Jupyter Notebooks in Dyalog APL

Adám Brudzewsky

What are notebooks?

A *notebook* combines the functionality of

a word processor — handles formatted text

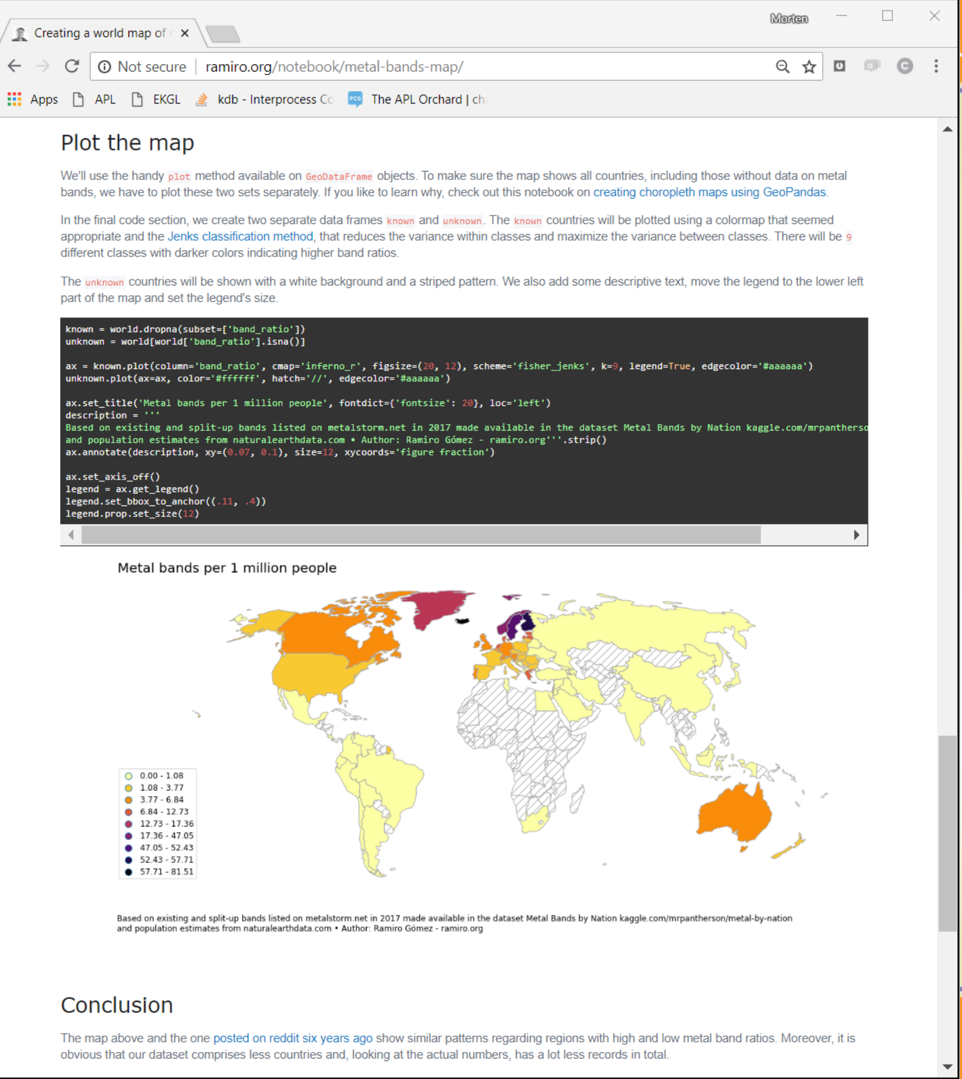
a "shell" or "kernel" — executes statements in a programming language and includes output inline

a rendering engine — renders HTML in addition to plain text

Example notebook

using Python

global density
of metal bands



Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out this notebook on creating choropleth maps using GeoPandas.

In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

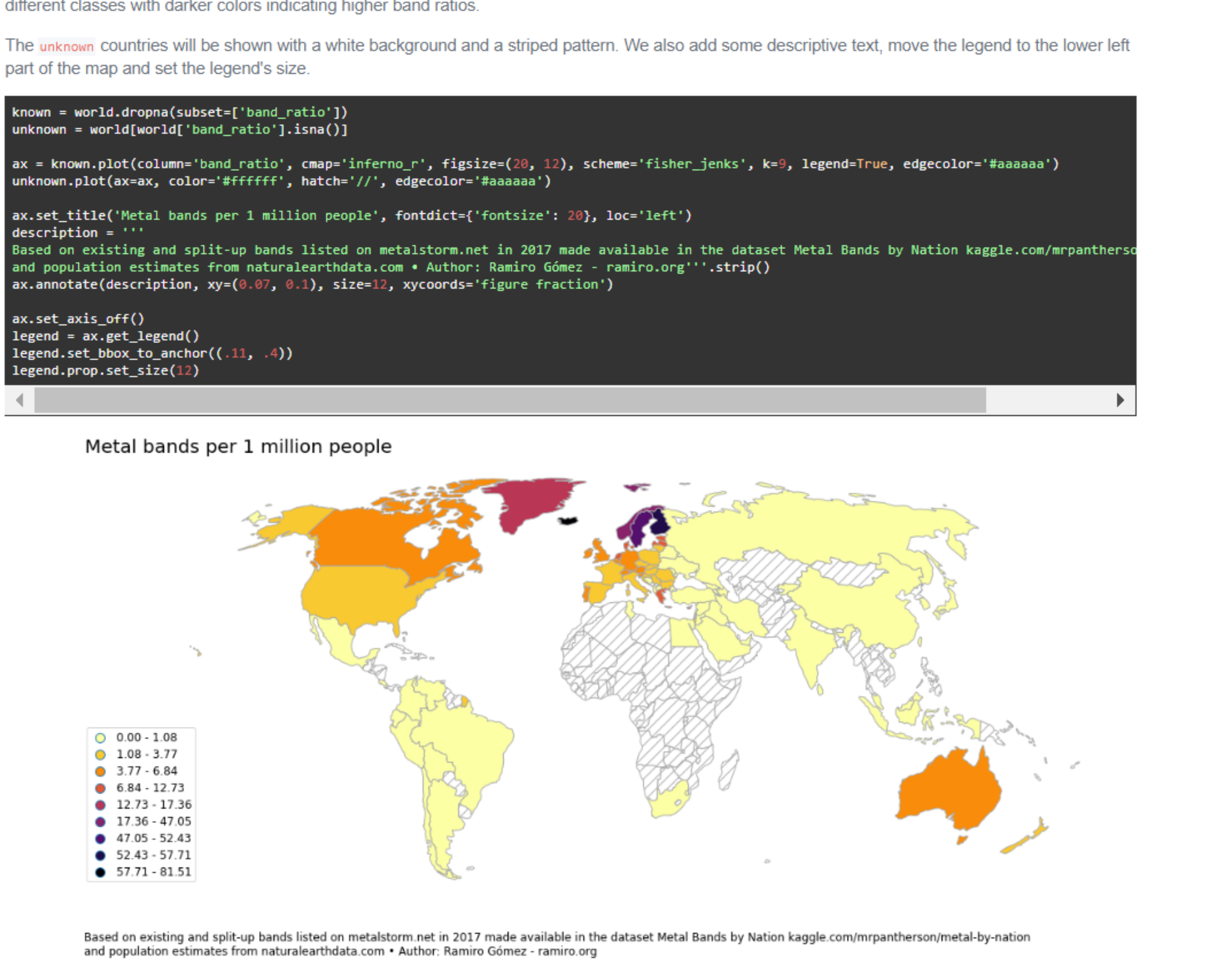
```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people

Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherson/metal-by-nation and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org

Conclusion

The map above and the one posted on reddit six years ago show similar patterns regarding regions with high and low metal band ratios. Moreover, it is obvious that our dataset comprises less countries and, looking at the actual numbers, has a lot less records in total.



different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people

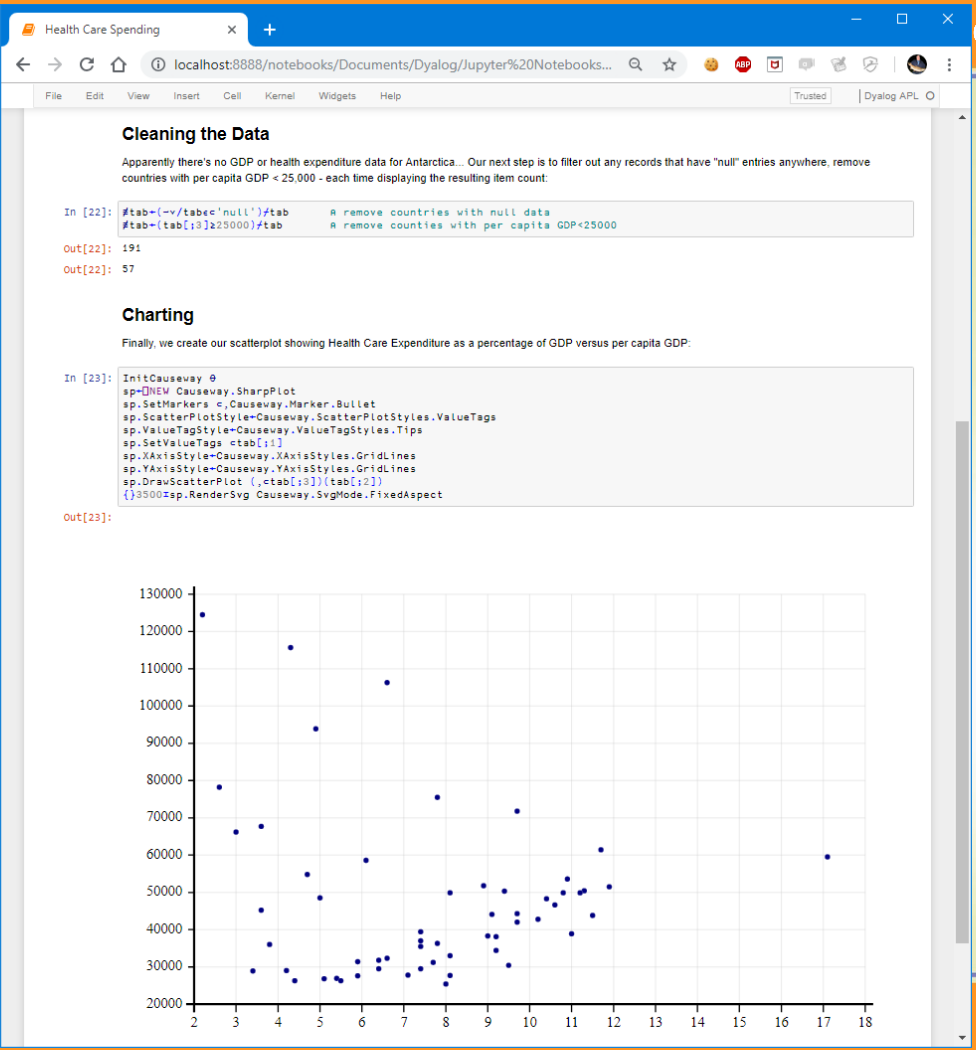
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherson/metal-by-nation
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org

notebook
on

sity
ands

Example notebook using Dyalog APL

health care expenditure vs GDP per capita



Cleaning the Data

Apparently there's no GDP or health expenditure data for Antarctica... Our next step is to filter out any records that have "null" entries anywhere, remove countries with per capita GDP < 25,000 - each time displaying the resulting item count:

```
In [22]: ≢tab←(~∨/tab∊⊂'null')⌿tab      ⍝ remove countries with null data
         ≢tab←(tab[;3]≥25000)⌿tab       ⍝ remove counties with per capita GDP<25000

Out[22]: 191

Out[22]: 57
```

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

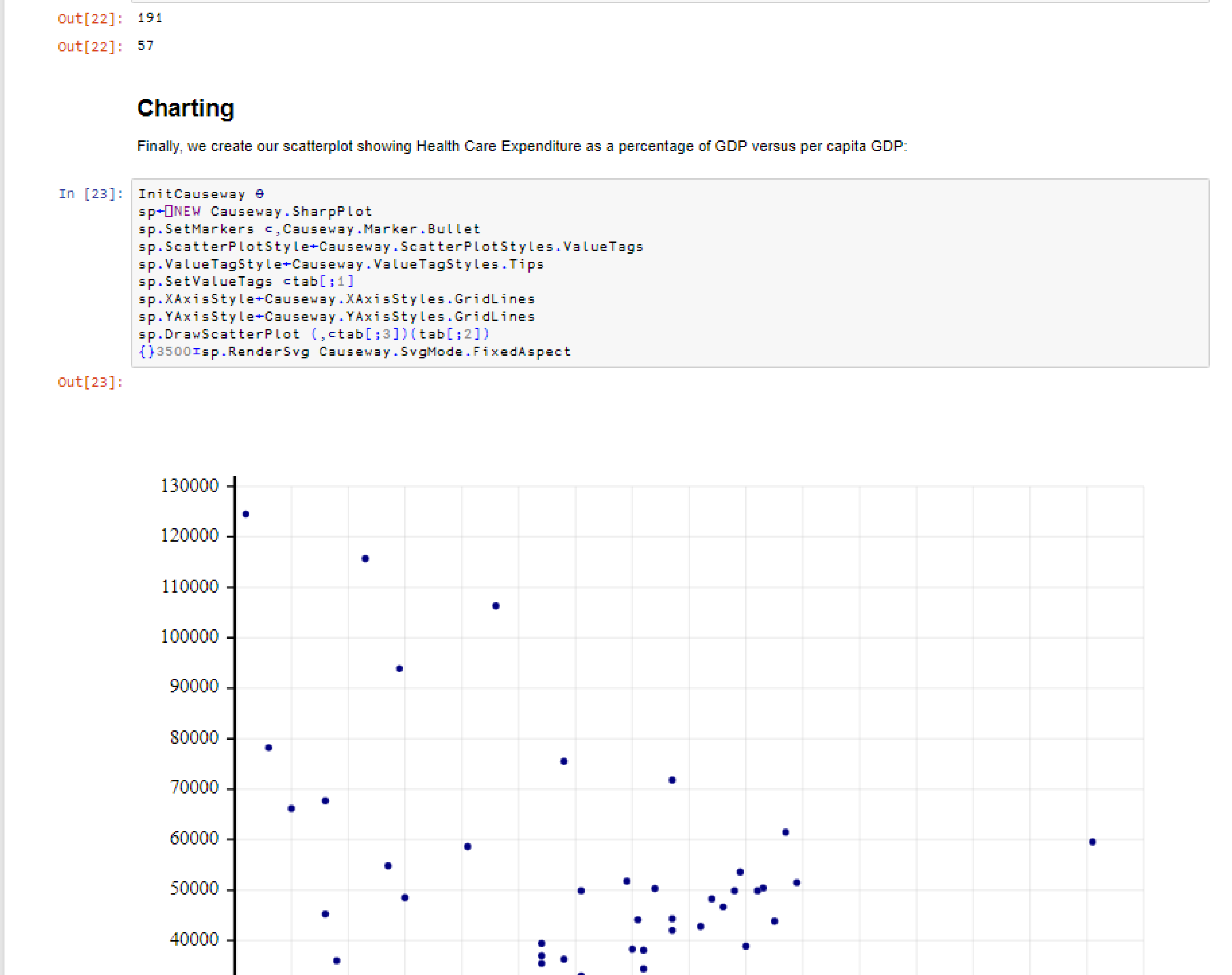
```
In [23]: InitCauseway ⍬
         sp←⎕NEW Causeway.SharpPlot
         sp.SetMarkers ⊂,Causeway.Marker.Bullet
         sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
         sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
         sp.SetValueTags ⊂tab[;1]
         sp.XAxisStyle←Causeway.XAxisStyles.GridLines
         sp.YAxisStyle←Causeway.YAxisStyles.GridLines
         sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
         {}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect

Out[23]:
```

notebook

g APL

expenditure

capita



```
Out[22]: 191
Out[22]: 57
```

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

```
In [23]: InitCauseway θ
         sp←⎕NEW Causeway.SharpPlot
         sp.SetMarkers ⊂,Causeway.Marker.Bullet
         sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
         sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
         sp.SetValueTags ⊂tab[;1]
         sp.XAxisStyle←Causeway.XAxisStyles.GridLines
         sp.YAxisStyle←Causeway.YAxisStyles.GridLines
         sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
         {}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect
Out[23]:
```

Notebook benefits

A single document that combines explanations with executable code and its output — an ideal way to provide:

- reproducible research results
- documentation of processes
- instructions
- tutorials and training materials of all shapes and sizes

A digital learning environment for computational thinking

What is *Jupyter* notebook?

First notebook: Mathematica 1.0 in '88

Jupyter notebook is a part of

Project Jupyter, a nonprofit to

> *develop open-source software, standards, and services for interactive computing across dozens of programming languages*

beginning with Julia, Python, R, and now over 70 languages, including Dyalog APL

What is *Jupyter* notebook?

First notebook: Mathematica 1.0 in '88

Jupyter notebook is a part of Project Jupyter, a nonprofit to

> *develop open-source software, standards, and services for interactive computing across dozens of programming languages*

beginning with Julia, Python, R, and now over 70 languages, including Dyalog APL

ONE OF THE MOST SIGNIFICANT ADVANCES IN THE SCIENTIFIC COMPUTING ARENA

UNIVERSITY CORPORATION FOR ATMOSPHERIC RESEARCH

Ways to use Jupyter notebooks

On your own computer after installing a Jupyter notebook server

With an online notebook server like cocalc.com

Save notebook with output and use a notebook viewer

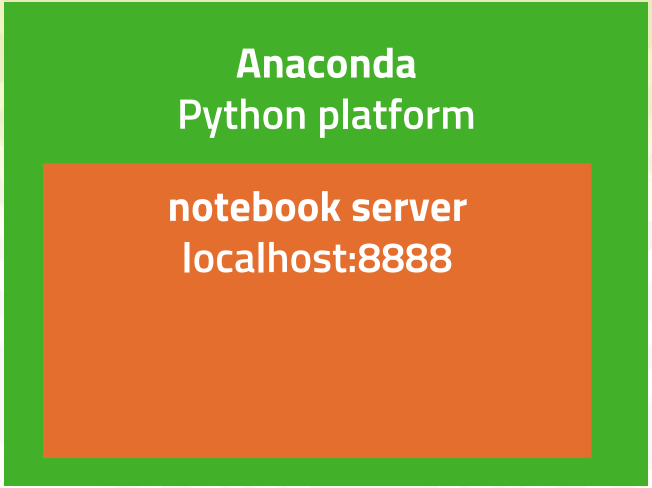
Export to HTML, PDF, LaTeX, etc.

Local notebook server — Python

Local notebook server — Python

Anaconda
Python platform

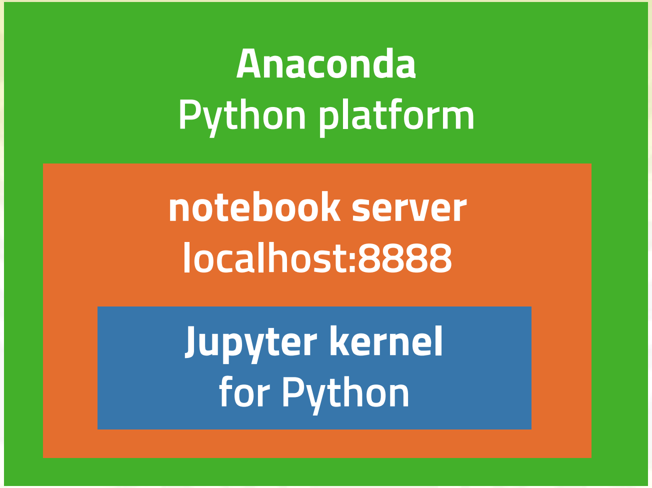
Local notebook server — Python



Anaconda
Python platform

notebook server
localhost:8888

Local notebook server — Python

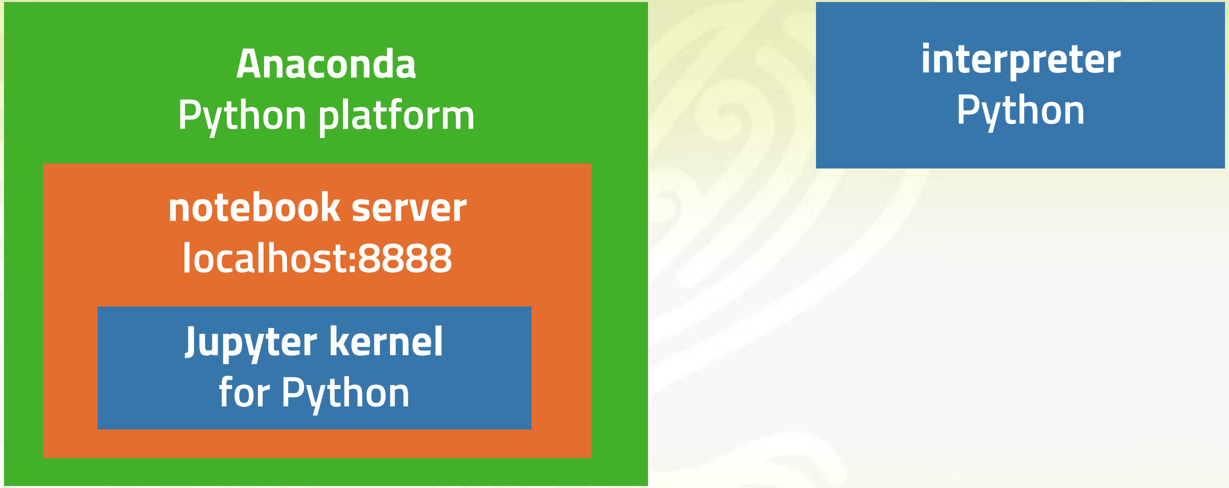


Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Python

Local notebook server — Python



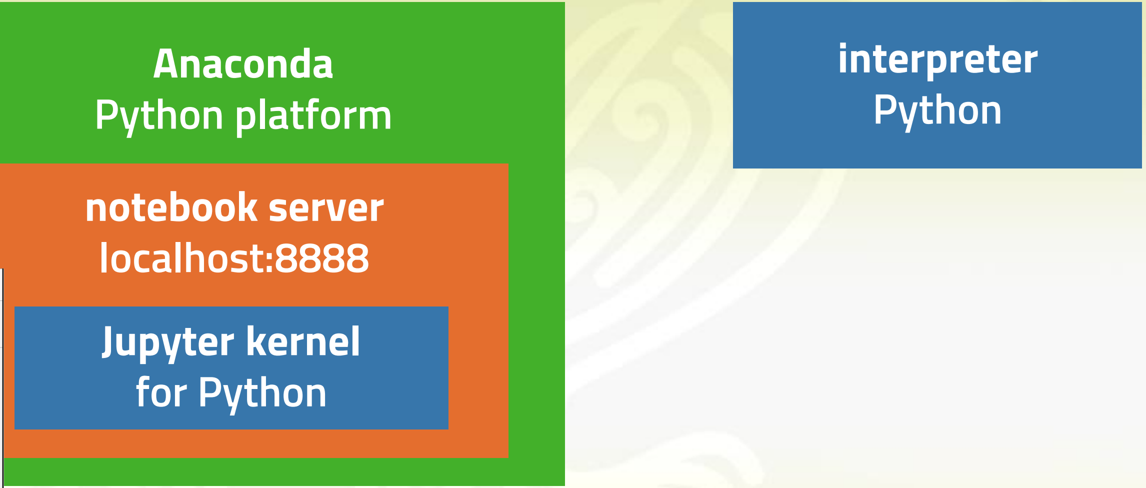
Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Python

interpreter
Python

Local notebook server — Python



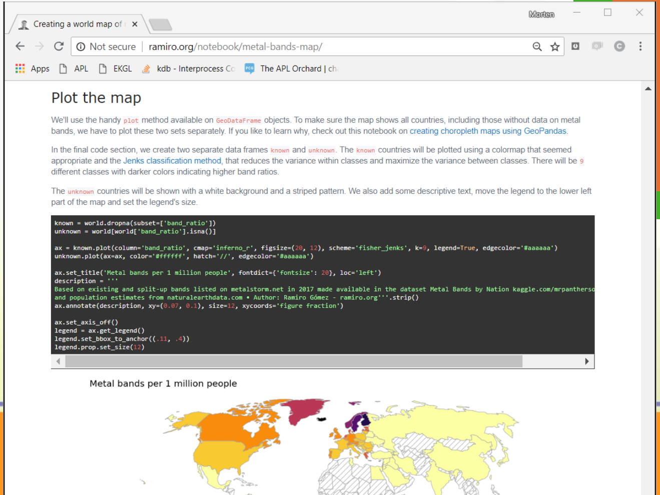
Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Python

interpreter
Python

web browser



Plot the map

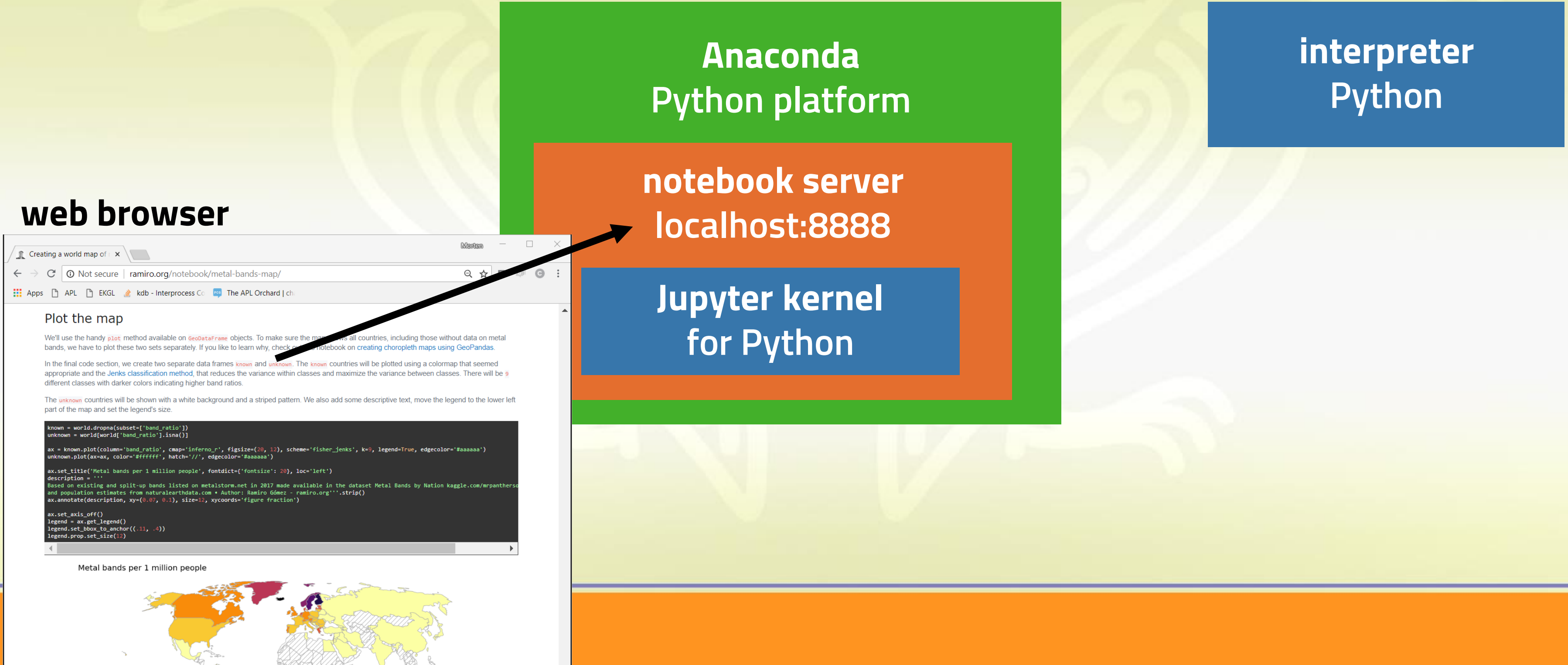
We'll use the handy plot method available on GeoDataFrame objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out this notebook on creating choropleth maps using GeoPandas.

In the final code section, we create two separate data frames known and unknown. The known countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be 9 different classes with darker colors indicating higher band ratios.

The unknown countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Local notebook server — Python



Plot the map

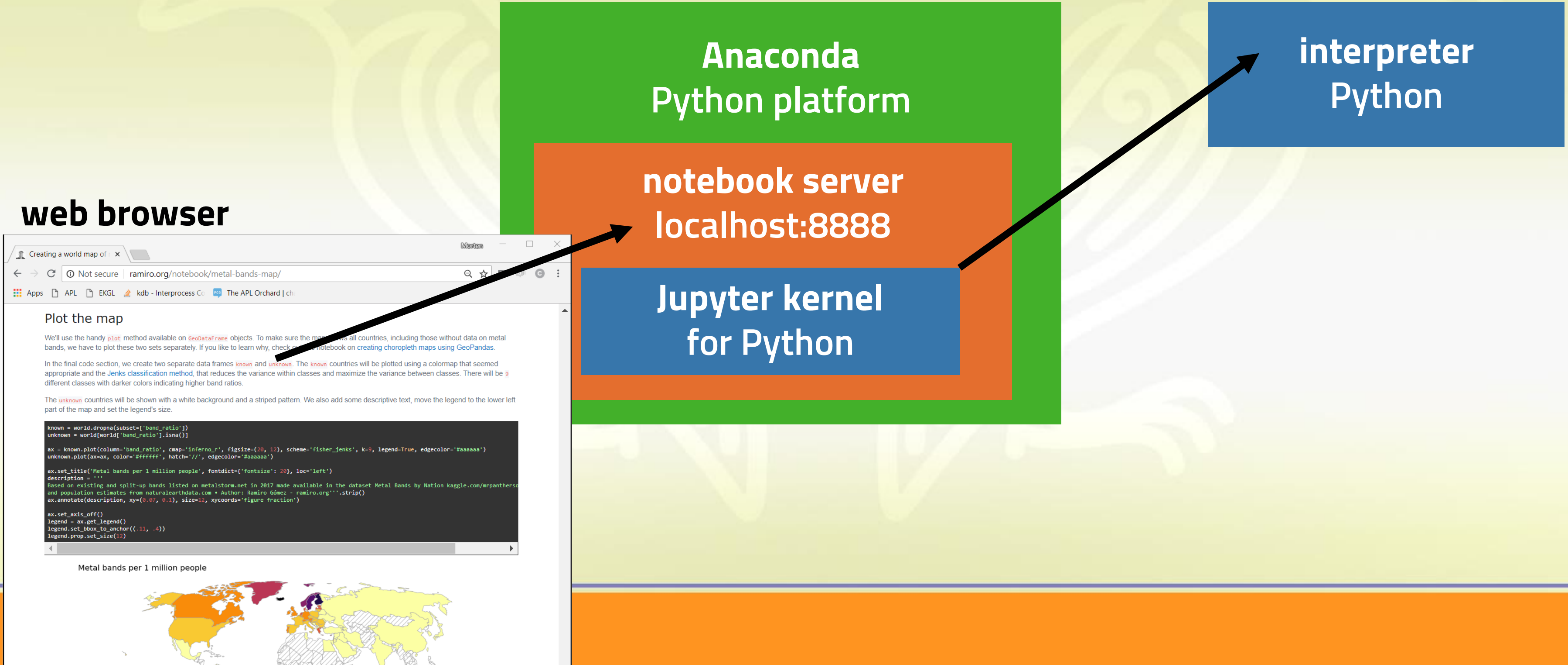
We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out the notebook on creating choropleth maps using GeoPandas.

In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Local notebook server — Python



Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the ma[illegible]ws all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check o[illegible] notebook on creating choropleth maps using GeoPandas.

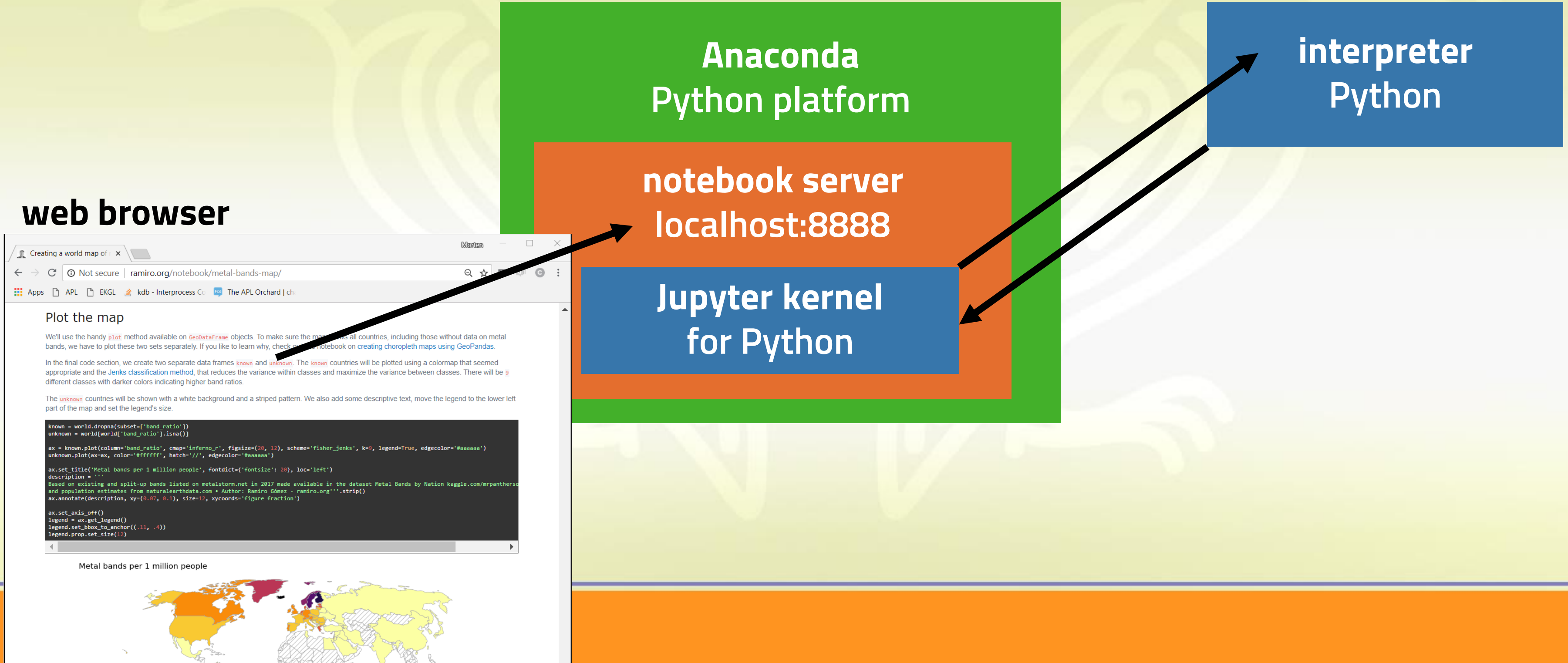
In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people

Local notebook server — Python



Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out this notebook on creating choropleth maps using GeoPandas.

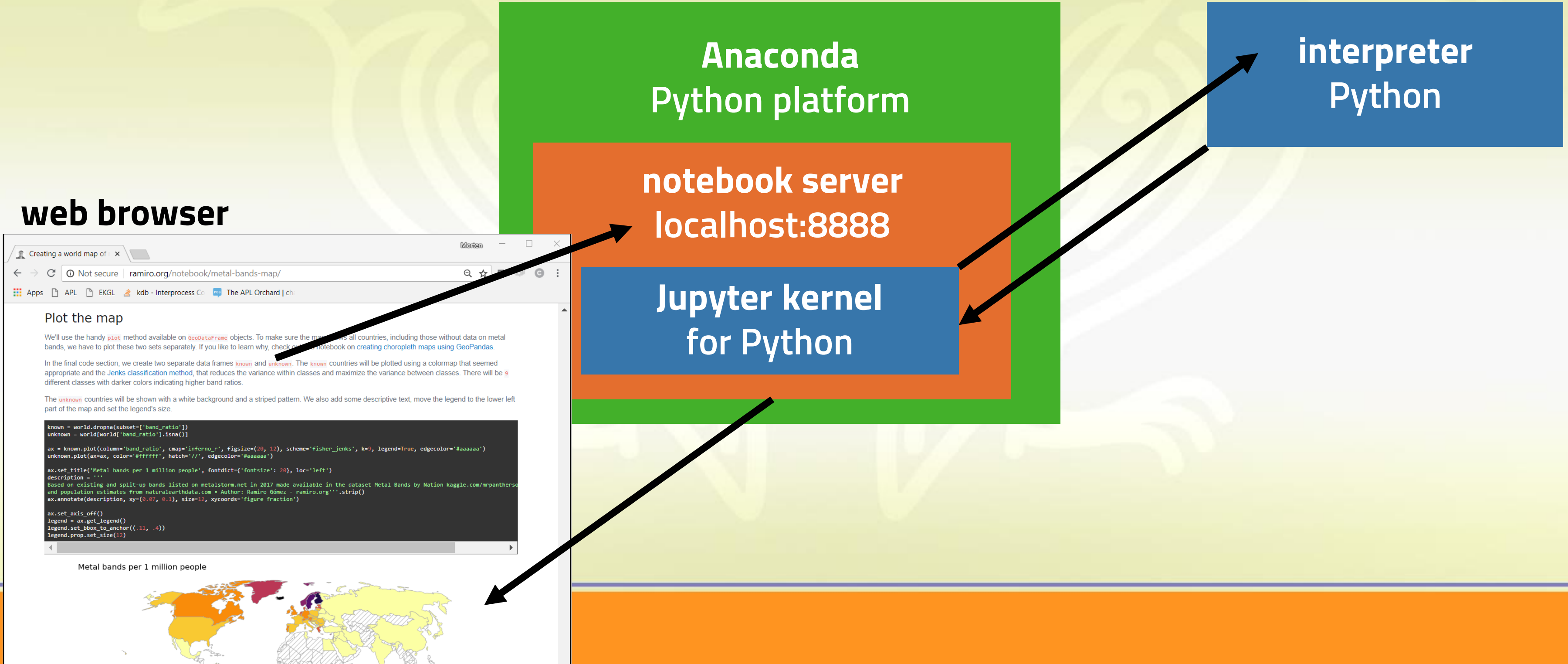
In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people

Local notebook server — Python



Plot the map

We'll use the handy `plot` method available on `GeoDataFrame` objects. To make sure the map shows all countries, including those without data on metal bands, we have to plot these two sets separately. If you like to learn why, check out the notebook on creating choropleth maps using GeoPandas.

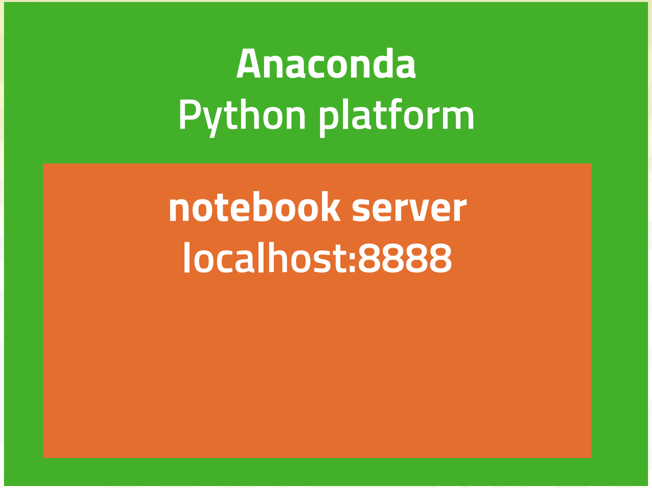
In the final code section, we create two separate data frames `known` and `unknown`. The `known` countries will be plotted using a colormap that seemed appropriate and the Jenks classification method, that reduces the variance within classes and maximize the variance between classes. There will be `9` different classes with darker colors indicating higher band ratios.

The `unknown` countries will be shown with a white background and a striped pattern. We also add some descriptive text, move the legend to the lower left part of the map and set the legend's size.

```
known = world.dropna(subset=['band_ratio'])
unknown = world[world['band_ratio'].isna()]

ax = known.plot(column='band_ratio', cmap='inferno_r', figsize=(20, 12), scheme='fisher_jenks', k=9, legend=True, edgecolor='#aaaaaa')
unknown.plot(ax=ax, color='#ffffff', hatch='//', edgecolor='#aaaaaa')

ax.set_title('Metal bands per 1 million people', fontdict={'fontsize': 20}, loc='left')
description = '''
Based on existing and split-up bands listed on metalstorm.net in 2017 made available in the dataset Metal Bands by Nation kaggle.com/mrpantherso
and population estimates from naturalearthdata.com • Author: Ramiro Gómez - ramiro.org'''.strip()
ax.annotate(description, xy=(0.07, 0.1), size=12, xycoords='figure fraction')

ax.set_axis_off()
legend = ax.get_legend()
legend.set_bbox_to_anchor((.11, .4))
legend.prop.set_size(12)
```

Metal bands per 1 million people

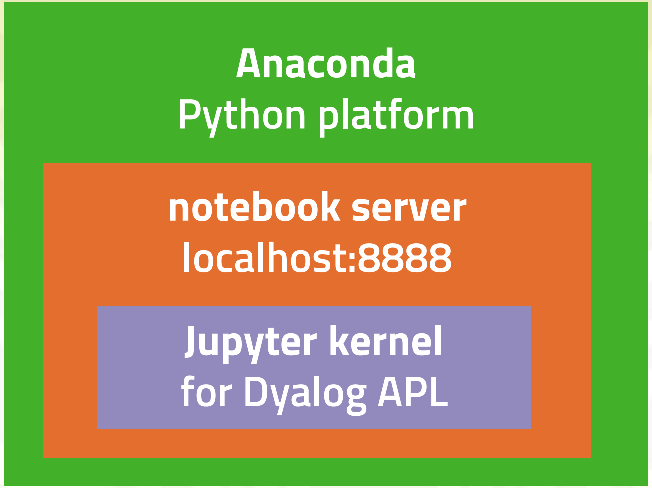
Local notebook server — APL



Anaconda
Python platform

notebook server
localhost:8888

Local notebook server — APL

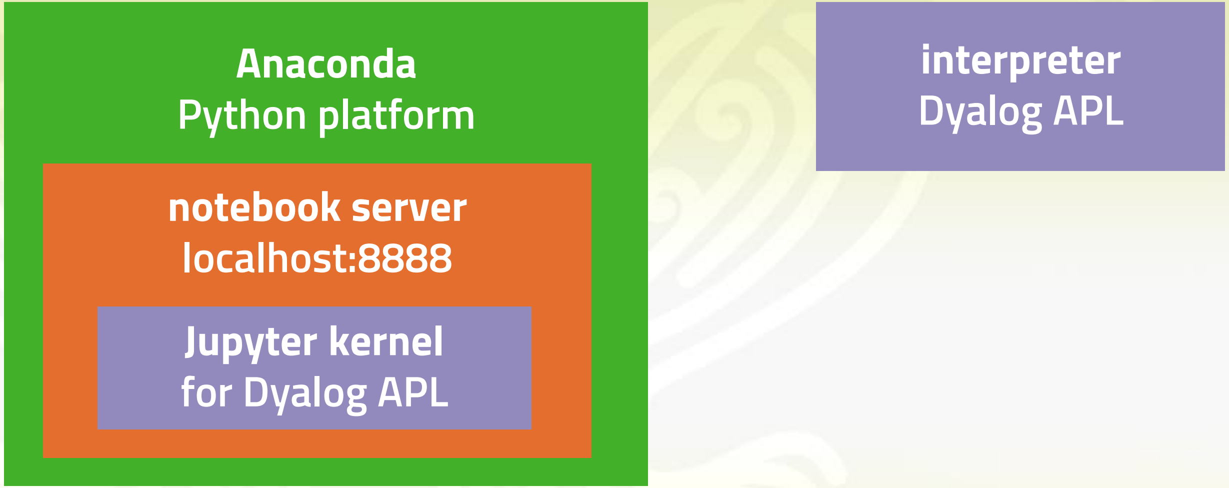


Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Dyalog APL

Local notebook server — APL



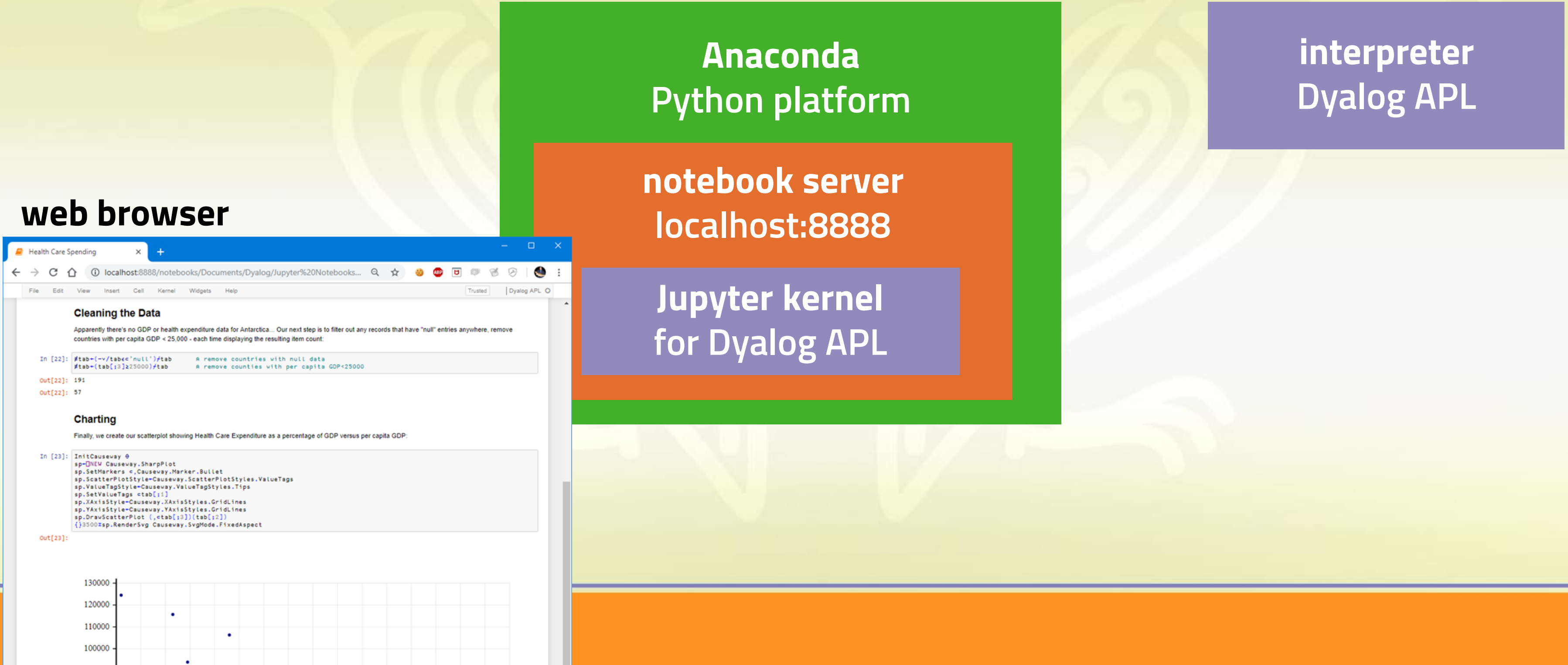
Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Dyalog APL

interpreter
Dyalog APL

Local notebook server — APL



Cleaning the Data

Apparently there's no GDP or health expenditure data for Antarctica... Our next step is to filter out any records that have "null" entries anywhere, remove countries with per capita GDP < 25,000 - each time displaying the resulting item count:

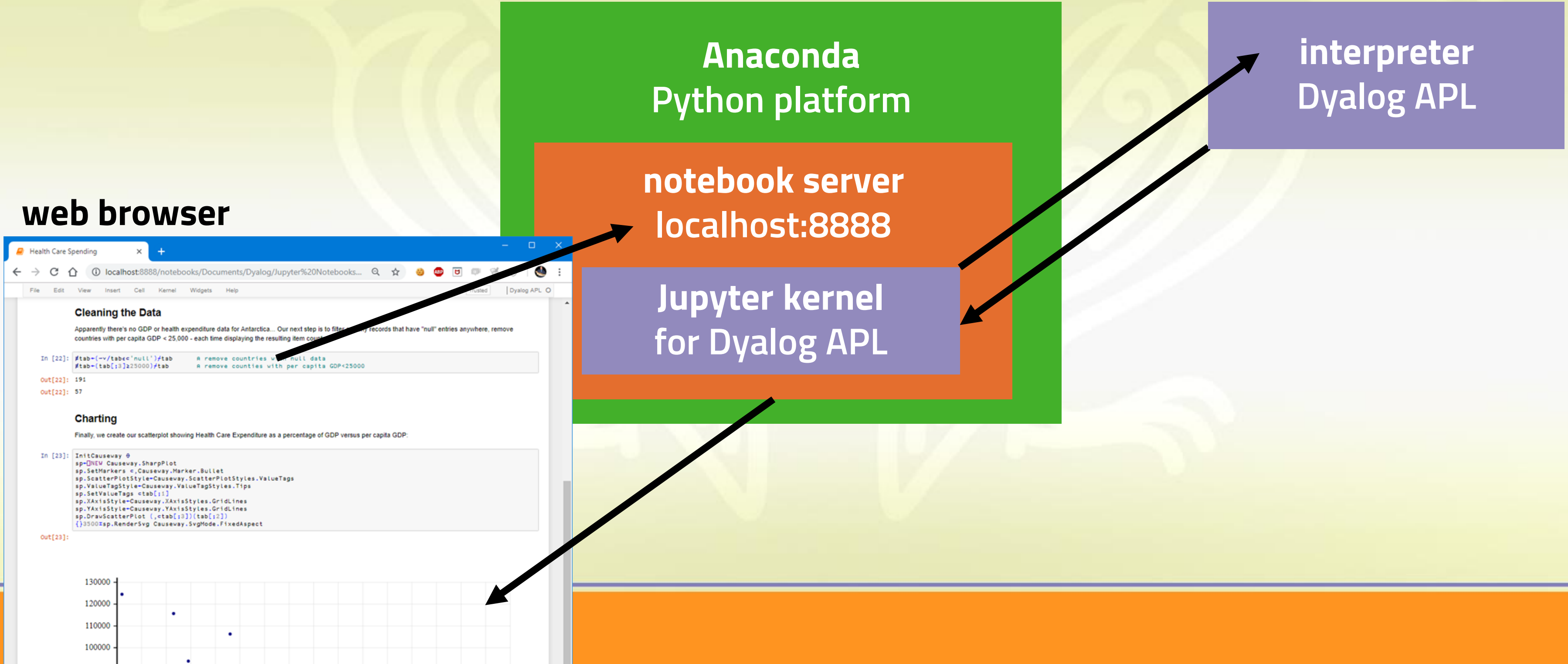
```
In [22]: ≢tab←(~∨/tab∊⊂'null')⌿tab       ⍝ remove countries with null data
         ≢tab←(tab[;3]≥25000)⌿tab         ⍝ remove counties with per capita GDP<25000
Out[22]: 191
Out[22]: 57
```

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

```
In [23]: InitCauseway ⍬
         sp←⎕NEW Causeway.SharpPlot
         sp.SetMarkers ⊂,Causeway.Marker.Bullet
         sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
         sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
         sp.SetValueTags ⊂tab[;1]
         sp.XAxisStyle←Causeway.XAxisStyles.GridLines
         sp.YAxisStyle←Causeway.YAxisStyles.GridLines
         sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
         {}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect
Out[23]:
```

Local notebook server — APL



Anaconda
Python platform

notebook server
localhost:8888

Jupyter kernel
for Dyalog APL

interpreter
Dyalog APL

web browser

Cleaning the Data

Apparently there's no GDP or health expenditure data for Antarctica... Our next step is to filter out any records that have "null" entries anywhere, remove countries with per capita GDP < 25,000 - each time displaying the resulting item count

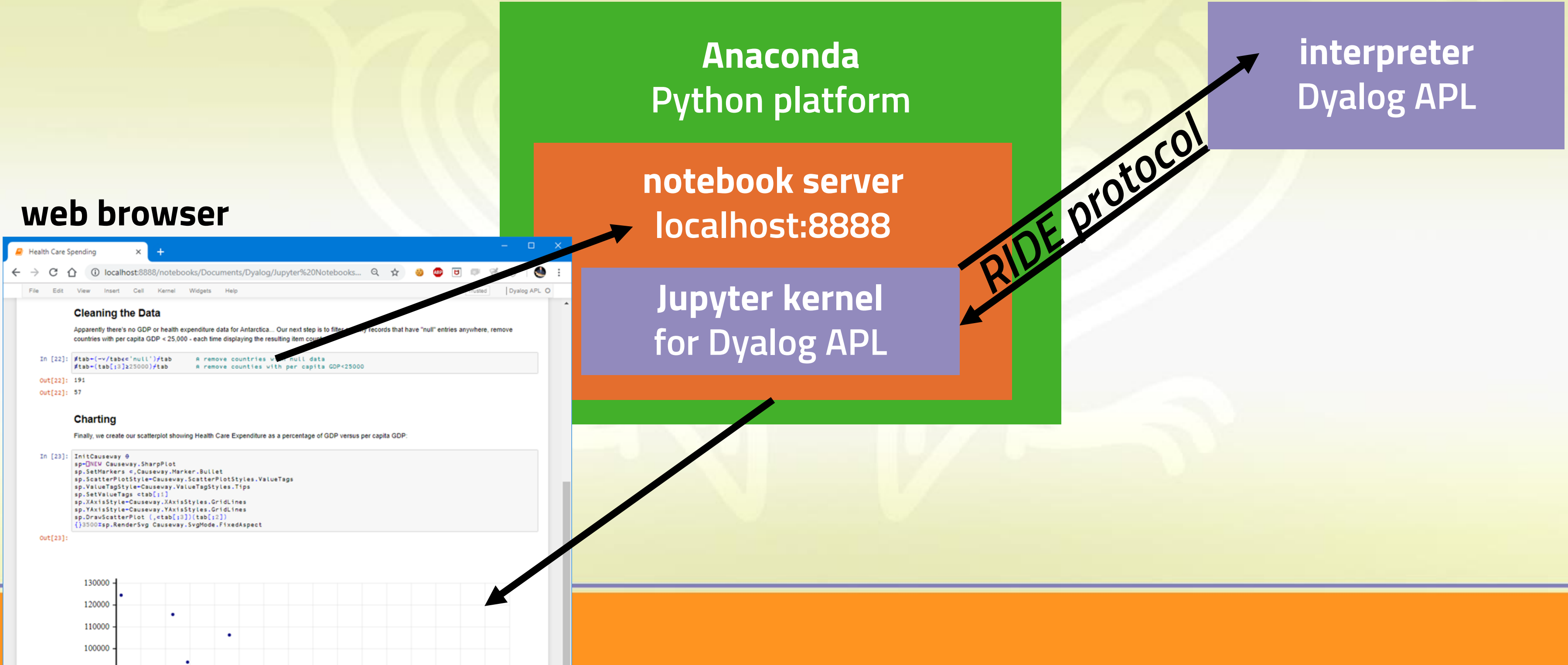
```
In [22]: ≢tab←(~∨/tab∊⊂'null')⌿tab       ⍝ remove countries with null data
         ≢tab←(tab[;3]≥25000)⌿tab        ⍝ remove counties with per capita GDP<25000
Out[22]: 191
Out[22]: 57
```

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

```
In [23]: InitCauseway ⍬
         sp←⎕NEW Causeway.SharpPlot
         sp.SetMarkers ⊂,Causeway.Marker.Bullet
         sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
         sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
         sp.SetValueTags ⊂tab[;1]
         sp.XAxisStyle←Causeway.XAxisStyles.GridLines
         sp.YAxisStyle←Causeway.YAxisStyles.GridLines
         sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
         {}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect
Out[23]:
```

Local notebook server — APL



Cleaning the Data

Apparently there's no GDP or health expenditure data for Antarctica... Our next step is to filter [illegible] records that have "null" entries anywhere, remove countries with per capita GDP < 25,000 - each time displaying the resulting item count

```
In [22]: ≢tab←(~∨/tab∊⊂'null')⌿tab       ⍝ remove countries with null data
         ≢tab←(tab[;3]≥25000)⌿tab       ⍝ remove counties with per capita GDP<25000
Out[22]: 191
Out[22]: 57
```

Charting

Finally, we create our scatterplot showing Health Care Expenditure as a percentage of GDP versus per capita GDP:

```
In [23]: InitCauseway ⍬
         sp←⎕NEW Causeway.SharpPlot
         sp.SetMarkers ⊂,Causeway.Marker.Bullet
         sp.ScatterPlotStyle←Causeway.ScatterPlotStyles.ValueTags
         sp.ValueTagStyle←Causeway.ValueTagStyles.Tips
         sp.SetValueTags ⊂tab[;1]
         sp.XAxisStyle←Causeway.XAxisStyles.GridLines
         sp.YAxisStyle←Causeway.YAxisStyles.GridLines
         sp.DrawScatterPlot (,⊂tab[;3])(tab[;2])
         {}3500⌶sp.RenderSvg Causeway.SvgMode.FixedAspect
Out[23]:
```

Setting up local notebook server

Install Dyalog ☺

Install Dyalog's Jupyter kernel

Install Anaconda

Launch Jupyter notebook server

github.com/Dyalog/dyalog-jupyter-kernel/wiki

Setting up local notebook server

Install Dyalog ☺

Install Dyalog's Jupyter kernel

Install Anaconda

Launch Jupyter notebook server

installation instructions

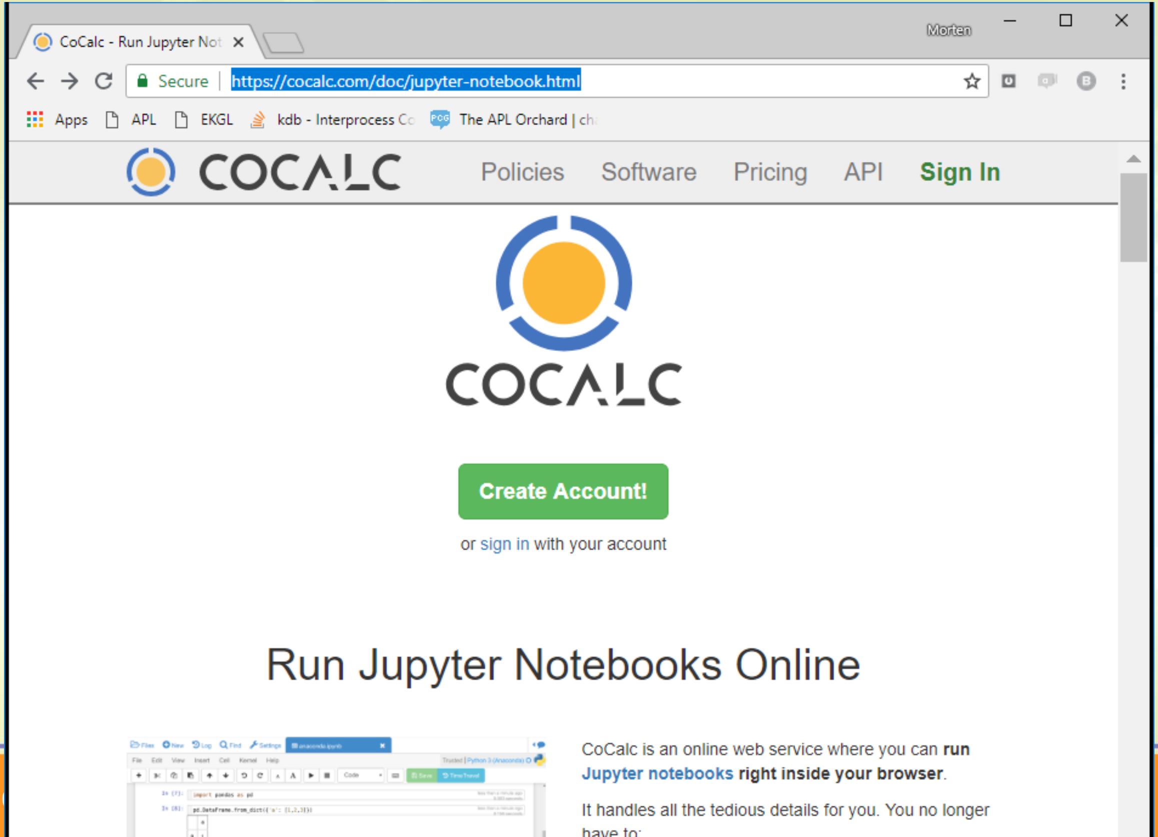
Demo

Installing Jupyter

Opening a notebook

Modifying content

Online notebook servers



or sign in with your account

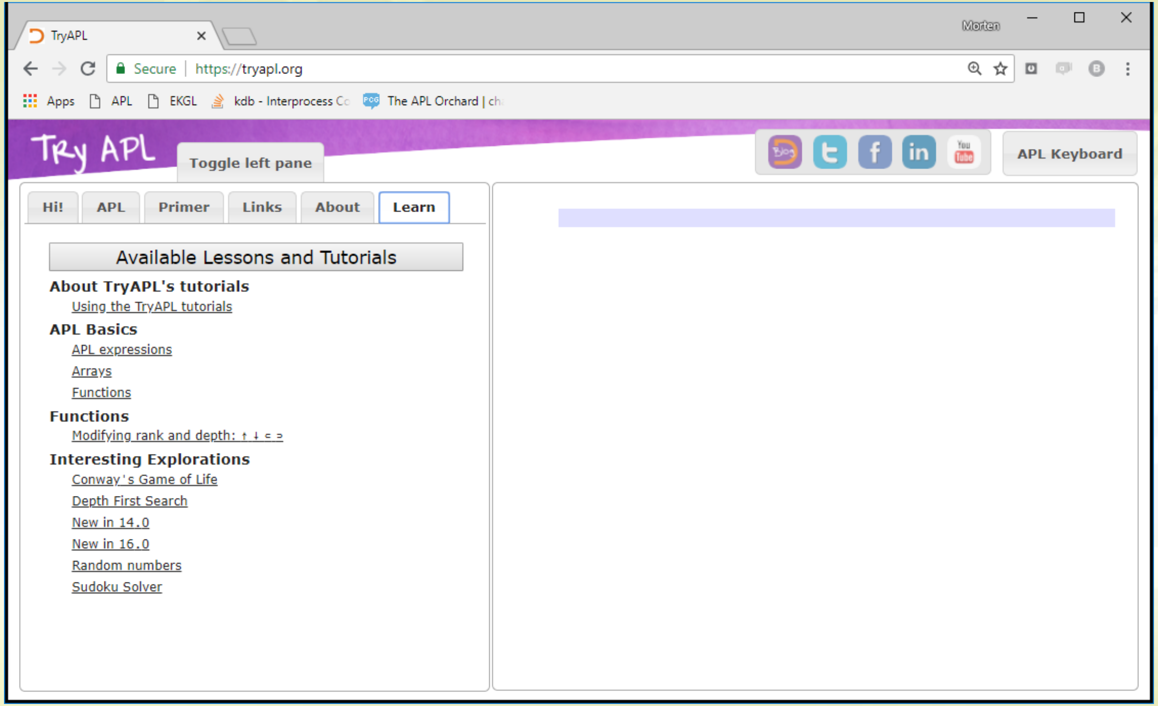
Run Jupyter Notebooks Online

CoCalc is an online web service where you can **run Jupyter notebooks right inside your browser**.

It handles all the tedious details for you. You no longer have to:

There are online services for various programming languages

Online notebook servers



Available Lessons and Tutorials

About TryAPL's tutorials

- Using the TryAPL tutorials

APL Basics

- APL expressions
- Arrays
- Functions

Functions

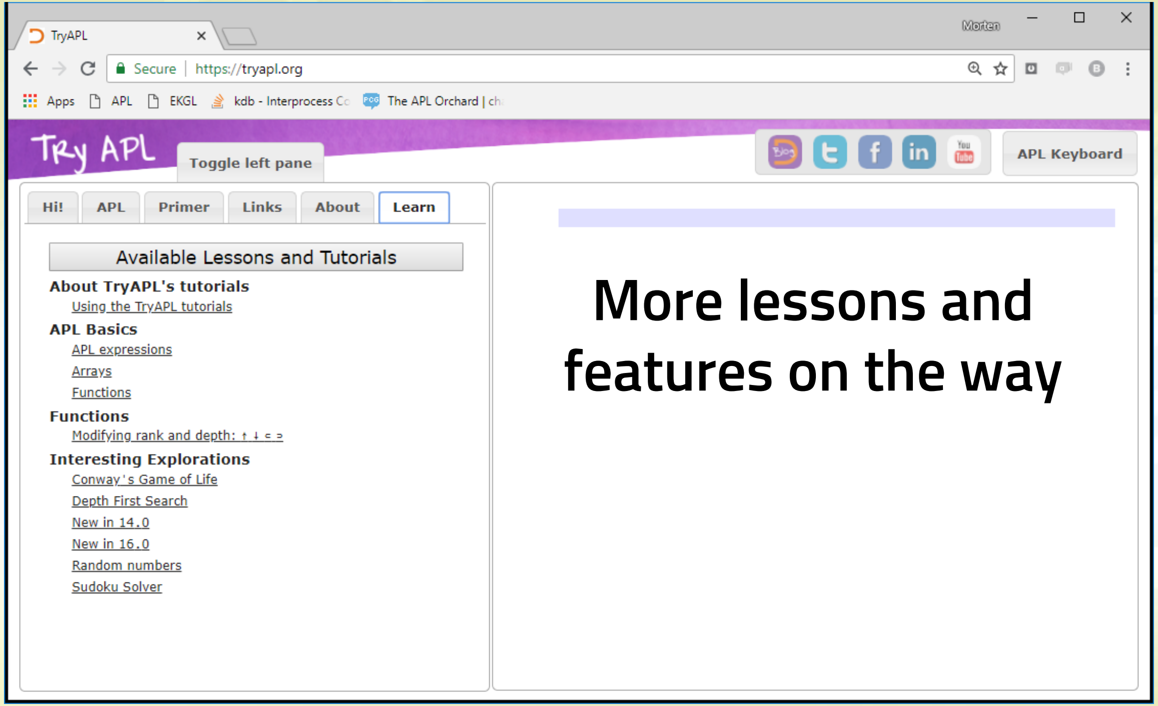
- Modifying rank and depth: ↑ ↓ ⊂ ⊃

Interesting Explorations

- Conway's Game of Life
- Depth First Search
- New in 14.0
- New in 16.0
- Random numbers
- Sudoku Solver

TryAPL's lessons are now Jupyter notebooks

Online notebook servers


TryAPL
Morten
Secure | https://tryapl.org
Apps APL EKGL kdb - Interprocess Co The APL Orchard | ch
Try APL
Toggle left pane
APL Keyboard
Hi! APL Primer Links About Learn

Available Lessons and Tutorials

About TryAPL's tutorials

- Using the TryAPL tutorials

APL Basics

- APL expressions
- Arrays
- Functions

Functions

- Modifying rank and depth: ↑ ↓ ⊂ ⊃

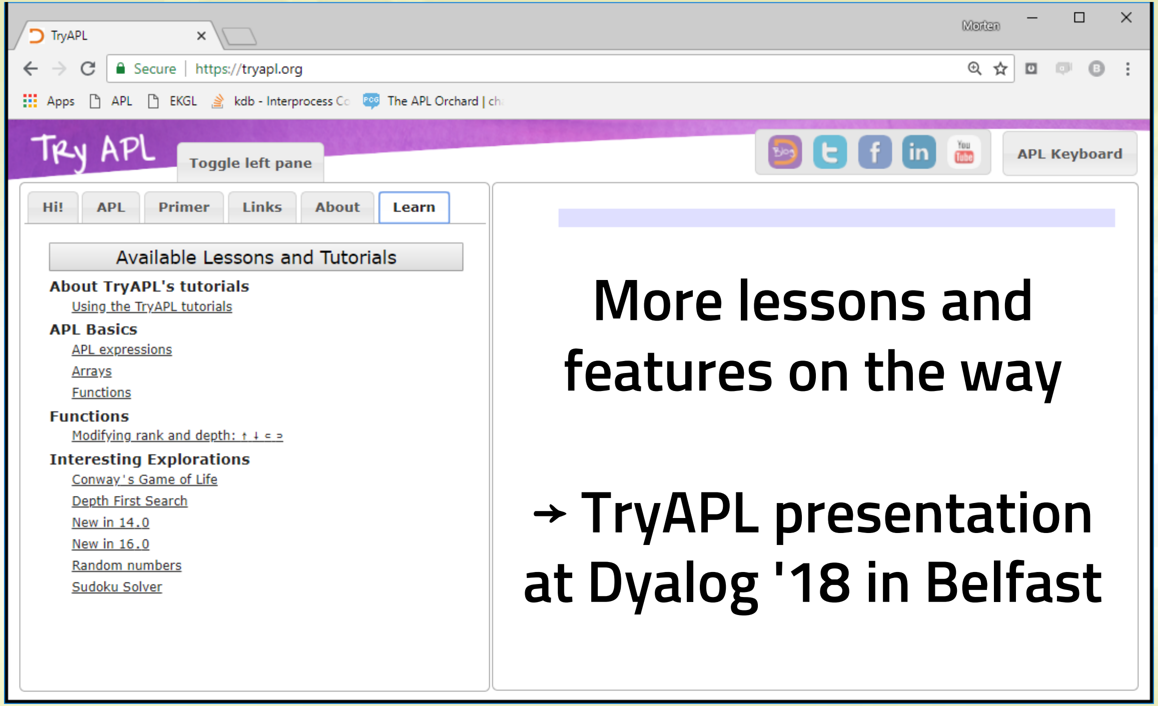
Interesting Explorations

- Conway's Game of Life
- Depth First Search
- New in 14.0
- New in 16.0
- Random numbers
- Sudoku Solver

More lessons and features on the way

TryAPL's lessons are now Jupyter notebooks

Online notebook servers



Available Lessons and Tutorials

About TryAPL's tutorials

- Using the TryAPL tutorials

APL Basics

- APL expressions
- Arrays
- Functions

Functions

- Modifying rank and depth: ↑ ↓ ⊂ ⊃

Interesting Explorations

- Conway's Game of Life
- Depth First Search
- New in 14.0
- New in 16.0
- Random numbers
- Sudoku Solver

More lessons and features on the way

→ TryAPL presentation at Dyalog '18 in Belfast

TryAPL's lessons are now Jupyter notebooks

]

Online notebook servers

Benefit: nothing to install

you may need to sign up for an account

To protect servers, host may place restrictions

or run in a sandbox with limited connectivity

Notebooks can execute any code

all code is run on the host server
same privileges as local execution

Static notebook viewers

Notebooks are stored as .ipynb files

- .ipynb files are in JSON format
- each code cell may include output from the last execution

You can share an .ipynb file

- anyone with a local notebook server can view it
- … but of course cannot execute anything new

Many online systems have viewers

- GitHub's file previewer
- Project Jupyter's nbviewer.jupyter.org

Exported notebooks

Notebooks can be exported to many standard

for example HTML, PDF, and $\LaTeX$

Some formats require 3rd party plug-ins

Exported notebooks are static

expressions cannot be re-executed

Demo

Creating a new notebook document

Generating rich output

Ways to use notebooks — recap

Installing a Jupyter notebook server on your PC

Use an online notebook server like cocalc.com

Store the notebook with output, then open in a notebook viewer

Export to HTML, PDF, $\LaTeX$, …

Ask questions now!

Wiki	github.com/Dyalog/dyalog-jupyter-kernel/wiki
Email	notebooks@dyalog.com

Ask questions now!

Wiki	github.com/Dyalog/dyalog-jupyter-kernel/wiki
Email	notebooks@dyalog.com

Thank you

Technology Partnership (tp.rs)

for the prototype APL kernel

Ask questions now!

Wiki	github.com/Dyalog/dyalog-jupyter-kernel/wiki
Email	notebooks@dyalog.com

Thank you

Will Robertson (our summer intern)
for working on the kernel
and creating many notebooks

Ask questions now!

Wiki github.com/Dyalog/dyalog-jupyter-kernel/wiki

Email notebooks@dyalog.com

Interested in an internship?

Email careers@dyalog.com

Thank you

Will Robertson (our summer intern)
for working on the kernel
and creating many notebooks

Ask questions now!

Wiki	github.com/Dyalog/dyalog-jupyter-kernel/wiki
Email	notebooks@dyalog.com

Thank you

Gil Athoraya (of Optima Systems)

for implementing syntax colouring

Ask questions now!

Wiki	github.com/Dyalog/dyalog-jupyter-kernel/wiki
Email	notebooks@dyalog.com

Thank *you*
for watching

Webinars on Thursdays at 16:00 UTC

Comment and suggest to webinar@dyalog.com or @Adám in chat.stackexchange.com/rooms/52405

No webinar in October due to Dyalog '18 in Belfast
October 28th–November 1st
many sessions will be livestreamed

General APL Questions
stackoverflow.com