基于Python的fMRI分析指南

**摘 要：**功能性磁共振成像（fMRI）作为一种广泛应用的脑成像技术，已被用于探索大脑功能与行为和疾病之间的关系。然而，fMRI研究在小样本、分析方法的灵活性以及缺乏标准化流程方面，存在假阳性和结果可重复性问题，制约了其应用和研究的可靠性。为应对这些挑战，开放科学倡导数据和分析流程的透明化与可重复性，已成为解决这些问题的有效手段。本研究基于Neuroimaging Analysis Replication and Prediction Study（NARPS）项目公开的fMRI数据，详细介绍了基于Python的fMRI分析流程，涵盖从数据预处理到统计建模与结果可视化的全流程，通过提供示例代码和具体操作步骤，使研究人员，尤其是初学者能够快速上手并应用这一流程，通过这一分析框架，我们期望能够提升fMRI研究的可重复性和透明性，并推动该领域科学进步。

**关键词：**fMRI分析；开放科学；Python

# 1 引言

**背景1：**可重复性危机、fMRI研究假阳性问题、目前主流的分析软件还是SPM，FSL，MATLAB等，工具的多样性和灵活性导致了研究者在数据分析选择的不确定性增加。这一背景下，近年来学界开始推动标准化的分析流程，以提高研究的透明度。

**背景2：**现有教程和标准化流程多以英语为主，并倾向于封闭式软件，对初学者和非英语母语者而言存在较高的学习门槛。Python 作为一种开源语言，具备代码透明、社区活跃、扩展性强等优点，近年来在神经影像学领域受到越来越多的关注(K. Gorgolewski et al., 2011)。基于 Python 的工作流程不仅可以增强研究的可重复性和开放性，还可以方便地整合已有工具（如 HeuDiConv、MRIQC 和 fMRIPrep），从而提供一套完整的、多步骤的标准化分析流程。

【**建立研究问题（目标）：**针对传统方法的局限性，基于Python语言构建一个标准化、开源的fMRI分析流程，为非英语母语者降低学习难度】

**【引用相关参考文献，丰富论点】**

针对背景1：可重复性危机与假阳性问题（篇幅小）

针对背景2：Python工具的优势（重点）

**【列出本研究outline】**

1. **环境配置**
2. **格式转换（DICOM, NIfTI，GIfTI）**
3. **预处理**
4. **基本分析（GLM等）**
5. **可视化**

**可视化做两个部分，NII的，surface的**

1. **进阶分析（MVPA等）**

**我想了一下，MVPA这个部分，我们就做两个简单例子吧，classification和RSA就够了。**

# 2 环境配置

在进行fMRI数据分析之前，必须配置一个适合的分析环境。本文将详细指导读者如何搭建这一环境，即便是初学者也能成功完成配置。通过合理的环境设置，有利于解决神经影像分析中的可重复性问题。在历时三个月完成数据分析后，研究者将稿件提交，若审稿人要求重新调整参数并重新运行分析，通过Git回退至历史版本、使用Conda重建环境，并通过Docker确保依赖一致性，仅需数小时即可获得新的结果。

本节将介绍三种工具，帮助构建有利于可重复性的分析环境：（a）Conda，用于避免软件环境间的差异；（b）Docker，帮助建立完整的分析环境；（c）Git，提供代码版本控制与团队协作功能。在这三种工具中，基于Python的fMRI数据分析流程大多依赖于Docker，因此，本节将重点介绍Docker的配置。

**表1** 神经影像数据分析环境配置工具及其作用

|  |  |  |
| --- | --- | --- |
| **阶段** | **工具** | **作用定位** |
| 基础环境 | Conda | Python环境隔离与管理 |
| 协作基石 | Git | 代码/分析流程版本控制 |
| 终极封装 | Docker | 完整环境与数据打包 |

## 2.1 Conda

Python语言因其灵活性、易用性和强大的数据处理能力在科学计算和数据分析领域受到广泛欢迎。然而，Python的一大挑战是库的管理和环境的隔离。解决这个问题的一个方案是使用分布式管理系统，Conda，类似于brew、apt或pip。Conda能够在封装的环境中跟踪用户的python安装、版本和依赖项，方便共享或复制环境。如果你的研究项目需要多个需不同（可能冲突）的软件版本，Conda非常有用，其可在不同的conda环境之间轻松切换。

[Anaconda](https://www.anaconda.com/)和[Miniconda](https://docs.anaconda.com/miniconda/)都是免费的开源项目，并提供了用于创建和管理虚拟环境的Conda包管理器。这使得在不同项目间的切换和环境隔离变得简单。由于本教程旨在帮助新手建立基于python的fMRI数据分析工作流，因此将以更加轻量级的Miniconda为例进行介绍。与Anaconda不同，Miniconda仅包含了Python和Conda，未预装其他库。用户需要手动安装所需的包，这使得Miniconda的环境更加简洁，并且可以根据实际需求安装必要的包，避免不必要的存储占用。此外，Miniconda的安装文件较小，因此安装速度较快。

本节将介绍如何在Linux、Windows和MacOS上安装 Miniconda，并验证安装是否成功。完成安装后，即可基于 Miniconda 创建多个 Conda 环境，以便更好地管理 fMRI 研究项目。

### 2.1.1 Linux系统

打开终端，可以在应用程序菜单中找到，或者使用快捷键（Ctrl + Alt + T）打开。运行以下命令下载并安装Miniconda：

# 下载最新版本的Miniconda安装脚本到用户的主目录

$ cd ~

$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh

# 运行安装脚本进行Miniconda安装

$ bash Miniconda3-latest-Linux-x86\_64.sh

# 如果不希望在登录时自动激活base环境，可以使用此命令

$ conda config --set auto\_activate\_base false

# 重新加载.bashrc文件，使配置变更生效

$ source .bashrc

# 如果需要，可以删除下载的安装文件以节省磁盘空间

$ rm -r Miniconda3-latest-Linux-x86\_64.sh

### 2.1.2 Windows系统

Windows用户可以前往Miniconda官网（https://docs.conda.io/en/latest/miniconda.html）下载适合的64位或32位安装包，或者使用以下命令安装（以64位Windows系统为例）：

# 使用 curl 下载 Miniconda 安装程序并将其保存为 miniconda.exe

curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86\_64.exe -o miniconda.exe

# 使用 start 命令运行安装程序，并使用 /S 参数进行静默安装（不显示用户界面），/wait 让命令行等待安装完成后再继续执行

start /wait "" miniconda.exe /S

# 删除安装文件 miniconda.exe 以节省磁盘空间

del miniconda.exe

Tips：如果使用 Windows 运行这些命令，需确保系统已安装 curl（Windows 10 及以上版本默认已包含）。

### 2.1.3 MacOS系统

打开终端（Terminal）并执行以下命令：

# 下载最新版本的Miniconda安装脚本到“下载”目录

$ cd ~/Downloads

$ curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86\_64.sh

# 运行安装脚本进行Miniconda安装（默认会安装到您的主目录）

$ bash Miniconda3-latest-MacOSX-x86\_64.sh

# 如果不希望在登录时自动激活base环境，可以使用此命令

$ conda config --set auto\_activate\_base false

# 重新加载.bash\_profile文件，使配置变更生效

$ cd ~

$ source .bash\_profile

# 如果需要，可以删除安装文件以节省磁盘空间

$ rm -r ~/Downloads/Miniconda3-latest-MacOSX-x86\_64.sh

Tips：如果您有多个fMRI研究项目，建议为每个项目设置一个独立的Conda环境。在处理项目时，始终使用特定的Conda环境，以确保每个项目中所用的软件版本一致，这将有助于报告您所使用的具体版本。此外，Conda环境可以随时激活或停用。您只需在命令行中输入conda activate <环境名>来激活，或者输入conda deactivate来停用。

当 Miniconda 安装成功后，可以创建一个专门用于 fMRI 数据分析的 Conda 环境。本教程将基于Neuroimaging Analysis Replication and Prediction Study（NARPS）项目的fMRI公开数据指导您如何创建并配置环境，此处将环境命名为“fMRI\_Narps”。

# 检查conda设置

$ cd ~ # 切换到当前用户的主目录

$ which conda # 查看conda命令的安装路径，确认conda的安装位置

$ conda env list # 列出所有conda环境，您应该看到一个名为base的默认环境

$ conda --version # 查看当前安装的conda版本，最新版本是24.9.2（截至2025年2月15日）

$ conda --help # 显示conda命令的帮助信息，列出所有可用命令

# 如果需要更新conda

$ conda update -n base -c defaults conda # 更新base环境中的conda到最新版本

# 创建一个名为pygers的新环境，使用Python 3.9

$ conda create -n fMRI\_Narps python=3.9 # 创建一个名为fMRI\_Narps的环境，并指定使用Python 3.9版本

$ conda env list # 再次列出所有环境，确认fMRI\_Narps环境已经成功创建

# 激活pygers环境

$ conda activate pygers # 激活fMRI\_Narps环境

# 检查当前环境中的Python版本

$ python --version # 查看当前激活的环境中Python的版本

# 如果需要更新Python版本

$ conda update python # 更新当前环境中的Python版本

# 如果想要在主版本之间升级，例如从Python 3.9升级到3.12

$ conda install python=3.12 # 安装指定版本的Python（例如3.12）

$ python --version # 查看更新后的Python版本

针对于fMRI数据分析，本教程建议安装以下软件包到已创建好的fMRI\_Narps的环境中。如果您有其他特定需求，也可以根据项目要求安装其他所需的软件包到当前环境中。

# 确保激活fMRI\_Narps环境

$ conda activate fMRI\_Narps # 激活名为fMRI\_Narps的conda环境

# 安装前四个包

$ conda install git numpy pandas jupyter # 安装git、numpy、pandas和jupyter包

# 检查安装情况；以下版本是截至2025年2月15日的最新版本

$ conda list # 列出当前环境中安装的所有包

$ git --version # 查看git版本（例如2.45.2）

$ jupyter --version # 查看jupyter版本（核心版本为5.7.2）

$ python -c "import numpy; print(numpy.version.version)" # 查看numpy版本（例如2.2.2）

$ python -c "import pandas; print(pandas.\_\_version\_\_)" # 查看pandas版本（例如2.2.3）

# 安装nodejs

$ conda install -c conda-forge nodejs # 安装nodejs

$ node -v # 检查nodejs的版本（例如20.12.2）

# 安装bids-validator

$ npm install -g bids-validator # 使用npm全局安装bids-validator

$ which bids-validator # 查看bids-validator的安装位置

$ bids-validator -v # 查看bids-validator的版本（例如1.15.0）

# 安装datalad（具体安装所依赖的包见3.1节）

$ conda install -c conda-forge datalad # 使用conda从conda-forge渠道安装datalad

# 查看datalad的版本（例如1.1.4）

$ datalad --version

# 安装pydeface和fsleyes（仅限本地环境）

$ pip install pydeface # 使用pip安装pydeface包

$ pydeface --help # 查看pydeface的帮助信息

$ conda install -c conda-forge fsleyes # 使用conda从conda-forge安装fsleyes

$ fsleyes --version # 查看fsleyes的版本（例如1.13.0）

# 查看fMRI\_Narps环境中已安装的所有包

$ conda list # 列出fMRI\_Narps环境中已安装的所有包

## 2.2 Docker

Docker是一个用于虚拟化和容器化复杂数据处理环境的工具，类似于Conda和renv，Docker允许创建和管理隔离的软件环境。Docker容器在概念上与Conda和renv环境类似：它们可以在不影响系统中其他进程的情况下，使用一组单独的软件包，并允许其他人在他们的机器上重现这些环境。与Conda和renv不同，Docker容器提供了更深层次的隔离，且支持安装几乎任何软件（需要相应的安装文件，以便未来可供下载）。从某种意义上说，Docker 模拟了计算机中的计算机，使其类似于虚拟机。然而，Docker容器与虚拟机之间存在一些技术差异，使得Docker更加易于使用，并在实际工作中具有更高的实用性。Docker提供了两个显著优势：更深层次的环境复现性，以及在日常工作中的便捷使用，使其成为理想的软件环境可重复性解决方案。

Docker特别适合虚拟化和容器化复杂的工作流程，尤其是在数据科学和医学成像（如 fMRI 数据处理）领域，Docker解决了配置环境的繁琐问题，对初学者也极为友好。因此，本教程将重点介绍如何使用 Docker镜像运行工具，如MRIQC和fMRIPrep等fMRI数据预处理工具。

本节将介绍如何在 Linux、Windows 和 macOS 系统上安装 Docker。

### 2.2.1 Linux系统

在Linux系统下有两种安装方式，可以选择APT或者手动安装。

前情提示：请勿在未配置 Docker APT 源的情况下直接使用 apt 命令安装 Docker。

由于 APT 源使用 HTTPS 进行安全传输，首先需要添加支持 HTTPS 的软件包和 CA 证书。可执行以下命令来进行配置：

# 更新 apt 包索引

sudo apt update

# 安装必要的工具和证书

sudo apt install \

apt-transport-https \

ca-certificates \

curl \

gnupg \

lsb-release

鉴于国内网络问题，推荐使用国内源。为了验证下载软件包的合法性，需要添加 GPG 密钥，如下：

# 添加阿里云源的 GPG 密钥

curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# 添加 Docker 官方源（如果需要使用官方源，请将下面注释取消）

# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

接下来，向`source.list`中添加Docker软件源，执行命令如下：

# 添加阿里云 Docker 镜像源

echo \

"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://mirrors.aliyun.com/docker-ce/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 如果需要使用官方源，请取消以下注释

# echo \

# "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

# $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

上述命令会将Docker APT镜像源添加为稳定版本。如果需要安装测试版Docker，即将stable改为test。

其次，更新apt包缓存并安装Docker，命令如下：

# 更新 apt 包缓存

sudo apt update

# 安装 Docker 及相关工具

sudo apt install docker-ce docker-ce-cli containerd.io

为了简化安装流程，在测试或开发环境中 Docker提供了一套便捷的安装脚本，另外可以通过—mirror选项使用国内源进行安装（若需要安装测试版的 Docker, 可从 test.docker.com 获取脚本）。

# 使用阿里云镜像安装 Docker

curl -fsSL get.docker.com -o get-docker.sh

sudo sh get-docker.sh --mirror Aliyun

安装完成后，启动Docker服务并设置开机自启：

# 启动 Docker 并设置为开机启动

sudo systemctl enable docker

sudo systemctl start docker

默认情况下，Docker命令会使用Unix Socket与Docker引擎通讯，只有root用户和docker组成员才可以访问Docker引擎的Unix Socket。出于安全考虑，一般Linux系统上不会直接使用root用户。因此，更好地做法是将需要使用docker的用户加入docker用户组。可在命令行执行以下命令：

# 创建 Docker 组

sudo groupadd docker

# 将当前用户添加到 Docker 组

sudo usermod -aG docker $USER

执行上述命令后，请退出当前终端并重新登陆。其次，使用以下命令验证Docker是否安装成功：

# 测试 Docker 安装是否成功

docker run --rm hello-world

如果输出以下类似内容，则说明Docker安装成功：

Hello from Docker!

This message shows that your installation appears to be working correctly.

注意，如果在使用过程中发现拉取Docker镜像十分缓慢，可以配置docker国内镜像加速。

### 2.2.2 Windows系统

在Windows系统下安装docker，一般会有系统要求，Docker Desktop for Windows 支持 64 位版本的Windows 10 Pro，且必须开启Hyper-V（若版本为v1903及以上则无需开启 Hyper-V），或者64位版本的Windows 10 Home v1903及以上版本。

若系统满足要求，可选择手动下载安装，也可以通过winget安装。访问docker官网下载Docker Desktop for Windows（https://www.docker.com/products/docker-desktop/）。下载完成后，双击`Docker Desktop Installer.exe`开始安装。若使用winget进行安装，可在命令行执行以下命令：

$ winget install Docker.DockerDesktop

如果使用的Windows版本为 Windows 10专业版或家庭版v1903及以上版本可以使用WSL2运行Docker，具体请查看Docker Desktop WSL 2 backend（https://docs.docker.com/desktop/features/wsl/）。安装完成后，点击任务栏中的Docker图标，启动Docker。当图标静止时，表示Docker启动成功。此时，您可以在PowerShell中运行 Docker命令。

Tips：推荐使用 Windows 终端（Windows Terminal）运行 Docker，更多信息请参见（<https://learn.microsoft.com/zh-cn/windows/terminal/install>）。

### 2.2.3 MacOS系统

Docker desktop for mac要求系统最低为MacOS必须是10.15或更高版本，Catalina、Big Sur或者Monterey，建议升级到最新版本的MacOS。在MacOS系统下安装Docker，可选择使用Homebrew或手动安装。若使用Homebrew安装，可在终端执行以下命令：

# 使用 Homebrew 安装 Docker Desktop for Mac

brew install --cask docker

若选择手动安装，可访问Docker官网下载适合你系统的版本。（https://www.docker.com/products/docker-desktop/）下载Docker Desktop for Mac。安装过程与macOS中其他应用程序类似。下载后双击.dmg文件，将Docker图标拖动到应用程序文件夹中。安装完成后，您可以在应用程序中找到Docker图标并启动。

安装后，您可以在终端中运行以下命令来检查Docker的版本：

$ docker --version

如果返回Docker版本信息，说明安装成功。您还可以运行一个Nginx服务器（https://hub.docker.com/\_/nginx/）来验证Docker是否正常工作：

$ docker run -d -p 80:80 --name webserver nginx

服务运行后，可以访问 http://localhost，如果看到"Welcome to nginx!"说明 Docker Desktop for Mac 安装成功了。要停止并删除该服务，运行以下命令：

$ docker stop webserver

$ docker rm webserver

## 2.3 Git

Git作为一种分布式开源版本控制系统（VCS），可用于存储代码、跟踪修订历史记录、合并代码更改，并在需要时恢复为较早的代码版本。在研究中，若要实现可重复性，通常需要在相同数据上运行相同的代码，确保得到一致的结果。然而，验证代码的相同与变化往往是一个挑战，尤其是在协作环境中。此时，Git 提供了一个有效的解决方案，它能清晰地展示代码变化的细节、变更的时间及其发生的原因。同时，Git 使得协作变得更为简单，特别是在进行开放研究时，它成为了不可或缺的工具(Peikert & Brandmaier, 2021)。

Git 通过将源代码及其完整的开发历史记录保存在本地存储库中，为开发提供了灵活的管理方式。开发者可以创建一个称为“分支”的源代码副本，在此基础上并行进行工作。完成工作后，开发者可以提交更改并保存进度，或者将修改合并回主分支。每次提交时，Git 会为该进度创建一个快照，并通过名为“diff”的可视化功能与先前版本进行对比。如果发生了修改，Git 会在存储库中保存一个新的快照。

GitHub则是一个基于Web的Git版本控制服务。它本质上是一个为开发者提供的社交网络平台，用户可以查看其他人的代码，识别其中的问题，并提出改进建议。简而言之，Git 是一个版本控制系统，而GitHub是用于托管Git仓库的服务平台

**表2** GitHub基本术语含义

|  |  |
| --- | --- |
| **基本术语** | **含义** |
| Repository | Git仓库，简称repo，是您告诉Git用于跟踪文件变化的文件夹。 |
| Branch | 分支是一个独立的开发分支，可将其视为一个全新的工作目录。 |
| Fork | 复制某个项目到自己的仓库。 |
| Clone | 将项目下载到本地。 |
| Commit | 提交修改，并且会生成一个唯一的ID，以便跟踪历史记录。 |
| Master | 默认的开发分支。当您创建一个新的Git仓库时，默认会创建一个名为“master”的分支，并将其设为活跃分支。 |

本节将介绍如何在Linux、Windows和MacOS上安装Git（更多问题可参阅官方教程文档<https://git-scm.com/book/zh/>）。

### 2.3.1 Linux系统

如果你想在Linux上用二进制安装程序来安装基本的Git工具，可以使用发行版包含的基础软件包管理工具来安装。 这里以基于Debian的发行版，如Ubuntu为例，可使用apt进行安装：

$ sudo apt install git-all

要了解更多选择，Git官方网站上有在各种Unix发行版的系统上安装步骤（可访问网址：https://git-scm.com/download/linux）。

### 2.3.2 Windows系统

在Windows上安装Git也有几种安装方法。 官方版本可以在Git官方网站下载（可访问链接： <https://git-scm.com/download/win>），打开该链接后，下载会自动开始。 要注意这是一个名为 Git for Windows 的项目（也叫做msysGit），和Git 是分别独立的项目（具体信息可访问 <http://msysgit.github.io/>）。

### 2.3.3 MacOS系统

在 Mac 上安装 Git 有多种方式。 最简单的方法是安装 Xcode Command Line Tools。 Mavericks （10.9） 或更高版本的系统中，在Terminal里尝试首次运行 'git' 命令即可。

$ git –-version

Tips：如果没有安装过命令行开发者工具，将会提示你安装，按照说明安装即可。

### 2.3.4 Git的基本用法

Git的工作就是创建和保存你项目的快照以及与之后的快照进行对比。Git常用的命令有git clone、git push、git add、git commit、git checkout和git pull。

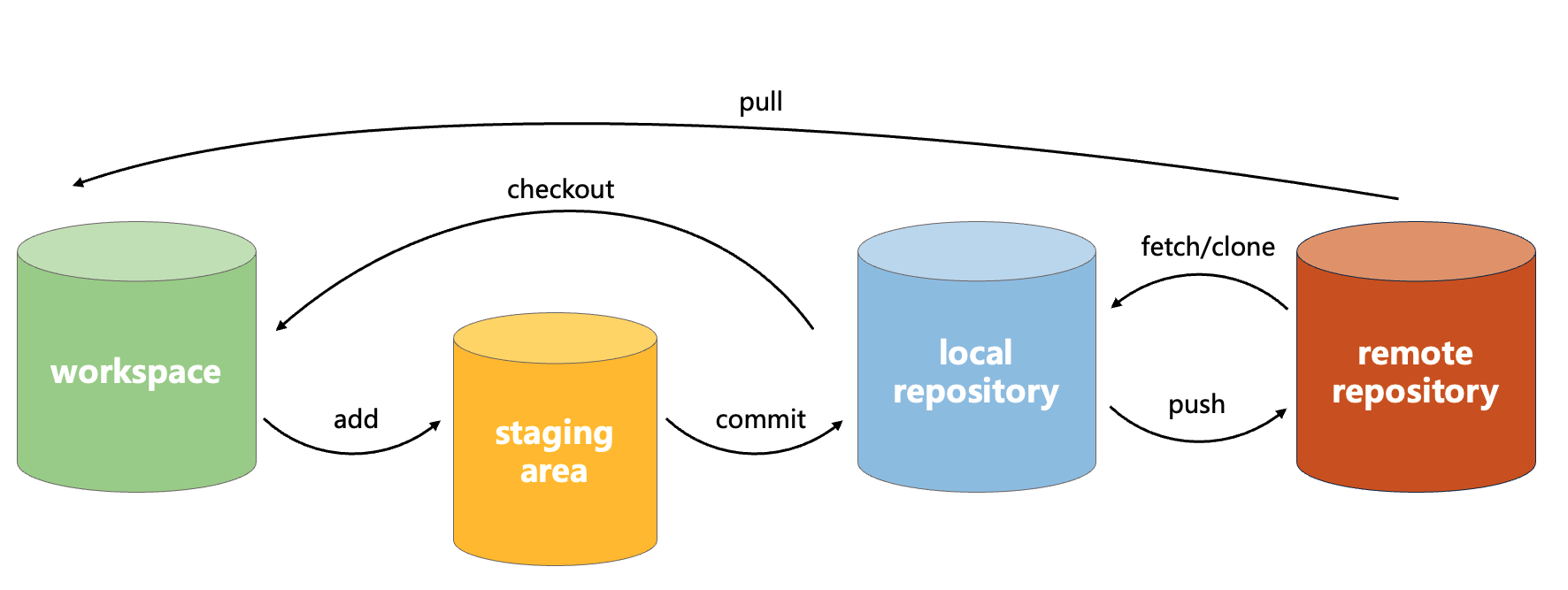


图1 Git的基本功能

**注：**workspace为工作区；staging area为暂存区；local repository为版本库；remote repository为远程仓库。

以下是关于使用Git的一些基本操作：

# 查看已配置的远程仓库

git remote # 显示所有远程仓库简称

git remote -v # 显示详细 URL 信息

# 添加远程仓库

git remote add <简称> <URL>

# 从远程仓库中获取数据

git fetch <远程简称> # 仅下载数据，不自动合并

# 拉取并自动合并

git pull <远程简称> <分支名> # 相当于 fetch + merge

# 推送本地提交到远程仓库

git push <远程简称> <分支名>

# 查看远程仓库详细信息

git remote show <远程简称>

# 重命名远程仓库

git remote rename <旧名称> <新名称>

# 删除远程仓库

git remote rm <远程简称>

# 3 基于Openneuro平台下载公开数据

开放科学作为一种实践，旨在通过提升数据的可获得性、公众意识和研究透明度来应对可重复性危机。Rosenthal在1979年首次指出了学术界长期存在的“文件抽屉问题”（File Drawer Problem），揭示了非显著性研究结果常被研究者选择性不报告，这一问题导致元分析结果可能高估了效应大小，从而降低了研究的可靠性（Scargle, 2000）。在神经影像学研究领域，这一问题因 “可重复性危机”而进一步加剧，Bennett 等人（2009）通过fMRI扫描死亡大西洋鲑鱼的实验发现，即使对无生命体进行检测，数据分析方法仍可能产生统计学显著的伪阳性激活信号。同年，Vul 等人（2009）在其论文“Puzzlingly High Correlations in fMRI Studies of Emotion, Personality, and Social Cognition”中指出，大量fMRI研究中由于研究者筛选和计算体素时使用了同一组数据，违反了统计独立性的原则，导致相关性被认为放大，从而降低了研究结果的可信度。上述问题表明，提升研究的公开透明度和数据可获取性已成为神经科学领域亟须解决的问题。

为应对这一挑战，Poldrack及其同事于2015年创立的斯坦福可重复神经科学中心（Stanford Center for Reproducible Neuroscience, SCRN）（Stanford CRN, 2021），致力于推动开放数据共享，尤其是脑成像数据结构（Brain Imaging Data Structure, BIDS）的推广（Gorgolewski et al., 2016）。BIDS格式的推广跨研究之间的数据兼容性，使研究者能够更高效地利用开放数据（Holdgraf et al., 2018; Pernet et al., 2018; Niso et al., 2018）（关于BIDS格式的具体信息见第4节）。

为进一步推动开放科学，Poldrack和Gorgolewski于2017年创建了OpenNeuro数据库（前身为OpenfMRI），旨在为全球神经科学研究人员提供一个免费、开放的数据存储和共享平台。OpenNeuro允许研究者上传任务相关的fMRI数据集，不受数据集规模或通用性的限制。目前，OpenNeuro已扩展支持多种数据类型，包括EEG、iEEG、MEG、ECoG、ASL和PET，并统一采用BIDS格式进行数据组织。本节将重点介绍如何通过OpenNeuro下载公开数据。当前常用的下载工具包括Datalad、命令行接口和Amazon S3，本文将主要介绍其中的两种：Datalad和Amazon S3。

## 3.1 安装Datalad

DataLad 是一款基于 Git 和 git-annex 的跨平台数据管理工具，其安装过程需预先配置以下依赖环境：

1. 基础环境：所有操作系统均需预先安装Git（版本在2.0及以上）和git-annex（版本在6.2及以上）；
2. Linux/MacOS系统：建议通过系统包管理器安装（例如，apt-get或homebrew）；
3. Windows系统：由于git-annex对Windows系统的原生支持存在功能限制，建议次啊用替代方案（详见3.3节内容）。

### 3.1.1 Linux系统

```bash

# 更新包管理器

sudo apt-get update

# 安装 Git

sudo apt-get install git

# 安装 git-annex

sudo apt-get install git-annex

# 创建并激活虚拟环境（可选但推荐）

python3 -m venv ~/datalad\_env

source ~/datalad\_env/bin/activate

# 安装 DataLad

pip install datalad

```

### 3.1.2 MacOS系统

```bash

# 安装 Homebrew（如果未安装）

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# 安装 Git 和 git-annex

brew install git

brew install git-annex

# 创建并激活虚拟环境

python3 -m venv ~/datalad\_env

source ~/datalad\_env/bin/activate

# 安装 DataLad

pip install datalad

```

## 3.2 使用Datalad下载公开数据

### 3.2.1 Linux系统和MacOS系统

```bash

# 基础操作

# 克隆元数据（不下载实际文件）

datalad clone https://github.com/OpenNeuroDatasets/ds002336.git

cd ds002336

# 查看文件夹数据的大小

du -sh

# 按需获取数据

datalad get sub-01/anat/ # 下载特定被试的解剖数据

datalad get . # 下载完整数据集

# 更新数据集

datalad update –merge

# 其他操作

# 访问数据集特定的版本

git tag

# 切换到某个特定的版本

git checkout 1.0.5

```

## 3.3 安装Amazon S3

AWS CLI 用于通过 Amazon S3 下载 OpenNeuro 数据，无需 AWS 账户即可访问公开数据。

### 3.3.1 Linux系统

```bash

# 安装 AWS CLI

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

# 验证安装

aws --version

```

### 3.3.2 Windows系统

```powershell

# 下载安装包

Invoke-WebRequest -Uri "https://awscli.amazonaws.com/AWSCLIV2.msi" -OutFile "$env:TEMP\AWSCLIV2.msi"

# 静默安装

Start-Process -Wait -FilePath "$env:TEMP\AWSCLIV2.msi" -ArgumentList "/quiet"

# 验证安装

aws --version

```

## 3.4 使用Amazon S3下载公开数据

```bash

# 下载完整数据集

aws s3 sync --no-sign-request s3://openneuro.org/ds002336 ./ds002336/

# 选择性下载（示例：只要 BIDS 原始数据）

aws s3 sync --no-sign-request \

s3://openneuro.org/ds001734 ./ds002336/ \

--exclude "\*" \

--include "sub-\*/ses-\*/\*" \

--include "dataset\_description.json"

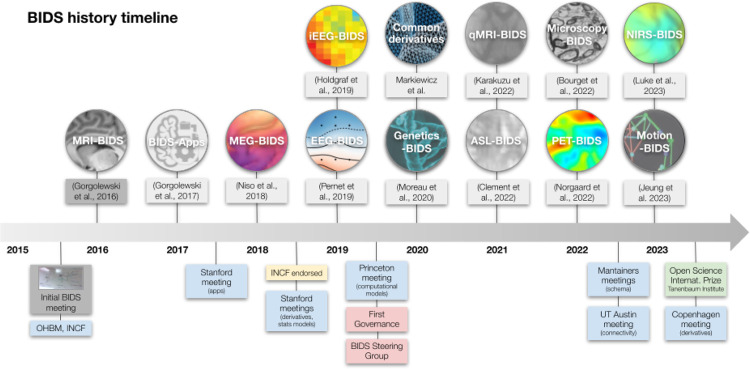
# 断点续传（网络不稳定时）

aws configure set default.s3.max\_concurrent\_requests 5 # 限制并发数

```

# 4 BIDS格式

Brain Imaging Data Structure（BIDS）是一种标准化的脑影像数据组织和存储结构（后文简称为BIDS），旨在为大规模的神经影像数据集提供一种通用的组织框架。它不仅规范了文件和目录结构，还包括对相关元数据的要求。BIDS格式的发展时间线如图2所示(Halchenko et al., 2024)。2016年6月，根据Gorgolewski等人发表的文章，该标准正式发布(K. J. Gorgolewski et al., 2016)。BIDS标准最初是为MRI数据开发的，但后来得到了更广泛的神经影像学界的欢迎，扩展到了MEG、EEG和iEEG等数据结构中(Holdgraf et al., 2019; Niso et al., 2018; Pernet et al., 2019)，包括在大型公开数据库OpenNeuro平台上也要求研究者上传的数据必须使用BIDS格式，确保了其广泛的适用性。此外，在Python中使用非常广泛的fMRIPrep也需要使用BIDS格式才可进行预处理，更加具体的说明请访问其官方网站（<http://bids.neuroimaging.io/>）。



**图2** BIDS格式的发展历程 (Poldrack et al., 2024)

## 4.1 BIDS格式用于fMRI数据

### 4.1.1 BIDS数据结构

为了研究者能够更好地管理研究的fMRI数据集，本文将介绍BIDS用于fMRI的基本数据结构，以及如何将原始的DICOM格式数据转为BIDS格式。以Narps数据集为例，BIDS结构的数据集主要的核心文件类型有：

1. NIfTI（一般为.nii.gz的压缩格式）：存储图像数据（如功能像、结构像、场图）；
2. JSON（附属的元数据）：记录扫描参数（如TE、TR、相位编码方向）；
3. TSV：存储表格数据（如被试信息）。

以NARPS数据集中的sub-001为例，符合BIDS标准的数据集树状结构如下：

ds001734/

├── participants.tsv

├── dataset\_description.json

├── README

├── sub-001/

│ ├── anat/

│ │ └── sub-001\_T1w.nii.gz

│ ├── fmap/

│ │ ├── sub-001\_magnitude1.nii.gz

│ │ ├── sub-001\_magnitude2.nii.gz

│ │ ├── sub-001\_phasediff.json

│ │ └── sub-001\_phasediff.nii.gz

│ └── func/

│ ├── sub-001\_task-MGT\_run-01\_bold.nii.gz

│ ├── sub-001\_task-MGT\_run-01\_events.tsv

│ ├── sub-001\_task-MGT\_run-01\_sbref.nii.gz

│ ├── sub-001\_task-MGT\_run-02\_bold.nii.gz

│ ├── sub-001\_task-MGT\_run-02\_events.tsv

│ ├── sub-001\_task-MGT\_run-02\_sbref.nii.gz

│ ├── sub-001\_task-MGT\_run-03\_bold.nii.gz

│ ├── sub-001\_task-MGT\_run-03\_events.tsv

│ ├── sub-001\_task-MGT\_run-03\_sbref.nii.gz

│ ├── sub-001\_task-MGT\_run-04\_bold.nii.gz

│ ├── sub-001\_task-MGT\_run-04\_events.tsv

│ └── sub-001\_task-MGT\_run-04\_sbref.nii.gz

在创建好所有的符合格式的文件夹以后，还需要注意以下几点。第一，BIDS数据中涉及的其他文件。首先来说要放在与“sub-\*”文件夹同一目录级别的文件，包括用于说明整个数据集改动的“CHANGES”文件和用于说明被试情况的“participants.json”和“participants.tsv”、用于说明整个数据集情况的“README”文件以及“dataset\_description.json”文件。这些文件在使用工具时会自动根据数据集内容生成，若有需要自己进一步补充的部分，比如数据的描述，被试的更多情况等等。这些数据是必要的，缺失则会导致无法validate。其次，对于每个被试文件夹（即“sub-\*”文件夹），应包括用于保存被试扫描参数的.json文件和.tsv文件。最后，功能数据“func”文件夹中还应有以“event”结尾的.tsv文件以及.json文件。

在DICOM格式数据转为BIDS格式过程中，文件的命名问题是导致BIDS格式不规范的一个主要原因。由于每个图像以及.json文件和.tsv文件的名字是数据处理人自己设置的，软件无法帮你设置，所以对于脑成像数据的文件都应规范命名。

### 4.1.2 场图数据的BIDS结构和命名规则

功能磁共振成像通常采用平面回波成像（EPI）的序列，该技术对于静态磁场（*B0*场）的均匀度非常敏感。磁场不均匀会导致显著的几何畸变，这种畸变主要发生在相位编码方向，在图像中表现为沿着相位编码方向平移、压缩或拉伸。

场图（fieldmap）是一种直接测量磁场不均匀性的图像类型，可用于校正EPI图像中的几何畸变。通过调整EPI序列的相位编码梯度运行方式，可以改变几何畸变的方向。例如，相位编码方向为后-前（posterior-anterior, PA）时，前额叶区域的EPI图像可能表现为向外拉伸；而相位编码方向为前-后（anterior-posterior, AP）时，则可能表现为向内压缩。为了有效校正因B0场不均匀引起的EPI图像畸变，通常需要采集场图数据。场图采集通常只需几分钟，但能显著提升后续预处理步骤（如图像配准）的质量。

场图数据主要有四种采集方式，包括梯度回波（Gradient Echo）、不对称自旋回波（Asymmetric Spin Echo）、EPI平面回波以及目标翻转b=0对（EPI）（Blip-reversed b=0 pair （EPI））。BIDS格式中，场图数据存储于每个被试的sub-<label>/fmap/子目录中，文件命名遵循BIDS规范。以下分别展示两种常见场图采集方式的BIDS数据结构（更多信息可参考BIDS官网<https://bids-specification.readthedocs.io/en/stable/modality-specific-files/magnetic-resonance-imaging-data.html#fieldmap-data>）：

1. 相位差场图结构：通过采集两个不同回波时间（TE）的梯度回波图像，生成一个相位差图（phase difference map）以及两个幅值图（magnitude images）；

**表3** 相位差场图在BIDS中的文件夹结构

|  |  |
| --- | --- |
| 文件名 | 含义 |
| sub-01\_ses\_mri\_magnitude1.nii.gz | 第一个回波的幅值图像 |
| sub-01\_ses\_mri\_magnitude2.nii.gz | 第二个回波的幅值图像 |
| sub-01\_ses\_mri\_phasediff.nii.gz | 两个回波的相位差图像（核心场图） |
| sub-01\_ses\_mri\_phasediff.json | 相位差图像的元数据（关键参数） |

1. 双相位场图结构：采集两个独立的相位图（phase maps）和两个幅值图，每张图像均附带元数据。

**表4** 双相位场图在BIDS中的文件夹结构

|  |  |
| --- | --- |
| 文件名 | 含义 |
| sub-01\_ses\_mri\_magnitude1.nii.gz | 第一个回波的幅值图像 |
| sub-01\_ses\_mri\_magnitude1.json | 第一个回波的幅值图像的元数据 |
| sub-01\_ses\_mri\_magnitude2.nii.gz | 第二个回波的幅值图像 |
| sub-01\_ses\_mri\_magnitude2.json | 第二个回波的幅值图像的元数据 |
| sub-01\_ses\_mri\_phase1.nii.gz | 第一个相位图像 |
| sub-01\_ses\_mri\_phase1.json | 第一个相位图像的元数据 |
| sub-01\_ses\_mri\_phase2.nii.gz | 第二个相位图像 |
| sub-01\_ses\_mri\_phase2.json | 第二个相位图像的元数据 |

## 4.2 Heudiconv

在环境配置中，本文已介绍了如何配置Docker，在进行BIDS格式转换的时候，我们可以利用docker环境进行。在进行DICOM转为BIDS格式数据的过程中，如dcm2niix(Li et al., 2016)、BIDScoin(Zwiers et al., 2022)以及HeuDiConv(Halchenko et al., 2024)的方法。HeuDiConv中用户只需要编写脚本，HeuDiConv就可以根据用户的数据集特点自动将DICOM数据组织为BIDS格式，减少了手动操作和出错的时间，其次，HeuDiConv作为Python工具，可以集成到基于Python的工作流程中，支持通过命令行或者Python脚本运行，并且HeuDiConv内部集成了 dcm2niix 的转换功能，因此用户不仅能享受到 dcm2niix 的高效，还能直接获得BIDS格式的输出（更多信息可访问HeuDiConv官方教程https://heudiconv.readthedocs.io/en/latest/）。互联网上也有很多其他优秀的教程，可查看该链接获取更为详细的HeuDiConv教程，https://reproducibility.stanford.edu/bids-tutorial-series-part-2a/。如果没有DICOM原始数据，但想先练习Heudiconv的初学者，可以下载在OpenNeuro上的练习数据（链接为https://openneuro.org/datasets/ds000031/versions/00001）。

### 4.2.1 安装Heudiconv

Heudiconv是一个强大的工具，可以将DICOM格式的数据转换为BIDS格式。安装Heudiconv之前，首先需要拉取Heudiconv的Docker镜像。以下是安装步骤：

拉取Heudiconv Docker镜像：

docker pull nipy/heudiconv:latest

### 4.2.2 组织DICOM数据

在使用Heudiconv转换数据之前，必须先对原始DICOM数据进行合理的组织。在使用Heudiconv之前，建议将原始DICOM数据按照以下结构组织：总目录/被试级别目录/session级别目录/不同扫描protocol的目录。

数据结构建议遵循以下层级格式：

HeuDiConv\_walkthrough/

├── Dicoms/

│ ├── sub-01/

│ │ ├── ses-001/

│ │ │ ├── dcmtoxnat.log

│ │ │ ├── SCANS/

│ │ │ │ ├── 3/

│ │ │ │ │ ├── DICOM/

│ │ │ │ │ └── SNAPSHOTS/

│ │ │ │ ├── 4/

│ │ │ │ ├── 5/

│ │ │ │ ├── 6/

│ │ │ │ ├── 7/

│ │ │ │ ├── 8/

│ │ │ │ └── 9/

具体而言，**每个被试的每个会话**（session）下的数据应该按“被试级别/会话级别/扫描协议”进行组织，确保每个被试的数据存放在独立的文件夹中，每个会话（session）下的数据存放在对应的子文件夹内。在组织好DICOM数据后，可以使用Heudiconv来读取这些数据并转换为BIDS格式。

**4.1.4执行Heudiconv转换**

执行Heudiconv转换时，Docker命令行格式如下：

**4.1.5 使用Python脚本进行批量处理**

在数据转换后，Heudiconv工具会生成一个.heudiconv的隐藏文件夹，其中包含了基本的DICOM信息（如.tsv文件）。可以通过编写Python脚本（上github链接）来自动化批量处理这些文件，进一步简化数据转换过程。该脚本的详细实现可以在[GitHub项目](https://github.com/shanshan717/fMRI_pipeline_narps.git)中找到，文件名为Step\_0.1\_Heudiconv.py。

**4.1.6 验证BIDS格式**

当完成格式转换后，可以使用BIDS格式验证工具（<https://bids-standard.github.io/bids-validator/>）（支持Firefox和Chrome浏览器）来验证数据是否符合BIDS标准。如果不行，查看error里的信息，进行调整。

# 5 fMRI预处理

## 5.1质量控制：MRIQC Package

**1. 通过Docker拉取MRIQC镜像：**

docker pull nipreps/mriqc:24.0.2

1. **运行MRIQC**

使用 MRIQC 对功能磁共振数据进行质量控制，并生成详细的质量检查报告。运行命令可以写入脚本文件（Step\_0.3\_MRIQC.sh）。

## 5.2 数据预处理（fMRIPrep packagge）

### 5.2.1 通过Docker拉取fMRIPrep镜像：

docker pull poldracklab/fmriprep:<latest-version>

下载好以后，还需要在freesurfer 官网（https://surfer.nmr.mgh.harvard.edu/registration.html）获取license。只要注册获取即可，下载好以后，把它保存在可用路径下，或者如果你的电脑已经装好了freesurfer，并且配置好了环境变量，即写入了.bashrc文件，那你可以省略这一步，fmriprep会搜寻freesurfer的安装路径和license路径。

### 5.2.2 运行fMRIPrep：

写脚本运行fMRIPrep，进行标准化的预处理流程，包括对功能数据和结构数据的标准化和空间配准。

fmriprep-docker /Users/wu\_script/Nifti /Users/wu\_script/derivatives participant

--participant-label 04 --fs-license-file /Users/license/freesurfer.txt

利用fMRIPrep进行数据预处理，使用 fMRIPrep 23.1.3 (Esteban et al., 2019, 2018)，基于 Nipype 1.8.6 (Gorgolewski et al., 2011, 2018)实现。需要特别注意的是，fMRIPrep 专注于最小化预处理流程，包括头动校正、时序校正和空间配准等标准化步骤，但不包含平滑（smoothing）和缩放（scaling）等后续处理。这些步骤可根据具体研究需求，使用其他工具（如 Nilearn）完成。由于本教程涵盖了表层空间（surface space）的处理，因此在数据预处理脚本中特别加入了 FreeSurfer 的表层重建步骤，以支持对表层空间的深入分析。

# 参考文献

Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011). Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in python. Frontiers in Neuroinformatics, 5, 13. https://doi.org/10.3389/fninf.2011.00013

Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., Flandin, G., Ghosh, S. S., Glatard, T., Halchenko, Y. O., Handwerker, D. A., Hanke, M., Keator, D., Li, X., Michael, Z., Maumet, C., Nichols, B. N., Nichols, T. E., Pellman, J., … Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. Scientific Data, 3(1), 160044. https://doi.org/10.1038/sdata.2016.44

Halchenko, Y. O., Goncalves, M., Ghosh, S., Velasco, P., Castello, M. V. di O., Salo, T., Wodder, J. T., Hanke, M., Sadil, P., Gorgolewski, K. J., Ioanas, H.-I., Rorden, C., Hendrickson, T. J., Dayan, M., Houlihan, S. D., Kent, J., Strauss, T., Lee, J., To, I., … Kennedy, D. N. (2024). HeuDiConv—Flexible DICOM conversion into structured directory layouts. Journal of Open Source Software, 9(99), 5839. https://doi.org/10.21105/joss.05839

Holdgraf, C., Appelhoff, S., Bickel, S., Bouchard, K., D’Ambrosio, S., David, O., Devinsky, O., Dichter, B., Flinker, A., Foster, B. L., Gorgolewski, K. J., Groen, I., Groppe, D., Gunduz, A., Hamilton, L., Honey, C. J., Jas, M., Knight, R., Lachaux, J.-P., … Hermes, D. (2019). iEEG-BIDS, extending the Brain Imaging Data Structure specification to human intracranial electrophysiology. Scientific Data, 6(1), 102. https://doi.org/10.1038/s41597-019-0105-7

Li, X., Morgan, P. S., Ashburner, J., Smith, J., & Rorden, C. (2016). The first step for neuroimaging data analysis: DICOM to NIfTI conversion. Journal of Neuroscience Methods, 264, 47–56. https://doi.org/10.1016/j.jneumeth.2016.03.001

Niso, G., Gorgolewski, K. J., Bock, E., Brooks, T. L., Flandin, G., Gramfort, A., Henson, R. N., Jas, M., Litvak, V., T. Moreau, J., Oostenveld, R., Schoffelen, J.-M., Tadel, F., Wexler, J., & Baillet, S. (2018). MEG-BIDS, the brain imaging data structure extended to magnetoencephalography. Scientific Data, 5(1), 180110. https://doi.org/10.1038/sdata.2018.110

Peikert, A., & Brandmaier, A. M. (2021). A reproducible data analysis workflow with R markdown, git, make, and docker. Quantitative and Computational Methods in Behavioral Sciences, 1–27. https://doi.org/10.5964/qcmb.3763

Pernet, C. R., Appelhoff, S., Gorgolewski, K. J., Flandin, G., Phillips, C., Delorme, A., & Oostenveld, R. (2019). EEG-BIDS, an extension to the brain imaging data structure for electroencephalography. Scientific Data, 6(1), 103. https://doi.org/10.1038/s41597-019-0104-8

Poldrack, R. A., Markiewicz, C. J., Appelhoff, S., Ashar, Y. K., Auer, T., Baillet, S., Bansal, S., Beltrachini, L., Benar, C. G., Bertazzoli, G., Bhogawar, S., Blair, R. W., Bortoletto, M., Boudreau, M., Brooks, T. L., Calhoun, V. D., Castelli, F. M., Clement, P., Cohen, A. L., … Gorgolewski, K. J. (2024). The Past, Present, and Future of the Brain Imaging Data Structure (BIDS). ArXiv, arXiv:2309.05768v2.

Zwiers, M. P., Moia, S., & Oostenveld, R. (2022). BIDScoin: A User-Friendly Application to Convert Source Data to Brain Imaging Data Structure. Frontiers in Neuroinformatics, 15. https://doi.org/10.3389/fninf.2021.770608

Scargle, J. R. (2000). Publication bias: The “file-drawer” problem in scientific inference. Journal of Scientific Exploration, 14(1), 91–106.