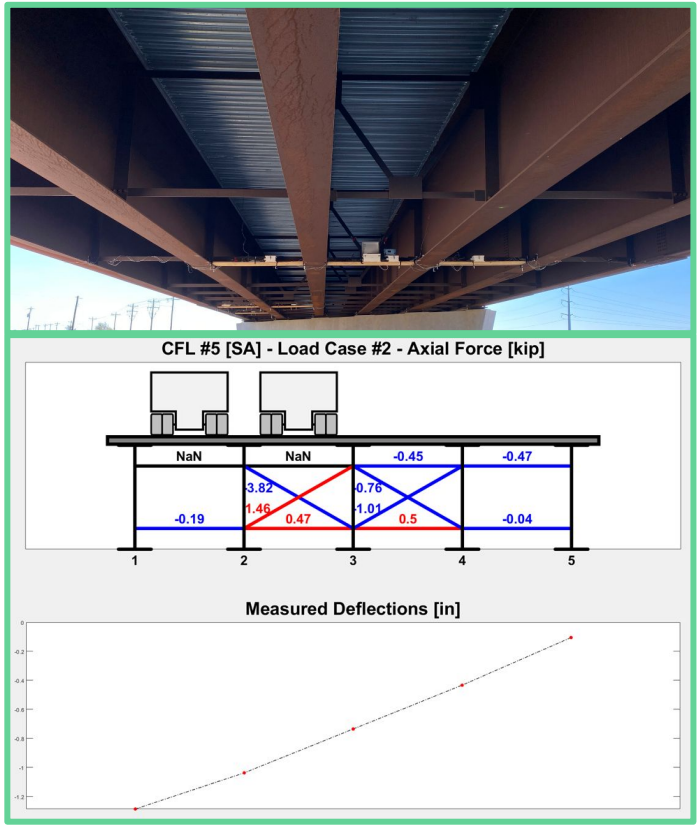
Bridge Building with Python & Abaqus

Just 7: Aidan Bjelland, Inrae Cho, Caroline Dolbear

Project Overview

Project Background / Motivation

- TxDOT Project 0-7093: Lean-on Bracing
  - Need lots of data to narrow in on factors affecting design...
  - Data has to come from somewhere (instrumenting 1000s of bridges in not feasible)...
    - Parametric studies can be run utilizing Abaqus and Python!

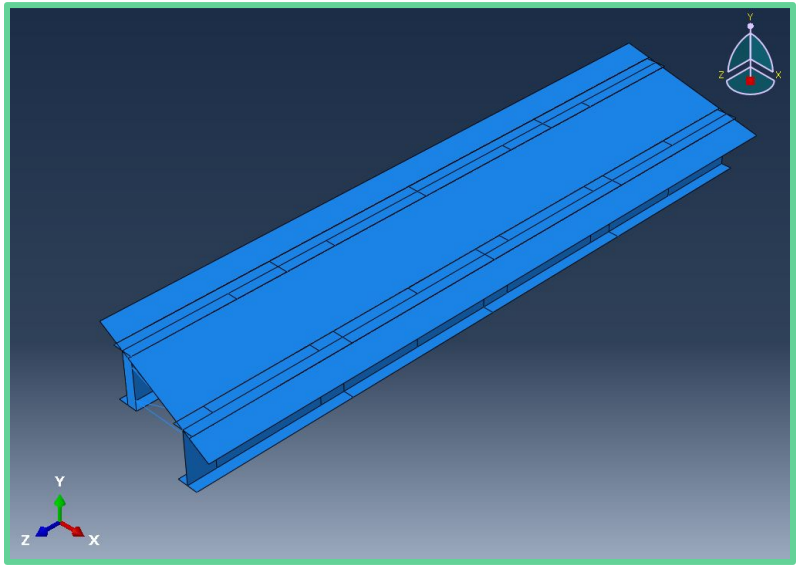


Background of Abaqus + Python Scripting

- Abaqus is a general Finite Element Analysis (FEA) program
- Abaqus, and other FEA programs, have built-in scripting functionality
  - Python is typically utilized for this scripting
- Abaqus has an inbuilt IDE (limited to Python 2.7)
  - Runs python scripts
  - Runs abaqus macros
- Enables automation of repetitive modelling tasks!

Project Objective (Final Project Edition)

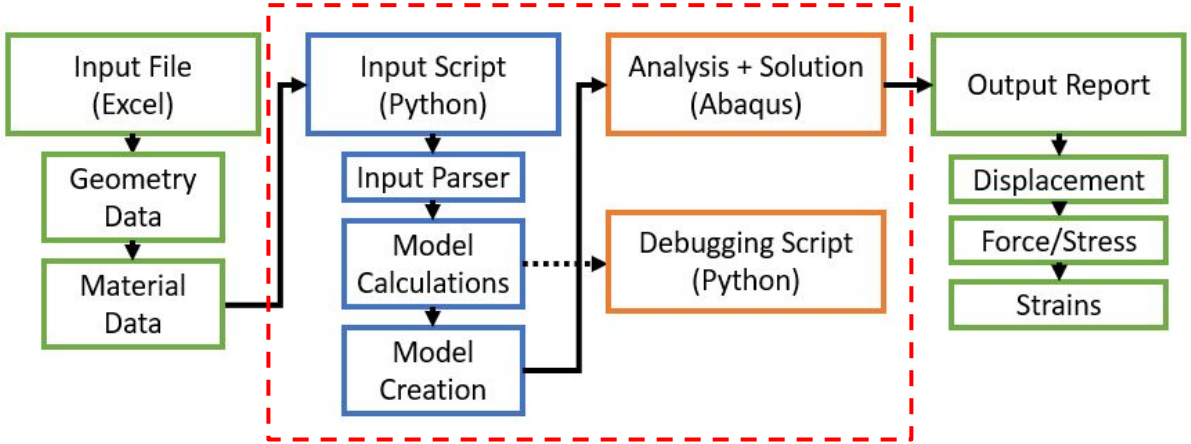
- Create a python code that reads an Excel spreadsheet as an input file, computes necessary calculations to create a bridge model, generates that model in Abaqus, and then analyzes that model



Project Approach

Project Requirements

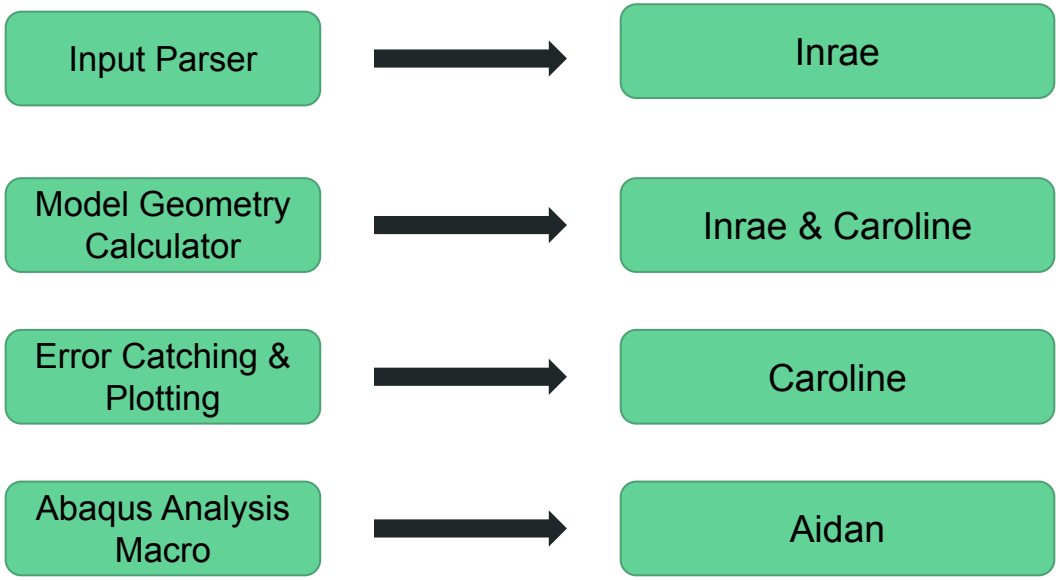
- Retrieves raw data from input csv file row by row
- Calculates and assigns position of critical points used in model creation and FEA
- Visualize input prior to model creation and FEA
- Creates bridge model compliant to user input in Abaqus
- Conducts FEA in Abaqus



Packages Utilized

- math, numpy
  - Assisted in model geometry calculations
- matplotlib
  - Tool to confirm correctness of model geometry calculations
- csv
  - Input parser
- abaqus
  - section
  - regionToolset
  - part
  - material
  - assembly
  - AND MORE!

Task Delegation



Programmatic Features + Capabilities

- Preprocessor
- Error Catching & Plotting Script
- Abaqus Macro Script

InputPreprocessor.py

- Input Parser & Data Manipulation
- Geometry Calculations

Purpose and Capabilities - Input Parser (Formatting)

What can the user do:

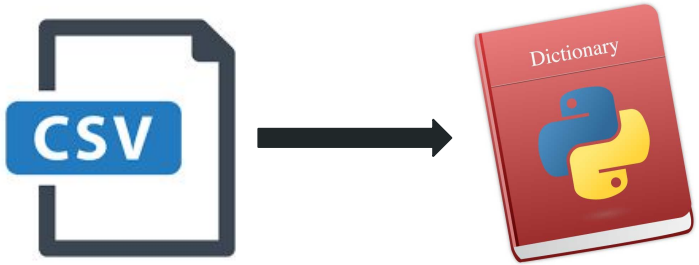
- Retrieve raw data from input csv file
- Correct format of raw data
- Store data for geometry calculation and model creation

How does it work:

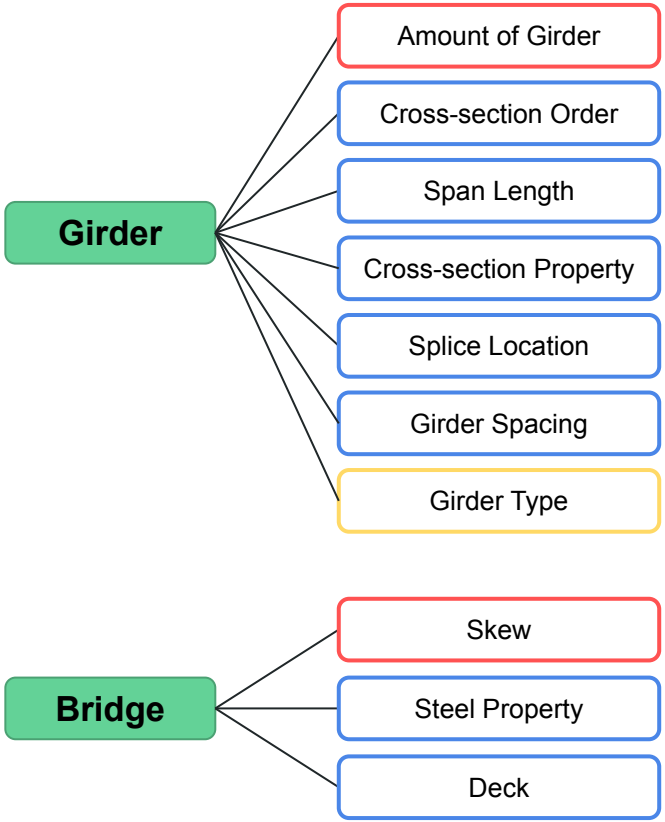
- User inputs multiple geometry and system parameters in a csv file
- Each cell is initially recognized as string or list of strings
- Input Parser converts raw data into desired format and stores them in multiple dictionaries

What are the limitations (assumptions):

- User must follow valid format for each parameter
- Only compatible with .csv file
- All numeric inputs must be unitless



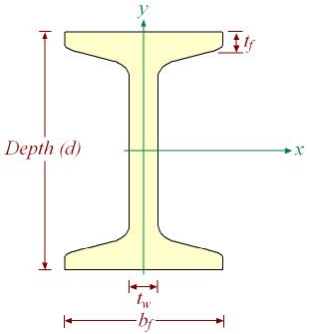
Input File Formats



Geometry Calculations

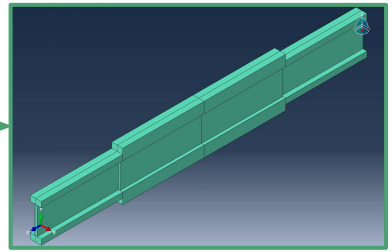
Girder Profile Sketch

`GirderSketch(xprops, xtype)`



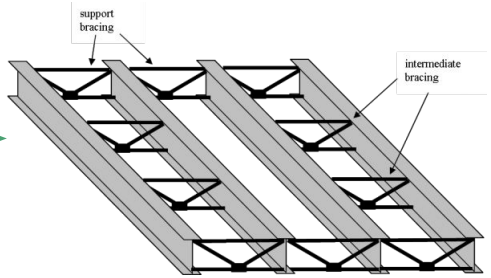
Splice Locations

`system_iden.splice_local(System_info)`



Bracing Locations

`system_iden.normal_bracing(System_info)`
`system_iden.parallel_bracing(System_info)`



InputDebug.py

Debugging Functions

Error Catching: `DEBUG_DataErrors(System_info)`

- Verifies that input file is of the proper format
  - Strings are accepted strings
  - Data types are correct
  - List lengths are compatible
  - Check that data makes sense

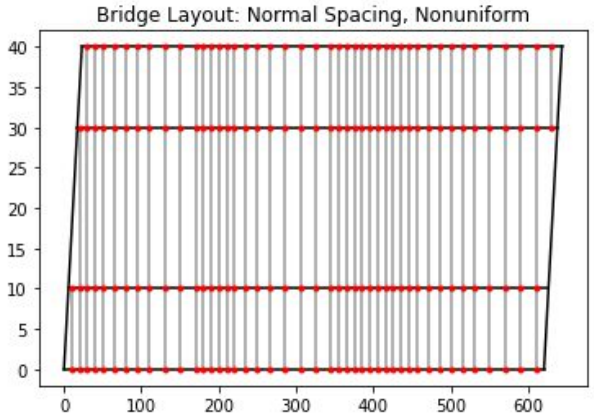
```
class DataTypeError(Exception):
    pass

class UnacceptedEntry(Exception):
    def __init__(self, entry, accepted_entry, message):
        self.entry = entry
        self.accepted_entry = accepted_entry
        self.message = message
    def __str__(self):
        errormessage = self.message + ': {} is not in {}'.format(self.entry, self.accepted_entry)
        return errormessage

class ListLengthError(Exception):
    pass
class DataError(Exception):
    pass
```

Plotting: `DEBUG_Plotting(x, y, System_info)`

- Verifies that bracing output is as expected
  - Visual check before running in Abaqus
  - Using matplotlib.pyplot



Demonstration #1

Error Catching & Plotting

AbaqusMacro.py

Functionality

Features:

- Part Creation
  - Girder Sketch to Part
  - Stiffener Sketch to Part
  - Strut Sketch to Part
  - Deck Sketch to Part
- Assembly
  - Bracing System (Stiffener + Struts)
  - Superstructure (Bracing System + Girders)
  - Bridge (Superstructure + Deck)
- Analysis
  - Define mesh
  - Define support conditions
  - Define loads
  - Define and execute job

What are the limitations:

- Depth of web is constant
- Base of flange is constant
- One girder definition
- One bracing system type per bridge

Key Design Choices:

- Sketch to Part
- Assembly Assumptions

Demonstration #2

- Normal Bridge with Uniform Normal ‘X’ Bracing
- Skew Bridge with Nonuniform Parallel ‘K’ Bracing

Q&A

Project Links

Code - Github:

https://github.com/ADBjelland/Final-Project---Team-7

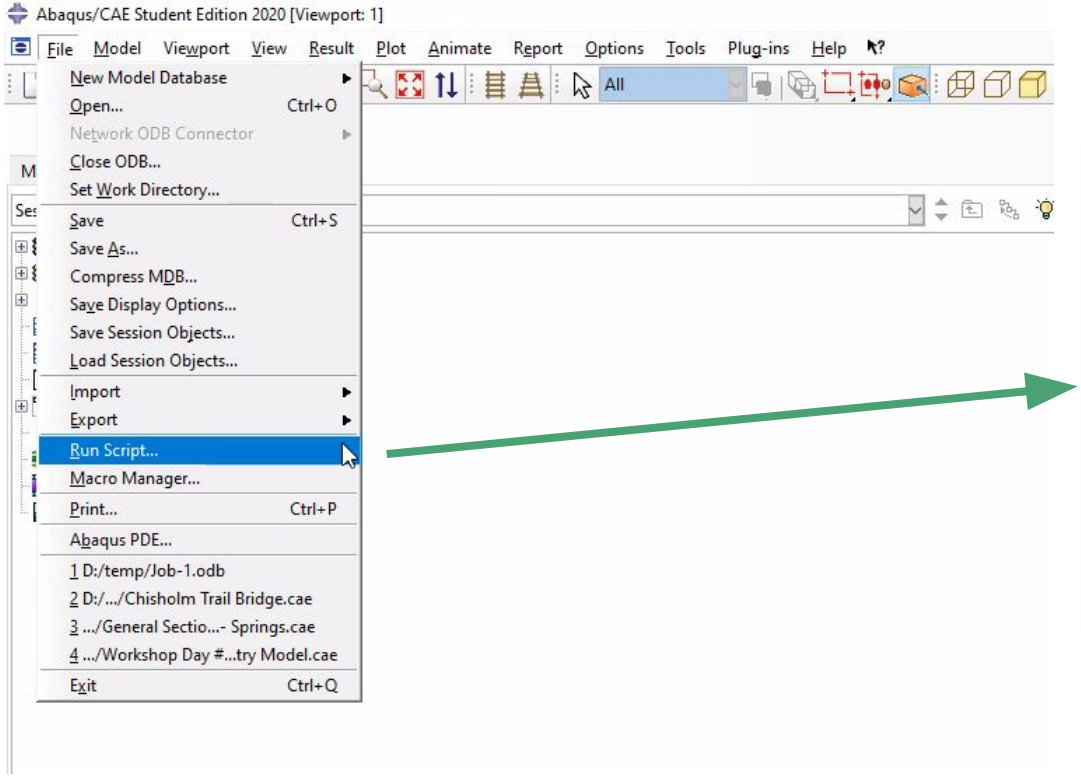
Code Documentation:

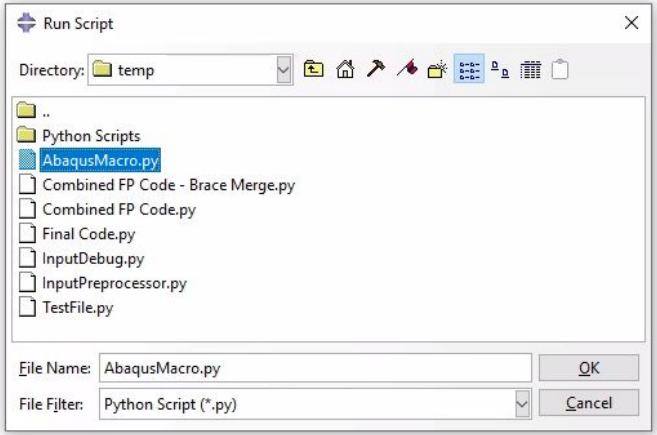
https://docs.google.com/document/d/10zO6yMPpG-RnKnAKklPowEqYQDxbSeN2/edit?usp=sharing&ouid=104216158585620858320&rtpof=true&sd=true

Input File:

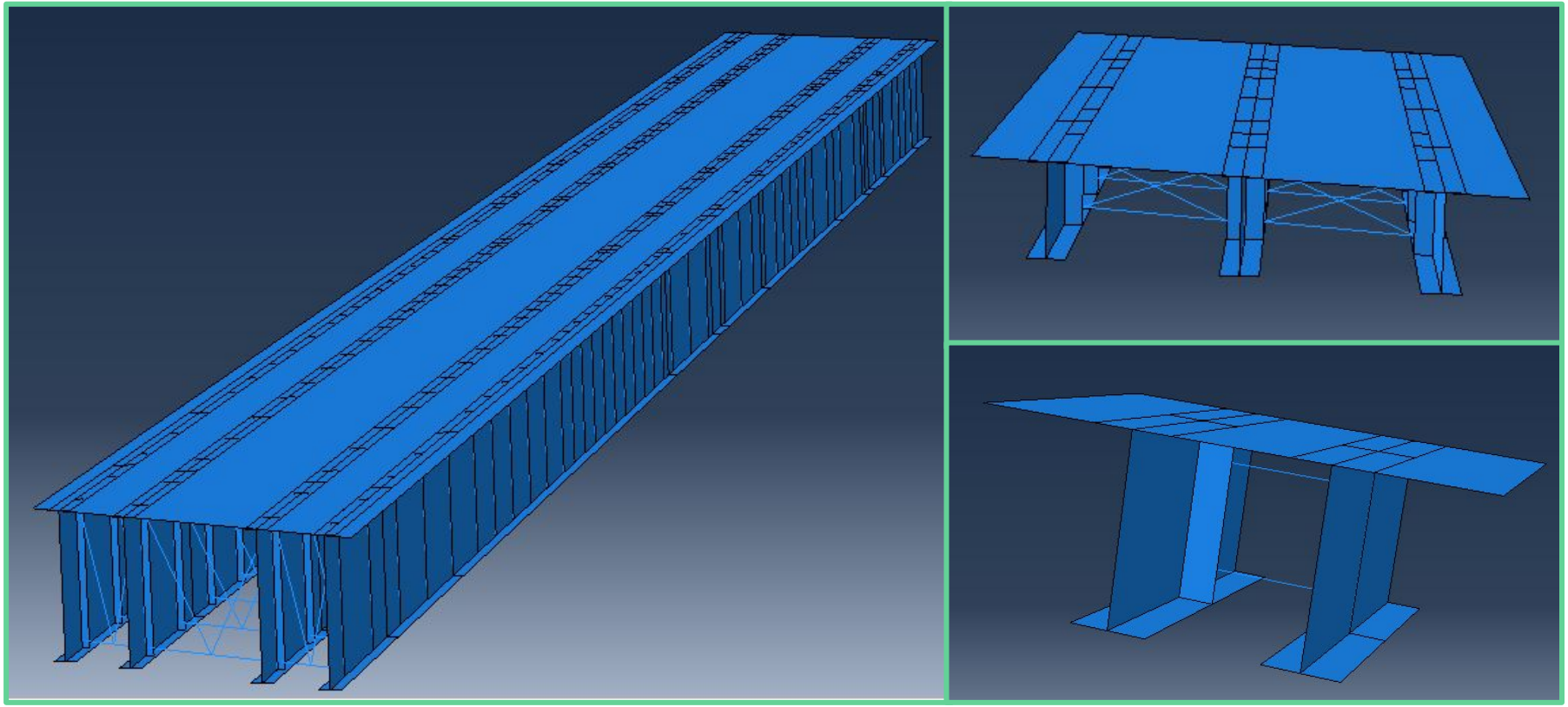
https://docs.google.com/spreadsheets/d/1d6-8EyLCfyEOkocTmbgUMpCe0gZMhizHeagAkf8dsB4/edit?usp=sharing

Abaqus Interface





Bridge Examples



Input File Example

Example CSV for Debug Demo in Github

Bridge	Girder	Girder	Girder	Girder	Girder
Dictionary Key	Girder Number	Span Length	Cross Section Order	Cross Section Property	Splice Location
ID	Amount of Girders	Span Length	Cross Section Order	Cross Section Properties [[tf, bf, tw, dw],...]	Splice Location (Assume new sections begin at
1	4	[100,200,500]	[1,1,1,1];[1,1,1,1];[1,1,1,1]	[0.5,2,3,6];[1,2,3,4];[1,2,0.5,2]	[20,50,20];[40,100,50];[100,300,50]
2	4	[100,200,340]	[1,1,1,1];[1,1,1,1];[1,1,1,1]	[0.25,3,0.1,20]	[20,50,20];[40,100,50];[50,200,50]
3	2	[100,200,500]	[1,1,1,1];[1,1,1,1];[1,1,1,1]	[0.5,2,3,6];[1,2,3,4];[1,2,0.5,2]	[20,50,20];[40,100,50];[100,300,50]
4	2	[100,200,500]	[1,1,1,1];[1,1,1,1];[1,1,1,1]	[0.5,2,3,6];[1,2,3,4];[1,2,0.5,2]	[20,50,20];[40,100,50];[50,200,50]
5	4	[100,200,340]	[1,3,2,1];[1,2,3,1];[1,2,2,1]	[0.5,2,3,6];[1,2,3,4];[1,2,0.5,2]	[20,50,20];[40,100,50];[50,200,50]
6	4	[100,200,300]	[1,3,2,1];[1,2,3,1];[1,2,2,1]	[0.5,2,3,6];[1,2,3,4];[1,2,0.5,2]	[20,50,20];[40,100,50];[50,200,40]

...many more columns!