Casey O'Hara

Econ 260A

2018-10-18

Code for Homework Challenge 1

DPE functions

`calc_dpe()` is the function to pass to `optim()`. Since `optim()` is attempting to minimize the function, we convert the value to a negative number so `optim()` determines a maximum.

$$-V_t(x_t) = -\max_{h_t}[\pi(h_t) + \delta V_{t+1}(x_{t+1})]$$

`calc_payoff()` calculates the payoff:

$$\pi(h, \alpha, \beta) = \alpha h - \beta h^2$$

`calc_motion()` calculates the equation of motion:

$$x_{t+1} = z_t f(y_t) = z_t \left(y_t + r y_t \left(\frac{1 - y_t}{K} \right) \right)$$

```
calc_payoff <- function(h, a, b) {
  ### payoff for a given harvest
  if(a <= 0 | b <= 0) {
    stop('Alpha and beta terms must be greater than zero.')
  }
  if(h < 0) {
    warning('Harvest must not be negative. Setting h = 0.')
    h <- 0
  }
  pay  <- a * h - b * h^2
  return(pay)
}

calc_motion <- function(y, r, K) {
  ### equation of motion with stochastic multiplier term
  f_y <- y + r * y * (1 - y / K)
  x_tplus1 <- f_y
  return(x_tplus1)
}

calc_dpe <- function(harvest, stock,
                     alpha, beta, delta,
                     r, K, theta = 0,
                     x_vec, V) {
  ### To be used in optimization...this function is minimized by choosing h

  ### Calc payment and x_{t+1} terms
```

```
  pay_t <- calc_payoff(harvest, a = alpha, b = beta)

  ### calc stock status for no shock, then low and high shocks from that
  x_no_shock <- calc_motion(y = stock - harvest, r = r, K = K)
  x_lo_shock <- (1 - theta) * x_no_shock
  x_hi_shock <- (1 + theta) * x_no_shock

  ### Note: if theta = 0, this just calcs v_exp as the un-shocked thing
  v_lo_shock <- spline(x_vec, V, xout = x_lo_shock, method = 'natural')
  v_hi_shock <- spline(x_vec, V, xout = x_hi_shock, method = 'natural')

  v_exp <- .5 * v_lo_shock$y + .5 * v_hi_shock$y

  ### return negative value; optim will minimize this, maximizing the payout
  negout <- -(pay_t + delta * v_exp)

  if(is.infinite(negout)) negout <- 0

  return(negout)
}
```

Optimization functions

`optimize_dpe()` iterates over t and x to determine optimal harvests and values. It returns a list with the h^* matrix and V matrix.

`assemble_df()` takes the outputs from `optimize_dpe()` and arranges them into a more convenient format: a dataframe with variables `index`, `x_vec`, `harvest_opt`, `t_end`, and `value_fn`.

```
optimize_dpe <- function(x_vec, TT, alpha, beta, delta, r, K, theta = 0) {
  ### Iterates for each time period and each stock value to determine optimal
  ### policy and optimal value functions.
  # cat('in optimize_dpe: z = ', z, '...\n')

  v_mat  <- matrix(0,  nrow = length(x_vec), ncol = TT + 1)
    ### initialize value matrix
  h_star <- matrix(NA_real_, nrow = length(x_vec), ncol = TT)
    ### Initialize the control vector
  v_new  <- rep(NA_real_, length = length(x_vec))
    ### Initialize the 'new' value function

  for (t in TT:1) { ### count down from end time to beginning time
    ### t <- TT
    message('Period t = ', t)
    for (i in seq_along(x_vec)) {
      ### i <- 1
      guess <- 0
      x <- x_vec[i]

      ### This finds optimal policy function
      thing <- optim(par = guess,
                     fn  = calc_dpe,
                     gr  = NULL,
```

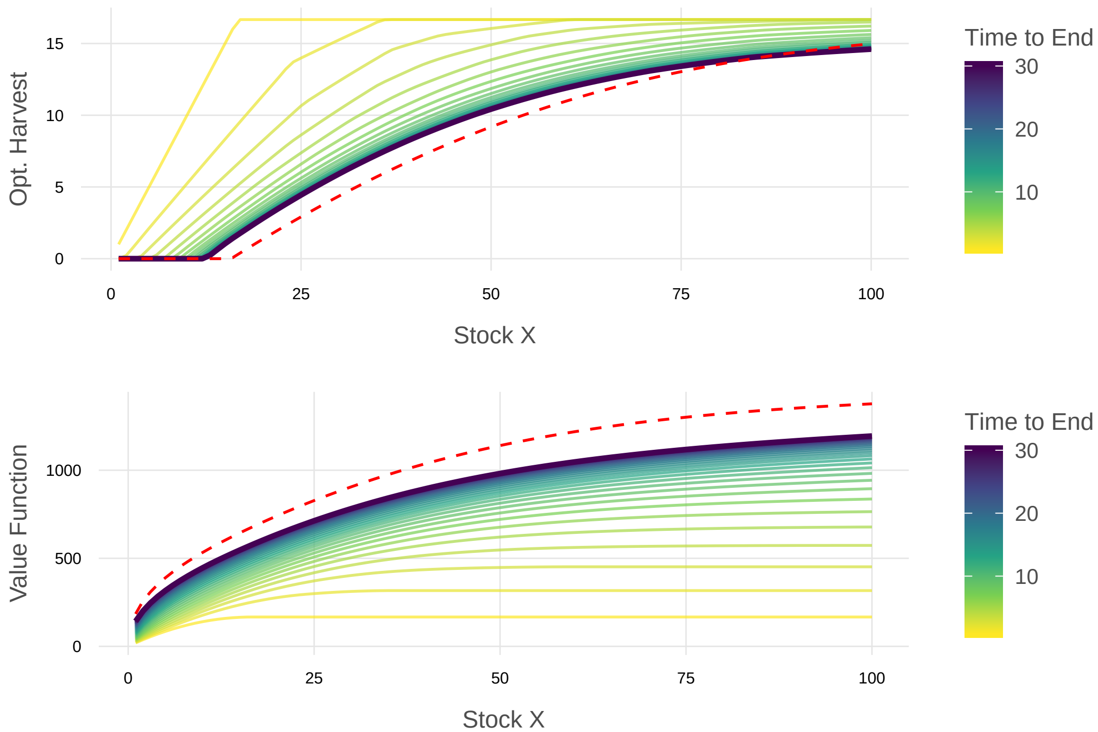
```r
                     lower = 0, upper = x, ### bounds on h
                     stock = x,
                     alpha = alpha, beta = beta, delta = delta, theta = theta,
                     r = r, K = K,
                     x_vec = x_vec, V = v_mat[ , t+1],
                     method = 'L-BFGS-B')

      h_star[i, t] <- thing$par
        ### the optimal parameter
      v_new[i] <- -thing$value
        ### the value of the dpe at the optimal parameter

    } ### end of loop for this value of x in time t

    v_mat[ , t] = v_new
  } ### end of t loop

  return(list(h_star, v_mat))
}

assemble_df <- function(opt_fxns, x_vec, TT, K, theta = 0) {
  ### convert matrices to data.frames, clip v_mat to T = 30
  h_df <- opt_fxns[[1]] %>% data.frame()
  v_df <- opt_fxns[[2]][ , 1:TT] %>% data.frame()

  harvest_df <- data.frame(index = 1:K,
                           x_vec,
                           h_df)
  value_df   <- data.frame(index = 1:K,
                           x_vec,
                           v_df)

  h_df_long <- harvest_df %>%
    gather(key = xtime, value = harvest_opt, paste0('X', 1:TT)) %>%
    mutate(xtime = as.numeric(str_extract(xtime, '[0-9]+'))) %>%
    mutate(t_end = TT + 1 - xtime) %>%
    select(-xtime)

  v_df_long <- value_df %>%
    gather(key = xtime, value = value_fn, paste0('X', 1:TT)) %>%
    mutate(xtime = as.numeric(str_extract(xtime, '[0-9]+'))) %>%
    mutate(t_end = TT + 1 - xtime) %>%
    select(-xtime)

  df <- left_join(h_df_long, v_df_long,
                  by = c('index', 'x_vec', 't_end')) %>%
    mutate(theta = theta)

  return(df)
}
```

Problem 5

Retaining the non-linear payoff function, now assume the stock evolves in a stochastic fashion: $x_{t+1} = z_t f(y_t)$, where $z_t = \{1-\theta, 1+\theta\}$ (i.e. it takes one of those two values, each with probability 0.5), and $f(y_t) = y_t + r y_t (1 - y_t/K)$. Write a computer program that solves this problem numerically. You can use the following parameters: $\delta = .9$, $\alpha = 20$, $\beta = .6$, $r = .3$, $K = 100$, $\theta = .3$. How does the converged optimal policy function depend on θ, r, δ, and α?

```
### set up parameters
alpha <- 20; beta <- .6; delta <- .9
theta <- .3
r  <- 0.3; K <- 100
TT <- 30

### Set up grid over state space
x_vec <-  1:K

### optimize_dpe() returns a list of (h_star, v_mat)
opt_fxns <- optimize_dpe(x_vec, TT, alpha, beta, delta, r, K, theta)
opt_df <- assemble_df(opt_fxns, x_vec, TT, K, theta)

opt_fxns_noshock <- optimize_dpe(x_vec, TT, alpha, beta, delta, r, K, theta = 0)
opt_df_noshock <- assemble_df(opt_fxns_noshock, x_vec, TT, K, theta = 0)

plot1 <- ggplot(opt_df, aes(x = x_vec, y = harvest_opt,
                            color = t_end, group = t_end)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(size = 0.5, alpha = .7) +
  geom_line(data = opt_df %>% filter(t_end == TT),
            size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1) +
  labs(x = 'Stock X',
       y = 'Opt. Harvest',
       color = 'Time to End')

plot2 <- ggplot(opt_df, aes(x = x_vec, y = value_fn,
                            color = t_end, group = t_end)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(size = 0.5, alpha = .7) +
  geom_line(data = opt_df %>% filter(t_end == TT),
            size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1) +
  labs(x = 'Stock X',
       y = 'Value Function',
       color = 'Time to End')

plot3 <- plot_grid(plot1, plot2, nrow = 2)
```

```r
print(plot3)
```




```r
ggsave(plot = plot3, filename = '5_convergence.png', width = 6, height = 4)
```

```r
### This will be used for question 6 as well - protect this variable
opt_df_converged <- opt_df %>%
  filter(t_end == TT)
opt_df_for_6 <- opt_df_converged
```

How does the converged optimal policy function depend on θ, r, δ, and α?

Test α:

Vary $\alpha \in 5, 10, 20, 30, 50$

```r
a_vec  <- c(5, 10, 20, 30, 50)
a_list <- vector('list', length = length(a_vec))

for(i in seq_along(a_vec)) {
  a <- a_vec[i]
  message('optimizing for alpha = ', a)
  opt_fxns <- optimize_dpe(x_vec, TT, a, beta, delta, r, K, theta)

  df_converge <- assemble_df(opt_fxns, x_vec, TT, K, theta) %>%
    filter(t_end == TT) %>%
    mutate(alpha = a)

  a_list[[i]] <- df_converge
}
a_df <- bind_rows(a_list)
```

```r
alpha_plot <- ggplot(a_df, aes(x = x_vec, y = harvest_opt,
                               color = alpha, group = alpha)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(size = .5, alpha = .7) +
  geom_line(data = a_df %>% filter(alpha == 20),
            size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1, breaks = a_vec) +
  labs(x = 'Stock X',
       y = 'Opt. Harvest',
       color = 'alpha')
```

Test θ:

Vary $\theta \in 0, 0.15, 0.3, 0.5, 0.7$

```r
th_vec  <- c(0, 0.15, 0.3, 0.5, 0.7)
th_list <- vector('list', length = length(th_vec))

for(i in seq_along(th_vec)) { ### i <- 2
  th <- th_vec[i]
  message('optimizing for theta = ', th)

  opt_fxns  <- optimize_dpe(x_vec, TT, alpha, beta, delta, r, K, theta = th)

  df_converge <- assemble_df(opt_fxns, x_vec, TT, K, theta = th) %>%
    filter(t_end == TT) %>%
    mutate(theta = th)

  th_list[[i]] <- df_converge
}
th_df <- bind_rows(th_list) %>%
  distinct()

theta_plot <- ggplot(data = th_df, aes(x = x_vec, y = harvest_opt,
                          color = theta, group = theta)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(data = th_df, size = .5, alpha = .7) +
  geom_line(data = th_df %>% filter(theta == 0.3), size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1, breaks = th_vec) +
  labs(x = 'Stock X',
       y = 'Opt. Harvest',
       color = 'theta')
```

Test δ:

Vary $\delta \in .5, .7, .9, .95, 1$

```
d_vec  <- c(.5, .7, .9, .95, 1)
d_list <- vector('list', length = length(d_vec))

for(i in seq_along(d_vec)) { ### i <- 1
  d <- d_vec[i]
  message('optimizing for delta = ', d)

  opt_fxns <- optimize_dpe(x_vec, TT, alpha, beta, d, r, K, theta)

  df_converge <- assemble_df(opt_fxns, x_vec, TT, K, theta) %>%
    filter(t_end == TT) %>%
    mutate(delta = d)

  d_list[[i]] <- df_converge
}
d_df <- bind_rows(d_list)

delta_plot <- ggplot(d_df, aes(x = x_vec, y = harvest_opt,
                               color = delta, group = delta)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(size = .5, alpha = .7) +
  geom_line(data = d_df %>% filter(delta == 0.9),
            size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1, breaks = d_vec) +
  labs(x = 'Stock X',
       y = 'Opt. Harvest',
       color = 'delta')
```

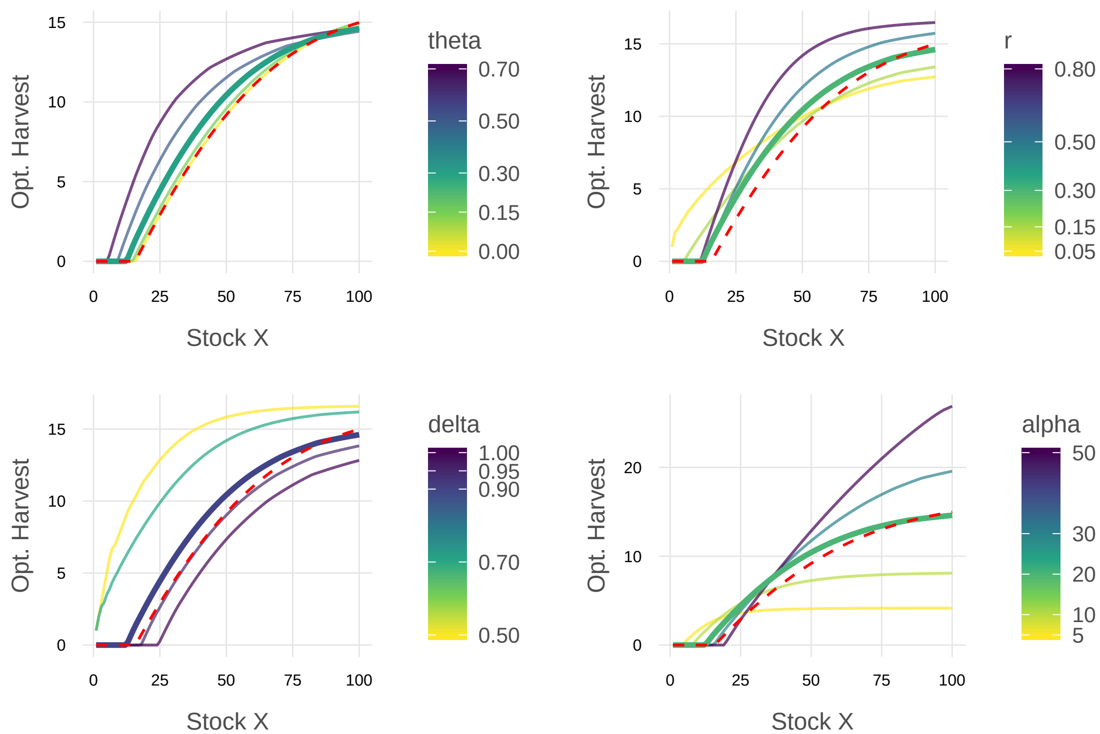
Test r:

Vary $r \in .05, .15, .3, .5, .8$

```
r_vec  <- c(.05, .15, .3, .5, .8)
r_list <- vector('list', length = length(r_vec))

for(i in seq_along(r_vec)) { ### i <- 1
  r_var <- r_vec[i]
  message('optimizing for r = ', r_var)

  opt_fxns <- optimize_dpe(x_vec, TT, alpha, beta, delta, r_var, K, theta)

  df_converge <- assemble_df(opt_fxns, x_vec, TT, K, theta) %>%
    filter(t_end == TT) %>%
    mutate(r = r_var)

  r_list[[i]] <- df_converge
}
r_df <- bind_rows(r_list)

r_plot <- ggplot(r_df, aes(x = x_vec, y = harvest_opt,
```

```r
                              color = r, group = r)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_line(size = .5, alpha = .7) +
  geom_line(data = r_df %>% filter(r == 0.3),
            size = 1, alpha = 1) +
  geom_line(data = opt_df_noshock %>% filter(t_end == TT),
            size = .5, alpha = 1, linetype = 'dashed', color = 'red') +
  scale_color_viridis_c(direction = -1, breaks = r_vec) +
  labs(x = 'Stock X',
       y = 'Opt. Harvest',
       color = 'r')
```

```r
quad_plot <- cowplot::plot_grid(theta_plot, r_plot, delta_plot, alpha_plot,
                                nrow = 2)
print(quad_plot)
```




```r
ggsave(plot = quad_plot, filename = '5_param_plots.png', height = 3.2, width = 6)
```

Problem 6

Suppose $x_0 = 15$. Using the infinite-horizon optimal policy function, simulate the optimized system forward for 20 years under baseline parameters above. Run 10 separate simulations and plot the trajectory of x_t and h_t over time for each simulation. (You should produce 2 plots, one for the 10 trajectories of x_t and one for the 10 trajectories of h_t).

```r
converge_df <- opt_df_for_6

### set up sim counts and such
```

```r
sim_yrs <- 50
sims    <- 500
sims_list <- vector('list', length = sims)

for(sim in 1:sims) {

  ### Set up vecs and initial values for this simulation
  x <- rep(0, length = TT)
  h <- rep(0, length = TT)
  x[1] <- 15
  # cat('sim #', sim, '\n...')

  for (t in 1:sim_yrs) { ### t <- 1

    ### calc a stochastic multiplier for this period on this simulation
    z <- ifelse(runif(1) > 0.5, 1 + theta, 1 - theta)

    # cat('t = ', t, '\n')
    h_tmp <-  spline(converge_df$x_vec, converge_df$harvest_opt,
                     xout = x[t], method = 'natural')
    h[t] <- h_tmp$y
    if(h[t] < 0) {
      warning('Harvest must not be negative... setting to zero')
      h[t] <- 0
    }

    # cat('t =', t, ': x[t] =', x[t], '... z =', z, '... h[t] =', h[t], '...\n')

    x[t+1] <- calc_motion(y = x[t] - h[t], r = r, K = K) * z
  }

  sims_list[[sim]] <- data.frame(sim     = sim,
                                 year    = 1:sim_yrs,
                                 stock   = x[1:sim_yrs],
                                 harvest = h[1:sim_yrs])
}

sims_df <- bind_rows(sims_list)

means_df <- sims_df %>%
  group_by(year) %>%
  summarize(mean_stock = mean(stock),
            mean_harvest = mean(harvest)) %>%
  ungroup()

stock_plot <- ggplot(sims_df, aes(x = year, y = stock, color = sim, group = sim)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_vline(xintercept = 20, color = 'grey40') +
  geom_line(show.legend = FALSE, alpha = 0.2, size = .5) +
  geom_line(data = means_df, aes(y = mean_stock), color = 'red4', size = 1) +
  ylim(0, NA) +
  scale_color_viridis_c()
```

```
harvest_plot <- ggplot(sims_df, aes(x = year, y = harvest, color = sim, group = sim)) +
  ggtheme_plot() +
  theme(axis.text = element_text(size = 6)) +
  geom_vline(xintercept = 20, color = 'grey40') +
  geom_line(show.legend = FALSE, alpha = 0.2, size = .5) +
  geom_line(data = means_df, aes(y = mean_harvest), color = 'red4', size = 1) +
  ylim(0, NA) +
  scale_color_viridis_c()

stock_harv_plot <- plot_grid(stock_plot, harvest_plot, nrow = 2)

ggsave(filename = '6_simulations.png', width = 6, height = 3.2)

print(stock_harv_plot)
```