Kotlin for Data Science

Thomas Nield
ChicagoMUG
April 17th, 2018

Thomas Nield

Business Consultant at Southwest Airlines

Author

Getting Started with SQL by O'Reilly

Learning RxJava by Packt

Trainer and content developer at O’Reilly Media

OSS Maintainer/Collaborator

RxKotlin

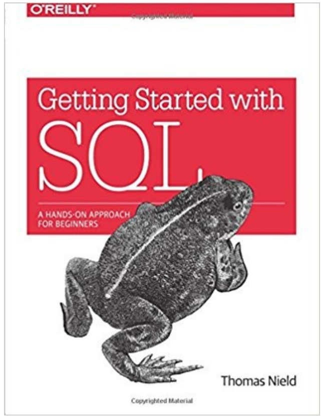
Kotlin-Statistics

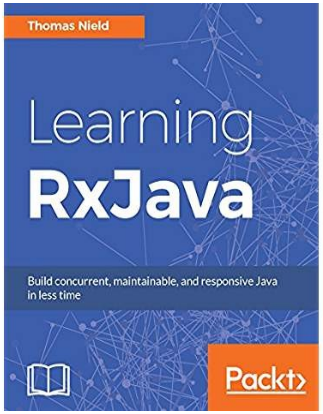
TornadoFX

RxKotlinFX

RxJavaFX

RxPy




Thomas Nield
Learning
RxJava
Build concurrent, maintainable, and responsive Java in less time
Packt>

thomasnield9727

https://github.com/thomasnield

Agenda

Anecdote – Monty Hall Problem

What Is Data Science?

Challenges in the Data Science Domain

Why Kotlin for Data Science?

Mathematical Modeling - Live Coding!

Getting Involved

The Monty Hall Problem

The Monty Hall Problem

DOOR 1

DOOR 2

DOOR 3

Choose a door, one has a prize.
Two others have goats.

The Monty Hall Problem

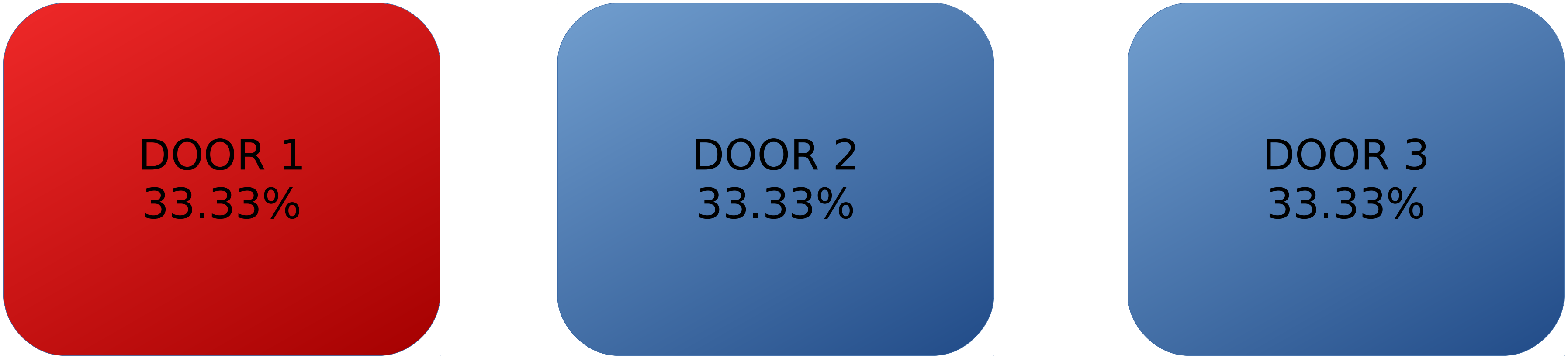
DOOR 1

DOOR 2

DOOR 3

You choose **Door 1**.
What is the probability it has the prize?

The Monty Hall Problem



DOOR 1
33.33%

DOOR 2
33.33%

DOOR 3
33.33%

You choose **Door 1**.
What is the probability it has the prize?

The Monty Hall Problem

DOOR 1

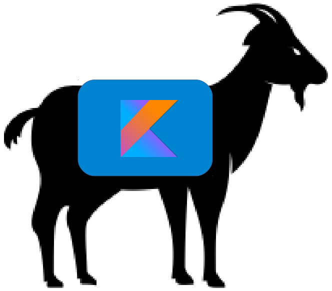
DOOR 2

Twist! **Door 3** was just opened. It’s a goat.
Did you want to switch from **Door 1** to **Door 2**?

The Monty Hall Problem

DOOR 1
?%

DOOR 2
?%



What is the prize probability of each door now?
Should you switch?

The Monty Hall Problem

DOOR 1
?%

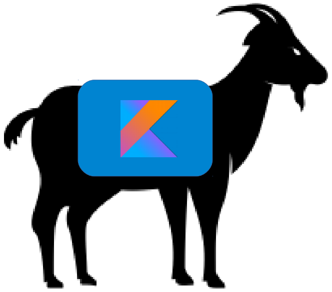
DOOR 2
?%

HINT: The prize probability of either door is not 50%

The Monty Hall Problem

DOOR 1
33.33%

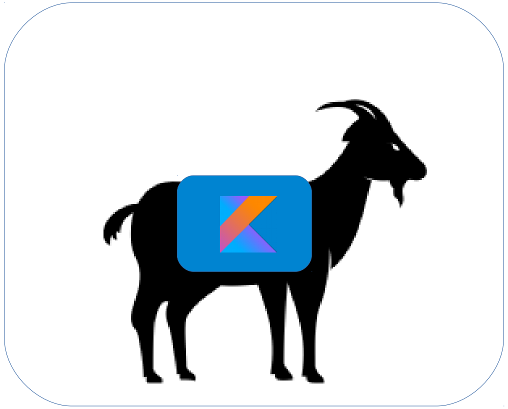
DOOR 2
66.66%



The probability of **Door 1** is 33.33% while **Door 2** is now 66.66%.
You should switch! But why?

The Monty Hall Problem

33.33%



66.66%

33.33%

The Monty Hall Problem

According to Bayes Theorem...

DOOR 1
33.33%

DOOR 2
66.66%

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(P_1|D_2) = \frac{P(D_2|P_1)P(P_1)}{P(D_2)} = \frac{(.5)(.33)}{(.5)} = .33$$

$$P(P_2|D_2) = \frac{P(D_2|P_2)P(P_2)}{P(D_2)} = \frac{(1)(.33)}{(.5)} = \boxed{.66}$$

The Monty Hall Problem

$$D_2 = \text{Probability of door 2 being left} = .5$$

$$P_1 = \text{Probability door 1 contains prize} = .33$$

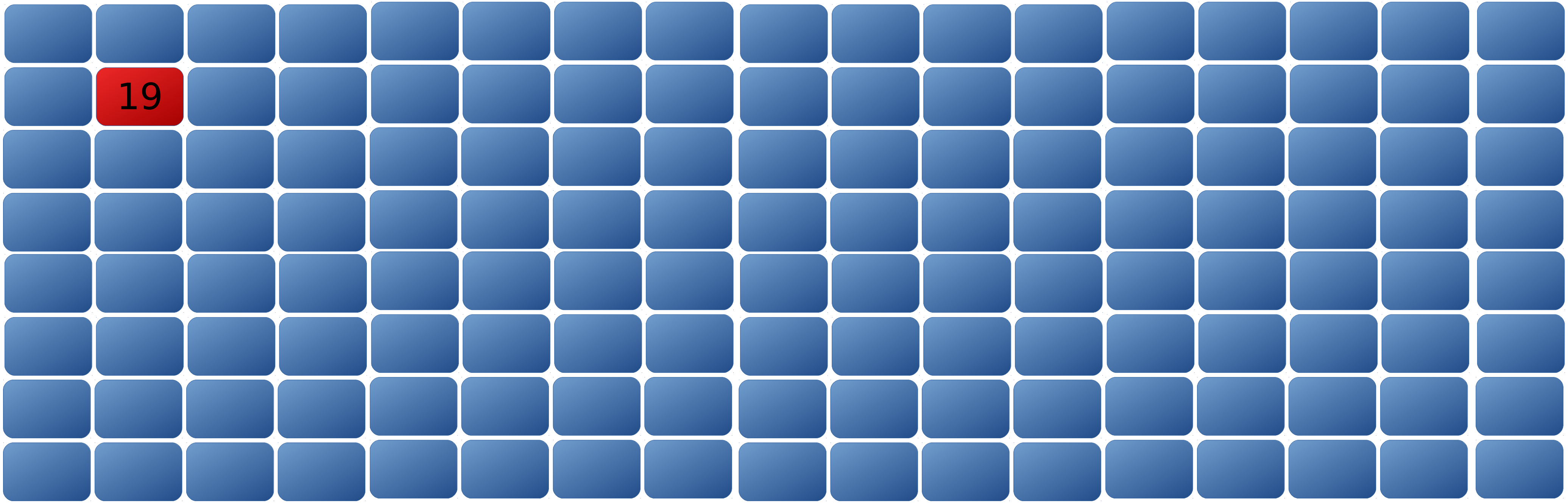
$$P_2 = \text{Probability door 2 contains prize} = .33$$

$$P(P_1|D_2) = \text{Probability door 1 contains prize given door 2 is left} = .33$$

$$P(P_2|D_2) = \text{Probability door 2 contains prize given door 2 is left} = .66$$

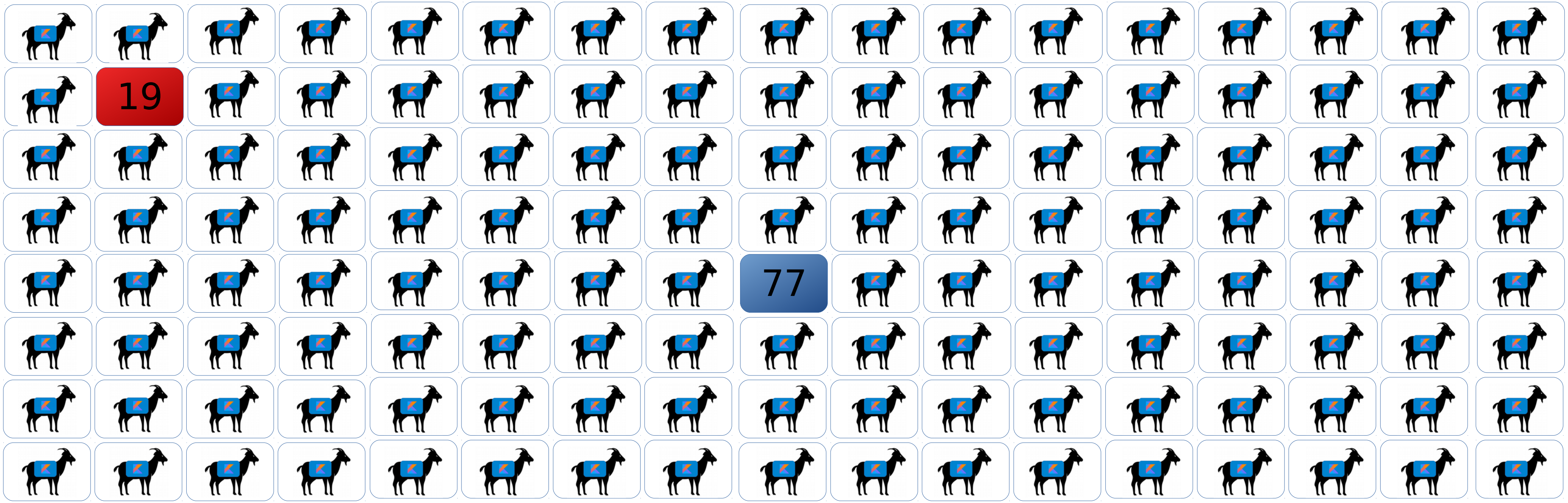
The Monty Hall Problem

Still confused? Hyperbolize! Imagine you had 1000 doors, and you chose **Door #19**.



The Monty Hall Problem

All other doors are opened but yours and **Door #77**. Inclined to switch now?

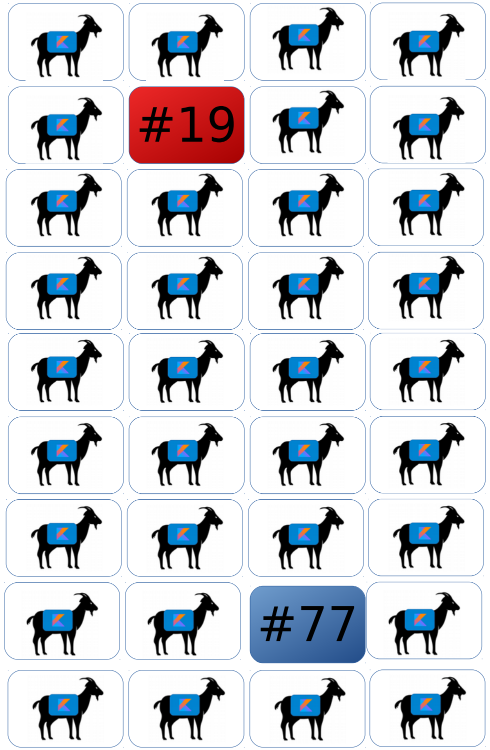


The Monty Hall Problem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(P_{19}|D_{77}) = \frac{P(D_{77}|P_{19})P(P_{19})}{P(D_{77})} = \frac{\frac{1}{999} * \frac{1}{1000}}{\frac{1}{1000}} = \frac{1}{999}$$

$$P(P_{77}|D_{77}) = \frac{P(D_{77}|P_{77})P(P_{77})}{P(D_{77})} = \frac{\frac{999}{1000} * \frac{1}{1000}}{\frac{1}{1000}} = \boxed{\frac{999}{1000}}$$



Yes, you should switch!

Learn More About Bayes Theorem

How Bayes Theorem Works – Brandon Rohrer

https://www.youtube.com/watch?v=5NMxiOGL39M

Why Am I Showing This?

The Monty Hall problem encapsulates the critical (and misunderstood) nuances of probability.

- Sometimes we only have incomplete data, but we need to make a decision anyway.
- When new partial data is available, we must merge (not abandon) the data we previously had.

As programmers, we thrive in certainty and exactness.

But the valuable, high-profile problems today often tackle uncertainty and approximation.

- Users and businesses expect “smarter” applications that will predict what they want.
- The future often needs to be forecasted and planned.
- Optimization programming is hard to solve and prove optimality.

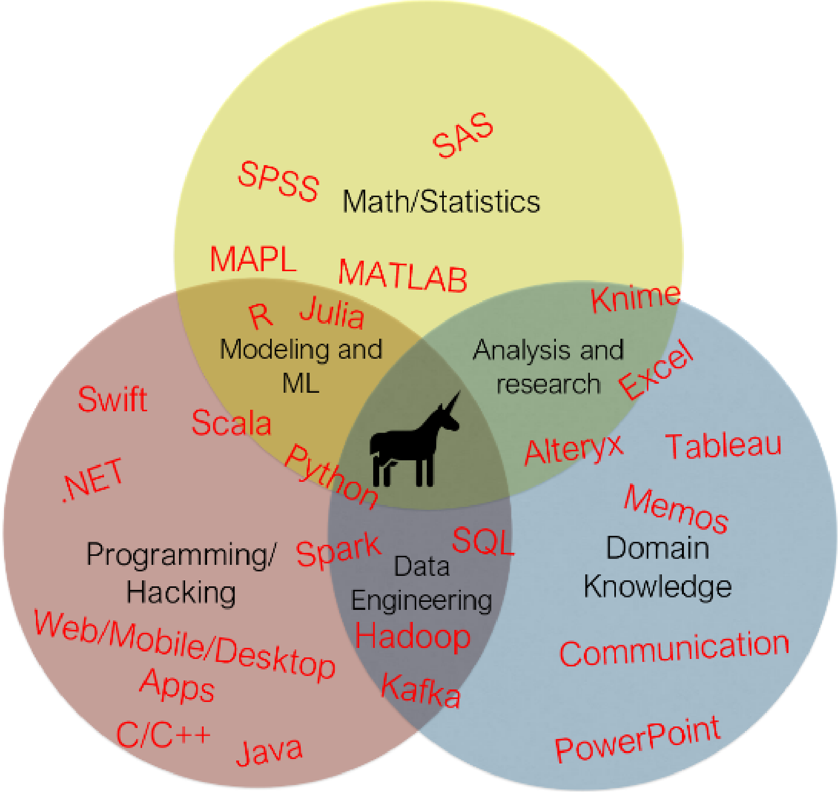
What Is Data Science?

What Is Data Science?

Data Science attempts to turn data into insight.

Insight can then be used to aid business decisions or create data-driven products.

A strong data science professional has some mix of programming, math/statistics, and business domain knowledge.



Data Scientist Archetypes

The Statistician

Summarizes data using classic statistical methods and probability metrics.

The Mathematician

Likes using the words “vector”, “matrix”, and “Bayes”.

The Data Engineer

Architect and wrangler of SQL, NoSQL, unstructured data, and “Big Data” pipelines, who makes data usable to rest of the team.

Data Scientist Archetypes

ML Engineer

A more advanced mathematician who specializes in machine learning algorithms/libraries.

The Programmer

A trained software developer who likely knows Java, Scala, or Python and often creates code from scratch tailored to specific business problems.

The Bard

The person who crafts communications about data findings to leaders and stakeholders, often telling stories with memos, charts, slides, infographics, spreadsheets, and other visual tools.

What is a Model?

What is a model?

A code representation of a problem, often mathematical in nature, that provides a solution in some form.

Examples:

A discrete optimization system that generates a train schedule maximizing the flow of passengers.

A k-means, decision tree, linear regression, or support vector machine model that groups customers based on their attributes.

Natural language processor that parses, interprets, and links legal documents.

Neural network that identifies or categorizes images or speech.

Data Science Challenges

A Model is Not A Product!

A current struggle in the data science domain is putting models in production

A model is often a hacky Python or R script that simply does not plug into a large enterprise ecosystem (which is often built on Java or .NET)

Models often use dynamically typed languages with tabular data structures and procedural code which is difficult to modularize, test, evolve, and refactor.

A model may have never been tested in a production environment, which may have drastically different dynamics and data patterns.

Twitter

There was only one problem—all of my work was done in my local machine in R. People appreciate my efforts but they don’t know how to consume my model because it was not “productionized” and the infrastructure cannot talk to my local model. Hard lesson learned!

- Robert Chang, Data Scientist at Airbnb (formerly Twitter)

Stitch Fix

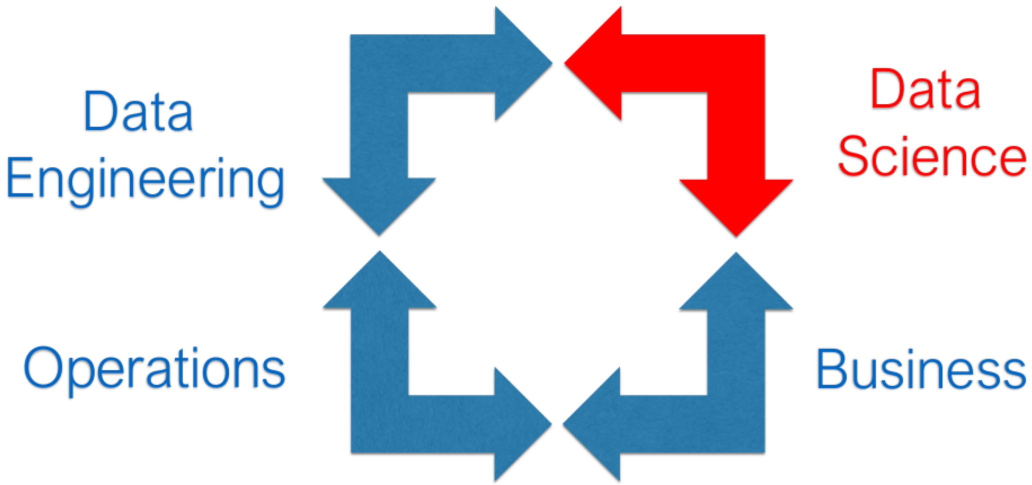
Data scientists are often frustrated that engineers are slow to put their ideas into production and that work cycles, road maps, and motivations are not aligned. By the time version 1 of their ideas are put into [production], they already have versions 2 and 3 queued up. Their frustration is completely justified.

- Jeff Magnusson, Director of Data at Stitch Fix

Slack

“The infinite loop of sadness.”

- Josh Wills, Director of Data Engineering



So What’s Wrong with Python and R?

Nothing is wrong with Python and R, they just are harder to use in a production capacity.

- Data science is often about “prototyping” and “proving” concepts, and the problem of putting models in production is still new.

- Lack of static typing, immutability, modern language features, concurrency, and design principles widen the gap between data science and software engineering as well.

The critical value of Python and R are in their libraries implemented with C, and the language is just the “glue”.

A production solution needs to prepare and feed live data into models and ingest the results.

Why Kotlin for Data Science?

Why Not Use Java For Data Science?

Java is a robust, production-grade platform with decent data science libraries.

Software engineers often use it to “productionize” data science models.

So why hasn’t Java gotten mainstream adoption in data science?

- Verbose, hard to script quickly with.
- Boilerplate makes it hard to experiment.
- Challenging language compared to Python, R.

Why Not Use Scala For Data Science?

Scala has been positioned into the data science domain with moderate success, especially on the “Big Data” front (e.g. Apache Spark).

So why hasn’t Scala taken significant share from Python?

- Python is simple, Scala is complex.
- Scala’s plethora of features-- good or overwhelming?
- Numerical libraries are increasingly going back to C, not JVM.

What About Kotlin? - YES!

Data scientists who code often need the following:

Rapid turnaround, quick iterative development

Easy to learn, flexible code language

Mathematical and machine learning libraries

Experienced software engineers often want the following:

Static typing and object-oriented code

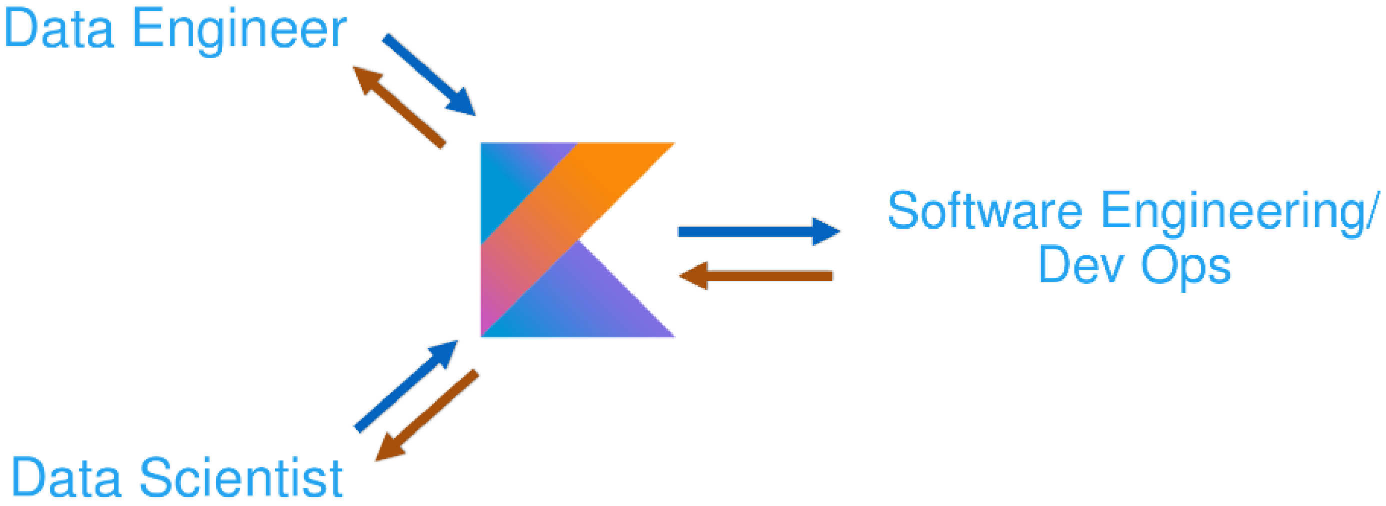
Production-grade architecture and support

Refactorability, reusability, concurrency, and scaling

Compatibility with established systems (Java codebases)

Kotlin encompasses all the qualities above.

One language, one codebase, one platform



Kotlin Strengths and Weaknesses

Strengths

Statically typed, reliable and refactorable

Accessible, inuitive coding language

Minimal boilerplate, fast turnaround

Interoperability with Java

Pragmatic features – data classes, DSL, nullable types, robust higher order functions

Weaknesses

Not dynamically typed – data structures have to be explicitly defined

Numerical efficiency – boxing hurtful for computations?

Libraries – Decent JVM data science libraries, but not comprehensive as Python or R

Java/Kotlin Data Science Libraries

Java/Kotlin Library	Python Equivalent	Description
ND4J	NumPy	Numerical computation Java library
DeepLearing4J	TensorFlow	Deep learning Java library
SMILE	scikit-learn	Comprehensive machine learning suite for Java
ojAlgo!	PuLP, NumPy	Linear algebra and optimization library for Java
Apache Commons Math	scikit-learn	Math, statistics, and ML for Java
TableSaw / Krangl	Pandas	Data frame libraries for Java/Kotlin
Kotlin-Statistics	scikit-learn	Statistical/probability operators for Kotlin
JavaFX / Vegas	matplotlib	Charting libraries

Mathematical Modeling with Kotlin

What is Mathematical Modeling?

Mathematical modeling is a broad discipline that attempts to solve real-world problems using mathematical concepts.

Applications range broadly, from *biology* and *medicine* to *engineering, business,* and *economics*.

Mathematical Modeling was rebranded under “data science” as data availability increased.

There is no “silver bullet” artificial intelligence, but rather the right technique for the right type of problem.

Real-World Examples

Product recommendations

Dynamic pricing

Sport event planning

Staff/resource scheduling

Inventory forecaster

Kidney exchanges

Airline network optimizer

DNA sequencing

Disaster Management

Why Learn Mathematical Modeling?

Technologies, frameworks, and languages come and go… but math never changes.

Classic mathematical techniques have survived decades (even centuries) and are not prone to obsolescence and deprecation.

Knowing mathematical modeling will greatly improve your professional marketability for decades to come, without constant investment in learning new technologies.

Work smarter, not harder

Many challenges, like staff scheduling, are impractical to solve with brute-force programming approaches.

Problems dealing with uncertainty (e.g. forecasting, predicting) are impossible without applied statistics.

Mathematical modeling allows you to gracefully find solutions to many categories of problems.

Mathematical Modeling Part I: Discrete Optimization

What Is Discrete Optimization?

Discrete optimization is a space of algorithms that tries to find a feasible or optimal solution to a constrained problem.

- Solving a Sudoku
- Creating a schedule for classrooms, staff, transportation, or manufacturing
- Maximizing fulfilled orders with constrained factory capacity
- Minimizing cost of materials for a product

Discrete optimization is a mixed bag of algorithms and techniques, which can often be abstracted away with an optimization solver library.

- Linear, integer, mixed programming
- Branch-and-bound
- Dynamic programming
- Metaheuristic approaches

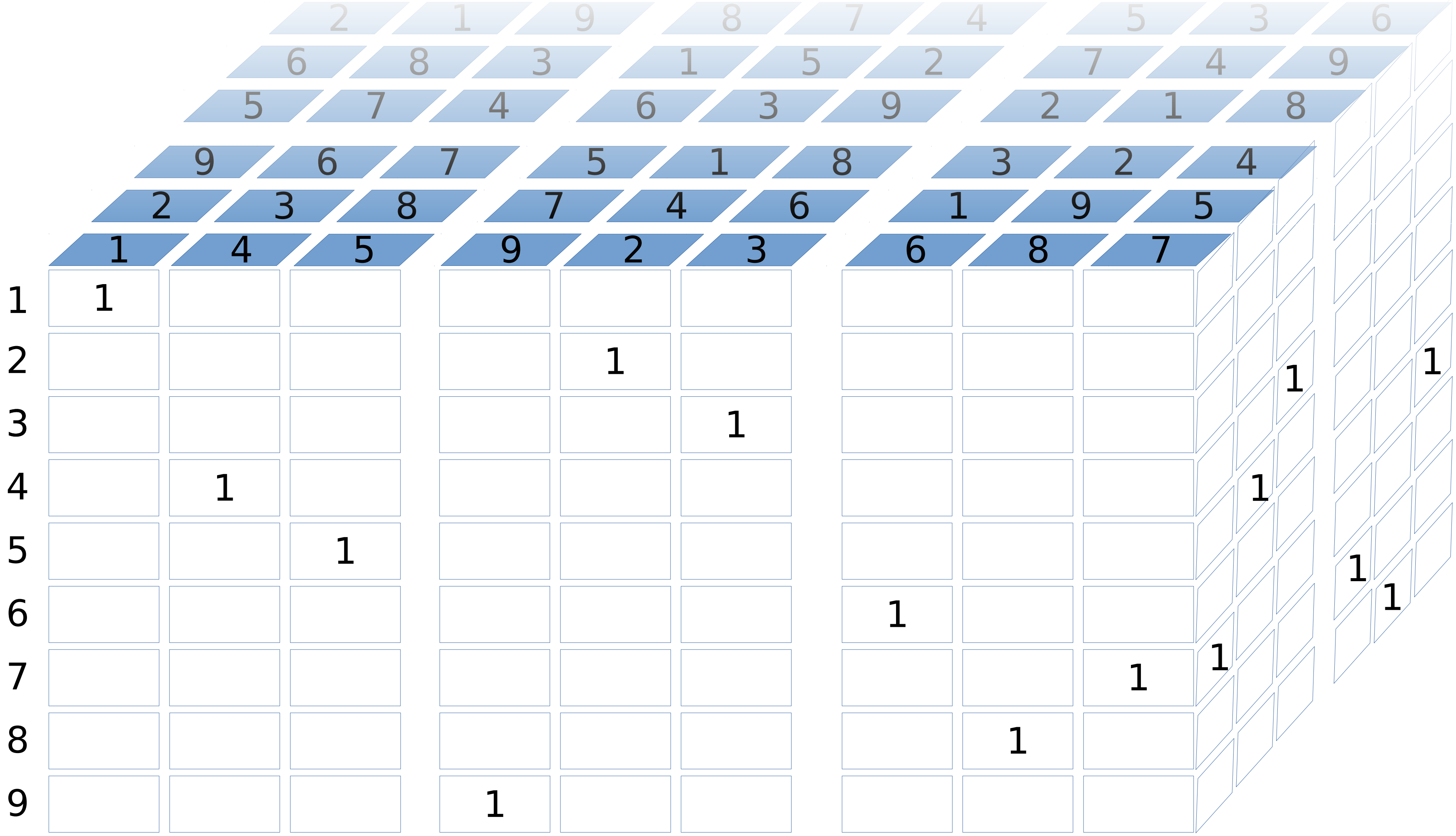
Solving a Sudoku

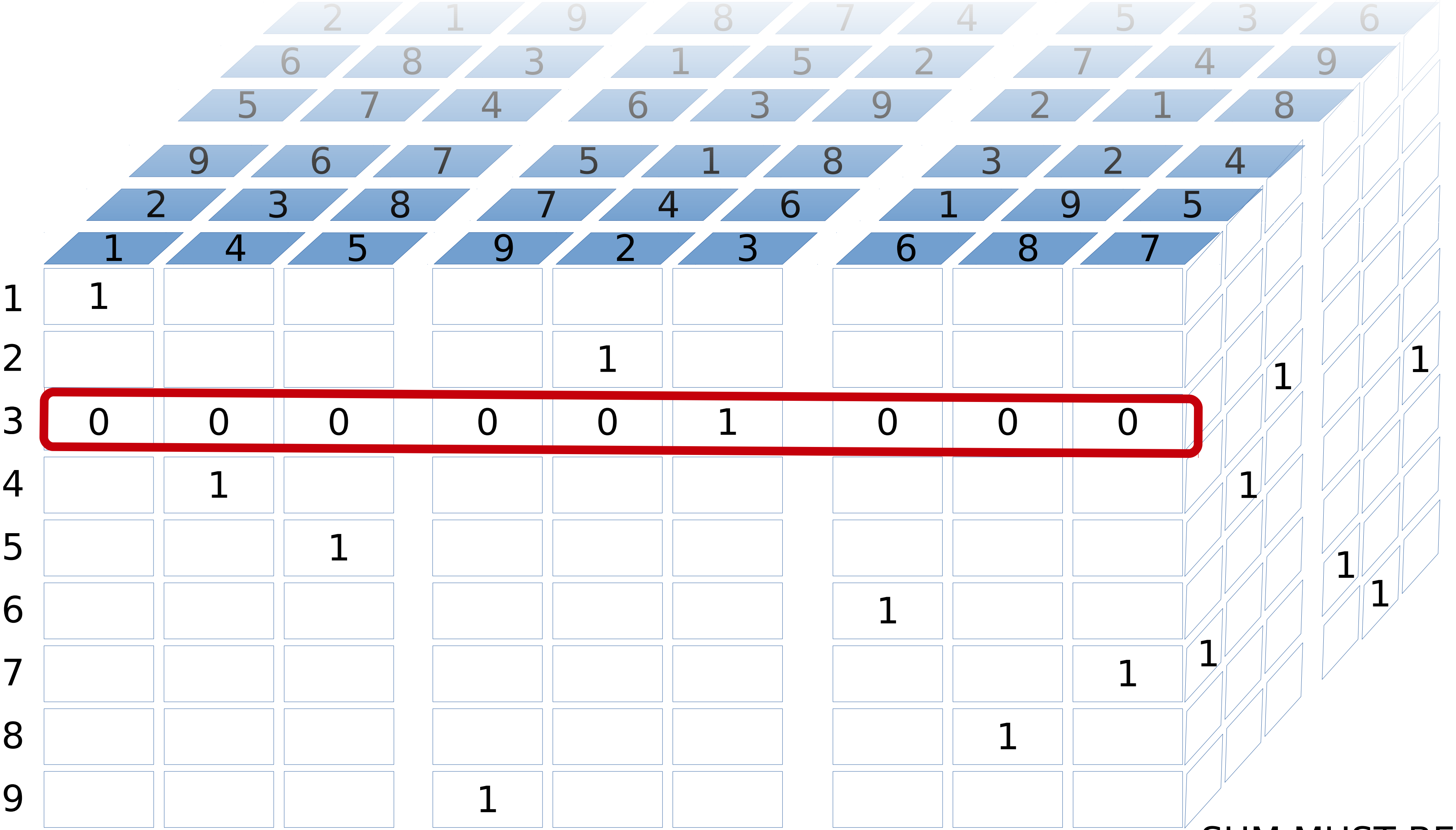
To solve a Sudoku, imagine each cell contains 9 variables corresponding for possible values 1 through 9.

Each variable is binary and can only be 1 or 0 indicating whether that number is used for that cell.

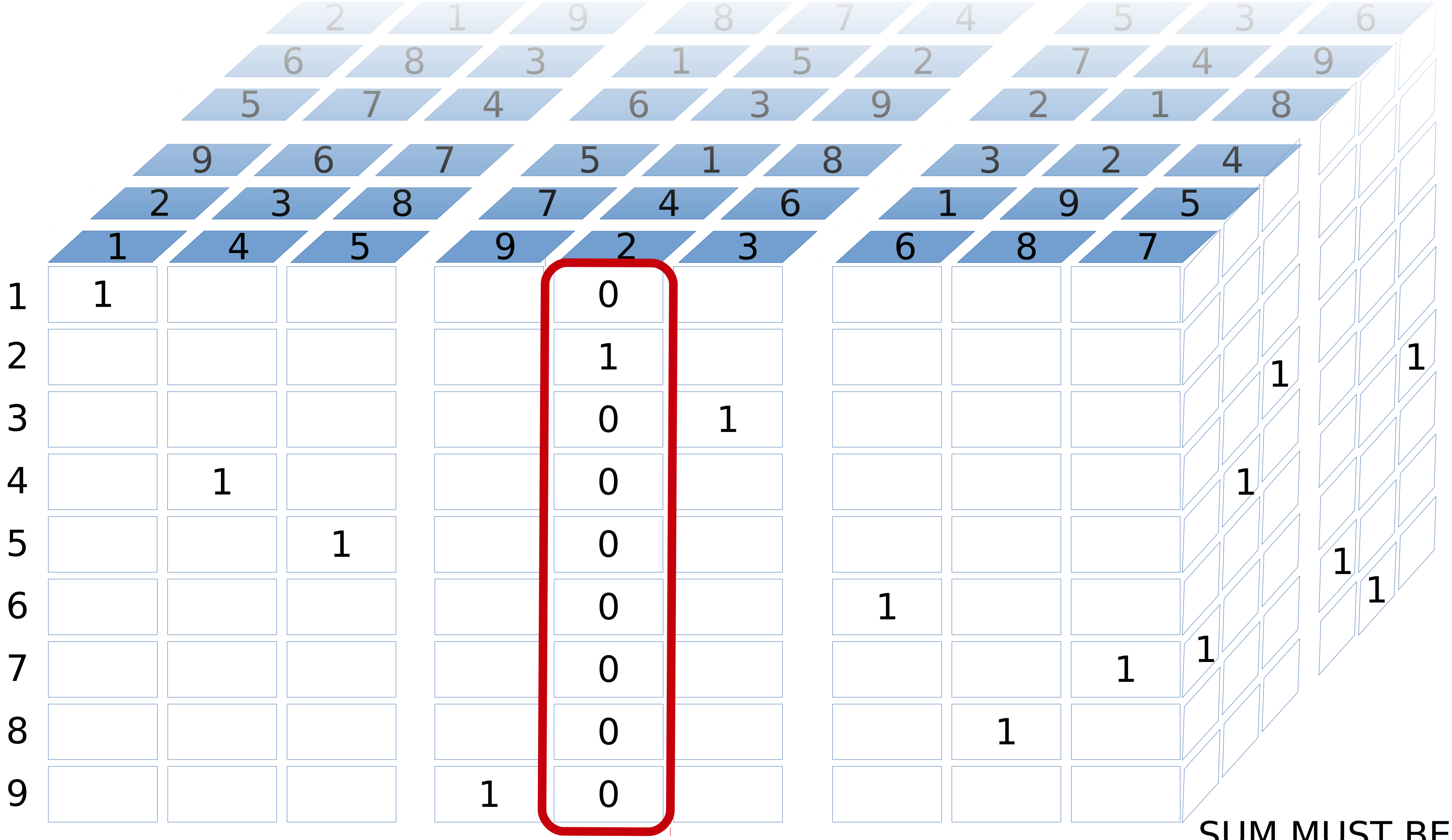
Constraints to solve a Sudoku:

- Each cell's 9 variables can only sum to 1.
- Variables for a given number (1-9) must sum to 1 for a given row, column, or square

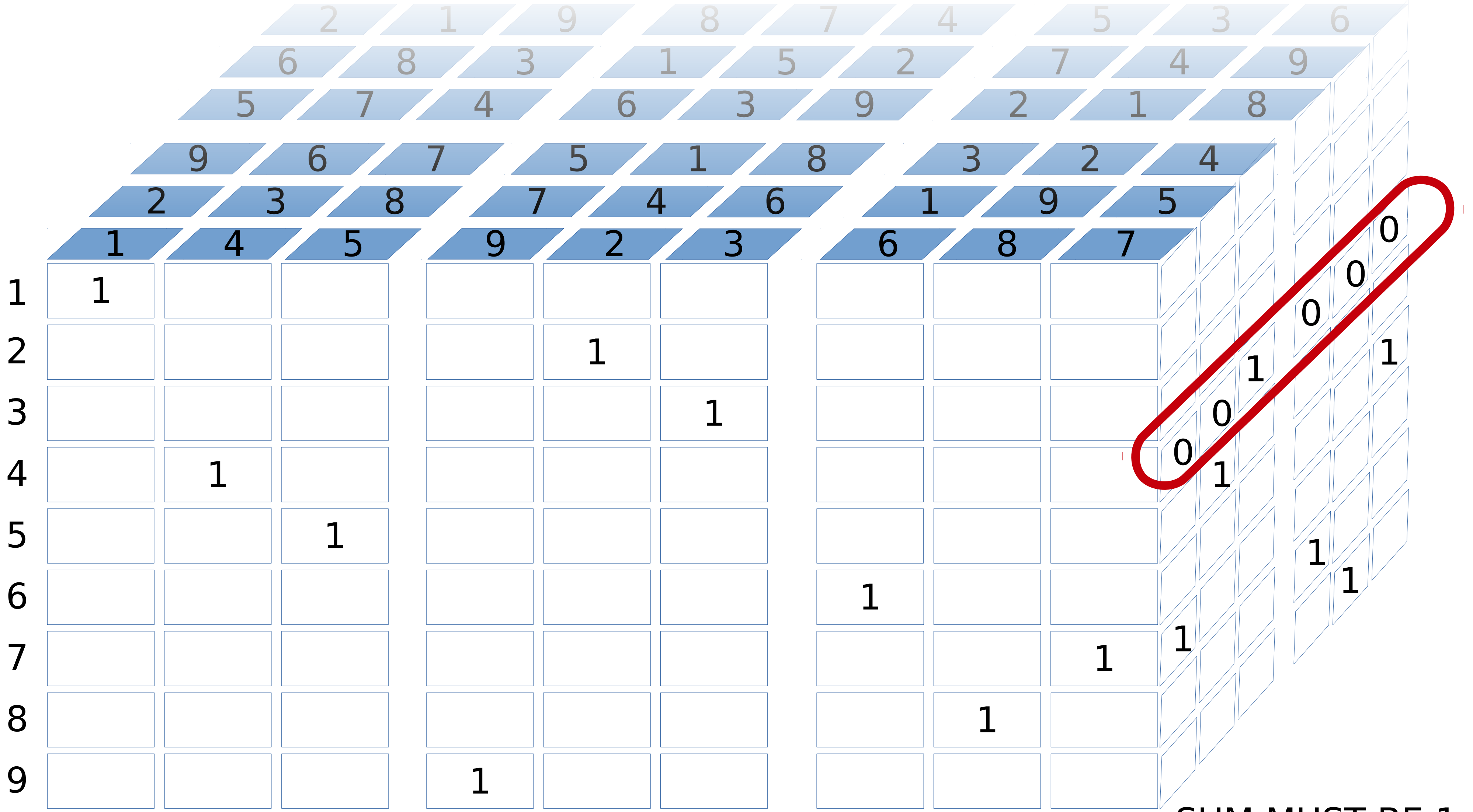

2 1 9 8 7 4 5 3 6
6 8 3 1 5 2 7 4 9
5 7 4 6 3 9 2 1 8
9 6 7 5 1 8 3 2 4
2 3 8 7 4 6 1 9 5
1 4 5 9 2 3 6 8 7
1
2
3
4
5
6
7
8
9
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1


2 1 9 8 7 4 5 3 6
6 8 3 1 5 2 7 4 9
5 7 4 6 3 9 2 1 8
9 6 7 5 1 8 3 2 4
2 3 8 7 4 6 1 9 5
1 4 5 9 2 3 6 8 7
1
2
3
4
5
6
7
8
9
1
1
0 0 0 0 0 1 0 0 0
1
1
1
1
1
1
1
1
1
1
1
1

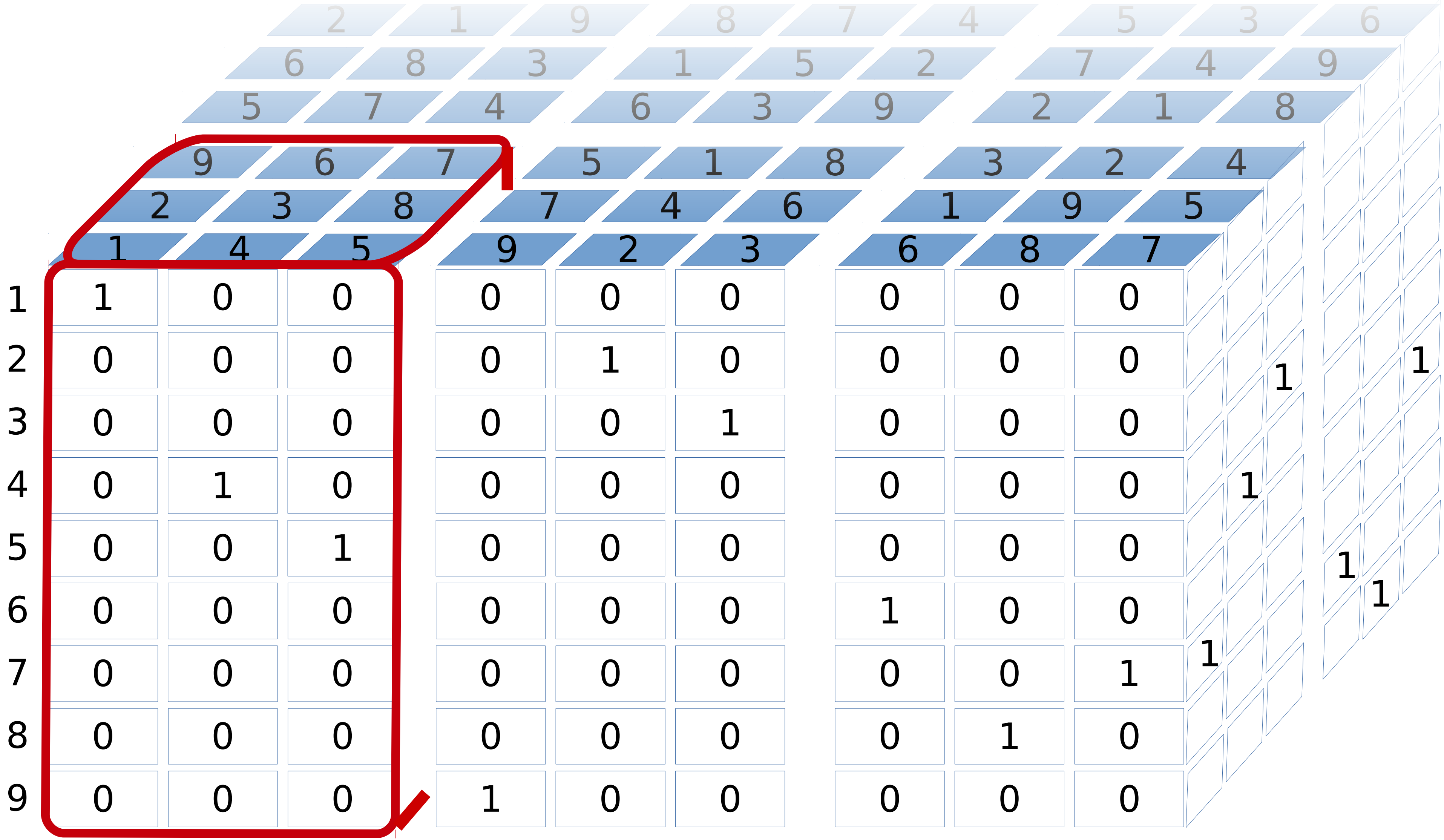
SUM MUST BE 1


2 1 9 8 7 4 5 3 6
6 8 3 1 5 2 7 4 9
5 7 4 6 3 9 2 1 8
9 6 7 5 1 8 3 2 4
2 3 8 7 4 6 1 9 5
1 4 5 9 2 3 6 8 7
1
2
3
4
5
6
7
8
9
1
0
1
0
1
0
1
0
1
0
0
1
0
1
0
1
0
1
1
1
1
1
1
1
1

SUM MUST BE 1


2 1 9 8 7 4 5 3 6
6 8 3 1 5 2 7 4 9
5 7 4 6 3 9 2 1 8
9 6 7 5 1 8 3 2 4
2 3 8 7 4 6 1 9 5
1 4 5 9 2 3 6 8 7
1
2
3
4
5
6
7
8
9
1
1
1
1
1
1
1
1
1
0
0
0
0
0
0
1
1
1
1
1
1

SUM MUST BE 1



	1	4	5	9	2	3	6	8	7
1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	1	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	1	0
9	0	0	0	1	0	0	0	0	0

SUM FOR EACH NUMBER MUST BE 1

Sudoku Example

We could create a solver from scratch, but a solver library like ojAlgo or OptaPlanner will make life easier.

- We just model our problem and define the constraints.
- The solver library will find the values for the variables.

The Sudoku model is a great example of binary variables constraining occupation to one resource.

- Only one “number” can occupy a square.
- This concept can apply to real-life, such as only one class can occupy a classroom at a given time.

Generating a Schedule

You need to generate a schedule for a single classroom with the following classes:

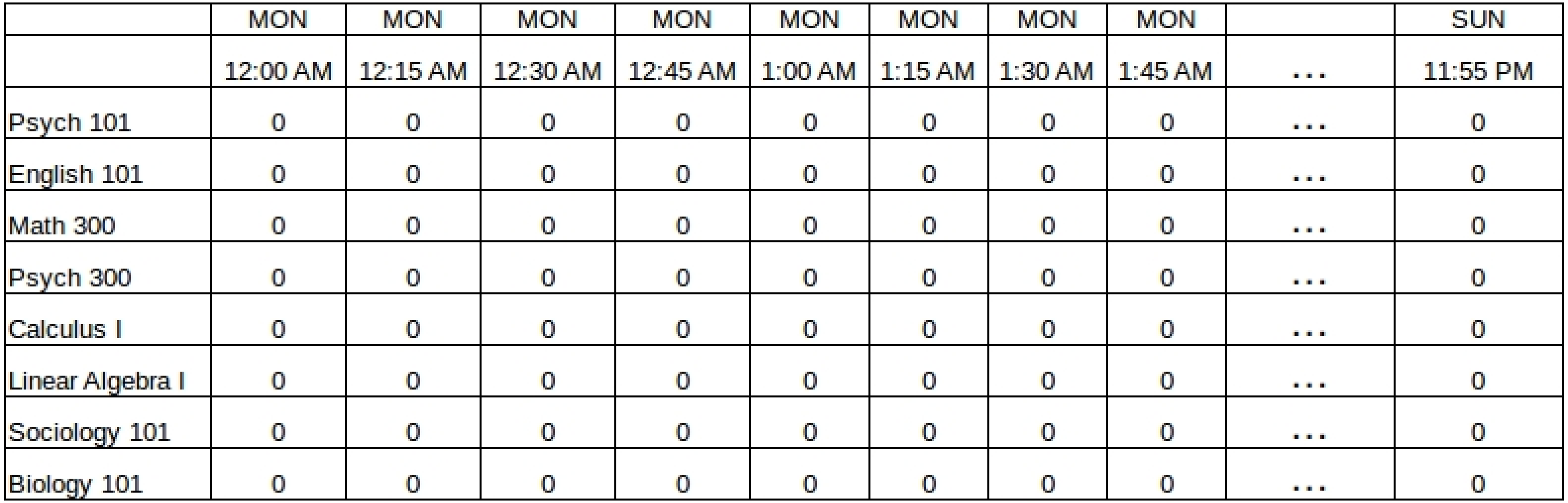
Psych 101 (1 hour, 2 sessions/week)
English 101 (1.5 hours, 2 sessions/week)
Math 300 (1.5 hours, 2 sessions/week)
Psych 300 (3 hours, 1 session/week)
Calculus I (2 hours, 2 sessions/week)
Linear Algebra I (2 hours, 3 sessions/week)
Sociology 101 (1 hour, 2 sessions/week)
Biology 101 (1 hour, 2 sessions/week)

Available scheduling times are **Monday through Friday**, **8:00AM-11:30AM**, **1:00PM-5:00PM**
Slots are scheduled in **15 minute increments**.

Generating a Schedule

Visualize a grid of each 15-minute increment from Monday through Sunday, intersected with each possible class.

Each cell will be a 1 or 0 indicating whether that's the start of the first class.



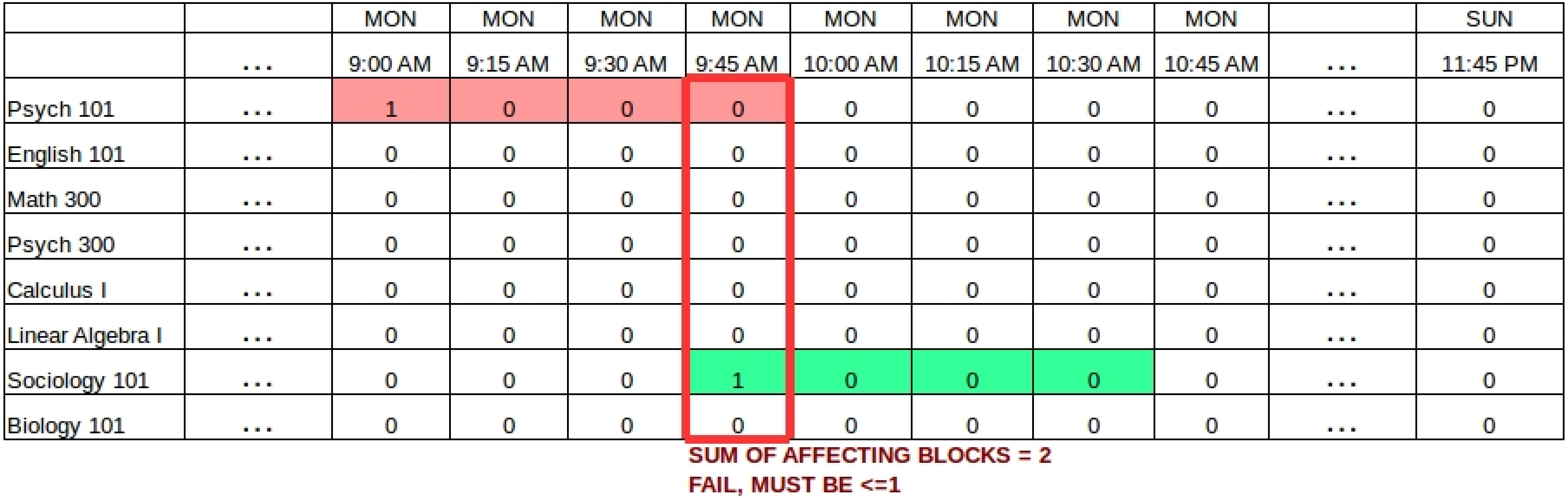
	MON	MON	MON	MON	MON	MON	MON	MON		SUN
	12:00 AM	12:15 AM	12:30 AM	12:45 AM	1:00 AM	1:15 AM	1:30 AM	1:45 AM	...	11:55 PM
Psych 101	0	0	0	0	0	0	0	0	...	0
English 101	0	0	0	0	0	0	0	0	...	0
Math 300	0	0	0	0	0	0	0	0	...	0
Psych 300	0	0	0	0	0	0	0	0	...	0
Calculus I	0	0	0	0	0	0	0	0	...	0
Linear Algebra I	0	0	0	0	0	0	0	0	...	0
Sociology 101	0	0	0	0	0	0	0	0	...	0
Biology 101	0	0	0	0	0	0	0	0	...	0

Generating a Schedule

Next visualize how overlaps will occur.

Notice how a 9:00AM Psych 101 class will clash with a 9:45AM Sociology 101.

We can sum all blocks that affect the 9:45AM block and ensure they don’t exceed 1.

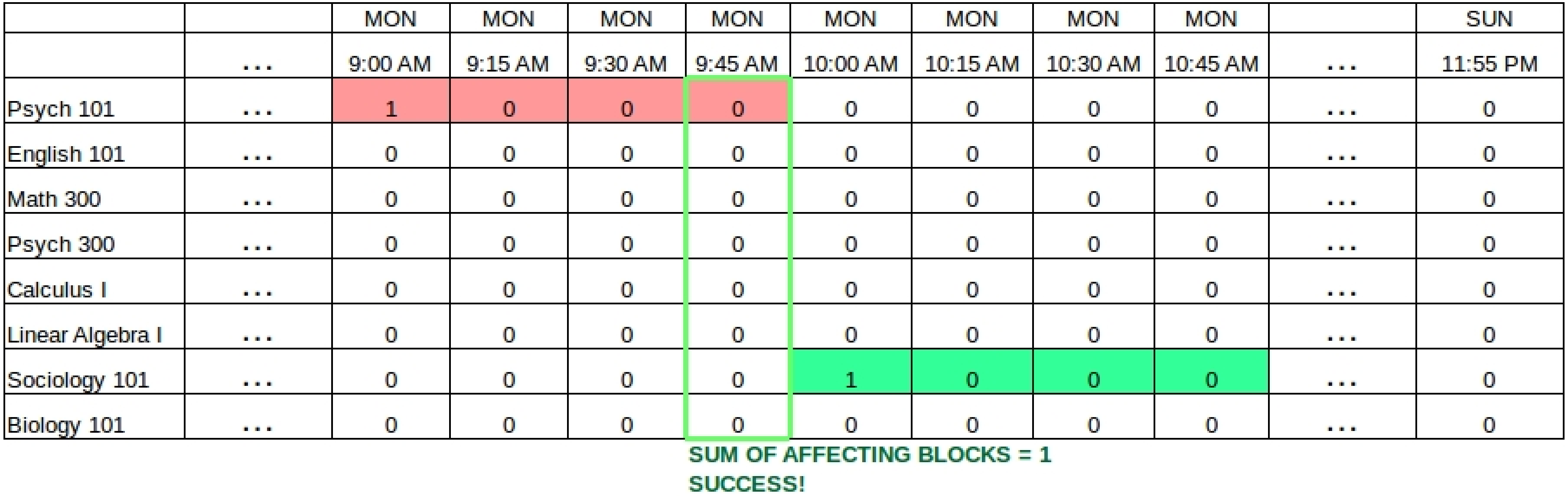


		MON	MON	MON	MON	MON	MON	MON	MON		SUN
	...	9:00 AM	9:15 AM	9:30 AM	9:45 AM	10:00 AM	10:15 AM	10:30 AM	10:45 AM	...	11:45 PM
Psych 101	...	1	0	0	0	0	0	0	0	...	0
English 101	...	0	0	0	0	0	0	0	0	...	0
Math 300	...	0	0	0	0	0	0	0	0	...	0
Psych 300	...	0	0	0	0	0	0	0	0	...	0
Calculus I	...	0	0	0	0	0	0	0	0	...	0
Linear Algebra I	...	0	0	0	0	0	0	0	0	...	0
Sociology 101	...	0	0	0	1	0	0	0	0	...	0
Biology 101	...	0	0	0	0	0	0	0	0	...	0

SUM OF AFFECTING BLOCKS = 2
FAIL, MUST BE <=1

Generating a Schedule

If the “sum” of all slots affecting a given block are no more than 1, then we have no conflicts!



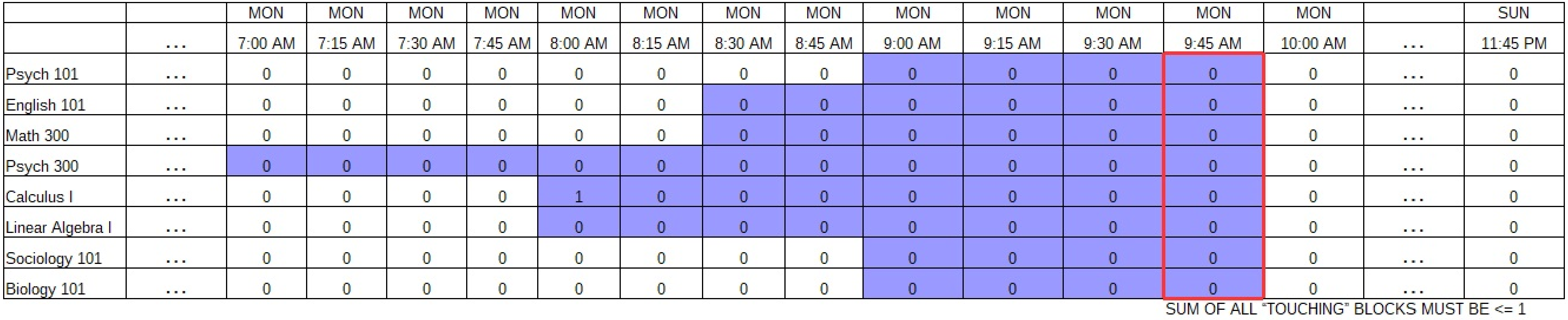
		MON	MON	MON	MON	MON	MON	MON	MON		SUN
	...	9:00 AM	9:15 AM	9:30 AM	9:45 AM	10:00 AM	10:15 AM	10:30 AM	10:45 AM	...	11:55 PM
Psych 101	...	1	0	0	0	0	0	0	0	...	0
English 101	...	0	0	0	0	0	0	0	0	...	0
Math 300	...	0	0	0	0	0	0	0	0	...	0
Psych 300	...	0	0	0	0	0	0	0	0	...	0
Calculus I	...	0	0	0	0	0	0	0	0	...	0
Linear Algebra I	...	0	0	0	0	0	0	0	0	...	0
Sociology 101	...	0	0	0	0	1	0	0	0	...	0
Biology 101	...	0	0	0	0	0	0	0	0	...	0

SUM OF AFFECTING BLOCKS = 1
SUCCESS!

Generating a Schedule

For every “block”, we must sum all affecting slots which can be identified from the class durations.

This sum must be no more than 1.



		MON	MON	MON	MON	MON	MON	MON	MON	MON	MON	MON	MON	MON		SUN
	...	7:00 AM	7:15 AM	7:30 AM	7:45 AM	8:00 AM	8:15 AM	8:30 AM	8:45 AM	9:00 AM	9:15 AM	9:30 AM	9:45 AM	10:00 AM	...	11:45 PM
Psych 101	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
English 101	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Math 300	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Psych 300	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Calculus I	...	0	0	0	0	1	0	0	0	0	0	0	0	0	...	0
Linear Algebra I	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Sociology 101	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Biology 101	...	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0

SUM OF ALL “TOUCHING” BLOCKS MUST BE <= 1

Generating a Schedule

Taking this concept even further, we can account for all recurrences.

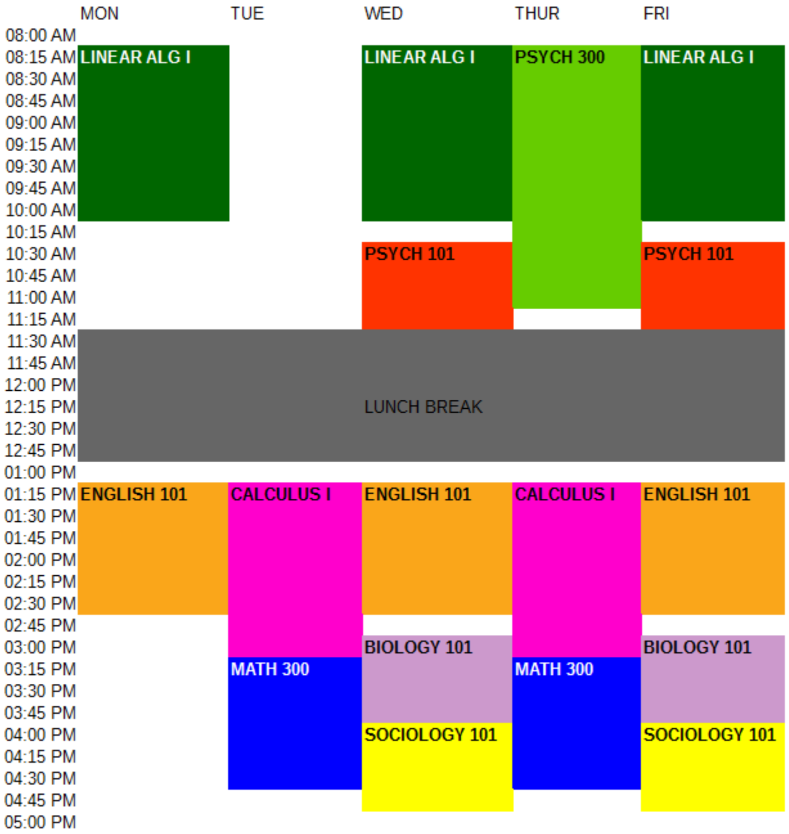
The “affected slots” for a given block can query for all recurrences for each given class.

View image here.

Generating a Schedule

Plug these variables and constraints into the optimizer, and you will get this possible solution.

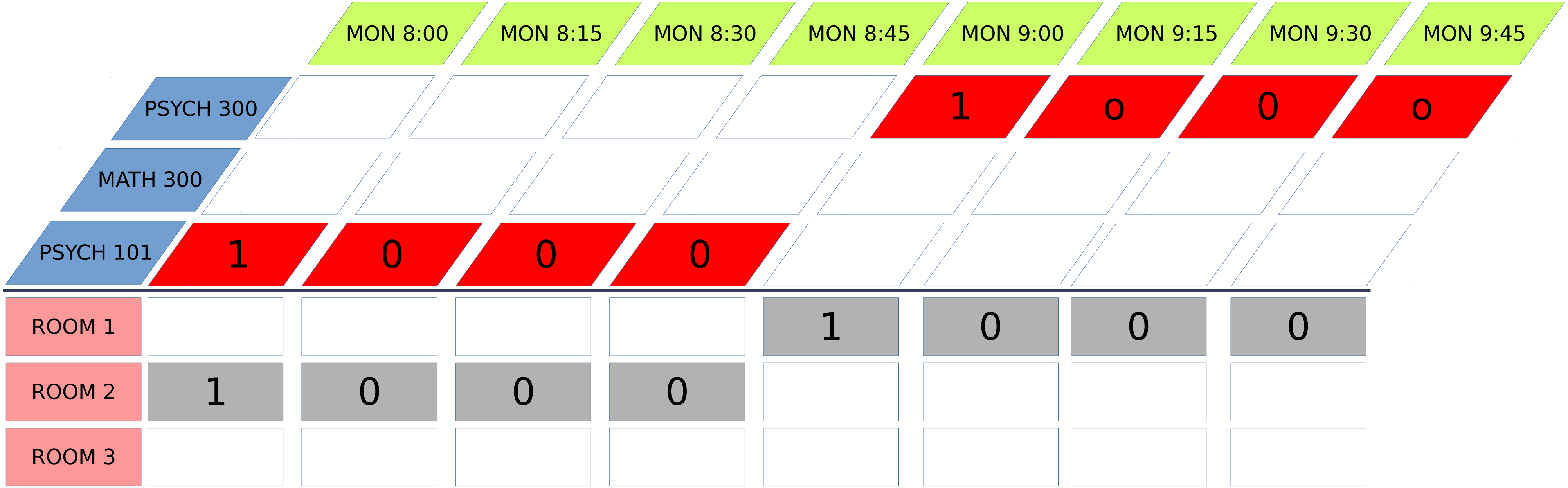
Most of the work will be finding the affecting slots for each block.



	MON	TUE	WED	THUR	FRI
08:00 AM					
08:15 AM	LINEAR ALG I		LINEAR ALG I	PSYCH 300	LINEAR ALG I
08:30 AM					
08:45 AM					
09:00 AM					
09:15 AM					
09:30 AM					
09:45 AM					
10:00 AM					
10:15 AM					
10:30 AM			PSYCH 101		PSYCH 101
10:45 AM					
11:00 AM					
11:15 AM					
11:30 AM					
11:45 AM					
12:00 PM					
12:15 PM			LUNCH BREAK		
12:30 PM					
12:45 PM					
01:00 PM					
01:15 PM	ENGLISH 101	CALCULUS I	ENGLISH 101	CALCULUS I	ENGLISH 101
01:30 PM					
01:45 PM					
02:00 PM					
02:15 PM					
02:30 PM					
02:45 PM					
03:00 PM			BIOLOGY 101		BIOLOGY 101
03:15 PM		MATH 300		MATH 300	
03:30 PM					
03:45 PM					
04:00 PM			SOCIOLOGY 101		SOCIOLOGY 101
04:15 PM					
04:30 PM					
04:45 PM					
05:00 PM					

Generating a Schedule

If you want to schedule against multiple rooms, plot each variable using three dimensions.



	MON 8:00	MON 8:15	MON 8:30	MON 8:45	MON 9:00	MON 9:15	MON 9:30	MON 9:45
PSYCH 300					1	o	0	o
MATH 300								
PSYCH 101	1	0	0	0				

ROOM 1					1	0	0	0
ROOM 2	1	0	0	0				
ROOM 3								

Continuous Labor Shifts

You have three drivers who charge the following rates:

Driver 1: $10 / hr

Driver 2: $12 / hr

Driver 3: $15 / hr

From 6:00 to 22:00, schedule one driver at a time to provide coverage, and minimize cost.

Each driver must work 4-6 hours a day. Driver 2 cannot work after 11:00.

Stay Calm

Variables and Constants

S_i = Shift start time of each i driver

E_i = Shift end time for each i driver

R_i = Hourly rate for each i driver

δ_{ij} = Binary (1,0) between two ij drivers

M = Length of planning window

Minimize

$$\sum_{i=1}^{3} R_i(E_i - S_i)$$

Constraints

$$4 <= E_i - S_i <= 6$$

$$16 = \sum_{i=1}^{3} E_i - S_i$$

$$E_2 <= 11$$

$$S_i >= E_j - M\delta_{ij}$$

$$S_j >= E_i - M(1 - \delta_{ij})$$

Discrete Optimization Summary

Discrete Optimization is a best-kept secret often overlooked in data science, but well-known in operations research.

Many data science professionals use neural networks and other machine learning when discrete optimization would be more appropriate.

Even neural networks have to minimize their cost penalty function and find optimal values for weights (backpropagation and gradient descent).

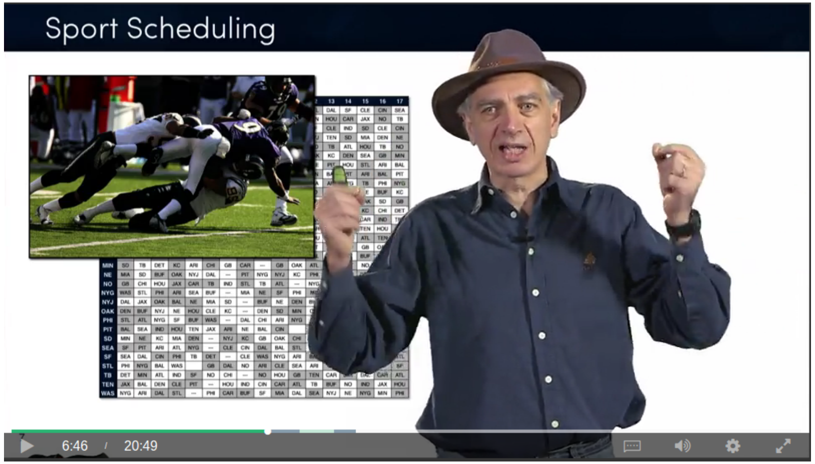
Recommended Java Libraries:

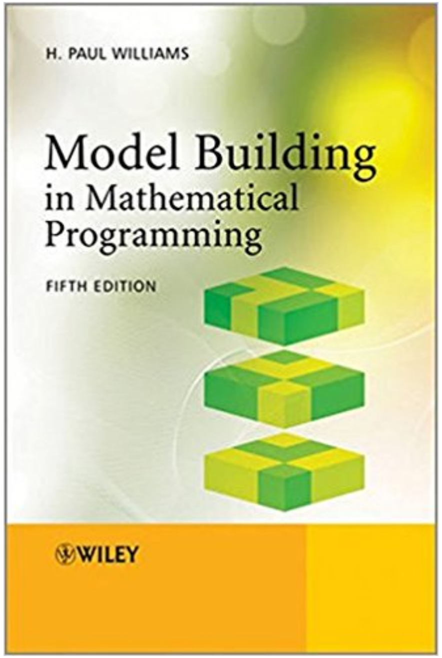
OjAlgo!

OptaPlanner

Learn More About Discrete Optimization

coursera *Discrete Optimization*





Mathematical Modeling Part II: Classification w/ Naive Bayes

Classifying Things

Probably the most common task in data science is classifying data:

- How do I identify images of ***dogs*** vs ***cats***?
- What ***words*** are being said in a piece of audio?
- Is this email ***spam*** or ***not spam***?
- What attributes define ***high***, ***medium***, and ***low*** profitability customers?
- How do I predict if a shipment will be ***late,* early**, or ***on-time***?

There are many techniques to classify data, with pros/cons depending on the ta:k:

- Neural networks
- Support Vector Machines
- Decision Trees/Random Forests
- Naive Bayes
- Linear/Non-linear regression

Naive Bayes

Let’s focus on Naive Bayes because it is simple to implement and effective for many use cases.

Naive Bayes is an adaptation of Bayes Theorem that can predict a category ***C*** for an item ***T*** with *multiple* features **F**.

A common usage example of Naive Bayes is email spam, where each word is a feature and **spam/not spam** are the possible categories.

It’s called “naive” because it assumes each feature is completely independent, but surprisingly it works well for many cases.

Implementing Naive Bayes

Naive Bayes works by mapping probabilities of each individual feature occurring/not occurring for a given category (e.g. a word occurring in ***spam***/***not spam***).

A category can be predicted for a new set of features by…

1) For a category, combine the probabilities of each feature for ***occur*** and ***not occur*** by multiplying them.

$$\text{Occur Product} = P_{f1} * P_{f2} * \ldots P_{fn}$$

$$\text{Not Occur Product} = !P_{f1} * !P_{f2} * \ldots !P_{fn}$$

2) Divide the products to get the probability for that category.

$$\text{Combined Probability} = \frac{(\text{Occur Product})}{(\text{Occur Product}) + (\text{Not Occur Product})}$$

Implementing Naive Bayes

3) Do this for every category, and select the one that has the highest probability.

Dealing with floating point underflow.

A big problem is multiplying small decimals for a large number of features may cause a floating point underflow.

To remedy this, transform each probability with **log()** or **ln()** and sum them, then call **exp()** to convert the result back!

$$\text{Occur Product} = exp(ln(P_{f1}) + ln(P_{f2}) + \ldots ln(P_{fn}))$$

$$\text{Not Occur Product} = exp(ln(!P_{f1}) + ln(!P_{f2}) + \ldots ln(!P_{fn}))$$

Implementing Naive Bayes

One last consideration, never let a feature have a 0 probability for any category!

Always leave *a little* possibility it could belong to any category so you don't have 0 multiplication or division mess anything up.

This can be done by adding a small value to each probability's numerator and denominator.

We will see all this in practice in our code.

Implementing Naive Bayes

One last consideration, never let a feature have a 0 probability for any category!

Always leave *a little* possibility it could belong to any category so you don't have 0 multiplication or division mess anything up.

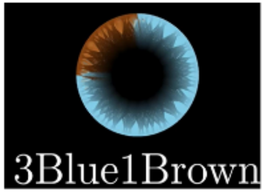
This can be done by adding a small value to each probability's numerator and denominator.

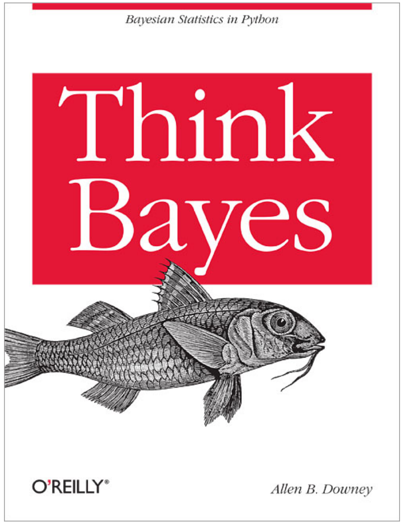
We will see all this in practice in our code.

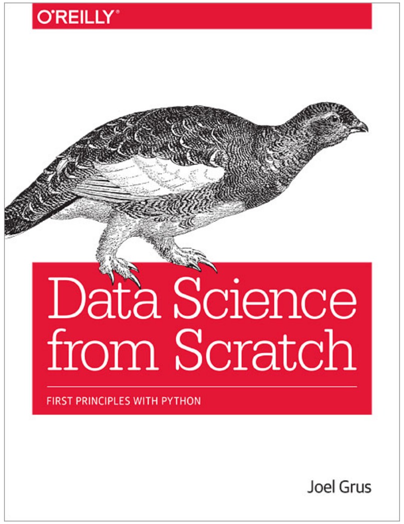
Helpful Resources on Machine Learning

Brandon Rohrer
https://brohrer.github.io/

Excellent YouTube Channel!







Mathematical Modeling Part III: Miscellaneous

Going Forward

Getting Involved

If you are interested in using Java and/or Kotlin for data science, learn what interests you:

Discrete Optimization

Machine Learning

Statistics

Probability

Data/Feature Engineering

Charting and visualizations

Don’t get discouraged when you find something overwhelming.

Always consult multiple resources and explore a mix of videos, books, blogs, and hands-on projects.

Although it is incredibly difficult to achieve, never stop striving for that unicorn status.

GitHub for this Session

https://github.com/thomasnield/kotlin_data_science_talk_ChicagoJUG