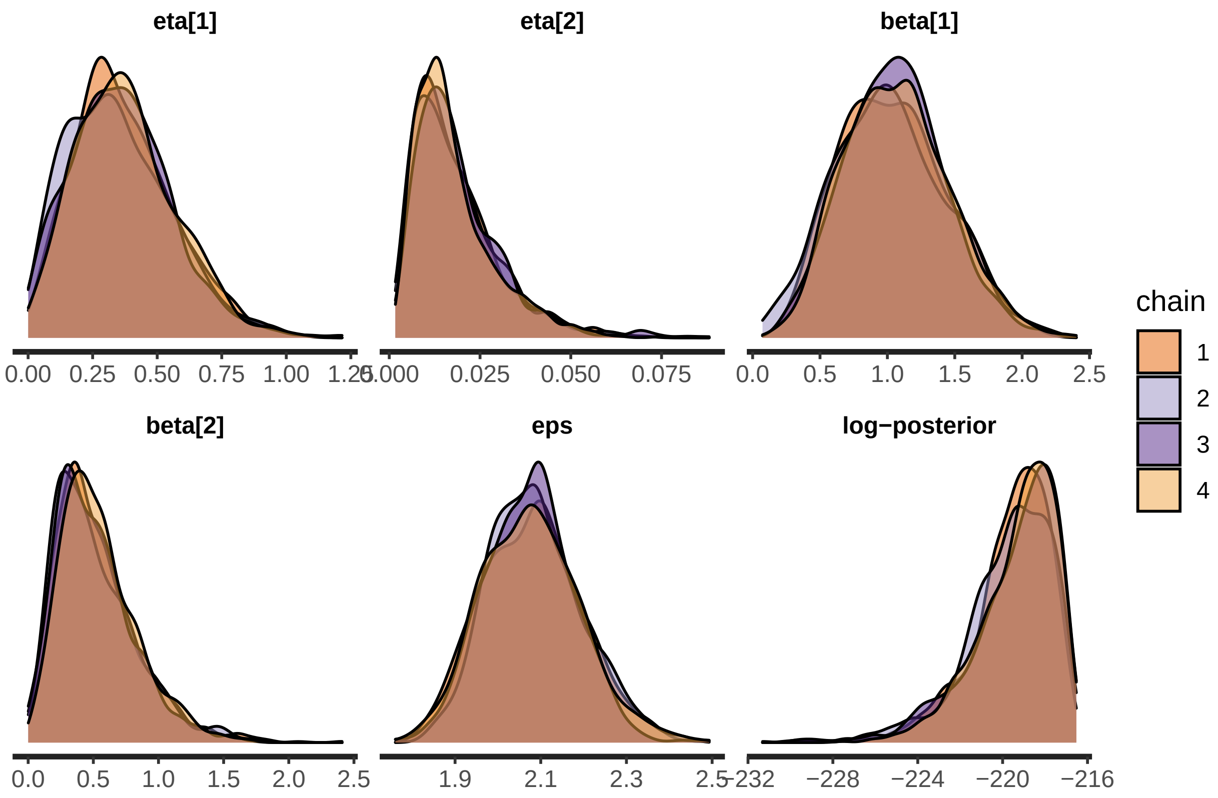
Inspired by this Stan case study and general best practices for checking the results of HMC, let's do some posterior checks to verify HMC ended up sampling a reasonable approximation of the true posterior. First, we can look at $\hat{R}$ and n_{eff}. $\hat{R}$ measures the level of agreement between 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.

```r
pars=c('eta', 'beta', 'eps', 'lp__')
print(fit_karin_et_al, pars = pars)
```

```
## 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%  97.5% n_eff Rhat
## eta[1]      0.37    0.01 0.20   0.05   0.22   0.34   0.49   0.80  1040    1
## eta[2]      0.02    0.00 0.01   0.00   0.01   0.02   0.02   0.05  1027    1
## beta[1]     1.05    0.01 0.39   0.35   0.76   1.03   1.30   1.84  1337    1
## beta[2]     0.52    0.01 0.31   0.09   0.30   0.46   0.69   1.28  1200    1
## eps        2.08    0.00 0.11    1.88    2.00    2.08    2.15    2.31  1786    1
## lp__    -219.52    0.07 1.97 -224.34 -220.55 -219.14 -218.05 -216.94   699    1
##
## Samples were drawn using NUTS(diag_e) at Wed Dec  2 10:33:31 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).
```

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!

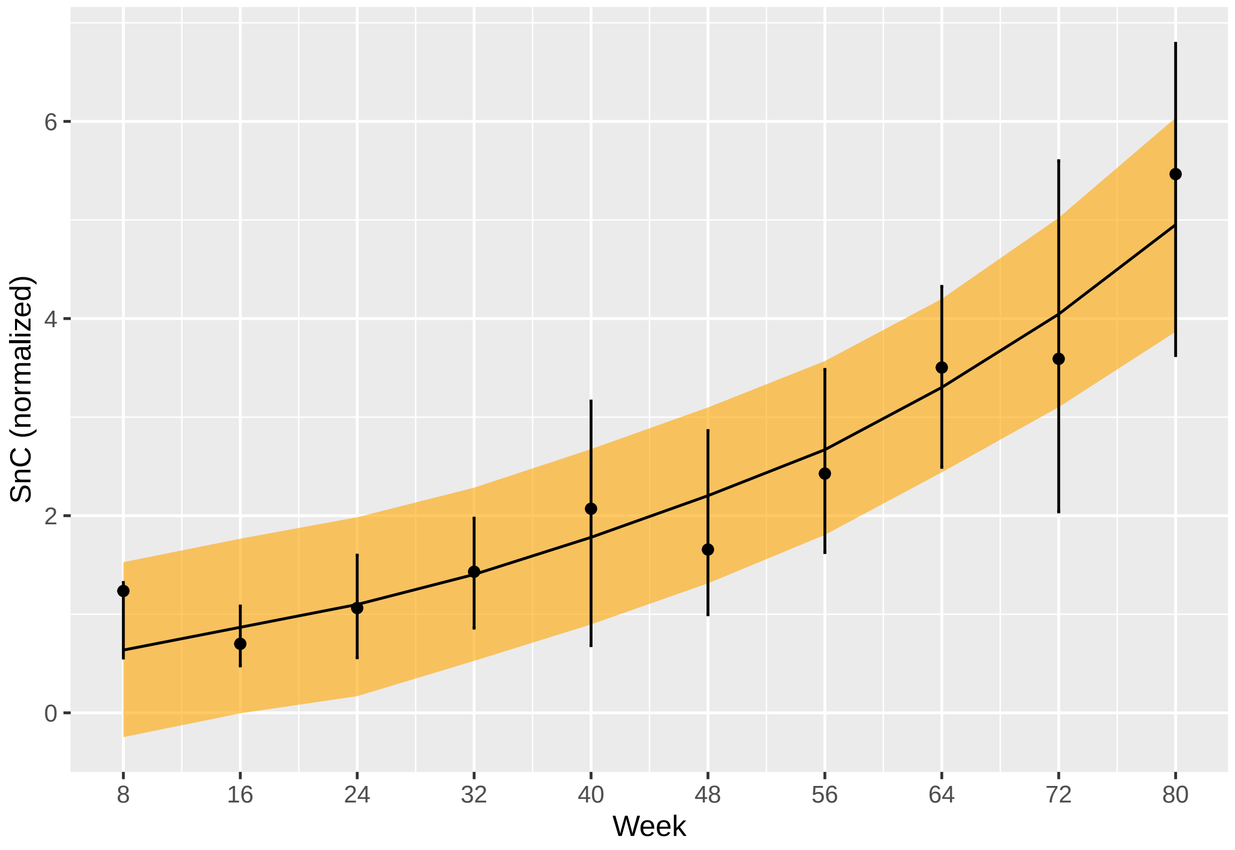
```
stan_dens(fit_karin_et_al, pars = pars, separate_chains = TRUE)
```



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_counts_pred <- cbind(
  as.data.frame(
    summary(
      fit_karin_et_al, pars = "pred_cells_means",
      probs = c(0.05, 0.5, 0.95)
    )$summary
  ),
  t,
  mean_abundances,
  stddev_abundances
)
```

```r
colnames(snc_counts_pred) <- make.names(colnames(snc_counts_pred)) # to remove % in the col names
ggplot(snc_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)")
```


SnC (normalized)
Week