Appendix for: "Bad Forecast, Good Decision"

Carl Boettiger

2020-11-19

Here we provide annotated code necessary to completely reproduce all of the analysis presented in the main paper. This analysis is run in R (R Core Team 2019) uses `MDPtoolbox` (Chades et al. 2017) for solving Markov Decision Processes (MDP) using stochastic dynamic programming functionality, `expm` for matrix exponentials (Goulet et al. 2020), and a few custom MDP functions provided by our package, `mdplearning` (Carl Boettiger 2018). We will also use `tidyverse` packages for basic manipulation and plotting (Wickham et al. 2019). This file is also available as an RMarkdown document (Xie, Allaire, and Grolemund 2018) at https://github.com/cboettig/bad-forecast-good-decision.

```
library(tidyverse)
library(MDPtoolbox)
library(expm)
# remotes::install_github("boettiger-lab/mdplearning")
library(mdplearning)
```

Our decision problem is defined as follows: A manager seeks to maximize the sum of the utility derived from such a harvest and such a state, $U(X_t, H_t)$, over all time, subject to discount rate δ:

$$\sum_{t=0}^{t=\infty} U(X_t, H_t)\delta^t \tag{1}$$

While in principle the utility could reflect many things, including the cost of fishing, market responses to supply and demand, the value of recreational fishing, the intrinsic value fish left in the sea (see Halpern et al. 2013), for simplicity we will assume utility is merely a linear function of the harvest quota set by the manager, i.e. a fixed price p per kilogram of fish harvested: $U(X_t, H_t) = p \min(H_t, X_t)$ (noting that realized harvest cannot exceed the available stock). As units are considered already non-dimensionalized in this example, without loss of generality we will set $p = 1$. This problem is already well studied, and it is worth noting that even under such a pessimistic assumption, the optimal strategy still seeks to sustain the fish population indefinitely; as Clark (1973) shows for the deterministic function f and Reed (1979) extended to the stochastic case. Given the function f with known parameters, it is straight forward to determine the optimal harvest policy by stochastic dynamic programming (SDP, see Mangel (1985); Marescot et al. (2013)).

To solve the decision problem using SDP, we define the state space and the action space on a discrete grid of 240 points. The maximum state is set well above the largest carrying capacity used in the models, which limits the influence of boundary effects introduced by the transformation to a discrete, finite grid. Available harvest actions match the state space, effectively allowing any harvest level to be possible.

```
states <- seq(0, 24, length.out = 240)
actions <- states
obs <- states
```

The utility (reward) of an action is set to the (realized) harvest (e.g. a fixed price per unit fish, with no cost applied to harvest effort). Future utility is discounted by fixed factor of 0.99. Classic maximum sustainable yield models (Schaefer 1954) ignore discounting, while modern economic optimization models insist on it, so a

small discount conforms to the latter while reasonably approximating the former. The qualitative conclusion is not sensitive to the discounting rate.

```
reward_fn <- function(x, h) pmin(x, h)
discount <- 0.99
```

Alternate reward functions with varying price and cost structures are possible but do not qualitatively impact the conclusions. This reward function is a limiting case of any more complex reward, and corresponds both to classic work that does not model utility explicitly (e.g. MSY theory, Schaefer 1954), as well as explicit assumptions typically made in more recent models (Reed 1979). Moreover, using a simple reward function makes it clear that model that performs best is not doing so merely because of particular features baked into the a carefully chosen reward rule.

Ecological Models

The manager chooses between two different logistic growth models, each of which can be thought of as an approximation to the underlying "true" population model:

$$f_i(Y) = Y + Y r_i \left(1 - \frac{Y}{K_i}\right) \tag{2}$$

Model 1 has $r_1 = 2$, $K = 16$, and $\sigma_1 = 0.05$. Model 2 has $r_2 = 0.5, K_2 = 10$, and $\sigma_2 = 0.075$.

Meanwhile, the true population growth rate is simulated using a function with non-linear per-capita growth:

$$f_i(Y) = Y + Y^4 r_i \left(1 - \frac{Y}{K_i}\right) \tag{3}$$

with $r_3 = 0.002$, $K_3 = 10$ and $\sigma_3 = 0.05$:

```
# K is at twice max of f3; 8 * K_3 / 5
f1 <- function(x, h = 0, r = 2, K = 10 * 8 / 5) {
  s <- pmax(x - h, 0)
  s + s * (r * (1 - s / K))
}
f2 <- function(x, h = 0, r = 0.5, K = 10) {
  s <- pmax(x - h, 0)
  s + s * (r * (1 - s / K))
}

# max is at 4 * K / 5
f3 <- function(x, h = 0, r = .002, K = 10) {
  s <- pmax(x - h, 0)
  s + s^4 * r * (1 - s / K)
}

## gather models together, indicate true model
sigma_g <- 0.05
models <- list("1" = f1, "2" = f2, "3" = f3)
model_sigmas <- c(sigma_g, 1.5 * sigma_g, sigma_g)
true_model <- "3"
```

On a discrete grid of possible states and actions, we can define the growth rate of a given state X_t subject to harvest H_t, $f(X_t, H_t)$ as set of matrices. Each matrix i gives the transition probabilities for any current state to any future state, given that action i is taken.

```
transition_matrices <- function(f, states, actions, sigma_g) {
  n_s <- length(states)
  n_a <- length(actions)
  transition <- array(0, dim = c(n_s, n_s, n_a))
  for (k in 1:n_s) {
    for (i in 1:n_a) {
      nextpop <- f(states[k], actions[i])
      if (nextpop <= 0) {
        transition[k, , i] <- c(1, rep(0, n_s - 1))
      } else if (sigma_g > 0) {
        x <- dlnorm(states, log(nextpop), sdlog = sigma_g)
        if (sum(x) == 0) { ## nextpop is computationally zero
          transition[k, , i] <- c(1, rep(0, n_s - 1))
        } else {
          x <- x / sum(x) # normalize evenly
          transition[k, , i] <- x
        }
      }
    }
  }
  transition
}
```

This follows the standard setup for standard stochastic dynamic programming, see Marescot et al. (2013). Having defined a function to compute the transition matrix, we can use it to create matrices corresponding to each of the three models:

```
transitions <- lapply(
  seq_along(models),
  function(i) {
    transition_matrices(
      models[[i]],
      states,
      actions,
      model_sigmas[[i]]
    )
  }
)
names(transitions) <- c("1", "2", "3")
```

Likewise, a corresponding matrix defining the rewards associated with each state X and each harvest action H can also be defined.

```
## Compute reward matrix (shared across all models)
n_s <- length(states)
n_a <- length(actions)
reward <- array(0, dim = c(n_s, n_a))
for (k in 1:n_s) {
  for (i in 1:n_a) {
    reward[k, i] <- reward_fn(states[k], actions[i])
  }
}
```

Optimal control solutions

We use value iteration to solve the stochastic dynamic program (Marescot et al. 2013; Chades et al. 2017) for each model. This determines the optimal harvest policy for each possible state, given each model. Because this step is the most computationally intensive routine, we cache the results using memosization conditioned on the transition matrices (Wickham et al. 2017). Running this code with alternate transition matrices automatically invalidates that cache, reducing the risk of loading spurious results.

```
mdp <- memoise::memoise(mdp_value_iteration,
  cache = memoise::cache_filesystem("cache/")
)

policies <-
  map_dfr(transitions,
    function(P) {
      soln <- mdp(P, reward,
        discount = discount,
        epsilon = 0.01, max_iter = 1000, V0 = rep(0, dim(P)[[1]])
      )
      escapement <- states - actions[soln$policy]
      tibble(states, policy = soln$policy, escapement)
    },
    .id = "model"
  )
```

Simulations and step-ahead forecasts

We simulate fishing dynamics under the optimal policy for each model, using a simple helper function from the `mdplearning` package. Because growth dynamics are stochastic, we perform 100 simulations of each model from identical starting condition to ensure results are not the result of chance alone.

```
library(mdplearning)
Tmax <- 100
x0 <- which.min(abs(states - 6))
reps <- 100
set.seed(12345)

## Simulate each policy reps times, with `3` as the true model:
simulate_policy <- function(i, policy) {
  mdp_planning(transitions[[true_model]], reward, discount,
    policy = policy, x0 = x0, Tmax = Tmax
  ) %>%
    select(value, state_index = state, time, action_index = action) %>%
    mutate(state = states[state_index])
}

sims <-
  map_dfr(names(transitions),
    function(m) {
      policy <- policies %>%
        filter(model == m) %>%
        pull(policy)
      map_dfr(1:reps, simulate_policy, policy = policy, .id = "reps")
    },
```

```
    .id = "model"
  )
```

Using the transition matrices directly, we can examine what each model would have forecast the future stock size to be in the following year when no fishing occurs (note that for each model, we use the transition matrix that corresponds to 'no fishing', `model[[state_index, ,1]]`) (Fig 1a, main text).

The transition matrices give the full (discretized) probability distribution, from which we can easily calculate both the expected value and the 95% confidence interval.

```
stepahead_unfished <- sims
stepahead_unfished$state_index <- rep(sims$state_index[sims$model == "1"], 3)

stepahead_unfished <- stepahead_unfished %>%
  filter(model != "3") %>%
  mutate(next_state = dplyr::lead(state_index), model = as.integer(model)) %>%
  rowwise() %>%
  mutate(
    expected = transitions[[model]][state_index, , 1] %*% states,
    var = transitions[[model]][state_index, , 1] %*% states^2 - expected^2,
    low = states[max(which(cumsum(transitions[[model]][state_index, , 1]) < 0.025))],
    high = states[min(which(cumsum(transitions[[model]][state_index, , 1]) > 0.975))],
    true = states[next_state]
  )
```

We also look at the forecast each model makes when implementing the corresponding optimal harvest:

```
stepahead_fished <- sims %>%
  filter(model != "3") %>%
  mutate(next_state = dplyr::lead(state_index), model = as.integer(model)) %>%
  rowwise() %>%
  mutate(
    prob = transitions[[model]][state_index, next_state, action_index],
    expected = transitions[[model]][state_index, , action_index] %*% states,
    var = transitions[[model]][state_index, , action_index] %*% states^2 - expected^2,
    low = states[max(which(cumsum(transitions[[model]][state_index, , action_index]) < 0.025))],
    high = states[min(which(cumsum(transitions[[model]][state_index, , action_index]) > 0.975))],
    true = states[next_state]
  ) %>%
  select(time, model, true, expected, low, high, var, prob, reps)
```

Proper scores

It is straight forward to apply the proper scoring formula of Gneiting and Raftery (2007) based on the first two moments of the distribution to score the respective forecasts under both the unfished and actively managed scenario sfor each model:

```
# Gneiting & Raferty (2007), eq27
scoring_fn <- function(x, mu, sigma) {
  -(mu - x)^2 / sigma^2 - log(sigma)
}

stepahead_unfished <- stepahead_unfished %>%
  mutate(
    sd = sqrt(var),
```

```
    score = scoring_fn(expected, true, sd)
  )

stepahead_fished <- stepahead_fished %>%
  mutate(
    sd = sqrt(var),
    score = scoring_fn(expected, true, sd)
  )
```

```
predictions <-
  stepahead_unfished %>%
  select(time, model, reps, expected, low, high, true, score) %>%
  mutate(scenario = "A_unfished") %>%
  bind_rows(stepahead_fished %>%
    select(time, model, reps, expected, low, high, true, score) %>%
    mutate(scenario = "B_fished")) %>%
  mutate(model = as.character(model))
```

Plotting

Having computed a single data frame of simulation data under both optimally fished (according to each model) and unfished scenarios for each model, along with the proper scores for the step-ahead forecasts a la Gneiting and Raftery (2007), we have all the results necessary to generate the figures presented in the main paper.

```
fig1ab <- predictions %>%
  filter(reps == "2", time < 10) %>%
  ggplot(aes(time, col = model, fill = model)) +
  geom_point(aes(y = expected)) +
  geom_errorbar(aes(ymin = low, ymax = high)) +
  geom_point(aes(y = true),
    pch = "*", size = 12,
    alpha = 0.6
  ) +
  facet_wrap(~scenario, ncol = 1, labeller = as_labeller(c(
    A_unfished = "A. Without fishing",
    B_fished = "B. Managed harvest"
  ))) +
  ylab("stock size")
```

Likewise we can plot the data on proper scores associated with each prediction of each model:

```
fig1cd <- predictions %>%
  ggplot(aes(x = score, group = model, fill = model)) +
  geom_histogram(binwidth = 2, show.legend = FALSE) +
  coord_cartesian(xlim = c(-100, 1), ylim = c(0, 4000)) +
  xlab("Proper score") +
  facet_wrap(~scenario, ncol = 1, labeller = as_labeller(c(
    A_unfished = "C.",
    B_fished = "D."
  ))) +
  scale_x_continuous(breaks = c(-100, 0)) +
  theme(
    axis.text.y = element_blank(),
```

```
    plot.margin = margin(0, 0, 0, 0, "cm"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  )
```

Forecast ecological and economic performance

The ecological performance of the model can easily be visualized by plotting the results of each simulation.

```
fig2a <-
  sims %>%
  group_by(model, time) %>%
  summarise(mean_state = mean(state), sd = sd(state), .groups = "drop") %>%
  filter(time < 25, model != "3") %>%
  ggplot(aes(time, mean_state)) +
  geom_line(aes(col = model), lwd = 1.5, show.legend = FALSE) +
  geom_ribbon(aes(
    ymin = mean_state - 2 * sd,
    ymax = mean_state + 2 * sd,
    fill = model
  ),
  alpha = 0.2, show.legend = FALSE
  ) +
  ylab("state") +
  ggtitle("A") +
  labs(subtitle = "Ecological")
```

To plot the economic value over time, we must sum up the discounted values at each time step, and then average over replicate simulations of each model:

```
##  Net Present Value accumulates over time
npv_df <- sims %>%
  group_by(model, reps) %>%
  mutate(npv = cumsum(value * discount^time)) %>%
  group_by(time, model) %>%
  summarise(mean_npv = mean(npv), .groups = "drop") %>%
  arrange(model, time)

optimal <- select(filter(npv_df, model == "3"), time, mean_npv)

fig2b <-
  npv_df %>%
  filter(model != "3", time %in% seq(1, 100, by = 5)) %>%
  ggplot(aes(time, mean_npv)) +
  geom_line(data = optimal, lwd = 1.5, col = "grey20") +
  geom_point(aes(col = model), size = 4, alpha = 0.8) +
  ylab("Net present value") +
  xlab("time") +
  ggtitle("B.") +
  labs(subtitle = "Economic")
```

Plotting the model functions themselves are shown in Fig 3a:

```
d <-
  map_dfc(models, function(f) f(states) - states) %>%
  mutate(state = states)
```

```
fig3a <-
  d %>%
  pivot_longer(names(models), "model") %>%
  ggplot(aes(state, value, col = model, lty = model)) +
  geom_hline(aes(yintercept = 0), lwd = 1) +
  geom_line(lwd = 2, show.legend = FALSE) +
  coord_cartesian(ylim = c(-5, 8), xlim = c(0, 16)) +
  ylab(bquote(f(x))) +
  xlab("x")
```

We can easily plot the optimal policy derived from each model, as shown in Fig3b.

```
fig3b <- policies %>%
  ggplot(aes(states, escapement, col = model, lty = model)) +
  geom_line(lwd = 2) +
  xlab("state")
```

References

Carl Boettiger, Milad Memarzadeh. 2018. *Mdplearning: Bayesian Learning Algorithms for Markov Decision Processes* (version 0.1.0). Zenodo. https://doi.org/10.5281/zenodo.1161519.

Chades, Iadine, Guillaume Chapron, Marie-Josee Cros, Frederick Garcia, and Regis Sabbadin. 2017. *MDPtoolbox: Markov Decision Processes Toolbox*.

Clark, Colin W. 1973. "Profit Maximization and the Extinction of Animal Species." *Journal of Political Economy* 81 (4): 950–61. https://doi.org/10.1086/260090.

Gneiting, Tilmann, and Adrian E Raftery. 2007. "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association* 102 (477): 359–78. https://doi.org/10.1198/016214506000001437.

Goulet, Vincent, Christophe Dutang, Martin Maechler, David Firth, Marina Shapira, and Michael Stadelmann. 2020. *Expm: Matrix Exponential, Log, 'Etc'*. http://R-Forge.R-project.org/projects/expm/.

Halpern, Benjamin S, Carissa J Klein, Christopher J Brown, Maria Beger, Hedley S Grantham, Sangeeta Mangubhai, Mary Ruckelshaus, et al. 2013. "Achieving the Triple Bottom Line in the Face of Inherent Trade-Offs Among Social Equity, Economic Return, and Conservation." *Proceedings of the National Academy of Sciences* 110 (15): 6229–34. https://doi.org/10.1073/pnas.1217689110.

Mangel, Marc. 1985. "Decision and control in uncertain resource systems," 255. http://dl.acm.org/citation.cfm?id=537497.

Marescot, Lucile, Guillaume Chapron, Iadine Chadès, Paul L. Fackler, Christophe Duchamp, Eric Marboutin, and Olivier Gimenez. 2013. "Complex Decisions Made Simple: A Primer on Stochastic Dynamic Programming." *Methods in Ecology and Evolution* 4 (9): 872–84. https://doi.org/10.1111/2041-210X.12082.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing* (version 3.6.2). Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Reed, William J. 1979. "Optimal escapement levels in stochastic and deterministic harvesting models." *Journal of Environmental Economics and Management* 6 (4): 350–63. https://doi.org/10.1016/0095-0696(79)90014-7.

Schaefer, Milner B. 1954. "Some aspects of the dynamics of populations important to the management of the commercial marine fisheries." *Bulletin of the Inter-American Tropical Tuna Commission* 1 (2): 27–56. https://doi.org/10.1007/BF02464432.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Jim Hester, Kirill Müller, and Daniel Cook. 2017. *Memoise: Memoisation of Functions*. https://github.com/hadley/memoise.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.