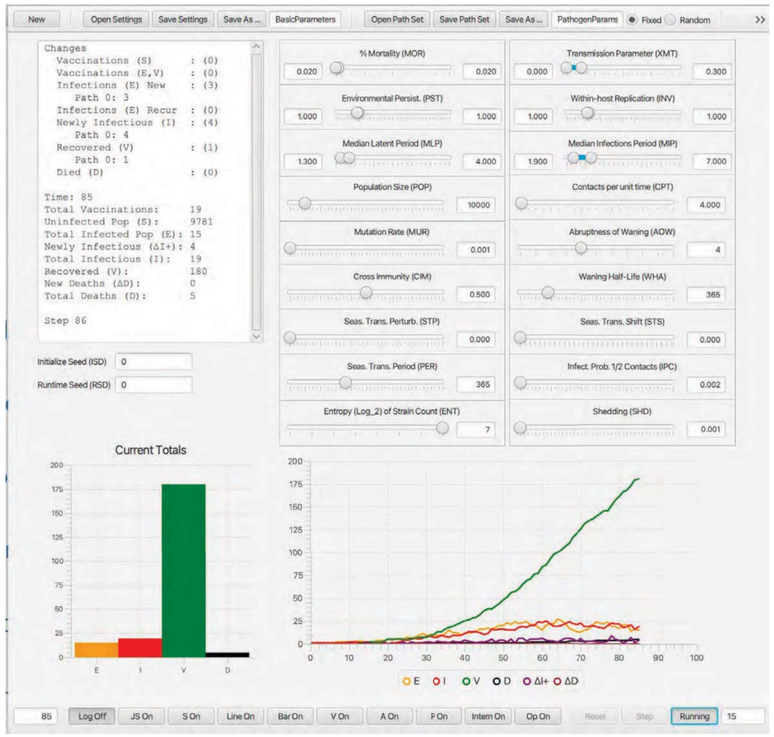
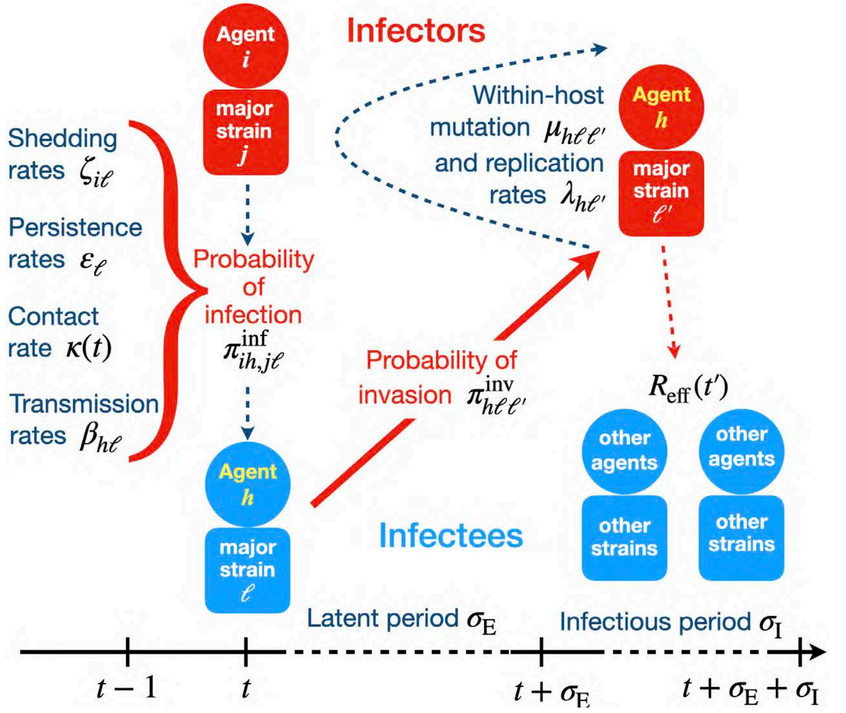
Adaptive vaccination may be needed to extirpate COVID-19

Results from a runtime-alterable strain-drift and waning-immunity model

Wayne Getz, UC Berkeley, Numerus Inc.,
Richard Salter, Oberlin College, Numerus Inc.
Ludovica Luisa Vissat, UC Berkeley
James Koopman, U of Michigan
Carl Simon, U of Michigan



Overview



- **J-RAMP:** Javascript runtime-alterable-model platform (Introduction: more in 2nd part of talk by Richard Salter)
- **SEIVD-IBM:** Susceptible, Exposed, Infected, Immune/Vaccination, Dead — Individual-Based Model
- **Parameters:** COVID-19, J-RAMP Dashboard, and data windows
- **Simulations:** sensitivity to ***waning*** rates and ***adaptive contact*** rates, single and multistrain vaccination
- **Vaccination:** last two years of a 3-year rollout; rates and single and multistrain vaccine valencies
- **Some sobering lessons:** need for timely strain monitoring; importance of adaptive multivalent vaccines in rollout

J-RAMP Java runtime-alterable model platform

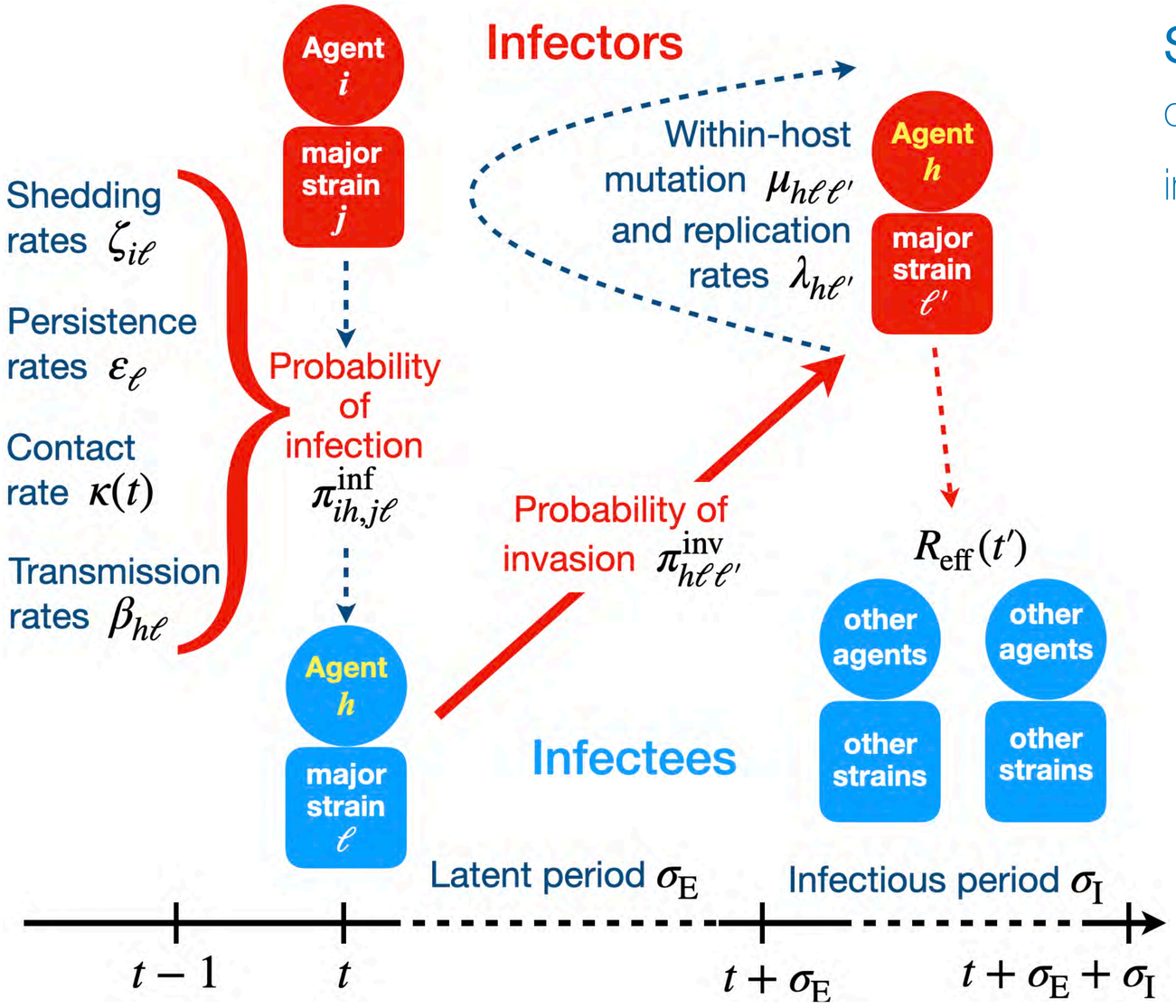
1. **Parameter values:** selected at start of simulation or changed midstream

2. **Set of runtime alternative modules (RAMs; functions):** added *de novo* at start of new simulation (open ended facility); or selection among alternatives at start of simulation or changed midstream

3. **Scripting windows:** API for remote or onboard run drivers; very simple block-box programming language (BPL) window using "airport codes"; and JavaScript interpreter window enhanced with BPL

4. **Integration with R** (or other options such as Python)**:** e.g., our **J-RAMP** acts as an **R routine** in an **R script** (e.g., set up in **R-Studio**)—essentially **J-RAMP** becomes a "***virtual package***" to the controlling R logic.

SEIVD-IBM (v= recovered/vaccinated, dead; individual-base model)



Shedding and transmission rates: modified by cross immunity between strains *m* and *j* (c_{mj}) and waning immunity in agent *i* wrt strain *m* (ω_{im}) with functions:

$$\phi_{ij}(t) = \prod_{m=0}^{2^J-1} \left(1 - c_{mj}\omega_{im}(t)\right)$$

Adaptive contact rates: depends a prevalence influence parameter $p_{\text{I}}^{\text{half}}$ that reduces the contact rate to half when prevalence rises to this parameter value:

$$\kappa(t) = \begin{cases} \dfrac{\kappa_0}{1+\left(\frac{N_{\text{I}}(t)}{N_0 - N_{\text{D}}(t)} \Big/ p_{\text{I}}^{\text{half}}\right)^2} & \text{when } p_{\text{I}}^{\text{half}} > 0 \\ \kappa_0 & \text{when } p_{\text{I}}^{\text{half}} = 0 \end{cases}$$

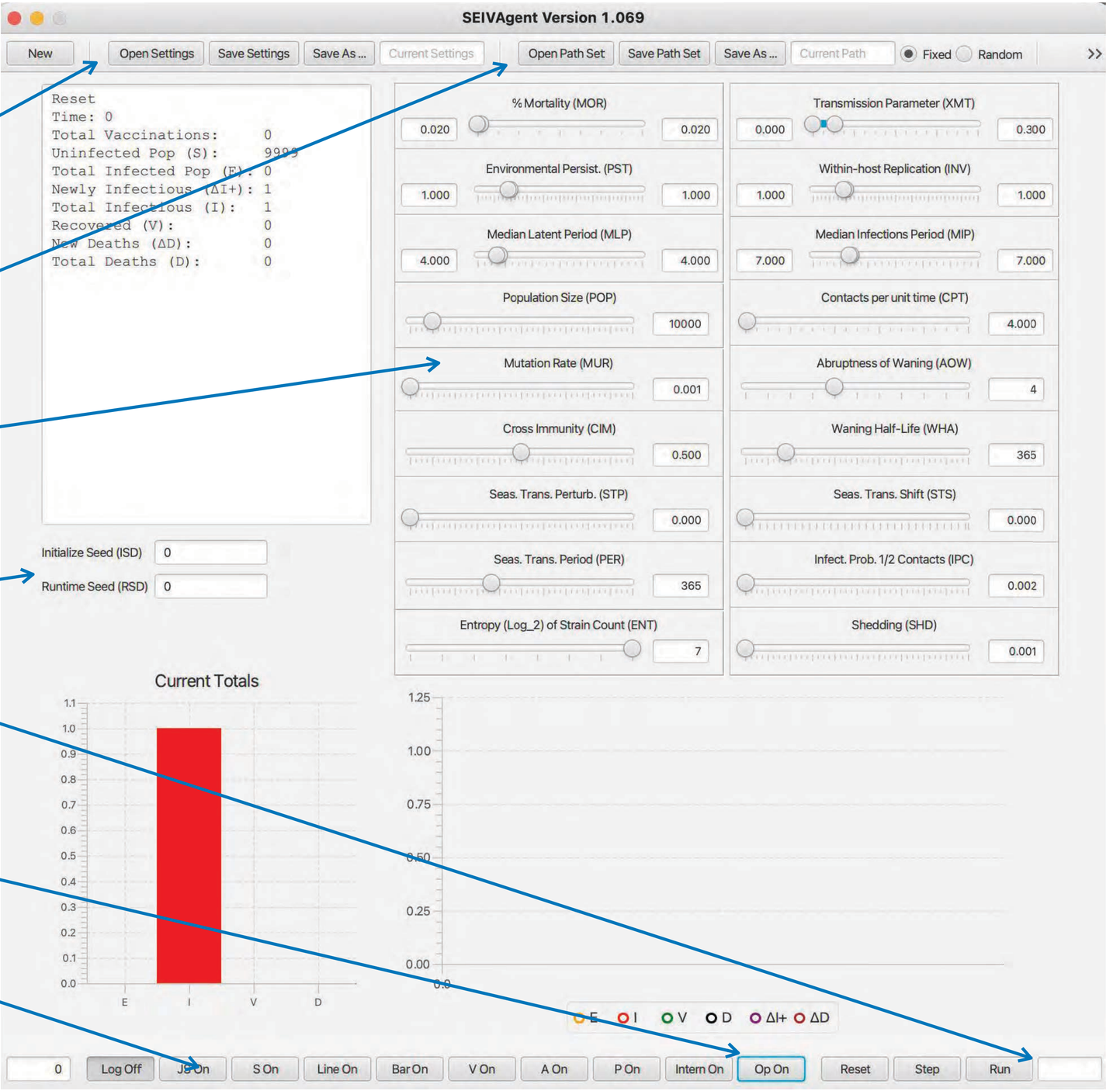
J-RAMP SEIVD-IBM

Extremely easy to use

Open “settings” and “path set”
or
set parameter values de novo

Insert pseudo random number “seeds”
and
Insert run length and press “run”

Make runtime alterations to functions
or
add JavaScipt drivers



J-RAMP SEIVD-IBM

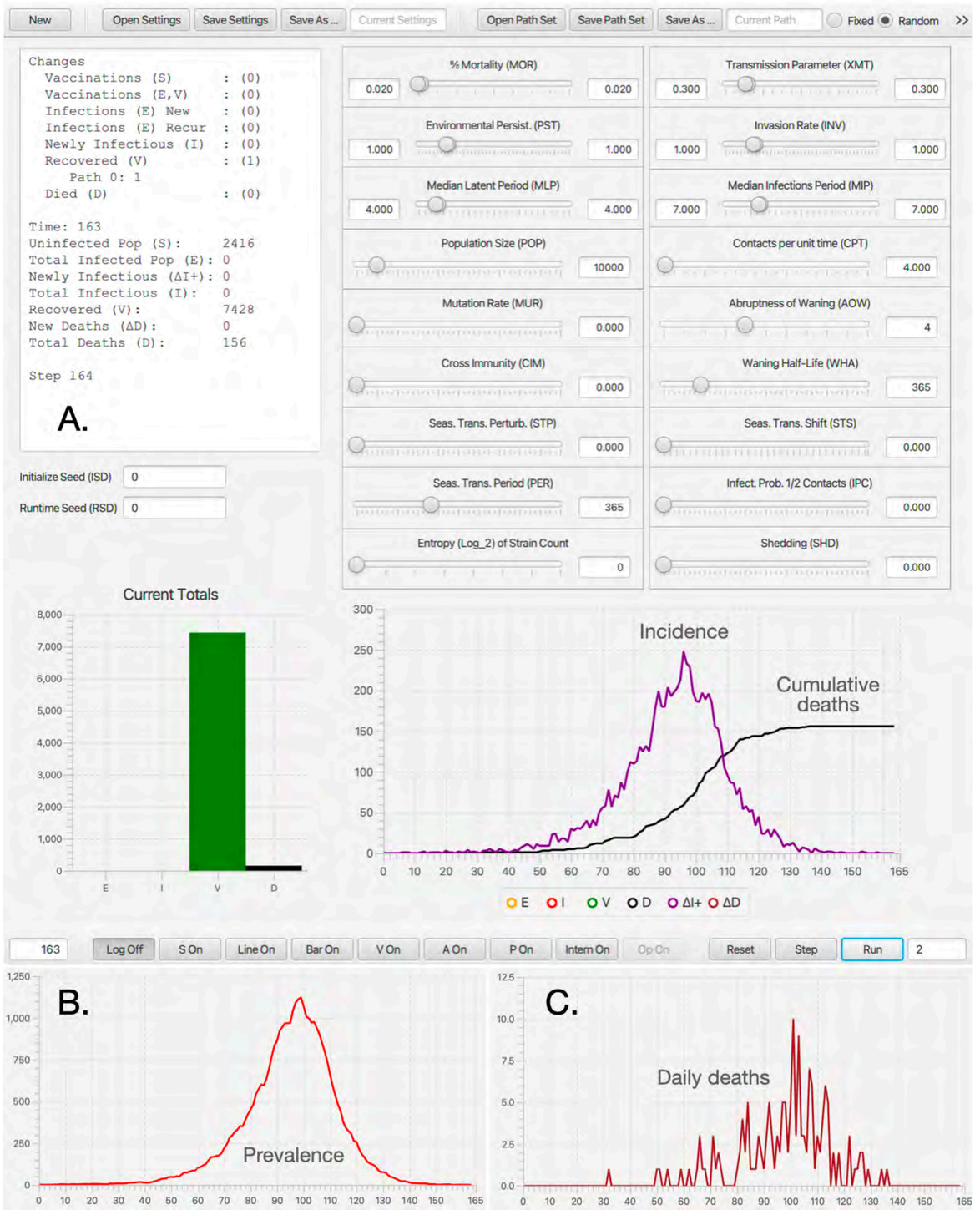
basic COVID-19 run

Table 1: Parameter values used to simulate single and multistrain outbreaks

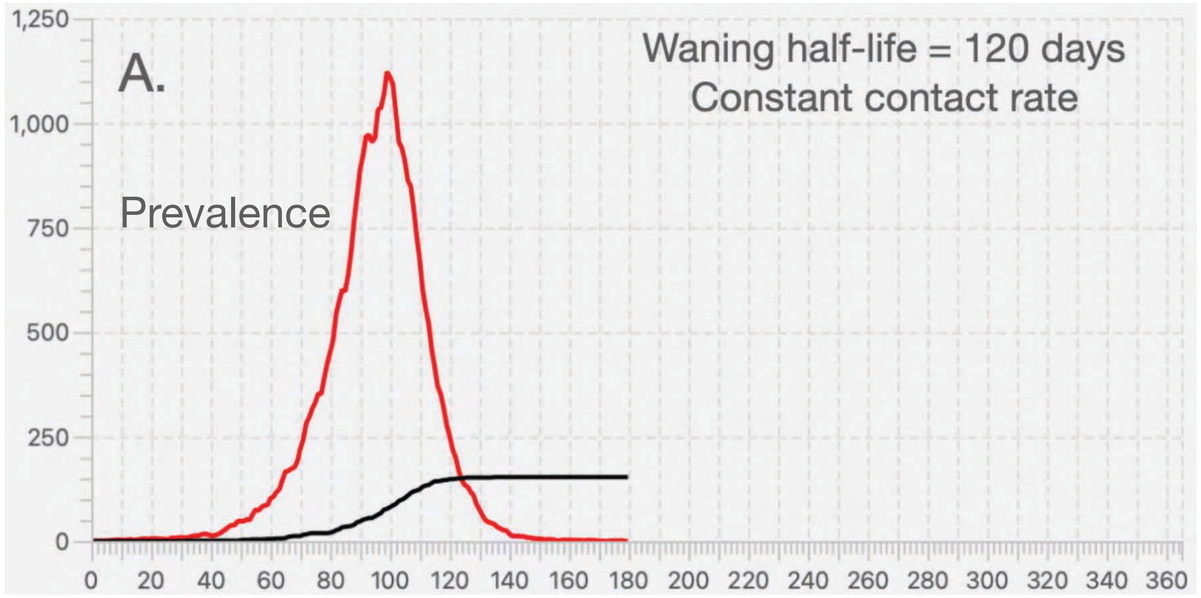
Parameter	Symbol	Value	Source/Comment
Single strain simulations			
Time unit	t	daily	Goldilocks value*
Nominal pop size	N_0	10^4 to 10^5	Goldilocks value‡
Basic contact rate	κ_0	4 per day	effective contacts†
Transmission	β	0.3	ensures $R_0 \approx 3^+$
Latent Period	σ_{E}	4 days	median time in E¶
Infectious period	σ_{I}	7 days	median time in I§
Immunity half-life	t^{half}	1/4 to 1 year	per run specs.‖
Disease-induced mort.**	p_α	2% of cases	mortality rate is $\alpha^{\#}$
Adaptive contact param.*†	$p_{\mathrm{I}}^{\mathrm{half}}$	0, 0.002, 0.05	reduces κ from κ_0
Seasonal fluctuation param.	δ	0	seasons ignored*+
Multi strain simulations (single strain parameter values for all strains if not mentioned below)			
Mutation rate*‡	μ	$0.001^{*\#}$	See Eq. A.16
Strain number	$j = 0, ..., 2^J - 1$	J is 1 to 8	i.e., 2 to 1024 strains
Cross immunity	c_{jm}	0 & 0.5	See Eq. 1
Pathogen shedding	$\bar{\zeta}_{jm}$	$0.001^{*\#}$	See Eq. A.12
Environmental persistence	$\bar{\eta}_j$	1 for all j	See Eq. A.13
Transmission	$\bar{\beta}_j$	[0,0.475]	per run specs.
Within-host replication rate	λ_j	1 for all j	See Eq. A.16
Disease-induced mort.**	p_{α_j}	[0,0.05]	per run specs.

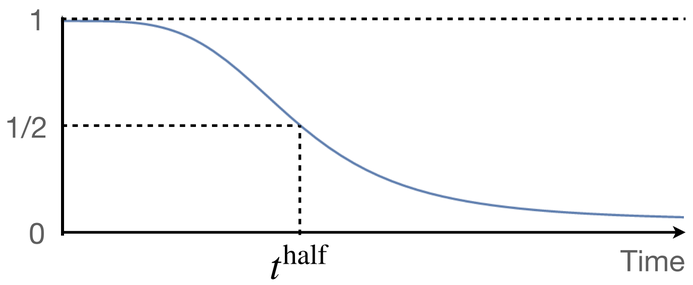
$$\rho \equiv 1/\sigma_{\mathrm{I}} = 1/7 \qquad \gamma \equiv 1/\sigma_{\mathrm{E}} = 1/4$$

$$R_0 = \frac{4 \times 0.3}{1/4 + 1/7} = 3.05$$

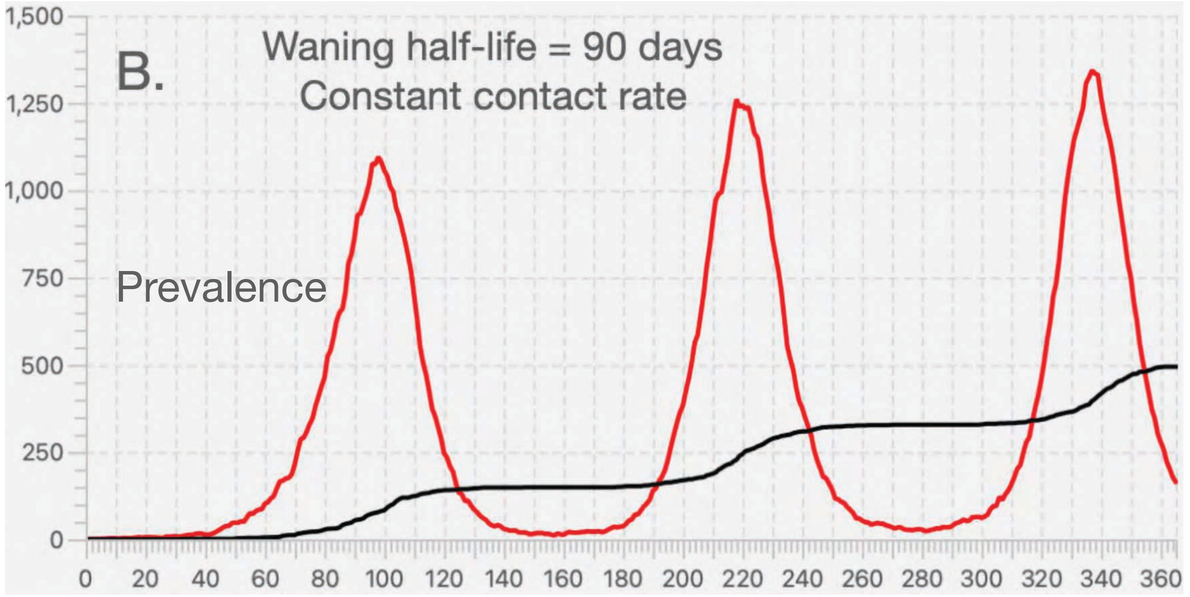


J-RAMP SEIVD-IBM Waning immunity

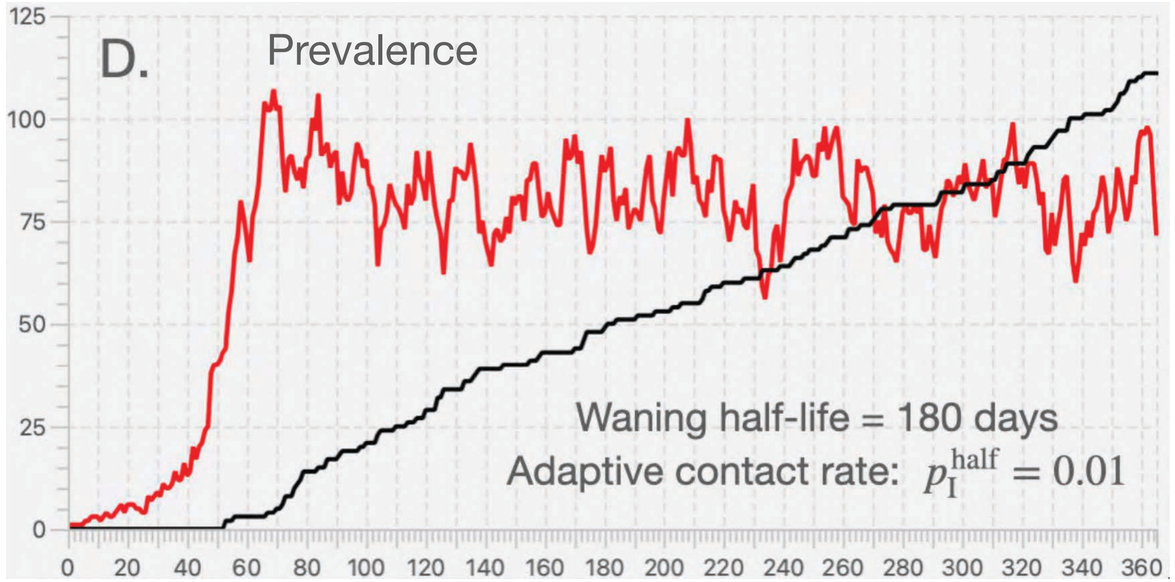


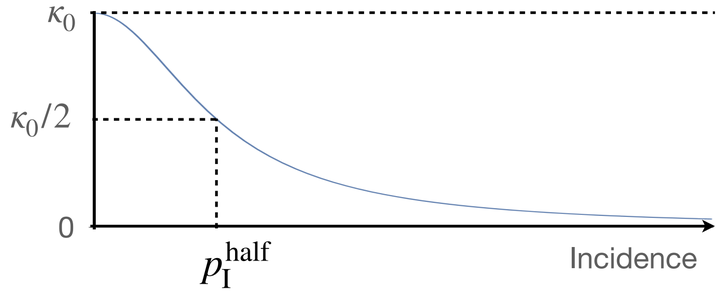


Immunity declines from 1 at recovery time to 0 over time like a *mirror-image logistic* and has a value of 1/2 after t^{half} days

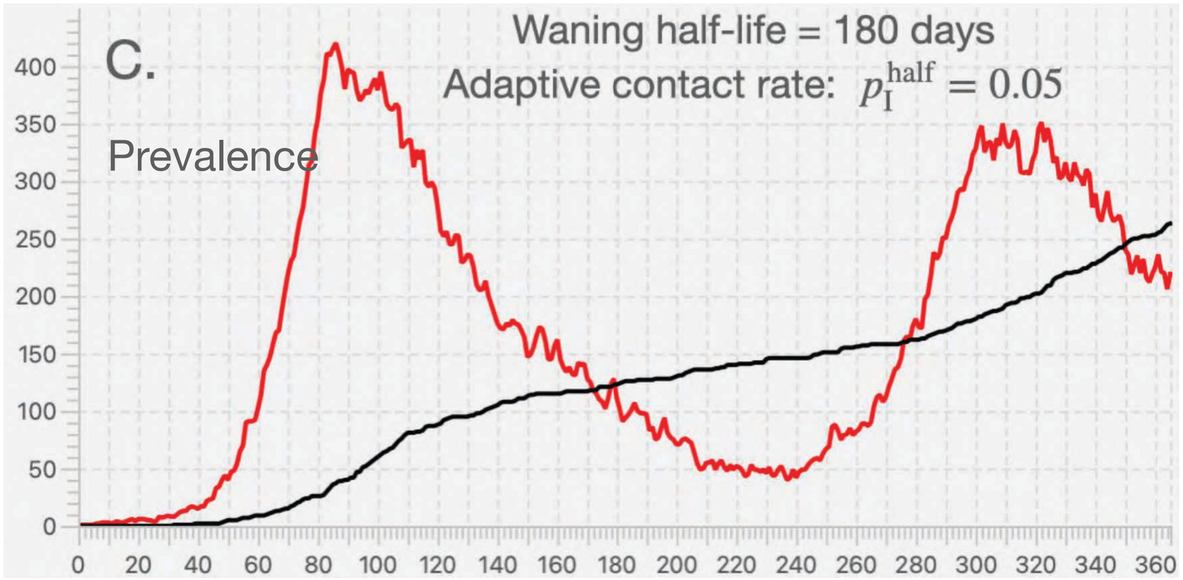


J-RAMP SEIVD-IBM adaptive contact



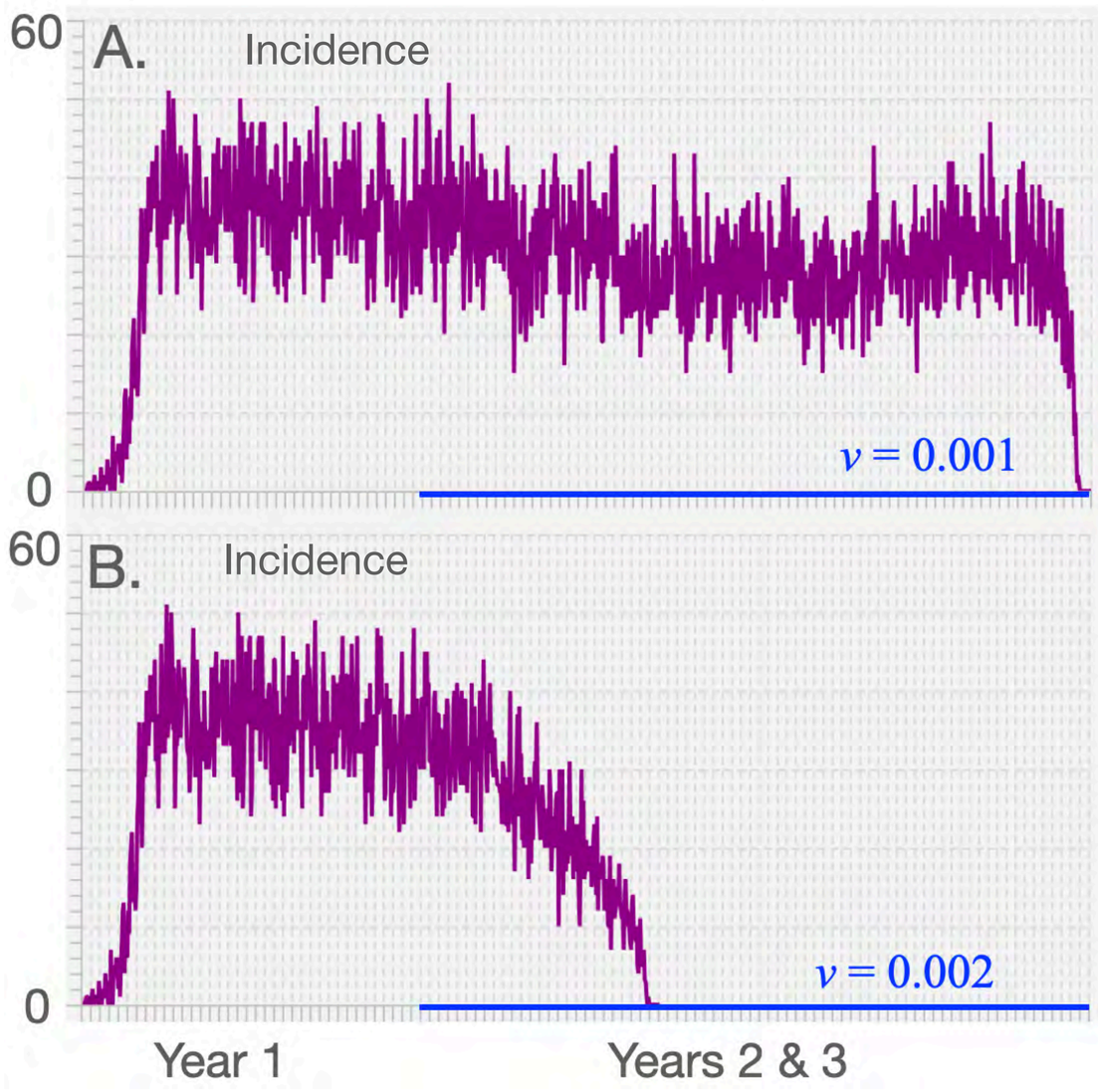


Adaptive contact rate $\kappa(t)$ declines from its constant value κ_0 when prevalence is 0 like a *mirror-image logistic* to $\kappa_0/2$ when prevalence is equal to $p_{\mathrm{I}}^{\mathrm{half}}$



J-RAMP SEIVD-IBM vaccination

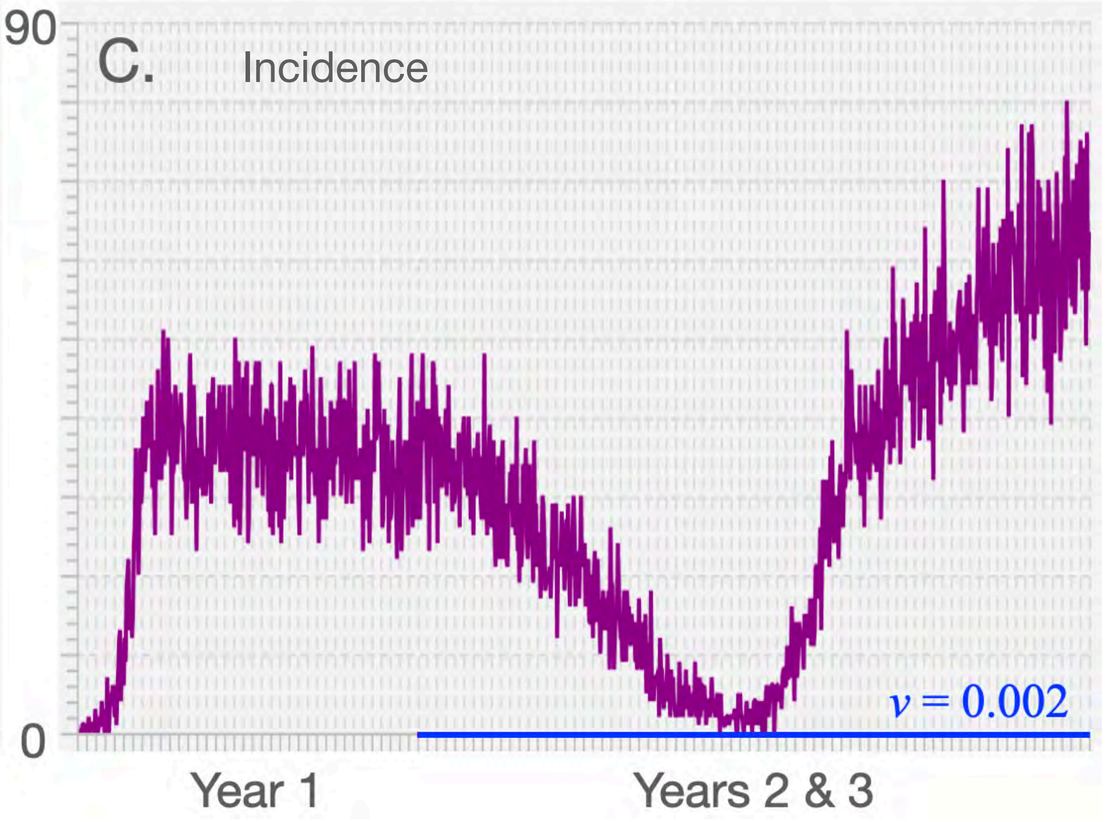
$$N = 100,000,\ p_{\mathrm{I}}^{\mathrm{half}} = 0.002,\ \text{and}\ t^{\mathrm{half}} = 365$$



Daily cases

A & B: vaccinate those not previously vaccinated

C: vaccinate those not previously vaccinated, but not those that have previously contracted COVID-19



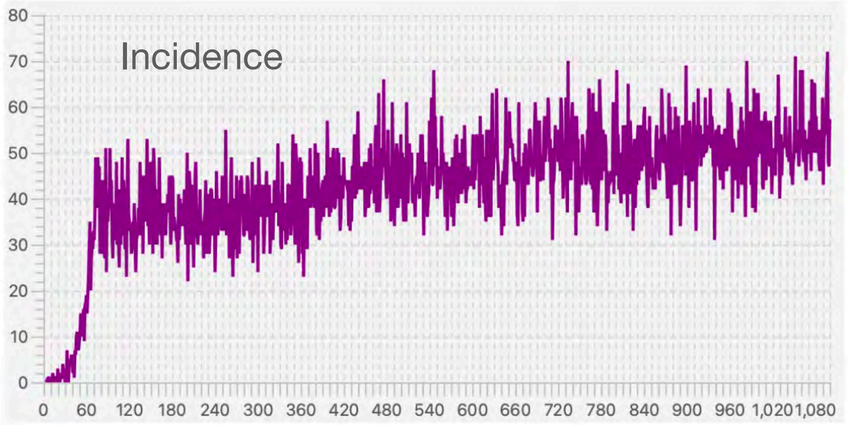
J-RAMP SEIVD IBM

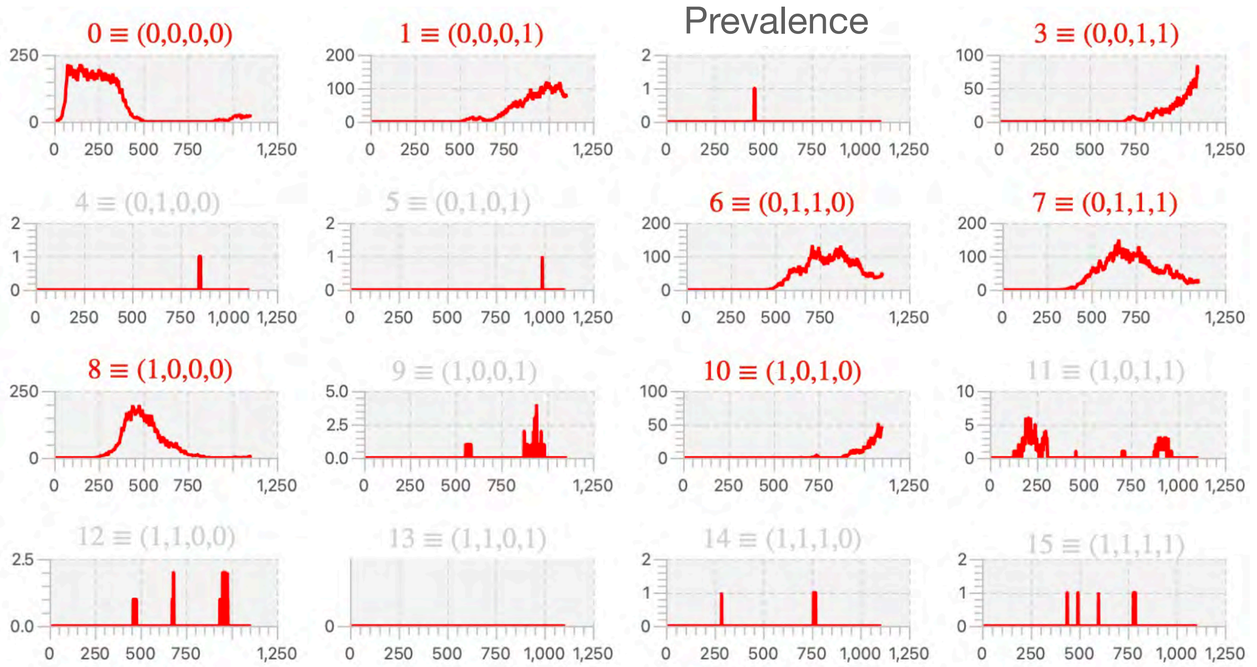
multistrain

Strains represented by strings of binary numbers (entropy of strain system is *J*: up to *J*=7 which is 128 strains)

A. cross immunity = 0.0

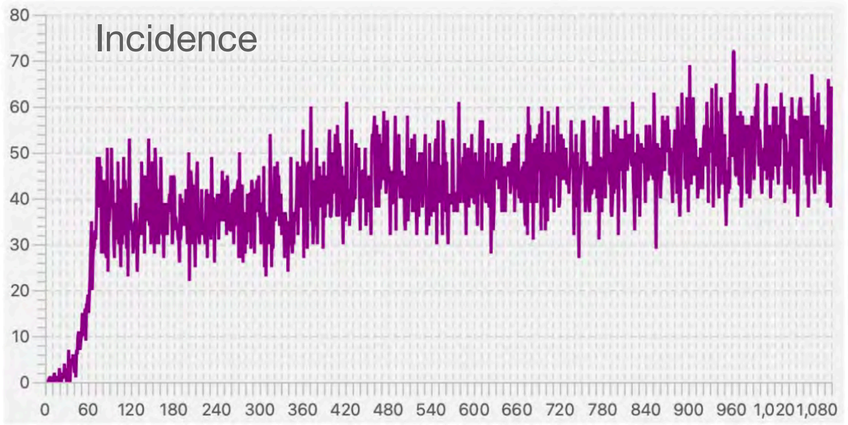
Time: 1100		Total Infectious (I):	292
Uninfected Pop (S):	66713	Recovered (V):	31810
Total Infected Pop (E):	244	New Deaths (ΔD):	0
Newly Infectious (ΔI+):	57	Total Deaths (D):	941

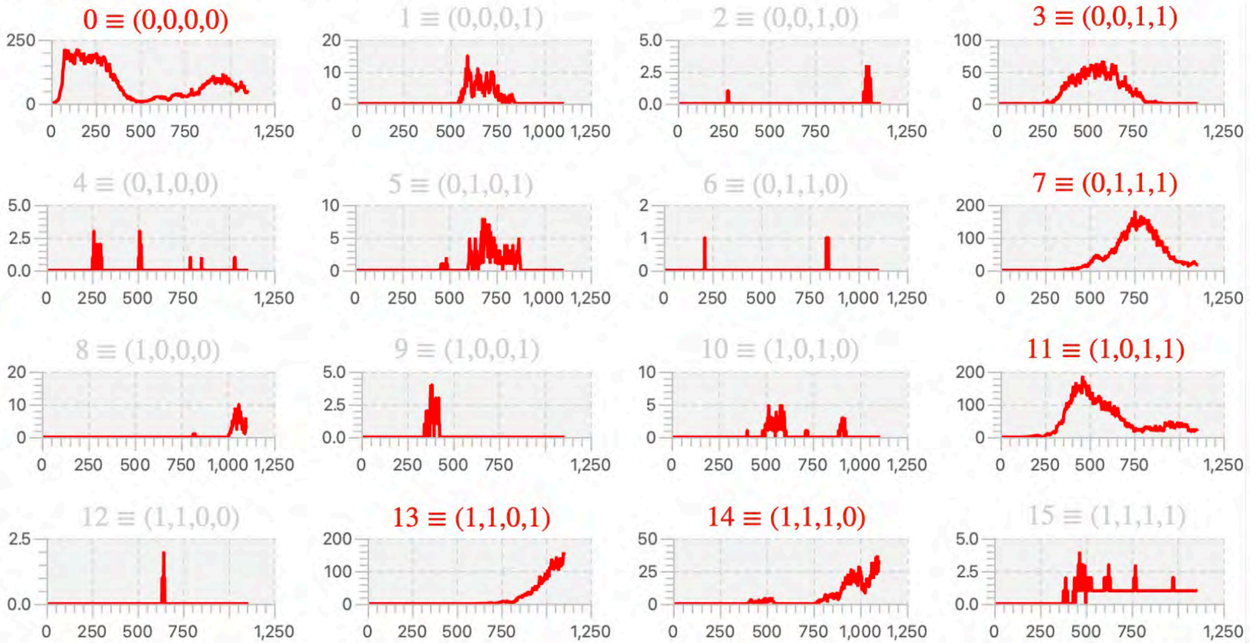




B. cross immunity = 0.5

Time: 1100		Total Infectious (I):	274
Uninfected Pop (S):	67061	Recovered (V):	31531
Total Infected Pop (E):	257	New Deaths (ΔD):	1
Newly Infectious (ΔI+):	64	Total Deaths (D):	877





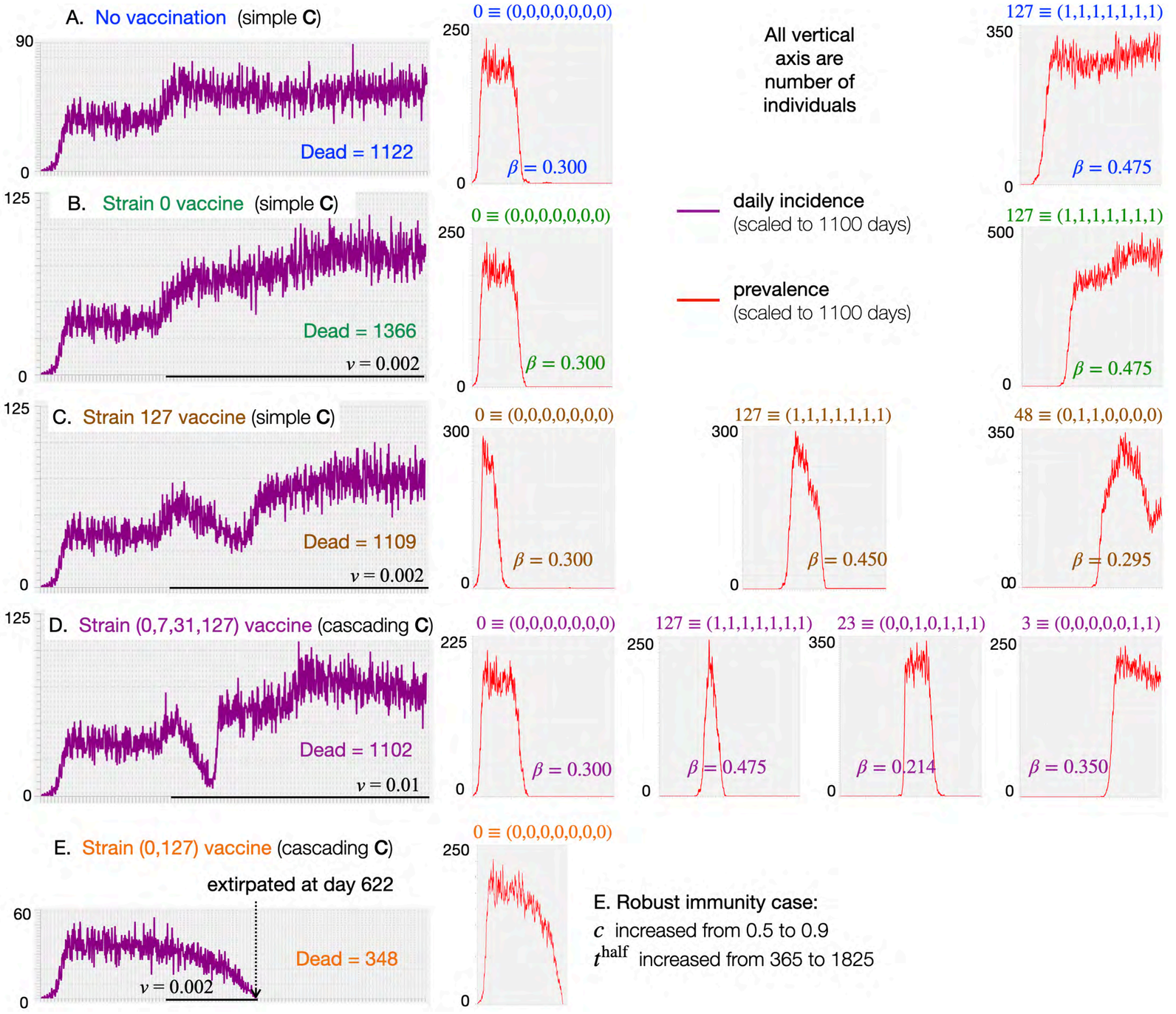
J-RAMP SEIVD IBM

multistrain

Univalent vaccine

Multivalent vaccine

Robust Immunity Case

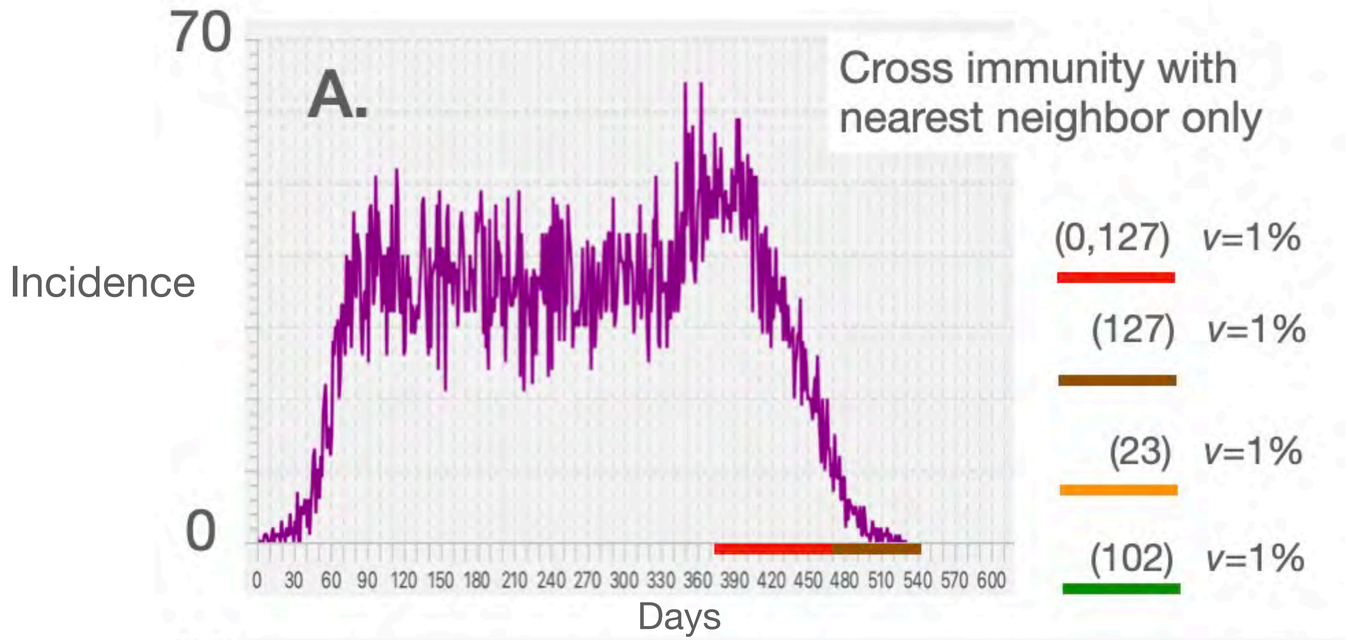


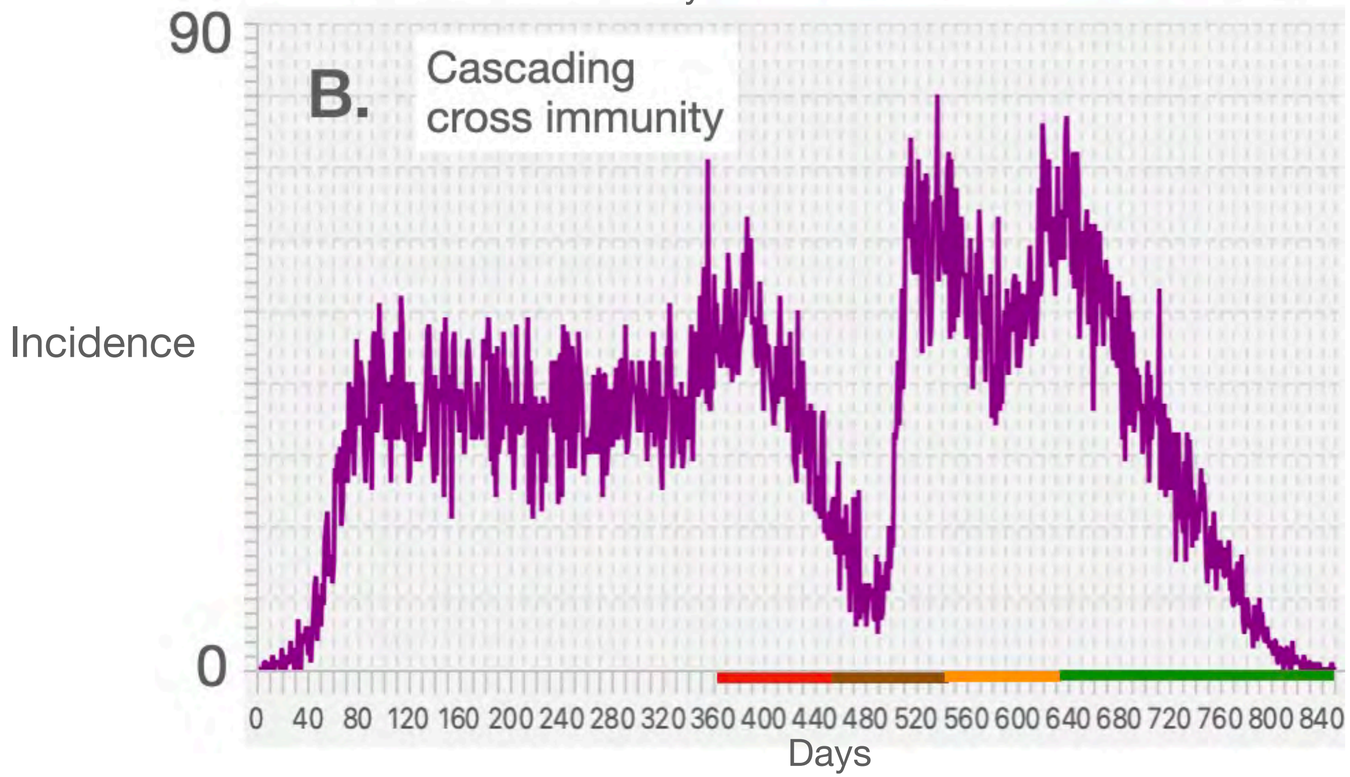
J-RAMP
SEIVD
IBM

multistrain

Adaptive vaccine rollout:

Every 90 days match the 2 top strains (if >10 prevalence)





Thanks

Questions before moving on to the next part of the talk

Overview

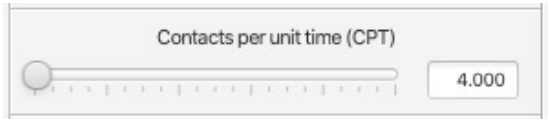
- An open and expressive design of the model platform aids in and encourages exploration and experimentation.
- The J-RAMP architecture is designed with this in mind.
- Novel Features:
  - An API covering all interactive elements of the simulation, with both local and remote access.
    - APIs are well known for many app-types, in particular Web apps.
  - Ability to safely change definitions of dependent terms (RAMS).
- I will try not to be too “computer-science-y”. Feel free to interrupt if you are unfamiliar with anything you hear.

API: Application Programming Interface

- A set of functions providing both control and data interfaces to an application.
- Can be included programmatically in a *script.*
- Allows computational logic to operate and manage an application.

API: Interactive elements

- Parameters and settings
  - Numerical parameters
  - Vaccine on/off
  - Vaccine mode



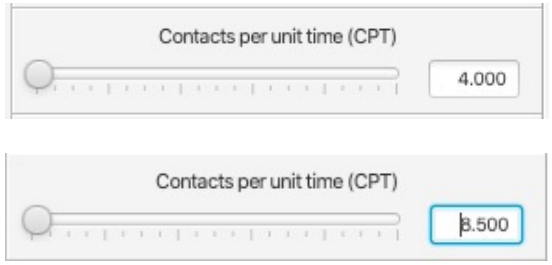
- Runtime operation

- Data retrieval

Time Course per Strain						
Strain 0						
Time	E	I	V	D	DI+	DD
0	0	1	0	0	1	0
1	1	1	0	0	0	0
2	2	1	0	0	0	0
3	2	1	0	0	0	0
4	3	1	0	0	0	0
5	3	1	0	0	0	0
6	3	2	0	0	1	0
7	3	2	1	0	1	0
8	3	3	1	0	1	0
9	4	3	1	0	0	0
10	4	3	1	0	0	0
11	6	2	2	0	0	0

API: Functions

- get/set
  - “Airport codes” identify all gettable/settable items.
  - e.g., `set CPT 8.5`



- reset, step, run_for, run
  - Control simulation dynamics.

- totPop, pathPop
  - Show current populations wrt state and pathogen variant, respectively.

- popTrail, pathTrail
  - Show time evolution of populations wrt state and pathogen variant, respectively.

BPL Reference

Commands

`reset`	Resets simulation with initial slider/switch values.
`restart`	Resets simulation with current slider/switch values.
`run_for T`	Run the simulation from the current time for ***`T`*** steps.
`run`	Run the simulation for the number of steps set by the last **`run_for`** command.
`step`	Single-step the simulation one time unit.
`get time`	Returns current simulation time.
`getLimit XXX`	Returns lower and upper values of limit control labeled ***`XXX`*** on separate lines.
`getPath XXX id`	Returns value of parameter ***`XXX`*** in pathogen ***`id`***.
`getOption op`	Returns the option number for operator ***`op`*** in the operator overrides.
`get XXX`	Returns value of non-limit control labeled ***`XXX`***.
`setLimit XXX lo hi`	Sets the limit control ***`XXX`*** to limits ***`lo hi`***.
`setPath XXX id Y`	Sets value of parameter ***`XXX`*** in pathogen ***`id`*** to ***`Y`***.
`setOption op Y`	Sets option for operator ***`op`*** in the operator overrides to option number ***`Y`***.
`set XXX Y`	Sets value of non-limit control ***`XXX`*** to ***`Y`***.
`pathPop id Z`	***`id`*** is a pathogen id and ***`Z`*** must be one of the agent state identifiers (S, E, I, V, DI+, DD). Returns the current population in state ***`Z`*** with respect to pathogen ***`id`***.
`totPop Z`	***`Z`*** must be one of the agent state identifiers (S, E, I, V, DI+, DD). Returns the total population in state ***`Z`***.
`timeTrail`	Returns the array of timesteps for use in a table with other trail data.
`pathTrail id Z`	***`id`*** is a pathogen id and ***`Z`*** must be one of the agent state identifiers (S, E, I, V, DI+, DD). The population array for that pathogen and for that state over time is returned.
`popTrail Z`	***`Z`*** must be one of the agent state identifiers (S, E, I, V, DI+, DD). The array of the entire population over time for that state is returned.
`sleep T`	NMBPL processor sleeps for ***`T`*** milliseconds. (May be used for synchronization during longer simulations.)

Notes

- Commands without explicit return do not return a value
- **`get`** and **`set`** use the 3-letter "airport codes" associated with each control
- Commands may be sequenced; e.g.

  ```
  run_for 50 set XMT 1.5 run trail PATH I
  ```

  will run for 50 timesteps, set the transmission paramter to 1.5, run for another 50, and return the 100 population values of I.
- Returned values are separated by newlines. When command sequences are sent remotely, left and right brackets (i.e. \[\ and \]\) followed by newlines enclose the sequence of returned values, even if the sequence is empty; e.g.,

  ```
  [\n50\n100\n1000\n]
  ```

- When using **`trail`**: if embedded in a command sequence, this must be the only command that returns a value (since the array returned by each of them will be embedded in the set of answers returned by the command sequence.)
- Use the console to test command sequences before using them remotely.
- BPL uses TCP/IP sockets to implement a client-server protocol on the local host. Clients initiate a connection on a specified port (default is 8080) and transmit a command sequence; the SEIVAgent server replies with the result and closes the connection.
- Changes to the port number, if saved to Preferences, will reappear in subsequent uses. Multiple instances of Numerus can run concurrently using different ports. Changes will automatically restart the server.
- **Restart** button restarts the server.

API: On-board Scripting

- A Javascript runtime is included extended with access to the API.

- Run for 180 days w/o vaccinations
- Vaccinate for two variants with greatest population.
- Re-evaluate every 365 days.
- Repeat for total of 1100 days.

```
var script = function() {
  var obj = new Object();
  obj.getIPop = function() {
    var ans = new Array();
    var entropy = SEIV.get(ENT);
    var pathSize = Math.pow(2, entropy);
    for (var j = 0; j < pathSize; j++) {
      ans[j] = +SEIV.pathPop(j, I);
    }
    return ans;
  };
  obj.getTwoBest = function(a) {
    var best0 = [-1, 0];
    var best1 = [-1, 0];
    for (var i = 0; i < a.length; i++) {
      if (a[i] > best0[1]) {
        best1 = best0;
        best0 = [i, a[i]]
      } else if (a[i] > best1[1]) {
        best1 = [i, a[i]];
      }
    }
    return [best0, best1];
  };
  obj.loop = function(len) {
    SEIV.run_for(len);
    obj.fixVAs();
  };
```

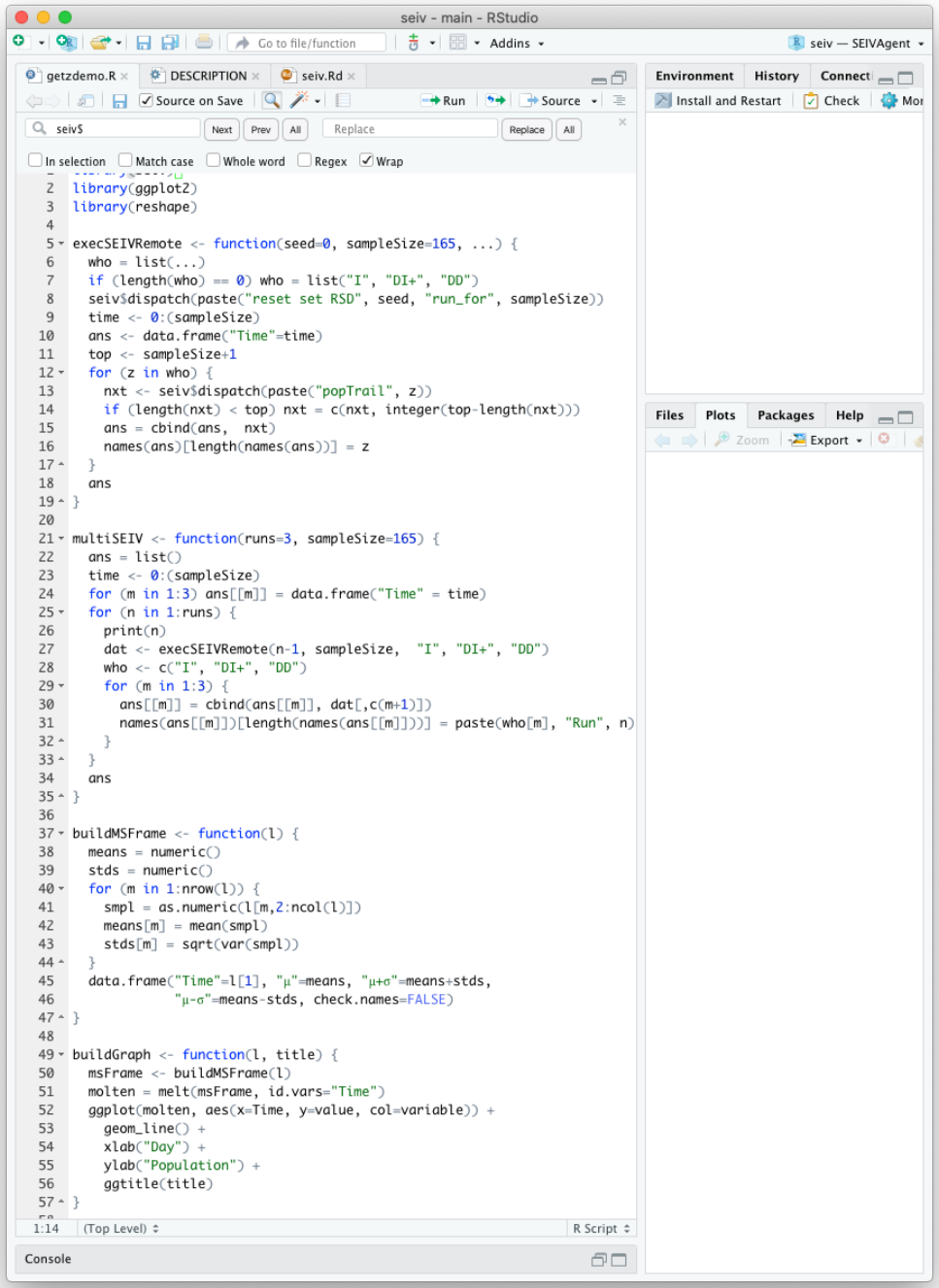
```
  obj.fixVAs = function() {
    var iPop = obj.getIPop();
    var twoBest = obj.getTwoBest(iPop);
    var best = twoBest[0];
    var nextBest = twoBest[1];
    if (best[1] <= 10 && nextBest[1] <= 10) {
      console.log("time", SEIV.get("time"), "No change",
                  "VA1="+SEIV.get(VA1), "VA2="+SEIV.get(VA2));
      return;
    }
    for (var j = 1; j <= 4; j++) {
      SEIV.set("VA"+j, "off");
    }
    SEIV.set(VA1, best[0]);
    if (nextBest[1] > 10) SEIV.set(VA2, nextBest[0]);
    else if (SEIV.get(VA1) == SEIV.get(VA2)) SEIV.set(VA2, null);
    console.log("time", SEIV.get("time"), "VA1="+SEIV.get(VA1),
                "VA2="+SEIV.get(VA2));
  };
  obj.init = function(vaccOn) {
    SEIV.reset();
    for (var j = 1; j <= 4; j++) {
      SEIV.set("VA"+j, "off");
    }
    SEIV.setLimit(VOO, vaccOn, 2000);
    SEIV.set(VA1, 0);
  };
  // n = total days, m = vacc period
  obj.run = function(n, m, delay) {
    console.log("\nStart: running for", delay, "days w/o vaccinations")
    obj.init(delay);
    SEIV.run_for(delay);
    console.log("time", SEIV.get("time"), "Vaccinations start");
    obj.fixVAs();
    p = Math.ceil((n - delay)/m);
    for (var i = 0; i < p; i++) {
      obj.loop(m);
    }
  };
  return obj;
}();

// 1100 = total days, 365 = vacc period, 180 = delay
var go = function() {
  script.run(1100, 365, 180);
}
```

API: Remote Scripting

- The API acts as a local Internet server to receive API requests from other platforms.
- Drivers can be easily configured to provide access from Python, Ruby, etc.
- In particular, a package for the R statistical platform is being distributed that fosters API communication from R programs.
- Other than installing the package, no other special installation is required.

Use with R

```
5  execSEIVRemote <- function(seed=0, sampleSize=165, ...) {
6    who = list(...)
7    if (length(who) == 0) who = list("I", "DI+", "DD")
8    seiv$dispatch(paste("reset set RSD", seed, "run_for", sampleSize))
9    time <- 0:(sampleSize)
10   ans <- data.frame("Time"=time)
11   top <- sampleSize+1
12   for (z in who) {
13     nxt <- seiv$dispatch(paste("popTrail", z))
14     if (length(nxt) < top) nxt = c(nxt, integer(top-length(nxt)))
15     ans = cbind(ans,  nxt)
16     names(ans)[length(names(ans))] = z
17   }
18   ans
19 }
```



seiv - main - RStudio

Go to file/function | Addins | seiv — SEIVAgent

getzdemo.R | DESCRIPTION

Source

```
  ans
}

buildMSFrame <- function(l) {
  means = numeric()
  stds = numeric()
  for (m in 1:nrow(l)) {
    smpl = as.numeric(l[m,2:ncol(l)
    means[m] = mean(smpl)
    stds[m] = sqrt(var(smpl))
  }
  data.frame("Time"=l[1], "μ"=means
             "μ-σ"=means-stds, chec
}

buildGraph <- function(l, title) {
  msFrame <- buildMSFrame(l)
  molten = melt(msFrame, id.vars="T
  ggplot(molten, aes(x=Time, y=valu
    geom_line() +
    xlab("Day") +
    ylab("Population") +
    ggtitle(title)
}

go <- function(runs=10, sampleSize=
```

5:1 execSEIVRemote(seed, sampleSize, ...)

Environment | History | Connections | Build | Git

Install and Restart | Check | More

Files | Plots | Packages | Help | Viewer

Zoom | Export

Console | Terminal | Jobs

~/Documents/eclipseWS/Nova3_WS/SEIVAgent/R/seiv

```
Restarting R session...

> source('~/Documents/eclipseWS/Nova3_WS/
SEIVAgent/R/getzdemo.R')
> go(10)
```

SEIVAgent Version 1.071

New | Open Settings | Save Settings | Save As ... | Current Settings | Open Path Set | Save Path Set | Save As ... | Current Path | Fixed | Random

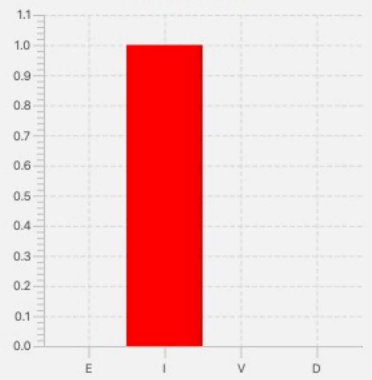
```
Reset
Time: 0
Total Vaccinations:      0
Uninfected Pop (S):      1999
Total Infected Pop (E):  0
Newly Infectious (ΔI+):  1
Total Infectious (I):    1
Recovered (V):           0
New Deaths (ΔD):         0
Total Deaths (D):        0
```

Initialize Seed (ISD) 0

Runtime Seed (RSD) 0

% Mortality (MOR)	0.027	0.027
Transmission Parameter (XMT)	0.300	0.300
Environmental Persist. (PST)	1.000	1.000
Within-host Replication (INV)	1.000	1.000
Median Latent Period (MLP)	4.000	4.000
Median Infections Period (MIP)	7.000	7.000
Population Size (POP)		2000
Contacts per unit time (CPT)		8.500
Mutation Rate (MUR)		0.000
Abruptness of Waning (AOW)		4
Cross Immunity (CIM)		0.000
Waning Half-Life (WHA)		365
Seas. Trans. Perturb. (STP)		0.000
Seas. Trans. Shift (STS)		0.000
Seas. Trans. Period (PER)		365
Infect. Prob. 1/2 Contacts (IPC)		0.000
Entropy (Log_2) of Strain Count (ENT)		3
Shedding (SHD)		0.000

Current Totals



0 | Log Off | JS On | S On | Line On | Bar On | V On | A On | P On | Intern On | Op On | Reset | Step | Run | 165

RAM: Runtime alterable module

- APIs limit programmatic interaction to runtime control and parameter settings.
- Many algorithms compute intermediate internal values derived from parameters and current state: **terms**.
- Often terms are computed with concise functions.
- However, experimenting with alternative definitions is often inconvenient and risky.
- RAMs offer a safe and easy approach to alternate definitions of terms.

Programming a RAM

- Each RAM is defined by a term definition from the algorithm and the corresponding code that implements that term.
- A User Guide shows both the definition and code, and describes the auxiliary functions needed to implement that code.
- The user may be interested in experimenting with alternate definitions of the term.

J-RAMP SEIVD IBM RAM Guide

rms wgetz

May 2021

ω (Equation 9)

$$\omega_{ij}(t) = \begin{cases} 0 & \text{if } \mathrm{V}_j \text{ is null} \\ \dfrac{1}{1 + \left((t - \tau_{ij} - \sigma_\mathrm{I} - \sigma_\mathrm{E})/t_j^{\mathrm{half}}\right)^\sigma} & \text{if } t \geq \tau_{ij} + \sigma_\mathrm{I} + \sigma_\mathrm{E} \\ 1 & \text{if } t < \tau_{ij} + \sigma_\mathrm{I} + \sigma_\mathrm{E} \end{cases}$$

Code

```
double omega(int i, int j)
{
  if (i == 0) return 0.0;        // Special case; A_0 never infected
  InfectedAgentState astate = getStateOf(i, j);
  if (astate == null || !(astate instanceof V)) return 0.0;
  double tau =  tau(i,j);
  double sigI = sigmaI(j);
  double sigE = sigmaE(j);
  double timeSinceInfect = tau + sigI + sigE;
  if (time() <= timeSinceInfect) return 1.0;
  return 1/(1 + Math.pow((time() - timeSinceInfect)/tHalf(j), Params.sigma()));
}
```

Notes

`astate = getStateOf(i,j)`	`astate` is assigned to a Java `InfectedAgentState` object for agent i with respect to pathogen j. `InfectedAgentState` objects can either be state `E`, `I`, `V` or `null` (the latter for never-infected or dead agents.)
`astate == null || !(astate == V)`	`astate` is something other than state `V`.
`tau(i,j)`	$= \tau_{ij}$.
`sigmaI(j)`	$= \sigma_\mathrm{I}$.
`sigmaE(j)`	$= \sigma_\mathrm{E}$.
`tHalf(j)`	$= \tau_j^{half}$.
`time()`	$= t$.
`Params.sigma()`	$= \sigma$.

Example: Cross Immunity

Default definition (nearest neighbor)

$$c_{j\ell} = \begin{cases} 1 & \text{if } \ell = j \\ c & \text{if } \ell \text{ differs from } j \text{ by one allele} \\ 0 & \text{otherwise} \end{cases}$$

```
public double crossImmune(int j, int l) {
    if (l == j) return 1;
    else if (Tools.hdist(j,l) == 1) return Params.cImmune();
    else return 0;
}
```

Cascading definition

$$c_{j\ell} = \begin{cases} 1 & \text{if } \ell = j \\ c^k & \text{if } \ell \text{ differs from } j \text{ by } k \text{ alleles} \end{cases}$$

```
public double crossImmune(int j, int l) {
  if (l == j) return 1;
  return Math.pow(Params.cImmune(), Tools.hdist(j, l));
}
```

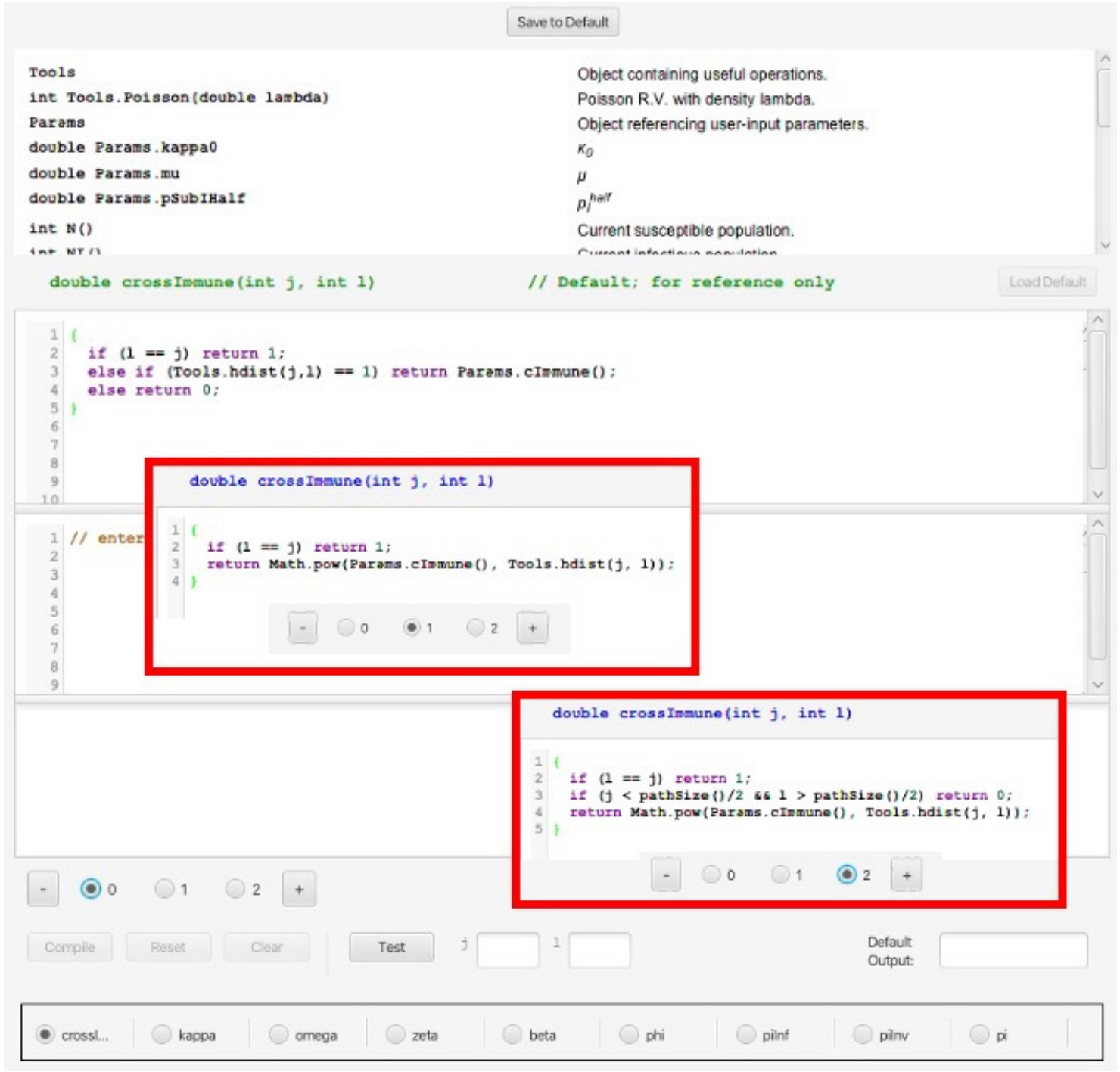
Possible escape mutation definition

$$c_{j\ell} = \begin{cases} 1 & \text{if } \ell = j \\ 0 & \text{if } j \leq 63 \text{ and } \ell > 63 \\ c^k & \text{otherwise, where } \ell \text{ differs} \\ & \text{from } j \text{ by } k \text{ alleles} \end{cases}$$

```
public double crossImmune(int j, int l) {
  if (l == j) return 1;
  if (j < pathSize()/2 && l > pathSize()/2) return 0;
  return Math.pow(Params.cImmune(), Tools.hdist(j, l));
}
```

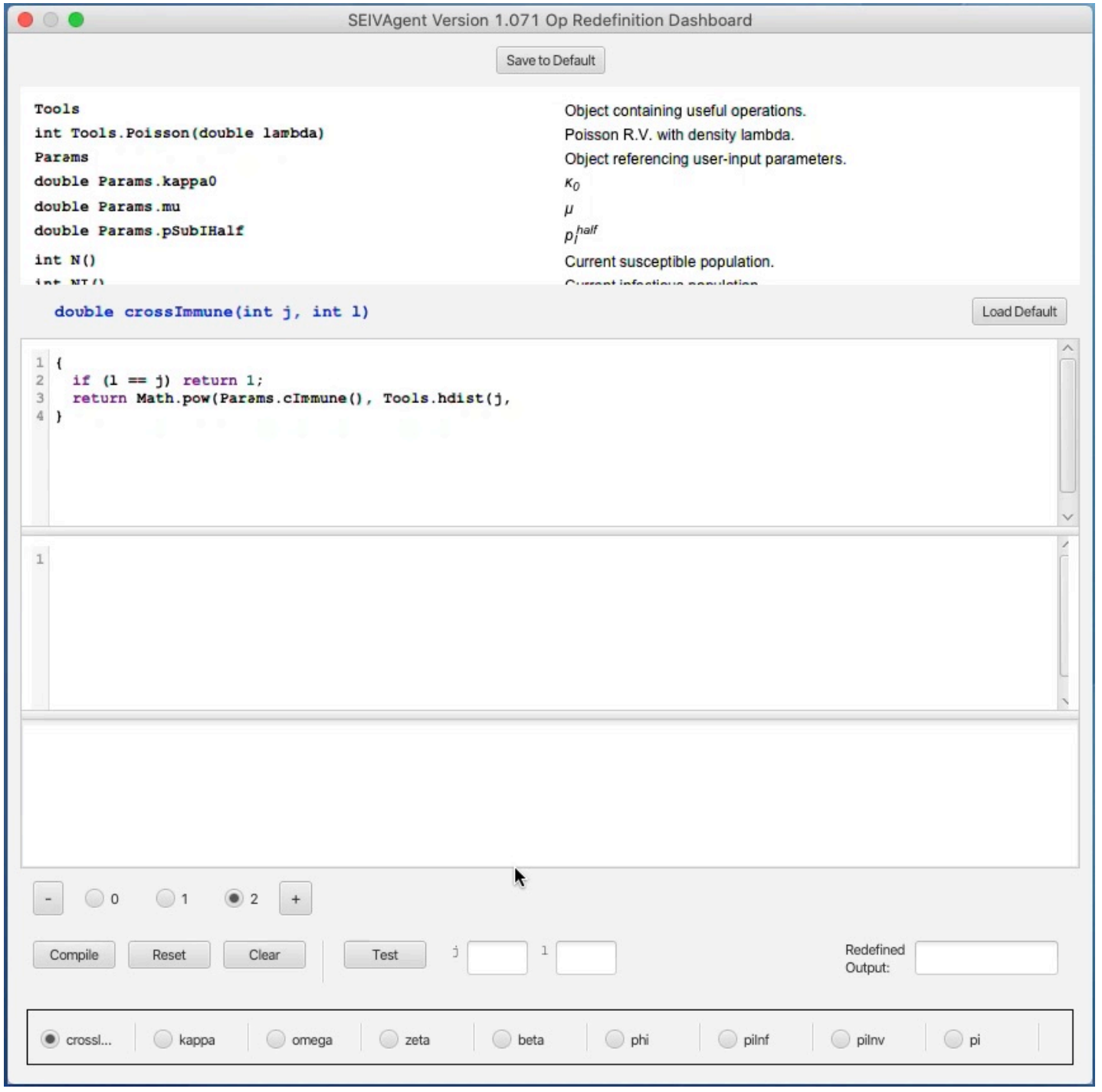
RAM frame

- Available RAMs are shown at the bottom of the window.
- Each RAM begins with the fixed internal default definition as Option 0.
- New definitions are added as additional options.
- RAMs *never* overwrite the default definition.
- RAM selection is part of the API.



RAM Programming

- A mini-development environment is used to write and compile alternative options.
- A reference window describes available help functions.
- Print statements can be included in the definitions.
- Options can be tested within the frame before being deployed.



Conclusion

- Versatility and expressiveness in applications promote experimentation and creativity when they are used.
- APIs provide programmatic control over an application’s user-level functionality.
- RAMs provide opportunities for alternate definitions of internal elements.
- RAM selection is part of the API.
- We are working on tools to simplify deployment of APIs and RAMs.
  - Working version already achieved with RAMs.