Appendix for: Deep Reinforcement Learning for Conservation Decisions

A Preprint

Marcus Lapeyrolerie
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Melissa Chapman
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Kari Norman
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California

Carl Boettiger
Department of Environmental Science, Policy, and Management
University of California, Berkeley
Berkeley, California
cboettig@berkeley.edu

April 7, 2021

Abstract

Enter the text of your abstract here.

***K*eywords** blah · blee · bloo · these are optional and can be removed

```
library(tidyverse)
library(patchwork)
library(reticulate)

np <- import("numpy")
np$random$seed(42L)

# force CPU-evaluation if needing perfect reproducibility
Sys.setenv("CUDA_VISIBLE_DEVICES"="0")
```

1 Finding a known optimal solution using RL

1.1 Sustainable Harvest Quotas

```
## Python dependencies
gym         <- import ("gym")
gym_fishing <- import("gym_fishing")
sb3         <- import ("stable_baselines3")

## initialize the environment
env <- gym$make("fishing-v1", sigma = 0.1)
```

```
# train an agent (model) on one of the environments:


#Trial 15 finished with value: 7.6994757652282715 and parameters:
hyper = list('gamma'= 0.995, 'lr'= 0.0001355522450968401, 'batch_size'= 128L,
         'buffer_size'= 10000L, 'episodic'= FALSE, 'train_freq'= 128L,
         'noise_type'= 'normal', 'noise_std'= 0.6656948079225263,
         'net_arch'= 'big')
policy_kwargs = list(net_arch=c(400L, 300L)) # big


#non-episodic:
hyper['gradient_steps'] = hyper['train_freq']
hyper['n_episodes_rollout'] = -1
n_actions = env$action_space$shape[0]
hyper$action_noise = sb3$common$noise$NormalActionNoise(
        mean=np$zeros(n_actions),
        sigma= hyper[['noise_std']] * np$ones(n_actions)
        )


td3 = sb3$TD3('MlpPolicy', env,  verbose=0L, seed = 42L,
            gamma = hyper[['gamma']],
            learning_rate = hyper[['lr']],
            batch_size = hyper[['batch_size']],
            buffer_size = hyper[['buffer_size']],
            action_noise = hyper[['action_noise']],
            train_freq = hyper[['train_freq']],
            gradient_steps = hyper[['train_freq']],
            n_episodes_rollout = hyper[['n_episodes_rollout']],
            policy_kwargs=policy_kwargs)


td3$learn(total_timesteps=300000L)
td3$save("cache/td3")
```

```
# Simulate management under the trained agent
td3 <- sb3$TD3$load("cache/td3")
td3_sims <- env$simulate(td3, reps = 500L)
td3_policy <- env$policyfn(td3, reps = 50L)
```

```
# train an agent (model) on one of the environments:
a2c <- sb3$A2C('MlpPolicy', env, verbose=0L, seed = 42L) # Must use L for integers!
a2c$learn(total_timesteps=300000L)
a2c$save("cache/a2c")
```

```
# Simulate management under the trained agent
a2c <- sb3$A2C$load("cache/a2c")
```

```
a2c_sims <- env$simulate(a2c, reps = 500L)
a2c_policy <- env$policyfn(a2c, reps = 50L)
```

```
# Simulate under the optimal solution (given the model)
opt <- gym_fishing$models$escapement(env)
opt_sims <- env$simulate(opt, reps = 500L)
opt_policy <- env$policyfn(opt)
```

```
sims_df <- bind_rows(td3_sims, a2c_sims, opt_sims, .id = "model") %>%
  mutate(model = c("TD3", "A2C", "optimal")[as.integer(model)])

policy_df <- bind_rows(td3_policy, a2c_policy, opt_policy, .id = "model") %>%
  mutate(model = c("TD3", "A2C", "optimal")[as.integer(model)])

gamma <- 1 #discount
reward_df <- sims_df %>%
  group_by(rep, model) %>%
  mutate(cum_reward = cumsum(reward * gamma^time)) %>%
  group_by(time, model) %>%
  summarise(mean_reward = mean(cum_reward),
            sd = sd(cum_reward), .groups = "drop")
```

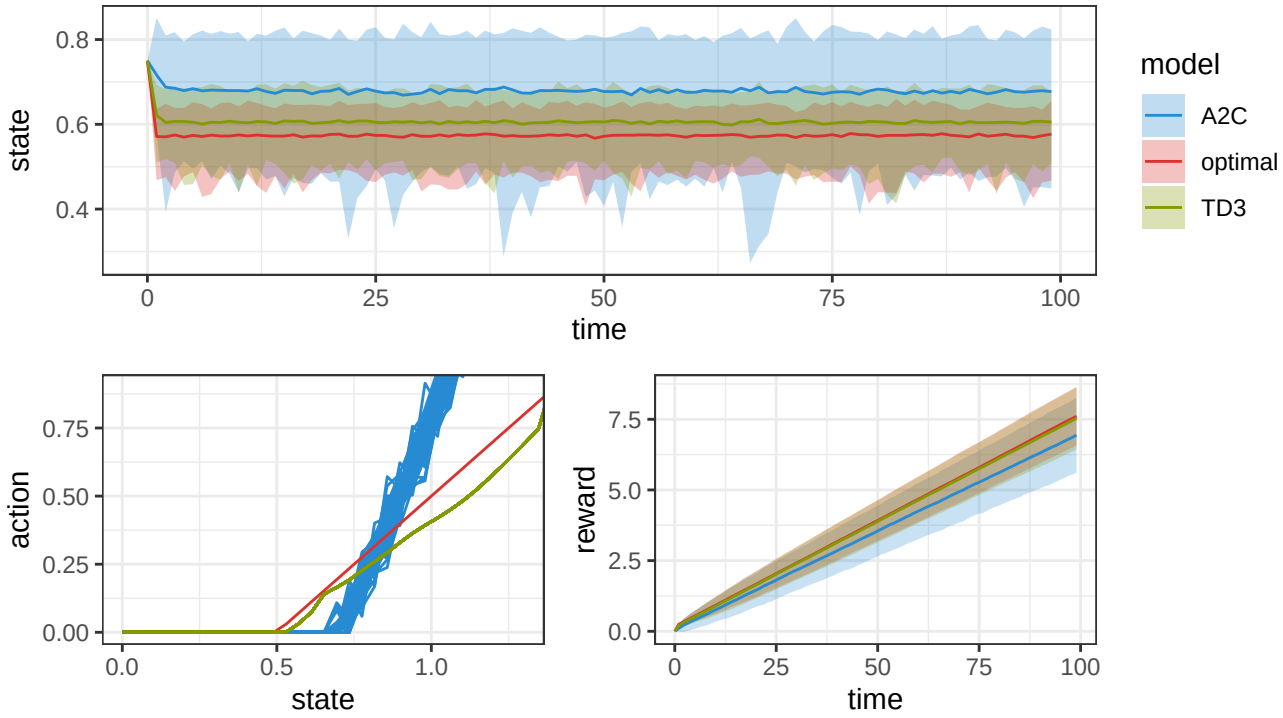
```
write_csv(sims_df, "figs/sims_df.csv")
write_csv(policy_df, "figs/policy_df.csv")
write_csv(reward_df, "figs/reward_df.csv")
```

```
ymin <- function(x) last(x[(ntile(x, 20)==1)])
ymax <- function(x) last(x[(ntile(x, 20)==19)])

fig_sims <-
sims_df %>%
  group_by(time, model) %>%
  summarise(ymin = ymin(state),
            ymax = ymax(state),
            state = mean(state), .groups = "drop") %>%
  ggplot(aes(time, state, ymin = ymin, ymax = ymax, fill=model)) +
  geom_ribbon(alpha= 0.3) + geom_line(aes(col = model))
```

```
fig_policy <-
  policy_df %>% ggplot(aes(state, action,
                           group=interaction(rep, model),
                           col = model)) +
  geom_line(show.legend = FALSE) +
  coord_cartesian(xlim = c(0, 1.3), ylim=c(0,0.9))
```

```
fig_reward <- reward_df %>%
  ggplot(aes(time, mean_reward)) +
  geom_ribbon(aes(ymin = mean_reward - 2*sd,
                  ymax = mean_reward + 2*sd, fill = model),
              alpha=0.25, show.legend = FALSE) +
  geom_line(aes(col = model), show.legend = FALSE) +
  ylab("reward")
```

```
fig_sims / ( fig_policy + fig_reward)
```



1.2 Ecological tipping points

```
gym_conservation <- import("gym_conservation")
env <- gym$make("conservation-v6")
```

```
# Simulate management under the trained agent
TD3 <- sb3$TD3$load("cache/td3-conservation")
TD3_sims <- env$simulate(TD3, reps = 100L)
TD3_policy <- env$policyfn(TD3, reps = 10L)
```

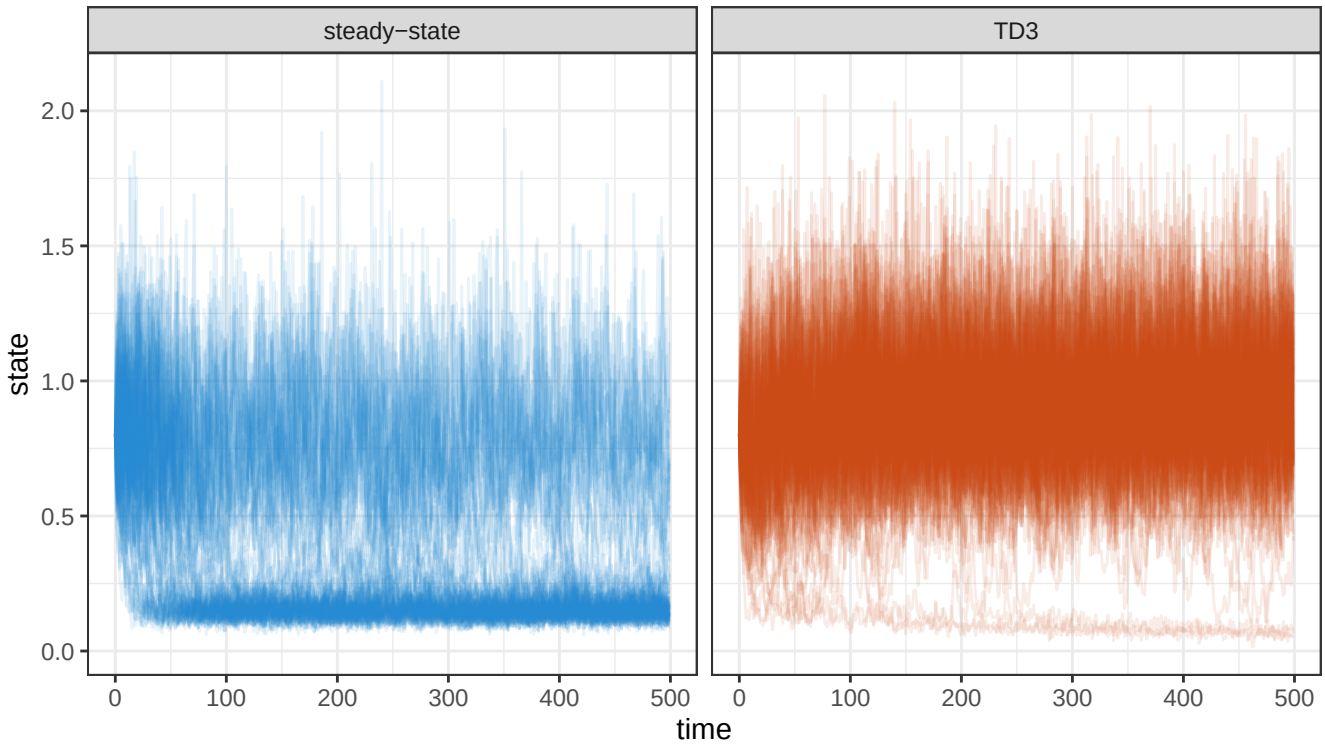
```
# Simulate under the steady-state solution (given the model)
K = 1.5
alpha = 0.001
opt <- gym_conservation$models$fixed_action(env, fixed_action = alpha * 100 * 2 * K )
opt_sims <- env$simulate(opt, reps = 100L)
opt_policy <- env$policyfn(opt)
```

```
sims_df <- bind_rows(TD3_sims, opt_sims, .id = "model") %>%
  mutate(model = c("TD3", "steady-state")[as.integer(model)])

policy_df <- bind_rows(TD3_policy, opt_policy, .id = "model") %>%
  mutate(model = c("TD3", "steady-state")[as.integer(model)])

gamma <- 1 #discount
reward_df <- sims_df %>%
  group_by(rep, model) %>%
  mutate(cum_reward = cumsum(reward * gamma^time)) %>%
  group_by(time, model) %>%
  summarise(mean_reward = mean(cum_reward),
            sd = sd(cum_reward), .groups = "drop")
```

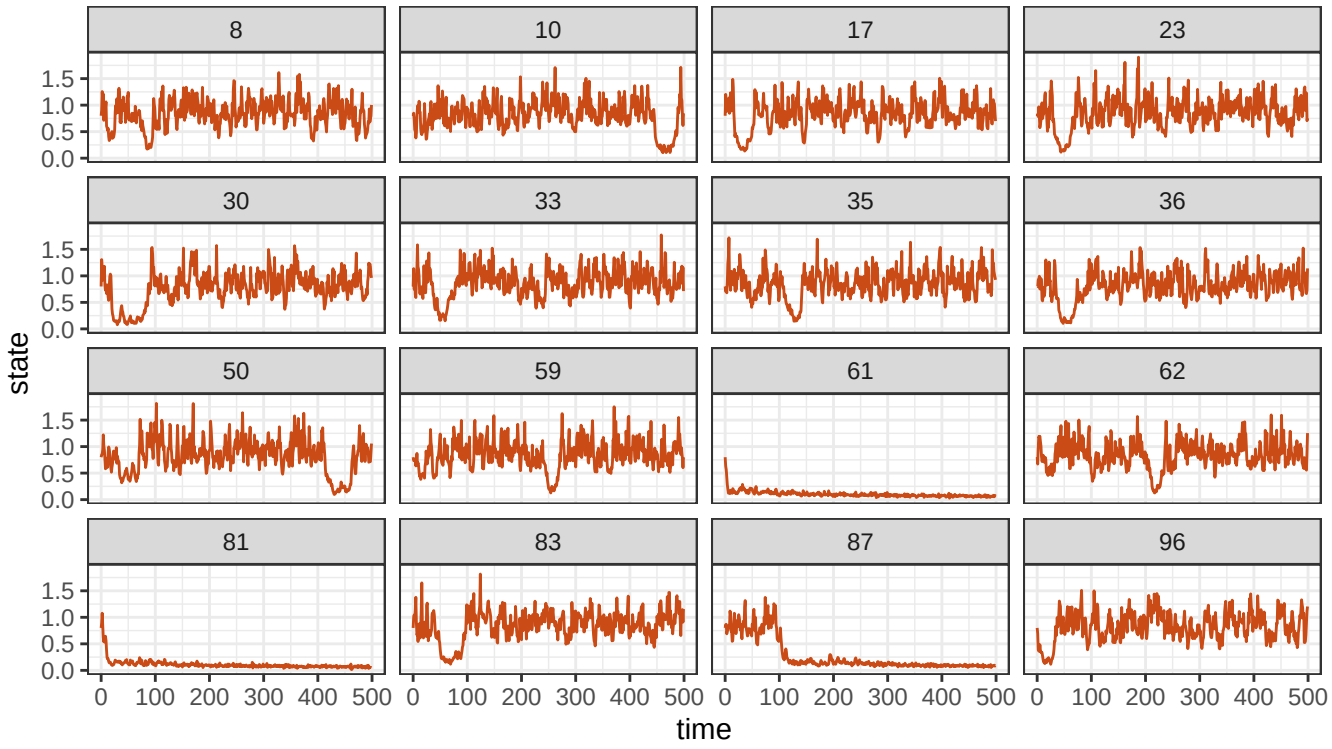
```
write_csv(sims_df, "figs/tipping_sims_df.csv")
write_csv(policy_df, "figs/tipping_policy_df.csv")
write_csv(reward_df, "figs/tipping_reward_df.csv")
```

```
# ensemble statistics
ymin <- function(x) last(x[(ntile(x, 20)==1)])
ymax <- function(x) last(x[(ntile(x, 20)==19)])

fig_sims <-
  sims_df %>%
   group_by(time, model) %>%
   summarise(ymin = ymin(state),
             ymax = ymax(state),
             state = mean(state), .groups = "drop") %>%
  ggplot(aes(time, state, ymin = ymin, ymax = ymax, fill=model)) +
   geom_ribbon(alpha= 0.3) + geom_line(aes(col = model))
```

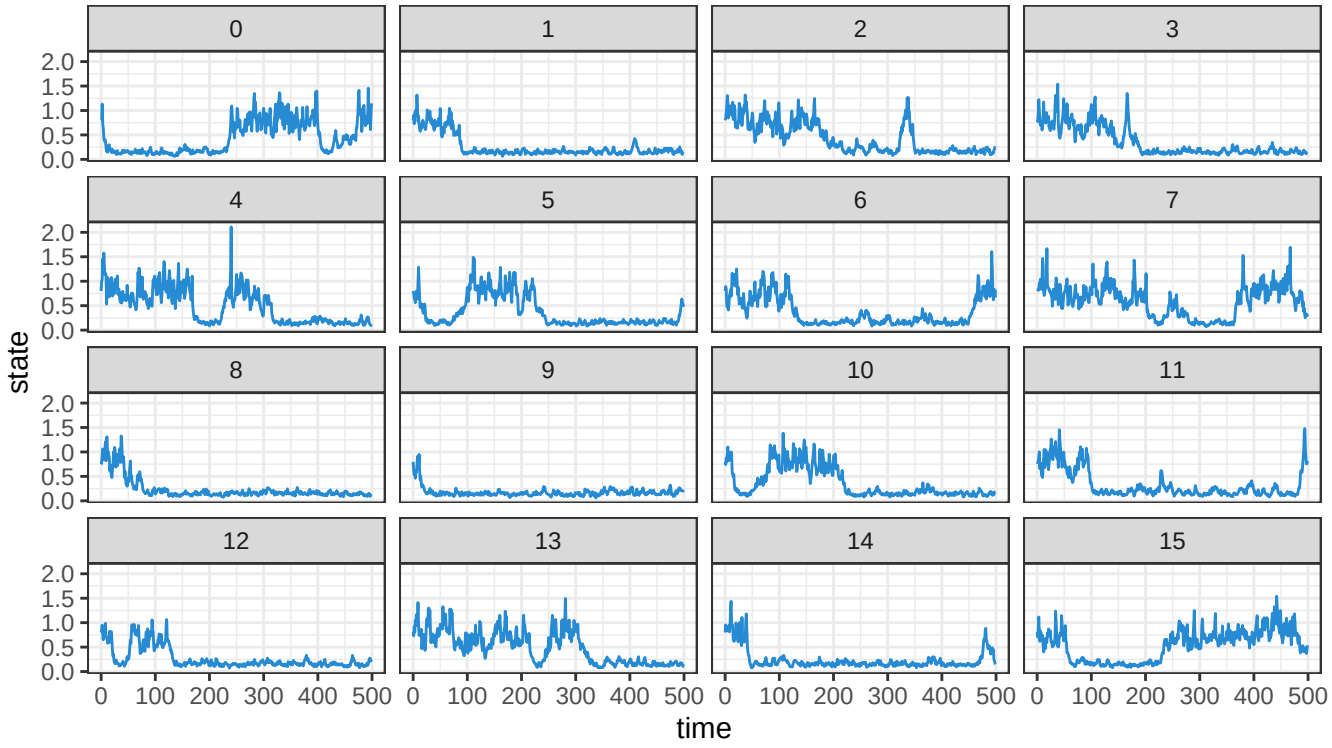
```
# Some individual replicates, for comparison
sims_df %>%
  filter(rep < 100, time < 500) %>%
  ggplot(aes(time, state, col=model, group=interaction(model, rep))) +
  geom_line(alpha = .1, show.legend = FALSE) + facet_wrap(~model)
```




```
# Some individual replicates, for comparison
is_low <- sims_df %>% filter(model == "TD3") %>% group_by(rep, model) %>% summarize(low = sum(state < .1
sims_df %>% inner_join(is_low) %>%
  ggplot(aes(time, state,  group=interaction(model, rep))) +
  geom_line(color = pal[3], show.legend = FALSE) + facet_wrap(~rep)
```




```
# Some individual replicates, for comparison
is_low <- sims_df %>% filter(model == "steady-state") %>% group_by(rep, model) %>% summarize(low = sum(s
sims_df %>% inner_join(is_low) %>%
  ggplot(aes(time, state, group=interaction(model, rep))) +
  geom_line(color = pal[1], show.legend = FALSE) + facet_wrap(~rep)
```



```
fig_policy <-
  policy_df %>% ggplot(aes(state, action,
                           group=interaction(rep, model),
                           col = model)) +
  geom_line(show.legend = FALSE)
```

```
fig_reward <- reward_df %>%
  ggplot(aes(time, mean_reward)) +
  geom_ribbon(aes(ymin = mean_reward - 2*sd,
                  ymax = mean_reward + 2*sd, fill = model),
              alpha=0.25, show.legend = FALSE) +
  geom_line(aes(col = model), show.legend = FALSE) +
  ylab("reward")
```

```
fig_sims / ( fig_policy + fig_reward)
```