Building the new Banff

An open-source data editing system based on GSDEM concepts

Darren Gray
Senior Methodologist, Statistics Canada

UNECE Expert Meeting on Statistical Data Editing
7-9 October 2024, Vienna

Statistics Canada Statistique Canada

Canada

Banff 3.0

- Release: January 2025
  - Internal release -> user testing -> external release
- Set of Python packages
  - Open-source (including C code)
  - No SAS dependencies
  - Shared via GitLab (internally) and GitHub (externally)
- Includes an overhauled processor
  - Redesigned to enable complex process flows
  - Based on concepts & terminology from GSDEM

Today's presentation

- Overview
- Generic Statistical Data Editing Model (GSDEM)
- The Banff procedures – what makes them special?
- The new Banff processor
- Building a catalogue of Banff-compatible modules
- Remarks

Key questions

Could you / would you use the Banff processor as a data editing production platform?

Within the data editing community, is there interest in building and maintain a catalogue of modular, Banff-compatible data editing tools?

A history of Banff

- Generalized edit and imputation system developed and maintained by Statistics Canada
- Features nine data editing procedures performing various data editing tasks, including outlier detection, error localization, and donor imputation
- Includes Banff Processor: metadata-driven process flow manager
- Current version runs on SAS architecture

A history of Banff

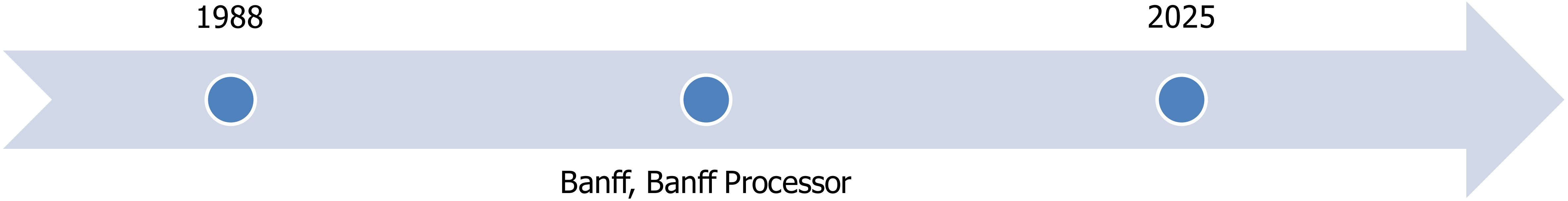
GEIS

(Generalized Edit and Imputation System)

1988

Banff 3.0

2025



Banff, Banff Processor

2002

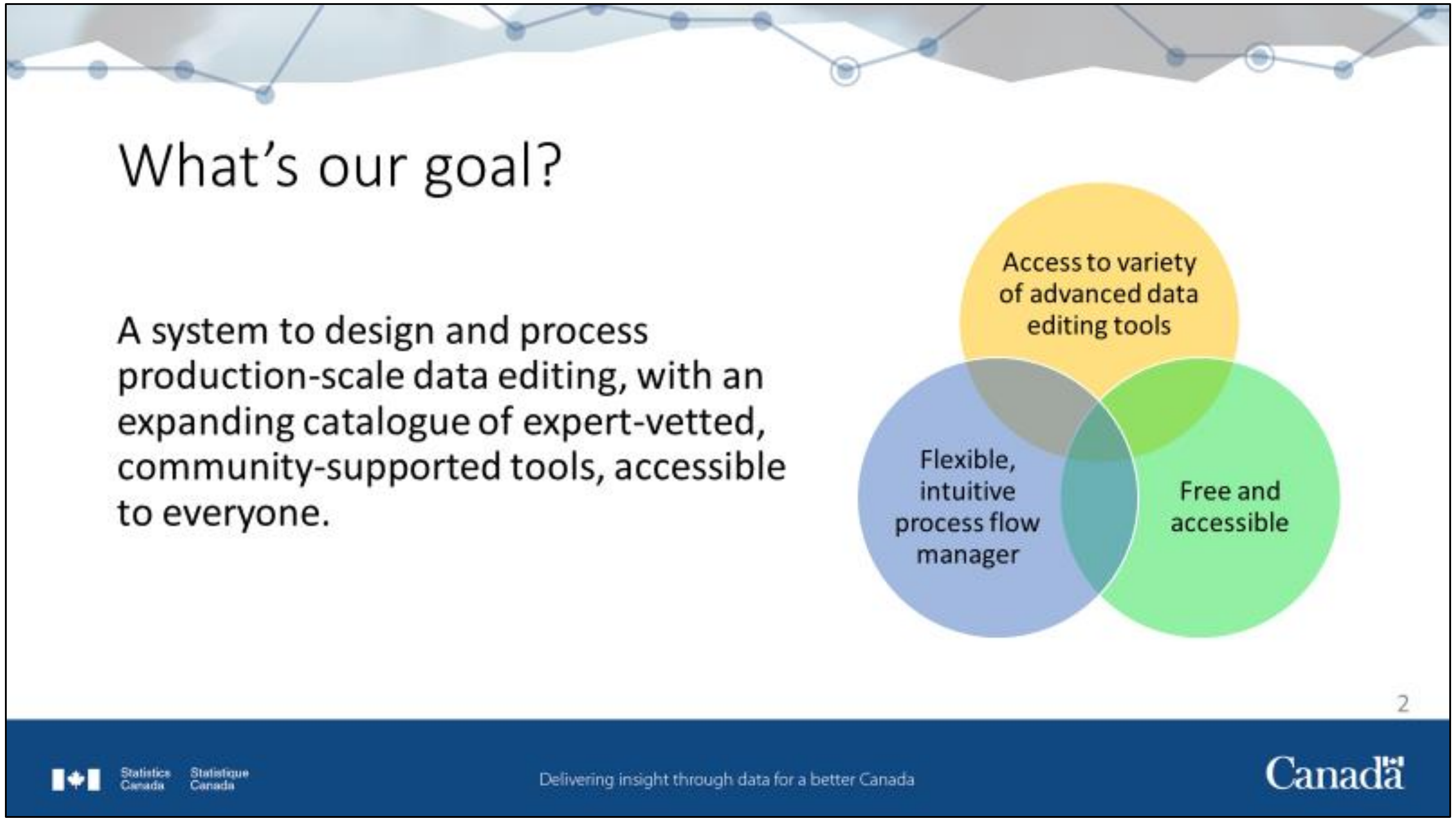
Users:

- Statistics Canada
- Canadian government agencies
- Other National Statistical Offices
- Private institutions

Plans for change

- Future development of Statistics Canada's Edit and Imputation System Banff (Thomas, 2017)
- The evolution of Banff in the context of modernization (Gray, 2018)
  - Standardizing the Banff procedures
  - Improving the Banff processor
- Banff’s next step: an open-source data editing system for advanced tools and collaboration (Gray, 2022)
  - Move to open-source

Banff's next step: an open-source data editing system for advanced tools and collaboration (Gray, 2022)
Expert Meeting on Statistical Data Editing | UNECE



What's our goal?

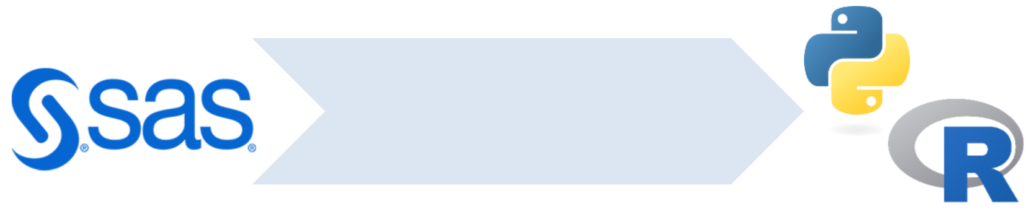
A system to design and process production-scale data editing, with an expanding catalogue of expert-vetted, community-supported tools, accessible to everyone.

Access to variety of advanced data editing tools
Flexible, intuitive process flow manager
Free and accessible
2
Statistics Canada Statistique Canada
Delivering insight through data for a better Canada
Canada

Changing environments

- Proliferation of free and/or open-source data editing tools:
  - Python and R packages
  - Awesome official statistics software
- Need / desire to modernize
  - New problems to solve
  - Modern tools, e.g., ML

Spring 2023: Statistics Canada makes decision to significantly reduce SAS footprint by 2028



Changes for Banff 3.0

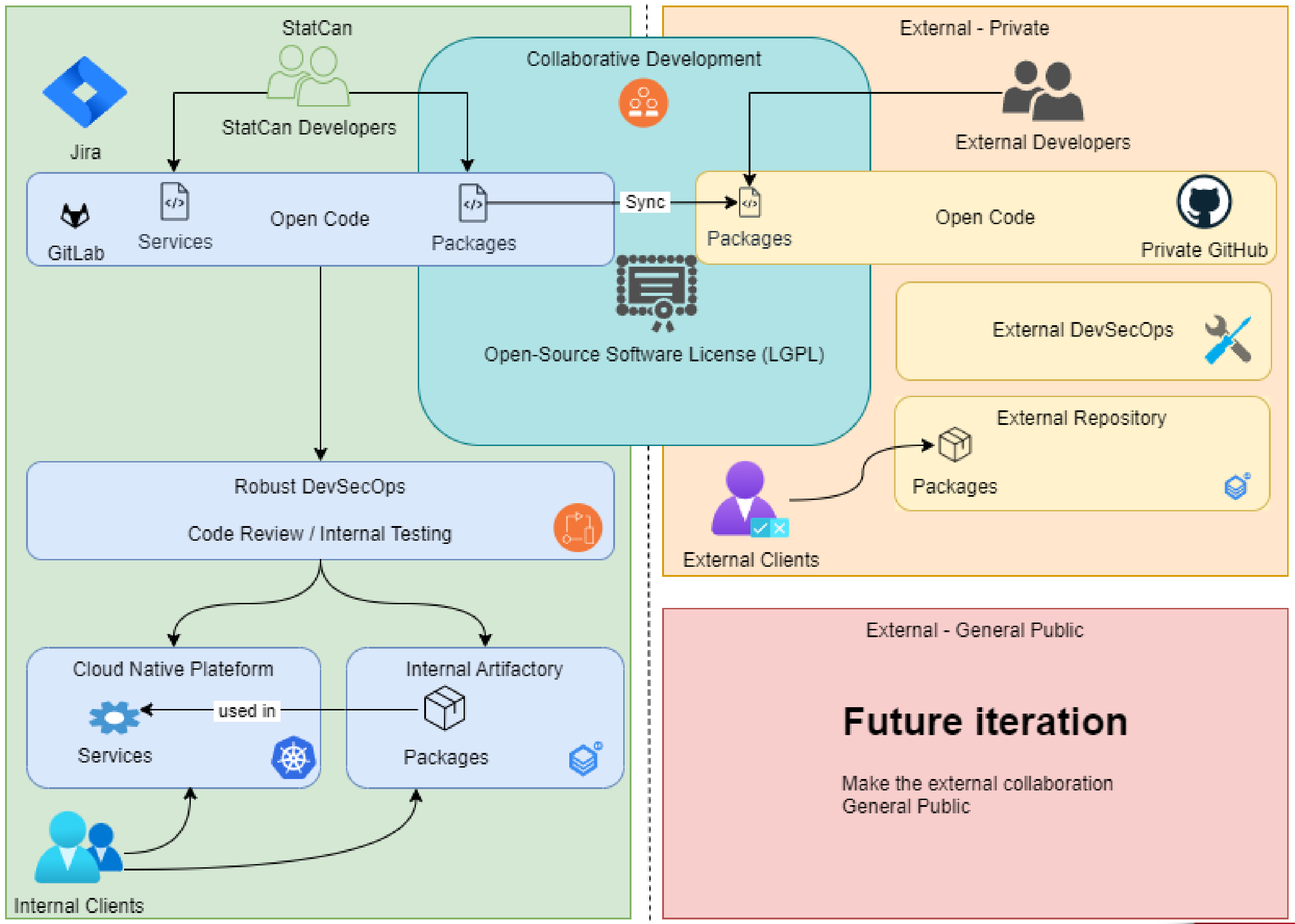
- Completely free and open-source
  - Primary languages are Python, C
  - Some Python package dependencies for data management
- Banff procedures (ErrorLoc, DonorImp, Outlier, etc.)
  - Python modules (instead of SAS procedures)
  - Underlying statistical functions remain unchanged
  - Minor updates –> mostly standardization
- Banff processor
  - Completely redesigned, with new features
- Dissemination and support via GitLab & GitHub

Generalized solutions – open-source distribution

Key messages

- Work in the open by default
- Enable external collaboration
- Adopt open-source licensing
- Private to public transition *when ready*
- Leverage robust internal tools (GitLab, Artifactory, Xray)
- Public sharing on industry platforms (GitHub, PyPI, CRAN)
- Maintain control (accept/reject external collaboration, issues)
- Pre-release internally first to ensure code quality

Distributing Generalized Solutions to internal and external clients - Conceptual Diagram

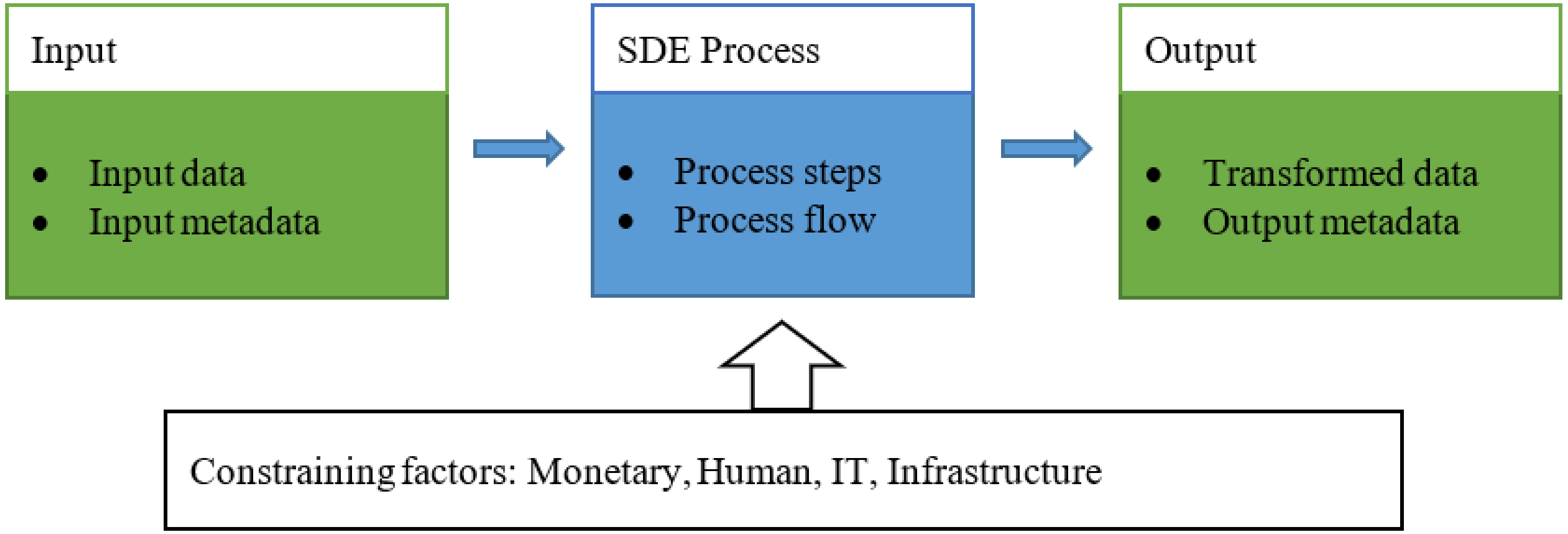


Generic Statistical Data Editing Model

- Developed under the High-Level Group for the Modernisation of Official Statistics (HLG-MOS)
- The GSDEM is envisaged as a "standard reference for statistical data editing" that provides "standard terminology and models" and facilitates "understanding, communication, practice and development in the field of statistical data editing".
- Topic covered in GSDEM:
  - Functions and methods
  - Metadata for the data editing process
  - SDE flow models

[GSDEM - Statistical Data Editing - UNECE Statswiki]

SDE Process


Input
Input data
Input metadata
SDE Process
Process steps
Process flow
Output
Transformed data
Output metadata
Constraining factors: Monetary, Human, IT, Infrastructure

SDE Process Flows

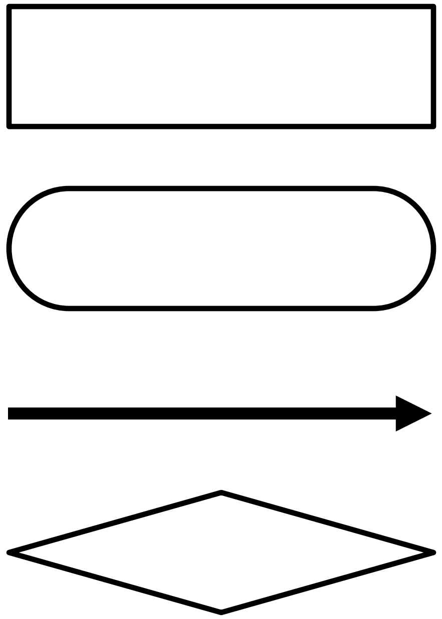
What elements are required to describe a specific SDE process?

- Overall process broken down into a limited number of **process steps**, a set of specific functions executed in an organized way for a specific SDE purpose.
- Navigation between process steps is managed by rule-based **process controls**.
- The **process flow** describes the sequencing and conditional logic among different process steps.

Process steps, functions and methods

- **Process steps** typically contains a considerable number of functions with specified methods that are executed in an organized way.
- SDE function categories:
  - **Review**: Functions that examine the data to identify potential problems
  - **Selection**: Functions that select units or fields within units for specified further treatment
  - **Treatment**: Functions that change the data in a way that is considered appropriate to improve data quality.
- **Methods** describe how the functions are performed.

SDE Process Flows

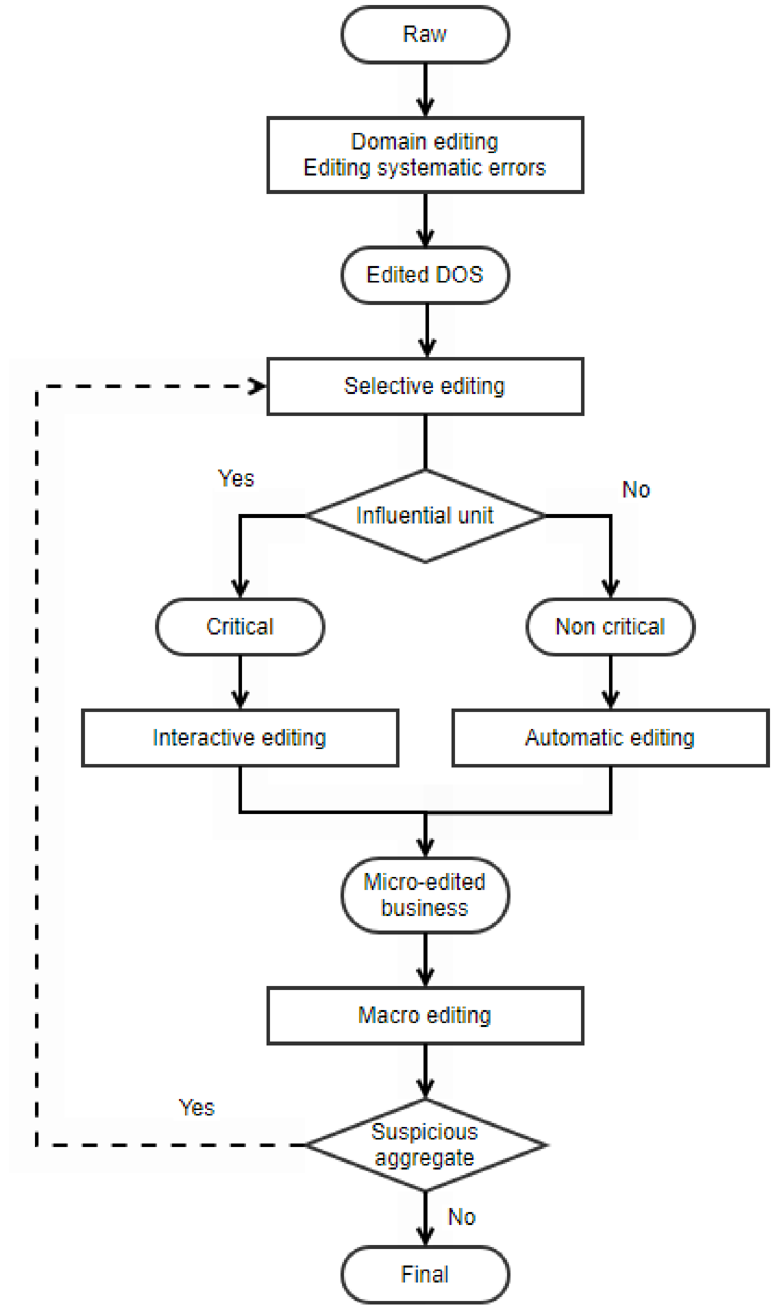


Process step

Data state

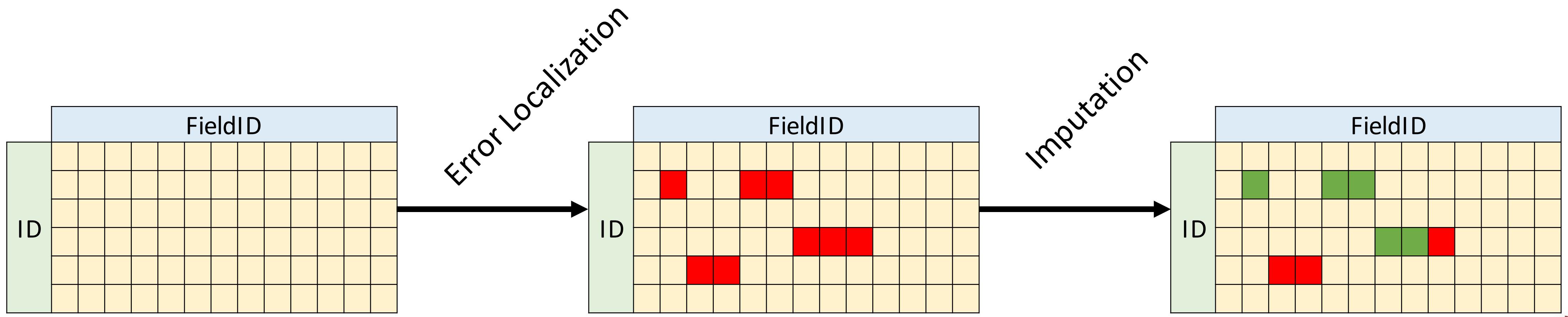
Trivial process control

Non-trivial process control



Banff's purpose

- GEIS / Banff was developed to process business data with specific needs such as:
  - Scale issues and distribution of business data (e.g., revenue, employment)
  - Linear relationships between variables
- Banff status flags play a key role in the system, as outputs from one procedure serve as inputs for subsequent ones



Design, review and selection functions

Procedure	Functions performed	Compatible with linear edits
VerifyEdits (design)	Evaluates a set of edits for consistency and redundancy; can optionally generate extremal points and implied edits.	Yes
EditStats (review)	Determines the number of records within a dataset that pass, miss, or fail each edit; provides summary statistics only.	Yes
ErrorLoc (selection)	For records that fail to pass a set of edits, identifies the minimum number of variables that must be amended to pass all edits, following the Fellegi-Holt error localization paradigm.	Yes
Outlier (selection)	Outlier detection using the Hidiroglou-Berthelot or Sigma-Gap methods.	No

Treatment functions

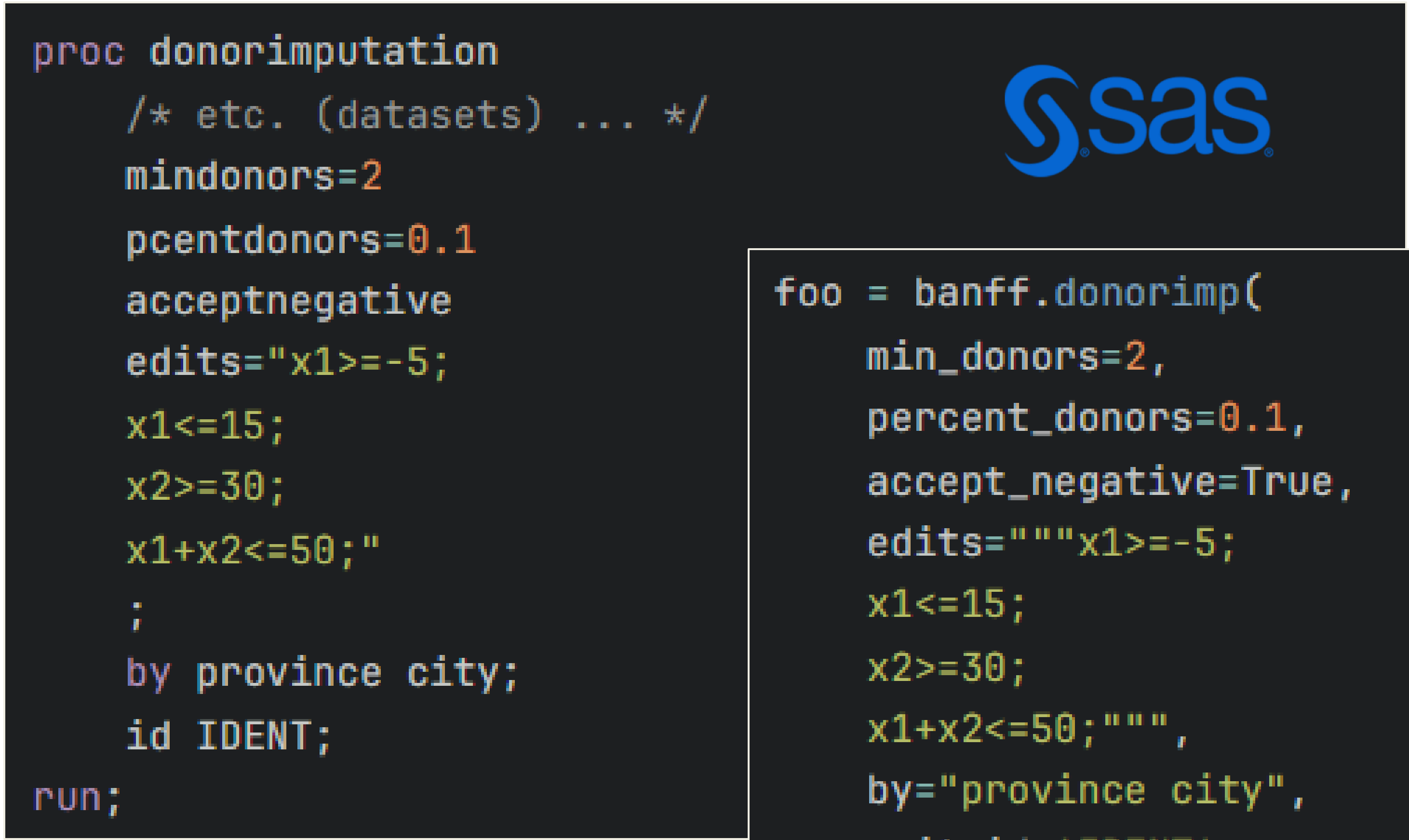
Procedure	Functions performed	Compatible with linear edits
Deterministic	Deterministic imputation for records where only one value (or vector of values) will pass all edits.	Yes
DonorImputation	Nearest-neighbour donor imputation ensuring that all amended records pass specified edits.	Yes
MassImputation	Donor imputation without edit constraints; generally used for blocks of missing data.	No
Prorate	Prorates record values to meet specified edit constraints.	Yes
Estimator	Estimator (or model-based) imputation; users can choose from a number of pre-defined algorithms or create their own; may reference auxiliary and historical data.	No

Are the Banff procedures still needed?

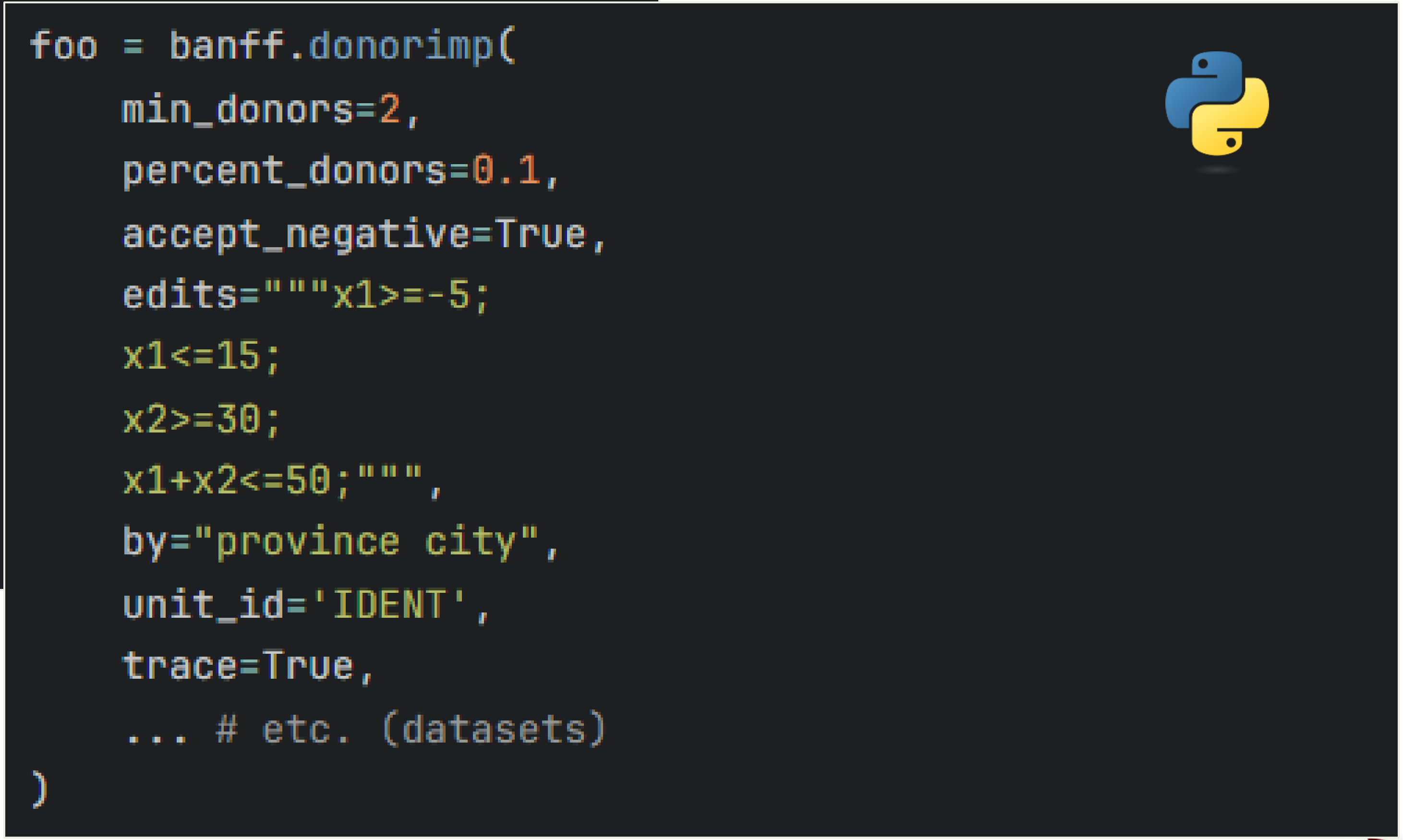
- Some still provide unique functionality, e.g., donor imputation:
  - Unique distance metric to account for scale issues within business data
  - Automated selection of auxiliary variables for distance calculation
  - Efficient search algorithm to find eligible donors to satisfy linear constraints
- For many, similar functionality (sometimes using different methods) can already be found elsewhere:
  - ErrorLoc: errorlocate (R package)
  - Outlier: univOutl (R package)
  - Estimator: simputation (R package)

Conversion plan

- Convert all nine procedures to maintain consistency
  - Minimize changes to C code
  - Swap SAS wrapper for Python wrapper
- Standardize the procedures
  - Consistent inputs, outputs and parameters across the functions
  - Use of input and output status flags
- A few minor improvements

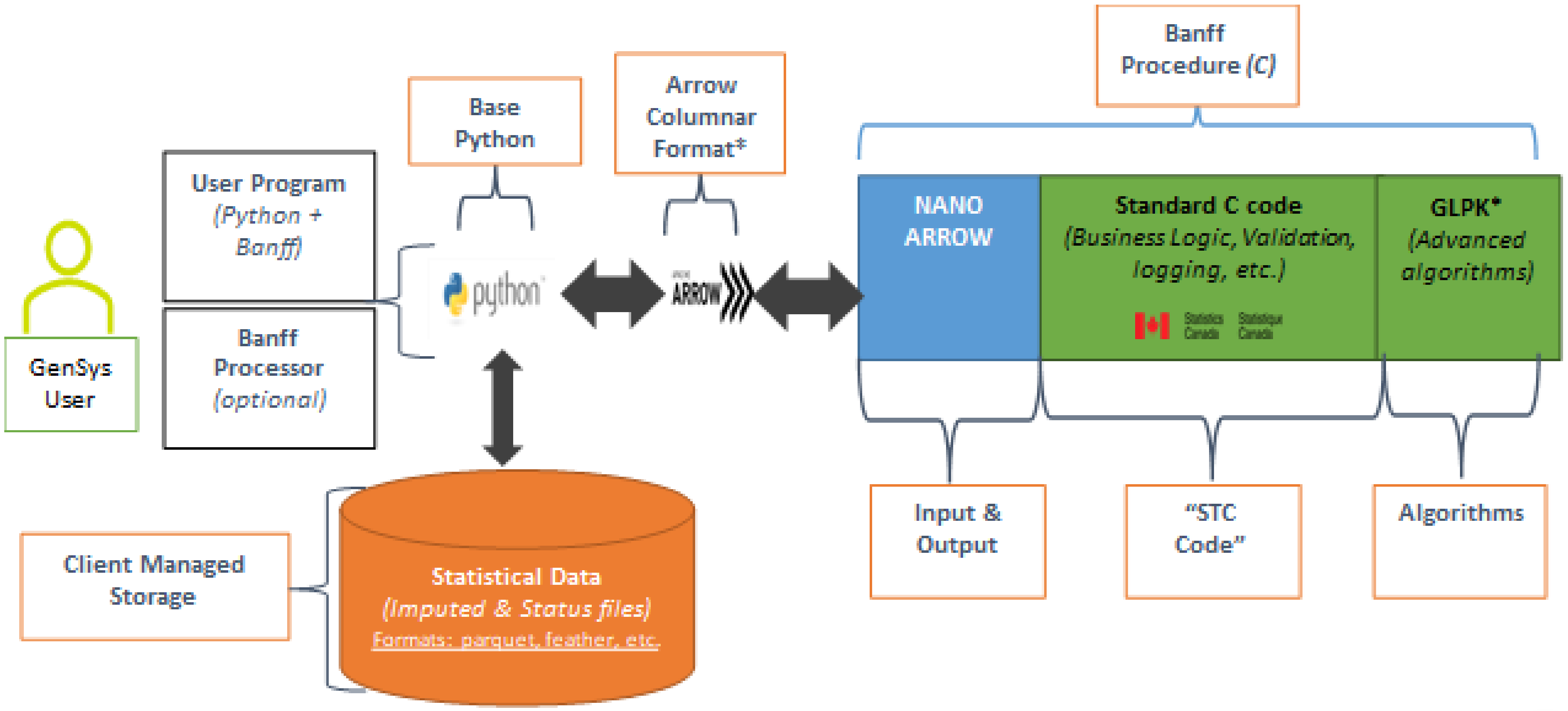


```
proc donorimputation
    /* etc. (datasets) ... */
    mindonors=2
    pcentdonors=0.1
    acceptnegative
    edits="x1>=-5;
    x1<=15;
    x2>=30;
    x1+x2<=50;"
    ;
    by province city;
    id IDENT;
run;
```



```
foo = banff.donorimp(
    min_donors=2,
    percent_donors=0.1,
    accept_negative=True,
    edits="""x1>=-5;
    x1<=15;
    x2>=30;
    x1+x2<=50;""",
    by="province city",
    unit_id='IDENT',
    trace=True,
    ... # etc. (datasets)
)
```

Banff procedure – data flow diagram


GenSys User
User Program (Python + Banff)
Banff Processor (optional)
Base Python
python
Arrow Columnar Format*
ARROW
Banff Procedure (C)
NANO ARROW
Standard C code (Business Logic, Validation, logging, etc.)
Statistics Canada Statistique Canada
GLPK* (Advanced algorithms)
Input & Output
"STC Code"
Algorithms
Client Managed Storage
Statistical Data (Imputed & Status files)
Formats: parquet, feather, etc.

Objectives

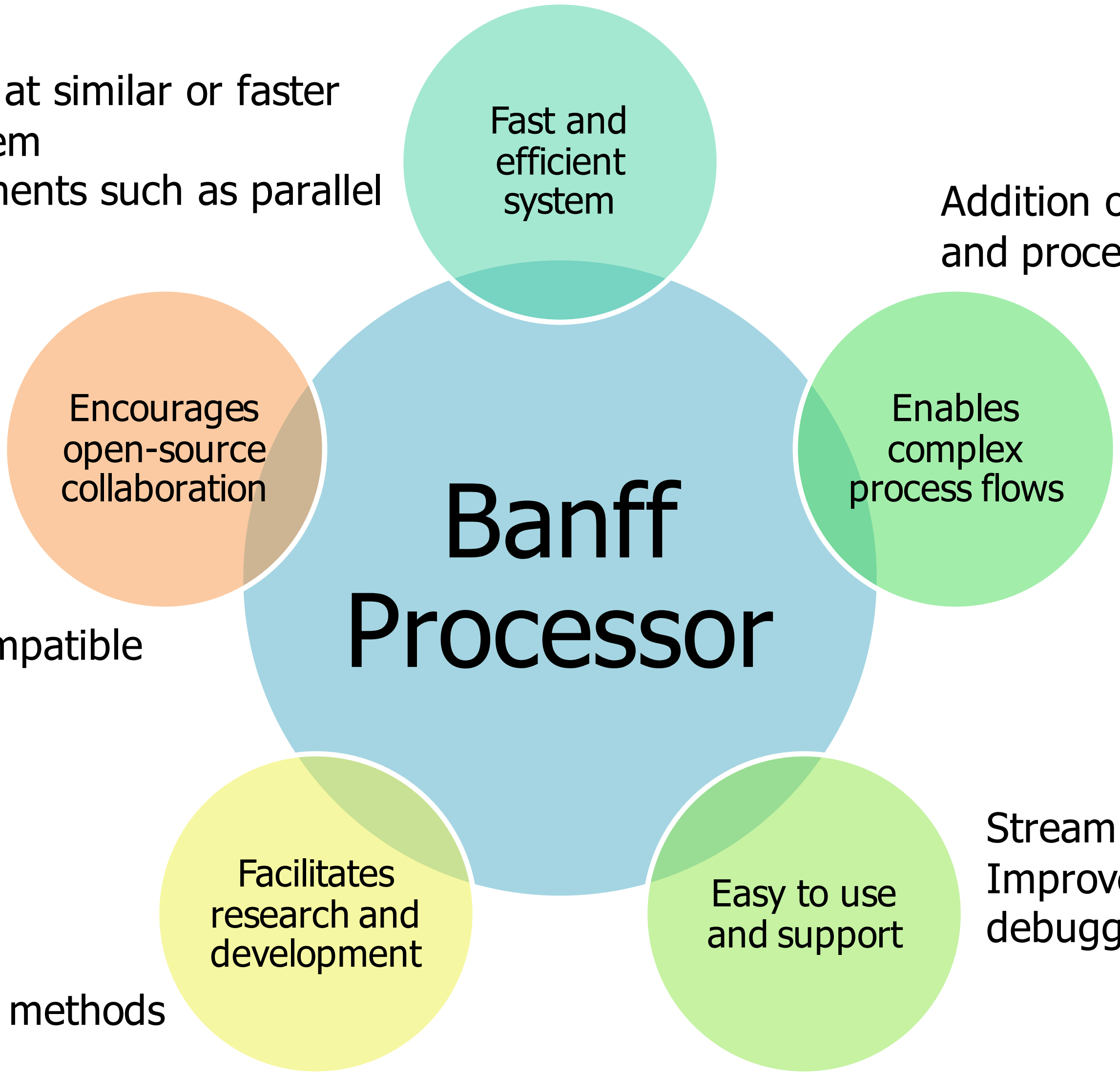
Process production tasks at similar or faster speeds than current system
Open to future enhancements such as parallel processing

Addition of process blocks and process controls

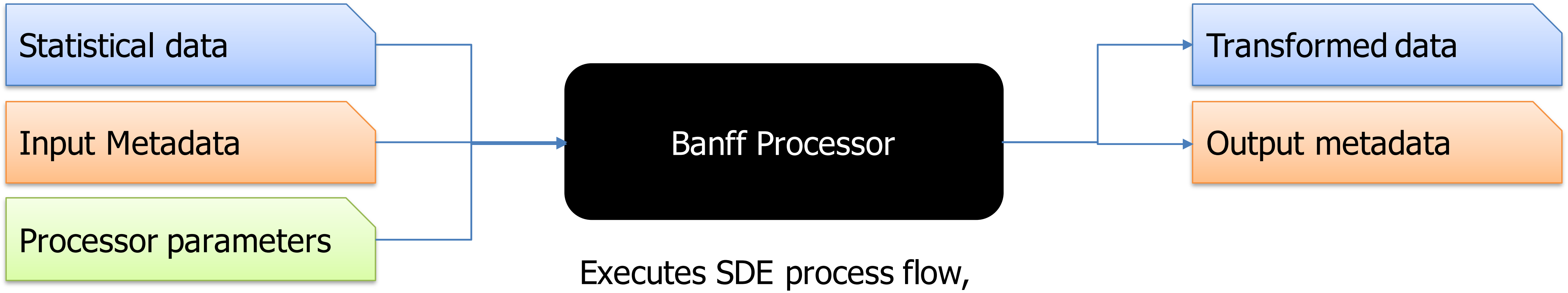
General application
A central repository for compatible modules

Streamlined design
Improved outputs and debugging options

Easy to test out new E&I methods


Fast and efficient system
Encourages open-source collaboration
Banff Processor
Enables complex process flows
Facilitates research and development
Easy to use and support

Inputs / Outputs



Executes SDE process flow, handles data management

Statistical data inputs

Statistical data

InData (mandatory)	Primary target of the SDE process
HistData (optional)	Historical dataset (used by some Banff procedures)
Other	Other statistical data required by plugins

Supported inputs formats: Apache Parquet (preferred), Apache Feather, CSV (not recommended), SAS Dataset (testing only)

Supported outputs: Parquet, CSV (for testing)

Metadata inputs

Input metadata

Processor metadata	Specifications for the SDE process flow and all process steps (XML files)
InStatus (optional)	Banff status flags from a previous process
Other	Other metadata required by plugins

Processor parameters

Processor parameters

- job_ID: Process identifier
- unit_ID: Unique record identifier on input data
- Location of processor input XML metadata
- Location of custom plugins
- Other parameters such as random seed, output file format, etc…

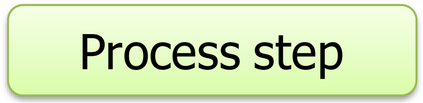
Processor Metadata

- Describes the complete SDE process flow, including parameters for all process steps
- Processor metadata can be categorized as follows:
  - Describing the overall process flow
  - Parameters specific to Banff built-in procedures
  - Parameters for custom procedures
  - Additional metadata common to multiple procedures
- Banff processors reads metadata from XML files
  - Users can design and edit processor metadata using Excel template, and then convert to XML
  - Point-and-click interface planned for future

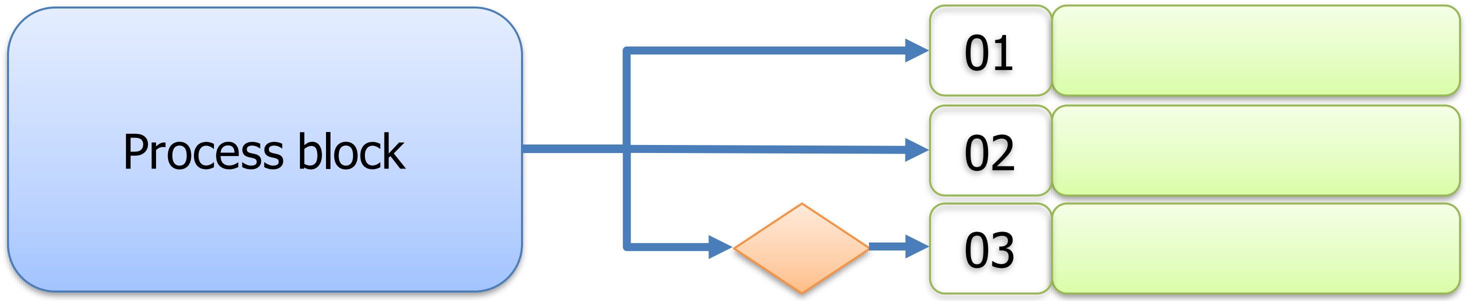
Outputs

- Key outputs:
  - Transformed statistical data
  - Final status flags
  - Full list of status flags
- Diagnostics:
  - High-level: Run-time for each process step
  - Detailed log of the complete process
- Other
  - Additional outputs generated by Banff procedures or custom plugins

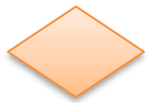
Elements of the Banff process flow



Executes either a built-in Banff process or a custom, user-defined process. Typically performs SDE functions such as review, selection, or treatment.



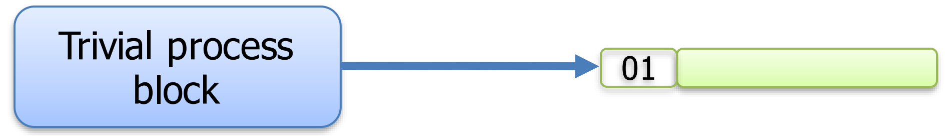
Consists of one or more ordered sub-processes, which can be process steps or other process blocks.



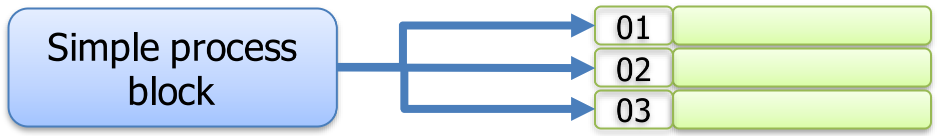
Process control

Associated with specific sub-processes in a process block, they apply logical conditions to determine when a sub-process runs, and for which data.

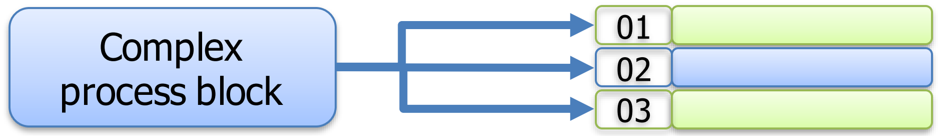
Process Blocks



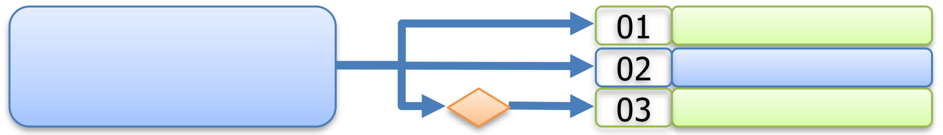
Consists of a **single process step**



Calls one or more **process steps**



Calls at least one other **process block**



Process controls can be assigned to any sub-process

Process Flow Metadata

Users can specify and edit process blocks using the Banff Processor template in Microsoft Excel, which is converted to xml files for the Processor to read.

The JOBS table includes columns to specify the complete process flow.

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

Process Flow Metadata

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

Process blocks are defined by a single **JobID** (e.g., “Main”). Each row in a process block calls a sub-process.

Ordering of the sub-processes is specified by the **SeqNo** column.

Another process block, “OutlierImpute”, with two sub-processes

Process Flow Metadata

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

The **ControlID** column indicates a **process control** is associated with the current sub-process.

The parameters for process controls are found in the ProcessControls metadata table and linked by the ControlID.

Process Flow Metadata

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

The **Process** column specifies which type of process to run. Options are:

- Job: Calls another process block
- One of the nine built-in Banff processes
- A user-defined plugin

Statistics Canada Statistique Canada

Canada

Process Flow Metadata

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

The **SpecID** column tells the Banff processor how to run the given process.

- When Process = JOB, SpecID must refer to another Process Block by the JobID
- For all other processes (built-in or plugins), SpecID indicates which parameters to apply, from another metadata table

Note: Some processes may not require specifications.

Process Flow Metadata

jobid	seqno	controlid	process	specid
Main	1		**ErrorLoc**	**ErrorLoc_specs**
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	**JOB**	**Rounding**
Main	7		Plugin_3	Custom_specs_3
Main	8		**ErrorLoc**	**ErrorLoc_specs**
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	**JOB**	**Rounding**
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

This table consists of four process blocks:

- Two complex process blocks
- Two simple process blocks

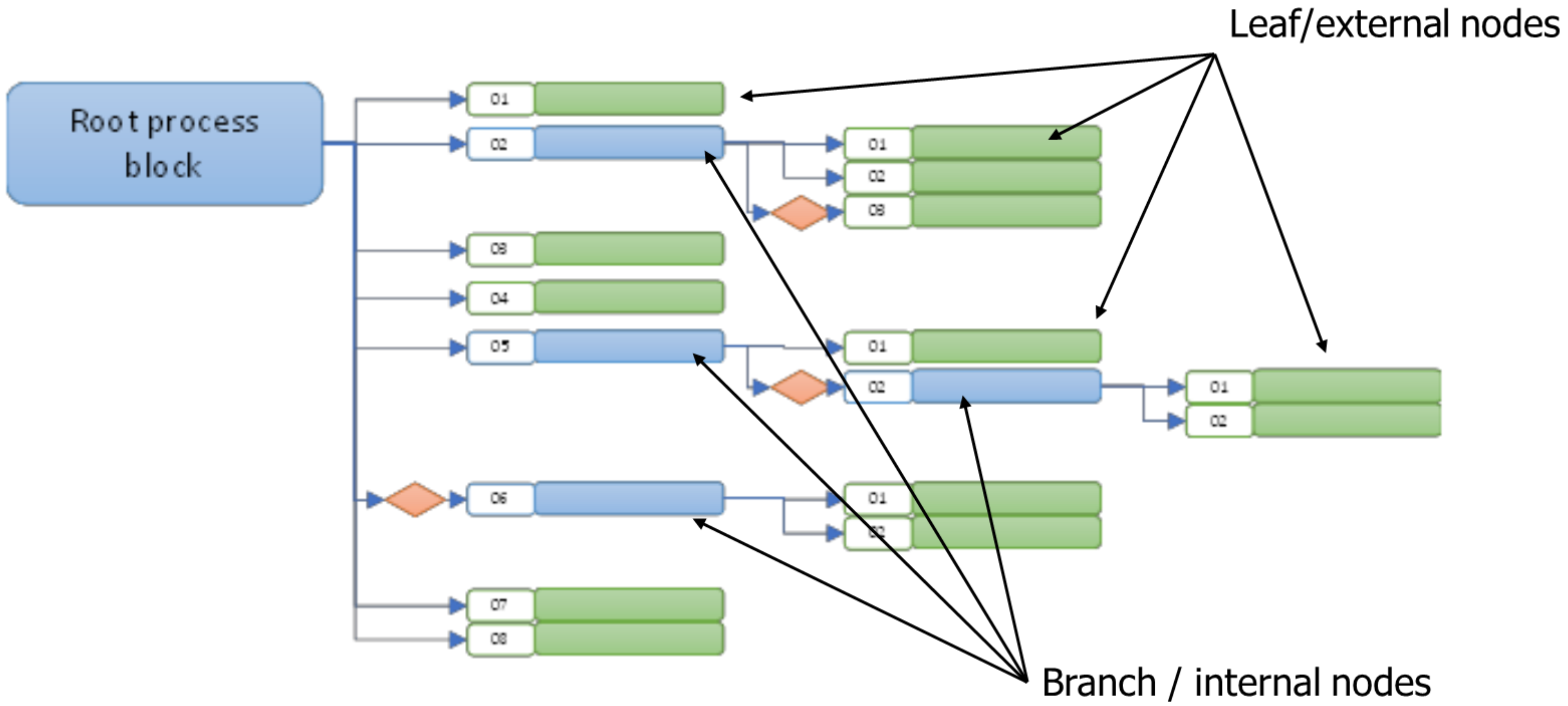
Note that the Main process block calls other process steps and process blocks more than once:

- ErrorLoc is called in steps 1 and 8
- The Rounding process block is called in step 6, and again in step 2 of OutlierImpute

Visualizing the Banff process flow

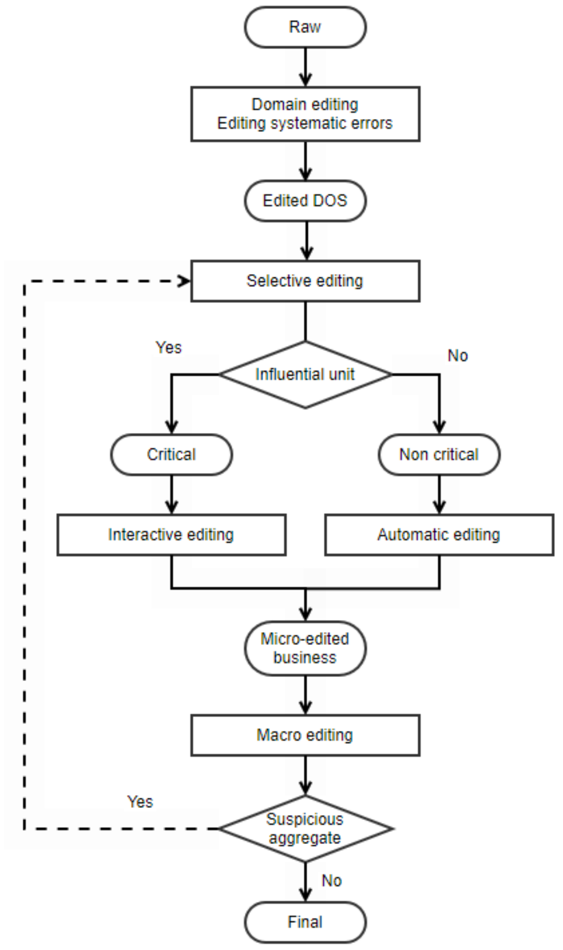
- When the Banff processor is called, the resulting process flow can be visualized as a "tree" data structure.
  - Process steps and process blocks are nodes in the tree, connected by links
  - Each process block consists of a "parent node with ordered children"
  - Some nodes have specific terms in the data structure
    - Root node: Process Block called by the user
    - Internal node (inner node / branch node): Process blocks
    - External node / Leaf: Node without any children -> Process step

Visualizing the Banff process flow



Process Blocks

- Process blocks allow users to organize the process flow in a logical way, improving readability and convenience, but do not change the resulting process flow
- When combined with **process controls**, allows users to specify complex process flows



Jobs table

jobid	seqno	controlid	process	specid
Main	1		ErrorLoc	ErrorLoc_specs
Main	2		JOB	EditImpute
Main	3		Outlier	Outlier1
Main	4		Outlier	Outlier2
Main	5		JOB	OutlierImpute
Main	6	Filter1	JOB	Rounding
Main	7		Plugin_3	Custom_specs_3
Main	8		ErrorLoc	ErrorLoc_specs
EditImpute	1		Deterministic	
EditImpute	2		DonorImp	Donor_specs_1
EditImpute	3	Filter2	Plugin_2	
OutlierImpute	1		Estimator	Estimator_specs
OutlierImpute	2	Filter3	JOB	Rounding
Rounding	1		Prorate	Prorate_specs
Rounding	2		Plugin_1	Custom_specs_1

Process controls that show up in the Jobs metadata table are specified in the ProcessControls table, linked by the ControlID.

ProcessControls table

controlid	targetfile	parameter	value
Filter1	Indata	row_filter	Province = "Ontario"
Filter2	Indata	column_filter	UnitID, Revenue, Expenses
Filter3	Indata	row_filter	Revenue ge 1,000,000
Filter3	Instatus	row_filter	Process = Outlier and Status = "FTI"

Process Filters

controlid	targetfile	parameter	value
Filter1	Indata	row_filter	Province = "Ontario"
Filter2	Indata	column_filter	UnitID, Revenue, Expenses
Filter3	Indata	row_filter	Revenue ge 1,000,000
Filter3	Instatus	row_filter	Process = Outlier and Status = "FTI"

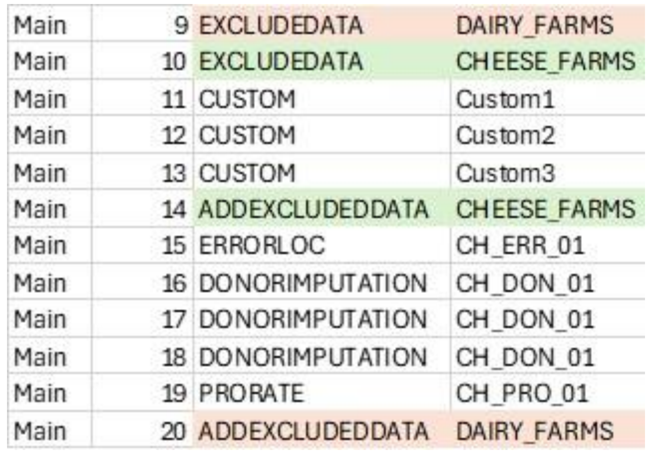
- ControlID: Process control identifier
- TargetFile: Input file to filter
- Parameter
  - row_filter: Filters the target file using an SQL WHERE clause from *value* field
  - column_filter: Filters target file to remove columns that don't appear in value field

Process Filters

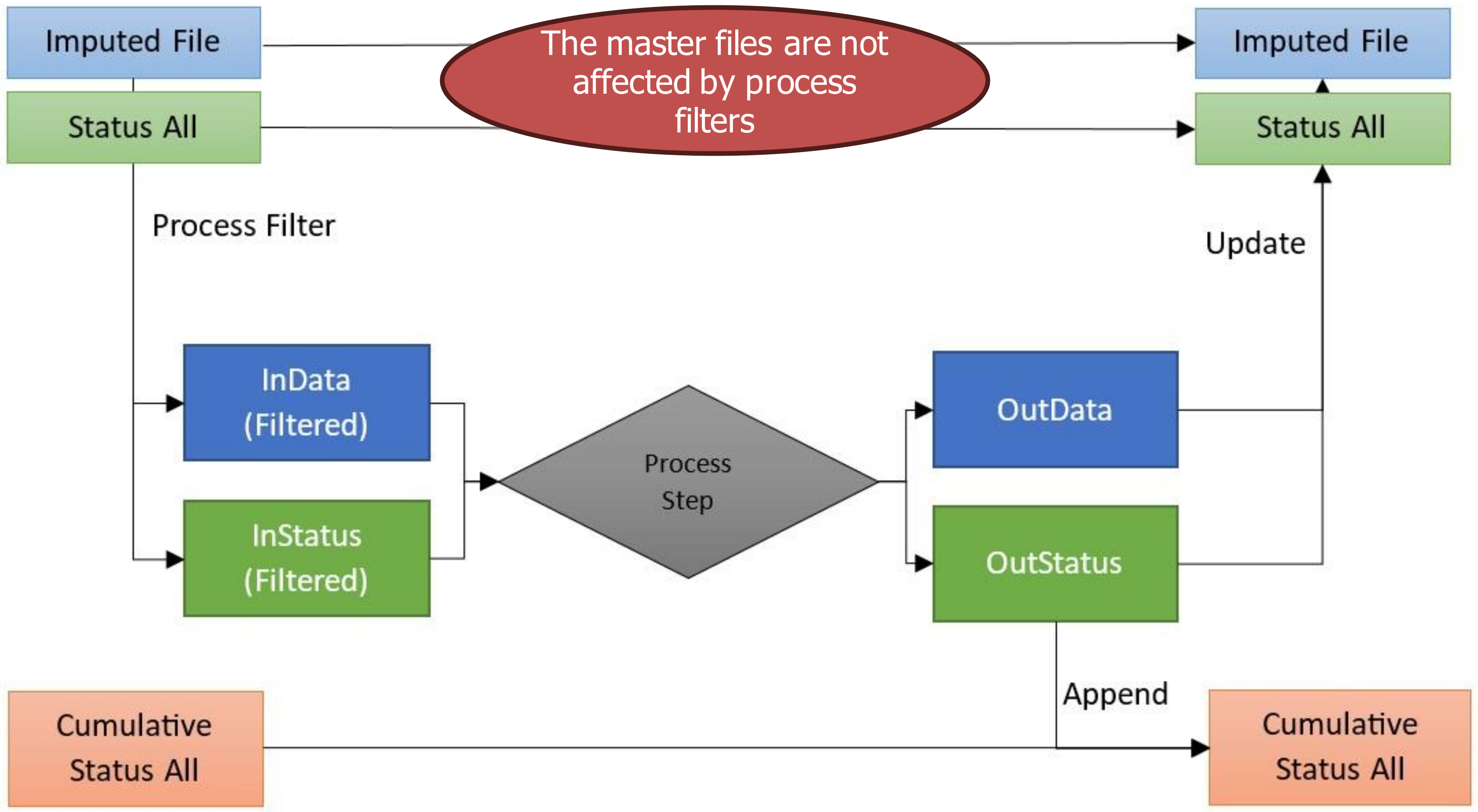
- Process filters of ***indata*** can be used to control which data flows through process blocks or process steps. Examples:
  - Process blocks to treat subsets of population, such as specific geographic regions or industries
  - Process blocks to treat subsets of variables
- Process filters of ***instatus*** can be used to limit the input selection flags. Examples:
  - A process block that only treats outliers
  - A process block that only treats records failing specific edits
- Multiple filters can be applied to the same process control
- Expressions use duckdb SQLite syntax.

Process Filters

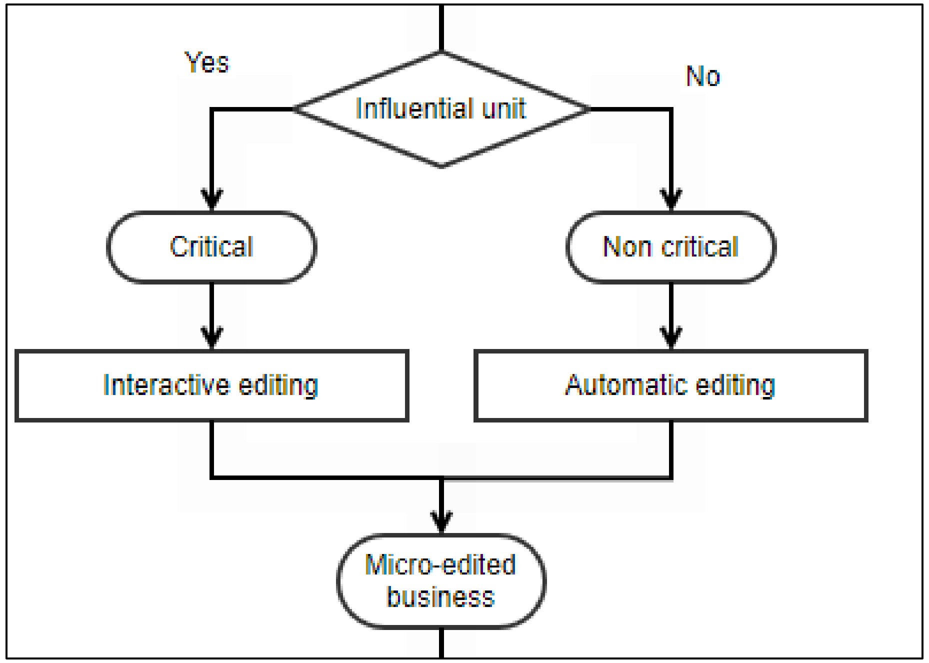
- Process filters offer a convenient and safe solution to common data management needs
  - In the current processor, users manipulate the statistical data directly using custom programs "ExcludeData" and "AddExcludedData"
  - Easy to see which steps in the process are affected
  - Removes risk of human error

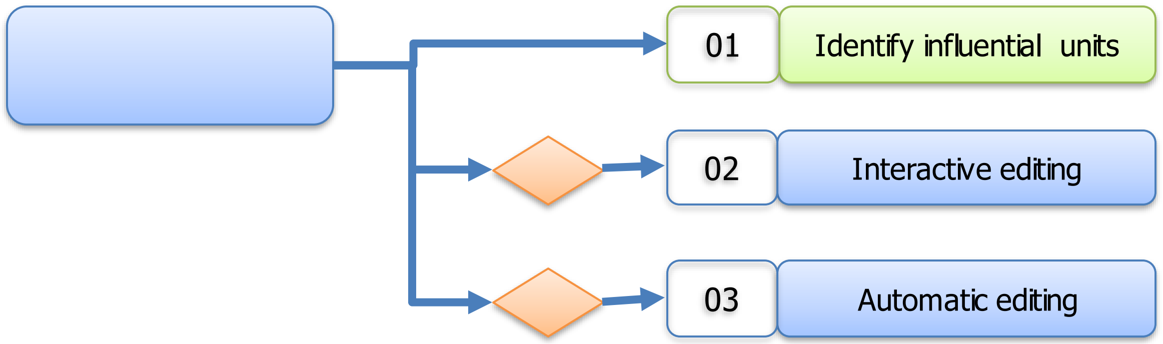


Main	9	EXCLUDEDATA	DAIRY_FARMS
Main	10	EXCLUDEDATA	CHEESE_FARMS
Main	11	CUSTOM	Custom1
Main	12	CUSTOM	Custom2
Main	13	CUSTOM	Custom3
Main	14	ADDEXCLUDEDDATA	CHEESE_FARMS
Main	15	ERRORLOC	CH_ERR_01
Main	16	DONORIMPUTATION	CH_DON_01
Main	17	DONORIMPUTATION	CH_DON_01
Main	18	DONORIMPUTATION	CH_DON_01
Main	19	PRORATE	CH_PRO_01
Main	20	ADDEXCLUDEDDATA	DAIRY_FARMS


Imputed File
Status All
The master files are not affected by process filters
Imputed File
Status All
Process Filter
Update
InData (Filtered)
InStatus (Filtered)
Process Step
OutData
OutStatus
Append
Cumulative Status All
Cumulative Status All

Example: treatment of influential units



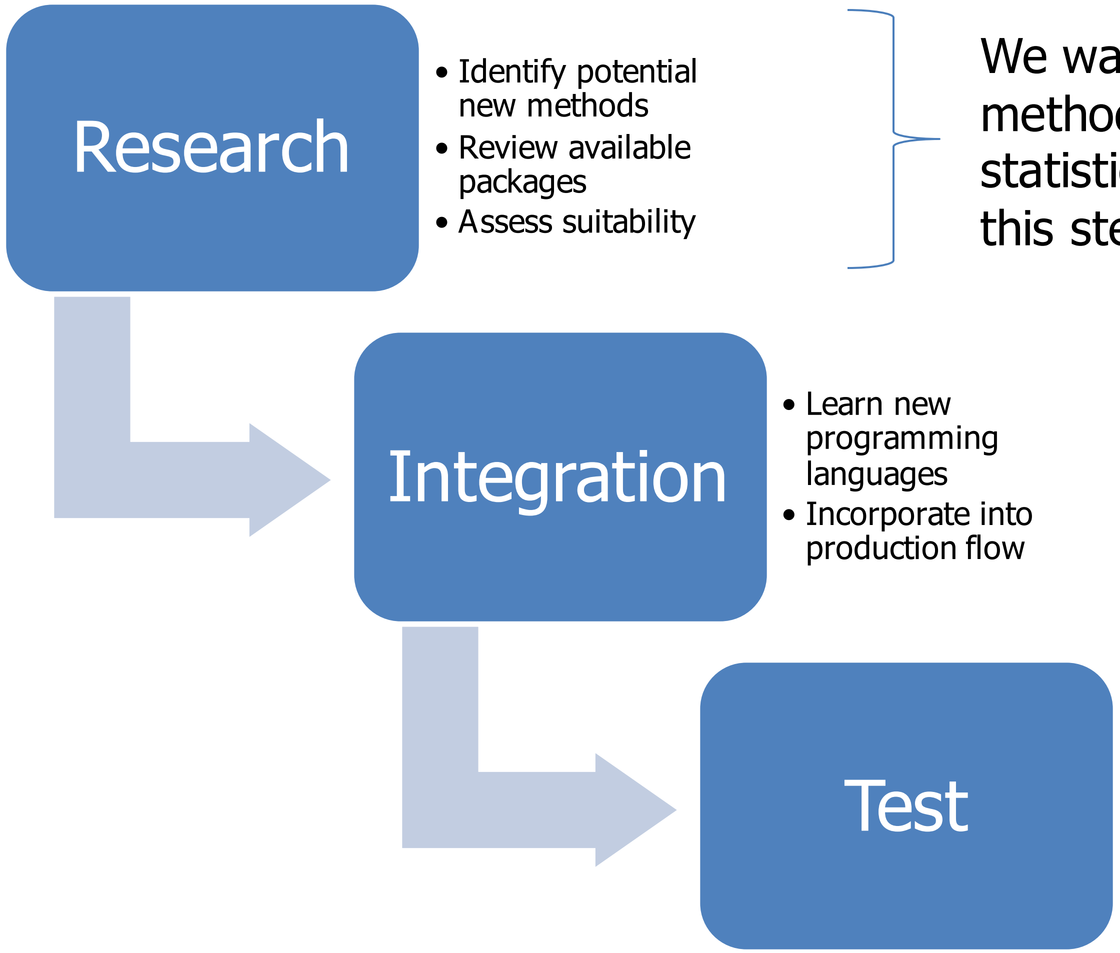


By applying consecutive process filters with converse conditions (influential unit = yes / no), we can create most of the logical conditions we require

Barriers to the adoption of new technology

- Resources: time, personnel, money
- Statistics Canada experience
  - Between production schedule, not a lot of time to test new methods
  - Data editing frequently performed by junior methodologists who transfer frequently between projects
  - Production systems are somewhat rigid
  - Transition from SAS to new programming languages (e.g., R, Python) is already overwhelming

Barriers to the adoption of new technology



Research

- Identify potential new methods
- Review available packages
- Assess suitability

We want methodologists and statisticians focused on this step

Integration

- Learn new programming languages
- Incorporate into production flow

Test

- Design and execute suitable tests
- Share results

Use Banff Processor to standardize and facilitate these steps

Building a catalogue of Banff modules

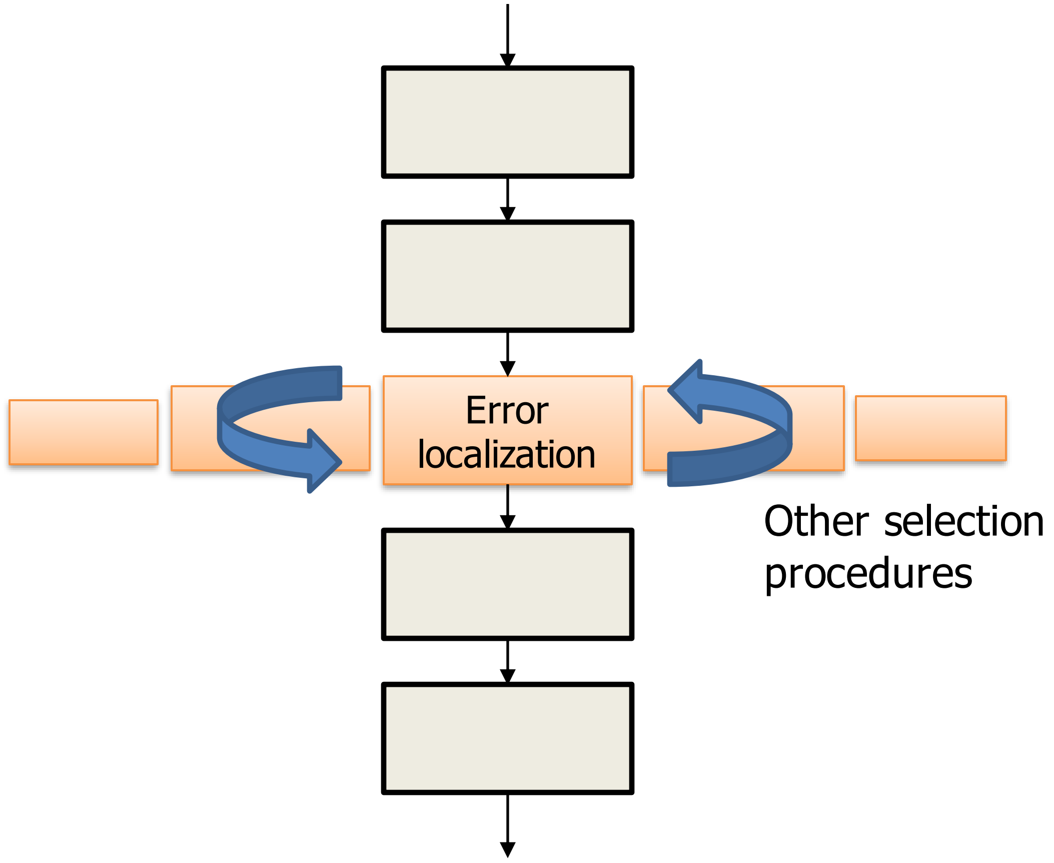
- Objective: build a catalogue of Banff-compatible modules (plugins) that are simple to integrate into Banff processor
  - Open-source, community-driven
  - Documented
  - Vetting and testing system
- Give methodologists more time to focus on important work:
  - Test new methods
  - Design effective SDE process flows

Modularity

Within the Banff processor, modules performing the same SDE functions should be interchangeable

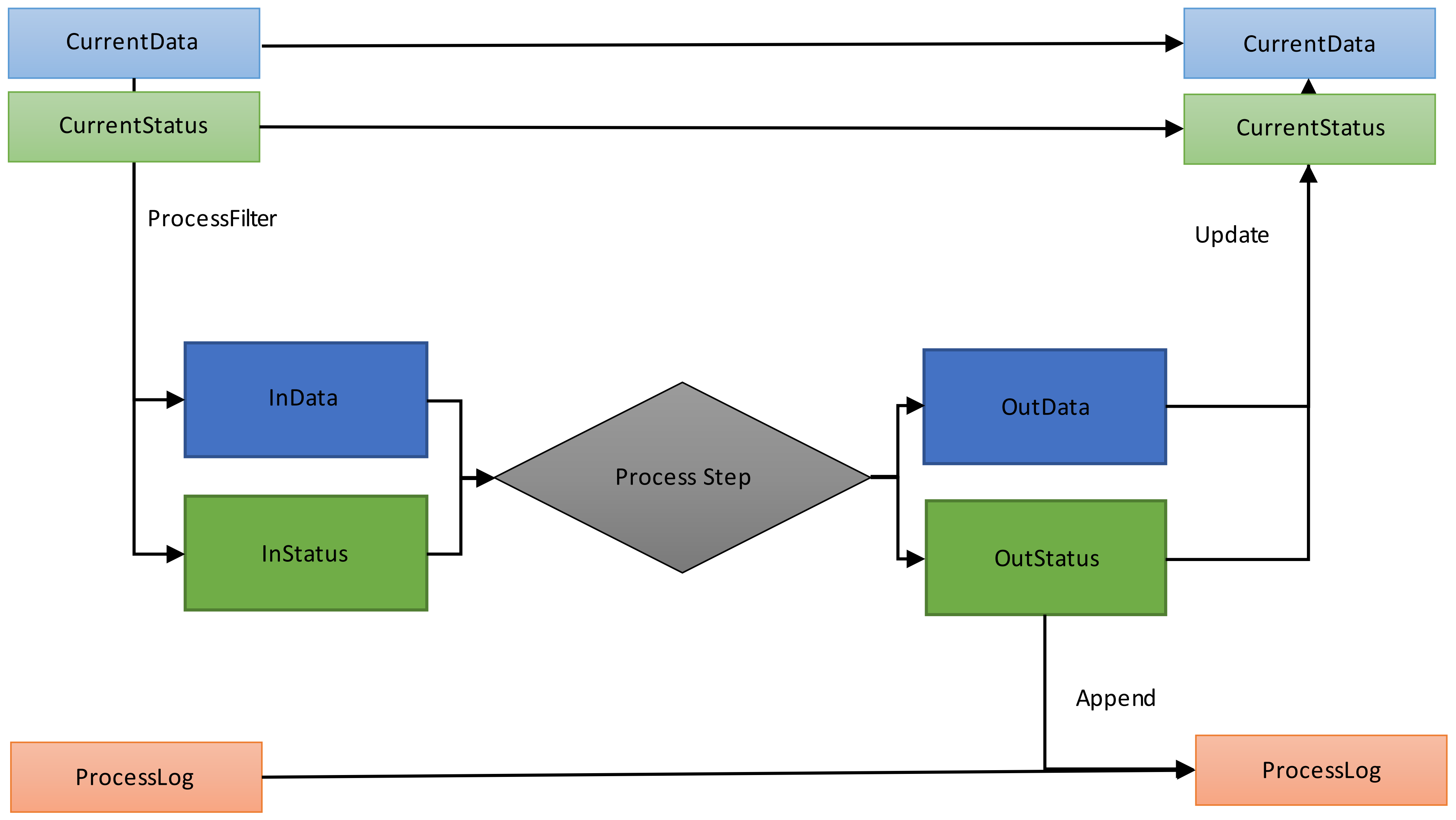
This makes it easy to test new functions, and assess their impact on the overall process flow

Statistical data editing process flow



Plugin structure

- General structure:
  1. Declare dependencies
  2. Import data and parameters
  3. Execute plugin code
  4. Save outputs
- Banff processor manages data management


CurrentData
CurrentStatus
ProcessFilter
InData
InStatus
Process Step
OutData
OutStatus
CurrentData
CurrentStatus
Update
Append
ProcessLog
ProcessLog

Plugin structure

- InData -> OutData
  - Plugins performing treatment functions (e.g., imputation) should produce an OutData object
  - OutData should have same structure as InData
    - Requires unique identifier <unit_id>
    - Field names should remain the same
    - Only transformed data required; not a complete copy of InData
- InStatus -> OutStatus
  - InStatus only required if plugin makes use of existing status flags
  - OutStatus is encouraged
    - For review and selection functions, should include FTI flags
    - For treatment functions, should include imputation flags

```
""" Custom Plugin Name """

# Import packages required by the plugin

class plugin_name:
    """Description of plugin and parameters."""

    @classmethod
    def execute(cls, processor_data):
        # Execution of plugin

        try:

            # Import indata and/or instatus as needed
            indata = processor_data.get_dataset("indata", format="pandas")
            instatus = processor_data.get_dataset("instatus", format="pandas")

            # Import uservars from metadata table as dictionary
            uservars_dict = processor_data.current_uservars

            # Import other parameters from metadata tale as needed
            unit_id_name = processor_data.input_params.unit_id
            by_var = processor_data.by_varlist

            ### EXECUTE PLUGIN CODE HERE ###

            # Save copies of outdata or outstatus
            processor_data.outstatus = outstatus_final
            processor_data.outdata = outdata_final

            # Banff processor automates remaining data management

# Registers all plugin classes to the factory
def register(factory):
    factory.register("plugin_name", plugin_name)
```

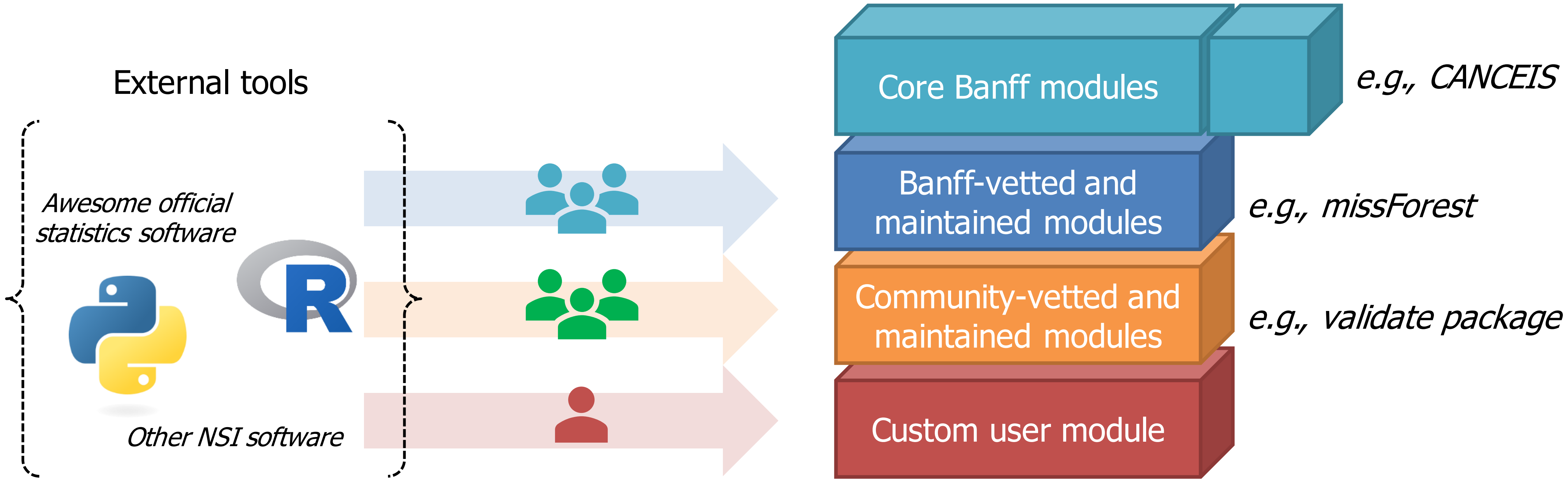
Plugins in development

- Banff team has already started developing plugins
- Some open-source packages already investigated:
  - scikit-learn (Python)
  - missForest (R)
- Packages developed for official statistics:
  - simputation
  - errorlocate
  - VIM

Ongoing work

- Continue to improve plugins
  - Improve template and documentation
  - Add automated features, e.g., auto-generation of imputation flags
- Begin building repository
  - Work with existing clients to convert custom SAS programs into plugins
  - Fill in gaps in existing Banff functionality
  - Add popular SDE packages from GitHub - SNStatComp/awesome-official-statistics-software: An awesome list of statistical software for creating and accessing official statistics
- Develop testing and vetting of plugins

Building a catalogue of Banff modules



Key takeaways

- Banff 3.0 will be free and open-source
  - Internal release January 2025
- Procedures
  - Underlying statistical methods remain the same
  - Changes to input/output structure, parameters
  - Updated to standard structure
- Processor
  - Major overhaul
  - Introduction of process blocks and process controls
- Banff plugins
  - Custom modules that can be called within the Processor

Final Remarks

- Two years later, have we achieved our objectives?
  - For the most part: yes
  - Aim to add some additional features after launch
- What challenges were faced migrating from SAS to open-source?
  - Required a team open to trying new things, and strong collaboration between methodology and IT partners
  - Needed to rethink our approach to a lot of problems (avoid “lift and shift”)

Acknowledgements

- Banff methodology team:
  - Management: Steve Matthews, Etienne Rassart
  - Development: Marouane Seffal
  - Consultation: Mark Stinner, Jonathan Baillargeon, Joël Bissonnette
- IT partners:
  - Management: Greg Ludwinski, Dany Brazeau, Martin Brière
  - Developers: Stephen Arsenault, Andrew Dombowsky, Anh Nguyen

Key questions

Could you / would you use the Banff processor as a data editing production platform?

Within the data editing community, is there interest in building and maintain a catalogue of modular, Banff-compatible data editing tools?

Thank you!

darren.gray@statcan.gc.ca

banff@statcan.gc.ca

Questions? Contact us: infostats@statcan.gc.ca