Challenge Scenario

Zombie Antidote

eX Modelo school

OpenMOLE

June 26, 2019

Story

There are some agents carrying a zombie **antidote**. When they are alerted to the zombie attack, they take the antidote.

This antidote has an **activation delay**. Before the actual activation, agents are weaker than normal humans. After activation, they are back to their normal stamina and they are immunized to zombie infection.

Alert mechanism:

- They become alerted individually when they see a zombie.

When they are immunized, they have specific behaviors:

- They alert everyone they meet.
- They run towards zombies to lure them away from normal humans: once a zombie detect an antidoter, he attacks him and they remain face to face for the future steps.

New parameters, we can act on

- Number of antidoters: *redCrossSize*
- Delay of activation of the antidote: *redCrossActivationDelay*
- Efficiency of the antidote: *redCrossEfficiencyProbability*
- Exhaustion probability modified during activation: *redCrossExhaustionProbability*. Being immunized decreases the stamina of antidoter during the period of activation.

By default, we consider the parameter *redCrossInformProbability* set equal to 1: antidoters transmit their information to every human they meet.

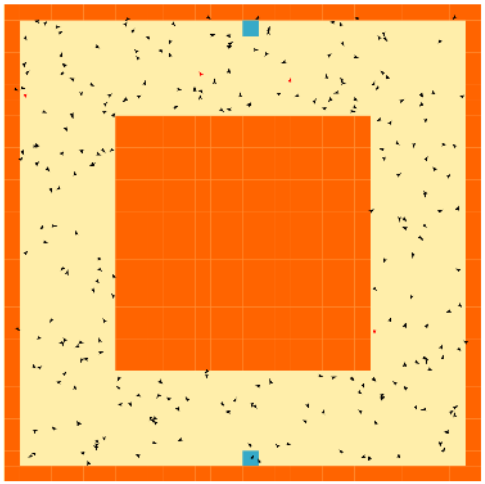
Questions on the model

Not all of the following questions were adressed in the present study.

- What is the effect of the initial number of antidoters on the number of zombifications, all other antidote features being set arbitrarily.
  - Direct sampling and replications
- Is the decoy technique of antidoters really efficient? Do they repeatedly "fight" the same zombie, preventing it to attack other humans?
  - Number of antidoters who took their vaccine but became zombified after the delay of activation (due to efficiencyProbability<1)
  - Number of antidoters who took their vaccine but then became zombified during the delay of activation

- Given an initial number of antidoters, what are the antidote features that maximize the number of rescued / minimize the number of zombifications?
- Given antidote features, study the compromise between having more antidoters (quantity) and their ability to transmit their information (quality).
- A zombie attack happened, anidoters were present and took their antidote. Howerver, despite we know they all took the same antiote, we do not know what were the antidotes features. Thanks to witness and some videos, we have data of the attack's dynamics. We want to find possible antidote features values.

Results

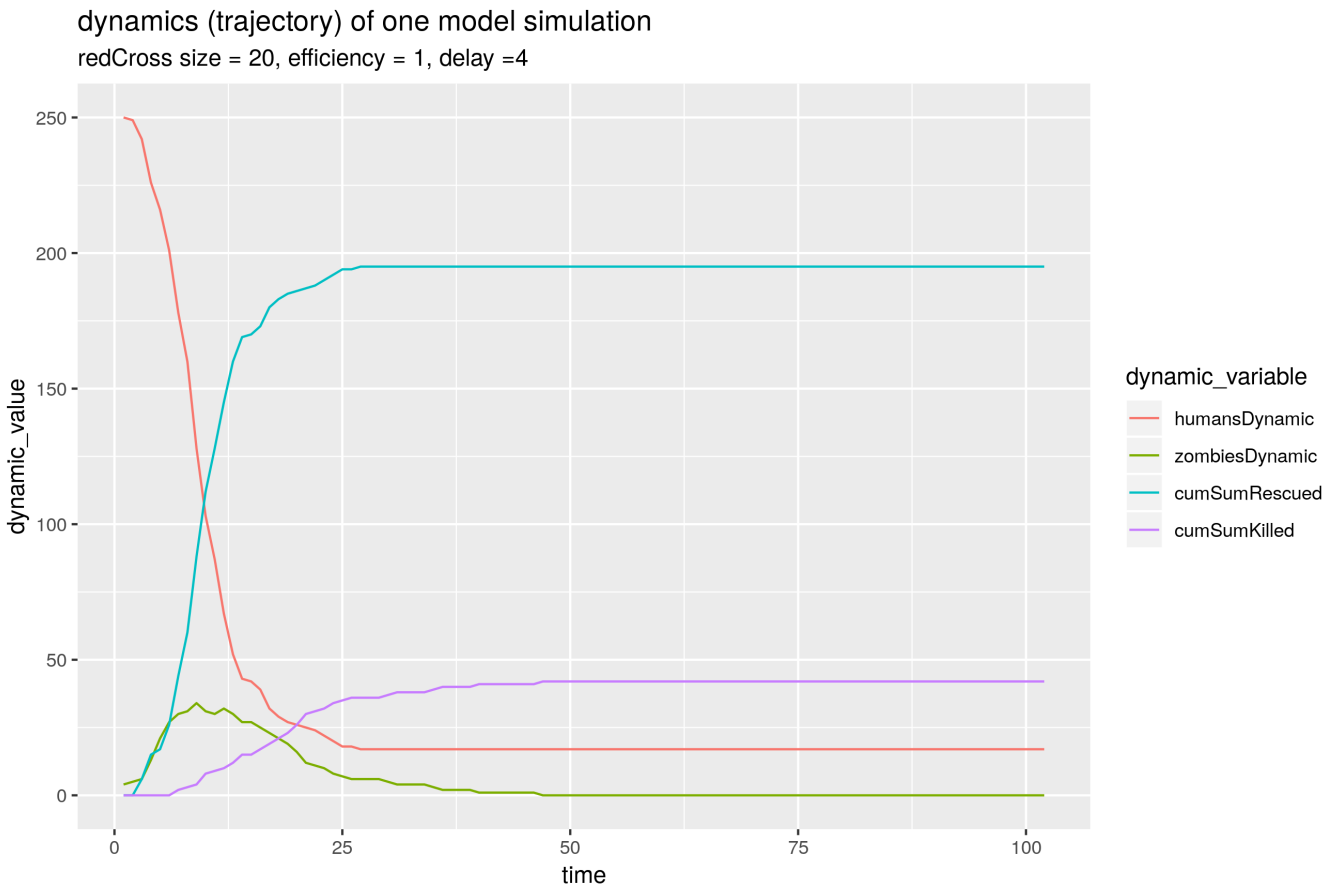


World for all simulations: *square* world.
It's a closed world with 2 rescue zones (only accessible by humans, not by zombies and redCross agents).

Four dynamics : *humans*, *zombies* (or *zombified*), *rescued* and *killed* (the sum of these dynamics is constant with time).

- Initial number of humans :250
  RedCross agents are part of humans (so their number is between 0 and 250).
- Initial number of zombies: 4

Example of dynamics: one simulation with given redCross parameters, and environment parameters.



Replications with 3 parameter values: *NoRedCross*, eff_low, eff_max.

Case	redCross	ActivationDelay	ExhaustionProbability	EfficiencyProbability
eff_low	20	4	0.6	0.95
eff_max				1.0

Mean dynamics (trajectories) over 50 replications in these three cases and we indicate confidence intervals.

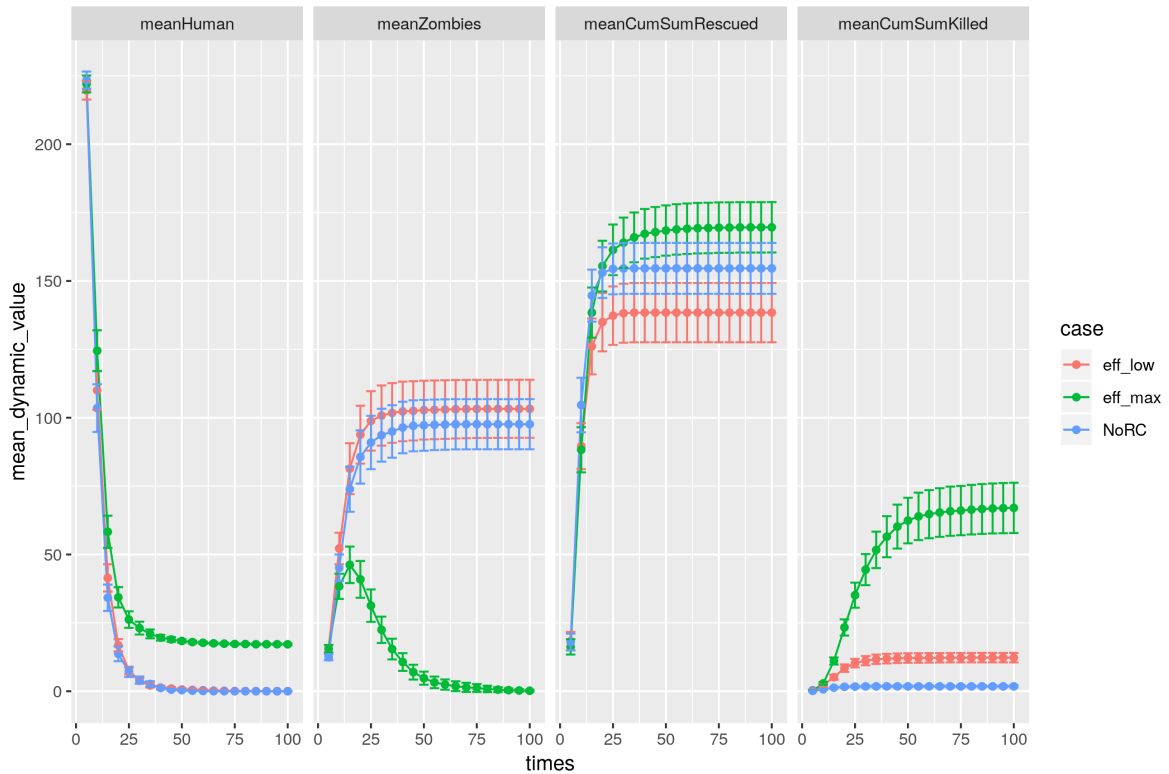
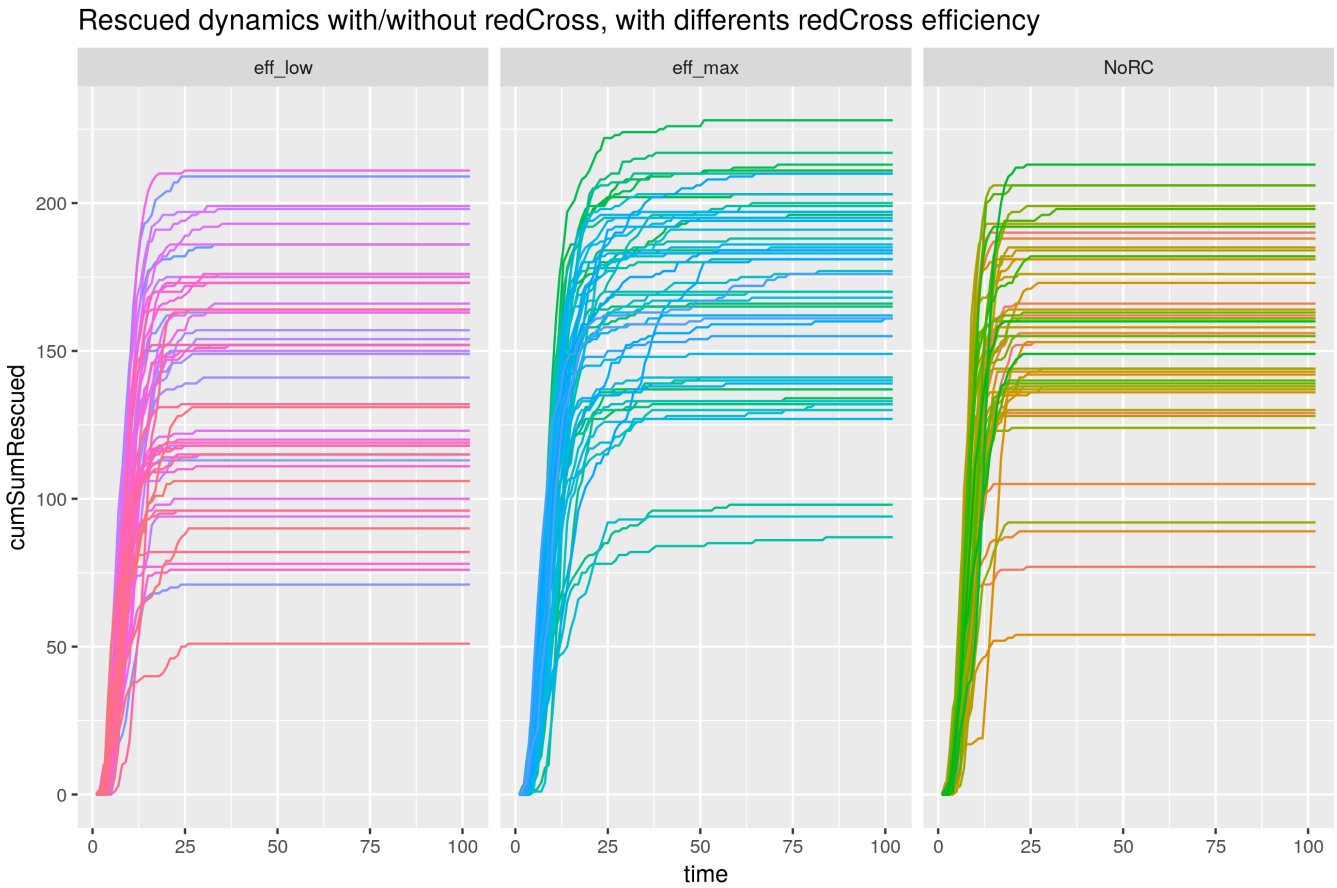
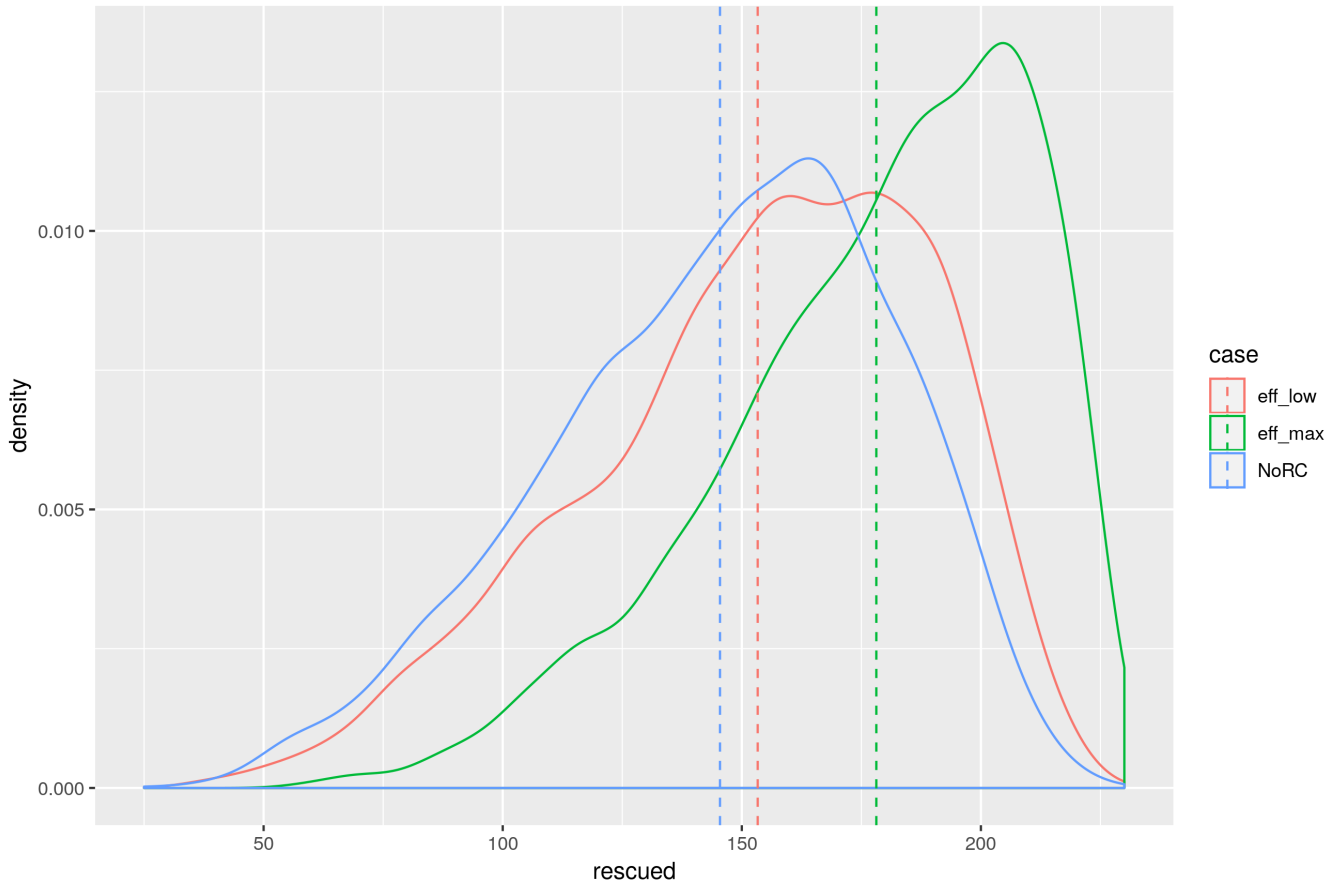


Illustration of the variability in trajectories due to stochasticity.



Histograms of the *rescued* (final time), performed with replications of 4000 simulations of the model in the three cases.



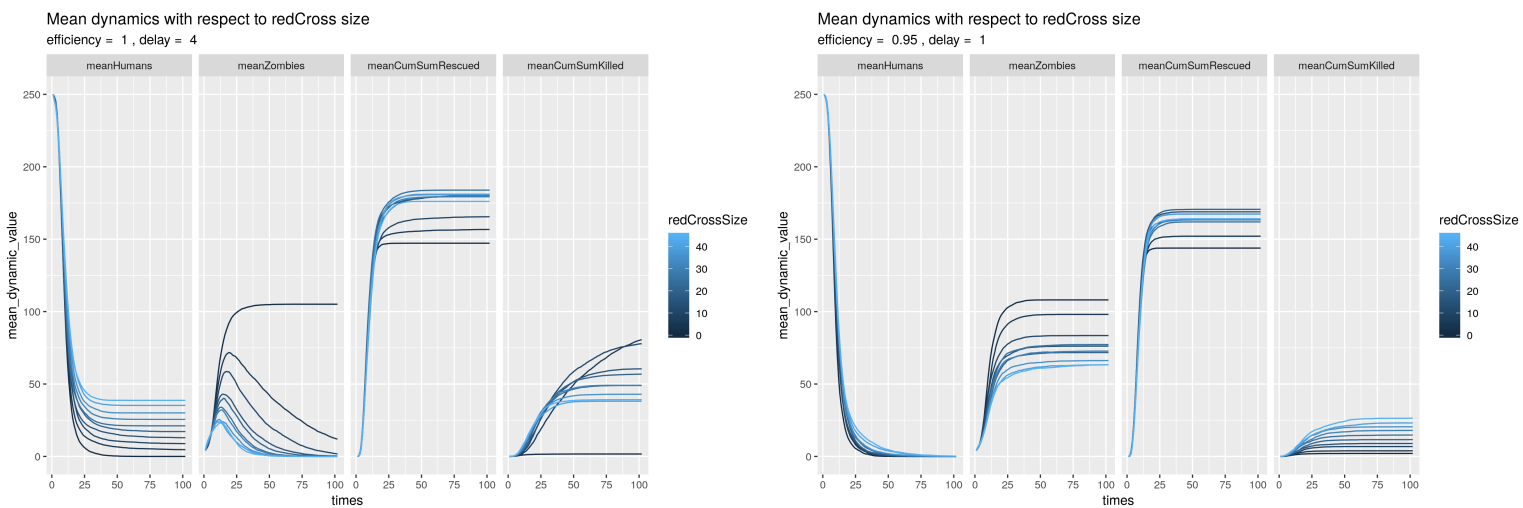
Influence of redCross size on dynamics

In this section, we consider 3 antidotes.

Aim : Analyse how vary output quantities (rescued,...) with *redCrosssize*, in each case.

Method: directSampling + replications.

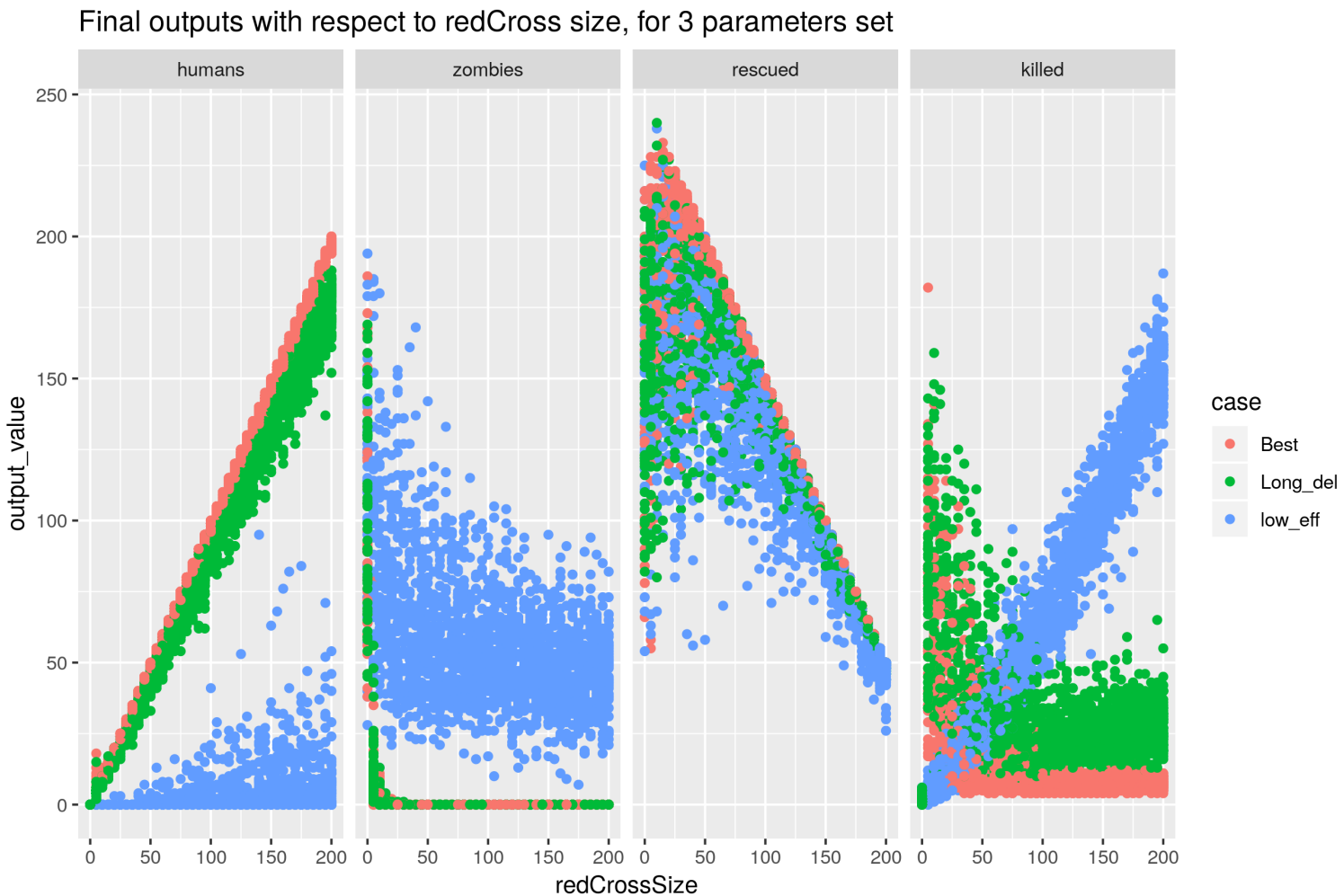
Case	ExhaustionProbability	ActivationDelay	EfficiencyProbability
Best	0.6	1	1
Long_del		4	1
low_eff		1	0.95



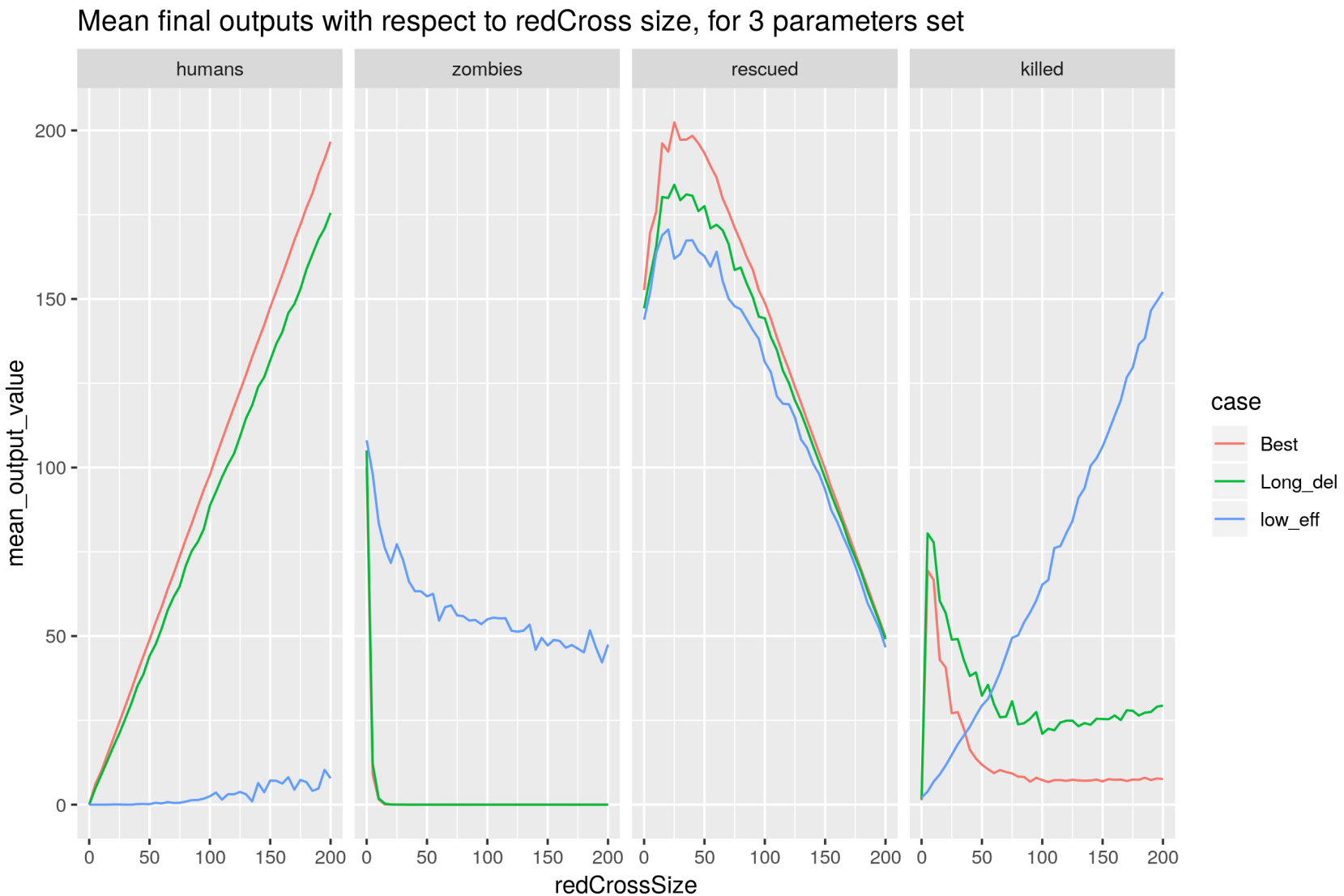
Mean trajectories over 50 replicationns.

Influence of redCross size on dynamics

Final time quantities vs *redCrosssize*, with colour varying with the three antidote considered



Mean final outputs



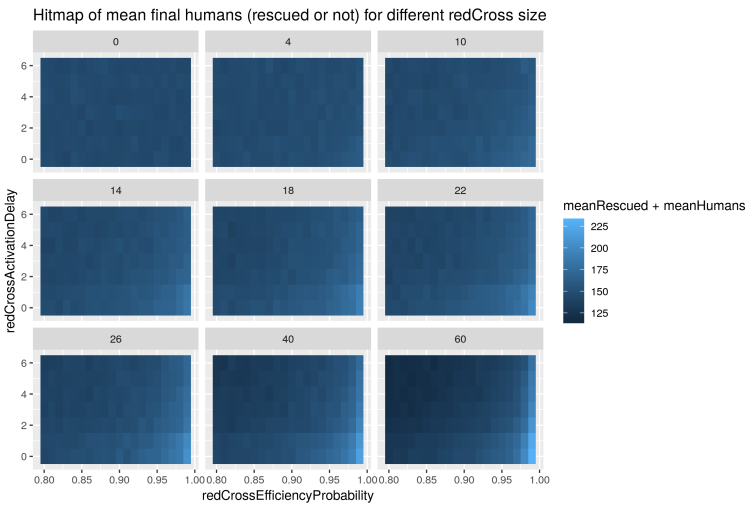
Find parameters values that minimise a final time output (DirectSampling)

OpenMOLE

Aim: find optimal antidote features.

Tool: DirectSampling + replications

```
val sampling = (redCrossSize in (0 to 60 by 2)) x
               (redCrossActivationDelay in (0 to 6 by 1)) x
               (redCrossEfficiencyProbability in (0.8 to 1.0 by 0.01))
```



Parameters value that maximises the number of humans (rescued + humans):

- $redCrossSize = 60$
- $activationDelay = 0$
- $efficiencyProbability = 1$

This section corresponds to the analogue for the RedCross of a question raised in the Army cheatsheet.

We consider that researchers currently have an antidote with known features. The aim of this section is to find on which parameters should we focus research effort to improve the antidote (for a given zombies attack scenario: $redCrossSize = 20$, $zombiesSize = 4$):

- redCrossActivationDelay
- redCrossEfficiencyProbability
- redCrossExhaustionProbability,

Improvement of an antidote (NSGA2)

The objective is to:

- maximize the number of rescued (+humans?) resulting the improved antidote
- minimize the "cost/ effort" to obtain it

... think of an emergency situation with limited time and budget.

Current antidote features are:

```
> currentRedCrossActivationDelay = 4
> currentRedCrossEfficiencyProbability = 0.98
> currentRedCrossExhaustionProbability = 0.6
```

Pourcentage of improvement compared to the current features of the antidote:
redCrossActivationDelayBonus, *redCrossEfficiencyProbabilityBonus* and *redCrossExhaustionProbabilityBonus* (varying in $[0, 1]$).

Antidote features are modified "linearly". For example,

```
ActivationDelay = floor(currentActivationDelay * (1-
ActivationDelayBonus))
```

Pourcentage of improvement compared to the current features of the antidote:
redCrossActivationDelayBonus, *redCrossEfficiencyProbabilityBonus* and *redCrossExhaustionProbabilityBonus* (varying in $[0, 1]$).

Antidote features are modified "linearly". For example,

```
ActivationDelay = floor(currentActivationDelay * (1-
ActivationDelayBonus))

EfficiencyProbability = currentEfficiencyProbability +
(1-currentEfficiencyProbability)* EfficiencyProbabilityBonus
```

Pourcentage of improvement compared to the current features of the antidote:
redCrossActivationDelayBonus, *redCrossEfficiencyProbabilityBonus* and *redCrossExhaustionProbabilityBonus* (varying in $[0, 1]$).

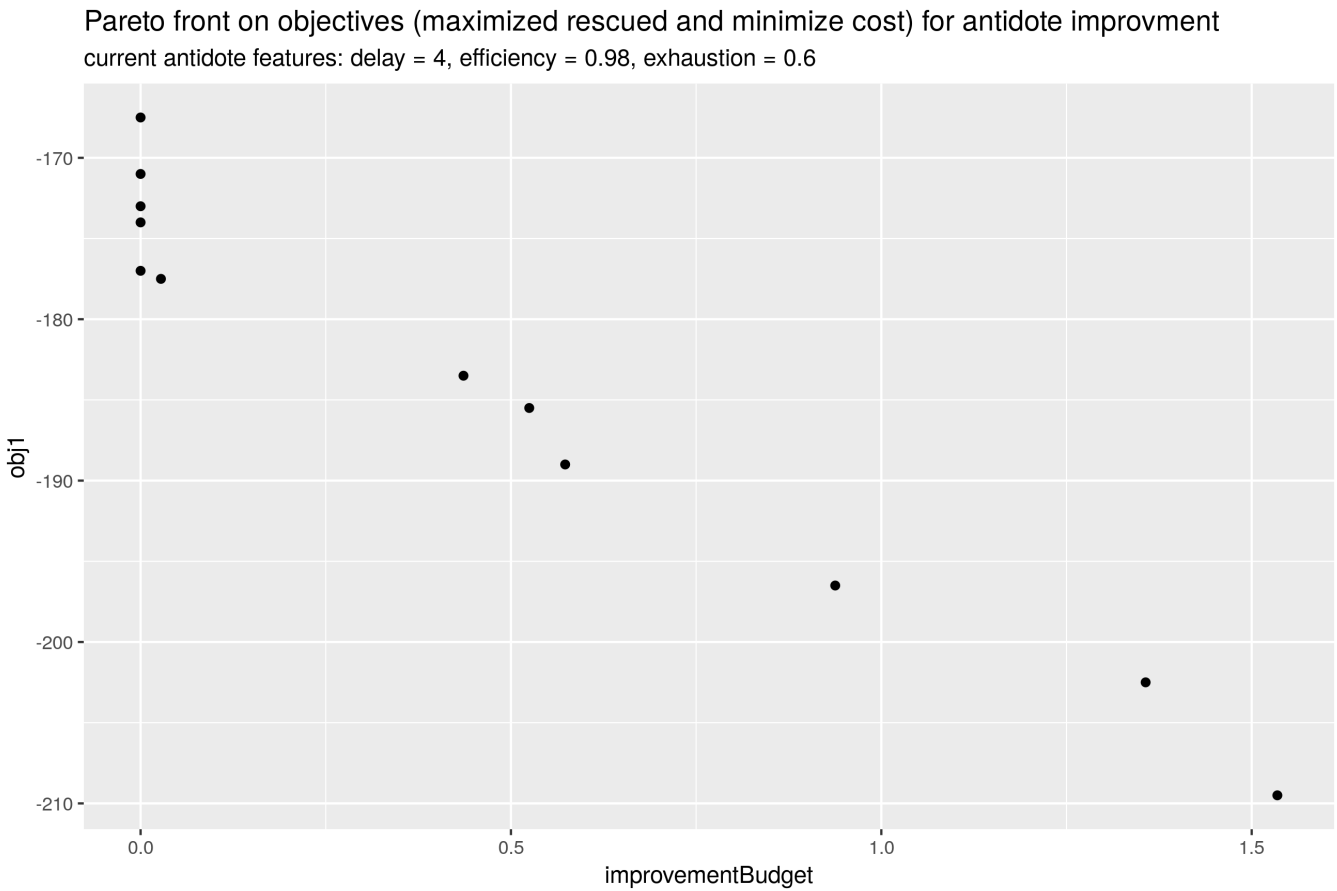
Antidote features are modified "linearly". For example,

```
ActivationDelay = floor(currentActivationDelay * (1-
ActivationDelayBonus))

EfficiencyProbability = currentEfficiencyProbability +
(1-currentEfficiencyProbability)* EfficiencyProbabilityBonus

ExhaustionProbability = humanExhaustionProbability +
(currentExhaustionProbability- humanExhaustionProbability)*(1-
ExhaustionProbabilityBonus)
```

Aim at minimizing the sum of pourcentage of improvement.

The question is thus to find a compromise between maximizing the number of rescued and minimizing the budget.

Pareto front obtained after 4000 iterations.



A zombies attack happened in a stadium. redCross agents were present and took their antidote. We know they **all took the same antidote** but we do not know what were the antidotes features they took.

Aim: Find antidote parameters values:

- redCrossActivationDelay
- redCrossEfficiencyProbability
- redCrossExhaustionProbability

A zombies attack happened in a stadium. redCross agents were present and took their antidote. We know they **all took the same antidote** but we do not know what were the antidotes features they took.

Aim: Find antidote parameters values:

- redCrossActivationDelay
- redCrossEfficiencyProbability
- redCrossExhaustionProbability

Data: Parts of the invasion dynamics:

- Initial conditions: 4 zombies, 250 humans (among them 20 redCross agent)
- Final values: all zombies were killed, and 183 humans were rescued
  ...in fact a value of 0.28 because we performed replications to get the data
- The number of humans at some time steps (humanDynamics are in a csv file)

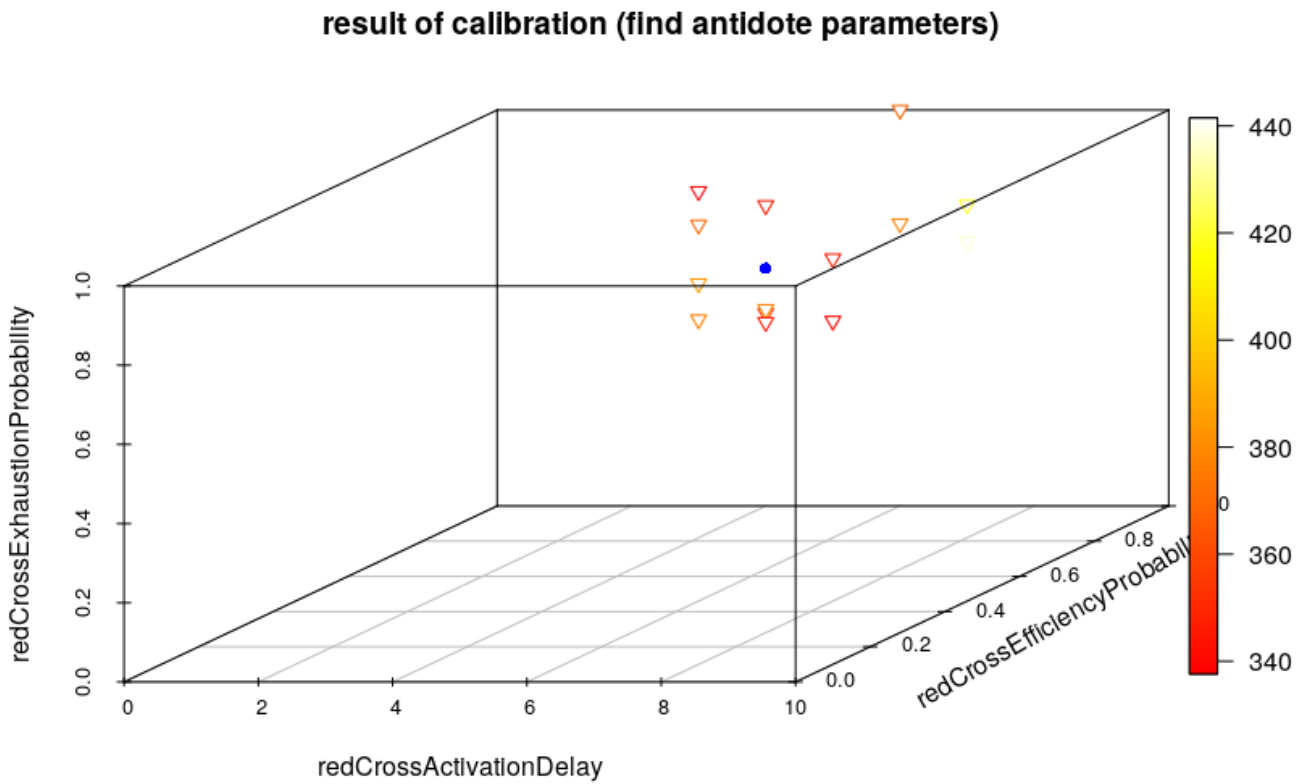
To answer this question, we perform a **calibration** on antidote parameters.

To compare model otuputs with data, we use the following **fitness**:

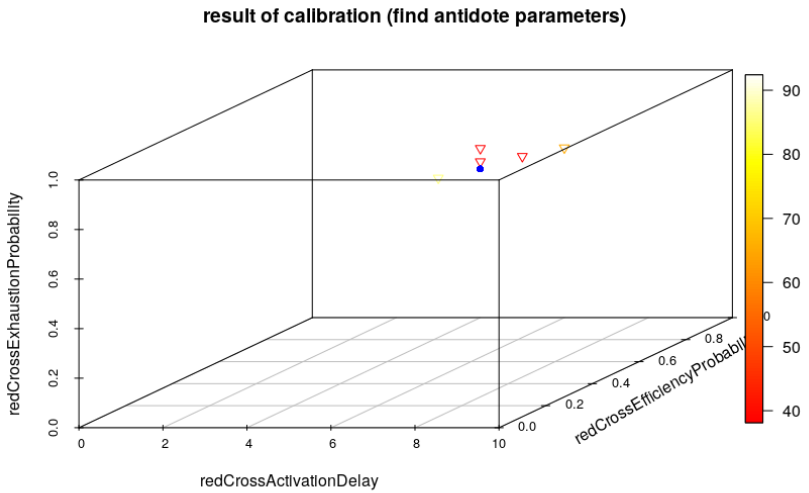
```
val objective =
ScalaTask("""
val fitness1 = math.abs(zombiesRef - zombies.toDouble)
val fitness2 = math.abs(rescuedRef - rescued.toDouble)
val fitness3 =
humansDynamicRef.zip(humansDynamic).map( x =>
math.abs(x._1 - x._2) ).sum
""")
```

Results : nsga2 algorithm managed to find the parameters values used to create the dataset !

The blue point represents parameters values used to create the data. The color from red to yellow for points refers to the value of the corresponding value of the objective *fitness3*.



Better solutions obtained by changing the objective and considering the **median** over the replications computed nsga2 algorithm (use of *aggregate*).



The solution with the lowest distance to the third objective is:

Variable	Data	Calibration result
redCrossActivationDelay	4	4
redCrossEfficiencyProbability	1	1
redCrossExhaustionProbability	0.6	0.62

Supplementary questions: tracks for the challenge

- Find dynamics not reachable without redcross agents i.e identify dynamics patterns only reachable if there is a sufficient initial number of antidoters.
- Use ABC algorithm to obtain an *a posteriori* distribution on parameters.
- Calibration with several dataset (not replications): we observed several zombies attacks in similar places (stadium) in which redCross agents were present, and all took the same antidote. Create data, then find parameter values for the antidote from these data.
- Is the decoy technique of antidoters really efficient? Do they repeatedly "fight" the same zombie, preventing it to attack other humans?
- Modelling: modify the alert mechanism, all antidoters are alerted at the same time.