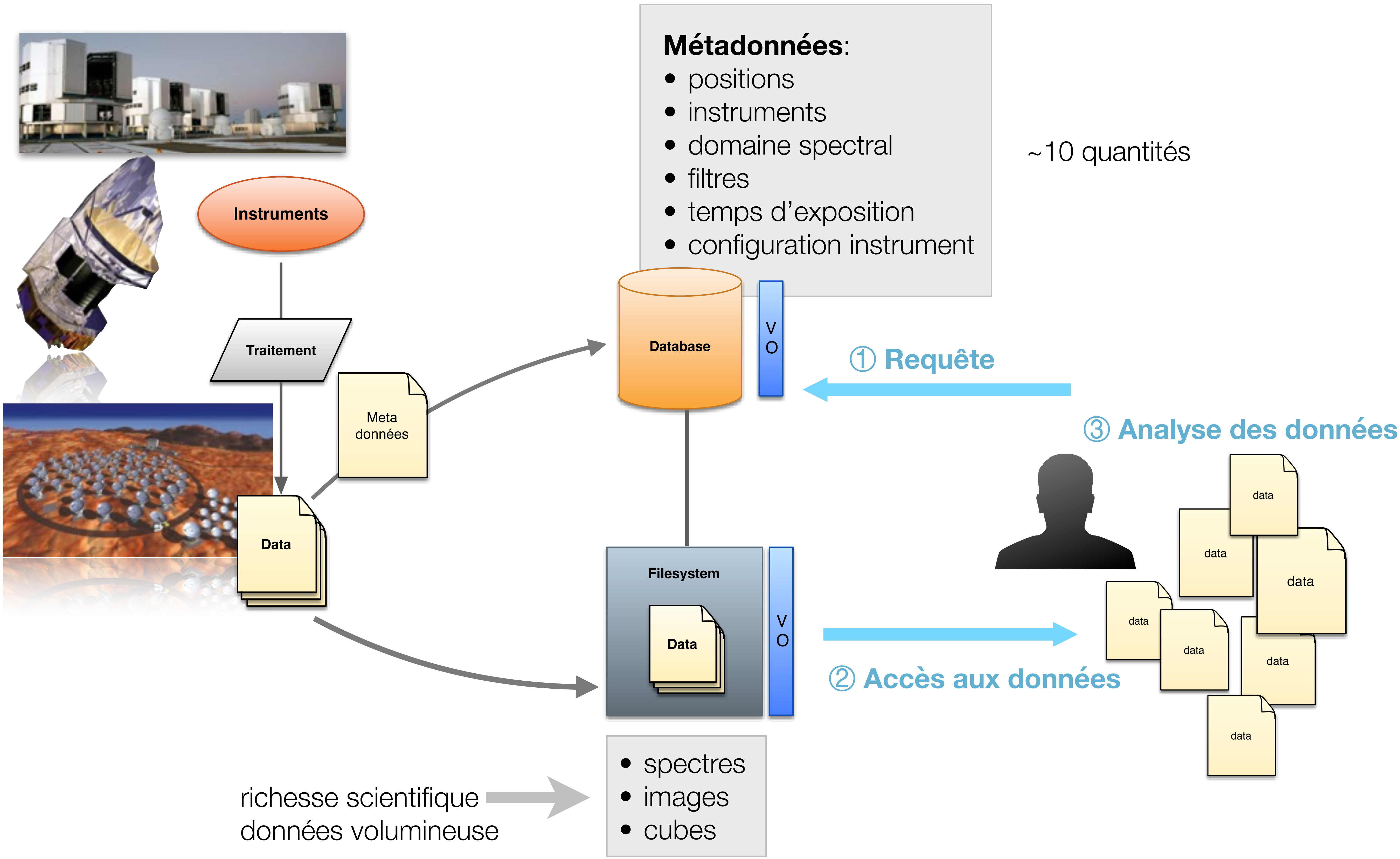
Usage of the “big” data

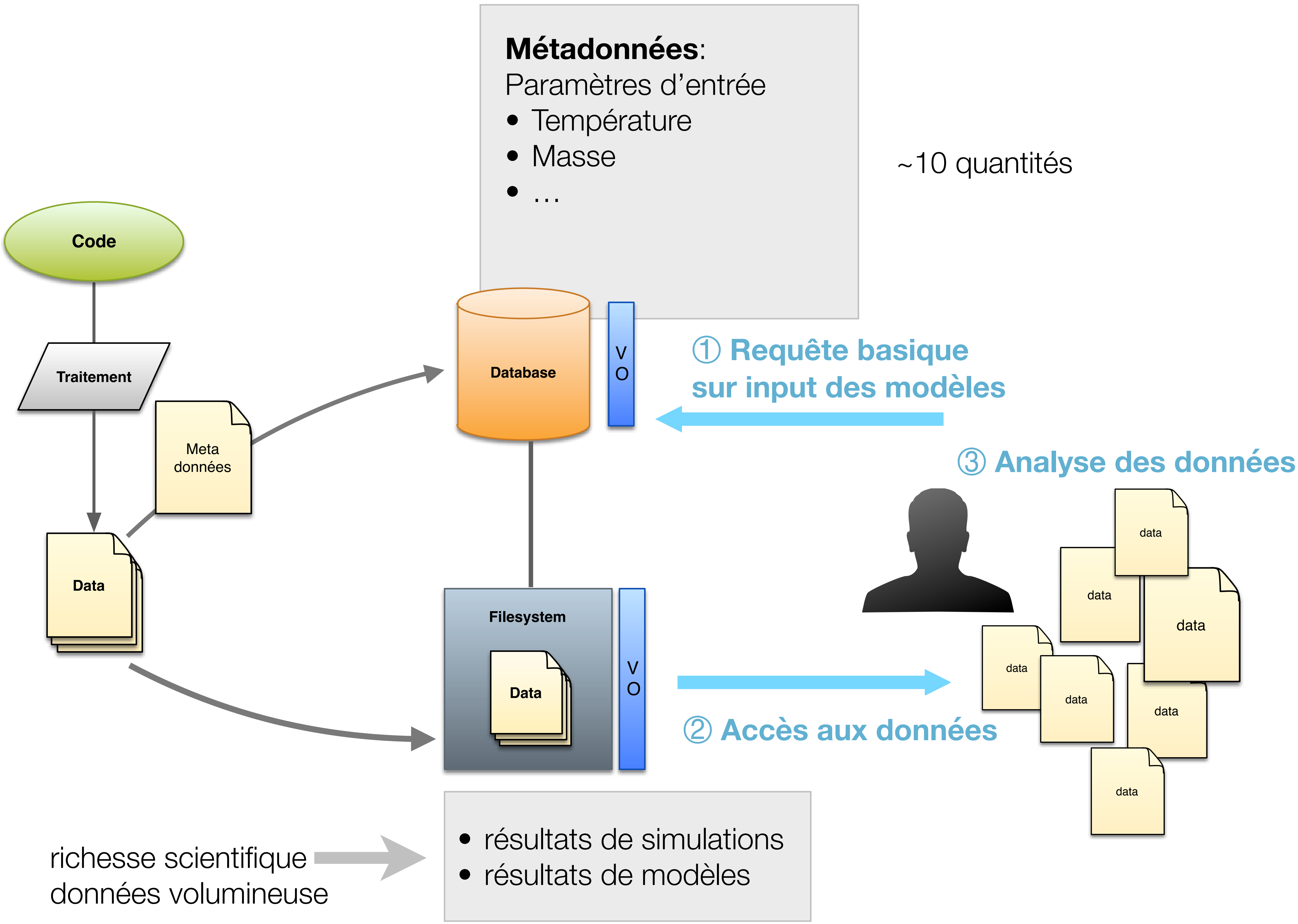
asov - paris - 2015.03.24

david.languignon@obspm.fr

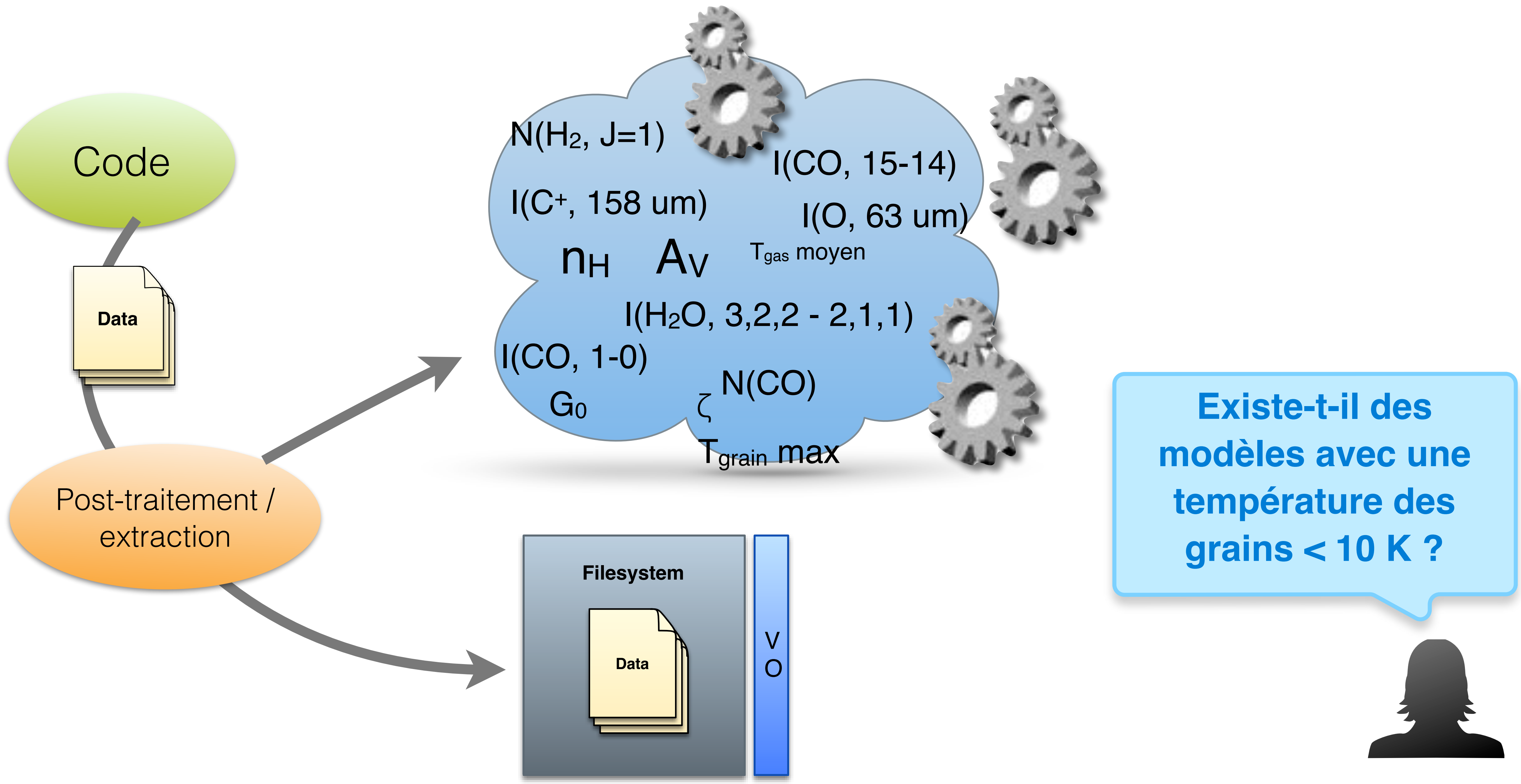
Astro data services: **Observations**



Astro data services: **Numerical simulations**



What we would like



- 10 **code input quantities** can be queried
- thousands of models
- tens of thousands of data

➔

- hundreds of thousands of **code output quantities** can be queried
- thousands of models
- millions of data

Technical challenges

- Simulated data very heterogeneous:
  - **dimension nature** (mass, line intensity, x, y, z, …)
  - **number of dimensions** ([10; 10^{5+}[)
  - **number of objects** (dm halos nbr < particles nbr < …)

- Human-computer interaction while manipulating large meta-data amounts

Technical challenges

few common dimensions

huge number of dimensions

huge number of objects

obj	A	B	C	D	E	F	G	…
1		[illegible]	23					
2	2E+12			3.12e2				
3					1.2	3E+12		
…								
10^{15+}								

sparse matrix

Technical challenges

Unfortunately, no one tech solution to rule them all

Handle a lot of columns

The current services are built on top of relational db / SQL
Ex: One of the most used VO standard: Table Access Protocol (TAP)

Problem: the number of columns a relational database can handle is limited

	Table Size	Number of col.	Nbr of rows	Col. name size
MySQL	64 Tb	4096	a lot	64
Postgress	32 TB	250 - 1600	unlimited	63
Oracle	4GB * block size	1000	unlimited	30
Microsoft	524272 TB	30 000	limited by storage	128

→ The classical relational db approach doesn't fit the number of columns, neither the data heterogeneity very well

How to manage a huge number of dimensions ?

Solutions ?

- noSQL & other new db designs
  - are the data more like documents than like table ?
  - do the db engine provide the management convenience we need ? (given the amount of data: clustering, memory setting etc…)
  - what logic must be moved to the application side when switching to schedules db ?

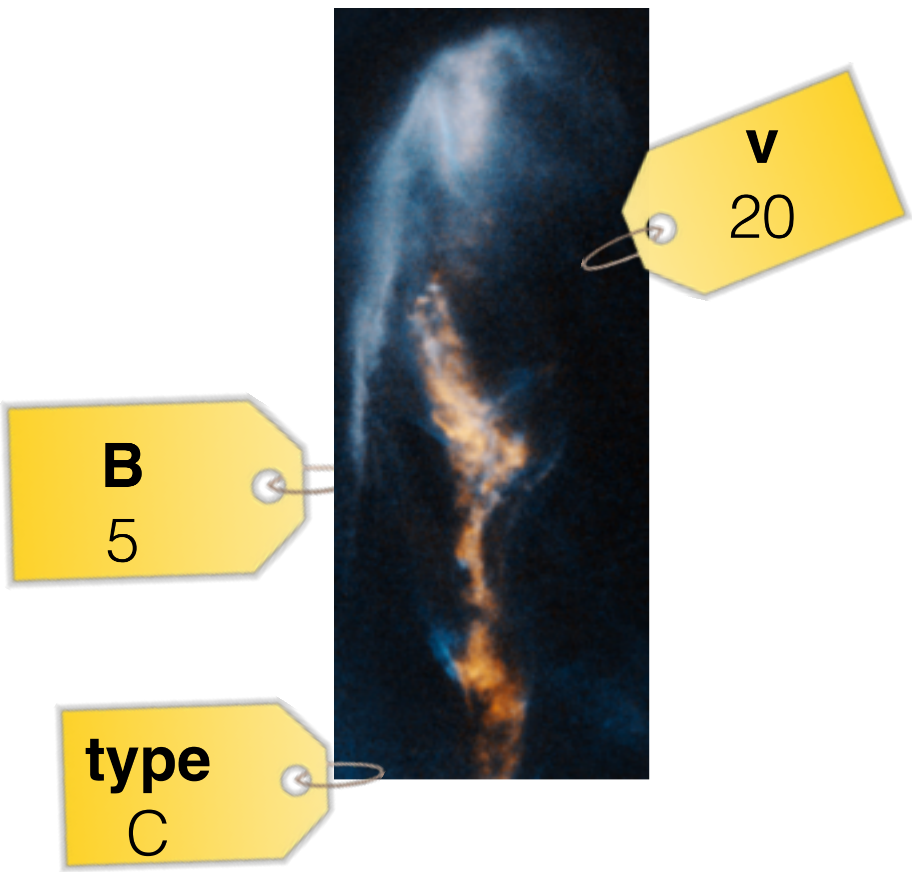
Actually, the problem is just moved, not solved
The new problems are often harder to solve

→ **Be careful about new tech trends…**
They are often intended to very specific cases, which are not yours

Back to basis

LISP (1958): association lists

(tag value)



shock

Interstellar cloud

Object = ((tag1 value1) (tag2 value2) …)

Data tagging

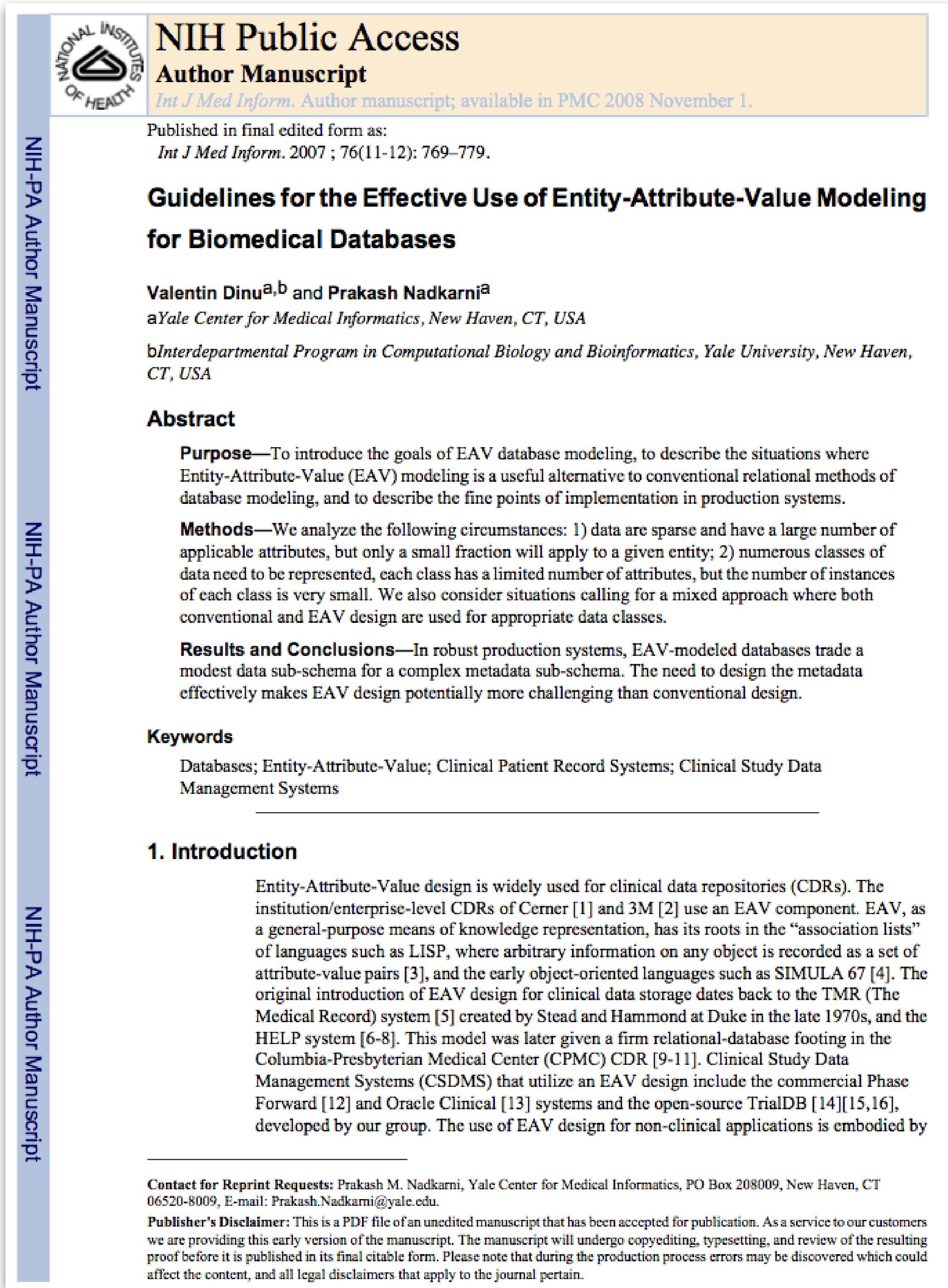
- **Unlimited number of tags for an object**
  - Solve the high dimensionality challenge
- **Unlimited tag combinations to a given object**
  - Solve the dimensions heterogeneity challenge (sparse matrix)
- **Unlimited number of objects can be tagged**
  - Solve the large number of object challenge

+

- Abstract enough to be implemented on top of many technologies
  - RDBMS (EAV)
  - key/value engines
  - noSQL

But at the cost of complex data query

Already used in biomed



NIH-PA Author Manuscript

NIH Public Access
Author Manuscript
Int J Med Inform. Author manuscript; available in PMC 2008 November 1.

Published in final edited form as:
Int J Med Inform. 2007 ; 76(11-12): 769–779.

Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases

Valentin Dinu[a,b] and **Prakash Nadkarni**[a]

[a]*Yale Center for Medical Informatics, New Haven, CT, USA*

[b]*Interdepartmental Program in Computational Biology and Bioinformatics, Yale University, New Haven, CT, USA*

Abstract

Purpose—To introduce the goals of EAV database modeling, to describe the situations where Entity-Attribute-Value (EAV) modeling is a useful alternative to conventional relational methods of database modeling, and to describe the fine points of implementation in production systems.

Methods—We analyze the following circumstances: 1) data are sparse and have a large number of applicable attributes, but only a small fraction will apply to a given entity; 2) numerous classes of data need to be represented, each class has a limited number of attributes, but the number of instances of each class is very small. We also consider situations calling for a mixed approach where both conventional and EAV design are used for appropriate data classes.

Results and Conclusions—In robust production systems, EAV-modeled databases trade a modest data sub-schema for a complex metadata sub-schema. The need to design the metadata effectively makes EAV design potentially more challenging than conventional design.

Keywords

Databases; Entity-Attribute-Value; Clinical Patient Record Systems; Clinical Study Data Management Systems

1. Introduction

Entity-Attribute-Value design is widely used for clinical data repositories (CDRs). The institution/enterprise-level CDRs of Cerner [1] and 3M [2] use an EAV component. EAV, as a general-purpose means of knowledge representation, has its roots in the "association lists" of languages such as LISP, where arbitrary information on any object is recorded as a set of attribute-value pairs [3], and the early object-oriented languages such as SIMULA 67 [4]. The original introduction of EAV design for clinical data storage dates back to the TMR (The Medical Record) system [5] created by Stead and Hammond at Duke in the late 1970s, and the HELP system [6-8]. This model was later given a firm relational-database footing in the Columbia-Presbyterian Medical Center (CPMC) CDR [9-11]. Clinical Study Data Management Systems (CSDMS) that utilize an EAV design include the commercial Phase Forward [12] and Oracle Clinical [13] systems and the open-source TrialDB [14][15,16], developed by our group. The use of EAV design for non-clinical applications is embodied by

Contact for Reprint Requests: Prakash M. Nadkarni, Yale Center for Medical Informatics, PO Box 208009, New Haven, CT 06520-8009, E-mail: Prakash.Nadkarni@yale.edu.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

- Used for a long time in Biomed
- Similar design used for years in RDF & Prolog
- Can benefit from robustness of RDBMS engine as implementation layer

There is **no silver bullet** here,
just choose the **right technology**
for the **right problem**

→ For simulated data with few dimensions,
a classical relational schema is the best solution.

What about huge number of objects ?

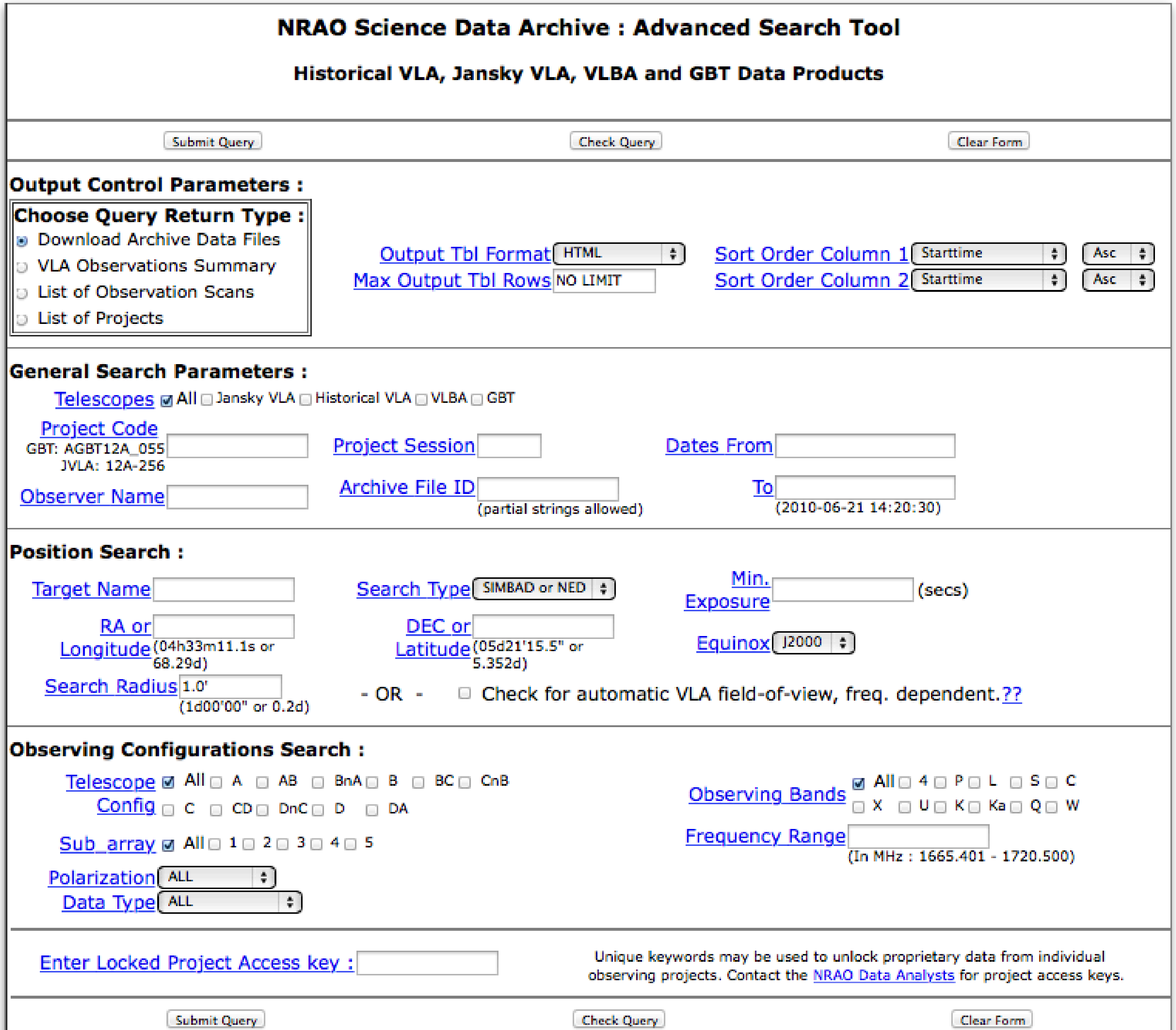
- Assume infinite collections
  - Use the **Stream abstraction** (infinite lazy list -> LISP)
- **Provide high level library** allowing easy handling of streams
  - ex: on top of a votable paginated api

Human - machine interactions

How would a human being easily interact with such a system ?

Human - machine interactions

Interface of the VLA archive:


NRAO Science Data Archive : Advanced Search Tool

Historical VLA, Jansky VLA, VLBA and GBT Data Products

Submit Query | Check Query | Clear Form

Output Control Parameters :

Choose Query Return Type :
- Download Archive Data Files
- VLA Observations Summary
- List of Observation Scans
- List of Projects

Output Tbl Format: HTML
Max Output Tbl Rows: NO LIMIT
Sort Order Column 1: Starttime, Asc
Sort Order Column 2: Starttime, Asc

General Search Parameters :

Telescopes: All, Jansky VLA, Historical VLA, VLBA, GBT

Project Code (GBT: AGBT12A_055, JVLA: 12A-256)
Project Session
Dates From
Observer Name
Archive File ID (partial strings allowed)
To (2010-06-21 14:20:30)

Position Search :

Target Name
Search Type: SIMBAD or NED
Min. Exposure (secs)
RA or Longitude (04h33m11.1s or 68.29d)
DEC or Latitude (05d21'15.5" or 5.352d)
Equinox: J2000
Search Radius: 1.0' (1d00'00" or 0.2d)
- OR - Check for automatic VLA field-of-view, freq. dependent. ??

Observing Configurations Search :

Telescope Config: All, A, AB, BnA, B, BC, CnB, C, CD, DnC, D, DA
Observing Bands: All, 4, P, L, S, C, X, U, K, Ka, Q, W
Sub_array: All, 1, 2, 3, 4, 5
Frequency Range (In MHz : 1665.401 - 1720.500)
Polarization: ALL
Data Type: ALL

Enter Locked Project Access key :

Unique keywords may be used to unlock proprietary data from individual observing projects. Contact the NRAO Data Analysts for project access keys.

Submit Query | Check Query | Clear Form

23 search parameters
classical “complexe” Interface

PDR services :
150 000+ parameters

Brian Glendenning, NRAO, InterOp Heidelberg 2013

Human - machine interactions

2 steps

- What are the available dimensions I can query ?
- What query can I do against a dimension ?

Google France

Recherche Google | J'ai de la chance

+

Semantic web

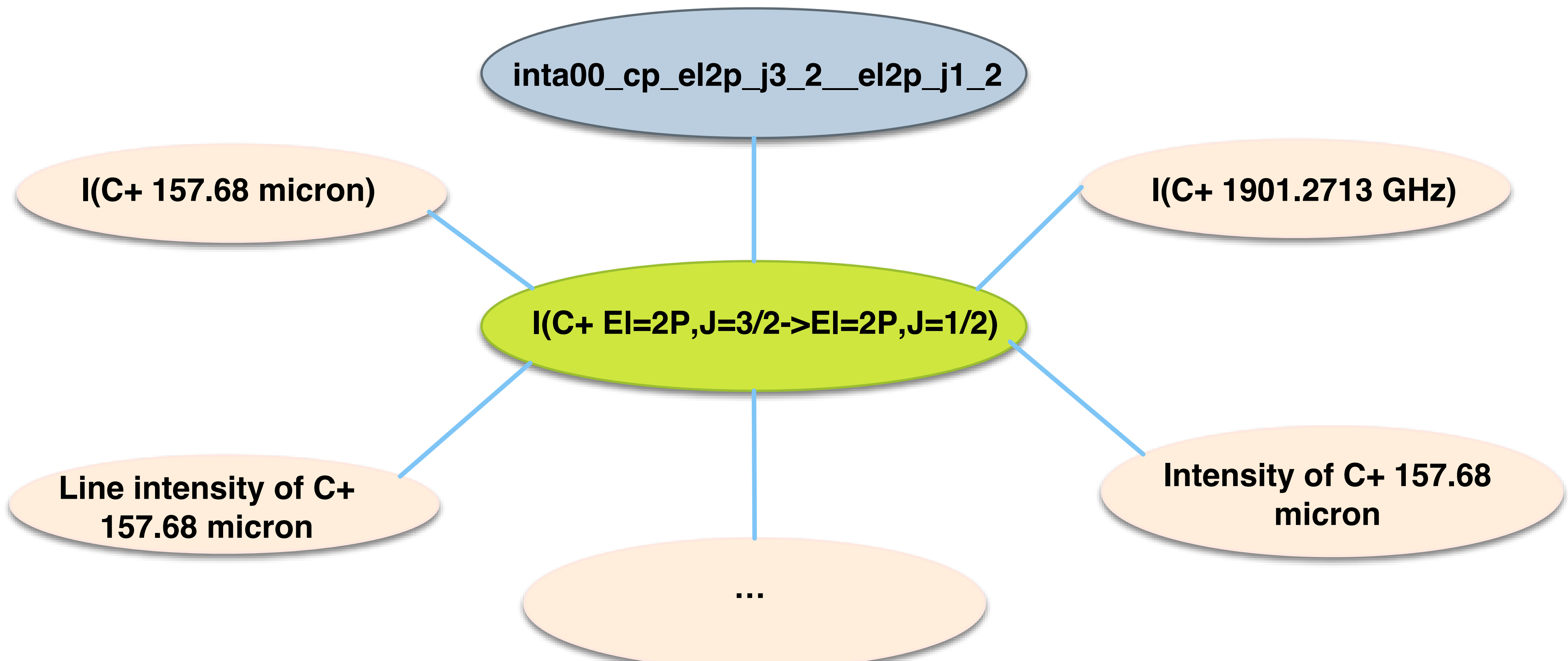
Dimension discovery

Vocabulary for the PDR code

- The thousands of quantities handled by the PDR code are tagged
  - ID
  - human readable name
  - unit + description

- Creation of the synonyms list

- Currently : ~ 300 000 terms



Dimension query

simple DSL: a tiny subset of SQL

Axis constraints

ex: N(Fe+) > 6e12

Add

N(H2)

N(H2) > 8.0E20
N(H2) < 8.8E20
N(CO) > 1.0E13
N(CO) < 1.0E14
I(C+ El=2P,J=3/2->El=2P,J=1/2 angle 00 deg) > 3.6E-6

Applications

- Observations analysis
- Statistical aggregated analysis on grid of models
- Cross grid (different codes) consolidation
- Machine learning (we have started that with Emetic Bron)

In particular

- Today
  - **Manual features extraction**
  - Manage a lot of features
- Tomorrow
  - **Automatic features extraction** (+ scientist checking)
  - Pattern recognition / analysis
    (ex: generic quantities relationships)

Demo

Plot axis

x | nH (input parameter) | (cm-3) | ☑ log scale

y | ISRF scaling factor (back side) | (Mathis_unit) | ☑ log scale

Fixed axis

AVmax | (mag) | 1

Axis constraints

ex: N(Fe+) > 6e12

Add | N(H2)

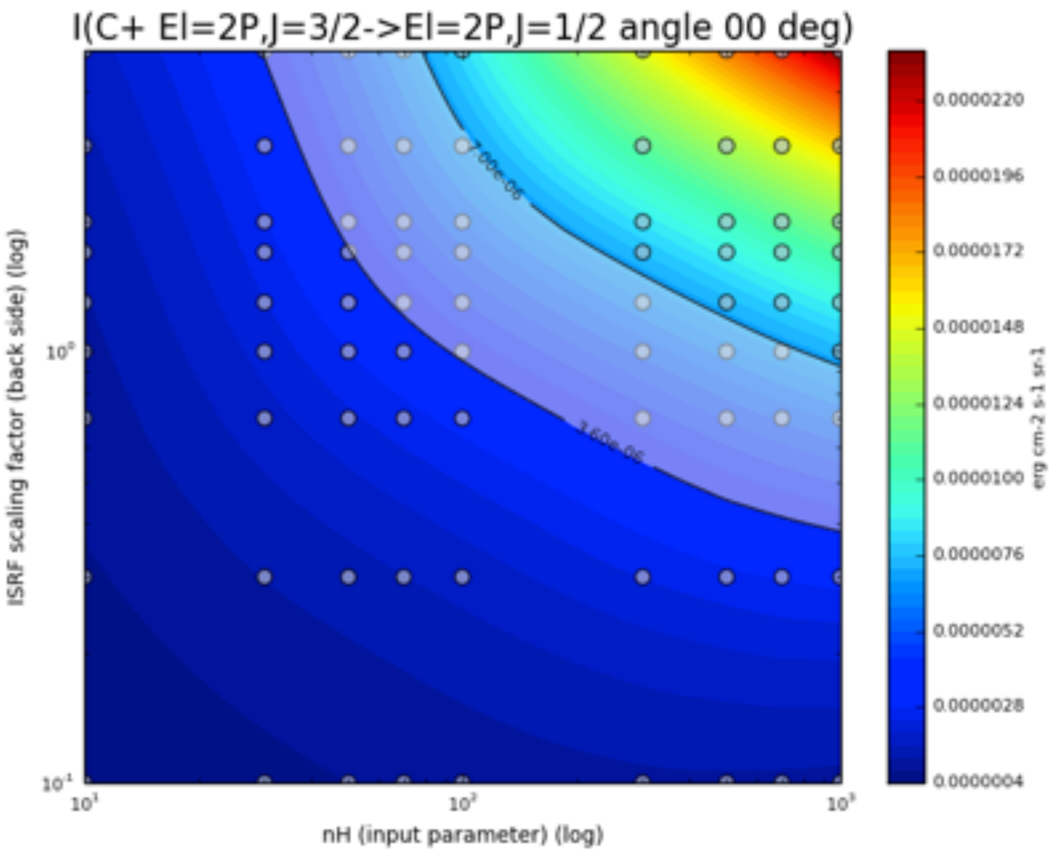
N(H2) > 8.0E20
N(H2) < 8.8E20
N(CO) > 1.0E13
N(CO) < 1.0E14
I(C+ El=2P,J=3/2->El=2P,J=1/2 angle 00 deg) > 3.6E-6

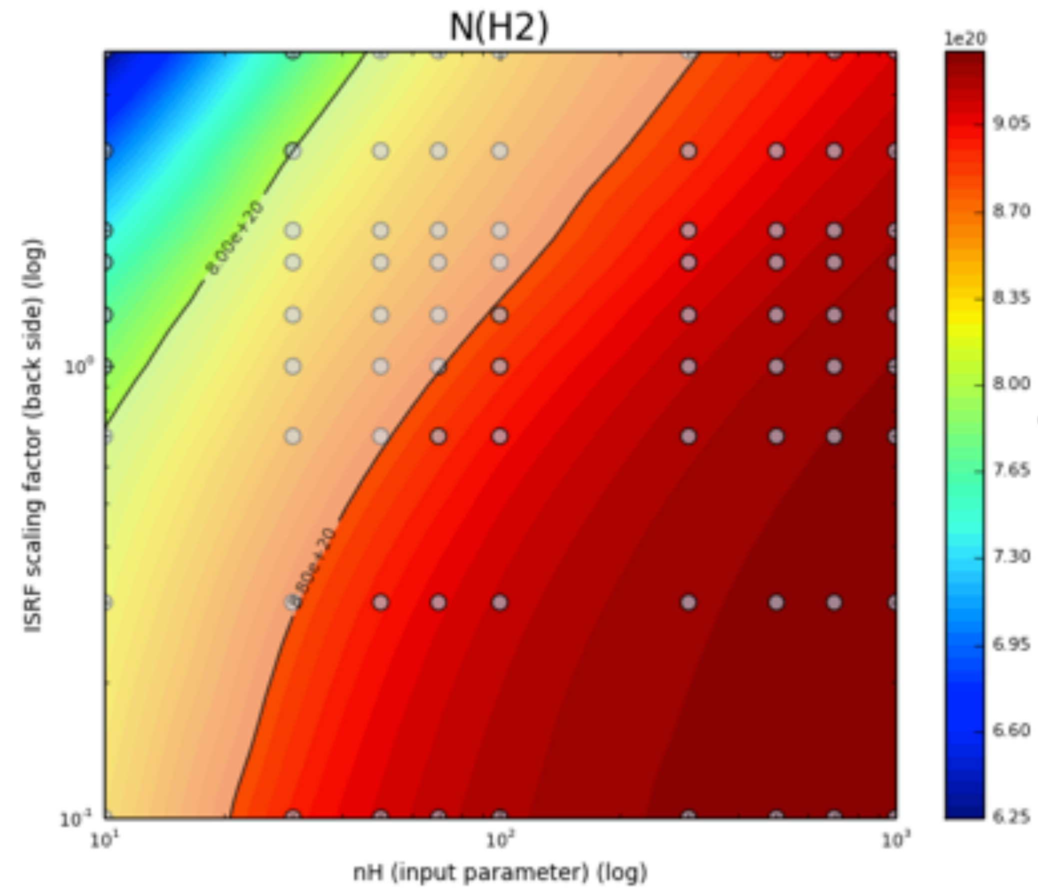
N(H2)

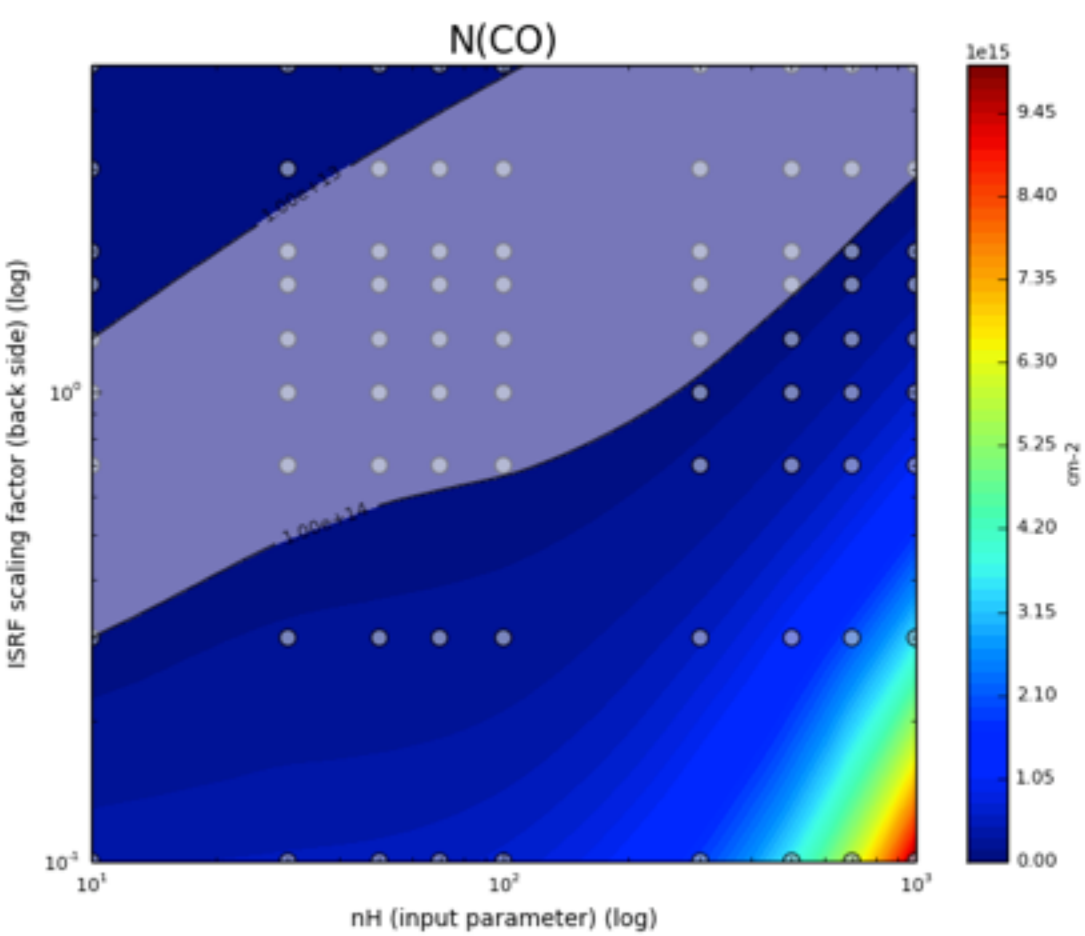
name: N(H2)
doc: none
range: [3.99e+19, 1.87e+21]
unit: cm-2

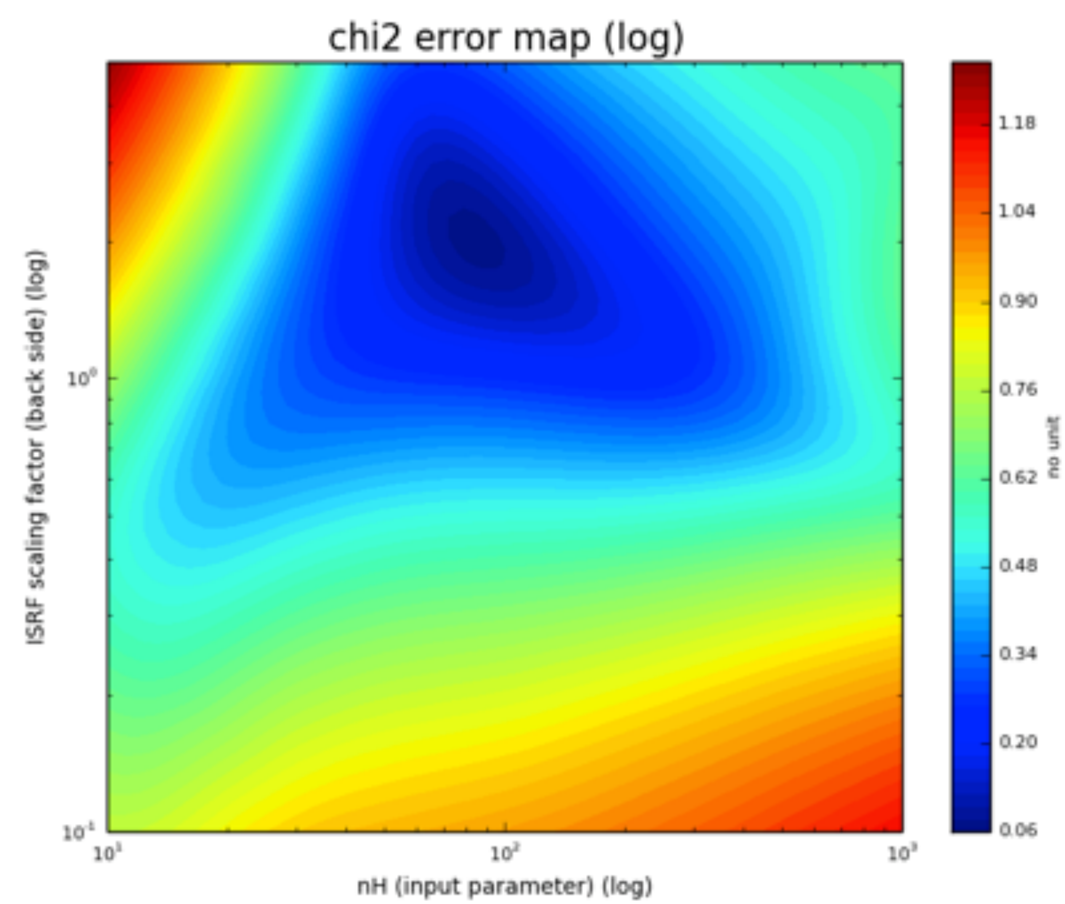
Plot

Demo


I(C+ El=2P,J=3/2->El=2P,J=1/2 angle 00 deg)
ISRF scaling factor (back side) (log)
nH (input parameter) (log)
erg cm-2 s-1 sr-1
0.0000220
0.0000196
0.0000172
0.0000148
0.0000124
0.0000100
0.0000076
0.0000052
0.0000028
0.0000004


N(H2)
ISRF scaling factor (back side) (log)
nH (input parameter) (log)
1e20
9.05
8.70
8.35
8.00
7.65
7.30
6.95
6.60
6.25
cm-2


N(CO)
ISRF scaling factor (back side) (log)
nH (input parameter) (log)
1e15
9.45
8.40
7.35
6.30
5.25
4.20
3.15
2.10
1.05
0.00
cm-2


chi2 error map (log)
ISRF scaling factor (back side) (log)
nH (input parameter) (log)
1.18
1.04
0.90
0.76
0.62
0.48
0.34
0.20
0.06
no unit

Conclusion

- The “Big Data” is under-utilised.
- Use the machine to increase the ROI of processing time slots / big shared equipments.
- Help to identify the quantities where a real scientific work is required.
  - Machine Learning
- Consolidate the semantic dbs (vocabularies)
- **Document how the machine works / process (because there is no magic !)**

Focus on the real scientific questions, let the machine do the dumb job

Summary:

Quite heavy to set up for now

- Complexity shifted from the client to the server
- Exploratory programming: perimeter of the problem at hand not always well known/defined

Opens interesting opportunities