Manual: Conversión de imágenes fMRI crudas a formato BIDS en macOS

Introducción

El **Brain Imaging Data Structure (BIDS)** es un estándar ampliamente adoptado para organizar y describir datos de neuroimagen, incluyendo fMRI, de forma consistente. Convertir sus datos fMRI "crudos" al formato BIDS facilita el uso de pipelines automatizados de preprocesamiento (como fMRIPrep) y el intercambio de datos con la comunidad científica. En este manual enseñaremos cómo convertir datos fMRI no procesados al formato BIDS, enfocándonos en herramientas compatibles con macOS y fáciles de usar, adecuadas para principiantes absolutos. Esto incluye tanto datos de fMRI en humanos como de animales (p. ej. ratas), asumiendo que los datos estén en formatos comunes como **DICOM**, **PAR/REC** (Philips) o **NIfTI** no estructurado, excluyendo formatos propietarios de Bruker (ya cubiertos en otros tutoriales del usuario).

¿Qué implican los datos "crudos"? Normalmente, los escáneres de MRI almacenan las imágenes en formato DICOM (un formato estándar médico). En investigación también es común encontrar datos en otros formatos propietarios (p. ej., PAR/REC de Philips) o ya convertidos a NIfTI pero sin una estructura organizada. El proceso de conversión a BIDS suele consistir en: (1) convertir los archivos crudos (p. ej., DICOM) a imágenes **NIfTI** con sus metadatos (archivos `.json` asociados), y (2) **renombrar y organizar** esos archivos en carpetas siguiendo las reglas BIDS (por ejemplo, `sub-001/ses-01/anat/sub-001_T1w.nii.gz`, etc.). Herramientas especializadas automatizan gran parte de este proceso. Casi todas utilizan internamente el convertidor **dcm2niix** de Chris Rorden para pasar de DICOM a NIfTI+JSON [1], evitando que el usuario deba hacerlo manualmente. Posteriormente agregan la capa de organización y nomenclatura BIDS requerida [2].

A continuación, presentaremos **cinco métodos populares** para lograr esta conversión en macOS. Se incluyen tanto herramientas de línea de comandos (CLI) como interfaces gráficas (GUI), priorizando aquellas de **instalación y uso sencillo**. Todas funcionan de forma nativa en macOS sin necesidad de entornos virtualizados complejos (Docker no será requerido en los pasos básicos, salvo que el usuario así lo prefiera). Las cinco herramientas seleccionadas son:

1. **HeuDiConv** – Conversión flexible mediante heurísticas (CLI en Python).
2. **Dcm2Bids** – Sencillo convertidor CLI que reorganiza NIfTI a BIDS (CLI en Python).
3. **BIDScoin** – Suite con interfaz gráfica para mapeo y conversión (Python, CLI+GUI).
4. **BIDS'em ALL** – Aplicación de escritorio con GUI (Java) para convertir DICOM a BIDS.
5. **ezBIDS** – Plataforma web para conversión asistida sin instalar nada (GUI en navegador).

En cada sección, explicaremos cómo descargar/instalar la herramienta en macOS y cómo usarla paso a paso para convertir datos fMRI crudos (incluyendo datos de ratas) al formato BIDS. Al final de la conversión, **se recomienda siempre validar** el conjunto de datos con la herramienta BIDS Validator para asegurar la conformidad con el estándar.

1. Conversión con HeuDiConv (CLI en Python)

HeuDiConv (Heuristic DICOM Converter) es una herramienta muy utilizada para convertir datos DICOM a BIDS de manera flexible. Se basa en **scripts heurísticos** en Python que definen las reglas de renombrado y organización, lo que le da gran potencia para adaptarse a distintas convenciones de adquisición [3]. HeuDiConv es útil tanto para fMRI humano como de animales, siempre que los datos provengan de DICOM (incluyendo DICOM obtenidos de estudios preclínicos). En el caso de datos ya en NIfTI, HeuDiConv puede organizarlos si se le provee la información necesaria, pero típicamente se usa sobre DICOM directamente.

Instalación en macOS: HeuDiConv está escrito en Python, por lo que la forma recomendada es instalarlo via pip. Previamente necesitamos instalar **dcm2niix** (la herramienta que HeuDiConv usa internamente para convertir DICOM a NIfTI). Puede instalar *dcm2niix* fácilmente en macOS, por ejemplo usando Homebrew (`brew install dcm2niix`) o con pip (`pip install dcm2niix`). Luego, para instalar HeuDiConv, abra la Terminal de macOS y ejecute:

```
pip install heudiconv
```

Esto instalará el comando `heudiconv`. Asegúrese de tener Python 3 instalado (macOS no lo incluye por defecto en versiones recientes, pero se puede obtener desde Python.org o via Homebrew).

Uso básico: HeuDiConv funciona especificando la ubicación de sus DICOM, un archivo de *heurística* que mapea los archivos a la estructura BIDS, y parámetros del sujeto/sesión. Por ejemplo, una ejecución típica podría ser:

```
heudiconv -d "/ruta/dicom/{subject}/*/*.dcm" -o /ruta/salida/BIDS \
  -f heuristic.py -s 001 -c dcm2niix -b --minmeta
```

Donde: - `-d` indica la ruta a los DICOM, usando `{subject}` como comodín para el ID del sujeto. - `-o` es la carpeta de salida donde se creará la estructura BIDS (p. ej. `.../BIDS/sub-001/...`).
- `-f` es el archivo de *heurística* en Python (en este ejemplo `heuristic.py`) que contiene las reglas de conversión.
- `-s` especifica el identificador del sujeto (001 en este caso).
- `-c dcm2niix` indica que use dcm2niix para la conversión de formato.
- `-b` le indica generar una estructura **BIDS completa** (con archivos como dataset_description.json, etc.) [4].
- `--minmeta` es una opción para que sólo incluya metadatos mínimos necesarios en los archivos JSON (evitando sobrecargarlos con campos DICOM irrelevantes) [5].

Al ejecutar este comando (adaptado a sus rutas y nombres), HeuDiConv leerá todos los DICOM del sujeto, aplicará la heurística y creará las carpetas y archivos en la carpeta de salida siguiendo BIDS. Por ejemplo, creará algo así como: `sub-001/ses-<etiqueta>/anat/sub-001_ses-<etiqueta>_T1w.nii.gz` para anatómicos, `sub-001/ses-<etiqueta>/func/sub-001_ses-<etiqueta>_task-<tarea>_bold.nii.gz` para funcionales, etc., dependiendo de lo definido en la heurística. También generará automáticamente archivos plantilla requeridos como `dataset_description.json`,

`participants.tsv`, `README`, etc. [6] [7]. **Nota:** Estos archivos JSON/TSV inicialmente estarán en blanco o con campos mínimos; el usuario debe completar información como el `dataset_description.json` antes de compartir los datos.

La parte más importante de HeuDiConv es el archivo de **heurística** (`heuristic.py`). Este es un script Python que define la lógica para asignar cada serie de DICOM a un tipo BIDS (anat, func, dwi, fmap) y nombrar los archivos resultantes. Para principiantes, escribir este archivo desde cero puede ser desafiante, pero existen **heurísticas de ejemplo** que se pueden reutilizar. Por ejemplo, la documentación oficial provee un ejemplo llamado *heuristic1.py* [3]. Si sus series DICOM incluyen en el campo "ProtocolName" o "SeriesDescription" información distinguible (p. ej., contienen palabras como "T1", "bold", "DWI"), entonces la heurística puede usar eso para clasificarlas.

Opción ReproIn: HeuDiConv incluye una heurística predefinida llamada **reproin** (`-f reproin`) que permite conversión *automática* si los nombres de las series siguen ciertas convenciones estandarizadas. Esta convención ReproIn propone nombrar las secuencias en el escáner con etiquetas que ya correspondan a entidades BIDS (por ej., `anat-T1w`, `func-bold_task-rest`, etc.) [8] [9]. Si usted controla la adquisición (por ejemplo, para futuros experimentos en ratas podría nombrar las secuencias siguiendo esta convención), HeuDiConv podrá mapearlas a BIDS sin un archivo heurístico personalizado. En ese caso, bastaría un comando simple:

```
heudiconv -f reproin --bids -o /ruta/salida/BIDS --files /ruta/dicom/001 --
minmeta
```

(asumiendo los DICOM del sujeto están en `/ruta/dicom/001`) [10]. Esto escaneará los DICOM, interpretará los nombres con la regla ReproIn y generará directamente la estructura BIDS [11].

Consejos para principiantes: Si bien HeuDiConv es muy potente, para un usuario novel puede ser un poco intimidante al principio debido a la necesidad de crear/editar la heurística. Sin embargo, su flexibilidad es útil cuando otras herramientas más automáticas no reconocen bien cierto estudio. Para iniciarse, se recomienda: 1) Usar heurísticas de ejemplo de estudios similares, 2) Probar primero con un sujeto para verificar la estructura generada, 3) Usar la opción `-b` para generar los archivos auxiliares BIDS, y 4) Ejecutar el **BIDS Validator** sobre la carpeta resultante para corregir cualquier error de nombre o campo faltante.

2. Conversión con Dcm2Bids (CLI en Python)

Dcm2Bids es otra herramienta popular y enfocada en la facilidad de uso para llevar sus datos crudos a BIDS. A diferencia de HeuDiConv (que usa un script Python libre), Dcm2Bids utiliza un enfoque basado en un archivo de **configuración JSON** sencillo, donde se definen las reglas de conversión sin necesidad de programar. Internamente también emplea `dcm2niix` para convertir DICOM a NIfTI, y luego **reorganiza los NIfTI generados** en la estructura BIDS especificada [12]. El objetivo de Dcm2Bids es hacer la conversión "lo más sencilla y amigable posible" para el usuario diario [13], reduciendo al mínimo la intervención manual en la creación de carpetas o renombrado de archivos.

Instalación en macOS: Dcm2Bids está escrito en Python, así que puede instalarlo fácilmente via pip o conda. Si tiene Python 3 instalado, simplemente ejecute en Terminal:

```
pip install dcm2bids
```

Esto instalará el comando `dcm2bids` y herramientas auxiliares. Alternativamente, los desarrolladores ofrecen **ejecutables precompilados para macOS** (a partir de la versión 3.x) que incluyen todo lo necesario [14]. Estos se pueden descargar desde la página de *Releases* del repositorio de GitHub de Dcm2Bids. Al descargar el paquete para macOS y descomprimirlo, obtendrá archivos binarios como `dcm2bids`, `dcm2bids_scaffold` y `dcm2bids_helper` [15] [16]. Puede mover estos binarios a alguna ruta incluida en su `$PATH` (por ejemplo `/usr/local/bin`) o ejecutarlos directamente indicando la ruta. La ventaja de usar estos binarios es que ya incorporan las dependencias (incluido dcm2niix, BIDS validator, etc. en la versión Docker/Apptainer), pero para propósitos de este manual asumiremos la instalación vía pip, que es sencilla.

Uso básico: Dcm2Bids suele usarse en tres pasos:

1. **Generar la estructura base BIDS y config de ejemplo:** Ejecute `dcm2bids_scaffold` para crear una carpeta de proyecto BIDS vacía con subdirectorios típicos y un archivo de configuración inicial. Por ejemplo:

```
dcm2bids_scaffold -o /ruta/proyectoBIDS
```

Esto creará en `/ruta/proyectoBIDS` subcarpetas como `sourcedata/`, `rawdata/` (o `BIDS` según versión) y un archivo de configuración JSON de muestra (`config.json`) dentro de `code/` [16].

1. **Identificar series y preparar el config:** A continuación, utilice `dcm2bids_helper` para analizar un conjunto de DICOM de ejemplo (generalmente de un sujeto) y obtener información de las series. Un comando típico:

```
dcm2bids_helper -d /ruta/DICOM/sujeto01 -o /ruta/proyectoBIDS
```

Esto ejecutará dcm2niix sobre los DICOM proporcionados, convertirá a NIfTI temporalmente y generará un archivo `dicom_metadata.tsv` o similar con los nombres de series, etiquetas, etc. También copiará muestras de los archivos NIfTI convertidos en una carpeta de *tmp* dentro del proyecto. Con esta información, usted edita el archivo de **configuración JSON** (`/ruta/proyectoBIDS/code/config.json`). En dicho JSON, para cada *serie* identificada, especificará a qué tipo BIDS corresponde y cómo nombrarla. Por ejemplo, puede mapear la serie cuyo `SeriesDescription` contenga "MPRAGE" a anat/T1w, o la serie "BOLD_resting" a func/rest. La sintaxis del JSON permite condiciones por palabras clave del nombre, por

número de serie, etc., y asignar campos BIDS como `Modality`, `TaskName`, `EchoTime` si es necesario. La documentación de Dcm2Bids brinda ejemplos detallados de cómo escribir estas reglas.

1. **Convertir todos los sujetos usando el config:** Finalmente, cuando el config está listo, ejecutamos el comando principal `dcm2bids` para procesar los datos completos. Un ejemplo de ejecución para un sujeto sería:

```
dcm2bids -d /ruta/DICOM/sujeto01 -p 01 -c /ruta/proyectoBIDS/code/config.json -
o /ruta/proyectoBIDS
```

Donde `-d` apunta al directorio con los DICOM del sujeto, `-p 01` es el identificador de participante que deseamos darle (esto producirá sub-01 en la estructura BIDS), `-c` proporciona la ruta al config JSON editado, y `-o` indica la carpeta base BIDS de salida (la misma que creó con scaffold) [17]. Al ejecutarlo, Dcm2Bids convertirá los DICOM a NIfTI con dcm2niix y luego *moverá/renombrará* esos NIfTI y JSON a la ubicación BIDS correcta según las reglas del config [12]. Repetiría este paso para cada sujeto (o hay formas de procesar múltiples sujetos en lote, e.g. con un script o utilizando `dcm2bids` en modo batch).

Al finalizar, obtendrá en `/ruta/proyectoBIDS` una estructura completa BIDS: subcarpetas `sub-01`, etc., con sus archivos `.nii.gz` y `.json` correctamente nombrados. **Dcm2Bids garantiza que los archivos cumplan con la nomenclatura BIDS**, aunque es recomendable correr el BIDS Validator para detectar si falta algún metadato obligatorio (por ejemplo, en fMRI de tarea se suele requerir un archivo `events.tsv` manualmente).

Ejemplo de configuración: Supongamos que para un experimento en ratas usted tiene series de MRI llamadas “T2w_TurboRARE” (anatómica) y “BOLD_rest_state” (funcional resting-state) en los DICOM. En el `config.json`, usted crearía entradas para mapearlas. Por ejemplo (pseudo-código JSON simplificado):

```
{
  "descriptions": [
    {
      "dataType": "anat",
      "modalityLabel": "T2w",
      "criteria": {"SeriesDescription": "T2w_TurboRARE"},
      "customLabels": "acq-rare"
    },
    {
      "dataType": "func",
      "modalityLabel": "bold",
      "criteria": {"SeriesDescription": "BOLD_rest"},
      "task": "rest"
    }
  ]
}
```

Esto le indica a Dcm2Bids que cualquier serie cuyo SeriesDescription contenga “T2w_TurboRARE” se guarde como un anat/T2w con etiqueta de adquisición “rare”, generando un nombre de archivo como `sub-01_acq-rare_T2w.nii.gz`. La segunda regla mapearía la serie BOLD resting a func/bold con `task-rest` en el nombre [18]. Dcm2Bids ofrece bastante flexibilidad con estas reglas, pudiendo usar también el *SeriesNumber*, *ProtocolName*, etc., si fuera más fiable que el nombre.

Ventajas para principiantes: Dcm2Bids evita la programación; solo requiere editar un JSON estructurado. Además, es **comunitario y bien documentado**: fue desarrollado con feedback de la comunidad para abarcar casos típicos manteniendo la sencillez [13]. Si sus datos DICOM están bien organizados (p. ej., separados por sujeto y sesión) y con descripciones consistentes, la configuración será muy directa. En caso de atascarse, los desarrolladores alientan preguntas en NeuroStars con la etiqueta `dcm2bids` [19], lo cual es útil.

Nota: Dcm2Bids también soporta contenedores Docker/Apptainer con todo incluido [20], lo que simplifica dependencias. Pero dado que nos enfocamos en macOS nativo y facilidad, puede no ser necesario usar Docker para la mayoría de usuarios de Mac (solo Python y dcm2niix instalados localmente).

3. Conversión con BIDScoin (Suite CLI+GUI en Python)

BIDScoin es una suite de herramientas en Python que se destaca por su enfoque *user-friendly* para convertir datos crudos a BIDS. A diferencia de las anteriores, BIDScoin combina **automatización basada en mapeos** e **interfaces gráficas interactivas** para que incluso usuarios sin experiencia en programación puedan realizar la conversión [21] [22]. Fue desarrollado en el Donders Institute (Radboud University) con el objetivo de eliminar la necesidad de lógica compleja en el código, sustituyéndola por un proceso de **descubrimiento de datos y mapeo editable por el usuario** [23].

Formatos soportados: BIDScoin es muy versátil. Acepta directamente **DICOM** (MRI, CT, PET), archivos Philips **PAR/REC**, archivos Siemens **Twix** (datos de espectroscopía MR), **NIfTI** sin estructurar, entre otros [24]. Esto cubre los formatos crudos más comunes, por lo que es ideal si sus datos de rata están en DICOM (o incluso si tuviera PAR/REC de algún equipo Philips preclínico). BIDScoin se basa en *plugins* para cada formato, usando por defecto `dcm2niix` para DICOM/PAR [25], `spec2nii` para espectroscopía, etc., de forma que siempre obtengamos NIfTI y JSON correctos.

Instalación en macOS: Al ser Python, la instalación recomendada es vía pip. Primero asegúrese de tener Python ≥3.8. En Terminal ejecute:

```
pip install bidscoin[dcm2niix2bids]
```

El *extra* `[dcm2niix2bids]` le indica que incluya el plugin de DICOM (dcm2niix) automáticamente [26]. Esto instalará los comandos principales de BIDScoin: `bidscoin`, `bidsmapper`, `bidseditor` y `bidscoiner`. Alternativamente, podría hacer `pip install bidscoin` a secas y luego instalar *dcm2niix* por separado si lo prefiere, pero la forma mostrada arriba es más sencilla ya que pip descargará dcm2niix por usted. Una vez instalado, puede verificar ejecutando `bidscoin --help` para ver si muestra información (si ve error de Qt en Mac M1, podría necesitar instalar una versión compatible, pero en general la última versión funciona en macOS moderno).

Flujo de trabajo (Workflow): BIDScoin se opera típicamente con **dos comandos y una intervención manual**: 1. Ejecutar `bidsmapper` para analizar los datos fuente. 2. Usar la interfaz gráfica `bidseditor` para verificar/ajustar el mapeo propuesto. 3. Ejecutar `bidscoiner` para realizar la conversión final con las definiciones aprobadas.

En realidad, el `bidsmapper` lanzará automáticamente el `bidseditor` por usted al terminar, facilitando el proceso [27]. Así, convertir sus datos puede reducirse a correr dos comandos en la Terminal. Por ejemplo:

```
bidsmapper /ruta/datos_origen /ruta/BIDS_destino
```

(El `bidsmapper` *escanea la carpeta de datos originales y genera un archivo de mapeo estudiantil, luego abre el editor gráfico)* [28]. En la GUI del **bidseditor**, usted verá una lista de los tipos de datos detectados (por ejemplo, "Series 001 – RARE_T2", "Series 002 – bold_rest") en una columna, y la propuesta de nombre BIDS en la columna de al lado [29]. BIDScoin habrá hecho "conjeturas inteligentes" sobre cómo mapear cada tipo de dato usando la información disponible (nombre de archivo, atributos DICOM como ProtocolName, etc.) [30]. Los elementos con texto verde indican que ya cumplen BIDS; en rojo, que requieren intervención (por ejemplo, falta especificar alguna etiqueta obligatoria) [31]. Usted puede editar cada mapeo directamente en la GUI: cambiar el sujeto o sesión asignado, renombrar campos, excluir datos irrelevantes (marcándolos para no convertir) [32] [33]. La interfaz provee menús desplegables con las opciones válidas según el esquema BIDS para evitar errores [33]. Una vez satisfecho, guarda los cambios y cierra el editor.

Luego ejecuta el segundo comando:

```
bidscoiner /ruta/datos_origen /ruta/BIDS_destino
```

Esto leerá el archivo de mapeo refinado y realizará la conversión completa de todos los sujetos/series al formato BIDS deseado [28] [34]. Al finalizar, tendrá en `/ruta/BIDS_destino` la estructura BIDS con sus sub- y ses- correspondientes, y todos los archivos convertidos (NIfTI+JSON, etc.). Si agrega más sujetos en el futuro, solo necesita volver a ejecutar `bidscoiner` para esos nuevos datos, **mientras no cambie el protocolo**; si el protocolo cambia (nuevos tipos de secuencia), es recomendable ejecutar de nuevo `bidsmapper` para incorporarlos y editarlos en la GUI [35].

Interfaz gráfica de BIDScoin (bidseditor) mostrando a la izquierda los tipos de datos detectados en la fuente y a la derecha los nombres BIDS propuestos, con indicadores de campos completados o pendientes. [29] [33]

En cuanto a los **datos de ratas**, BIDScoin no discrimina por especie durante la conversión; simplemente creará sujetos `sub-xx`. Sin embargo, para cumplir completamente con BIDS, deberá especificar la especie en el archivo `participants.tsv` o en `dataset_description.json` (por ejemplo, usando el campo `Species` y `ParticipantSpecies` si sigue extensiones BIDS recientes). Esto es algo a revisar tras la conversión, ya que actualmente BIDScoin se centra en la estructura de archivos más que en editar los archivos de participantes.

BIDScoin también incluye utilidades opcionales ("BIDS-apps") para tareas post-conversión como deface (anonimizar caras en anatómicos), combinar echos de multi-echo, generar informes de calidad, etc. [25] [36].

Estas no son obligatorias pero pueden ser útiles. Por ejemplo, podría ejecutar `bidscoin_deface` si quisiera borrar la identidad de cráneos en ratas (no tan relevante como en humanos, pero la opción existe).

En resumen, BIDScoin ofrece una ruta muy amigable: automatiza la detección y propone un plan de conversión, permite al usuario refinarlo visualmente, y luego convierte todo sin necesidad de escribir código. Es ideal si desea una solución *semi-automática* guiada. La curva de aprendizaje es baja, pues **no requiere conocimiento de programación**, aunque conocer las categorías BIDS (anat, func, etc.) es útil para tomar decisiones en el editor [22]. Por último, conviene hacer una validación BIDS final: BIDScoin genera una estructura generalmente correcta, pero siempre es bueno correr el BIDS Validator manualmente para confirmar que no falten descripciones de tareas, TR en archivos JSON, etc., antes de proceder al análisis.

4. Conversión con BIDS'em ALL (GUI de escritorio, Java)

BIDS'em ALL es una aplicación de escritorio con interfaz gráfica pura, diseñada para convertir automáticamente carpetas de imágenes DICOM a un conjunto de datos BIDS, con interacción mínima del usuario. Es una solución *multiplataforma* (Windows, Mac, Linux) desarrollada en Java, por lo que funciona en macOS siempre que tengamos Java instalado [37]. Para principiantes absolutos que prefieren evitar la terminal, esta herramienta ofrece una experiencia point-and-click muy sencilla.

Instalación en macOS: No se "instala" como tal, sino que se descarga un archivo JAR ejecutable. Los pasos son: - Asegúrese de tener **Java Runtime Environment (JRE) 8** o superior en su Mac. Si no, descárguelo de java.com [38] e instálelo (en macOS usualmente es un paquete `.pkg`). - Descargue el archivo `BIDS\` *em* `ALL.jar` desde la página oficial del proyecto. En la página de BIDS'em ALL hay un enlace de descarga directa [39]. También puede provenir de su repositorio o NeuroStars (donde el autor lo compartió). - Una vez descargado el `bidsemall.jar`, simplemente **haga doble click** sobre él (o ejecútelo con `java -jar bidsemall.jar` desde Terminal) [40]. Se abrirá la interfaz gráfica de BIDS'em ALL.

Uso de la aplicación: Al abrir BIDS'em ALL, verá una ventana con opciones para seleccionar la carpeta de entrada y la carpeta de salida. El flujo típico de conversión es: 1. **Seleccionar datos de entrada:** Clic en "Input DICOM directory" y elija la carpeta raíz que contiene sus archivos DICOM crudos (puede ser una carpeta con subcarpetas por sujeto; la herramienta escaneará recursivamente). No necesita un `DICOMDIR` especial, puede manejar múltiples subcarpetas [41]. 2. **Seleccionar carpeta de salida:** Clic en "Output directory" y elija dónde quiere que se guarde el dataset BIDS resultante. 3. **Configurar identificador de sujetos:** Al darle a "Convert", si la herramienta detecta múltiples sujetos, le preguntará cómo asignar el identificador BIDS para cada uno. De forma predeterminada usará el Patient ID encontrado en los DICOM para nombrar `sub-XXX` [42]. Usted puede aceptar eso o cambiarlo (por ejemplo, podría querer renombrar "Patient 001" a `sub-rat01`). 4. **Clasificar series:** La primera vez que procese un cierto protocolo, BIDS'em ALL le presentará cada **serie** encontrada que no reconozca previamente, y le pedirá que indique de qué se trata en términos BIDS. Por ejemplo, puede aparecer un diálogo diciendo: *Serie "TurboRARE T2" – seleccione tipo de dato:* y usted escogería "anat" y luego "T2w" de listas desplegables, completando además campos obligatorios (como `EchoTime` si aplica, etc.) [43]. Para una serie "BOLD Resting State", escogería "func" -> "bold" y especificaría `Task=rest` en el formulario. Los campos obligatorios se resaltan en rojo para que no se omita nada esencial [44]. Esta configuración solo se pregunta la **primera vez** para cada nombre de serie distinto; BIDS'em ALL guardará sus elecciones en un archivo de configuración (`DIC.yml`) dentro del directorio de la app [45]. De este modo, si en el futuro convierte más sujetos con las mismas secuencias, ya

no preguntará de nuevo – aplicará automáticamente la misma clasificación [46]. 5. **Conversión automática:** Tras clasificar las series nuevas, la herramienta **organiza los DICOM** en una estructura tipo BIDS bajo una carpeta `sourcedata` temporal y luego lanza *dcm2niix* para convertirlos a NIfTI, moviéndolos a la carpeta BIDS final (`rawdata` o directamente la raíz BIDS) [47]. Este proceso es totalmente automático; verá una barra de progreso mientras convierte cada serie. 6. **Resultado:** Cuando termine, usted tendrá en la carpeta de salida una estructura BIDS completa con subcarpetas por sujeto (`sub-001`, `sub-002`, etc.), sus archivos NIfTI `.nii.gz` y `.json` correspondientes. BIDS'em ALL generará también los archivos complementarios requeridos (dataset_description.json, participantes, etc.) con la información que tenga disponible.

La filosofía de BIDS'em ALL es que el usuario **solo toma decisiones mínimas** (asignar IDs y clasificar series) a través de una GUI intuitiva, y la aplicación hace el resto en 6 pasos predefinidos [48]. Por ejemplo, en el paso 3 cuando pide la clasificación de una serie, presentará campos obligatorios en rojo hasta que se llenen [49], guiando así al usuario para cumplir BIDS. En pasos subsiguientes, cualquier serie igual se procesa del mismo modo automáticamente usando la configuración guardada [46], lo que acelera conversiones por lotes.

BIDS'em ALL soporta macOS sin inconvenientes – simplemente requiere la máquina virtual de Java para correr el `.jar` [37]. Dado que es una herramienta joven (versión 1.0 lanzada en 2023 [50]), es posible que en el futuro se agreguen más funciones. Actualmente, **no incluye un validador BIDS integrado**, así que es recomendable que, tras la conversión, ejecute usted mismo el BIDS Validator (puede usar la versión web o instalarlo) para verificar que todo esté perfecto. No obstante, si clasificó correctamente las series, BIDS'em ALL crea todos los archivos necesarios y respeta la estructura, por lo que típicamente debería **pasar la validación sin errores**.

Caso de uso ratas: Si está convirtiendo datos de fMRI de ratas, la interfaz de BIDS'em ALL le permitirá, por ejemplo, definir `anat-T2w` para sus secuencias anatómicas (o `T1w` si fuera el caso), `func-bold` con un nombre de tarea (o `rest` si es reposo) para sus fMRI funcionales, etc., igual que para humanos. Solo asegúrese de poner un identificador de sujeto adecuado (quizá "rat1", "rat2"; la herramienta no pone restricciones, pero por consistencia use solo caracteres alfanuméricos). Más tarde, puede editar el `participants.tsv` para añadir columna de especie = Rat (o lo que corresponda).

En resumen, **BIDS'em ALL** es de los métodos más directos para un principiante: todo con ventanas gráficas, sin código ni terminal. Es perfecto si quiere convertir unos cuantos sujetos manualmente. Para centenares de sujetos, quizás herramientas como HeuDiConv o Dcm2Bids permitirían automatizar más con scripting; pero para volúmenes pequeños o moderados, la comodidad de BIDS'em ALL es difícil de superar. Y dado que guarda la configuración de series, manejar un estudio completo con la misma configuración experimental se vuelve trivial tras configurar las primeras series.

5. Conversión con ezBIDS (GUI web, sin instalación)

ezBIDS es una herramienta única en esta lista: es completamente web-based, lo que significa que se usa a través del navegador y **no requiere instalación** alguna [51]. Está pensada específicamente para usuarios con poca experiencia técnica, proporcionando una interfaz paso a paso que guía el proceso de conversión de datos a BIDS de forma interactiva. Al ser web, también aprovecha servidores remotos para realizar las

conversiones, así que su computadora local no necesita tener instaladas herramientas como dcm2niix o Python – solo requiere conexión a internet y un navegador moderno.

Acceso a ezBIDS: Puede acceder a ezBIDS visitando su página web oficial (por ejemplo, desde la documentación se enlaza a *ezBIDS homepage*, que típicamente será un enlace desde la página de Brainlife.io o un subdominio dedicado). Es posible que necesite registrarse con una cuenta gratuita (Brainlife) para utilizarla, ya que sus datos se subirán a un servidor seguro [52]. Una vez en la página principal de ezBIDS, verá un botón "Get Started" – al pulsarlo, se inicia el flujo de conversión [53].

Proceso guiado paso a paso: ezBIDS divide la conversión en una serie de pasos claros, presentados como páginas secuenciales en la web. A continuación resumimos las etapas principales:

1. **Subir datos de imagen:** El primer paso le pedirá seleccionar o arrastrar sus archivos de imagen crudos para subirlos al servidor seguro de ezBIDS [54]. Puede subir carpetas DICOM (comprimidas en un .zip) o incluso NIfTI no organizados. La documentación enfatiza que la transferencia es cifrada y segura, y que usted mantiene control sobre los datos [51] [55]. Dependiendo del tamaño, la subida puede tardar un tiempo; una vez completada, ezBIDS procede al siguiente paso.

2. **Descripción del dataset:** Se le solicitará ingresar información general del estudio: nombre del dataset, descripción, investigadores, etc., equivalente a llenar el archivo `dataset_description.json` [56]. Esto solo se hace la primera vez para un nuevo estudio; ezBIDS guarda esta info en una *plantilla* para reutilizarla si procesa datos similares más adelante [57].

3. **Identificación de sujetos y sesiones:** La herramienta examina los datos subidos y detecta cuántos sujetos y (si corresponde) sesiones hay, proponiendo IDs para cada uno [58]. En esta página usted puede editar esos identificadores. Por ejemplo, si sus archivos tenían IDs poco claros, puede renombrarlos a `sub-001`, `sub-002`, etc., de forma consistente. EzBIDS permite renombrado manual o semiautomático agrupando por patrones [58]. Para datos de ratas, podría aprovechar para asignar IDs significativos (p. ej., `sub-rat01`), aunque BIDS oficialmente esperaría un número; cualquier string sin espacios funciona siempre que sea consistente.

4. **Agrupación y mapeo de series (Series Mapping):** EzBIDS luego agrupa los archivos en *series* presumiblemente correspondientes a adquisiciones específicas [59]. Usa metadatos DICOM (Serie Description, TR, TE, etc.) para inferir qué archivos pertenecen a la misma secuencia y qué tipo de imagen es [18]. Por ejemplo, puede agrupar todos los volúmenes de una secuencia BOLD juntos, distinguir un anat T1 de un T2 por su nombre, etc. Estas agrupaciones se muestran al usuario junto con la **"mejor suposición"** de ezBIDS sobre su correspondencia BIDS [18]. La interfaz presentará algo como: Grupo 1 – 128 archivos – posiblemente `anat/T1w`; Grupo 2 – 300 archivos – posiblemente `func/bold (task-rest)`, etc. EzBIDS marca con símbolos de advertencia (amarillo) o error (rojo) los casos donde no está segura [60] [61]. El usuario puede entonces asignar o corregir las etiquetas BIDS a cada grupo usando menús desplegables. Por ejemplo, si ezBIDS no pudo adivinar el TaskName de una serie funcional, pondrá un aviso rojo y usted deberá especificarlo (e.g. "rest" o "maze", según el experimento) [62]. Si agrupó mal alguna serie, también puede ajustar la asignación de archivos a grupos mediante la interfaz.

5. **Mapeo de archivos de eventos (si aplica):** Si subió archivos de **eventos/estimulación** asociados a fMRI (p. ej., registros de estímulos en formato CSV, TSV, etc.), ezBIDS le guiará para integrarlos [63].

Podrá indicar qué archivo de eventos corresponde a qué tarea y run, y mapear columnas de tiempo de inicio (onset) y duración al formato estándar de BIDS (`events.tsv`). EzBIDS incluso ofrece ayuda si sus archivos de eventos no tienen directamente duración: permite calcularla restando columnas de inicio/fin, etc. [64]. Esta característica es muy útil y reduce la necesidad de procesar manualmente los archivos de comportamiento. (En estudios con ratas, este paso podría no aplicarse si no hay tareas definidas, pero si tuviera por ejemplo registros de estímulos olfativos, podría cargar un CSV con tiempos y ezBIDS lo convertiría al formato BIDS apropiado).

6. **Revisión del dataset:** Antes de finalizar, ezBIDS presenta una página de **resumen** donde lista todos los archivos e imágenes tal como quedarían organizados [65]. Aquí usted puede hacer ediciones finales, como marcar alguna serie para *excluir* (por ejemplo, imágenes de localización o exploraciones donde hubo mucho movimiento) [66]. Esta es la última oportunidad de ajuste manual. EzBIDS aplica automáticamente algunas reglas, por ejemplo, asigna números de `run-` si detecta múltiples ocurrencias de la misma serie en una sesión [67], o advierte si cierta secuencia parece incompleta (menos volúmenes de los esperados según TR y duración) para que el usuario confirme si debe excluirse [68].

7. **Opcional – Desface de anatómicos:** EzBIDS ofrece un paso opcional para **"deface"** las imágenes anatómicas (quitar rasgos faciales) [69]. En datos de animales esto usualmente no es relevante (no hay identidad que proteger), pero la opción está disponible. Se puede elegir entre algoritmos (Quickshear o pyDeface) [70]. Este proceso corre en el servidor y puede tardar unos minutos por sujeto.

8. **Información de participantes:** Se presenta un formulario para agregar metadata de sujetos en el `participants.tsv`: por defecto ezBIDS intenta rellenar sexo y edad desde los DICOM si existen (suelen venir en Patient Characteristics) [71]. Usted puede verificar y añadir otros campos como peso, cepa (strain) en caso de animales, etc. Para ratas, podría añadir columna "species" con "Rat" aquí. EzBIDS permite agregar columnas libremente para cualquier dato demográfico que quiera incluir [71].

9. **Finalizar y descargar:** En el último paso, se pulsa el botón **Finalize**. EzBIDS entonces aplica todas las decisiones y genera el dataset BIDS completo en el servidor [53]. Inmediatamente corre el **BIDS Validator** sobre el conjunto y muestra el resultado en pantalla [72]. Los errores (rojo) significarían que algo no quedó compliant y debe corregirse antes de poder continuar; las advertencias (amarillo) son opcionales pero conviene revisarlas [73]. Si hubo errores, puede volver atrás en la interfaz para corregir (ezBIDS permite navegar entre los pasos previos, luego habría que pulsar "Rerun Finalize" para validar de nuevo) [74]. Una vez el dataset está libre de errores, ya puede **descargar** el paquete BIDS (un archivo .zip con toda la estructura) a su computadora con el botón "Download BIDS" [75]. Alternativamente, ezBIDS ofrece enviar directamente los datos a repositorios abiertos como OpenNeuro o a su espacio en Brainlife.io [76], si así lo desea, lo cual facilita compartirlos o procesarlos en la nube.

Pantalla final de ezBIDS mostrando a la izquierda la estructura de carpetas BIDS generada (sub-001, etc. con sus archivos) y a la derecha el resultado de la validación BIDS, indicando que no hay errores. [72] [75]

Plantillas reutilizables: Una gran ventaja de ezBIDS es que, tras procesar una sesión o dataset, genera automáticamente un archivo de **configuración plantilla** (`ezBIDS_template.json`) [77]. Este JSON

almacena todas las correspondencias que usted confirmó (mapping de series, nombres de tareas, etc.). Si en el futuro tiene más datos del mismo experimento (p. ej., nuevas ratas escaneadas con idéntico protocolo), puede cargar esos datos *junto con* la plantilla en ezBIDS, y el sistema aplicará todas las mismas clasificaciones automáticamente [78] [79]. Solo intervendría usted si surge algo nuevo. Esto acelera muchísimo el procesamiento de estudios longitudinales o con muchos sujetos. Básicamente, la primera vez es un poco interactiva, pero las siguientes se vuelven semiautomáticas gracias a la plantilla.

Consideraciones de privacidad: Dado que se suben datos a un servidor, ezBIDS implementa varias medidas: los datos se almacenan de forma privada (solo accesibles a su cuenta), y puede borrarlos después de convertir. Aun así, si trabaja con datos sensibles, es recomendable verificar que estén anonimizados antes de subir (ezBIDS tiene la opción de deface para anatómicos humanos). Para datos preclínicos de ratas, la privacidad no suele ser un problema, pero igual es bueno saber que los identificadores personales (como PatientName en DICOM) podrían estar en los headers – ezBIDS en su proceso los sustituye por los nuevos IDs BIDS y puede omitir información confidencial.

Ventaja para principiantes: EzBIDS prácticamente le "toma de la mano" durante el proceso. Proporciona explicaciones en cada paso, resaltando con asteriscos campos obligatorios [80], dando **mensajes informativos y de advertencia** para guiar sus decisiones [60] [61], y asegurando al final que el dataset pase la validación BIDS [72]. Según sus autores, pretende eliminar la necesidad de conocer a fondo la especificación BIDS para poder convertir los datos correctamente [81] [82]. Es ideal si usted no se siente cómodo editando archivos de configuración o usando la terminal. A cambio, hay que invertir tiempo subiendo datos y revisando páginas web, lo cual es razonable para datasets pequeños/medianos. Para un dataset enorme (cientos de GB), quizá una herramienta local sería más práctica para no mover tantos datos por internet.

En conclusión, **ezBIDS** es una herramienta potente para principiantes: con cero instalación y mediante un flujo visual, logra convertir datos crudos a BIDS cumpliendo con las reglas, incluso convirtiendo archivos de eventos y validando el resultado. Es difícil lograr más simplicidad que esta para organizar un estudio fMRI. Recomendamos ezBIDS especialmente si está dando sus primeros pasos con BIDS o si quiere asegurarse de no olvidar ningún detalle obligatorio en la conversión.

Conclusiones y pasos siguientes

Hemos explorado cinco métodos eficaces para convertir datos fMRI crudos a formato BIDS en un entorno Mac, privilegiando la facilidad de instalación y uso para principiantes. Para recapitular:

- **HeuDiConv:** Ofrece máxima flexibilidad mediante heurísticas personalizables. Requiere algo de configuración técnica, pero es muy poderoso cuando se tienen protocolos complejos o altamente personalizados [3]. Útil a largo plazo si desea automatizar completamente conversiones repetitivas mediante scripting. Apto para usuarios dispuestos a ensuciarse las manos con un poco de Python.

- **Dcm2Bids:** Un equilibrio entre automatización y simplicidad. Mediante un archivo JSON de configuración, permite mapear series a BIDS de forma declarativa [12]. Ideal si sus datos siguen convenciones razonablemente consistentes, ya que con unas pocas reglas podrá convertir muchos sujetos. Muy amigable para iniciarse, con buen soporte comunitario.

- **BIDScoin:** Herramienta moderna con GUI interactiva que simplifica la conversión descubriendo automáticamente tipos de datos y solicitando al usuario solo aclaraciones puntuales [22]. Perfecta si quiere rapidez en montar la estructura BIDS sin escribir código. Soporta múltiples formatos (DICOM, PAR/REC, etc.), cubriendo necesidades tanto clínicas como preclínicas. Requiere instalar Python/Qt, pero luego la experiencia es muy cómoda.

- **BIDS'em ALL:** Aplicación de escritorio Java completamente gráfica. Permite conversiones rápidas *ad-hoc*, preguntando al usuario por la clasificación de cada secuencia y recordando esas elecciones [83]. Recomendable para usuarios que prefieren un entorno visual y tienen datasets de tamaño pequeño a mediano. No necesita conocimientos técnicos, solo seguir los diálogos.

- **ezBIDS:** Plataforma web paso-a-paso. Sin instalar nada, guía al usuario mediante formularios y visualizaciones a través de todo el proceso [84]. Garantiza conformidad BIDS al 100% gracias a sus validaciones integradas [72]. Es la opción con menor barrera de entrada – ideal para un primer intento de convertir sus datos o para asegurar que no se omita ningún detalle. Útil también si desea compartir o procesar datos en la nube tras la conversión.

Todas estas herramientas sirven tanto para datos de fMRI **humanos** como de **animales (rata)**, siempre y cuando los datos estén en formatos estándar (DICOM principalmente). En BIDS, la diferenciación de especie se hace vía metadatos (no en la estructura de carpetas), así que después de convertir, recuerde anotar la especie "Rat" en el archivo de participantes o en `dataset_description.json` según corresponda (algunas extensiones BIDS proponen campos `Species` y `ParticipantSpecies` para esto). Ninguna de las herramientas presentadas incorpora explícitamente el manejo de especies porque la organización de archivos es la misma independientemente de si el sujeto es humano o animal – simplemente **un sujeto es una carpeta** `sub-XX`. Es responsabilidad del investigador aclarar la naturaleza de esos sujetos en la documentación del dataset.

Por último, un **paso fundamental** después de cualquier conversión es correr el **Validador BIDS** sobre el dataset resultante (muchas herramientas ya lo integran o generan reportes). Esto le informará si hay errores o advertencias que atender. Por ejemplo, podría avisarle si olvidó incluir un campo obligatorio en algún JSON (como RepetitionTime en fMRI), o si algún archivo no sigue la nomenclatura. Todas las herramientas intentan producir datos válidos de primera mano – Dcm2Bids y BIDScoin se alinean a las especificaciones, ezBIDS explícitamente chequea compliance, etc. – pero como buena práctica, **valide sus datos** antes de alimentar un pipeline de análisis.

Con los datos ya en **formato BIDS**, estará listo para aplicar pipelines como **fMRIPrep**, **MRIQC**, u otros análisis sabiendo que la estructura de sus datos es estándar. Por ejemplo, fMRIPrep podrá leer directamente la carpeta BIDS que creó con cualquiera de estos métodos y ejecutar el preprocesamiento de principio a fin, ya que encontrará los archivos en los lugares esperados [85]. Esto le ahorrará mucho tiempo en el largo plazo y le permitirá compartir su dataset con colaboradores o en repositorios públicos con facilidad.