

Lecture I - Welcome and Introduction

Applied Optimization with Julia

Dr. Tobias Vlćek

University of Hamburg - Fall 2024

About this Course

About me

- **Field:** Optimizing and simulating complex systems
- **Languages:** of choice: Julia, Python and Rust
- **Interest:** Modelling, Simulations, Machine Learning
- **Teaching:** OR, Algorithms, and Programming
- **Contact:** tobias.vlcek@uni-hamburg.de

Course Structure

Lectures

- Every Tuesday between 10.15 AM and 11.45 AM
- First four lectures repeat modelling and programming
- Later lectures discuss practical problems and implementation
- Lectures are interactive → We discuss approaches!
- Communication takes place via OpenOlat and E-Mail

Tutorials

- Tutorials every Friday between 8.15 AM and 9.45 AM
- In these tutorials we are working on assignments
- Please bring a laptop with Windows, macOS, or Linux!
- This Friday there is **no tutorial!**

Assignments

- Based on applied problems of the lecture
- Up to 3 students can solve assignments together
- Submitted solutions earn bonus points for the exam
- Max. 0.5 point per tutorial

Course Objective

Applied Optimization

- Real-world problems can be addressed with models
- Our objective is to foster your interest in the topic
- Enable you to recognize and solve problem structures
- Includes problem understanding and implementation

Research in Operations Research

- Part of the University of Hamburg Business School
- Aiming to solve real-world problems
- Or improving our theoretical understanding
- Publication in journals and conferences

National and international journals

- European Journal of OR
- Journal of the Operational Research Society
- Journal on Applied Analytics
- Management Science
- Operations Research
- OR Spectrum

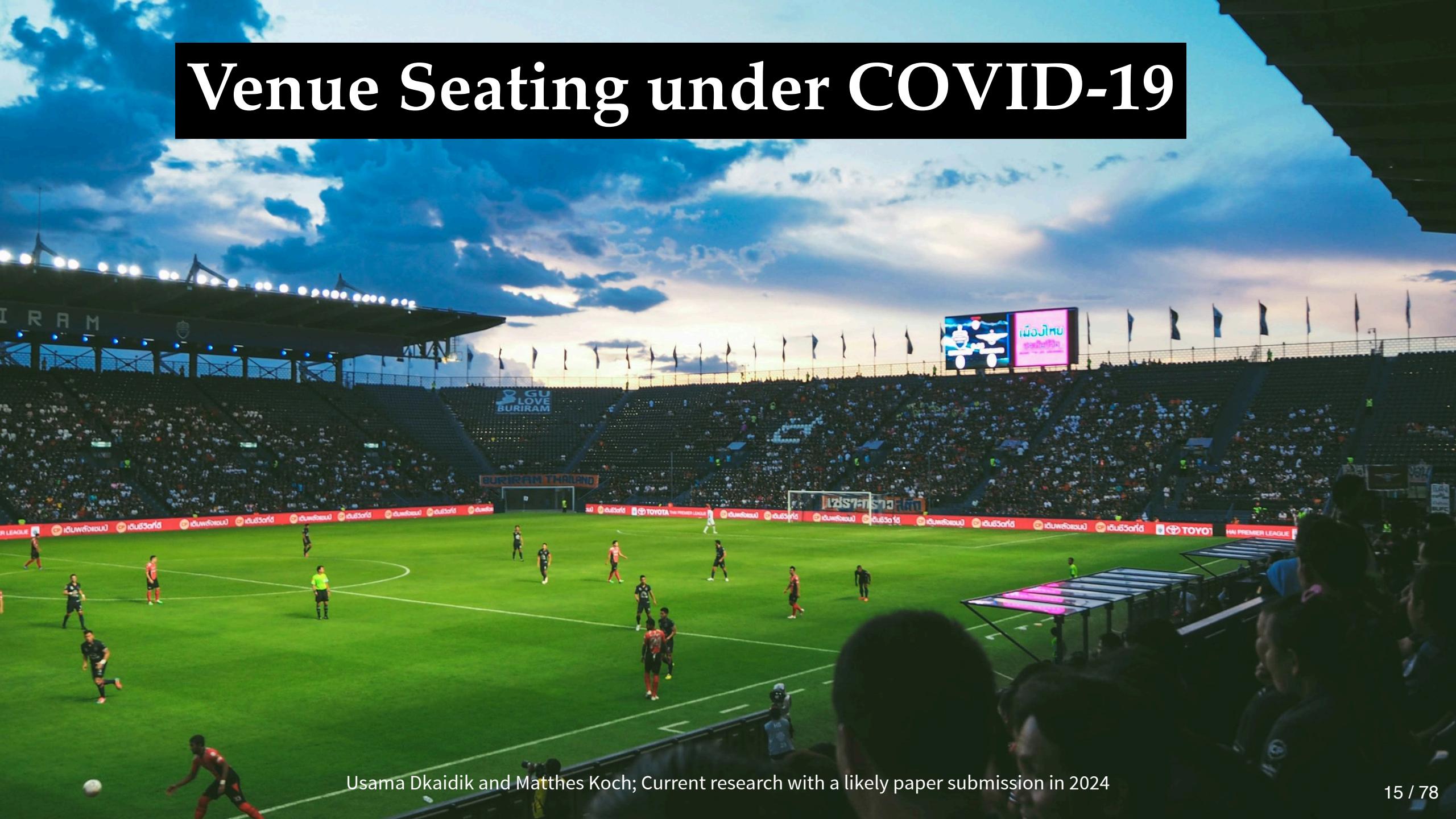
Real-World Applications

Brewery Production Planning

Police Service District Planning



Venue Seating under COVID-19



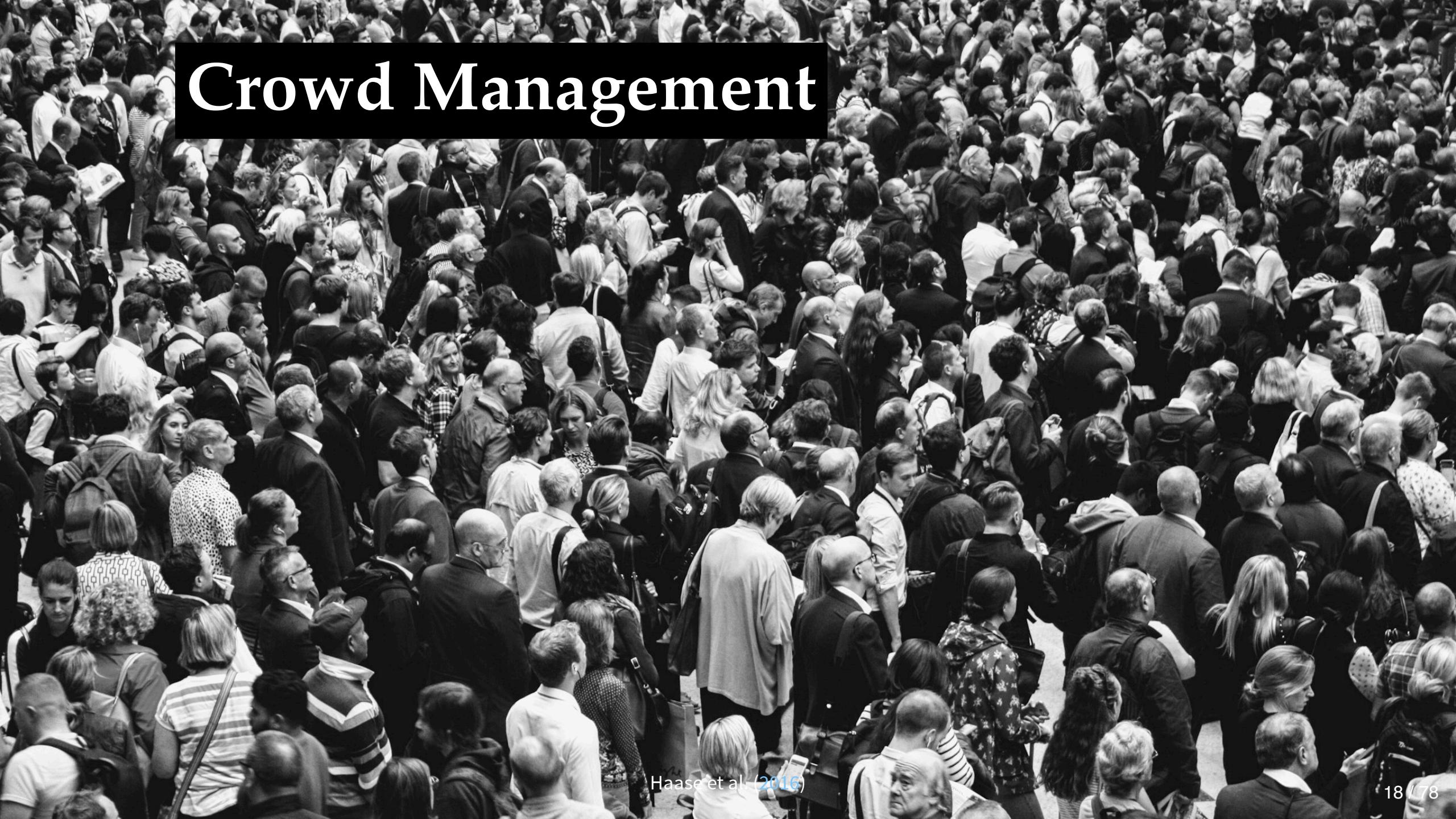
Metro Inflow Management



Split-Order Minimization



Crowd Management



Lecture Preview

Part I

1. Welcome and Introduction
2. First Steps in Julia
3. Packages and Data Management
4. Modelling with JuMP

Part II

5. Beer Production Planning
6. Minimizing Split Orders in E-Commerce
7. Periodic Library Routing
8. Police Districting

Part III

9. Safety Planning for the Islamic Pilgrimage in Mecca
10. Passenger Flow Control in Urban Rail
11. Arena Seat Planning under Distancing Rules
12. Sales Force Deployment for Teams
13. Recap and Discussion

Julia Programming Language

Choice of Programming Language



Why Julia?

- Designed to be:
 - as general as Python
 - as statistics-friendly as R
 - as fast as C++!

Syntax

- Dynamically-typed syntax just like Python
- Similar to R, Matlab and Python - **not like C++**
- In comparison, accessible and easy to learn!
- **No need to worry about memory management!**

JuMP

- Package for **algebraic modeling** in Julia
- Simplifies solving complex optimization problems
- Provides a high-level, user-friendly interface
- Useful for operations research and data science

Must it be a new language?

- Yes, but no need to worry!
- Julia is quite similar to Python and R
- We will learn the syntax together in the first part
- It is helpful to switch languages from time to time

Algebraic Modeling

Do you have
experience with
algebraic modeling?

What is algebraic modeling?

- A “mathematical language” for optimization problems
- Allows for describing complex systems and constraints
- Based on linear algebra (Equations and Inequalities)

How to learn algebraic modeling?

- Practice, practice, and practice!
- Understand standard models and their approach
- Develop an understanding of constraints
- Understand the structure of a models solution space
- Use an available algorithms to determine solutions

Central Questions

- What is to be decided?
- What is relevant to the decision?
- What information is given and relevant?
- What parameters (data) are needed?
- Which variables and of which type are needed?

Model Components

1. Objective function
2. Constraints
3. Variables

Linear Optimization Model

Basic Model Formulation

$$\text{maximize} \quad F = \sum_{j \in \mathcal{J}} c_j \times X_j$$

subject to

$$\begin{aligned} \sum_{j \in \mathcal{J}} a_{i,j} \times X_j &\leq b_i & \forall i \in \mathcal{I} \\ X_j &\geq 0 & \forall j \in \mathcal{J} \end{aligned}$$

Model Components

- \mathcal{I} : set of $i \in \mathcal{I}$,
- \mathcal{J} : set of $j \in \mathcal{J}$,
- F : Objective function variable,
- X_j : decision variables,
- c_j : objective function coefficients,
- $a_{i,j}$: parameters,
- b_i : parameters

What is this good for?

- **Good Question!** A lot of things:
 - Modeling real-world problems
 - Solving complex systems
 - Optimizing resource allocation
 - Decision-making under constraints
 - Simulation and prediction

Solar Panel Transport

Case: Solar Panel Transport

Description

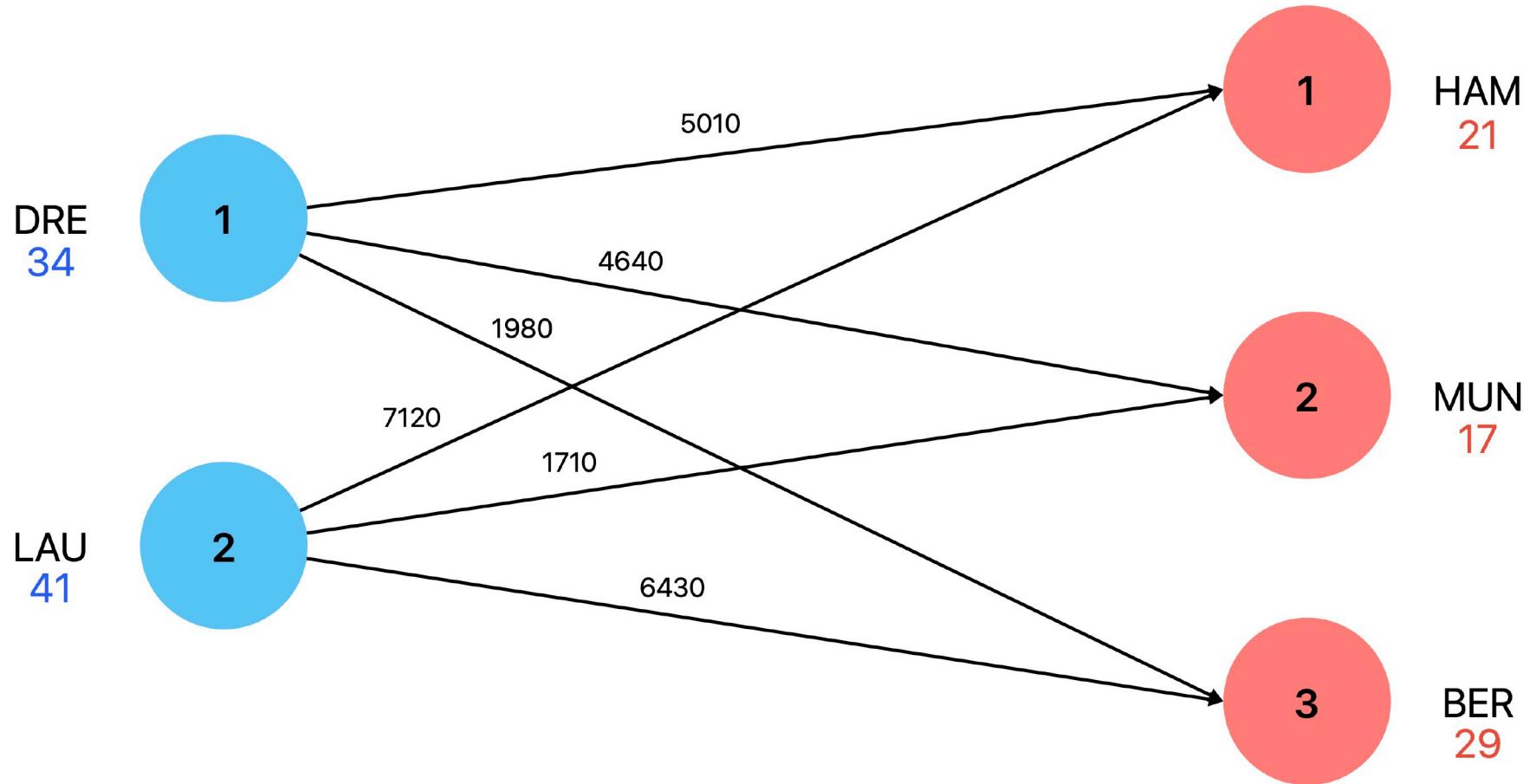
A company is producing **solar panels** in Dresden and Laupheim and has to transport them to new solar farms near Hamburg, Munich, and Berlin. The quantities offered and demanded (truckloads) and the transport costs per truckload in Euro are summarized in the following table.

Transport Costs

Origin/Destination	Hamburg	Munich	Berlin	Available
Dresden	5010	4640	1980	34
Laupheim	7120	1710	6430	41
Demand	21	17	29	

Example: A truckload from Dresden $i = 1$ to Munich $j = 2$ costs $c_{12} = 4640$ Euro. Moreover, it is necessary to **fulfil all customer demands**, as the contract has been signed.

Graphical Illustration



Understanding the Problem

What are we trying?

First, we always need to understand the objectives.

Let's break it down
step by step!

Sets

Remember, sets are **collections of elements**

Parameters

Parameters are **fixed values that are given.**

Decision Variable

- Decision variables are the values we are trying to find
- Here, our objective is to minimize the transport costs

These are our
building blocks!

Objective Function

- The objective function is the value we are trying to minimize
- Formalized as a sum of decision variables and parameters

Minimizing the Transport Costs

Minimize the transport costs over all truckloads while meeting the demand based on the available solar panels adhering to the available panels.

Constraints

- Constraints are **conditions that must be met**
- They limit the solution space!

Question: Objective value without any constraints?

Supply Constraints

Ensure that the number of panels transported from a location does not exceed the available panels.

Demand Constraints

Ensure that the demand of each customer is covered.

Non-negativity Constraints

Ensure no negative number of truckloads are transported.

Transport Problem

The complete model can then be written as:

$$\text{Minimize} \quad F = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{i,j} \times X_{ij}$$

$$\text{subject to:} \quad \sum_{j \in \mathcal{J}} X_{i,j} \leq a_i \quad \forall i \in \mathcal{I}$$

$$\sum_{i \in \mathcal{I}} X_{i,j} = b_j \quad \forall j \in \mathcal{J}$$

$$X_{i,j} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$$

Inequality Constraints

Question: Could we replace $=$ by \geq in the demand constraint?

Any questions?

Profit Maximization

Description

Unfortunately, the margins on solar panels are low. After the previous contract has been fulfilled, the company produced the **same number of panels** as before. In addition, all three customers want to order the **same number of truckloads** with solar panels again. The sales volume per truckload of panels is 11,000 Euros. The complete production of a truckload of solar panels, including materials, costs 6,300 Euros.

New Objective

In the new contract, the company wants to **maximize its profits** while the demand does not have to be fulfilled.

New Parameters

p : Sales volume per truckload of solar panels,

c : Production costs per truckload of solar panels.

Former Model

$$\text{Minimize} \quad F = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{i,j} \times X_{i,j}$$

$$\text{subject to:} \quad \sum_{j \in \mathcal{J}} X_{i,j} \leq a_i \quad \forall i \in \mathcal{I}$$

$$\sum_{i \in \mathcal{I}} X_{i,j} \geq b_j \quad \forall j \in \mathcal{J}$$

$$X_{i,j} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$$

New Model

$$\text{Maximize} \quad F = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} (p - c - c_{i,j}) \times X_{i,j}$$

$$\text{subject to:} \quad \sum_{j \in \mathcal{J}} X_{i,j} \leq a_i \quad \forall i \in \mathcal{I}$$

$$\sum_{i \in \mathcal{I}} X_{i,j} \leq b_j \quad \forall j \in \mathcal{J}$$

$$X_{i,j} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$$

Do you have
any questions?

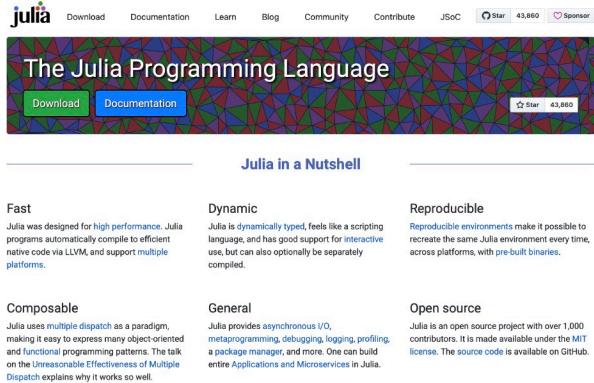
Installing Julia

Download and Install Julia



To prepare for the upcoming lectures, we start by installing the Julia Programming Language and an Integrated Development Environment (IDE) to work with Julia.

Installating Julia



- Head to julialang.org and follow the instructions.



Tip

If you are ever asked to add something to your “PATH”, do so!

VS Code



- Next, we are going to install VS Code
- Alternatively, you can install VS Codium
- It is essentially VS Code but without any tracking by MS

Installing VS Code

- Head to the website code.visualstudio.com
- OR to the website vscodium.com
- Download and install the latest release

Verify the Installation

- Start the IDE and take a look around
- Search for the field “Extensions” on the left sidebar
- Click it and search for “Julia”
- Download and install “Julia (Julia Language Support)”

Create a new file

- Create a new file with a “.jl” ending
- Save it somewhere on your computer
- e.g., in a folder that you will use in the course

```
1 print("Hello World!")
```



- Run the file by clicking “run” in the upper right corner
- OR by pressing “Control+Enter” or “STRG+Enter”

Everything working?

- If the terminal opens with a Hello World! → perfect!
- If not, it is likely that the IDE **cannot find the path** to Julia
- Try to determine the path and save it to VS Code
- After saving it, try to run the file again



Don't worry if it is not running right away. We will fix this together!

Starting with Julia

How to get started?

- Learning a new programming language is a daunting task
- It is best to start with some small, interactive problems
- Then, slowly increase the scope of the tasks
- We will do this **together in class!**



Note

And that's it for todays lecture!

We now have covered a first introduction and are ready to start solving some problems in the upcoming lectures.

Questions?

Literature

Literature I

For more interesting literature to learn more about Julia, take a look at the [literature list](#) of this course.

Literature II

- Haase, Knut, Habib Zain Al Abideen, Salim Al-Bosta, Mathias Kasper, Matthes Koch, Sven Müller, and Dirk Helbing. 2016. “Improving Pilgrim Safety During the Hajj: An Analytical and Operational Research Approach.” *Interfaces* 46 (1): 74–90.
- Mickein, Markus, Matthes Koch, and Knut Haase. 2022. “A Decision Support System for Brewery Production Planning at Feldschlösschen.” *INFORMS Journal on Applied Analytics* 52 (2): 158–72.
- Vlćek, Tobias, Knut Haase, Malte Fliedner, and Tobias Cors. 2024. “Police Service District Planning.” *OR Spectrum*, February. <https://doi.org/10.1007/s00291-024-00745-3>.
- Vlćek, Tobias, Knut Haase, Matthes Koch, Lena Dolz, Anneke Weygandt, and Jan Pape. 2024. “Controlling Passenger Flows into Metro Systems to Mitigate Overcrowding During Large-Scale Events.” *Submitted to Transportation Science*.
- Vlćek, Tobias, and Guido Voigt. 2024. “Optimizing SKU-Warehouse Allocations to Minimize Split Parcels in E-Commerce Environments.” *Submitted to Decision Sciences*.