NiDB Pipeline Guide

August 2015

Greg Book - [gregory.a.book@gmail.com](mailto:gregory.a.book@gmail.com)

Contents

[1 NiDB Pipelines 3](#_Toc427329948)

[1.1 Definitions 3](#_Toc427329949)

[1.2 Viewing Pipelines 3](#_Toc427329950)

[1.3 Create a Pipeline 4](#_Toc427329951)

[1.3.1 New Pipeline 4](#_Toc427329952)

[1.3.2 Copy Existing pipeline 9](#_Toc427329953)

[1.4 Modify a pipeline 9](#_Toc427329954)

[1.5 Analyses 10](#_Toc427329955)

[1.5.1 Understanding the Analysis Process 10](#_Toc427329956)

[1.5.2 Data Selection 11](#_Toc427329957)

[1.5.3 Analysis Directory Structure 11](#_Toc427329958)

[1.5.4 Analysis Listing 12](#_Toc427329959)

[1.6 Example Pipelines 13](#_Toc427329960)

[1.6.1 Freesurfer 13](#_Toc427329961)

[1.6.2 FSL fMRI pipelines 13](#_Toc427329962)

[1.7 Checking Pipeline Logs 14](#_Toc427329963)

[1.8 Troubleshooting 14](#_Toc427329964)

[1.8.1 Why Didn’t the Pipeline Do Anything? 14](#_Toc427329965)

[1.8.2 Why did only some studies start processing? 15](#_Toc427329966)

[2 Examples 15](#_Toc427329967)

[2.1 Example 1 – Analyze all Go NoGo data (repeating series names) 15](#_Toc427329968)

[2.2 Example 2 – Analyze all Go NoGo data (unique series name) 17](#_Toc427329969)

[2.3 Example 3 – Analyze all Go NoGo data, but add in T1 and fieldmaps 18](#_Toc427329970)

[2.4 Example 4 – Analyze DTI data (keeping series number) 20](#_Toc427329971)

[2.5 Example 5 – Analyze Go NoGo, T1 from different study 21](#_Toc427329972)

[2.6 Example 6 – Analyze similar protocol names 23](#_Toc427329973)

# NiDB Pipelines

The NiDB pipeline is designed to allow you to process data automatically, and with little behind the scenes intervention. It is best used when you have an established command-line based analysis that you want to run on a large batch of data, or you want to run it automatically each time new data arrives on NiDB.

## Definitions

* **Pipeline** – A script that is run on a set of data. Also includes data specifications, dependencies, groups, and compute cluster options.
* **Pipeline Group** – A set of related pipelines. They may all be separate steps of a larger process.
* **Dependency** – A pipeline may depend on another pipeline for results that it requires for its analysis. An example is an fMRI stats pipeline depending on an fMRI preprocessing pipeline.
* **Descendent pipeline** – A pipeline that depends on another pipeline
* **Level 0** – a pipeline that does not operate on any data
* **Level 1** – study/subject level pipeline. This pipeline will download data as part of the analysis
* **Level 2** – second/stats level pipeline that depends on one or more first level pipelines. No data is downloaded as part of a level 2 pipeline
* **Ignored studies** – As a pipeline scans the database, looking for matching data, it will mark non-matching studies so they are not checked again. If you do something like change the data criteria, then some of the ignored studies may get picked up to run, but you must clear the ignore flag.
* **Analysis** – The result of a pipeline running against an imaging study
* **Primary data** – The data criteria on which studies are search first.
* **Associated data** – will only be download if the primary data is found.

## Viewing Pipelines

To view the pipeline page, click the **Analysis** tab. This will display a list of the currently enabled pipelines. By default it will only display the visible pipelines, if you want to view the hidden pipelines, click the **View**: **Normal** link at the top left of the page.

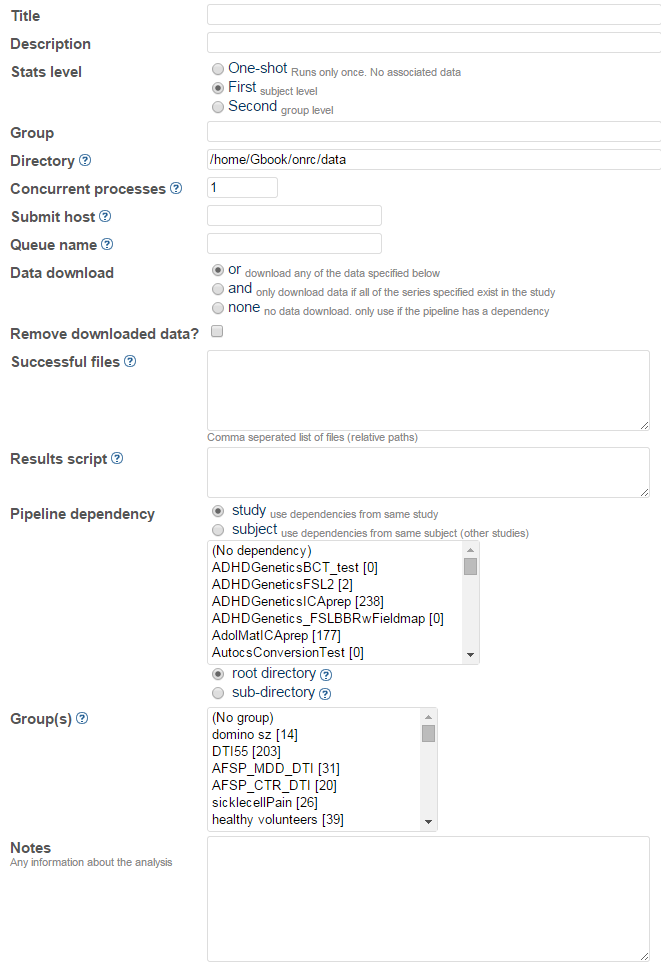


Mousing over the rows will give more information about the item. The first column contains the pipeline group, and when mousing over the label, you will see a graph describing the relationships between the pipeline, its dependencies and the study groups going into them. The second column displays the pipeline, and any descendent pipelines, along with their version numbers. Click the pipeline name to view the pipeline details. The next columns display the pipeline level (0,1,2) and the owner. The **Status** column shows the status and a checkbox to enable/disable the pipeline. If the pipeline is enabled, it will run and process analyses, otherwise it will not run. The **reset** link can be clicked if you absolutely know that a pipeline has stopped processing. The Analysis column displays the number processing and number complete. Click the http://ado2:8080/images/preview.gif icon to view the analysis list for that pipeline. The Disk size column lists the total space used by the analyses. This total includes any data from dependent (parent) pipelines, which are either hard- or soft-linked, and may be higher than the actual disk usage. Finally, the last two columns show the data path and the SGE queue name.

## Create a Pipeline

### New Pipeline

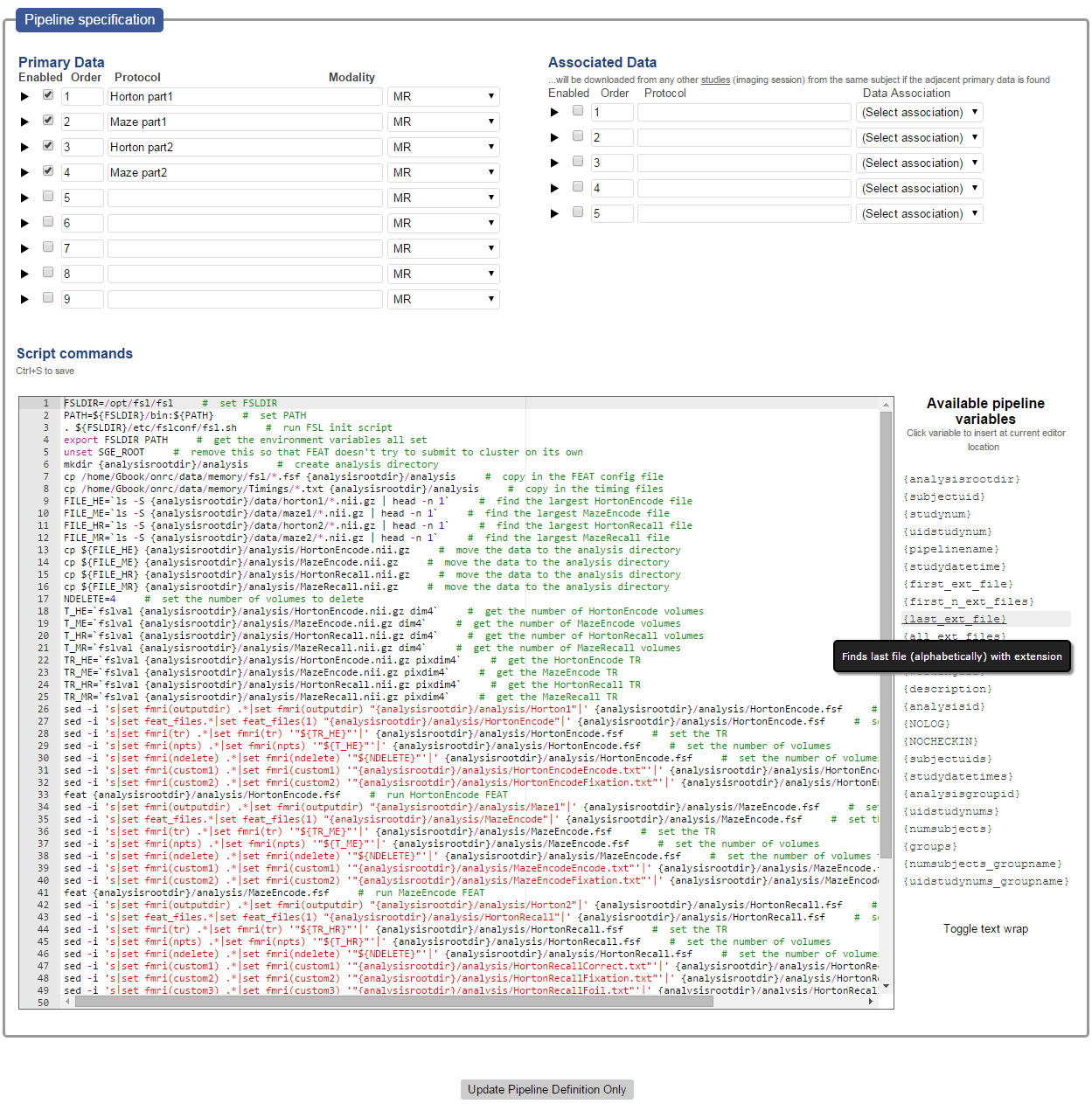
At the top of the pipeline page, click the **New Pipeline** link. You’ll see a pipeline page with many options. Some options have descriptions and help bubbles.



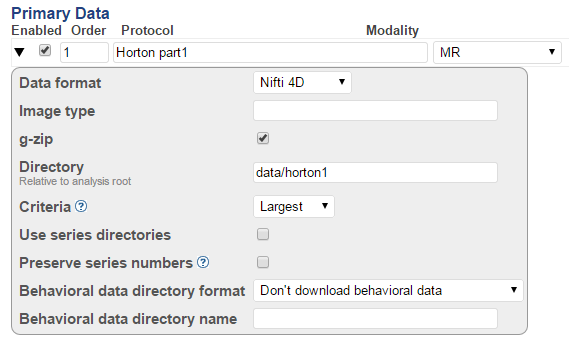
Some descriptions of the pipeline options

* **Title** - an alphanumeric string, with no spaces or special characters. This will become the directory in which an analysis is done. Once the pipeline is created, it cannot be changed.
* **Description** – longer description of the pipeline
* **Stats level** – Corresponds to the levels 0,1,2
* **Group** – (optional) to associate several pipelines together
* **Directory** – Where you want the data to go. If blank, it will be placed the default pipeline analysis directory
* **Concurrent processed** – The total number of analysis jobs allowed to running at the same time. This is a different number than the queue limits in SGE
* **Submit host** – If you have access to more than one cluster, there may be different submit hosts
* **Queue name** – The SGE queue name
* **Data download** – specifies how to interpret the data specification
* **Remove downloaded data** – doesn’t really work, don’t use it
* **Successful files** – a list of files that should exist at the end of the analysis. If they all exist, the analysis is marked complete.
* **Results script** – a script that is run at the end of the pipeline that ports results back into the database
* **Pipeline dependency**
  + **Study** – dependencies will only be used from the same study. For example an fMRI preprocessing pipeline is run on S1234ABC1, so this option will only check the same study for a dependency when running this pipeline
  + **Subject** – this will check the same subject, regardless of study for a dependent pipeline
  + **Dependency list** – the pipeline on which this pipeline depends. Can be multiple. Ctrl+click to select multiple
  + **Root directory** – this will place the dependency(s) in the root directory
  + **Sub**-directory – this will place the dependency(s) in subdirectories of the same name as the dependency
* **Group(s)** – the groups which will be part of this pipeline. This is ONLY study level groups
* **Notes** – extra notes about the pipeline

When you’ve filled out all the information, click **Add Pipeline Info**. This will create the basic pipeline, and the information that generally does not change. On the pipeline list, click your newly made pipeline. Click the pipeline to view it. At the bottom you’ll see a **Pipeline specification** section.



On the upper left is the primary data specification, the upper right the associated data specification, and on the bottom the pipeline script. Clicking the arrows next to each row of the data specifications will display more data options.



Description of the options

* **Enabled** – whether the data criteria will be used or not
* **Order** – change these numbers to reorder the download list
* **Protocol** – This is an exact protocol name. If you list a single protocol name, it will look for only that name. If you use a list like [“Task1” “Task 1” “task one”] it will search for any of those protocol names.
* **Directory** – relative to the analysis root directory
* **Criteria**
  + **All** – all matching series will be downloaded
  + **First** – only the lowest numbered matching series will be downloaded
  + **Last** – only the highest numbered matching series will be downloaded
  + **Largest** – Only one series with the most number of volumes or slices will be downloaded
  + **Smallest** – Only one series, with the least number of volumes or slices will be downloaded
* **Use series directories** – Will place the data in multiple numbered directories, with the series number
* **Preserve series numbers** – if the series directories are used, it will retain the original series number, otherwise it will start the numbering at 1

Below that are the script commands. This is a simple bash script with embedded pipeline variables. These variables (list on the left) are converted to real values when the script is run on each analysis. The bash script is a modified interpretation of bash, it will not run exactly as listed in the script window. Variables are replaced, check-in points are added, and some output is redirected to log files. Trailing semi-colons are also removed from each line.

If you use a loop, each line of the loop should contain the {NOLOG}{NOCHECKIN} variables in the comments, which will prevent the insertion of check-in code or logging, which may interrupt the loop. Each line should have a comment, even an empty comment. If you do any kind of output redirection using > or >>, make sure to include the {NOLOG} variable. If you use a command such as find, which requires a trailing semi-colon, add a second semi-colon to the end of it.

#### Script Variables

Variables are listed to the right of the script area, and clicking the variable will insert it into the script at the current cursor location. Here’s a description and example of the script variables.

|  |  |  |
| --- | --- | --- |
| Variable | Description | Example |
| {analysisrootdir} | Analysis root directory. The absolute path to the analysis directory | /path/to/the/pipelines |
| {subjectuid} | The UID of the subject being analyzed | S1234ABC |
| {studynum} | The study number of the being analyzed | 2 |
| {uidstudynum} | UID and study number | S1234ABC2 |
| {pipelinename} | The name of the pipeline | fMRIPreProc |
| {studydatetime} | Date/time of the start of the study. YYYYMMDDHHMISS | 20150323164523 |
| {first\_ext\_file} | Tries to find the first file with the extension “ext”, replace ext with the extension you want to find. You would want this to be part of a path, and that path and files must exist before the analysis is run | /path/to/{first\_nii\_file}  becomes …  /path/to/S1234ABC1\_001.nii |
| {first\_n\_ext\_files} | Same as above, but finds the first n files with extension | /path/to/{first\_2\_nii\_files}  becomes …  /path/to/S1234ABC1\_001.nii  /path/to/S1234ABC1\_002.nii |
| {last\_ext\_file} | Save as above, but finds last file with extension | /path/to/{last\_nii\_file}  becomes …  /path/to/S1234ABC1\_098.nii |
| {all\_ext\_files} | Save as above, but finds all files with extension | /path/to/{all\_nii\_files}  becomes …  /path/to/S1234ABC1\_001.nii  /path/to/S1234ABC1\_002.nii  …  /path/to/S1234ABC1\_097.nii  /path/to/S1234ABC1\_098.nii |
| {command} | The script command for that line, excluding the comment | feat fmritask.fsf |
| {workingdir} | No idea what this does |  |
| {description} | The comment. The text after the hash (#) character | This is a comment |
| {analysisid} | A unique, but not random number identifying the analysis. Necessary for re-inserting results back into the database | 2354199 |
| {NOLOG} | Will not append a > stepXX.log to the end of the line |  |
| {NOCHECKIN} | Will not prepend a line to checkin |  |
| {subjectuids} | Creates a space separated list of the UIDs involved in the analysis. Will only be used if there are groups | S1223ABC S2349GJH S3589FKW … |
| {studydatetimes} | Creates a space separated list of the study datetimes | 20150304184523 20130304124523 … |
| {analysisgroupid} | The database ID of the analysis group | 23 |
| {uidstudynums} | List of the UID/study numbers | S1359DJJ1 S3952VNS2 S7842WIT1 … |
| {numsubjects} | Number of subjects in the group(s) in total | 345 |
| {groups} | List of groups that the pipeline depends on | fMRIPreProc fMRIStats1 fMRIStats2 |
| {numsubjects\_groupname} | Number of subjects for the specified group | {numsubjects\_fMRIPreProc}  32 |
| {uidstudynums\_groupname} | List of UID/study numbers for the specified group | {uidstudynums\_fMRIPreProc}  Becomes…  S1359DJJ1 S3952VNS2 S7842WIT1 |

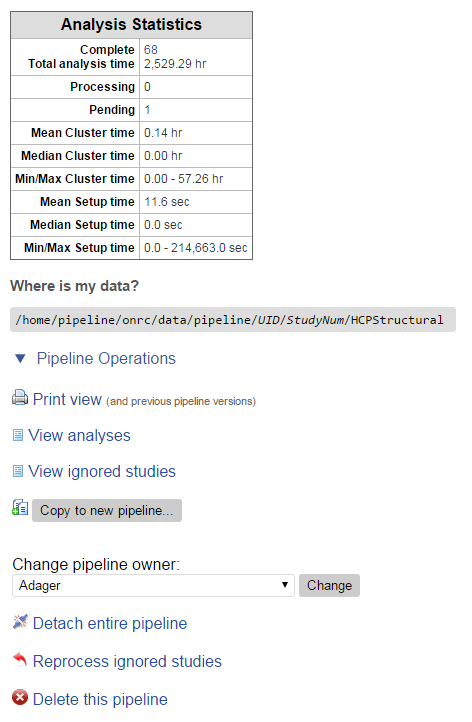
### Copy Existing pipeline

To copy an existing pipeline into a new pipeline, go to the pipeline’s page, and click the **Pipeline Operations** arrow, then click the http://ado2:8080/images/copy16.gif **Copy to new pipeline…** button. It will prompt you for a new pipeline name.

## Modify a pipeline

When clicking on the pipeline name on the main pipeline page, you will be brought to the pipeline’s page. You will be able to modify the pipeline (if you are the owner or administrator) on this page. Items that cannot be changed are gray out. The pipeline is separated into two section 1) the main pipeline details 2) the data criteria and pipeline script. Only the data criteria and script are versioned; so when you save the script or data criteria, a new version will be saved, and old version can be viewed. This is useful when you make a change and suddenly it’s not working anymore. The versioning allows you to check a recently working copy of the script.

At the top of the pipeline page, you’ll see a box with the Analysis Statistics and below that a box for Pipeline Operations.



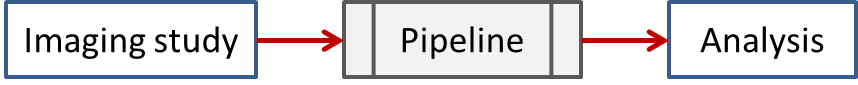
Description of the options:

* **Print view** – View the current (and previous versions) of the pipeline in a print form
* **View analyses** – Displays a list of the current analyses
* **View ignored studies** – Displays a list of imaging studies which were included in the search for data, but were subsequently not run because the data criteria did not match
* **Copy to new pipeline** – Copies this pipeline to a new one with a newly specified name
* **Change pipeline owner** – Changes the owner of the pipeline to another user
* **Detach entire pipeline** – removes the pipeline from the control of the NiDB pipeline system and moves the data to a specified directory (Not Implemented).
* **Reprocess ignored studies** – For any studies which were ignored, this option will try them again to see if they match the data criteria.
* **Delete this pipeline** – This will delete the pipeline and any data associated with it. Dependent or descendent pipelines will not be deleted.

## Analyses

### Understanding the Analysis Process

NiDB pipelines operate on single imaging studies, so that a pipeline is applied to an imaging study and the outputs are associated with that imaging study. In other words the analysis and the imaging study are on the same granular level.



### Data Selection

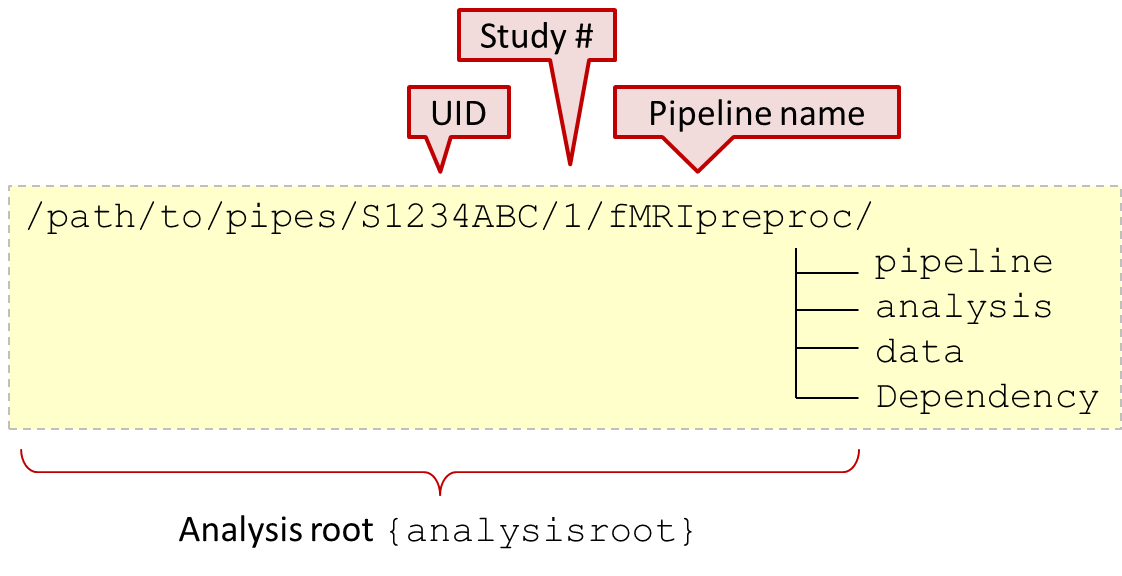
The first step to check which imaging studies will be run through the pipeline starts with the dependencies and groups. Only studies which are part of the specified group (or all studies in the database if there is no group specification) and which have the specified dependency(s) (or all studies is there are no dependency specifications) will be checked. After a list of possible studies is selected, each one is checked against the data criteria. Within each imaging study are several series, each with a protocol name, and these are the data that checked for inclusion in the analysis.

In the data specification, there are two parts 1) primary data 2) associated data. When the pipelines are searching for matching data to determine whether to run the pipeline on that study, they are only looking at the primary data. So if the primary data criteria are met, then the study is analyzed. If there is an associated data spec, then the associated data is also searched for (from studies belonging to the same subject), but if it is not found, the analysis will still run.

For primary data, there is an option to require all of spec, or any of it. In other words if you select “and”, the pipeline will only run on the study if the study contains all of the data criteria specified. If you select “or”, the pipeline will run on the study if the study contains any of the data specified.

### Analysis Directory Structure

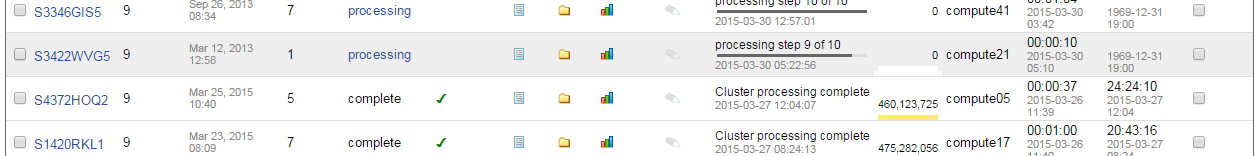
The following figure describes the analysis directory structure.



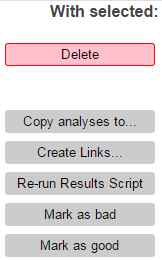
The {analysisroot} variable is considered the root the analysis and should be used in scripts to specify absolute paths. Your pipeline should be assumed to have access to all directories including and below the analysisroot directory. The pipeline subdirectory is always created and stores log files, SGE job files. The analysis directory is optional, but is very useful to contain an analysis. The data directory is optional as well, but is often created when a data directory is specified in the data specification. The Dependency directory is created if the pipeline has a dependency. Any copied (or linked) data will be placed in the Dependency directory.

### Analysis Listing

When viewing the analysis listing page, you’ll be able to see the currently processing and completed analyses. If there a lot of analyses (more than 5000) it will be divided onto multiple pages. You can navigate those pages using the navigation links on the top left.  The analysis listing provides a lot information right on the page, through mouseover messages, or through dialog boxes when clicking icons.



The first column contains a link to the imaging study. Second column contains the pipeline version that was run on that analysis. Next columns contain the imaging study date, and the number of series downloaded. Then the analysis status: processing, pending, or complete. Clicking on a processing link will show the current SGE status for the running job. A green checkmark  indicates the analysis is complete because the file(s) specified in the pipeline for a successful analysis exist. The next three icons allow viewing of analysis details in dialog boxes, without leaving the page. The http://ado2:8080/images/preview.gif icon will display the logs associated with the analysis. The  icon will display a file/directory listing of everything in the analysis root directory. The  icon will display any results that were associated with the analysis. The pencil icon allows you to add notes to each analysis. Click the icon to add or edit notes. The most recent status message and date are next, followed by the disk size of the completed analyses. The hostname of the server the analysis was run is next. This can be helpful in diagnosing random failed analyses: ie, it might be related to one particular incorrectly configured server. Then the analysis setup time, which is basically the time it took to copy in the data. Last is the compute time, this the span of time between when the analysis was first started by the SGE, and when it ended. The right-most column contains checkboxes. Select specific analyses to perform the following options (buttons at bottom of page).



* **Delete** – Completely deletes the analysis.
* **Copy analyses to**… - Will copy the analyses to a specified directory, leaving the original copy
* **Create links** – Will create a directory containing links to the selected data
* **Re-run Results Script** – Will rerun the results script specified in the pipeline
* **Mark as bad** – Will mark the analyses as bad, so **descendent pipelines will not use the analyses**
* **Mark as good** – will unmark bad analyses

## Example Pipelines

Below is a sampling of pipelines that demonstrate the nuances of the pipeline bash format and also the generic quality of the programming necessary to accommodate any data that passes through the pipeline.

### Freesurfer

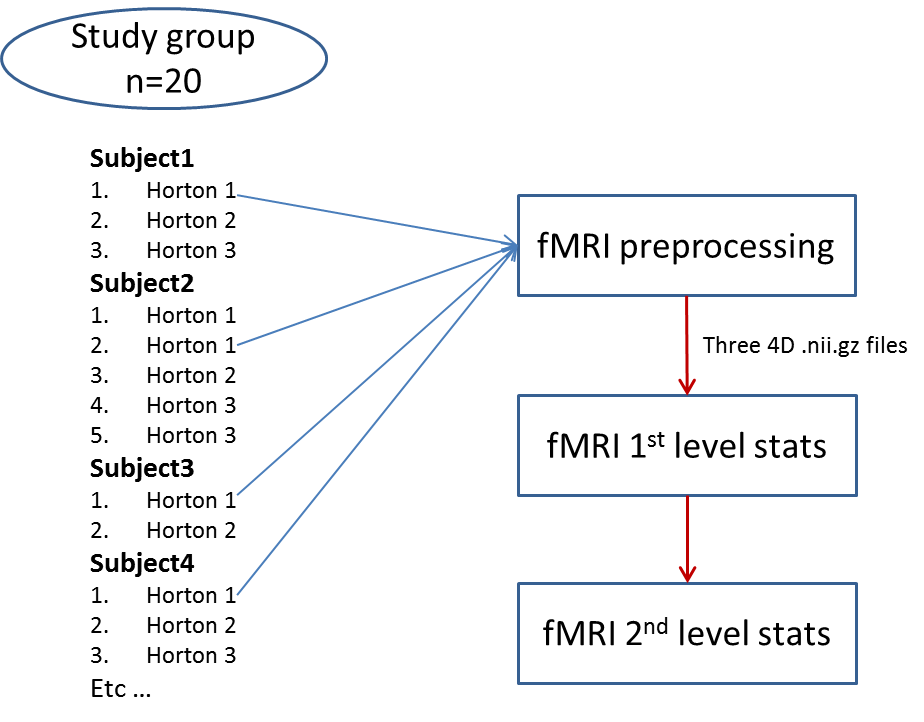
This simple pipeline will take any type and number of T1 images, and pass them through freesurfer’s recon-all process.

|  |  |
| --- | --- |
| **Data** | Multiple T1 images, placed in {analysisroot}/data |
| **Script** | 1 export FREESURFER\_HOME=/opt/freesurfer # Freesurfer home directory  2 export FSFAST\_HOME=/opt/freesurfer/fsfast # ENV variables  3 export MNI\_DIR=/opt/freesurfer/mni # ENV variables  4 source $FREESURFER\_HOME/SetUpFreeSurfer.sh # setup Freesurfer to run  5 export SUBJECTS\_DIR={analysisrootdir} # all FS data will go here  6 freesurfer > {analysisrootdir}/version.txt # {NOLOG} get the freesurfer version  7 perl /opt/autoCS/ImportFreesurferData.pl {analysisrootdir}/data analysis # import data  8 recon-all -clean-tal -notal-check -mprage -no-isrunning -autorecon1 -subjid analysis #  9 recon-all -mprage -no-isrunning -autorecon2 -subjid analysis # Autorecon 2  10 recon-all -mprage -no-isrunning -autorecon3 -subjid analysis # Autorecon 3 |

This script first sets up the environment variables required to find freesurfer and those variables required by freesurfer. Freesurfer requires a SUBJECTS\_DIR which contains the subjects to be processed. This directory is expected to contain a bunch of other directories which would contain their respective subject data. Freesurfer also expects data to be in a specific format prior to running it. Since this analysis only operates on one subject, the SUBJECTS\_DIR directory will have only one directory, and for simplicity, it’s called ‘analysis’. This implies the Subject ID is analysis, and so that ID is passed to the freesurfer calls. The perl script called on line 7 will take the raw data in the data directory and put it into the format expected by freesurfer in the analysis directory.

### FSL fMRI pipelines

This is an example of a dependent pipeline containing a preprocessing step, first level stats, and second level stats. Limited code will not be provided, so we can focus on the concepts of how dependent pipelines work. Let’s start with the simplest case, you have a group of 20 subjects who did an fMRI task called Horton. Each subject should have 3 runs, labeled Horton 1, Horton 2, Horton 3. In this example, the data criteria will be to select the largest (most number of BOLD reps) for each protocol name. Subject 1 is normal, subject 2 has repeated runs, and subject 3 is missing the third run. All of these go into the preprocessing pipeline. The output of the preprocessing pipeline is a 4D preprocessed .nii.gz file, or in the case of this whole pipeline setup: 3 .nii.gz files, one for each run. Each subject will have 3 .nii.gz files.



## Checking Pipeline Logs

This is only available to system administrators of NiDB. Under the Admin tab, go to Modules. Click the pipeline module, and you’ll see a list of the recent log files. If you suspect the pipeline module has failed, you may see an error message at the end of any of the log files. Multiple instances of the pipeline module normally run at the same time, so while one instance may have failed, others may continue to run. You can also see the progress and other debugging information in the log files.

## Troubleshooting

### Why Didn’t the Pipeline Do Anything?

If you created the pipeline, it has data, it has a script, and its enabled, you expect it to run… but it’s not, nothing at all is happening. It’s just sitting there, for an hour, taunting you. It could be many things. If nothing is happening, check these things one by one.

* Are the SGE submit node and queue name valid
* Does the pipeline directory exist, and is it writeable. Especially if you are not using the default pipeline directory
* Is the # of concurrent processes at least 1
* Does the group you selected have any studies in it
* Does the dependency you selected have any analyses in it
* Has it been 6 hours since the data was collected (the pipeline will only process studies older than 6 hours)

Most problems are related to the data criteria, so check the data criteria carefully against the imaging studies you are trying to process.

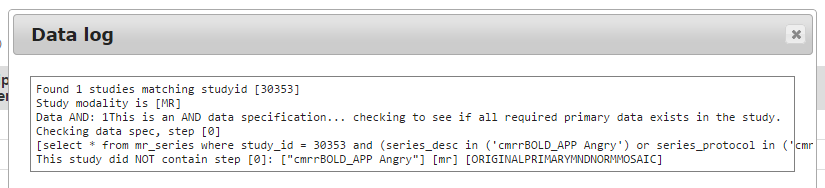
* Is the criteria step enabled (they are not enabled by default)
* Do your analyses contain the exact protocol name and modality specified
* Make sure the Image type is also valid for your data
* If your data selection criteria is an ‘and’, make sure every study has all the data

### Why did only some studies start processing?

It may, or may not, be more frustrating when only some of the studies you expect to process actually process. Not that some of the analyses started and failed, but that some didn’t start at all and were apparently ignored. For example, you specified a group of 20 studies, but only 18 of them were picked. What happened to the other 2 studies? Check the same things here that you would check if nothing processed, but the problem is most likely

* A dependent pipeline analysis failed
* The data criteria do not match all of the studies in your group

To check the ignored studies, go to the pipeline’s page, and click the **View ignored studies** link. This will display a list of studies that were checked. Click the **view log** link to check the reason they were not included in the analysis. The message may be cryptic, but the last line often describes what was not found.



# Examples

## Example 1 – Analyze all Go NoGo data (repeating series names)

**Explanation:** For each imaging study, check if they have any series that exactly match the “GoNoGo” protocol name. This will check the entire database, searching across projects to match the protocol. Since the same protocol name is repeated several times, you’d want to put them into separate directories, and it also helps to number those directories consecutively. So, the data will go into directories numbered 1,2,3,etc.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | GoNoGo | 2 | 1000 | 100,000,000 |
| GoNoGo | 3 | 1000 | 100,000,000 |
| GoNoGo | 4 | 1000 | 100,000,000 |

**Data Specification (Pipeline: GNG-Basic)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Gzip | Directory | Criteria | Use series dirs | Preserve series dirs. |
|  | GoNoGo | Study | - | Nifti 4D | 🗸 | data/GNG | All | 🗸 |  |

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG/1/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG/2/S1234ABC\_1\_3.nii.gz

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG/3/S1234ABC\_1\_4.nii.gz

## Example 2 – Analyze all Go NoGo data (unique series name)

**Explanation:** This is the same as Example 1, except the GoNoGo series are named uniquely. For example, you may have 3 runs of the task, but you needed to rerun run 2 because the subject squeezed the emergency ball to go to the bathroom. You only want one of each of the unique protocol names, and chances are you want the one with the most BOLD reps.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | GoNoGo run 1 | 2 | 1000 | 100,000,000 |
| GoNoGo run 2 | 3 | 802 | 79,000,000 |
| GoNoGo run 2 | 4 | 1000 | 100,000,000 |
| GoNoGo run 3 | 5 | 1000 | 100,000,000 |

**Data Specification (Pipeline: GNG-Basic)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Gzip | Directory | Criteria | Use series dirs. | Preserve series dirs. |
|  | GoNoGo run 1 | Study | - | Nifti 4D | 🗸 | data/GNG1 | Largest |  |  |
|  | GoNoGo run 2 | Study | - | Nifti 4D | 🗸 | data/GNG2 | Largest |  |  |
|  | GoNoGo run 3 | Study | - | Nifti 4D | 🗸 | data/GNG3 | Largest |  |  |

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG1/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG2/S1234ABC\_1\_4.nii.gz

{pipelinedir}/S1234ABC/1/GNG-Basic/data/GNG3/S1234ABC\_1\_5.nii.gz

## Example 3 – Analyze all Go NoGo data, but add in T1 and fieldmaps

**Explanation:** This pipeline will do the same things as Example 1, but will also download T1 and Fieldmaps from the same imaging session. In other words, this pipeline will only run on the study if it contains all of the specified data. When checking if all of the data exists for each imaging study; if the subject did a T1 on a different day in a different study, but not in the one being checked, it will be not analyzed because there was no T1 in the imaging study. If the T1 may exist outside of the imaging study, see Example 5.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | GoNoGo | 2 | 1000 | 100,000,000 |
| GoNoGo | 3 | 1000 | 100,000,000 |
| GoNoGo | 4 | 1000 | 100,000,000 |
| T1w | 5 | 1 | 20,000,000 |
| T1w | 6 | 1 | 20,000,000 |
| Fieldmap | 7 | 1 | 1,000,000 |
| Fieldmap | 8 | 1 | 1,000,000 |

**Data Specification (Pipeline: GNG-withT1)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Image Type | Gzip | Directory | Criteria | Use series dirs | Preserve series dirs |
|  | GoNoGo | Study | - | Nifti 4D |  | 🗸 | data/GNG | All | 🗸 |  |
|  | T1w | Study | - | Nifti 3D |  | 🗸 | data/T1 | All |  |  |
|  | Fieldmap | Study | - | Nifti 4D | ORIGINALPRIMARYMND | 🗸 | data/fmmag | Largest |  |  |
|  | Fieldmap | Study | - | Nifti 4D | ORIGINALPRIMARYPND | 🗸 | data/fmphase | Largest |  |  |

\* Note the Fieldmap protocols. There are two Fieldmap series, one phase and one magnitude. They have the same protocol name, but the Image Type is different.

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/1/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/2/S1234ABC\_1\_3.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/3/S1234ABC\_1\_4.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/T1/S1234ABC\_1\_5.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/T1/S1234ABC\_1\_6.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/fmmag/S1234ABC\_1\_7.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/fmphase/S1234ABC\_1\_8.nii.gz

## Example 4 – Analyze DTI data (keeping series number)

**Explanation:** Sometimes scanners can write out extra series but it has the same protocol name as the series from which it was derived. Or you may have an interest in the exact series number. Perhaps your scan sequences are automated and you only want to download runs 1,3,5 of some task. This will place the series in their own directories but also name the directory the same as the series number.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | DTI | 2 | 64 | 10,000,000 |
| DTI | 3 | 1 | 1,000,000 |
| DTI | 4 | 1 | 1,000,000 |
| DTI | 5 | 64 | 10,000,000 |
|  |  | DTI | 6 | 1 | 1,000,000 |
|  |  | DTI | 7 | 1 | 1,000,000 |

In this example, series 3,4,6,7 are actually derived data (FA and ADC maps) though they are named DTI. If you download all of the DTI series, you can then utilize only series 2,5.

**Data Specification (Pipeline: DTI)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Gzip | Directory | Criteria | Use series dirs. | Preserve series dirs. |
|  | DTI | Study | - | DICOM |  | data/DTI | All | 🗸 | 🗸 |

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/DTI/data/DTI/2/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/3/S1234ABC\_1\_3.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/4/S1234ABC\_1\_4.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/5/S1234ABC\_1\_5.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/6/S1234ABC\_1\_6.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/7/S1234ABC\_1\_7.nii.gz

{pipelinedir}/S1234ABC/1/DTI/data/DTI/8/S1234ABC\_1\_8.nii.gz

## Example 5 – Analyze Go NoGo, T1 from different study

**Explanation:** This pipeline will do the same things as Example 3, except it will search outside of the imaging study for the T1w image. It will also download a T2w image from the subject if it is found. If there is no T2w image, the rest of the data will still be downloaded.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | GoNoGo | 2 | 1000 | 100,000,000 |
| GoNoGo | 3 | 1000 | 100,000,000 |
| GoNoGo | 4 | 1000 | 100,000,000 |
| Fieldmap | 5 | 1 | 1,000,000 |
| Fieldmap | 6 | 1 | 1,000,000 |
| S1234ABC | 2 | T1w | 2 | 1 | 20,000,000 |
| T1w | 3 | 1 | 20,000,000 |
| T2w | 4 | 1 | 20,000,000 |

**Data Specification (Pipeline: GNG-withT1)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Image Type | Gzip | Directory | Criteria | Use series dirs | Preserve series dirs |
|  | GoNoGo | Study | - | Nifti 4D |  | 🗸 | data/GNG | All | 🗸 |  |
|  | T1w | Subject | Nearest in time | Nifti 3D |  | 🗸 | data/T1 | All |  |  |
| 🗸 | T2w | Subject | Nearest in time | Nifti 3D |  | 🗸 | Data/T2 | All |  |  |
|  | Fieldmap | Study | - | Nifti 4D | ORIGINALPRIMARYMND | 🗸 | data/fmmag | Largest |  |  |
|  | Fieldmap | Study | - | Nifti 4D | ORIGINALPRIMARYPND | 🗸 | data/fmphase | Largest |  |  |

\* Note the Fieldmap protocols. There are two Fieldmap series, one phase and one magnitude. They have the same protocol name, but the Image Type is different.

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/1/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/2/S1234ABC\_1\_3.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/GNG/3/S1234ABC\_1\_4.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/T1/S1234ABC\_2\_2.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/T1/S1234ABC\_2\_3.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/T2/S1234ABC\_2\_4.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/fmmag/S1234ABC\_1\_5.nii.gz

{pipelinedir}/S1234ABC/1/GNG-withT1/data/fmphase/S1234ABC\_1\_6.nii.gz

## Example 6 – Analyze similar protocol names

**Explanation:** Imagine you have scanned several subjects, but the protocol name changed a couple of times during the time all the subjects were scanned. You want to analyze protocols named “Go No Go”, “GNG”, and “GoNoGo” all the same way. For this example, we’ll also assume only 1 run (the largest) is needed.

**Raw Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Subject | Imaging Study | Series Description | Series # | Num BOLD reps | Size (bytes) |
| S1234ABC | 1 | GoNoGo | 2 | 1000 | 100,000,000 |
| S5678ABC | 4 | GNG | 4 | 1000 | 100,000,000 |
| S5678XYZ | 2 | Go No Go | 3 | 1000 | 100,000,000 |

**Data Specification (Pipeline: GNG)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optional | Protocol | Data Source | Subject Linkage | Data format | Gzip | Directory | Criteria | Use series dirs. | Preserve series dirs. |
|  | “GoNoGo” “GNG” “Go No Go” | Study | - | Nifti 4D | 🗸 | data/GNG | Largest |  |  |

Basically, the only change it to make the Protocol field a space separated list of protocol names. The protocol names must also be quoted.

**Data on disk** (Paths are correct, but actual filenames may be different)

{pipelinedir}/S1234ABC/1/GNG/data/GNG/S1234ABC\_1\_2.nii.gz

{pipelinedir}/S5678ABC/4/GNG/data/GNG/S5678ABC\_4\_4.nii.gz

{pipelinedir}/S5678XYZ/2/GNG/data/GNG/S5678XYZ\_2\_3.nii.gz