

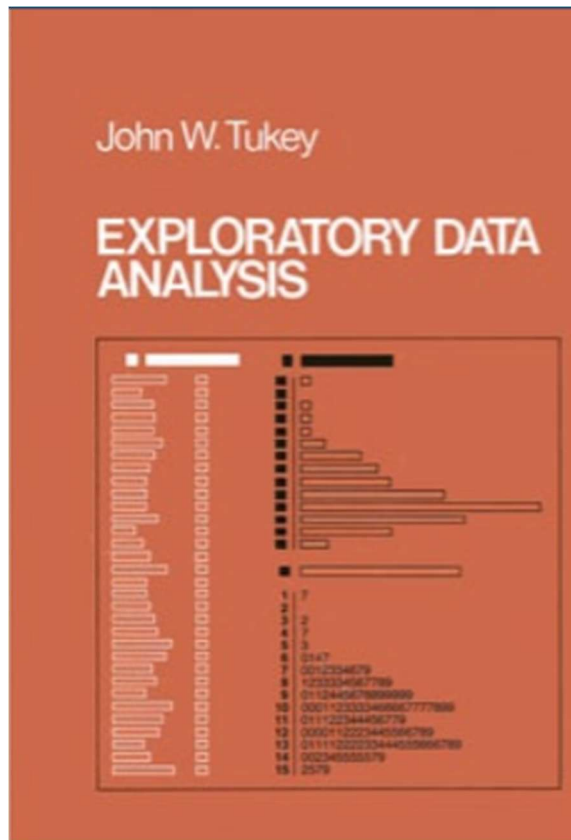
EDA for QC of ADaM

Trust requires traceability

James Joseph

- [EDA for QC of ADaM](#)

EDA



“Don’t expect standard summaries to reveal the unusual.”

QC Levels

- L1 = self-review
- L2 = peer-reviewed
- L3 = double-coded, compared

*QC level is not enforced, only planned

The Problem We Solve

“We're finding same issues, again?!”

- QC logs scattered, usually Excel
- Task tracking is inconsistent
- Verification is hard to trace
- Issues repeat across cycles

What We've Learned

Actually Quadruple Programming!

- Excel doesn't scale
- Traceability is non-negotiable
- Double-Independent QC = Biased.
- Roles must be enforced

EDA Solution

[Event-driven architecture](#) is a software design model built around the publication, capture, processing and storage of events.

Where EDA Fits In

Extract → **Transform**

Between raw input and formal output.

→ Load

Where EDA Fits In

QC Plan → QC Log + Deliverables

→ Live Tracker → **Archive**

Who This EDA For

Reviewers

- Biostatisticians verifying deliverables
 - Programming managers
 - QC leads and reviewers
-

Our Terminology

- **Study** = container
 - **QC Plan** = qc levels + assignments
 - **Deliverable** = grouped files
 - **File** = assigned unit
 - **Artifact** = required file
-

+ CLINICAL_DEI
Ttest Analysis

Stage
3
Status
In Progress
Due Date
03/15/2024

2f
2u
0i

F001
ttest.py
QC L3
In Progress

1u 0i 2a

Assignments (1)
+

L1
Developer
Florian Reihl (florian.reihl@edaclinical.com)
x

Issues (0)
+

Add assignment
No issues yet

Artifacts (2)
+

F011
Define-XML
Doc
https://github.com/cdisc-org/DataExchange-DatasetJson/blob/master/exar
x

F012
Validation Results
Doc
https://github.com/eda/validation/paired_ttest_validation.html
x

+ F002
ttest.r
QC L2
In Progress

1u 0i 1a

+ Add File

Enforcing QC Level

- **L1** = self-review•,
- **L2** = peer-reviewed
- **L3** = double-coded, compared

NOTE: not optional;

Our Roles

- **Manager:** assign, compare, approve
 - **Developer:** implement
 - **Lead** = Developer + Manager
 - **Reviewer:** approve
 - **Admin:** assign roles in system
-

Developer View of Tracker

 **ttest.py**

deliv_001 - Paired T-Test Analysis

Study: STUDY002 • [Lead Developer](#)

Due: 2024-03-15T00:00:00.000Z

Stage 3  2

File Details

Status

Stage Level

 Planning

Stage 3

Artifacts

 Define-XML
F011

protocol [View](#) [External](#)

 Validation Results
F012

report [View](#) [External](#)

How QC Levels Are Enforced

Role boundaries protect Level 3 QC integrity:

- Developers can't view other developers' files.
 - Developers cannot run compares on their own outputs.
 - Developers do not share the same input files.
-

Triggers

- File assigned by manager, e.g. upload/update qc plan →
 - Event = logged activity
-

IssueLogged

LBTEST ⌵ Lab Test or Exami...		⏮ ⏪ ⏩ ⏭ ⏴ ⋮
	LE Ca	ⓘ Show Metadata
Albumin	CH	🏷 Use Column Labels
Alkaline Phosphatase	CH	📄 Column Visibility
Alanine Aminotransferase	CH	🔍 Build Query
Aspartate Aminotransferase	CH	⚠ Report Issue
		Add an issue report
		↔ Compare with Table
		📄 Export
		📄 Copy Content

StatusChanged

[illegible]

Notification of Event

● Signed In



Notifications

[Mark all as read](#)



Deliverable Ttest Analysis Ready for Compare

CLINICAL_DEL_001 changed from in-progress to review

🕒 2 minutes ago



File ttest.py has been Submitted

F001 changed from in-progress to review

🕒 2 minutes ago



File ttest.r has been Submitted

F002 changed from in-progress to review

🕒 40 minutes ago

[View all notifications](#)

Tracker

QC Plan in Action

- NDJSON = full event log
- Reconciler builds tracker from log
- Every change is derived
- No manual edits allowed

Types of events

Setup & Submission Events

Issue Management Events

System-Derived Events

Setup & Submission Events

Event	Purpose	Tracker	Notify?
QCPlanUploaded	Start plan	File view shown	No
FileSubmitted	Submit code or compare	Artifact marked	Yes
DevCommented	Add dev note	Note shown	No

FileSubmitted

ttest.py Python

▶ Run Analysis

:

ttest_results.json Success

⬇️ ↺

```
1 import numpy as np
2 import scipy.stats as stats
3 import json
4 from datetime import datetime
5
6 def perform_ttest():
7     """
8     Performs an independent samples t-test comparing two groups.
9     Returns results in JSON format.
10    """
11
12    # Load data from md.json file
13    with open('md.json', 'r') as f:
14        data = json.load(f)
15
16    group_a = data['group_a']
17    group_b = data['group_b']
18
19    # Calculate descriptive statistics
20    mean_a = np.mean(group_a)
21    mean_b = np.mean(group_b)
22    std_a = np.std(group_a, ddof=1)
23    std_b = np.std(group_b, ddof=1)
24
```

```
{
  "analysis_type": "Independent Samples T-Test",
  "timestamp": "2025-06-11T15:34:43.059Z",
  "sample_sizes": {
    "group_a": 20,
    "group_b": 20
  },
  "descriptive_statistics": {
    "group_a": {
      "mean": 87.4,
      "std_dev": 2.89,
      "data": [
        85,
        89,
        76,
        92,
        88,
        84,
        91,
        87,
        83,
        90,
        86,
        88
      ]
    }
  }
}
```

● Analysis completed successfully

⌂ Submit Results

FileSubmitted

POST

/api/v1/events

Submit Event

^

Submit an event to the system

Parameters

Try it out

No parameters

Request body

required

application/json

▼

Example Value

Schema

```
{
  "action": "FileSubmitted",
  "file_id": "string",
  "deliverable_id": "string",
  "study_id": "string",
  "object_id": "string",
  "metadata": {}
}
```

Issue Management Events

Event	Purpose	Tracker	Notify?
IssueLogged	Block a file	File = blocked	Yes
IssueResolved	Reopen workflow	File status reevaluated	Yes

IssueLogged

POST

/api/v1/files/{file_id}/issues

Create Issue

^

Open a new issue for a specific file (Authorized users only)

Parameters

Try it out

Name	Description
file_id required	file_id
string	
(path)	

Request body required

application/json

Example Value

Schema

```
{
  "title": "string",
  "description": "string",
  "priority": "medium",
  "created_by": "string"
}
```

System-Derived Events

Event	Trigger	Tracker	Notify?
StatusChanged	Artifact state check	Status changes	Yes
StageCompleted	All stage files complete	Status changes	Yes
DeliverableComplete	All stages passed	Status changes	Yes

UAT

The CAMIS Project

- Assigned devs to build R + Py tests
 - Same input, independent code
 - Compare after both uploaded
 - Compare result logs QC pass/fail
-

Execute

ttest.py Python

▶ Run Analysis

:

ttest_results.json Success

⬇️ ↺

```
1 import numpy as np
2 import scipy.stats as stats
3 import json
4 from datetime import datetime
5
6 def perform_ttest():
7     """
8     Performs an independent samples t-test comparing two groups.
9     Returns results in JSON format.
10    """
11
12    # Load data from md.json file
13    with open('md.json', 'r') as f:
14        data = json.load(f)
15
16    group_a = data['group_a']
17    group_b = data['group_b']
18
19    # Calculate descriptive statistics
20    mean_a = np.mean(group_a)
21    mean_b = np.mean(group_b)
22    std_a = np.std(group_a, ddof=1)
23    std_b = np.std(group_b, ddof=1)
24
```

```
{
  "analysis_type": "Independent Samples T-Test",
  "timestamp": "2025-06-11T15:34:43.059Z",
  "sample_sizes": {
    "group_a": 20,
    "group_b": 20
  },
  "descriptive_statistics": {
    "group_a": {
      "mean": 87.4,
      "std_dev": 2.89,
      "data": [
        85,
        89,
        76,
        92,
        88,
        84,
        91,
        87,
        83,
        90,
        86,
        88
      ]
    }
  }
}
```

● Analysis completed successfully

📄 Submit Results

Submit

Submit File



Once submitted, file will be available for review and you may receive comments.

Study

Select study



Deliverable

Select deliverable

Stage

Select stage



Status

Select status



- ☐ I followed EDA's SOP-001 QC Plan
- ☐ I followed EDA's WI-042 Work Instruction

Submit the file

Seed

ttest.pyPython▶ Run Python

```
1 import numpy as np
2 import scipy.stats as stats
3 import json
4 from datetime import datetime
5
6 def perform_ttest():
7     """
8     Performs an independent samples t-test comparing two groups.
9     Returns results in JSON format.
10    """
11
12    # Load data from medical_study.json file
13    with open('medical_study.json', 'r') as f:
14        data = json.load(f)
15
16    group_a = data['group_a']
```

Run Python analysis to see results

ttest.RR▶ Run R

```
1 library(jsonlite)
2 library(dplyr)
3
4 perform_ttest <- function() {
5     # Load data from medical_study.json file
6     data <- fromJSON("medical_study.json")
7
8     group_a <- data$group_a
9     group_b <- data$group_b
10
11    # Calculate descriptive statistics
12    mean_a <- mean(group_a)
13    mean_b <- mean(group_b)
14    sd_a <- sd(group_a)
15    sd_b <- sd(group_b)
16}
```

Run R analysis to see results

N

🔗 Run Comparison Analysis

Compare

ttest.py Python ▶ Run Python	ttest.R R ▶ Run R
<pre>1 import numpy as np 2 import scipy.stats as stats 3 import json 4 from datetime import datetime 5 6 def perform_ttest(): 7 """ 8 Performs an independent samples t-test comparing two groups. 9 Returns results in JSON format. 10 """ 11 12 # Load data from medical_study.json file 13 with open('medical_study.json', 'r') as f:</pre>	<pre>1 library(jsonlite) 2 library(dplyr) 3 4 perform_ttest <- function() { 5 # Load data from medical_study.json file 6 data <- fromJSON("medical_study.json") 7 8 group_a <- data\$group_a 9 group_b <- data\$group_b 10 11 # Calculate descriptive statistics 12 mean_a <- mean(group_a) 13 mean_b <- mean(group_b)</pre>
<pre>{ "language": "Python", "analysis_type": "Independent Samples T-Test", "timestamp": "2025-06-11T18:02:29.645Z", "sample_sizes": { "group_a": 20, "group_b": 20 }, "descriptive_statistics": { "group_a": { "mean": 87.4, "std_dev": 2.890275 }, "group_b": { "mean": 80.05,</pre>	<pre>{ "language": "R", "analysis_type": "Independent Samples T-Test", "timestamp": "2025-06-11T18:02:29.946Z", "sample_sizes": { "group_a": 20, "group_b": 20 }, "descriptive_statistics": { "group_a": { "mean": 87.4, "std_dev": 2.890275 }, "group_b": { "mean": 80.05,</pre>
🔗 Run Comparison Analysis	

Roadmap: Validation

- SOPs + user docs
- Verify test modules (t-test, ANOVA)
- Output audit trail across tools
- Regulatory-ready (SOC2, Part 11)

Integration Strategy

- Microsoft Enterprise ID
- Azure container
- API- first (integrate with other tools)
- MS Graph API for SharePoint, Teams, GitHub

Our Differentiators

Real double programming

- Append-only event log automates tracker
- Role-based views to maintain independence
- Compare engine across R, Py, SAS
- Cloud-native, Microsoft-integrated

Thank You

TP - SOP & Work Instructions

JJ – Product & Architecture

FR – Frontend + UI

XH – System + Reconciliation

Let's talk QC. cal.com/eda

Demo

- [Viewer Demo](#)
- [API Demo](#)

-
- [EDA for QC of ADaM](#)