Verification and Validation

CS1538: Introduction to Simulations
Steps in a Simulation Study

- Problem & Objective Formulation
  - Model Conceptualization
  - Data Collection
    - Model translation, Verification, Validation
      - Experimental Design
        - Experimentation & Analysis
          - Documentation, Reporting, Implementation
Verification and Validation

First we must distinguish between the two

Verification

- Is the process of building / implementing the model correct?
  - Is the final result an accurate implementation of the model?
  - Note that verification does not address the issue of whether the model itself is accurate or not

Validation

- Is the model that is being implemented the correct one (does it accurately model the system in question)?
- This is testing the correctness of the model itself
Verification

- General “program correctness” techniques
  - for software engineering and general programming

- Good software development practices

- Plus, simulation-specific things to look out for
Validation

- How do we know that the model we have developed is an accurate representation of the real system?
  - Often an iterative process (calibration)
    - A system is evaluated against the real system, modified, and evaluated again
    - The process continues until the model is deemed satisfactory overall
  - Subjective evaluation
    - Experts examining the model and perhaps some test output, to determine if it fits the real system
  - Objective evaluation
    - If we have some real data, we can compare them against the simulation data using statistical tests
3-Step Approach

Naylor and Finger’s 3-step approach to validation:

1) Build a model with high face validity

2) Validate model assumptions

3) Compare model I/O with real system I/O
Face Validity

- Does the model seem reasonable to domain experts?
  - This can be done prior to implementation

- Does the model behave in an expected way in a given situation? Is the output reasonable?
  - We are not analyzing the data in detail (done in step 3)
  - Just looking for blatant incorrectness
    - e.g. negative wait times or counts
Validate Model Assumptions

- Banks classifies these into two groups:
  - **Structural assumptions**
    - Relate to the operation of the system
    - How data is manipulated once inside
    - Deals with the data structures and algorithms used
      - Ex: Single FIFO Queue
  - **Data assumptions**
    - This takes the data issues from the face validity and looks at them in more detail
    - Are the distributions used correct / accurate?
      - Goodness-of-Fit tests: Kolmogorov-Smirnov Test or Chi-Squared Test
      - Graphical methods
Validate Input / Output Transformations

- Need an objective way to test the model for correctness / accuracy
  - Identify the input and output variables of interest
  - Obtain sample data of the real system of interest if possible
  - Run the simulation and compare the results to those obtained for the real system
    - Analyze mathematically to see if the model “fits” the system
    - If the results are poor / incorrect, we need to find out why and revise the model
We want to consider two different kinds of variables needed by our simulation

- **Uncontrollable variables**, whose values are determined by the external environment
  - These could be interarrival times, delays, and possibly service times
  - Denote these by the array $X$

- **Decision variables**, whose values are controllable within the simulation (i.e. can be changed)
  - Denote these by the array $D$

Given $X$ and $D$, our simulation will produce some output data, $Y$, based on the protocols implemented – Call this transformation $f$, leading to:

$$f(X, D) = Y$$
Comparing Generated and Measured Data

- How do we actually test to see if our output matches the expected data?
  - Consider some output variable $Y_k$, taken over multiple, independent simulation runs
    - We calculate the average value and standard deviation of the output over those runs
  - We’d like to see if this “matches” our expected value for the variable
  - We will use the t-test
    - Could also use confidence interval and see if it matches expected data
One-Sample T-test

- The basic routine for applying the t-test is similar to how we applied the chi-square test before
  - Assuming our outputs follow a normal distribution, the difference between the observed mean and the expected mean follows what is known as the student t-distribution
  - We have a null hypothesis that the observed average is not too far off from the expected value
One-Sample T-test

- The basic routine for applying the t-test is similar to how we applied the chi-square test before
  - We compute the t value
    \[ t_0 = \frac{\overline{Y}_k - \mu_0}{S / \sqrt{n}} \]
    Where:
    - $\overline{Y}_k$ = mean of simulation results
    - $\mu_0$ = hypothesized mean
    - $S$ = standard deviation of simulation results
    - $n$ = number of simulation runs used
  - Reject the null hypothesis if $t_0 \geq t_{\alpha/2, n-1}$ or if $t_0 \leq -t_{\alpha/2, n-1}$
One-Sample T-Test

- For example, assume we know \( \mu = 28 \)
- We have run a simulation that generated the following 10 values:

  19.60598106
  25.87288018
  24.25071858
  19.48842905
  33.80513566
  35.27167241
  24.78146019
  25.81875958
  24.62708968
  22.79642059

\[
\begin{align*}
Y' &= 25.63 \\
S &= 5.22
\end{align*}
\]

Thus, we look up \( t_{0.05,9} \) in the table and find: 1.83

Since the table is symmetric, we can use the absolute value, and \(|-1.436| < 1.83\) so we do not reject the null hypothesis.
Comparing Generated and Measured Data

- So far, our tests have been focused on **Type I error**: \[ \Pr(H_0 \text{ rejected } | H_0 \text{ is true}) = \alpha \]
- However, we should also make sure that we don’t accept a hypothesis when it is false – a **Type II error**
  \[ \Pr(\text{failure to reject } H_0 | H_1 \text{ is true}) = \beta \]
- Type II error is harder to check than Type I error (because \( H_1 \) states an inequality: “\( Y' \neq \mu \)”)
- We have to consult the **operating-characteristic** curves, which shows the probability of a Type II error versus a quantity \( \delta \) which is defined as \( \delta = |Y' - \mu|/S \).
  - One thing the curves can tell us is how big \( n \) has to be in order for us to reduce \( \beta \) to a reasonable value.
Example: Single-Server Grocery Checkout
(Ex. 10.1 from Banks et al.)

- A small grocery store has one checkout counter. Customers arrive at random times from 1 to 8 minutes apart (discrete uniform distribution). Service times vary from 1 to 6 minutes (integer values only, following distr. below).

- An analyst ran the simulation over 16 time units and found $\hat{L}_Q = 0.4375$
  - Is this reasonable?

- Trace of simulation on next slide

<table>
<thead>
<tr>
<th>Service Time (minutes)</th>
<th>Pr(T=t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.10</td>
</tr>
<tr>
<td>2</td>
<td>0.20</td>
</tr>
<tr>
<td>3</td>
<td>0.30</td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
</tr>
<tr>
<td>5</td>
<td>0.10</td>
</tr>
<tr>
<td>6</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Example: Single-Server Grocery Checkout
(Ex. 10.1 from Banks et al.)

- Is $\hat{L}_Q = 0.4375$ reasonable?
- Trace for simulation

<table>
<thead>
<tr>
<th>Clock</th>
<th>Event Type</th>
<th>Ncustomers</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Arrival</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Depart</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Arrival</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Arrival</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>Depart</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- $N_{customers} = \#$ customers in system at time
- Status = status of server (0 = idle, 1 = busy)
Example: Homework 4

- Are there any unreasonable/unrealistic assumptions?

- Are there any unstated assumptions?