Exploring Modeling with Data and Differential Equations Using R

John M. Zobitz

Version 2.0.0

Contents

Welcome

This book is written for you: the student learning about modeling and differential equations. Perhaps you first encountered models, differential equations, and better yet, building plausible models from data in your Calculus course.

This book sits "at the intersection" of several different mathematics courses: differential equations, linear algebra, statistics, calculus, data science - as well as the partner disciplines of biology, chemsitry, physics, business, and economics. An important idea is one of *transference* where a differential equation model applied in once context can also be applied (perhaps with different variable names) in a separate context.

I intentionally emphasize models from biology and the environmental sciences, but throughout the text you can find examples from the other disciplines. I hope you see the connections of this content to your own intended major.

This book is divided into 4 parts:

1. Modeling with differential equations and data.
2. Model parameter estimation.
3. Stability analysis for differential equations.
4. Modeling with stochastic differential equations.

This may seem like a different order than traditionally presented. This is a "modeling first" paradigm that first introduces models, and equally important, how to estimate parameters for a model using data. This conversation between models and data are important to help build plausibility. Stability analysis helps to solidify the connection between models and parameters (which may change the underlying dynamical stability). Finally the notion of *randomness* is extended with the introduction of stochastic differential equations.

Computational code

This book makes heavy use of the R programming language, and unabashedly develops programming principles using the `tidyverse` syntax and programming approach. This is intentional to facilitate direct connections to courses in introductory data science or data visualization. Throughout the years learning (and teaching) different programming languages I have found R to be the most versatile and adaptable. The `tidyverse` syntax has also been transformational for me in my own work and as my students - the barrier to compute and write code is lowered.

There is a companion R package available to run programs and functions in the text. Instructions to do so are given in Section **??**. The minimum version of R Version 4.0.2 (2020-06-22) and RStudio is Version 1.4.1106.

Acknowledgments

This book has been developed over the course of several years and has been written across different continents.

- **Augsburg University:** You have been my professional home for over 14 years and given me the space to be intellectually creative in my teaching. I have great colleagues to work with.
- **Augsburg University students:** Thank you for your interest in this topic, providing honest and insightful feedback about the course. This has been a work in progress (albeit bumpy at times).
- **My family:** Shannon, Colin, Grant, and Phoebe for humoring me while this project has been completed.
- **Waterparks, coffee shops, soccer practices:** Many times this was written "in the spaces" between work and home, and especially during downtimes when my kids could play. Turns out my kids love waterparks. Who knew?

Copyright

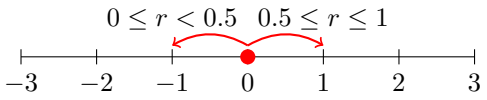
Chapter 1

Random Walks

In the last section we saw how to introduce stochasticity into a discrete dynamical system and examined the differences in the results. In this section we will begin to develop some tools about how to understand stochastics by studying *random walks*.

1.1 Random walk on a number line

The conceptual idea of a random begins on a number line. Let's begin at the origin (so at $t = 0$ then $x = 0$). Based on this number line we can only to the left or the right, with equal probability. At a given time we decide to move in a direction based on a random number r drawn between 0 and 1 (in `R` we do this with the command `runif(1)`). Figure **??** conceptually illustrates this random walk



For each iteration of this process we will draw a random number using `runif(1)`. We can code this process using a `for` loop:

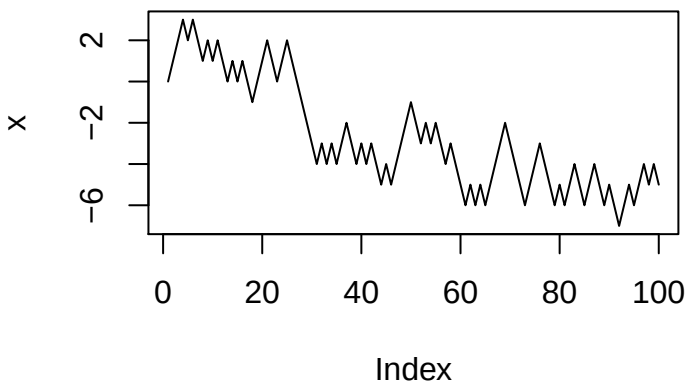
```
number_steps <- 100     ### Number of times we run our simulation

### Set up vector of results
x <- array(0,dim=number_steps)

for (i in 2:number_steps) {
  if (runif(1)< 0.5) {x[i]=x[i-1]-1}   # Move right
    else {x[i] <- x[i-1]+1}   # Move left
  }

# Let's take a peek at our result:

plot(x,type='l')
```



```
print(mean(x))  # Where our average position was over the time interval
```

```
## [1] -2.72
```

```
print(sd(x))  # Where our standard deviation was over the time interval
```

```
## [1] 2.640324
```

Let's remind ourselves what this code does:

- `number_steps <- 100`: The number of times we draw a random number, referred to this as steps.
- `x <- array(0,dim=number_steps)`: We are going to pre-allocate a vector (`array`) of our results. Values in this array are all set at 0 for convenience.
- The for loop starts at the second step and then either adds or subtracts one from the prevoius position `x[i-1]` and updates the result to `x[i]`.
- `plot(x,type='l')` makes a simple line plot of the results.

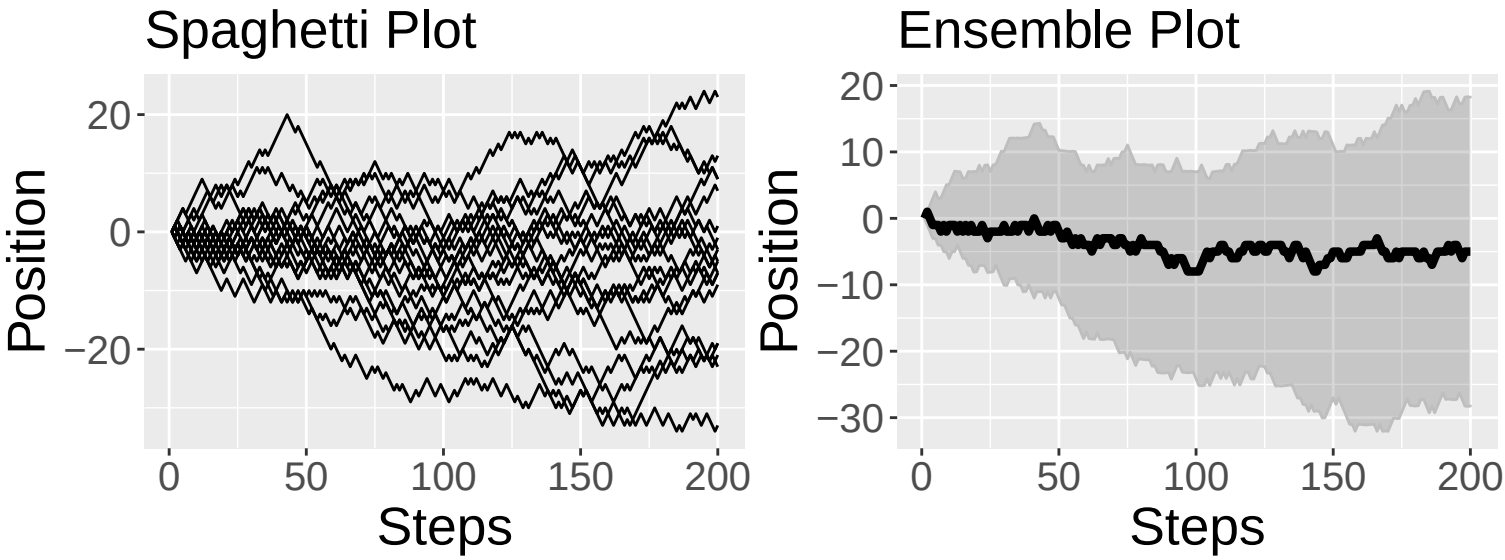
Now that you have run this code, try running it again. Do you get the same result? I hope you didn't - because this process is random! It is interesting to run it several times because there can be a wide variance in our results - for some realizations of the sample path, we end up being strictly positive, other times we go negative, and other times we just hover around the middle.

1.2 More realizatons

Similar to section on stochastic simulation, One thing that would be helpful is if we ran the simulation for multiple times, or multiple realizations. The code `randomWalk` can help us do that:

```
number_steps <- 200  # How long we run our random walk
number_realizations <- 20 # How many separate realizations we do

random_walk(number_steps,number_realizations)
```

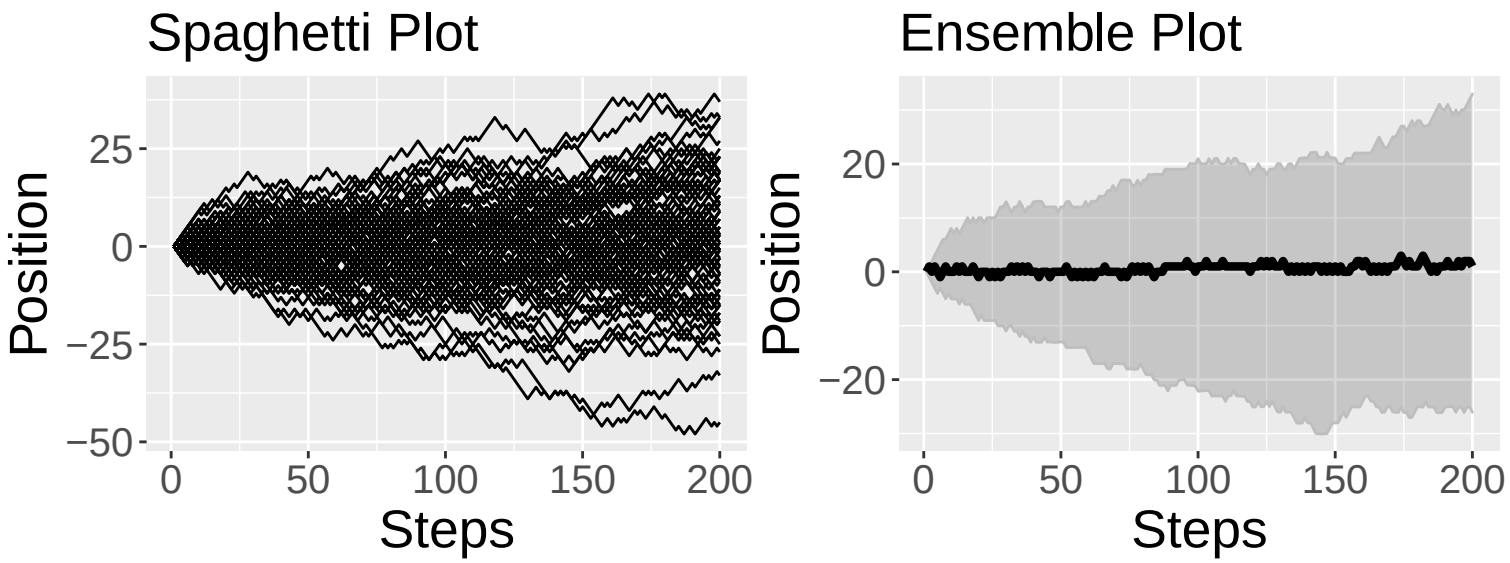


You will notice two plots get produced: (1) A **spaghetti plot** that plots all the different sample paths for this realization, and (2) a **ensemble plot** that takes the median and 95% confidence interval of the results.

Something interesting looks like it is going on here. The ensemble plot looks like a sideways parabola, but let's check to make sure that is the case. Perhaps rerun `random_walk` but set `number_realizations` to be 100:

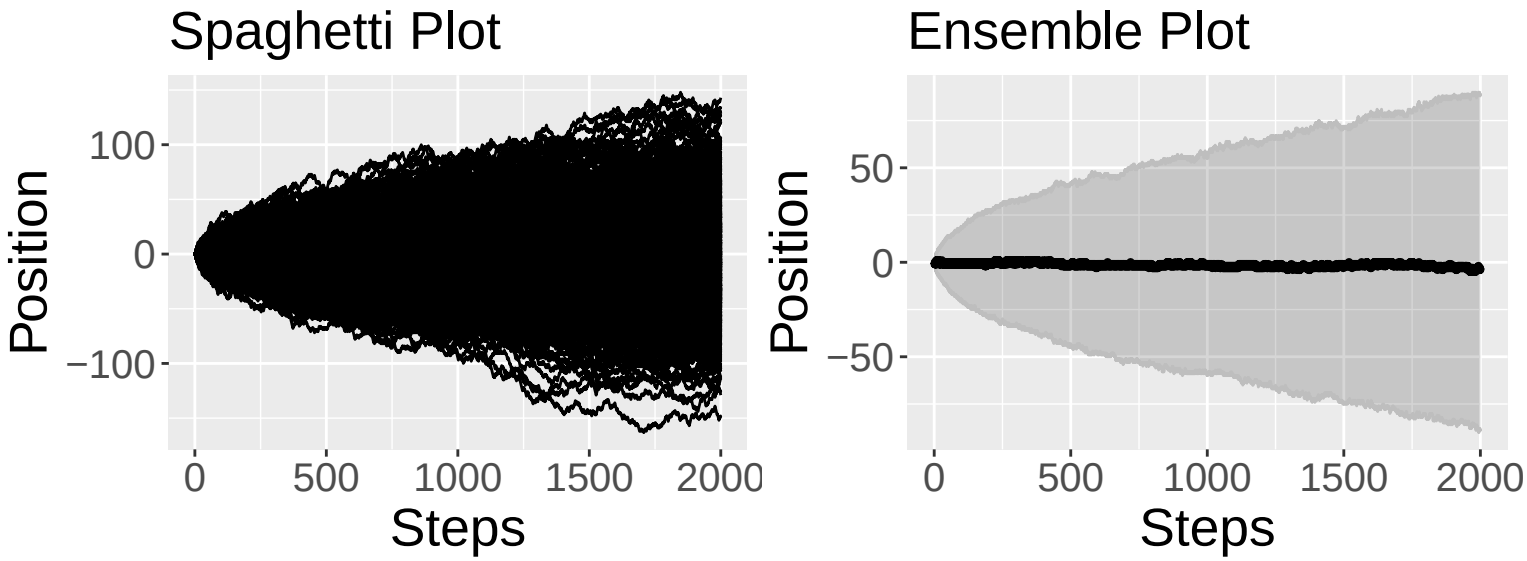
```
number_steps <- 200
number_realizations <- 100

random_walk(number_steps,number_realizations)
```



As the confidence interval increases as the number of steps increase, we get an interesting observation. These results suggest that on *average* you go nowhere (in other words, the average position is $x = 0$), but as the number of steps increase, you are very likely to be *somewhere* (in other words, the confidence interval increases as the number of steps increase). The more realizations we can do the more robust this pattern becomes. Figure @(fig:high-steps-random) shows the spaghetti and ensemble plots when the number of realizations is 1000:

```
random_walk(2000,1000)
```



Let's investigate our observations a little more mathematically.

1.3 Random walk mathematics

Call x^i the position x at step i in a random walk. While we have set this up to be a unit walk, more generally $x^i = x^{i-1} + r(p)\Delta x$, with Δx being the jump size (in this case $\Delta x = 1$) and $r(p)$ being a random variable:

$$r(p) = \begin{cases} -1 & 0 \leq p < 0.5 \\ 1 & 0.5 \leq p < 1 \end{cases} \tag{1.1}$$

Note that in the above equation p is drawn from a uniform distribution.

The equation $x^i = x^{i-1} + r(p)\Delta x$ is sometimes referred to as the *evolution equation*. If we have multiple simulations then x^i_j is the position at step i for simulation j.

Let's introduce some terminology to help us out here. $\langle x^i \rangle = \frac{1}{n}\sum_{j=1}^{n} x_j^i$ is the *expected value* of our position at step i, summed over all of the simulations at that timestep. Notice the connection here to the ensemble average at a given point.

You will learn in a probability theory class that the expected value is a linear operator. Because of this, we can compute the expected value of our evolution equation:

$$\langle x^i \rangle = \langle x^{i-1} + r(p)\ \Delta x \rangle = \langle x^{i-1} \rangle + \langle r(p)\ \Delta x \rangle = \langle x^{i-1} \rangle + \langle r(p) \rangle \Delta x \tag{1.2}$$

Let's focus in particular on the second term of this expression: $\langle r(p) \rangle$. Because $r(p)$ is a discrete random variable we can also compute its expected value as well. Let's write Equation (1.3) a little differently:

$$r_i = \begin{cases} -1 & p_1 = 0.5 \\ 1 & p_{12} = 0.5 \end{cases} \tag{1.3}$$

Written in this way, r has two possible outcomes: $r_1 = 1$ with probability $p_1 = 0.5$ or $r_2 = -1$ with probability $p_2 = 0.5$. Note that $p_1 + p_2 = 1$.

Now to calculate $\langle r(p) \rangle$ we adding up the possible outcomes for r (either 1 or -1) when multiplied by their associated probabilities (either 1/2 or 1/2 - note how these probabilities sum to 1):[1]

$$\langle r(p) \rangle = 1 \cdot \frac{1}{2} - 1 \cdot \frac{1}{2} = 0 \tag{1.4}$$

So, the expected value of the evolution equation is $\langle x^i \rangle = \langle x^{i-1} \rangle$. This may not seem helpful, but let's start thinking of this value from the beginning (that is when $i = 0$, or when we begin at $x = 0$).

We assume for all of the simulations begin at $x = 0$, so $\langle x^0 \rangle = 0$. Let's evolve to the next timestep. By what we found, then $\langle x^1 \rangle = \langle x^0 \rangle = 0$. Connect these together: $\langle x^1 \rangle = 0$ as well. What do you think happens at $\langle x^2 \rangle$? If you guessed 0, you are correct, because $\langle x^2 \rangle = \langle x^1 \rangle = \langle x^0 \rangle = 0$. In fact, this pattern continues so that $\langle x^i \rangle = \langle x^{i-1} \rangle = 0$.

What does this tell us? Between all of this notation, $\langle x^i \rangle$ is the expected value at each timestep, so we showed that *the expected value at any time* is zero. In other words, *random walk goes nowhere*!

1.3.1 Random walk variance

Now that we have characterized the expected value or the average displacement let's do the variance of this random walk. We calculate this by computing the mean square displacement, or $\langle (x^i)^2 \rangle$. First we compute the square displacement using the evolution equation and multiplying out:

$$(x^i)^2 = (x^{i-1} + r(p)\Delta x)^2 = (x^{i-1})^2 + 2x^{i-1}r(p)\Delta x + r(p)^2 \Delta x^2. \tag{1.5}$$

Next what we will compute the expectation $\langle (x^i)^2 \rangle$ of our multiplied expression term by term:

- $\langle (x^{i-1})^2 \rangle$: There is not much we can do with this term.
- $\langle 2x^{i-1}r(p) \rangle$: This term is zero:

$$= 2x^{i-1}\langle r(p) \rangle = 2x^{i-1} \cdot \frac{1}{2} - 2x^{i-1} \cdot \frac{1}{2} = 0 \tag{1.6}$$

- $\langle r(p)^2 \Delta x^2 \rangle$: Since $r(p)$ is either positive or negative 1 depending on p, it must be the case that $r(p)^2 = 1$ for any value of p. So in the expectation, the last term is just Δx^2.

So the end result for the variance is:

$$\langle (x^i)^2 \rangle = \langle (x^{i-1})^2) \rangle + (\Delta x)^2 \tag{1.7}$$

[1] Generally speaking for a discrete random variable x, $\langle x \rangle = \sum p_i \cdot x_i$, with $\sum p_i = 1$.

In order to understand this let's write out the first few terms of this recursive relationship:

$$\left\langle (x^0)^2 \right\rangle = 0 \tag{1.8}$$
$$\left\langle (x^1)^2 \right\rangle = \left\langle (x^0)^2 \right\rangle + (\Delta x)^2 = (\Delta x)^2 \tag{1.9}$$
$$\left\langle (x^2)^2 \right\rangle = \left\langle (x^1)^2 \right\rangle + (\Delta x)^2 = 2(\Delta x)^2 \tag{1.10}$$
$$\left\langle (x^3)^2 \right\rangle = \left\langle (x^2)^2 \right\rangle + (\Delta x)^2 = 3(\Delta x)^2 \tag{1.11}$$

There is a pattern here, in fact $\langle (x^i)^2 \rangle = i(\Delta x)^2$. So the variance, or the mean square displacement grows, proportional to the step size. Another way to state this is that the standard deviation (the square root of the variance) is equal to $\pm\sqrt{n}\ \Delta x$, where n is the current step. This matches up with our graphs from earlier since $\Delta x = 1$! Informally this tells us that on *average* you go nowhere, but *eventually* you travel everywhere - how cool!

1.4 Continuous random walks (diffusion)

One final thought can be made here. We are taking discrete steps but we can transform our results to a continuous time analog. Let $t = n\Delta t$ be the approximation from discrete time to continuous time. Equivalently $n = \dfrac{t}{\Delta t}$. With this information we can rearrange the square displacement equation to the following:

$$\left\langle (x^n)^2 \right\rangle = \frac{t}{\Delta t}(\Delta x)^2. \tag{1.12}$$

The quantity $D = \dfrac{(\Delta x)^2}{2\Delta t}$ is known as the *diffusion coefficent*. So then the mean square displacement can be arranged as $\langle (x^n)^2 \rangle = 2Dt$, confirming again that the variance grows proportional to t.

To connect this back to our discussion of stochastics, understanding a random walk helps us to understand how demographic and environmental stochasticity affect a dynamical system and the types of behaviors in the solution this random walk introduces to the system. In the following sections we will investigate the ways in which the random walk connects to stochastic differential equations.

1.5 Exercises

Exercise 1.1. In class we found that the diffusion coefficient is equal to $D = \frac{(\Delta x)^2}{2\Delta t}$.

a. Solve the expression for D in terms of Δt.

b. The diffusion coefficient for oxygen in water is approximately 10^{-5} cm^2 sec^{-1}. Use that value to complete the following table:

Distance (Δx)	1 μm = 10^{-6} m	10 μm	1 mm	1 cm	1 m
Diffusion time (Δt)	____ sec	____ sec	____ min	____ hours	____ years

c. Navigate to the following website, which lists sizes of different cells: https://en.wikibooks.org/wiki/Cell_Biology/Introduction/Cell_size. For what cells would diffusion be a reasonable process to transport materials?

Exercise 1.2. You are playing a casino game. If you win the game earn a dollar. If you lose the game you lose one dollar. The probability of winning or losing is 50-50 (0.50). You start the game with $100.

a. Write a one-dimensional random walk to simulate your money after playing the game 50 times. Make a few sample plots.

b. Based on the results of this section, what do you think your long-term expected winnings will be?

c. Now assume the house win probability is 0.52. Modify your random walk to simulate your money after playing the game 50 times. Make a few sample plots.

d. What do you think your long-term expected winnings of this modified game will be? (You may need to play the game for 100, 200 times to see a pattern.)

Exercise 1.3. Compute $\langle r \rangle$ for the following random variable:

$$r = \begin{cases} -1 & p_1 = 0.52 \\ 1 & p_2 = 0.48 \end{cases} \tag{1.13}$$

Exercise 1.4. Compute $\langle r \rangle$ for the following random variable, where $0 \leq q \leq 1$:

$$r = \begin{cases} -1 & p_1 = q \\ 1 & p_2 = (1-q) \end{cases} \tag{1.14}$$

Exercise 1.5. Consider the following random variable:

$$r = \begin{cases} -1 & p < 1/3 \\ 0 & 1/3 \leq p < 2/3 \\ 1 & 2/3 \leq p \end{cases} \tag{1.15}$$

a. Modify the code for the one dimensional random walk to generate a simulation of this random walk and plot your result. You can do this by applying a 'if' 'else' statement:

```
p <- runif(1)
if (p < 1/3) {x[i]<-x[i-1]-1}
else if (1/3 <= p & p < 2/3) {x[i] <- x[i-1]}
else {x[i] <- x[i-1]+1 }
```

b. Compute $\langle r \rangle = \int_0^1 r \, dp$ and $\langle r^2 \rangle = \int_0^1 r^2 \, dp$.

c. Based on your last answer, explain how this random variable introduces a different random walk than the one described in this section. In what ways do you tihnk this would change our calculations for the mean and variance of the ensemble simulations?

Exercise 1.6. In this exercise you will write code for a two dimensional random walk.

a. Modify the code for the one dimensional random walk to have both an x and a y position. One way to do this is to create a variable y structured similar to x, and to make a second 'if' statement in your for loop that moves 'y'.

b. Plot a few different realizations of your sample paths.

c. If we were to compute the mean and variance of the ensemble simulations, what do you think they would be?