Checking Plausibility in Exploratory Data Analysis

Hermann Stolte
supervised by Prof. Matthias Weidlich[1] and Dr. Elisa Pueschel[2]
[1] Humboldt-Universität zu Berlin, [2] Deutsches Elektronen-Synchrotron DESY
hermann.stolte@hu-berlin.de

ABSTRACT

Exploratory data analysis is widespread across many scientific domains. It relies on complex pipelines and computational models for data processing, that are commonly designed collaboratively by scientists with diverse backgrounds for a variety of software stacks and computation environments. Here, a major challenge is the uncertainty about the correctness of analysis results, due to the high complexity of both, the actual data and the implemented analysis steps, and the continuous reuse and adaptation of data analysis pipelines in different application settings. This PhD project investigates how the design, adaptation, and evaluation of exploratory data analysis pipelines can be supported through automated plausibility assessment. To this end, we outline the requirements, our approach, and initial results for models and methods to enable plausibility checking in the context of exploratory data analysis.

1 INTRODUCTION

Today's large-scale research projects in domains such as Materials Science, Astrophysics, or Remote Sensing, to name just a few examples, often involve exploratory data analysis. Here, data from multiple distributed sources is integrated and analyzed collaboratively by scientists with diverse backgrounds, from various disciplines, and from different organizations. Key to the process are complex pipelines for scientific data processing, sometimes referred to as scientific workflows [10], which may be designed for a variety of software stacks and computation environments [13].

Main challenges here arise from the complexity of both, datasets and analysis steps, that makes results difficult to interpret and error-prone. With data being collected by multiple internal or external stakeholders each using their own methods in varying environments, bias and noise need to be accounted for. For complex research questions, interdisciplinary teams collaborate to combine domain and technical expertise, e.g., for creating computational models. Researchers from different backgrounds have an individual set of hidden assumptions about data and models, which can easily cause miscommunication and thus introduce errors in the design of the data processing pipelines.

An important aspect of exploratory data analysis is that the specific research questions evolve over time. As a consequence, pipelines and datasets are also subject to frequent changes [3]. While program code for complex analysis is error-prone in general, the evolution of pipelines and data is inherent to exploratory data analysis and amplifies the resulting challenges. Moreover, changes

Proceedings of the VLDB 2021 PhD Workshop, August 16th, 2021. Copenhagen, Denmark. Copyright (C) 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

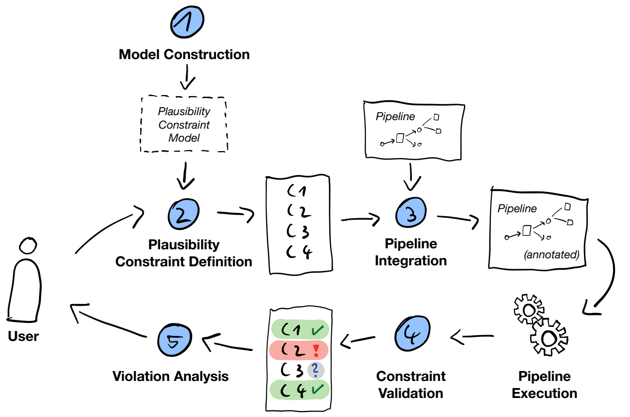

Figure 1: An illustration of the steps for plausibility analysis of scientific data processing pipelines.

originate not only from the restricted scope of a single research group or laboratory, but from external collaborators with limited or no involvement in the subsequent use of the updated code and data. In the absence of established routines for version management of datasets and data processing pipelines, such a setting can lead to hard-to-find bugs, especially when time-pressure is involved.

The overall uncertainty about the correctness of analysis results makes the development, maintenance, and evaluation of pipelines difficult. Support is needed for users to assess the plausibility of the results obtained by data processing pipelines. Therefore, this PhD project is dedicated to answer the following research question:

> *How to support the design, adaptation, and evaluation of exploratory data processing pipelines through automated plausibility analyses?*

We aim to answer this research question by providing the foundations for automated plausibility analysis, as illustrated in Fig. 1.

(1) Construct a meta-model for plausibility constraints
(2) Support users in defining plausibility constraints
(3) Integrate constraints into a given data processing pipeline
(4) Validate constraints during pipeline execution
(5) Enable users to identify root causes of constraint validations

The next section illustrates the need for automated plausibility assessment by a specific application case from the field of Astrophysics. §3 gives an overview of relevant related work. In §4, we outline our solution approach, before we conclude in §5.

2 BACKGROUND

In Astrophysics research, extreme acceleration processes (e.g., around black holes, exploding or merging stars) are of high scientific interest [4]. Here, energy is emitted as radiation across the whole electromagnetic spectrum, including gamma radiation. When gamma-rays

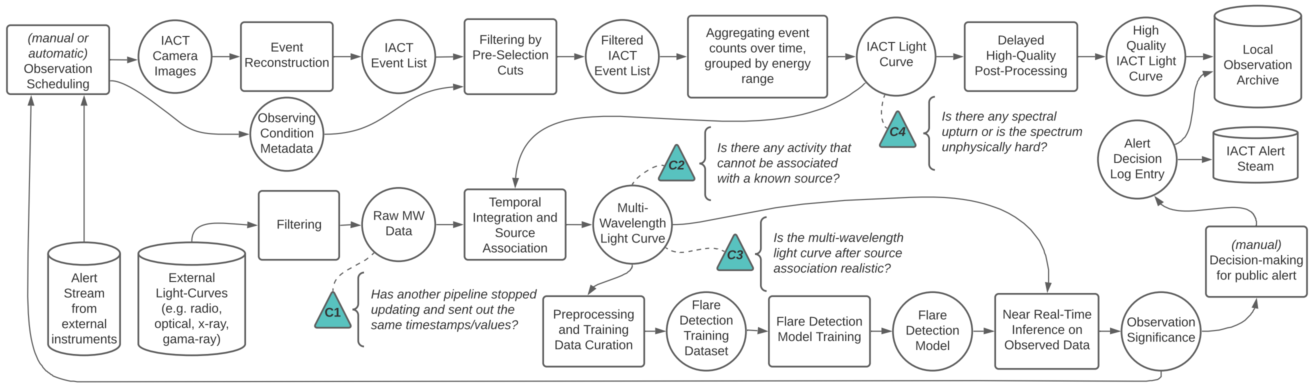
(manual or automatic) Observation Scheduling
IACT Camera Images
Event Reconstruction
IACT Event List
Filtering by Pre-Selection Cuts
Filtered IACT Event List
Aggregating event counts over time, grouped by energy range
IACT Light Curve
Delayed High-Quality Post-Processing
High Quality IACT Light Curve
Local Observation Archive
Observing Condition Metadata
C4
Is there any spectral upturn or is the spectrum unphysically hard?
Alert Decision Log Entry
IACT Alert Steam
C2
Is there any activity that cannot be associated with a known source?
Temporal Integration and Source Association
Filtering
Raw MW Data
Multi-Wavelength Light Curve
C3
Is the multi-wavelength light curve after source association realistic?
(manual) Decision-making for public alert
Alert Stream from external instruments
External Light-Curves (e.g. radio, optical, x-ray, gama-ray)
C1
Has another pipeline stopped updating and sent out the same timestamps/values?
Preprocessing and Training Data Curation
Flare Detection Training Dataset
Flare Detection Model Training
Flare Detection Model
Near Real-Time Inference on Observed Data
Observation Significance

Figure 2: A pipeline for near real-time flare detection in IACT and multi-wavelength data with four plausibility checks.

hit the Earth's atmosphere, they cause faint light showers, which are observable using ground-based telescopes. With so called Imaging Atmospheric Cherenkov Telescopes (IACT) [8], the physics behind acceleration processes can be studied, e.g., by analyzing temporal and spectral changes over time.

However, the respective data analysis is challenging. Background noise from cosmic rays needs to be filtered out and data can only be captured when specific observing conditions are met (e.g., given source visibility in the sky, no clouds, no bright moonlight). Also, reconstructing the properties of gamma ray photons from images of light showers introduces systematic bias (e.g., related to the visibility of light showers at low energies and different zenith angles).

To study acceleration processes effectively, IACT data is combined with data from other instruments (e.g., other gamma-ray, optical, x-ray or radar observatories), which is referred to as multi-wavelength (MW) data. The integration and joint analysis of such data is particularly challenging when research questions demand a near real-time analysis of transient phenomena, i.e., events that may be observable for only a few minutes.

For instance, consider the use case of near real-time IACT and multi-wavelength blazar[1] flare detection. Fig. 2 illustrates a pipeline for detecting flaring states and scheduling observations of blazars based on IACT and MW data. Here, data from different distributed sources need to be jointly analyzed to detect transient flaring states of blazars within hours to minutes. When a flaring state of interest is detected, the local observation schedule can be updated accordingly, and other observatories will be alerted.

Challenges for the analysis of IACT and MW data are imposed by the properties of the data, mainly its heterogeneity, sparsity, and inherent bias and noise. Moreover, the research questions tend to evolve, as for example, the definition of an interesting flare may be revised, which then leads to changes in the data processing pipeline. In addition, the setting in which the analysis is conducted imposes challenges on the technical side, as a variety of software stacks and computational environments is used, from machine learning frameworks, through batch processing systems and middleware for stream processing, to transient alert brokers. To cope with the challenges induced by the data, the analysis, and the infrastructure, the combined expertise and interdisciplinary collaboration of astrophysicists, computer scientists, and further engineers, is required. Against this background, the design of data processing pipelines is an error-prone process, which continuously raises the question of how to assess the plausibility of the produced data.

For the pipeline in Fig. 2, for instance, the plausibility of intermediate data can be assessed based on the following constraints:

- **C1** It may happen that an external pipeline to deliver and filter light curves from external instruments is faulty and sends identical measurements repetitively, which is implausible.
- **C2** In MW data, source activity can usually be observed by multiple instruments. A source with high activity being detected only by a single instrument is therefore suspicious.
- **C3** MW instruments can have a limited spatial resolution, so that observed activity may be associated with several source candidates. An erroneous source association may be discovered based on an implausible multi-wavelength light curve.
- **C4** Based on the current physical understanding of gamma-ray emission from blazars, certain features of an IACT spectrum, such as a spectral upturn, may be unexpected.

Note that violations of plausibility constraints are not necessarily related to errors in the data processing pipeline. Rather, they may also hint at unexpected and, therefore, particularly interesting physical phenomena. In any case, a plausibility assessment is beneficial and supports the design, adaptation, and evaluation of the pipeline.

3 RELATED WORK

Scientific workflows. Our ideas are related to the field of scientific workflows and workflow engines [10]. They have been researched for decades to provide accessibility and reproducibility. To this end, workflow engines (e.g., Apache Taverna, Galaxy, KNIME, and Pegasus) support for the design and execution of pipelines by large catalogs of operators, graphical user interfaces, and techniques for data integration. Yet, plausibility analysis as envisioned here is not part of their functionality.

Analysis of provenance data. Some scientific workflow engines collect provenance data [7] as a source for analyzing data lineage, i.e., to explain how a certain data item was created. Various models for provenance data have been defined in the literature, most

[1]A blazar is an astronomical object of particular scientific interest. When pointed to earth, it can be observed as a highly variable source of multi-wavelength radiation.

prominently PROV [2]. Moreover, there are tools available to explore and analyze provenance data, e.g., VisTrails [6], or to use it for distributed debugging, e.g., with BugDoc [12]. We argue that models for provenance data are a useful starting point for automatic plausibility analysis of data processing pipelines.

Program verification. Pipelines can be seen as programs, so that inspiration for plausibility analysis may be taken from the field of software engineering, especially software verification. Here, work on automatic test case generation [1] or invariant mining [11], which aims at identifying state-based expressions known to hold true, are particularly promising. Considering system faults to be a special case of implausible pipeline executions, and system invariants to be a special case of plausible pipeline behavior, inspiration can be drawn from approaches for discovering both.

4 CHECKING PLAUSIBILITY

This section outlines our approach to automated plausibility analysis. For each step outlined in Fig. 1, we discuss the requirements, our approach and preliminary results of how to address them.

4.1 Model

Automated plausibility assessment requires a model for making statements about data. The model needs to enable the specification of dependencies between data that is processed in a pipeline, while incorporating the following considerations:

Single or multi data item. Constraints may not only consider single data items, but the relation of multiple data items. For instance, considering our use case, a change in the rate of reconstructed events by an IACT telescope array is considered suspicious, if it coincides with a telescope entering or leaving the array. The plausibility of an IACT event list may therefore be assessed using observing condition metadata.

Single or multi pipeline execution. A basic constraint checks the value of one or more data items in isolation. However, plausibility assessment may require the joint analysis of a sequence of data items as a value distribution across multiple pipeline executions. For instance, for the example constraint C1, the sequence of n previous values of a data item is required to analyze the distribution variance.

Independent or dependent on external data. A constraint may refer to external data sources. For example, the constraint C2 requires access to an exhaustive catalog of known sources, that is not available as a data item during pipeline execution.

Our idea is to develop a model for plausibility constraints based on provenance graphs, which capture the relation between data items in a pipeline as a directed acyclic graph (DAG) [7]. Nodes in the graph are named entities and represent data items, whereas edges are causal dependencies. Data items also need to be linked to computational steps. We intend to realize this using the ProvONE model [5], which links data provenance with pipeline definitions (i.e., workflows in ProvONE terminology). Based thereon, the definition of a plausibility constraint includes:

(1) A collection of one or more ProvONE entities representing data items in a pipeline.
(2) Optionally, the location of external data referenced in the definition of the constraint.
(3) A constraint function that maps from the domains of the entities (and external data, if required) to a probability distribution, thereby expressing the plausibility of data items.

According to this model, C4 from Fig. 2 is captured as a function that maps from the domain of an IACT light curve to a probability distribution. Specifically, the constraint function assigns high probabilities when expected spectral features are present.

4.2 Constraint Definition

The effort needed to define constraints shall be minimized, since it is the crucial step in which a user has to invest time to benefit from automated plausibility checking down the line. Ideally, constraints could be found and suggested to a user for review. Research questions here are hence (1) *How to support the user in the manual definition of plausibility constraints,* and (2) *how to mine plausibility constraints automatically and assess their usefulness?*

To this end, we observe in our use case that constraints are closely related to physical models and laws. For example, the constraint C4 requires defining a function that maps features of a source's IACT light curve to a probability of plausibility. The latter is based on the source type, so that we aim to leverage physical models and laws for mining and defining constraints.

We further intend to investigate whether textual documentation (e.g., instrument specifications) can serve as a basis for constraint mining. This requires mapping the data entities of a provenance graph to named entities in natural language text, i.e., using techniques for named entity recognition and relation extraction. Once causal relations between entities have been discovered, they can be employed for suggestions in the manual definition of constraints.

4.3 Pipeline Integration

A key question in automated plausibility analysis is how to integrate the definition of plausibility constraints into existing pipelines. The objective of pipeline integration is two-fold:

- Data entities from the underlying provenance model need to be linked to software parameters and function arguments;
- Plausibility constraints need to be placed in a pipeline execution graph in the first possible position, where all required data items are available. This ensures that implausibilities can be detected and reacted to as early as possible.

Embedding plausibility constraints in a pipeline definition is not straightforward. Even in pipelines modeled as directed acyclic graphs, the same type of data may occur multiple times in an execution graph. Also, a constraint may concern multiple data items.

To enable pipeline integration, the functionality for plausibility analysis first needs to be implemented. Since exploratory analysis often relies on software toolkits that offer standard solutions for data management and analysis methods in a specific domain, these provide a suitable starting point for such an implementation. For example, for the analysis of IACT data, the libraries ctools and gammalib [9] are examples for such toolkits.

To integrate an implementation of plausibility analysis in the definition of a pipeline, our idea is to exploit the fact that processing steps in a pipeline are commonly defined as parameterized software interfaces, e.g., a function, a software class, or a service. Our approach for integrating plausibility checks, therefore, is to create a wrapper layer around common software components. In our use case, for instance, there are common types of plots for visualizing specific data entities, such as source spectra. If a pipeline uses pre-defined calls to construct common plot types, the link from function arguments to data entities and from plausibility constraints to execution placements can be derived automatically.

Listing 1 illustrates a Python wrapper for a call (`ctbutterfly()`) of the ctools library to create a so-called butterfly plot. Knowing that the input of the call is a light curve, the wrapper fetches applicable constraints (brown), and instantiates (green) and checks them (red).

```
def plauscheck_plot_ctbutterfly(iact_light_curve):
    entity = plauscheck.entities.IACT_LIGHT_CURVE
    constraintTypes = plauscheck.getConstraintsFor(entity)
    for constraintType in constaintTypes:
        constraint = constraintType(iact_light_curve)
        plauscheck.validate(constraint)
    ctools.ctbutterfly(iact_light_curve)

iact_light_curve = derive_light_curve(iact_event_list)
plauscheck_plot_ctbutterfly(iact_light_curve)
```

Listing 1: A wrapper function for integrating plausibility analysis into a library call for plotting IACT light curves.

4.4 Constraint Validation

Once plausibility constraints have been formulated and integrated, they need to be validated during pipeline execution. Given the common properties of scientific data, such validation is challenging:

- Data may be sparse, meaning that data is available only for certain (spatial or temporal) contexts. For our example of MW data integration, time series are available only in short time periods with varying cadences between instruments.
- Data may be uncertain in various ways. In the context of IACT and MW data, for example, there exists uncertainty in the distance estimation between source and observer, as well as in the source association, where activity in a region of interest may be associated with several source candidates.
- Data can be multi-resolution, e.g. measuring a phenomena in varying level of detail. In our use case, the sensitivity of different MW instruments varies greatly in terms of captured energy resolution, requiring a careful integration that respects uncertainty associated with the instrument type.

While we consider plausibility constraints to be stochastic, see §4.1, the above properties also motivate a probabilistic validation of them. This way, the confidence into the result of constraint validation is quantified. In the extreme case, some plausibility constraints cannot be validated due to data sparsity, uncertainty, or resolution differences. Moreover, the evolution of confidence over time needs to be taken into account. For some constraints, such as the constraint C1, the confidence changes over time. The more data is processed, the more reliable will be the assessment of the state of up-stream data sources, which increases the confidence in the ability to validate constraint C1. Against this background, we strive for algorithms for constraint validation that are rooted in Bayesian modeling.

4.5 Violation Analysis

Once a constraint indicates that certain data is implausible with a high-confidence, a user needs to assess whether there is indeed some error in the pipeline or whether the phenomenon is due to an unexpected, and often particularly interesting trend in the data. While the lineage of the data items for which implausibility is indicated can be derived directly from a respective provenance graph, effective violation analysis aims at a more targeted analysis. That is, given the set of all upstream data items, we strive for a separation of those that are actually correlated with the violation of the constraint from those that are irrelevant.

We intend to approach this use case from two angles. First, a correlation analysis between the data items created at intermediate, upstream steps of the pipeline may help to identify which type of data is likely to have a causal effect. Second, standard means for outlier detection for the distributions of data items created by upstream steps in the pipeline may provide clues on abnormal trends that led to downstream constraint violations.

5 CONCLUSIONS

In this paper, we motivated the need to support the design, adaptation, and evaluation of data processing pipelines with a case from Astrophysics. Against this background, we outlined the requirements, our approach, and initial results on how to enable comprehensive plausibility checking in exploratory data analysis. Having a first version of the model of plausibility constraints, our current research focuses on the derivation of constraints from the physical models underlying the illustrated pipeline for flare detection.

ACKNOWLEDGMENTS

We thank Iftach Sadeh (DESY) for valuable insights to gamma-ray astrophysics. The work is supported by the Helmholtz Einstein International Berlin Research School in Data Science (HEIBRiDS).

REFERENCES

[1] Saswat Anand, Edmund K Burke, Tsong Yueh Chen, et al. 2013. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software* 86, 8 (2013), 1978–2001.
[2] Khalid Belhajjame, Helena Deus, Daniel Garijo, et al. 2012. Prov model primer. *WWW Consortium* (2012).
[3] Jeffrey Chang. 2015. Core services: Reward bioinformaticians. *Nature* 520, 7546 (April 2015), 151–152.
[4] The Cherenkov Telescope Array Consortium. 2015. Science with the Cherenkov Telescope Array. *International Journal of Modern Physics D* (2015).
[5] Vıctor Cuevas-Vicenttın et al. 2016. ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance.
[6] Juliana Freire and Cláudio T. Silva. 2012. Making Computations and Publications Reproducible with VisTrails. *Comput. Sci. Eng.* 14, 4 (2012), 18–25.
[7] Melanie Herschel, Ralf Diestelkämper, and Houssem B. Lahmar. 2017. A Survey on Provenance: What for? What Form? What From? *VLDB J.* 26, 6 (2017), 881–906.
[8] Jamie Holder. 2015. Atmospheric Cherenkov Gamma-ray Telescopes. *arXiv e-prints*, (Oct. 2015), arXiv:1510.05675.
[9] Jea Knödlseder, M Mayer, C Deil, et al. 2016. GammaLib and ctools-A software framework for the analysis of astronomical gamma-ray data. *Astronomy & Astrophysics* 593 (2016), A1.
[10] Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, et al. 2016. Scientific Workflows: Moving Across Paradigms. *ACM Comput. Surv.* 49, 4, (2016).
[11] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. 2010. Mining Invariants from Console Logs for System Problem Detection. In *USENIX ATC*.
[12] Raoni Lourenço, Juliana Freire, and Dennis Shasha. 2020. BugDoc: Algorithms to Debug Computational Processes. In *SIGMOD*, 463–478.
[13] Victoria Stodden, Marcia McNutt, David H Bailey, et al. 2016. Enhancing reproducibility for computational methods. *Science* 354, 6317 (2016), 1240–1241.