Overview

When an analysis that requires advanced analytics is initiated, both the code repository in bitbucket and the corresponding folder structure on EFS are created. Users whose names are associated with studies/analyses created via Analysis Wizard (eg. Analysis Lead and Contributors) will automatically get access to the Bitbucket code repository and EFS server folder that is automatically created if they choose to use Advanced Analytics Tools such as R, SAS, JupyterHub during the creation process in Analysis Wizard.

The folder structure is organized by Therapeutic Area and individual Analyses, and the repository owner and code reviewer(s) are from the respective Project teams - as specified during the creation in Analysis Wizard when adding the "Contributors".

EFS Folder Structure and Naming Convention

The master folder structure shows how each analysis is organized in relation to a Therapeutic Area or across Therapeutic Areas.

The overall goals of the unified folder structure are as follows:

All folders and code repository follow GxP guidelines

GxP policies apply to all Analyses created through the Analysis Wizard

The code sub-folder containing the versioned code for individual studies is cloned into Bitbucket

The folder structure is tool-agnostic

Folder structure is organized by Therapeutic Area and Analysis

Every subfolder is organized by RWDEx generated Analysis no#

Every analysis follows GxP storage guidelines; each analysis may follow additional GxP guidelines as required

1. Therapeutic Area

Any analyses that fall under a one specific Therapeutic area, for instance, oncology or cardiovascular or neuroscience etc. are placed in this folder. If analyses are across therapeutic Areas or Therapeutic area is unknown, they can be added to the cross\_therapeutic\_area folder

1.1. (analysisnumber)\_indication\_projecta\_teamnameplaceholder/

Each analysis in a specific area is organized by analysis number, indication, project description and team name

Folder Naming Convention:

Analysis number is a unique identifier generated from the RWDEx platform (eg. stdy123)

“Indication” refers to the therapeutic area subcategory. For instance, if Therapeutic area is oncology indication can be ‘melanoma’, ‘small cell lung cancer’ etc. (selected from a drop down box)

‘projecta’ is a short project description for the analysis (limited number of characters)

The last qualifier in 1.1 is the team name (for instance, ‘bards’, ‘tigermed’, ‘sp4core’, ‘cheors’, ‘eds’ etc)

This team name should be a unique to your team and should serve as a quick identifier

If a project is between Core and BARDS, the team name can be ‘core0bards’

1.1.1 adhoc/

Under each analysis there will be several standard folders. The first folder is the adhoc folder. This is used for any adhoc/code/input/outputs that don’t need to be version controlled. Users are free to create subfolders under the adhoc folder as they see fit

The adhoc folder is to be used only when rest of the subfolder don't meet requirements. If the code in adhoc folder needs to be moved to production, it should be placed in teh code/src folder

1.1.2 code/

Under each analysis, the second folder is code. There are several characteristics to be noted here:

Code folder is language/tool agnostic. Code written in any language should be placed here

Code folder mirrors a repository in Bitbucket (for detailed information see SAS, JH or R reproducibility WFs)

All code will be version controlled upon commiting into Bitbucket

Access to the code repository for any additional user (not marked as Contributors in Analysis Wizard) in bitbucket will be granted by the Support Team upon request

Only the Support Team can delete repositories from Bitbucket

This folder on EFS shoud always reflect the latest production version of the code from BitBucket. There is a triangular relationship in between the Local/home workspace, BitBucket, and EFS study folder:

The local workspace, aka. ~home folder, is used for everyday development

The BitBucket is to be used as target remote repository for storing the approved code, and represents the single source of truth

The EFS study folder is to be an access restricted folder where the finalized version of code is stored for reproducibility and archival purposes, and should mirror the latest BitBucket version

Detailed Drill down

1.1.2.1 conf/

Under each code folder, the first folder is the configuration folder. This will contain any configuration or property files that are specific to the code

1.1.2.2 lib/

Under each code folder, the second folder is the library folder. This will contain any custom macros that are relevant to the analysis

1.1.2.3 src/

Under each code folder, the statistical programming/analysis should occur in the source folder. Users are free to create subfolders under source as they see fit

Subfolders such as modeling, preprocessing, preparation, extraction, transformation etc. can be created

1.1.2.4 study\_files/

Any documentation/ study files/SOPs for the analysis should be placed here (eg. protocol, programming specifications, table mockups, icd/ndc/cpt code lists, etcl). Since the 1.1.2 code folder is a repository in bitbucket, the study files will be version controlled once checked into bitbucket. Users are free to create subfolders under study files as they see fitt

1.1.2.5 utility/

Under each code folder, the last folder is the utilities folder. This will contain any common procedures that are relevant to the analysis

1.1.3 input/

The third folder under each analysis is the input folder. All inputs files that are required for the analysis should be placed here

Users are free to create additional folders under the input folder aside from the standardized structure defined

Detailed Drill down

1.1.3.1 data\_raw/

Under each input folder, the first folder is the data raw folder. All preliminary/raw input datasets should be placed here

Datasets that are extracted from databases should be placed here

1.1.3.2 data\_processed/

Under each input folder, the second folder is the data processed folder. All intermediary input datasets should be placed here.

Datasets that are outputs from code and are inputs to successive code should be placed here

1.1.3.3 ref/

Under each input folder, the third folder is the reference folder. Any lookup files, data dictionaries, ICD9/10/NDC codes or other reference material can be placed in this folder. Note: The source/raw version of icd/ndc/cpt codes is considered part of the analysis specifications and needs to be saved in the code repository/bitbucket so it is stored in /study\_files often in excel format. The converted SAS/R version to be used in programs is stored here in /ref

1.1.4 output/

The fourth folder under each analysis is the output folder. All output files that are created for the analysis should be placed here.

Users are free to create additional folders under the output folder aside from the standardized structure defined

Detailed Drill down

1.1.4.1 out\_data/

Under each output folder, the first folder is the out data folder. All proprietary datasets such as .sas datasets that are generated from the code should be placed here

1.1.4.2 out\_graphics/

Under each output folder, the second folder is the out graphics folder. All graphics or visualizations that are generated from the code should be placed here

1.1.4.3 out\_logs/

Under each output folder, the third folder is the out logs folder. All logs that are generated from executing the code should be placed in this folder

1.1.4.4 out\_lst/

Under each output folder, the fourth folder is the out lst folder. All data list files that are generated from the code should be placed here

1.1.4.5 out\_tables/

Under each output folder, the last folder is the out tables folder. All exportable datasets (such as excel, csv, pdf files) that are generated from the analysis should be placed in this folder

1.1.5 validation\_developer\_testing/

Every analysis requires developer testing. All material related to the Developer Testing should be placed here

Each validation folder will have a similar structure. It will contain three standard subfolders:

Code: All developer-tested code should be checked in the validation folder-specific repository. All study files should be placed here as well

Input: All inputs required for developer testing should be placed here

Output: All outputs created from developed testing should be placed in this folder

The subfolders under code, input, output will be the same as detailed in the previous slides

Detailed Drill down

1.1.6 validation\_double\_programming/

Any analysis that requires double programming should use this folder. The double programmer should conduct analysis using the guidelines set forth in the code repository

Each validation folder will have a similar structure. It will contain three standard subfolders:

Code: All double programming code should be checked in the validation folder-specific repository. All study files should be placed here as well

Input: All inputs required for double programming should be placed here

Output: All outputs created from double programming should be placed in this folder

The subfolders under code, input, output will be the same as detailed in the previous slides

Detailed Drill down

1.1.7 validation\_independent\_review/

Any analysis that requires independent review should use this folder. The independent reviewer should use the guidelines set forth in the code repository

Each validation folder will have a similar structure. It will contain three standard subfolders:

Code: All independent review code should be checked in the validation folder-specific repository. All study files should be placed here as well

Input: All inputs required for independent review should be placed here

Output: All outputs created from independent review should be placed in this folder

The subfolders under code, input, output will be the same as detailed in the previous slides

Detailed Drill down

2. RWE Collaborative Code Hub (“The Hub”)

This folder is to be used as a code library.

The root of the RWE Collaborative Code Hub is located at:

EC2 Linux - /efs/analysis/rwe\_collaborative\_code\_hub

For each code available on RWE CCH, a program is created or updated in the development area:

EC2 Linux (for Redshift within Real World Data Exchange (RWDEx)):

/efs/analysis/rwe\_collaborative\_code\_hub/inprogress

(for RWE standardization initiative following the project folder structure on EC2)

CPI Linux for PE studies:

\\bardsar-test\rwe\pe-standards

(for RWE standardization initiative following the project folder structure on CPI)

Once finalized, validated, and documented, the code program is promoted to production and checked into Bitbucket\* master branch of RWE CCH at:

RWE\_Collaborative\_Code\_Hub project overview

The Library Management Committee will approve the promotion of code from ‘inprogress’ to Bitbucket

\*(Please refer to B-S004.ER98 for RWE Guidelines for RWDEx Bitbucket Code Check-in for checking code into Bitbucket)

2.1 inprogress/

This folder serves as a “staging area” for the code library. Once analysis-specific code is developed in therapeutic area-specific folders, it can be moved to this folder and prepared to be moved into the Sharable folder. The subfolders under ‘inprogress’ and ‘Sharable’ are broken down by the level of testing done and the number of projects the code has been tested on

The naming convention under ‘inprogress’ is as follows:

Type of folder:

Jobaid: Contains code or macros identified as of interest (re-used through projects utility or statistically interesting). No documentation or testing is required at this location, but may be put here if available.

Knowledgelib: Contains generalized codes or macros that have been developer-tested intended to be used across projects. Any documentation is stored in this area with the minimum being programming specification. During the process of standardization, the developer-tester has the responsibility to make available the developer tested code to the users through KnowledgeLib.

Macrolib: Contains generalized codes or macros that have been developer-tested intended to be used across projects. Formal developer testing and independent review should be completed in CCH. PRS, instructional program header, documentation (e.g., specification, user instructions), and sample code package are provided in this area.

Cohort Standards: Each cohort has its own folder following naming conventions as: cohort-TA-indication-datasource-year, for example: cohort-antiviral-pneumonia-optum-2020 Cohorts are shared through the RWDEx Collaborative Code Hub via Bitbucket. Each cohort will have its own Bitbucket repository following RWDEx standard folder structure. Code used in creating cohorts, code lists and PRS (programing request specifications) are shared in Bitbucket repository

Study Automation: Codes have been added to jobaid/knowledgelib/macrolib library, can be identified in CCH tracker

PE Standard Code: Since PE Standards code is run on the CPI Linux platform, there is no testing required to add the code to CCH. CCH simply serves as a repository for the code. Each updated program is copied to the code/lib folder and a sample call program is placed in the code/src folder. Documentation is also stored code/study\_files. The library level request is the same as described above for Jobaid, KnowledgeLib or Macrolib.

The subfolders for knowledgelib and macrolib mirror the subfolders for jobaid

2.1.1 jobaid\_analysis#\_projecta\_teamname/

Code that has only been developer-tested on one project is placed in the Jobaid folder

It will contain the following subfolders.

1. code

2. input

3. output

4. validation\_developer\_testing

5. validation\_double\_programming

6. validation\_independent\_review

Each of these folders will have a similar structure as the folders specific to a therapeutic area

2.1.2 knowledgelib\_analysis#\_projecta\_teamname/

Code that has only been developer-tested on a couple of projects is placed in the Knowledgelib folder

It will contain the following subfolders:

1. code

2. input

3. output

4. validation\_developer\_testing

5. validation\_double\_programming

6. validation\_independent\_review

Each of these folders will have a similar structure as the folders specific to a therapeutic area

2.1.3 macrolib\_analysis#\_projecta\_teamname/

Code that has undergone different levels of testing on multiple projects is placed in the Macrolib folder

It will contain the following subfolders:

1. code

2. input

3. output

4. validation\_developer\_testing

5. validation\_double\_programming

6. validation\_independent\_review

Each of these folders will have a similar structure as the folders specific to a therapeutic area

Bitbucket folder structure

The jobaid, knowledgelib, macrolib and cohort standards folders uses the same folder structure.

Assigning Access to CCH

All RWDEx users should have read access to the CCH bitbucket repositories.

The programmers/developers must apply via help desk ticket for

bitbucket repository write access;

/efs/analysis/rwe\_collaborative\_code\_hub read/write/execute access to SAS and R folders for

Jobaid

Knowledge

Macrolib

Development Process - high level POV

As part of the collaboration workflow, users should use the directory structure that has been created specifically for their analysis - check the link for detailed process access in SAS, JH or R

image2021-4-10\_14-49-57.png

The above process should be followed by each developer working on the study/analysis

The code should reference the input and output files in the shared directory of /efs/ so the code can be executed from the home directory of any developer or the shared folder (/efs/analysis/)

The code being saved on the home directory should be pushed to Bitbucket and copied on the server at the same time

Bitbucket should always have the most up to date code. Any new developer working on the code, should clone the code from bitbucket to their home directory before starting the dev work

Code should not be checked in directly to the Shared folder.

Once the study has been archived, users should delete the repo clone in their home directory after 1 year (starting from the archival date)

Modifying EFS folder structure

To change a folder name, user submits a support ticket request that is routed to the AMS team. User provides the Analysis # along with each request which is the unique identifier for the AMS team to identify the folder which gets affected

Only certain parameters can be changed:

Therapeutic Area studies: Indication, project description or team name

RWE Collab Hub Code: Naming Convention, project description, team name

NOTE: \*\*The folder name needs to be changed on the analytical tool server as well as Bitbucket\*\*

Modifying Access - repository

Repository access can be controlled from Bitbucket GUI or via an AMS ticket. To request access for a new team member or to revoke access, please follow these steps

Modifying Name- repository

Repository Name can be controlled from Bitbucket GUI or via an AMS ticket. To request a name change, please follow these steps