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)
library(regimeshifts) ## devtools::install_github("cboettig/regimeshifts")
library(adaptivetau)
library(tidyverse)
library(ggthemes)
library(gridExtra)
colours <- ptol_pal()(2)
```

Gillespie algorithm

Levins' Patch model 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), 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
c <- 1

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:

```
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")
```

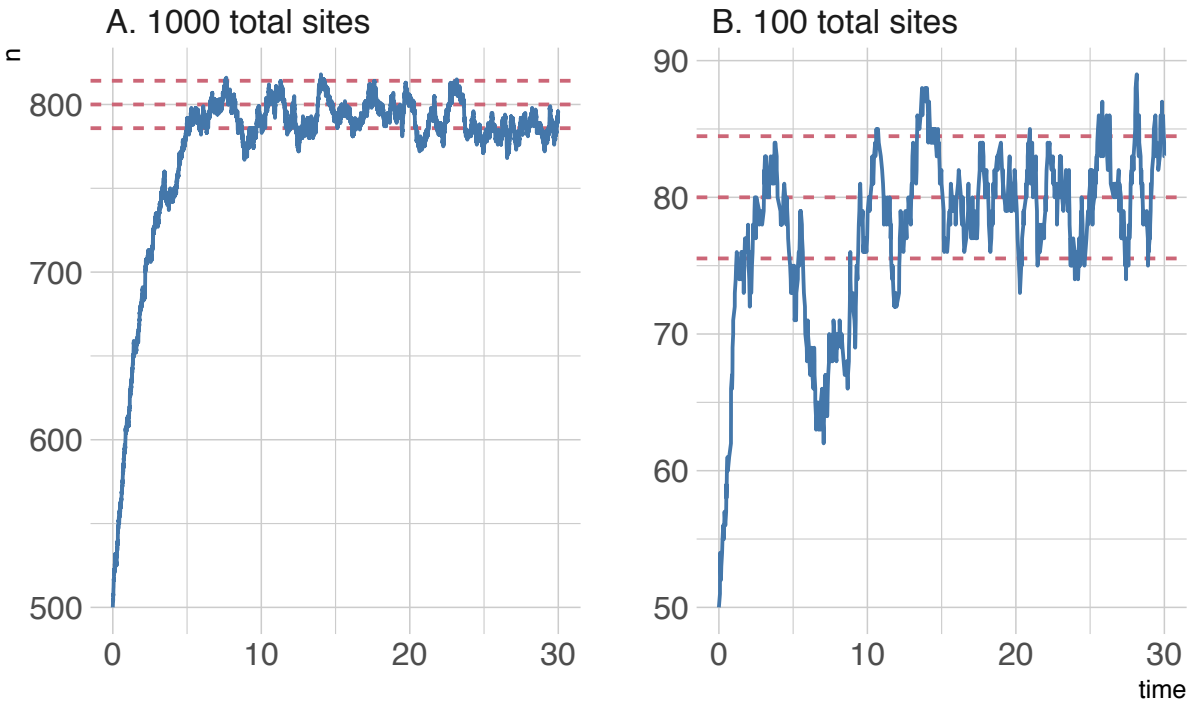
Bind together both simulation results and theory results and store output data in a plain text `.csv` file.

```
gillespie <- bind_rows("large"=sim_large,
                       "small"=sim_small,
                       .id = "system_size") %>%
```

```
  bind_rows(theory)
write_csv(gillespie, "gillespie.csv")
```

Example plotting command to generate the figure as shown in main text:

```
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

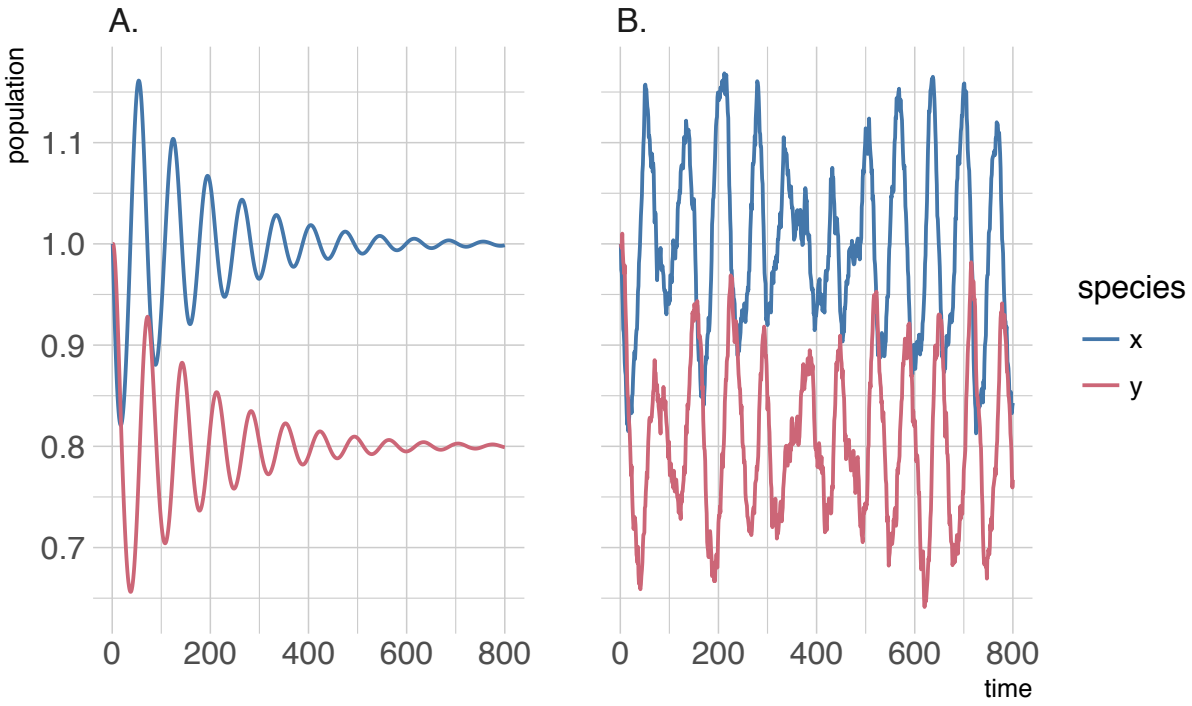
Simulations of the quasi-cycles can easily be performed in NIMBLE, which allows us to express the stochastic models in the familiar BUGS syntax. Change the values of the model and noise parameters below to explore these results.

```
quasicycle  <- nimble::nimbleCode({

  x[1] <- x0
  y[1] <- y0

  for(t in 1:(N-1)){
    mu_x[t] <- x[t] + x[t] * r * (1 - x[t] / K) - b * x[t] * y[t]
    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)
  }

})

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)))

f <- function(constants){
  model <- compileNimble(nimbleModel(quasicycle,
                                     constants = constants,
                                     inits = list(x0 = 1, y0 = 1)))
  set.seed(123)
  simulate(model)
  tibble(t = seq_along(model$x),
         x = model$x,
         y = model$y,
         sigma = constants$sigma_x)
}

quasicycle_df <- p %>% rowwise() %>% do(f(.))
write_csv(quasicycle_df, "quasicycles.csv")
```

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

```
read_csv("quasicycles.csv") %>%
  gather(species, population, -t, -sigma) %>%
  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 osccilato is based on May's model:

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

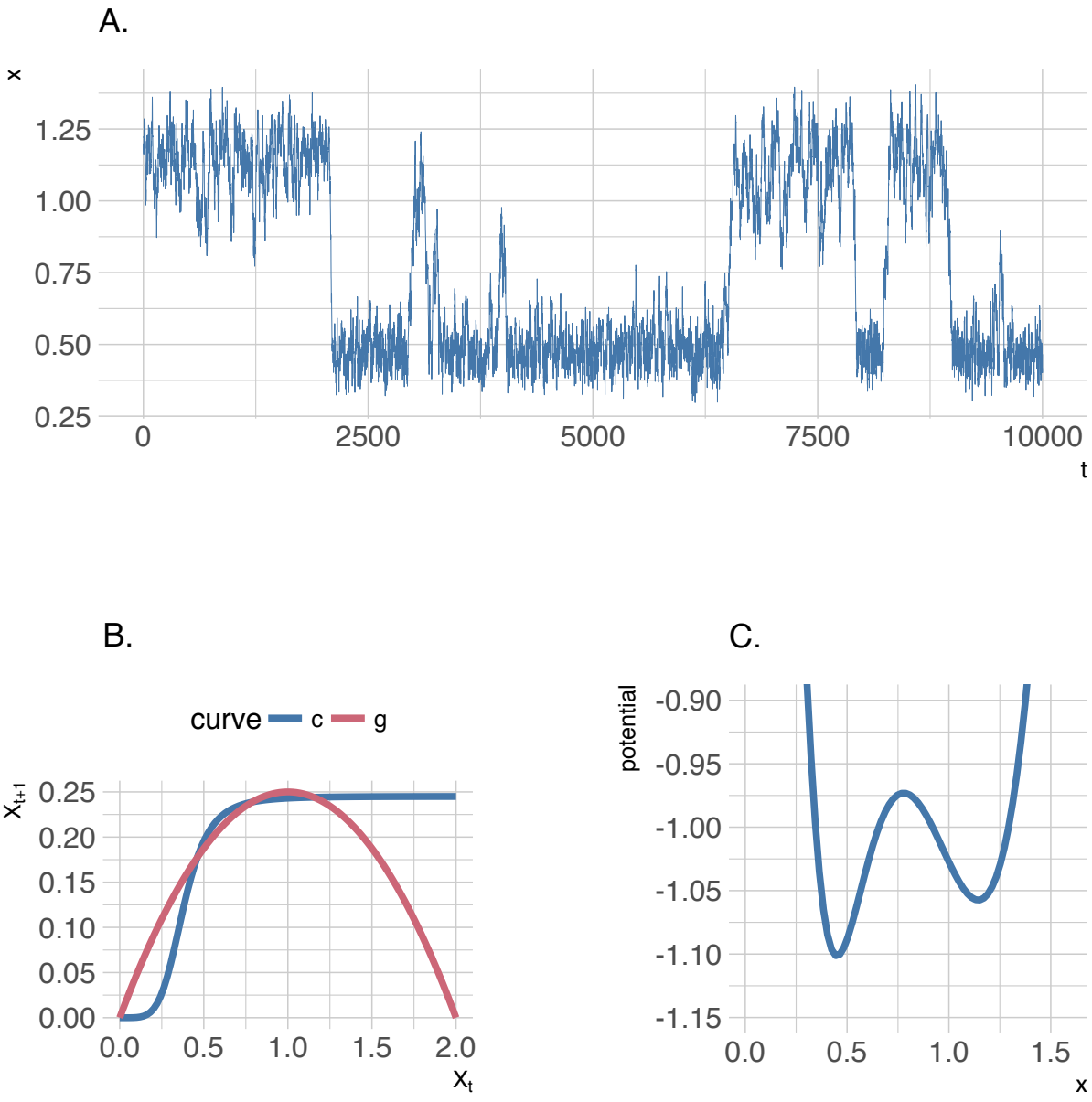
```
p <- list(r = .5, K = 2, Q = 5, H = .38, sigma = .04, a = 0.245, N = 1e4)

may  <- nimble::nimbleCode({

  x[1] <- x0
  for(t in 1:(N-1)){
    mu[t] <- x[t] + x[t] * r * (1 - x[t] / K)  - a * x[t] ^ Q / (x[t] ^ Q + H ^ Q)
    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")
```

Seperately, 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)) %>%
  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)) %>%
  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

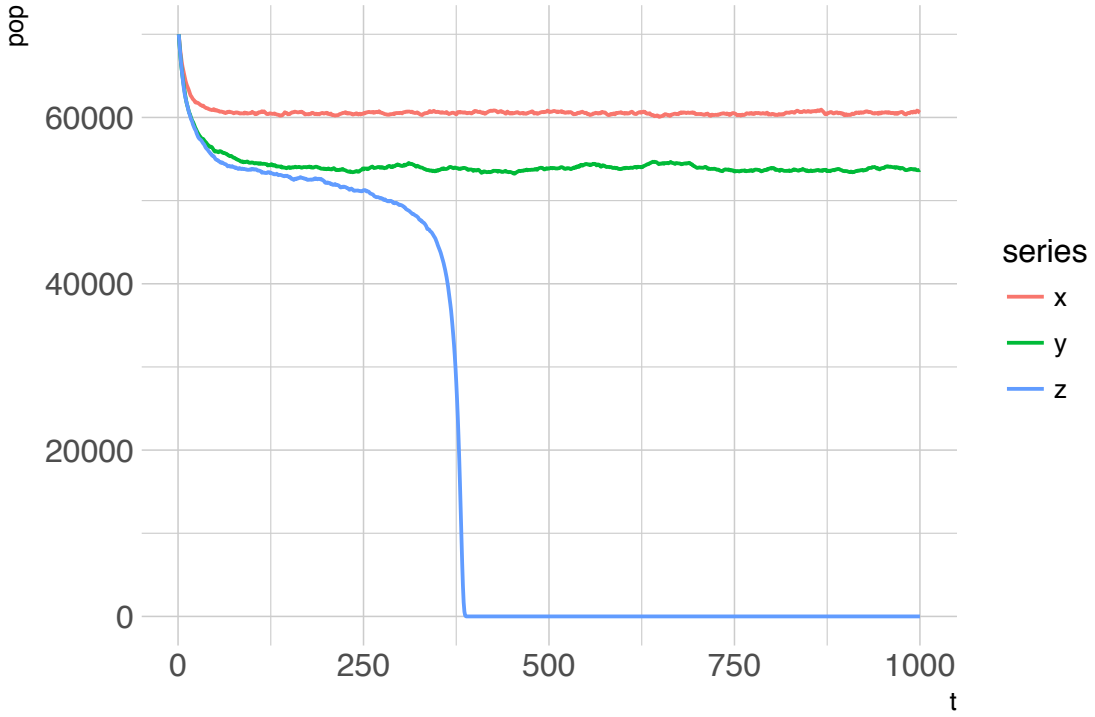
Configuration for Dai model

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

z <- y <- x <- numeric(max_days)
```

```
z[1] <- y[1] <- x[1] <- 7e4 # 1.7e5
set.seed(123)
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)

}

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

ggplot(df, aes(t, pop, col=series)) +
  geom_line()
```



We will also need a function for rolling autocorrelation for our warning sign calculation:

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

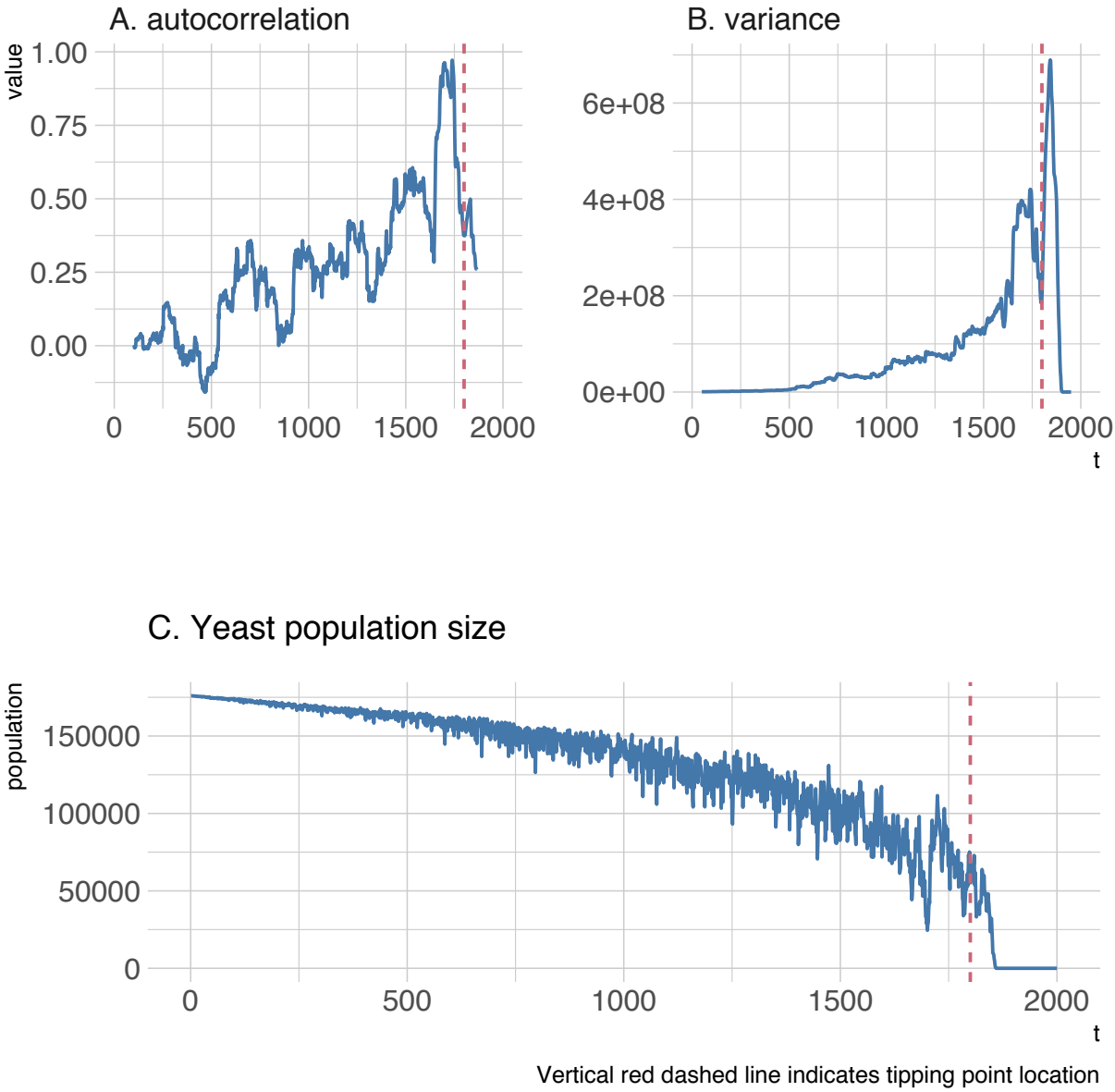
```
roll_acor <- function(x, lag = 1, ...){
  x_t1 = lag(x, n = lag)
  mu_t = roll_mean(x, ...)
  mu_t1 = roll_mean(x_t1, ...)
  s_t = roll_sd(x, ...)
  s_t1 = roll_sd(x_t1, ...)
  acor = roll_mean( (x - mu_t) * (x_t1 - mu_t1) / (s_t * s_t1),...)
  acor
}
```

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 Figre 4:

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

p1 <- tipping %>%
  select(-x) %>%
  rename("A. autocorrelation" = autocorrelation, "B. 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")
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()  +
  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)))
```



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.

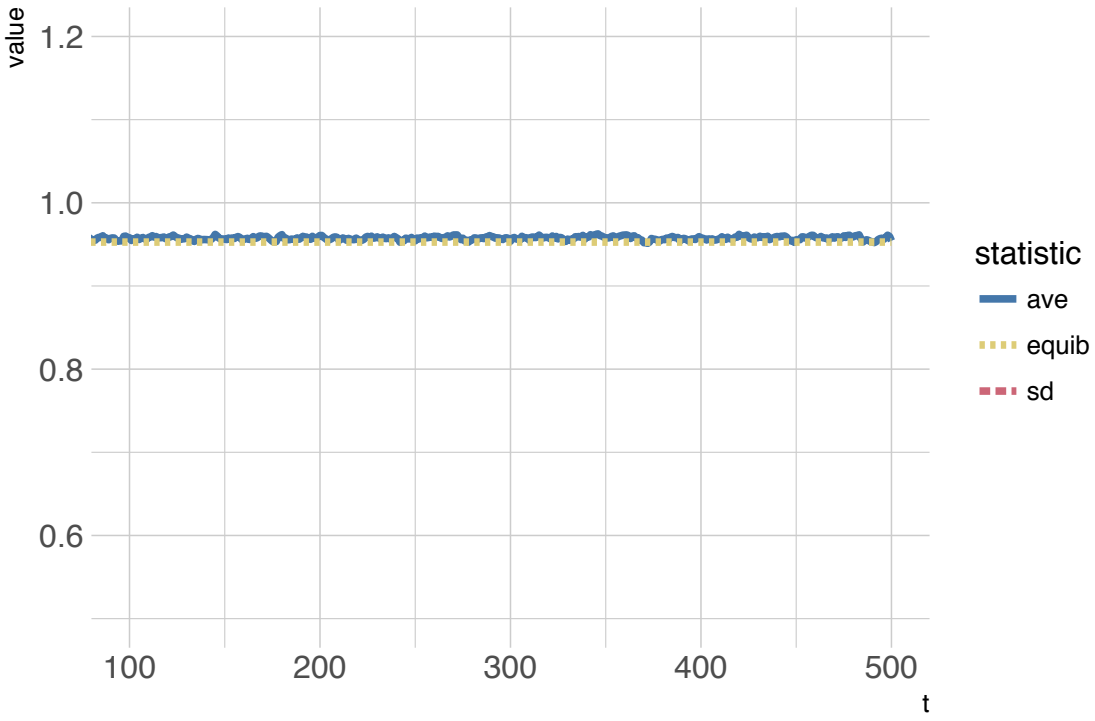
Analytical predicted population size, as a fraction of K, is $\frac{1}{2} + \frac{1}{2}\sqrt{1 - 8\sigma^2/K^2}$. Note this is independent of r, and increases with σ/K, the larger the noise σ_g as a fraction of the carrying capacity K. Population sizes can also be inflated relative to their deterministic 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.

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


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)
  }

})


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")
```

```
read_csv("inflation.csv") %>%
  gather(statistic, value, -t) %>%
  ggplot(aes(t,value, col=statistic, lty=statistic)) +
 # ggplot(aes(t,ave)) +
  geom_line(lwd=1) +
  scale_color_ptol() +
  coord_cartesian(xlim=c(100,500), ylim=c(1.2, .5))
```



References

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. doi:10.1126/science.1219805.