Crash Course: Spreadsheet Modeling and Computer Programming (i.e., Coding)

Kyle McKay
601-415-7160
Kyle.McKay@usace.army.mil

Ecological Modeling Workshop
New England District
Course Overview

- Introduction
- Modeling Basics
- Modeling Process + Lab
  - Conceptualization
  - Quantification
  - Evaluation
  - Iteration, Application, and Communication
- Crash Courses
  - Model certification
  - Coding Basics
  - Model Selection, Adaptation, and Integration
  - Incorporating Spatial Dynamics
Lecture Overview

- Avoiding spreadsheet errors
- Thinking like a computer
- Basic computation
- Logic statements and loops
- Model documentation
Avoiding Spreadsheet Errors

Why discuss spreadsheets?

<table>
<thead>
<tr>
<th><strong>Strengths</strong></th>
<th><strong>Shortcomings</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Inexpensive to develop</td>
<td>▪ False sense of confidence due to the easy output of “answers”</td>
</tr>
<tr>
<td>▪ Readily available</td>
<td>▪ Can quickly become complex</td>
</tr>
<tr>
<td>▪ Transferable to a broad user base</td>
<td>▪ Error rates can be high (90% of spreadsheets, 1-5% of cells)</td>
</tr>
<tr>
<td>▪ Simple, logical interfaces</td>
<td>▪ Troubleshooting and error checking may be time-consuming and difficult (as is the case with all computer programs)</td>
</tr>
<tr>
<td>▪ Easily understood and applied without extensive training</td>
<td>▪ Written quality assurance and control techniques are not often applied due to reliance on self-policing</td>
</tr>
<tr>
<td>▪ Built-in tools and third-party add-ins provide computational power without extensive programming</td>
<td></td>
</tr>
<tr>
<td>▪ Current spreadsheet programs have many capabilities previously associated with more sophisticated software (e.g. optimization routines)</td>
<td></td>
</tr>
</tbody>
</table>
Reducing Errors

- High error rates suggest that thorough checking, testing, and auditing are uncommon

- Four Key Techniques for Reducing Errors:
  - Planning spreadsheet development
  - Avoiding errors during development
  - Finding errors
  - Self-improvement
Planning Development

Who uses spreadsheets?
- Sponsor – assigns the task
- Developer – designs the spreadsheet
- Auditor – checks the validity and verifies results
- User – inputs data and operates the model
- Reader – uses the results of the model to inform decisions

What constitutes a “good” spreadsheet?
- Simple
- Relevant
- Adaptable
- Reliable
- Appropriate
Avoiding Errors: General

- Work in groups
- Build complex spreadsheets in stages and test results as you go
- Fix problem immediately – Avoid “quick fixes”
- Take advantage of Visual Basic for Applications (VBA)
- Document your work

### Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Length (ft)</td>
<td></td>
<td></td>
<td>Kyle McKay:</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Width (ft)</td>
<td></td>
<td></td>
<td>Maximum length of the property.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Area (ft²)</td>
<td></td>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>

**HYDRAulic ROughness CALCulator (HYDROCAL): User’s Manual**

S. Kyle McKay and J. Craig Pickens

U.S. Army Engineer Research and Development Center
Environmental Laboratory

197 Galbraith Ave, Apt. B, Athens, GA 30606
Kyle McKay@usace.army.mil

3909 Main Perry Rd., Okolona, MS 38949
Craig J. Pickens@usace.army.mil

February 11, 2008
Draft for Internal Review
Avoiding Errors: General (cont)

- Descriptively name cells, rows, columns, worksheets, and workbooks
- Use reference styles and locations correctly.
- Use security features
- Delete extraneous information and cells
- Be forewarned. Copy/paste and autofill errors are common

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input</td>
<td>x</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Calculation</td>
<td>-10</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>-9</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>y = ax^2+2bx</td>
<td>-8</td>
<td>=B8*(D4^2)+2<em>B9</em>D4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td>2</td>
<td>-6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>0</td>
<td>-5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>-4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>-3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>-1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Avoiding Errors: Logic and Calculation

- Complicated expressions are potentially hazardous
- Avoid “hard-wiring” constants into formulas
- Maintain a common interval for calculation
- Arrange inputs close to calculations
- Reserve circular references as error detection tools. Learn to apply GoalSeek and Solver instead.

![Excel spreadsheet](image)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>2</td>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>8</td>
<td>Calculation</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>$y = x^2 + 2x$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Solver Parameters dialog box](image)
Avoiding Errors: Formatting and Labeling

- Make your spreadsheet read top-to-bottom & left-to-right
- Do not turn the grid off
- Format for function, not for decoration
- Organize your spreadsheet based on functionality (e.g. input, calculation, output)
- Remember significant figures
- Make blank cells look blank

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Length (ft)</td>
<td>10</td>
<td>Input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Width (ft)</td>
<td>50</td>
<td><strong>Calculation /Output</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Area (ft²)</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Avoiding Errors: Figures

- Use the right chart for the job
- Consider changing chart location
- Modify charts such that all data sets are visible
- Think twice about scale on charts
Finding Errors

- Who should look for errors?
  - THE WHOLE TEAM: Developers, Auditors, Users, Readers
- Review during construction
- Use cross-checks and validation points
- Test spreadsheets
- Inspect all formulas
- Use auditing tools (next slide)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Length (ft)</td>
<td>10</td>
<td></td>
<td>Input</td>
</tr>
<tr>
<td>2 Width (ft)</td>
<td>50</td>
<td></td>
<td>Calculation /Output</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Area (ft²)</td>
<td>=B1*B2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Perimeter (ft)</td>
<td>=2<em>B1+2</em>B2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How are audits conducted?

Auditing Tools
- Microsoft Excel Built-In Auditing Tools
  - "Formula Auditing" toolbar
  - Warning messages
- XL Analyst – [http://www.xlanalyst.co.uk](http://www.xlanalyst.co.uk)
- Spreadsheet Professional – [www.spreadsheetinnovations.com](http://www.spreadsheetinnovations.com)
- Operis Analysis Kit – [www.operis.com/oak.htm](http://www.operis.com/oak.htm)
- XLSior – [www.xlsior.com](http://www.xlsior.com)

Comparison Tools
- Synkronizer – [http://www.synkronizer.com](http://www.synkronizer.com)
Excel Formula Auditing Toolbar

- Find error cells
- Trace precedents
- Trace dependents
What about other errors?

- Many errors lie outside the realm of what model developers can control
  - Data errors
  - Input errors
  - User errors
  - Inappropriate application
  - Conceptual errors
  - Misinterpretation of results
  - Deliberate errors
Self-Improvement

- Be humble
- Be consistent
- Don’t be caught by avoidable errors
- Stay organized
- Be a student of spreadsheets
- Develop corporate policy (e.g. Model Certification initiative)
  - Risk-based
  - Train developers, users, and readers in potential hazards, avoidance, and self-audit.
  - Modeler accountability must be addressed.
  - Train auditing specialists
Where else can I go for spreadsheet best practices?

- European Spreadsheet Interest Group (EUSPRIG), www.eusprig.com
- Ray Panko, University of Hawaii, http://panko.shidler.hawaii.edu/
- Systems Modeling Ltd., http://www.sysmod.com/
Trade-offs between Coding & Excel

- Up-front investment in time
- Capacity to trouble-shoot
- Potential modification of code
- “Clean” appearance
- Speed of computation
- Computational limits
Thinking like a computer
A computer’s inner monologue
Programmers must tell computers what to think about

Multiply 5 times 2

10!
Computers think quickly!

Multiply 5 times every number from 1 to 1,000,000

5, 10, …
BUT they get confused easily

Multiply 5 times “X”

ERROR!
So you must always be very explicit

If $x=4$, multiply 5 times $x$. 

20!
Computer Programming

- Computer programming is an explicit instruction for the computer to do something

- A scripting language is simply a mechanism for you to communicate with the computer
Coding... just another language

- FORTRAN
- PASCAL
- C++
- Java
- MATLAB
- JMP
- WinBugs
- R
- Python
- French
- Spanish
- Mandarin
- English
- Italian
- Farsi
- Flemish
- Hindi
- German
- ...
- ...

Innovative solutions for a safer, better world
R Project

- Free scripting language
- Open source
- User developed

http://cran.us.r-project.org/

4,782 freely downloadable add-ins for specific functions ("packages")
The R interface

- Base-R
- “Wrapper” – Rcommander, Rstudio,…

- Command line – manually entered each time
- Script – saved for later
Defining Variables

- Must tell the program what a variable is called
- Can be done using either “<-” or “=“
  - “<-” is more general
Naming Variables

- Name them something recognizable
- Use comments to define them in more detail
  - e.g. n #Manning’s n
- Punctuation and numbers can be helpful
  - e.g. x3, x.3, x_3
- Things that DO NOT WORK
  - Numbers before letters (e.g. 3x)
  - Punctuation that means something to R (e.g. , ! <)
# A couple of notes about the R language

<table>
<thead>
<tr>
<th>Mark</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Hash marks denote comments within the code. These tell the program to ignore everything else on this line.</td>
</tr>
<tr>
<td>;</td>
<td>Semicolons combine multiple lines of code on the same line.</td>
</tr>
<tr>
<td>Space / tab</td>
<td>R ignores these (a few exceptions exist).</td>
</tr>
</tbody>
</table>
Clearing the Deck

- R stores variables until you over-write them or tell the program to clear memory

- This can lead to problems when you’re developing code. Your code may look like it’s working, but it could be using stored data.
Whenever you get lost...

- `?function` returns the help menu for any "function"
- Google and help-boards are amazing because of the huge R user community

Go to R for demonstration.
Basic Computation
Basic computation

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Exponentiation (^)

Order of operations is critical: PEMDAS
Vectors

- User-specified vector: `c(0, 1, 2, 3)`
- Sequence from 0 to 3 by 1: `seq(0, 3)`
- Sequence from 0 to 3 by 0.1: `seq(0, 3, 0.1)`
- Sequence from 3 to 0 by -0.1: `seq(3, 0, -0.1)`
Matrices can be your friend!

- \( \text{xyz} \leftarrow \text{matrix}(0, \text{nrow}=20, \text{ncol}=5) \)

- Call values from a matrix
  - One value \( \text{xyz}[\text{row},\text{column}] \)
  - One row \( \text{xyz}[\text{row},] \)
  - One column \( \text{xyz}[,\text{column}] \)
Basic functions

- sum
- mean
- min
- max
- median
- length
- quantile

- function(arguments)

- Many pre-existing functions in “Base R”
- Load other functions from packages
- Develop your own functions
Loading “packages”

- What package do you need?
  - [http://cran.us.r-project.org/](http://cran.us.r-project.org/)

- Installing the package
  - Packages -> Install Package -> Select mirror

- Loading the package
  - Packages -> Install Package -> Select package
  - Library(package)

Go to R for demonstration.
Logic and Loops
Logic Statements

- If-then
  - If time = 0300, then Kyle is asleep.

- Testing alternative models
  - Looping through many competing models

- Threshold based models
  - If sunset < time < sunrise, then primary production = 0.
  - If sunrise < time < sunset, then primary production = \(x\) g/m\(^2\).

\[
Q_w = \begin{cases} 
Q_{w,\text{max}} & Q_r > Q_{w,\text{max}} + Q_{r,\text{min}} \\
Q_r - Q_{r,\text{min}} & Q_{w,\text{max}} > Q_r > Q_{r,\text{min}} \\
0 & Q_{r,\text{min}} > Q_r
\end{cases}
\]
Loops

- **For**
  - Iterating over a specific interval and period
  - e.g., Calculate AAHUs for year-1 to year-50

- **While**
  - Iterating until a condition is met
  - e.g., calculate AAHUs while time < 51 years
Nested loops

- Combining multiple types of loops to do fun things!
- Multiple for loops to populate a matrix
- A for loop and an if statement to apply multiple models to a single vector

Go to R for demonstration.
Documentation
What should you document?

- Why did you develop the model (purpose)?
- How was the model developed (process)?
- Who is the target user? How is the model operated?
- What algorithms are used in the model (literature support)?
- What is the general flow of data and logic?
- What assumptions were made?
- Who can be contacted about the model?
- Are there multiple versions or releases?
- What error checking, peer review, testing, verification, or validation has been conducted?
More documentation is always better!
(but this doesn’t mean longer reports)

- **Internal documentation**
  - Annotating code with 3X more comments than you think.
    - Section comments
    - Line-by-line comments
    - Descriptive naming of variables (and associated units)!
  - In Excel, “How to” tabs, internal comments, and descriptive naming can be invaluable.
  - Can someone not involved in the project understand any line of code (even without programming experience)?

- **External documentation**
  - Model development process
  - Error checking
  - Maintain consistency between documentation and version
References for Further Reading

- [http://cran.us.r-project.org/](http://cran.us.r-project.org/)
  - The HUGE user community has web support and packages for nearly anything you could image. Google can get you a long way…


  - Freely available here: [http://ms.mcmaster.ca/~bolker/emdbook/](http://ms.mcmaster.ca/~bolker/emdbook/)
Take-away Points:

- Spreadsheets are powerful tools, but they are prone to errors (and are not always the right tool for the job).

- Our goal was NOT to teach you a programming language, but instead to give you an appreciation for the potential power of scripting languages and provide enough references to send you on your way.

Next Lecture:

- Model selection, adaptation, and integration
Other important concepts in coding

- Functions
- Probability distributions
- Plotting
- Random numbers
- User communities
- Importing data