PRINCETON CONSULTANTS

Information Technology and Management Consulting

Rapid Optimization Model Development with Python and Pandas

Rob Randall, PhD – Senior Optimization Specialist
Irv Lustig, PhD – Optimization Principal

2022 INFORMS Business Analytics Conference

April 3, 2022

PRINCETON CONSULTANTS

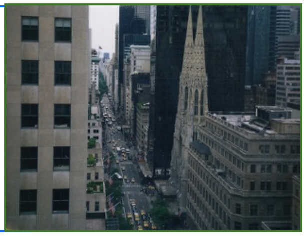
Information Technology and Management Consulting

www.princeton.com

2 Research Way
Princeton New Jersey

Information Technology & Management Consulting

Advanced Analytics *Strategy*
Application Development *Process Improvement*



590 5th Ave
New York City

Transformation through AI and Optimization

- **Clients**: Leading transportation and industrial supply chain organizations
- **Track Record**: 42 years in business under founding leadership; Over 1800 successful completed projects
- **Experience**: Senior Staff (top 28 consultants) average 15+ years; Firm Leadership (6 Directors) average 20+ years experience
- **Mission**: Helping clients Accelerate their Digital Transformation

“Life-Altering” Improvements

“The increase in speed and flexibility from the new model impacts every piece of our business across multiple teams. Now we can strategically invest that time in everything from building more personal customer experiences to optimizing our production process. With this new model, Birchbox has truly entered a new operating universe.”

David Bendes

Vice President of Global Business Technology at Birchbox

Introduction

Princeton Consultants has developed a 7-step approach that we believe improve the likelihood of success in optimization projects. This approach:

- Creates useful documentation
- Shortens the overall development time
- Improve code maintainability
- Provides a natural feedback loop to business requirements

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Introduction

- The main tools used in the development are python with the pandas library
  - Pandas provides a very clean, sparse way of containing and manipulating the data
  - Python has a rich ecosystem which aids in deployment
- This is often used in concert with Gurobi’s python interface (gurobipy)
  - They can also be used with other python APIs such as IBM ILOG CPLEX python interface (docplex), FICO Xpress-MP, and Google’s OR Tools (ortools)
- These techniques have been applied to multiple clients and verticals successfully

Example Problem

A client has tasked you with with helping them solve a scheduling problem.

There are a set of tasks to be scheduled

There are a set of resources to schedule the tasks

There are groups of resources

Tasks have different processing times

Each resource has a maximum length of time that they can be actively working on tasks

A Resource has a specific operating cost per time unit

Some tasks cannot be completed by some resources

Each resource group has a minimum and a maximum number of tasks that they need to do in a day

The goal is to minimize the total cost of processing all the tasks

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Understand the Data

- Understanding the data required is an ABSOLUTE NECESSITY for an optimization project to be successful
- For this step work with your clients to understand:
  - What data is available
  - The semantic meaning of that data
  - How they organize that data
  - How different pieces of data are related to each other
- This is the MOST IMPORTANT step in an optimization project, because without a deep understanding of the data available for optimization, we cannot build a good model
  - Irv is giving a detailed talk on data first modeling on Tuesday morning.

Determining the Available Data

› How many times have you (or seen some one who has) built a model, probably a beautiful, elegant model, only to find out that half of the data that you planned to use doesn’t exist?

› That’s why before we ever start modeling, we work with our clients to determine what data they have and what state that data is in

  – Sometime there may be data, but it may be incomplete or inaccurate

› This is a key step in determining what type of model can be built

The semantic meaning of that data

- Are there universally agreed upon names for data fields?
  - In my experience there is not, so while one company might call a piece data X, for another that may be Y, and X may be something entirely different.
  - Another issue can be mismatched units
    - Column may be named Minutes but the data is saved in Seconds
- Therefore, not only do we work with our clients to determine what data is available to model, but to make sure we know what each piece of data means and how it was created.
  - This generally involves creating a data dictionary so that we have definitions of what each table and each field in those tables represent
  - This is a good practice for the client to get into as well, as often with turnover there are fields that the client isn’t even always sure the meaning of

How different pieces of data are related to each other

› Not only do we need to know how each piece data is defined but need to know how the different pieces of data work together.

› Determine relationships and other linking pieces that can be used when building the model
  – Some of these may determine what variables you can use
  – And what constraints may need to be built on those relationships

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Define and Document Datasets

- **A format needs to be defined for the client to deliver the datasets in a consistent and meaningful format to the team**
  - We generally use the concept of “tidy data” as a starting point for this format
    - Columns contain variables, rows contain observations
  - For more information on ”tidy data” see Irv’s talk on Tuesday
- **When creating these definitions, we define mathematical notation to represent the data items**
  - Use Markdown where we can include LaTeX notation.
    - This is generally done in either Visual Studio Code or a Jupyter Notebook
    - Can export using Pandoc to a distributable (doc, pdf) form
      - This document is often reviewed with the client

Define and Document Datasets

- The notation that we have developed involves the following:
  - For defining sets, we prefer using the LaTeX `\mathcal` prefixing upper case letters so they look like this, $\mathcal{P}$:
    - Using the corresponding lower-case letter to refer to an individual element of the set
  - Greek letters are used to represent input data defined on those sets.
    - For example, the set labeled as $\mathcal{P}$ might have a data vector δ for which the individual elements of the vector are written δ_p where $p \in \mathcal{P}$.

Example – Data Table Documentation

Tasks have different processing times

Each resource group has a minimum and a maximum number of tasks that they need to do in a day

A Resource has a specific operating cost per time unit

The goal is to minimize the total cost of processing all the tasks

Table	Column
Tasks	Task – The unique identifier for the task
	Time – The task duration, in hours
ResourceGroups	ResourceGroup – The unique identifier for the resource group
	MinTasks – The minimum number of tasks this resource group can perform in a day
	MaxTasks – The maximum number of tasks this resource group can perform in a day
Resources	Resource – The unique identifier for the resource
	HoursAvailable – The number of hours this resource can be used in a day
	CostPerHour – The cost, in dollars, per hour for using this resource
TaskResource	Task – A reference to a Task instance
	Resource – A reference to a Resource that can perform this Task
	Cost – The cost to assign a task to a resource

Example – Model Data Documentation

Sets

Let $\mathcal{T}$ be the set of tasks that need to be completed.

- **Tasks** table, **Task** column

Let $\mathcal{G}$ be the set of resource groups.

- **ResourceGroups** table, **ResourceGroup** column

Let $\mathcal{R}$ be the set of resources that can complete the tasks.

- **Resource** table, **Resource** column

Let $\mathcal{Q}_r$ be the set of tasks that resource $r \in \mathcal{R}$ can perform.

- All values of the **Task** column in the **TaskResource** table that have Resource *r*

Let $\mathcal{P}_t$ be the resources that task t ∈ $\mathcal{T}$ can be scheduled on.

- All values of the **Resource** column in the **TaskResource** table that have Task *t*

Let $\mathcal{S}_g$ be the resources that belong to resource group $g \in \mathcal{G}$.

- **Resource** table, **ResourceGroup** column

Example – Model Data Documentation

Data Inputs

Let $\eta_t > 0$ be the number of hours for a task $t \in \mathcal{T}$.

- **Task** table, **Time** column

Let $\lambda_r \geq 0$ be the cost per hour for a resource $r \in \mathcal{R}$.

- **Resource** table, **CostPerHour** column

Let $\gamma_r \geq 0$ be the number of hours that a resource $r \in \mathcal{R}$ can work in one day.

- **Resource** table, **HoursAvailable** column

Let $\alpha_g \geq 0$ be the min number of tasks, $t \in \mathcal{T}$, that resource group $g \in \mathcal{G}$ can work.

- **ResourceGroup** table, **MinTasks** column

Let $\beta_g \geq 0$ be the max number of tasks, $t \in \mathcal{T}$, that a resource group $g \in \mathcal{G}$ can work.

- **ResourceGroup** table, **MaxTasks** column

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Define and Document Datasets

› Generally, comes in one of two ways
  - Client just dumps their data to us
  - Work with client to determine our data needs and specify what data to send and the format for that data

› All data should be in a tabular format
  - CSV
  - Excel with multiple sheets

Client Data Dump

- When the client dumps data to us, we refer to that as the "raw data format".
  - Production data
  - Not necessarily model friendly
- Using Python and Pandas, transform this data to a format that's model friendly, the "model data format"
  - Calendar dates transformed to discrete periods
  - Dropping unnecessary data
  - Combining multiple production tables into data tables we've defined

Use Created Data Tables

- Client uses the data tables we create to feed data to us
  - This is the best way for us, which means it almost never happens
  - Cuts down on the data passed and work we do, since the data matches our tables
- Still may need some processing
  - Calendar to discrete periods

Collect Data From the Client

- As we work on this code, sometimes we must update our documentation of the tidy data format that we are expecting
  - This is a living document
    - Update based on new understanding
    - When new data is added for features
- Once the input data to the model has been defined there should be some data validation tests added.
  - This will help prevent data issues that pop up in production and “break” the model.
    - Not going to catch them all
    - Continue refining throughout project

Example – Data Validations

- What are some validations that should be added for this model?
  - Ensure no task takes longer than a resource can work
  - Ensure every task can be done by at least one resource
  - The total resource hours available exceeds the total task work hours required
  - Ensure there are enough tasks that can be assigned to resources in each resource group to meet the minimum number of tasks for that group
  - Ensure that the total number of tasks that can be performed by all resource groups exceed the number of input tasks
- Do this in terms of data definitions, doesn’t have to be an optimization person who codes that

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. **Document Data Transformations**
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Document Data Transformations

- Need to document any transformations that are made to the tidy data
  - Especially, any that combine data into a new data input
- New data input definition documentation
  - If we have two inputs, α and β that are indexed on the same set $\mathcal{P}$ with the individual elements denoted as α_p and β_p and we need to compute the sum of the minimum of the two quantities denoted as γ :

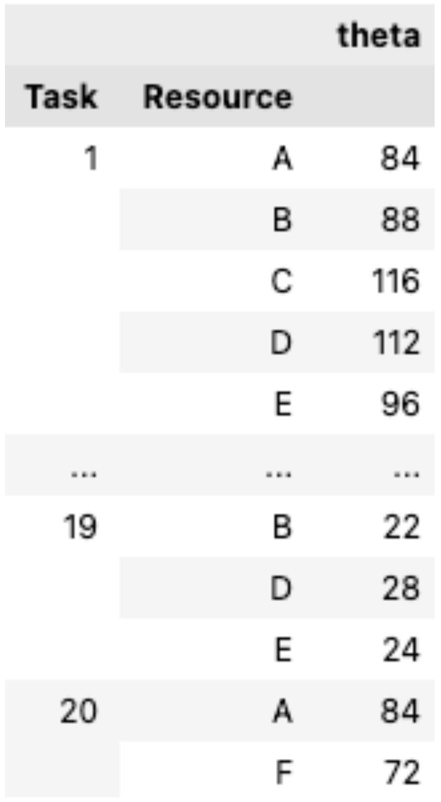
$$\gamma = \sum_{p \in \mathcal{P}} \min\{\alpha_p, \beta_p\}$$

Example

› Data Transformations

– Let θ_{rt} be the cost for a resource $r \in \mathcal{R}$ performing a task $t \in \mathcal{T}$.

$$\theta_{rt} = \eta_t \cdot \lambda_r \qquad \forall r \in \mathcal{R}, t \in Q_r$$



		theta
Task	Resource	
1	A	84
	B	88
	C	116
	D	112
	E	96
...	...	...
19	B	22
	D	28
	E	24
20	A	84
	F	72

Task	Time
1	2
2	3
3	2
4	3

Resource	Cost
A	29
B	26
C	25
D	26
E	17
F	22

Task	Resource
1	A
1	B
1	C
1	D
1	E
2	D
2	E
2	F
3	E
4	A
4	B
4	C
4	D
5	B

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. **Create Model Documentation**
6. Code Optimization Model
7. Validate Results and Iterate

Create Model Documentation

› Now that the data inputs and transformations are documented, we can write a document that describes the optimization model that we will build. This includes:

- Decision variables
  - › Our convention is to use a single lower-case English letter for a decision variable
  - › The sets that describe these variables are listed using subscripts
- Constraints
  - › Add constraints to the model one at a time to allow for constant model testing
- Objective function(s)
  - › A multi-objective model, or
  - › Allow users to choose from multiple objectives

Create Model Documentation

Each item is also described in “Plain English” to ensure an understanding of what the model is trying to accomplish

- Auditability
- Show understanding of rules to business
  - Create an Optimization Business Logic Document, that is shared specifically with business users
- Used to create a solution checker

Example Problem

A client has tasked you with with helping them solve a scheduling problem.

There are a set of tasks to be scheduled

There are a set of resources to schedule the tasks

There are groups of resources

Tasks have different processing times

Each resource has a maximum length of time that they can be actively working on tasks

A Resource has a specific operating cost per time unit

Some tasks cannot be completed by some resources

Each resource group has a minimum and a maximum number of tasks that they need to do in a day

The goal is to minimize the total cost of processing all the tasks

Example Problem

Let’s try and characterize the requirements.

Decisions

There are a set of tasks to be scheduled on a set of resources	Determine the number of tasks scheduled by a group of resources

↓

Rules

Each task must be scheduled	Each resource has a maximum length of time that they can be actively working on tasks	Each resource group has a minimum and a maximum number of tasks that they need to do in a day

↓

Goals

Minimize the total cost of processing all the tasks

Example – Model Documentation

Decision Variables

There are a set of tasks to be scheduled on a set of resources

- Let $x_{rt} \in \{0,1\}$ be 1 if the resource $r \in \mathcal{R}$ is assigned to perform task $t \in \mathcal{Q}_r$, 0 otherwise.

Determine the number of tasks scheduled by a group of resources

- Let $y_g \in \mathbb{Z} \geq \mathbf{0}$ be the number of tasks assigned to resources in $g \in \mathcal{G}$.

Example – Model Documentation

Constraints

Each task must be scheduled

– Each task can only be assigned to be performed on one resource.

$$\sum_{r \in \mathcal{R}} x_{rt} = 1 \qquad \forall t \in \mathcal{T}$$

Example – Model Documentation

Constraints

Each resource has a maximum length of time that they can be actively working on tasks

– Each resource can only be assigned tasks whose total time must be no more than their maximum hours for a day.

$$\sum_{t \in Q_r} \eta_t \cdot x_{rt} \leq \gamma_r \qquad \forall r \in \mathcal{R}$$

Example – Model Documentation

Constraints

Each resource group has a minimum and a maximum number of tasks that they need to do in a day

- Determine the number of tasks assigned to resources in each resource group

$$\sum_{r \in \mathcal{S}_g} \sum_{t \in \mathcal{Q}_r} x_{rt} = y_g \qquad \forall g \in \mathcal{G}$$

Example – Model Documentation

Constraints

Each resource group has a minimum and a maximum number of tasks that they need to do in a day

- Ensure each resource group is scheduled for at least its minimum number of tasks

$$y_g \geq \alpha_g \qquad \forall g \in \mathcal{G}$$

- Ensure each resource group is only scheduled up to its maximum number of tasks

$$y_g \leq \beta_g \qquad \forall g \in \mathcal{G}$$

Example – Model Documentation

Objective

Minimize the total cost of processing all the tasks

– The objective function for the model is to minimize the cost of assigning the resources to tasks.

$$\text{Minimize} \sum_{r \in \mathcal{R}} \sum_{t \in Q_r} \theta_{rt} \cdot x_{rt}$$

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Code Optimization Model

› Data inputs and transformations are documented, created a formulation that we believe solves the problem for the client
  – Now we FINALLY start writing code for the model.

For complicated models, often start with a prototype in a Jupyter Notebook.

- This allows for rapid testing to ensure that the model is built correctly
- Then we can take the prototype and convert it into a class object that can be integrated into whatever system we're building
- Add a single constraint at a time, continuously testing the model

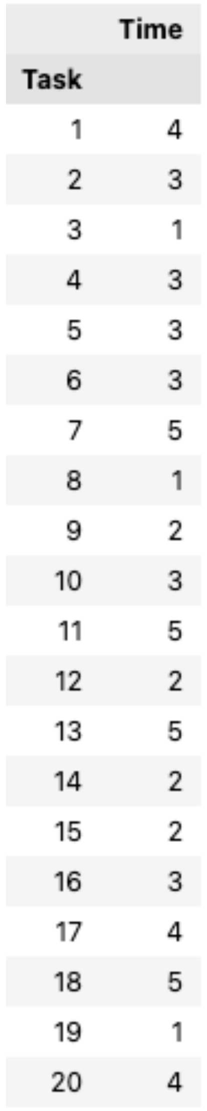
Often, we use a RESTful service endpoint that receives data for an instance of the model.

- This is a wrapper that takes data in and passes that data to the model code
- Believe in separating the model code from the service
- Model can be created and tested without the service
- Allows for more parallel development

Example – Code Optimization Model

- Tasks
  - Data Set
    - Let $\mathcal{T}$ be the set of tasks that need to be completed.
  - Data Inputs
    - Let $\eta_t > 0$ be the number of hours for a task $t \in \mathcal{T}$.

```
tasks = pd.read_excel("../data/data.xlsx",
                      sheet_name="Tasks",
                      index_col=0)
```



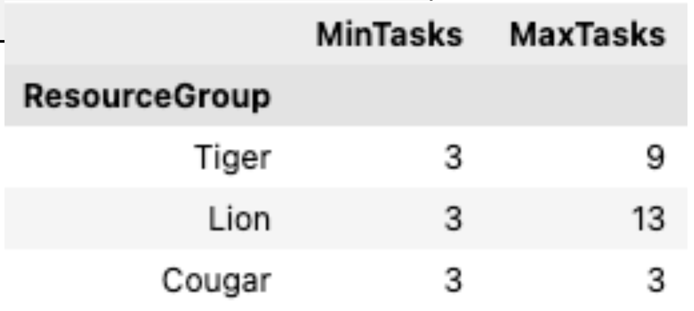
	Time
Task	
1	4
2	3
3	1
4	3
5	3
6	3
7	5
8	1
9	2
10	3
11	5
12	2
13	5
14	2
15	2
16	3
17	4
18	5
19	1
20	4

Example – Code Optimization Model

› Resource Groups

- Data Set
  - › Let $\mathcal{G}$ be the set of resource groups.
- Data Inputs
  - › Let $\alpha_g \geq 0$ be the minimum number of tasks, $t \in \mathcal{T}$, that a resource group $g \in \mathcal{G}$ can work in one day.
  - › Let $\beta_g \geq 0$ be the maximum number of tasks, $t \in \mathcal{T}$, that a resource group $g \in \mathcal{G}$ can work in one day.

```
resource_groups = pd.read_excel("../data/data.xlsx",
                                 sheet_name='Resource Groups’,
                                 index_col=0)
```



	MinTasks	MaxTasks
ResourceGroup		
Tiger	3	9
Lion	3	13
Cougar	3	3

Example – Code Optimization Model

› Resources

– Data Set

› Let $\mathcal{R}$ be the set of resources that can complete the tasks.

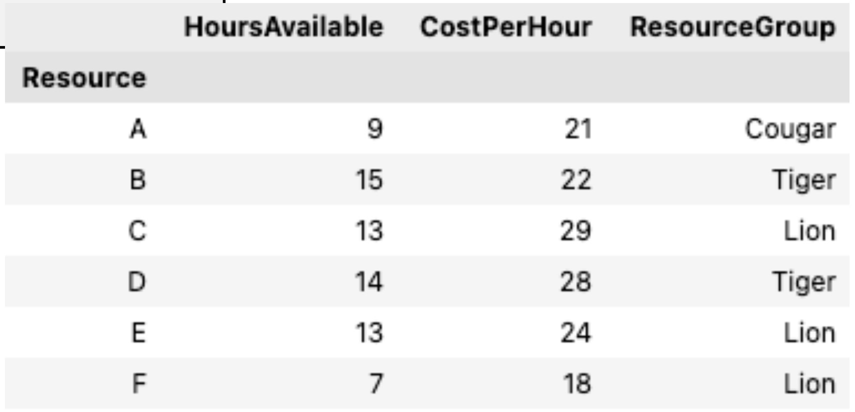
› Let $\mathcal{S}_g$ be the set of resources that belong to resource group $g \in \mathcal{G}$.

– Data Inputs

› Let $\lambda_r \geq 0$ be the cost per hour for a resource $r \in \mathcal{R}$.

› Let $\gamma_r \geq 0$ be the number of hours that a resource $r \in \mathcal{R}$ can work in one day.

```
resources = pd.read_excel("../data/data.xlsx",
                          sheet_name='Resources',
                          index_col=0)
```



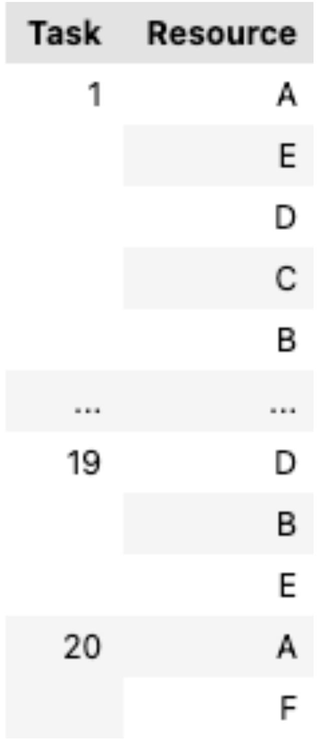
	HoursAvailable	CostPerHour	ResourceGroup
Resource			
A	9	21	Cougar
B	15	22	Tiger
C	13	29	Lion
D	14	28	Tiger
E	13	24	Lion
F	7	18	Lion

Example – Code Optimization Model

› Data Set

- Let $\mathcal{Q}_r$ be the set of tasks that resource $r \in \mathcal{R}$ can perform.
- Let $\mathcal{P}_t$ be the set of resources that task t $\in \mathcal{T}$ can be schedule on.

```
res_task = pd.read_excel("../data/data.xlsx”,
                         sheet_name='Tasks For Resources’,
                         index_col=[0,1])
```



Task	Resource
1	A
	E
	D
	C
	B
...	...
19	D
	B
	E
20	A
	F

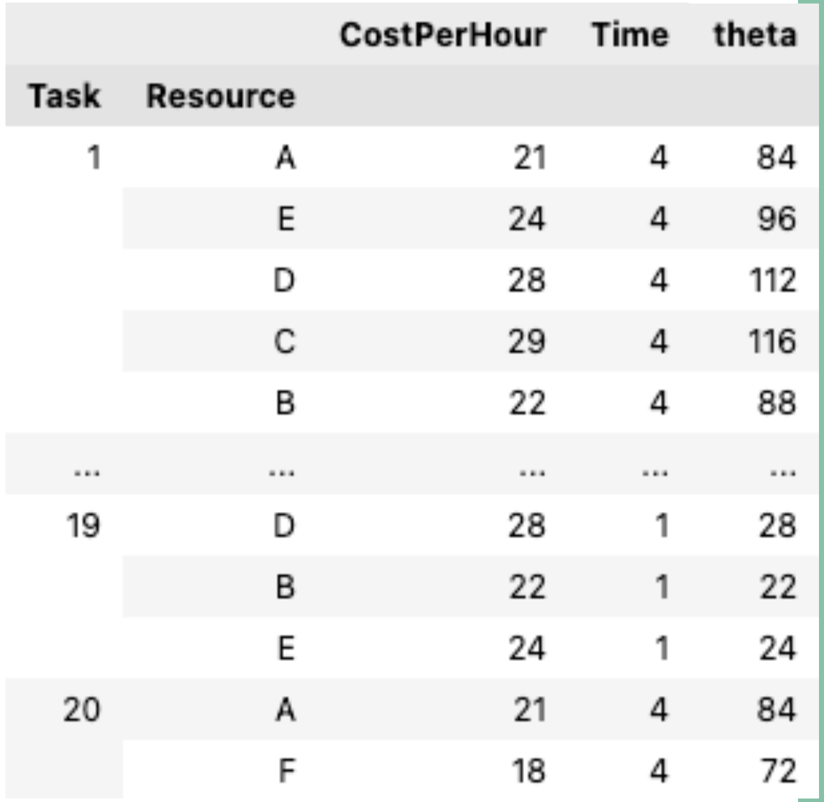
Example – Code Optimization Model

› Data Transformations

– Let θ_{rt} be the cost for a resource $r \in \mathcal{R}$ performing a task $t \in \mathcal{T}$.

$$\theta_{rt} = \eta_t \cdot \lambda_r \qquad \forall r \in \mathcal{R}, t \in Q_r$$

```
theta = (
         res_task
          .merge(resources[["CostPerHour"]],
                 how="left”, left_on="Resource",
                 right_index=True)
          .merge(tasks[["Time"]], how="left",
                 left_on="Task", right_index=True)
          .assign(theta=lambda df:
                        df["CostPerHour"] * df["Time"])
          ["theta"]
)
```



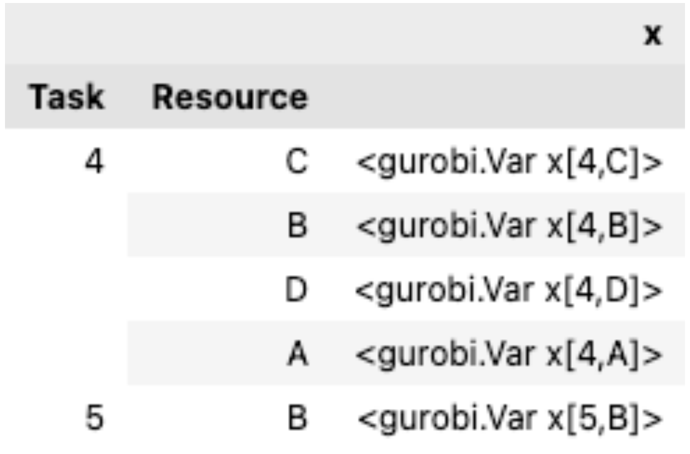
		CostPerHour	Time	theta
Task	Resource			
1	A	21	4	84
	E	24	4	96
	D	28	4	112
	C	29	4	116
	B	22	4	88
...	...	...	...	...
19	D	28	1	28
	B	22	1	22
	E	24	1	24
20	A	21	4	84
	F	18	4	72

Example – Code Optimization Model

› Decision Variables

– Let $x_{rt} \in \{0,1\}$ be 1 if the resource $r \in \mathcal{R}$ is assigned to perform task $t \in \mathcal{Q}_r$, 0 otherwise.

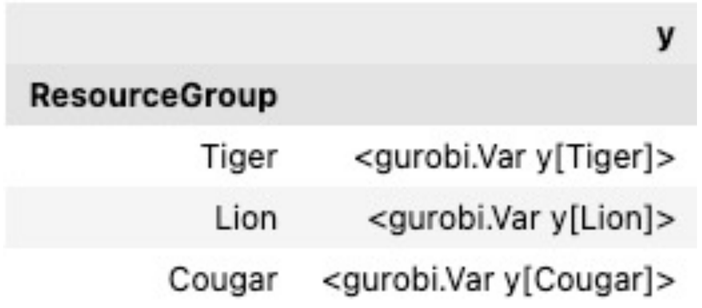
```
x = pd.Series(
        m.addVars(res_task.index,
                  vtype=grb.GRB.BINARY,
                  name="x"),
        index=res_task.index, name="x”
)
```



```
                          x
Task  Resource
   4         C  <gurobi.Var x[4,C]>
             B  <gurobi.Var x[4,B]>
             D  <gurobi.Var x[4,D]>
             A  <gurobi.Var x[4,A]>
   5         B  <gurobi.Var x[5,B]>
```

– Let $y_g \in \mathbb{Z} \geq 0$ be the number of tasks assigned to resources in $g \in \mathcal{G}$.

```
y = pd.Series(
        m.addVars(resource_groups.index,
                  vtype=grb.GRB.INTEGER,
                  name="y"),
        index=resource_groups.index, name="y"
)
```



```
                                  y
ResourceGroup
        Tiger   <gurobi.Var y[Tiger]>
         Lion    <gurobi.Var y[Lion]>
       Cougar  <gurobi.Var y[Cougar]>
```

Example – Code Optimization Model

› Constraints

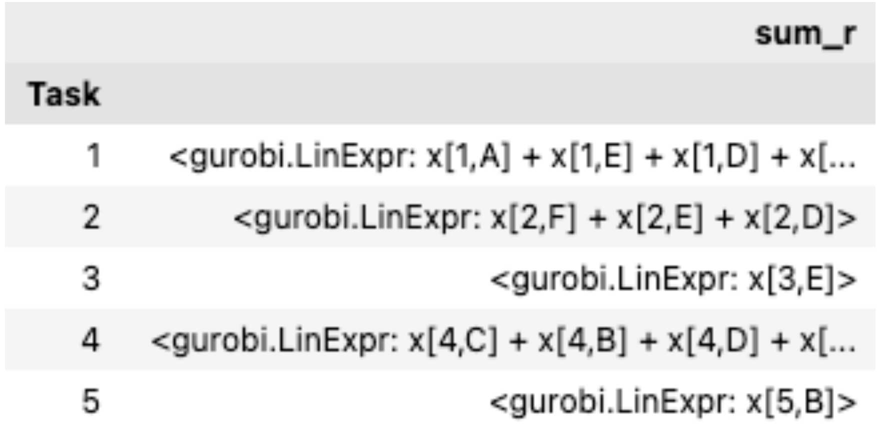
– Assign Each Task to only One Resource

› Each task can only be assigned to be performed one task in the model.

$$\sum_{r \in \mathcal{R}} x_{rt} = 1 \qquad \forall t \in \mathcal{T}$$

```
df = x.groupby('Task').agg(grb.quicksum)
```

```
for ind, sum_r in df.items():
    m.addConstr(sum_r == 1,
                name=namer(
                  "AllTasksMustBeScheduled",
                  ind))
```



	sum_r
Task	
1	<gurobi.LinExpr: x[1,A] + x[1,E] + x[1,D] + x[...
2	<gurobi.LinExpr: x[2,F] + x[2,E] + x[2,D]>
3	<gurobi.LinExpr: x[3,E]>
4	<gurobi.LinExpr: x[4,C] + x[4,B] + x[4,D] + x[...
5	<gurobi.LinExpr: x[5,B]>

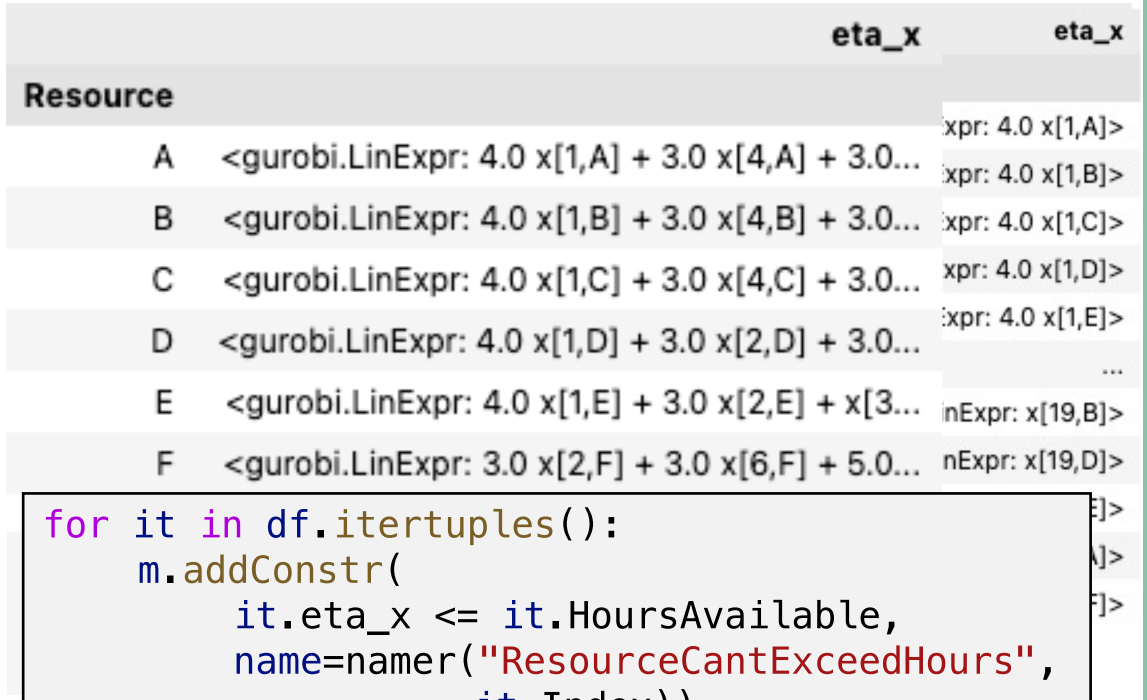
Example – Code Optimization Model

– Ensure Each Resource Stays within their Allowed Hours

› Each resource can only be assigned tasks whose total time must be no more than their maximum hours for a day.

$$\sum_{t \in Q_r} \eta_t \cdot x_{rt} \leq \gamma_r \qquad \forall r \in \mathcal{R}$$

```
df = pd.concat(
        [
         pd.merge(x, tasks["Time"],
                  how="left", left_on="Task",
                  right_index=True)
           .assign(eta_x=lambda df:
                  df["Time"] * df["x"])
           .groupby(["Resource"])
                     ["eta_x"]
           .agg(grb.quicksum),
         resources["HoursAvailable"],
        ],
      axis=1)
```



Resource	eta_x
A	<gurobi.LinExpr: 4.0 x[1,A] + 3.0 x[4,A] + 3.0...
B	<gurobi.LinExpr: 4.0 x[1,B] + 3.0 x[4,B] + 3.0...
C	<gurobi.LinExpr: 4.0 x[1,C] + 3.0 x[4,C] + 3.0...
D	<gurobi.LinExpr: 4.0 x[1,D] + 3.0 x[2,D] + 3.0...
E	<gurobi.LinExpr: 4.0 x[1,E] + 3.0 x[2,E] + x[3...
F	<gurobi.LinExpr: 3.0 x[2,F] + 3.0 x[6,F] + 5.0...

eta_x
xpr: 4.0 x[1,A]>
xpr: 4.0 x[1,B]>
xpr: 4.0 x[1,C]>
xpr: 4.0 x[1,D]>
xpr: 4.0 x[1,E]>
...
nExpr: x[19,B]>
nExpr: x[19,D]>
E]>
A]>
F]>

```
for it in df.itertuples():
    m.addConstr(
        it.eta_x <= it.HoursAvailable,
        name=namer("ResourceCantExceedHours",
                   it.Index))
```

Example – Code Optimization Model

– Count Tasks Assigned to Resource Group

› Determine the number of tasks assigned to resources in each resource group

$$\sum_{r \in \mathcal{S}_g} \sum_{t \in \mathcal{Q}_r} x_{rt} = y_g \qquad \forall g \in \mathcal{G}$$

```
df = pd.concat(
        [
```

```
         pd.merge(x, resources["ResourceGroup"],
                  how="left",
                  left_on="Resource",
                  right_index=True,
                  )
```

```
           .groupby("ResourceGroup")["x"]
           .agg(sum_x=grb.quicksum),
         y,
        ], axis=1)
```

	sum_x	y
ResourceGroup		
Cougar	<gurobi.LinExpr: x[1,A] + x[4,A] + x[6,A] + x[...	<gurobi.Var y[Cougar]>
Lion	<gurobi.LinExpr: x[1,E] + x[1,C] + x[2,F] + x[...	<gurobi.Var y[Lion]>
Tiger	<gurobi.LinExpr: x[1,D] + x[1,B] + x[2,D] + x[...	<gurobi.Var y[Tiger]>

```
for it in df.itertuples():
   m.addConstr(it.sum_x == it.y,
               name=namer(
                  "CountTasksForGroup",
                  it.Index))
```

Example – Code Optimization Model

– Ensure Resource Group Meets Minimum Task Processing

› Ensure each resource group is scheduled for at least its minimum number of tasks

$$y_g \geq \alpha_g \qquad \forall g \in \mathcal{G}$$

– Ensure Resource Group Meets Maximum Task Processing

› Ensure each resource group is only scheduled up to its maximum number of tasks

$$y_g \leq \beta_g \qquad \forall g \in \mathcal{G}$$

```
df = pd.concat([y, resource_groups], axis=1)
```

```
for it in df.itertuples():
    m.addConstr(it.y >= it.MinTasks,
                name=namer("GroupMeetsMinTasks",
                           it.Index))
    m.addConstr(it.y <= it.MaxTasks,
                name=namer("GroupMeetsMaxTasks",
                           it.Index))
```

	y	MinTasks	MaxTasks
ResourceGroup			
Tiger	<gurobi.Var y[Tiger]>	3	9
Lion	<gurobi.Var y[Lion]>	3	13
Cougar	<gurobi.Var y[Cougar]>	3	3

Example – Code Optimization Model

› Objective

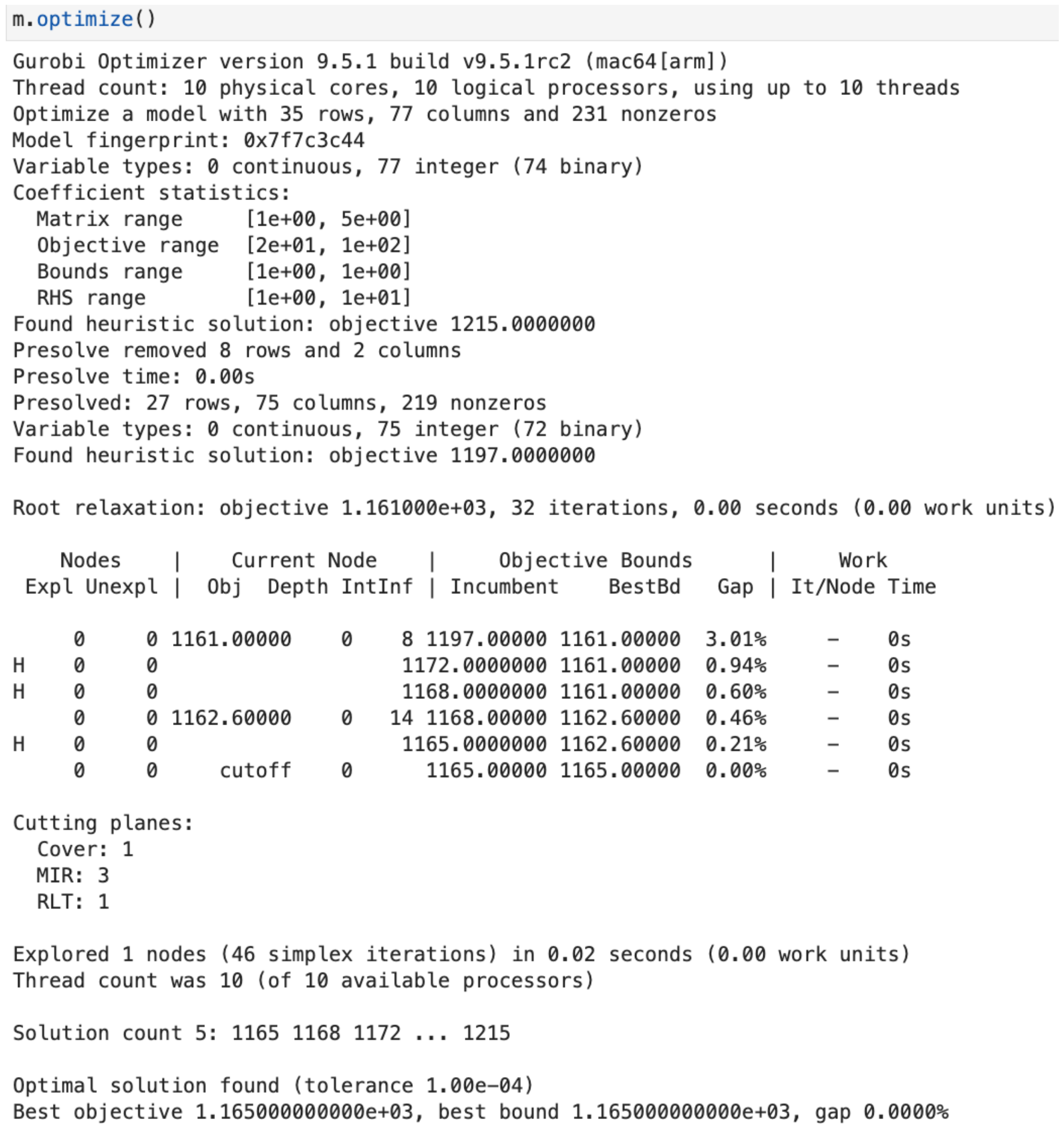
– The objective function for the model is to minimize the cost of assigning the resources to tasks.

$$\text{Minimize} \sum_{r \in \mathcal{R}} \sum_{t \in Q_r} \theta_{rt} \cdot x_{rt}$$

```
z = grb.quicksum(theta * x)
m.setObjective(z, grb.GRB.MINIMIZE)
```

Example – Code Optimization Model

› Run the model with Gurobi



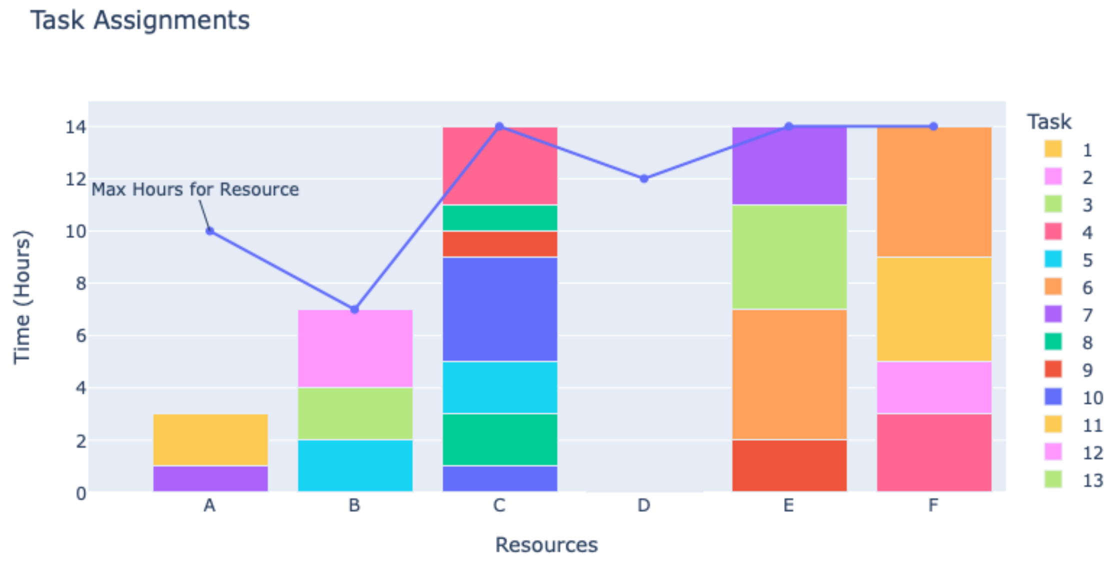
```
m.optimize()
```

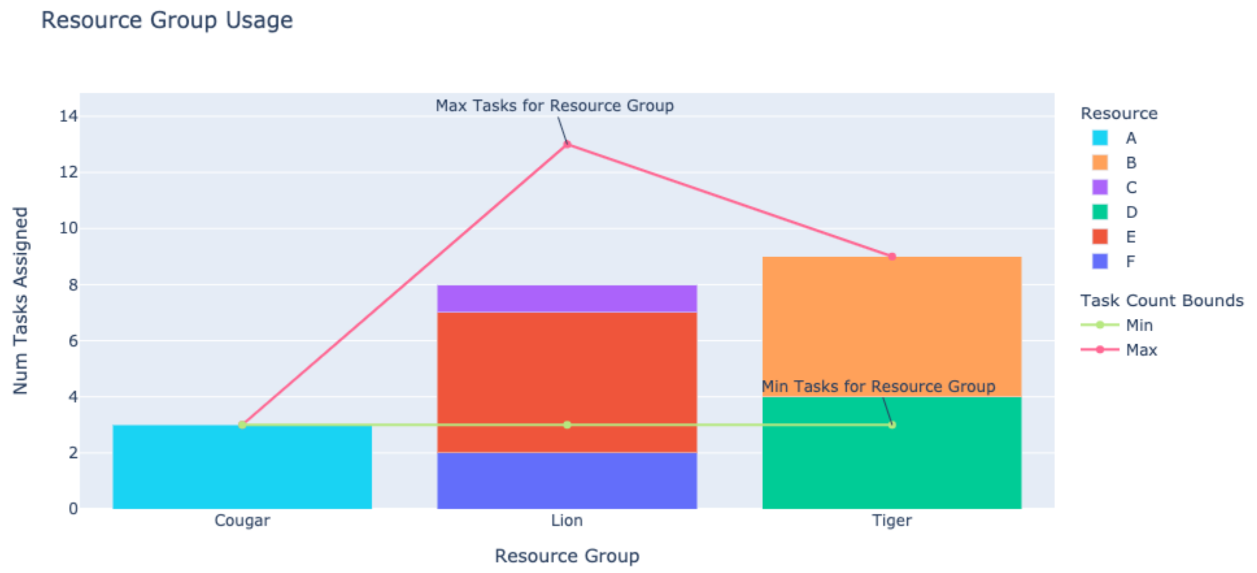
```
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (mac64[arm])
Thread count: 10 physical cores, 10 logical processors, using up to 10 threads
Optimize a model with 35 rows, 77 columns and 231 nonzeros
Model fingerprint: 0x7f7c3c44
Variable types: 0 continuous, 77 integer (74 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+00]
  Objective range  [2e+01, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+01]
Found heuristic solution: objective 1215.0000000
Presolve removed 8 rows and 2 columns
Presolve time: 0.00s
Presolved: 27 rows, 75 columns, 219 nonzeros
Variable types: 0 continuous, 75 integer (72 binary)
Found heuristic solution: objective 1197.0000000

Root relaxation: objective 1.161000e+03, 32 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 1161.00000    0    8 1197.00000 1161.00000  3.01%     -    0s
H    0     0                    1172.0000000 1161.00000  0.94%     -    0s
H    0     0                    1168.0000000 1161.00000  0.60%     -    0s
     0     0 1162.60000    0   14 1168.00000 1162.60000  0.46%     -    0s
H    0     0                    1165.0000000 1162.60000  0.21%     -    0s
     0     0     cutoff    0      1165.00000 1165.00000  0.00%     -    0s

Cutting planes:
  Cover: 1
  MIR: 3
  RLT: 1

Explored 1 nodes (46 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 10 (of 10 available processors)

Solution count 5: 1165 1168 1172 ... 1215

Optimal solution found (tolerance 1.00e-04)
Best objective 1.165000000000e+03, best bound 1.165000000000e+03, gap 0.0000%
```

1. Understand the Data
2. Define and Document Datasets
3. Collect Data from the Client
4. Document Data Transformations
5. Create Model Documentation
6. Code Optimization Model
7. Validate Results and Iterate

Review Results

- For reviewing results, it helps to have a visual representation to enhance our understanding of the results and to review with clients
  - A tool that we like is plotly
- The top plot here shows the assignment of the tasks to resources and how much time each resource is working, with another line for the max time for each resource
- Bottom plot shows the resource group usage for tasks with separate lines for the min and max tasks
- Don't be surprised if the client hates the initial results
  - Most are bad at verbalizing what they really want/need. Therefore, iterating on solutions is very important





Solution Checker

- Written by someone other than the modeler
  - Does not even need to be an optimization person
- Model Solution is output as data frames
- They can use the same data as input data
  - Do not copy, or preferably even look at, code written by the modeler
  - Only use the data and business rules documents should be used
- Validates the constraints and value of the objective function

Development Tools

- Here are some of the tools that we use, and recommend, for developing models
  - Anaconda – Package manager that allows for easy environment setup
  - pandas – Python library for data management
  - Visual Studio Code – Rich IDE with git (and svn) integration, along with many other extensions
    - We use pylance type checking as a tool inside Visual Studio Code to evaluate our python code for potential issues
  - mypy – Type checks code for potential issues
  - pytest – This is a testing harness in python that is straightforward to set up and create tests
  - Docstrings – Tool in python that allows for functions to describe their inputs and outputs as well as the intent of the function
  - Markdown – Used to create data and model documentation
  - Pandoc – Is used to generate PDF documents from the model documentation markdown
  - Jupyter – Used to analyze data and create proofs of concept
- Solver APIs
  - gurobipy – Gurobi library for python development
  - docplex – IBM ILOG CPLEX and CPO library for python model development
  - xpress – FICO Xpress-MP library for python model development
  - ortools – Google OR-Tools library for python model development

Review

1. Understand the Data

2. Define and Document Datasets

3. Collect Data from the Client

4. Document Data Transformations

5. Create Model Documentation

6. Code Optimization Model

7. Validate Results and Iterate

Resources

- Princeton blog post on rapid model development: https://www.princetonoptimization.com/blog/blog/princeton%E2%80%99s-rapid-optimization-model-development-approach-python-and-pandas
- Demo code shown in this presentation: https://github.com/rrandall1471/2022-INFORMS-Demo

Questions