Boettiger (2020) Theoretical Ecology

T.J. Clark

2020-08-14

```r
library(tidyverse)
library(mdplearning) # Markov Decision Process
library(MDPtoolbox)
```

Reconstructing Boettiger's (2020) Theoretical Ecology paper

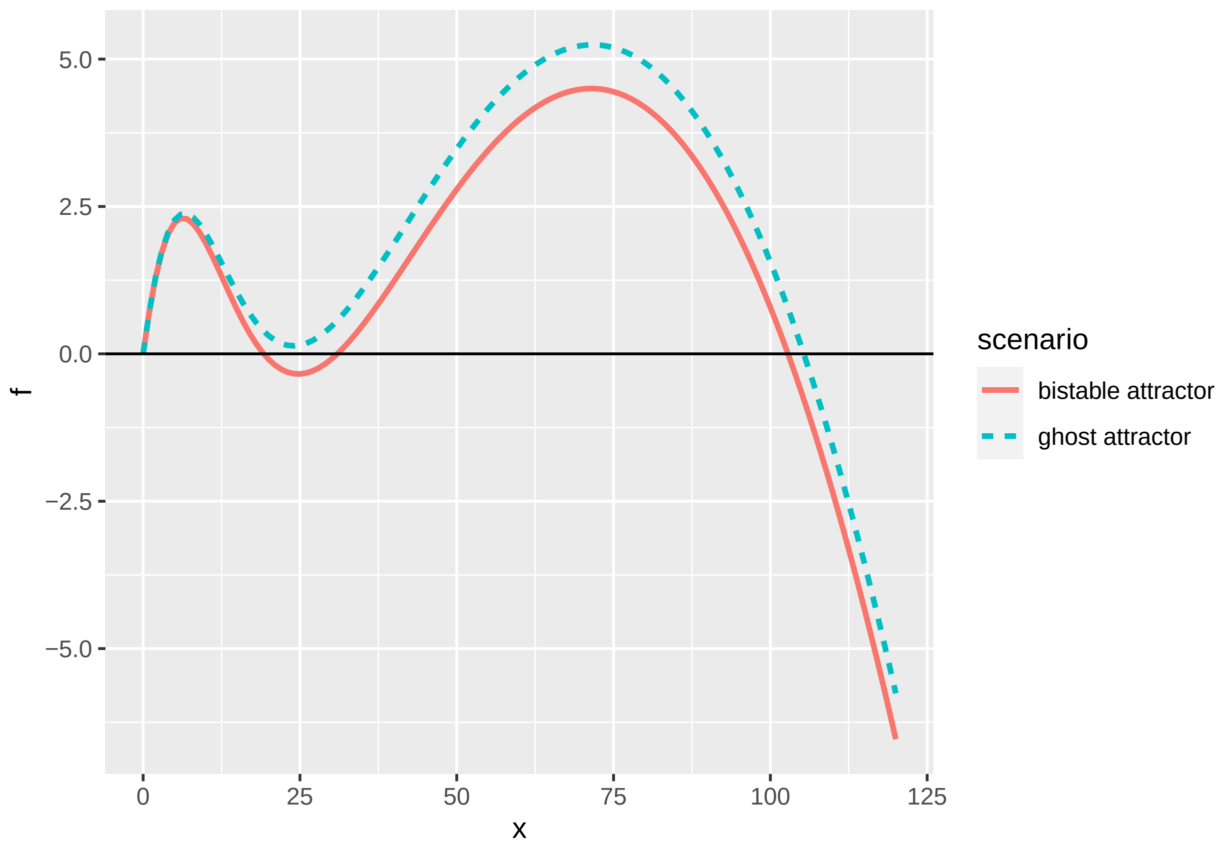
Idea is to optimately manage under different understanding of stable states…

```r
# discrete version of the state space (beetle) and the action space (culling)
n_s <- 121
states <- seq(0,120, length=n_s)
actions <- seq(0,120,length=n_s)
```

```r
# Model constants - also used to compute transition probabilities
efficiency <- 0.4
p <- list(r= 0.8, K = 153, q = 2, b = 20, sigma = 0.05, x0 = 20) # fixed parameters

may <- function(a){
  function(x, h=0){
    y <- x - efficiency*h
    pmax(
      y + y * p$r * (1 - y / p$K) - a * y ^ p$q / (y ^ p$q + p$b ^ p$q),
      0) # dont allow below 0
  }
}
```

```r
# Graph different models - allows manipulation by "a"
c("ghost attractor" = 27.2,
  "bistable attractor" = 28) %>%
  map_dfr(function(a) tibble(x = states, f = may(a)(x,0) - x, a = a), .id = "scenario") %>%
  ggplot(aes(x, f)) +
  geom_line(aes(lty = scenario, col = scenario), lwd = 1)+
  geom_hline(aes(yintercept = 0))
```



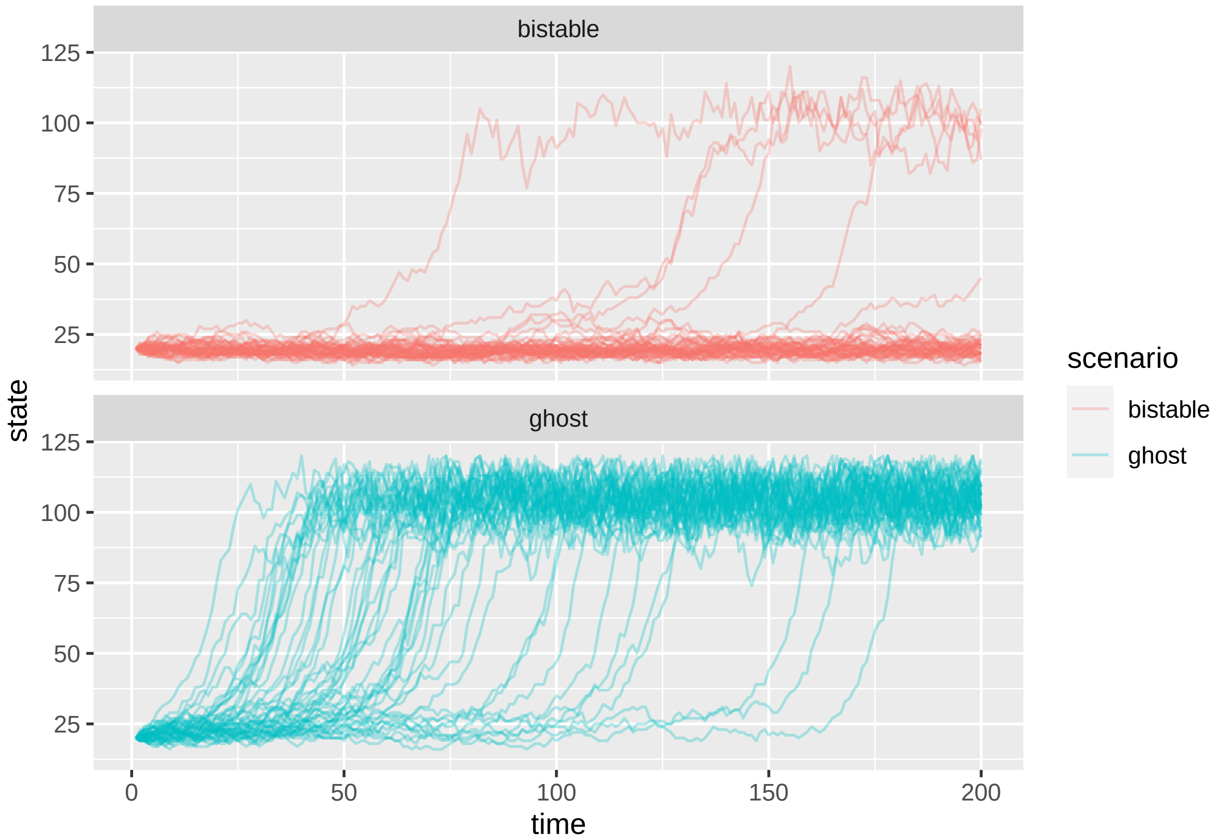

```
# transition matrix
# expresses probability that state Xt will transition to state Xt+1 with action At
transition_matrix <- function(states, actions, f, sigma){
  n_s <- length(states)
  n_a <- length(actions)
  transition <- array(0, dim = c(n_s, n_s, n_a))# grid of state space vals between 0 and 121, and 121 a
  for (i in 1:n_a){
    for (k in 1:n_s){
      nextpop <- f(states[k], actions[i])
      if (nextpop <= 0){
        x <- c(1, rep(0, n_s - 1))
      } else {
        x <- truncnorm::dtruncnorm(states, 0, max(states), nextpop, sigma*nextpop) # generates new x...
        if(sum(x) <= 0){
          x <- c(1, rep(0, n_s - 1))
        } else {
          x <- x/sum(x) # larger values are more likely!
        }
      }
      transition[k,,i] <- x
    }
  }
  if(any(is.na(transition))) stop("error creating transition matrix")
  transition
}
```

```r
# generate transition matrices
P_ghost <- transition_matrix(states, actions, may(27.2), p$sigma)
P_bistable <- transition_matrix(states, actions, may(28), p$sigma)
```

```r
# sample from transition matrix based off of matrix probabilities
sim <- function(transition, x0, Tmax, action = rep(1, Tmax)){
  n_states <- dim(transition)[2]
  state <- numeric(Tmax + 1)
  state[1] <- x0
  time <- 1:Tmax
  for (t in time){
    state[t + 1] <- base::sample(1:n_states,
                                 1,
                                 prob = transition[state[t], , action[t]])
  }
  data.frame(time = 1:Tmax, state = state[time])
}
```

```r
# try out simulate function via Stochastic dynamic Programming
x0 <- which.min(abs(states - p$x0)) # start point for utility function?
Tmax <- 200 # time of simulation
set.seed(12345)
reps <- 50
ghost_sim <- map_dfr(1:reps, function(i)
  sim(P_ghost, x0, Tmax) %>% mutate(state = states[state], scenario = "ghost"),
  .id = "rep")
bistable_sim <- map_dfr(1:reps, function(i)
  sim(P_bistable, x0, Tmax) %>% mutate(state = states[state], scenario = "bistable"),
  .id = "rep")
fig1_sims <- bind_rows(ghost_sim,bistable_sim)
```

```r
# show 50 reps for each of the simulations
fig1_sims %>%
  ggplot(aes(time, state, group = interaction(rep, scenario), col = scenario)) +
  geom_line(alpha = 0.3) + facet_wrap(~scenario, ncol=1)
```




```r
# utility functions

damage <- 0.5
control <- 1
endemic <- 50
discount <- 0.999 # dynamics on scale of days
reward_fn <- function(x,h) -(damage * pmax(x-endemic, 0))^2 - (control*h)

# apply reward function across states and actions
reward <- array(dim=c(length(states), length(actions)))
for(i in 1:length(states)){
  for(j in 1: length(actions)){
    reward[i,j] <- reward_fn(states[i], actions[j])
  }
}
```

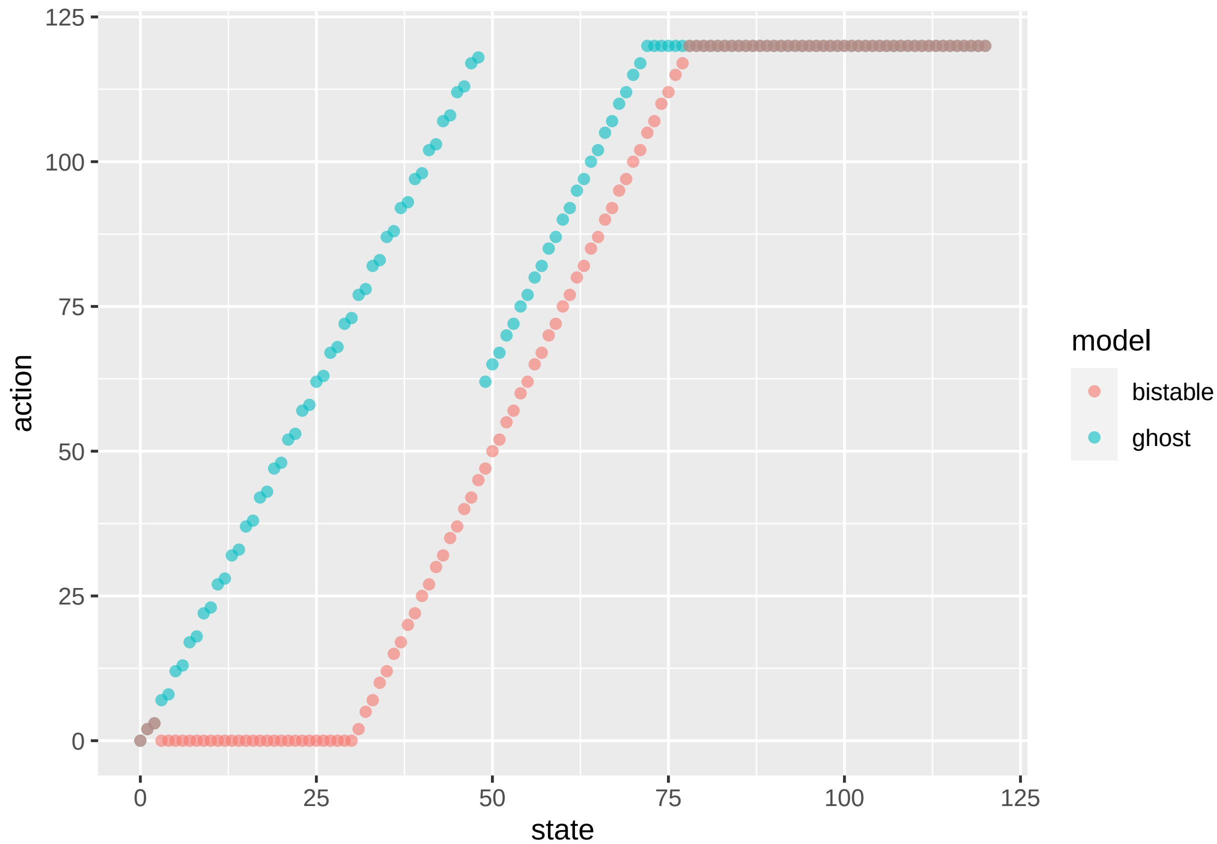
Optimal Planning under Dynamics. . . what would the manager do?

```r
# Uses Stochastic Dynamic Programming - see Marescot et al. (2013) sometime...
soln_ghost <- mdp_compute_policy(list(P_ghost), reward, discount, max_iter = 1e4, epsilon = 1e-2)
soln_bistable <- mdp_compute_policy(list(P_bistable), reward, discount, max_iter = 1e4, epsilon = 1e-4)

policy_plot <-
  tibble(state = states,
         ghost = actions[soln_ghost$policy], # optimal ghost policy
         bistable = actions[soln_bistable$policy]) %>% # optimal bistable policy
```

```r
  pivot_longer(c(ghost,bistable),
               names_to="model",
               values_to="action")
```

```r
# graph optimal policy
# NOTE: with more noise the management policies converge (since bistable can jump from 1 state to anoth
policy_plot %>%
  ggplot(aes(state, action, color = model)) +
  geom_point(alpha=0.6)
```

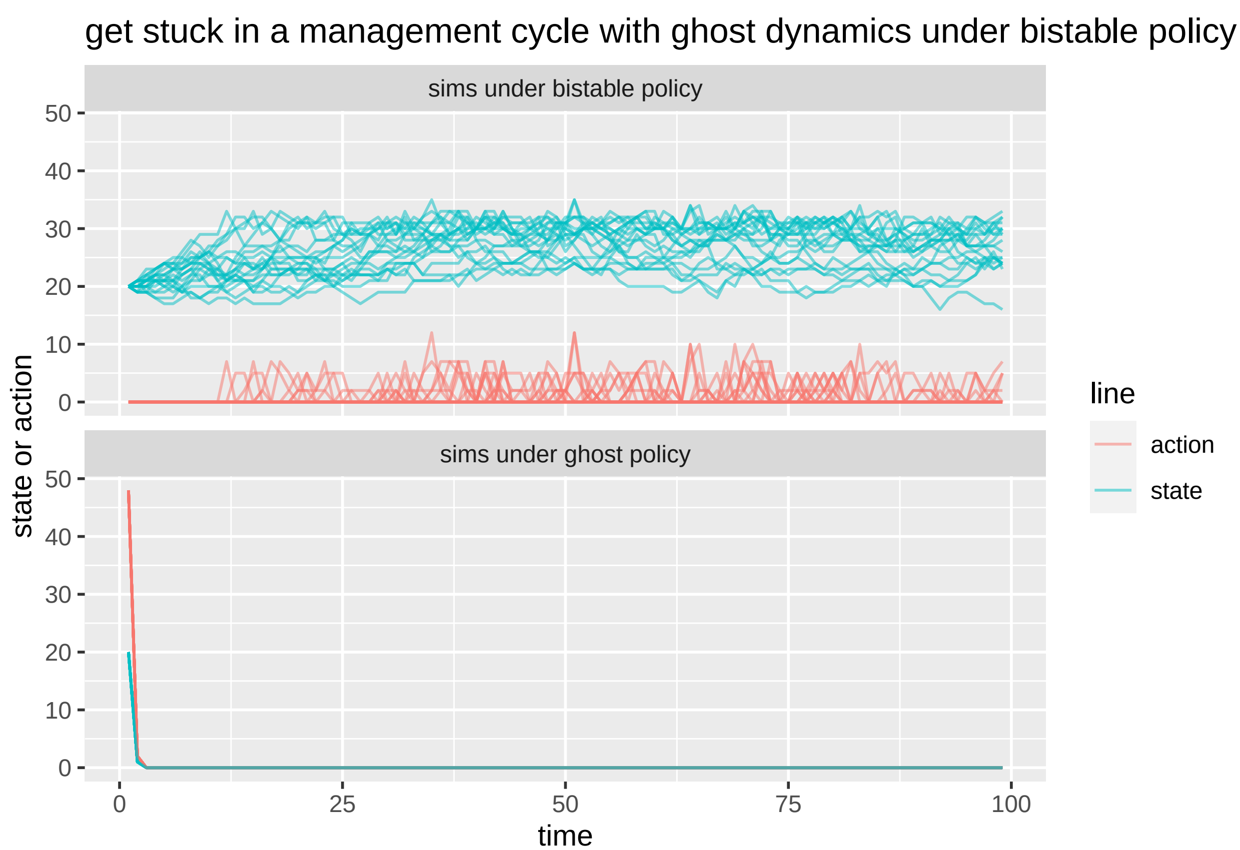


What happens if the you have ghost dynamics under each management policy?

```r
# run simulations under different management scenarios
reps <- 20
Tmax <- 200
set.seed(12345)
# ghost policy with ghost dynamics
sim_ghost <- map_dfr(1:reps, function(i)
                   mdp_planning(P_ghost, reward, discount,
                   policy = soln_ghost$policy,
                   x0 = x0, Tmax = Tmax), .id = "reps")
# bistable policy with ghost dynamics
sim_bistable <- map_dfr(1:reps, function(i)
                   mdp_planning(P_ghost, reward, discount,
                   policy = soln_bistable$policy,
```

```
                    x0 = x0, Tmax = Tmax), .id = "reps")
fixed_sims <-
 bind_rows("sims under ghost policy" = sim_ghost,
           "sims under bistable policy" = sim_bistable,
           .id = "prior") %>%
  mutate(state = states[state],
         action = actions[action])
```

```
# graph
fixed_sims %>% select(state, action, time, prior, reps) %>%
  pivot_longer(c(-time, -prior, -reps), names_to = "line", values_to  = "state") %>%
  filter(time < 100) %>%
  ggplot(aes(time, state, group = interaction(line, reps), col=line)) +
  geom_line(alpha=0.5) +
  facet_wrap(~prior, ncol=1) + ylab("state or action")+
  ggtitle("get stuck in a management cycle with ghost dynamics under bistable policy")
```

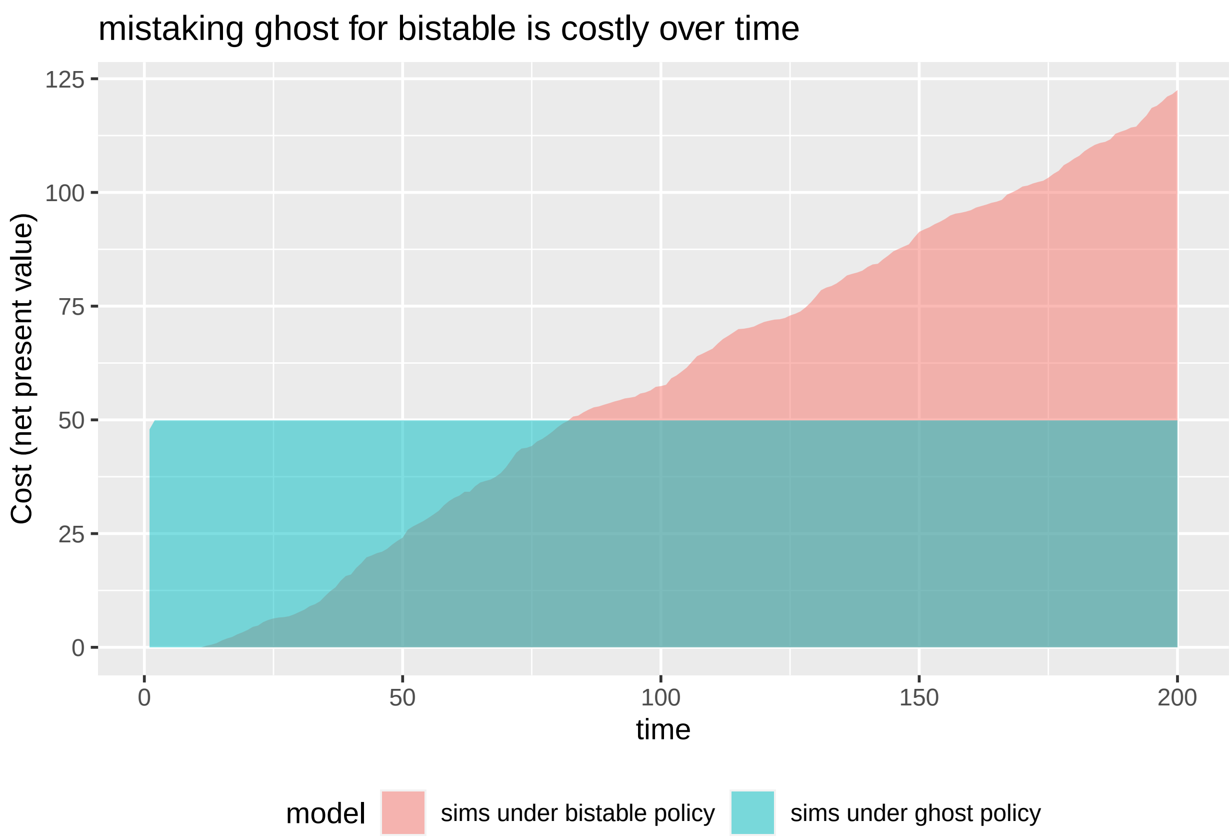


Average costs due to mistakes in policy with ghost dynamics

```
costs <- fixed_sims %>% group_by(time, prior) %>%
  summarize(cost = mean(value)) %>% # summarize across value gained
  group_by(prior) %>%
  mutate(cost = abs(cumsum(cost * .999 ^ time))) %>% # .99 is discounting by 81.86% of starting val at
  rename(model = prior)
```

```
## `summarise()` regrouping output by 'time' (override with `.groups` argument)
```

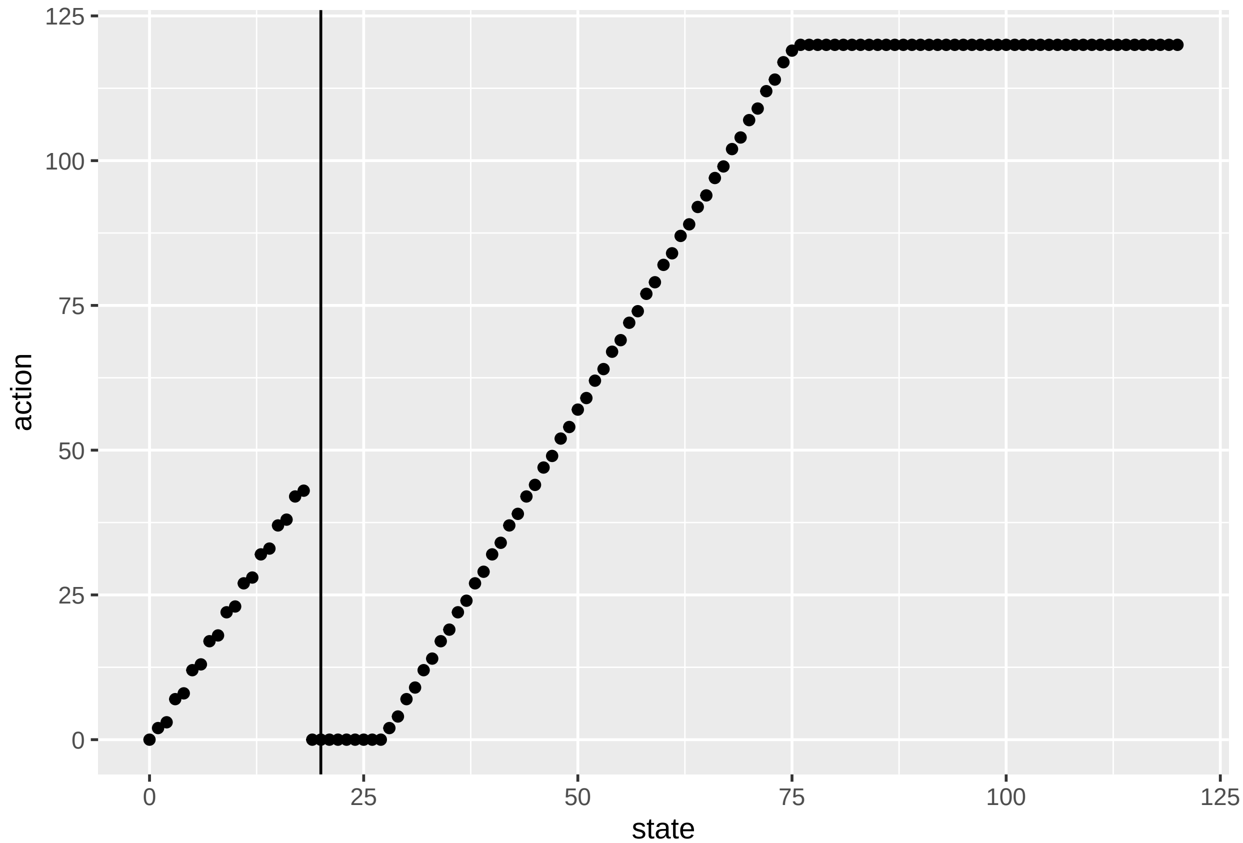
```r
# graph
costs %>%
  ggplot(aes(time, cost, ymin = 0, ymax = cost, fill = model)) +
  geom_ribbon(alpha = 0.5) +
  ylab("Cost (net present value)") +
  xlab("time")+
  theme(legend.position = "bottom") +
  ggtitle("mistaking ghost for bistable is costly over time")
```



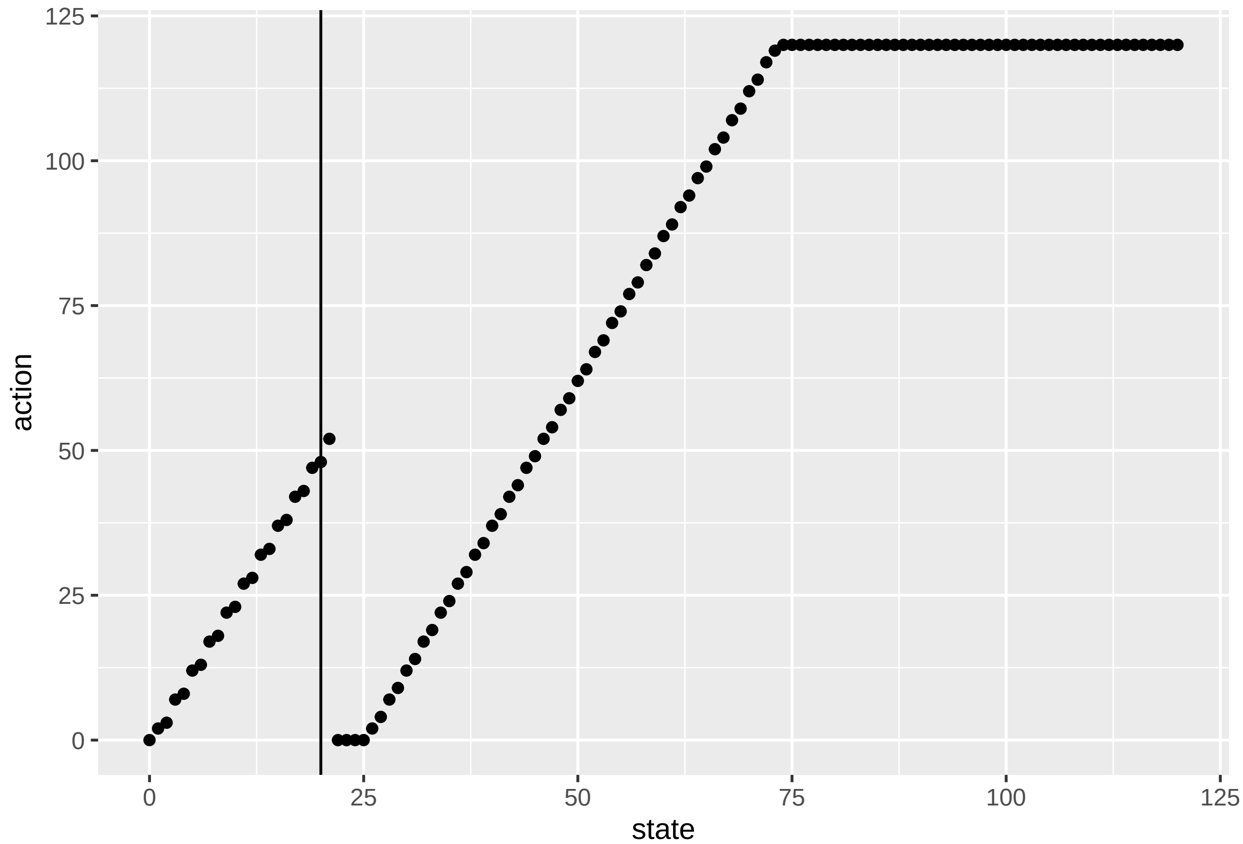
Planning for uncertainty - what if we don't know which model is correct?

```r
# calculate with different prior beliefs of models
fifty_fifty <- mdp_compute_policy(list(P_ghost, P_bistable), reward, discount,
                                  model_prior = c(0.5, 0.5),
                                  max_iter = 1e4, epsilon = 1e-2)
seventy_thirty <- mdp_compute_policy(list(P_ghost, P_bistable), reward, discount,
                                  model_prior = c(0.7, 0.3),
                                  max_iter = 1e4, epsilon = 1e-2)

# graph optimal policies for 50/50
tibble(state = states,
       action = actions[fifty_fifty$policy]) %>%
  ggplot(aes(state, action)) + geom_point() +
  geom_vline(aes(xintercept = states[x0]))
```




```
# graph optimal policies for 70/30 ghost/bistable
# basically, more likely that it's a ghost, more likely you'll be aggressive from the get go!
tibble(state = states,
       action = actions[seventy_thirty$policy]) %>%
  ggplot(aes(state, action)) + geom_point() +
  geom_vline(aes(xintercept = states[x0]))
```

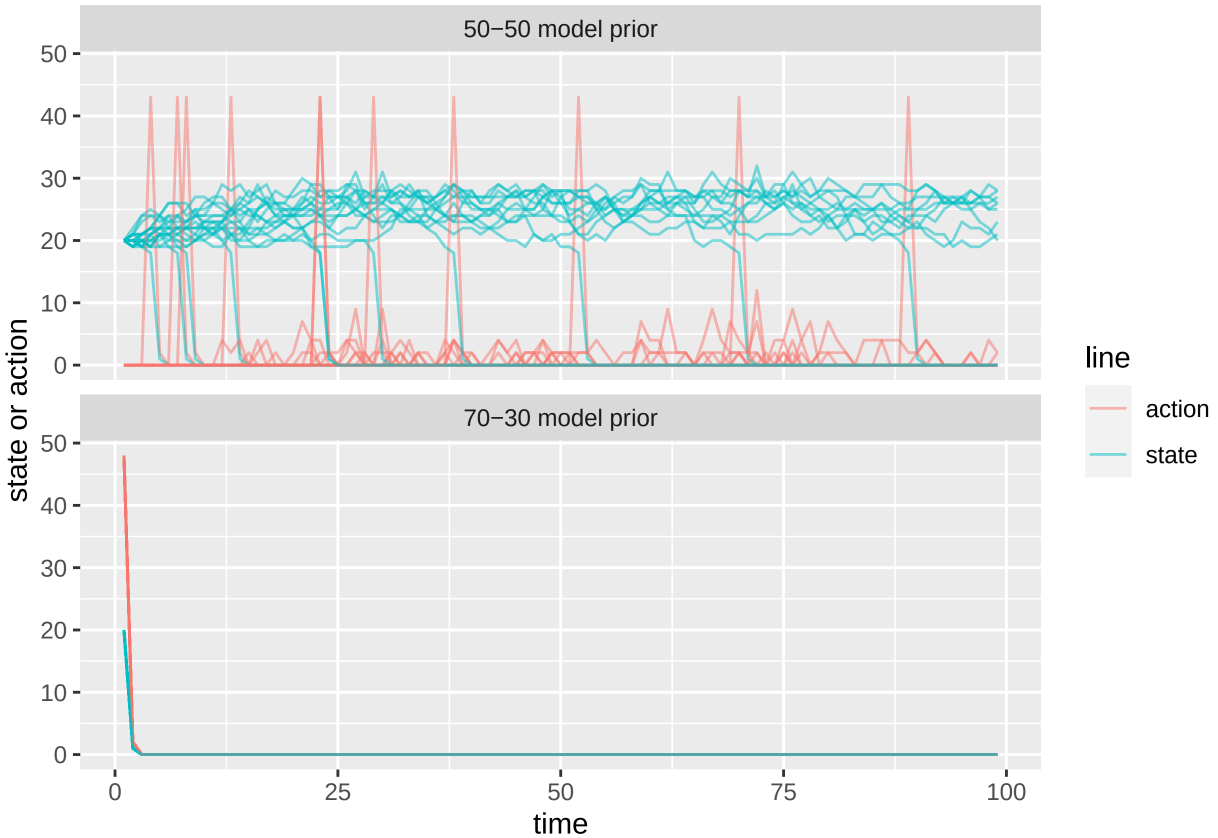



```r
# run simulations showing manager's uncertainty in the true model, with real dynamics being a ghost
sim_fiftyfifty <- map_dfr(1:reps, function(i)
  mdp_planning(P_ghost, reward, discount,
               policy = fifty_fifty$policy,
               x0 = x0, Tmax = Tmax), .id = "reps")
sim_seventythirty <- map_dfr(1:reps, function(i)
  mdp_planning(P_ghost, reward, discount,
               policy = seventy_thirty$policy,
               x0 = x0, Tmax = Tmax), .id = "reps")

sims_by_prior <-
  bind_rows("50-50 model prior" = sim_fiftyfifty,
            "70-30 model prior" = sim_seventythirty,
            .id = "prior") %>%
  mutate(state = states[state],
         action = actions[action])

# graph
# 50-50 prior = some resemble large actions to eliminate pest, but ususally just look like bistability f
# 70-30 prior = super large action right away, eliminates pest altogether
sims_by_prior %>% select(state, action, time, prior, reps) %>%
  pivot_longer(c(-time, -prior, - reps), names_to = "line", values_to = "state") %>%
  filter(time < 100) %>%
  ggplot(aes(time, state, group = interaction(line, reps), col = line)) +
  geom_line(alpha = 0.5) +
  facet_wrap(~prior, ncol=1)+
```

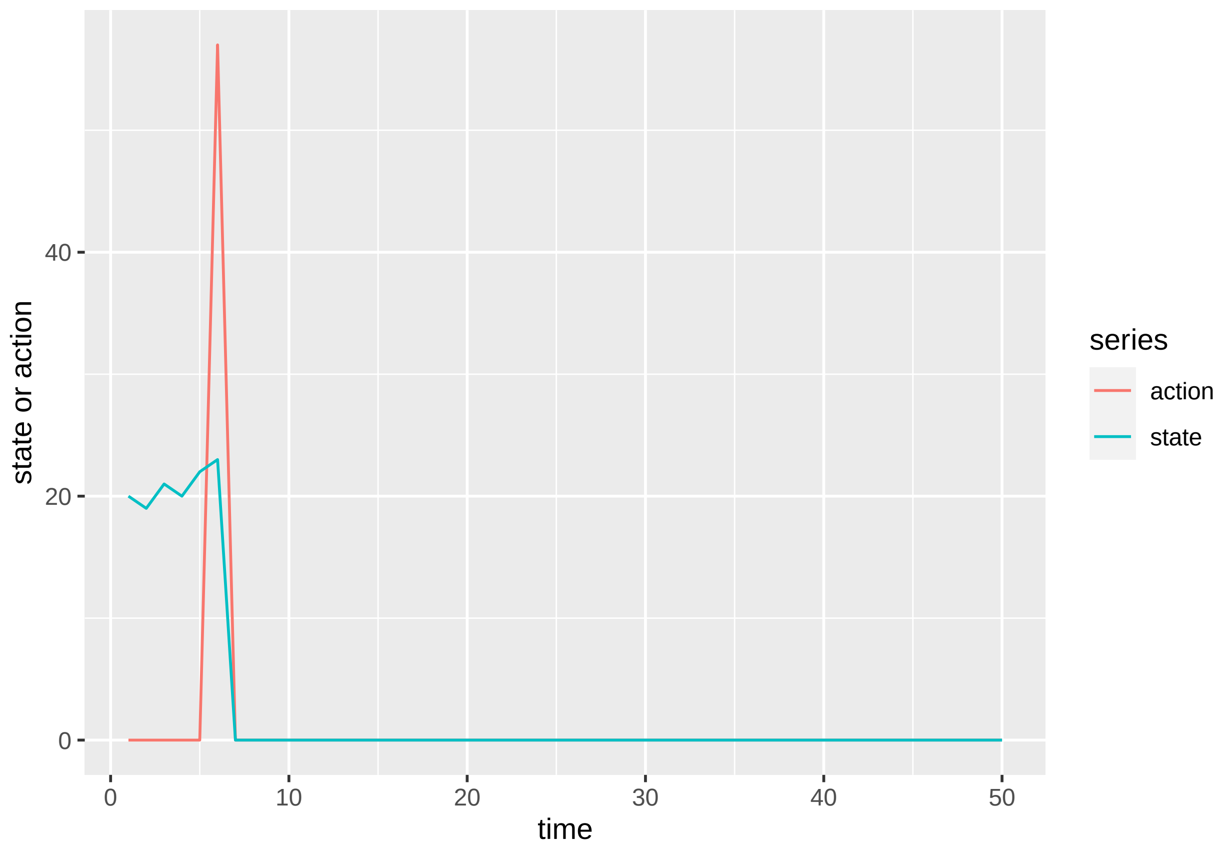
```r
  ylab("state or action")
```



What if the manager can learn using Adaptive Management? Because each time step gives the manager more information about which dynamics are occurring...

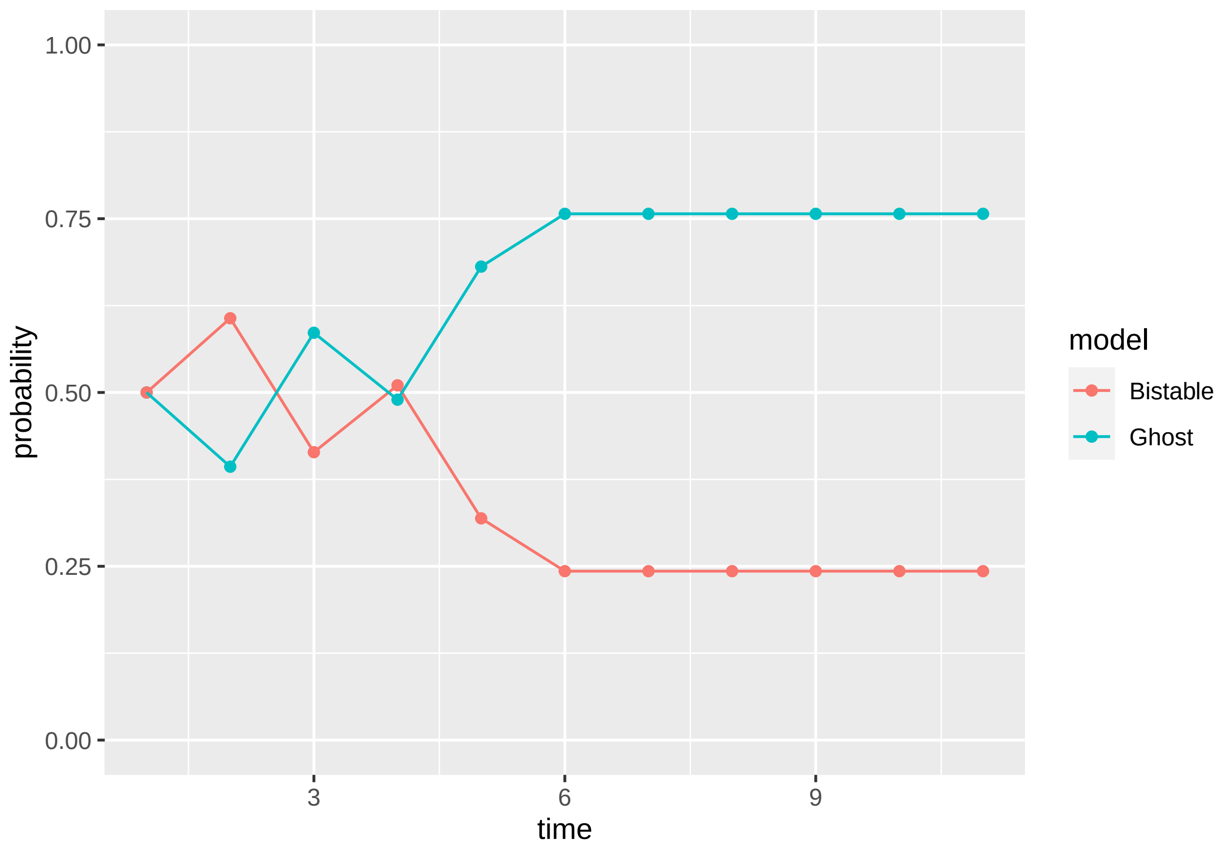
```r
# adaptive management, 50/50 prior belief, actual ghost dynamics
set.seed(12345)
Tmax <- 50
learning <- mdp_learning(list(P_ghost, P_bistable), reward, discount,
                         model_prior = c(0.5, 0.5),
                         x0=x0, Tmax=Tmax,
                         true_transition = P_ghost,
                         max_iter = 1e4, epsilon = 1e-2)

# graph
# short period of low pest management, followed by a large action
learning$df %>%
  select(-value, -obs) %>%
  gather(series, state, -time) %>%
  ggplot(aes(time, states[state], color = series)) +
  geom_line()+
  ylab("state or action")
```



Learning through Adaptive Management… When do we know the dynamics of the system? Posterior shows belief state of the system

```
# increasing prob of ghost being correct, decreasing prob of bistable being incorrect
model_names = c("Ghost", "Bistable")
learning$posterior %>%
  data.frame(time = 1:Tmax) %>%
  filter(time < 12) %>% # learns very quickly
  gather(model, probability, -time, factor_key = T) %>%
  mutate(model = model_names[as.integer(model)]) %>%
  ggplot(aes(x=time, y=probability, col = model)) +
  geom_point() +
  geom_line() +
  ylim(c(0,1))
```


1.00
0.75
0.50
0.25
0.00
probability
3
6
9
time
model
Bistable
Ghost