HELMHOLTZ
Helmholtz-Institut Mainz

Papermill

Jupyter Notebooks run in Batch

Jupyter Notebooks

Jupyter Notebooks:

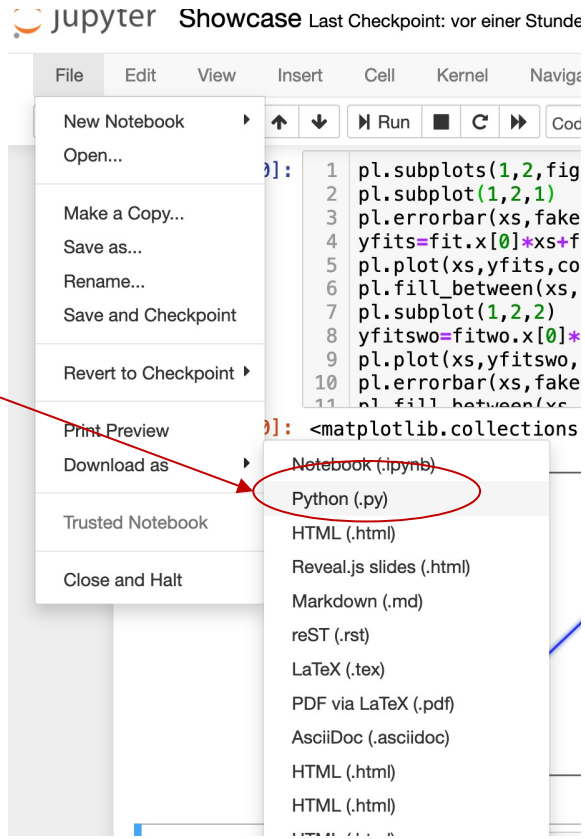
- Nice Interface for FAST implementations (Mathematica Notebook-like)
- Data Exploration
- Visualization

Caveats:

- Nonlinear execution
- Not easily scriptable - input parameters?

Jupyter Notebooks

- Usual workflow(s):
  - Type A:
    - Develope in Jupyter notebook
    - Export notebook to script
    - Run script in batch system
  - Type B:
    - Explore data
    - Fix some aspect of data analysis
    - Run script in batch system with fixed parameters



Jupyter Notebooks

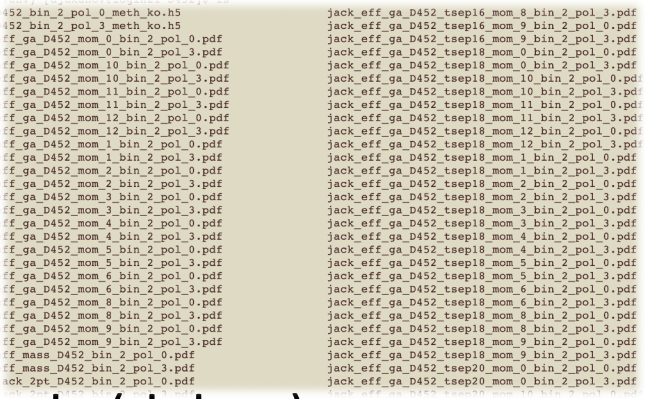
- Usual workflow(s) problems:
  - Type A/B:
    - Passing arguments needs some effort modifications of the scripts (e.g. argparse)

```
parser = argparse.ArgumentParser(
    description='Extract baryon two point functions')

parser.add_argument('-e','--exact-src-dir',dest='exact_src_dir',
                    help='Directory containing exact solves.',
                    nargs=1, required=True)
parser.add_argument('-s','--sloppy-src-dir',dest='sloppy_src_dir',
                    help='Directory containing sloppy solves.',
                    nargs=1, required=True)
parser.add_argument('-p','--prefix',dest='prefix',
                    help='Prefix of measurment files.',
                    nargs=1, required=True)
parser.add_argument('-o','--outfile',dest='out_file',
                    help='Outputfile name without suffix.',
                    nargs=1, required=True)
parser.add_argument('-n','--number',dest='numconfs',
                    help='Run on first n configs.',
                    nargs=1)
```

    - Keep output of script (stdout,plots) in the same file (not easy) maybe including errors

      Having a directory with:
      - pdf for every plot
      - Maybe an explaining README



    - Some exceptions only occur rarely (debug)

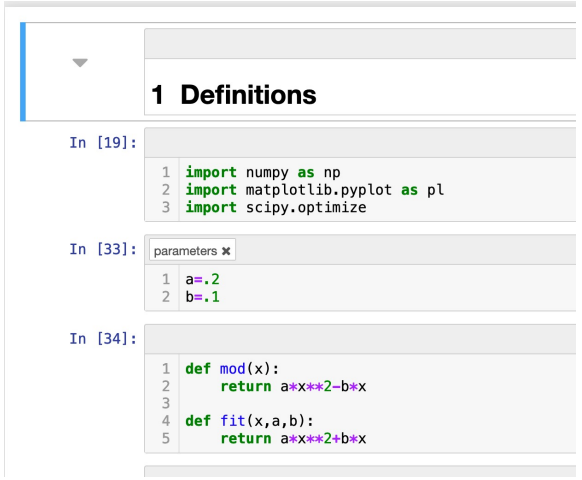
Papermill

- Would be nice to
  - Run notebooks with arguments
  - Create output-notebooks with mixed output (strings, plots, etc.)
  - Interactively debug failing notebooks
  - Run this in batch

https://papermill.readthedocs.io/en/latest/#

Papermill

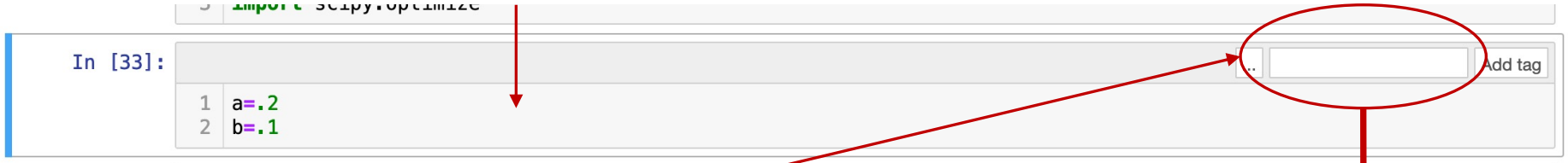
- Idea of papermill
  - Have a „template“ notebook
  - Declare parameters as input
  - Run the template notebook with input parameters
  - Save the output in a new notebook
- Example:
  - Template notebook template.ipynb
  - Two paramaters: a and b
  - Run in command line




```
(venv) [djukanov@login21 papermill]$ papermill template.ipynb run1.ipynb -p a 2.2 -p b 2.55
Input Notebook:  template.ipynb
Output Notebook: run1.ipynb
Executing: 100%|                                                                | 16/16 [00:18<00:00,  1.17s/cell]
```

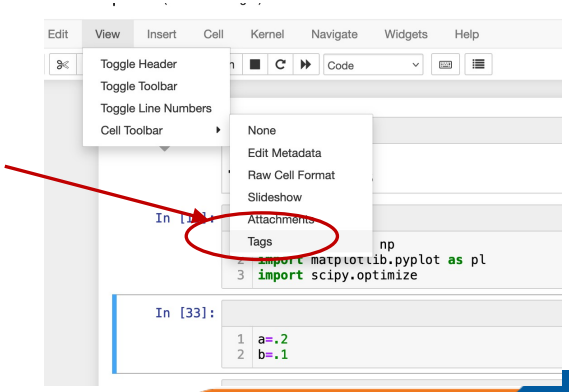
Papermill

- How to get input
  - Take ordinary cell



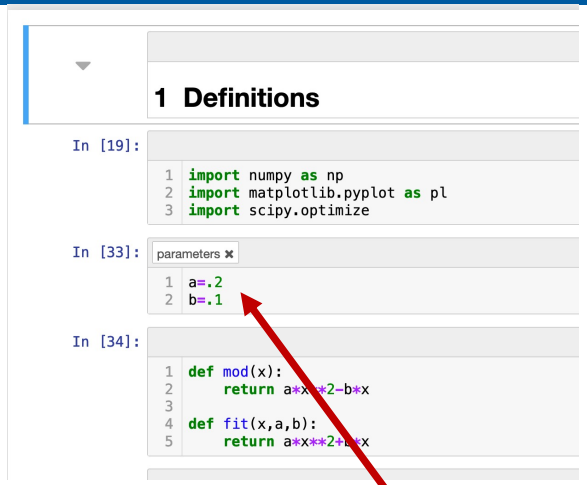
  - Set tag of the cell as: parameters

  - If **Add tag** is not visible toggle it

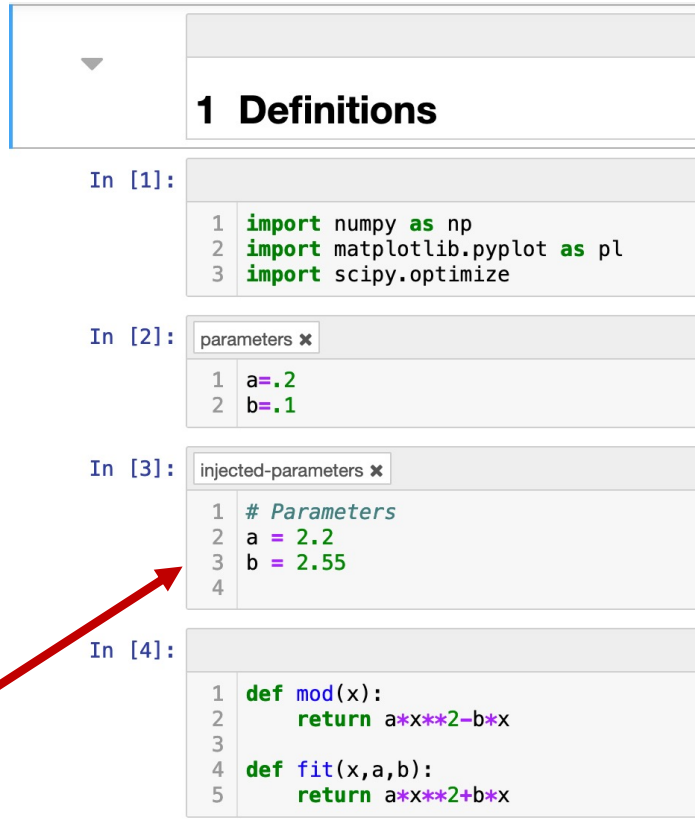


  - That‘s all!

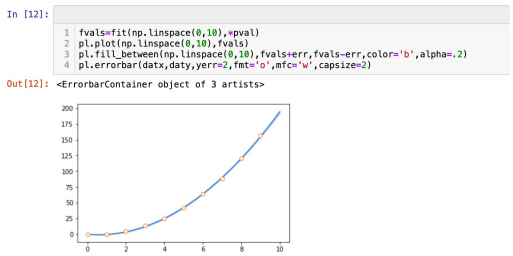
Papermill



papermill



- Example Output:
  - Output notebook run1.ipynb
  - Two paramaters: a and b
  - Set in command line
  - Some result based on injected parameters



Papermill

- Command line has reduced set of options
- Run from within python

```
#!/usr/bin/env python

import papermill as pm

pm.execute_notebook('./template.ipynb','./run2.ipynb',parameters=dict(a=2,b=25))
~
```

- Parameters as dict:

  Can pass more complicated things (e.g. arrays)

```
def run_D200():
    for bins in [2]:
        for binloops in [1]:
            pol=3
            for po in ['pol0_','']:
                ff=ret_runs('D200')
                if po == 'pol0_':
                    flo=16
                    fup=28
                else:
                    flo=16
                    fup=31
                try:
                    pm.execute_notebook('./Sigma_term_papermill.ipynb','./output/Sigma_D200_bin_'+str(bins)+'_po_'+po+'.ipynb',parameters=
                            dict(ensn='D200',ensr='D200r000',datadir='./',files_2pt=ff,flower=flo,fupper=fup,
                                bins=bins,smd=1,po=po,pol=3,bins_loops=binloops,
                                outfname_sigmapinff='gs_eff_D200_bin_'+str(bins)+'_bin_loop_'+str(binloops)+po+'_sigmapin_ff.h5',
                                outfname_scalarff='gs_eff_D200_bin_'+str(bins)+'_bin_loop_'+str(binloops)+po+'_scalar_ff.h5',
                                outdir='./output/',rmcfgs=[[757]]))
                except:
                    print(traceback.format_exc())
                    continue
```

Papermill

- Run in Slurm just as any other script

```
#!/bin/bash
#SBATCH -J papermill           # Job name
#SBATCH -p himster2_exp                  # Queue name 'short' or 'long' on Mogon I
                                 #            'smp' on Mogon II
#SBATCH -n 1
#SBATCH --ntasks-per-node=32                    # Total number of tasks, here explicitly 1
#SBATCH -t 05:30:00              # Run time (hh:mm:ss) - 0.5 hours
#SBATCH --hint=nomultithread

#SBATCH -A m2_him_exp            # Specify allocation to charge against

export HDF5_USE_FILE_LOCKING='FALSE'

module load mpi/OpenMPI/4.0.3-GCC-9.3.0
source /home/djukanov/venv/bin/activate

export PYTHONPATH=$PYTHONPATH:/home/djukanov/LaQAPack/src/python/LaQAPack

./run_papermill.py -e  $1
```

- In case of an error Cell is output

Papermill

- In case of an error Cell is output

- Run the cell up to the error
- Debug interactively
- Include changes in template
- Rerun

END