Sandtable

White Paper:
The Sandtable Model Foundry

BACKGROUND: DATA SCIENCE AS DATA PRODUCT DEVELOPMENT

Data science - the practice of deriving insight from data through a combination of computational and statistical techniques - is a fast-growing field. People from a wide range of backgrounds are attempting to become data scientists by reading about it online, in books, and by meeting other actual and aspiring data scientists in real and virtual forums.

As with any discipline in its very early stages, there are few, if any, rules or standards about how to do data science. There are notions of good and bad practice that are becoming better known through books and blogs but no formal code.[1]

One of the biggest issues for data science is the ease with which it is possible to 'get going' on a data science project. Given a data set, a computing environment, access to the Internet, the work of other data scientists, and intermediate programming skills in Python or R, anyone can start to generate apparently interesting insights. Indeed, this is how data science is practiced in many organisations today.

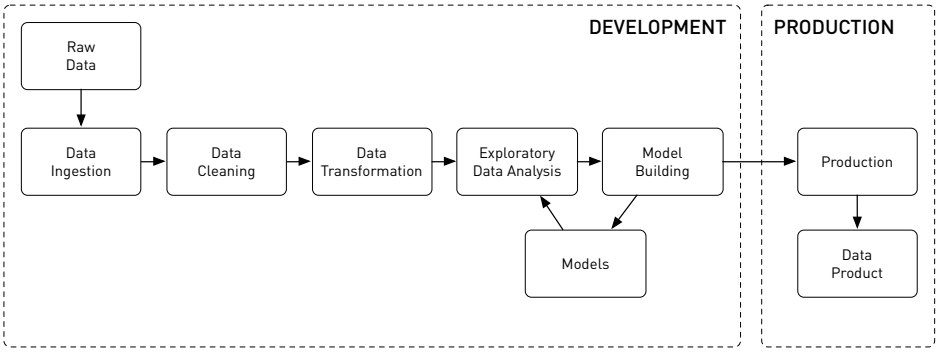
Needless to say, throwing hackery and resourcefulness at data is not a sustainable or indeed scalable approach for most organisations. There's an analogy here with the early days of the World Wide Web, back in the late 1990s, when big corporations began investing in 'webmasters'. The purpose of a webmaster was to build and maintain the company web site, and an ability to write HTML was the main, if not the only, prerequisite for the role. Some companies were lucky, and hired someone who quickly integrated what they were doing into the processes of the wider organisation. Others found out the hard way that coding in HTML was a necessary but by no means sufficient qualification for building and sustaining a web presence. Over time, standards of best practice emerged and found their way into papers, conferences, courses and professional qualifications.

So, turning back to data science, what does best practice look like right now?

---

[1] O'Neil & Schutt, *Doing Data Science* (2013) O'Reilly Media and Provost & Fawcett *Data Science for Business* (2013) O'Reilly Media are good starting points.

At a very high level, there is an emerging model[2] of the end-to-end data science modelling process that looks like this:



Development is where most data scientists spend majority of their time; it is the process that generates the insights, the breakthroughs. Development starts with ingesting data from external sources, cleaning and preparing it for analysis – abstracting from it or otherwise transforming it. Then there is the process of exploratory data analysis, where a variety of tools and visualization techniques are brought to bear in understanding the data under investigation, and ultimately building and refining models to investigate hypotheses and arrive at a suitable representation of the underlying phenomena with sufficient predictive performance[3].

A robust development process should possess the following characteristics:

- Organised - data is ingested, stored, transformed and managed in a systematic, structured way.
- Open and collaborative – people can work on models together with a common toolset in a shared environment.
- Versioned - models are versioned so that it is easy to keep track of them and determine which model does what.
- Elastic – it's possible to spin up computing resources for larger-scale training and evaluation tasks as we need them.

The tools provided to support the development process should:

- Be powerful – enabling, flexible and not restrictive. They should be as powerful as programming languages.
- Be easy to use and learn – they should be designed around proven, familiar concepts.
- Be open – as in open source – giving the benefit of the expertise of the

[2] The emerging model of the data science process is loosely based on a chart from O'Neil & Schutt (2013), p. 41.

[3] Sandtable's approach focuses mainly on explanatory modelling, in contrast with predictive modelling that dominates much of Data Science. For an excellent exposition on the differences, see Schmueli 'To Explain or to Predict?' (2010) Statistical Science, Vol 25, No 3, pgs. 289-310.

wider community and an opportunity to contribute back.

The process should support and be supported by an enabling culture within the team that is built around collaboration, openness and agility.

Insights are valuable, but what most organisations investing in data science need are **data products**[4], that can be used repeatedly and improved upon to deliver sustainable value. These products are best thought of as the end result of a manufacturing process (production), with the following characteristics:

- Repeatable - the same product can be made, again and again, with variation only as intended.
- Scalable – the product can be built using larger and larger quantities of data.
- Maintainable - the product can be kept current by updating or re-releasing it when new data comes in.
- Auditable – it is clear how the product generates its results from its inputs, and all of its inputs are traceable.
- Quality assured - the product will reliably do the job it is required to do.
- Available - the product when can be accessed when needed.

The rest of this paper describes the platform that Sandtable has developed in order to support the processes of a data science modelling project from development to production.

INTRODUCING THE SANDTABLE MODEL FOUNDRY

The Sandtable Model Foundry has been created in order to provide the features and tools needed by both sides of the data science process: development and production.

The Foundry is a Cloud-based platform, architected as a set of services that, together, form a workspace in which a single data scientist or a team of data scientists can operate effectively. The goal has been to create a platform that strikes the right balance between prescription and flexibility in supporting data science work.

The core organizing principle of the Foundry is the notion of a **project**. Once

[4] 'A data application acquires its value from the data itself, and creates more data as a result. It's not just an application with data; it's a data product. Data science enables the creation of data products.' from Loukides 'What is Data Science?' (2011) O'Reilly Media. An example of a data product would be a recommendation engine used to suggest new products to visitors of an online shop. Well-known data products include Google's PageRank and LinkedIn's 'People You May Know'.

created, a project becomes a hook around which a set of services can be grouped and used.

The services provided by the Foundry can be split into three components: the **data store**; the **model development environment**; and the **deployment platform**.

DATA STORE

The beginning of the development process starts with data.

Data is ingested, stored in and made available to other services via Sandtable's custom-built **data store,** Sandgit. The data store provides a repository-based schema-on-read store in the Cloud. Data in the store is immutable – transformation creates new versions, supporting a clear audit trail from the original data. Uniform resource indicators (URIs) are used to refer to data from other services used in the Foundry. Data in the store can be pulled to a notebook environment or loaded into a database for analysis and model development.

Other data stores can be used in place of Sandgit, such as AWS S3 or HDFS. For consistency, however, we suggest using a single data store across an organisation.

MODEL DEVELOPMENT ENVIRONMENT

The development environment is built around notebooks and pipelines, two powerful approaches to exploring data and developing models.[5]

Notebooks enable data scientists to keep work organised through data wrangling and exploratory data analysis, and have been hailed as 'the new spreadsheets'. We have integrated Jupyter[6] notebooks because they are interactive, support inline visualisations, and are language-agnostic. They are also effective environments in which to write data narratives for disseminating results. Jupyter notebooks are emerging as the de facto data science environment, particularly in the Python data science community[7].

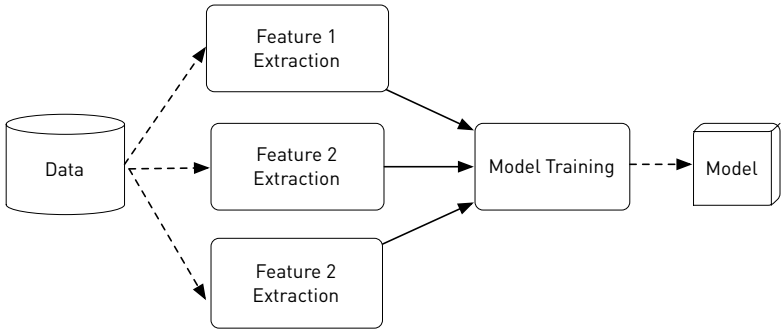
---

[5] For an overview of the approaches, see, for example: http://radar.oreilly.com/2015/04/more-tools-for-managing-and-reproducing-complex-data-projects.html

[6] See https://jupyter.org for more information. Note that Jupyter notebooks were previously known as IPython notebooks. The name 'Jupyter' is inspired by the three programming languages: Julia, Python, and R.

[7] For an overview of the Jupyter (IPython) ecosystem, see, for example: http://radar.oreilly.com/2014/01/ipython-a-unified-environment-for-interactive-data-analysis.html

In order to make the process of developing models efficient and reproducible, we have implemented a **pipeline development framework**, based on Spotify's Luigi[8], which supports the creation of complex data pipelines in Python[9]. This is mainly intended for model development pipelines, in particular data transformation and model training tasks, though it does have wider application. Luigi tasks integrate well with a variety of Big Data frameworks, including Apache Hadoop[10] and Apache Spark[11].



Example model development pipeline (data in; model out)

Both notebooks and pipelines can be published. **Publishing** makes notebook snapshots and pipelines available to other users for reviewing and comparison, and tracked for audit.

In order to train and test models, there is a custom-built **model execution service** - Sandman - that handles a set of standard tasks for developing models that conform to a straightforward API, including model training and exploration.[12]

The Foundry gives users the ability to **manage services and resources** attached to their projects. The associated resources are then available to data scientists from within the development environment. In addition to providing access to a custom-built service, the Foundry also facilitates integration with a wide range of open frameworks and services, such as:

- Apache Spark
- Apache Hadoop
- Amazon Redshift
- MongoDB

[8] Luigi is an open-source project released by Spotify, available here: https://github.com/spotify/luigi. Luigi comes with a set of tasks for working with systems like Hadoop, Spark, MongoDB, and AWS Redshift.
[9] Luigi pipelines are written in Python but R scripts, for example, can be easily integrated.
[10] https://hadoop.apache.org/
[11] https://spark.apache.org/
[12] This includes, for example, models built using the Python machine learning package scikit-learn (http://scikit-learn.org).

- MySQL

DEPLOYMENT PLATFORM

The final step in the process involves the deployment of models.

The Foundry provides a simple API so that models developed using the pipeline framework can be easily stored and versioned. The models can then be viewed in a **model library.** From there models can be compared based on training metrics and reports, and deployed to provide predictive services. The services can be configured to satisfy both batch and real-time requirements, and are accessible via RESTful APIs. This means that model results can be made available to, for example, interactive web-based dashboards, or to other applications, that are registered with the Foundry.

A DAY IN THE LIFE

In an attempt to bring some of the ideas we have been outlining to life, we've written a couple of stories that reflect our experiences of how data science can be at its worst – and what it could become with the help of the Sandtable Model Foundry. Specifically, the stories highlight the following benefits of using the Foundry:

- Maintaining a more transparent connection between data, model code and results
- Collaboration and sharing built in to the workflow (rather than using email)
- Finding the current state of work quickly and reliably
- Easily locating data sources
- Keeping results and charts together with analysis code
- Automatically saving work (and keeping a history of what's been saved)
- Easy access to more computing resources
- Easy access to open frameworks (such as Spark)

Both stories concern Amy, member of small data science team. The first covers a particularly bad day in her life, pre-Foundry. The second looks at a better day using the Foundry.

STORY 1

Amy gets in to work and logs in to her laptop. She needs to update the model of smoking behaviour in the UK she's been working on. She looks for the most recent version of the model in her local Git repository. It's been a while since she worked with it, so she thinks she can remember which branch she was using but has to take a few minutes to make sure. Once satisfied she's working on the right version, she looks at her code to figure out which data sources were used for one of the key functions in the model: the calculation of relapse rates[13]. Amy finds the reference to the source and locates it in her local folder using the search function. Now that she knows where the data is, she can start on her task for the day: working on a new version of the function.

She's already done some analysis work on this data source so she looks for it on her machine; eventually she finds a folder with some python scripts, charts and a couple of Excel files. She goes through all of them one by one to

[13] Relapse rates specify the rates at which smokers relapse back to smoking after an attempt to quit.

re-familiarise herself with what's been done. Once she's on top of these, she feels ready to tackle the new analysis.

For the main part of the analysis task, which involves a large dataset, she thinks to herself that she'd like to use Spark, but unfortunately doesn't have easy access to a Spark cluster. As a compromise, she takes a sample from the larger dataset and loads it onto her laptop.

She starts working on some scripts and before long she's generating results, which she outputs as a set of .png charts that open on her screen as the scripts complete and are saved to a folder. She iterates through a process of revise /run / review results a few times until she's happy with the analysis. She wants to share the results of her analysis with her colleagues so she attaches the charts to an email and fires it off.

The next thing she needs to do is incorporate the results of her analysis into a new model version. To do this, she creates a new version of the model code and commits it Git. She then has to write a series of inter-dependent scripts, based on her analysis, that will take data from the data source file, transform it, and use it to derive a relapse function for different segments of the UK population. It's a bit fiddly, and she realises she could probably re-use some things she's written before, but it seems easier to write from scratch. About two thirds of the way in, her local machine hangs before she's done her first Git commit. She has to start again, almost from scratch.

She tests the new scripts and spends a fair bit of time de-bugging, mainly because of their inter-dependencies and references to the local filesystem. It all feels rather fragile. She runs the model to generate a new set of results. Because of the uncertainty in her model, it has to be run multiple times and the data from each run aggregated together. She's written a script to handle this process but her machine only has four cores so it can only execute four runs at a time. To make matters worse, there's an error on several of the runs that only gets picked up as the results are being processed, so she has to run everything again.

To compare the results with previous models, she has to figure out what the most recent relevant models were called, which folders they are kept in, and bring up their associated results data. She manages this with the help of a model versions text file she maintains by hand in the top-level folder for the project.

Having convinced herself that the model is a more or less natural evolution of previous models, she runs off a standard set of reports and sends them over to her client, with a lengthy email describing the model update and explaining why it should replace the previous model version. After she's pressed 'send'

on the email she relaxes slightly, but that doesn't stop her from re-reading the report.

STORY 2

Amy gets in to work and logs in to her laptop. She needs to update the model of smoking behaviour in the UK she's been working on. She opens up a browser and logs in to the Foundry. This automatically brings up the last project workspace she was working in, which is a model of smoking behaviour on the UK. If necessary, she can easily switch to other projects through the menu, but today she needs to continue work on the smoking project so she's happy where she is. The project workspace shows the project dashboard: statistics for the project, such as the number of models created, and an activity log so she can see what has been done on the project by the team over the past few days (and beyond).

She clicks on the Notebook tab to take a look at the notebooks she has been working on. She's been working on a new approach to deriving smoking relapse curves from data, so she goes down the list to find the notebook she's stored that work in and opens it up. She scrolls down to the bottom through all of the code and charts she's been working on to start a new section. For this new piece of analysis she needs to bring some new data into the development environment. She received the new data yesterday and checked it into the Sandtable data store, Sandgit. She loads the new data directly into Spark from Sandgit and starts work.

She has a good idea of what needs to be done to the data to derive the functions she needs, so it's a simple task to write the scripts to transform the data, perform the derivations and plot the results. Once she's happy with the work, she decides to publish a snapshot so there's a record of this step and so that she can share her results for review with other data scientists on the team. A link to the published notebook snapshot is automatically sent to the project's Slack channel.

Now that she's happy with her analysis, she wants to use the new dataset to derive a relapse curve for part of her model population. To connect the data with the model in a sustainable way, she sets about creating a new pipeline: she clicks on the pipeline tab, opts to create a brand new pipeline (as opposed to editing a previously-created one) and starts writing the pipeline code. She's done the hard analytical work so this bit is easy. Once she's made the edits, she tests the pipeline in her development environment on a sample of the data to make sure it behaves as expected. Once she's happy, she publishes the pipeline so that others can see what she's done, and she has a

permanent record of the connection between model and data.

She now selects and runs the pipeline on the full dataset and generates a new version of the model. The new model appears staged in the model library, and she can see how it compares with other models on the relevant metrics (for this project, these metrics include quit attempts and smoking prevalence in the population). The metadata associated with this version of the model contains a record of which data and pipelines are linked to it. Once she is happy with the progress, she deploys the model to provide a predictive service available for use by the project's dashboard application. Finally, her client is automatically sent a message letting them know that the dashboard (and the model) has been updated.

Email us with comments at foundry@sandtable.com.