Data Citation
Working Group Mtg @ P13
April 3 2019, Philadelphia

research data sharing without barriers

rd-alliance.org

Agenda

- 12:00 Introduction, Welcome
- 12:10 Short description of the WG recommendations
- 12:30 Reports by adopters / pilots
- 13:00 Review of Recommendations text / lessons learned
- 13:20 Other issues, next steps

Welcome and Intro

Welcome!

to the maintenance meeting of the

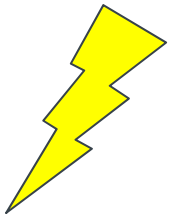
WGDC

Agenda

- 12:00 Introduction, Welcome
- 12:10 Short description of the WG recommendations
- 12:30 Reports by adopters / pilots
  - WGDC Recommendations in ESIP Guidelines
  - Implementation in Dendro
  - Implementation in CKAN
  - Implementation in OpenEO
- 13:00 Review of Recommendations text / lessons learned
- 13:20 Other issues, next steps

Identification of Dynamic Data

- Usually, datasets have to be static
  - Fixed set of data, no changes:
    no corrections to errors, no new data being added
- But: (research) data is **dynamic**
  - Adding new data, correcting errors, enhancing data quality, …
  - Changes sometimes highly dynamic, at irregular intervals
- Current approaches
  - Identifying entire data stream, without any versioning
  - Using “accessed at” date
  - “Artificial” versioning by identifying batches of data (e.g. annual), aggregating changes into releases (time-delayed!)



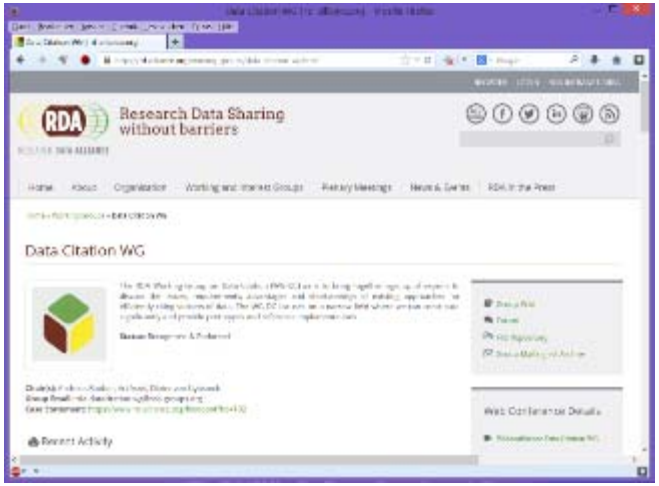
- Would like to identify precisely the **data as it existed at a specific point in time**

Granularity of Subsets

- What about the **granularity** of data to be identified?
  - Enormous amounts of CSV data
  - Researchers use specific subsets of data
  - Need to identify precisely the subset used
- Current approaches
  - Storing a copy of subset as used in study -> scalability
  - Citing entire dataset, providing textual description of subset -> imprecise (ambiguity)
  - Storing list of record identifiers in subset -> scalability, not for arbitrary subsets (e.g. when not entire record selected)

- Would like to be able to identify precisely the **subset of (dynamic) data used** in a process

RDA WG Data Citation



- Research Data Alliance
- WG on **Data Citation:**
  Making Dynamic Data Citeable
- March 2014 – September 2015
  - Concentrating on the problems of **large, dynamic (changing) datasets**
- Final version presented Sep 2015 at P7 in Paris, France
- Endorsed September 2016 at P8 in Denver, CO

- Since: support for take-up/adoption, lessons-learned

https://www.rd-alliance.org/groups/data-citation-wg.html

Dynamic Data Citation

We have: Data + Means-of-access (“query”)

Dynamic Data Citation

We have: Data + Means-of-access (“query”)

Dynamic Data Citation:
Cite (dynamic) data dynamically via query!

Dynamic Data Citation

We have: Data + Means-of-access ("query")

Dynamic Data Citation:
Cite (dynamic) data dynamically via query!

Steps:

1. Data → versioned (history, with time-stamps)

Dynamic Data Citation

We have: Data + Means-of-access ("query")

Dynamic Data Citation:
Cite (dynamic) data dynamically via query!

Steps:

1. Data → versioned (history, with time-stamps)

Researcher creates working-set via some interface:

Dynamic Data Citation

We have: Data + Means-of-access ("query")

Dynamic Data Citation:
Cite (dynamic) data dynamically via query!

Steps:

1. Data → versioned (history, with time-stamps)

Researcher creates working-set via some interface:

2. Access → **store & assign PID to "QUERY"**, enhanced with
   - **Time-stamping** for re-execution against versioned DB
   - **Re-writing** for normalization, unique-sort, mapping to history
   - **Hashing** result-set: verifying identity/correctness

   leading to landing page

Data Citation – Deployment

- Researcher uses workbench to identify subset of data
- Upon executing selection („download“) user gets
  - Data (package, access API, …)
  - PID (e.g. DOI) (Query is time-stamped and stored)
  - Hash value computed over the data for local storage
  - Recommended citation text (e.g. BibTeX)
- PID resolves to landing page
  - Provides detailed metadata, link to parent data set, subset,…
  - Option to retrieve original data OR current version OR changes
- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned
- Query store aggregates data usage

Data Citation – Deployment

Note: query string provides excellent provenance information on the data set!

- ubset of data
- er gets
  - Data (package, acce s API, …)
  - PID (e.g. DOI) (Que is time-stamped and stored)
  - Hash value compute over the data for local storage
  - Recommended citati n text (e.g. BibTeX)
- PID resolves to land g page
  - Provides detailed metadata, link to parent data set, subset,…
  - Option to retrieve original data OR current version OR changes
- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned
- Query store aggregates data usage

Data Citation – Deployment

- Note: query string provides excellent provenance information on the data set!
- ubset of data
- er gets
  - Data (pac
  - PID (e.g.
  - Hash valu
  - Recommended citat Text (e.g. BibTeX)
- This is an important advantage over traditional approaches relying on, e.g. storing a list of identifiers/DB dump!!!
- PID resolves to land g page
  - Provides detailed metadata, link o parent data set, subset,…
  - Option to retrieve original data OR current version OR changes
- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned
- Query store aggregates data usage

Data Citation – Deployment

- Note: query string provides excellent provenance information on the data set!
- ubset of data
- er gets
  - Data (pac
  - PID (e.g.
  - Hash valu
  - Recommended citati Text (e.g. BibTeX)
- This is an important advantage over traditional approaches relying on, e.g. storing a list of identifiers/DB dump!!!
- PID resolves
  - Provides det
  - Option to ret
- Identify which parts of the data are used. If data changes, identify which queries (studies) are affected
- Upon activating PID associated w h a data citation
  - Query is re-executed against time-st mped and versioned DB
  - Results as above are returned
- Query store aggregates data usage

Data Citation – Recommendations

Preparing Data & Query Store

- R1 – Data Versioning
- R2 – Timestamping
- R3 – Query Store

When Resolving a PID

- R11 – Landing Page
- R12 – Machine Actionability

When Data should be persisted

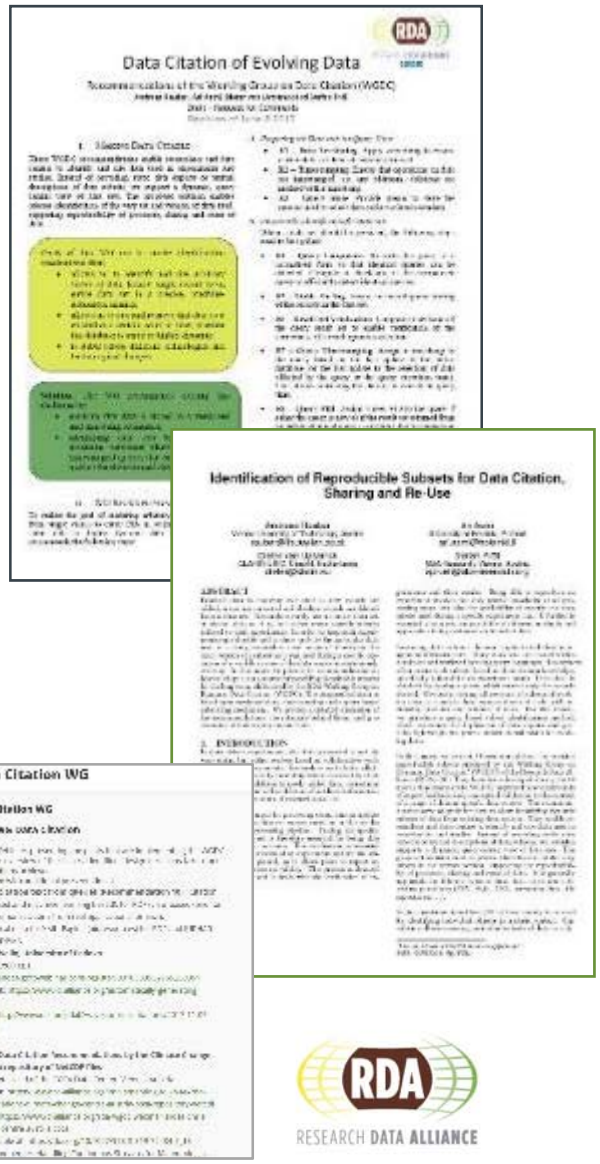
- R4 – Query Uniqueness
- R5 – Stable Sorting
- R6 – Result Set Verification
- R7 – Query Timestamping
- R8 – Query PID
- R9 – Store Query
- R10 – Citation Text

Upon Modifications to the Data Infrastructure

- R13 – Technology Migration
- R14 – Migration Verification

Data Citation – Output

- **14 Recommendations grouped into 4 phases:**
- **2-page flyer**
  https://rd-alliance.org/recommendations-working-group-data-citation-revision-oct-20-2015.html
- **More detailed report: Bulletin of IEEE TCDL 2016**
  http://www.ieee-tcdl.org/Bulletin/v12n1/papers/IEEE-TCDL-DC-2016_paper_1.pdf
- **Adopter's presentations, webinars and reports**
  https://www.rd-alliance.org/group/data-citation-wg/webconference/webconference-data-citation-wg.html



Adopters

- Series of Webinars presenting implementations
  - Recordings, slides, supporting papers
  - https://www.rd-alliance.org/group/data-citation-wg/webconference/webconference-data-citation-wg.html
  - **Automatically generating citation text from queries (Recommendation 10) for RDBMS and XML data sources**
  - Implementing of the RDA Data Citation Recommendations by the **Climate Change Centre Austria (CCCA) for a repository of NetCDF files**
  - Implementing the RDA Data Citation Recommendations for **Long-Tail Research Data / CSV files**
  - Implementing the RDA Data Citation Recommendations in the **Distributed Infrastructure of the Virtual and Atomic Molecular Data Center (VAMDC)**
  - Implementation of Dynamic Data Citation at the **Vermont Monitoring Cooperative**
  - Adoption of the RDA Data Citation of Evolving Data Recommendation to **Electronic Health Records**

RDA Recommendations - Summary

- ***Benefits***
  - Allows **identifying, retrieving and citing the precise data subset** with minimal storage overhead by only storing the versioned data and the queries used for extracting it
  - Allows retrieving the data both **as it existed** at a given point in time as well as the **current view** on it, by re-executing the same query with the stored or current timestamp
  - It allows to cite even an **empty set**!
  - The query stored for identifying data subsets provides valuable **provenance data**
  - Query store collects **information on data usage**, offering a basis for data management decisions
  - **Metadata** such as checksums support the verification of the correctness and **authenticity** of data sets retrieved
  - The same principles work for **all types of data**

Agenda

- 12:00 Introduction, Welcome
- 12:10 Short description of the WG recommendations
- 12:30 Reports by Adopters / Pilots
  - ESIP Guidelines
  - ISO 690 Standard
  - NICT Plattform
  - No-SQL (MongoDB) for CKAN
  - Dendro Data Repository
  - OpenEO: Earth Observation Data Center (EODC)
- 13:00 Review of Recommendations text / lessons learned
- 13:20 Other issues, next steps

Reference Implementations & Standards

Reference Implementations

- MySQL – Stefan Pröll
- CSV via MySQL – Stefan Pröll
- CSV-files via GIT – Kristof Meixner
- NoSQL via MongoDB in CKAN – Florian Wörister

Standards

- ISO 690:2010, Information and documentation — Guidelines for bibliographic references and citations to information resources, p.43, Sec. 16.13.5, 4th ed. 2018-09-11 – Juha Hakala
- ESIP: Data Citation Guidelines for Earth Science Data Version 2 (draft) – Mark Parsons

Key Adopters so far…

- Electronic Health Records at the Univ. of Washington in St. Louis – Leslie McIntosh
- Vermont Monitoring Cooperative – James Duncan
- Virtual Atomic and Molecular Data Center (VAMDC) – Carlo Maria Zwölf
- Climate Change Centre Austria (CCCA)
- Open Earth Observation - Earth Observation Data Center (EODC) – Wolfgang Wagner, Bernhard Gößwein
- IPSL (CNRS) – Sebastian Denville

Ongoing Adoption Projects

- Ocean Networks Canada - Reyna Jenkyns
- Deep Carbon Observatory - Mark Parsons
- MyHealth MyData – Rudolf Mayer
- Smart Data Platform at NICT, Japan – Koji Zettsu
- Dendro Data Repository – João da Silva

RDA WGDC Recommendations in ESIP Guidelines

Mark Parsons

research data sharing without barriers

rd-alliance.org

ESIP: Data Citation Guidelines for Earth Science Data Version 2

Data Citation Guidelines for Earth Science Data Version 2

Suggested Citation:

ESIP Data Preservation and Stewardship Committee. 2019. *Data Citation Guidelines for Earth Science Data. Ver. 2.* Earth Science Information Partners. https://doi.org/10.....

Table of Contents

ESIP
The Federation of Earth Science Information Partners
Fostering connections to make data matter

Dynamic and Micro-citation

This may be the most challenging aspect of data citation. It is necessary to enable "micro-citation" or the ability to refer to the specific data used—the exact files, granules, records, etc. from a particular version. Scientifically, this is to enable reproducibility by providing a precise reference to the data used. It may, however, impact the credit or attribution functions of a citation. Different subsets of a larger collection may have been created by different people. As discussed in Data Within a Larger Work, mechanisms for crediting at finer granularity are still being developed.

Mechanisms for referencing and providing access to precise subsets of data are more established. Ideally, the repository should provide a PID that resolves to the precise subset and version of the data used. We recommend that repositories implement the Research Data Alliance (RDA) Recommendation on Scalable Dynamic Data Citation, which provides a PID for a particular query.

We recognize, however, that not all repositories have the ability to implement the RDA Recommendation so other approaches that can work reasonably well, at least for human interpretation, may be used.

Versioning

In all cases, it is very important to carefully track and document versions of the data set. Individual stewards and data centers will need to develop and follow their own practices, but

research data sharing without barriers
rd-alliance.org

RDA
RESEARCH DATA ALLIANCE

RDA WGDC Recommendations in ISO 690, Information and documentation — Guidelines for bibliographic references and citations to information resources

Juha Hakala

research data sharing without barriers

rd-alliance.org

ISO 690:2010

Information and documentation — Guidelines for bibliographic references and citations to information resources

Information et documentation — Principes directeurs pour la rédaction des références bibliographiques et des citations des ressources d'information

Contents Page

RESEARCH DATA ALLIANCE

ISO 690:2010

Information and documentation — Guidelines for bibliographic references and citations to information resources

Information et documentation — Princip
des références bibliographiques et des
d'information

Contents

16.13.5 Component parts

Research data sets may be dynamic: new data can be added, errors corrected and new items added. A data set may also be available in different formats, enabling different kind of uses. The data used shall be cited accurately so that the version used can be identified from the reference.

Researchers may filter and sort data sets in order to create subsets, which are then used instead of the original data set. Citations should facilitate re-creation of the data used in the original research, even when this requires technological changes such as migrations[22].

This can be accomplished if data sets are time-stamped and versioned, and if there are time-stamped queries which can be used to retrieve the desired subset of the data. This approach, promoted by the Research Data Alliance Data Citation WG[23], supports a dynamic, query centric view of research data sets, which is a prerequisite for accurate data citations. In this approach, persistent identifiers are assigned to queries to be executed against data sets[24].

EXAMPLE 5 LEUPRECHT [et al.]. tas_CNRM-CERFACS-CNRM-CM5_RCP4.5_r1i1p1_CLMcom-CCLM4-8-17, [data set]. Version 2. [Subset used: January to June 2014]. Vienna, Austria. CCCA Data Centre [distributor], 2016. Available from: https://hdl.handle.net/20.500.11756/93887ecf. [accessed 2017-06-28].

[21] https://www.w3.org/TR/prov-dm/
[22] https://www.rd-alliance.org/system/files/documents/iPRES2016-Proell.pdf
[23] https://www.rd-alliance.org/group/data-citation-wg/outcomes/data-citation-recommendation.html
[24] https://www.rd-alliance.org/system/files/RDA-DC-Recommendations_151020.pdf

Data Provenance for Smart Data Platform

Koji Zettsu, Yasuhiro Murayama
National Institute of Information and Communications Technology
Japan

research data sharing without barriers
rd-alliance.org

Data Provenance for Smart Data Platform

A Draft Plan

Koji Zettsu, Yasuhiro Murayama

National Institute of Information and Communications Technology

Japan

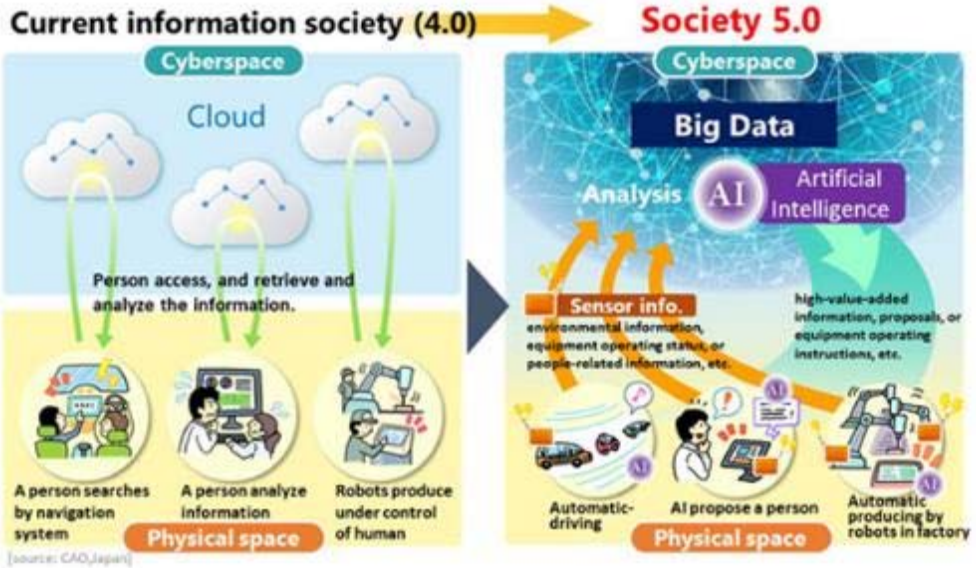
April 1, 2019

History

Sep. 2013	Contributed article in *"out of cite, out of mind"* *(Data Science Journal 12(13))* published by CODATA-ICSTI Task Group on Data Citation Standards and Parities
Sep. 2013	**[RDA-P2]** Started discussion with A. Rauber (RDA Data Citation WG chair)
Nov. 2014	Research presentation at SciDataCon 2014 (New Delhi): ***"Mining Data Citation for Usage Analysis of Open Science Data"***
Oct. 2016	RDA Data Citation WG recommendation published (DOI: http://dx.doi.org/10.15497/RDA00016)
Apr. 2017	**[RDA-P9]** Kick-off presentation of Japanese adoption (***dynamic data citation***) at RDA Data Citation WG
Sep. 2017	**[RDA-P10]** "Interim report" talk of the *dynamic data citation* work
Oct. 2017	Research presentation at CODATA 2017 (St. Petersburg) : "***A Data Citation System Framework for Identification of Evolving Data***"
Apr. 2018	Start *present work* **data provenance for a Smart Data Platform** in NICT Real Space Information Analysis project

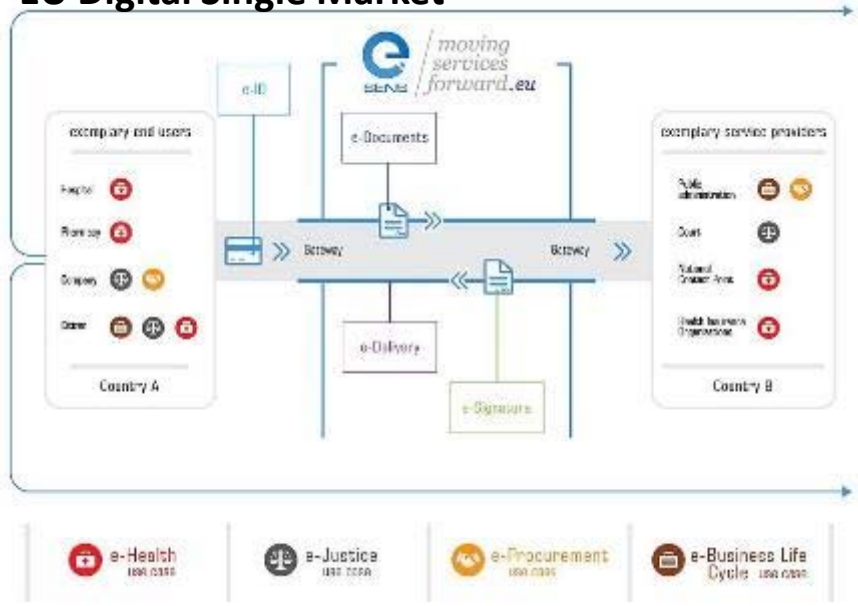
Background: Super Smart Society

- A high degree of convergence between cyberspace (virtual space) and physical space (real space) through IoT
- Interdisciplinary data collaboration for complex problem solving in smart societies
- Data-driven AI with Smart Data
  - IoT big data collected and processed to be turned into 'actionable information'
  - Fairness, Accountability, Transparency in Machine Learning (FAT/ML)



Source: Society 5.0, Cabinet Office of Japan, https://www8.cao.go.jp/cstp/english/society5_0/index.html

EU Digital Single Market



Source: EU Digital Single Market, https://en.wikipedia.org/wiki/Digital_Single_Market#/media/File:E-SENS_architecture.jpg

NICT Cross-Data Collaboration Platform

BIG DATA ANALYTICS LABORATORY

Smart Service Applications

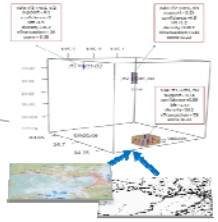
APIs

Dynamic Map Creation & Navigation

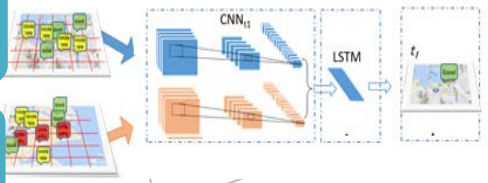
Complex Event Analysis & Prediction

Event Data Collection

Event Data Mining



Deep Learning for Event Data

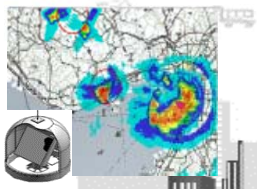


Event Data Warehouse

NICT Integrated Testbed

11 domains of sensing data
(as of Jan., 2019)

Weather



Atmosphere

Traffic

Health

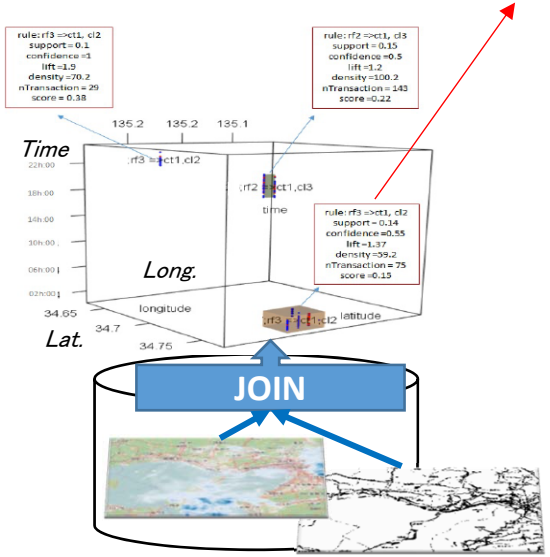
SNS, etc.

Application: Smart Sustainable Mobility

BIG DATA ANALYTICS LABORATORY

- Discovery of association rules between traffic and environmental events

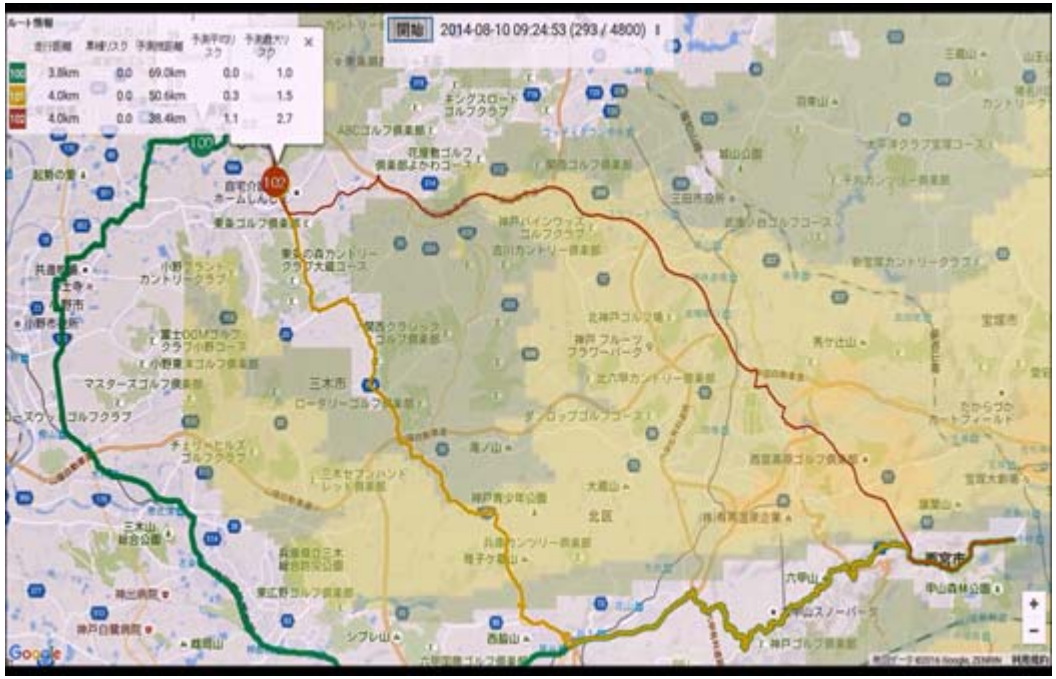
rainfall:15-20mm/h ⇒
speed: <10km/h,
congestion_length:300m-600m
• support=0.14, confidence=0.55, lift=1.37
• # transactions=75, density=59.2



Precipitation radar data

Traffic congestion data

- Realtime prediction of mobility risks
- Dynamic search for safer route



Shortest route
Safer route (25%-lower risk)
Risk-free route

Application: Smart Environmental Healthcare

BIG DATA ANALYTICS LABORATORY

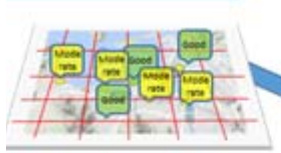
Past 24-hours atmospheric observation data (input)

Deep Learning (CRNN)

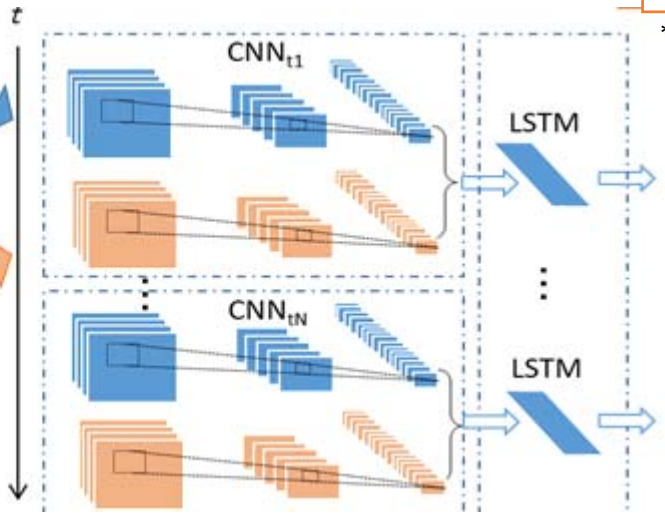
Next 1-12 hours prediction of air quality index* (output)

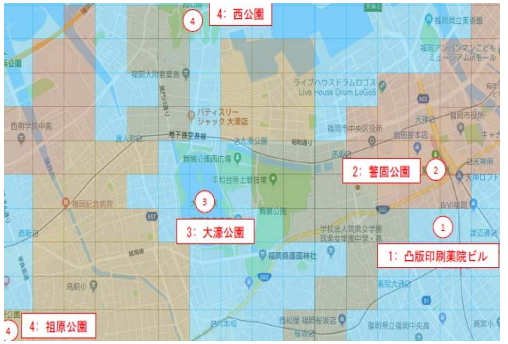
*) Significance of health effect by polluted air

- Monitoring station data (local)
- Monitoring station data (regional - transborder)



$PM_{2.5}$, Ox, CO, NO_2, SO_2, SPM, PM_{10}, 温湿





Participatory Sensing

- Small Atmospheric sensor
- Lifelog sensor

- Wearable health sensor

- Onboard atmospheric sensor
- Drive recorder

AQI-optimal route guide app.

Training data augmentation using user-collected data

Cross-Data Collaboration Framework

BIG DATA ANALYTICS LABORATORY

Collection → Association → Training → Prediction

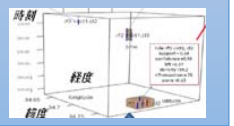
Sensing data

- Weather (DIAS, PANDA, etc.)
- Atmosphere (AEROS, etc.)
- Traffic (ITARDA, JARTIC, etc.)
- Probe car
- Healthcare (medical recept, wearable sensor, etc.)
- SNS (Twitter)

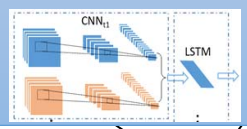
Service-specific sensing data collection

Data Loader API

Association Mining API



Associative Prediction API



Map Creation API

Route search API

Alert API

Training dataset

Prediction result data

Event data (common format)

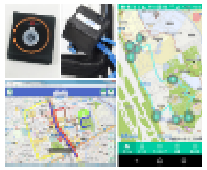
11 domains・24 kinds・13.3 billion records・13.8TB (as of Jan. 2019)

NICT Integrated Testbed

DB servers×8, Storage servers× 2 ,
Analysis server×10, Cluster server×36

● Smart sustainable mobility

● Smart environmental healthcare

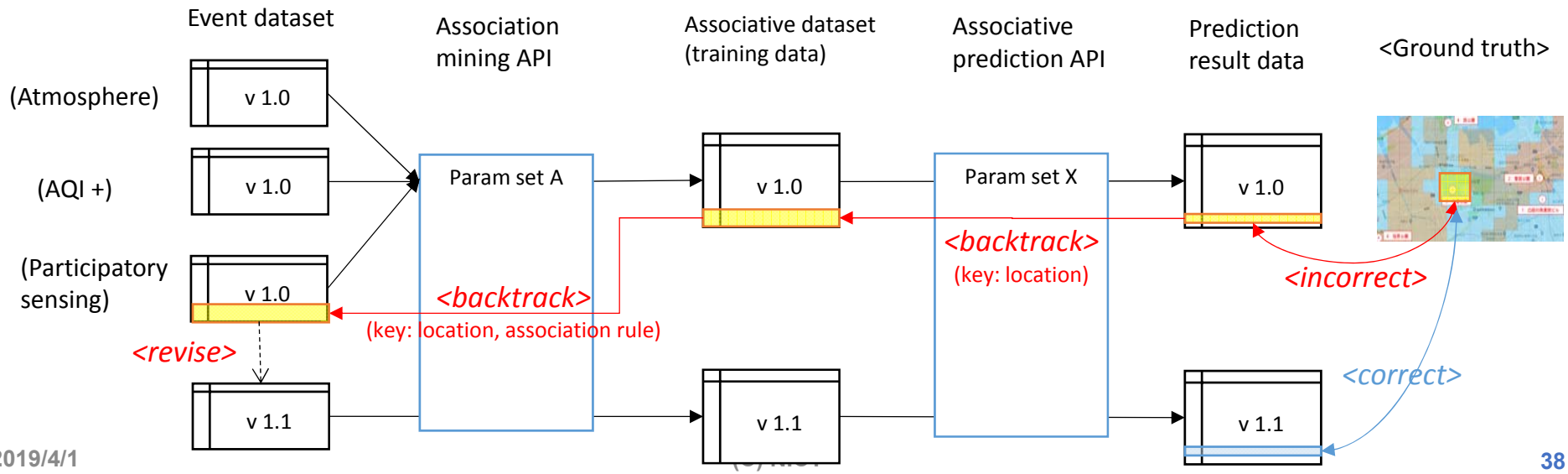


Feedback

Basic Concepts

- Generate data provenance information as a data flow graph in Cross-Data Collaboration Framework based on API logging and dataset versioning
  - Enhance the ***dynamic data citation*** method presented at CODATA2017*
    *) *Zettsu, K. and Murayama, Y.: A Data Citation System Framework for Identification of Evolving Data, CODATA2017, St. Petersburg (October, 2017)*
- Revise API input datasets and/or parameters by "back-tracking" a data provenance graph
- A **workbench tool** for supporting trial-and-error revision work
  - Backtracking a data provenance graph
  - Revision control of datasets and parameters in a data provenance graph
  - Application-specific summary (or *view*) of a large data provenance graph

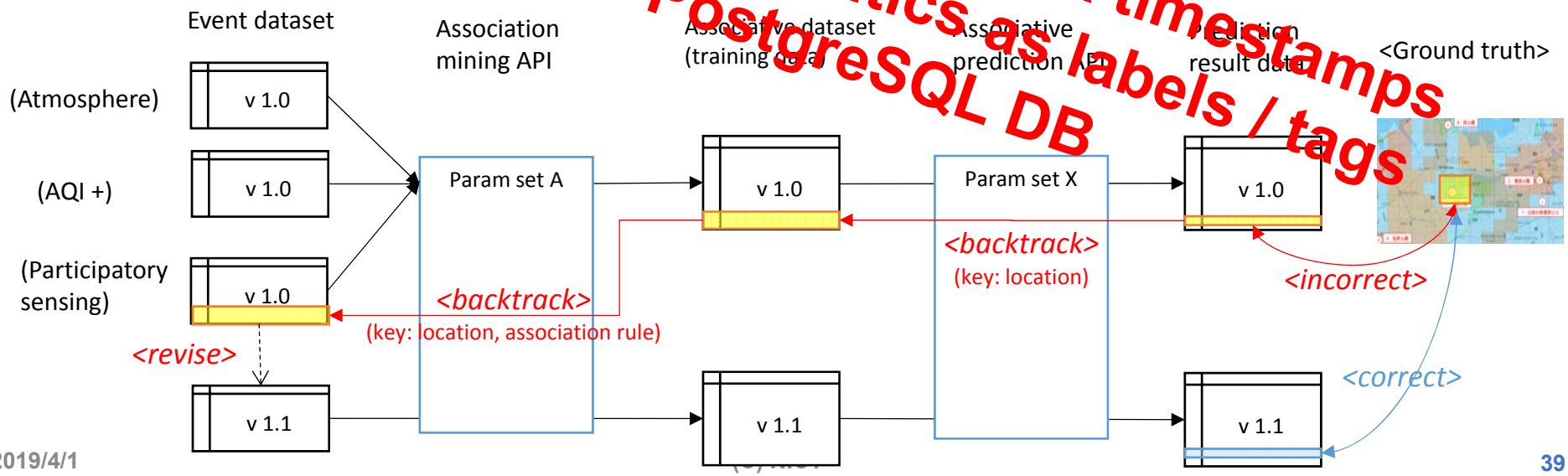
[Example of Smart Environmental Healthcare]



Basic Concepts

- Generate data provenance information as a data flow graph in Cross-Data Collaboration Framework based on API logging and dataset versioning
  - Enhance the ***dynamic data citation*** method presented at CODATA2017*
    *) *Zettsu, K. and Murayama, Y.: A Data Citation System Framework for Identification of Evolving Data, CODATA2017, St. Petersburg (October, 2017)*
- Revise API input datasets and/or parameters by "back-tracking" a data provenance graph
- A **workbench tool** for supporting trial-and-error revision work
  - Backtracking a data provenance graph
  - Revision control of datasets and parameters in a data provenance graph
  - Application-specific summary (or *view*) of a large data provenance graph

[Example of Smart Environmental Healthcare]



Note:
- Versioning via timestamps
- Semantics as labels / tags
- PostgreSQL DB

R&D Plan from April 2019 to March 2020

BIG DATA ANALYTICS LABORATORY

- Prototype implementation on NICT Cross-Data Collaboration Platform
  - API call logging and dataset versioning based on database management system (PostgreSQL/Greenplum): UDF, virtual table, materialized view etc.
  - Workbench tool for Cross-Data Collaboration Framework in SQL and Python language
- Usability testing by platform users: developers and data scientists of target smart applications:
  - **Smart sustainable mobility**: mobility risk prediction based on real time collection of traffic and environmental data (Fujisawa, etc.)
  - **Smart environmental healthcare**: personal AQI prediction for walking /running courses with participatory sensing (Tokyo Olympic Game 2020 marathon course, etc.)
- Publish a technical report on the architecture, use case, and reference implementation based on the prototype and preliminary experiments
  - To be published at Smart IoT Acceleration Forum (Japan)

- Opportunities for joint work with RDA Data Citation WG and/or related groups/projects (?)

Dynamic Data citation support for CKAN Repositories

Florian Wörister

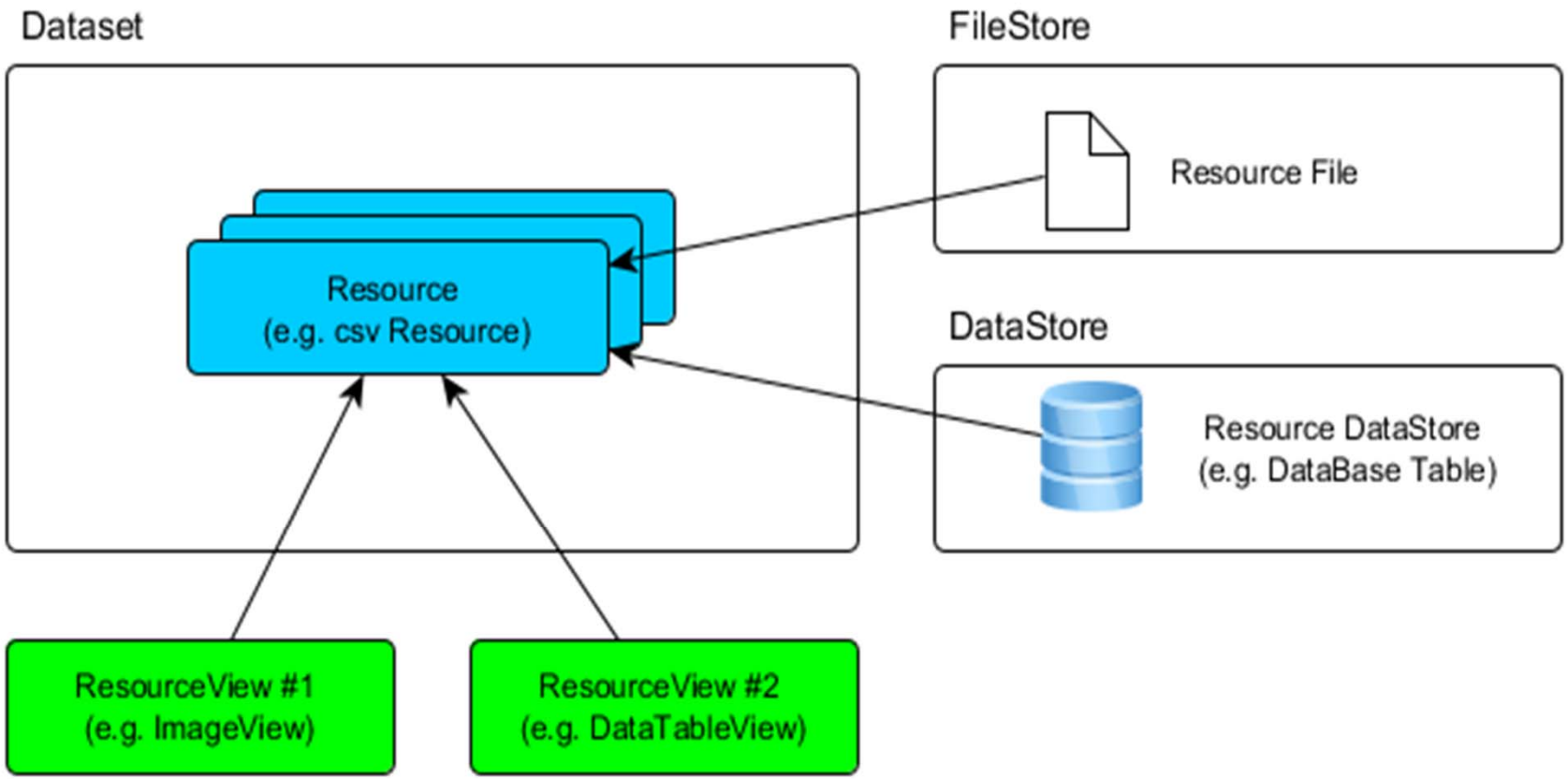
research data sharing without barriers

rd-alliance.org

CKAN

- One of the leading Open Source data portal solutions → “Wordpress for Research Data”
- Organized in datasets that contain several resources (e.g. csv files, images, text documents, etc.)
- Provides a plugin API (python)

CKAN Architecture



CKAN DataStore

- CKAN has a FileStore and a DataStore
  - FileStore → stores resources (e.g. a csv File) in directory
  - DataStore → stores resources in a structured form into a database

- There is a 1-to-1 relationship between a FileStore resource and a DataStore resource

- A DataPusher service extracts content of tabular files (e.g. csv) and automatically pushes the data into a datastore

- **It is possible to implement a custom DataStore Backend!**

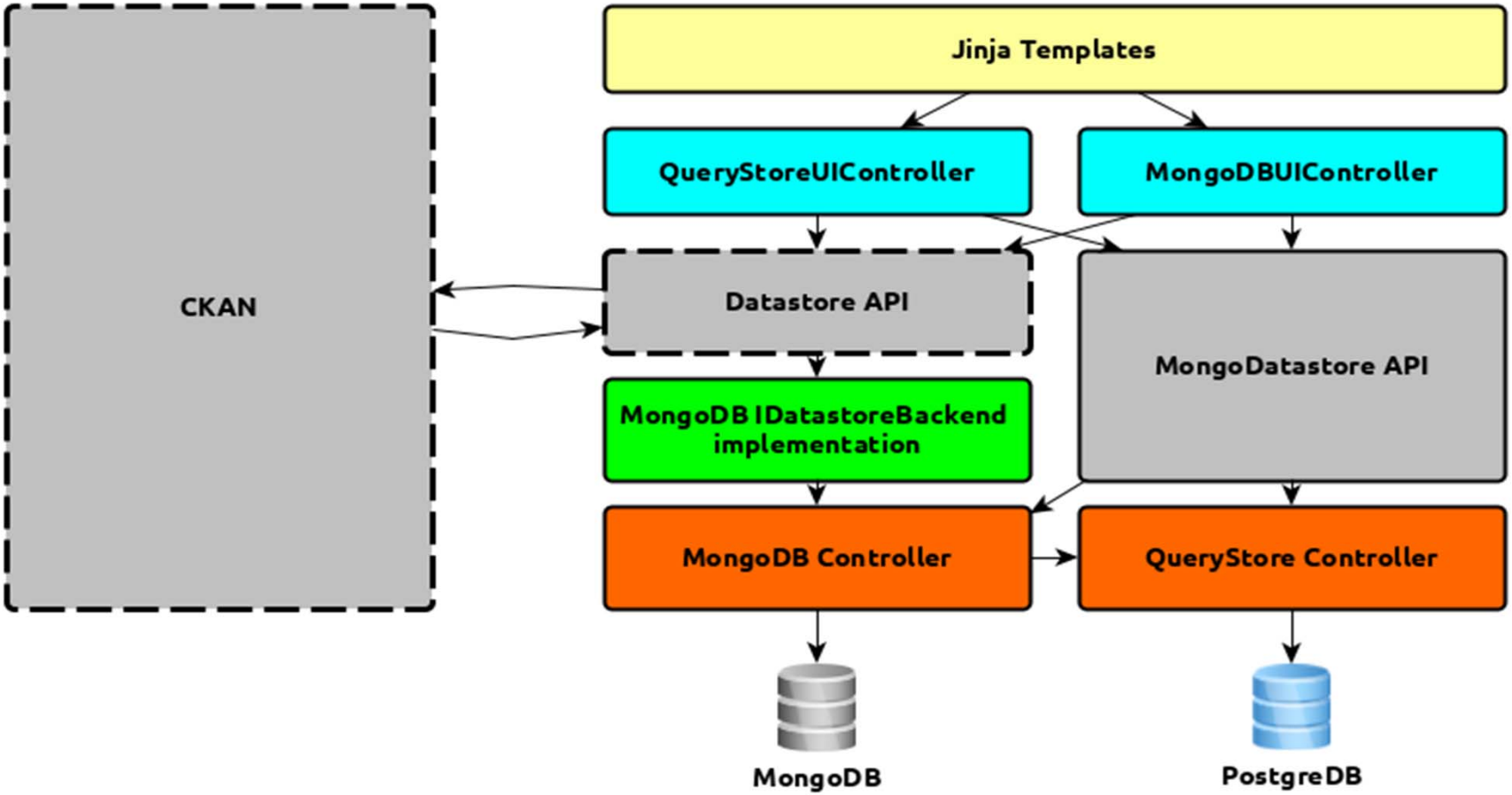
The CKAN Plugin API

- Plugins are implemented as Python Modules
- Plugins can:
  - Provide implementations of already defined Interfaces
  - Can override Jinja Templates of CKAN
  - Can extend the CKAN API

The Goal

- Implement a CKAN plugin that:
  - follows the **RDA recommendations for Data Citation**
  - provides an interface to fetch data from **external data sources** (relational database, NoSQL database, filestorage, etc.)
  - sticks to the **DataStore Backend interface**, **compatible** with already existing CKAN instances and plugins

The *MongoDatastore* CKAN extention



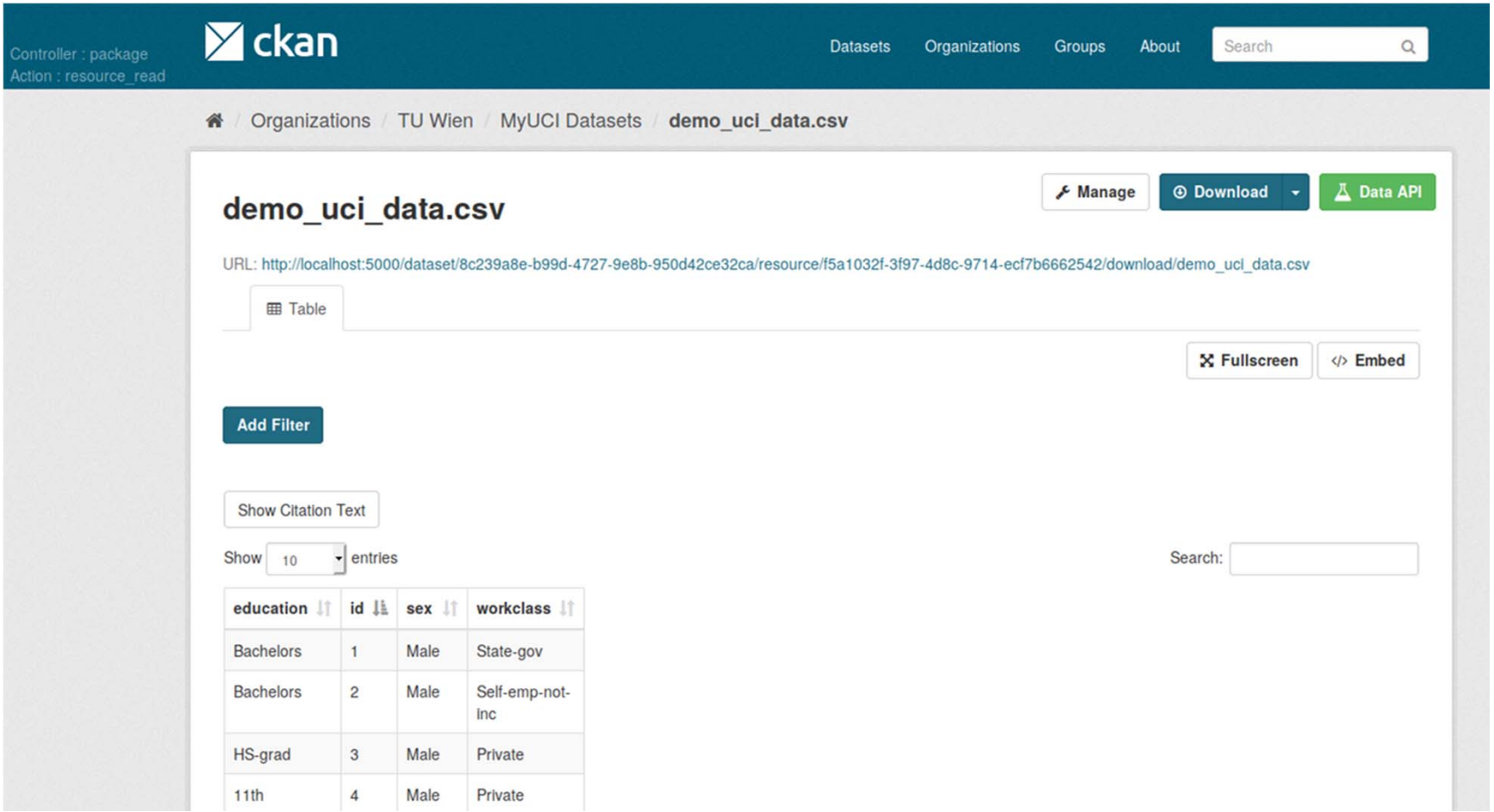
MongoDB terminology in a nutshell

Relational Database	MongoDB
Database	Database
Table	Collection
Row (Tuple, Entry, …)	Document
Column	Field

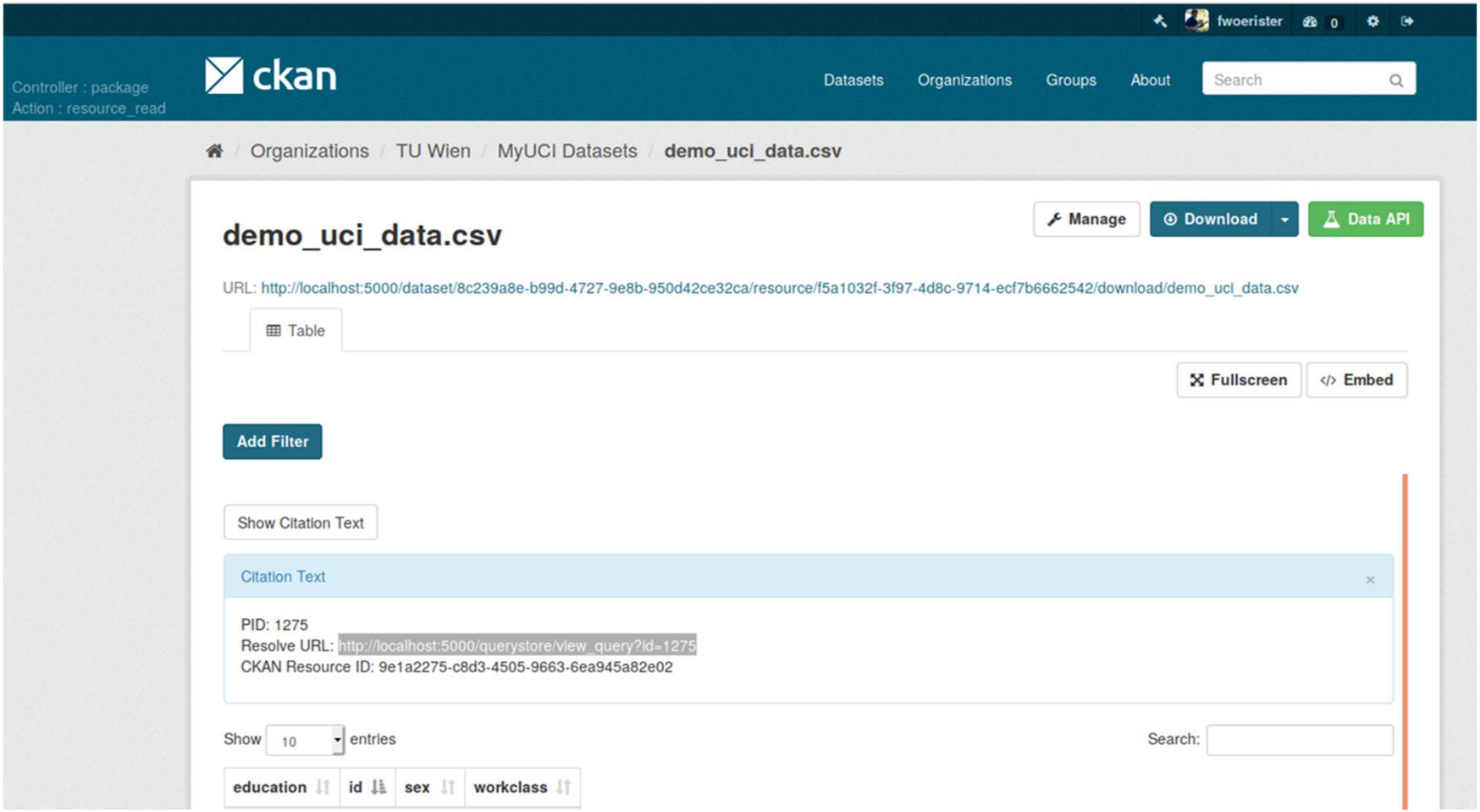
The MongoDB Aggregation Framework

- A query consists of several ‘stages’
- Each stage modifies the documents it receives and passes them to the next stage
- In the plugin 3 main stages are applied:
  - **History Stage** → transform the documents to the state how the database looked at a certain timestamp
  - **Query Stage** → query the data
  - **Paging Stage** → only retrieve the page of interest

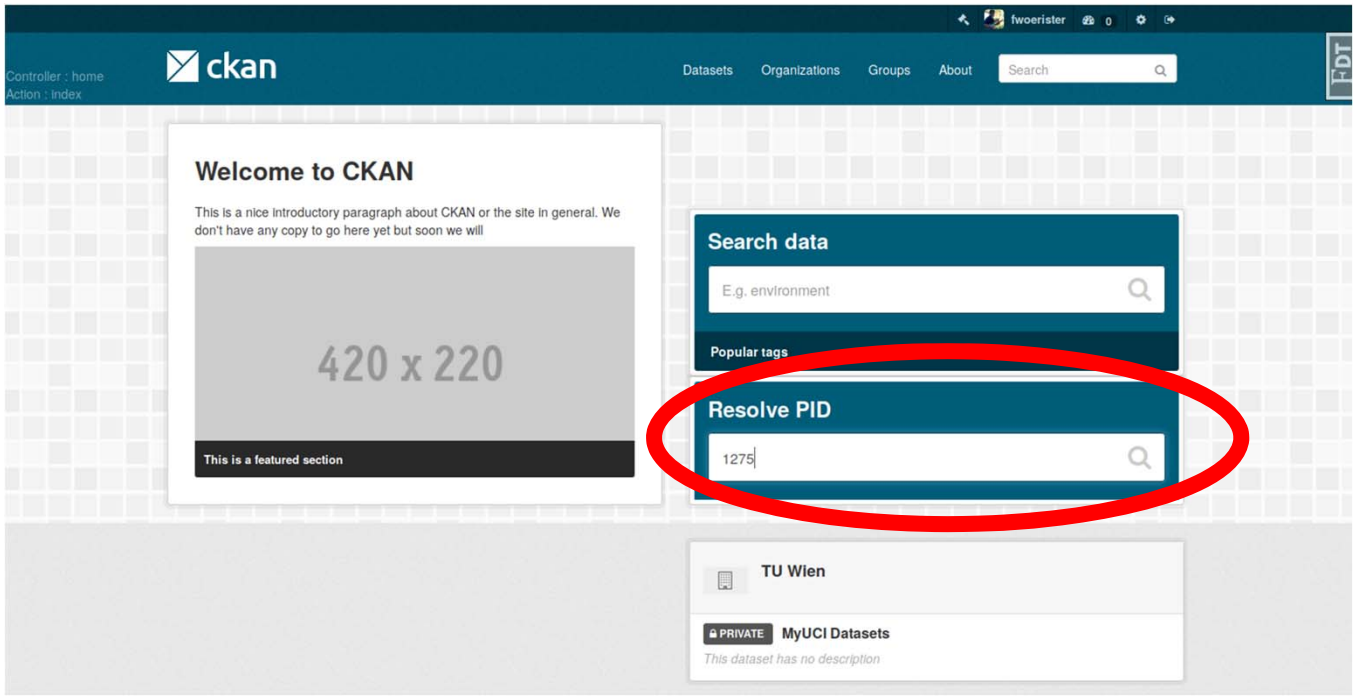
Table View in CKAN



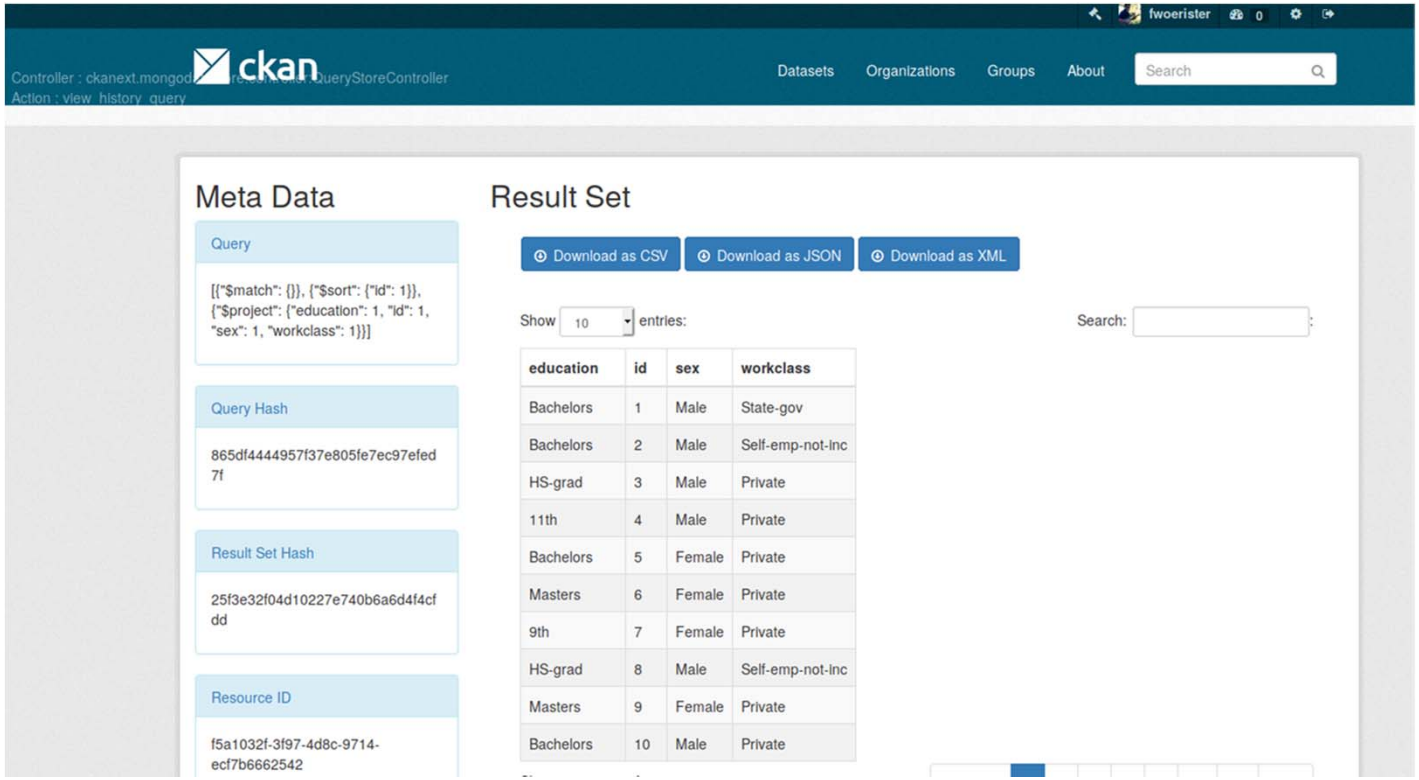
Screenshots



Screenshots



Walk-Through



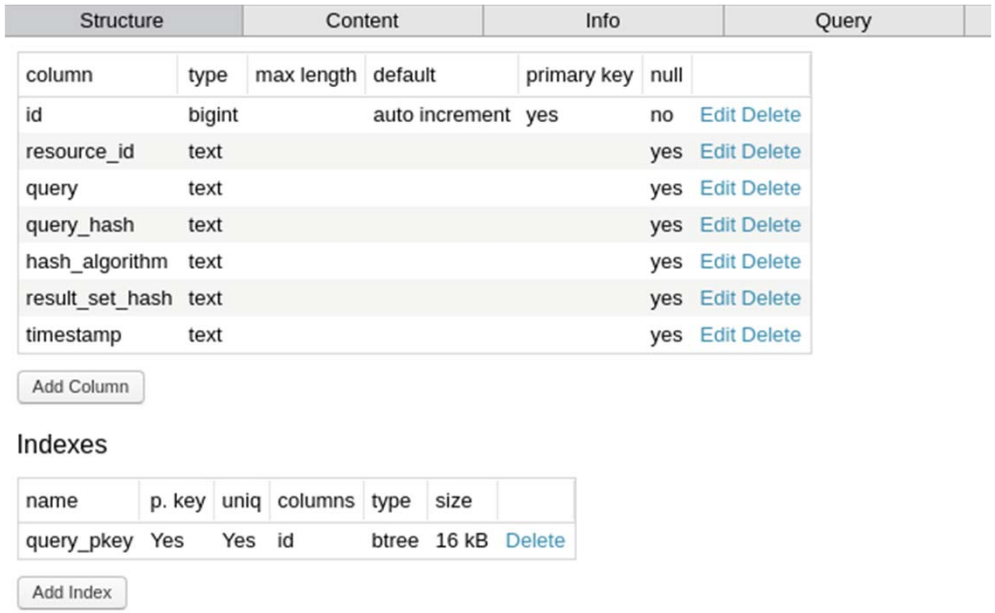
R1 - Data Versioning

- Every collection has an id field and the timestamp, when it was added
- In case of an **update**, the record is not modified, but the new version is added to the collection
- In case of an **deletion**, a document is added, that marks a certain id as deleted

R2 - Timestamping

- When inserting a document to a MongoDB collection an ObjectID is issued, which contains a timestamp
- ***Notice**: In general it is bad practice to add semantics to an identifier, that is why in future, a separate timestamp will be used for keeping record of the time of creation!*

R3 - Query Store Facilities



Structure | Content | Info | Query

column	type	max length	default	primary key	null	
id	bigint		auto increment	yes	no	Edit Delete
resource_id	text				yes	Edit Delete
query	text				yes	Edit Delete
query_hash	text				yes	Edit Delete
hash_algorithm	text				yes	Edit Delete
result_set_hash	text				yes	Edit Delete
timestamp	text				yes	Edit Delete

Add Column

Indexes

name	p. key	uniq	columns	type	size	
query_pkey	Yes	Yes	id	btree	16 kB	Delete

Add Index

- The information for query re-execution is stored in a PostgreDB
- The plugin accesses this database via an OR-Mapper (*SQLAlchemy*)

R4 - Query Uniqueness

Queries are normalized (fields are sorted by alphabet) and then an md5 hash is calculated!

Structure	Content	Info	Query

column	type	max length	default	primary key	null	
id	bigint		auto increment	yes	no	Edit Delete
resource_id	text				yes	Edit Delete
query	text				yes	Edit Delete
query_hash	text				yes	Edit Delete
hash_algorithm	text				yes	Edit Delete
result_set_hash	text				yes	Edit Delete
timestamp	text				yes	Edit Delete

Add Column

Indexes

name	p. key	uniq	columns	type	size	
query_pkey	Yes	Yes	id	btree	16 kB	Delete

Add Index

R5 - Stable Sorting

- Every query is automatically sorted by id prior to user-defined sorts

R6 - Result Set Verification

Based on an md5 hash

Structure	Content	Info	Query	

column	type	max length	default	primary key	null	
id	bigint		auto increment	yes	no	Edit Delete
resource_id	text				yes	Edit Delete
query	text				yes	Edit Delete
query_hash	text				yes	Edit Delete
hash_algorithm	text				yes	Edit Delete
result_set_hash	text				yes	Edit Delete
timestamp	text				yes	Edit Delete

Add Column

Indexes

name	p. key	uniq	columns	type	size	
query_pkey	Yes	Yes	id	btree	16 kB	Delete

Add Index

R7 - Query Timestamping

Structure	Content	Info	Query

column	type	max length	default	primary key	null	
id	bigint		auto increment	yes	no	Edit Delete
resource_id	text				yes	Edit Delete
query	text				yes	Edit Delete
query_hash	text				yes	Edit Delete
hash_algorithm	text				yes	Edit Delete
result_set_hash	text				yes	Edit Delete
timestamp	text				yes	Edit Delete

Add Column

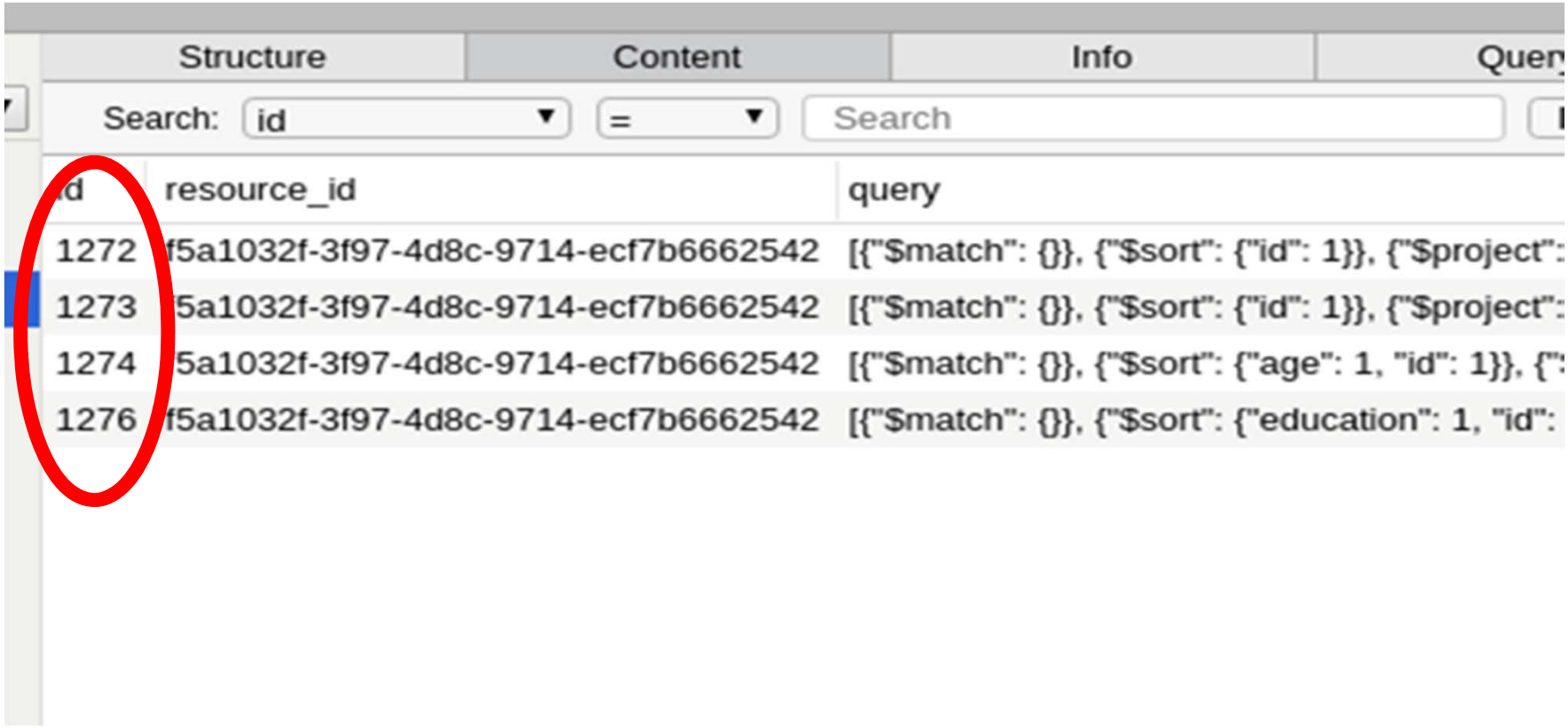
Indexes

name	p. key	uniq	columns	type	size	
query_pkey	Yes	Yes	id	btree	16 kB	Delete

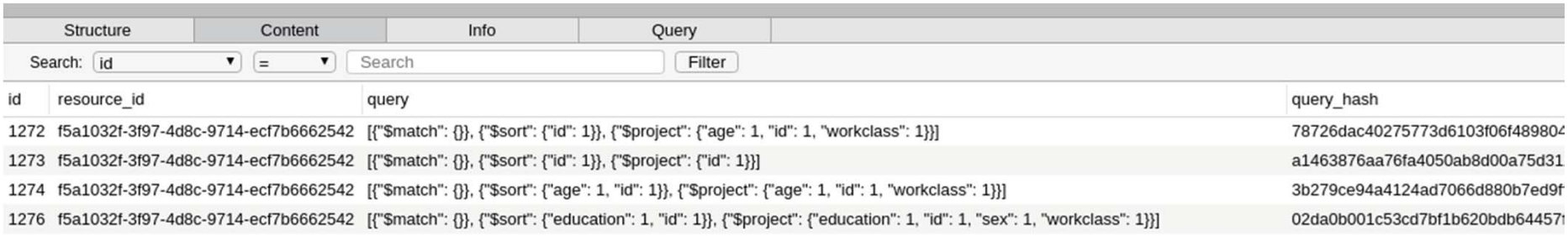
Add Index

R8 - Query PID

→ As PID, the id value of the query table is used



R9 - Store the Query

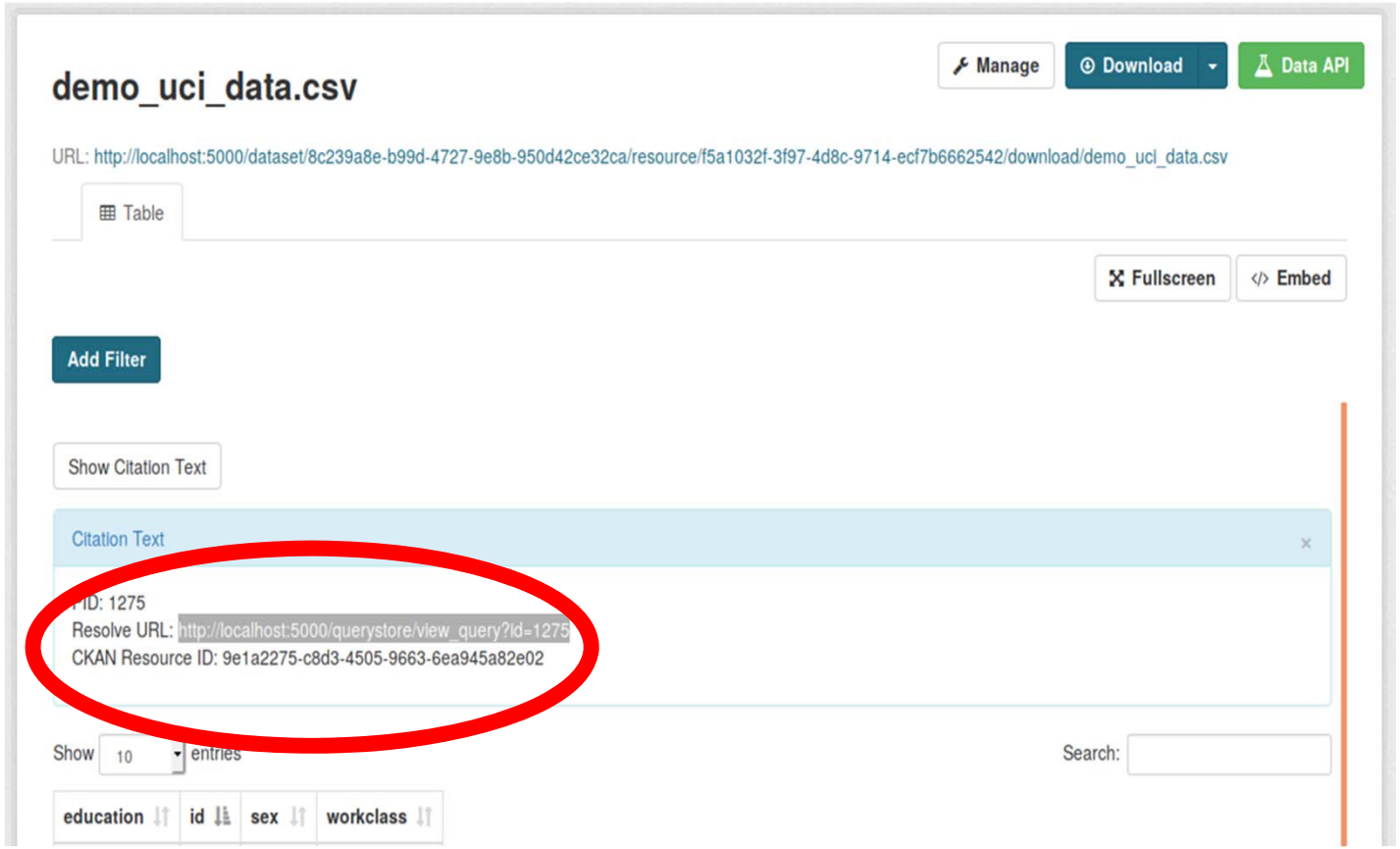


Structure | Content | Info | Query

Search: id = Search Filter

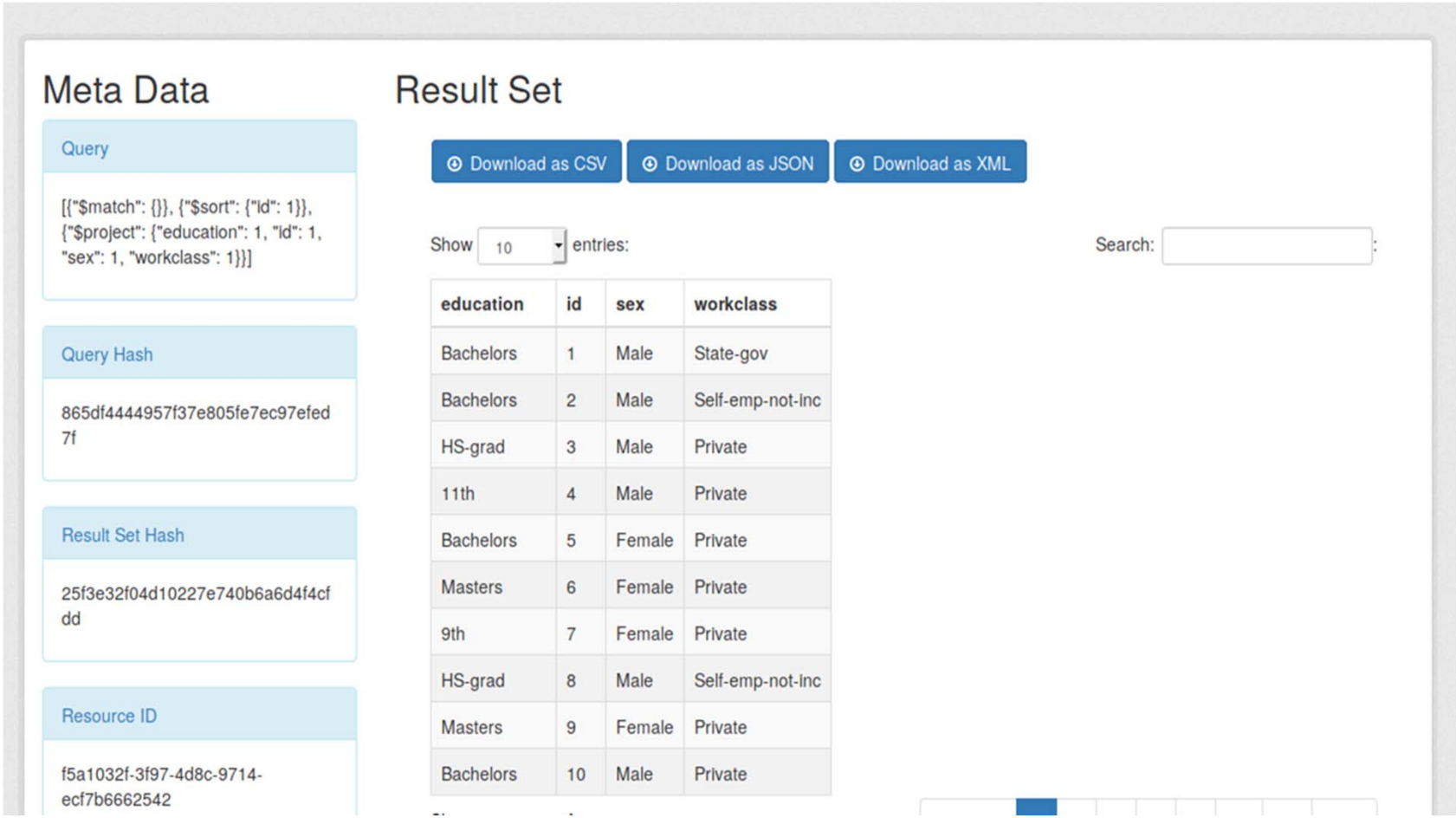
id	resource_id	query	query_hash
1272	f5a1032f-3f97-4d8c-9714-ecf7b6662542	[{"$match": {}}, {"$sort": {"id": 1}}, {"$project": {"age": 1, "id": 1, "workclass": 1}}]	78726dac40275773d6103f06f489804
1273	f5a1032f-3f97-4d8c-9714-ecf7b6662542	[{"$match": {}}, {"$sort": {"id": 1}}, {"$project": {"id": 1}}]	a1463876aa76fa4050ab8d00a75d31
1274	f5a1032f-3f97-4d8c-9714-ecf7b6662542	[{"$match": {}}, {"$sort": {"age": 1, "id": 1}}, {"$project": {"age": 1, "id": 1, "workclass": 1}}]	3b279ce94a4124ad7066d880b7ed9f
1276	f5a1032f-3f97-4d8c-9714-ecf7b6662542	[{"$match": {}}, {"$sort": {"education": 1, "id": 1}}, {"$project": {"education": 1, "id": 1, "sex": 1, "workclass": 1}}]	02da0b001c53cd7bf1b620bdb64457

R10 - Automated Citation Texts



TODO: *Add Author, Dataset Name, Year, etc. to the generated Citation Text!*

R11 - Landing Page



R12 - Machine Actionability

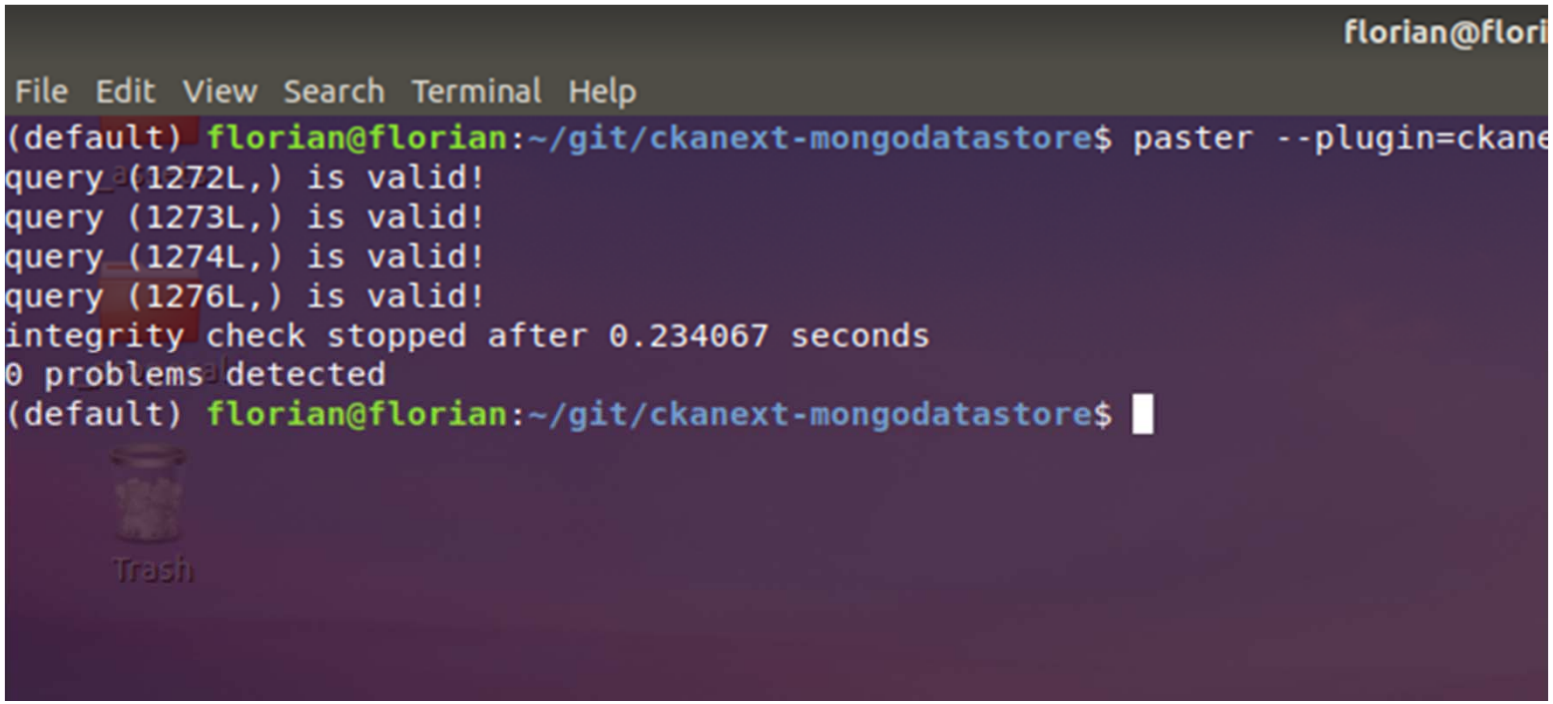
- The plugin provides a *retrieve_stored_query* method, that can be called via a REST interface.
- Exposes the result set + meta information (fields of the *query* table)

R13 - Technology Migration

*This recommendation addresses a more organisational issue. The plugin can handle this migrations, as after any technology change it is possible to validate the whole query store via the **CLI** (see R14!)*

R14 - Migration Verification

There is a paster command, that verifies every query in the query store!



GitHub

GitHub

Plug-in available on GitHub:

https://github.com/fwoerister/ckanext-mongodatastore

Implementation of WGDC Recommendations in Dendro System

João da Silva

research data sharing without barriers

rd-alliance.org

Scalable Citation of Subsets of Dynamic Data

An Europe Adoption Grant application
Based on the 14 Recommendations provided by the RDA Data Citation WG

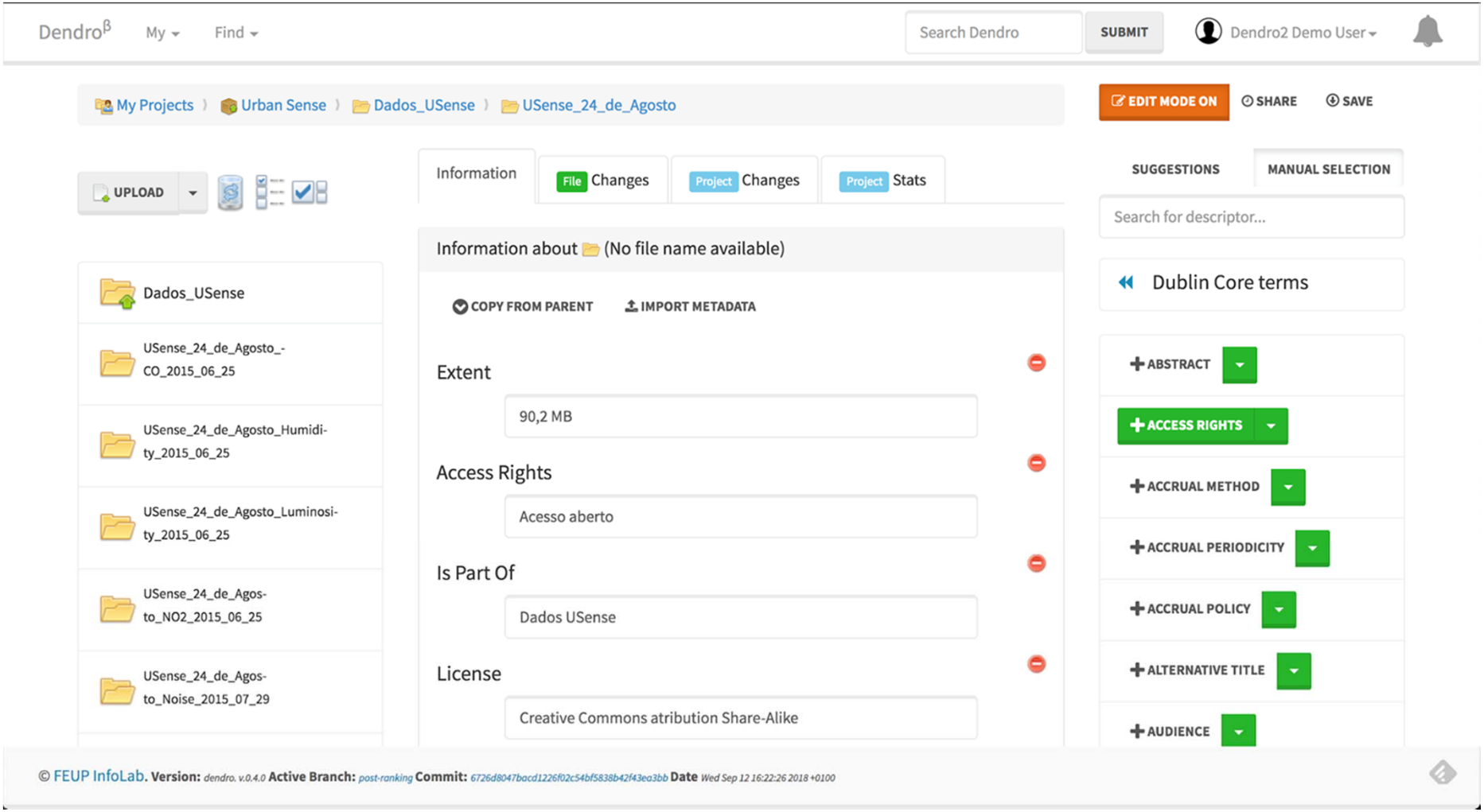
William Fukunaga
(wnfukunaga@gmail.com)
João Rocha da Silva
(joaorosilva@gmail.com)

INESC TEC / Faculdade de Engenharia da Universidade do Porto

Dendro

- Open-source “Dropbox” + Repository for research data
  - Fully developed by INESC TEC and University of Porto
- Domain-specific metadata via extensible graph data model
- Data visibility / access control
- Integrates with any repository (CKAN, ePrints, DSpace…)

https://github.com/feup-infolab/dendro


Dendro β
My
Find
Search Dendro
SUBMIT
Dendro2 Demo User
My Projects
Urban Sense
Dados_USense
USense_24_de_Agosto
EDIT MODE ON
SHARE
SAVE
UPLOAD
Information
File Changes
Project Changes
Project Stats
SUGGESTIONS
MANUAL SELECTION
Search for descriptor...
Information about (No file name available)
COPY FROM PARENT
IMPORT METADATA
Dados_USense
USense_24_de_Agosto_-CO_2015_06_25
USense_24_de_Agosto_Humidity_2015_06_25
USense_24_de_Agosto_Luminosity_2015_06_25
USense_24_de_Agosto_NO2_2015_06_25
USense_24_de_Agosto_Noise_2015_07_29
Extent
90,2 MB
Access Rights
Acesso aberto
Is Part Of
Dados USense
License
Creative Commons atribution Share-Alike
Dublin Core terms
ABSTRACT
ACCESS RIGHTS
ACCRUAL METHOD
ACCRUAL PERIODICITY
ACCRUAL POLICY
ALTERNATIVE TITLE
AUDIENCE
© FEUP InfoLab. Version: dendro. v.0.4.0 Active Branch: post-ranking Commit: 6726d8047bacd1226f02c54bf5838b42f43ea3bb Date Wed Sep 12 16:22:26 2018 +0100

- Microservice architecture **JUST ADDED (1.0-beta)**
  - OS/infrastructure independent, easy setup & backups
  - Dendro and **all dependencies** powered by **Docker containers**
  - **Scale** and **plug in** existing software
  - Machine learning for **automatic metadata production**

- Repository **BETA (2.0-beta)**
  - **DataCite PID + citation snippet generation** for datasets
  - **Custom Visibility** Embargoed/Custom access
  - **Faceted** Dataset Search

Problem

- Contents of CSV, XLS, TSV files are extracted and queryable via MongoDB, but **not versioned**
- **We need to make subsets of versioned data citable and their contents verifiable**
  - **RDA Data Citation WG Recommendations**

Goals

- Comply with 14 RDA Data Citation WG Recommendations
- Handle large datasets and their versions
- HTTP interface with fully documented REST API
- Publish as a Docker image: scalability through Docker Swarm
- Streamed I/O: retrieve and process arbitrarily large datasets
- Integrate with any platform, not only Dendro

Current data model

Row: {

creator:	String,
date:	Date,
resourceId:	String,
tag:	String,
hidden:	Boolean,
sheetName:	String,
sheetIndex:	Number,
rowIndex:	Number,
rowContent:	[]

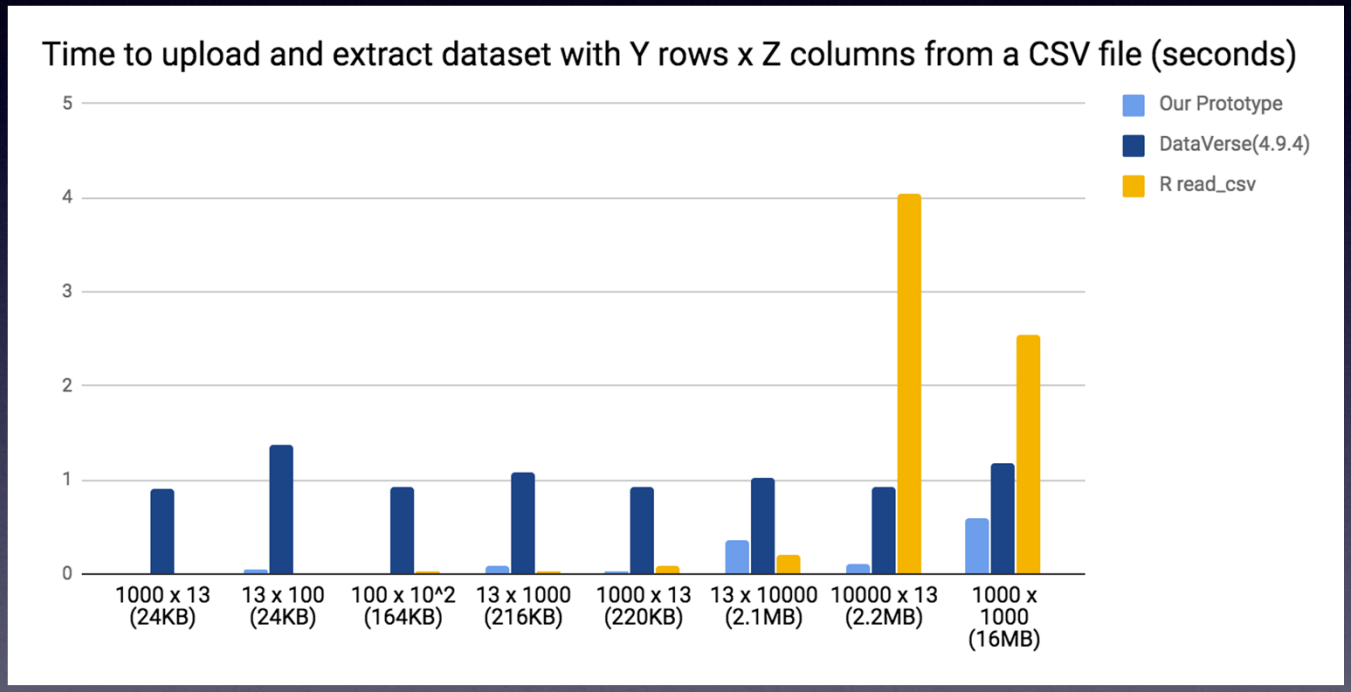
}

Query: {

hashResult:	String,
pid:	String,
queryText:	String,
date:	Date,
creator:	String,

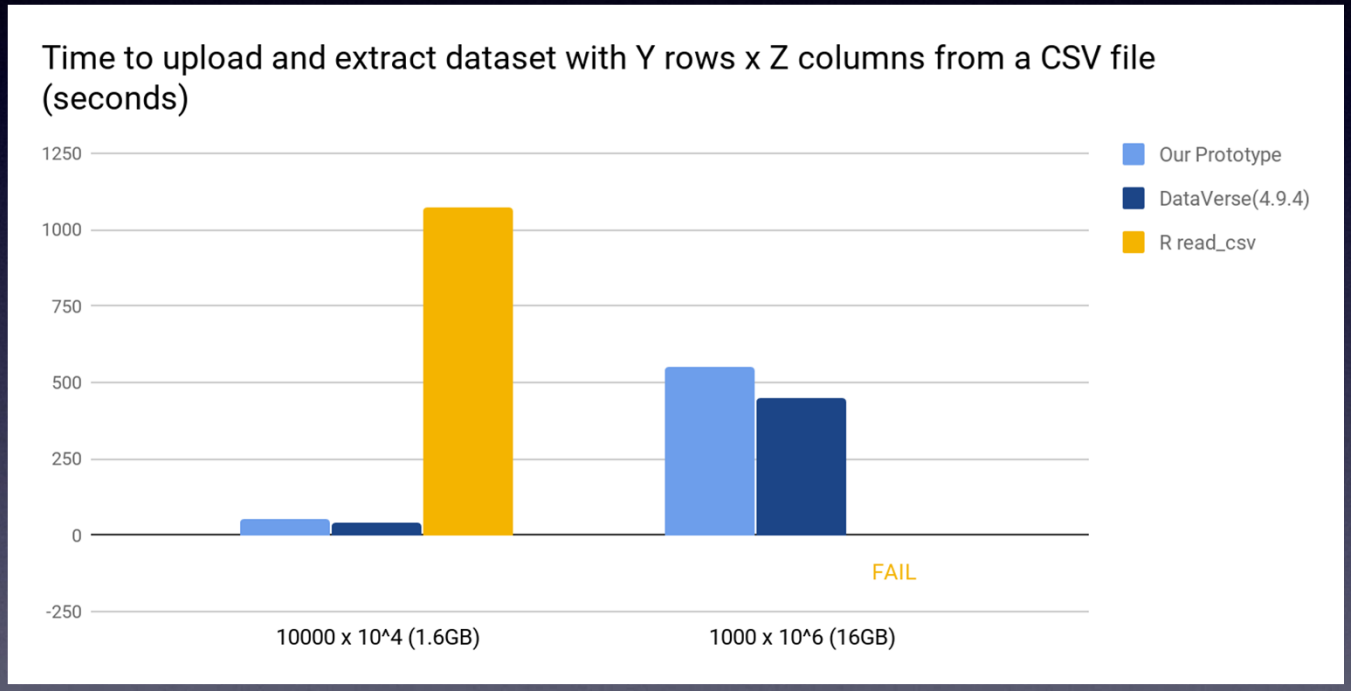
}

A quick experiment...



Both HTTP servers running inside Docker containers on the same hardware. DataVerse selected for its versioning features. Preliminary results.

Now some larger data!



Both HTTP servers running inside Docker containers on the same hardware. DataVerse selected for its versioning features. Preliminary results.

Current Status

Requirement	Compliant?
Data Versioning	**Yes**
Timestamping	**Yes**
Query Store Facilities	**No**
Query Uniqueness	**No**
Stable Sorting	**Yes**
Result Set Verification	**No**
Query Timestamping	**No (yes?)**
Query PID	**No (internal GUIDs only)**
Store the Query	**No**
Automated Citation Texts	**No (DataCite - yes)**
Landing Page	**No**
Machine Actionability	**Yes**
Technology Migration	**No**
Migration Verification	**No**

RDA Europe Adoption Grant proposal

GOAL Satisfy all recommendations (currently 4/14)

GOAL Integrate with Dendro and CKAN

GOAL Compare performance vs. CKAN DataStore and DataVerse

OpenEO

Tomasz Miksa, Bernhard Gößwein

research data sharing without barriers
rd-alliance.org

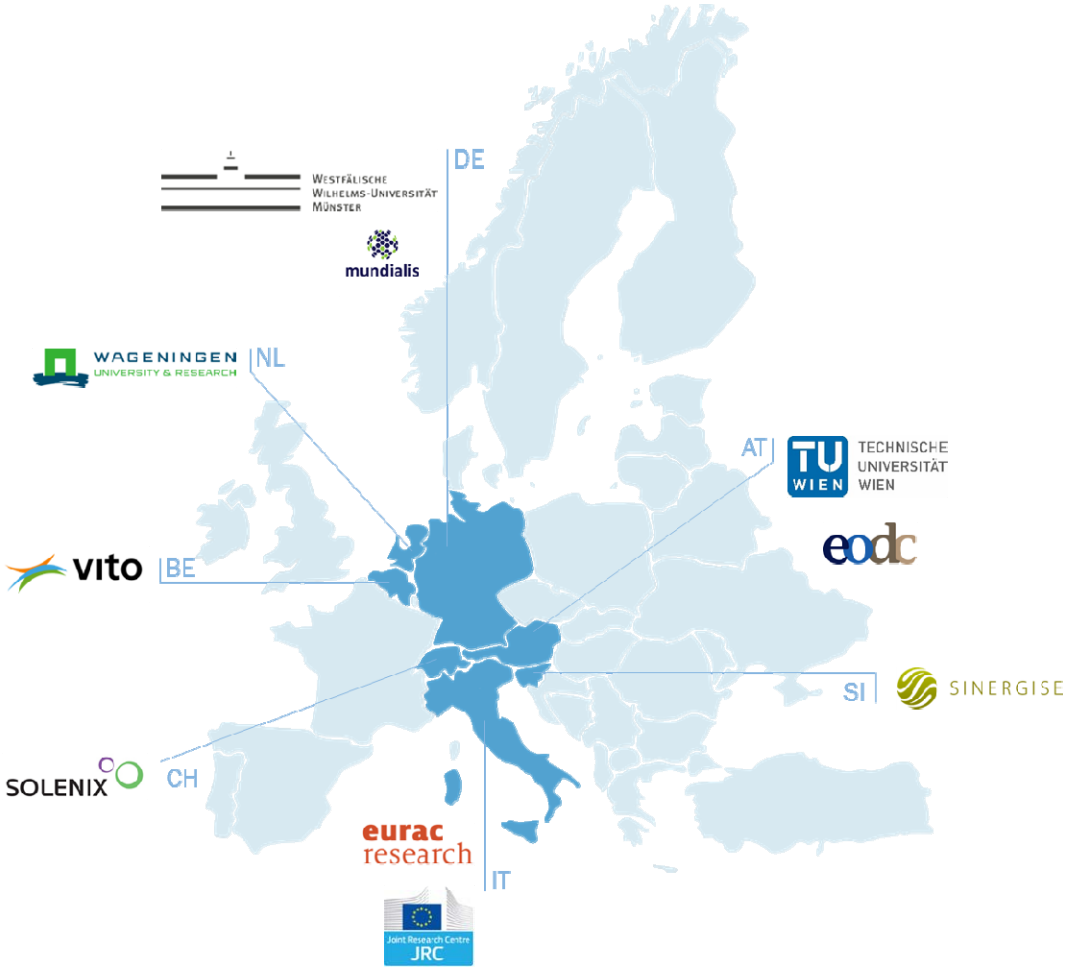
OpenEO Product Reproducibility

Grant agreement No 776242 | 4/3/2019

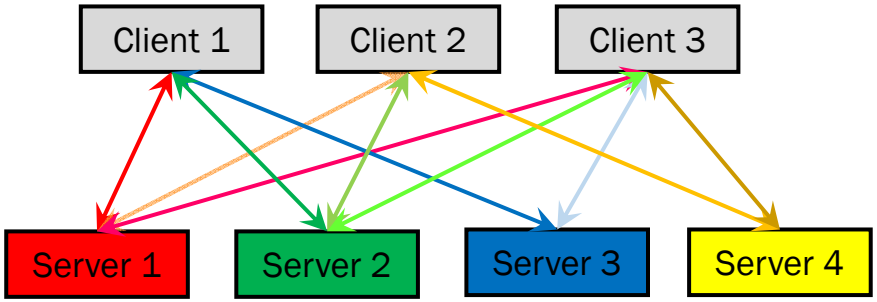
Insert title here

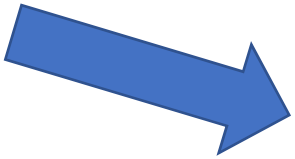
OpenEO Overview

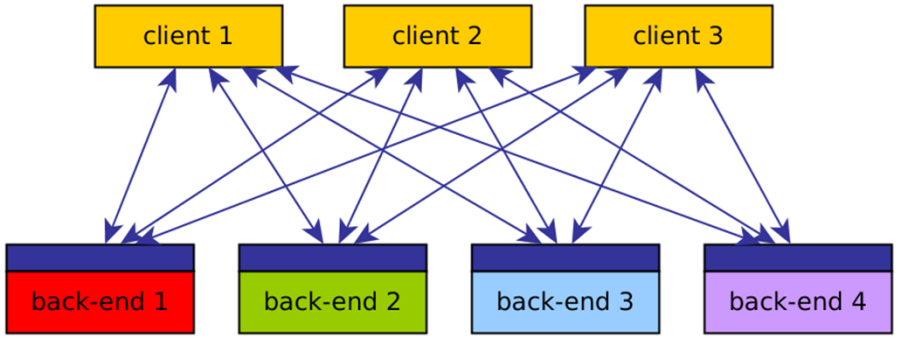
- Earth Observation
- Data
  - Too big for local processing
  - Code visits data
- Back-end operators
- Goal
  - Develop common API



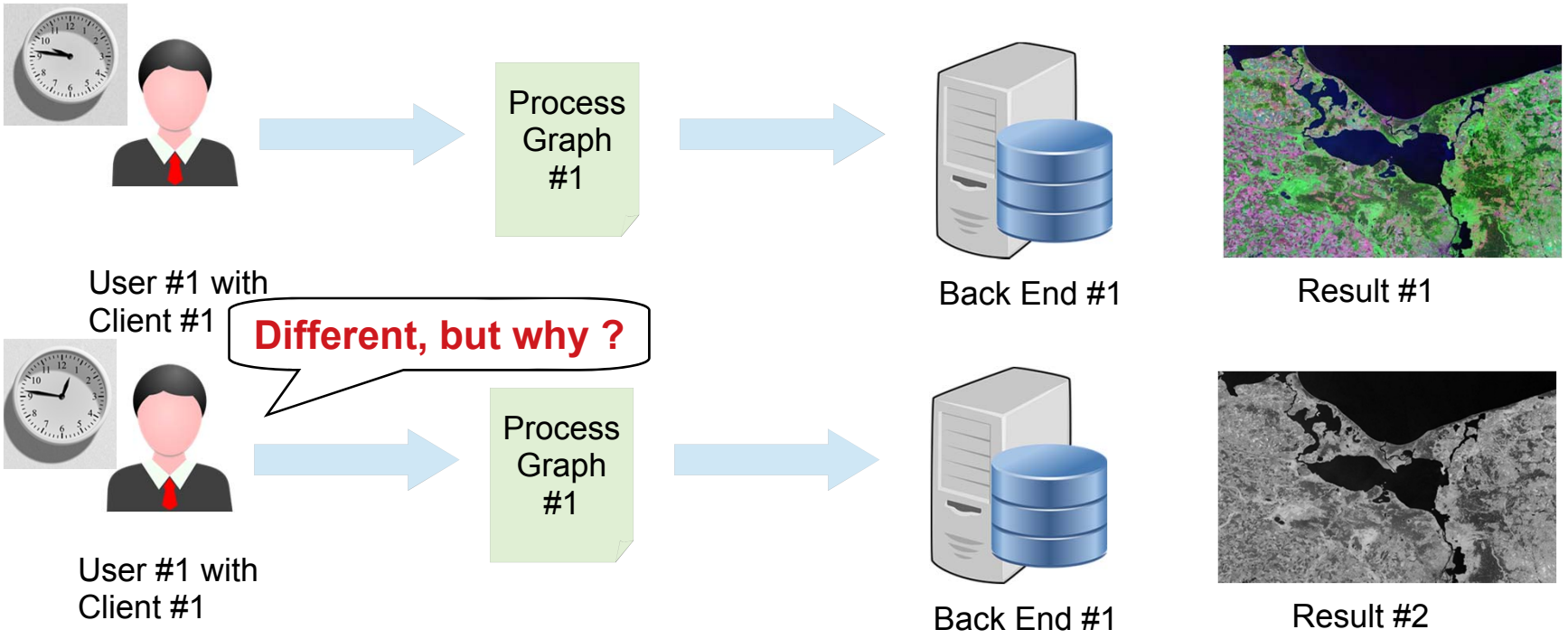
OpenEO Overview







Data Citation – why?



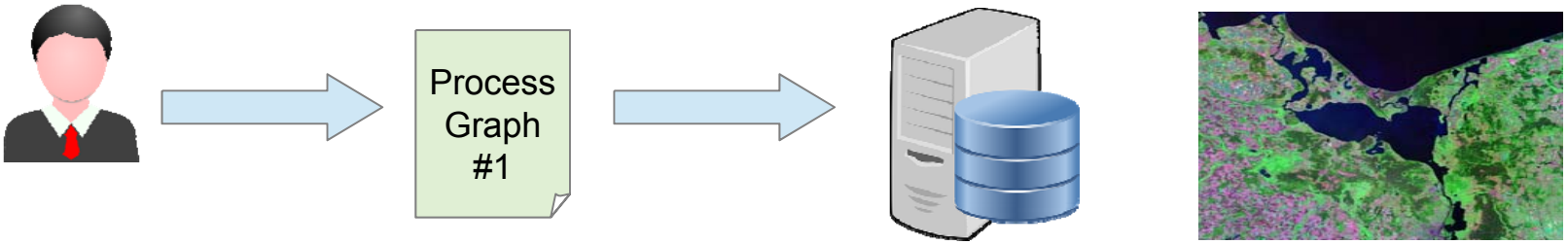
OpenEO: Overview

- OpenEO Client (e.g. Python, R, …)
  - The tooling the openEO user uses to connect to a back end and to define a source data and process chain (process graph) that gets executed at the back end.
    3 Clients planned: Python, R and Javascript.
- OpenEO coreAPI (Specification)
  - The definition how the clients and back ends communicate including guidelines how to implement it. Communication protocol definition using JSON and RESTful Webservices.
- OpenEO Back end (e.g. EODC, Google Earth Engine)
  - Host of source data and processing framework. Every back end has a back end driver layer handling openEO requests and translating them to the own processing framework.
  - 8 back ends planned: EODC, Google Earth Engine, R Backend, EURAC, Mundialis, Sinergise, VITO and JRC.

EODC: Overview

- Back end of the OpenEO Project
- Processing:
  - Uses python code for processing
  - Uses Docker container to run the python code
  - Uses OpenShift to manage the Docker containers
- Every process of the Process Graph is represented by a piece of python code.
- So the process graph gets transformed to a docker container including the python code of every process.

Problem



User #1 with
Client #1

Back End #1

Result #1

- User defines a process graph using the client library and sends it to the back end of his choice.
- Back end calculates result and returns it with a download link for the user.

Solution: Back-End Context Model

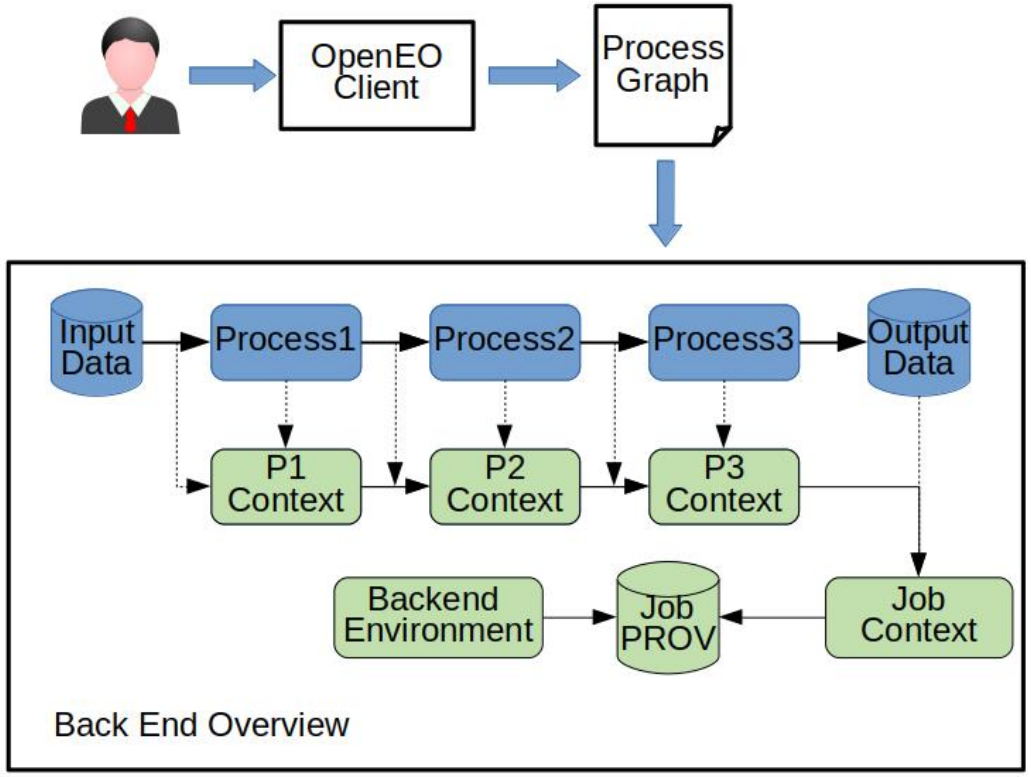
- Standalone tool to automatically capture changes on the back end settings
- **Captured Data:**
  - GitHub Repo (OpenSource Project, the code of the back-ends are on GitHub)
  - Changes of (Git independent) used folders, excluding temporary folders.
  - Operating System and packages installed.
  - Supported coreAPI version.

Solution: Data Citation

- **Current situation at OpenEO**
  - The source data gets versioned through different filenames. Persistence of older versions are not guaranteed. (R1)
  - Versions of the source data is not done by every back end provider, the reason for this is the big data amount that is too costly to be persisted in several versions.
- **Our suggestions to OpenEO Backends**
  - R1: Persistence of old data versions have to be applied.
  - R2: Timestamps of data changes have to be persisted on source data changes.
  - R3-9: A Query Store needs to be introduced to the back ends. Most back ends already use an OGC standard for querying data.
  - R10-R14: Need to be added into the core API to have a common standard for all back ends.

Solution: Processing Context Model

- **Process Context:**
  - Input Data
  - Process Code
  - Output Data
  - Timestamps begin/end
  - Parameters
  - Execution Environment



Solution: Processing Context Model

- Captures the back end context model at the time of the execution
- There are two levels of detail planned for the capturing of the processes:
  - Capture the result between every process step (optional)
  - Capture the result of the whole process chain
- Depends of the back end, what is possible to implement, but the capturing of the whole process chain is at least doable on every back end.

What will be developed: Process Capturing

Definition of the context models for OpenEO

- Implementation of the processing Context Model on the EODC Back end (in Python).
- Implementation of the back end Context Model capturing on the EODC Back end (in Python/Linux)
- Implementation of the data citation tools on the EODC Back end (in Python/Linux)
- Changes to the Python Client to make the captured data available to the User and to let the user compare two different jobs.
- Define suggestions to the OpenEO core API (e.g. a possibility to choose the data version).

EODC Data Citation

- **Initial situation at EODC**
  - File based data management: Every source data file has a unique path. Updates on the data result in a new filename.
  - There is no guaranteed persistence of deprecated data objects. These are kept and can be retrieved again from the source data store (ESA)
  - Creation time stamps of the files are persisted in a meta-database.
  - Querying the data happens through a Web API using the OGC standard CSW (see https://csw.eodc.eu)
  - Queries and Query results are not persisted

EODC Data Citation

- **RDA Recommendations at the EODC back end**
  - **R1:** Persistence of old data versions not at EODC (size), but versions tracked for data identification, old versions available at ESA.
  - **R2:** Timestamps are already available at the existing meta-database of the EODC back end.
  - **R3:** A Query Store is implemented as an additional table in the relational database (PostgreSQL).
  - **R4:** Query is defined by the filter processes of the process graph, these are captured by the EODC back end and stored in a JSON object. Sorted alphabetically to be used as the unique query.
  - **R5:** Stable sorting is assured by the CSW query, where the resulting file list is sorted in ascending order (alphabetically).
  - **R6:** The resulting file list is given as a list object sorted alphabetically in ascending order. This is transferred to a string object, cleaned of irrelevant characters, fed into SHA-256 hash function.

EODC Data Citation

- **RDA Recommendations at the EODC back end**
  - **R7:** The timestamp of the query execution is stored
  - **R8:** Query PID is created (using Python UUID), if the same unique query and resulting hash combination is not in the database yet
  - **R9:** Query Store is implemented as an additional table in the relational data base (PostgreSQL). The original query is the original CSW query in XML format. The checksum of the unique query is implemented by a SHA-256 hash of the unique query. The data-set description is represented by the data-set identifier used at the EODC back end (e.g. Sentinel-2A). There is one column added to the query store to store additional information in a JSON object (e.g. number of resulting files).
  - **R10:** Citation text for the data-set is already available at EODC, the generated data PID is added to it, as well as the information that it was retrieved at the EODC back end.

EODC Data Citation

- **RDA Recommendations at the EODC back end**
  - **R11 & R12:** Landing page at the EODC back end defined as OpenEO Endpoint. Publicly accessible, JSON format. Contains link to re-execute the query and list the result files. Since it is part of the OpenEO API, it can be accessed from OpenEO clients and be used in future jobs as input data.
  - **R13 & R14:** These recommendations are not implemented at the moment, since there are no migrations. There are unit tests written in the OpenEO Python client testing the basic functionality of the data identification of the back end.

EODC Data Citation

```
import openeo
#connect to back end
con = openeo.connect(EODC_DRIVER_URL)
# Choose dataset
processes = con.get_processes()
pgA = processes.get_collection(name="s2a_prd_msil1c")
pgA = processes.filter_daterange(pgA, extent=["2017-05-01", "2017-05-31"])
pgA = processes.filter_bbox(pgA, west=10.288696, south=45.935871,
east=12.189331, north=46.905246,crs="EPSG:4326")
# Choose processes
pgA = processes.ndvi(pgA, nir="B08", red="B04")
pgA = processes.min_time(pgA)
# Create and start job at the back end
# This generates the job context model and the input data (query) PID
jobA = con.create_job(pgA.graph)
jobA.start_job()
# Returns resolveable Query PID e.g. EODC_DRIVER_URL/collections/qu-d1701f4e-
e7c5-4a83-92e0-9facbd401a06
pidA = jobA.get_data_pid_url()
# Re-executes the query and returns the resulting file list.
file_list = con.get_filelist(pidA)
# Reusing the data PID with a different workflow
pgB = processes.get_collection(data_pid=pidA)
# Choose processes for the new workflow
pgB = processes.ndvi(pgB, nir="B08", red="B04")
pgB = processes.max_time(pgB)
# ...
```

RESEARCH DATA ALLIANCE

Others?
Plans, On-going, Feedback

Anybody

research data sharing without barriers

rd-alliance.org

Adoption Stories

- Let us know if you are (planning to) implement (part of) the recommendations

- Submit your adoption story to the RDA Webpage:

https://www.rd-alliance.org/recommendations-outputs/adoption-stories

Agenda

- 12:00 Introduction, Welcome
- 12:10 Short description of the WG recommendations
- 12:30 Reports by adopters / pilots
- 13:00 Review of Recommendations text / lessons learned
- 13:20 Other issues, next steps

R1: Data Versioning

- **R1 – Data Versioning**: Apply versioning to ensure earlier states of data sets can be retrieved.
- Most common issues:
  - Audit files, transaction logs are not sufficient as roll-back is too costly to allow recreating an earlier state
  - Semantic versioning does not make sense with data (SW: changes that do not break APIs, purely syntactic, not semantic)
  - Granularity of versions often mentioned, but never encountered in any actual pilot (microsec-updates)
  - Size: if versions cannot be kept – then one cannot go back to the respective state of data
  - Principles still ok with higher-granularity versioning, i.e. “stable” versions when needed (suboptimal, but necessary in specific settings)
  - Legally requested deletions must, of course, be physically executed → such states of data can no longer be re-created

R1: Data Versioning (cont.)

Semantic Versioning

- Semantic versions are "only" **assertions on states of the data at certain points in time**, eg
  - Data may be transient / still undergoing changes, whereas after a certain points in time it has reached a state where no further changes are expected
  - Certain states of data may not be intended for permanent retention, whereas others may have guarantees of availability over time
- Assertions specified as tags associated to queries, e.g.
  - Query *"Select * FROM <table> WHERE timestamp_added < ts1 and ts_deleted >ts1"* may carry the assertions *"status: not expected to change"* and *"availability: 7 years" (preferably from controlled vocabularies)*
- *Subset queries are "nested queries" on such "stable versions"*

R2: Timestamping

- **R2 – Timestamping**: Ensure that operations on data are timestamped, i.e. any additions, deletions are marked with a timestamp.
- Most common issues:
  - R1 & R2 need to be addressed together
  - Should actually be the first recommendation:
    - R1: Timestamp all write operations on data
    - R2: Ensure that no write operation overwrites/deletes earlier states of data that one needs to get back to

R3: Query Store Facilities

- **Query Store Facilities**: Provide means for storing queries and the associated metadata in order to re-execute them in the future.
- Most common issues:
  - Actually an operational recommendation: set up a query store
  - Not at the same level as the previous two recommendations
  - However, R1 – R3 are “compulsory” while most of the subsequent ones (except for R7 and R9) are optional
  - Query store is perceived to get huge (not encountered so far) (to be addressed in R9: staging area / temporary storage, not persisting ALL queries forever)
  - Some suggest that the user should store the queries -> dangerous, as any data schema migration would require the data center to always maintain dynamic query migration services (user cannot migrate to new data representations)

R4: Query Uniqueness

- **R4 – Query Uniqueness**: Re-write the query to a normalised form so that identical queries can be detected. Compute a checksum of the normalized query to efficiently detect identical queries.
- Most common issues:
  - Usually less of an issue as most centers support structured access to data via APIs and "query builders" / faceted browsing → most queries come in standardized form anyway, little to no re-writing necessary
  - Optional, worst case: two semantically equivalent queries get different PID
  - May need guidance on creating the hash input string for the checksum computation (e.g. removing spaces, CR/LF, …) (but: standard for checksum computation in many settings)

R5: Stable Sorting

- **R5 – Stable Sorting**: Ensure that the sorting of the records in the data set is unambiguous and reproducible.
- Most common issues:
  - Optional, only needed if sequence of results is important
  - May need further guidance (e.g. sorting by primary key)
  - Causing few issues or concerns so far

R6: Result Set verification

- **R6 – Result Set Verification**: Compute fixity information (checksum) of the query result set to enable verification of the correctness of a result upon re-execution.
- Most common issues:
  - Sometimes confusion on terminology (fixity information vs. checksum vs. hash key, all quasi-synonymous)
  - May require guidance on hash input string computation (removal of white spaces, more difficult for non-standard data that has internal structure)
  - May need additional guidance on how to do this for very large data sets (currently: equivalents to primary key column plus row headers)
  - Hardly any issues so far

R7: Query Timestamping

- **R7 – Query Timestamping**: Assign a timestamp to the query based on the last update to the entire database (or the last update to the selection of data affected by the query or the query execution time). This allows retrieving the data as it existed at the time a user issued a query.
- Most common issues:
  - In practice worries about overlaps between write updates and query execution -> requires locking or query execution with “available” timestamp one time delta before the current time, i.e. with according write operations completed.
  - Some groups still use semantic versioning or sequential numbers
    - Nothing wrong with numbers, but less generic and: no way to determine according state of system when only query execution time is known (as commonly used in references)
      → mapping sequence number to timestamp

R7: Query Timestamping – Semantic Versioning

Why timestamps, why not semantic versioning

- Some prefer to use semantic versioning (minor/major updates that do not / do change behaviour/interface)
  - Advantage: version number indicates relationship btw. versions
  - Disadvantage:
    - Something that was expected to be a not-changing update may turn out to induce changes / side-effects later-on
    - With data, “minor” updates are hard to think of: changing a typo may result in a record being found / not found by a query, encoding changes may break subsequent processing pipelines
    - Different semantics / types of use across different communities
- Recommendation
  - No semantics in identifier (mantra!)
  - Keep identification (version timestamp) and semantics separate
  - Semantic version number in addition to timestamp

R7: Query Timestamping – Distributed Settings

Distributed Setting

- No need for synchronized timestamps across nodes
- Each node keeps local time
- Solution with one central query store (master node):
  - Master node distributes queries
  - Distributed nodes return query result with local execution timestamp
  - Master stores timestamps per node where response received
- Solution with individual query stores
  - Distributed nodes store own query and timestamps, return their PIDs
  - Central/original query processing node stores query ids of distributed nodes
  - Central node only aggregator

R8: Query PID

- **R8 – Query PID**: Assign a new PID to the query if either the query is new or if the result set returned from an earlier identical query is different due to changes in the data. Otherwise, return the existing PID.
- Most common issues:
  - Questions on whether it has to be a DOI, or whether one could use a dual system, e.g. internally some PID and externally a DOI
  - Dual DOI assignment in strong demand: one for the <*Specific Subset*> of data, emerging from an evolving <*Data Source*> - mentioned in **R9** and documentation, but not explicitly in R8 -> change?
  - Theoretically all manageable by the machine-actionable landing page, but still seems relevant for direct impact accumulation
    (Note: danger of requesting 20 DOIs including the meta-data source, the data infrastructure, … and all forms of contributors…)

R9: Store Query

- **R9 – Store Query**: Store query and metadata (e.g. PID, original and normalized query, query & result set checksum, timestamp, superset PID, data set description, and other) in the query store.
- Most common issues:
  - Again a procedural rather than a conceptual recommendation
  - Sometimes concerns about “query” as basis
    (but: cf. filename is query on file system pulling together segments)
  - Only reference to *<Superset PID> - make 2 PID aspect more visible*
  - *Concerns on massive volume of queries to be stored*
    - In most cases: relatively small in size
    - Some pilots chose a “staging area” where they keep queries for a certain period of time (1-4 weeks) allowing researchers to select from their “shopping basket” which data sets they finally used → assig a PID only to these

R10: Automated Citation Texts

- **R10 – Automated Citation Texts:** Generate citation texts in the format prevalent in the designated community for lowering the barrier for citing the data. Include the PID into the citation text snippet.
- Most common issues:
  - The heading of this recommendation sounds odd (“automated”…)
  - No guidelines from our side on how to do this
  - In many cases reference to <*Superset PID*>
  - Sometimes reveals real concerns by data centers: fight about the actual purpose of a citation, from identification of the data to attribution of credit
  - Conflict between human and machines
  - Can be automated, even for complex cases (c.f. Webinar by Gianmaria Silvello)

R11: Landing Page

- **R11 – Landing Page**: Make the PIDs resolve to a human readable landing page that provides the data (via query re-execution) and metadata, including a link to the superset (PID of the data source) and citation text snippet.
- Most common issues:
  - Very few issues, seems accepted
  - Sometimes discussion whether landing page equals download page → to be solved elsewhere
  - Would – in combination with R12 - potentially solve all issues concerning the number of <super*>set PIDs and ORCIDs

R12: Machine Actionability

- **R12 – Machine Actionability**: Provide an API / machine actionable landing page to access metadata and data via query re-execution.
- Most common issues:
  - Essential! But not always fully implemented
    XML is not necessarily sufficient → common vocabularies are not yet always present → may need time to mature
  - May need explicit link to FAIR principles and FAIR metrics
  - Helps in attribution, credit sharing, etc.

R13: Technology Migration

- **R13 – Technology Migration**: When data is migrated to a new representation (e.g. new database system, a new schema or a completely different technology), migrate also the queries and associated fixity information.
- Most common issues:
  - Very little experience so far
  - Essential, as it makes clear why the query store needs to be at the data infrastructure
  - May be challenging wrt. fixity information – the migration of which is explicitly addressed – may need explanation to stress assertion that previously computed fixity information corresponds to this new fixity information (change in structure or change in fixity function)
  - May cause conceptual issues: if the data representation (formatting) changes, is it still the same subset of data (breaking API)

R14: Migration Verification

- **R14 – Migration Verification**: Verify successful data and query migration, ensuring that queries can be re-executed correctly.
- Most common issues:
  - Seems obvious, was added to avoid having 13 recommendations ☺
  - But: led to implementation of a few test-routines that can also be run continuously in a repository, testing correct recreation of old data sets in low-load time windows → good practice that should be shared!

R15?: Anything missing ???

- **Is there anything missing that should be added?**
- Start a process of collecting feedback from
  - Adopters on their experience
  - Potential adopters on their concerns and confusions
  - All others on whatever they consider worth addressing
- Note: Recommendations describe an “ideal world”
  - Reality requires pragmatism
  - Partial implementations of recommendations, variations
  - Provide feedback on issues, concerns, successes and failures

Next Steps

- Support in adoption: what kind of support is needed?
  (in the end it all boils down to money, but apart from this…)
  - Webinars: generic
  - Focused workshops for individual pilots
  - Joint projects: proposals, …
  - Further sessions at plenaries?
- Dissemination of information from on-going pilots
  - Structuring: contact, descriptions, results, lessons learned
  - Outcomes: reports, slides, publications, code, discussions
  - Summary paper on pilots
- New Webinars?
- Anything else? AOB? Wishes?

Thanks

Thanks!

And hope to see you at the next meeting

of the

WGDC