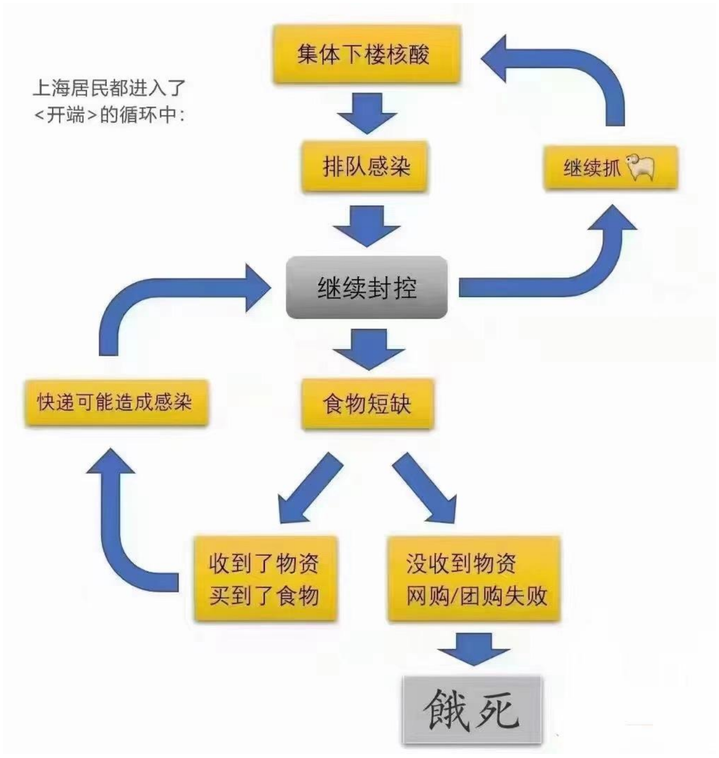
Dynamic Zero Policy for COVID-19

Weibin Gu, April 9, 2022 | Last update: April 10, 2022

This live script models people's life under dynamic zero policy in the situation of COVID-19 in China. The idea is not coming from pandemics in a strict sense, but from the self-mockery of Chinese people as illustrated below.



The code is self-explanatory and have quite a few parameters to play with. So far, it is only supported for MATLAB but can be easily implemented in other programming languages. Again, this is only a toy model that was developed for joyfulness. Feel free to make modifications or re-invent on top of this script and leave comments on mistakes or ways to improve. Thank you.

Stamina

TODO: Classify people into the elder, baby, adult, and teenager.

```
S0 = 1; % initial condition
%S = S0; % declared later
maximum_lifetime = 5% maximum lifetime under starving [day]
```

```
maximum_lifetime = 5
```

```
delta_S = 1/maximum_lifetime; % stamina loss per day [%]
```

Food stock

F is the number of days that current food stock can support your life. At the end of each day, this will be reduced by 1 unless food stock is already empty.

Food consumption can be related to stamina loss by $\Delta F = \Delta S$ when food is sufficient (i.e., $F \geq 1$), and $\Delta F = F * \Delta S$ when $0 \leq F < 1$. Of course, $\Delta F = 0$ when $F = 0$.

```
maximum_foodstock =7 % (initial condition) maximum food stock [day]
```

```
maximum_foodstock = 7
```

```
    % This is a critical variable affected by government annoucement, which
    % can be set as the anounced (expected) quarantined period.
%F = maximum_foodstock; % declared later
```

Amount of food delivery F_d, which will be directly added to food stock on its arrival: $F = F + F_d$. It is assumed that only one food delivery can be ordered at a time and the second can be only ordered when the first one is completed. This can be assumed if we can smooth the sequence of number of orders and then discretize it, from my imagination...

```
food_delivery = 3% number of days that food delivery can support your life [day]
```

```
food_delivery = 3
```

```
Fd = food_delivery;
```

Infection probability

COVID is powerful, and can be spread when you are in the queue to do nucleic acid test or even receiving deliveries (viruses are attached on the surface of the goods).

```
Pi_q = 5/100 % during nucleic acid test queue [%]
```

```
Pi_q = 0.0500
```

```
Pi_d = 1/100 % during receiving deliveries [%]
```

```
Pi_d = 0.0100
```

Time interval of nucleic acid test and the arrival of deliveries

Basically, these are frequencies.

I also added randomness to food delivery frequency (in the final loop) to make things more interesting and realistic.

```
tq = 3 % (tq)-days per nucleic acid test
```

```
tq = 3
```

```
td = 7 % (td)-days per arrival of deliveries
```

```
td = 7
```

```

%td_rdm = round(td+randn(1)) % if you want to see the ideal case, delete
%randn(1)
```

Let's roll the dice!

Stamina model: $S_{k+1} = S_k - \Delta S + \Delta F$, where ΔF is defined previously.

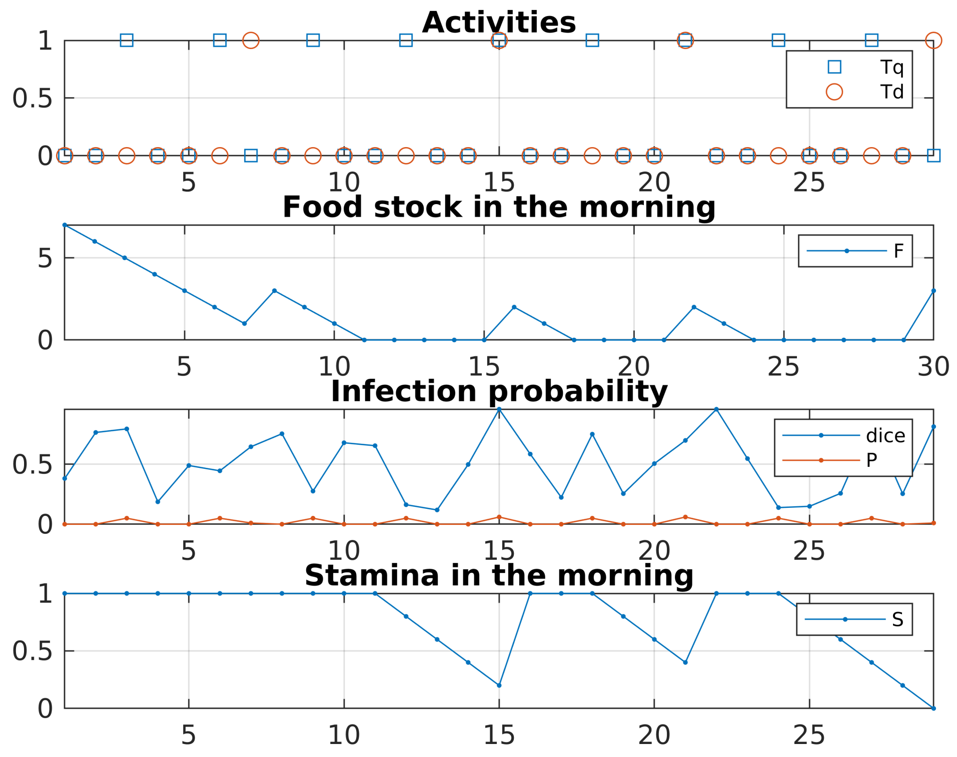
Infection model: $p = p_{i,q} * T_q + p_{i,d} * T_d$, assuming two infections are independent, and T_q, T_d are "trigger parameters" which can be calculated by $T_i = (k \bmod t_i) == 0, i = \{q, d\}$ for the ideal case. The following code also takes into account randomness of food delivery.

```
clear Tq Td P dice S F
S(1) = S0; % intialization
F(1) = maximum_foodstock; % intialization
td_ref = 0; % reference day to receive food delivery [day]
is_delivered = true;

for k = 1:1000
    Tq(k) = mod(k,tq)==0;
    if is_delivered
        is_delivered = false;
        td_rdm = round(td+randn(1)); % random frequency of food delivery [day]
        td_ref = td_ref + td_rdm; % the day of food to be delivered [day]
    end
    if k == td_ref
        is_delivered = true;
        Td(k) = true;
    else
        Td(k) = false;
    end
    P(k) = Tq(k)*Pi_q + Td(k)*Pi_d; % infection model
    if Td(k)
        F(k+1) = F(k) + Fd; % food delivery
    else
        F(k+1) = F(k);
    end
    dice(k) = rand(1); % today's dice [%]
    if (S(k) <= 0.001) % FIXED: 0.001 (instead of 0) is used to increase numerical stab
        fprintf('You starve to death on Day %d.',k);
        break
    elseif (dice(k) <= P(k))
        fprintf('You get infectious on Day %d.',k);
        break
    else
        % Congratulations! You survived and started your day :)
        if F(k+1) >= 1
            refuel = (1-S(k))/delta_S; % refuel your stamina if it is not full [day]
            delta_F = delta_S * (1 + refuel);
        elseif F(k+1) >=0 && F(k+1) < 1
            delta_F = F(k+1)*delta_S; % start hungry
        else
            delta_F = 0; % woops, no food to eat
        end
        S(k+1) = S(k) - delta_S + delta_F; % stamina model
    end
    if F(k+1) > 0
        F(k+1) = F(k+1) - 1;
    else
        F(k+1) = F(k+1);
    end
end
```

```
You starve to death on Day 29.
```

Let's look back over your life... (so sad)

```
figure,
subplot(4,1,1), plot(Tq,'s'), hold on, plot(Td,'o'), legend('Tq','Td'), grid on, axis t
subplot(4,1,2), plot(F,'.-'), legend('F'), grid on, axis tight, title('Food stock in th
subplot(4,1,3), plot(dice,'.-'), hold on, plot(P,'.-'), legend('dice','P'), grid on, ax
subplot(4,1,4), plot(S,'.-'), legend('S'), grid on, axis tight, title('Stamina in the m
```



```
fprintf("Conclusions: Assuming you can live without eating and drinking for %d days," +
    "and start quarantining with food stock of %d days. " + ...
    "You order food delivery every %d days," + ...
    "but finally your life ends on %d day, unfortunately.", ...
    maximum_lifetime, ...
    maximum_foodstock, ...
    food_delivery, ...
    k);
```

```
Conclusions: Assuming you can live without eating and drinking for 5 days,and start quarantining with food
```

Let's look at a big picture.

```
clear num_death num_inf num_day
num_death = 0;
num_inf = 0;
num_day = 0; % survival day

% THE FOLLOWING IS THE SAME CODE WITH MINOR MODIFICATIONS
% AS SEEN IN 'ROLLING THE DICE', BUT HERE WE"RE RUNNING
% A NUMBER (j) OF CAMPAIGNS TO CALCULATE AVERAGE VALUES
for j = 1:100
    clear Tq Td P dice S F
    S(1) = S0;
    F(1) = maximum_foodstock;
    for k = 1:1000
        Tq(k) = mod(k,tq)==0;
        if is_delivered
            is_delivered = false;
            td_rdm = round(td+randn(1));
            td_ref = td_ref + td_rdm;
        end
        if k == td_ref
            is_delivered = true;
            Td(k) = true;
        else
            Td(k) = false;
        end
        P(k) = Tq(k)*Pi_q + Td(k)*Pi_d;
        if Td(k)
            F(k+1) = F(k) + Fd;
        else
            F(k+1) = F(k);
        end
        dice(k) = rand(1);
        if (S(k) <= 0.001) % FIXED: 0.001 (instead of 0) is used to increase numerical
            num_death = num_death + 1; % accumulate
            break
        elseif (dice(k) <= P(k))
            num_inf = num_inf + 1; % accumulate
            break
        else
            if F(k+1) >= 1
                refuel = (1-S(k))/delta_S;
                delta_F = delta_S * (1 + refuel);
            elseif F(k+1) >=0 && F(k+1) < 1
                delta_F = F(k+1)*delta_S;
            else
                delta_F = 0;
            end
            S(k+1) = S(k) - delta_S + delta_F;
        end
        F(k+1) = F(k+1) - 1;
    end
    num_day = num_day + k; % accumulate
end

avg_inf = num_inf / j % probability of getting infectious
```

```
avg_inf = 0.1300
```

```
avg_death = num_death / j % probability of death
```

```
avg_death = 0.8700
```

```
avg_day = num_day / j
```

```
avg_day = 12.3600
```

```
fprintf("Running %d campaigns, " +...
    "it is found that getting infectious has probability %0.2f%%, " +...
    "and starving to death has probability %0.2f%%. " +...
    "The average normal day length is %0.2f.", ...
    j, ...
    avg_inf*100, ...
    avg_death*100, ...
    avg_day);
```

```
Running 100 campaigns, it is found that getting infectious has probability 13.00%, and starving to death h
```

Seeking the best policy

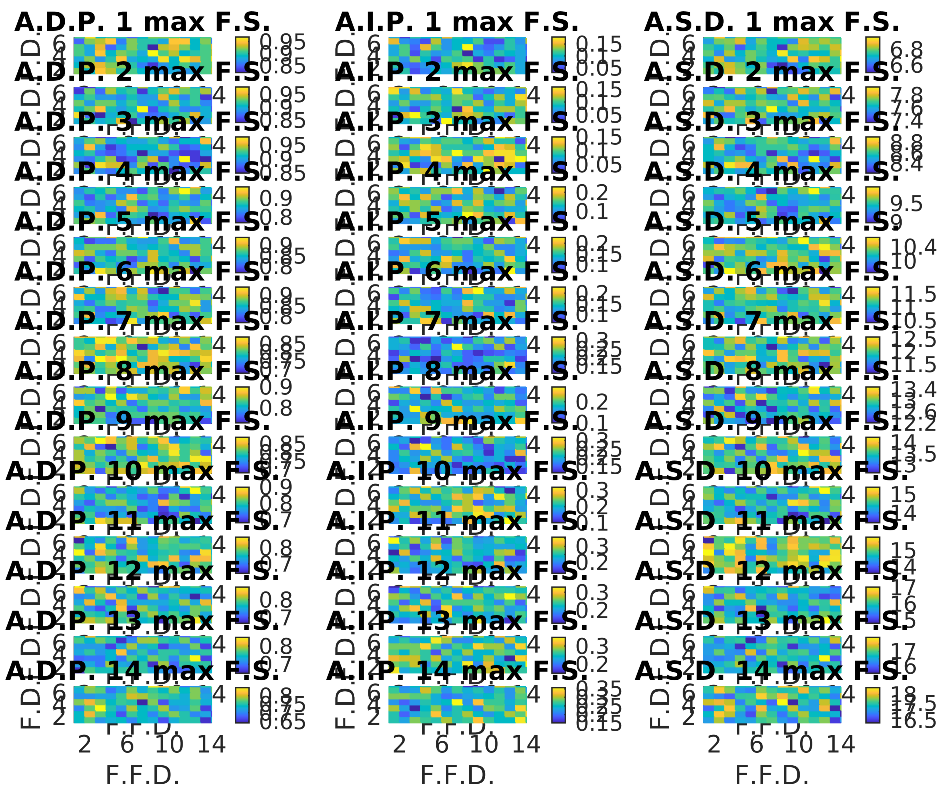
Here only 3 variables are considered as controllable (hence, decision variables) which are:

- maximum_foodstock
- food_delivery
- td

Similar to the campaign run in the "Big Picture" session, I rerun the campaign while spanning through the region of interest of these decision variables.

```
% Region of interest
maximum_foodstock = 1:14;
food_delivery = 1:7;
Fd = food_delivery;
td = 1:14;

% Campaign
for iter_mfs = maximum_foodstock
    for iter_Fd = Fd
        for iter_td = td
            clear num_death num_inf num_day
            num_death = 0;
            num_inf = 0;
            num_day = 0;

            for j = 1:100
            clear Tq Td P dice S F
            S(1) = S0;
            F(1) = maximum_foodstock(iter_mfs);
            for k = 1:1000
                Tq(k) = mod(k,tq)==0;
                if is_delivered
                    is_delivered = false;
                    td_rdm = round(td(iter_td)+randn(1));
                    td_ref = td_ref + td_rdm;
                end
                if k == td_ref
                    is_delivered = true;
                    Td(k) = true;
                else
                    Td(k) = false;
                end
                P(k) = Tq(k)*Pi_q + Td(k)*Pi_d;
                if Td(k)
                    F(k+1) = F(k) + Fd(iter_Fd);
                else
                    F(k+1) = F(k);
                end
                dice(k) = rand(1);
                if (S(k) <= 0.001)
                    num_death = num_death + 1;
                    break
                elseif (dice(k) <= P(k))
                    num_inf = num_inf + 1;
                    break
                else
                    if F(k+1) >= 1
                        refuel = (1-S(k))/delta_S;
                        delta_F = delta_S * (1 + refuel);
                    elseif F(k+1) >=0 && F(k+1) < 1
                        delta_F = F(k+1)*delta_S;
                    else
                        delta_F = 0;
                    end
                    S(k+1) = S(k) - delta_S + delta_F;
                end
                F(k+1) = F(k+1) - 1;
            end

            num_day = num_day + k;
        end

        avg_inf = num_inf / j;
        avg_death = num_death / j;
        avg_day = num_day / j;

        ret_inf(iter_Fd,iter_td,iter_mfs) = avg_inf;
        ret_death(iter_Fd,iter_td,iter_mfs) = avg_death;
        ret_day(iter_Fd,iter_td,iter_mfs) = avg_day;
    end % end of iter_td = td
    end % end of iter_Fd = Fd
end % end of iter_mfs = maximum_foodstock
```

```
figure,
for iplot = maximum_foodstock
    subplot(max(maximum_foodstock),3,1+(iplot-1)*3),
    mesh(ret_death(:,:,iplot),'FaceColor','Flat'), axis tight
    xlabel('F.F.D.')%Frequency of food delivery [day]
    ylabel('F.D.')%Food delivery [day]
    title(['A.D.P. ',num2str(iplot),' max F.S.'])%Averaged death probability; Max foods
    colorbar, view([0 90])

    subplot(max(maximum_foodstock),3,2+(iplot-1)*3),
    mesh(ret_inf(:,:,iplot),'FaceColor','Flat'), axis tight
    xlabel('F.F.D.')
    ylabel('F.D.')
    title(['A.I.P. ',num2str(iplot),' max F.S.'])%Averaged infection probability
    colorbar, view([0 90])

    subplot(max(maximum_foodstock),3,3+(iplot-1)*3),
    mesh(ret_day(:,:,iplot),'FaceColor','Flat'), axis tight
    xlabel('F.F.D.')
    ylabel('F.D.')
    title(['A.S.D. ',num2str(iplot),' max F.S.'])%Averaged survival day
    colorbar, view([0 90])
end
```


A.D.P. 1 max F.S.
A.I.P. 1 max F.S.
A.S.D. 1 max F.S.
A.D.P. 14 max F.S.
A.I.P. 14 max F.S.
A.S.D. 14 max F.S.
F.D.
F.F.D.