SCIENTIFIC REGISTRY OF TRANSPLANT RECIPIENTS

Liver Simulated Allocation Model

Version 2019

User's Guide

Last Updated: May 16, 2019

This work is supported by the Scientific Registry of Transplant Recipients (contract number HHSH250201000018C), with funding and oversight by the Health Resources and Services Administration, an agency of the U.S. Department of Health and Human Services. The Scientific Registry of Transplant Recipients is operated by the Minneapolis Medical Research Foundation, Chronic Disease Research Group.

Contributors to SAM refinement and redesign

Minneapolis Medical Research Foundation
Yi Peng
Eugene Shteyn
Xinyue Wang
Josh Pyke

United States Naval Academy
Sommer Gentry

Johns Hopkins University
Dorry Segev

University of Minnesota
Diwakar Gupta
Hao-Wei Chen
Zhi Zhang
Sang Phil Kim

Contributors to initial SAM development

Arbor Research Collaborative for Health
David Dickinson
Shannon Li
Keith McCullough
Robert M. Merion, MD
Friedrich K. Port, MD
Ann M. Rodgers
Caroline Shevrin
Robert A. Wolfe

University of Michigan
Cora Brunton
Susan Murray

Altarum Institute
David Thompson
Larry Waisanen

Table of Contents

1 Getting started

1.1 Purpose of the Program

The purpose of the liver simulated allocation model (LSAM) is to simulate the allocation of livers to candidates on the Organ Procurement and Transplantation Network (OPTN) waiting list. LSAM is a computer simulation program developed by the Scientific Registry of Transplant Recipients (SRTR).

The program has been designed to support studies of alternative organ allocation policies. It can use a variety of allocation rules to determine how a series of organs would be allocated to a list of potential recipients under each of the rules considered. The allocation process involves some random components reflecting the uncertainty of acceptance decisions when an organ is offered to a potential recipient, and reflecting the unpredictable life expectancy related to undergoing or not undergoing transplant. To account for such random variation, the program can repeatedly simulate organ allocation with the same set of allocation rules, candidate lists, and organs, to determine what happens on average.

This guide describes how program users can create or edit an Allocation Run package, which consists of a named collection of input files, and then instruct the program to simulate the organ allocation process. Before LSAM can be used, several data files must be created including candidates, candidate histories, and organ characteristics. Default versions of these files are created when LSAM is installed.

1.2 Installing the Software

You must be running Microsoft Windows XP, Windows Vista, or Windows 7 to install LSAM. You may need administrator rights to install LSAM. The software for the current version of the LSAM program is available on CD from SRTR.

1. Insert the software CD into your CD drive.
2. Using Windows Explorer, select your CD drive and double-click on LSAM 2014 Installer.exe. The Setup Wizard will start.
3. The Setup Wizard will prompt you to select Next to continue installation.
4. The Setup Wizard will show the recommended default folder for installing the program, which can be changed. Choose the default folder or another folder, then select Next.

5. The Setup Wizard will ask if you want to create a start menu folder. Indicate your preference, then select Next.
6. The Setup Wizard will ask if you want to create a desktop icon or a Quick Launch icon. Indicate your preference, the select Next.
7. The Setup Wizard gives a summary of your installation settings. Select Install to continue.
8. Software program files will be copied to the appropriate folders. A folder called \SRTR\LSAM 2014\LSAM will be created on your hard disk, and sample data files will be created in \SRTR\LSAM 2014\LSAM\Input\.
9. Select Finish to exit the Setup Wizard Completed screen. The process should take less than 2 minutes. Remove the CD from your drive and store in CD case.

1.3 Deleting the Software

On the taskbar at the bottom left of your screen, click the Start button to bring up the Start menu. Choose Settings, then Control Panel. Double-click on the icon Add/Remove Programs. Then highlight LSAM version 2014, click Add/Remove, and follow the on-screen instructions. (In Windows 7, the Control Panel icon is titled Programs and Features, and the removal command is Uninstall.)

You may also select the uninstall option from the Start menu folder if you chose to create one during installation.

1.4 Moving the Software

First delete the software as described above, then reinstall it to a new location.

1.5 Starting the Program

The Setup Wizard creates two shortcuts you can use to start the LSAM program:

1. On the taskbar at the bottom-left of your screen, click the Start button; the Start menu appears. Choose Programs, SRTR, then LSAM.
2. Click on the LSAM icon on your desktop if you chose to create one during installation.

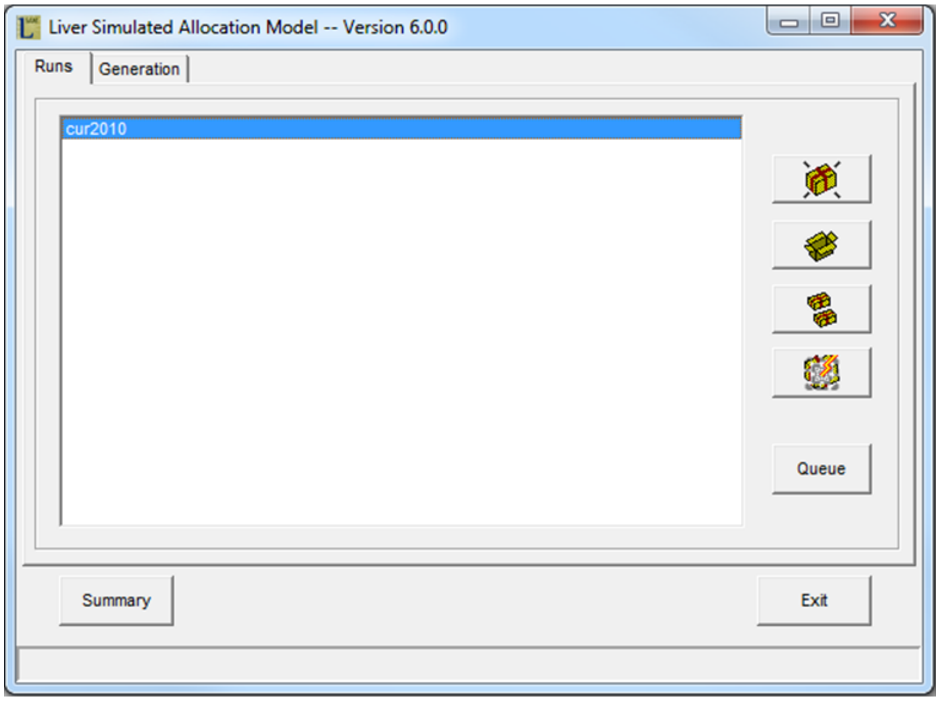
You may also open the folder the program was installed in and click on the LSAM icon.

1.6 Setting Input Parameters

The information that LSAM uses must be prepared according to very detailed specifications. Descriptions of the required files appear in Chapters 6 and 7. After preparing these files, you can use them with LSAM.

The collection of program parameters and input files defining an LSAM run is referred to as an Allocation Run package. The LSAM user interface prompts users to identify the files that contain the input data (organ and candidate characteristics and other information related to organ allocation, acceptance, and posttransplant survival) that define a single Allocation Run. The interface also supports direct input of selected aspects of the allocation rules in dialogue boxes.

The next few pages describe the steps needed to create, run, and edit an Allocation Run package. Below is the first screen of the program. The Runs tab contains previously defined Allocation Run packages (the Generation tab is not currently used in LSAM).Click on the wrapped package button on the right of the screen to create a new Allocation Run package.



cur2010

If any Allocation Run packages have been defined previously, the program will list them on this screen. Select an Allocation Run package, then select the open box button. This will take you to the Allocation Run Specification screen, where you may run that package or edit its definition.

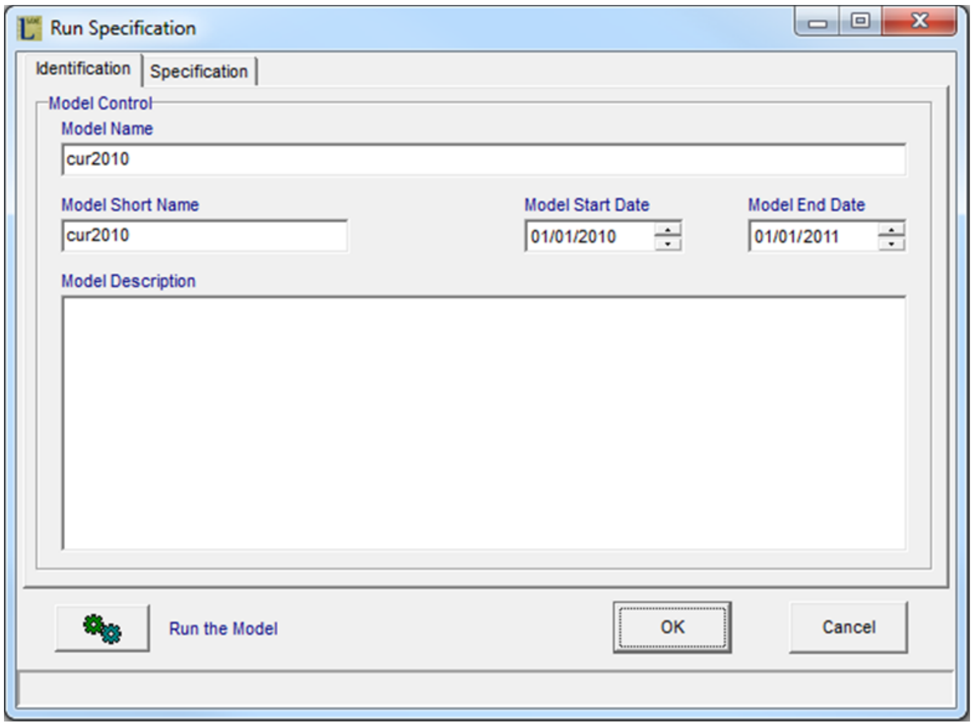
Other buttons allow you to duplicate an Allocation Run package (two wrapped packages), delete an Allocation Run package (lightning bolt), run the Queue (section 1.7), or create a Summary file (section 1.8).

1.6.1 Run Specification

The Run Specification screen contains two tabs, which are used to select the files from which the program will read its input data for this Allocation Run Specification.

Identification
On the Identification tab of the Run Specification screen, set the:

- **Model Name,** listed on the main screen and used to identify the Model Allocation Run. Use abbreviations that describe the model parameters, i.e., "CY01RegShare" instead of "Model02."
- **Model Short Name,** used to assign output file names by appending the short name of the Allocation Run to the output file names. For example, the file "CY01RegShareMatch.out" contains the match run results for an allocation run with Model Short Name "CY01RegShare."
- **Model Start Date,** the beginning date of this Model Allocation Run.
- **Model End Date,** the ending date of this Model Allocation Run.


Run Specification

Identification | Specification

Model Control

Model Name: cur2010

Model Short Name: cur2010

Model Start Date: 01/01/2010

Model End Date: 01/01/2011

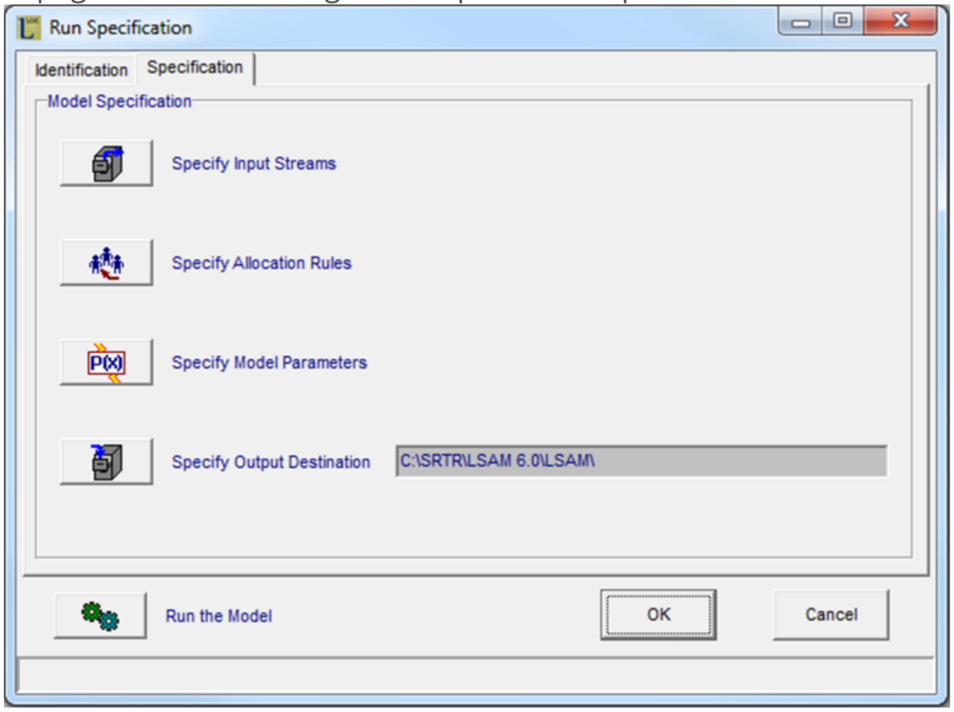
Model Description:

Run the Model | OK | Cancel

Select the **Specification** tab to continue.

Specification

The next few pages describe setting model specification parameters.


Run Specification

Identification | Specification

Model Specification

Specify Input Streams

Specify Allocation Rules

Specify Model Parameters

Specify Output Destination: C:\SRTR\LSAM 6.0\LSAM\

Run the Model | OK | Cancel

Select the **Specify Input Streams** button to continue.

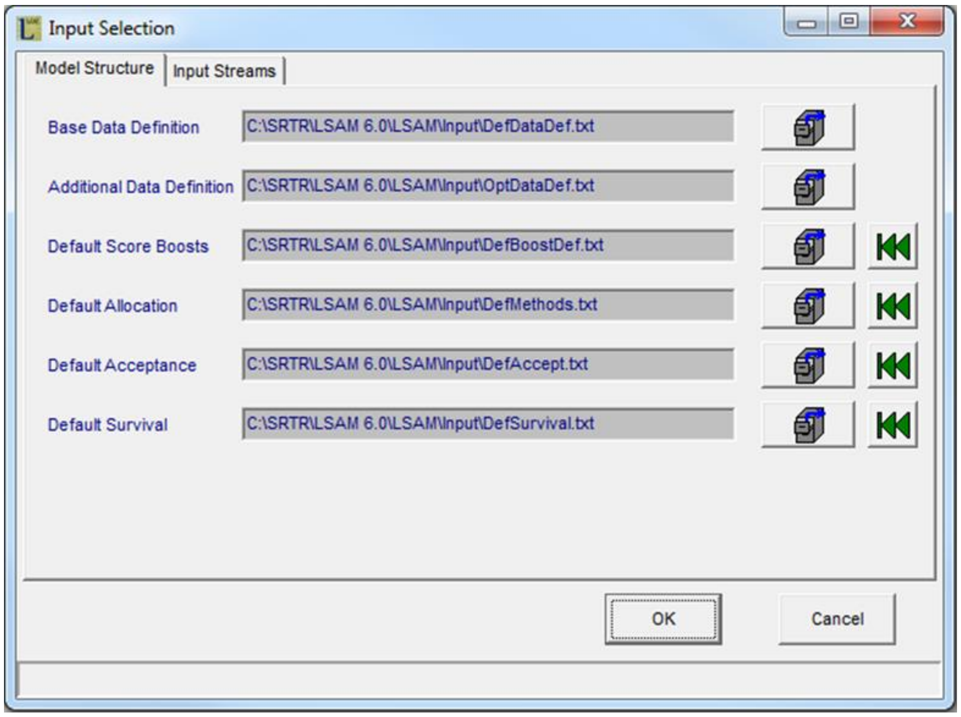
1.6.2 Input Selection

The Input Selection screen contains two tabs, which are used to select the files from which the program will read its input data for this Allocation Run.

Model Structure

The Model Structure tab allows you to specify where the Base and Additional Data Definition, Score Boost, Allocation, Acceptance, and Survival files are located. See Chapter 6, Input File Specifications, for more detailed information about each input file.

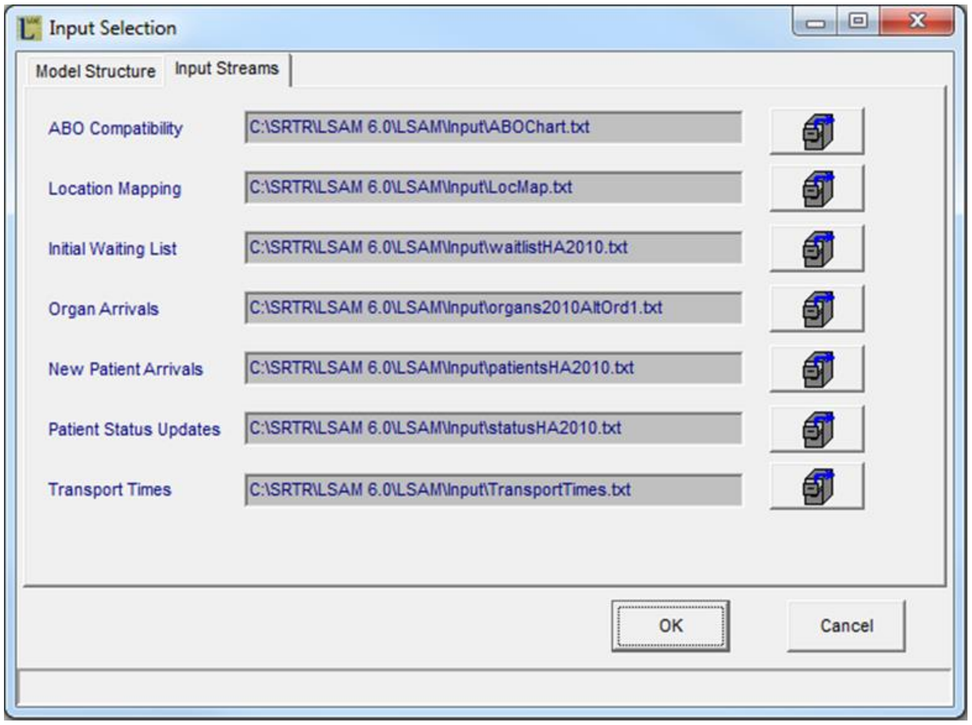
To change a data source from its default value, select the filing cabinet button on the right. This will bring up a standard Windows file selection dialog.



Base Data Definition	C:\SRTR\LSAM 6.0\LSAM\Input\DefDataDef.txt
Additional Data Definition	C:\SRTR\LSAM 6.0\LSAM\Input\OptDataDef.txt
Default Score Boosts	C:\SRTR\LSAM 6.0\LSAM\Input\DefBoostDef.txt
Default Allocation	C:\SRTR\LSAM 6.0\LSAM\Input\DefMethods.txt
Default Acceptance	C:\SRTR\LSAM 6.0\LSAM\Input\DefAccept.txt
Default Survival	C:\SRTR\LSAM 6.0\LSAM\Input\DefSurvival.txt

Input Streams

The Input Streams tab allows you to specify where the ABO Compatibility, Location Mapping, Initial Waiting List, Organ Arrivals, New Patient Arrivals, Patient Status Updates, and Transport Times files are located. To change a location, click on the filing cabinet icon to the right of the file name. See Chapter 6, Input File Specifications, for more detailed information about each input file.



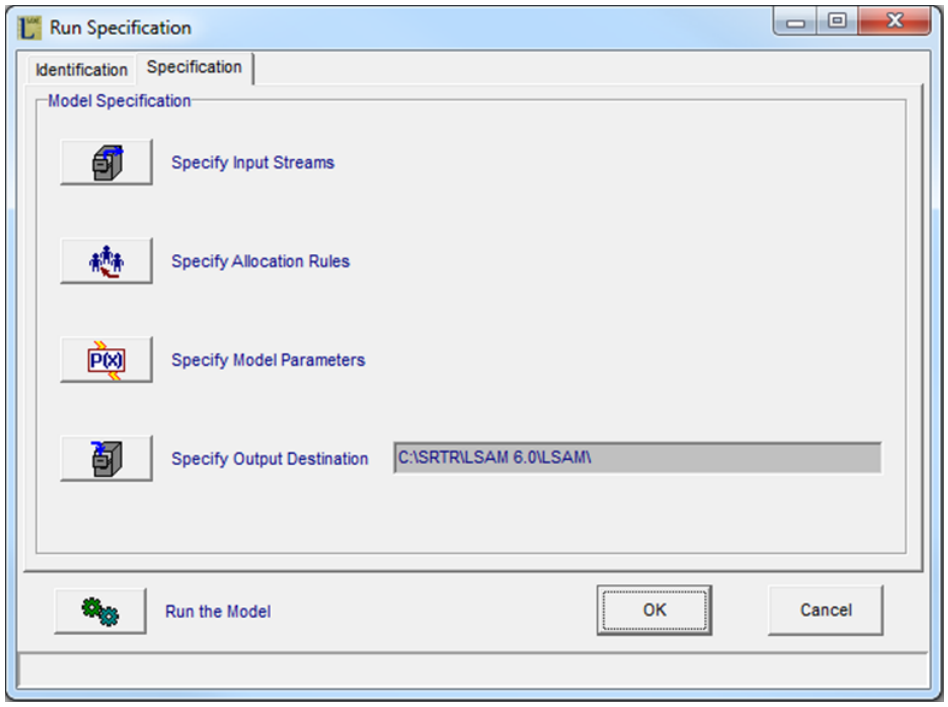
ABO Compatibility	C:\SRTR\LSAM 6.0\LSAM\Input\ABOChart.txt
Location Mapping	C:\SRTR\LSAM 6.0\LSAM\Input\LocMap.txt
Initial Waiting List	C:\SRTR\LSAM 6.0\LSAM\Input\waitlistHA2010.txt
Organ Arrivals	C:\SRTR\LSAM 6.0\LSAM\Input\organs2010AltOrd1.txt
New Patient Arrivals	C:\SRTR\LSAM 6.0\LSAM\Input\patientsHA2010.txt
Patient Status Updates	C:\SRTR\LSAM 6.0\LSAM\Input\statusHA2010.txt
Transport Times	C:\SRTR\LSAM 6.0\LSAM\Input\TransportTimes.txt

Multiple sample organ arrival files have been provided with this installation. The same organs become available in each of the files, but their arrival dates and times have been shuffled in each of the files to more closely simulate the randomness with which donor organs become available. Users may choose to specify that LSAM use a different ordering of organ arrivals for each iteration by choosing the organ arrival file with the phrase "AltOrd1" in its name on this screen. LSAM will then use AltOrd1 for its first iteration, AltOrd2 for its second iteration, etc.

1.6.3 Allocation Rules

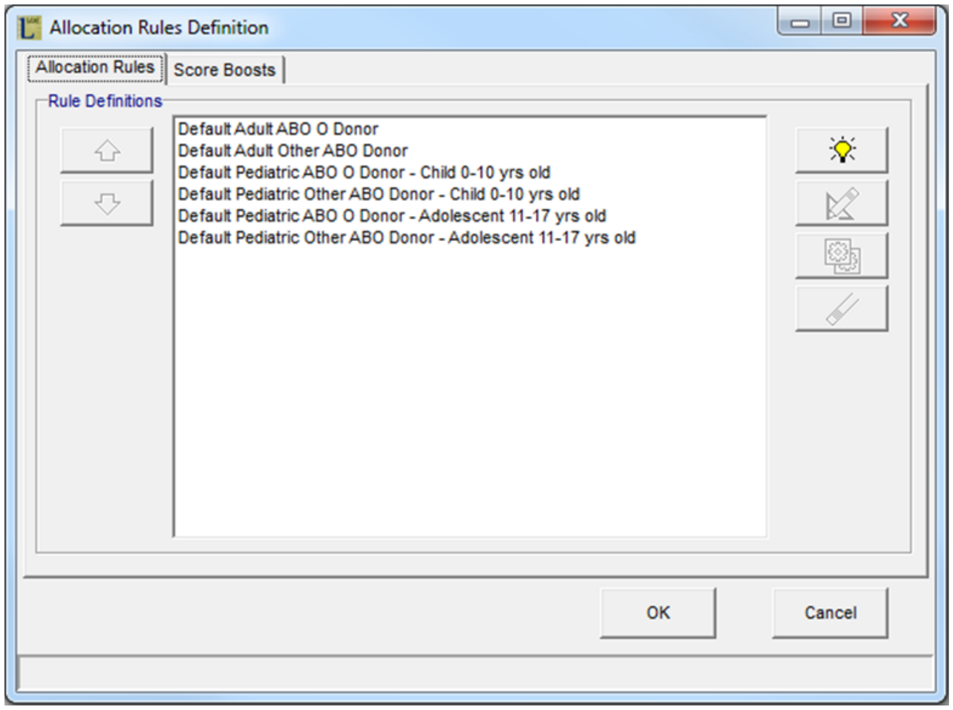
Run Specification: Allocation Rules

LSAM provides interfaces for adjusting allocation rules and other model parameters. All of these settings are also specified in the default input files, as discussed in Chapter 6, Input File Specifications. The graphical interfaces may be convenient for small adjustments, but using a text editor to manually copy and edit the input files is recommended for large-scale changes.



C:\SRTR\LSAM 6.0\LSAM\

To explore the first graphical parameter interface, select the **Specify Allocation Rules** button.



Default Adult ABO O Donor
Default Adult Other ABO Donor
Default Pediatric ABO O Donor - Child 0-10 yrs old
Default Pediatric Other ABO Donor - Child 0-10 yrs old
Default Pediatric ABO O Donor - Adolescent 11-17 yrs old
Default Pediatric Other ABO Donor - Adolescent 11-17 yrs old

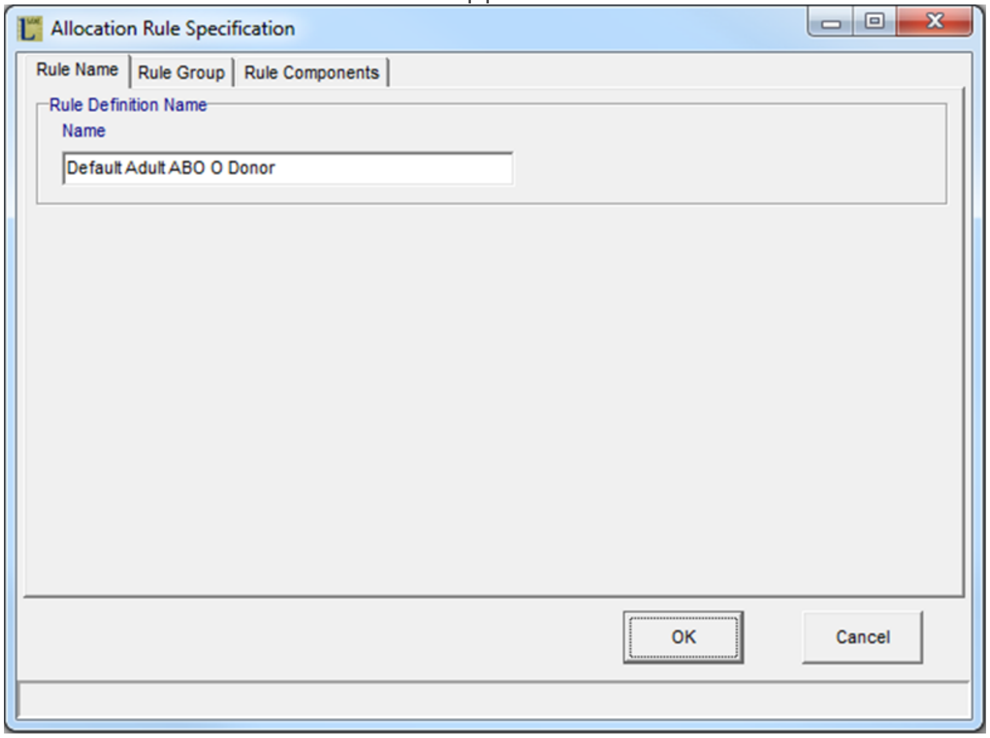
Highlight one of the Allocation Rules, such as Default Adult Donor, and select the edit button (blue pencil) to continue.

Allocation Rule Specification

The Allocation Rule Specification screen contains three tabs that are used to define the rules for organ allocation.

Rule Name

Use the Rule Name tab to specify the name of this allocation rule. The name should describe the set of donors to whom this rule applies.



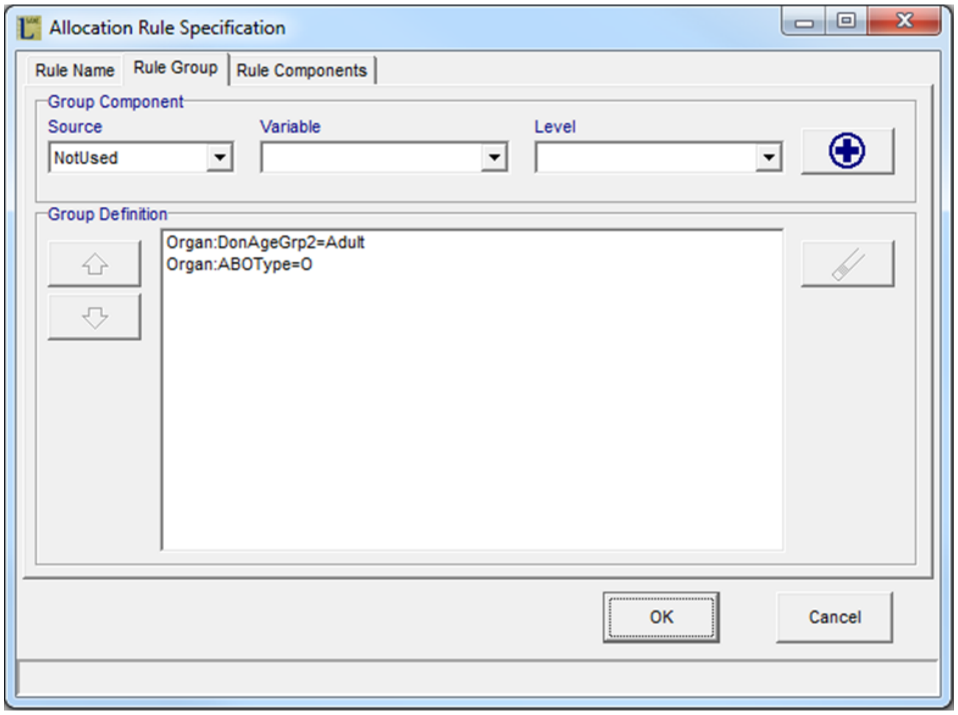
Rule Definition Name
Name: Default Adult ABO O Donor

Rule Group

Use the Rule Group tab to define the group of donors to whom this allocation rule applies.

The source must be the organ record. Choose a variable from the drop-down list. This list contains all fields whose usage has been defined as “category” in the default or optional data definitions files. Choose a level for this variable from the level drop-down list. These, too, have been defined in the default or optional data definitions files. See Chapter 6, Input File Specifications, for more detailed information about the data definitions files.

You may combine any number of rules using the plus sign button at the right to narrow the donor list to which this rule applies. An organ donor must meet all of the group definitions for this allocation rule to apply.

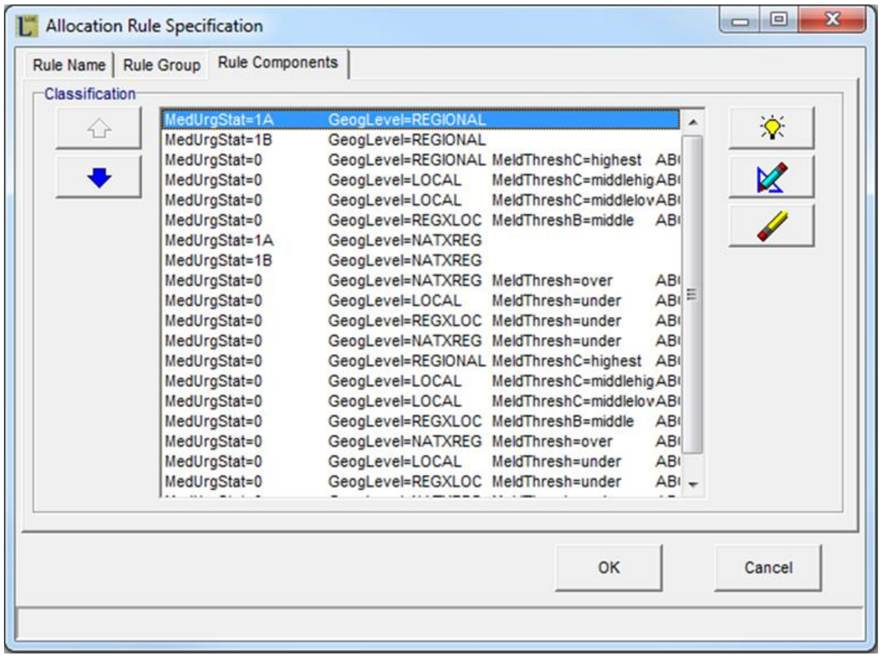


Source: NotUsed

Organ:DonAgeGrp2=Adult
Organ:ABOType=O

Rule Components

The Rule Components tab displays the ordered list of patient categories for the selected allocation rule. To change the ordering, select a category and use the up- and down-arrow keys on the left to promote or demote the category. To create a new category, select the create button (light bulb) on the right side. To edit a category, select the category and then select the edit button (blue pencil) on the right side. To delete a category, select it and then select the erase button (pencil eraser). See Chapter 3 for more detailed information about the allocation methods.



```
MedUrgStat=1A    GeogLevel=REGIONAL
MedUrgStat=1B    GeogLevel=REGIONAL
MedUrgStat=0     GeogLevel=REGIONAL MeldThreshC=highest   AB
MedUrgStat=0     GeogLevel=LOCAL    MeldThreshC=middlehigAB
MedUrgStat=0     GeogLevel=LOCAL    MeldThreshC=middlelovAB
MedUrgStat=0     GeogLevel=REGXLOC  MeldThreshB=middle    AB
MedUrgStat=1A    GeogLevel=NATXREG
MedUrgStat=1B    GeogLevel=NATXREG
MedUrgStat=0     GeogLevel=NATXREG  MeldThresh=over       AB
MedUrgStat=0     GeogLevel=LOCAL    MeldThresh=under      AB
MedUrgStat=0     GeogLevel=REGXLOC  MeldThresh=under      AB
MedUrgStat=0     GeogLevel=NATXREG  MeldThresh=under      AB
MedUrgStat=0     GeogLevel=REGIONAL MeldThreshC=highest   AB
MedUrgStat=0     GeogLevel=LOCAL    MeldThreshC=middlehigAB
MedUrgStat=0     GeogLevel=LOCAL    MeldThreshC=middlelovAB
MedUrgStat=0     GeogLevel=REGXLOC  MeldThreshB=middle    AB
MedUrgStat=0     GeogLevel=NATXREG  MeldThresh=over       AB
MedUrgStat=0     GeogLevel=LOCAL    MeldThresh=under      AB
MedUrgStat=0     GeogLevel=REGXLOC  MeldThresh=under      AB
```

Highlight one of the rule components, and then select the edit button to continue.

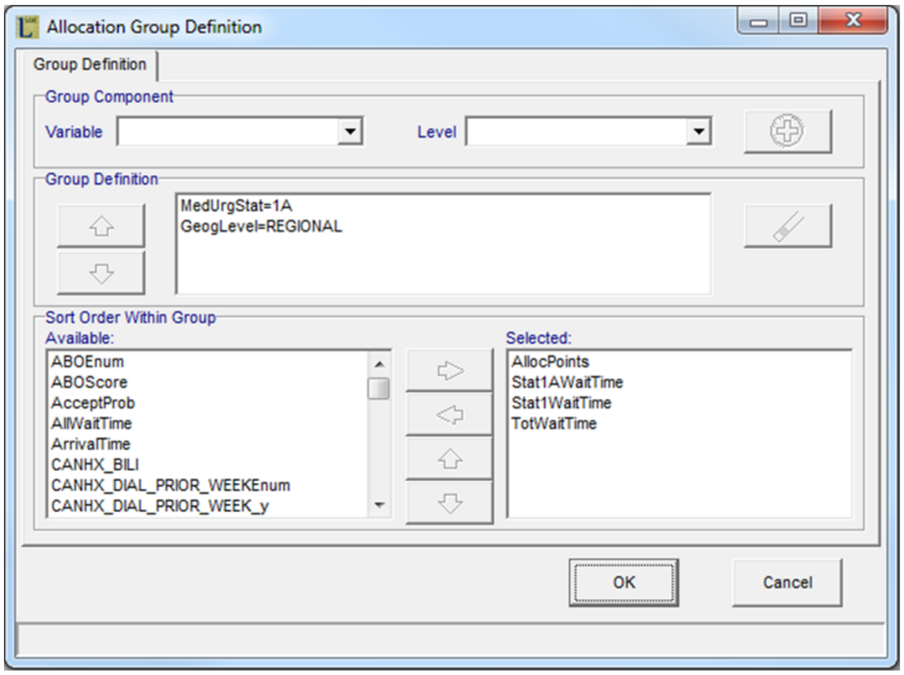
<u>Allocation Group Definition</u>

Group Definition

The Group Definition tab allows users to define the patient categories. The drop-down list of variables contains all patient variables whose usage has been defined as “category” in the default or optional data definition files. Choose a variable from the list, then choose a level for this variable from the level drop-down list. Rearranging the group definitions list using the up- and down-arrows to the left changes the order in which the criteria are checked. Ordering the more restrictive criteria closer to the top of the list may speed processing. These variables have been defined in the default or optional data definitions files. See Chapter 6, Input File Specifications, for more detailed information about the data definitions files.

Within each patient subgroup, you may want to order the patients by different criteria. Use the Sort Order Within Group section of the screen for subgroup ordering. To add a patient characteristic to the list, highlight that characteristic in the available list, then select the right-arrow button. To remove a characteristic, highlight it in the selected list, then select the left-arrow button. To promote or demote a characteristic, highlight it in the selected list, then select the up- or down-arrow key.

The next section describes how to boost the value of any of the subgroup fields for patient and organ combinations with certain characteristics.



MedUrgStat=1A
GeogLevel=REGIONAL

Available: ABOEnum, ABOScore, AcceptProb, AllWaitTime, ArrivalTime, CANHX_BILI, CANHX_DIAL_PRIOR_WEEKEnum, CANHX_DIAL_PRIOR_WEEK_y

Selected: AllocPoints, Stat1AWaitTime, Stat1WaitTime, TotWaitTime

From the Allocation Group Definition screen, select the OK or Cancel button to return to the Allocation Rule Specification screen. Select the OK or Cancel button from there to return to the Allocation Rules Definition screen.

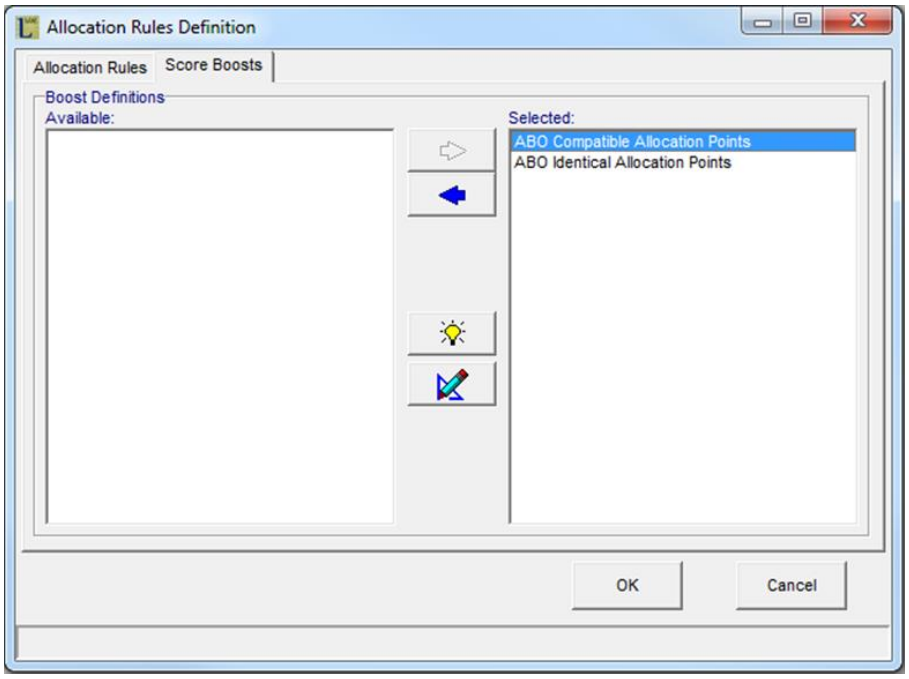
Next, select the **Score Boosts** tab from the Allocation Rules Definition screen.

Allocation Rules Definition

Score Boosts

Use the Score Boosts tab to increase the value of certain patient fields. Those fields may then be used for subgroup sorting in the allocation process. Refer to the previous section, Allocation Group Definition, for information on how to use those fields in the allocation process.

Here you will define the patient and organ characteristics that must be present, and the calculation for the points those patients will gain. This section of the interface corresponds to the DefBoostDef.txt file.


Allocation Rules Definition
Allocation Rules
Score Boosts
Boost Definitions
Available:
Selected:
ABO Compatible Allocation Points
ABO Identical Allocation Points
OK
Cancel

Highlight a score boost from either the available list or the selected list and select the edit button (blue pencil).

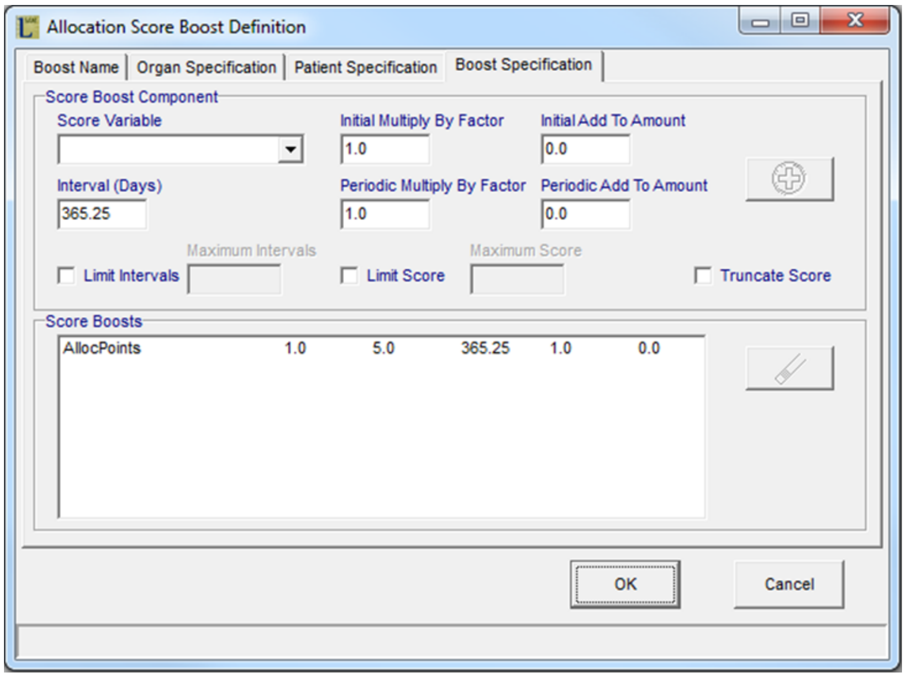
It is also possible to enter a linear equation as a score boost. Linear equation score boosts cannot be entered, viewed, or modified using the LSAM input panels, but must be entered using the DefBoostDef.txt input file. Refer to the Allocation Score Boost Definition section of Chapter 6, Input File Specification, for more information.

The Boost Name, Organ Specification, and Patient Specification tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Refer to Allocation Rule Specification, page 12, for more information on those two tabs.

Select the **Boost Specification** tab to continue.

Boost Specification

The Boost Specification tab is used to specify which patient characteristic(s) will be increased, and by how much. You may multiply the field by a factor and/or add an amount to the field. In addition, you can specify that the field increase at a given interval if the patients remain on the waiting list. You may also set upper limits on the number of intervals or on the score itself, and you may truncate a calculated score.



Allocation Score Boost Definition

Boost Name | Organ Specification | Patient Specification | Boost Specification

Score Boost Component

Score Variable:

Initial Multiply By Factor: 1.0

Initial Add To Amount: 0.0

Interval (Days): 365.25

Periodic Multiply By Factor: 1.0

Periodic Add To Amount: 0.0

Maximum Intervals:

☐ Limit Intervals

Maximum Score:

☐ Limit Score

☐ Truncate Score

Score Boosts

```
AllocPoints        1.0    5.0    365.25    1.0    0.0
```

We have completed our allocation rules definitions. Next, we will explore the model parameters you may specify for random number generation, acceptance probability, and post-graft survival.

Select the OK or Cancel button to return to the Allocation Rules Definition screen, then select OK or Cancel again to return to the Run Specification screen. Finally choose the **Specify Model Parameters** button to continue.

1.6.4 Model Parameter Specification

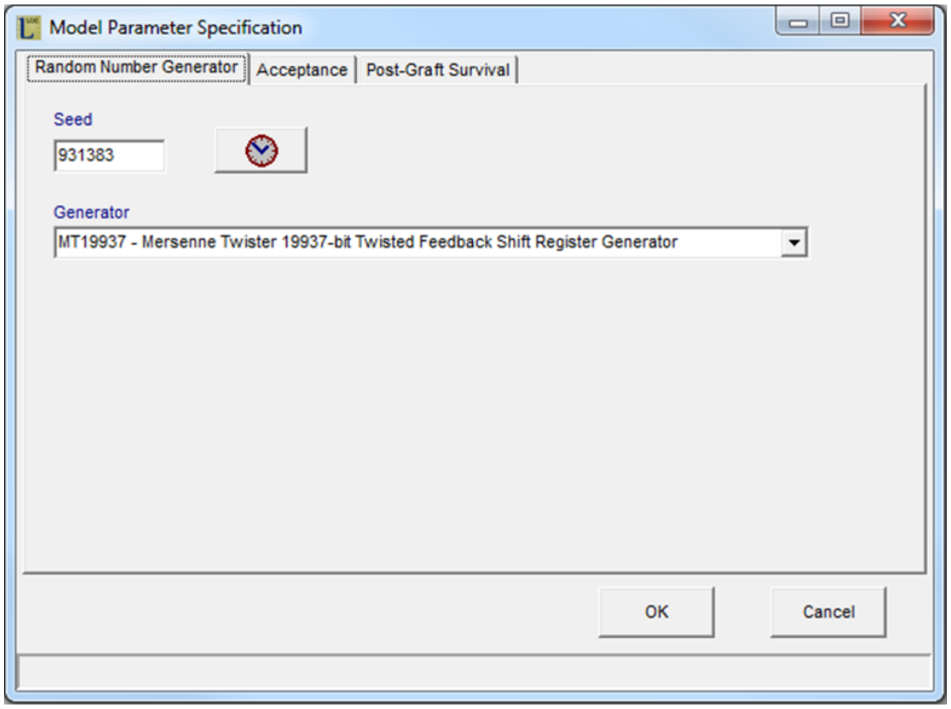
Random Number Generator

This list of choices allows you to choose how the program will generate random numbers for allocation. Random numbers are used in several places in this simulation; for example, if the probability of organ acceptance for a certain organ/recipient pair is calculated as 70%, the program will use random numbers from this generator to decide whether or not that recipient accepts that particular organ. For most purposes, the choice of generator does not matter.

The default generator, MT19937, is preferred, although the stream of random numbers produced by any of these generators should be adequate.

If you wish to duplicate the results of a run exactly, you must provide the same seed and the same generator.

You can accept the existing seed, enter a new seed in the window, or select the clock button and the program will select a new seed. See Chapter 2.1 for information on how the model uses the random number generator.


Model Parameter Specification

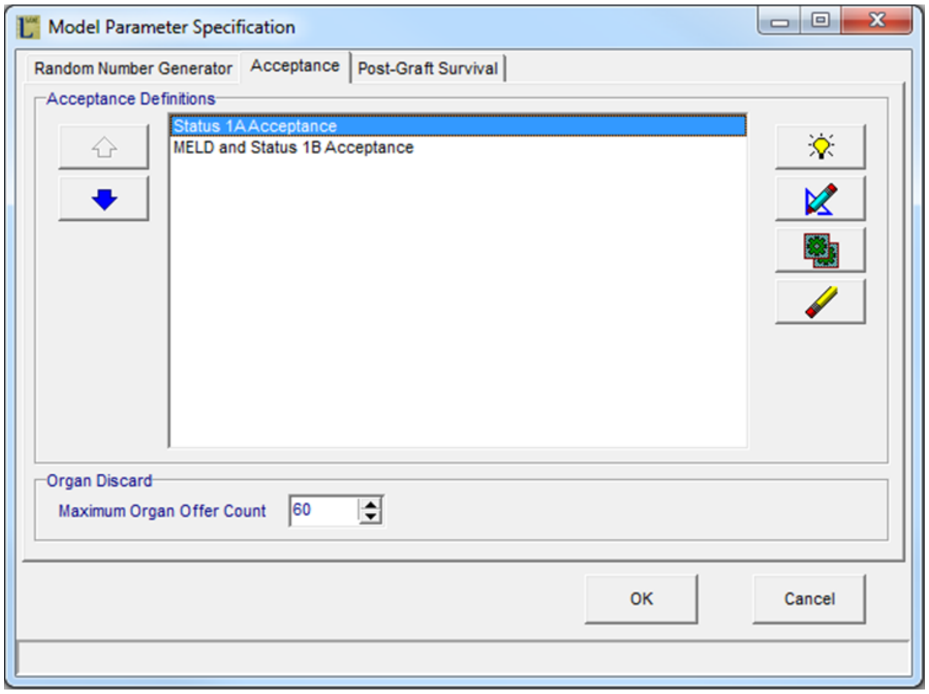
Random Number Generator | Acceptance | Post-Graft Survival

Seed: 931383

Generator: MT19937 - Mersenne Twister 19937-bit Twisted Feedback Shift Register Generator

Acceptance

Acceptance probability is determined by model input. The mechanism is described in detail below. You may also limit the number of times a given organ is offered before it is discarded. Lowering this limit will speed processing and result in more discarded organs. The acceptance model you use should reflect the offer limit you set. The default acceptance model parameters are also available in DefAccept.txt.



Status 1A Acceptance
MELD and Status 1B Acceptance

Maximum Organ Offer Count 60

Highlight one of the acceptance definitions and select the edit button.

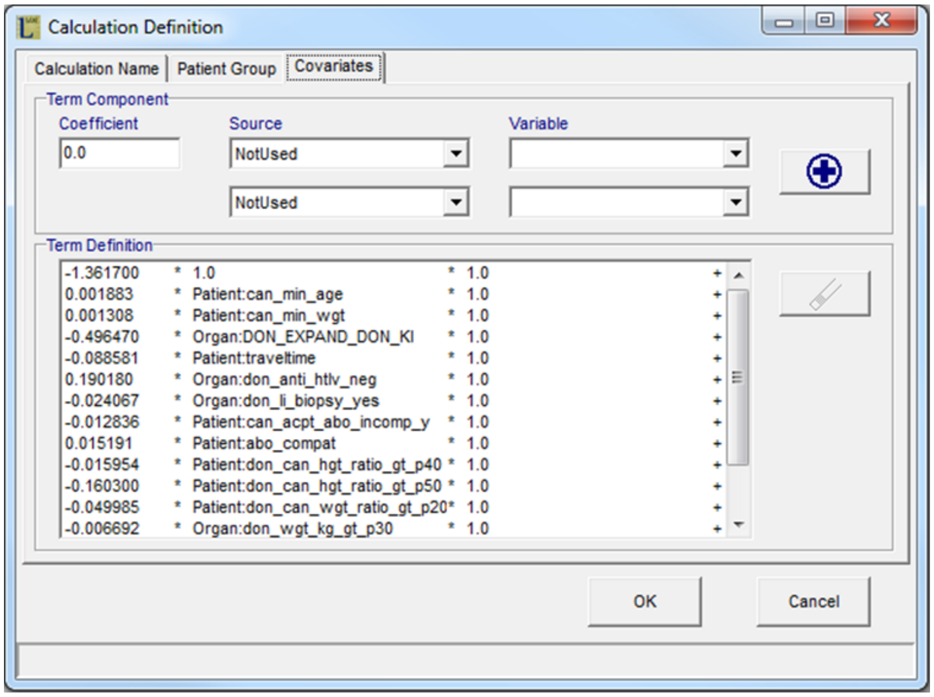
The Calculation Name and Patient Group tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Refer to Allocation Rule Specification, page 12, for more information on those two tabs.

Select the **Covariates** tab to continue.

Calculation Definition (Acceptance)

Covariates

Here you may define an equation to represent organ acceptance probability. Values that may be used in this equation are scalar variables (coefficients), characteristics of the organ or donor, characteristics of the potential organ recipient, and/or values calculated from characteristics of the specific organ and patient combination under consideration. For each organ offered, this equation will be solved by the model, and the resulting value, β, transformed using an inverse logit transformation (exp(β) / 1+exp(β)). That value will then be compared to a random number between 0 and 1. If it is greater than the random number, then the organ is accepted; otherwise, it is refused.



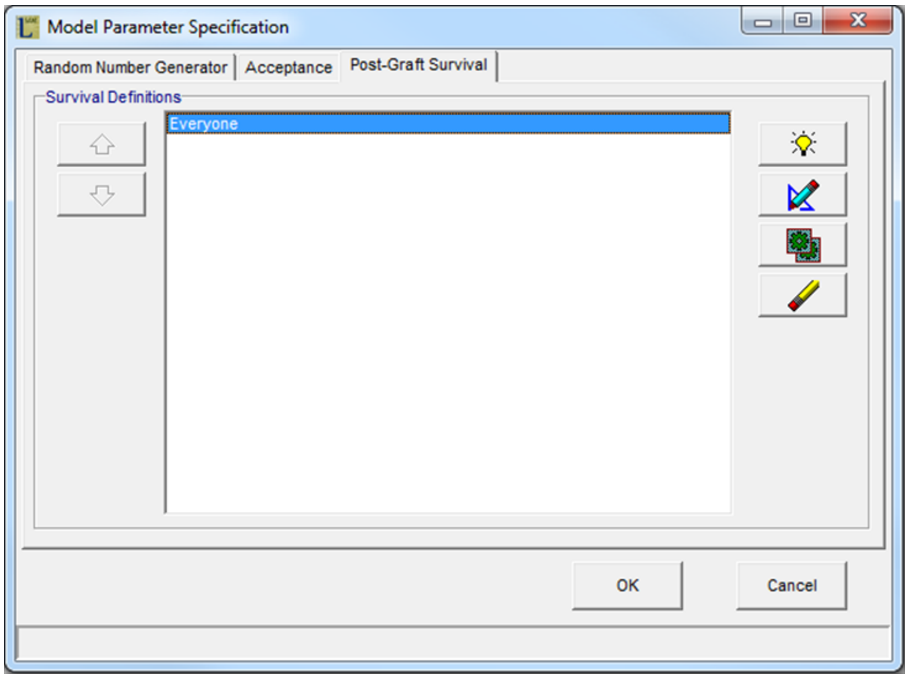
Coefficient: 0.0
Source: NotUsed
Source: NotUsed

```
-1.361700    *  1.0                            *  1.0   +
0.001883     *  Patient:can_min_age            *  1.0   +
0.001308     *  Patient:can_min_wgt            *  1.0   +
-0.496470    *  Organ:DON_EXPAND_DON_KI        *  1.0   +
-0.088581    *  Patient:traveltime             *  1.0   +
0.190180     *  Organ:don_anti_htlv_neg        *  1.0   +
-0.024067    *  Organ:don_li_biopsy_yes        *  1.0   +
-0.012836    *  Patient:can_acpt_abo_incomp_y  *  1.0   +
0.015191     *  Patient:abo_compat             *  1.0   +
-0.015954    *  Patient:don_can_hgt_ratio_gt_p40 *  1.0   +
-0.160300    *  Patient:don_can_hgt_ratio_gt_p50 *  1.0   +
-0.049985    *  Patient:don_can_wgt_ratio_gt_p20*  1.0   +
-0.006692    *  Organ:don_wgt_kg_gt_p30        *  1.0   +
```

Select the OK or Cancel button to return to the Model Parameter Specification screen, then choose the **Post-Graft Survival** tab to continue.

Model Parameter Specification: Post-Graft Survival

Post-graft survival time and intermediate patient status changes are determined by model input. The mechanism for specifying these post-graft survival parameters is described in detail below. The default survival model is also specified in the file DefSurvival.txt.



Everyone

Highlight one of the survival definitions and select the edit button.

The Calculation Name and Patient Group tabs on the next screen are similar to the Rule Name and Rule Group tabs in the Allocation Rule Specification. Refer to the Allocation Rule Specification, page 12, for more information on those two tabs.

Select the **Covariates** tab to continue the post-graft survival definitions.

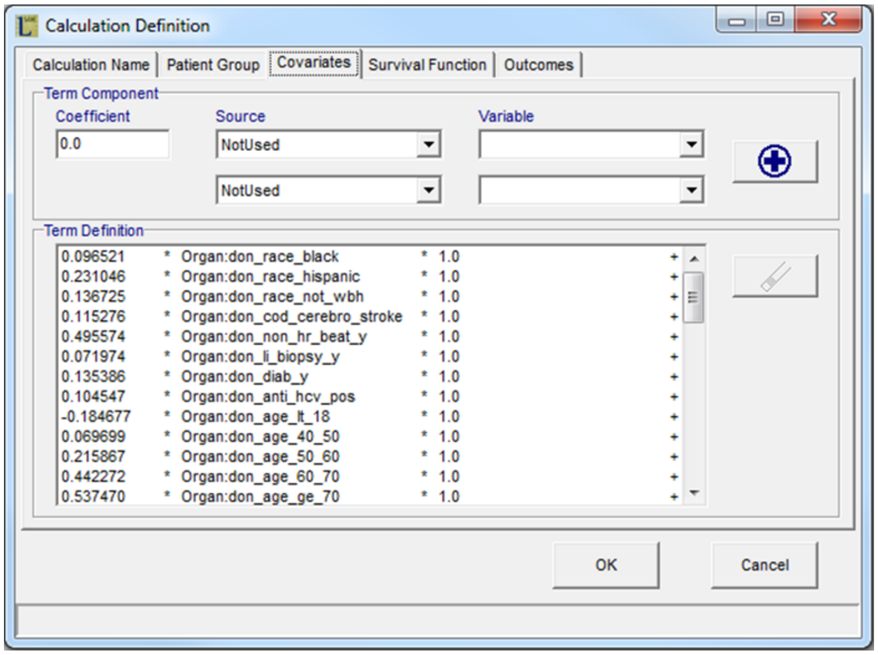
Calculation Definition (Post-Graft Survival)

Covariates

The post-graft survival equations are specified by the user. Users define an equation for each patient group defined. This equation represents either the placement into a step table, or the calculation of a Weibull function.

The equation, similar to the acceptance equations, may be made up of scalar variables, organ characteristics, patient characteristics, and/or calculations that depend on information from both the organ and the patient. Each time a transplant is performed in the allocation run, this equation is solved, and the resulting value is combined with a random number; this result is then used to determine the organ failure date.

The information from the covariates screen is used to determine the relative expected survival time between patients. The actual estimated survival time is calculated based on the survival model. The survival model is made up of the covariates and a survival function. A survival function is an overall curve, describing the basic pattern of survival for patients with regard to the proportion of patients alive at a certain time. This curve is modified for each patient through the individual covariate pattern; patients with covariates indicating poor health will tend to die sooner than patients with covariates indicating good health.



Coefficient: 0.0
Source: NotUsed
Source: NotUsed

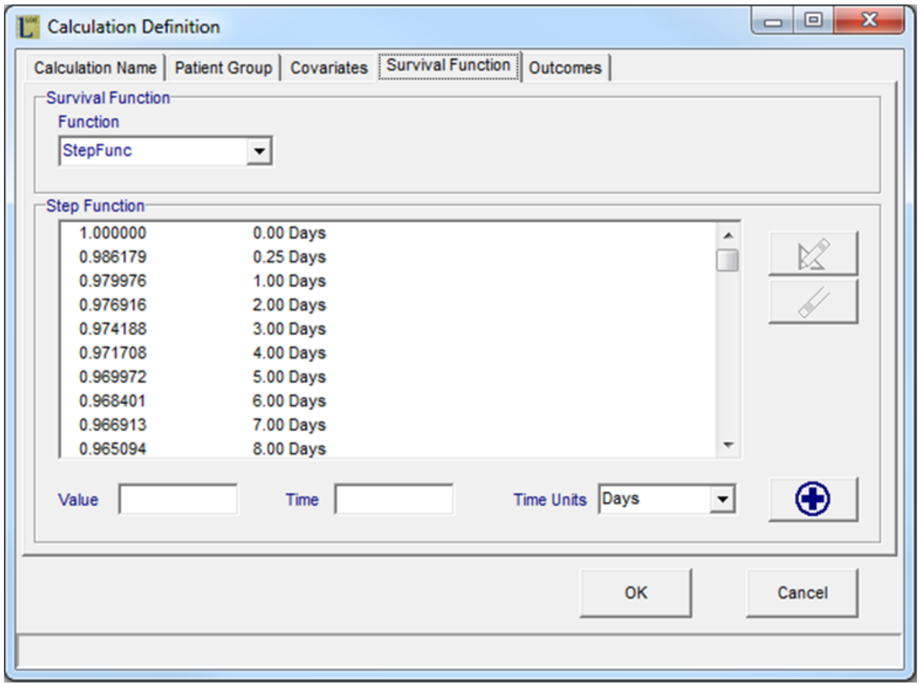
```
0.096521   * Organ:don_race_black          * 1.0   +
0.231046   * Organ:don_race_hispanic       * 1.0   +
0.136725   * Organ:don_race_not_wbh        * 1.0   +
0.115276   * Organ:don_cod_cerebro_stroke  * 1.0   +
0.495574   * Organ:don_non_hr_beat_y       * 1.0   +
0.071974   * Organ:don_li_biopsy_y         * 1.0   +
0.135386   * Organ:don_diab_y              * 1.0   +
0.104547   * Organ:don_anti_hcv_pos        * 1.0   +
-0.184677  * Organ:don_age_lt_18           * 1.0   +
0.069699   * Organ:don_age_40_50           * 1.0   +
0.215867   * Organ:don_age_50_60           * 1.0   +
0.442272   * Organ:don_age_60_70           * 1.0   +
0.537470   * Organ:don_age_ge_70           * 1.0   +
```

Select the **Survival Function** tab to continue.

Calculation Definition (Post-Graft Survival)

Survival Function – Cox Model Step Function

On this tab, you may choose between a Cox model step function or a Weibull distribution to determine the graft failure date. If you choose the Cox model, then you must enter information indicating the post-graft survival steps and associated failure times. Refer to Chapter 4, Modeling Post-Graft Survival, Graft Failures, and Relistings for more information about post-graft survival input.



Function: StepFunc

1.000000	0.00 Days
0.986179	0.25 Days
0.979976	1.00 Days
0.976916	2.00 Days
0.974188	3.00 Days
0.971708	4.00 Days
0.969972	5.00 Days
0.968401	6.00 Days
0.966913	7.00 Days
0.965094	8.00 Days

Value Time Time Units: Days

Choose the **Outcomes** tab to continue.

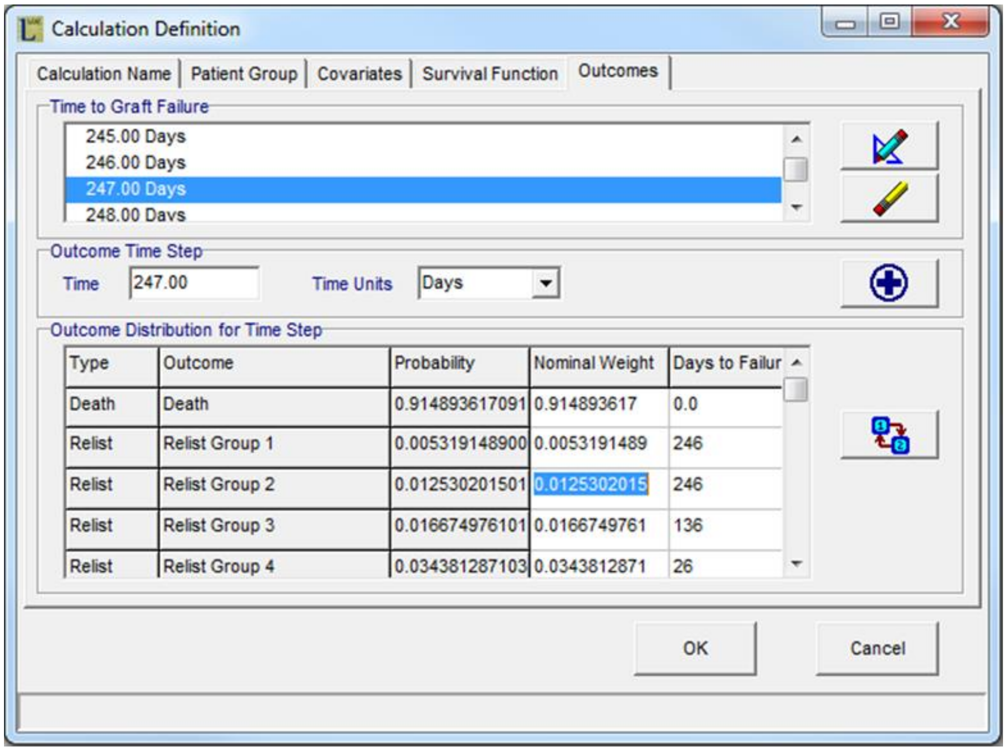
Calculation Definition (Post-Graft Survival)

Outcomes

Each row in the Time to Graft Failure section represents one step in the step table, and an associated organ failure date (which is equivalent to death date). Here, we will associate a set of outcomes with each step. Each outcome has a probability, also defined by the user. To see the Outcome Distribution for a particular step, highlight the step, then select the edit button to the right of the Time to Graft Failure section.

Each outcome has a probability, and a number of days before failure when this outcome will occur. In addition, if the type of outcome is relist, one or more status updates may occur between relisting and organ failure. These are also defined by the user.

For example, suppose a patient undergoes transplant and arrives at this outcome step table with a failure time of 247 days after transplant. Another random number is drawn, and the patient is assigned an outcome from the list at the bottom of the screen. If this patient ended up in Relist Group 2, then at 246 days before organ failure, this patient would relist. The patient may also have status updates, described below.



245.00 Days
246.00 Days
247.00 Days
248.00 Days

Time 247.00 Time Units Days

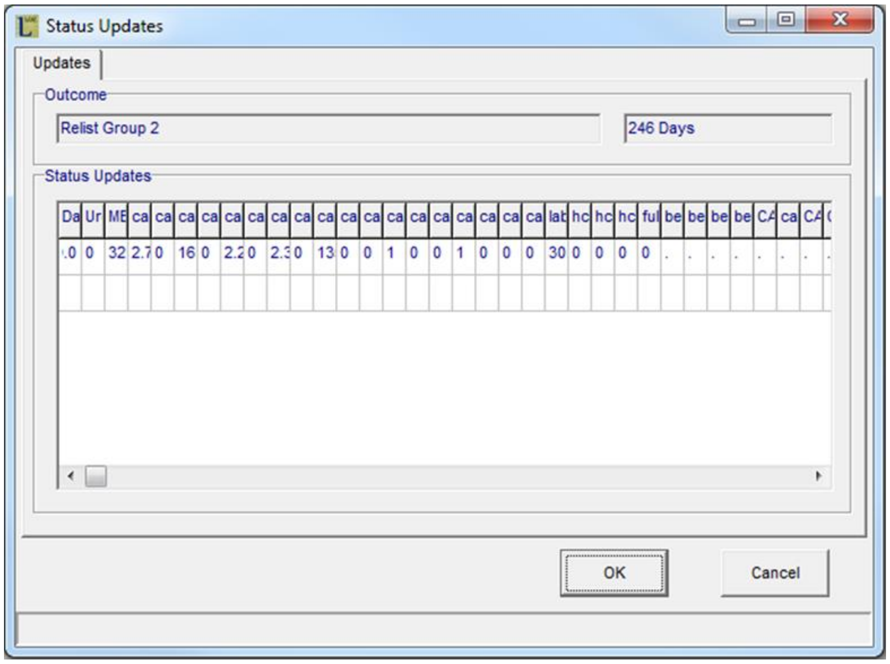
Type	Outcome	Probability	Nominal Weight	Days to Failur
Death	Death	0.914893617091	0.914893617	0.0
Relist	Relist Group 1	0.005319148900	0.0053191489	246
Relist	Relist Group 2	0.012530201501	0.0125302015	246
Relist	Relist Group 3	0.016674976101	0.0166749761	136
Relist	Relist Group 4	0.034381287103	0.0343812871	26

Highlight Nominal Weight for a Relist outcome, and select the update button (squares and arrows) on the right to continue.

1.6.5 Status Updates

All status updates that you want this patient to receive after transplant should be defined on this screen. Each update is relative to the date the patient is scheduled to relist, which is relative to the organ failure date. Possible status updates are changes to urgency status and changes to model for end-stage liver disease (MELD) scores. In addition, users may define other patient characteristics to be updated. Each of these additional characteristics will be represented by a column in this table. Here we've defined several additional patient characteristics to be updated after transplant, such as serum creatinine, bilirubin, laboratory MELD score, and HCC type 1 and type 2 status.

In our example above, the patient will relist 246 days before organ failure. On that date, the patient's urgency status will be set to 0 and MELD score to 32. The patient's serum creatinine, bilirubin, lab MELD, and HCC status fields will be set as specified. At 247 days after transplant, the patient's graft will fail (from the Time to Graft Failure on the previous page) unless the patient undergoes a second transplant before that date.


Status Updates

Updates

Outcome

Relist Group 2 | 246 Days

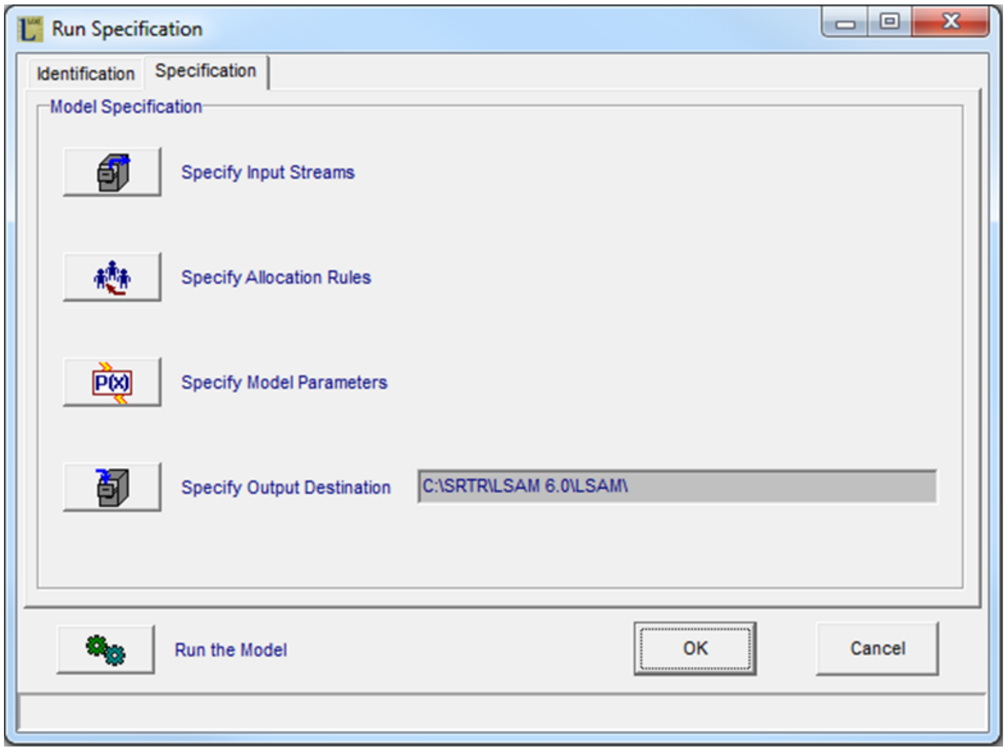
Status Updates

Da	Ur	ME	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	ca	lab	hc	hc	hc	ful	be	be	be	be	CA	ca	CA	(
.0	0	32	2.7	0	16	0	2.2	0	2.3	0	13	0	0	1	0	0	1	0	0	0	30	0	0	0	0	.	.	.	.	.	.	.	.

1.6.6 Output Destination

Run Specification: Specification

The final step is to specify the destination of outputs. The current output destination appears next to the Specify Output Destination button of the Specification tab.



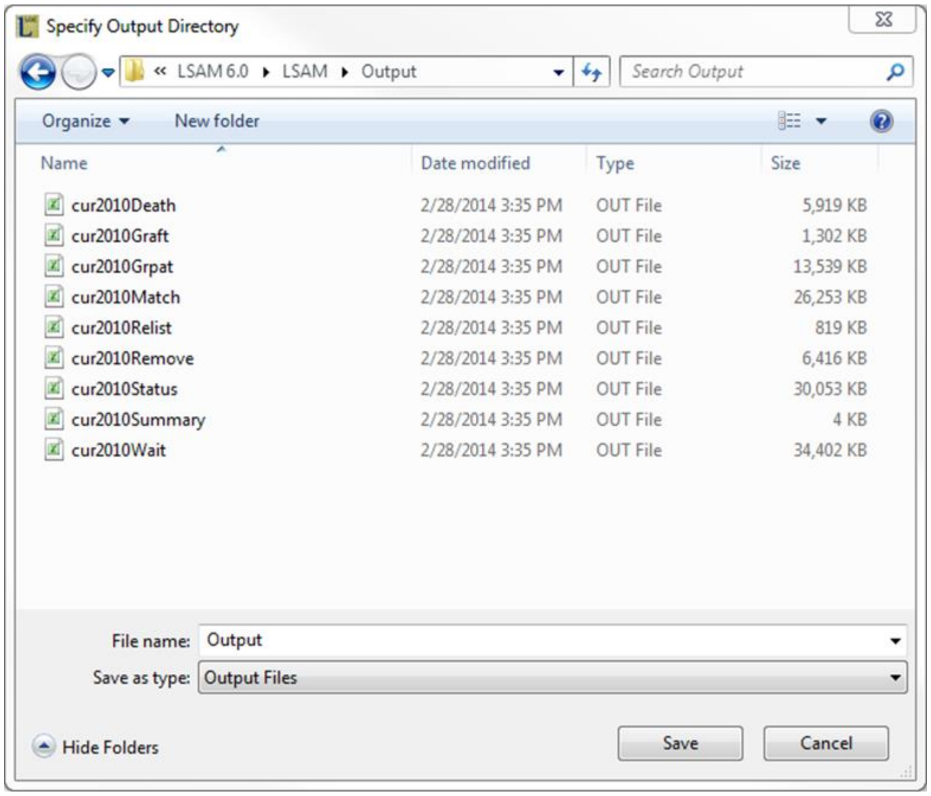
C:\SRTR\LSAM 6.0\LSAM\

Select the Specify Output Destination button, and the Specify Output Directory dialog appears.

Specify Output Directory

The simulation creates output files for each Allocation Run in the selected directory. The file names are those used in the source code. The model assigns each output file a different name by prefixing the model short name of the Allocation Run. See Chapter 8, Frequently Asked Questions, for more details.

This screen operates with the usual conventions of a Windows save dialog. The process for selection or creation of an output folder should be familiar to Windows users.



Name	Date modified	Type	Size
cur2010Death	2/28/2014 3:35 PM	OUT File	5,919 KB
cur2010Graft	2/28/2014 3:35 PM	OUT File	1,302 KB
cur2010Grpat	2/28/2014 3:35 PM	OUT File	13,539 KB
cur2010Match	2/28/2014 3:35 PM	OUT File	26,253 KB
cur2010Relist	2/28/2014 3:35 PM	OUT File	819 KB
cur2010Remove	2/28/2014 3:35 PM	OUT File	6,416 KB
cur2010Status	2/28/2014 3:35 PM	OUT File	30,053 KB
cur2010Summary	2/28/2014 3:35 PM	OUT File	4 KB
cur2010Wait	2/28/2014 3:35 PM	OUT File	34,402 KB

File name: Output
Save as type: Output Files

The example lists the output files from an Allocation Run named cur2010. You also may find it useful to create new folders to hold the inputs and outputs of different sets of Allocation Runs.

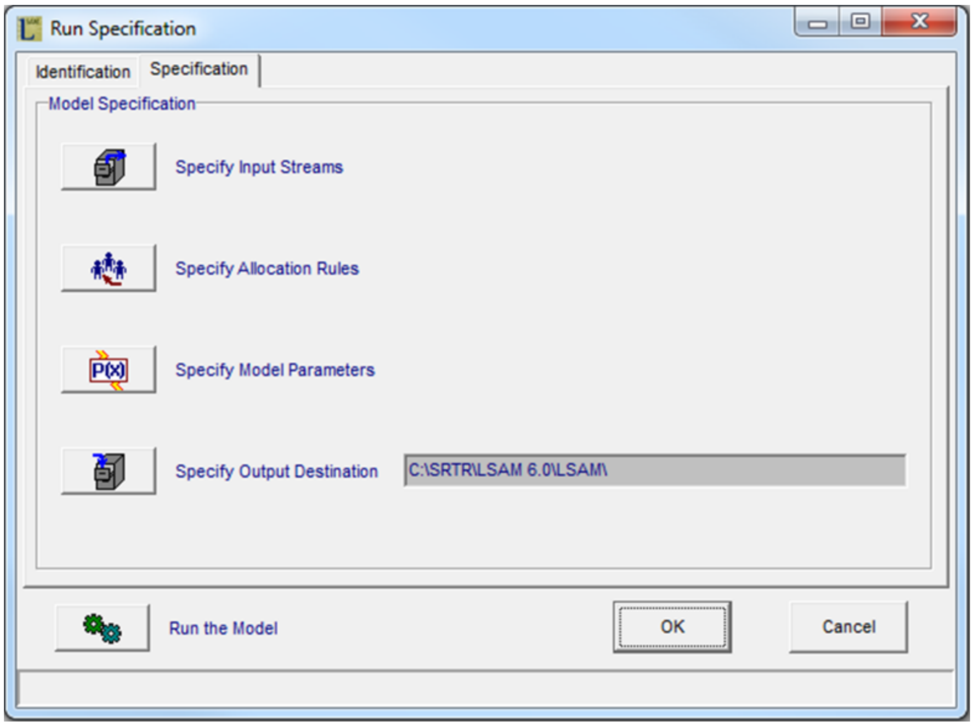
Select Save to return to the Allocation Run Specification screen.

1.6.7 Running the Model

We have visited the screens that specify the input data for the model and have returned to the Run Specification screen.

Run Specification

Now all the Allocation Run parameters are set, and we can run the model.



C:\SRTR\LSAM 6.0\LSAM\

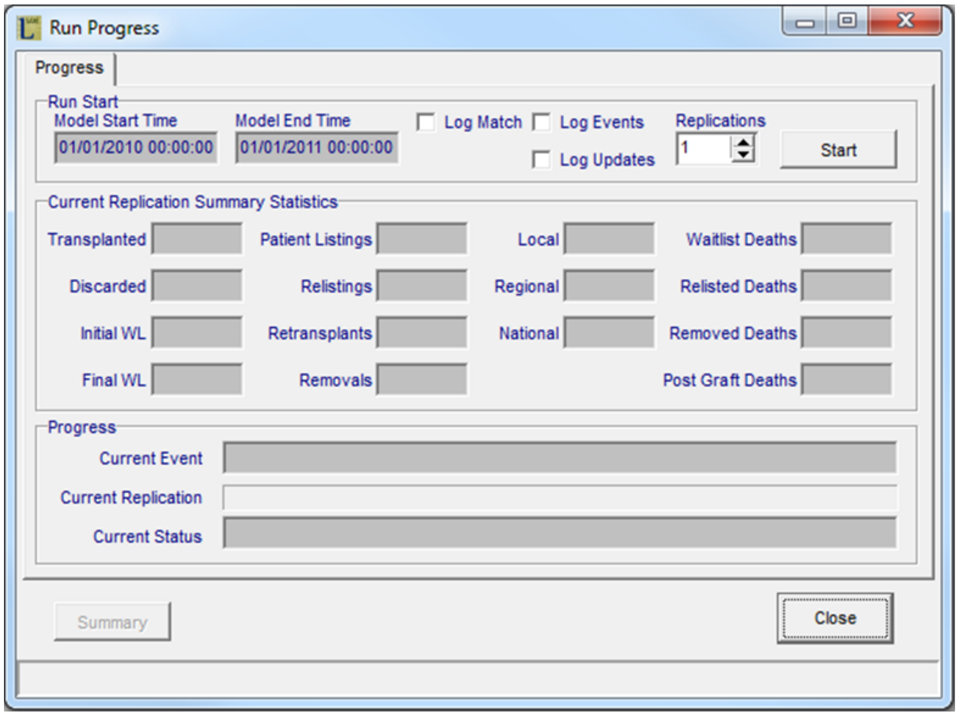
Select the Run the Model button. The Run Progress screen appears.

Run Progress

On this screen, you can select the number of replications to run and, optionally, you can choose to Log Match, Log Events, and/or Log Updates. See Chapter 7, Output File Specifications, for more details about these options.

Select Start. You will see the summary statistics and progress bars updated periodically as the Allocation Run progresses. While the model is processing, the Start button changes to a Stop button. This permits you to interrupt the Allocation Run process if necessary. The Start/Stop button changes to a faded Done message and the total running time is displayed in the Current Event box when the Model Allocation Run is complete.

The random number seed stream is initialized before the first replication. Subsequent iterations continue drawing from that stream where the previous iteration left off.



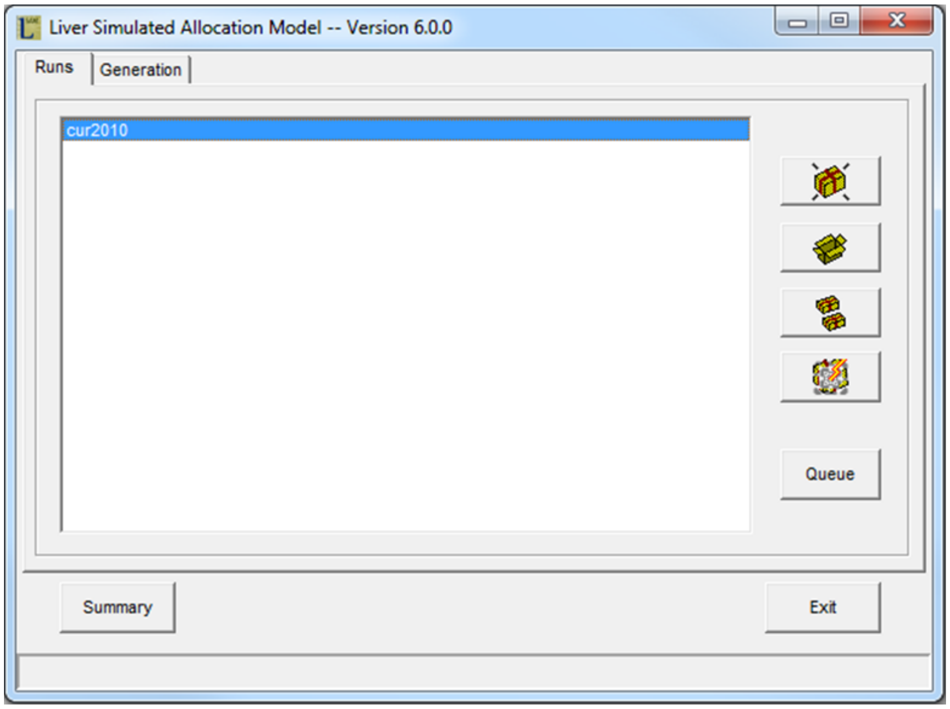
Model Start Time: 01/01/2010 00:00:00
Model End Time: 01/01/2011 00:00:00
Replications: 1

At the completion of the Allocation Run, you may view the summary statistics by selecting the Summary button. (Summary statistics also are recorded in an output file.) On the Summary screen (section 1.8), the Close button returns you to the Run Progress screen.

When the Allocation Run is complete, select the Close button, which takes you to the Allocation Run Specification screen. Closing out of that screen takes you back to the LSAM main screen.

1.6.8 Main Screen

If you have no more Allocation Runs to perform, select the Exit button, and the program will terminate.



If you have just defined a new Allocation Run package, its name will be listed on the main screen, along with the names of any previously defined packages. Start another Allocation Run by selecting one of the listed Allocation Run packages and then selecting the open box button. This will take you to the Run Specification screen, where you may run that package or edit its definition. Other button options allow you to create another new Allocation Run package, duplicate an Allocation Run package, or delete an Allocation Run package from the list.

You may also run the Queue (section 1.7), which lets you perform multiple runs without user input, and create a Summary file (section 1.8) from the available output.

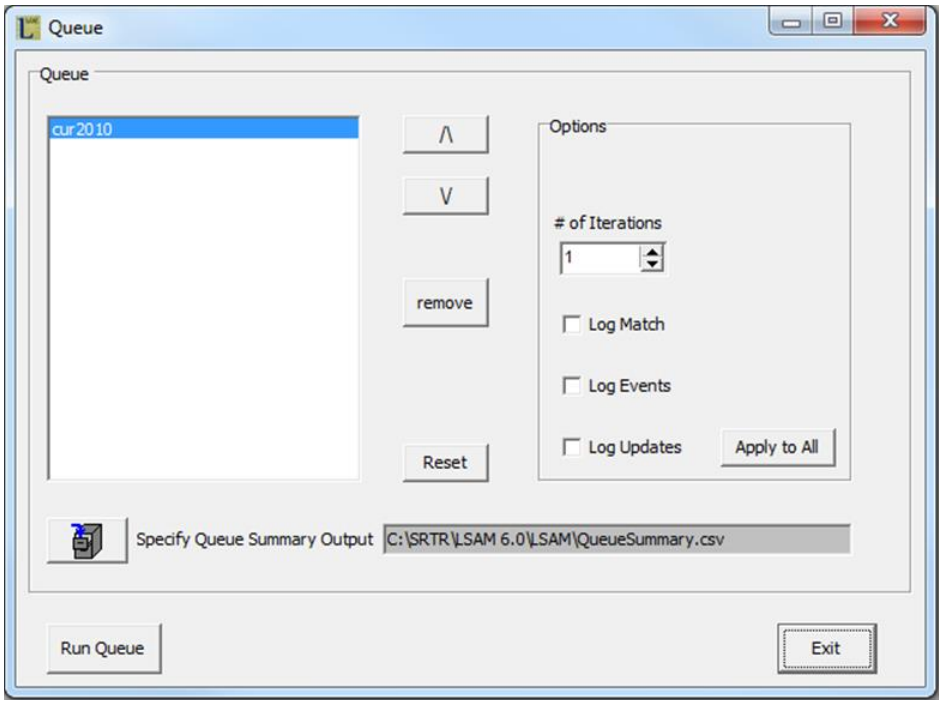
Four operations are allowed on run packages:

- Create a new Allocation Run package (wrapped package).
- Open the selected Allocation Run package (open box).
- Copy the selected Allocation Run package (two wrapped packages).
- Delete the selected Allocation Run package (lightning bolt destroying package).

1.7 Queue

The Queue feature allows users to select multiple models to run one after another without further user input. The user specifies the models, their order, and their settings, and then the program performs all the runs and returns to the Queue screen.

To open it, select the Queue button at the LSAM main screen.



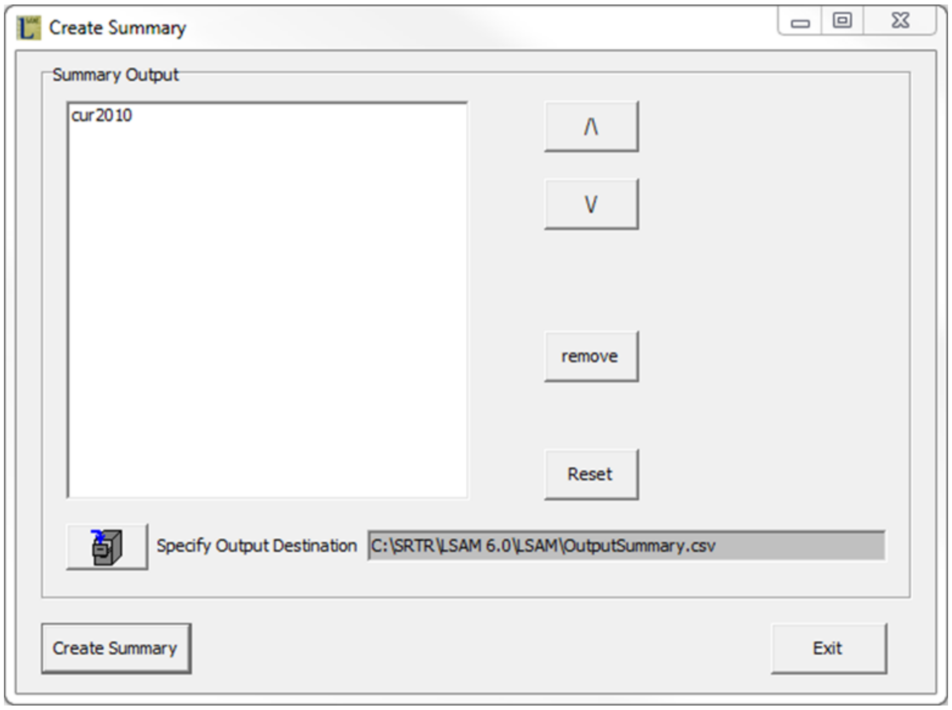
All the models are listed when the Queue is opened. Users can select the buttons in the middle to change the model order (runs go from top to bottom), remove runs, or reset the queue. Users can select options for each model on the right side, and apply the settings to all models in the queue.

After the run is completed, the Queue creates a QueueSummary.csv file, which includes all the model summary files in an Excel-friendly format.

To run the queue, select the Run Queue button on the bottom left. After the queue is completed, the program will return to the Queue screen, informing the user of the run time and if any errors occurred.

1.8 Summary

The Summary feature allows users to create an output file from any available model summaries. To open it, select the Summary button on the bottom left of the main screen of the LSAM.



cur2010

Specify Output Destination C:\SRTR\LSAM 6.0\LSAM\OutputSummary.csv

The program scans the models and checks whether an output summary exists in the model's specified location (for example, if you create a model and specify placing output in My Documents\LSAM Output, the program will look for the output summary in that folder). Users will see the models for which the output summary exists.

The user can specify the order, remove models, or reset the list, and also select the OutputSummary destination.

Select the Create Summary button on the bottom left to create the summary file. The file is in .csv format.

1.9 Viewing Output Files

The simulation creates output files for each Allocation Run, and it assigns file names by appending the short title of the Allocation Run to a specific identifier. For example, the file xxxMatch.out contains the match run results for an Allocation Run titled "xxx."

To view the output files, you can use Excel or another spreadsheet application and open the file with the vertical bar "|" as the **delimiter**.

2 Overview

This chapter provides an overview of the modeling methods and of the way computation is organized in the organ allocation model. The discussion covers the basic modeling approach (including which processes are random in the model), the top-level organization of the model, the main data structures referenced by the model, and the event handler routines.

2.1 Basic Approach and Random Processes

The model simulates the organ allocation system with an event-sequenced Monte Carlo technique. That is, some of the modeled processes are random in nature, and the model samples pseudo-random numbers to simulate a realization of processes over the specified time period. Each such realization of the organ allocation system constitutes a single replication. The model generates a user-specified number of replications, saves the detailed histories of these replications in user-specified files, and describes the set of replications with respect to the means, medians, and standard deviations of selected variables.
Some important assumptions:

1. Arrivals of candidates are input to the model with a data file.
2. The initial waiting list is input to the model with a data file.
3. An entire history of waiting list status changes to the end of the Allocation Run, or to death or removal from the waiting list, must be input to the model for each patient. These waiting list changes include, for example, MELD score changes, time of removal, time of death, and any user-defined variables that change over time. This is the history that will be used for the patient up until the time, if any, that the patient is allocated an organ during the simulation. Note that this history does NOT specify the time of transplant. This history can be based on actual experience; however, ,a hypothetical history must be prepared for patients who actually underwent transplant to indicate what would have happened had they not undergone transplant. Alternatively, the histories can be based on data generated from hypothesized models.
4. Once a candidate receives an organ, his or her input stream of status changes no longer applies. If the patient relists, the model assigns a status change history by randomly selecting a set of status changes from a pool of user-defined histories specifically provided for this purpose.
5. The values of several other parameters are specified in the program or tables and remain constant during a run. These include the geographic membership re-

lationships among institutions, local units (OPOs), and regions. For example, it is assumed that patients do not move among institutions, institutions do not change affiliation with OPOs, and the boundaries of national regions do not change. These relationships are controlled by input data and can be customized for each run, but changes within a run are not supported.

As this list shows, the model represents several important processes—candidate and organ arrivals and MELD score changes—through user input. Sample histories of these processes must be created outside the model (e.g., using historical data) and formatted into time-ordered streams of records for input into the model.

Other processes are represented internally to the model. The following processes are simulated within the model using Monte Carlo techniques, i.e., a pseudo-random number generator:

1. **Organ acceptance:** The user defines a calculation used to compute the organ acceptance probability. Values that may be used in this calculation are scalar variables, characteristics of the organ or donor, characteristics of the potential organ recipient, and/or values calculated from characteristics of the specific organ and patient combination under consideration. This calculation is performed for each patient to whom the organ is offered, and the resulting value, β, is transformed using an inverse logit transformation (exp(β) / 1+exp(β)). That value is then compared to a random number between 0 and 1. If the random number is less than this value, then the organ is accepted; otherwise, it is refused. An organ is considered discarded after it has been offered to all potential recipients and each offer has been refused, or after it reaches the maximum organ offer count specified on the Acceptance Definitions screen.

2. **Post-graft survival:** Post-graft survival is also specified in the model by the user. The user defines a calculation that is then used to determine the patient's death date after transplant. The calculation may be made up of scalar variables, organ characteristics, candidate characteristics, and/or calculations that depend on information from both the organ and the candidate. Each time a transplant is performed in the model, this calculation is performed, and the resulting value is combined with a random number; the result of this is used to determine the death date, using the method chosen by the user, either a Cox proportional hazard model or a Weibull distribution. A set of possible outcomes and their relative probabilities is associated with each death date. The possible outcomes are relisting at differing times before the death date and with differing medical characteristics, or not relisting.

Here, a relisted graft recipient is one who received a graft during the time span of the simulation, not necessarily one who entered the model with a prior graft. The latter set of recipients enter the model on the arrival stream or on the initial waiting list, and valid records presumably exist for them in the status change input stream (until the simulation awards new grafts to them).

The model draws all random numbers from a single random number stream.

2.2 Top-Level Overview: Event Queues

The simulation maintains time-ordered queues of several kinds of events, from which it can choose the next event in order for processing:

1. Organ arrivals (input-driven).
2. Status changes for waitlisted candidates who have not yet received grafts (input-driven).
3. Candidate arrivals (input-driven).
4. Status changes for relisted graft recipients (sampled from a list of possible outcomes).
5. Relisting events of recipients whose grafts fail (sampled by the model).
6. Deaths of graft recipients not on the waiting list (sampled by the model).

Note that status change events include changes of MELD scores and of status 1A or 1B, and changes indicating that the candidate has been removed from the waiting list or died while on the waiting list.

2.2.1 Organ Arrival Event

This event handler selects a candidate to receive the organ that has become available. It applies the allocation rules that have been defined in the model. It performs the match run by reordering the waiting list according to the rules and offering the organ to candidates in order. It simulates the organ acceptance process by sampling from a uniformly distributed random variable and comparing this to the probability of acceptance, which is calculated using acceptance inputs to the model. If no candidate accepts the organ, the organ is discarded. If the organ is accepted, the event handler removes the recipient from the waiting list. Whatever the outcome, the event handler writes a record with the results of the match run if the log match box was checked on the Run Progress screen before starting the run.

The event handler adds the recipient to a list of graft recipients and schedules a death event. It uses the model (as described in Chapter 5) to determine a possible outcome before death and the time that will elapse until this outcome. It then schedules the outcome events, which will include a relisting and possibly some post-graft status change events. Note that these status change events are defined differently in the model from those that take place before transplant. The latter are described next.

2.3 Event Handlers

2.3.1 Status Change Event

The status change file contains records that describe the medical status and MELD score history of every waitlisted candidate from the time of the candidate's arrival in the model (either the initial waiting list snapshot or arrival on the waiting list) until the candidate's death. This history is valid until the candidate receives a graft. If a transplant recipient re-lists, the recipient's status history is provided by a different source. Whichever source of status changes applies, the model invokes the status change event handler when a status change event occurs.

If the candidate's medical status changes, the model updates variables that keep track of the time that the candidate was at the previous medical status, including imposing the 30-day limit on credit for waiting time in inactive status (status 7). The event handler updates the candidate's medical status to the new value and updates other variables that keep track of status occupancy time. Similar processing applies to changes in MELD scores, including keeping track of the time that the candidate holds the current value of the MELD score.

If the new status is 9 (removal), the event handler removes that candidate from the waiting list and adds the candidate to a list of removed candidates. If the new status is 8 (death), the event handler removes the candidate from the waiting list and writes an outcome record for the candidate. (If the candidate dies after being removed from the waiting list, the model removes the candidate from the list of living removed candidates.)

2.3.2 Candidate Arrival Event

A candidate arrival event occurs when a patient joins the waiting list. The event handler adds the candidate to the waiting list and initializes all candidate descriptions, e.g., demographic descriptors, medical status and MELD score, previous transplant status, and listing institution. The specifications of the candidate arrival record appear in Chapter 6.

2.3.3 Posttransplant Events

When a candidate receives a graft (i.e., at the time of an organ arrival event), the simulation samples the future time of graft failure and determines whether the recipient will relist or not. The simulation accordingly schedules either the relist event or the recipient's death. The event handler for posttransplant events processes these events.

In the case of a death event, the event handler removes the recipient from the list of living graft recipients and writes an outcome record. In the case of a relisting event, the simulation removes the recipient from the list of non-waitlisted graft recipients and adds the recipient to the waiting list. A status change history was already selected for the recipient at the time of the organ arrival event, and the event handler initializes the candidate's medical status and MELD score to the initial set of values provided in this history.

2.4 Limitations

As discussed at the beginning of this chapter, LSAM relies on the inputs selected at runtime to simulate liver allocation. Any bias or omissions in the input data may affect simulation results. In addition to this general caveat, several specific limitations are worth noting when using LSAM and interpreting its results.

2.4.1 Reliance on Historical Data

LSAM uses statistical models to simulate multiple parts of the organ allocation process. These models were trained on historical data from OPTN and SRTR transplant archives; thus, they may not respond to changes in allocation policy in the same way the real system would. This is particularly true of the organ acceptance model, which simulates the decision-making behavior of patients and clinicians based on historical acceptance data. LSAM also does not identify trends over time in the training data, so gradual changes in the characteristics of donors and candidates will not be reflected in LSAM results.

2.4.2 Standardized Behavior

The OPTN allocation guidelines are extensive, but they do not eliminate all ambiguity. In practice, there is some amount of variation in policy and behavior between different OPOs and transplant centers. However, LSAM assumes that all centers and OPOs implement allocation policies in the same way and exhibit the same organ acceptance behavior. LSAM also does not model any directed or expedited allocation of donated organs.

2.4.3 Status Matching and HCC

Patients who underwent transplant in real life have no status updates in their patient records past the transplant date, so status updates from other patients were appended to fill out their LSAM patient histories in the SRTR-provided status input file. These patients are matched on expected mortality but not on specific diagnosis, so LSAM's ability to predict outcomes in diagnosis subgroups (such as HCC patients) is limited.

2.4.4 Single Listings, Single Organs

LSAM does not model patients who list at multiple transplant centers or who are listed for multiple organs. LSAM also does not model multi-organ transplants or split livers.

2.4.5 Discards

Organs are discarded after a fixed number of declined offers, regardless of organ and donor characteristics.

3 MELD-Based Allocation Rules

The model is sufficiently flexible to allow users to specify a wide range of allocation policies through the input files that are provided specifically for that purpose. The model is also distributed with a set of input files that contains an example of an allocation policy, which is available to the model as the default, should users elect not to specify new allocation rules via the appropriate input files.
The current OPTN liver allocation policy is a mixed rule comprising a MELD score and a medical category. The complete rule, described by OPTN policy 3.6, is available at http://optn.transplant.hrsa.gov/PoliciesandBylaws2/policies/pdfs/policy_8.pdf; historical overviews and summary flowcharts can be downloaded at http://srtr.org/allocationcharts/Default.aspx. The implementation of the current policy in LSAM is discussed below.

3.1 Mixed MELD Allocation Order

The policy orders candidates in status 1A and 1B by the number of points awarded. All other candidates are prioritized by MELD score or a related mortality risk score. For simplicity, MELD is used to refer both to MELD and pediatric end-stage liver disease (PELD) scores.

Adult Donors

The rule offers a liver from an adult donor to candidates in the following order of classification, and it allocates the liver to the highest priority candidate who accepts it:

1. Combined local and regional status 1A candidates in descending point order.
2. Combined local and regional status 1B candidates in descending point order.
3. Local and regional candidates with MELD scores ≥ 35 in descending order of MELD score, with local candidates ranked above regional candidates at each MELD level.
4. Local candidates with MELD scores 29-34 in descending order of MELD score.
5. National liver-intestine candidates in descending order of status and MELD score. *(Liver-intestine listing not implemented in LSAM.)*
6. Local candidates with MELD scores 15-28 in descending order of MELD score.
7. Regional candidates with MELD scores 15-34 in descending order of MELD score.
8. National status 1A candidates in descending point order.
9. National status 1B candidates in descending point order.

10. National candidates with MELD scores ≥ 15 in descending order of MELD score.
11. Local candidates with MELD scores < 15 in descending order of MELD score.
12. Regional candidates with MELD scores < 15 in descending order of MELD score.
13. National candidates with MELD scores < 15 in descending order of MELD score.

Within each classification, the rule offers a liver to candidates in descending point order. For an adult donor, each classification is drawn from both adult and pediatric candidates.

Pediatric Donors

The current OPTN rule allocates livers from donors aged 0-10 and 11-17 years in descending order of candidate mortality risk, similar to the adult donor policy. However, due to the importance of size matching organs from pediatric donors, pediatric candidates are given different levels of preference in these rules. The full rules can be reviewed at the internet locations given above.

LSAM Implementation

The current mixed MELD allocation rules are implemented in the LSAM input file DefMethods.txt. A second file, DefMethods-2010.txt, implements the previous version of the allocation rule and may be useful for historical comparisons.

The grouping of candidates in the DefMethods files is achieved through matching by medical urgency status, geographic sharing level (local, regional, or national), and candidate age. The sort order within each group is defined with allocation points, waiting time, and a variable called GeogRank, which prioritizes local over regional candidates at each MELD level for the MELD ≥ 35 rules. The syntax rules for the DefMethods files are discussed in Chapter 6, section 8.

3.2 Point Award for Status 1A and 1B Candidates

Ties are broken by points and then by waiting time in statuses 1A and 1B. Points are awarded on blood type and waiting time criteria. Waiting time rank is calculated within each classification. For example, a regional status 1A candidate is ranked only against other regional status 1A candidates. Likewise, when the liver under consideration is from a pediatric donor, a regional pediatric status 1A candidate is ranked only against other regional pediatric status 1A candidates.

In LSAM, these points are referred to as a variable called AllocPoints.

Blood type +10 points for ABO identical.

+5 points for ABO compatible.

0 points for status 1A or 1B candidates who accept an incompatible blood type.

Waiting time +10 points for longest waiting time within classification.

Each candidate receives a fraction of 10 points proportional to waiting time rank within classification. This scheme applies to waiting time within classification for status 1A or 1B candidates.

3.3 Medical Status

The following table summarizes the medical statuses for adult candidates. For the model, medical status incorporates both the medical urgency status and the removal codes from the SRTR data into one field.

Status	Definition
0	Candidate is not status 1A or 1B, but is active on the waiting list. Status 0 indicates a candidate who is ranked by MELD or PELD score.
1A	A candidate aged ≥ 18 years with fulminant liver failure and a life expectancy without a liver transplant of less than 7 days, or a recipient whose graft failed within 7 days of transplant, or a candidate with acute decompensated Wilson's disease. Under certain circumstances, a candidate aged < 18 years may be classified as status 1A.
1B	A candidate aged < 18 years with fulminant liver failure and a life expectancy without a liver transplant of less than 7 days, or a recipient whose graft failed within 7 days of transplant, or a candidate with acute decompensated Wilson's disease.
7	A candidate who is temporarily inactive; however, the candidate continues accruing waiting time up to a maximum of 30 days. United Network for Organ Sharing (UNOS) staff will confirm the inactive status at the end of 30 days. Candidates who are considered temporarily unsuitable for transplant are listed as status 7.

8	A candidate who dies while waiting for transplant.
9	A candidate who has been removed from the waiting list and is no longer considered a part of the system.

There is a separate MELD/PELD score field for candidates who are not status 1A or 1B and are, therefore, ranked by MELD or PELD.

3.4 MELD Scores for STATUS 1A and 1B Candidates

This policy uses the MELD/PELD score to prioritize candidates who are not assigned to medical urgency status 1A or 1B. Within a whole integer match MELD or PELD score, organs are awarded first to candidates with identical ABO blood type, then to candidates with compatible blood types. In this delivery of LSAM, the ranking by ABO compatibility is done using a user-defined variable called ABOScore. Within that group, ties are broken by waiting time at the current or higher score, which is represented, in this delivery, by a variable called MeldTime.

3.5 Screening Criteria

The rules screen out candidates from consideration for any of the following reasons:

1. **ABO:** donor and candidate types incompatible.
2. **Exception:** accept if candidate status is 1A or 1B or MELD score is ≥ 25 and candidate will accept an incompatible type. ABOChart.txt reflects the MELD 25 cutoff.
3. **Age:** donor age not within candidate's acceptable age range.
4. **Weight:** donor organ weight not in candidate's acceptable weight range.
5. **HCV status:** donor's hepatitis C virus (HCV) core antibody status is positive and candidate will not accept HCV positive donor.
6. **Hepatitis B:** donor's hepatitis B virus (HBV) core antibody status is positive and candidate will not accept HBV positive donor.
7. **Program status:** the liver program at the candidate's transplant center is temporarily inactive.
8. **Candidate status:** candidate's status is temporarily inactive.

3.6 ABO Blood Typing

The following table summarizes blood type compatibilities:

Donor	Candidate's Blood Type			
	O	**A**	**B**	**AB**
O	I	C/X	C/X	C/X
A	X	I	X	C
A1	X	I	X	C
A2	C*/X*	I	X	C
B	X	X	I	C
AB	X	X	X	I

C*/X*:	Compatible if candidate will accept an A2 donor, otherwise incompatible.
I:	Identical.
C:	Compatible.
C/X:	Compatible if candidate is status 1A, 1B, or ABO type B and MELD 30, otherwise incompatible.
X:	Incompatible.

The model is written to accept the parameters of the point award schemes as inputs, so the baseline values listed above can be changed.

Note that candidates whose blood type is defined as incompatible are not necessarily screened from OPTN match runs, or automatically from LSAM. For instance, an ABO type O donor organ will be offered to an ABO type A or AB candidate, or to an ABO type B candidate with MELD score < 30 after the organ has been offered to all other candidates nationally.

4 Modeling Post-Graft Survival, Graft Failures, and Relistings

Posttransplant survival is an important factor in organ allocation. This chapter discusses how LSAM, based on model inputs, estimates how long a recipient would live after receiving a particular organ at a particular time. This can be used to calculate, after a series of model runs, whether a given policy would result in greater organ utility as measured by overall posttransplant survival. Of note, in LSAM, a graft failure equals a death.

The model inputs can be in the form of a Cox proportional hazards model or a Weibull distribution equation. The following sections describe these methods and the form of the inputs that LSAM requires.

4.1 Time to Graft Failure Determined with a Cox Proportional Hazards Model

This section describes the process for entering the parameters required for LSAM to determine the time to graft failure using a Cox proportional hazards model.

When a candidate receives a graft, the model samples the time, T, remaining before death. The user specifies the survival model by specifying the model variables and the coefficients of the following survival function:

$$\text{Prob [survival time > T]} = S_j(T)^{\exp(\Sigma b_{ij} x_{ij} y_{ij})}$$

Where:

Sj(T) is a step function specified by the model user,

j is an indicator for recipient group as specified by fields in the recipient and organ records. The user may select a different set of variables and coefficient values for each group, e.g., diagnosis-specific survival models,

b_{ij} is the ith coefficient within group j, i.e., $\Sigma b_{ij} x_{ij} y_{ij}$ = the sum of the product bxy over all covariates (i) within group j,

x_{ij} is either the ith covariate within group j or the first part of an interaction term within group j,

y_{ij}, if specified, is the second part of an interaction term within group j.

Covariates and Parameters

Each element x_{ij} and y_{ij} is the value of a model variable or of unity. (A model variable here means the value of a variable describing the recipient or the organ involved in the graft event.) The user selects the variables x_{ij} and y_{ij}, and provides the corresponding coefficients b_{ij}. This is done separately for each recipient group.

$x_{ij} = 1$ and $y_{ij} = 1$ implies that this is the intercept term.
x_{ij} = variable and $y_{ij} = 1$ implies that this is an ordinary term within the model.
x_{ij} = variable1 and y_{ij} = variable2 implies that this is an interaction term be-
tween
variable1 and variable2.

Survival Time

At each transplant event, the model determines the remaining survival time, T, by sampling a value u from a U(0,1) distribution and inverting the complementary cumulative probability distribution of the survival time for this recipient:

$$\text{Prob [survival time} > T] = S_j(T)^{\exp(\Sigma b_{ij} x_{ij} y_{ij})} = u$$

$$T = S_j^{-1}(\exp(\ln(u)/\exp(\Sigma b_{ij} x_{ij} y_{ij})))$$

Step Function

The model reads data for $S_j(t_{kj}) = v_{kj}$ in tabular form (separately for each group j). These are specified by the user in the Default Survival input file, described in Chapter 6, or on the Step Function screen, described in Chapter 1, section 6.4. Specifically, the table includes values of t increasing from t=0 (time of transplant) to t=maxtime, and corresponding values of v declining from v=1 (i.e., all recipients are alive at time zero) at t=0 to v=0 for the largest t (eventually all the recipients will die).

The model reads the N_j time values for group j as input: $0=t_{0j}<t_{1j} <t_{2j} < \ldots < t_{N\,j}$ and the values as $1=v_{0j} > v_{1j} > \ldots > 0=v_{Nj}$.

Note that in any given dataset, it would be rare for all recipients to be followed until they die. Some recipients live a long time and some drop out of contact. The user must decide how the survival curve should be extrapolated to $S(t_{Nj}) = 0$ when creating the input file for this step function.

At each graft event, the model samples a value u from a U(0,1) distribution and solves the equation S(T) = u for the survival time T. Solving for the survival time requires putting the sample value, u, into the inverted survival function:

$$T = S^{-1}(u) = S_j^{-1}(\exp(\ln(u)/\exp(\Sigma b_{ij}x_{ij}y_{ij})))$$

To perform this inversion, the model first computes the value of the transformed probability v:

$$v = \exp(\ln(u)/\exp(\Sigma b_{ij}x_{ij}y_{ij}))$$

We are guaranteed that $0 \leq v \leq 1$. The model then uses the step function to determine the corresponding survival time. That is, if the value of v lies between v_{nj} and $v_{n-1,j}$, the survival time is $T = t_{nj}$. This condition will be true for some value of n for n = 1 to N.

The step function for the proportional hazards survival function is the baseline survival curve $s_0(t)$. The step function is a flexible way to approximate survival curves of any shape. Users must remember to enter the baseline survival function[1] rather than the survival function for the average recipient on this screen. In SAS, the baseline survival function can be created as follows:

First, create a separate dataset where all the covariates in the posttransplant survival model are set to zero, e.g.,

```
data covzeros;
  age = 0;  male = 0; weight = 0; height = 0;
run;
```

Then run the posttransplant survival model using this dataset to supply the covariates for the survival curve to be output, e.g.,

```
proc phreg data=analysis_data;
 model recipient_days*died(0) = age male weight height;
baseline out = ph_step_function covariates = covzeros  survival = survival
       /nomeans;
```

[1] In the baseline survival function, all covariates in the model are set to zero. This includes covariates that normally cannot equal zero, such as patient weight.

```
        run;
```

The proportional hazards baseline step function data are now saved in the output dataset (ph_step_function in the example above). These data can be entered into the SAM manually or through an input file (Default Survival under Model Structure).

4.2 Time to Graft Failure Determined with a Weibull Survival Model

This section describes the process for entering the parameters required for LSAM to determine the time to graft failure using a Weibull survival model.

The Weibull model is one of the exponential family of survival models. A strictly exponential survival model assumes that the same proportion of recipients die over the same length of time, a situation known as a constant hazard rate. For example, if 10% of the recipients die in the first year, then 10% of the remaining 90% will die in the second year, leaving 81% of the original population alive. The parameters entered into the model alter this overall percentage for certain groups of recipients, but they do not alter the shape of the curve. For example, older recipients may die at a rate of 15% per year while younger recipients die at a rate of 5% per year, but in the strictly exponential model both groups are assumed to have a constant mortality rate at all times. Weibull models are more flexible than the strictly exponential models in that their hazard rates can be monotone increasing, decreasing, or constant. This helps model situations such as posttransplant mortality, in which the hazard is expected to be high at first (due to the operative stresses), then to decrease over time. Aside from the covariates in the model, two parameters define the overall shape of the survival curve, defining this change in the hazard. These two parameters can differ depending on the model formulation.

There are three Weibull model formulations, each corresponding to a commonly used parameterization of the two Weibull parameters needed. The parameterization to be used depends on the software used to generate the model. For example, SAS output from the lifereg procedure includes the intercept, scale, and shape parameters. In SAS, the scale and the shape parameter are the inverse of each other, so only the shape and the intercept parameters are used in the LSAM. If the results of a lifereg procedure in SAS are to be used to describe posttransplant mortality, the formulation used should be the Accelerated Failure Time formulation: AFT: exp[-exp(shape*(ln(t)-intercept))]. The shape and intercept should be supplied in the spaces provided. To use another formulation, such as the proportional hazards PH: exp[-(t/scale)^shape], use the shape parameter as before and enter exp(intercept) in the space provided for the scale. The answers received via each formulation will be identical if the scale and the shape parameters are correctly entered.

4.3 Outcomes

Associated with each survival time is a table of probabilities for various possible outcomes (death or various relisting events). The model samples a value u from a U(0,1) distribution and compares it with the table of probabilities to determine the outcome.

4.4 Relisting

Associated with each relisting outcome is a time before graft failure at which relisting occurs, and a sequence of status updates for the relisted recipient. The model schedules a relisting event at the specified time before failure, and schedules the status update events. The model also schedules a death event for the recipient at the calculated failure time if a retransplant has not occurred.

5 Creating Input Data Fields

Creating suitable input files is not a trivial task. A number of problems can arise when data from real-world data sets are fed into the model. For example, if a status update file is created from real data, then candidates who actually underwent transplant will receive no status updates after the date of transplant. Thus, no candidate who actually underwent transplant will ever die or be removed from the list in the model, even if this means that these candidates survive for months or years as status 1 without undergoing transplant. To obtain realistic results, users should devise a method to generate simulated death or removal dates and status changes for candidates who underwent transplant.

5.1 Validating Input Data

LSAM does not catch inconsistencies in the input data. For instance, if a candidate starts with status 1 time longer than total time on the waiting list, LSAM does not check for consistency.

5.2 Sharing Input Files Among Users

If input files will be shared among users on different computers, note that the Allocation Run Specification file refers to each data file with its full path. Some editing of the file may be needed if the directory organization or location of the application on the target machine is different from on the source machine.

5.3 General Principles for All Input Files

Selecting Time Frame for Data

The model will simulate new arrivals, status changes, deaths, and transplants beginning with the snapshot date on the waitlist input data. Statistics will be reported only for the time period designated by the start and end dates selected when the model is run. Any changes that occur before the start date are not reported in the summary statistics. However, they are applied to the waiting list to make it current at the start date.

Formatting DateTime Fields

Format: (mm/dd/yyyy hh:mm:ss) As time information is not included in SRTR data, the time portion was randomly generated during the creation of the supplied input files.

Medical Urgency Status and MELD

Candidates with status 1A or 1B were assigned MELD = 0. If the MELD score was missing at the start date because of inactive status, MELD was set to 0. MELD changes were not applied to candidates listed as status 1A or 1B or status 7 (inactive).

Format of Input Files

All input files are ASCII text files, one line per record, with fields separated by a vertical bar delimiter (“|”).

6 Input File Specifications

Each specification lists fields in the order they occur in a record, followed by an example. Optional fields are italicized. Non-italicized fields are required. The specification of each field gives the name of the field, its data type, and a short definition. (The field names are those used in the source code.) Files are listed using generic filenames; a particular release may have more specific filenames related to release parameters (described in the Input Files Notes for the release).

6.1 Organ Arrivals

The organs file (organs.txt) contains organs that become available from the model start date through and including the model end date. For inclusion, we suggest selecting records using dispositions 5 (recovered for transplant but not transplanted) and 6 (transplanted).

Sort order: EventTime ascending.

Field Name	**Data Type**	**Short Definition**
EventTime	DateTime	organ arrival date-time (mm/dd/yyyy hh:mm:ss)
lsam_don_id	string	item identifier (donor ID)
InstitID	string	institution where donated
DonorAge	double	age of donor (same units as in candidate record)
ABOType	string	ABO blood type of donor (codes defined in blood compatibility file)
WeederFld(1)	double	first referenced weeder field
…		
WeederFld(n)	double	nth referenced weeder field
WeedFlag(1)	boolean	first referenced weed flag field
…		
WeedFlag(n)	boolean	nth referenced weed flag field
OptOrganFld(1)	specified	first specified optional organ data field
…		
OptOrganFld(n)	specified	nth specified optional organ data field

Example: 10/01/2001 2:39:28|OJA010|741|64.6|A|64.6|161.4|False|False|161.4

Note: This example contains two weeder fields, two weed flags, and one optional fields (after the last required field, the ABOType field). For this example data to work with LSAM, these flags and fields must have been defined in the Optional Data Definition file. Refer to section 6.7, which describes the Optional Data Definition input file.

6.2 Candidate Waiting List and Candidate Arrivals Files

The Waiting List (waitlist.txt) and Patient Arrivals (patients.txt) files are of the same format. They are both described in this section.

waitlist.txt contains all candidates on the waiting list (active and inactive) as of the day before the model start.

patients.txt contains new patients added to the waiting list with listing dates from the model start date through and including the model end date.

The time portion of the EventTime (hh:mm:ss) had to be randomly set since that information is not in the SRTR data. Sometimes a status change occurs on the same day as a new candidate record. As we want all status changes to occur after the new candidate listing, we intentionally set all the new candidate listings to take place in the AM and the status changes in the PM to insure that all status changes submitted on the same day as a new candidate listing occur after the new listing.

Sort order: EventTime ascending.

To avoid over-counting, new candidates who are relisted after undergoing transplant during the model run time period should be excluded from the input data. Relistings after transplant are simulated in the model.

Field Name	**Data Type**	**Short Definition**
SnapDate	DateTime	as-of date for current status of candidate (mm/dd/yyyy hh:mm:ss)
EventTime	DateTime	candidate arrival date-time (mm/dd/yyyy hh:mm:ss)
lsam_cand_id	string	item identifier (candidate ID)
CenterID	string	transplant center identifier
PatientAge	double	patient age as of snapdate for initial waitlist file {years} patient age at time of listing for patient arrivals file {years}
ABOType	string	ABO blood type (codes defined in blood compatibility file)
OrgUrgStat	string	original medical urgency status at time of listing {1|1A|1B|2A|2B|3|7}
MedUrgStat	string	current medical urgency status {1|1A|1B|7}
OrgMeld	double	original MELD score at time of listing {0.0 … 40.0}

MeldScore	double	current MELD score {0.0 … 40.0}
PrevTrans	boolean	current liver is a transplant {True|False}
WillATwo	boolean	if blood type O, will accept A2 donor {True|False}
WillIncBlood	boolean	will accept incompatible blood type {True|False}
ElapsedActive*	double	cumulative active waiting time {days}
ElapsedStat(1)*	double	cumulative time in status 1 {days}
ElapsedStat(1A)*	double	cumulative time in status 1A {days}
ElapsedStat(1B)*	double	cumulative time in status 1B {days}
ElapsedStat(2A)*	double	cumulative time in status 2A {days}
ElapsedStat(2B)*	double	cumulative time in status 2B {days}
ElapsedStat(3)*	double	cumulative time in status 3 {days}
ElapsedStat(7)*	double	cumulative time in status 7 (inactive){days}
ElapsedMeld(6)*	double	cumulative time in MELD score 6 {days}
…		
ElapsedMeld(40)*	double	cumulative time in MELD score 40 {days}
WeederFld(1) Min		
WeederFld(1) Max	double	first referenced weeder field maximum value
…		
WeederFld(n) Min	double	nth referenced weeder field minimum value
WeederFld(n) Max	double	nth referenced weeder field maximum value
WeedFlag(1)	boolean	first referenced weed flag field
…		
WeedFlag(n)	boolean	nth referenced weed flag field
OptPatientFld(1)	specified	first specified optional candidate data field
…		
OptPatientFld(n)	specified	nth specified optional candidate data field

*Cumulative times can be waiting times accrued under a previous waitlist registration for reinstated candidates. Complete definitions appear on the OPTN website (www.optn.org, Policies/Bylaws, Policies, Section 3.2 Organ Distribution, 3.2.1.8 Waiting Time Modification.)

Example: (long line has been wrapped)
10/01/2001 0:20:35|10/01/20010:20:35|193080|756|57|O|2B|2B|33|33|
False|False|False|0|0|0|0|0|0|0|0|.|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0.0|99.0|100.0|440.0|False|False|False|M|Mid-east/Arabian|2|Liver : Non-Cholestatic Cirrhosis|232.0

Note: Create the waitlist input file to be fixed as the existing waiting list on the day before the Model Allocation Run start date; i.e., if the Model Allocation Run is to start on January 1, 2010, create the waiting list ending December 31, 2009. Cumulative times are calculated up to and including the last status change date (SnapDate). For candidates without status changes, cumulative times will all be 0 (see exception below) and SnapDate will be the listing date/time. The first 30 days' credit for inactive status is added to the ElapsedActive field during the Model Allocation Run. Waitlist input file sort order: none. Patient Arrivals input file sort order: EventTime ascending.

The number of weeder fields expected is determined by what is found in the optional data definition file. Their order here must be the same as their order in the optional data definition file. The number and order of weed flag fields are also determined by what is found in the optional data definition file. The number and order of optional fields expected are also determined by what is found in the Optional Data Definition file. Refer to section 6.7.

6.3 Status Updates

The simulation uses actual waitlist arrivals, status changes, and removal information for candidates on the waiting list during the model run timeframe. Status.txt contains records that describe the medical status and MELD score history of every waitlisted candidate from the time of the candidate's arrival in the model (either the initial waitlist snapshot or arrival on the waiting list) until the candidate's removal or death.

We had to randomly set the EventTime (hh:mm:ss) since that information is not in the SRTR data. Sometimes a status change occurs on the same day as a new candidate record. As we want status changes to occur after the new candidate listing, we intentionally set all the new candidates listings to take place in the AM and the status changes to take place in the PM when input files are created. Thus, if a status change occurs on the same day as a listing, the status change will occur later. We did not add logic to evaluate the order if more than one status change occurs on the same date.

Input data should include all status changes, from the model start date through and including the model end date. Use current status codes. Status changes should *not* include candidates removed from the waiting list due to transplant (status 4). Candidates removed due to death are status 8. All other status changes for candidates removed from the waiting list should be set as 9, i.e., due to improvement in or deterioration of the candidate's condition.

Candidates who in real life underwent transplant will receive no status updates after the date of transplant unless their status histories are completed in some manner, such as the predictive mean matching described in the input files manual. Thus, no candidate who actually underwent transplant will ever die or be removed from the list in the model, even if this means these candidates survive for months or years at status 1 without undergoing transplant. To obtain realistic results, generate artificial status changes and death or removal dates for these candidates.

Sort order: EventTime ascending. Note: MedUrgStat 7=inactive; 8=death; 9=removed

Field Name	Data Type	Short Definition
EventTime	DateTime	event date-time (mm/dd/yyyy hh:mm:ss)
lsam_cand_id	string	item ID (Patient ID)
MedUrgStat	string	new medical urgency status {1|2A|2B|3|7|8|9}
MELDScore	double	new MELD score {0 … 40}
OptStatusFld(1)	specified	first specified optional status data field
…		
OptStatusFld(n)	specified	nth specified optional status data field

Example: 10/01/2001 12:00:31|167712|2B|19

Note: The number of optional fields expected, and their order, is determined by what is found in the Optional Data Definition file. Refer to the section 6.7, which describes that input file.

6.4 Location Mapping

LocMap.txt is a listing of all transplant center IDs with a code for their related local and regional areas.

Sort order: CenterID unique, ascending. *Note: in the input data supplied, Center ID is masked for confidentiality.*

Field Name	Data Type	Short Definition
CenterID	string	transplant center ID institution where donation occurred
LocalID	string	local unit ID where center/institution is located
RegionID	string	region ID where center/institution is located

Example:

```
2|USVA|01
3|LCPA|11
4|LCMO|11
5|LCRI|11
6|LCAK|06
```

6.5 Blood Compatibility

ABOChart.txt has two parts. The first part contains two MELD threshold values. The first MELD threshold is used to determine whether a candidate with ABO blood type B may be considered identical for purposes of allocation to a donor with ABO type O; the second is used to determine whether a candidate who has specified willingness to accept an incompatible blood type donor may be offered the organ. The OPTN rules state that the candidate's MELD score must be above this threshold for the candidate to be considered, even if the candidate specified willingness to accept an incompatible donor organ. Below is an example of the first part of the ABOChart.txt file.

```
<COMPRULE>
MELD|30|25
</COMPRULE>
```

The second part of the ABOChart.txt file is a listing of all possible donor and candidate blood type combinations. Compatibility type "special" is used when the organ ABO is A2 and the candidate ABO is O, since compatibility is then determined by whether or not a type O candidate will accept a type A2 organ.

Field Name	Data Type	Short Definition
OrganABO	string	blood type of organ donor {O|A|A1|A2|B|AB|A1B|A2B}
PatientABO	string	blood type of patient {O|A|A1|A2|B|AB|A1B|A2B}
Match 1A or 1B	string	status 1 {I=identical, C=compatible, X=incompatible, S=special)
Match not 1A or 1B	string	MELD (not status 1) {I=identical, C=compatible, X=incompatible, S=special)

Example:

```
O|O|I|I
O|A|C|X
O|A1|C|X
O|A2|C|X
```

6.6 Default Data Definition

DefDataDef.txt contains the definition of the default fields that are available for forming allocation rules, score boost specifications, acceptance calculations, and post-graft survival calculations. The program must contain code to support each field that is referenced in this file. Note that optional fields are also available for those definitions, and are defined in the Optional Data Definition file, described in section 6.7.

DefDataDef.txt is an ASCII text file that contains blocks surrounded by defined block begin and block end tags.

Field List: <FIELDLIST>…</FIELDLIST>
The field list tag is the outer-most tag. All other blocks are contained inside the field list start and end tags.

Field Definition: <FIELDDEF>…</FIELDDEF>
The field definition block defines one field. The format for this block is:

```
<FIELDDEF>
fldname|fldsource|fldstate|fldtype|fldusage|fldbasis
</FIELDDEF>
```

fldname provides the name of the field. This must be a recognized field name.
fldsource identifies the source {Patient; Organ} of the field.
fldstate must be one of the following {Default; DefCalc; DefComp; DefCut} which defines how the program determines the value of the field.

Default: a field on the default patient or organ input record.
DefCalc: a field calculated by default by the program.
DefComp: a field whose value is determined by the program by comparing the patient and organ records.
DefCut: a field calculated by the program using cutpoints on a known field.

fldtype must be one of the following: {String; Double; DateTime}, which defines how the program stores the value of the field.
fldusage must be one of the following: {Category; Score}, which defines how the field is used by the program.
fldbasis identifies the numeric field to which the cutpoints are applied if the field state is a cutpoint field. Otherwise, this item is omitted.

Level Definition: <LEVELDEF>…</LEVELDEF>

The level definition block defines the acceptable values for the field. The format for this block is:

```
<LEVELDEF>
label|lowcut|highcut
</LEVELDEF>
```

label identifies, for category fields, which string values can be used for allocation, boost, etc.
lowcut defines the lower cutpoint for this level. Used only for cutpoint (fldstate = DefCut) fields. Otherwise, this item is omitted
highcut defines the upper cutpoint for this level. Used only for cutpoint (fldstate = DefCut) fields. Otherwise, this item is omitted.

Example:

```
<FIELDLIST>
<FIELDDEF>
ABOCompat|Patient|DefComp|String|Category
<LEVELDEF>
I
C
</LEVELDEF>
</FIELDDEF>
<FIELDDEF>
AdultMort|Patient|DefCut|String|Category
<LEVELDEF>
Low|0|32
High|33|40
</LEVELDEF>
</FIELDDEF>
</FIELDLIST>
```

6.7 Optional Data Definition

OptDataDef.txt consists of three sections: weeder definitions, flag definitions, and optional data definitions.

OptDataDef.txt is an ASCII text file that contains blocks surrounded by defined block-begin and block-end tags.

Weeder Definition: <WEEDDEF>...</WEEDDEF>

Each weeder definition specifies a value field in the organ record and a pair of range fields (minimum and maximum) in the patient (and initial waiting list) record. Candidates for whom the organ value does not fall within the specified range are weeded from the allocation for that organ. The weeder definition block may define multiple weeder fields. The format for this block is:

```
<WEEDDEF>
fieldname|organidx
</WEEDDEF>
```

fieldname provides the name of the field. The name must be unique within the data source.
organidx identifies the organ to which this weeder field applies. For the LSAM, this should always be set to 1.

Flag Field Definition: <FLAGDEF>…</FLAGDEF>
Each flag field definition specifies a boolean value in the organ record that indicates whether a condition is present and a boolean value in the candidate record that indicates whether presence of that condition in an offered organ is acceptable to the candidate. Thus, if the organ value is true and the candidate value is false, the candidate will be weeded from the allocation for that organ. In all other cases, the candidate will not be weeded. The flag field definition block may define multiple flag fields. The format for this block is:

```
<FLAGDEF>
fieldname|organidx
</FLAGDEF>
```

fieldname provides the name of the field. The name must be unique within the data source.
organidx identifies the organ to which this weeder field applies. For the Liver Model, this should always be set to 1.

Optional Data Field Definitions Group: <DATADEF>…</DATADEF>
The Optional Data Field Definitions Group block contains definitions for the optional data fields, each of which is contained in its own <FIELDDEF> block. The format for the optional data definitions group block is:

```
<DATADEF>
<FIELDDEF> … </FIELDDEF>
…
```

</DATADEF>

Three types of data field definitions may be defined in the Optional Data Field Definitions Group. They are the Optional Data Field, the Optional Cutpoint Field, and the Optional Calculation Field. Each of these types is described below. The following definitions apply to all three optional field definitions.

fldname provides the name of the field. The name must be unique within the data source.
fldsource identifies the source {Patient; Organ; Status} of the field. Patient fields also apply to waitlist records, which have the same format as patient records.
fldtype must be one of the following: {String|Double|DateTime}, which defines how the program stores the value of the field.

Optional Data Field Definition: <FIELDDEF>…</FIELDDEF>
The optional field definition block defines one optional field. The value Optional indicates to the program that the field will be supplied on the appropriate input file. The field source parameter specifies which file(s) will include the field—organ, patient and waiting list, or status update. The level definition block within the field definition block contains a list of possible values of Category fields. The format for the optional data field definition is:

```
<FIELDDEF>
fldname|fldsource|Optional|fldtype|fldusage
<LEVELDEF>              { when fldusage=Category }
value-1
…
value-n
</LEVELDEF>
</FIELDDEF>
```

fldusage must be one of the following: {Category|Score|PassThru}, which defines how the field is used by the program.
Category identifies a category field. This must be a string.
Score identifies a numeric field. This must be Double or DateTime.
PassThru identifies a field that is read from the input stream and written to the output without being used by the program.

Optional Cutpoint Field Definition: <FIELDDEF>…</FIELDDEF>
When the field state is *OptCut*, a category field will be created by the program using the specified cut points on the specified basis field, **fldbasis**. The level definition block contains

a list of possible values for this field. The **label** gives a name to values of the basis field that fall between the lower and upper cutpoints (**lowcut**, **highcut**) defined for this level.

```
<FIELDDEF>
fldname|fldsource|OptCut|fldtype|Category|fldbasis
<LEVELDEF>
label-1|lowcut|highcut
...
label-n|lowcut|highcut
</LEVELDEF>
</FIELDDEF>
```

Optional Calculation Field Definition: <FIELDDEF>...</FIELDDEF>
When the field state is *OptCalc*, a score field will be created by the program using the specified calculation operation.

```
<FIELDDEF>
fldname|fldsource|OptCalc|fldtype|Score
<OPTCALC>
optfunc|source:variable|source:variable|operator
</OPTCALC>
</FIELDDEF>
```

The calculation specification (<OPTCALC> block) for an OptCalc field consists of one of the following functions:
EnumerNum|source:variable converts the levels of a categorical variable into a number 0,1,... based on the order in which the levels of that field were specified. *Source* must be {Patient; Organ}.
Sum|source:variable|source:variable adds the values of the first and second variables. *Source* must be {Patient; Organ, Literal}, where Literal indicates that a numeric value will be specified instead of a field.
Difference|source:variable|source:variable subtracts the values of the second variable from the value of the first variable. *Source* must be {Patient; Organ, Literal}.
Ratio|source:variable|source:variable divides the value of the first (score) variable by the value of the second (score) variable. *Source* must be {Patient; Organ, Literal}.
Product|source:variable|source:variable multiplies the value of the first (score) variable by the value of the second (score) variable. *Source* must be {Patient; Organ, Literal}.
CompareNum|source:variable|source:variable|operator compares the values of the two specified variables using the specified comparison operator, returning 1=True or 0=False.

In these calculation specifications, *Source* must be {Patient; Organ; Literal}, as above. *Operator* must consist of one of the following comparison operators {GE; GT; LE; LT; EQ; NE}.
NaturalLog|source:variable gives the natural log of the value of the specified variable. *Source* must be {Patient; Organ, Literal}.
Exponential|source:variable gives the natural log of the value of the specified variable. *Source* must be {Patient; Organ, Literal}.
Power|source:variable|source:variable gives the value of the first variable to the power of the value of the second variable. *Source* must be {Patient; Organ, Literal}.
BoolOp|source:variable|source:variable|operator gives a boolean calculation of the value of the two specified variables. The calculation performed depends on the value of the operator. *Operator* must consist of one of the following boolean operators {OR, XOR, AND}. *Source* must be {Patient; Organ, Literal}.
Equation|equation gives the result of the specified mathematics formula.
The mathematics formula can be constructed using the elements:

+: operand, executes adding operation.
-: operand, executes subtraction operation.
*: function, executes multiplying operation.
/: function, executes division operation.
Sqrt: functions, root of a number. Root can have any degree.
Div: functions, executes integer division operation.
Mod: functions, executes remainder operation.
Int: function, returns the integer part of a number.
Frac: function, returns the fractional part of a number.
Random: function, returns random number within the range $0 \leq \text{value} < 1$.
Trunc: function, truncates a number to an integer.
Round: function, returns the value rounded to the nearest whole number.
Sin: function, returns the sine of the angle in radians.
ArcSin: function, returns the inverse sine of a number.
Sinh: function, returns the hyperbolic sine of an angle.
ArcSinh: function, returns the inverse hyperbolic sine of a number.
Cos: function, returns the cosine of the angle in radians.
ArcCos: function, returns the inverse cosine of a number.
Cosh: function, returns the hyperbolic cosine of an angle.
ArcCosh: function, returns the inverse hyperbolic cosine of a number.
Tan: function, returns the tangent of the angle.
ArcTan: function, returns the arctangent of a number.
Tanh: function, returns the hyperbolic tangent of an angle.
ArcTanh: function, the inverse hyperbolic tangent of a number.
CoTan: function, returns the cotangent of the angle.

ArcCoTan: function, returns the inverse cotangent of a number.
CoTanh: function, returns the hyperbolic cotangent of an angle.
ArcCoTanh: function, returns the inverse hyperbolic cotangent of a number
Sec: function, returns the secant of an angle.
ArcSec: function, returns the inverse secant of a number.
Sech: function, returns the hyperbolic secant of an angle.
ArcSech: function, returns the inverse hyperbolic secant of a number.
Csc: function, returns the cosecant of an angle.
ArcCsc: function, returns the inverse cosecant of a number.
Csch: function, returns the hyperbolic cosecant of an angle.
ArcCsch: function, returns the inverse hyperbolic secant of a number.
Abs: function, returns an absolute value.
Ln: function, returns the natural log of an expression.
Lg: function, returns log base 10.
Log: function, returns the log of expression for a specified base.
Pi: function, returns 3.1415926535897932385.
Exp: function, returns the exponential of an expression .
!: function, returns factorial of an expression.
^: function, raises expression to any power.
ArcTan2 [Y, X: Double] function, calculates ArcTan(Y/X), and returns an angle in the correct quadrant. The values of X and Y must be between –2^64 and 2^64. In addition, the value of X cannot be 0. The return value will fall in the range from -Pi to Pi radians.
Hypot [X, Y: Double] function, returns the length of the hypotenuse of a right triangle. Specify the lengths of the sides adjacent to the right angle in X and Y. Hypot uses the formula Sqrt(X**2 + Y**2).
RadToDeg function, converts radians to degrees.
RadToGrad function, converts radians to grads.
RadToCycle function, converts radians to cycles.
DegToRad function, returns the value of a degree measurement expressed in radians.
DegToGrad function, returns the value of a degree measurement expressed in grads.
DegToCycle function, returns the value of a degree measurement expressed in cycles.
GradToRad function, converts grad measurements to radians.
GradToDeg function, converts grad measurements to degrees.
GradToCycle function, converts grad measurements to cycles.
CycleToRad function, converts an angle measurement from cycles to radians.
CycleToDeg function, converts an angle measurement from cycles to degrees.
CycleToGrad function, converts an angle measurement from cycles to grads.
LnXP1 function, returns the natural log of (X+1).
Log10 function, calculates log base 10.
Log2 function, calculates log base 2.

IntPower [Base: Double; Exponent: Integer] function, calculates the integral power of a base value.
Power [Base: Double; Exponent: Double] function, raises base to any power.
Ldexp [X: Double; P: Double] function, calculates X times (2 to the power of P).
Ceil function, rounds variables up toward positive infinity
Floor function, rounds variables toward negative infinity.
Poly [X: Double; Coefficients(1)..Coefficients(N): Double] function, evaluates a uniform polynomial of one variable at the value X.
Mean [Data(1)..Data(N): Double] function, returns the average of all values in an array.
Sum [Data(1)..Data(N): Double] function, returns the sum of the elements in an array.
SumInt [Data(1)..Data(N): Integer] function, returns the sum of the elements in an integer array.
SumOfSquares [Data(1)..Data(N): Double] function, returns the sum of the squared values from a data array.
MinValue [Data(1)..Data(N): Double] function, returns smallest signed value in an array.
MinIntValue [Data(1)..Data(N): Integer] function, returns the smallest signed value in an integer array.
Min [A,B: Double] function, returns the lesser of two numeric values.
MaxValue [Data(1)..Data(N): Double] function, returns the largest signed value in an array.
MaxIntValue [Data(1)..Data(N): Integer] function, returns the largest signed value in an integer array.
Max [A,B: Double] function, returns the greater of two numeric values.
StdDev [Data(1)..Data(N): Double] function, returns the sample standard deviation for elements in an array.
PopnStdDev [Data(1)..Data(N): Double] function, calculates the population standard deviation.
Variance [Data(1)..Data(N): Double] function, calculates statistical sample variance from an array of data.
PopnVariance [Data(1)..Data(N): Double] function, calculates the population variance.
TotalVariance [Data(1)..Data(N): Double] function, returns the statistical variance from an array of values.
Norm [Data(1)..Data(N): Double] function, returns the Euclidean 'L-2' norm.
RandG [Mean, StdDev: Double] function, generates random numbers with Gaussian distribution.
RandomRange [AFrom, ATo: Integer] function, returns a random integer from a specified range.
RandomFrom [Value(1)..Value(N): Double] function, returns a randomly selected element from an array.
EnsureRange [AValue, AMin, AMax: Double] function, returns the closest value to a specified value within a specified range.

Example:

```
<WEEDDEF>
DonAge|1
DonWeight|1
</WEEDDEF>
#
<FLAGDEF>
HCVDonor|1
HepDonor|1
</FLAGDEF>
#
<DATADEF>
<FIELDDEF>
DoesReqCross|Patient|Optional|String|Category
<LEVELDEF>
True
False
</LEVELDEF>
</FIELDDEF>
<FIELDDEF>
PatientWeight|Patient|Optional|Double|Score
</FIELDDEF>
<FIELDDEF>
DonorWeight|Organ|Optional|Double|Score
</FIELDDEF>
<FIELDDEF>
WtRatio|Patient|OptCalc|Double|Score
<OPTCALC>
Ratio|Organ:DonorWeight|Patient:PatientWeight
</OPTCALC>
</FIELDDEF>
<FIELDDEF>
distance|Patient|OptCalc|Double|Score
<OPTCALC>
Equa-
tion|3443.9*(arccos(cos(Patient:("lat1"))*cos(Patient:("long1"))*cos(Organ:("lat2"))*cos(Orga
n:("long2")) +
cos(Patient:("lat1"))*sin(Patient:("long1"))*cos(Organ:("lat2"))*sin(Organ:("long2")) +
sin(Patient:("lat1"))*sin(Organ:("lat2"))))
```

```
</OPTCALC>
</FIELDDEF>
</DATADEF>
```

6.8 Allocation Method Definition

DefMethods.txt contains a list of allocation method definitions, consisting of a list of default rule definitions. Within a rule definition, the <GROUDEF> block defines the subset of organs the rule applies to. Fields used in the group definition must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the groupdef section) are joined using a logical AND. When an organ arrives, the program checks the allocation methods in the order in which they are presented to find the first one that specifies a group the organ belongs to. If the organ characteristics don't match any of the groups, the organ is discarded and an error message is written to the input stream errors output file.

The lines in the <ORDERDEF> block specify subsets of the candidate list. The order of the order definitions determines the order in which subsets of the waiting list are handled. Within a subset, candidates are sorted by the sortfields.

Rule List: <RULELIST>…</RULELIST>
The rule list tag is the outer-most tag. All other blocks are contained inside the rule list start and end tags.

Allocation Definition: <ALLOCDEF>…</ALLOCDEF>
The allocation definition block defines the allocation rules for one group of organs. There may be multiple allocation definition blocks within the rulelist block. The format for the allocation definition block is:

```
<ALLOCDEF>
title
<GROUPDEF>
source:fieldname=level
...
</GROUPDEF>
<ORDERDEF>                    these groups | are sorted this way
patfield=level,...|sortfield,...
patfield=level,...|sortfield,...
...
</ORDERDEF>
</ALLOCDEF>
```

source:fieldname=level defines the group an organ belongs to, based on its characteristics. The source is always organ. The field name is any field in the default or optional data definition files that applies to the organ. Level is the value this field must have for this organ to be in this group. The organ characteristics must match all definitions in the group.
patfield=level defines a subset of candidates to whom this order definition applies.
sortfield defines the sort order of the subset of candidates who meet the candidate criteria of this order definition statement.

Example:

```
<RULELIST>
<ALLOCDEF>
Default Adult Donor
<GROUPDEF>
Organ:DonAgeGrp=Adult
</GROUPDEF>
<ORDERDEF>
MedUrgStat=1A,GeogLevel=LOCAL|AllocPoints,Stat1WaitTime,TotWaitTime
MedUrgStat=1A,GeogLevel=REGXLOC|AllocPoints,Stat1WaitTime,TotWaitTime
MedUrgStat=1B,GeogLevel=LOCAL|AllocPoints,Stat1WaitTime,TotWaitTime
MedUrgStat=1B,GeogLevel=REGXLOC|AllocPoints,Stat1WaitTime,TotWaitTime
MedUrgStat=0,GeogLevel=LOCAL, ABOIdentOrCom-
pat=True|TruncMELD,ABOScore,MeldTime,MeldScore
MedUrgStat=0,GeogLevel=REGXLOC, ABOIdentOrCom-
pat=True|TruncMELD,ABOScore,MeldTime,MeldScore
MedUrgStat=1A,GeogLevel=NATXREG|AllocPoints,Stat1WaitTime,TotWaitTime
MedUrgStat=0,GeogLevel=NATXREG, ABOIdentOrCom-
pat=True|TruncMELD,ABOScore,MeldTime,MeldScore
</ORDERDEF>
</ALLOCDEF>
</RULELIST>
```

6.9 Allocation Score Boost Definition

DefBoostDef.txt is used to specify rules that define how scores used to sort candidates might be boosted. If the offered organ meets specified criteria (listed in the <ORGGRP> block) and the candidate meets specified criteria (listed in the <PATGRP> block), then the value of that candidate’s score will be boosted in the defined manner (defined in the <BOOST> block) for this particular organ being allocated. The boost consists of multiplying the existing score by a value and then adding a second value. Boost definitions may only reference the fields that are described in either the default data definition file or the optional data definition file.

Boost List: <BOOSTLIST>…</BOOSTLIST>
The boost list tag is the outer-most tag. All other blocks are contained inside the boost list start and end tags.

Boost Definition: <BOOSTDEF>…</BOOSTDEF>
The boost definition block contains the definition of one boost rule. Multiple boost rules may be contained within the boost list block.

There are two possible formats for the boost definition block. The format for the standard boost definition block is:

```
<BOOSTDEF>
boostname
<ORGGRP>
fldname=level
…
</ORGGRP>
<PATGRP>
fldname=level
…
</PATGRP>
<BOOST>
fldname|boostamnt|boostfact
…
</BOOST>
</BOOSTDEF>
```

boostname: gives the name of this boost rule.
fldname=level: specifies the criteria the organ and candidate must meet for the candidate to get this boost.
fldname: within the <BOOST> block specifies the field in the candidate record that will get the new, booostedscore, where boostedscore = **fldname** * **boostfact** + **boostamnt.**

Example:

```
<BOOSTLIST>
<BOOSTDEF>
ABO Compatible Allocation Points
```

```
<ORGGRP>
</ORGGRP>
<PATGRP>
ABOCompat=C
</PATGRP>
<BOOST>
AllocPoints|5|1
</BOOST>
</BOOSTDEF>
</BOOSTLIST>
```

The alternate boost definition block allows users to define a linear equation by which to boost the underlying variable. Note that this alternate boost definition can be entered only through input files. There are no input panels in LSAM from which the inputs may be made.

The format for the alternate, linear equation, boost definition block is:

```
<BOOSTDEF>
boostname
<ORGGRP>
fldname=level
...
</ORGGRP>
<PATGRP>
fldname=level
...
</PATGRP>
<CALCDEF>
fldname
<TERMDEF>
coeff1|source1a:fieldname1a|source1b:fieldname1b
...
</TERMDEF>
</CALCDEF>
</BOOSTDEF>
```

boostname: gives the name of this boost rule.
fldname: specifies the field in the patient record that will get the new, boostedscore, where boostedscore = **fldname** *operator* (coeff1*source1a:fieldname1a*source1b:fieldname1b) *operator* ... *operator* (coeff(i)*source(i)a:fieldname(i)a*source(i)b:fieldname(i)b).

operator: the optional operator field is used to indicate the mathematical operation to perform. The default is Sum. Other option is Product. The example below yields this equation:

lyft=lyft + (21.7258 + (-0.1944 * can_age) + (-0.1852*can_age*can_dgn_kp_dm)) * 0.8*one_minus_DPI_rank

```
<BOOSTLIST>
<BOOSTDEF>
Life Years From Transplant (LYFT) Equation
<CALCDEF>
lyft
<TERMDEF>
21.7258
-0.1944|Patient:can_age
-0.1852|Patient:can_age|Patient:can_dgn_kp_dm
Product|0.8|Organ:one_minus_DPI_rank
</TERMDEF>
</CALCDEF>
</BOOSTLIST>
</BOOSTDEF>
```

6.10 Default Acceptance

DefAccept.txt is the default definition of the calculation to perform to determine whether a candidate accepts an offered organ.

Acceptance List: <ACCLIST>…</ACCLIST>
The acceptance list tag is the outer-most tag. All other blocks are contained inside the acceptance list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>
Each calculation definition block defines the calculation that will take place, the mathematical function that will be used, and the characteristics of the organ and/or candidate that will be inputs to the mathematical function. The format for the calculation definition block is:

```
<CALCDEF>
accname
<GROUPDEF>
source:fieldname=level
```

```
...
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
...
</TERMDEF>
<FUNCTION>
funcname
</FUNCTION>
</CALCDEF>
```

accname is the name of this acceptance calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ that will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists only of the coefficient.

The function section identified the function to be applied to the result of the linear combination specified in the termdef section. The inverse logit function, InvLogit(y) = Exp(y)/(1+Exp(y)), where y = the result of the linear combination specified in the termdef section, is the only choice available. Specify INVLOGIT for the **funcname**.

Example:

```
<ACCLIST>
<CALCDEF>
Sample Acceptance
<GROUPDEF>
</GROUPDEF>
<TERMDEF>
3
```

```
0.02|Patient:PatientAge
-0.04|Organ:DonorAge
</TERMDEF>
<FUNCTION>
INVLOGIT
</FUNCTION>
</CALCDEF>
</ACCLIST>
```

6.11 Default Post-Graft Survival Definition

DefSurvival.txt is the default definition of the calculation to perform to determine how long a patient will live after receiving a graft and to assign a series of possible outcomes, such as relisting and status changes, to that patient before graft failure.

Survival List: <SURVLIST>…</SURVLIST>
The survival list tag is the outer-most tag. All other blocks are contained inside the survival list start and end tags.

Calculation Definition: <CALCDEF>…</CALCDEF>
Each calculation definition block defines the characteristics of the organ and/or the patient to which this survival calculation applies. The format for the calculation definition block is:

```
<CALCDEF>
survname
<GROUPDEF>
source:fieldname=level
…
</GROUPDEF>
<TERMDEF>
coeff|source1:fieldname1|source2:fieldname2
…
</TERMDEF>
<FUNCTION>
funcname
<STEPFUNC>
<STEP>
value|time
<OUTCOMES>
outtype|outcomename|outprob|outtime
<UPDATES>
```

```
atime|medurgstat|meldscore|opt(1)|…
…
</UPDATES>
…
</OUTCOMES>
</STEP>
…
</STEPFUNC>
</FUNCTION>
</CALCDEF>
```

survname is the name of this survival calculation.

Fields used to define groups must be category fields that are defined in either the default data definition file or the optional data definition file. Group components (lines in the <GROUPDEF> block) are joined using a logical AND.
source:fieldname=level: lists the numeric fields associated with the patient and the organ that will be inputs to the mathematical function. Source: {Patient, Organ}.

Fields used to define terms must be score fields that are defined in either the default data definition file or the optional data definition file. Each term component (line in the <TERMDEF> block) consists of the coefficient (**coeff**) times the first field (**source1:fieldname1**) times the second field (**source2:fieldname2**). Term components (lines in the <TERMDEF> block) are added together. If the second field of a term component is omitted, then the term component consists of the coefficient times the first field. If both fields are omitted, then the term component consists only of the coefficient.

The <FUNCTION> block identifies the type of function to be used to determine survival time. The choices for **funcname** are STEPFUNC or WEIBULL. If **funcname** is StepFunc, then the next block will be <STEPFUNC>. If **funcname** is Weibull, then the next block will be <WEIBULL>

The <STEPFUNC> block identifies the step function to be inverted to determine survival time:

Prob[survival to time > T] = $S(T)^{\exp(y)}$
where y = the result of the linear combination specified in the termdef section.

The <STEPFUNC> block defines the function as a set of steps connecting the listed points. Associated with each point is a **value** (probability) and a **time** (in days) until failure.

The <WEIBULL> block identifies the values of the parameters and form of a Weibull equation.

parm1|parm2|form name

The parameters required depend on the chosen form. For example, if the chosen form is "Intercept," then the required parameters are shape and intercept. The following table outlines the Weibull equation form choices and their required parameters.

Weibull Equation form	**parm1**	**parm2**	**form name**
AFT: exp[-exp(shape*(ln(t)-intercept))]	shape	intercept	Intercept
PH: exp[-scale * t^shape]	shape	scale	Lambda
PH: exp[-(t/scale)^shape]	shape	scale	Alpha

Whether the death date is determined by a Cox PH StepFunction or a Weibull Distribution, the patient outcomes before death are determined by a series of steps that depend on the days to death.

Each <STEP> block refers to patients with a certain number of days until death. Within each step, there can be one or more outcomes.

Each <OUTCOME> block includes a probability (**outprob**) and a time (**outtime**, in days) before failure when the outcome occurs. For a death outcome, the time is zero. For a relist outcome, the time is the number of days before failure when relist occurs.

Associated with each relist outcome is a set of status updates (<UPDATES> block) for the relisted patient. The relisted patient updates consist of a time (**atime**, in days) after relist when the update occurs. The set of updates does not include an update for death, since that event will be generated from the failure time for the step. The format of <UPDATES> block is the same as that for the Status Updates input file, which is described in section 6.3, except that the patient ID is not specified because these post-graft survival status updates may be applied to any patient who undergoes transplant.

Example:
```
#
<SURVLIST>
<CALCDEF>
Status 1 patient survival
<GROUPDEF>
</GROUPDEF>
```

```
<TERMDEF>
Patient:MedUrgStat=1
</TERMDEF>
<FUNCTION>
StepFunc
<STEPFUNC>
.9823|0
.9615|1
.9365|3
.9212|6
</STEPFUNC>
<STEP>
1.00
<OUTCOMES>
Death|Death|0.95
Relist|Relist Group 1|0.05|7.0
<UPDATES>
0.0|1|25
</UPDATES>
</OUTCOMES>
</STEP>
<STEP>
0.90
<OUTCOMES>
Death|Death|0.90
Relist|Relist Group 1|0.05|30.0
<UPDATES>
0.0|1|25
</UPDATES>
Relist|Relist Group 2|0.05|60.0
<UPDATES>
0.0|2A|20
30.0|1|25
</UPDATES>
</OUTCOMES>
</STEP>
</FUNCTION>
</CALCDEF>
</SURVLIST>
```

6.12 Model Allocation Runs

ModelRuns.txt is created by the program, not by the user. The ModelRuns.txt contains the names and file prefixes of all Allocation Runs that have been defined using the program. The model saves the specification when the user selects the OK button or the Run button in the user interface.

Annotated Example: (the annotations in italics do not appear in the input file)

NAME OPTN Test	*name of the Allocation Run*
FILE OPTN	*short name used as file prefix*
EOF	*end of definition*
NAME Meld Test	*name of another run*
FILE Meld	*short name used as second file prefix*
EOF	*end of second definition*

6.13 Allocation Run Definition

xxxxModel.inp is generated by the program and is used to save an Allocation Run definition (for Allocation Run xxxx). Each file contains the specifications of one of the defined Allocation Runs. The model saves the specification when the user selects the OK button or the Run button in the user interface.

The definitions include paths where the directory can locate the input files for the Allocation Run. If the Allocation Run definition is to be shared among users on different computers, the path definitions must be edited to correspond to the locations on the new machine where the program can find the files.

In addition, the Allocation Run definition file will contain the entire configuration of the Allocation Run, which will look similar to the contents of the input files previously defined.

7 Output File Specifications

This chapter gives the specifications of output files for the program. The specifications of each output file lists fields in the order in which they occur in a record, followed by an example. Most of these files are generated automatically with every LSAM run, but the match list, status updates, and event log are optional outputs, controlled by checkboxes on the Run Progress screen. The format of each table list gives the name of the field, its data type, and a short definition. The field names are those used in the source code. The program assigns each output file a different name by prefixing the short title of the Allocation Run.
Note: *xxxx is the Short Name given during the Model Allocation Run.*

Output Files:

- Automatically created
  - Waiting list at end of Model Allocation Run.
  - Patients receiving grafts.
  - Death list (candidates who died during the Model Allocation Run).
  - Removed list (candidates who were removed from the waiting list during the Model Allocation Run).
  - Graft list (all organs grafted during the Model Allocation Run and the candidates receiving the grafts).
  - Summary statistics.
- Optionally created
  - Match list (all candidates to whom each organ was offered).
  - Status updates (log of the arriving status update events).
  - Event log (log of the arriving events).

The following four output files all consist of the same layout. For simplicity, the output layout is specified only once. All are text files, one line per record, "|" delimiter separating fields.

7.1 Waiting List

xxxxWait.out: list of candidate records that represents the status of the waiting list at the end time of the Model Allocation Run.

7.2 Grafted Patients

xxxxGrPat.out: list of patient records that represents all candidates receiving a graft during the Model Allocation Run. Recipients receiving more than one graft appear multiple times.

7.3 Death Listing

xxxxDeath.out: list of candidate records that represents all patients who died during the Model Allocation Run. It includes waitlist deaths, removed candidate deaths, and post-transplant deaths.

7.4 Removed Listing

xxxxRemove.out: list of patients who were removed from the waiting list during the Model Allocation Run.

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number
SnapDate	DateTime	Snapshot for existing waiting list date-time (mm/dd/yyyy hh:mm:ss)
ArrivalTime	DateTime	Date-time arriving or relisting on waiting list (mm/dd/yyyy hh:mm:ss)
lsam_cand_id	string	Item ID (patient ID)
CenterID	string	Transplant center ID
PatientAge	double	Patient age at time of listing
ABOType	string	ABO blood type {O|A|A1|A2|B|AB|A1B|A2B}
OrgUrgStat	string	Original medical urgency status at time of listing {1|2A|2B|3|7}
MedUrgStat	string	Current medical urgency status {1|2A|2B|3|7}
OrgMeld	double	Original MELD score at time of listing {0.0 … 40.0}
MeldScore	double	Current MELD score {0.0 … 40.0}
PrevTrans	boolean	Current liver is a graft {True|False}
WillATwo	boolean	If type O, will accept A2 donor {True|False}
WillIncBlood	boolean	Will accept incompatible blood type {True|False}
CumActiveTime	double	Cumulative waiting time {days}
CumStat1Time	double	Cumulative time in status 1 {days}
CumStat1ATime	double	Cumulative time in status 1 or 1A {days}
CumStat1BTime	double	Cumulative time in status 1, 1A or 1B {days}
CumStat2ATime	double	Cumulative time in status 1, 1A, 1B or 2A {days}
CumStat2BTime	double	Cumulative time in status 1, 1A, 1B, 2A or 2B {days}
CumStat3Time	double	Cumulative time in status 1, 1A, 1B, 2A, 2B or 3 {days}
CumInactTime	double	Cumulative time in inactive status {days}
Meld6WaitTime	double	Cumulative waiting time in Meld 6 (or lower) {days}
…	double	… (Melds 7-39)

Meld40WaitTime	double	Cumulative waiting time in Meld 40 (or lower) {days}
Optional Fields (1..n)		Optional Fields defined in the Optional Data Definition file
GraftCount	double	Number of grafts, including this one, that this patient has received
DeathDate	DateTime	Date-time the patient died in the model (mm/dd/yyyy hh:mm:ss)
RelistDate	DateTime	Date-time of patient relisting (mm/dd/yyyy hh:mm:ss)
lsam_don_id	string	Organ identifier
{AllocPoints}	double	Allocation points this patient was awarded for this organ
{ProbAccept}	double	Calculated probability that this patient would accept this organ
{GraftDate}	DateTime	Date-time of transplant (mm/dd/yyyy hh:mm:ss)

Example: (long line has been wrapped)

```
1|10/01/2001 02:39:28|09/30/2001
22:29:45|183529|739|55.00|A|2B|2B|20|20|False|False|False|
0.17|0|0|0|0|0.17|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0|0|0|0|0|0|0|0|0|99|60|350|False|
False|False|M|White|2|Liver : Non-Cholestatic Cirrhosis|1|12/18/2009
02:39:28|0.0|OJA010
|0|0.08|10/01/2001 02:39:28
```

7.5 Grafted Organs

xxxxGraft.out: list of all organs grafted during the Model Allocation Run and the patients receiving the grafts.

Field Name	Data Type	Short Definition
Iter	integer	Replication number
lsam_don_id	string	Organ ID
EventTime	DateTime	Date-time of graft (mm/dd/yyyy hh:mm:ss)
OrganInstId	string	Donor Institution ID
DonorAge	double	Donor age
OrganABO	string	ABO blood type {O|A|A1|A2|B|AB|A1B|A2B}
lsam_cand_id	string	Patient ID
ProbAccept	double	Probability of acceptance calculated by the model
PatCenterID	string	Transplant center ID
PatientAge	double	Patient age at time of listing
PatientABO	string	ABO blood type {O|A|A1|A2|B|AB|A1B|A2B}

OrgUrgStat	string	Original medical urgency status at time of listing {1|1A|1B|2A|2B|3|7}
MedUrgStat	string	Current medical urgency status {1|1A|1B|2A|2B|3|7}
OrgMeld	double	Original MELD score at time of listing {0.0 … 40.0}
MeldScore	double	Current MELD score {0 … 40}
CumActiveTime	double	Cumulative waiting time {days}
CumOneTime	double	Cumulative time in status 1 {days}
CumOneATime	double	Cumulative time in status 1A {days}
CumOneBTime	double	Cumulative time in status 1B{days}
CumTwoATime	double	Cumulative time in status 2A {days}
CumTwoBTime	double	Cumulative time in status 2B {days}
CumThreeTime	double	Cumulative time in status 3 {days}
CumInactTime	double	Cumulative time in inactive status {days}
ArrivalTime	DateTime	Date-time arriving on the waiting list (mm/dd/yyyy hh:mm:ss)

Example: (long line has been wrapped)

```
1|OJA010|10/01/2001
02:39:28|741|64.6|A|183529|0.08|739|55.00|A|20|0.17|0|0|0.17|0|0|
09/30/2001 22:29:45
```

7.6 Match Offers

xxxxMatch.out: list of all candidates to whom each organ was offered. Produces only if the log match box was checked on the Run Progress screen at the start of the run. Since this option writes one record for each offer made, it can occupy a significant amount of disk space, particularly when multiple iterations are being performed.

It-
er|lsam_don_id|EventTime|OrganInstId|OrganLocId|OrganRegId|DonorAge|OrganABO|
Lsam_cand_id|OfferNumber|Decision|ProbAccept|SortField0|Sort0|SortField1|Sort1|Sor
tField2|Sort2|SortField3|Sort3|PatCenterID|PatLocId|PatRegId|PatientAge|PatientABO|
Or-
gUrgStat|MedUrgStat|OrgMeld|MeldScore|CumActiveTime|CumOneTime|CumOneATim
e|CumOneBTime|CumTwoATime|CumTwoBTime|CumThreeTime|CumInactTime|AllocRu
leName|AllocRuleEntry

Field Name	**Data Type**	**Short Definition**
Iter	integer	Replication number
lsam_don_id	string	Organ ID
EventTime	DateTime	Organ arrival date-time (mm/dd/yyyy hh:mm:ss)
OrganInstId	string	Donor institution ID
OrganLocId	string	Donor location ID (four-character center abbreviation)
OrganRegId	string	Donor region
DonorAge	double	Donor age
OrganABO	string	ABO blood type {O|A|A1|A2|B|AB|A1B|A2B}
lsam_cand_id	string	Patient ID (or Discard if no patient accepted organ)
OfferNumber	double	Number of times this organ has been offered, counting this offer
Decision	boolean	Patient accepted the offered organ
ProbAccept	double	Probability of acceptance calculated by the model
SortField0	string	Name of the first field used to sort candidates in this category
Sort0	double	Value of the first sort field for this candidate
SortField1	string	Name of the second field used to sort candidates in this category
Sort1	double	Value of the second sort field for this candidate
SortField2	string	Name of the third field used to sort candidates in this category
Sort2	double	Value of the third sort field for this candidate
SortField3	string	Name of the fourth field used to sort candidates in this cate-

		gory
Sort3	double	Value of the fourth sort field for this candidate
PatCenterID	string	Transplant center ID
PatLocId	string	Transplant location ID (four-character center abbreviation)
PatRegId	string	Transplant center region
PatientAge	double	Patient age at time of listing
PatientABO	string	ABO blood type {O|A|A1|A2|B|AB|A1B|A2B}
OrgUrgStat	string	Original medical urgency status at time of listing {1|2A|2B|3|7}
MedUrgStat	string	Current medical urgency status {1|2A|2B|3|7}
OrgMeld	double	Original MELD score at time of listing {0.0 … 40.0}
MeldScore	double	Current MELD score {0.0 … 40.0}
Cu-mActiveTime	double	Cumulative waiting time {days}
CumOneTime	double	Cumulative time in status 1 {days}
CumO-neATime	double	Cumulative time in status 1A {days}
Cu-mOneBTime	double	Cumulative time in status 1B {days}
CumTwoATime	double	Cumulative time in status 2A {days}
CumTwoB-Time	double	Cumulative time in status 2B {days}
CumThreeTime	double	Cumulative time in status 3 {days}
CumInactTime	double	Cumulative time in inactive status {days}
AllocRu-leName	string	Name of the allocation rule which was used for this offer
AllocRuleEntry	integer	Number (0-based) of the line of the allocation rule used for this offer

Example:
1|TCH03634|01/01/2006
00:37:06|200|INOP|10|64|B|396300|39|Declined|0.00271106225780784|AllocPoints|10||0||0||0|434|NYRT|9|12.483505800033|O|1A|1A|0|0|6.47567129629169|0|6.47567129629169|0|0|0|0|0|Default Adult Other ABO Donor|8

7.7 Status Updates

xxxxStatus.out: log of the arriving status update events. This file is produced only if requested during the Model Allocation Run.

Field Name	Data Type	Short Definition
Iter	integer	Replication number
EventTime	DateTime	Date-time of arrival of the update (mm/dd/yyyy hh:mm:ss)
Lsam_cand_id	string	Patient ID
MedUrgStat	string	New medical urgency status {1|1A|1B|2A|2B|3|7}
MeldScore	double	New MELD score {0 … 40}
Optional Fields (1..n)		Optional Fields defined in the Optional Data Definition file

Example:
1|10/01/2001 12:00:31|167712|2B|19

7.8 Event Log

EventLog.log: log of the arriving events. This file is produced only if requested.

Field Name	Data Type	Short Definition
EventTime	DateTime	Date-time of arrival of the update (mm/dd/yyyy hh:mm:ss)
EventID	string	Event ID {PatientID|OrganID}
EventDesc	string	Description of the event

Example:
01/02/2001 17:12:15 OI4006 Organ arrives.

7.9 Summary

xxxxSummary.out: summary statistics for each iteration, followed by the mean and standard deviation of each statistic across all the iterations in the Allocation Run.

Field Name	Data Type	Short Definition
Iteration	integer	block of summary statistics for each iteration
Initial Waiting list	integer	patients on waiting list at start of run
Patient Listings	integer	new patient arrivals between start and end date
Relistings	integer	relisted between start and end date
Removed from Waiting list	integer	status 9 (removed) between start and end date
Final Waiting list	integer	patients on waiting list at end time

Discarded Organs	integer	number of organs discarded between start and end date
Median Days Wait For Graft	double	median days on waiting list before graft
Transplanted Organs	integer	number of organs transplanted between start and end date
Retransplants	integer	subset of transplants that were retransplants
Pediatric Transplants	integer	transplants to pediatric recipients
Status 1 Transplants	integer	transplants to status 1 patients
Status 1A Transplants	integer	transplants to status 1A patients
Status 1B Transplants	integer	transplants to status 1B patients
MELD > 40 Transplants	integer	transplants to patients with MELD > 40
MELD 35-40 Transplants	integer	transplants to patients with 35 ≤ MELD ≤ 40
MELD 30-34 Transplants	integer	transplants to patients with 30 ≤ MELD < 35
MELD 25-29 Transplants	integer	transplants to patients with 25 ≤ MELD < 30
MELD 20-24 Transplants	integer	transplants to patients with 20 ≤ MELD < 25
MELD 15-19 Transplants	integer	transplants to patients with 15 ≤ MELD < 20
MELD < 15 Transplants	integer	transplants to patients with MELD < 15
Local Transplants	integer	transplants to patients in same local unit as donor
Regional Transplants	integer	transplants to patients in same region as donor (not local unit)
National Transplants	integer	transplants to patients in different region than donor
Local Status 1 Transplants	integer	transplants to local status 1 patients
Local Status 1A Transplants	integer	transplants to local status 1A patients
Local Status 1B Transplants	integer	transplants to local status 1B patients
Local MELD > 40 Transplants	integer	transplants to local patients with MELD > 40
Local MELD 35-40 Transplants	integer	transplants to local patients with 35 ≤ MELD ≤ 40
Local MELD 30-34 Transplants	integer	transplants to local patients with 30 ≤ MELD < 35
Local MELD 25-29 Transplants	integer	transplants to local patients with 25 ≤ MELD < 30
Local MELD 20-24 Transplants	integer	transplants to local patients with 20 ≤ MELD < 25
Local MELD 15-19 Transplants	integer	transplants to local patients with 15 ≤ MELD < 20
Local MELD < 15 Transplants	integer	transplants to local patients with MELD < 15
Regional Status 1 Transplants	integer	transplants to regional, status 1 patients
Regional Status 1A Transplants	integer	transplants to regional, status 1A patients
Regional Status 1B Transplants	integer	transplants to regional, status 1B patients
Regional MELD > 40 Transplants	integer	transplants to regional patients with MELD > 40

Regional MELD 35-40 Transplants	integer	transplants to regional patients with 35 ≤ MELD ≤ 40
Regional MELD 30-34 Transplants	integer	transplants to regional patients with 30 ≤ MELD < 35
Regional MELD 25-29 Transplants	integer	transplants to regional patients with 25 ≤ MELD < 30
Regional MELD 20-24 Transplants	integer	transplants to regional patients with 20 ≤ MELD < 25
Regional MELD 15-19 Transplants	integer	transplants to regional patients with 15 ≤ MELD < 20
Regional MELD < 15 Transplants	integer	transplants to regional patients with MELD < 15
National Status 1 Transplants	integer	transplants to national status 1 patients
National Status 1A Transplants	integer	transplants to national status 1A patients
National Status 1B Transplants	integer	transplants to national status 1B patients
National MELD > 40 Transplants	integer	transplants to national patients with MELD > 40
National MELD 35-40 Transplants	integer	transplants to national patients with 35 ≤ MELD ≤ 40
National MELD 30-34 Transplants	integer	transplants to national patients with 30 ≤ MELD < 35
National MELD 25-29 Transplants	integer	transplants to national patients with 25 ≤ MELD < 30
National MELD 20-24 Transplants	integer	transplants to national patients with 20 ≤ MELD < 25
National MELD 15-19 Transplants	integer	transplants to national patients with 15 ≤ MELD < 20
National MELD < 15 Transplants	integer	transplants to national patients with MELD < 15
ABO Identical Transplants	integer	transplants to patients with ABO type identical to donor
ABO Compatible Transplants	integer	transplants to patients with ABO type compatible with donor
ABO Incompatible Transplants	integer	transplants to patients with ABO type incompatible with donor
Total Deaths	integer	deaths of patients within run period
Waitlist Deaths	integer	deaths of patients while on the waiting list (not relisted)
Initial Listing Status 1 Waitlist Deaths	integer	deaths of status 1 patients on waiting list (not relisted)
Initial Listing Status 1A Waitlist	integer	deaths of status 1A patients on waiting list (not

Deaths		relisted)
Initial Listing Status 1B Waitlist Deaths	integer	deaths of status 1B patients on waiting list (not relisted)
Initial Listing, MELD > 40 Waitlist Deaths	integer	deaths of MELD > 40 patients on waiting list (not relisted)
Initial Listing, MELD 35-40 Wait-list Deaths	integer	deaths of MELD 35-40 patients on waiting list (not relisted)
Initial Listing, MELD 30-34 Wait-list Deaths	integer	deaths of MELD 30-34 patients on waiting list (not relisted)
Initial Listing, MELD 25-29 Wait-list Deaths	integer	deaths of MELD 25-29 patients on waiting list (not relisted)
Initial Listing, MELD 20-24 Wait-list Deaths	integer	deaths of MELD 20-24 patients on waiting list (not relisted)
Initial Listing, MELD 15-19 Wait-list Deaths	integer	deaths of MELD 15-19 patients on waiting list (not relisted)
Initial Listing, MELD < 15 Waitlist Deaths	integer	deaths of MELD < 15 patients on waiting list (not relisted)
Initial Listing, Status Inactive Waitlist Deaths	integer	deaths to status 7 (inactive) patients on waiting list (not relisted)
Initial Listing, Status Other Wait-list Deaths	integer	deaths to unknown status patients on waiting list (not relisted)
Removed Patient Deaths	integer	deaths of patients who were removed from wait-ing list
Post-Graft Deaths	integer	deaths of patients who have received grafts (to-tal)
Post-Graft, Non-Relist Deaths	integer	deaths of patients who have received grafts (not relisted)
Post-Graft, Relist Deaths	integer	deaths of patients who received a graft and re-listed
Relisting Status 1 Waitlist Deaths	integer	deaths of status 1 patients who received a graft and relisted
Relisting Status 1A Waitlist Deaths	integer	deaths of status 1A patients who received a graft and relisted
Relisting Status 1B Waitlist Deaths	integer	deaths of status 1B patients who received a graft and relisted
Relisting MELD > 40 Waitlist Deaths	integer	deaths of MELD > 40 patients who received a graft and relisted
Relisting MELD 35-40 Waitlist Deaths	integer	deaths of MELD 35-40 patients who received a graft and relisted
Relisting MELD 30-34 Waitlist	integer	deaths of MELD 30-34 patients who received a

Deaths		graft and relisted
Relisting MELD 25-29 Waitlist Deaths	integer	deaths of MELD 25-29 patients who received a graft and relisted
Relisting MELD 20-24 Waitlist Deaths	integer	deaths of MELD 20-24 patients who received a graft and relisted
Relisting MELD 15-19 Waitlist Deaths	integer	deaths of MELD 15-19 patients who received a graft and relisted
Relisting MELD <15 Waitlist Deaths	integer	deaths of MELD < 15 patients who received a graft and relisted
Relisting, Inactive Waitlist Deaths	integer	deaths of inactive patients who received a graft and relisted
Relisting, Status Other Waitlist Deaths	integer	deaths of patients who received a graft and re-listed, current status unknown
Post-Graft Surviving	integer	patients with grafts still alive at end time
Median Post-Graft Life-Years	double	median years of life after graft (not relisted)
Transplants in Region r1	integer	transplants to patients in region r1
…	integer	
Transplants in Region rN	integer	transplants to patients in region rN
{…snip…}	integer	similar block for each replication
Average of Replications	double	block of averages and standard deviations across replications

8 Frequently Asked Questions

8.1 Data Input Parameters

Model Allocation Run Dates: 1/1/2010 through 1/1/2011
Livers available between 1/1/2010 and 12/31/2010
Waiting list statistics as of 12/31/2009
New Patients added between 1/1/2010 and 12/31/2010
Status Updates between 1/1/2010 and 12/31/2010

8.2 Summary Report: Standard Deviation Calculation

Each iteration represents an independent sample from an infinite set of possible iterations. The variance around each statistic is thus given by the sum of the squared differences from the mean, divided by number of iterations minus one. The standard deviation is the square root of the variance.

8.3 What is Median Days Wait for Graft?

For each patient who receives a graft, days wait for graft is the number of days between waitlisting and receiving a graft. It includes only patients who receive grafts during the user-specified period of statistics collection. (Users can direct statistics collection to begin at a time later than the start time of the simulation.) The program uses the sorted list of these numbers to determine the median.

8.4 What is Median Post-Graft Life-Years?

For each patient who receives a graft during the user-specified period of statistics collection, post-graft life-years is the difference between the scheduled date of death and the date of receiving a graft. The program uses the resulting sorted list of post-graft life-years to determine the median.

8.5 Do Transplanted Organs Include Retransplants?

Yes. It counts the organs that are transplanted.

8.6 Do Patient Listings include Relistings?

No. It includes only the initial listing.

8.7 Should the inputs and outputs balance?

Yes. This is the initial list + all entries to the list - all exits from the list.

```
  Initial waiting list
+ Patient listings
+ Relistings
-  Transplanted organs
-  Removed from waiting list
-  Waitlist deaths
-  Relisted patient deaths
===================
Final Waiting list
```

Importantly, the output file Death.out includes patients who died after undergoing transplant; that is, some candidates are listed both in the list of transplanted organs and in the list of deaths. The numbers will not balance until this overlap is accounted for.

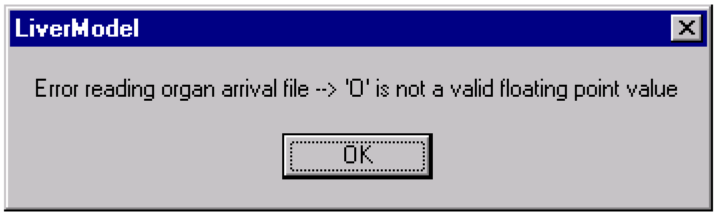
8.8 What records are included in Grafts.out vs. GrPats.out?

Grafts.out contains one record for each organ transplanted. GrPats.out contains one record for each transplant that occurred during the model run, including information about the patient as well as the organ.

8.9 What are the types of Input/Output Errors?

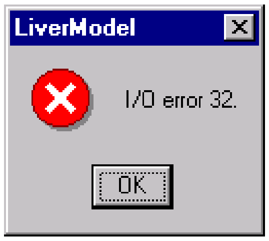
Typically, the LSAM program will not run with errors in the input files. Below are the most frequent errors encountered.

If a value read from a text file does not conform to the expected format, an error message will be displayed during the Model Allocation Run specifying the location of the error. The input data must be corrected and the model restarted.

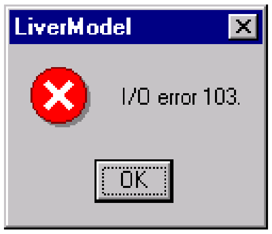

LiverModel

Error reading organ arrival file --> '0' is not a valid floating point value

If an output file is in use by another program, i.e., Excel or Word, the error message I/O error 32 appears. Close the file in use and restart the model run. If a file you have selected in your input parameters does not exist, the error message I/O error 103 appears.


LiverModel

I/O error 32.


LiverModel

I/O error 103.

9 Additional Resources

OPTN maintains a glossary of terminology related to transplantation: http://optn.transplant.hrsa.gov/resources/glossary.asp

SRTR maintains summary information on organ allocation policy, including flowcharts and descriptive articles: http://www.srtr.org/allocationcharts/Default.aspx

The LSAM data input files are based on the SRTR Standard Analysis File (SAF), and the SAF data dictionary may be useful in understanding these fields in more detail: http://www.srtr.org/data_request/saf.aspx. Requests for SAF data may be made to the SRTR: http://www.srtr.org/data_request/datause_policy.aspx

Please send questions to SRTR: srtr@srtr.org or 1-877-970-SRTR.