Zombie Population Modelling

Victoria Lang
e-mail: vml13@skku.edu
Sungkyunkwan University
Suwon, 440-746, Korea

Sang-Gu Lee[1]
e-mail: sglee@skku.edu
Sungkyunkwan University
Suwon, 440-746, Korea

Jae Hwa Lee
e-mail: jhlee2chn@skku.edu
Sungkyunkwan University
Suwon, 440-746, Korea

Abstract
While the usage of powerful mathematics software packages plays a key role in mathematics courses such as calculus, linear algebra, etc., shortcomings in these software packages exist – namely, issues of price, portability, and integration into a dynamic classroom technological environment. With these limitations in mind, this research expounds upon the mentally stimulating "Zombie Population Models" first developed by Munz, Hudea, Imad, and Smith[8]. Specifically, we modify these models to be visualized online via Sage, an open-source mathematics software based in the Python programming language, that allows for direct user interaction. Sage is very portable and does not require the user to download large software packages or learn extremely confined programming languages. We then focus on viewing mathematically "realistic" population trajectories for the different classes of zombies from Left 4 Dead, Valve Corporation's immensely popular zombie video game series. All outcomes are numerically (and visually!) realized with online Sage tools, constructed using the @intreract command in Sage, and are available for viewing, manipulation, and use in a mobile environment at http://matrix.skku.ac.kr/2014-Zombie-Model/main.htm.

1. Introduction

The outbreak and spread of virulent diseases, such as malaria, measles, smallpox, etc., is a subject that has remained a subject of interest in the academic community. By creating mathematical models of an epidemic, scientists can identify trends and patterns inherent in the spread of the disease and, accordingly, implement isolation or vaccination plans to stop its transmission. As well, mathematical models of infections (part of the larger study of the distribution and effects of epidemics called *epidemiology*) can also provide clues to the cause of the disease and lead to eradication from the source. Various epidemic models including SIR, SIS, SIRS, SEIS, SEIR, etc. have been studied for many years. For more detailed information, see [2].

The reanimated dead (i.e. "zombies") have a strong story-telling basis throughout history and have fascinated cultures across the world for centuries. In this paper, we will briefly examine basic models for zombie infection introduced by Munz, Hudea, Imad, and Smith [8], expanding upon the well-

[1] Corresponding author

known SIR epidemiological model[2]. Specifically, we modify these models to be visualized in order to allow for direct user interaction and easy viewing of "realistic" population trajectories for the different classes of zombies from *Left 4 Dead*[3], Valve Corporation's video game series that revolves around players surviving a pandemic of aggressive zombie-ism.

Figure 1: *Left 4 Dead, a cooperative zombie video game, and similar video game titles*

Sage, System for Algebra and Geometry Experimentation, was developed by William Stein with a development group at the University of Washington and mathematicians from around the world [5,12]. It was developed for the express purpose of doing mathematical computations without having to purchase, download, install, and learn the complex code confined to other gargantuan computer algebra system (CAS). *Sage* has been released on its website: http://www.sagemath.org. Furthermore, *Sage* has a client-server model which is well-adapted to the internet and allows for easy embedding and programming of *Sage* commands (called *Sage* cells) into any website [6].

In light of this, we have developed *Sage* interactive visualization modules for "Zombie population" models. In particular, these Zombie models introduce different classes, subtypes, and dynamics into the standard population model. The construction of interactive models via *Sage*, which have been uploaded both to our *Sage* server and hosted privately, is a significant tool in understanding both the mathematics of population modeling and how small changes in initial parameters – initial population, infection rate, encounter rate, etc. - can result in vastly dissimilar dynamics later. We seek to provide a comprehensive look at using *Sage* and interactive commands to create web tools that suit the user's mathematical needs.

2. The Basic Zombie Population Modelling

The model we will be analyzing and modifying is the basic zombie model ('SZR' model) of Munz, Hudea, Imad, and Smith [8]. This model was first developed by David Joyner and is detailed in his lecture notes titled "Love, War, and Zombies – Systems of Differential Equations using *Sage*"[9]. (For consistency purposes, we will use the same variable names as Joyner in order to avoid confusion and for overall ease of comprehension.) This model is extremely well-known in the epidemiology community, so we will provide only a cursory explanation of the variables and equations used. For the complete analysis, refer to [8]. This simple model considers the following three classes, Susceptible (S), Zombie (Z), and Removed (R). They are governed by their corresponding system of differential equations:

[2] Kermack, W.O. and McKendrick, A.G. A Contribution to the Mathematical Theory of Epidemics, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115 (772), pp. 700-721, 1927.
[3] https://en.wikipedia.org/wiki/Left_4_Dead

(1) $S' = \Pi - \beta SZ - \delta S$

(2) $Z' = \beta SZ + \zeta R - \alpha SZ$

(3) $R' = \delta S + \alpha SZ - \zeta R$

Solutions to the above model are numerically approximated within *Sage* using the `desolve_system_rk4` function[4], which numerically solves the initial value problem for a system of first order equations and returns a list of points (that are then plotted in a easy-to-view graph) using the 4th order Runge-Kutta method. And we have adapted the model to an interactive framework where students can access the mode and manipulate variables without any prior programming knowledge or the need to install supplementary software. The base for the code, developed within *Sage*, is provided here:

```
@interact
def zombies(s1= slider(1,50,1,20,label='initial amount of humans'),
            z1 = slider(1,50,1,5,label='initial amount of zombies'),
            r1 = slider(1,50,1,5, label='initial amount of removed (infected) humans'),
            a = slider(0,1,0.001,0.005,label='humans kill zombies rate'),
            b = slider(0,1,0.001,0.004,label='zombies kill human rate'),
            zeta = slider(0,1,0.001,0.009,label='resurrection rate'),
            d = slider(0,1,0.001,0.002,label='death rate of humans (from natural causes)'),
            timelim = slider(1,80,1,30,label='maximum time')):
    x,y,t,s,z,r=var('x,y,t,s,z,r')
    B=0.0
    P=desolve_system_rk4([B-b*s*z-d*s,b*s*z-zeta*r-a*s*z,d*s+a*s*z-zeta*r],[s,z,r],
                         ics=[0,s1,z1,r1] ,ivar=t,end_points=timelim)
    Ps = list_plot([[t,s] for t,s,z,r in P],plotjoined=True)
    Pz = list_plot([[t,z] for t,s,z,r in P],plotjoined=True,rgbcolor='red')
    Pr = list_plot([[t,r] for t,s,z,r in P],plotjoined=True,rgbcolor='black')
    show(Ps+Pz+Pr)
```

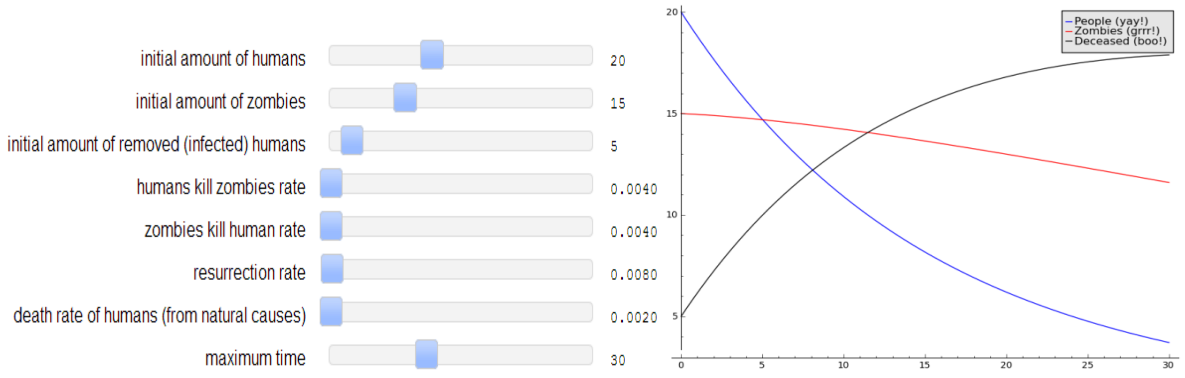
Table 1: *Sage code (web address 5)*

We shall provide an overview for the construction of this interactive *Sage* cell: first, the entire display tool is defined as a function (here called *zombies*) that takes in eight distinct parameters. Unlike more primitive data types as arguments for a function, since this is an interactive model, the arguments are labeled "sliders" that can take on a range of values as defined by the user. The very first line - `@interact` – wraps the function *zombie* is what truly transforms it into an interactive cell, allowing for those with only a cursory understanding of Python or other programming languages to edit, comprehend the code, and plot solutions. For example, instead of a static value for a, the "humans killing zombie rate, the *users* (not the programmer) can define the value with an easy-to-use sliding tool that ranges from 0 to 1. This makes the model not only easier to alter as a student, but also makes the resulting changes more intuitive and does not rely on the user needing an extensive knowledge of programming. Then, using the `desolve_system_rk4` function, the

[4] Solve numerically a system of first-order ordinary differential equations using the 4th order Runge-Kutta method. See http://doc.sagemath.org/html/en/reference/calculus/sage/calculus/desolvers.html. For the mathematical background of R unge- Kutta methods, see Griffiths, D. F. and Higham, D. J. *Numerical Methods for Ordinary Differential Equations: In itial Value Problems*, Springer, 2010.

system of differential equations is solved numerically. Finally, all trajectories are plotted for each population in question, labels are provided for ease of reading, and one graph containing all trajectories is shown to the user.

And a sample of the developed *Sage* code, all parameters that can be altered directly by the user, and the resulting trajectories are provided below[5].



initial amount of humans 20
initial amount of zombies 15
initial amount of removed (infected) humans 5
humans kill zombies rate 0.0040
zombies kill human rate 0.0040
resurrection rate 0.0080
death rate of humans (from natural causes) 0.0020
maximum time 30

Figure 2: *Basic Zombie model*

While the 4th order Runge-Kutta method is not the only method for solving this system, it is the method that has the most literature associated with it; especially for the specific topic of zombie population modeling with *Sage*, `desolve_system_rk4` is used almost solely. However, for those seeking a more mathematically rigorous method, readers can refer to the “dopri5” set integrator method in scipy.integrate.ode [11], implemented in SciPy (also callable within *Sage*), which uses the adaptive Dormand-Prince model [4] allowing for step-size control. As might be evident by the topic contents (zombies), for the purposes of this paper, we choose to focus less on mathematical robustness as much as simplicity for students, establishment of `desolve_system_rk4` in previous literature, and ease of viewing in a mobile/web platform. The subject matter (Zombie infection) is itself improbable, so exact accuracy and the Dormand-Prince adaptive step size method, while undoubtedly providing more "realistic" trajectories, was not deemed necessary for our purposes (which are mostly educational). For more information on this specific implementation of `desolve_system_rk4`, readers are encouraged to consult [9].

3. Extensions of Zombie Population Modelling

3.1 Quarantine Zombie model

As an easy extension of the base zombie infection model, Munz et. al [8] provide equations for a “Quarantine” Zombie model. This model resembles the situation above but with a new added class of infected humans that are removed from the general populace; specifically, infected humans (represented by a “*Q*” in the model) are removed and sent to private, quarantine areas, thereby ensuring they cannot infect new individuals and spread the disease. The equations governing this scenario are:

$$(4)\quad S' = \Pi - \beta SZ - \delta S$$

[5] http://matrix.skku.ac.kr/2014-Zombie-Model/SKKUMath_files/standard_graph.html (Use Chrome and Firefox browsers.)

(5) $I' = \beta SZ - \rho I - \delta I - \kappa I$
(6) $Z' = \rho I + \zeta R - \alpha SZ - \sigma Z$
(7) $R' = \delta S + \delta I - \alpha SZ - \zeta R + \gamma Q$
(8) $Q' = \kappa I + \alpha Z - \gamma Q$

We, naturally, adapted the equations to *Sage* in a similar vein as above. The *Sage* code and an example of "possible" trajectory, as well as a more in-depth discussion of the model, are given below and on our mobile site[6].

```
@interact
def zombies(human_initial= slider(1,50,1,20,label='initial amount of humans'),
            zombies_initial = slider(1,50,1,5,label='initial amount of zombies'),
            a = slider(0,1,0.001,0.005,label='human reproduction rate'),
            beta1 = slider(0,1,0.001,0.004,label='infection rate'),
            rho1 = slider(0,1,0.001,0.009,label='infect to zombie rate'),
            delta1 = slider(0,1,0.001,0.002,label='death rate (from natural causes)'),
            kappa1 = slider(0,1,0.001,0.002,label='infected human --> quarantine rate'),
            zeta1= slider(0,1,0.001,0.002,label='dead people --> reanimate as zombie rate'),
            alpha1=slider(0,1,0.001,0.002,label='zombies attack humans --> zombies die rate'),
            sigma1=slider(0,1,0.001,0.002, label='zombie quarantine rate'),
            gamma1 =slider(0,1,0.001, 0.002, label='escape rate of quarantined individuals (all killed)'),
            timelim = slider(1,80,1,30,label='maximum time')):

    x,y,t=var('x y t')
    B=0
    t,s,z,r,j,q = var("t,s,z,r,j,q")
    B=0.0
    P = desolve_system_rk4([(a*s)-(beta1*s*z)-(delta1*s),
                            (beta1*s*z)-(rho1*j)-(delta1*j)-(kappa1*j),
                            (rho1*j)+(zeta1*r)-(alpha1*s*z)-(sigma1*z),
                            (delta1*s)+(delta1*j)+(alpha1*s*z)-(zeta1*r)+(gamma1*q),
                            (kappa1*j)+(sigma1*z)-(gamma1*q)],[s,j,z,r,q],
                           ics=[0,human_initial,0,zombies_initial,0,0],
                           ivar=t,end_points=timelim)
    #Pquar=desolve_system_rk4([B-b*s*z-rho*j-d*j-kappa*j, kappa*j+a*z-gammadie*q],
    #[s,z,j,q], ics=[0,10,10,5,5], ivar=t, end_points=timelim)

    Ps = list_plot([[t,s] for t,s,j,z,r,q in P], plotjoined=True, legend_label='People (yay!)')
    Pz = list_plot([[t,z] for t,s,j,z,r,q in P], plotjoined=True, rgbcolor='red',
               legend_label='Zombies (grrr!)')
    Pr = list_plot([[t,r] for t,s,j,z,r,q in P], plotjoined=True, rgbcolor='black',
               legend_label='Removed')
    Pj = list_plot([[t,j] for t,s,j,z,r,q in P], plotjoined=True, rgbcolor='green',
               legend_label=' Latent Infected')
    Pq = list_plot([[t,q] for t,s,j,z,r,q in P], plotjoined=True, rgbcolor='purple',
               legend_label='Quarantined')
    show(Ps+Pz+Pr+Pj+Pq)
```

Table 2: *Sage code (web address 6)*

[6] http://matrix.skku.ac.kr/2014-Zombie-Model/SKKUMath_files/latent_graph.html

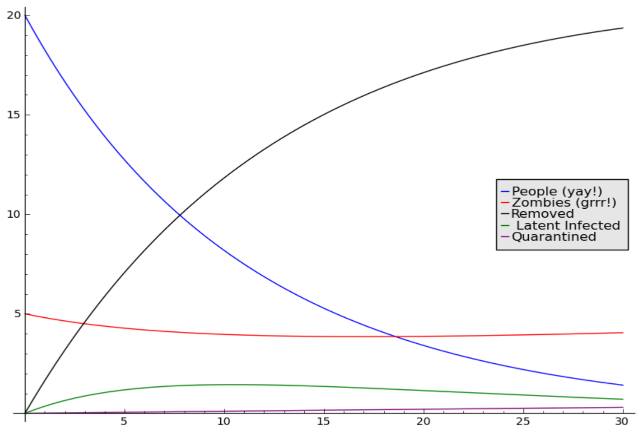


Figure 3: *Zombie model with Quarantine*

3.2 Adding the "Left 4 Dead" zombies to the model

Unlike previous portrayals of the undead as roughly the same in physical appearance, aggressiveness, and attack capability across individuals, *Left 4 Dead*, a popular video game created by gaming powerhouse Valve Corporation, is unique in creating two distinct subfamilies of zombies: the *Common Infected*[7] and *Special Infected*. The most common zombie group the player encounters is, as anticipated, the *Common Infected*. However, by some mysterious biological process, a small percentage of humans experience a more physiologically impressive transformation into *Special Infected*, with heightened strength, intelligence, and particular anatomical nuances. Approximate percentages of each infection strain are given below in a pie chart found on a poster during game-play:

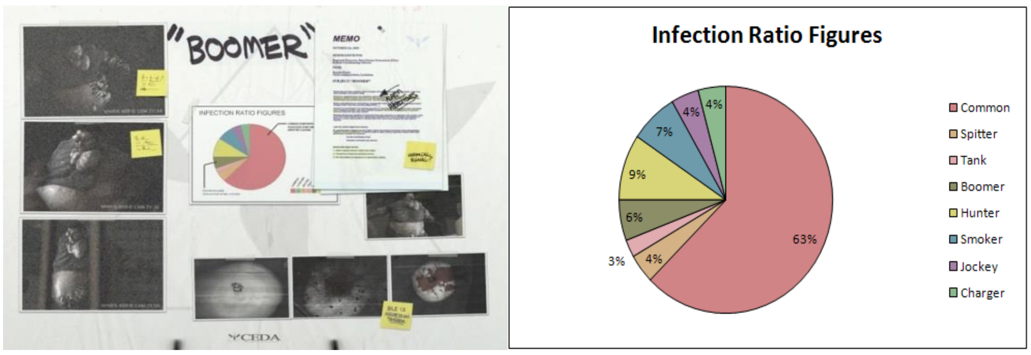


Figure 4: *The ratios of the stands of Green Flu infections, in-game and reformatted for easy viewing*[8]

The *Common Infected* zombies are *Left 4 Dead*'s version the archetypal "zombie" that has remained a fixture in nearly all forms of popular media. While not as slow-moving and shambling as other depictions (such as George Romero's *Night of the Living Dead*[9] zombies), they are still comparatively weak, lack most higher-order motor control functions, and are "as easy to kill as any normal human is" ("Common Infected"). The power of the *Common Infected* lies in their ability to

[7] http://left4dead.wikia.com/wiki/Common_Infected

[8] http://left4dead.wikia.com/wiki/Green_Flu
[9] https://en.wikipedia.org/wiki/Night_of_the_Living_Dead

attack in large, synchronized groups, known as "*The Horde*." Ajraldi, Pittavino, and Venturino [1] provide a method for modeling herd behavior in two interacting species via a square root term- that is, since our population is spread over a two-dimensional domain, the density square root will account for the individuals lying along the edge of their respective region. Thus, replacing the general **Z** equation (1) with the more specialized Horde zombies, the equations governing our model become:

$$(9)\ \ S' = \Pi - \beta_H S\sqrt{H} - \omega S$$

$$(10)\ \ R' = \omega S + \alpha_H S\sqrt{H} - \zeta_H R$$

$$(11)\ \ H' = \beta_H S\sqrt{H} + \zeta_H R - \alpha_H S\sqrt{H}$$

3.3 "General" Dynamics: The Hunter and The Charger[10]

These two zombies, distinct in respective traits and powers, will be modeled from roughly the same differential equation; for ease of organization, they are grouped together by their *mathematical*, not anatomical, relatedness. The *Hunter* is a *Special Infected* notable for its increased speed, agility, and relative absence of conspicuous physical mutations. Their equation, hence, is merely a special case of the general Zombie class from the model in [1], and accounting for all respective rates:

$$(12)\ \ N' = \beta_N SN + \zeta_N R - \alpha_N SN$$

The *Charger* is a large *Special Infected* that also has increased speed abilities but uses them to instead charge at a group of Survivors, sending them flying through the air. Following the equation setup above, we have:

$$(13)\ \ C' = \beta_C SN + \zeta_C R - \alpha_C SN$$

3.4 "In Tandem" Dynamics: The Jockey and The Boomer[11]

The *Jockey* and the *Boomer*, denoted by **J** and **B**, respectively, display "in tandem" population dynamics - that is, the effectiveness of other zombies increases given their presence in the system. Specifically, the *Boomer* attacks the human players by vomiting on them. This vomit attracts the *Common Infected* (Horde, **H**) zombies, and hence the likelihood of successful *Horde* attacks is increased with the presence of Boomers. The related equations are:

$$(14)\ \ B' = \beta_B SB + \zeta_B R - \alpha_B SB$$

$$(15)\ \ J' = \beta_J SJ + \zeta_J R - \alpha_J SJ$$

$$(16)\ \ H' = \beta_H S\sqrt{H}\left(\frac{B}{S+1}\right) + \zeta_H R - \alpha_H S\sqrt{H}$$

[10] http://left4dead.wikia.com/wiki/The_Hunter http://left4dead.wikia.com/wiki/The_Charger

[11] http://left4dead.wikia.com/wiki/The_Jockey http://left4dead.wikia.com/wiki/The_Boomer

3.5 "Survivor Presence Proportional" Dynamics: The Smoker, The Tank, and The Witch[12]

The *Smoker*, the *Tank*, and the *Witch*, (**M**, **K**, and **W**, respectively) are characterized by varying success rates based on the number of human survivors present in the system and their interactions with these survivors. Markedly stronger and more resistant than the aforementioned zombie classes, their corresponding equations are:

$$(17)\ \ K' - \beta_K SK + \zeta_K R - \alpha_K SK\left(\frac{S}{K+1}\right)$$

$$(18)\ \ M' - \beta_M SM + \zeta_M R - \alpha_M SM\left(\frac{S}{M+1}\right)$$

$$(19)\ \ W' = \beta_W SW\left(\frac{1}{1+\eta W}\right) + \zeta_W R - \alpha_W SW$$

By visiting website http://matrix.skku.ac.kr/2014-Zombie-Model/main.htm, you can view the open-source *Sage* code and these mobile-tailored population trajectories in a convenient and interactive format. The following *Sage* code, while extensive, should be fairly easy to follow by the reader at this point. All zombies are added continuously, and attack/defense rates were calculated by in-game values are mentioned above.

```
@interact
def zombies(human_initial= slider(1,300,1,50,label='initial amount of humans'),
            zombies_initial_h = slider(1,50,1,5,label='initial amount of common infected zombies'),
            zombies_initial_b = slider(1,50,1,2,label="initial amount of The Boomer"),
            zombies_initial_m = slider(1,50,1,2,label='initial amount of The Smoker'),
            zombies_initial_n=slider(1,50,1,2,label='initial amount of the Hunter'),
            zombies_initial_k = slider(1,50,1,1,label='initial amount of the Tank'),
            zombies_initial_w = slider(1,50,1,1,label='initial amount of the Witch'),
            mm=slider(0,1,0.001,0.2, label='The Witch elusiveness rate'),
            zombies_initial_c = slider(1,50,1,1,label='initial amount of the Charger'),
            zombies_initial_j = slider(1,50,1,1,label='initial amount of the Jockey'),
            a2 = slider(0,1,0.01,0.69,label='human birth rate'),
            deathtimes= slider(0,1,0.001,0.005, label='human natural death rate'),
            timelim = slider(0,400,10,60,label='maximum time')):

    alpha_common = 0.09
    alpha_boomer=0.09
    alpha_smoker=0.083544
    alpha_hunter=0.083544
    alpha_tank=0.005
    alpha_witch=.067404
    alpha_charger=0.076012
    alpha_jockey=0.08193

    #infection rates from Removed population
    zeta_common = 0.063
    zeta_boomer = 0.006
    zeta_smoker = 0.007
    zeta_hunter = 0.009
    zeta_tank = 0.003
    zeta_witch = 0.004
    zeta_charger = 0.004
```

[12] http://left4dead.wikia.com/wiki/The_Smoker http://left4dead.wikia.com/wiki/The_Tank http://left4dead.wikia.com/wiki/Witch

```
zeta_jockey = 0.004

#attack rates from damage done by primary attack
beta_common = 0.005
beta_witch=0.09
beta_boomer=.007157
beta_smoker=0.010897
beta_hunter=0.012335
beta_tank=0.046853
beta_charger=0.012335
beta_jockey=0.007157

t,s,h,b,m,r,n,k,w,j = var("t,s,h,b,m,r,n,k,w,j")
P = desolve_system_rk4([(a2)-(beta_common*s*sqrt(h)*(b/(s+1)))-(beta_boomer*s*b)
                        -(beta_smoker*s*m*(1/((s^2)+1)))-(beta_hunter*s*n)-(beta_tank*s*k)
                        -(beta_witch*s*w*(1/(1+mm*w)))-(beta_jockey*s*j)-(deathtimes*s),
                        (beta_common*s*sqrt(h))+(zeta_common*r)-(alpha_common*s*sqrt(h)),
                        (beta_boomer*s*b)+(zeta_boomer*r)-(alpha_boomer*b*s),
                        (deathtimes*s)+(alpha_common*s*sqrt(h))+(alpha_boomer*s*b)
                        +(alpha_smoker*s*m)+(alpha_hunter*s*n)+(alpha_tank*k*s*(s/(k+1)))
                        +(alpha_witch*w*s)-(zeta_common*r)-(zeta_boomer*r)-(zeta_smoker*r)
                        -(zeta_hunter*r)-(zeta_tank*r)-(zeta_witch*r)-(zeta_jockey*r),
                        (beta_smoker*s*m)+(zeta_smoker*r)-(alpha_smoker*m*s),
                        (beta_hunter*s*n)+(zeta_hunter*r)-(alpha_hunter*n*s),
                        (beta_tank*s*k)+(zeta_tank*r)-(alpha_tank*k*s*(s/(k+1))),
                        (beta_witch*s*w*(1/(1+mm*w)))+(zeta_witch*r)-(alpha_witch*w*s),
                        (beta_jockey*s*j)+(zeta_jockey*r)-(alpha_jockey*j*s)],[s,h,b,r,m,n,k,w,j],
 ics=[0,human_initial,zombies_initial_h, zombies_initial_b,0,
      zombies_initial_m, zombies_initial_n, zombies_initial_k, zombies_initial_w,
      zombies_initial_j],ivar=t,end_points=timelim)

#survivors
Ps = list_plot([[t,s] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,
               legend_label='People (yay!)')
#common infected
Ph = list_plot([[t,h] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='red',
               legend_label='Common Infected')
#the Boomer
Pb = list_plot([[t,b] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='green',
               legend_label='The Boomer')
#the Smoker
Pm = list_plot([[t,m] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='purple',
               legend_label='The Smoker')
#the Hunter
Pn = list_plot([[t,n] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='orange',
               legend_label='The Hunter')
#the Tank
Pk = list_plot([[t,k] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='black',
               legend_label='The Tank')
#the Witch
Pw = list_plot([[t,w] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='pink',
               legend_label='The Witch')
#the Jockey
Pj = list_plot([[t,j] for t,s,h,b,r,m,n,k,w,j in P],plotjoined=True,rgbcolor='yellow',
               legend_label='The Jockey')

show(Ps+Ph+Pb+Pm+Pw+Pj+Pn+Pk)
```

Table 3: *Sage code*

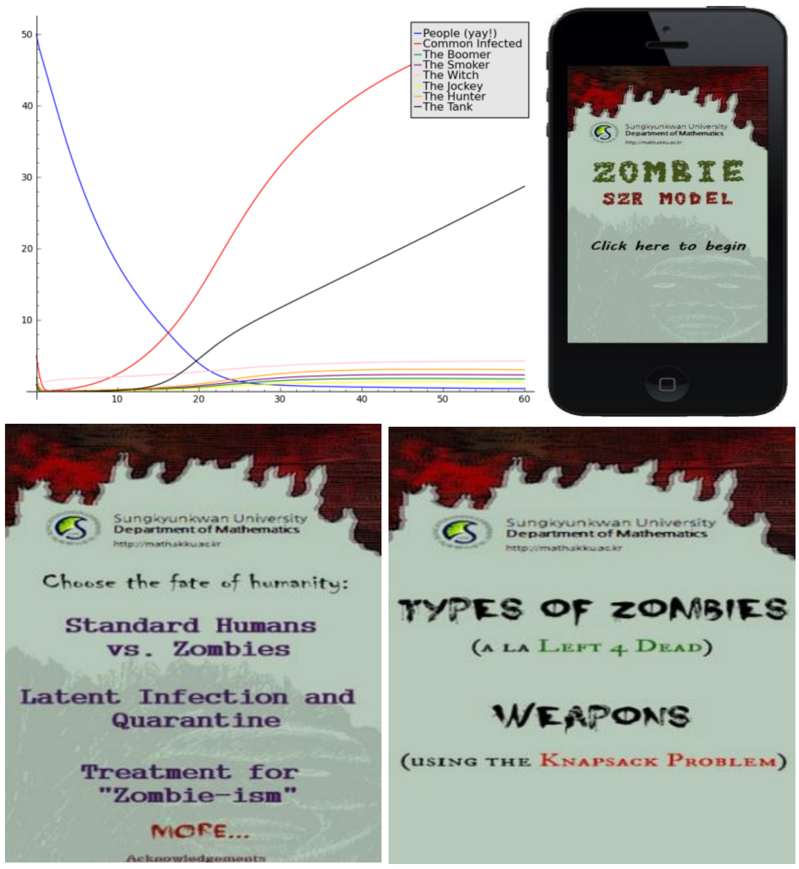

People (yay!)
Common Infected
The Boomer
The Smoker
The Witch
The Jockey
The Hunter
The Tank
Sungkyunkwan University Department of Mathematics
ZOMBIE
SZR MODEL
Click here to begin
Choose the fate of humanity:
Standard Humans vs. Zombies
Latent Infection and Quarantine
Treatment for "Zombie-ism"
MORE...
Acknowledgements
TYPES OF ZOMBIES
(A LA LEFT 4 DEAD)
WEAPONS
(USING THE KNAPSACK PROBLEM)

Figure 5: *The Zombie SZR model and its various links and applications*

These population trajectories, as well as in-depth zombie profiles, mathematical analyses, and other interesting "apocalyptic" math resources and fun tidbits, are available through our mobile-adapted website above.

4. Conclusions

Through this research, we have introduced our easy-to-view Zombie Population Model as a pedagogical Sage Tool. Readers need resources that appeal to their extracurricular interests and bolster academic curiosity by showing mathematics in a way previously unexplored; namely, how mathematics relates to pop culture in the form of zombies and video games [3]. In general, educators are finding academically stimulating ways to incorporate mathematics learning into video games and vice versa. Being able to connect a reader's extracurricular interests to an academic area he or she previously struggled in can not only increase interest in the subject as a whole, but encourage the student to preserve through difficult moments and, hence, find satisfaction in their work. Technological tools that can combine a reader's independent interests with relevant mathematically-driven content have been shown to improve student learning in various areas [7]. Bolstering interest in population models and differential equations can, in turn, bolster interest in all areas of mathematics [10]. Being able to present these tools in an easy-to-view, open-source, free format online furthers this goal and gives the power of mathematical manipulation directly to students of various math levels and disciplines.

5. References

[1] Ajraldi, V., Pittavino, M. and Venturino, E., Modeling herd behavior in population systems, *Nonlinear Analysis-Real World Applications*, 12(4), pp.2319-2338, 2011.

[2] Daley, D.J. and Gani, J., *Epidemic Modelling: An Introduction*, Cambridge University Press, 2005.

[3] Devlin, K., *Mathematics Education for a New Era: Video Games as a Medium for Learning*, A K Peters, 2011.

[4] Dormand, J. R. and Prince, P. J., A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics*, 6(1), pp. 19–26, 1980.

[5] Joyner, D. and Stein, W., *Sage Tutorial: www.sagemath.org*, Create Space Independent Publishing Platform, 2008.

[6] Lee, S., Kim, K. and Sun, S., Modeling of Mobile Sage and Graphing Calculator, *Journal of Modern Education Review*, 3(12), pp.243-250, 2013.

[7] McCab, D. B. and Meuter, M. L., A student view of technology in the classroom: Does it enhance the seven principles of good practice in undergraduate education? *Journal of Marketing Education*, 33(2), pp.149-159, 2011.

[8] Munz, P., Hudea, I., Imad, J. and Smith, R., When Zombies Attack! Mathematical Modelling of an Outbreak of Zombie Infection, *Infectious Disease Modelling Research Progress*, pp.133-150, 2009. Retrieved from http://mysite.science.uottawa.ca/rsmith43/Zombies.pdf.

[9] Joyner, D., *Love, War, and Zombies – Systems of Differential Equations using Sage* [Powerpoint slides]. Retrieved Oct 1, 2015, from: http://www.sagemath.org/files/love-war-zombies_slides.pdf.

[10] Maron, M., *Modelling Populations: From Malthus to the Threshold of Artificial Life*. Retrieved April 17, 2015, from http://brainoff.com/easy/report.pdf. (2003, December 9).

[11] Peterson, P., Travers, J. and Virtanen, P., *Scipy.integrate.ode*. (2015, Oct 24). Retrieved Oct 25, 2015, from http://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.integrate.ode.html.

[12] SageMath, *Mathematical Software System - Sage*. (2007, June 7). Retrieved May 27, 2015, from http://www.sagemath.org/.