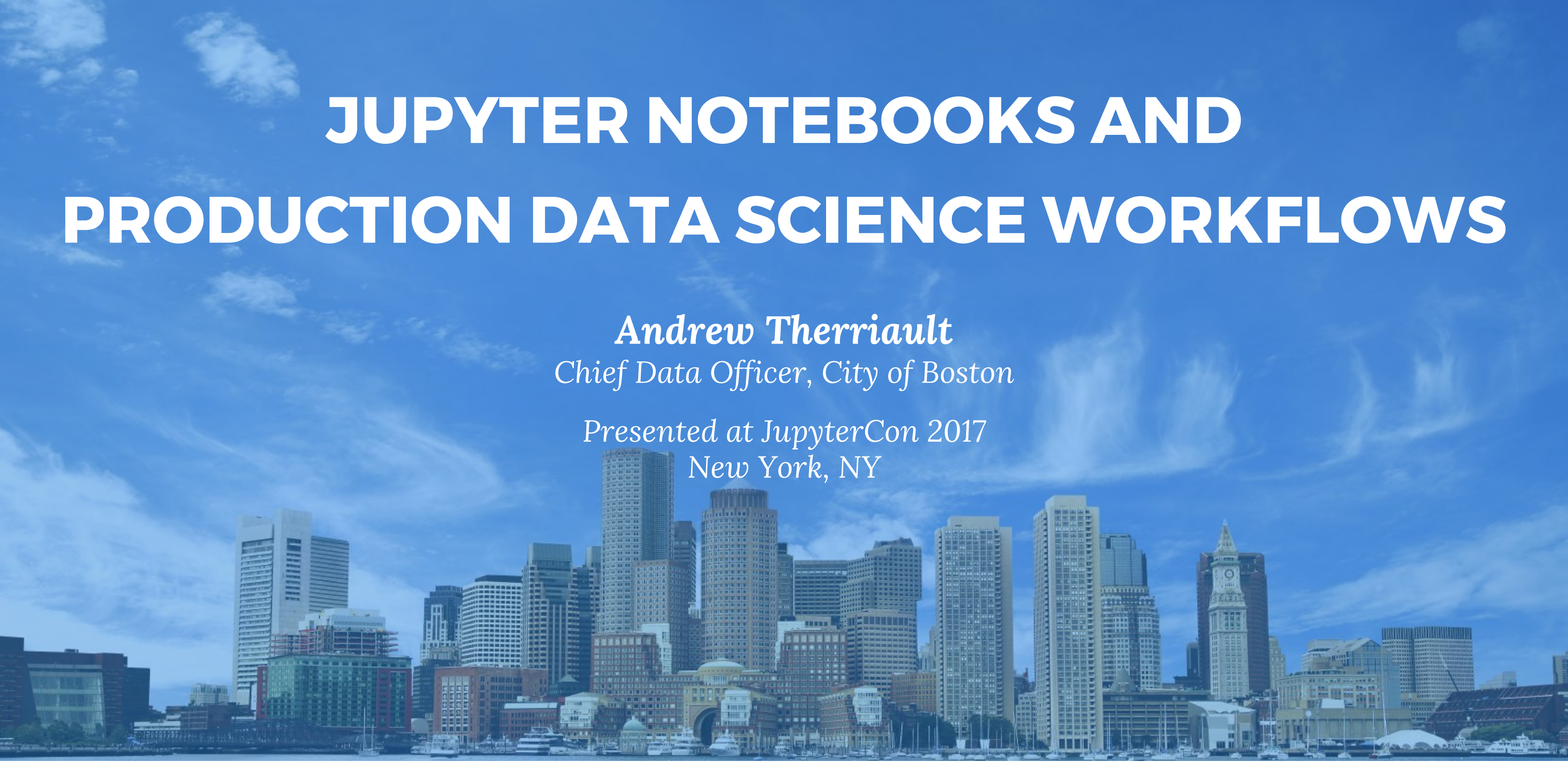


JUPYTER NOTEBOOKS AND PRODUCTION DATA SCIENCE WORKFLOWS

Andrew Therriault
Chief Data Officer, City of Boston

Presented at JupyterCon 2017
New York, NY

City of Boston
Mayor Martin J. Walsh

Innovation & Technology

GETTING STARTED

MY BACKGROUND

Chief Data Officer for the City of Boston, leading our Analytics Team and steering the city's overall plans for using data.

Other resume highlights:

- *PhD in poli-sci from NYU*
- *Post-doc at Vanderbilt*
- *Data science consultant*
- *Director of Data Science at the Democratic National Committee (2014 and 2016 election cycles)*

WHICH TRANSLATES TO

> 100 machine learning models used in production, including:

- Vote choice & turnout
- Donation propensity
- Phone & address quality
- Restaurant inspections
- Service request coding

Since 2014, almost all models built in Jupyter notebooks

Also, hundreds of notebooks for analysis, data viz, and reporting.

WHAT'S IN THIS TALK

The basic question: ***where do Jupyter notebooks fit into production workflows?***

How I'll tackle it:

- My history with Jupyter
- Workflow design patterns
- How to choose?
- Real-world examples
- Looking ahead

An important caveat: this is ***not*** an authoritative tutorial. I'm mainly sharing my experiences to help others as they think through their own challenges.

MY HISTORY WITH JUPYTER FOR DATA SCIENCE

THE BEGINNING (2013-14)

My first Python model:

- Data in CSVs extracted from MySQL
- Developed in Spyder IDE, run at command line

When I started at DNC:

- Inherited a pretty decent server in our datacenter
- Set up notebooks after failing to get Spyder to connect to remote kernel
- Discovered `pyodbc`, stopped using separate SQL client

GOING STRONG (2014-16)

Soon got my team on notebooks:

- Started by setting up notebook servers for each user, eventually deployed JupyterHub
- First approach was a unreliable and a pain, second took quite a while to figure out the bugs, neither was great for collaboration

Discovered the wonders of `nbconvert` (before which I spent way too many nights "model-sitting" notebook runs)

STARTING OVER (2016-17)

When I got to Boston:

- No organized system for running data science code (where to run, how to collaborate, etc.)
- Rather than solve the problem from scratch again, paid Continuum for their solution instead

Now, we have a shared notebook server on AWS, and use it for most data analysis and model development

But what about production?

SO MANY THINGS TO THINK ABOUT FOR PRODUCTION WORKFLOWS

HOW DO YOU GET THE DATA?

Are you using an extract? Connecting to a database? Pulling from a data store like S3?

*And is it already prepped for you? Do you do it before or after you import it to Python?**

HOW IS YOUR CODE ORGANIZED?

Are you calling Python as a Jupyter kernel, or converting it into a standard Python script?

Is all your code in one file? Are you using functions? Objects? Are your parameters in the same file or a different one? Are you making packages?

WHERE DOES THE CODE RUN?

Are you doing production work on your personal machine? Running it on a server? Deploying it with a service like AWS Lambda?

How are you handling dependencies? Creating environments? Making Docker containers?

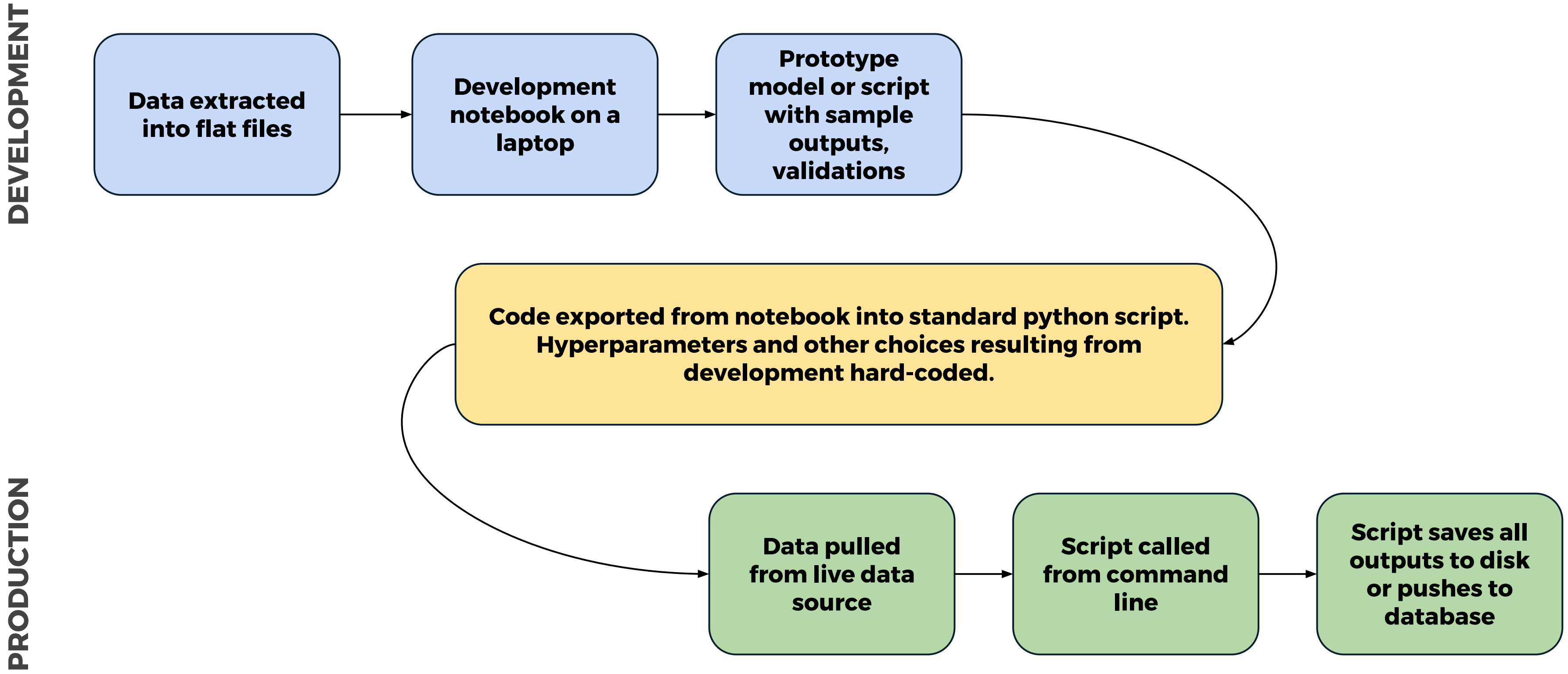
WHERE DO YOUR OUTPUTS GO?

Are the outputs something that can be represented in a notebook, or do you need them in another format? Are you sharing them? Using for multiple purposes?

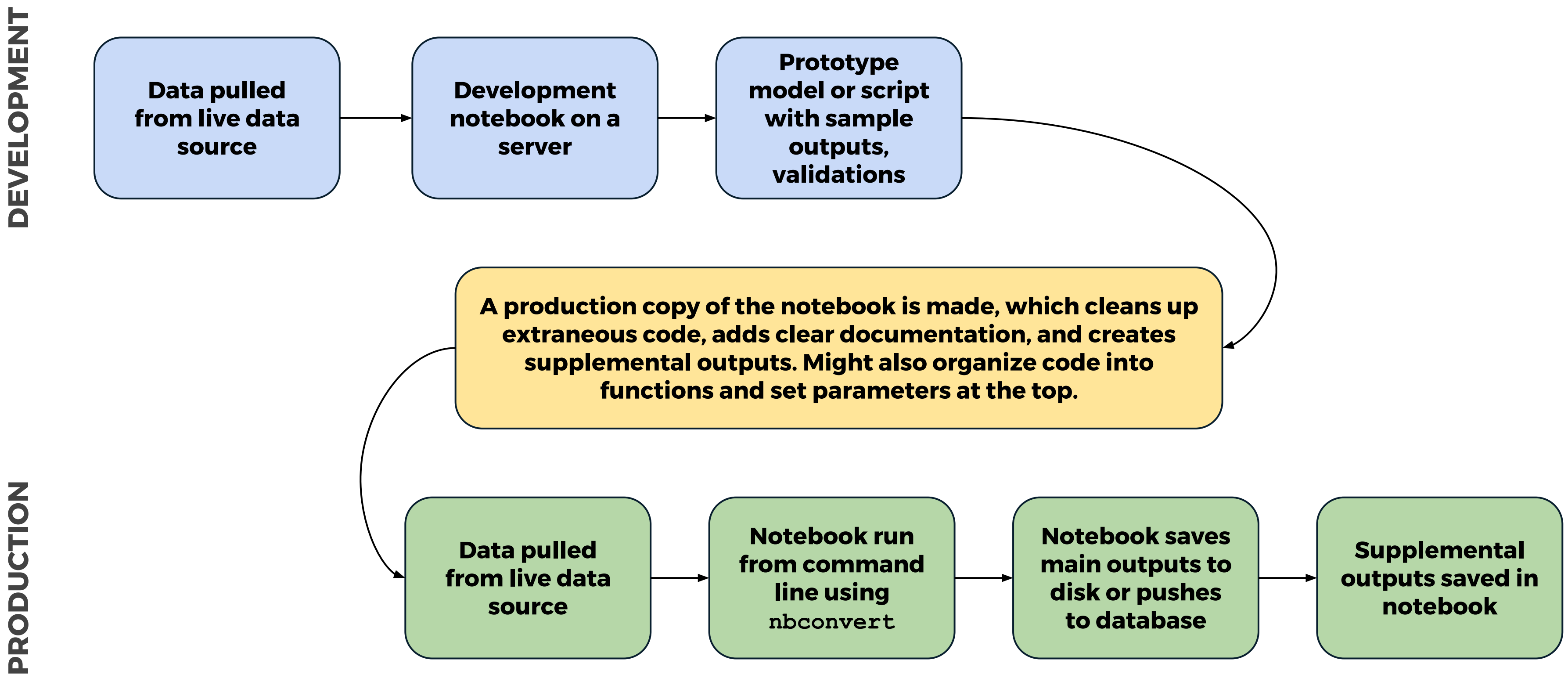
Are there supplemental outputs (descriptive stats, validation checks, etc.) that work well in a notebook?

* - *I'm just going to say Python here, but substitute R or some other kernel instead if you prefer.*

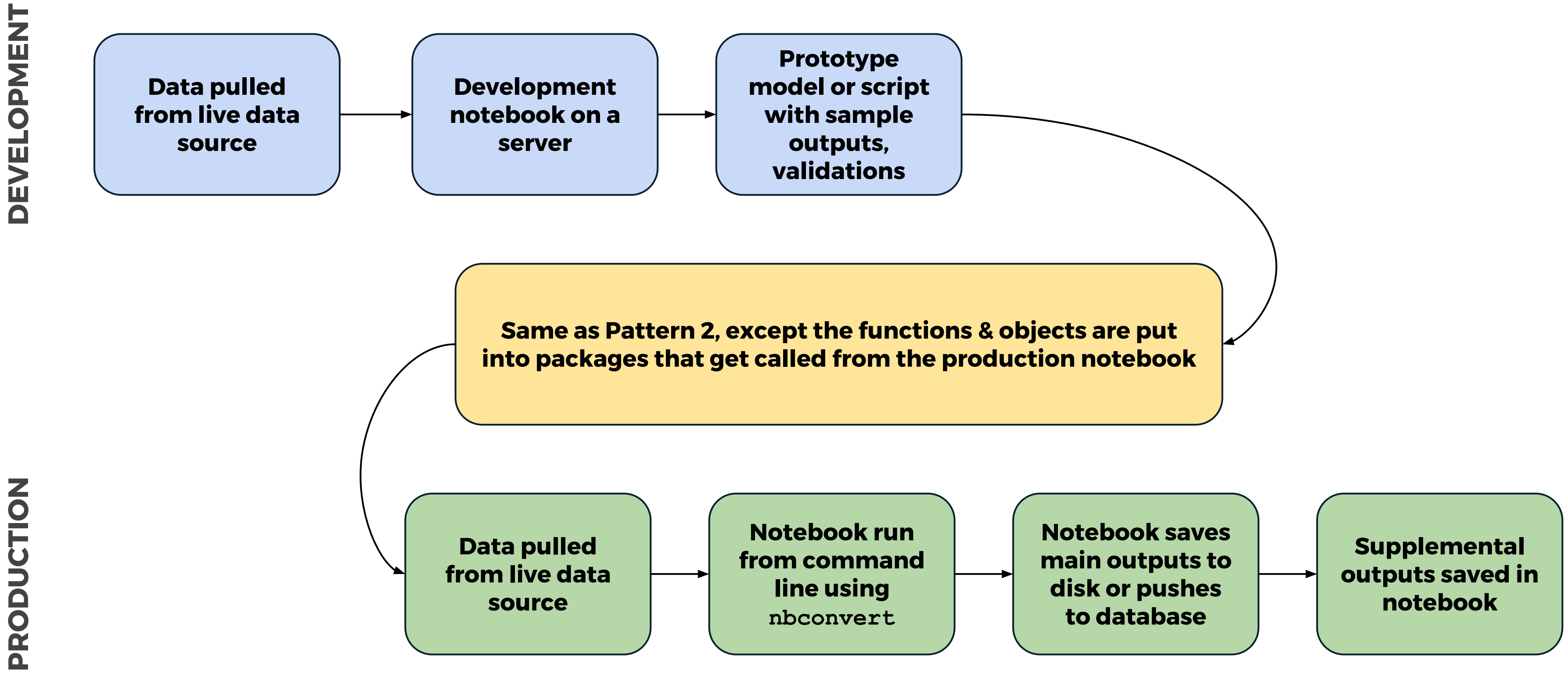
DESIGN PATTERN 1: THE OLD WAY


DEVELOPMENT
Data extracted into flat files
Development notebook on a laptop
Prototype model or script with sample outputs, validations
Code exported from notebook into standard python script. Hyperparameters and other choices resulting from development hard-coded.
PRODUCTION
Data pulled from live data source
Script called from command line
Script saves all outputs to disk or pushes to database

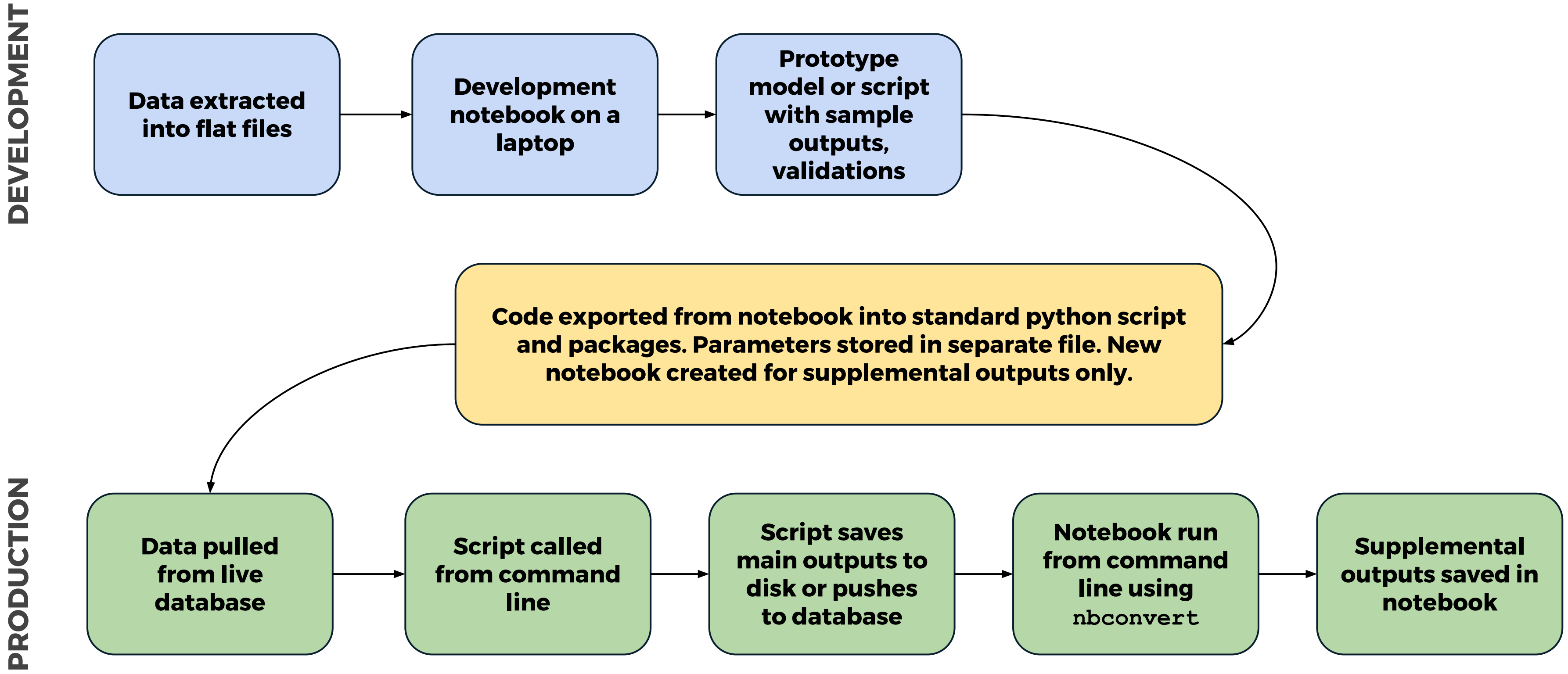
DESIGN PATTERN 2: FULL-CYCLE PRODUCTION NOTEBOOKS


DEVELOPMENT
Data pulled from live data source
Development notebook on a server
Prototype model or script with sample outputs, validations
A production copy of the notebook is made, which cleans up extraneous code, adds clear documentation, and creates supplemental outputs. Might also organize code into functions and set parameters at the top.
PRODUCTION
Data pulled from live data source
Notebook run from command line using nbconvert
Notebook saves main outputs to disk or pushes to database
Supplemental outputs saved in notebook

DESIGN PATTERN 3: SCALABLE PRODUCTION NOTEBOOKS


DEVELOPMENT
Data pulled from live data source
Development notebook on a server
Prototype model or script with sample outputs, validations
Same as Pattern 2, except the functions & objects are put into packages that get called from the production notebook
PRODUCTION
Data pulled from live data source
Notebook run from command line using nbconvert
Notebook saves main outputs to disk or pushes to database
Supplemental outputs saved in notebook

DESIGN PATTERN 4: THE MULTI-TOOL APPROACH


DEVELOPMENT
Data extracted into flat files
Development notebook on a laptop
Prototype model or script with sample outputs, validations
Code exported from notebook into standard python script and packages. Parameters stored in separate file. New notebook created for supplemental outputs only.
PRODUCTION
Data pulled from live database
Script called from command line
Script saves main outputs to disk or pushes to database
Notebook run from command line using nbconvert
Supplemental outputs saved in notebook

OTHER VARIANTS ON THESE PATTERNS

DATA PREPARATION

You could run a distinct ETL step at the start of production, have it as a totally separate pipeline, or include it in your main analysis / modeling code.

VERSION CONTROL AND CI

Often the split between dev and prod isn't nearly as clean, so you may go back and forth between the two using tools like Jenkins or Travis. (That also means adding tests into the workflow!)

CONTAINERS AND ENVIRONMENTS

At production time, you'll probably want to package up all your dependencies along with your scripts, especially if you plan to automate

SERVICES AND DEPLOYMENT

These patterns describe your typical batch workflow, but in more complex use cases "production" will also include on-demand or streaming use, and services like APIs and Lambda have unique requirements

WHAT TO CONSIDER WHEN CHOOSING YOUR WORKFLOW

RELIABILITY	Jupyter's *way* more stable now than it was a few years ago, but it still introduces more ways for code to break, especially if your notebook server is not something you're spinning up uniquely for every project
ACCESSIBILITY	While apps like Github and nbviewer have made it easier to share notebooks, there's still a lot more flexibility to be had from exporting results into text files, spreadsheets, databases, or saved images
REUSABILITY	Modules and packages exist because copied-and-pasted code is inefficient to produce and nearly impossible to maintain
INTERPRETABILITY	By putting code alongside documentation and results, notebooks make it much easier for others (*or future you*) to figure out what your code does, and that's as valuable for production as development
FLEXIBILITY	Notebooks have a Swiss Army Knife quality to them - you can do almost anything in them, but they're not the best tool for every job
AGILITY	We love notebooks because they make data science easier and faster, and getting something new into production quickly is a big deal

REAL-WORLD EXAMPLES OF ML MODELS IN PRODUCTION

PATTERN 1: PHONE QUALITY

Used each day's results of phone contact attempts to update our predictions of each voter's best number to call. The production scripts were standard Python, with predictions uploaded to the production database and validation results & summary stats saved to disk.

PATTERN 2: DONATION PROPENSITY

To target monthly fundraising mailings, a series of notebooks was used to run SQL to prep data, tune & train models, generate predictions, validate models, and summarize results. Used `nbconvert` *to run in the background on a production server.*

PATTERN 3: ISSUE POSITIONS

Developed and released 10 unique issue position models in 1 month by reusing the modeling package we built to standardize pulling data, tuning and training models and ensembles, running validation checks, and parallelizing prediction.

PATTERN 4: 311 SERVICE REQUESTS

Our new 311 website allows users to type their issue in plain language and be routed to the proper request form. We developed an NLP model in Jupyter using a sample of case descriptions, then converted to Python modules. Prediction API is deployed on Lambda, and the model will be retrained with new data and redeployed automatically throughout each day.

THINKING AHEAD TO THE FUTURE

NOTEBOOKS AS APPLICATIONS

As notebooks grow from development environments to shareable applications, they are often becoming end products themselves, rather than just tools

DATA SCIENCE PLATFORMS

A growing number of data science platforms (such as Anaconda, Civis, and Domino) make notebooks a first-class part of their toolkit, enabling much easier deployment and automation for data scientists

THE RISE OF CONTAINERS

Containers continue to expand their place in the data science ecosystem, and so notebooks are becoming a more practical tool for production deployments, even for serverless architectures like Lambda

NEW JUPYTER CAPABILITIES

Upcoming developments in the Jupyter ecosystem (especially JupyterLab) will likely blur the lines between production applications and development tools even further - for example, by substituting extensions for traditional modules and packages.

For more information about the City of Boston's Analytics Team, go to boston.gov/analytics.

And if you want to talk more:
andrew.therriault@boston.gov (work)
andrew.therriault@gmail.com (personal)
@therriaultphd (twitter)

THANK YOU!

City of Boston
Mayor Martin J. Walsh

Innovation & Technology