**HCV Prison Model**

**Technical manual and user’s manual**

Prepared by:

Neil Arvin Bretaña, MSc, PhD

Viral Immunology Systems Program

The Kirby Institute

UNSW Sydney

Table of contents

1. Introduction
2. Data structure for an individual
3. Functions of the model
4. Main function
5. Model versions
6. Input file
7. Running the code
8. Reading output files

**Introduction**

This is a user manual for the prison HCV model developed by Dr. Neil Bretaña of the Viral Immunology Systems Program of the Kirby Institute, UNSW Sydney under the supervision of Dr. Richard Gray and Professor Andrew Lloyd. The model is written in C and utilises the GNU scientific library (GSL). The model utilises specific custom R scripts for processing the output.

Throughout this manual, the HCV model is referred to as the *model*, and prisoners simulated in the model are referred to as *individuals*.

For more information on this model and its applications, please refer to the following publications:

**Data structure**

A data structure refers to a format of data values representing an individual during model simulations. The data structure for an individual used by the model is stored in the header file *individual.h*. The variables, descriptions, data types, and values of the individual data structure are listed in Table 1.

The model can be extended to keep track of additional individual characteristics and behaviour by modifying this file.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Description** | **Data type** | **Value** |
| ID | Stores the identification number for an individual | Integer | n |
| gender | Stores the gender of an individual | Integer | 0 (male)  1 (female) |
| age | Stores the age of an individual | Integer | 18 – 100 |
| atsi | Stores the aboriginal/torres strait islander identity of an individual | Integer | 0 (no)  1 (yes) |
| injecting | Stores the injecting behaviour of an individual | Integer | 0 (no)  1 (yes) |
| sharing | Stores the sharing behaviour of an individual | Integer | 0 (no)  1 (yes) |
| injectingFreq | Stores the injecting frequency of an individual | Integer | 0 (not applicable)  1 (less than daily)  2 (daily or more) |
| shaFreq | Stores the sharing frequency of an individual | Integer | 0 (not applicable)  1 (less than daily)  2 (daily or more) |
| location | Stores the prison location of an individual | Integer | 1 (minimum security prison)  2 (medium security prison)  3 (maximum security prison) |
| group | Stores the overall group of an individual | Integer | 0 (IDU+ HCV+ ATSI)  1 (IDU+ HCV+ Non-ATSI)  2 (IDU+ HCV- ATSI Previously exposed)  3 (IDU+ HCV- ATSI Susceptible)  4 (IDU+ HCV- Non-ATSI Previously exposed)  5 (IDU+ HCV- Non-ATSI susceptible)  6 (IDU- HCV+ ATSI)  7 (IDU- HCV+ Non-ATSI)  8 (IDU- HCV- ATSI Previously exposed)  9 (IDU- HCV- ATSI Susceptible)  10 (IDU- HCV- Non-ATSI Previously exposed)  11 (IDU- HCV- Non-ATSI Susceptible) |
| placeInfected | Stores the location of an individual where HCV infection occurred | Integer | 0 (N/A)  1 (minimum security prison)  2 (medium security prison)  3 (maximum security prison) |
| risk | Stores the risk group of an individual | Integer | 0 (non-injecting)  1 (injecting less than daily; opioid; not sharing)  2 (injecting less than daily; opioid; sharing less daily)  3 (injecting less than daily; opioid; sharing daily more)  4 (injecting less than daily; non-opioid; no sharing)  5 (injecting less than daily; non-opioid; sharing less daily)  6 (injecting less than daily; non-opioid; sharing daily or more)  7 (injecting daily or more; opioid; not sharing)  8 (injecting daily or more; opioid; sharing less daily)  9 (injecting daily or more; opioid; sharing daily more)  10 (injecting daily or more; non-opioid; no sharing)  11 (injecting daily or more; non-opioid; sharing less daily)  12 (injecting daily or more; non-opioid; sharing daily or more) |
| timeOfImprisonment | Stores the simulation day of imprisonment of an individual | Integer | n |
| infectionNumber | Stores the number of times an individual has been infected with HCV | Integer | n |
| Metavir | Stores the liver disease stage of an individual | Integer | 0 (liver disease stage 0)  1 (liver disease stage 1)  2 (liver disease stage 2)  3 (liver disease stage 3)  4 (liver disease stage 4)  5 (clearer)  6 (non-infected) |
| cirrhosis | Stores the status of cirrhosis of an individual | Integer | 0 (no)  1 (yes assuming METAVIR 4) |
| viralLoad | Stores the viral load of an individual | Integer | n |
| timeOfInfection | Stores the simulation day when an individual got infected with HCV | Integer | n |
| injOpd | Stores the opioid injecting status of an individual | Integer | 0 (no)  1 (yes) |
| her | Stores the heroin use of an individual | Integer | 0 (no)  1 (yes) |
| herDose | Stores the dosage of heroin use of an individual | Integer | n |
| herDaysMissed | Stores the number of simulation days an individual has missed heroin use | Integer | n |
| met | Stores the methamphetamine use of an individual | Integer | 0 (no)  1 (yes) |
| metDose | Stores the dosage of methamphetamine use of an individual | Integer | n |
| metDaysMissed | Stores the number of simulation days an individual has missed methamphetamine use | Integer | n |
| bup | Stores the buprenorphine use of an individual | Integer | n |
| bupDose | Stores the dosage of buprenorphine use of an individual | Integer | n |
| bupDoseMissed | Stores the number of simulation days an individual has missed buprenorphine use | Integer | n |
| OST | Stores the OST status of an individual | Integer | 0 (not on OST)  1 (OST version from 2006 to 2018)  2 (optimised OST) |
| timeStartOST | Stores the simulation day when an individual commenced OST | Integer | n |
| NSP | Stores the NSP status of an individual | Integer | 0 (not on NSP)  1 (NSP version) |
| timeStartNSP | Stores the simulation day when an individual commenced NSP | Integer | n |
| bleach | Stores the bleach use of an individual | Integer | 0 (no)  1 (yes) |
| DAA | Stores the DAA status of an individual | Integer | 0 (not on NSP)  1 (NSP version) |
| timeStartDAA | Stores the simulation day when an individual commenced DAA | Integer | n |
| everIDU | Stores the IDU history of an individual | Integer | 0 (no history of IDU)  1 (has a history of IDU) |
| \*prevIndiv | Pointer to the previous individual | struct | Address of the previous individual |
| \*nextIndiv | Pointer to the next individual | struct | Address of the next individual |

**Functions of the model**

A function refers to a compartmentalised chunk of code that performs a specific task required in model simulations. The functions of the model are stored in the C file *individual.c*. The function file required the inclusion of the following libraries and header files: *stdio.h*, *stdlib.h*, *string.h*, *math.h*, *gsl/gsl\_randist.h*, and *individual.h*.

generateRand()

The generateRand() function performs the generation of a random integer. This function returns an integer.

draw\_multinom(*gsl\_rng \*r, int nEvents, double probsInput[]*)

The draw\_multinom() function computes a random sample from the multinomial distribution from an underlying distribution *probsInput[nEvents]*. This is done by calling the gsl\_ran\_multinomial() function of the gsl library. This function returns an integer.

distributePop(*int input*)

The distributePop() function draws a random integer from a range (0 to a specified integer *input*). This function returns an integer.

newIndiv(int idGlobal, int HCVentry, int HCVentryAb, int currDay, sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopym gsl\_rng \*\*r, int nEvents, int (\*pLocArray2)[ROWPRIS][COLCTR], int prison, int category)

The newIndiv() function creates an instance for an individual. For individuals created to populate the model from an input file, this function sets individual characteristics based on the parameters sent from the main function. For individuals representing new incarcerations during the simulation, this function sets individual characteristics based on hard-coded probabilities based on literature.

traverse(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The traverse() function goes through all the individuals currently in the simulation from the first one generated to the last one. This function is usually used for printing individuals and their characteristics for testing.

countHCVpris(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countHCVpris() function counts the total number of individuals currently in the simulation that have been infected with HCV inside the prison setting. This function returns an integer.

countOpd(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countOpd() function counts the total number of individuals currently in the simulation that are using opioids. This function returns an integer.

countOpdNotOST(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countOpd() function counts the total number of individuals currently in the simulation that are using opioids but are not on OST. This function returns an integer.

countEverIDU(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countEverIDU() function counts the total number of individuals currently in the simulation that have a history of IDU. This function returns an integer.

countHCVCom(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countHCVCom() function counts the total number of individuals currently in the simulation that have been infected with HCV outside of the prison setting. This function returns an integer.

countHCVantibody(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countHCVantibody() function counts the total number of individuals currently in the simulation that are HCV Ab+. This function returns an integer.

countHCVRNA(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countHCVRNA() function counts the total number of individuals currently in the simulation that are HCV RNA+. This function returns an integer.

countAveStay(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy, int currDay*)

The countAveStay() function counts the average number of simulation days for all individuals in the simulation. This function returns an integer.

count0(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count0() function counts the total number of individuals currently in the simulation that belong to risk group 0. This function returns an integer.

count1(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count1() function counts the total number of individuals currently in the simulation that belong to risk group 1. This function returns an integer.

count2(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count2() function counts the total number of individuals currently in the simulation that belong to risk group 2. This function returns an integer.

count3(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count3() function counts the total number of individuals currently in the simulation that belong to risk group 3. This function returns an integer.

count4(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count4() function counts the total number of individuals currently in the simulation that belong to risk group 4. This function returns an integer.

count5(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count5() function counts the total number of individuals currently in the simulation that belong to risk group 5. This function returns an integer.

count6(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count6() function counts the total number of individuals currently in the simulation that belong to risk group 6. This function returns an integer.

count7(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count7() function counts the total number of individuals currently in the simulation that belong to risk group 7. This function returns an integer.

count8(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count8() function counts the total number of individuals currently in the simulation that belong to risk group 8. This function returns an integer.

count9(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count9() function counts the total number of individuals currently in the simulation that belong to risk group 9. This function returns an integer.

count10(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count10() function counts the total number of individuals currently in the simulation that belong to risk group 10. This function returns an integer.

count11(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count11() function counts the total number of individuals currently in the simulation that belong to risk group 11. This function returns an integer.

count12(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The count12() function counts the total number of individuals currently in the simulation that belong to risk group 12. This function returns an integer.

countR(*sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy*)

The countR() function counts the total number of individuals currently in the simulation that have been infected with HCV more than once (re-infection). This function returns an integer.

probNtrDeath(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probNtrDeath() function draws a double from a uniform distribution as the probability of natural death in prison. This function calls the gsl\_ran\_flat() function of the gsl library and returns a double.

probStopInj(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probStopInj() function draws a double from a uniform distribution as the probability of injecting cessation prison. This function calls the gsl\_ran\_flat() function of the gsl library and returns a double.

probInjLdOpNoSh(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdOpNoSh() function returns the probability of transitioning IDU behaviour into injecting less than daily, while using opioids, but not sharing syringes. This function returns a double.

probInjLdOpShLd(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdOpShLd() function returns the probability of transitioning IDU behaviour into injecting less than daily, while using opioids, and sharing syringes less than daily. This function returns a double.

probInjLdOpShDm(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdOpShDm() function returns the probability of transitioning IDU behaviour into injecting less than daily, while using opioids, and sharing syringes daily or more. This function returns a double. This function is not currently used.

probInjLdNoNoSh(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdNoNoSh() function returns the probability of transitioning IDU behaviour into injecting less than daily, but not using opioids, and not sharing syringes. This function returns a double.

probInjLdNoShLd(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdNoShLd() function returns the probability of transitioning IDU behaviour into injecting less than daily, not using opioids, but sharing syringes less than daily. This function returns a double.

probInjLdNoShDm(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjLdNoShDm() function returns the probability of transitioning IDU behaviour into injecting less than daily, not using opioids, but sharing syringes daily or more. This function returns a double. This function is not currently used.

probInjDmOpNoSh(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmOpNoSh() function returns the probability of transitioning IDU behaviour into injecting daily or more, and using opioids, but not sharing syringes. This function returns a double.

probInjDmOpShLd(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmOpShLd() function returns the probability of transitioning IDU behaviour into injecting daily or more, using opioids, and sharing syringes less than daily. This function returns a double.

probInjDmOpShDm(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmOpShDm() function returns the probability of transitioning IDU behaviour into injecting daily or more, using opioids, and sharing syringes daily or more. This function returns a double. This function is not currently used.

probInjDmNoNoSh(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmNoNoSh() function returns the probability of transitioning IDU behaviour into injecting daily or more, but not using opioids, and not sharing syringes. This function returns a double.

probInjDmNoShLd(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmNoShLd() function returns the probability of transitioning IDU behaviour into injecting daily or more, not using opioids, but sharing syringes less than daily. This function returns a double.

probInjDmNoShDm(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probInjDmNoShDm() function returns the probability of transitioning IDU behaviour into injecting daily or more, not using opioids, but sharing syringes daily or more. This function returns a double. This function is not currently used.

probMoveCom(*gls\_rng \*\*r, sIndiv \*\*pTarget, int currDay*)

The probMoveCom() function returns the probability of release from prison based on an individual’s current location and the current simulation day. This function returns a double.

probMoveP1(*gls\_rng \*\*r, sIndiv \*\*pTarget)*

The probMoveP1() function returns the probability of moving to a minimum security prison based on an individual’s current location. This function returns a double.

probMoveP2(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probMoveP2() function returns the probability of moving to a medium security prison based on an individual’s current location. This function returns a double.

probMoveP3(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probMoveP3() function returns the probability of moving to a maximum security prison based on an individual’s current location. This function returns a double.

probInfectIndiv(*gls\_rng \*\*r, sIndiv \*\*pTarget, int currDay*)

The probInfectIndiv() function returns the probability of infecting an individual assuming a sharing event based on an individual’s characteristics. This function returns a double.

probNtrClear(*gls\_rng \*\*r, sIndiv \*\*pTarget, int currDay*)

The probNtrClear() function returns the probability of an individual for spontaneous clearance of HCV assuming. This function returns a double.

probProgress(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probProgress() function returns the probability of an individual’s liver disease stage to progress assuming infection. This function returns a double.

probHCVdeath(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probHCVdeath() function returns the probability of an individual to die due to HCV-related liver disease. This function returns a double.

probOSTstart(*gls\_rng \*\*r, sIndiv \*\*pTarget, int OST, int nDailyOST, int nOSTDailyCap, int currDay*)

The probOSTstart() function returns the probability of an individual to commence OST. This function returns a double.

probOSTstop(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probOSTstop() function returns the probability of an individual to stop OST. This function returns a double.

probDAAstart(*gls\_rng \*\*r, sIndiv \*\*pTarget, int DAA, int currDay, int nDaily, int nDailyCap*)

The probDAAstart() function returns the probability of an individual to commence DAA treatment. This function returns a double.

probDAAstop(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probDAAstop() function returns the probability of an individual to stop DAA. This function returns a double.

probNSPstart(*gls\_rng \*\*r, sIndiv \*\*pTarget, int NSP, int currDay, int nDailyNSP, int nNSPDailyCap*)

The probNSPstart() function returns the probability of an individual to commence NSP. This function returns a double.

probNSPstop(*gls\_rng \*\*r, sIndiv \*\*pTarget*)

The probNSPstop() function returns the probability of an individual to stop DAA. This function returns a double.

age(*sIndiv \*\*pTargetCopy*)

The age() function increments the age of the individual pointed by *\*\*pTargetCopy* by 1.

removeIndiv(*sIndiv \*\*pTarget, sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy, int (\*pLocArray2)[ROWPRIS][COLCTR])*)

The removeIndiv() function removes the individual pointed by *\*\*pTarget* from the simulation and re-adjusts the linked list of individuals in the simulation so as to not have an gap. It also updates the changes in the population summary *pLocArray2*.

changeRisk(*sIndiv \*\*pTargetCopy, int (\*pLocArray2)[ROWPRIS][COLCTR], int newRisk*)

The changeRisk() function updates the injecting, injecting frequency, sharing, sharing frequency, injecting opioids, and risk variables of the individual pointed by *\*\*pTargetCopy* according to the parameter *newRisk*. It also updates the changes in the population summary pointed by *pLocArray2*.

moveLocation(*sIndiv \*\*pTargetCopy, int (\*pLocArray2)[ROWPRIS][COLCTR], int newLoc*)

The moveRisk() function updates the location of the individual pointed by *\*\*pTargetCopy* according to the parameter *newLoc*. It also updates the changes in the population summary pointed by *pLocArray2*.

infect(gsl\_rng \*\*r, int nEvents, *sIndiv \*\*pTargetCopy, sIndiv \*\*pHeadCopy, sIndiv \*\*pTailCopy, int (\*pLocArray2)[ROWPRIS][COLCTR], int currDay*)

The infect() function initiates an infection event to transmit HCV from the infected individual pointed by *\*\*pTargetCopy* to another individual. First, it goes through the list of individuals to check if there are potential individuals to infect (individuals who are in the same prison location, who are injecting and sharing). For each susceptible individual, this function assigns a probability of infection calculated by the formula:

From the list of susceptible individuals, an individual is selected based on a multinomial distribution on all probabilities of infection by calling the draw\_multinom() function. A per-event probability is then applied based on an exponential distribution by calling the gsl\_ran\_exponential() function to decide if the selected individual is to be infected with HCV. Assuming a success of infection, this function then updates the metavir, infection number, liver disease stage, place of infection, and the time of infection variables of the selected individual. The individual’s group is then updated as well as the population summary pointed by *pLocArray*.

clearHCV(*sIndiv \*\*pTargetCopy, int (\*pLocArray2)[ROWPRIS][COLCTR]*)

The clearHCV() function updates the liver disease stage, place of infection, time of infection, and DAA therapy variables of the individual pointed by *\*\*pTargetCopy*. The individual’s group is then updated as well as the population summary pointed by *pLocArray*.

progressHCV(*sIndiv \*\*pTargetCopy*)

The progressHCV() function updates the liver disease stage variable of the individual pointed by *\*\*pTargetCopy*.

deathHCV(*sIndiv \*\*pTargetCopy*)

The deathHCV() function generates a random integer by calling the generateRand() function. Based on a hard-coded threshold, it refers to the random integer to decide the success of a death due to HCV event. This function returns an integer. This function is not currently used.

startOST(*sIndiv \*\*pTargetCopy, int currDay*)

the startOST() function updates the OST participation variable of the individual pointed by *\*\*pTargetCopy* to signify OST participation. The version of OST applied by this function depends on the *currDay*.

stopOST(*sIndiv \*\*pTargetCopy*)

the stopOST() function updates the OST participation variable of the individual pointed by *\*\*pTargetCopy* to signify dropping out of OST participation.

startNSP(*sIndiv \*\*pTargetCopy, int currDay*)

the startNSP() function updates the NSP participation variable of the individual pointed by *\*\*pTargetCopy* to signify NSP participation.

stopNSP(*sIndiv \*\*pTargetCopy*)

the stopNSP() function updates the NSP participation variable of the individual pointed by *\*\*pTargetCopy* to signify dropping out of NSP participation.

startDAA(*sIndiv \*\*pTargetCopy, int currDay*)

the startDAA() function updates the DAA treatment, and time when DAA treatment started variables of the individual pointed by *\*\*pTargetCopy* to signify DAA treatment.

stopDAA(*sIndiv \*\*pTargetCopy*)

the startDAA() function updates the DAA treatment, and time when DAA treatment started variables of the individual pointed by *\*\*pTargetCopy* to signify cessation of DAA treatment.

clearDAA(*sIndiv \*\*pTargetCopy*)

the clearDAA() function updates the liver disease stage, place of infection, time of infection, and DAA therapy variables of the individual pointed by *\*\*pTargetCopy*. The individual’s group is then updated as well as the population summary pointed by *pLocArray*.

**Main function**

The main function is the code that performs the actual simulation of the model. It is stored in the C file *main.c*. The function file required the inclusion of the following libraries and header files: *stdio.h*, *stdlib.h*, *string.h*, *time.h, math.h*, *gsl/gsl\_randist.h*, and *individual.h*.

The main function follows the algorithm described to perform the simulation:

1. The main function first reads through the input parameters (number of days to simulate *Tmax*, number of prisoners in each group per location, output file name) and prepares the output file upon program initiation.
2. The main function populates the model according to the input.
3. The main function then starts the simulation at simulation time *t*=1. For each time step, the total prison population is first updated by incarcerating *n* individuals from the community into the prison. The number of newly incarcerated individuals *n* is drawn using an exponential distribution on a specified. Newly incarcerated individuals are instantiated by calling the newIndiv function.
4. The main function then goes through every incarcerated individual to apply in-prison events. This version of the model accommodates up to 26 events.
   1. For each individual, an event probability for each of the events is assigned by calling the respective probability assignment function.
   2. Based on each event probability, the occurrence of each event is then set by drawing from a binomial distribution. This sets whether an event will be performed or not for this particular time step in the model.
   3. Events set to occur are then performed in a stochastic order until all occurring events are executed or a death or move out of prison event is executed, in which the individual agent is removed from the simulation.
5. After going through all individuals, the main function prints the state of the model at time *t* into the output file.
6. The main function increments simulation time *t* by 1.
7. The main function repeats steps 3 through to 6 until simulated time *Tmax* is reached.

**Models**

1. Status quo

The base version of the model is called the status quo. In this version of the model, DAA treatment is set to *n*=100 annually from simulation day 2921 (year 2014) to simulation day 3650 (end of year 2016) with a daily cap *n*=1. DAA treatment is then increased to *n*=700 with a daily cap *n*=2 from simulation day 3651 to 4015 (year 2016), and then to *n*=1000 with a daily cap *n*=3 from simulation day 4016 (year 2017) onwards.

1. DAA scenario

Two versions of the model were created for two scenarios of DAA therapy implementation in the prison setting. While both were built on top of the status quo version, one version has DAA treatments reduced to *n*=0 from simulation day 4381 (year 2018) onwards and the other version has DAA treatments doubled to *n*=2000 from simulation day 4381 onwards.

1. NSP

Two versions of the model were created for two scenarios of NSP implementation in the prison setting. While both were built on top of the status quo version, one version has NSP distributed to all PWID with 10% reduction in sharing behavior from simulation day 4381 (year 2018) onwards and the other version has NSP distributed to all PWID with 25% reduction in sharing behavior from simulation day 4381 onwards.

1. OST

Two versions of the model were created for two scenarios of OST implementation in the prison setting. While both were built on top of the status quo version, one version has *n*=1400 OST places at any one time from simulation day 4381 (year 2018) onwards and the other version has OST provided to all opioid users at any one time from simulation day 4381 onwards.

1. Reduction of community HCV prevalence

Two versions of the model were created for two scenarios of OST implementation in the prison setting. While both were built on top of the status quo version, one version assumes an annual reduction in community chronic HCV prevalence based on pessimistic Australian estimates from simulation day 4381 (year 2018) onwards and the other version assumes an annual reduction in community chronic HCV prevalence based on optimistic Australian estimates from simulation day 4381 onwards.

1. DAA (optimistic) + OST (intermediate)

This version of the model combines the effects of optimistic DAA settings and intermediate OST settings.

1. DAA (optimistic) + OST (intermediate) + NSP (intermediate)

This version of the model combines the effects of optimistic DAA settings and intermediate OST and NSP settings.

1. DAA (optimistic) + Reduction of community HCV prevalence (intermediate)

This version of the model combines the effects of optimistic DAA settings and intermediate reduction of community HCV prevalence settings.

1. DAA (optimistic) + Reduction of community HCV prevalence (intermediate) + OST (intermediate)

This version of the model combines the effects of optimistic DAA settings and intermediate OST and reduction of community HCV prevalence settings.

1. DAA (optimistic) + Reduction of community HCV prevalence (intermediate) + OST (intermediate) + NSP (intermediate)

This version of the model combines the effects of optimistic DAA settings and intermediate OST, NSP, and reduction of community HCV prevalence settings.

**Input file**

Two types of input files: ***b*** - basic and ***d*** - detailed.

Basic input file

Basic requires only the total population in the community and each prison security setting. The model distributes this number randomly into sub-populations.

Detailed input file

Basic requires the number for each sub-population in the community and each prison security setting.

Running the model:

1. Compile individual.c

*gcc -c individual.c -lgsl -lgslcblas -lm*

1. Compile main.c

*gcc -c main.c -lgsl -lgslcblas -lm*

1. Instantiate the model

*gcc -Werror -Wall main.o individual.o -lgsl -lgslcblas -o <modelName.exe>*

1. Run the model

There are two types of model run depending on the input

* 1. Basic –

*./<modelName.exe> ./<inputFileName.txt> ./<outputFileName.txt>* ***b***

* 1. Detailed –

*./<modelName.exe> ./<inputFileName.txt> ./<outputFileName.txt>* ***d***

Reading output files

The model comes with R scripts to summarize the output:

1. PopulationCalibration2.R

This script is used to check the prison population and HCV incidence calibration of the model.

1. Projections3.R

This script is used to summarize the output of the DAA version of the model.

1. Projections3NP.R

This script is used to summarize the output of the NSP version of the model.

1. Projections3OST.R

This script is used to summarize the output of the OST version of the model.

1. ProjectionsRedux.R

This script is used to summarize the output of the reduction version of the model.

1. ProjectionsCombined2.R

This script is used to summarize the output of any combined version of the model.