MCIN Open Science Guidance

Open Science Data Preparation Checklist

Contact: Samir Das (samir.das ‘at‘ mcgill.ca)

Ethics considerations

- ❏ Contact Dr. Stephanie Dyke (stephanie.dyke ‘at’ mcin.ca) & see MCIN Open Science Guidance - Ethics Considerations

Data quality & Open Science information

- ❏ Data entry completion
- ❏ Manual QC checks of data entry from the source documents (on a subsample dataset), including some automatic checks for scoring algorithms
- ❏ Automated QC checks on the original database
- ❏ Selection of participant data & variables to be shared and their “sharing level” (open vs registered) - See Ethics Considerations above
- ❏ Selection of Consent Codes for shared data - See Ethics Considerations above
- ❏ Creation of new IDs for the datasets to be shared
    - ❏ Map randomly the original candidate IDs to the new IDs for the open instance.
    - ❏ Keep the mapping information on a CSV file separate from the open dataset
    - ❏ See appendix 1 for instructions on how to do this within LORIS

MRI data anonymization

- ❏ De-identification of the imaging dataset
    - ❏ Non-exhaustive list of DICOM identifying fields that need to be scrubbed
    - ❏ See appendix 2 for instructions on how to do this using DICAT for LORIS
- ❏ Scan type identification
    - ❏ See appendix 3 for instructions on how to do this within LORIS
- ❏ Run defacing algorithm on all anatomical files
    - ❏ See appendix 4 for instructions on how to do this within LORIS
- ❏ QC ALL defaced images
    - ❏ See appendix 5 for instructions on how to do this within LORIS
- ❏ Backup and remove non-defaced images from the dataset to be released
    - ❏ See appendix 6 for instructions on how to do this within LORIS
- ❏ Map MRI QC information if any are available
    - ❏ See appendix 7 for instructions on how to do this within LORIS
- ❏ Insertion of Lego Phantom datasets if they were done. See appendix 8
    - ❏ Note, still resolving issues on that front, during the creation of this document
- ❏ Make sure to not share the DICOM files since they are not defaced!
    - ❏ See appendix 9 for instructions on how to do this within LORIS

Behavioural data anonymization

- ❏ Make sure that no dates are available in the open dataset
    - ❏ Instead provide age of the participant at the time the data was collected
- ❏ Re-architecture of some data (e.g. mother tongue changed to “French”, “English”, “Other”, instead of allowing unique data points) - See Ethics Considerations at top of document

- ❏ Carefully go through open text field if some need to be shared to make sure it does not contain any PII. Video and audio data should also be checked with care (depending on the ethics of the project, some of these data may be shared through registered access).

Data documentation

- ❏ Gather list of publications
- ❏ Gather list of variables (per instrument or on the candidate level) and their dictionary
- ❏ Prepare a short summary of the study
- ❏ Creation of the dataset descriptor (JSON-LD file)
- ❏ Prepare documentation on how to get the data
- ❏ LORIS-specific additional steps:
    - ❏ Creation of a script to download MRI data via the API (with documentation)
    - ❏ Setup Document Repository for LORIS instances and upload the documentation
- ❏ Write a paper to describe the dataset and announce the public release of the dataset

LORIS-specific additional steps

- ❏ Set up a new dev, staging and production VM for the open instance
    - ❏ Ensure compatibility across the 3 VMs (same OS, MySQL, PHP, Python, Node...)
- ❏ Upgrade to the latest LORIS release (see Appendix 10)
- ❏ Customize test battery, instrument_subtests, instrument_subgroups, test_names so that only the instrument data to be shared are populated
- ❏ Customize Battery class (see appendix 11)
- ❏ Insertion of demographic information (Gender but no date of birth!) See appendix 12
- ❏ Creation of the visit labels in the new LORIS instance (must customize which fields you want inserted in the session before running script) See appendix 13
- ❏ Creation of the `CommentIDs` using `assign_missing_instruments.php`

- ❏ Insertion of instrument fields into the new LORIS instance (again, only identified variables to be inserted - no dates or sensitive information) See appendix 14
- ❏ Map out special cases whether it'd be candidates or visits or instruments (*e.g.* export as instrument but import as non-instrument)
- ❏ John's age_at_mri.php can be modified and used for age calculations at visit time
- ❏ Determine default user's permissions
- ❏ Set up the document repository (so dataset documentation can be uploaded in LORIS)
- ❏ Set up the mail server
- ❏ Set up the API (see Appendix 15)
- ❏ Tests (all modules, API endpoints, data sanity checks...)
- ❏ Set up the DQT (to be developed)
- ❏ Firewall modifications if crawling & extracting

APPENDICES

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 1: Creation of new set of IDs

- ❏ Generate a new set of IDs for the candidates to be shared
- ❏ Map randomly the original candidate IDs to the new IDs for the open instance.
- ❏ Keep the mapping information on a CSV file outside of LORIS

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 2: De-identification of the imaging dataset using DICAT

1) Grep the `ArchiveLocation` and `PatientName` from the `tarchive` table of the original LORIS instance:

- ❏ Query to use:

```
SELECT PatientName, ArchiveLocation, DateAcquired
  FROM tarchive t
  JOIN session s ON (s.ID=t.SessionID)
  WHERE (PatientName LIKE '%MTL0002%' OR PatientName LIKE '%MTL0003%' ... )
    AND Current_stage!='Recycling Bin'
  ORDER BY DateAcquired;
```

Notes:

- ★ Replace the PatientName by the original PSCID to grep the list of tarchive that will need to be processed in the open VM
- ★ Visit with `Current_stage=”Recycling Bin”` will be ignored

2) Transfer the DICOM archives in the `incoming` folder of the open VM

- ❏ Copy the list of `ArchiveLocation` from the above query in a text file and run the following command:

```
rsync  -a -e 'ssh -p PORT_NUMBER'
--files-from=tarchive_list_to_transfer_to_open_VM
/data/loris/data/tarchive/ lorisadmin@open-VM.loris.ca:/data/incoming
```

3) Extract the DICOM folders on the open VM

- ❏ For each archive untar (twice) the archive in the incoming folder of the new open VM.
- ❏ Since the extraction can take a lot of processing time, they can be run in parallel using QSUB.

4) Run all DICOM studies on the open VM through DICAT

Version 2.2 of DICAT or later should be used using the XML file called `fields_to_zap_for_open_science.xml` instead of the original XML file used with DICAT.

Note: we want all the PatientName to be labelled: `PSCID_CandID_VisitLabel` for candidates using their new open IDs (not their original PSCID and CandID from the original database)

a) Install DICAT on the open VM as instructed in the README of the DICAT repository

- ❏ Link to the DICAT repository: https://github.com/aces/DICAT/tree/master

b) Create a CSV file (`file.csv`) with one DICOM study per row where each row must contain the following: " Path_To_DICOM_folder,New_Patient_Name,, "

- ❏ Example of what the CSV file should look like:

```
/data/not_backed_up/MTL0123_456789_V1,MTL0123_456789_V1,,
/data/not_backed_up/MTL0123_456789_V2,MTL0123_456789_V2,,
/data/not_backed_up/MTL0123_456789_V3,MTL0123_456789_V3,,
```

c) Execute DICAT to de-identify and relabel all DICOM studies

- ❏ Modify the config file `dicat/data/fields_to_zap_for_open_science.xml` to include additional DICOM headers that should be erased from the DICOM studies of the open datasets.
- ❏ Call the DICAT mass script to de-identify the datasets using the XML file called `fields_to_zap_for_open_science.xml` in `%DICAT_dir%/dicat/data`:

```
python mass_deidentify.py -c file.csv -x
%DICAT_dir%/dicat/data/fields_to_zap_for_open_science.xml
```

Notes:

- ★ Ensure to use the file `dicat/data/fields_to_zap_for_open_science.xml` in your DICAT directory for the de-identification procedure so that ALL the date and time information are removed from the DICOM datasets in addition to the other header identified as containing potential identifying information. (Always verify your datasets afterwards to make sure no other identifiable fields should be removed from the DICOMs).
- ★ Since the `SeriesUID` contains the date and time in it, it has to be removed from datasets shared openly.

d) Instructions on how to run DICAT in parallel using QSUB

- ❏ Create X number of CSV files (X = number of CPUs available to QSUB)
- ❏ In the terminal run:

```
qsub -V -S /bin/sh
```

- In STDIN enter:

```
python %DICAT_dir%/dicat/mass_deidentify.py -x
%DICAT_dir%/dicat/data/fields_to_zap_for_open_science.xml -c
/data/incoming/list_1.csv
```

- To signal the end of STDIN input use CTRL+D
- Repeat this process for the ‘X-1’ other CSV files to be run through `mass_deidentify.py`

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 3: Scan type identification

Run the insertion pipeline on the de-identified DICOM datasets (after appendix 2 step)

1) Important note: use the latest `dcm2mnc` provided by Bert that resolves many issues with `dcm2mnc`. To do so you need to:

- ❏ Download the `dcm2mnc` binary (for Ubuntu; for CentOS)
- ❏ Copy the `dcm2mnc` binary into `%LORIS-MRI%/uploadNeuroDB/bin` (with `%LORIS-MRI%` being the path to the LORIS-MRI code)
- ❏ Ensure the path `%LORIS-MRI%/uploadNeuroDB/bin` is in the `$PATH` variable of the `environment` file. Then source the `environment` file
- ❏ Ensure the file is readable and executable by lorisadmin and the apache user (`www-data` for Ubuntu, `apache` for CentOS)
- ❏ In the Config module, modify the “dcm2mnc binary to sue when converting” field of the “Imaging pipeline” section to the name of the new `dcm2mnc` binary
- ❏ Pipeline ready to be run!

2) Special acquisition consideration when populating `mri_protocol`

If the dataset to be shared openly contains fieldmaps or other types of acquisition producing multiple types of images (for instance phase and magnitude images), make sure to use the LORIS release 21.0.0 or later.

In LORIS 21.0.0 and after, it is possible to specify an image type in the `mri_protocol` table (`image_type` field) that can be used to distinguish between image types (for example, phase and magnitude images have different `image_type` values).

3) A few checks after insertion

- ❏ Verify that the MRI violations present in the module are all expected violations
- ❏ Ensure that no uploads failed
- ❏ Ensure that no dates are available anywhere in the imaging tables
- ❏ Ensure that no incidental findings have been inserted by mistake (if so, verify with your ethics to see if they can be shared and under which umbrella (registered vs open)

Appendix 4: Run defacing algorithm on all anatomical files

Note: This step requires at least LORIS release 21.0.0

1) Determine which modalities should be defaced

Usually, all anatomical scans (T1W, FLAIR, T2W, T2 star, MP2RAGE, fieldmap magnitude…) need to be defaced. Once the modalities have been determined, set up the following settings in the Config module under the “Imaging Pipeline” section:

- ❏ “Scan type to use as a reference for defacing...”: select the T1W scan type
- ❏ “Modalities on which to run the defacing pipeline”: select the modalities to deface

2) Run the defacing pipeline

a) Command to run the defacing pipeline on one session:

```
run_defacing_script.pl -profile $profile -sessionIDs $session_id
```

`$profile`: name of the profile file (typically prod); `$session_id`: session ID to process

b) Command to run the defacing pipeline on multiple sessions in parallel:

When defacing needs to be run on multiple session IDs, it is highly recommended to use QSUB to run the defacing in parallel.

- ❏ Ensure QSUB is installed and setup on the VM
- ❏ Ensure the config setting “Project batch management used” is set to “Yes”
- ❏ Create a text file with the list of session IDs to process (one session ID per line)
- ❏ Run the following command using the list of session IDs created in a. (sessionIDs_list)

```
batch_run_defacing_script.pl -profile prod < sessionIDs_list
```

3) Sanity checks to run to make sure all session IDs have been defaced

Run the 2 following queries for each scan type to check that all images have been defaced:

```
SELECT COUNT(*) FROM files f JOIN mri_scan_type mst ON
(f.AcquisitionProtocolID=mst.ID) WHERE Scan_type='T1W';
SELECT COUNT(*) FROM files f JOIN mri_scan_type mst ON
(f.AcquisitionProtocolID=mst.ID) WHERE Scan_type='T1W-defaced';
```

Appendix 5: QC ALL defaced images

Note: This step requires at least LORIS release 21.0.0
For easy QC of the defaced images, 3D rendering of the face can be created. To do so:

- ❏ Make sure that the command `which pipeline_qc_face.pl` points to the file in `uploadNeuroDB/bin` and not the one in the MINC tools
- ❏ Create a temporary file called `foo` that would be creating a file with a date timestamp that you want to use to narrow the find command of the next point

```
touch --date "2019-01-01" /tmp/foo
```

- ❏ Run the command below to find defaced files for which a 3D face rendering JPEG image has not been created yet

```
find /data/loris/data/assembly -name *defaced* -newer /tmp/foo -print >list
```

- ❏ Run the batch script on the list of files that need to be processed

```
batch_run_pipeline_qc_face_script.pl -profile prod -out_basedir
/data/preventad/data/deface_qc/deface_qc_`date` < list
```

- ❏ QC all the images to make sure the defacing script worked on all subjects (Note: this can be done by anyone as long as they have access to JPEG files)

What to do if the defacing failed on a session

If the defacing failed on all modalities of a given session, it is possible that the linear registration step failed. In that case, try in the following order:

1) Rerun the defacing pipeline by trying a different version of the `bestlinreg.pl` tool
    - ❏ Modify the script `uploadNeuroDB/bin/deface_minipipe.pl` so that it calls bestlinreg.pl instead of bestlinreg_claude.pl
    - ❏ Run the defacing pipeline on the session ID that failed defacing the first time
    - ❏ QC the new images
    - ❏ Don't forget to revert the changes to `uploadNeuroDB/bin/deface_minipipe.pl` as `bestlinreg_claude.pl` has a lower failure rate than `bestlinreg.pl`
2) If modifying the version of `bestlinreg.pl` did not work, Dr. Louis Collins suggests to perform manual registration.

Appendix 6: Backup and remove non-defaced images

Note: This step requires at least LORIS release 21.0.0

This can be done by using the `tools/delete_imaging_upload.pl` script.

1) What does the delete script do?

It will back up the SQL entries and the non-defaced files before removing them from the dataset (after having checked that the non-defaced file has an associated defaced file). Note that the `TarchiveSource` field of the defaced entry in the `files` table will be populated with the `TarchiveSource` of the non-defaced files (allowing for QC mapping if needed).

If one day those non-defaced files need to make their way back into LORIS, one would just need to run the SQL patches and extract the archives with the backed up non-defaced files.

2) To run `delete_imaging_upload.pl`:

- ❏ Get the list of all upload IDs on which the script must be run (`list_uploadID`)
- ❏ Run the delete script with the option `-defaced`

```
cat list_uploadID | while read f; do perl delete_imaging_upload.pl -profile
prod -uploadID $f -defaced; done &> log_defacing_`date`
```

Important note: do not run this script in parallel as the script locks the database while it is running

3) Sanity checks after deletion

- ❏ Ensure that all non-defaced images have been removed from the open LORIS instance
- ❏ Ensure the `TarchiveSource` field was populated for the defaced images so that QC can be mapped afterwards (see appendix 7)

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 7: Map MRI QC information

Note: the `DicomArchiveID`/`SeriesNumber` combination will act as the `SeriesUID`

Steps overview:

In the original database:

1) Ensure that `TarchiveSource` is populated for native files
2) Create a table with `FileIDs`, `DicomArchiveID`, `EchoTime` & `SeriesNumber`
3) Create tables with QC information, `DicomArchiveID`, `EchoTime` & `SeriesNumber`
4) Export tables from the original database

In the open database:

5) On the open VM, repeat step 2
6) Import the tables created in the original database to map the QC information
7) Insert QC status & MRI Comments with the proper FileIDs
8) Map SessionID comments
9) Remove all possible identifying information from the text comments

1) Ensure that TarchiveSource is populated for native files

- ❏ Create a temporary table with the `ArchiveLocation` stored in `parameter_file` so that we can modify the `ArchiveLocation`'s path to match the one stored in `tarchive`

```
CREATE TEMPORARY TABLE archive_locations AS
  SELECT FileID, TarchiveSource, Value AS ArchiveLocation
  FROM files
    JOIN parameter_file USING (FileID)
   JOIN parameter_type USING (ParameterTypeID)
  WHERE Name="tarchiveLocation" AND TarchiveSource IS NULL;

UPDATE archive_locations SET ArchiveLocation=REPLACE(archivelocation,
'tarchive/', '');
```

- ❏ Create another temporary table with all the fields from the first temporary table and the `TarchiveID` matching the archive location that was stored in `parameter_file`

```
CREATE TEMPORARY TABLE archive_locations_with_tarchiveID AS
  SELECT archive_locations.*, TarchiveID
  FROM archive_locations JOIN tarchive USING (ArchiveLocation);
```

- ❏ Update the `files` table with the `TarchiveID` stored in the last temporary table

```
UPDATE files f, archive_locations_with_tarchiveID alwt SET
f.TarchiveSource=alwt.TarchiveID WHERE f.FileID=alwt.FileID;
```

2) Create a table with FileIDs, DicomArchiveID, EchoTime & SeriesNumber

❏ Create a table with the original `FileID`, `EchoTime` & `DicomArchiveID` (a.k.a. `StudyUID`)

```
CREATE TEMPORARY TABLE map_fileID_DicomArchiveID_EchoTime AS
  SELECT f.FileID, DicomArchiveID, f.EchoTime
  FROM files f
    JOIN tarchive t ON (t.TarchiveID=f.TarchiveSource);
```

❏ Create a table with `FileID` and the `SeriesNumber` from `parameter_file`

```
CREATE TEMPORARY TABLE map_fileID_SeriesNumber AS
  SELECT FileID, Value AS SeriesNumber
  FROM parameter_file
    JOIN parameter_type USING (ParameterTypeID)
  WHERE Name="series_number";
```

❏ Create a final table `FileID`, `EchoTime`, `DicomArchiveID` and `SeriesNumber` that can be used later on to map QC and feedback information

```
CREATE TABLE map_fileID_DicomArchiveID_SeriesNumber_EchoTime AS
  SELECT mfs.FileID, DicomArchiveID, SeriesNumber, EchoTime
  FROM map_fileID_DicomArchiveID_EchoTime
    JOIN map_fileID_SeriesNumber mfs USING (FileID);
```

3) Create tables with QC info, DicomArchiveID, EchoTime & SeriesNumber

❏ Create a table with QC info linked to `EchoTime`, `DicomArchiveID` & `SeriesNumber`

```
CREATE TABLE files_qcstatus_DicomArchiveID_SeriesNumber_EchoTime AS
  SELECT FileQCID, fq.FileID, QCStatus, QCFirstChangeTime,
QCLastChangeTime, Selected, DicomArchiveID, SeriesNumber, mfdse.EchoTime
  FROM files_qcstatus fq
    JOIN map_fileID_DicomArchiveID_SeriesNumber_EchoTime mfdse USING
(FileID);
```

❏ Create a table with feedbacks linked to `EchoTime`, `DicomArchiveID` & `SeriesNumber`

```
CREATE TABLE feedback_mri_comments_DicomArchiveID_SeriesNumber_EchoTime AS
  SELECT CommentID, fmc.FileID, SessionID, CommentTypeID,
PredefinedCommentID, Comment, ChangeTime, DicomArchiveID, SeriesNumber,
mfdse.EchoTime
  FROM feedback_mri_comments fmc
    JOIN map_fileID_DicomArchiveID_SeriesNumber_EchoTime mfdse USING
(FileID);
```

4) Export tables from the original database

- ❏ Export tables from the original database

```
mysqldump -u $USER -h $HOST $DATABASE
files_qcstatus_DicomArchiveID_SeriesNumber_EchoTime -p >
files_qcstatus_DicomArchiveID_SeriesNumber_EchoTime.sql
mysqldump -u $USER -h $HOST $DATABASE feedback_mri_comment_types -p >
feedback_mri_comment_types.sql
mysqldump -u $USER -h $HOST $DATABASE feedback_mri_predefined_comments -p >
feedback_mri_predefined_comments.sql
mysqldump -u $USER -h $HOST $DATABASE
feedback_mri_comments_DicomArchiveID_SeriesNumber_EchoTime -p >
feedback_mri_comments_DicomArchiveID_SeriesNumber_EchoTime.sql
```

5) On the open VM, repeat step 2

Note: the name of the created tables should be different than the ones created on the original database so that the import in step 6 do not overwrite the tables created in step 5.

6) Import the tables created in step 4 to map the QC info

- ❏ Import tables into the open VM (tested on the dev open VM)

```
source ~/files_qcstatus_DicomArchiveID_SeriesNumber_EchoTime.sql
source ~/feedback_mri_comment_types.sql
source ~/feedback_mri_predefined_comments.sql
source ~/feedback_mri_comments_DicomArchiveID_SeriesNumber_EchoTime.sql
```

7) Insert QC status & MRI Comments with the proper `FileID`s

- ❏ Insert into `files_qcstatus` table the QC status

```
CREATE TEMPORARY TABLE FileIDs AS
  SELECT FileQCID, mdse.FileID, QCStatus, QCFirstChangeTime,
QCLastChangeTime, Selected, DicomArchiveID, SeriesNumber, EchoTime
  FROM files_qcstatus_DicomArchiveID_SeriesNumber_EchoTime fqdse
    JOIN map_fileID_DicomArchiveID_SeriesNumber_EchoTime mdse USING
(DicomArchiveID, SeriesNumber, EchoTime);

INSERT INTO files_qcstatus (FileQCID, FileID, QCStatus, QCFirstChangeTime,
QCLastChangeTime, Selected)
  SELECT FileQCID, FileID, QCStatus, QCFirstChangeTime, QCLastChangeTime,
Selected FROM FileIDs;
```

- ❏ Insert into `feedback_mri_comments` table the MRI comments

```
CREATE TEMPORARY TABLE FeedbackIDs AS
  SELECT CommentID, mdse.FileID, SessionID, CommentTypeID,
PredefinedCommentID, Comment, ChangeTime, DicomArchiveID, SeriesNumber,
EchoTime
  FROM feedback_mri_comments_DicomArchiveID_SeriesNumber_EchoTime fmcdse
    JOIN map_fileID_DicomArchiveID_SeriesNumber_EchoTime mdse USING
(DicomArchiveID, SeriesNumber, EchoTime);

INSERT INTO feedback_mri_comments (CommentID, FileID, SessionID,
CommentTypeID, PredefinedCommentID, Comment, ChangeTime)
  SELECT CommentID, FileID, SessionID, CommentTypeID, PredefinedCommentID,
Comment, ChangeTime FROM FeedbackIDs;
```

- ❏ Remove date information from the `feedback_mri_comments` table

```
 UPDATE feedback_mri_comments SET ChangeTime=CURRENT_TIMESTAMP;
```

- ❏ Remove date information from the `files_qcstatus` table

```
UPDATE files_qcstatus SET QCFirstChangeTime=NULL;
UPDATE files_qcstatus SET QCLastChangeTime=NULL;
```

8) Map `SessionID` comments

- ❏ Some QC information are in the `session` table and can be imported using John Saigle's data import/export scripts on the behavioural side (see Appendix 13).
- ❏ Session level feedback are located in the `feedback_mri_comments` tables. However, should be careful as to not import identifying information as those feedback are an open text field. This should be dealt with on a case by case.

9) Remove all possible identifying information from the text comments

- ❏ Look at the `Comment` field of `feedback_mri_comments` and make sure that there is no comments that might be identifying (incidental findings, doctor names etc…)

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 8: Insertion of Lego Phantom datasets

To be developed...

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 9: Make sure DICOM files are not accessible

Since DICOM datasets are not defaced, there is a need to ensure that the DICOM files are not accessible via the frontend or the API. In order to do so, ensure to:

- ❏ Modify the permissions of `/data/loris/data/tarchive` and `/data/loris/data/trashbin` folders so that they cannot be accessed by Apache
- ❏ Remove the download links for the DICOM archive and MRI violated scans (in the DICOM archive, Imaging Browser and MRI violations modules)
- ❏ Remove the download DICOM archive portion of the API

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 10: Upgrade to the latest LORIS release

The upgrade to the latest LORIS release includes but is not limited to:

- ❏ Syncing frontend data tables with data from customized queries (to include specific fields that would not be present in standard LORIS data tables for example) along with menu filters
- ❏ Customize metadata fields (for example: Candidate DoB, Gender, Within Optimal, Within Permitted information when accessing candidate/timepoint. May be prohibited/irrelevant for the project)
- ❏ Remove customized modules (Note: ideal approach might be to start with no modules and only add/customize the necessary ones)
- ❏ Customize LorisMenu based on the modules to be accessed (see point above)
- ❏ Customize and fix broken queries
- ❏ Customize and fix broken scripts
- ❏ Customize SQL tables - extra tables or modified columns
- ❏ Incorporating bugs that were fixed after the latest release

Appendix 11: Customize Battery class

Some studies may want to exclude populating certain visits, or some may want to exclude populating certain instruments for certain visits etc… The battery class can be a good place to handle these kinds of restrictions.

The handling of DDE can also be done here if needed although it is highly probable that there will be no DDE for open LORIS.

Appendix 12: Insertion of demographic information

Use the scripts written by John Saigle based on LORIS 20.1 release to populate Gender on the open LORIS instance: https://github.com/johnsaigle/Loris/pull/15.

Reminder: only identified variables should be inserted (dates, registeredBy and other identifiable fields should not be copied over).

- ❏ Extract the gender information from the original database

```
php dataExporter.php candidate Gender [date]
```

- ❏ Run the `OpenScienceDataImporter.php` script to create the SQL statements needed to update the `candidate` table (Note: `IDmapping.csv` contains the PSCID mapping between the original database and the new open database `data.csv` contains the Gender information with the original PSCID)

```
php OpenScienceDataImporter.php candidate IDmapping.csv data.csv
```

- ❏ Source the SQL statements produced by `OpenScienceDataImporter.php`
- ❏ Sanity checks: make sure that all the information was properly inserted (for example, all candidates have Gender information populated, the number of candidates in the candidate table matches the number of candidates to be shared etc…)

Note: if using LORIS release 21.0 and later, the Gender field has been renamed to Sex, therefore, the scripts created by John Saigle might need to be updated.

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 13: Creation of visit labels in the new LORIS instance

Use the scripts written by John Saigle (https://github.com/johnsaigle/Loris/pull/15) based on LORIS release 20.1 to populate the fields of the `session` table on the open LORIS instance (including MRI session level QC status mentioned in Appendix 7).

Reminder: only identified variables should be inserted (dates, registeredBy and other identifiable fields should not be copied over).

- ❏ Extract the `session` information from the original database

```
php dataExporter.php visits [date]
```

- ❏ Run the `OpenScienceDataImporter.php` script to create the SQL statements needed to update the `session` table (Note: `IDmapping.csv` contains the PSCID mapping between the original database and the new open database `data.csv` contains the session information with the original PSCID)

```
php OpenScienceDataImporter.php visits IDmapping.csv data.csv
```

- ❏ Source the SQL statements produced by `OpenScienceDataImporter.php`
- ❏ Sanity checks: run a few sanity checks to ensure all data were properly imported (no missing information, correct number of visits etc…)

MCIN MCGILL CENTRE for INTEGRATIVE NEUROSCIENCE

Appendix 14: Insertion of instrument fields in the new LORIS

Use the scripts written by John Saigle (https://github.com/johnsaigle/Loris/pull/15) based on LORIS release 20.1 to populate the instrument tables on the open LORIS instance.

Reminder: only identified variables should be inserted (dates, registeredBy and other identifiable fields should not be copied over).

- ❏ Extract the instrument information from the original database

```
php dataExporter.php instrument RBANS col1,col2,col3 [date]
```

- ❏ Run the `OpenScienceDataImporter.php` script to create the SQL statements needed to insert in the instrument table (Note: `IDmapping.csv` contains the PSCID mapping between the original database and the new open database `data.csv` contains the instrument information with the original PSCID)

```
php OpenScienceDataImporter.php instrument IDmapping.csv data.csv
```

- ❏ Source the SQL statements produced by `OpenScienceDataImporter.php`
- ❏ Sanity checks: run a few sanity checks to ensure all data were properly imported (no missing information, correct number of instruments, numbers should line up in terms of visits, flags, instruments, scores all imported etc…)

Appendix 15: Setup the API

- ❏ In the “API Keys” section of the Config module, set up a valid JWT Secret Key so that the API can be used for the open VM
- ❏ In the API code, remove all POST/PUT requests (except for the login endpoint) so that the data cannot be modified via the API.
- ❏ In the API code, remove DICOM images download as mentioned in Appendix 9