Appendix A: Stochastic simulations of noisy phenomena

Contents

Getting started

This appendix includes the code to create all simulated data shown in the paper (and one additional example involving stochastic inflation that is not shown.) We must first load a variety of packages we use to create and display the simulations. The `regimeshifts` package provides an example of the model of Dai et al. (2012) and must be installed separately from GitHub, all other packages can be found on CRAN. A reproducible Dockerfile for building an image containing all necessary software to run the examples is also provided in the associated GitHub repository for this paper: https://github.com/cboettig/noise-phenomena.

```
library(nimble)
library(RcppRoll)
# devtools::install_github("cboettig/regimeshifts")
library(regimeshifts)
library(adaptivetau)  # For Gillespie simulation only
library(tidyverse)
library(ggthemes)
library(gridExtra)
library(scales)
# Set a default color pallete
colours <- ptol_pal()(2)
```

Gillespie algorithm

Levins' Patch model (Levins 1969) as an individual-based birth-death process:

$$\frac{dn}{dt} = \underbrace{cn\,(1 - n/N)}_{\text{birth rate, } b(n)} - \underbrace{en}_{\text{death rate, } d(n)}$$

We simulate this using the Gillespie's exact stochastic simulation algorithm (SSA) (Gillespie 1977; see overview in Hastings and Gross 2012), as implemented in the `adaptivetau` R package. We declare the transition events: birth increases the state n to $n+1$, and death n to $n-1$, and describe the rates associated with them:

```
# indicate state variable and the change
transitions = list(c(n = +1), # birth event
                   c(n = -1)) # death event

rateF <- function(state, params, t) {
  c(params$c * state[["n"]] * (1 - state[["n"]] / params$N), # birth rate
    params$e * state[["n"]])                                # death rate
}
```

We compare the stochastic simulation with identical parameters, $c = 1$, $e = 0.2$ except for the total number of available sites. In our "small" system scenario, we have a total of $N = 100$ sites, while in the "large" system we have $N = 1000$ sites.

The theory predicts an equilibrium population size $\bar{n}$ at $b(\hat{n}) = d(\hat{n})$, or

$$\left(1 - \tfrac{e}{c}\right) N,$$

and a variance of

$$\left.\frac{b(n) + d(n)}{2d'(n) - b'(n)}\right|_{n=\hat{n}} = \tfrac{e}{c} N$$

Exact Gillespie simulations can be run efficiently in R using the `adaptivetau::ssa.exact()` function:

```
set.seed(1234) # set random number generator seed to be reproducible

e <- 0.2 # patch death rate
c <- 1   # patch colonization rate

# Run simulation for small and large N
sim_small <- adaptivetau::ssa.exact(init.values = c(n = 50),
                        transitions, rateF,
                        params = list(c=c, e=e, N=100),
                        tf=30) %>% as_tibble()

sim_large <- adaptivetau::ssa.exact(init.values = c(n = 500),
                        transitions, rateF,
                        params = list(c = c, e = e, N = 1000),
                        tf=30) %>% as_tibble()
```

We will also create a separate data frame (`tibble`) corresponding to our theoretical predictions from the system size expansion. We use popular functions from the `tidyverse` suite of R packages for the manipulation of `data.frames`. Below, the `mutate` function is used to add new columns for plus and minus one standard deviation based on theoretical calculation of $\sqrt{N\frac{e}{c}}$, and the `map_dfr` function is used to apply this to both `small` and `large` cases. We use `tidyverse` syntax (see Ross, Wickham, and Robinson 2017) instead of base R in these examples for the following reasons:

- the resulting code is more concise, due to use of higher-level relational data abstractions
- the resulting code is closer to natural semantics. This is in part because consecutive steps are connected by pipes `%>%`rather than temporary variable assignment, and in part by the use of functions named by verbs corresponding to standard relational data concepts.
- `tidyverse` is widely used and well documented, see http://r4ds.had.co.nz/ for an accessible and thorough introduction.

```
theory <-
list(small = 100,large = 1000) %>%
```

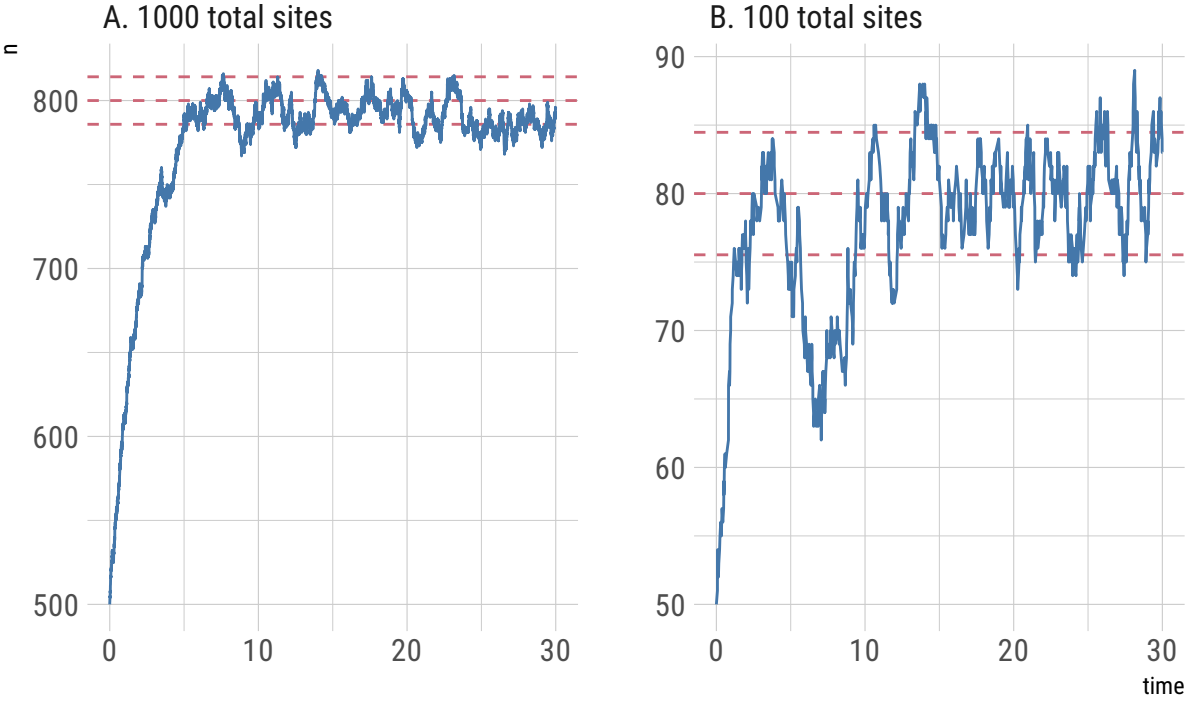
```
  map_dfr(function(N){
    data.frame(mean = N * (1-e/c)) %>%
    mutate(plus_sd = mean + sqrt(N * e / c),
           minus_sd = mean - sqrt(N * e / c))
  }, .id = "system_size")
```

Bind together both simulation results and theory results and store output data in a plain text `.csv` file.

```
# combine data.frames, using column called "system_size"
# to indicate large noise or small noise simulations
gillespie <-
  bind_rows("large" = sim_large,
            "small" = sim_small,
            .id = "system_size") %>%
  bind_rows(theory)

# Write data to file
write_csv(gillespie, "gillespie.csv")
```

Example plotting command to generate the figure as shown in main text, (using `ggplot2` plotting methods from `tidyverse`):

```
read_csv("../appendixA/gillespie.csv", col_types = "cdiddd")  %>%
  mutate(system_size = recode(system_size,
                              large = "A. 1000 total sites",
                              small= "B. 100 total sites")) %>%
  ggplot(aes(x = time)) +
  geom_hline(aes(yintercept = mean), lty=2, col=colours[2]) +
  geom_hline(aes(yintercept = minus_sd), lty=2, col=colours[2]) +
  geom_hline(aes(yintercept = plus_sd), lty=2, col=colours[2]) +
  geom_line(aes(y = n), col=colours[1]) +
  facet_wrap(~system_size, scales = "free_y")
```



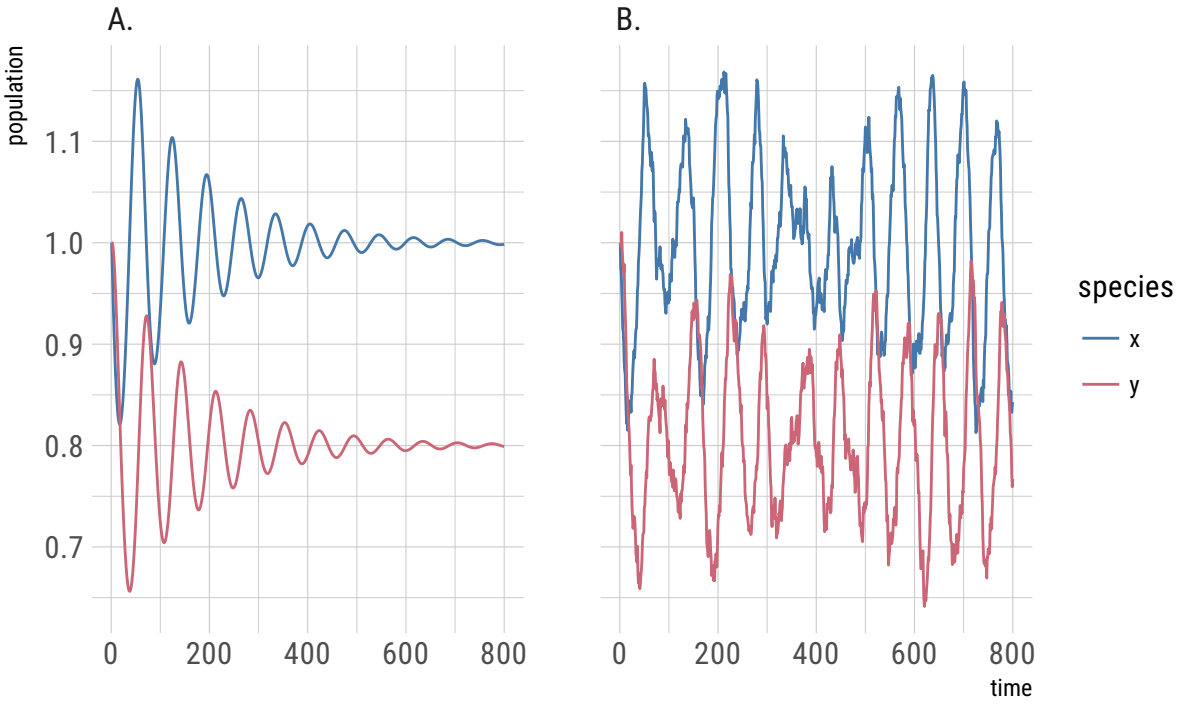
Quasi-cycles

Simulations of the quasi-cycles can easily be performed in NIMBLE, which allows us to express the stochastic models in the familiar BUGS syntax. Change the values of the model and noise parameters below to explore these results. While the models below can easily be implemented in base R, the BUGS syntax for expressing a stochastic model provides a more generic formulation that can easily be used in parameter estimation as well; see de Valpine et al. (2017) for details. This can also improve computational performance because NIMBLE compiles these models into C++ code.

```
# Define the BUGS model:
quasicycle  <- nimble::nimbleCode({
  x[1] <- x0
  y[1] <- y0
  for(t in 1:(N-1)){
    # Deterministic terms look like normal R code:
    mu_x[t] <- x[t] + x[t] * r * (1 - x[t] / K) - b * x[t] * y[t]
    # Stochastic assignment uses ~ to indicate 'distributed as' normal (Gaussian)
    x[t+1] ~ dnorm(mu_x[t], sd = sigma_x)
    mu_y[t] <- y[t] + c * x[t] * y[t] - d * y[t]
    y[t+1] ~ dnorm(mu_y[t], sd = sigma_y)
  }
})

# creates a data.frame of parameter combinations for each noise level
p <-
  data.frame(
  data.frame(r = .1, K = 5,  b = .1, c = .1, d = .1, N = 800),
  data.frame(sigma_x = c(.00001,0.01),  sigma_y = c(.00001,0.01)))

# define a function to perform a simulation given a set of parameters (constants)
f <- function(constants){
  model <- compileNimble(nimbleModel(quasicycle,
                                     constants = constants,
                                     inits = list(x0 = 1, y0 = 1)))
  # set random seed for reproducibility
  set.seed(123)
  # Run the actual simulation, using nimble
  simulate(model)

  # Assemble the results into a data.frame (tibble).
  tibble(t = seq_along(model$x),
         x = model$x,
         y = model$y,
         sigma = constants$sigma_x)
}

# Do this simulation for each set of parameters, given
quasicycle_df <- p %>% rowwise() %>% do(f(.))

# Write results to file for later reference and plotting
write_csv(quasicycle_df, "quasicycles.csv")
```

From this quasi-cycle simulation data we can recreate the plot shown in Figure 2 as follows:

```r
read_csv("quasicycles.csv") %>%
  # Data in long from
  gather(species, population, -t, -sigma) %>%
  # Re-arrange and and re-lable for the plots:
  mutate(sigma = as.factor(sigma)) %>%
  mutate(sigma = recode(sigma, "1e-05" = "A.", "0.01" = "B.")) %>%
  rename(time = t) %>%
  ggplot(aes(time, population, col = species)) + geom_line() +
  facet_wrap(~ sigma, ncol=2) + scale_color_ptol()
```



Stochastic oscillator

Our example of a stochastic oscillator is based on May's model:

$$X_{t+1} = X_t + \underbrace{X_t r \left(1 - \frac{X_t}{K}\right)}_{\text{Vegitation growth}} - \underbrace{\frac{a X_t^Q}{X_t^Q + H^Q}}_{\text{Vegitation consumption}} + \xi_t,$$

```r
p <- list(r = .5, K = 2, Q = 5, H = .38, sigma = .04, a = 0.245, N = 1e4)

# Define stochastic model in BUGS notation
may  <- nimble::nimbleCode({

  x[1] <- x0
  for(t in 1:(N-1)){
    # Determinstic mean looks like standard R
    mu[t] <- x[t] + x[t] * r * (1 - x[t] / K)  - a * x[t] ^ Q / (x[t] ^ Q + H ^ Q)
    # Note the use of ~ in BUGS to show 'distributed as normal'
    y[t+1] ~ dnorm(mu[t], sd = sigma)
```

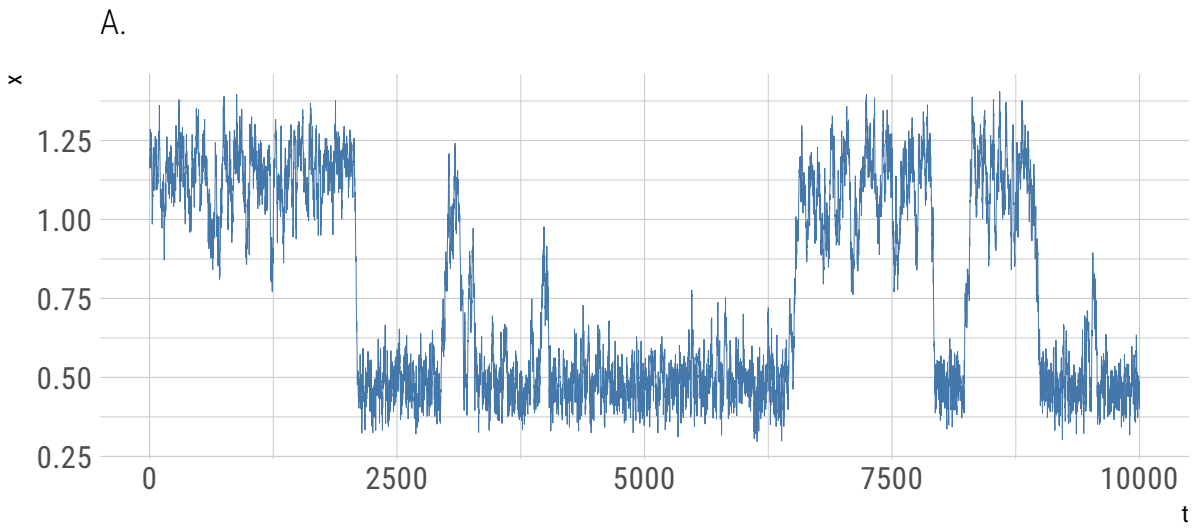
```
    x[t+1] <- max(y[t+1],0)
  }

})

model <- nimbleModel(may,constants = p, inits = list(x0 = 1.2))
cmodel <- compileNimble(model)
set.seed(123)
simulate(cmodel)

tibble(t = seq_along(cmodel$x), x = cmodel$x) %>%
  write_csv("noisy_switch.csv")
```

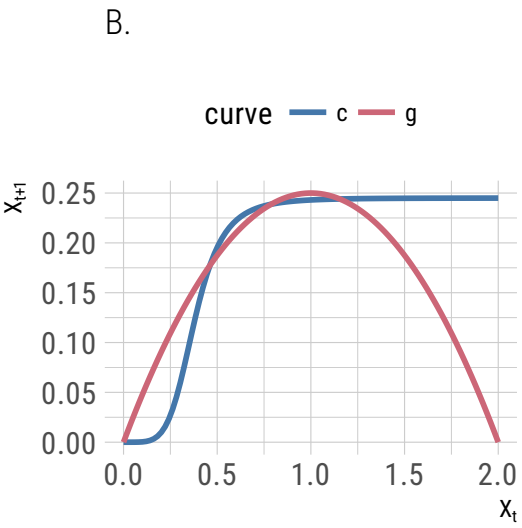
Separately, we will also compute the growth and consumption curves and the potential function for comparison with theory:

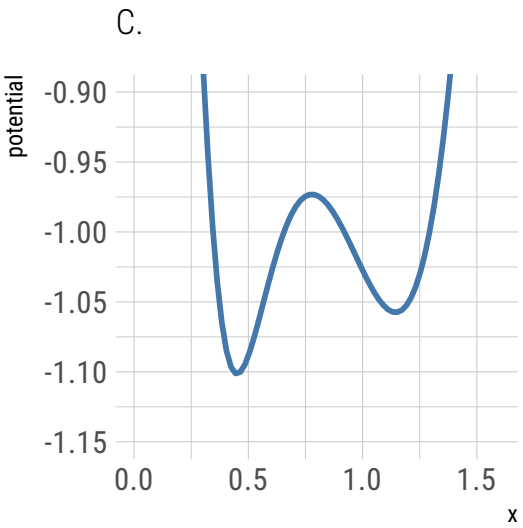
```
theory <-
  tibble(x= seq(0,2, length.out = 100)) %>%
  # As before, we add columns with mutate
  mutate(g = x * p$r * (1 - x / p$K),
         c = p$a * x ^ p$Q / (x^p$Q + p$H^p$Q)) %>%
  mutate(potential = - cumsum(g - c)) %>%
  # assemble resulting data into "long" form with the
  # column headings as named here:
  gather(curve, y, -x, -potential)
```

We can now create each of the subplots shown in the paper in Figure 3:

```
p1 <- read_csv("../appendixA/noisy_switch.csv")  %>%
  ggplot(aes(t,x)) +
  geom_line(lwd=.1, col=colours[[1]]) +
  labs(subtitle = bquote(bold(A.)))

p2 <-  theory %>%
  ggplot(aes(x, y, col=curve)) +
  geom_line(lwd=1) +
  scale_color_ptol() +
  labs(x = bquote(X[t]), y = bquote(X[t+1]), subtitle= bquote(bold(B.))) +
  theme(legend.position="top")

p3 <-  theory %>%
  ggplot(aes(x, potential)) +
  geom_line(col=colours[1], lwd=1) +
  coord_cartesian(xlim=c(0,1.6), ylim=c(-1.15,-0.9)) +
  labs(subtitle = bquote(bold(C.)))

gridExtra::grid.arrange(p1,p2,p3, layout_matrix = rbind(c(1,1), c(2,3)))
```







Tipping points

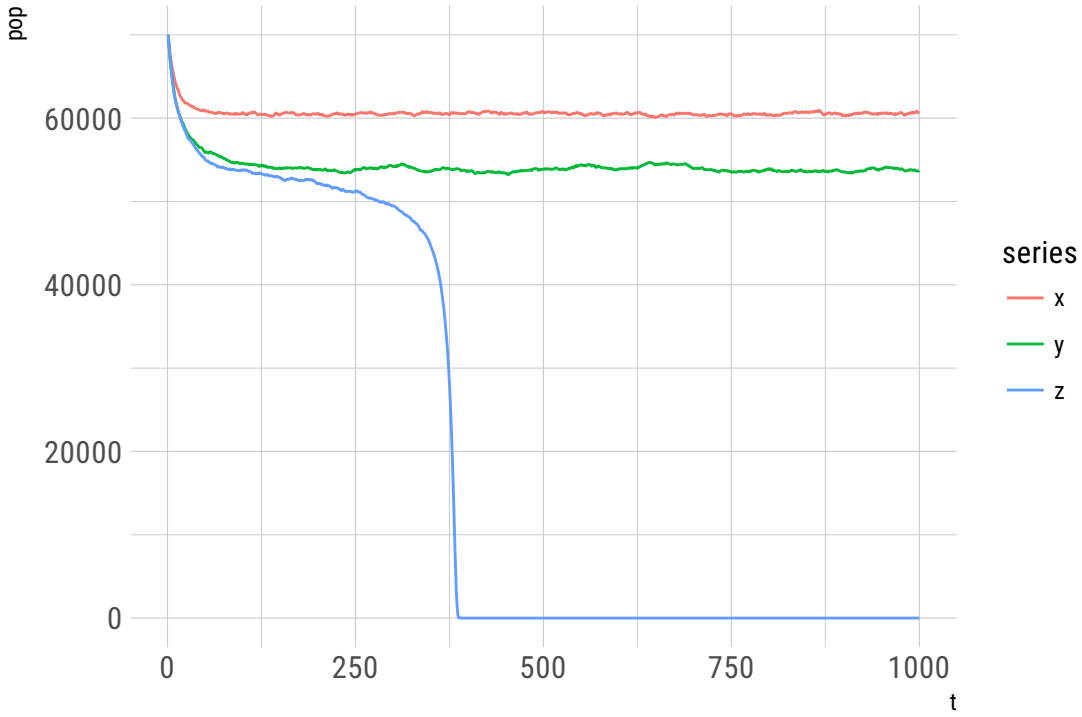
Configuration for Dai model

Though not shown in the paper, it is useful to illustrate tipping point location using very small noise in a constant environment and varying only the initial condition. Starting just above the tipping point leads to a steady stable state, starting just below the tipping point leads to a sudden crash.

Details of the Dai model are found in the appendix of Dai et al. (2012), my best interpretation of that description can be found in R code in the `regimeshifts` package, https://github.com/cboettig/regimeshifts, with archival version available at https://doi.org/10.5281/zenodo.1013975.

```
max_days <- 1000
# Perform three simultaneous simulations, x,y,z, using
# different dilution factors DF
```

```r
# Intialize variables
z <- y <- x <- numeric(max_days)
z[1] <- y[1] <- x[1] <- 7e4 # 1.7e5

# set random seed for reproduciblity
set.seed(123)

# simulate for max_days
for(day in 1:(max_days-1)){
  x[day+1] <- dai(x[day], epsilon = rnorm(1,0, 0.001), DF = 1790)
  y[day+1] <- dai(y[day], epsilon = rnorm(1,0, 0.001), DF=1799)
  z[day+1] <- dai(z[day], epsilon = rnorm(1,0, 0.001), DF = 1800)

}

# gather results into long data.frame
df <- data.frame(t = 1:max_days, x = x, y = y, z = z) %>%
  gather(series, pop, -t)

# and plot results
ggplot(df, aes(t, pop, col=series)) +
  geom_line()
```



We will also need a function for rolling auto-correlation for our warning sign calculation:

$$\rho = \frac{1}{n-1} \frac{\sum_{i=1}^{n} \left(x_{i,t} - \mathrm{E}(x_t)\right) \left(x_{i,t+1} - \mathrm{E}(x_{t+1})\right)}{\sigma_{x_t} \sigma_{x_{t+1}}}$$

```r
roll_acor <- function(x, lag = 1, ...){
  x_t1 = lag(x, n = lag)
```

```
  mu_t = roll_mean(x, ...)
  mu_t1 = roll_mean(x_t1, ...)
  s_t = roll_sd(x, ...)
  s_t1 = roll_sd(x_t1, ...)
  acor = roll_mean( (x - mu_t) * (x_t1 - mu_t1) / (s_t * s_t1),...)
  acor
}
```

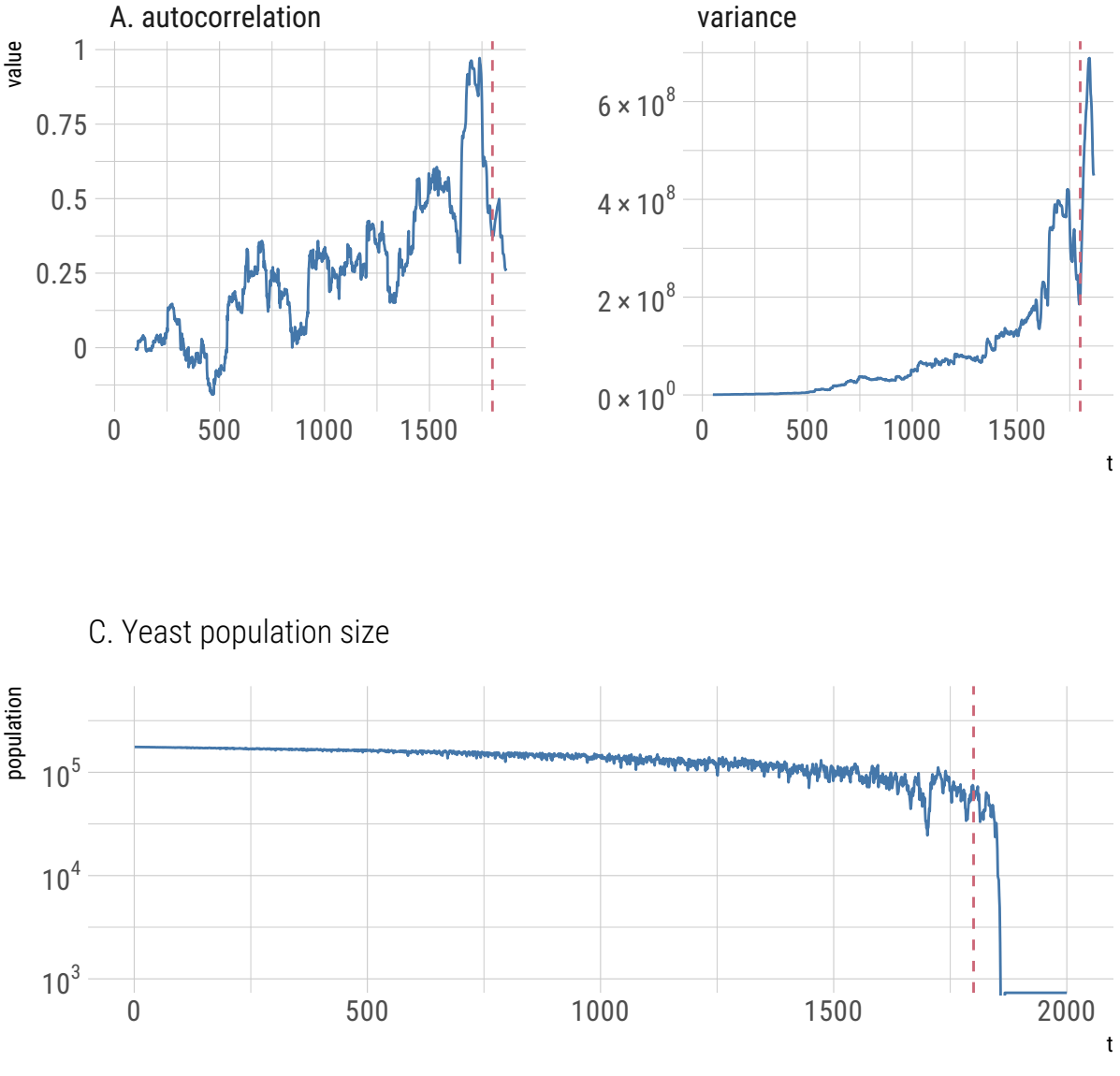
Simulation of Dai model:

```
# Slow environmental degredation through small stepwise changes
# (increasing the serial dilution factor, as in the Dai et al experiment.)
DF <- as.numeric(sapply(seq(0, 2000, length=9), rep, 40))

# continuous linear increase
DF <- seq(1, 2000, length=2000)
tip_time <- 1800
max_days <- length(DF)
y <- numeric(max_days)

set.seed(1111)
y[1] <- 1.76e5
for(day in 1:(max_days-1)){
  y[day+1] <- dai(y[day], DF = DF[day])
}

tip_df <- tibble(t = seq_along(y), x = y)

# also compute warning signs, include in single table
tip_df %>%
  mutate(autocorrelation = roll_acor(x, n = 100, fill=NA),
         variance = roll_var(x, n = 100, fill=NA),
         tip_time = tip_time) %>%
  write_csv("tipping.csv")
```

We are now ready to create the panels shown in Figure 4. A little data tidying and `ggplot` layering creates the final plot:

```
tipping <- read_csv("../appendixA/tipping.csv")

p1 <- tipping %>%
  select(-x) %>%
  filter(t < 1866) %>%
  rename("A. autocorrelation" = autocorrelation, "variance" = variance) %>%
  gather(series, value, -t, -tip_time) %>%
  ggplot(aes(t, value)) + geom_line(col=colours[[1]]) +
  geom_vline(aes(xintercept = tip_time), col=colours[[2]], lty=2) +
  facet_wrap(~series, ncol = 2, scales="free_y") +
  scale_y_continuous(labels=scientific)
p2 <- tipping %>%
  select(t, population = x, tip_time) %>%
  ggplot(aes(t, population)) +
  geom_line(col=colours[1]) +
```

```
  geom_vline(aes(xintercept = tip_time), col=colours[2], lty=2) +
  scale_y_log10(labels=scientific)  +
  coord_cartesian(ylim=c(1e3, 5e5)) +
  labs(subtitle = "C. Yeast population size",
       caption = "Vertical red dashed line indicates tipping point location")

gridExtra::grid.arrange(p1,p2, layout_matrix = rbind(c(1,1), c(2,2)))
```



Note that Dai, Korolev, and Gore (2015) plots coefficient of variation, (standard deviation over mean), rather than the variance. This is a popular choice since the resulting indicator is a dimensionless measure of the noise relative to the mean, but also confounds changes in the mean dynamics with changes in the noise dynamics. This can give a misleading impression of the performance of the indicator: in many models, the mean (that is, the value of the equilibrium population size, $\hat{x}$) will also decrease prior to the transition, resulting in a more pronounced increase in the coefficient of variation than in the variance. However, changes in mean per se should are not a generic indicator of stability predicted by the theory – indeed, the whole point of the theory was to detect changes that would not be manifested in the mean. In general, systems

going through a saddle-node bifurcation can vary greatly in how much the mean changes. The mean will generally decrease, since it must eventually meet the 'saddle' point below it – it is a question of whether the saddle point does most of the increasing or the node does most of the decreasing.

Stochastic inflation

While stochastic inflation is mentioned only in a footnote of the manuscript, I include this example to illustrate simulations of an ensemble of replicates, commonly needed to get a better picture of stochastic phenomena. This example code is written for clarity rather than performance, and will take longer to run than the other examples.

Analytically predicted population size, as a fraction of K, is $\frac{1}{2} + \frac{1}{2}\sqrt{1 - 8\sigma^2/K^2}$. Note this is independent of r, and increases with σ/K, the larger the noise σ_g as a fraction of the carrying capacity K. Population sizes can also be inflated relative to their deterministic equilibrium whenever the second derivative is positive (as in the lower equilibrium in the May model of alternative stable states, considered below.) As predicted by the system size expansion, the deviation from the deterministic result is of order N^{-1}.

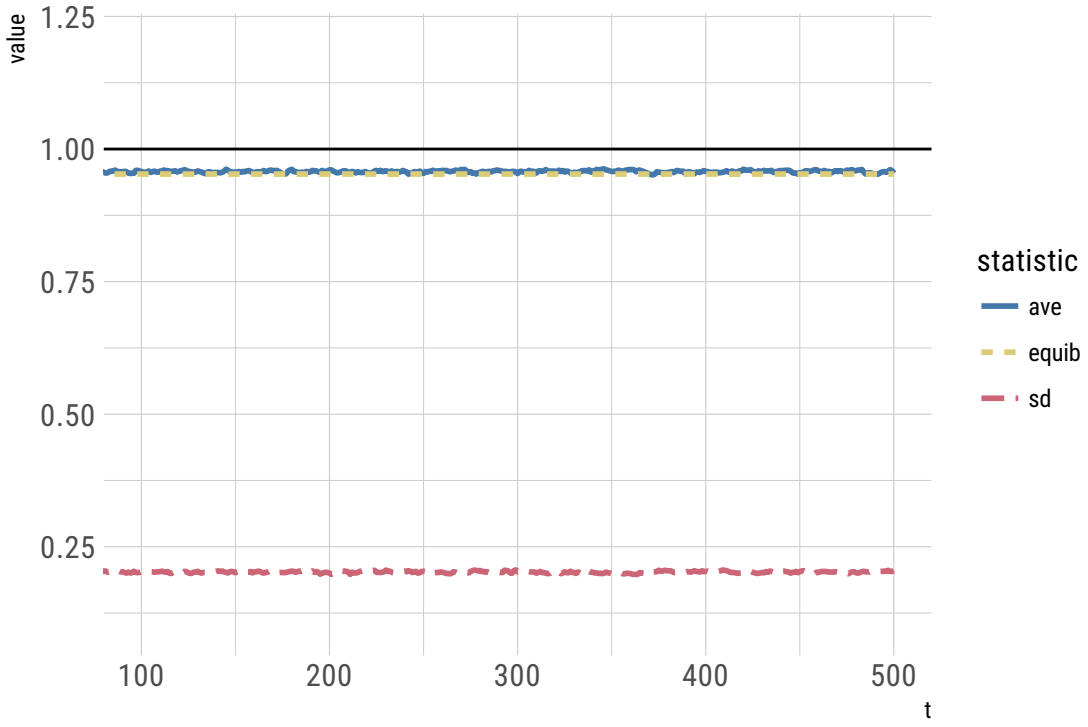
Because a substantial number of replicate simulations are required to derive a precise mean state, this example will be significantly slower to run than others.

```
# Parameters
p <- list(r = .4, K = 1, sigma = .15, N = 500)
equib <- (1 + sqrt(1 - 8 * p$sigma^2/p$K^2) )/ 2

# BUGS model
logistic  <- nimble::nimbleCode({
  x[1] <- x0
  for(t in 1:(N-1)){
    mu[t] <- x[t] + x[t] * r * (1 - x[t] / K)
    x[t+1] ~ dnorm(mu[t], sd = sigma)
  }
})

# Compile and run model
model <-
nimbleModel(logistic,
            constants = p,
            inits = list(x0 = 1))
cmodel <- compileNimble(model)
set.seed(123)

# Perform the simulations
df <- map_dfr(1:10000,
              function(rep){
                simulate(cmodel)
                data.frame(t = seq_along(cmodel$x), x = cmodel$x)
              },
              .id = "rep")

# Tidy up the result data and write to file,
# along with analytic prediction for equilibrium
df %>%
```

```
  filter(x > 0) %>%
  group_by(t) %>%
  summarise(ave = mean(x), sd = sd(x)) %>%
  mutate(equib = equib) %>%
  write_csv("inflation.csv")
```

Note that the average across stochastic replicates is slightly less than K=1 (black line), and matches almost perfectly the equilibrium predicted by the higher order correction theory. The standard deviation of fluctuations shown for reference as well.

```
read_csv("inflation.csv") %>%
  gather(statistic, value, -t) %>%
  ggplot(aes(t,value, col=statistic, lty=statistic)) +
  geom_line(lwd=1) +
  geom_hline(yintercept = 1, col="black") +
  scale_color_ptol() +
  coord_cartesian(xlim=c(100,500), ylim=c(0.1, 1.2))
```



References

Dai, Lei, Kirill S. Korolev, and Jeff Gore. 2015. "Relation between stability and resilience determines the performance of early warning signals under different environmental drivers." *Proceedings of the National Academy of Sciences*, 201418415. https://doi.org/10.1073/pnas.1418415112.

Dai, Lei, Daan Vorselen, Kirill S Korolev, and J. Gore. 2012. "Generic Indicators for Loss of Resilience Before a Tipping Point Leading to Population Collapse." *Science (New York, N.Y.)* 336 (6085):1175–7. https://doi.org/10.1126/science.1219805.

de Valpine, Perry, Daniel Turek, Christopher J. Paciorek, Clifford Anderson-Bergman, Temple Lang Duncan, and Rastislav Bodik. 2017. "Programming With Models: Writing Statistical Algorithms for General Model

Structures With NIMBLE." *Journal of Computational and Graphical Statistics* 26 (2). Taylor & Francis:403–13. https://doi.org/10.1080/10618600.2016.1172487.

Gillespie, Daniel. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions." *Journal of Physical Chemistry* 81.25:2340–61.

Hastings, Alan, and Louis J. Gross, eds. 2012. *Encyclopedia of Theoretical Ecology*. Oakland, CA: University of California Press.

Levins, Richard. 1969. "Some demographic and genetic consequences of environmental heterogeneity for biological control." *Bulletin of the Entomological Society of America* 15 (3). Entomological Society of America:237–40.

Ross, Zev, Hadley Wickham, and David Robinson. 2017. "Declutter your R workflow with tidy tools." *PeerJ Preprints* 5:e3180v1. https://doi.org/10.7287/peerj.preprints.3180v1.