

Introduction to Mathematical Modeling using Magnolia

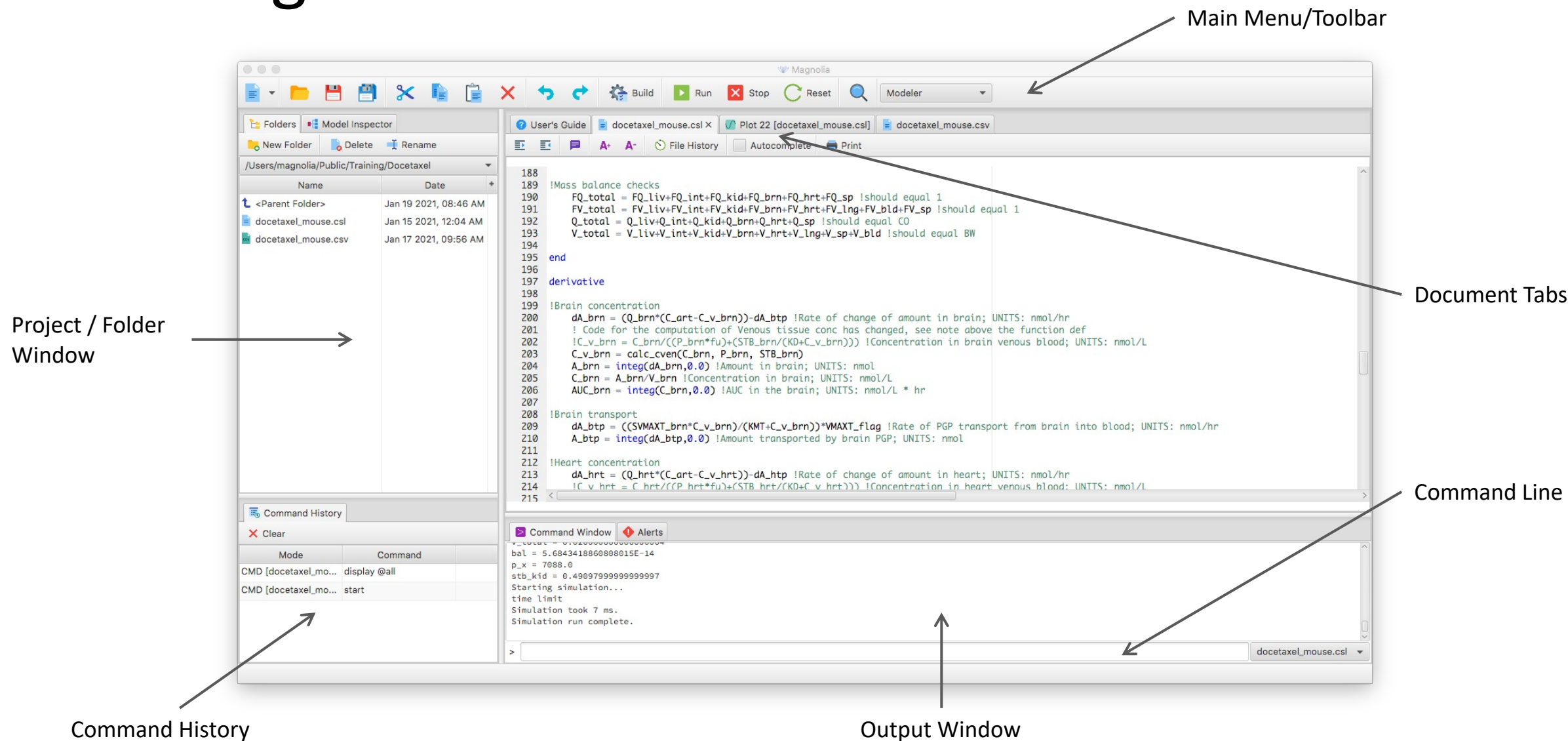
With Applications to Pharmacokinetic Analysis

Section 2: Overview of the Magnolia User
Interface and Modeling and Analysis Workflow

Outline

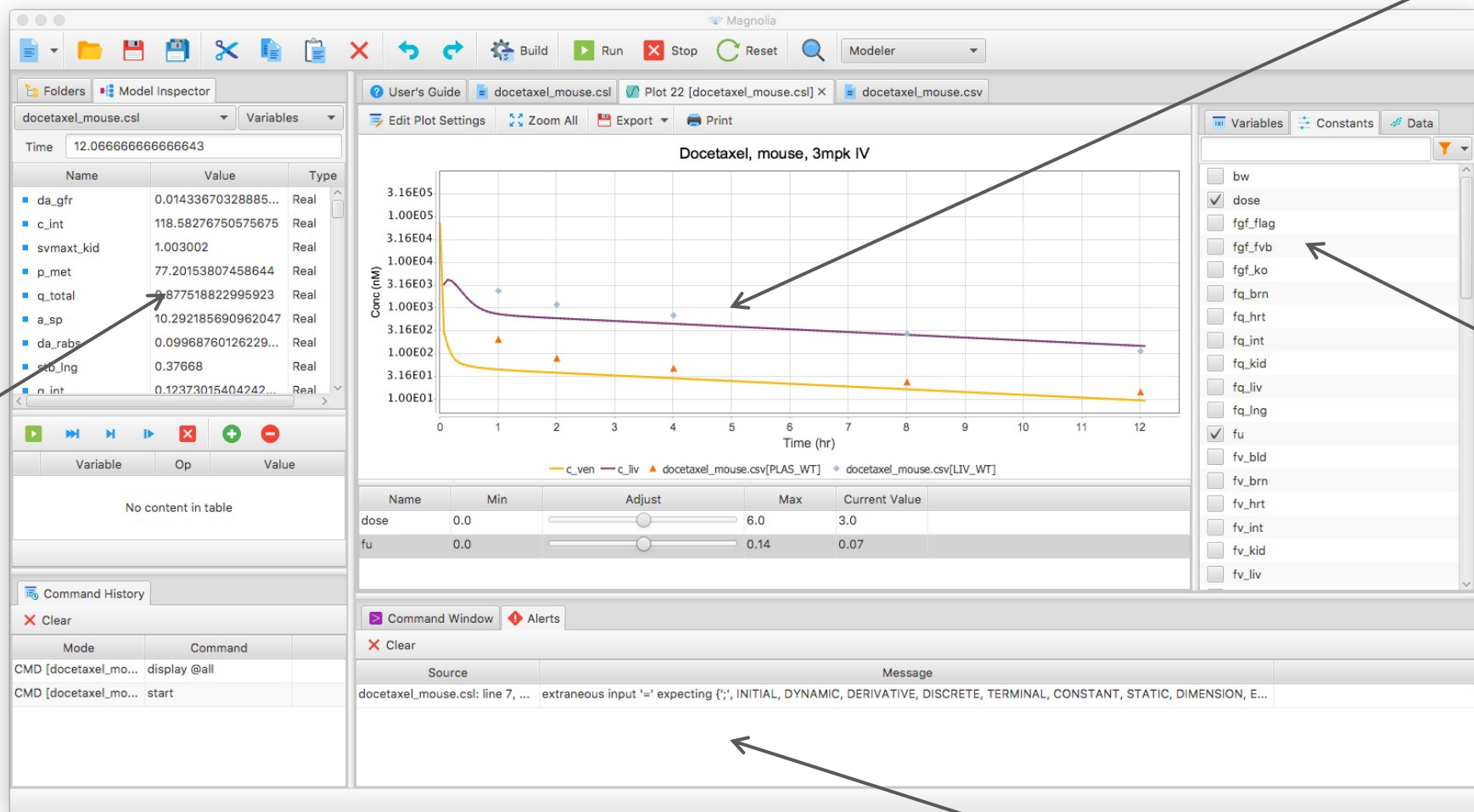
1. Introduction to Magnolia
- 2. Overview of the Magnolia user interface and modeling and analysis workflow**
3. Specifying models using the CSL language
4. Scripting simulation runs using the CMD and Python languages
5. Additional analytical methods: parameter estimation, sensitivity analysis, Monte-Carlo analysis

The Magnolia Environment



The Magnolia Environment

Model Inspector
Window

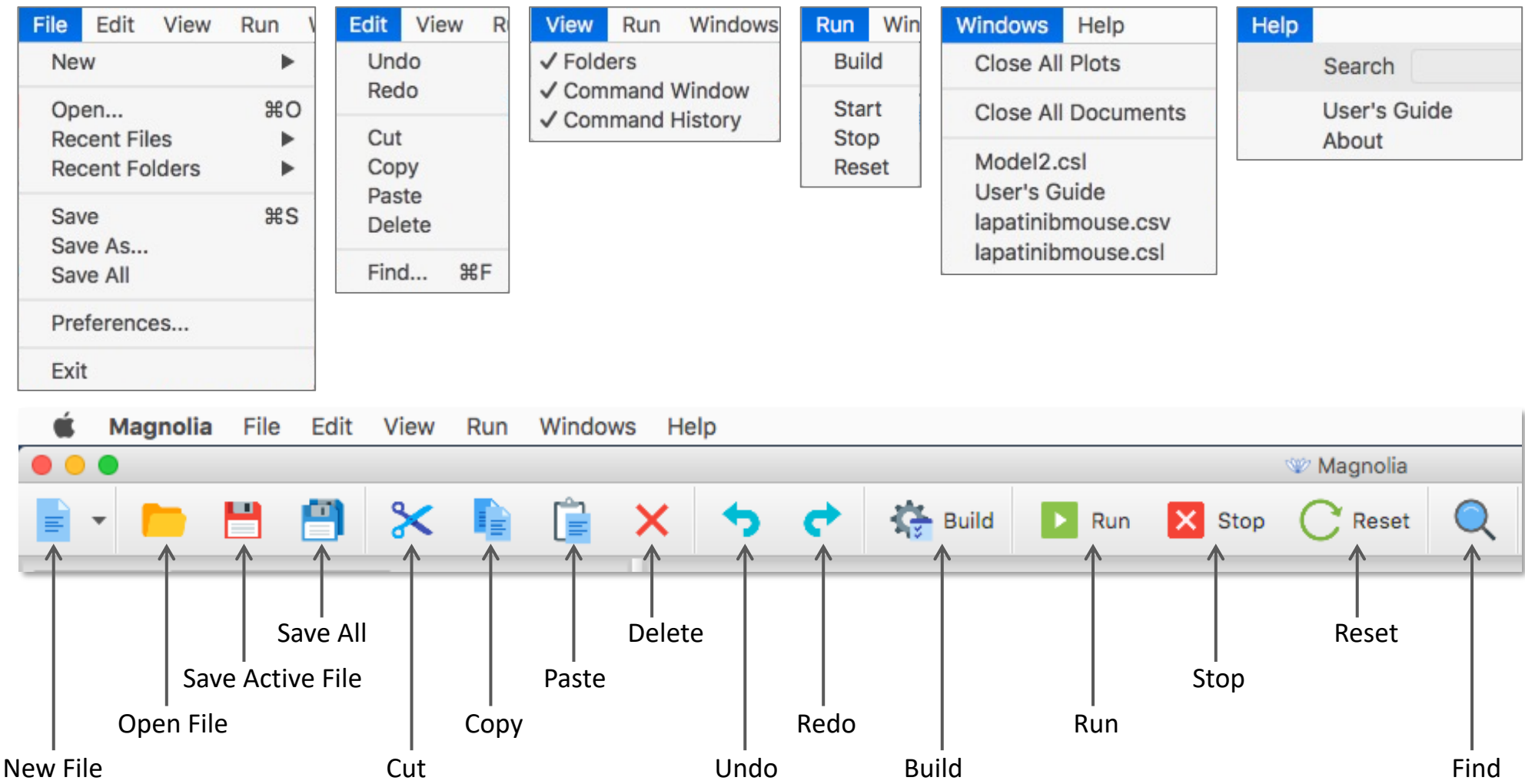


Interactive Plot
Window

Model Variable,
Parameter and Data
Sets Lists

Alerts List

Main Menu/Toolbar



Menu/Toolbar Item Groups

- **File:** opening, closing, saving files
 - See subsequent slide for the types of files recognized by Magnolia
- **Edit:** cut, copy, paste, delete, undo, redo
 - These commands operate on the active document; not all of these command will be relevant to every document type
- **View:** control visibility of some of the small windows surrounding the document tab area
- **Run:** operations related to controlling simulation runs
- **Windows:** select or close open document windows
- **Help:** open the “About” dialog or the Magnolia help system

Menu/Toolbar: “File”

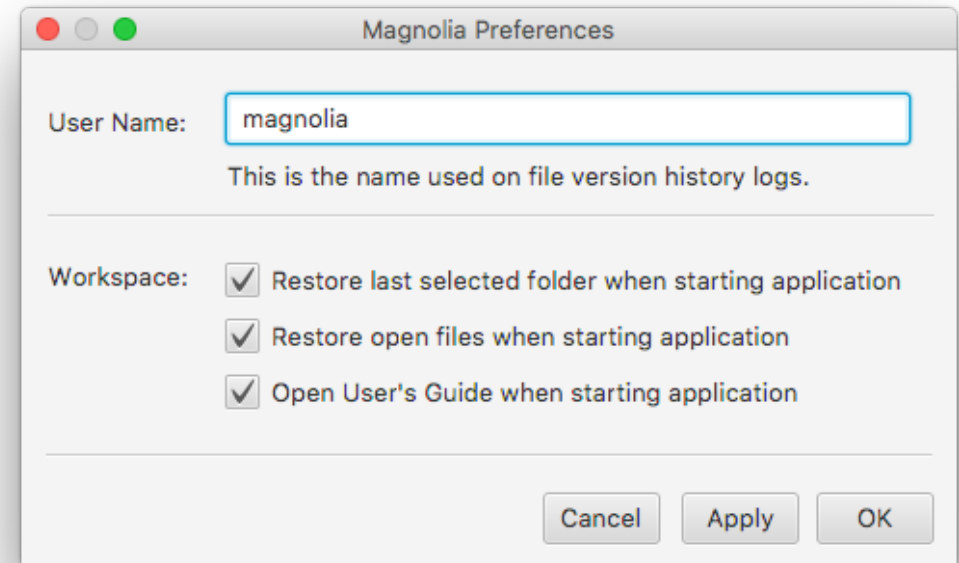
- **New:** opens a new file; use the sub-menus or drop-down toolbar selections to indicate what kind of file to create
- **Open:** opens a file browser to select an existing file to be opened
- **Recent Files:** selects a file to re-open from a list of recently edited files
- **Recent Folders:** navigates to a recently opened folder and displays its contents in the project/folder browser window (see the project/folder browser slide below for additional information)

Menu/Toolbar: “File” (continued)

- **Save:** save the contents of the currently selected file (this is the file currently visible in the document view in the middle area of the application)
- **Save As:** open a file dialog in order to save the currently selected file using a specific file name
- **Save All:** save all currently open files; if one or more of these files have not yet been saved, a file dialog will be displayed prompting for the name to give the file
- **Exit:** exit Magnolia; a prompt will be displayed if and open documents have unsaved changes

Menu/Toolbar: “File” (continued)

- **Preferences:** opens the dialog used to assign various application-wide settings
 - User Name: identifies the author of code edits tracked on the file version history maintained by the code editor (see the code editor slide below for additional details)
 - Workspace settings: specifies whether open files/folders should be restored the next time Magnolia is started, and whether the User’s Guide should be automatically opened when the application starts

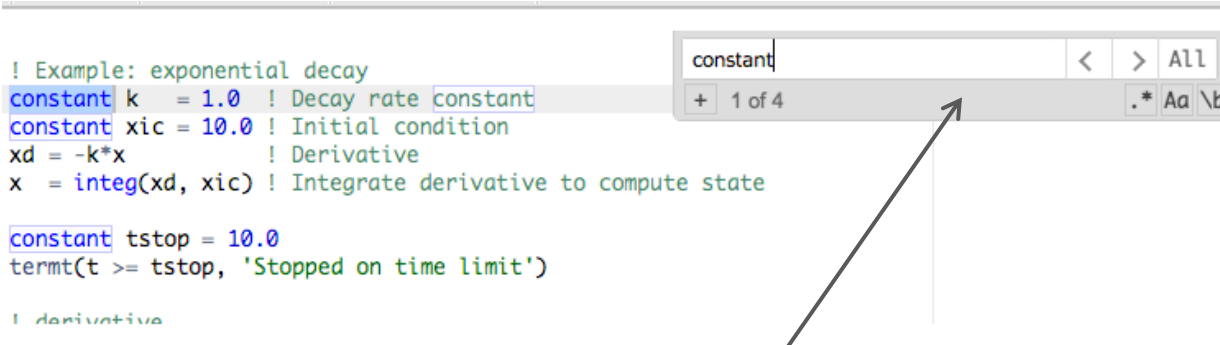


Menu/Toolbar: “Edit”

These menu items are active when a source code file is the active document (i.e., they don't have an effect on plot windows):

- **Undo/redo:** undo or redo the last source code edit on the active document (each open document maintains a separate undo/redo history)
- **Cut:** copy the selected text to the clipboard and delete the selection
- **Copy:** copy the selected text to the clipboard without deleting it
- **Paste:** insert the contents of the clipboard into the active document at the current cursor location
- **Delete:** delete the selected text in the active document

Menu/Toolbar: “Edit” (continued)



```
! Example: exponential decay
constant k = 1.0 ! Decay rate constant
constant xic = 10.0 ! Initial condition
xd = -k*x ! Derivative
x = integ(xd, xic) ! Integrate derivative to compute state

constant tstop = 10.0
term(t >= tstop, 'Stopped on time limit')

! derivative
```

- **Find:** search for the specified string in the active document (this comment opens a small field at the top right of the code editor)
- **Replace:** to find and replace text in the code editor, click the small “+” at the lower left of the search box and specify the original and replacement strings
- Note that this popup window can also be opened by typing Control-f on the keyboard (or Command-f on Mac OS)

Options are also available to use regular expression-based searches, case-sensitive searches, and whole-word searches



Menu/Toolbar: “View”

Several of the small “tool” windows docked to the edge of the main application window may be hidden/displayed as desired to optimize available screen area; check or un-check these as desired

See subsequent slides for detailed information related to each of these windows:

- Folders
- Command Window
- Command History

Menu/Toolbar: “Run”

- **Build:** translate and compile the model in the active document, but do not run it; this is useful while building and debugging model code
- **Start:** translate, compile and run the model in the active document. If the model builds and runs successfully, an interactive plot will be displayed at the conclusion of the run
- **Stop:** halts a simulation in progress. This is useful if a simulation is taking an unexpectedly long time to run, for example
- **Reset:** resets all model parameters back to the values specified in the CSL code; i.e., this menu item is used to revert any parameter changes which have been made via parameter sliders or via commands at the command prompt

Menu/Toolbar: “Windows”

- **Close All Plots:** closes any open plots windows, but leaves all other documents open
- **Close All Documents:** closes all open documents, including model and script files, plots and data files
- **Window List:** lists the names of all open documents (as reflected in the text in their associated tab at the top of the document area
 - Click an entry in this list to make the corresponding document active; this is useful when many documents are open at one time

Menu/Toolbar: “Help”

- **Search (Mac OS only):** can be used to quickly locate a menu item by entering a search string
- **User’s Guide:** opens the Magnolia online help system and makes it the active document (see subsequent slides for additional details)
- **About:** opens the Magnolia “About” dialog, which displays the running version of Magnolia

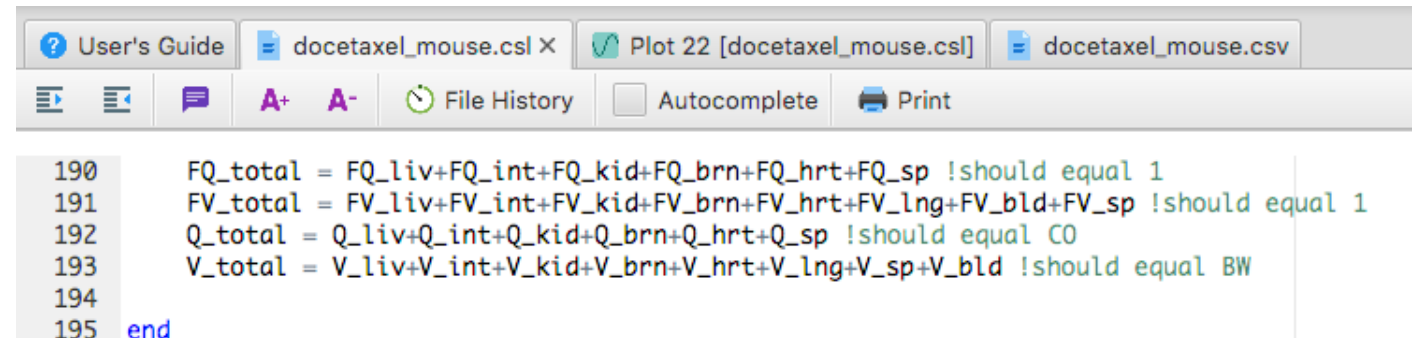
The text field in the center of the About window contains system diagnostic information which can be used to help troubleshoot installation and operation issues



Document Area/Tabs

In Magnolia, “document” refers to both files associated with projects (data files, model files, scripts), and plots. Each of these documents is displayed in a associated editor in the middle of the Magnolia main window. These editors are arranged on tabs; selecting one of these tabs makes the corresponding document active. Most buttons on the main toolbar act on the active document.

- Some editors have a “mini” toolbar just below the tab and above the editor itself which contains options specific to that type of editor; more information on those toolbars can be found in the subsequent section describing each type of editor
- A document can be closed by clicking the small “x” on its tab; you’ll be prompted to save the file if there are any unsaved edits to the document
- A list of the open documents is displayed on the main menu -> windows submenu; clicking one of these items will make the corresponding document active



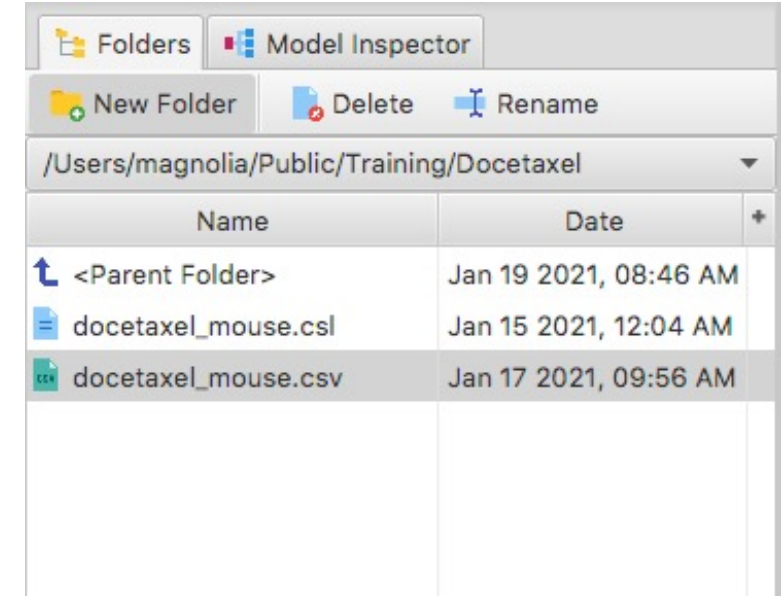
The screenshot shows the Magnolia software interface. At the top, there is a tab bar with four tabs: "User's Guide" (with a question mark icon), "docetaxel_mouse.csl" (with a document icon), "Plot 22 [docetaxel_mouse.csl]" (with a checkmark icon), and "docetaxel_mouse.csv" (with a document icon). Below the tabs is a toolbar with icons for document navigation, a mini toolbar with "A+" and "A-" buttons, a "File History" button with a clock icon, an "Autocomplete" checkbox, and a "Print" button with a printer icon. The main editor area displays a code snippet with line numbers 190 to 195. The code is as follows:

```
190 FQ_total = FQ_liv+FQ_int+FQ_kid+FQ_brn+FQ_hrt+FQ_sp !should equal 1
191 FV_total = FV_liv+FV_int+FV_kid+FV_brn+FV_hrt+FV_lng+FV_bld+FV_sp !should equal 1
192 Q_total = Q_liv+Q_int+Q_kid+Q_brn+Q_hrt+Q_sp !should equal C0
193 V_total = V_liv+V_int+V_kid+V_brn+V_hrt+V_lng+V_sp+V_bld !should equal BW
194
195 end
```


Project/Folder Window

You control the organization of project files (data, scripts, models) in Magnolia by simply using the filesystem of your computer. To browse folders and open files, Magnolia provides a “Folders” view.

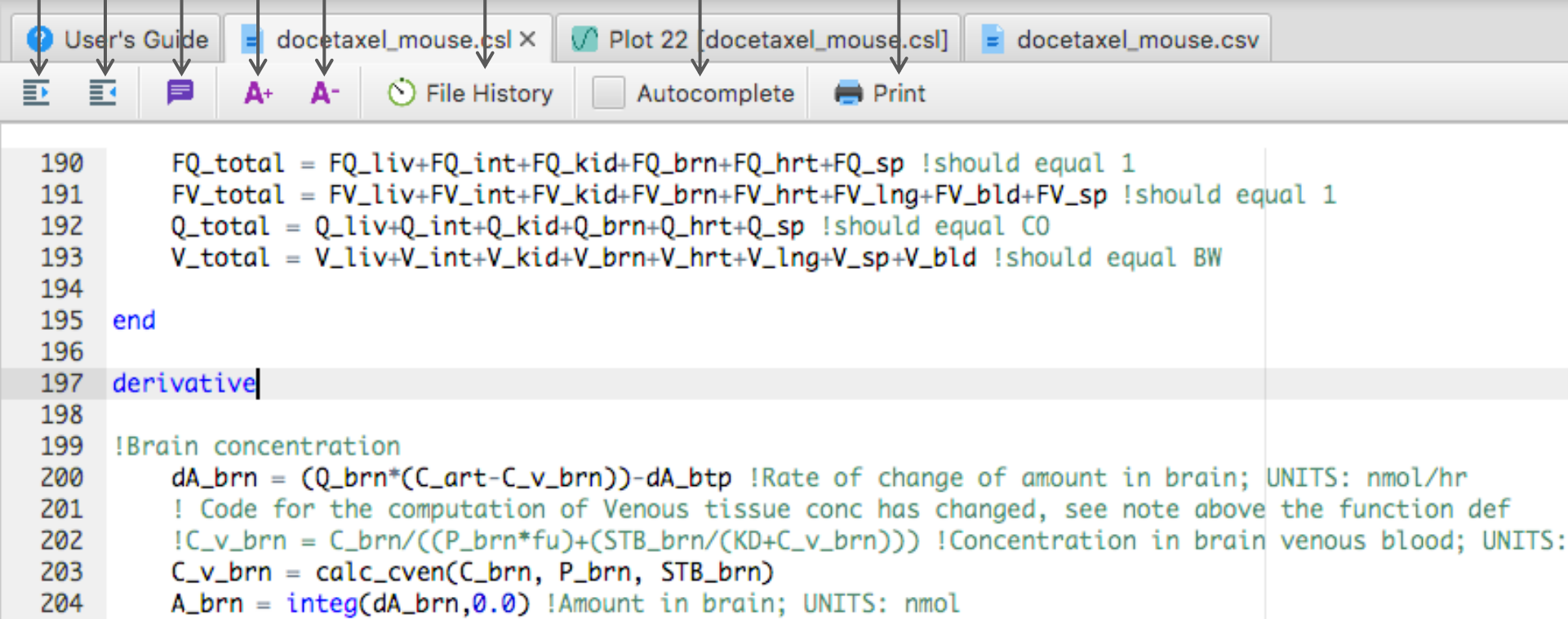
- Toolbar buttons are provided to create a new folder, delete the selected file/folder, or rename the selected file/folder
- A drop-down list displays the currently viewed folder, and can be used to quickly navigate to recently viewed folders
- Within the file/folder list, double-clicking a folder navigates down into that folder
- Double-clicking the <Parent Folder> item navigates up into the parent folder
- Double-clicking a recognized file type opens that file into the corresponding editor in the documents area



Code Editor and Toolbar

Indent/Unindent Show File History Toggle Autocomplete Print Document

Toggle Comment Increase/Decrease Font Size



The toolbar includes the following icons and labels:

- User's Guide
- docetaxel_mouse.csl
- Plot 22 [docetaxel_mouse.csl]
- docetaxel_mouse.csv
- Indent/Unindent
- Toggle Comment
- Increase/Decrease Font Size
- Show File History
- Autocomplete
- Print

```
190      FQ_total = FQ_liv+FQ_int+FQ_kid+FQ_brn+FQ_hrt+FQ_sp !should equal 1
191      FV_total = FV_liv+FV_int+FV_kid+FV_brn+FV_hrt+FV_lng+FV_bld+FV_sp !should equal 1
192      Q_total = Q_liv+Q_int+Q_kid+Q_brn+Q_hrt+Q_sp !should equal C0
193      V_total = V_liv+V_int+V_kid+V_brn+V_hrt+V_lng+V_sp+V_bld !should equal BW
194
195  end
196
197  derivative
198
199  !Brain concentration
200      dA_brn = (Q_brn*(C_art-C_v_brn))-dA_btp !Rate of change of amount in brain; UNITS: nmol/hr
201      ! Code for the computation of Venous tissue conc has changed, see note above the function def
202      !C_v_brn = C_brn/((P_brn*fu)+(STB_brn/(KD+C_v_brn))) !Concentration in brain venous blood; UNITS:
203      C_v_brn = calc_cven(C_brn, P_brn, STB_brn)
204      A_brn = integ(dA_brn,0.0) !Amount in brain; UNITS: nmol
```

Code Editor and Toolbar

The code editor window is used to create and edit all languages used by Magnolia, including model source code (CSL), scripting languages (CMD and Python) and the MCMC language used in Magnolia to define Bayesian models. The editor provides color syntax highlighting, code completion, search/replace and change history for all languages. A context menu with common edit operations (cut, copy, paste, etc...) is displayed via a right-mouse click. The mini toolbar above the editor area additionally provides buttons for the following:

- **Indent/Unindent:** indent or unindent the highlighted block of code by inserting/deleting a tab at the beginning of the line. This is particularly useful in Python scripts where indentation is used to specify code blocks. Indentation can also be performed by highlighting a block of code and pressing the <tab> key (indent), or <shift><tab> (unindent).
- **Toggle Comment:** applies or removes the language-specific comment character to all selected lines of code. This can be useful when debugging models/scripts to quickly enable or disable sections of code.
- **Increase/Decrease Font Size:** increases or decreases the font size used in the active editor window. This can be useful when running Magnolia on systems with large, high-resolution monitors.

Code Editor and Toolbar

- **Show File History:** in the lower section of the code editor, a list showing a summary of changes to the file is displayed; this list is updated each time the file is saved
- The combo box displays the time and date of a particular set of edits to the file, and the name of the user who made those edits
- In the list of changes, lines of code which were replaced or removed are highlighted in red; new lines of code are highlighted in green
- To revert the code to a particular version, select that version in the combo box and press the “Revert to this Version” button
- To clear the history of the file, select the “Clear History” button; this will permanently erase the list of changes to the file

The screenshot shows a code editor with a toolbar at the top containing buttons for 'User's Guide', 'docetaxel_mouse.csl X', 'File History', 'Autocomplete', and 'Print'. The code in the editor includes dosing parameters and timing commands. The 'History' panel at the bottom shows a list of changes, with the most recent change highlighted in green. The change is a modification to the 'CINT' variable, with the new value '0.1' highlighted in green and the old value '10' in red. The history entry is dated 'Fri, 29 Jan 2021 19:55:53 -0500 [magnolia]'.

```
40
41 !Dosing parameters
42 constant Dose = 3 !intravenous dose; UNITS: mg/kg; tail vein injection
43 constant MW = 861.9 !UNITS: g/mol
44 DDose = (Dose*1000000*BW)/MW !delivered dose; UNITS: nmol; scaled to bodyweight
45 constant fu = 0.07 !fraction docetaxel unbound in blood; 93% bound to plasma proteins
46
47 !Timing Commands
48 constant TSTOP = 12 !length of experiment; UNITS: hr
49 constant POINTS = 180 !number of points
50 !CINT = TSTOP/POINTS !interval of data collection (every 10 min); UNITS: hr
51 ! Changed to fixed CINTERVAL
52 cinterval CINT = 0.1 ! hours
53
54 !Organ blood flow parameters
55 CO = 0.275*(BW**0.75)*60 !cardiac output; UNITS: L/hr; from Brown (9249929) page 440
56 constant FO liv = 0.020 !flow to liver; UNITS: fraction CO: liver hepatic artery from
57
```

History: Fri, 29 Jan 2021 19:55:53 -0500 [magnolia] Revert to this Version Clear History

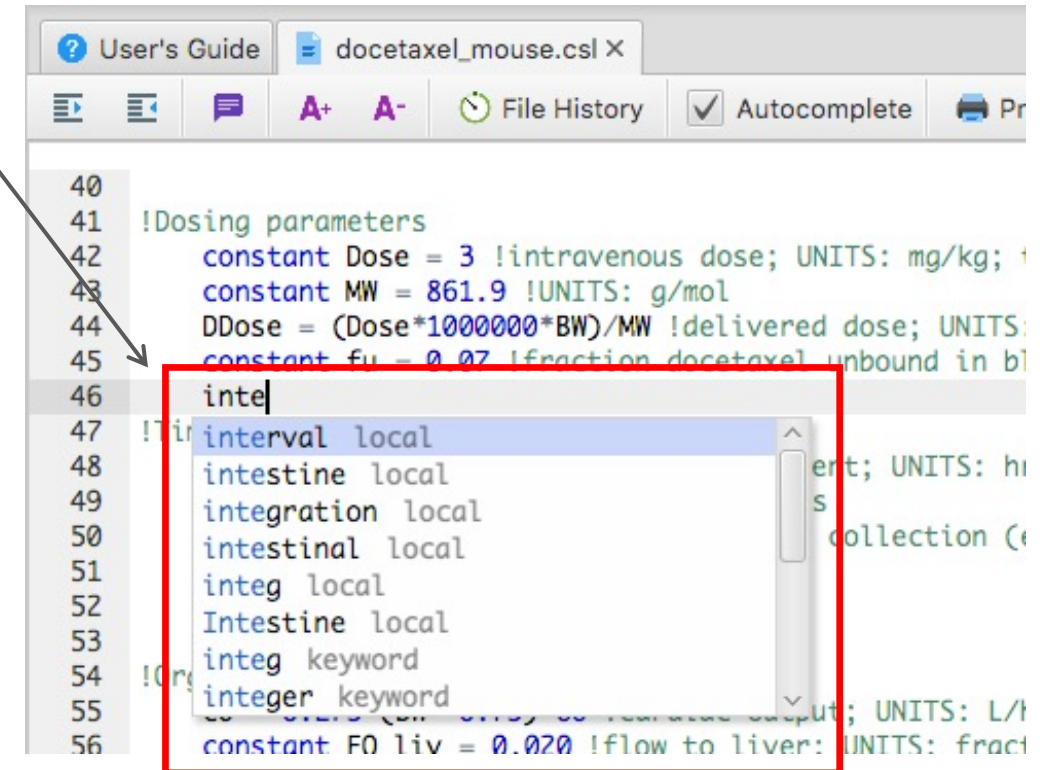
@@ -47,7 +47,9 @@

```
!Timing Commands
constant TSTOP = 12 !length of experiment; UNITS: hr
constant POINTS = 180 !number of points
- CINT = TSTOP/POINTS !interval of data collection (every 10 min); UNITS: hr
+ !CINT = TSTOP/POINTS !interval of data collection (every 10 min); UNITS: hr
+ ! Changed to fixed CINTERVAL
+ cinterval CINT = 0.1 ! hours
```

!Organ blood flow parameters

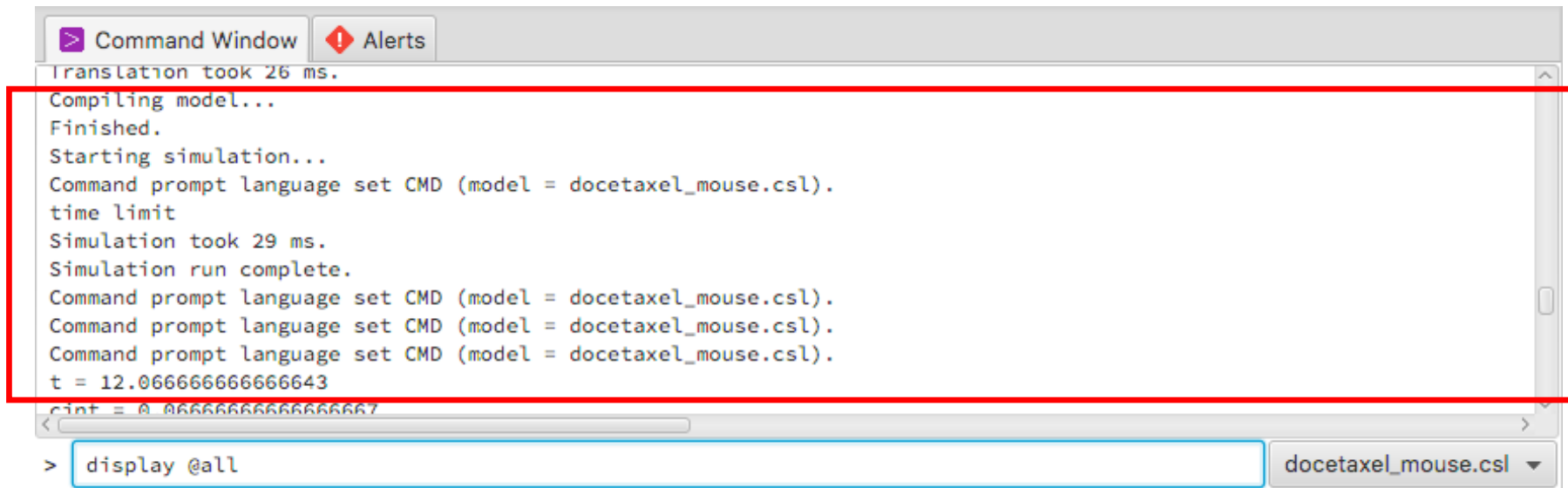
Code Editor and Toolbar

- **Toggle Autocomplete:** optionally enables “code completion; when enabled, this feature cases a pop-up list of keywords or variable names based on the last few letters typed in the editor
- To select one of the suggestions which match the partially-typed variable or symbol name, highlight the desired entry in the list using the arrow keys and hit the <tab> key
- If the list doesn't contains the keyword/variable/etc you intend to type, dismiss the list by hitting the <escape> key
- **Print Document:** open the operating-system-specific print dialog for printing the active source code document



Output Window and Command Line

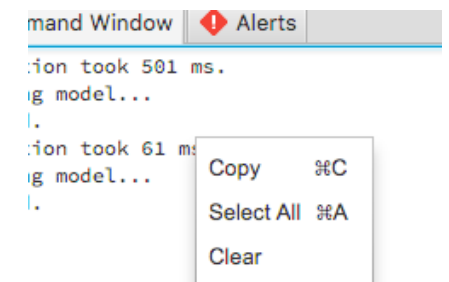
Textual output generated by Magnolia is displayed in the output window, a text box located at the bottom of the main window. This window displays information including: error/diagnostic information generated by Magnolia at model build- or run-time, runtime simulations outputs requested by the user (e.g., the values of specific variables at each time step as the simulation runs), or the results of commands interactively issued at the command prompt or in script files. A context menu is provided in this window to access a few useful actions. To open this menu, right-mouse click in the window; this menu includes options to select and copy text within the window, and to clear the existing contents of the window.



The screenshot shows the 'Command Window' in the Magnolia software. The window has a title bar with 'Command Window' and 'Alerts' buttons. The main area contains the following text:

```
Translation took 26 ms.  
Compiling model...  
Finished.  
Starting simulation...  
Command prompt language set CMD (model = docetaxel_mouse.csl).  
time limit  
Simulation took 29 ms.  
Simulation run complete.  
Command prompt language set CMD (model = docetaxel_mouse.csl).  
Command prompt language set CMD (model = docetaxel_mouse.csl).  
Command prompt language set CMD (model = docetaxel_mouse.csl).  
t = 12.066666666666643  
cint = 0.0666666666666667
```

A red rectangle highlights the main text area. At the bottom, there is a command prompt with the text '> display @all' and a dropdown menu showing 'docetaxel_mouse.csl'.



Output Window and Command Line

Magnolia supports multiple scripting languages, and allows multiple simulations to be open (in memory) at the same time. Because of this, it's necessary to tell Magnolia which scripting language should be used at the prompt when a command is issued, and which simulation is intended to be the target of the command. A drop-down button is provided just to the right of the command prompt to do this.

To issue a Python command, select "Python" from this list; Magnolia supplied command within the Python language to specify what loaded simulation is the target of a particular command. Refer to subsequent sections on details of Python scripting in Magnolia for more information regarding these commands.

To issue a command in the CMD language, select from this list the name of the source code file associated with the loaded simulation you wish to control. Any subsequent commands issued at the prompt will be sent to that particular simulation. Refer to subsequent sections on details of CMD scripting in Magnolia for more information regarding these commands.

```
set CMD (model = docetaxel_mouse.csl).
```

667

docetaxel_mouse.csl

✓ Python

docetaxel_mouse.csl

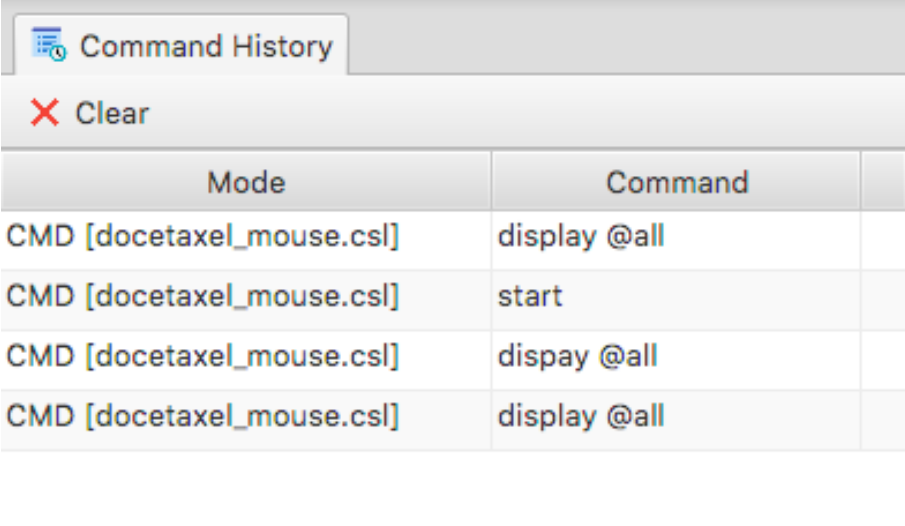
Python

Command History Window

Any commands issued interactively at the command prompt cause an entry to be added to the list in the command history windows. To re-execute a previous command, simply double-click the corresponding entry in the list.

Entries in this list contain two pieces of information: the text of the command itself (in the “Command” column) and for CMD commands, the simulation that was the target of the command (in the “Mode” column). If a CMD command is selected for re-execution and the simulation is no longer open in Magnolia, the command will have no effect.

This list contains only commands issues since Magnolia was last started. To clear out the contents of the list, select the “Clear” button on the toolbar.

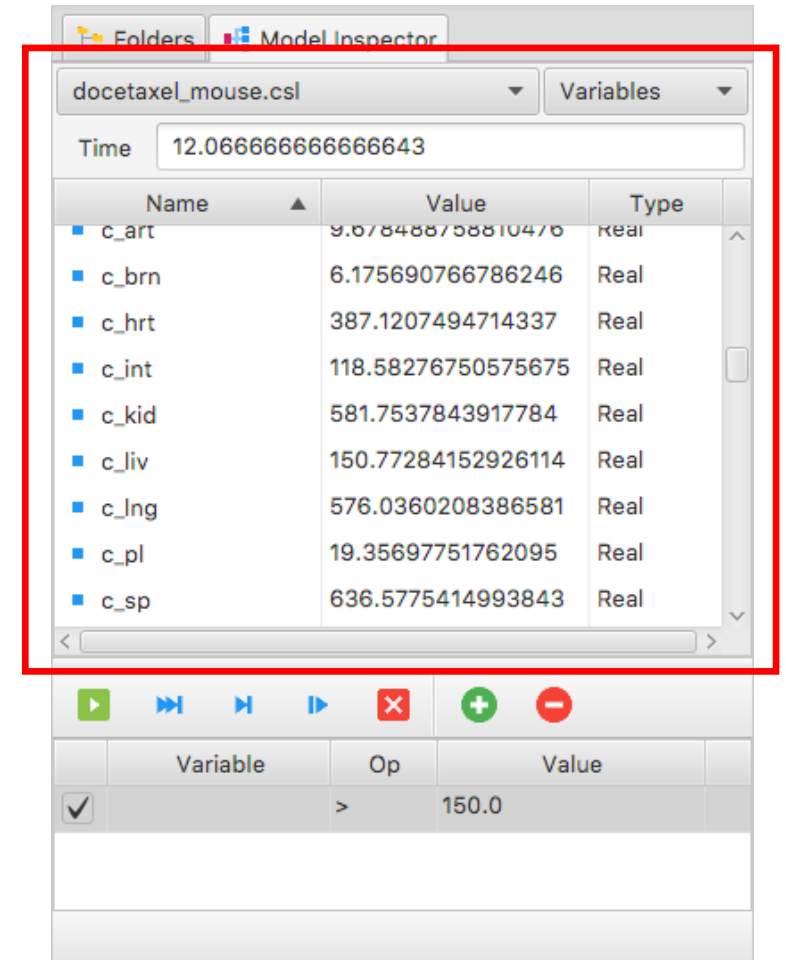


Command History	
✖ Clear	
Mode	Command
CMD [docetaxel_mouse.csl]	display @all
CMD [docetaxel_mouse.csl]	start
CMD [docetaxel_mouse.csl]	dispay @all
CMD [docetaxel_mouse.csl]	display @all

Model Inspector

When debugging complex models, it's frequently useful to have the ability to step through the simulation execution timestep-by-timestep in order to monitor the evolution of particular model outputs. To facilitate this, Magnolia provides the "Model Inspector." This window is used to display the values of model outputs or parameters as the simulation runs, pause and step simulation execution, and set a list of conditions on which the simulation execution should pause.

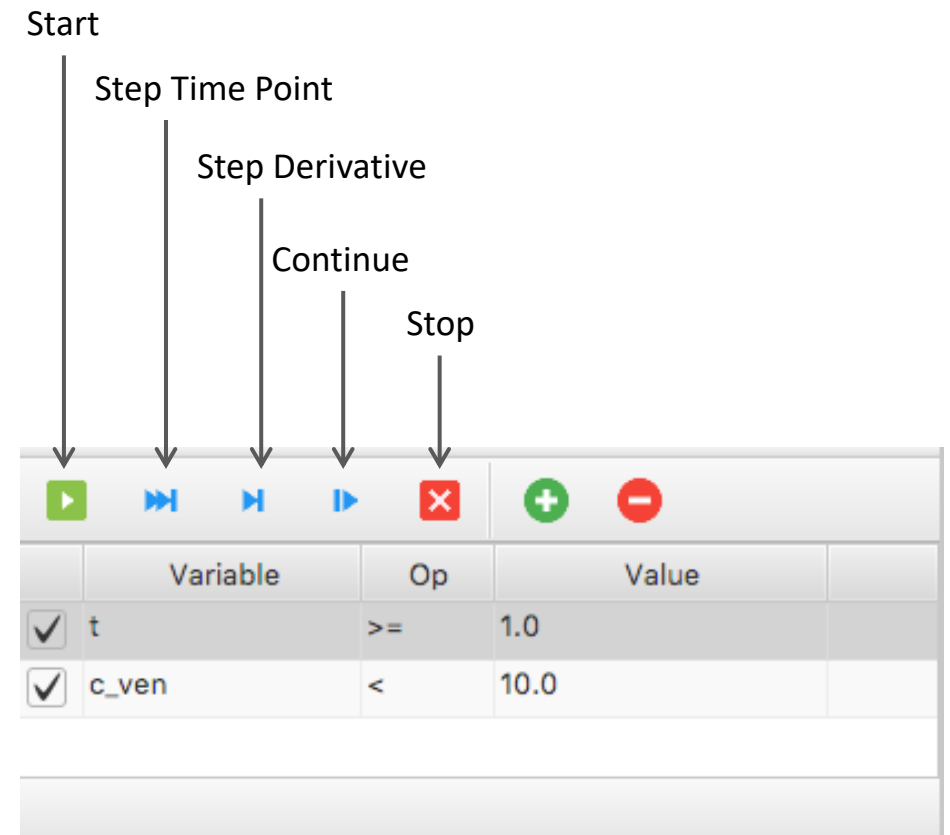
To use the Model Inspector, select the target model in the combo box near the top left of the window. Then select whether you want model variables or parameters to populate the list in the middle of the window. This list will be populated with the name of each variable or parameter in the model, its current value, and the datatype of that variable. The values in this list will be updated as the simulation run progresses, and you may switch between viewing parameters and variables at any time. To sort the list by a particular field, just click the corresponding column header for that field.



Model Inspector

Below the model variables/parameters table, the Model Inspector provides a “mini” toolbar with buttons for controlling how the simulation advanced in time:

- **Start:** start the simulation run and advance to the first derivative calculation
- **Step Time Point:** advance to the next simulated time points, as determined by the value of CINTERVAL
- **Step Derivative:** advance to the end of the next derivative calculations (note, there almost always multiple derivative calculations per time point, so this step provides a more granular level of detail than stepping by time point)
- **Continue:** continue time advancement and do not stop on subsequent time steps or derivative evaluations, but do stop on any breakpoints listed in the box below
- **Stop:** abort the debug run prior to conclusion



Model Inspector

Breakpoints in Magnolia consist of four pieces of information, as displayed in the four columns in the breakpoint table.

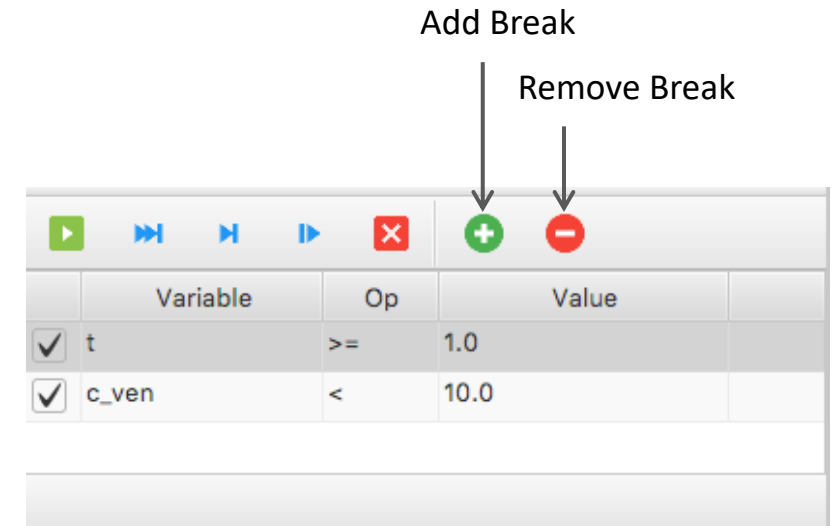
- **Check box (first column):** indicated whether the corresponding breakpoint is enabled. This is useful for turning off a breakpoint without completely removing it from the table.
- **Variable (second column):** indicates the model variable (output) on whose value the corresponding breakpoint is conditioned.
- **Op (operation, third column):** indicates the condition on which the stopping criteria is evaluated in comparison to the “Value” column: possible values are == (equality), ~= (inequality), > >=, <, and <=.
- **Value (fourth column):** specifies the value used in the comparison indicated by the value in the “Op” column.

Example: to pause the simulation run when time reaches 10, select the target model in the dropdown list at the top of the inspector window. In the breakpoint section, hit the green “+” button above the breakpoint list to add a breakpoint. Double-click in the “Variable” cell of the breakpoint you just added and type “t” to indicate that the stopping condition will be based on time (then hit <enter> to accept the value). In the “Op” column, select “>=“ to indicate that the simulation should pause as soon as time reaches (or exceeds) the value in the “Value” column. Finally, type the value “10” in the value column hit <enter> to accept the change. Press the “start debug” button on the mini toolbar, then press the “continue” to simulate forward to the breakpoint.

Model Inspector

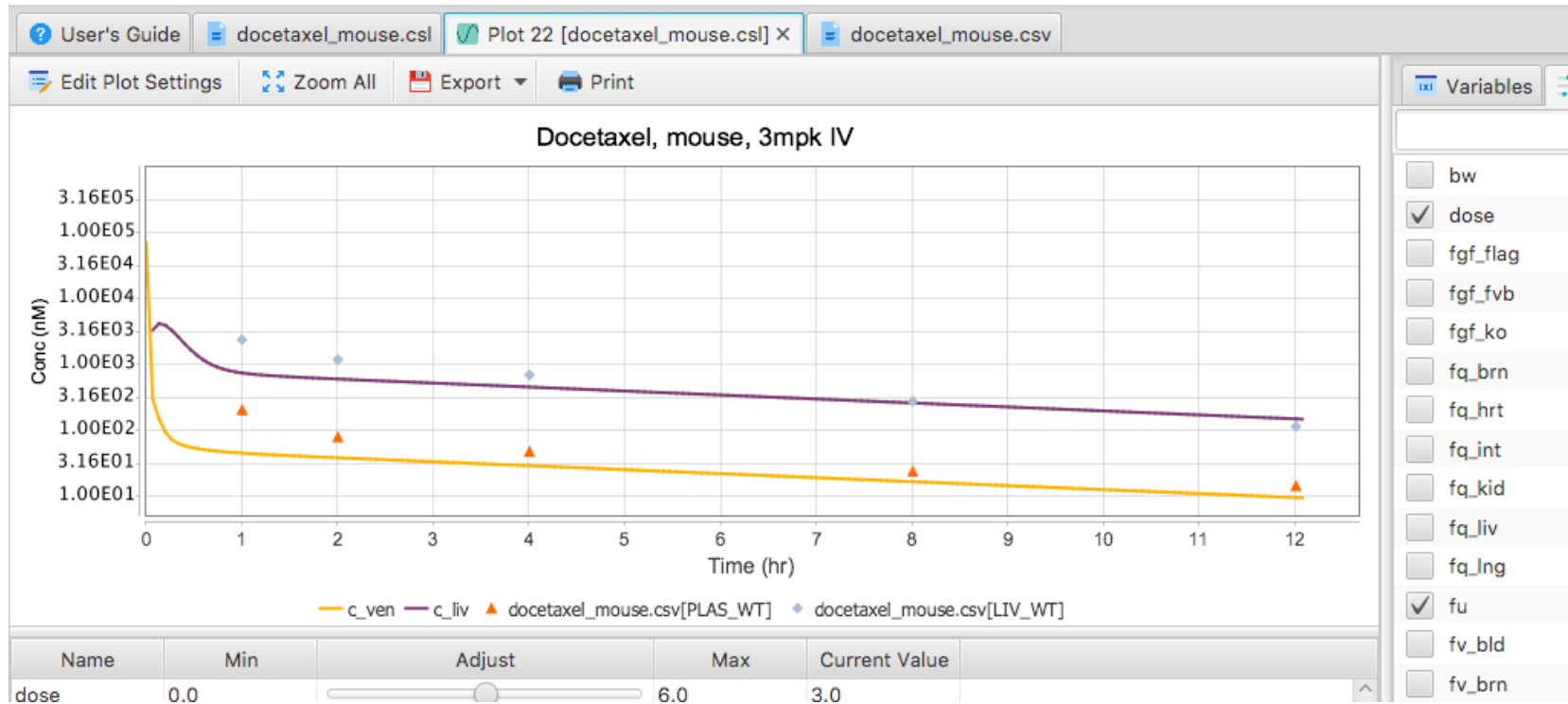
When trying to troubleshoot an event that's occurring at a point in simulated time long after the simulation starts, or when a model output crosses some threshold value, it's not practical to start the run and step by time point to the time at which the problem occurs. In order to tell the simulation to pause on a specific condition, Magnolia allows the construction of “break points,” which are displayed in the table near the bottom of the Model Inspector window:

- **Add Break:** add to the table a condition on which simulation execution should pause
- **Remove Break:** remove the selected condition from the table



Interactive Plot

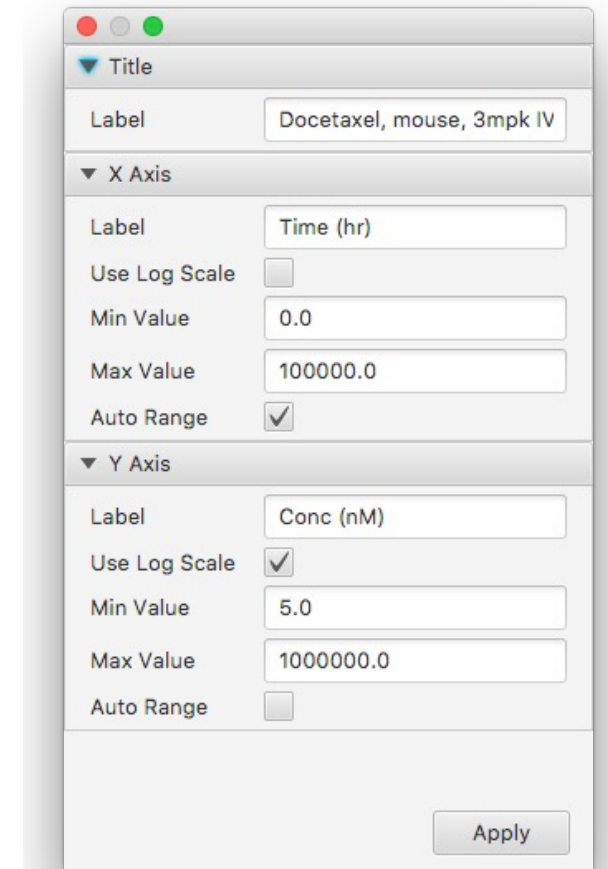
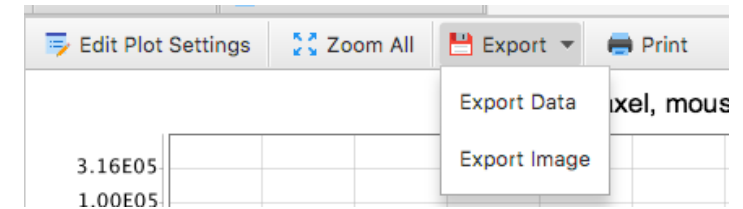
More outputs may be plotted in a variety of ways in Magnolia. To gain some intuitive understanding of the relationship between model outputs and input, it's useful to have a way to run the model interactively: that is, to make a change in one or model parameter values and have the simulation immediately run and update the plots or the outputs of interest. To support this, Magnolia provides *Interactive* plots. This allows you to quickly specify (i.e., without using scripts) which outputs are to be plotted, and which model parameters can be interactively controlled using sliders.



Interactive Plot Toolbar

The interactive plot windows has a mini toolbar just above the plot area. Buttons on this toolbar are used to do the following:

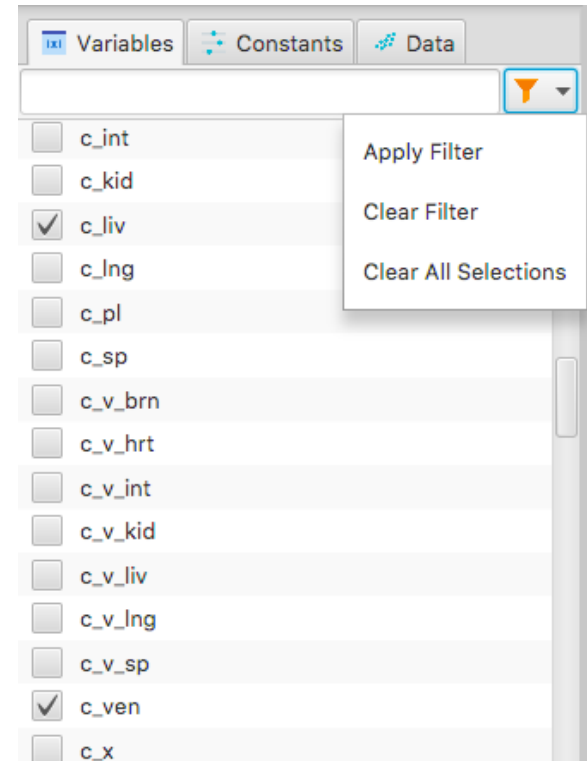
- **Edit Plot Settings:** displays a dialog from which properties of the plot can be adjusted:
 - **Title:** the text to be displayed above the plot area.
 - **X Axis:** the label, scale and range parameters associated with the X axis. The “Label” field can be used to enter a title for the X axis. Check the “Use Log Scale” check box to use a logarithmic scale on this axis. Check the “Auto Range” check box to automatically compute upper and lower bounds for the axis based on the range of the data to be plotted; otherwise, the values contained in the “Min Value” and “Max Value” fields will be used to determine the axis range.
 - **Y Axis:** the label, scale and range parameters associated with the Y axis.
- **Zoom All:** automatically recalculate the axis ranges to fit the displayed data series.
- **Export:** export the current plot at either an image (.png file) or as raw data (.csv file).
- **Print:** open the operating system-specific print dialog to print the current plot.



Model Variable/Parameters Lists

To select outputs which should be plotted, parameters which should be assigned to an interactive slider, and observed data which should be superimposed (as a scatterplot) over the model output plots, three checklist windows (arranged on tabs) are displayed to the right of the plot area.



- **Variables:** contains a list of all the model outputs which may be selected for plotting (presently, this includes on scalar variables). To add an output to the plot, simply check the corresponding entry in the list. If the particular output was not logged on the previous simulation run, a message will be displayed in the “Alerts” window indicating that the simulation needs to be re-run. Uncheck a output to remove it from the plot.
- **Constants:** contains a list of model parameters (as defined by the CSL “constant” keyword) that may be assigned to sliders in the table below the plot (more about this on the next slide). Checking a parameter adds a corresponding row to the sliders table; unchecking the parameter removed it. Only scalar model parameters are presently supported.
- **Data:** presents a list of open data files (CSV), Checking an entry in this list causes the corresponding data to be added to the plot as a scatter plot. See the “Data Table View” section for additional details.
- **Filter Button:** provides selection for filtering the entries in the list by specifying a matching wildcard pattern, and for clearing the filter or selected entries.



Parameter Sliders

Any selected parameters in the “constants” tab/list to the right of the plots area will have a corresponding row in the parameter “sliders” table. This table provides a way of adjusting parameter values, re-running the simulation, and updating the plots in an interactive way. The five columns in this table are used as follow:

- Name: cells in this column display the name of the corresponding model parameter, as selected in the list to the right of the plot window. The value in this cell can’t be changed.
- Min: the minimum value associated with the left end of the slider’s range. The value in this window can be changed by double-clicking the cell, typing the new value, and hitting <enter> to accept the new value. If the new value exceeds the current maximum, the max value will be increased accordingly.
- Max: the minimum value associated with the right end of the slider’s range. The value in this window can be changed by double-clicking the cell, typing the new value, and hitting <enter> to accept the new value. If the new value is below the current minimum, the min value will be decreased accordingly.
- Current value: displays the current value of the parameter, as determined by the slider’s position. To provide a new value precisely, the value can be typed into this cell; the slider will update accordingly.

c_ven c_liv docetaxel_mouse.csv[PLAS_WT] docetaxel_mouse.csv[LIV_WT]					
Name	Min	Adjust	Max	Current Value	
dose	0.0		6.0	3.0	
fu	0.0		0.14	0.07	

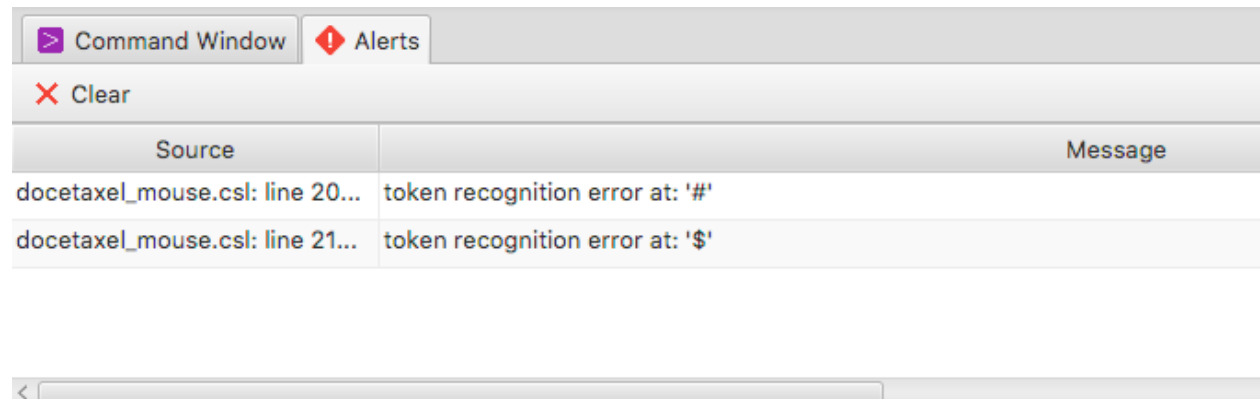
Alerts Window

The Alerts window displays a list of notifications of errors or warnings associated with building or running a model or script. Each row in the table displayed in the alerts window represents an individual warning/error message. The list is automatically cleared when a model/script is re-executed. Columns in this table are as follows:

Source: if the warning/error occurred upon build or execution of a model/script source code, the name of the file from which the error originated is displayed in this column. If this cell is not empty, double-clicking that row in the list will make the associated file active in the documents area, and position the cursor on the relevant line of code.

Message: text describing the associated with the error/warning. For some errors, additional detail is presented in the output window.

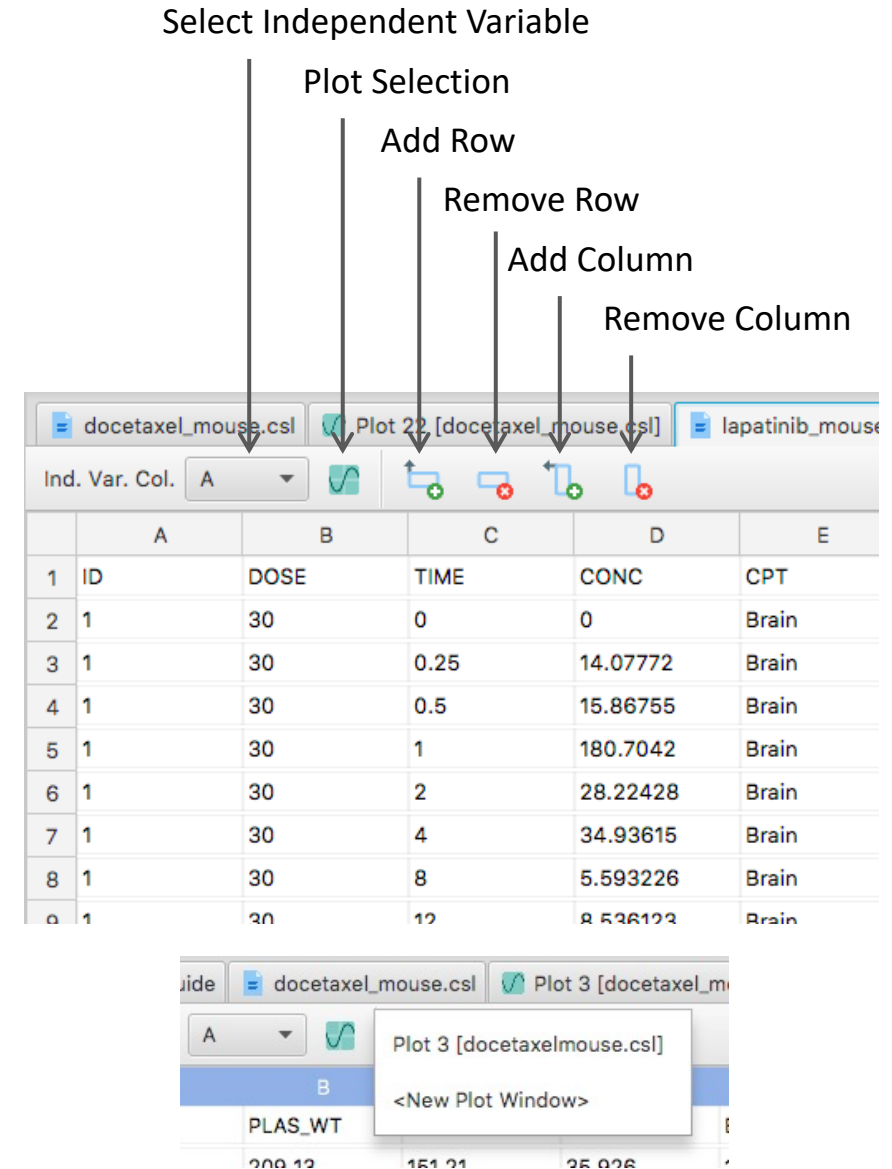
The “Clear” button in the mini toolbar just above the table may be used to remove all alert messages from the list.



Data Table View

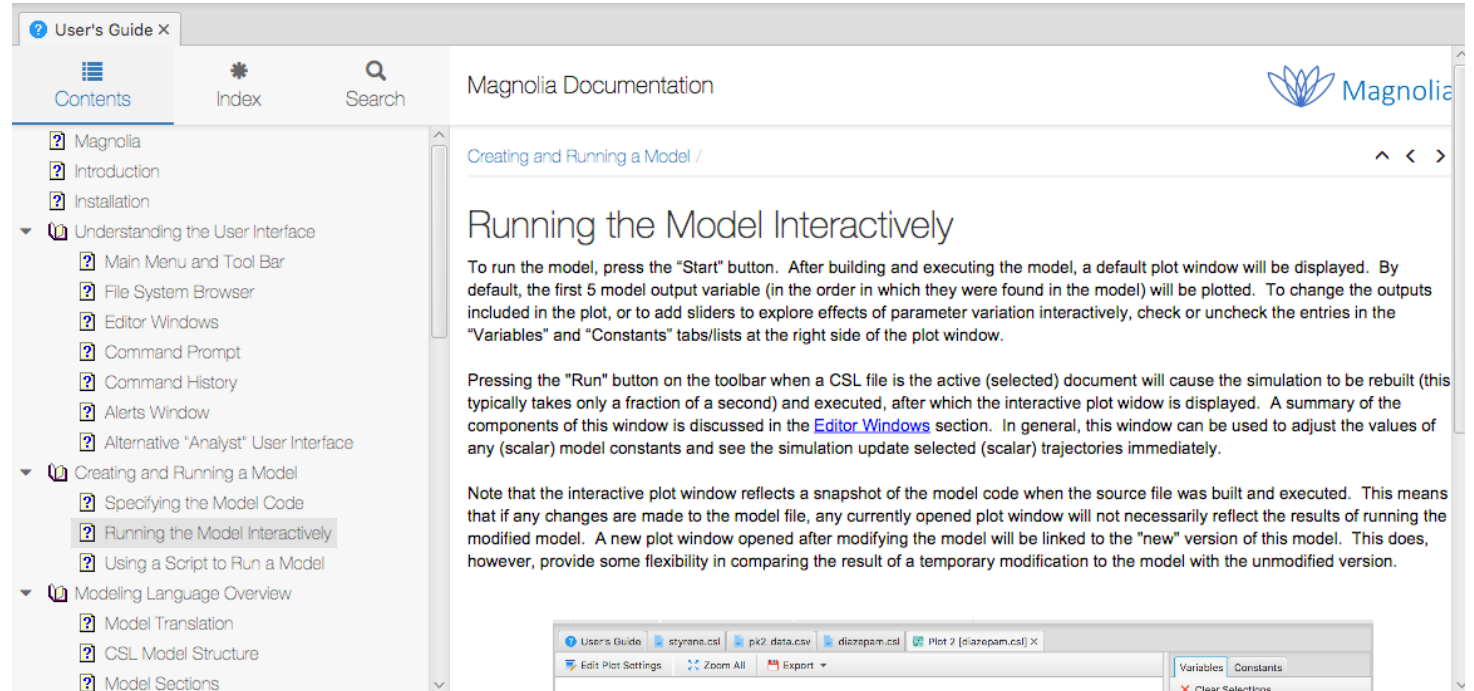
Comma-separated value (CSV) files may be opened and edited in Magnolia via the Data Table document view. Double-clicking a CSV file in the file browser window opens an editor in the document tab area of the main window. This editor contains a table displaying the file contents, along with a toolbar options plotting plotting the data and adding/removing columns/rows in the data set:

- **Select Independent Variable:** specifies which column in the spreadsheet should be used for the x-coordinate when the data is used to create a scatter plot. If multiple columns are selected, each column other than the independent variable column is rendered as a separate series
- **Plot Selection:** displays a list of open plots to which the selected data range may be added; optionally the data can be added to a new plot. If the current selection does not contain the specified independent variable column, an error message is displayed
- **Add Row:** adds a new row above the current selection
- **Remove Row:** removes currently selected rows
- **Add Column:** adds a new column to the right of the current selection
- **Remove Column:** remove currently selected columns



Interactive Help Window (User's Guide)

- Provides searchable HTML-based help on Magnolia user interface, basic workflow and modeling/scripting language references
- Use the tree view in the left panel to navigate a hierarchical list of help topics
- By default, the help window is opened when Magnolia starts
- If the help window is closed, it can be reopened from the main menu: Help -> User's Guide



File Types Recognized by Magnolia

- Model source code files: these are translated and compiled to generate executable models
 - **.csl, .csls**: CSL model and submodel files. These files contain the specification of systems of ODEs representing the behavior of the system being modeled.
 - **.mcmc**: statistical model descriptions used to estimate CSL/CSLS model parameters using Bayesian inference via Markov Chain Monte-Carlo sampling. Details of this language are described in separate training materials.
- Scripts: these files contain sets of commands to control simulation runs and analyses, specify any pre- or post-processing of results, and control data visualization.
 - **.cmd**: script files specified in the CMD interactive command language, appropriate for simple tasks
 - **.py**: Python language scripts which allow the flexibility of the general-purpose programming language for scripting complex analyses.
- Data files (**.csv**): the main format for importing tabular data into Magnolia for use in plots or parameter estimation
- Magnolia notebook files (**.mnb**): an experimental feature of Magnolia, intended to provide a tool for maintaining documentation to accompany models

Typical Workflow

The particular workflow used in a modeling/analysis project is, of course, problem-specific. However, a typical project workflow will likely consist of one or more of the following activities:

- 1. Data exploration:** this step usually involves plotting the data to be modeled and optionally performing some basic data analysis (e.g., summary statistics). Plotting can be accomplished by either opening the data from CSV files into a Magnolia data view and selecting ranges of data for plotting. Alternatively, plots can be created from Python scripts using commands detailed in a subsequent section. Use of the Python language along with a variety of Python libraries allows many kinds of exploratory data analyses to be performed on the data.
- 2. Model development/adaptation:** this step involves either creating a new model from scratch, or adapting an existing model for a new use. Generally, this consists of defining the equations representing the dynamic behavior to be modeled and encoding these equations in the CSL language using the Magnolia code editor.
- 3. Model debugging:** model development is generally incremental and iterative, meaning the model will frequently be executed and outputs checked (commonly visually) for reasonableness during development. Magnolia interactive plots are useful for doing this. When model predictions are inconsistent with expected results, however, the Magnolia model inspector can be a useful aid for carefully stepping the model execution incrementally in order to observe where specific model outputs start to go awry. Breakpoints also allow the model to be paused at specific points during the simulation run to investigate the source of model errors.

Typical Workflow

4. **Model qualification:** to assess confidence in model predictions, the model may be qualified (tested) in a variety of more- or less-rigorous ways to determine if the model is fit for intended use. At a minimum, the model outputs may be plotted against corresponding observed data (if available) to perform a visual predictive check. On the other hand, Python scripts can be constructed to quantitatively assess consistency of predictions with data (e.g., by computing residual sum of squares). If the model fails the qualification tests, further model refinement or debugging is required (steps 2 & 3).
5. **Sensitivity analysis:** most models contain parameters whose values are only known to a certain degree of accuracy. In order to understand the degree to which model outputs may be affected by variation in model inputs, Magnolia provides several tools for sensitivity analysis within the CMD scripting language. These commands, which are used to quantitatively assess sensitivity of one or more model outputs to variation in model inputs. Refer to the section on scripting languages for additional information on these features.
6. **Parameter estimation:** frequently it's necessary to estimate model parameters by fitting model outputs to observed data. This can be accomplished in Magnolia using specific commands in the CMD language (described in the scripting language section). Commands are provided for estimating model parameters using simple least-squares or maximum likelihood techniques, or by constructing statistical model for parameter estimation using Bayesian techniques.

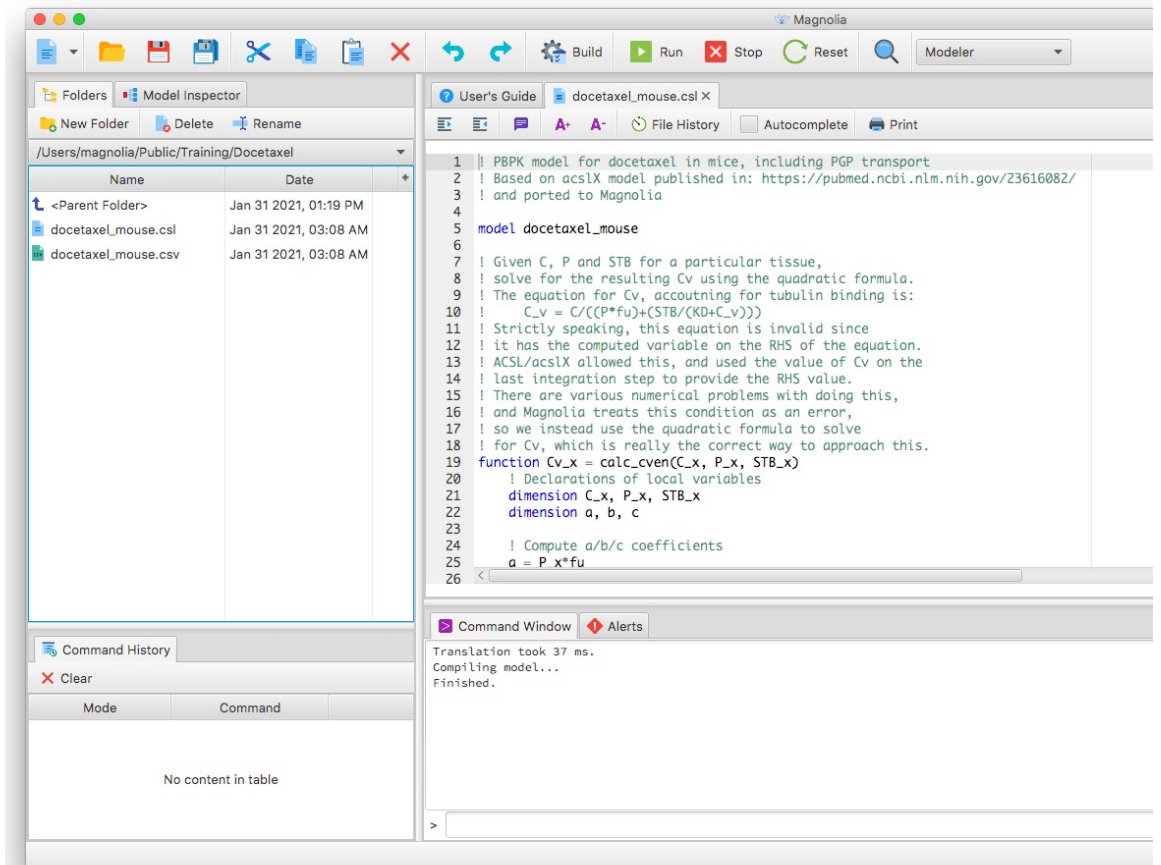
Typical Workflow

7. **Pre-processing of data:** in the event that data required for parameter estimation, or to directly specify the values of model parameters, need to be processed (e.g., unit conversions), any required processing can be specified using Python scripts. Python scripting also provides a way to open CSV files, read the contents in to Python arrays/matrices, and write that data back out to CSV files after manipulation.
8. **Simulation:** control of simulation runs in Magnolia is accomplished in a variety of way. Simulations may be run interactively by simply pressing the “Run” button on the main toolbar to open an interactive plot, and control simulation runs using parameter sliders. Alternatively, simulation runs can be controlled by scripting simulation scenarios in the CMD or Python languages. Each scenario typically consists of a set of model parameter values to set prior to starting the simulation run, followed by a list of commands to create the desired plots or printouts at the conclusion of the run.
9. **Variability analysis:** interactive plots in Magnolia only display “mean” behavior: that is, model outputs corresponding to mean/typical/putative values of model parameters. When some information about parameter variability is known, it’s usually useful to perform Monte-Carlo calculations to assess how variability in model parameters is propagated into variability in the model outputs. This can be accomplished in Magnolia through specific commands in the CMD and Python scripting languages. Using Python, for example, the model outputs collected over many model runs can be evaluated to compute confidence intervals, which can be plotted and overlaid with a scatterplot of the observed data.
10. **Post-processing of results:** any other necessary post-processing (e.g., statistical analysis) can be accomplished by creating appropriate Python scripts to manipulate simulation output directly at the end of the simulation run.

Exercise 2.1

Use the project/file explorer window to find and open a model file

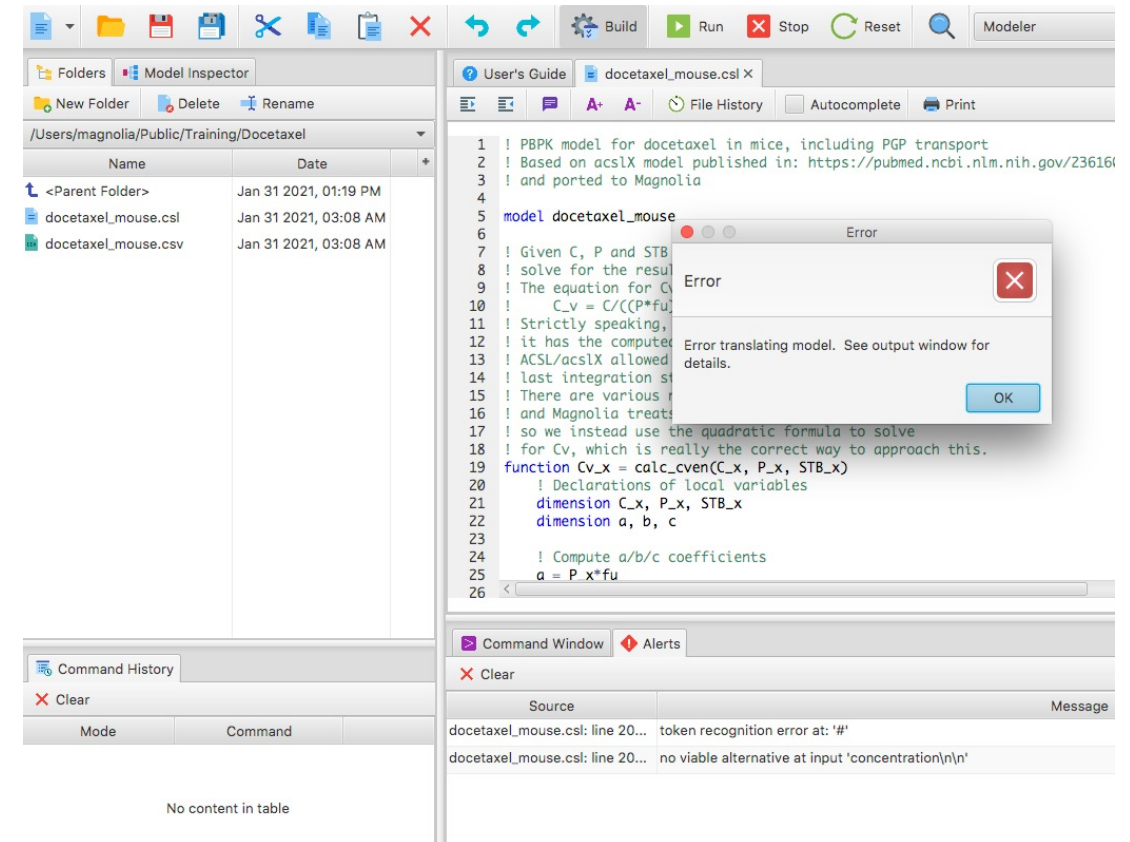
In the "Folders" window, navigate to the folder in which you extracted the training examples and find the "Docetaxel" subfolder. You should see a file named "docetaxel_mouse.csl" in this folder. Double-click that file in the folder list. The file should open and display in a code editor window. You should also see a small dialog indicating that the model is being built, but the dialog may be visible for only a fraction of a second, as the model builds very quickly. A few lines of text will be visible in the output window indicating the build was successful.



Exercise 2.2

Fix a model syntax error

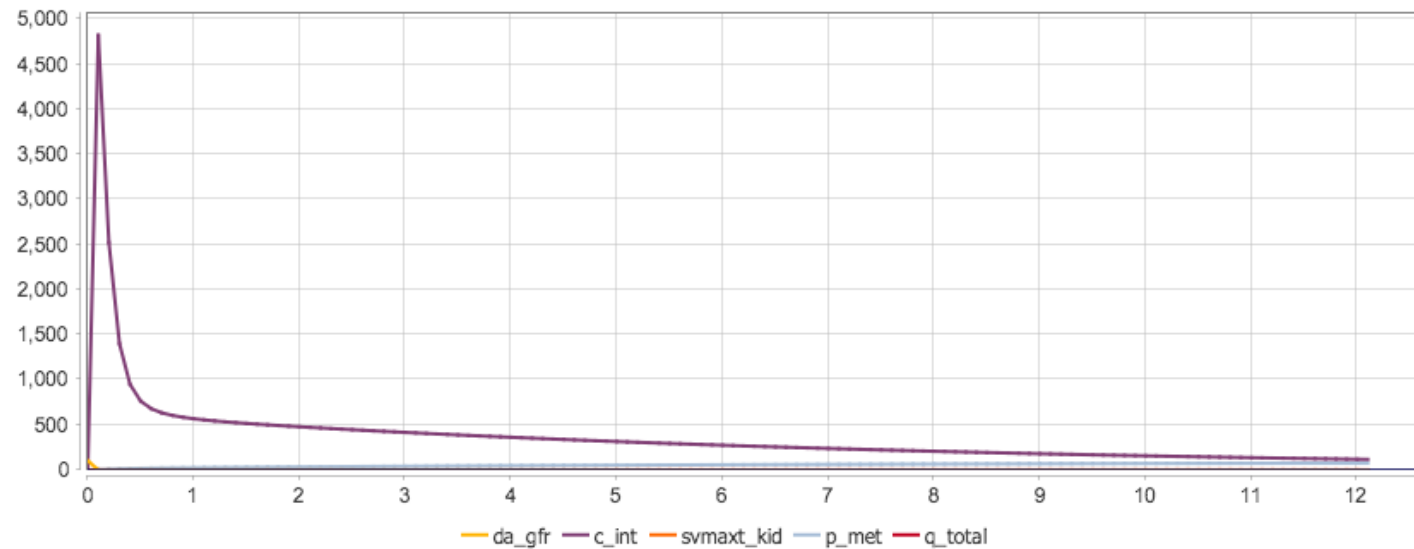
Let's intentionally add an error to the model we just opened. To do that, scroll to line 200 in the code editor (this should be just after the keyword "derivative" in the code. Just after "derivative," add a line containing a hash mark ("#") and immediately after that, another line containing only the word "concentration" (without quoted). Both of these are syntactical errors on the CSL language. Scroll back up to the top of the file, and hit the "Build" button on the main menu. You should see an error dialog near the middle of the screen, and two entries in the Alerts window notifying you of the two errors. Dismiss the error dialog by hitting "OK" and then double-click one of the alerts in the alert list. You should be taken to the lines in the code where the error was detected. Correct the error and rebuild the model to verify that the error has been corrected.



Exercise 2.3

Run the model and create an interactive plot

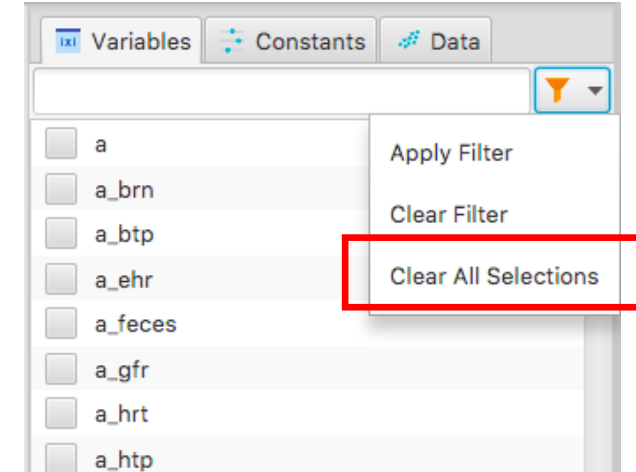
With the corrected model now open, hit the “Run” button on the main toolbar. The model should quickly build again and the interactive plot window should open in the document tabs area of the main window. If this is the first time you’ve run the model, the plot will contain a few default outputs, the sliders area below the plot area will be empty, and the plot will have no title or axis labels.



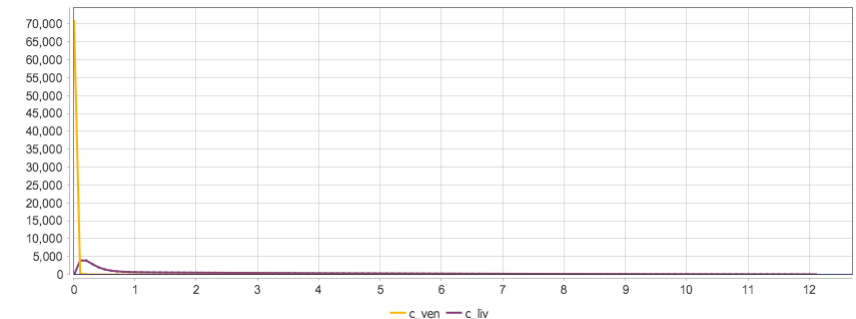
Exercise 2.3 (continued)

Now, let's select the model outputs we want to plot. To do this, first remove the currently plotted variables by pressing the filter button on the "Variables" tab to the right of the plot area and press "clear all selections." All series should disappear from the plot.

On the same list we can now select the outputs we want included in the plot. Let's select venous concentration (c_ven) and liver concentration (c_liv). You won't immediately see plots displayed because those outputs were not logged the last time the simulation ran (you'll also see two alerts in the alerts window notifying you of this). Re-run the model using the "Run" button on the toolbar to collect these outputs and update the plot.



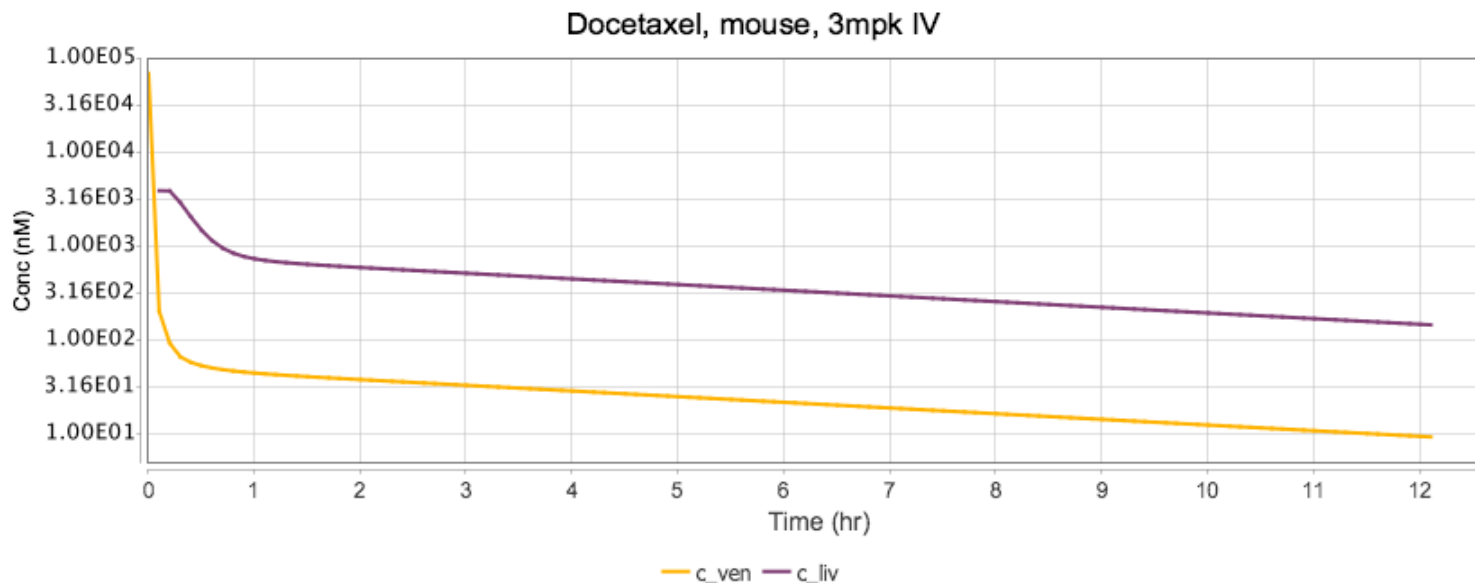
Command Window Alerts	
Clear	
Source	Message
Plot 6 [docetaxel_mouse.csl]	The output [c_ven] was not recorded on the last run. Please re-run the simulation to generate a plot for this output.
Plot 6 [docetaxel_mouse.csl]	The output [c_liv] was not recorded on the last run. Please re-run the simulation to generate a plot for this output.



Exercise 2.3 (continued)

This default plot could be improved by using a logarithmic y axis, adjusting the axis range, and adding some annotation. To do this, click the “Edit Plot Settings” button on the mini toolbar just above the plot.

You should see a dialog from which various plot properties can be adjusted. In that dialog, enter values as shown in the picture to the right and hit the “Apply” button. Dismiss the dialog by hitting the close button on the dialog title bar. The updated plot should appear as shown below.



▼ Title

Label Docetaxel, mouse, 3mpk IV

▼ X Axis

Label Time (hr)

Use Log Scale ☐

Min Value 0.0

Max Value 100000.0

Auto Range ☒

▼ Y Axis

Label Conc (nM)

Use Log Scale ☒

Min Value 5

Max Value 100000.0

Auto Range ☐

Apply

Exercise 2.3 (continued)

The plot settings you just specified (selected outputs, titles, axis scale, range, labels) are automatically saved when you close the interactive plot window and re-applied the next time the interactive plot is opened (i.e., the next time you run the model).

To illustrate this, close the plot window you just created, then make sure the “docetaxel_mouse.csl” model is the selected tab in the document area. Re-run the model by hitting the “Run” button on the main toolbar.

The plot should re-open and appear just as it did when you closed it a few moments ago.

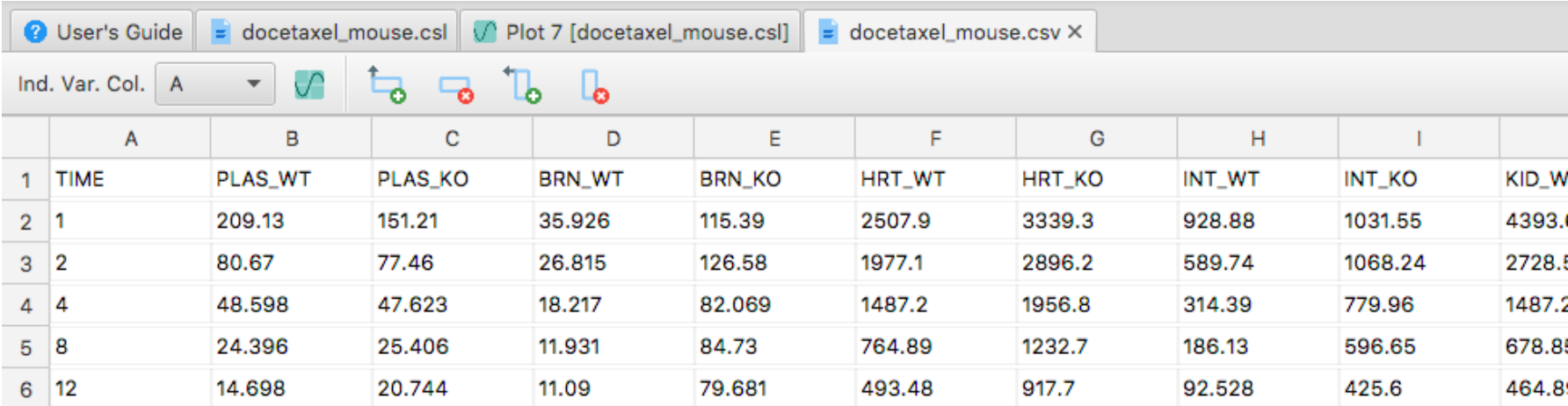
You’ll see shortly that model parameters assigned to sliders in the sliders table are also saved save with the plot settings when an interactive plot window is closed, so you won’t need to recreate the sliders (and associated ranges and current values) when re-creating the plot.

Exercise 2.4

Overlay observations on model predictions

When modeling phenomena for which observed data exists, or when estimating model parameters from observed data, we'll need at some point to overlay the observed data on the plots of model predictions. There are a number of ways to do this using the scripting language supplied in Magnolia, but let's look at a way to do this using on the user interface.

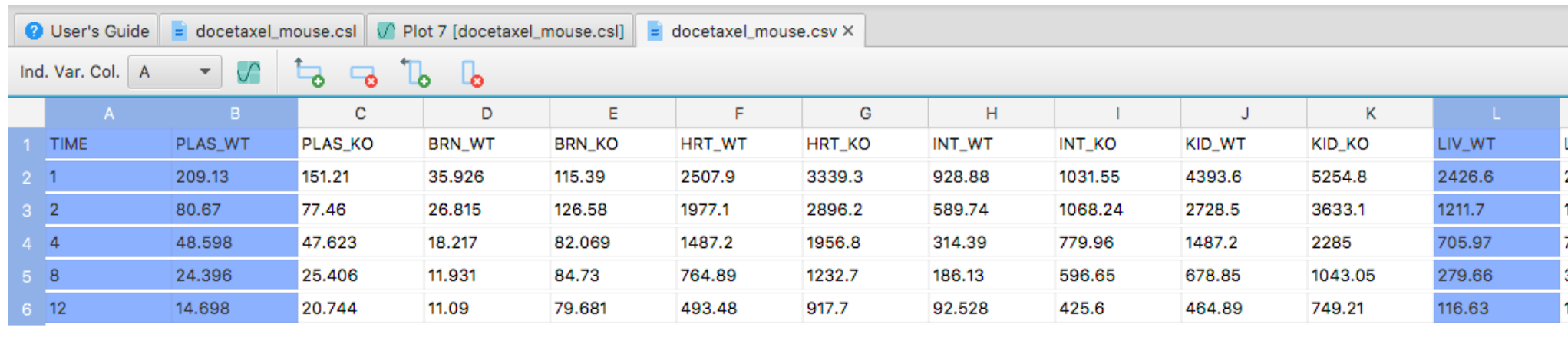
If your "Folder" window still has the docetaxel example model directory selected, you'll see a file named "docetaxel_mouse.csv." Double-click this file to open it in a Magnolia data table editor. You should then see something like the below picture in the document area of the main window.



	A	B	C	D	E	F	G	H	I	
1	TIME	PLAS_WT	PLAS_KO	BRN_WT	BRN_KO	HRT_WT	HRT_KO	INT_WT	INT_KO	KID_W
2	1	209.13	151.21	35.926	115.39	2507.9	3339.3	928.88	1031.55	4393.0
3	2	80.67	77.46	26.815	126.58	1977.1	2896.2	589.74	1068.24	2728.5
4	4	48.598	47.623	18.217	82.069	1487.2	1956.8	314.39	779.96	1487.2
5	8	24.396	25.406	11.931	84.73	764.89	1232.7	186.13	596.65	678.85
6	12	14.698	20.744	11.09	79.681	493.48	917.7	92.528	425.6	464.85

Exercise 2.4 (continued)


Make sure your interactive plot window plotting the venous and liver concentration profiles is still open. If it's not, simply re-run the model. We'd like to plot the observed venous and liver concentration samples from wildtype mice over the predicted curves. To do this, return to the data view documents and highlight the TIME, PLAS_WT and LIV_WT. This can be done by selecting the header (the "A", "B", "C" etc. buttons) above each desired column of data values. You'll need to select three columns; hold down the control/command (depending on your operating system) button when selecting the second and third columns to do this. Note that if instead of selecting the entire column, you only wanted to select a specific range of cells, that can be done by clicking and dragging to select the desired range using the mouse (as is commonly done in most spreadsheet programs). The data view should now look similar to the following image:



	A	B	C	D	E	F	G	H	I	J	K	L
1	TIME	PLAS_WT	PLAS_KO	BRN_WT	BRN_KO	HRT_WT	HRT_KO	INT_WT	INT_KO	KID_WT	KID_KO	LIV_WT
2	1	209.13	151.21	35.926	115.39	2507.9	3339.3	928.88	1031.55	4393.6	5254.8	2426.6
3	2	80.67	77.46	26.815	126.58	1977.1	2896.2	589.74	1068.24	2728.5	3633.1	1211.7
4	4	48.598	47.623	18.217	82.069	1487.2	1956.8	314.39	779.96	1487.2	2285	705.97
5	8	24.396	25.406	11.931	84.73	764.89	1232.7	186.13	596.65	678.85	1043.05	279.66
6	12	14.698	20.744	11.09	79.681	493.48	917.7	92.528	425.6	464.89	749.21	116.63

Exercise 2.4 (continued)

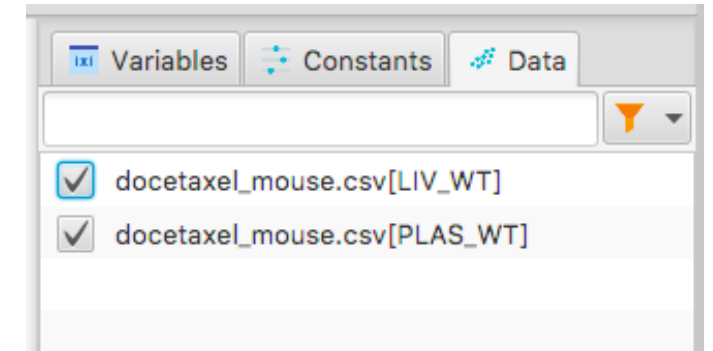
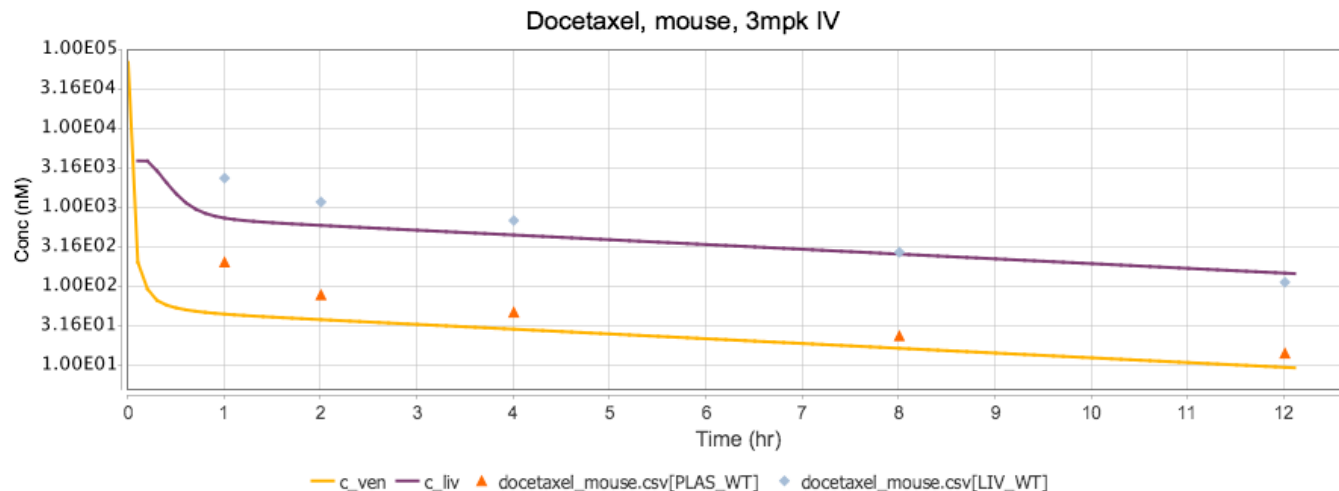
To send the selected data to the plot window and have it rendered as a scatter plot superimposed on the predicted curves, ensure that the “Independent Variable Column” combo box in the toolbar still has column “A” selected, then hit the small plot icon/button to display a list of target plots to which the data can be sent. We want to send the data to the interactive plot, which should still be visible, select the plot window containing the name of the model. The plot window should then become the active document again, and the observed data should be included on the plot as a scatter plot (i.e., symbols not connected by lines). You’ll also see two new entries in the list on the “Data” tab to the right of the plot area. These can be used to show or hide observed data sets.

Ind. Var. Col. A 

Plot 7 [docetaxelmouse.csv]

<New Plot Window>

	A	B		
1	TIME	PLAS_WT		
2	1	209.13	151.21	35.926
3	2	80.67	77.46	26.815

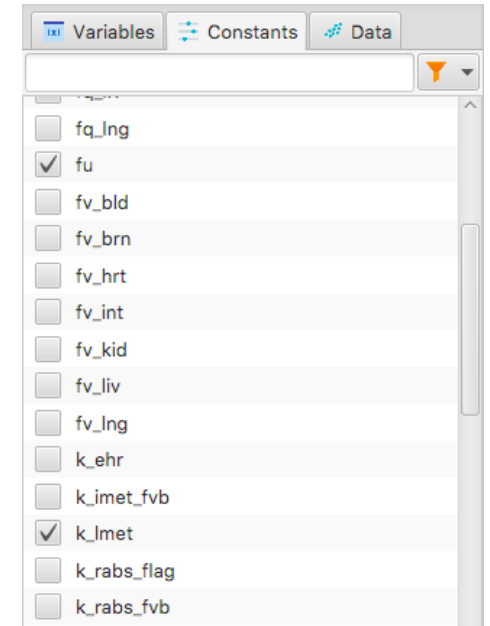


Exercise 2.5

Add sliders and adjust model parameters interactively

Let's add a few sliders so we can interactively adjust some model parameters and see how the predicted curves change in real time. Specifically, let's assign sliders to the protein binding fraction (f_u), liver partition coefficient (p_{liv}) and hepatic metabolism rate (k_{lmet}). Simply check the corresponding entries in the "Constants" tab/list to the right of the plot to do this. The sliders table below the plot should now appear similar to the image below.

Name	Min	Adjust	Max	Current Value	
k_{lmet}	0.0		7328.0	3664.0	
p_{liv}	0.0		14176.0	7088.0	
f_u	0.0		0.14	0.07	

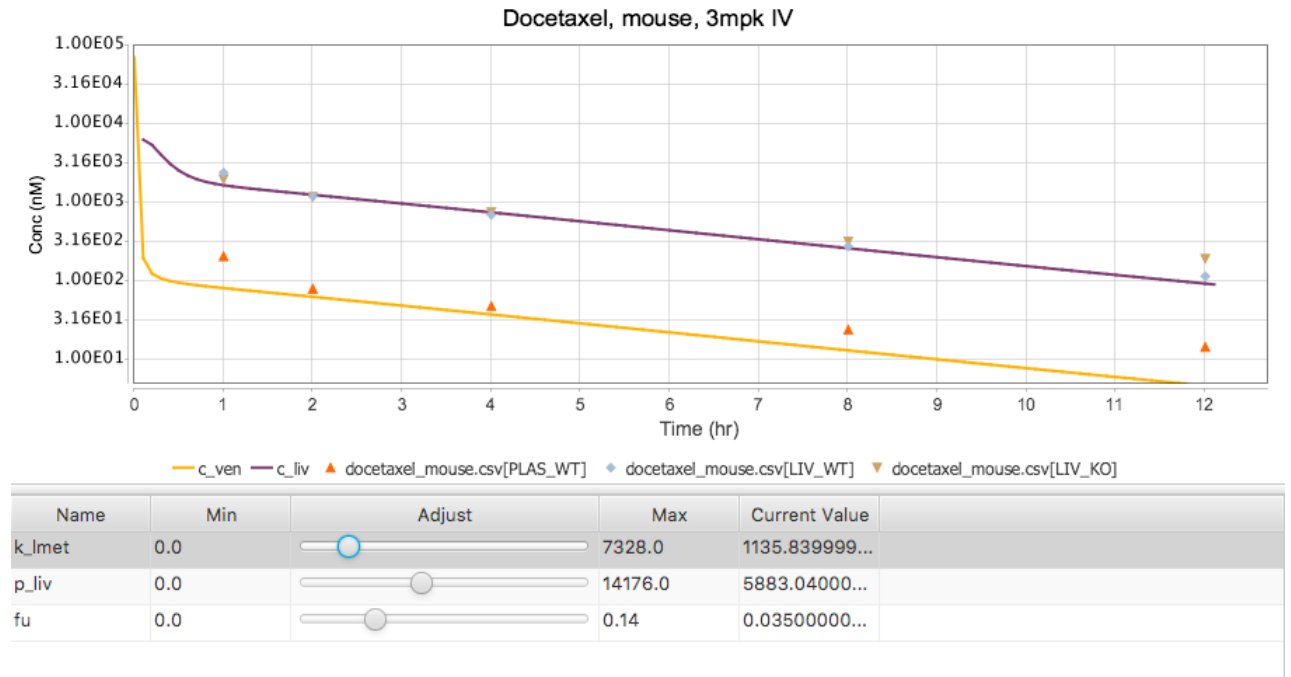


Exercise 2.5

If you'd like to adjust any of the slider ranges, simply double-click the corresponding cell under the "Min" of "Max" column, enter the desired value, then click <enter> to commit the change. For now, let's just leave the default value. Now try adjusting any/all of the sliders by dragging the small circle on the slider with the mouse. You should see the plotted curves update as soon as you release the mouse. Try adjusting the parameters as shown in the picture below, either by dragging the slider button or by entering a numeric value explicitly in the "Current Value" column. Verify that your plots looks similar to the one in the image.

Note that if you close this plot, then re-run the model, the sliders and associated settings will be re-created and the corresponding parameter values set in the model before it runs (data sets are not automatically reloaded and plotted, however).

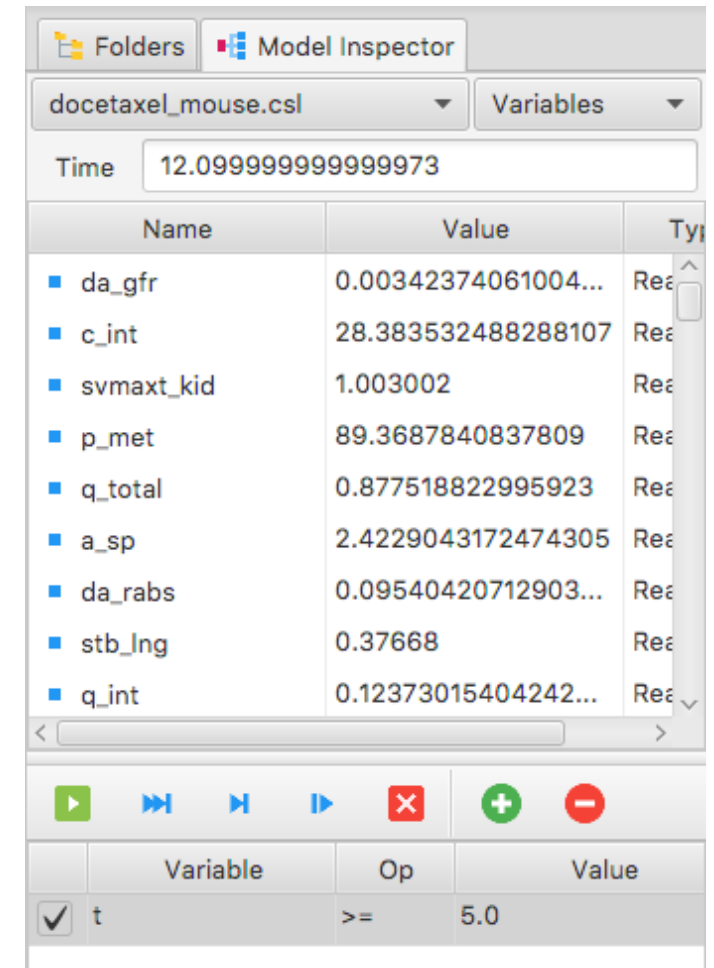
If you'd like to reset the sliders to their default values as specified in the model code (and re-run the simulation), click the "Reset" button on the main toolbar with the interactive plot window open and selected in the document area.



Exercise 2.6

Use the model inspector to monitor model variables


Now let's run the model using the "Model Inspector" to illustrate how one might go about debugging a model. Let's say we're seeing the venous concentration grow unreasonably large shortly after time = 5 in the models. We can investigate the model quantities which might contribute to the error by creating a couple of breakpoint in the model inspector. First, create a breakpoint to pause model execution at $T \geq 5$. In the model inspector window, make sure the selected model is "docetaxel_mouse.csl." Push the circular "+" button just above the breakpoints window to add a breakpoints. Double-click the cell in the "Variable" column and type "t" to specify that the breakpoint will be predicated on time. Then double-click the cell in the "Op" column and select " \geq " from the dropdown list to indicate that we want break when time exceeds a specified value. Finally in the "Value" column, enter the value 5. Your model inspector window should now look something like the image to the right.



Exercise 2.6 (continued)

Repeat the preceding steps to add a breakpoint to stop when `c_ven` exceeds $1e8$ nM. The updated breakpoints table should look something like the image at the bottom of this slide.

Now start a debug run by hitting the green “play” button in the mini toolbar just above the breakpoints table. The simulation will start and quickly pause at the first communication interval (i.e., just before the first set of model outputs is logged). The time will still equation zero at this point, and you can see initial values of model variables displayed in the table just above the breakpoints list. You’ll also see a message at the bottom of the window indicating where/why the breakpoint occurred.




	Variable	Op	Value
<input checked="" type="checkbox"/>	t	>=	5.0
<input checked="" type="checkbox"/>	c_ven	>=	1.0E8

docetaxel_mouse.csl Variables

Time 0.0

Name	Value	Type
da_gfr	0.00342374061004...	Real
c_int	28.383532488288107	Real
svmaxt_kid	1.003002	Real
p_met	89.3687840837809	Real
q_total	0.877518822995923	Real
a_sp	2.4229043172474305	Real
da_rabs	0.09540420712903...	Real
stb_lng	0.37668	Real
q_int	0.12373015404242...	Real



	Variable	Op	Value
<input checked="" type="checkbox"/>	t	>=	5.0
<input checked="" type="checkbox"/>	c_ven	>=	1.0E8

Stopped on communication interval.

Exercise 2.6 (continued)

Even though the problem we're attempting to debug doesn't occur until about $T = 5$, let's step forward in the simulation a few times to observe how the variables table updates. Hit the "step time" button a few times and notice that the value of time in the field at the top of the window updates in increments of 0.1. Now press the "step derivative" button a few times and notice how time updates. The displayed time values are no longer at increments of 0.1, since the derivative calculations may happen at timepoints determined by the ODE solver algorithm.

Step to next time (communication interval)

Step to next derivative calculation

Continue (step to next breakpoint, or end of simulation)

The screenshot shows the 'docetaxel_mouse.csl' simulation window. At the top, the 'Time' field displays '0.2012374930224586'. Below this is a table of variables:

Name	Value	Type
da_gfr	0.09274092231603...	Real
c_int	981.8907388044913	Real
svmaxt_kid	1.003002	Real
p_met	11.325955444373669	Real
q_total	0.877518822995923	Real
a_sp	49.92218400387828	Real
da_rabs	0.00532904810791...	Real
stb_lng	0.37668	Real
q_int	0.12373015404242...	Real

Below the table is a control panel with several buttons: a green play button, a blue double right arrow, a blue single right arrow, a red X, a green plus, and a red minus. Below these are two rows of checkboxes and labels:

Variable	Op	Value
<input checked="" type="checkbox"/> t	>=	5.0
<input checked="" type="checkbox"/> c_ven	>=	1.0E8

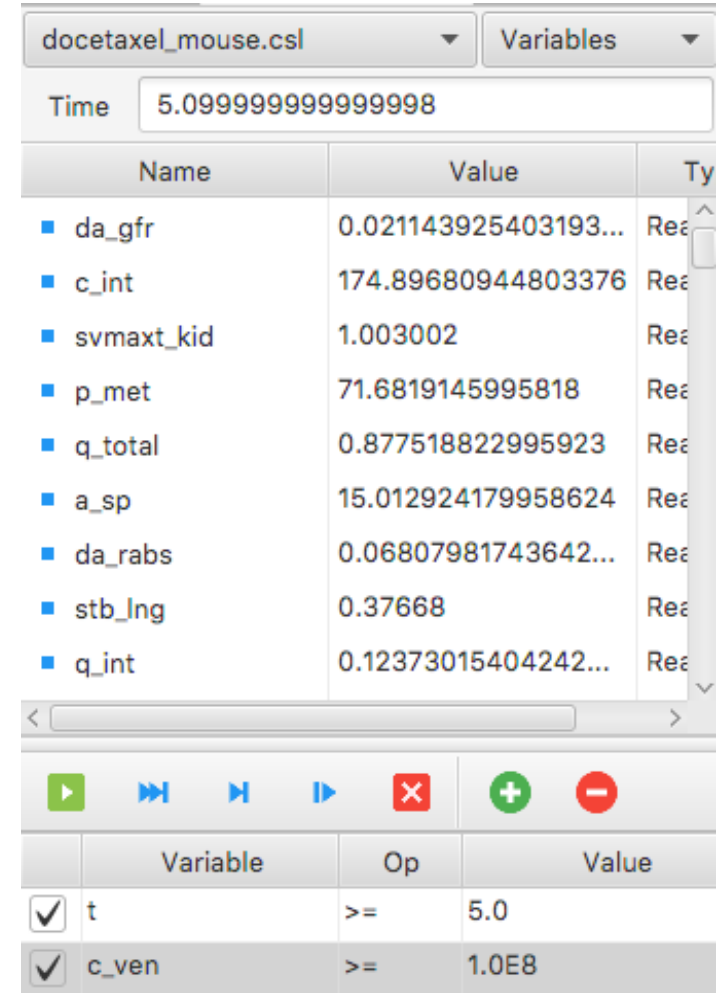
At the bottom, a status bar reads 'Stopped on derivative evaluation.' Three arrows point from the text on the left to the control panel: the first arrow points to the blue double right arrow button, the second arrow points to the blue single right arrow button, and the third arrow points to the red X button.

Exercise 2.6 (continued)

Now hit the “continue” button on the mini toolbar to simulate until a breakpoint condition is satisfied. Since we’re just pretending that the model encounters an error around $T = 5$, the time-based breakpoint will be the condition that triggers the simulation to pause (because c_{ven} will never exceed $1E8$). The model inspector window will look something like the image to the right. Note that we overshoot our desired time by a small amount, likely due to numerical imprecision in the “ ≥ 5 ” comparison. If necessary, adjust to the stopping condition and re-start the debug run.

Once the simulation pauses, all model output values can be inspected in the variables table to help understand what quantities/equations are causing the problem. You can also continue stepping by time or by derivative calculation by hitting the corresponding buttons on the mini toolbar.

Many scripting commands work as well while the simulation is paused. For example, you can display the values of model variables and constants at the command prompt, or create static (non-interactive) plots using the CMD language (see the section on scripting languages for details). Once you’re done debugging, hit the red “stop” button to end the debugging session.



The screenshot shows the 'docetaxel_mouse.csl' model with the 'Variables' table. The 'Time' field is set to 5.099999999999998. The table lists various model variables and their values. Below the table is a mini toolbar with buttons for running, stepping, and stopping the simulation. At the bottom, a table shows the current breakpoint conditions.

Name	Value	Type
da_gfr	0.021143925403193...	Real
c_int	174.89680944803376	Real
svmaxt_kid	1.003002	Real
p_met	71.6819145995818	Real
q_total	0.877518822995923	Real
a_sp	15.012924179958624	Real
da_rabs	0.06807981743642...	Real
stb_lng	0.37668	Real
q_int	0.12373015404242...	Real

Variable	Op	Value
<input checked="" type="checkbox"/> t	\geq	5.0
<input checked="" type="checkbox"/> c_ven	\geq	1.0E8