Developing reproducible analytical pipelines for the transformation of consumer price statistics: rail fares

Authors: Matthew Price and Diogo Marques, Office for National Statistics

Paper submitted to the Meeting of the Group of Experts on Consumer Price Indices, June 2023

Abstract

At the Office for National Statistics, we are transforming our consumer price statistics by introducing alternative data sources such as scanner data. This paper discusses the process of developing a new production system that can integrate these data as part of our consumer price index production.

Firstly, we will discuss the choices made for the infrastructure of the project, including choice of platform and language, and how this has been designed to aid research as well as ensuring an efficient and user-friendly production system.

We then discuss how best practice guidance for reproducible analytical pipelines (for example, code structure, testing, version control and code deployment) have been implemented in the project.

Finally, we focus on more specific areas within the end-to-end system. This includes the steps taken for data engineering, including the standardisation of data. We will also cover the choices we have made to implement these within our existing production round, including the steps needed on an annual basis to reset the basket and allow for the introduction of new consumption segments.

1 Overview

At the Office for National Statistics (ONS), we are currently undertaking an ambitious programme of transformation across our consumer price statistics, including identifying new data sources, improving methods, developing systems and establishing new processes. These new data sources allow us to measure inflation from an improved coverage of price quotes, introduce weights at a lower level of aggregation than before, and allow for automated data acquisition.

This year, the Consumer price inflation, UK: February 2023 release was the first time our headline measure of inflation included these new data sources, using expanded data on rail fares. Over the next couple of years, we are looking to develop the use of these new data sources to include additional categories such as used cars and groceries, as well as expanding our new system to include the processing of the existing data collection. These existing data will continue to be used when they cannot be replaced by scanner or web-scraped data, such as for small independent shops and service providers who do not have a website.

In designing our production systems, we needed to ensure that they were flexible enough so that future categories of data could be incorporated into the design with minimal additional development. In this paper, we will focus on how the system works for rail fares but still with an eye to future development. In particular, this includes:

- The team structure and skills required for the transformation project
- The key considerations for the project's infrastructure and how the platform is designed to facilitate all aspects of development, research, and production
- The steps taken in engineering supplier data to produce standardised datasets
- A technical summary of how the pipelines used to produce indices are structured
- The end-to-end processes for producing rail fares indices in production
- How coding best practice is implemented into the project to ensure quality systems

2 Team structure for the Consumer Prices Transformation project

This Consumer Prices Transformation project is a multidisciplinary project within ONS, comprising of distinct teams who are experts in their areas.

The owners of this project are the Prices Division within ONS, who oversee the production of the UK's consumer, business, and house price statistics. The main project management hub sits within Prices as well as several teams dedicated to the transformation project. The Data Transformation team is responsible for acquiring and managing the new alternative data sources. This team is also leading on building in-house web scraping capability. The Methods Transformation team is responsible for all aspects of methods research in the transformation project, including the publication of our research updates. The teams are largely resourced by statisticians/methodologists, with some additional specialism in data science. The Business Change team then works with the CPI Production teams to implement the agreed systems into the current ongoing monthly production of the consumer price inflation measures.

Outside of the Prices division, there are other teams working on delivering the technical side of the project. Some of the key roles are:

- Business analysts who spec out the various requirements from Prices for both the underlying infrastructure and more general functionality
- Methodologists who specialise in index number theory and support the Prices Methods team in their research
- IT architects who design the overall system based on the Prices use case and ensure it fits in with the wider ONS technology base
- Data engineers who manage and build data processing pipelines to ingest and standardise data for downstream production processes and wider research, build data validation tools and manage part of the data strategy of the project
- A data acquisitions team who, along with the Data Transformation team in Prices, manage the relationship with the data supplier and engage with new ones

- Data science/software engineers who are responsible for building the data processing pipelines based on the requirements specified by Prices
- Testers who ensure that the final development meets the specified requirements

Together, these teams ensure that the final system meets Prices requirements and is developed in line with other ONS infrastructure to enable sustainable delivery in future.

3 Technical decisions for the project

3.1 Identifying key requirements for the platform

ONS' cloud strategy follows the Government Digital Service (GDS) Cloud Strategy. The GDS Cloud Strategy promotes a "Cloud First" policy, whereby public sector organisations should consider and fully evaluate potential cloud solutions before considering any other options. This strategy strongly advocates the use of infrastructure as a service (IaaS) and platform as a service (PaaS) offerings from cloud providers.

For the UK ONS Consumer Prices Transformation project, there were several key factors that needed to be considered when determining the most appropriate cloud platform to host the project on. Alternative data sources are inherently large data sources and therefore the project needed access to a scalable data storage solution that allows resilient and efficient storage and tools to query big data. To enable big data processing using Apache Spark, we also needed to have a distributed computing system that allows the use of more complex data transformations (such as applying complex methodologies to data) to enrich and aggregate data in a reproducible manner (i.e. through code that follows the RAP principles).

For the production teams, there is also a need to have dashboards in place to cover processes such as validating the data being delivered to ONS and visualising the calculated indices. Additionally, it was identified that interactive web applications would be required to enable accessible user input on enriching and validating certain data sources (such as in manually labelling, flagging, and amending observations in various data sources).

The final requirement was to have the functionality for researchers to perform ad-hoc analysis and data manipulations to enable ongoing transformation activity for future data categories alongside live production.

3.2 Use of environments to separate workstreams

To allow the stable processing of the production indices while allowing continued development, the project makes use of several environments to separate the different stages of work. The configuration for each environment is managed through code. This means that it is trivial to ensure that each environment has the exact same configuration of resources with the only difference being the access permission groups being tailored to the sensitivity of data present.

There are four types of environments within the project as outlined in Table 1.

1) Develop, where the users have complete freedom over all aspects of their individual sandboxes which can be created and destroyed at will. Because of this, this environment is unique in that it is not managed by the infrastructure code. This space is where the development of pipelines occurs, and all data used is synthetic.

2) Test, the first environment where all the separately developed systems should always work together. Here the test team will perform their tests to ensure that the systems and platform meet the agreed business requirements. Again, all data used in this space is synthetic.

3) Pre-Production, the first environment that contains live data and is identical to Production. This is to allow further end-to-end testing like in the Test environment, but now with the real data.

4) Production, the environment where all delivered data (whether it is being used in production or research) is ingested into the platform and the data processing pipelines run automatically to produce the production price indices. As this environment contains the up-to-date classifications and mappers, it is also where the research team do their work exploring the data and investigating new methodology. Permission groups in this environment are configured so that the researchers are not able to see the current month's data for

production data sources, nor be able to interact with any of the outputs from the various production data processing pipelines.

Table 1 Summary of the different environments used.

Environment	Description	Main users	Data used	Stability
Develop	Sandbox where teams have full access to explore, test, and do their work.	• Software engineers • Data engineers • Infrastructure engineers	Synthetic	Not stable
Test	Test environment where all systems can work together allowing testing of individual and multiple systems.	• Testers	Synthetic	Stable
Pre-production	Where testing on live data can occur to ensure changes will be stable before moving into production.	• Business change team	Production (data duplicated from the prod environment)	Very stable
Production	Where production occurs via automated scheduling of pipelines. Also, where research on data and methods can occur.	• Production team • Research team	Production (where all new data is ingested, permissions set to prevent researchers from seeing "current month" data for production datasets)	Most stable

This structure for the environments creates an intuitive development cycle for new methodologies and pipelines (Figure 1). First researchers do an initial investigation on a given data source in their enclave within the production environment. From their analysis and recommendations, they work with the business architects to write out new requirements to pass to the various development teams. These are then actioned and implemented by the relevant team in the developer sandboxes. Once that has undergone the appropriate review, new or updated pipelines are released into the test environment, where they are end-to-end tested. Success here results in runs in the pre-production environment, and if successful the new pipelines will be promoted into the production environment as part of the annual process (see Section 6).

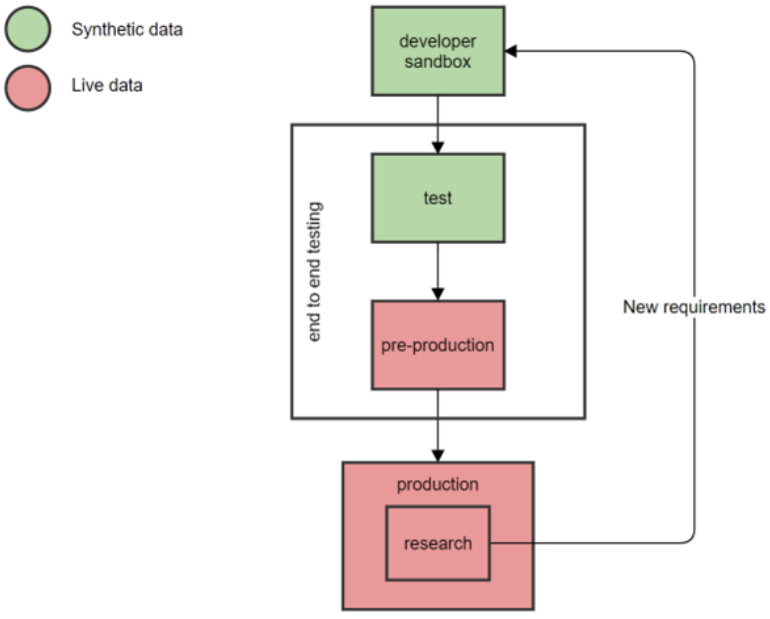

Figure 1 Development workflow across environments

4 Data Engineering

To prepare the data sources for use in producing indices they undergo several data engineering stages; data ingestion, validation, standardisation, and engineering for general use. The data ingestion stage brings the data into the cloud project, performs virus checks and decryption where applicable, and moves the data into an output bucket in both Production and Pre-production environments. Once the data enters the environment, the workflow orchestration tool AirFlow will start and trigger the data validation pipeline. If none of the validation fails, the data engineering pipeline is initiated and it standardises and appends the latest data to a table. The end result of the data engineering work is to create staged data ready for the data processing pipelines (section 5). All engineering steps have extensive logging and will also send appropriate notifications to external communication tools so that users can action any alerts.

4.1 Ingestion

The ingest process uses Apache NiFi, a tool for automating the movement of data between disparate systems, for ingestion into both our cloud and on-prem systems. Following that, an in-house IT solution is used that does data integrity checks with supplied manifests, decryption, de-compression and virus scan.

4.2 Data quality – validation pipeline

We have built an in-house data validation package using Python and Spark to replace what were previously weekly manual data checks. This package has been built off the back of the need to automate manual data checks and standardise them using tested and scrutinised methods.

The validation checks can either be set to warning or error level. If a validation check raises an error the pipeline stops the process and notifies the data team instantly of the issue, minimising delays in requesting a redelivery from the supplier. A warning can be used instead of an error to keep an eye on trends or notify the production team of a change in the data that may require an action, e.g. with rail fares, a new value in the station column could mean the station mapper that links stations to regions may need revising. However, warnings do not stop the process from completing.

We considered a range of different tools when building this and used PySpark because it allows us to create one method that can scale to work on any data size, in our case both on the smaller regular deliveries but also can be used on datasets of over 10TB, without the need to build and test separate methods. The validation checks include schema, min/max row count and expenditure and other more nuanced checks.

4.3 Standardisation

The main data engineering pipelines read and standardise the data from the raw files creating a continuous time series in the most accessible format as a table, which has both the data and column names in the format expected by downstream systems. The pipeline also performs some secondary engineering when the raw data doesn't contain all the variables needed to produce the indices. It creates new columns that are either derived by applying rules to existing columns or mapping data from other sources, for example, an external mapper to assign train stations to Government Office Regions (GORs). Finally, the pipeline also adds junk columns, which are Boolean columns used to filter out observations. While this can be seen as a data cleaning step, it doesn't actively remove data but allows the user to choose to remove these columns as part of the downstream processes (see data processing pipelines). The junk columns each apply a separate rule so filtering rules can easily be added/removed, for example, whether to keep transactions that pertain to "business only" travel (that are out of scope for CPI).

Further reading around some of the data engineering rules applied can be found in our previous article on using transaction-level rail fares data to transform consumer price statistics.

4.4 Data quality – insights dashboard

To ensure data users always have the best understanding of the data that they're using, we have built a dashboard with visualisations that provide an up-to-date overview of key metrics and more niche insights that could aid research, for example, expenditure trends (nationally and by region) and null rates in the data. Providing these visualisations upfront has many benefits, it provides powerful insights achieved through complex SQL queries to users that don't have advanced SQL knowledge. It also saves multiple data users running similar queries on terabytes of data, saving time and computing resources. Powering the visualisations we have a set of materialised views that

contain the SQL queries needed for the visualisations, either in full or to pre-aggregate the data from the terabytes large tables to a size that's more manageable for the visualisation tool. Both the views and dashboard are refreshed within a few minutes of new data being engineered and appended onto the main table.

4.5 Resilient and efficient storage

To ensure downstream processes use their resources and data most efficiently, all tables are partitioned, and we apply other methods such as clustering where needed. ONS back up all raw data in a separate service and all tables have a type of version control that allows any table to be rolled back to a previous snapshot, or a deleted table to be reinstated within an agreed window of time.

5 Data processing pipelines

The data processing pipelines are the pipelines that transform the staged data sources (or one of the derived tables detailed in Table 2) to further enrich, clean and aggregate the data to produce a full set of indices from the elementary aggregate level to the overall headline inflation rate. More information about this process can be found in Section 6: Producing indices for production.

5.1 Pipeline code structure

The data processing pipelines serve dual purposes of being used by the researchers to inform decisions on what methodologies to implement in production, as well as the pipelines used in production. To allow this flexibility each pipeline is controlled by a configuration file which contains settings for all options in each pipeline. For example, in the data cleaning pipeline, the configuration file allows the turning on of several different outlier methods and associated parameters. In the elementary indices pipeline, over 20 permutations of index methods can be applied to the data. Researchers can explore a wide parameter space at will to make the most informed decisions by changing the values in this file at run time. In production, the configuration files are locked down to only the chosen methods for publication with no option to change the values.

Each data processing pipeline is designed to the same template. Following this template allows us to quickly setup new systems when required for new methodologies or data sources. Furthermore, it means developers and end users can familiarise themselves with a new pipeline based on their knowledge of other pipelines in the project.

The general structure of a pipeline is to have a “main” script that operates as follows:

1. Take a minimum amount of command line inputs (file paths for a “user” and “backend” configuration file, the name of the user running the code [for audit purposes], and the desired level of logging [as code contains substantial debug logs that can be used to diagnose issues]).
2. Read in and validate the “user” configuration file which contains options such as what date source to use, the date range of the data to use, which methods to apply etc.
3. Read in and validate the “backend” configuration file which contains information that would not need changing between pipeline runs such as the table paths for inputs and outputs or the stratification for a data source.
4. Read in all the data needed for the pipeline (using the information from the various configuration files).
5. Pass all relevant configuration parameters as well as the data through to the “core” pipeline code that will perform all the data transformations of the pipeline (e.g. any pre-processing, applying any methodologies, performing any aggregations).
6. Add a unique run identification to the output from the previous step.
7. Append the output to the appropriate table (where the unique run ids will allow distinguishing data between runs).
8. Append key information (e.g. the version of the pipeline, the user and timestamp for the submitted run and the configuration files used) about the run to a separate pipeline run log table to allow auditing of pipeline runs.

Developing the pipeline code in such a manner also has the added benefit of separating all the methodological parts of the code from the input and output code. This is of importance, as given all the code is written in open-source languages, the only parts of the code that would not be immediately portable to a different cloud platform are the parts that perform the input and output functions. By placing these at the very start and end of the code only, the

systems can be readily ported to another platform without having to disentangle these I/O processes from deep within the code base. It also means that retesting a pipeline in such a situation could be performed with ease as the underlying methods' code will not need altering to account for any code changes required for the different cloud platform.

5.2 Pipeline tables

To have a clear audit trail for all the data generated by the pipelines, several tables are generated during the production of the indices (as discussed in section 6). A summary of these tables and where they are created can be found in Table 2.

Table 2 Overview of main data tables produced by systems

Table	Description	Produced by
Staged	The standardised form of the data delivered by suppliers.	Staging of raw data files delivered to ONS by the data engineering pipelines
Cleaned	The data after applying any pre-processing, with additional columns denoting whether a row is identified as being an outlier or not based on the chosen data cleaning methods.	Data cleaning pipeline
Invalid strata	A mapper that denotes whether for each elementary strata if an index is calculable at the start of the annual round	Invalid strata pipeline
Processed	The aggregated monthly price and quantity observation for each unique product. This is the data used to calculate the elementary aggregate indices.	Elementary indices pipeline
Elementary aggregate indices	The monthly price indices for each elementary aggregate.	Elementary indices pipeline
Aggregated indices	The full set of indices from elementary aggregate, aggregated all the way up to COICOP1. Includes additional information on growth rates and contributions.	Index aggregation pipeline

Storing the data like this allows deeper interrogation by researchers or production users if anything is identified in the final output. As an example, if there was a sudden drop in a particular strata's index, all the data that fed into that value can be explored from the initial data as provided by the supplier, to the outcome of the data cleaning methods on the data, through to the final monthly price and quantity observations for each product in the strata.

6 Producing indices for publication

Given we will restrict the number of consumption segments to only include those that have representative items in the traditional collection, we will also need to continue the process of annually updating our basket of goods and services to include new consumption segments to reflect changing consumption patterns. The production system has been designed to be run in an annual round, which initialises the data for subsequent monthly rounds, which are used to produce the monthly outputs.

The workflow outlined in Figure 2 primarily relates to the processing of the rail fares alternative data sources but forms the standard workflow that will be applied to future data categories (e.g. used cars or groceries), and allows us to integrate alternative data sources more readily with traditional data sources. More detail on our hierarchies and how we are integrating new and traditional data sources is discussed in Integration of alternative data into consumer price statistics: the UK approach.

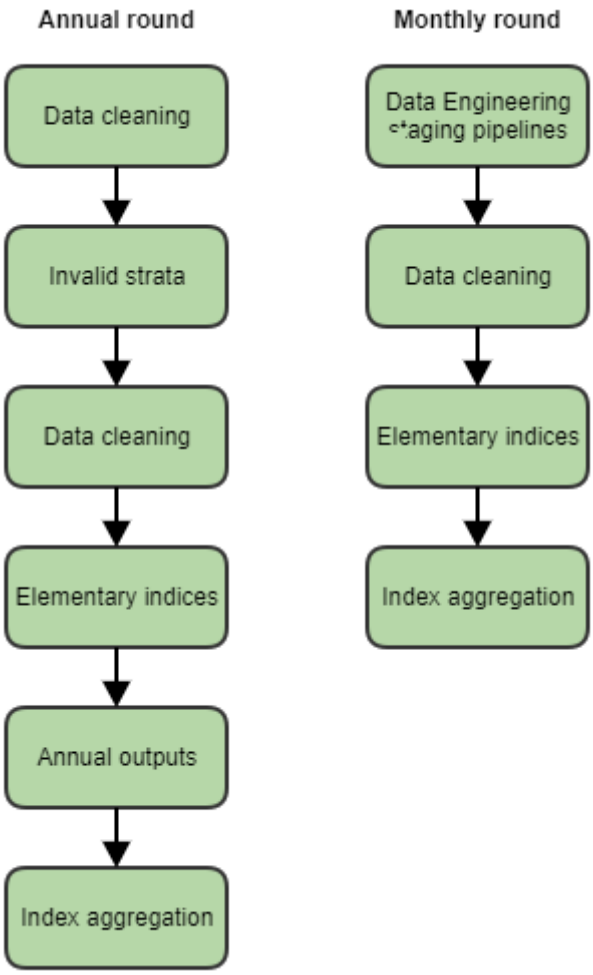

Figure 2 Overview of pipeline workflow for the annual and monthly rounds.

6.1 Annual processing

We will only introduce new consumption segments (or new alternative data sources retailers) where we already have a 25-month period of data so we can consistently use a 25-month window for all alternative data based elementary aggregate indices. This means that every year the consumption segments will essentially be reset, the data re-classified to the new consumption segments over a historic 25-month period, and then the new index for the following 13 months would splice onto this recalculated historic index.

The annual round therefore initialises a 25-month historic series based on the classification system in the current year. For example, given a January 2024 base, the system would initialise the data from January 2022 through to January 2024. This is important if any classifications have changed or any new consumption segments have been added to, or removed from, the CPI basket of goods and services.

A 25-month period of data are first cleaned (data cleaning pipeline – see "Outlier detection for rail fares and second-hand cars dynamic price data" for details on our chosen data cleaning methodology for rail fares) before it is decided whether a stratum index can be appropriately calculated based on the presence of bilateral pairs of price relatives (invalid strata pipeline). If no bilateral pair exists in the January of the current year, this is considered an invalid stratum as it's unlikely that an index will be calculable in the following months. This is output as a mapper file that can be reviewed, monitored, and amended if necessary.

The data cleaning pipeline is then run again, removing any data within the 25-month historic series that pertain to an invalid stratum. Now our data should only contain products from which we would want to calculate an index.

The cleaned data are then used to produce elementary aggregate indices (elementary indices pipeline) for the initial 25-month window that will be used to splice onto during the monthly round. For rail fares we use the GEKS-Törnqvist index method (see "Introducing multilateral index methods into consumer price statistics" for a technical explainer).

The processed data (output from the elementary indices pipeline) are then used in the annual outputs pipeline to calculate the stratum-level weights for the year (when created from the data directly – some stratum-level weights are derived offline using existing methods and ingested to be used alongside the other weights).

The aggregation pipeline can then use the output from the elementary indices pipeline and the annual outputs pipeline, along with weights, to aggregate the indices and impute any remaining missing indices in the 25-month historic window.

Now we have initialised our historic back series for all the data and produced the required weights, we can use this information in the monthly round.

6.2 Monthly processing

The first step in the monthly round process is where the data cleaning pipeline cleans the latest month of staged data using the new staged data and the cleaned back series. Next, the elementary indices pipeline uses the cleaned data to produce a new index based on the most recent month of data along with 24 historic months of data from the 'processed' data output in the previous month. This produces new elementary indices with a 25-month window i.e. the monthly round in February 2024 would initially produce an index from February 2022 to February 2024. These are then spliced onto the series' produced in the index aggregation pipeline from the annual round (in February) or the index aggregation module from the monthly round (in the remaining months for that year), to ensure no revisions occur.

These elementary indices are then passed on to the aggregation pipeline to aggregate to higher levels, using the previously constructed weights. Indices are all re-referenced to Jan=100 to align with, and allow aggregation with, traditional data sources. Any missing index values are imputed. For the UK, missing stratum indices are imputed based on the consumption segment level index. Missing consumption segment level indices are imputed based on the COICOP 4 level index. Missing COICOP indices are imputed based on their nearest parent. More detail on our aggregation and imputation methods can be found in: Introducing alternative data into consumer price statistics: aggregation and weights.

6.3 Research capabilities

Alongside the processes for creating the publication indices, researchers need the ability to mimic the production cycle in order to perform analysis on the impact of proposed changes to the output inflation numbers. Reasons to perform such analysis include; to see the effect if different methodologies are used, to investigate the incorporation of a new alternative data source or category, or to check the effects of reclassifying data. The split permissions set in the production environment mean that researchers can access all the data they need to perform this analysis without interfering with the production tables in a space where they have full control over what additional data and methods they utilise.

7 Implementing best practices to quality assure systems

The data engineering and data processing pipelines built for this project have been done in line with the Reproducible Analytical Pipelines (RAP) strategy defined by the UK Government Analysis Function (GAF). This strategy outlines the software engineering and infrastructure best practices to ensure that pipelines are reproducible, auditable, efficient, and high quality. The following sections details how the UK ONS Consumer Prices Transformation project meets and exceeds the minimum standard for a RAP.

7.1 Implementing RAP for the code

7.1.1 Minimise manual steps

Due to the size of the data being used there is naturally little opportunity for tools like spreadsheets, which would involve manual steps, to be present in the workflows. The only area where manual steps are involved is in the creation of mapper files that modify behaviour within the pipeline. For example, we may use a mapper to define the link between lower-level ONS-specific aggregates (described as consumption segments) to COICOP5. This is a manual process as these mappings can be revised on an annual basis as part of the annual basket refresh (see section 6). To minimise any unintended consequences, the processes for creating these files are documented and these inputs are

validated when loaded into the platform to ensure any issues are caught. Using mappers also allows us to avoid annually updating hard-coded references, reducing the risk of errors by having to update the code directly.

7.1.2 Built using open-source software

All the systems are written using the Python language. Due to the size of the data being processed, the pipelines are also built to leverage distributed computing using Apache Spark and its Python API, PySpark.

7.1.3 Quality assurance processes with peer review

Business analysts work with the Prices methods and business change teams to define new functionality as requirements with clear acceptance criteria, captured via Jira tickets. Each requirement is then coded by the relevant data or software engineer, and the code is then peer reviewed by other developers within the teams. Doing this allows both quality assurance and dissemination of technical knowledge across teams.

7.1.4 Use of version control software

All code is kept in git repositories on the GitHub platform.

The transformation work requires research and impact analysis to be carried out using a breadth of methods before selecting the final publication method. Because of this, during development new methods are added via "Research" releases of the pipeline. Once the methods have been determined and accepted by stakeholders, those methods will be formally tested via "Release candidate" releases, which after passing will be promoted to "Production" releases. This workflow is managed via the use of Jira tickets to ensure traceability in what functionality is available in each release.

Table 3 Overview of releases used by for the Consumer Prices Transformation project

Release	Frequency	Description
Production	Annual	These are versions of the pipeline that have been fully tested for all methods required to produce publication outputs. These can only be changed in exceptional circumstances due to the annual production round of producing the indices.
Release candidate	Annual	These pipeline versions are created prior to the release of a new production version of a pipeline, as the pipeline must undergo formal testing before it can be signed off as ready to enter production. To enable this, these release candidates of the code are produced for use by the testing team.
Research	Monthly	These are versions of the pipeline that have undergone testing by the developers, but not the formal test process that the production releases undergo as the methods introduced may not be used in production. These versions are produced every month between annual rounds for a pipeline that is in active development.
Bug patch	Ad-hoc	These are versions that are created only when required due to bugs being identified in a Production or Research version of a pipeline. For Production bug patches, the code must undergo another round of formal testing by the test team. Research bug patches only undergo standard developer testing.

This development of code is managed through a GitFlow workflow. Using the GitFlow branching strategy aids the development of "Production" and "Research" releases simultaneously (see Figure 3) and creates a dedicated space for functional testing to be conducted. Using git tags to "tag" pipeline releases across these branches, new versions can be released to the processing platform via CI/CD (see section 7.1.9).

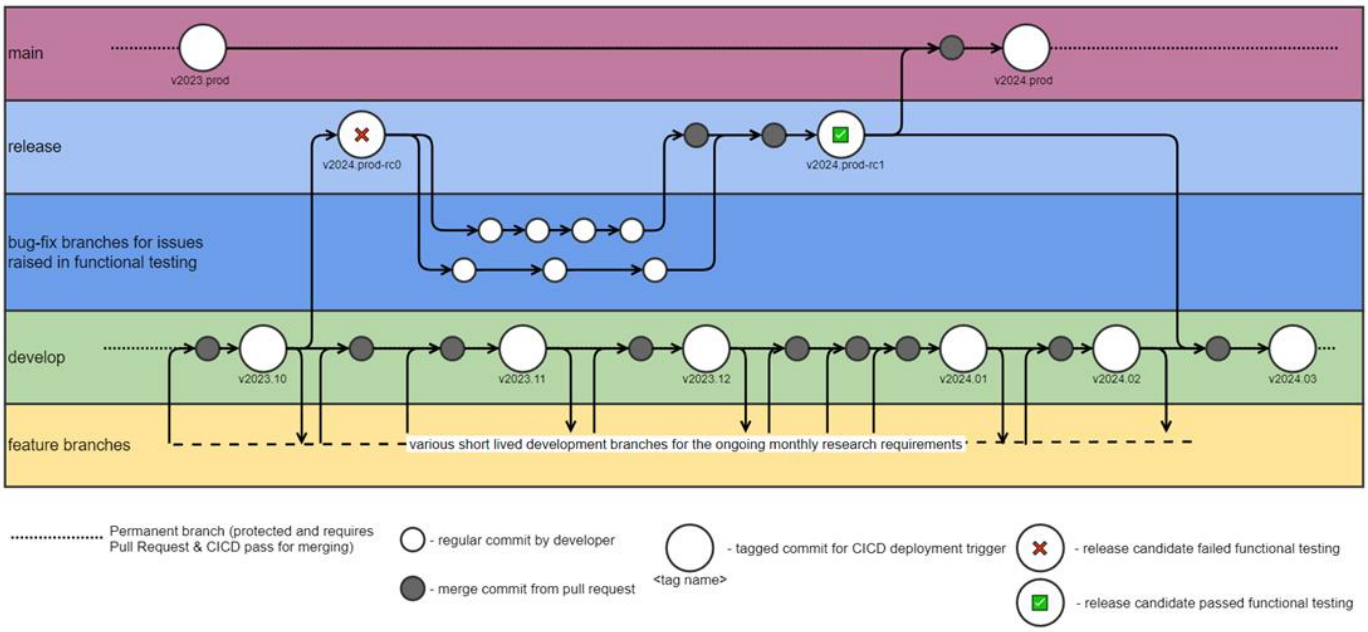

Figure 3 Example of the GitFlow workflow.

7.1.5 Open source of code and data used

Currently, the code is not publicly available due to the confidential information that must be present in some of the code. There is however the intention to release code packages of the various methodologies in the future. The UK data used in calculating our consumer price inflation statistics are commercially sensitive and cannot be shared.

7.1.6 Follow existing good practice guidance set by department

The Consumer Prices Transformation project builds on the best practice guidance set within ONS, but as the first transformation project of its kind in the organisation, it is also setting best practices. To ensure the quality of the choices being made on the project we have engaged with the GAF team to provide external assurance.

7.1.7 Fully documented code

All the project code is documented within the pipelines, with this supplemented by documentation covering a range of aspects of the project from platform design, data dictionaries to code release patterns kept in a central project wiki hosted on the ONS intranet.

7.1.8 Testing

The pipelines and systems are all thoroughly tested in several ways. For each pipeline, all the code undergoes:

- Unit testing – ensuring that each individual function behaves as expected. These tests are typically constructed by the developer.
- Component testing – ensuring that each “module” of code behaves as expected. A module can be thought of as an entire method which is made up of several functions. In the instance where a method is a single function the component and unit test would be the same thing. The data used for these tests is provided by the appropriate methodologist.
- Module integration testing – this ensures that the sequence of modules interact with each other as expected. An example would be seeing that a pipeline can run end to end (but not confirming if the results are valid).
- Functionally testing – this confirms that each pipeline and its output meets the defined requirements as written. These tests confirm for example, that for a given input data source and input parameters the pipeline produces the correct expected output (including negative testing).
- Integration testing – confirming that a sequence of pipelines all run together. For this project, as the output of one pipeline feeds directly into another, these tests would confirm the entire end-to-end process from new data being received through to the production of indices.
- Acceptance testing – the final checks to ensure that the entire processes meet the needs of every defined business scenario. For example, “new data are delivered, are new indices produced and alerts sent” or “bad data are delivered, are the systems stopped and alerts sent”.

Several of these test stages are automated using the continuous integration tools.

7.1.9 Continuous Integration and Continuous Deployment (CI/CD)

Using CI tools we automate the checking of unit, component and integration tests for every change made to each codebase prior to being incorporated into the pipeline. These checks also cover ensuring consistent code style and code documentation. The CD tools for the project ensure that the packaging and deployment of code to the production platform is automated in a reproducible manner. The deployment of code is triggered by the presence of tagged releases of the code base.

7.1.10 System logging

All pipelines utilise a range of logging across the range of severities (i.e. debug, info, warning, error) to allow robust traceability of processes occurring. Information logging includes information such as how many rows are removed by a filtering command to allow quick diagnosis of data issues if more rows are dropped than expected, or the schemas of data being read or written to tables. Debug statements are switched off in systems by default but can be turned on in the case of runs that error to allow better diagnosis of issues. Warnings are used to alert for behaviour which while not terminal is not expected behaviour, whilst errors are reported when issues occur that prevent a pipeline from proceeding further. These logs are captured both within the cloud platform, but also relevant alerts are sent to the production team to inform them of successes and failures of the systems.

7.2 Implementing RAP for the platform

7.2.1 Automated pipelines

The project makes use of Apache Airflow to manage workflow orchestration in several ways speed to minimise user input and error, for the normal production of monthly indices. Event driven workflows are in place to handle the staging into tables of new data when new data files are ingested onto the cloud platform. Time based workflows are used to handle the automated scheduling of the production of indices by passing the staged data through the several data processing pipelines required for producing indices.

7.2.2 Reproducible infrastructure

The entire cloud platform infrastructure is prescribed using Terraform. By doing this all aspects of the cloud platform from the tools and datasets to user permissions are prescribed in code and can be easily audited, reproduced or expanded.

8 Future developments

The roadmap for the transformation of UK consumer price statistics has several data categories in active development. These data categories will reuse the workflow as outlined in this paper for producing rail fare indices for producing their own indices. Due to the modular nature of the approach we have taken in our systems design, it will be trivial to add extra pipelines if needed for a data category at any point in the data journey. For instance, research has already identified the need for pipelines to perform automated classification for the clothing data category, or for identifying potential relaunched products in the grocery data category.