Modeling - Framework and Falling Penny myth

August 31, 2023

Contents

1 Modeling framework **1**

2 Modeling Examples **4**

3 The Falling Penny Myth **5**

4 References **8**

1 Modeling framework

- If modeling is "defined" by the relationships shown in the diagram (from Downey's book p.4), where does coding with interactive notebooks fit in? More specifically, which parts of the workspace play a role in which part of the diagram? (link)

    1. The workspace is a software system. It can be subjected to measurements, which generate data (about the workbook), e.g. session time.
    2. The notebook can be used to create a model of a real system: e.g. the unicorn data frame (and data set) is an abstraction because only certain features (columns) are retained.
    3. The model (data frame) can then be subjected to further analysis, e.g. we can build a model to predict the frequency of unicorns in industries based on the collected data (what type of model?)

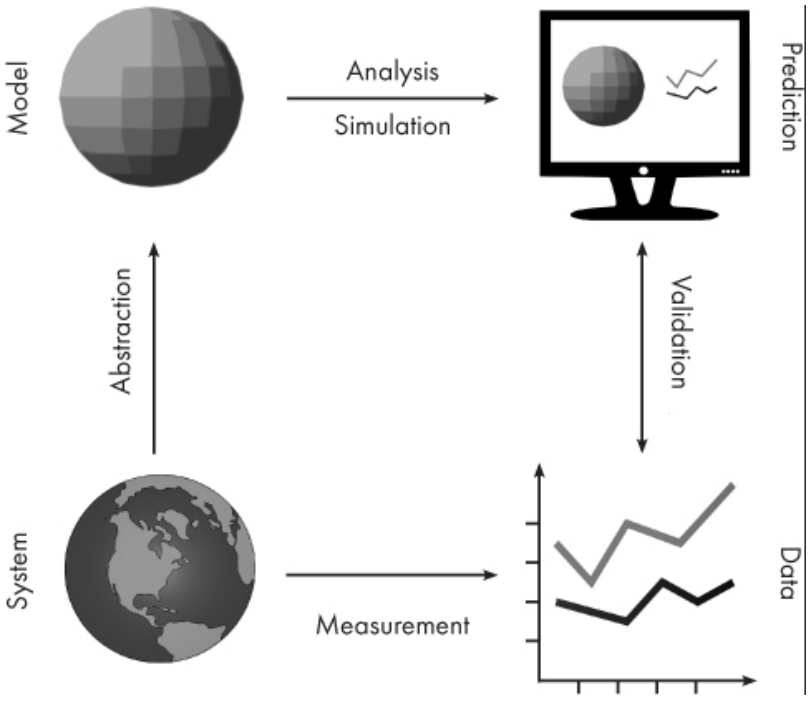

Figure 1: Source: Downey (2023)

4. Sticking to the fact that the workspace itself is a system, we could predict future session time lengths based on previous usage. A lineplot would show the session times over time (that is a time series), and linear extrapolation would predict the evolution of the session times.
5. To validate the prediction, further measurements can be taken and plotted alongside the prediction.

- These three results are the three outcomes of analytics:

  1. descriptive analysis tells us what has already happened;
  2. predictive analysis shows us what could happen;
  3. prescriptive analysis informs us what should happen.

- An alternative model: discuss the differences! (link)

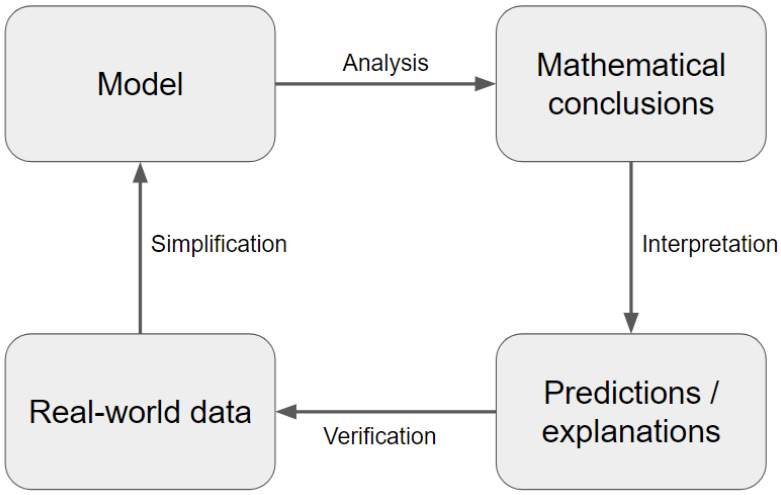

Figure 2: Source: Giordano et al. (2014)

  - "Simulation" (from Model to Prediction) is missing
  - "Validation" (from Prediction to Data) goes both ways while "interpretation" of a mathematical model only goes one way.
  - System is equivalent to Real-world data but instead of "verification" of the predictions or explanations, the first

model posits measurements to obtain data which enter a feedback loop with the predictions.
- The first model is more general, the second one only deals with mathematical modeling of real-world data.
- Deep learning models for example, which are trained on real-world data and can be validated using test data, are not covered here.

2 Modeling Examples

Weather forecasting

1. **System:** Earth's atmosphere (layers, temperature gradients, pressure differences, wind patterns).
2. **Model**: Computational model representing atmospheric interactions (Navier-Stokes PDEs, thermodynamic laws, radiation balance).
3. **Prediction**: Weather forecast (temperature, humidity, chance of precipitation).
4. **Data**: Real-time meteorogical data (current temperature, wind speed, satellite images).

Stock market analysis

1. **System:** Stock market (stocks, indices, traders, economic indicators like interest rates, unemployment).
2. **Model**: Quantitative models to predict stock prices (Time Series forecasting, sentiment analysis, neural nets).
3. **Prediction**: Stock prices or trends.
4. **Data**: Historical stock prices, trading volumes, economic indicators, social media sentiment data.

Epidemic spread

1. **System:** Population of infected individuals who are exposed to, infected with, or recovered from a contagious disease.

2. **Model**: Compartmental SIR model (Susceptible, Infected, Recovered) that uses ODEs to describe how individuals move between compartments over time.

3. **Prediction**: Project number of individuals in SIR compartments over time.

4. **Data**: Actual case counts, hospitalization rates, recovery rates, collected from hospitals, labs, public health agencies.

3 The Falling Penny Myth

Workspace problem: `https://tinyurl.com/FallingPennyMythProblem`

Question: Would a penny dropped from the top of the Empire State Building go so fast that it would be embedded in the concrete; or if it hit a person, would it break their skull?

Solution 1: strong assumptions.

$v = at$ is the velocity of an object after t seconds.

The distance travelled is $x = at^2/2$.

The time until the penny reaches the sidewalk: $t = \sqrt{2x/a}$

For constant gravity $a = 9.8m/s^2$ and the height of the Empire State Building $x = 381m$, we get $t = 8.8s$.

The velocity after that time is $v = 86m/s$ (190 mph).

Modeling assumptions: constant gravity (not true, varies with distance even in classical mechanics), zero air resistance.

Solution 2: downward gravitational force and upward drag (or air resistance) force are opposed and will eventually cancel each other out so that the object will no longer be accelerated. It has now reached *terminal velocity*.

Terminal velocity is reached when:

$$m \cdot g = \frac{1}{2} \cdot C_d \cdot A \cdot \rho \cdot V_t^2 \tag{1}$$

Solving for the velocity, we get:

$$V_t = \sqrt{\frac{2 \cdot m \cdot g}{C_d \cdot A \cdot \rho}} \tag{2}$$

Where m is the mass of the object, g is the acceleration due to gravity, C_d is the drag coefficient (empirical dimensionless weight depending on the

object in relation to the fluid), A is the cross-sectional area of the object, and ρ is the density of the fluid through which the object is falling.

Substituting for our penny, this comes out to about 17 m/s or 38 mph (while a sky diver, for example, will experience 53 m/s or 120 mph).

To see if and how this hurts, watch this video by MythBusters (Your Discovery Science, 2015).

So the first model was wrong and not useful to determine the truth of the myth. The second model is still wrong but it's better and good enough to refute the myth (via falsification).

Falsification here means: if I make a claim (the myth), and I can find one empirical example that it is not true, my claim has lost its generality and will no longer be scientifically valid.

Computation in Python

Create a variable `a` to represent acceleration in meters per second squared:

```
a = 9.8
```

Create another variable to represent the time for the penny to drop, let's say 3.4 seconds:

```
t = 3.4
```

Now we can compute the velocity `v` of the penny after `t` seconds (ignoring air resistance):

```
v = a * t
print(v)
```

What distance `x` would the penny travel during that time?

```
x = a * t**2 / 2
print(x)
```

How long would it take for the penny to fall 381 m, the height of the Empire States building? First, let's store the height in a variable `h`:

```
h = 381
```

To compute t, we need the square root function. We can import the function from the NumPy package:

```
from numpy import sqrt
```

Now we can use it to compute `t` in seconds:

```
t = sqrt(2 * h / a)
print(t)
```

Finally, let's calculate the penny's velocity `v` in m/s:

```
v = a * t
print(v)
```

Exercise: can you compute the terminal velocity for the penny given this formula:

$$V_t = \sqrt{\frac{2 \cdot m \cdot g}{C_d \cdot A \cdot \rho}} \tag{3}$$

Where m is the mass of the object ($2.5 \times 10^{-3} kg$), g is the acceleration due to gravity ($9.8 m/s^2$), C_d is the drag coefficient (0.47), A is the cross-sectional area of the object ($2.85 \times 10^{-4} m^2$ - a penny has a radius of $9.525 mm$), and ρ is the density of the fluid through which the object is falling ($1.204 kg/m^3$).

Solution:

```
# Assign constants
m = 2.5e-3  # kg
g = 9.8     # m/s^2
c_d = 0.47
A = 2.85e-4 # m^2
rho = 1.204

# Compute terminal velocity
v = sqrt((2 * m * g) / (c_d * A * rho))
print(v)
```

Exercise: check the penny area computation computationally.

Solution: we need the constant value for π, or we could approximate it to 3.14, and the known area formula (πr^2):

```
r = 9.525e-3
A = r**2 * 3.14
print(A)
print(f'{round(A*1e4,2):.2f}')
```

Let's do it with a higher precision `pi` using `NumPy`:

```
from numpy import pi
A = r**2 * pi
print(A)
print(f'{round(A*1e4,2):.2f}')
```

4 References

Downey AB. Modeling and Simulation in Python. NoStarch Press; 2023. https://allendowney.github.io/ModSimPy/

Giordano FR, Fox WP,Horton SB. A First Course in Mathematical Modeling (5e). Cengage Learning 2013.

Google LLC. Google Colaboratory. Accessed August 19, 2023. https://colab.research.google.com

Python Software Foundation. Python (Version 3.8.10). Python Software Foundation. Published 2021. Accessed August 19, 2023. https://www.python.org