



CDISC India Day

Automating ADNCA Dataset Preparation with Per-Step CDISC Conformance Gates

Ragav Rajan S · Pumas-AI
CDISC India Day 2026



Speaker



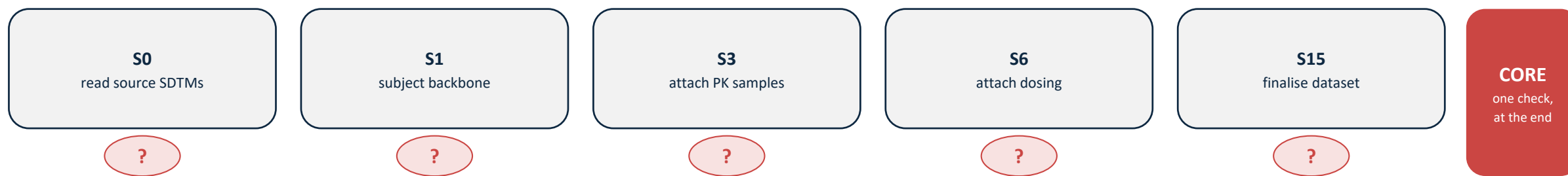
Ragav Rajan S
Data Scientist
Pumas-AI

I am Ragav Rajan S, a data scientist at Pumas-AI Inc, with expertise in Julia and R programming, machine learning, and statistics. My work focuses on developing tools and strategies for data handling, reporting, and downstream analytics. I am also extending that work into automating these workflows using AI and large language models.

Your ADNCA failed. Which step broke it?

An ADNCA, the analysis dataset every PK parameter is computed from, is built one derivation at a time.

We split the build into seventeen steps; whatever the split, today they are checked once, together, at the end.



any one of them could have caused the finding below

What comes back

```
dataset  ADNCA
variable DOSEA
record   2,118
severity Error
message  value not permitted
```

What it does not say

Which step caused it?

Tracing it back is manual, and every fix means re-deriving and re-validating.

The finding tells you what is wrong. It never tells you which step made it wrong. That is the black box.

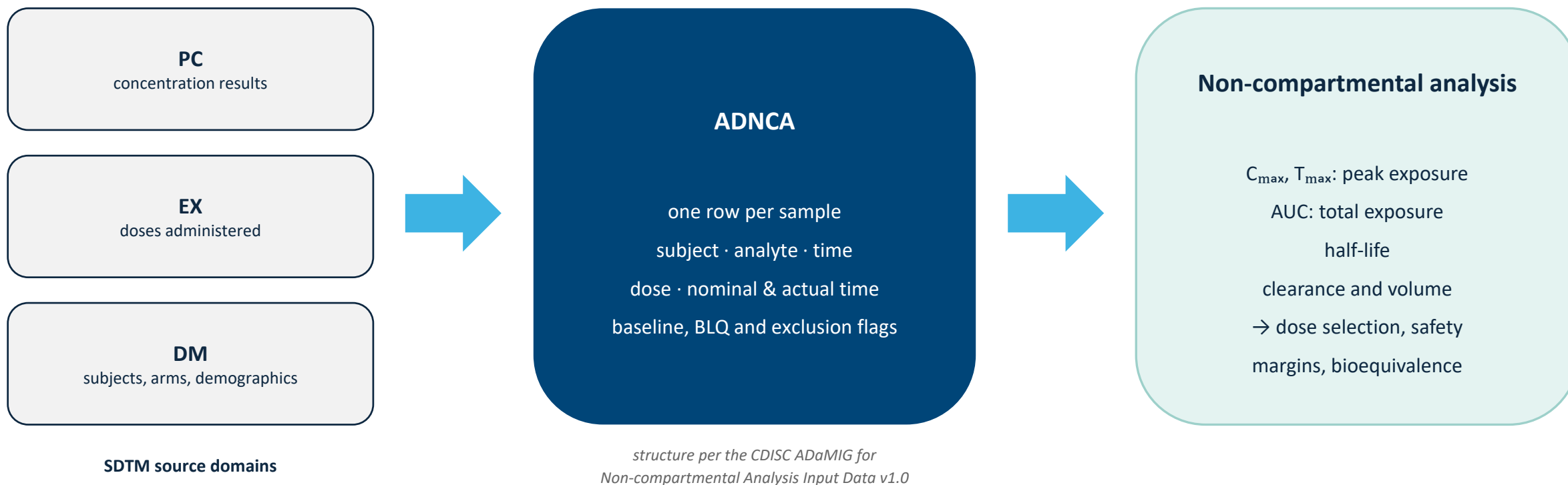
Agenda

1. What an ADNCA is, and why conformance matters
2. The change: a gate after every step
3. What CORE is, and the two pieces of technology
4. The pipeline, and what happens at each stage
5. A worked example on public CDISC pilot data
6. What we take from it



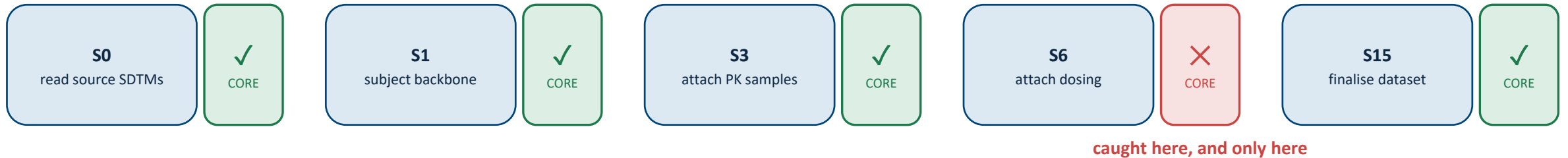
What is an ADNCA?

ADNCA, Analysis Data for Non-compartmental Analysis, is one analysis-ready dataset: every measured concentration sits next to the dose and the clock that explain it.



Shift the check left

The change: a conformance gate after every step.



What a gate is

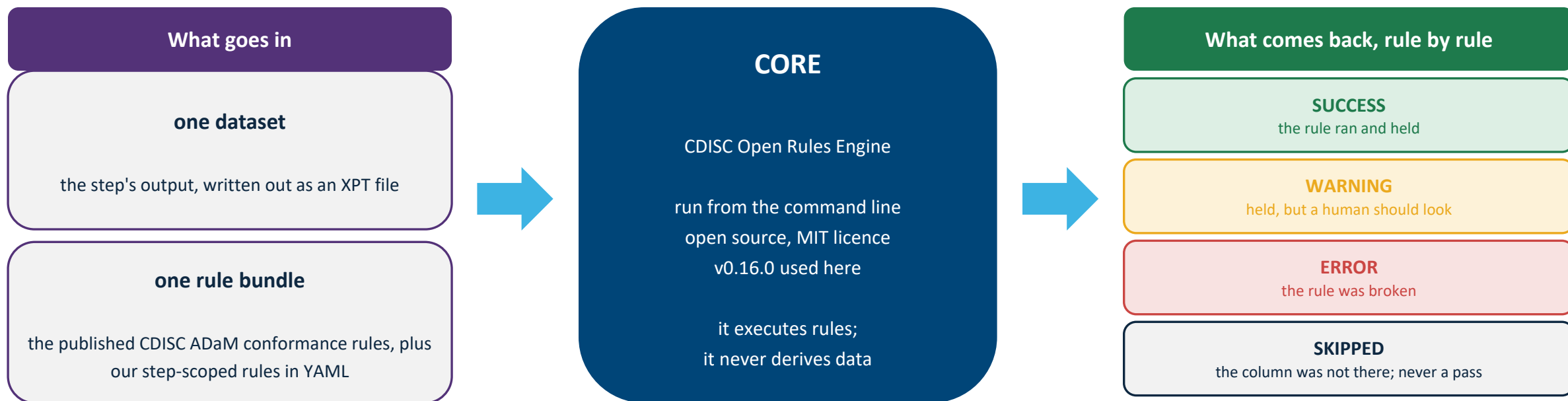
- The step's output is written out as an XPT file
- CORE validates it against that step's own rules
- The next step does not start until CORE validates it

What it buys

- A finding names the step that introduced it
- The fix is usually a small edit, not a rebuild
- Defects cannot compound into the next stage

What CORE is, and what it gives back

CORE is CDISC's own open-source conformance engine. It is the only thing in this pipeline that decides pass or fail.



One JSON report per gate, and one Excel report at the end: every number in this talk is a line in one of those files.

We did not write the judge. CORE decides; the pipeline only decides when to ask it.

Three pieces of technology, in plain terms

ADaM.jl

the machinery · dev version

- Tested Julia functions, one per step
- `attach_dosing`, `derive_relative_times`, ...
- All 17 stages of the build, covered
- Documented and versioned
- It knows how to derive, not what

PumasAide

the bridge

- Connects the agent to a live Julia session
- Runs each stage's ADaM.jl call there
- Serves the real Pumas API docs, so a signature is read, never guessed
- It carries out; it decides nothing

A Claude Code skill

the procedure

- A written procedure: a checklist in a file
- Authored and reviewed by a human, like an SOP
- Names the stages, their order and their rules
- Says what to do when a check fails
- It decides what runs, not what passes

It is NOT

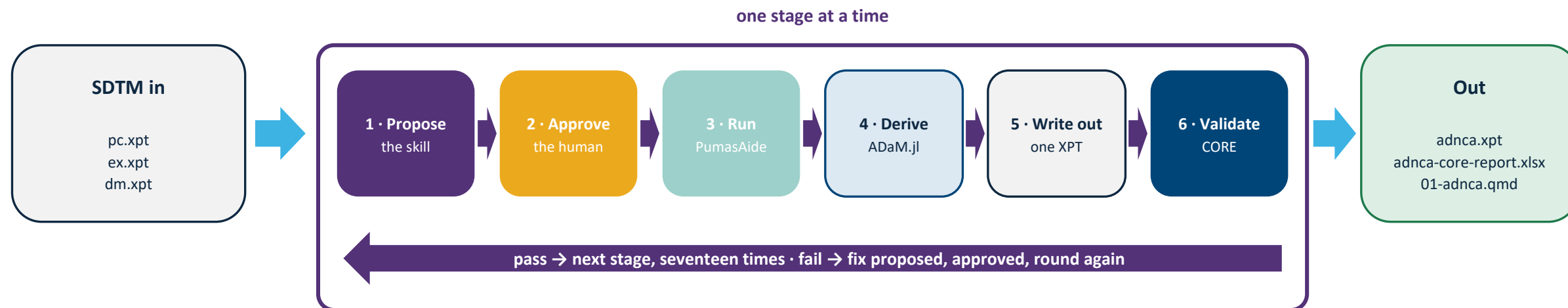
a model improvising a pipeline, choosing its own checks, or deciding what counts as a pass

It IS

a checklist a human wrote, executed the same way every time, without drift or shortcuts

The library derives. PumasAide runs it. The procedure decides what runs, in what order, and when to stop.

Every stage, and who decides



The skill · executes

picks the next stage, writes its code, calls CORE, proposes a fix on a fail; never edits the source data

The human · approves

every stage before it runs and every fix after a fail; adjudicates warnings in writing; can abort; owns the dataset

ADaM.jl · derives

one tested Julia function per stage, run through PumasAide in a live session; knows how to derive, never what to derive

CORE · judges

the only source of pass or fail; CDISC ADaM rules plus our step-scoped ones, run at the owning stage; a skip is never a pass

The model proposes. CORE decides. The human approves.

What it looks like on disk

The procedure: a folder of plain text

```
.claude/skills/adnca-prep/
├── SKILL.md           ← the procedure
├── references/
│   ├── stage-map.md   stage → function → rules
│   ├── core-cli.md    how to call CORE
│   ├── remediation.md what to do on a failure
│   └── gotchas.md
├── templates/
│   ├── adnca-template.qmd
│   └── rules/
│       ├── adnca_rules_s0.yaml
│       ├── adnca_rules_s1.yaml
│       └── ... one bundle per stage
```

One rule: adnca_rules_s1.yaml

```
- Core: {Id: ADNCA-S1-SPINE-001}
  Severity: Error
  Description: "One row per USUBJID
    on the spine (unique key)."
```

Scope:

```
  Domains: {Include: [ADNCA_S1_SPINE]}
```

Check:

```
  all:
    - {name: USUBJID,
      operator: is_not_unique_set,
      value: STUDYID}
```

Outcome:

```
  Message: "USUBJID duplicated
    on the S1 spine."
```

The rule bundles live in the skill, hand-authored from ADaMIG v1.3 and the ADaMIG for NCA Input Data v1.0, then reviewed like an SOP.

Scope pins the rule to one stage's output, so an S1 rule cannot fire at S6. That is what makes a finding name its step.

This ADNCA pipeline: 17 stages, 17 gates



☐ ACT 0 · read & inspect ☐ ACT 1 · build the spine ☐ ACT 2 · describe & flag ☐ ACT 3 · finalise & validate

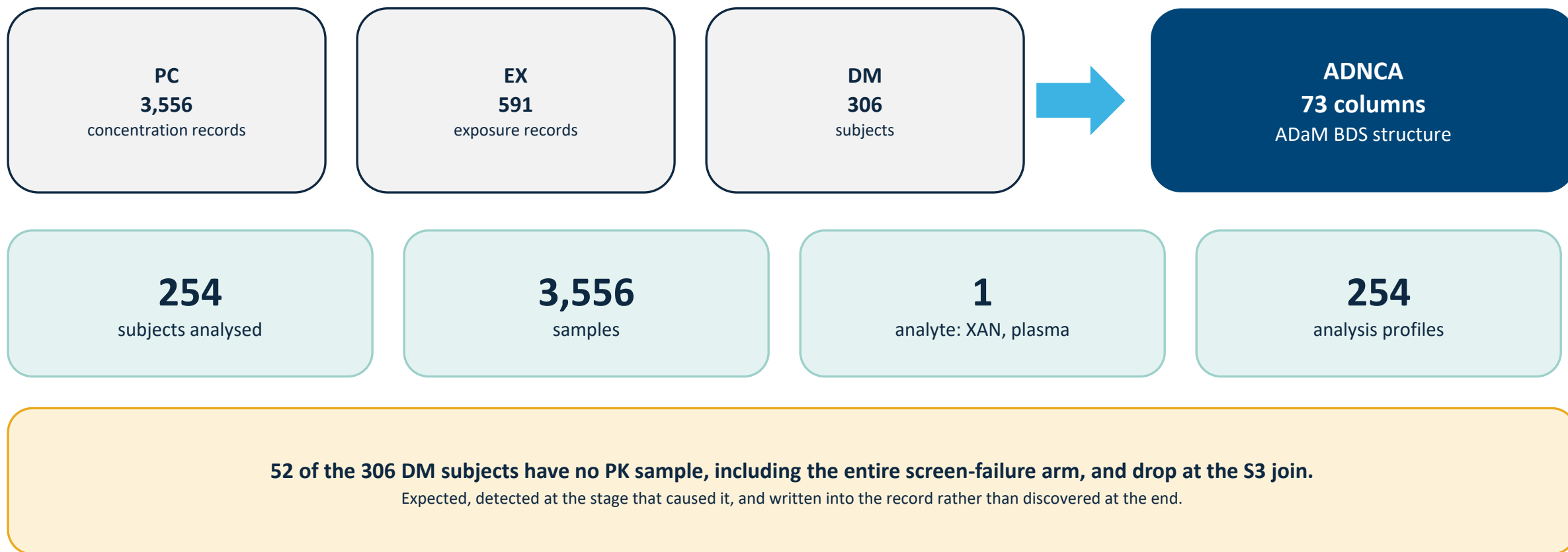
92 rules across 17 gates, one after every stage.



S15's bundle was empty on an earlier run, so CORE asserted nothing and the gate reported "not validated". It now carries 7 rules.

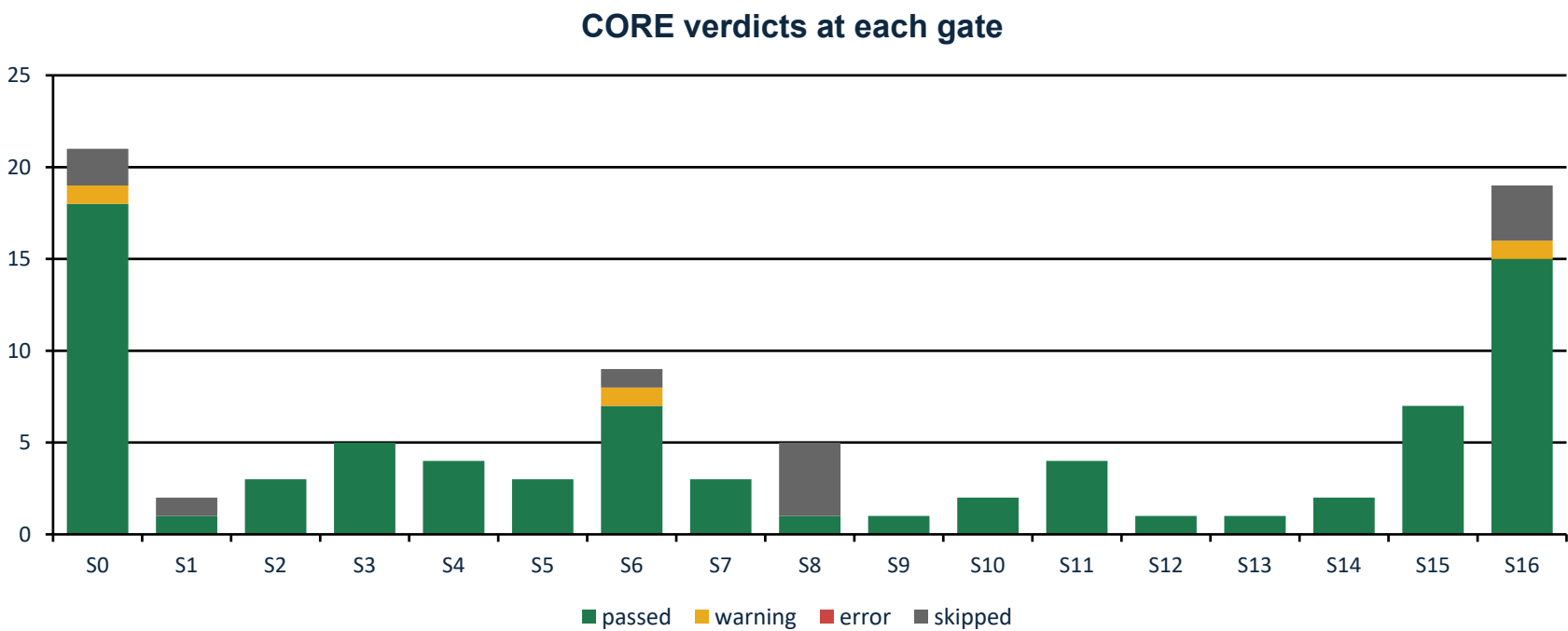
A worked example

CDISC pilot study SDTM: public data, so the whole run is reproducible by anyone.



Results

One bar per gate; the height is the rules that ran there, the segments are how they came back. Every number is a CORE result, not a self-report.



0
errors
across all 17 gates

3
warnings
all the placebo arm,
adjudicated in writing

11
skips
each diagnosed at the stage
that owns the column

92 rules · 17 gates · 0 errors · 3 adjudicated warnings · 11 skips diagnosed

the deliverable every CORE run
datasets/derived/adnca.xpt
results/00-dataprep/01-adnca/reports/s0 ... s16.json

the Excel report the record
results/00-dataprep/01-adnca/reports/adnca-core-report.xlsx
programs/00-dataprep/01-adnca.qmd

The audit trail

01-adnca.qmd: the code, and the record of running it

```
## Stage S6: attach dosing
```{julia}
adnca_s6 = attach_dosing(adnca_s5, ex;
 reference = REFERENCE, ...)
write_stage_xpts("s6", [(:adnca_s6, "dose")])
```
```

****CORE validation****: 9 rules, 0 errors, 1 warning

- ok ADNCA-S6-DOSE-001/002/003/4: ADTM,
PCRFTDTM, FANLDTM on every sample.

- warn ADNCA-S6-DOSE-007: DOSEA ≤ 0
on 1204 rows. Exactly the
Placebo arm. Flagged, not
dropped.

- skip ADNCA-S6-DOSE-011: DOSEFRQ
absent. Inapplicable by
configuration.

results/: the run record

Every intermediate kept · data/s0 ... s16/: one XPT per stage, 17 folders

Every CORE run kept · reports/s0 ... s16.json, plus the final Excel report

Pinned environment · Manifest.toml pins every version; reruns are byte-identical

What that one line records

CORE raised this at S6, the stage that attached dosing.
Not at the end, against a finished dataset.

A human read it, recognised the placebo arm, and wrote the
adjudication into the file. Flagged, not silenced, not dropped.

The skill wrote the line as it ran. Nobody typed it afterwards.

Ask “why is this value what it is?” and the answer is in the file, at the stage that produced it.

What to take away

- **Move the conformance check upstream, not just automate it**
 - A finding at the end names the data. A finding at the step names the derivation.
- **The AI executes a procedure a human wrote; it does not judge**
 - CORE alone decides pass or fail; every fix needs a human accept.
- **Treat a skipped or empty check as a gap, never as a pass**
 - Silent gaps are the failure mode that end-only validation hides best.
- **Earlier localisation is the contribution; speed is a side effect**
 - Much faster than assembling by hand, but the reason to do it is that defects stop compounding.



Questions?

CDISC standards

ADaMIG for Non-compartmental Analysis Input Data v1.0 cdisc.org/standards/foundational/adam/adamig-non-compartmental-analysis-input-data-v1-0

ADaMIG v1.3 cdisc.org/standards/foundational/adam/adamig-v1-3

ADaM Conformance Rules v4.0 cdisc.org/standards/foundational/adam/adam-conformance-rules-v4-0

Conformance checking

CDISC Open Rules Engine (CORE) cdisc.org/core

cdisc-rules-engine CLI, MIT licence; v0.16.0 used here github.com/cdisc-org/cdisc-rules-engine

The automation layer

Claude Code skills code.claude.com/docs/en/skills

Agent Skills, the open standard behind SKILL.md platform.claude.com/docs/en/agents-and-tools/agent-skills/overview

PumasAide, the MCP server the skill runs Julia through docs.pumas.ai/stable/aitools/overview

The build library

ADaM.jl ADNCA build API, Pumas-AI; development version docs.pumas.ai/stable/tutorials/adam/

Ragav Rajan S · Pumas-AI

CDISC India Day 2026

r.rajan@pumas.ai



Appendix: the summary the skill prints

The table the skill prints when the last gate closes. Every number below is a CORE result, not a self-report.

| stage | rules | warn | err | skip |
|-------|-------|------|-----|------|
| S0 | 21 | 1 | 0 | 2 |
| S1 | 2 | 0 | 0 | 1 |
| S2 | 3 | 0 | 0 | 0 |
| S3 | 5 | 0 | 0 | 0 |
| S4 | 4 | 0 | 0 | 0 |
| S5 | 3 | 0 | 0 | 0 |
| S6 | 9 | 1 | 0 | 1 |
| S7 | 3 | 0 | 0 | 0 |
| S8 | 5 | 0 | 0 | 4 |

| stage | rules | warn | err | skip |
|-------|-------|------|-----|------|
| S9 | 1 | 0 | 0 | 0 |
| S10 | 2 | 0 | 0 | 0 |
| S11 | 4 | 0 | 0 | 0 |
| S12 | 1 | 0 | 0 | 0 |
| S13 | 1 | 0 | 0 | 0 |
| S14 | 2 | 0 | 0 | 0 |
| S15 | 7 | 0 | 0 | 0 |
| S16 | 19 | 1 | 0 | 3 |

92 rules · 17 gates · 0 errors · 3 adjudicated warnings · 11 skips diagnosed

the deliverable datasets/derived/adnca.xpt
the Excel report results/00-dataprep/01-adnca/reports/adnca-core-report.xlsx
every CORE run results/00-dataprep/01-adnca/reports/s0 ... s16.json
the record programs/00-dataprep/01-adnca.qmd

Appendix: the delivered ADNCA

| No. | STUDYID | USUBJID | SUBJID | SITEID | AGE | AGEU | SEX | RACE | ANL01FL | NCAXFL | NCAXFN | PKSUMXF |
|-----|--------------|-------------|--------|--------|-----|-------|-----|-------|---------|--------|--------|---------|
| 1 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 2 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 3 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 4 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 5 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 6 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 7 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 8 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 9 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 10 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 11 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 12 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 13 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 14 | CDISCPIL0T01 | 01-701-1015 | 1015 | 701 | 63 | YEARS | F | WHITE | Y | | 0 | Y |
| 15 | CDISCPIL0T01 | 01-701-1023 | 1023 | 701 | 64 | YEARS | M | WHITE | Y | | 0 | Y |
| 16 | CDISCPIL0T01 | 01-701-1023 | 1023 | 701 | 64 | YEARS | M | WHITE | Y | | 0 | Y |

datasets/derived/adnca.xpt: 3,556 rows x 73 columns, 254 subjects, one analyte.

Appendix: the CORE report

| Version | CDISC RuleID | FDA RuleID | Message | Status |
|---------|--------------|------------|---|----------------|
| 1 | | | A required key column is missing on an ADNCA record. | SUCCESS |
| 1 | | | A required NCA core column is missing on an ADNCA record. | SUCCESS |
| 1 | | | PARAMCD violates the sanitized-code shape at output. | SUCCESS |
| 1 | | | ROUTE not in {iv, ev, inf} — map from CDISC SDTM ROUTE before validating. | SUCCESS |
| 1 | | | PARAM and PARAMCD are not one-to-one. | SUCCESS |
| 1 | | | PARAM and PARAMN are not one-to-one. | SUCCESS |
| 1 | | | Duplicate record key (USUBJID, PARAMCD, ATPTN, DTYPE). | SUCCESS |
| 1 | | | Tied MRRLT within a profile — read_nca needs strictly increasing time. | SUCCESS |
| 1 | | | ABLFL is not in {Y, N, missing}. | SUCCESS |
| 1 | | | BLOFL is not in {Y, N, missing}. | SUCCESS |
| 1 | | | DOSEU or AVALU missing on a populated record. | SUCCESS |
| 1 | | | AVISIT and AVISITN are not one-to-one. | SUCCESS |
| 1 | | | ATPT and ATPTN are not one-to-one. | SUCCESS |
| 1 | | | PARCAT1 and PARCAT1N are not one-to-one. | SUCCESS |
| 1 | | | TRTP and TRTPN are not one-to-one. | SKIPPED |
| 1 | | | TRTA and TRTAN are not one-to-one. | SKIPPED |
| 1 | | | TRTSEQP and TRTSEQPN are not one-to-one. | SKIPPED |
| 1 | | | ATPTREF differs from AVISIT on a record. | SUCCESS |
| 1 | | | Analysis-flagged record will be dropped by the read_nca filter. | ISSUE REPORTED |

results/00-dataprep/01-adnca/reports/adnca-core-report.xlsx: the S16 gate, 19 rules, 0 errors, 1 warning, 3 skips.

ADNCA preparation

ADNCA preparation

/adnca-prep

```

#| label: s0d-pc
inspect_pc(pc)

#| label: s0d-ex
inspect_ex(ex)

#| label: s0d-dtc
check_datetime_missing(pc, [:PCDTC]; id = :USUBJID, key = :PCTESTCD)

#| label: s0d-ids
check_id_overlap(dm, pc; a_name = "DM", b_name = "PC")

#| label: s0d-prefilter
# pc = @rsubset(pc, coalesce(:PCSTAT, "") != "NOT DONE")

```

Its four tables are what I read the scoping answers ('KEEP_ANALYTES', 'BLQ_TOKENS', 'FREQ_TO_TAU', ...) e

AskUserQuestion

Stage S0D

Insert Stage S0D (the four source diagnostics) into 01-adnca.qmd?

- ☐ Accept
Insert and evaluate. Its tables drive the ADNCA Setup scoping answers I'll propose next.
- ☐ Skip
Advance without diagnostics — the scoping answers would then stay at template defaults, unjustified.
- ☐ Abort
Halt the loop and print the status table.
- ☐ Other

1 Submit answers

Esc to cancel

01-adnca.qmd U

```

programs > 00-dataprep > 01-adnca.qmd > ...
237 # ADNCA PIPELINE
252 ## Stage S0 - read source SDTHs - ACT 0
254 ```{julia}
256 #| eval: true
257 pc = read_sdtm(SDTM_DIR, "pc")
258 ex = read_sdtm(SDTM_DIR, "ex")
259 dm = read_sdtm(SDTM_DIR, "dm")
260 write_stage_xpts("s0", [[:pc, "pc"], [:ex, "ex"], [:dm, "dm"]])
261 ```
262
263 **CORE validation** - 21 rules, 0 errors, 1 warning
264
265 - ok ADNCA-S0-DM-001/002, PC-001..010, EX-001/002/003/005/006/007 - all
Error-severity
266 source checks pass on CDISCPLOT01.
267 - warn ADNCA-S0-EX-004 - EXDOSE ≤ 0 on 226 EX rows. These are the 'Placebo' arm
('EXTRT = "PLACEBO"', 'EXDOSE = 0'), which is a real dose record, not a data
268 defect.
269 Warning-severity, so not remediated; placebo subjects carry no measurable
XAN and
270 will drop out at S3's inner join or wash out as BLQ.
271 - skip ADNCA-S0-DM-007 (not evaluated: TRT01P/TRT01A absent - the source is
SDTM DM,
272 which carries ARM/ACTARM. S2 maps ':ARM => :TRTP', ':ACTARM => :TRTA';
verify TRTP/TRTA
273 are populated there rather than treating this skip as a pass.)
274 - skip ADNCA-S0-PC-012 (not evaluated: PCSTAT absent from source PC, so there
are no
275 NOT DONE rows to pre-filter. The 's0d-prefilter' chunk stays commented out.)
276
277 # REPRODUCIBILITY
278
279 The program code for this analysis is in the following location:
280
281 - PumasAI-Consulting Services GitHub repo: ```{julia} pumasai_repo```
282 - Program file: ```{julia} qmd_relative_path```
283
284 The packages used in this analysis are listed below:
285
286 D>Run Cell | Run Above
287 ```{julia}
288 import Pkg
289 Pkg.status()
290 ```

```