Modeling - Orientation

August 31, 2023

Contents

1 README

- Course overview

- Getting started (infrastructure: GMail, DataCamp, Canvas, GitHub)

- DataCamp workspace practice

- Modeling and simulation frameworks

- Testing the falling Penny myth

- Pint Python library for unit computations

2   Getting started

- Can you see the GMail space "Modeling Chat"?

- Did you get my invitation to join datacamp.com? The courses in the "assignments" section are optional but you will get bonus points (10 per completed course - by end of the term).

- Can you log into lyon.instructure.com and see Canvas?

- Can you see my GitHub repo in github.com/birkenkrahe/mod23?

3   Course overview

Course and instructor overview information. The original version of this lecture is on GitHub. There are changes for technical details.

What has modeling meant to me on my journey?

1. Theoretical particle physics, especially lattice gauge theory: modeling "the universe" on a discrete spacetime-lattice with multigrid algorithms (very similar to neural nets).

2. Interest in complex and dynamical systems (many variables, many degrees of freedom) - domain of 2021 Nobel Prize winner Giorgio Parisi (Complex systems - A Physicist's Viewpoint, 2013) - against reductionism. Don't suppress but simulate the details of living systems (but not by brute force if possible).

3. I'm now once more interested in the interface between machine learning and physics, e.g. applications of the Renormalization Group (Liu, 2017).

What are your expectations for this course?

What will you learn in this course? I am not sure yet.

What I did in 2021: `https://tinyurl.com/57hv47td` focus on "decision intelligence" - quite conceptual, going through different modeling methods, graphical/computational/conceptual.

Python libraries we'll be using:

1. NumPy for numerical computation (`https://www.numpy.org`)

2. SciPy for scientific computation (`https://www.scipy.org`)

3. Matplotlib for visualization (`https://matplotlib.org`)

4. pandas for working with data (`https://pandas.pydata.org`)

5. SymPy for symbolic computation (`https://www.sympy.org`)

6. Pint for units like kilograms and meters (`https://pint.readthedocs.io`)

7. Jupyter for reading, running, and developing code (`https://jupyter.org`)

Grading/schedule: see syllabus in GitHub and in Canvas.

What will you learn in this course? - Different types of models, Python basics, different cases, scientific computing. Discrete systems (changes occur at discrete time instants), e.g. computer systems; queuing systems; network protocols. First order systems that are described by a first-order differential equation. Examples include: electrical circuits (voltage changes in response to a step input); thermal systems (temp changes in response to heating/cooling); mechanical systems (velocity of mass changes due to an applied force - e.g. car suspension system). Epidemiological models.

How will you be evaluated? - 25% each for 15 weekly assignments, monthly sprint reviews, weekly tests and one final exam.

Which tools are we going to use? - Canvas, GitHub, DataCamp workspace, bpmn.io for diagrammatic modeling purposes.

Textbooks? Free online book by Alan Downey accompanied by Colab notebooks. Quite densely written and coded (own library), needs unpacking. I'll be working through the book with you.

Infinite skills exercise: come up with at least one model you would create if you had infinite programming skills and if you could build anything you wanted using any computer and any programming language.

First assignment: read the article by Parisi (2013) and summarize it. Be prepared to present your summary.

Next: using the DataCamp coding platform.

4   Introducing DataCamp workspace

Our integrated development and interactive notebook environment is DataCamp workspace at workspace.datacamp.com. For more Python platforms,

see the GitHub practice file: Command line, IDLE, Google Colaboratory, Kaggle, replit.com, etc.

DataCamp workspace has a notebook interface with an IPython shell, a file manager, text cells with Markdown, auto-completion and many pre-installed packages. There is (free, for you) access to a Linux terminal, AI-assistance, and co-coding.

More information about DataCamp workspace: tinyurl.com/bdzfpkzh

Can you think of any reasons not to make it too convenient to develop, test and execute your programs? (Sounds crazy, right?) Is development and analysis speed the only goal?

> *Answer:* the notebook and the graphical UI are additional levels between you and the machine. This is very convenient for quick exploration, but you don't learn much about the internals and the infrastructure. The problem with that is that infrastructure changes often and has a strong impact on performance - infrastructure knowledge is quickly becoming a secret science, and only the initiated have access.

> How do you feel about AI-assisted coding?

> How I feel about AI-assisted coding: I noticed the dementia-inducing effect that it has on me as an expert but I don't know if it might help you learn faster or more broadly, or not. When you have access to an AI, it is important to know what you can use it for, and to resist its allure continuously so that you don't become dependent. This could easily be said for any

5   Workspace demo - setup

In DataCamp, open the `workspace` tab at the top to get to the workspace overview. You can also open this link to get directly to the workbook: tinyurl.com/WorkspaceDemoPython.

If you're in the overview, take a look around: You have access to all shared workspaces, and you can limit the view to your own. You can view bookmarked workbooks (favorites). There is also a menu for "Code Alongs". Open `DataCamp Python Demo (problem)`.

Click on `Make Copy` to copy the workbook - rename the workspace to reflect your ownership, and save it to the `Account` "Lyon College Data Science Fall 2023".

Go to the `Workspace overview` by clicking the symbol at the top left of your dashboard. You should now see your own workbook there. While you could only comment on my workbook , you can edit and run this one.

If you do leave a comment, I will be notified via GMail and will respond as soon as I see the email and find the time.

Dashboard

Our target data is the "unicorn company" dataset - we're going to analyse the data of companies with a valuation $>$ USD 1 bn.

Get the CSV file here: tinyurl.com/unicornCompaniesCSV

The workspace has two main areas:

1. Left sidebar for work environment

2. Text, code and output cells or blocks in the center. Text cells can be edited, commented upon, AI-assisted, or deleted. Code cells can be run, commented upon, AI-assised, or deleted.

3. There are some extra choices at the top:

   - `View` $>$ `Switch to JupyerLab` opens a launcher for a bunch of different apps. You'll see a more traditional view of your notebook. You can add tabs to get to a console, a notebook, a terminal etc.

   - `Run` $>$ `Open Terminal` (CTRL-.) opens a terminal or command line interface (CLI) to enter commands for the shell. You can also enter some from within the notebook but this is much more convenient when you want to muck around with files.

The purpose of the notebook format is that you can build a data report as you go along, including any idea or input, any code (in Python), and any output generated by your code.

Finished notebooks can be published to registered DataCamp users only. To publish to a larger audience, you need to use Kaggle or Google Colaboratory, or another platform.

You can always download your workbook = notebook + files to a with `File` $>$ `Download`. Don't try this on Chromebook.

Within data science (including AI, machine learning, data analysis) this interactive notebook format is the gold standard for data storytelling - developing and presenting data-driven computational insights to a human audience.

Jupyter notebook (`.ipynb` files) are an open source standard so there is no lock-in: you can import and export notebooks to and from this platform, and if you lose access, no big deal. You can e.g. download and use a free, offline version of "Jupyter Lab" to your PC or work in another online environment.

Code along notebook

To begin, you should have an editable copy of my workspace in your personal workspace: tinyurl.com/WorkspaceDemoPython.

The practice file's text is complete but all code chunks are missing and you will have to add them as well as text blocks where needed.

The demo involves:

1. Explaining how this works

2. Explaining the data set

3. Importing CSV data as a pandas data frame (a data table)

4. Viewing the unique values of company categories

5. Cleaning the data frame column for company categories

6. Grouping all records (rows) by industry category

7. Plotting the number of unicorn companies by industry category

The code covers much of what you'll learn in this class. Don't get discouraged if you cannot follow in detail. Let it be a lesson and a motivation.

A live solution of the workbook is available here: tinyurl.com/WorkspaceDemoPythonSolution. The published notebook is available, too: tinyurl.com/WorkspaceDemoPublic.

Understanding the sidebar

Open the `Files` menu in the sidebar: you see the notebook (open) and the CSV file.

Click on the three dots next to name of the CSV file to see different options.

The option `Query in new SQL cell` opens a new code cell (at the very end of the notebook) with a SQL query command on all features (columns) of the CSV file. To execute this command, the CSV data are converted to a dataframe first.

Create the SQL cell and run it, then press CTRL-Z twice to get back to the original notebook. You don't have to test the other option, `Load as DataFrame` because we're going to do this explicitly. But if you wanted to, this would create a Python cell with the commands to import the CSV data as a DataFrame.

Click on the CSV file `unicorn_companies.csv` to open it.

You see a headline with several features and 917 records of these features, one for each unicorn company. This is what is called 'raw' data: in a Comma-Separated-Values (CSV) file, all values are separated by commas. The first line is special: it contains the headers, the names for the different columns.

Importing a CSV file as a pandas `DataFrame`

Get back to your notebook. Next to the CSV file, select `Copy path to clipboard`. Click on `Files` to close the menu. Now all you see is the (minimized) sidebar and the notebook.

Get the CSV file here: tinyurl.com/unicornCompaniesCSV

```
# import pandas
import pandas as pd
# read CSV file
df = pd.read_csv('unicorn_companies.csv')
# show data frame
df
```

When you run this cell, either with the mouse or by entering CTRL-ENTER, the first 10 records of the DataFrame `df` and the headline with the features. You can also download the CSV dataset from here, and try to create a chart - better wait with that until you understand the data set better.

Though the data look quite clean and appealing, a table view is not the best way to get an overview - there are many records.

Viewing `unique` column (`pd.Series`) data

For investment purposes, the `Category` column or feature is most interesting: this is the type of company. How many of these types are there?

To print out all unique categories, we can use the `unique` function, which will return all unique entries in the `Category` column if we index the data frame accordingly:

```
help(pd.unique)
```

There's a lot of information in this helpfile. You can look for help using
`?` or the `help` function:

```
?pd.unique
help(pd.unique)
```

Testing the AI coding assistant

This is a good place to show off your `AI` assistant: you may not know how
to look for help for `unique`. Entering `help(unique)` or `?unique` will give an
useless (to the beginner) error message: `Object 'unique' not found.`

Add an `AI` code block. The assistant will ask you for a prompt. For simple
questions like these, almost any prompt will do, e.g. `I need help for the`
`function 'unique'.` The marks around `unique` will help the computer
understand that you mean a command (these marks are also used for coding
font markdown in text blocks).

The information given by the AI is pretty exhausting and does not quite
fit our problem - the issue is our prompt. Below the block you find another
input field `Tell our AI what to do...` - Enter another prompt:

```
I need the docstring for the function 'pd.unique'.
```

This time, we get a better but still quite verbose answer in a code block that
is automatically executed.

We only want a short explanation that an absolute beginner can under-
stand. Let's ask for that directly:

> As an absolute beginner in Python, I need a very short explana-
> tion of what 'pd.unique' does and how I can use it on a column
> of a data frame.

Let's apply this knowledge to the `'Category'` column but instead of using
the functional notation `pd.unique(series)`, let's use the dot operator:

```
df["Category"].unique()
```

To test the AI yet again, remove the parentheses after the function call
to `unique`. This yields an error. At the bottom of the output, you can click
on `Fix & explain`.

The first part of the AI response is correct - the parentheses are reconstituted. But then a `NameError` is unnecessarily generated because the AI does not have access to the Python environment, which includes the user-defined data frame `df`. To correct this, you need to re-run the respective code and re-run this block thereafter!

These experiments show that we're still quite far away from getting fully relieved of our coding burdens. This was (much) more work than necessary. A simple Google search ("Explain pd.unique in Python") yields a quicker and better answer:

> "The unique function in pandas is used to find the unique values from a series. A series is a single column of a data frame. We can use the unique function on any possible set of elements in Python. It can be used on a series of strings, integers, tuples, or mixed elements."

Back to viewing the unique 'Category' values

To remove the extraneous information about data types in the printout (`array`) and print the list one item per line, you can also use a for loop or a *list comprehension*:

```
# Print out all categories - one per line
for category in df['Category'].unique():
  print(category)
# With a list comprehensionN
[print(i) for i in df["Category"].unique()];
```

Here, we generate a new line with `print` for every unique record of the column. The semi-colon at the end stops a bunch of `None` values to be printed afterwards (an IPython artefact).

You can see that there are duplicates because of typos (`Finttech`) and capitalization (`Artificial Intelligence`). Let's remove the ambiguities.

Clean data frame column `Category`

We can use `df.replace` to replace one value by another value inside our dataframe. We do not need to repeat the command but we can append methods to one another:

```
df_clean = df.replace(to_replace='Artificial intelligence',
                      value='Artificial Intelligence')\
          .replace(to_replace='Finttech',
                      value='Fintech')
```

Share editing rights

One of the neater properties of DataCamp Workspace is the ability to share your notebook and edit synchronously like in GoogleDocs.

Click on the sharing sign at the top and share **editing** access with your neighbor by using his/her email. Also, reduce "General access" to "Disable access" - now nobody except those you invite via email can see your file.

You have to use the person's email used for DataCamp - make sure it's their Lyon College email. Once they've been invited, you can let them access to edit, view, comment or remove their access.

Print the new dataframe `df_clean` in each other's notebooks by adding a new code block with the command `df_clean`.

Once this is done, `Remove` access from your workspace for the other person.

Grouping data by column values

To find out how many unicorn companies are there in each `Category` (aka industry), we group the corresponding records using the function `pd.DataFrame.groupby`. The command in the code cell below performs several operations on the `df_clean` dataframe:

We use three functions: `df.groupby()` on the `Category` column (Chat-GPT summary), `size` to extract the number of records in each group, and `sort_values` to sort the result in descending order:

```
category_counts =\
    df_clean.groupby(by = 'Category', as_index=False)\
```

```
            .size()\
            .sort_values(by=['size'])
```

groupby(by = 'Category', as_index = False): This groups the dataframe by the 'Category' column. The as_index = False parameter ensures that the resulting groups retain 'Category' as a column rather than using it as an index.

size(): After grouping, this function is used to compute the size of each group. In the context of groupby, the size() function returns a pd.Series (a vector or 1-dim array) with the number of items in each group. This is essentially a count of rows for each 'Category'.

.sort_values(by=['size']): This sorts the resulting pd.Series based on the size/count.

Now, when you use the size() function with groupby, the resulting pd.Series will have the counts of each group as its values. When you sort this and convert it back into a dataframe (which happens implicitly because of as_index=False), the counts become a new column. By default, this column is named size – hence the creation of a new column named size in the output.

The result, category_counts, is a pandas data frame with two columns sorted by size of group rather than alphabetically. When you let Colab suggest a graph, you get a line plot, a histogram (distribution) and a time series. type returns the data structure of its argument, and pd.DataFrame.shape is an attribute of the dataframe that contains its dimensions.

```
# show the data type of category_counts
print(type(category_counts))
# show the dimension of category counts
print(category_counts.shape)
```

Plotting data

The result, category_counts, is a pandas data frame with two columns sorted by size of group rather than alphabetically. When you let Colab suggest a graph, you get a line plot, a histogram (distribution) and a time series. type returns the data structure of its argument, and pd.DataFrame.shape is an attribute of the dataframe that contains its dimensions.

There are many different graphics packages available. The one most often mentioned is matplotlib. It is a great package to get a quick overview but you usually need to customize the graphs quite a bit before they look publishable.

Instead, we use the `plotly` package, which has an express module that does most of the heavy lifting for us. All it needs is the data and the names of the x and y column, and a title:

```
# import plotly.express
import plotly.express as px
# Create a bar plot of category group size vs. category
px.bar(category_counts,
    x = 'Category',
    y = 'size',
    title='Unicorn company distribution across industries')
```

`plotly` is a plotting library, and `plotly.express` is a module to provide a range of plot types quickly (ChatGPT help and online doc).

Compare the result when using `matplotlib.pyplot`: instead of one line, we need several lines of code to get a similarly appealing result. However, as I said, for quick data exploration, this is the way to go.

```
# import matplotlib.pyplot
import matplotlib.pyplot as plt
# plot category group size vs. Category
plt.bar(category_counts['Category'],
        category_counts['size'])
# rotate the x ticks by 90 degrees to make them readable
plt.xticks(rotation=-90)
# add a title
plt.title("Unicorn company distribution across industries")
# label the y-axis
plt.ylabel('Frequency')
# draw a grid to increase readability
plt.grid()
# show the final plot
plt.show()
```

6 Workspace - Summary (exercise)

- Workspace offers Jupyter notebooks in Python, R and SQL.

- WS Notebooks contain text, code, output ("literate programming").

- WS Notebooks have pre-installed libraries and sample data

- WS notebooks run an IPython shell

- WS notebooks can be downloaded/uploaded as `.ipynb` files

- WS notebooks can be shared with other [DataCamp] users

- WS notebooks can be published to [DataCamp] portfolios

7   References

CB Insights. The Complete List of Unicorn Companies. CB Insights. Published 2023. Accessed August 19, 2023. `https://www.cbinsights.com/research-unicorn-companies`

Downey AB. Modeling and Simulation in Python. NoStarch Press; 2023. `https://allendowney.github.io/ModSimPy/`

Giordano FR, Fox WP,Horton SB. A First Course in Mathematical Modeling (5e). Cengage Learning 2013.

Google LLC. Google Colaboratory. Accessed August 19, 2023. `https://colab.research.google.com`

Liu, Z (2017). Machine Learning and the Renormalization Group. `https://tinyurl.com/57nyk3y7`

Parisi G (2013). Complex Systems: A Physicist's Viewpoint. `https://arxiv.org/pdf/cond-mat/0205297.pdf`

Pérez F, Granger BE. IPython (Version 8.14.0). IPython Development Team. Published 2023. Accessed August 19, 2023. `https://ipython.org`

Python Software Foundation. Python (Version 3.8.10). Python Software Foundation. Published 2021. Accessed August 19, 2023. `https://www.python.org`

Schouwenaars F, Cotton R. Unicorn companies. DataCamp. Published 2022. Accessed August 19, 2023. `http://bit.ly/ws-unicorn`