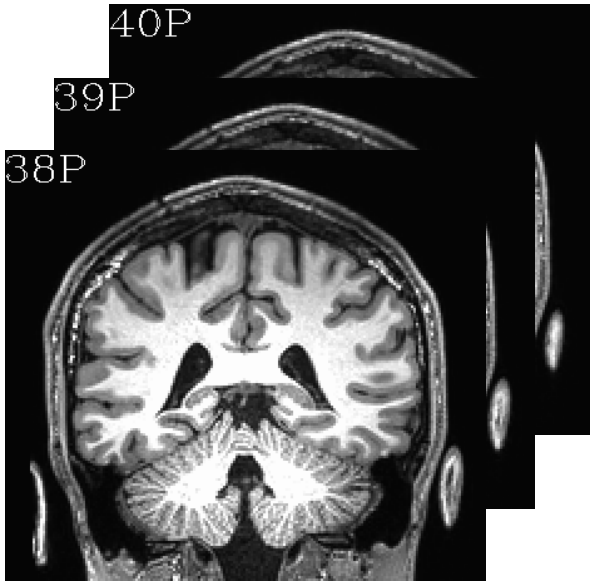
What is a volumetric data set?

Abbrevs used here

abbrev	= abbreviation
AKA	= also known as
anat	= anatomical
diff	= difference
dset	= dataset
e.g.	= exempli gratia (= “for example”)
EPI	= echo planar image
Ex	= example
FOV	= field of view
i.e.	= id est (= “that is”)
ijk	= coordinate indices (integer)
NB	= nota bene (= “note well”)
phys	= physics or physical
ref	= reference
subj	= subject
vol	= volume
vox	= voxel(s)
xyz	= physical coordinates (units of mm)

Volumetric data structure

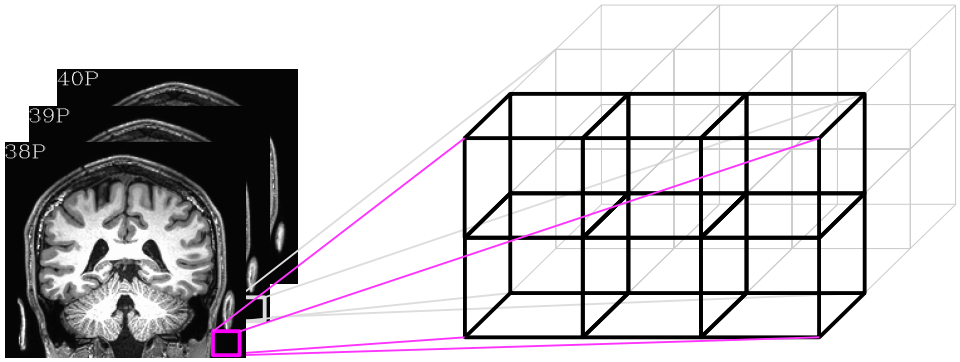
- What is a volumetric data set?



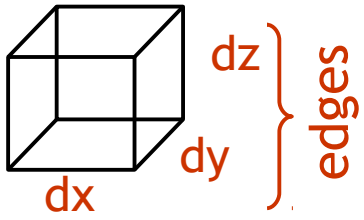
Volumetric data structure

- What is a volumetric data set?
  → *It a **grid** made up of voxels (basic case: 3D).*

3D grid

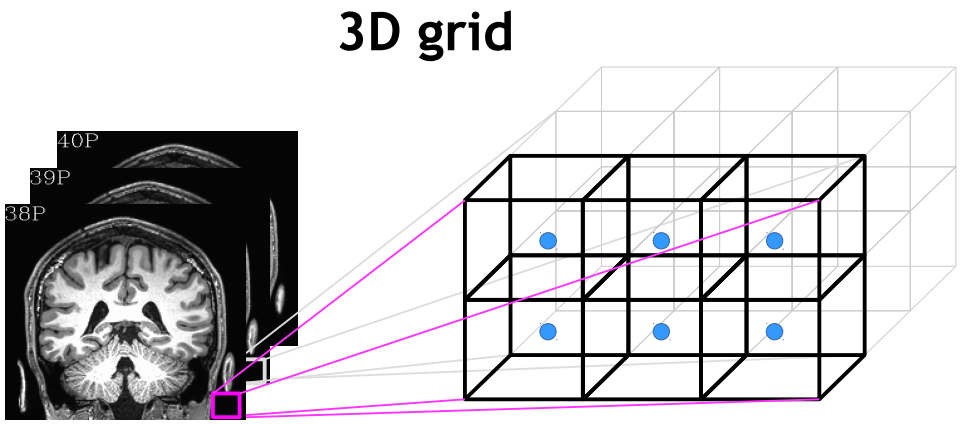


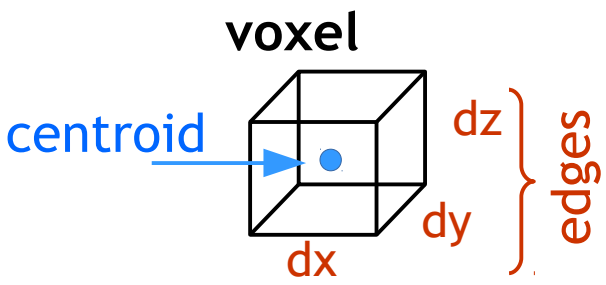
voxel



Volumetric data structure

- What is a volumetric data set?
  → *It a **grid** made up of voxels (basic case: 3D).*



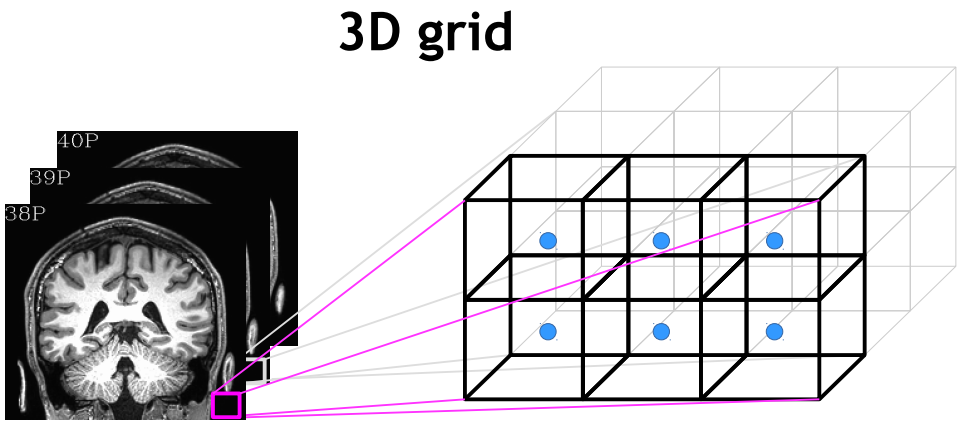


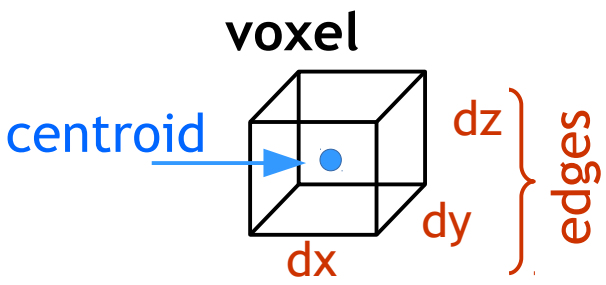
<u>2 ways to describe each voxel location</u>

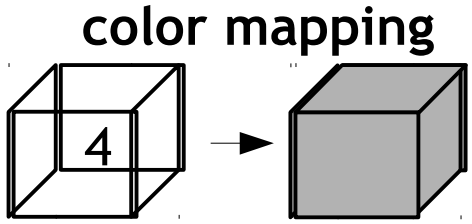
+ "*i, j, k*": integer indices, counting voxels from one corner
  - independent of voxel size (just storage indices on disk)
+ "*x, y, z*": units of mm, physical location of voxel *centroid*
  - depends on voxel size *(dx, dy, dz)*

Volumetric data structure

- What is a volumetric data set?
  → *It a **grid** made up of voxels (basic case: 3D).*
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).







<u>2 ways to describe each voxel location</u>

+ "*i, j, k*": integer indices, counting voxels from one corner
  - independent of voxel size (just storage indices on disk)
+ "*x, y, z*": units of mm, physical location of voxel *centroid*
  - depends on voxel size *(dx, dy, dz)*

Volumetric data structure

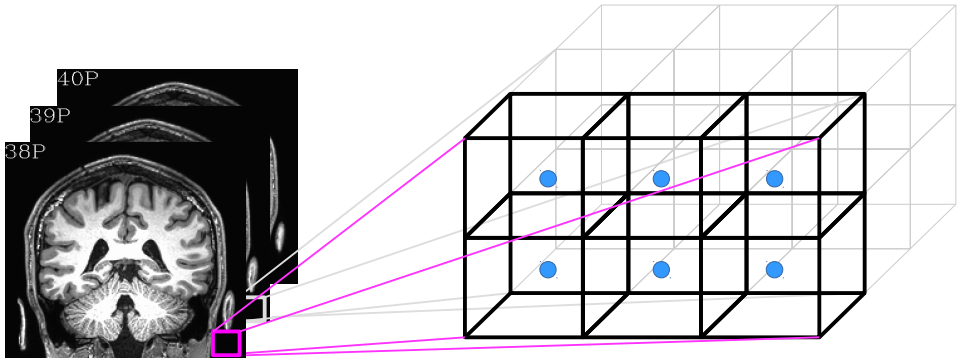
- What is a volumetric data set?

  → *It a **grid** made up of voxels (basic case: 3D).*

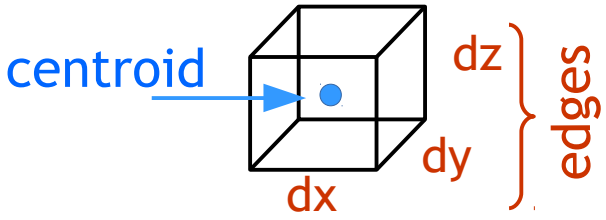
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).

  A time series data set is an ordered set of 3D vols (→ a '4D data set').

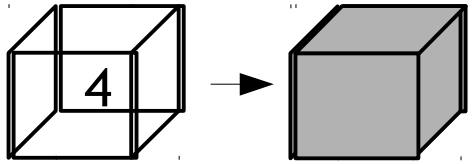
3D grid



voxel



color mapping



<u>2 ways to describe each voxel location</u>

+ "*i, j, k*": integer indices, counting voxels from one corner
  - independent of voxel size (just storage indices on disk)
+ "*x, y, z*": units of mm, physical location of voxel *centroid*
  - depends on voxel size *(dx, dy, dz)*

Volumetric data structure

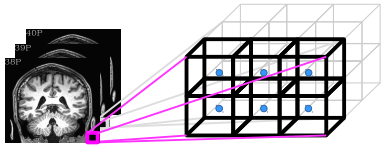
- What is a volumetric data set?

  → *It a **grid** made up of voxels (basic case: 3D).*

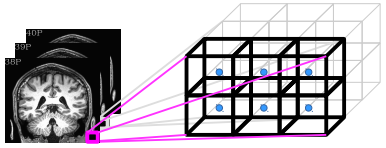
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).

  A time series data set is an ordered set of 3D vols (→ a '4D data set').

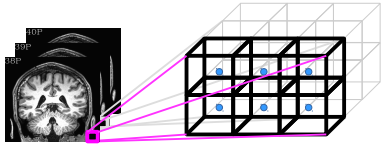
3D grid



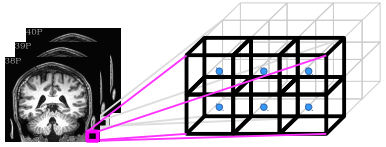
… at *t*=0



… at *t*= 1TR



… at *t*= 2TR



… at *t*= 3TR

…

Volumetric data structure

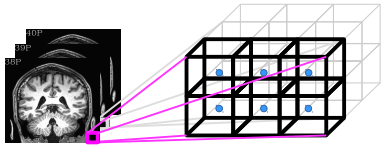
- What is a volumetric data set?

  → *It a **grid** made up of voxels (basic case: 3D).*

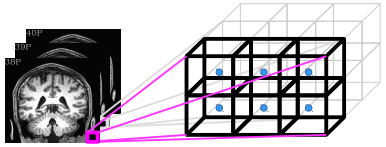
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).

  A time series data set is an ordered set of 3D vols (→ a '4D data set').

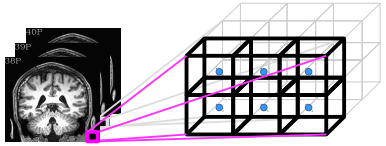
3D grid



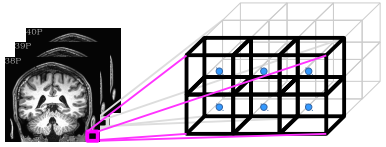
… at *t*=0



… at *t*= 1TR



… at *t*= 2TR



… at *t*= 3TR

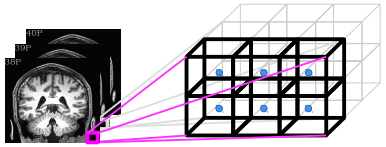
…

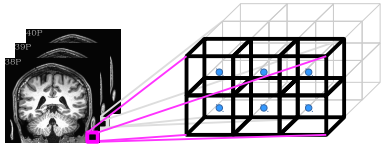
3D grid + time dimension
→ **"4D data set"**
+ Can talk about time as "*t*" in physical units of seconds, or as "*n*" in index units of simple counting.

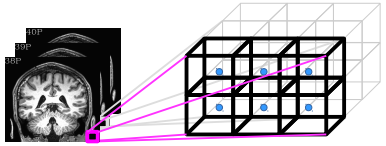
Volumetric data structure

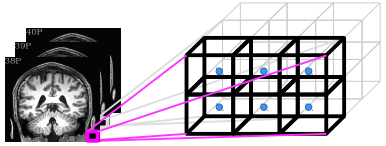
- What is a volumetric data set?
  → *It a **grid** made up of voxels (basic case: 3D).*
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).
  A time series data set is an ordered set of 3D vols (→ a '4D data set').

3D grid


… at *t*=0

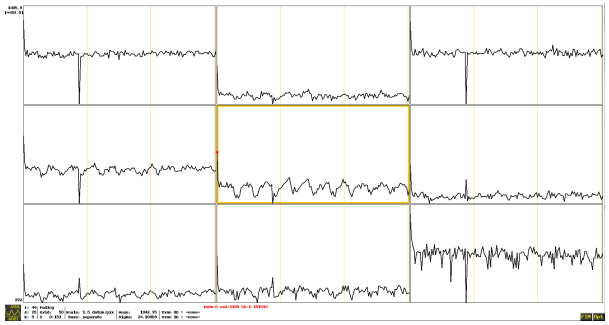

… at *t*= 1TR


… at *t*= 2TR


… at *t*= 3TR

…

3D grid + time dimension
→ **"4D data set"**
+ Can talk about time as "*t*" in physical units of seconds, or as "*n*" in index units of simple counting.
+ Also say that each voxel contains a "time series", e.g.:



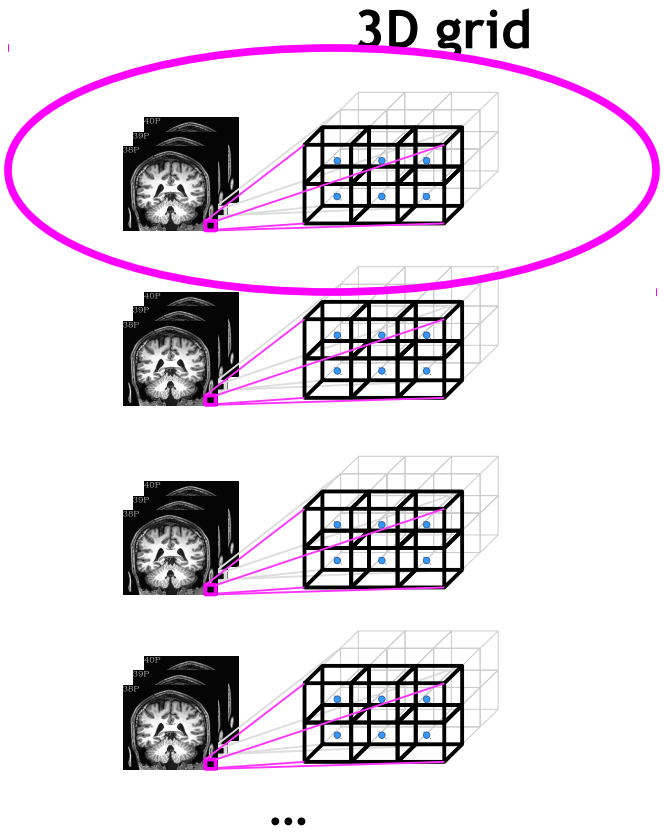
AFNI terminology sidenote

- What is a volumetric data set?

  → *It a **grid** made up of voxels (basic case: 3D).*

  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).

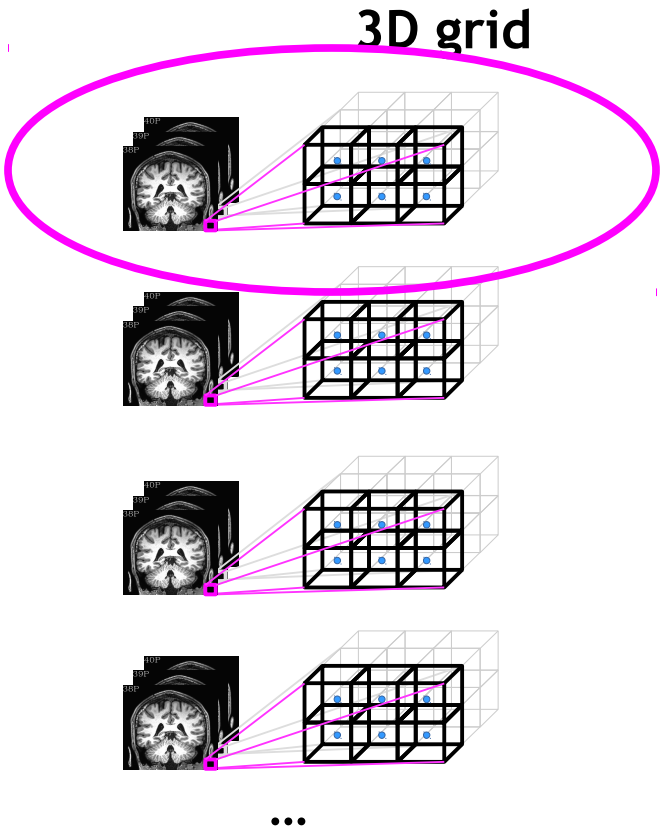
  A time series data set is an ordered set of 3D vols (→ a '4D data set').



We often refer to a 3D volume as a **brick**, because, well, it is an example of a solid, similar-looking 3D shape.

AFNI terminology sidenote

- What is a volumetric data set?
  → *It a **grid** made up of voxels (basic case: 3D).*
  Each voxel contains a number, which is represented by a color (gray, RGB, etc.).
  A time series data set is an ordered set of 3D vols (→ a '4D data set').



We often refer to a 3D volume as a **brick**, because, well, it is an example of a solid, similar-looking 3D shape.

Particularly in the context of 4D data sets, we also call a 3D volume a **sub-brick**.

This is an odd lingual quirk. But to date, this appears to be the only quirk in the AFNI software (or its developers).

AFNI terminology sidenote

- Sub-brick selection by volume index

  AFNI has a convenience feature of being able to select subset(s) of volumes for copying, calculation, etc. from a 4D set.

  This works by putting the index or index range in square brackets ***and*** quotation marks "[ ]" ("" keep the terminal from interpreting the square brackets specially).

AFNI terminology sidenote

- Sub-brick selection by volume index

  AFNI has a convenience feature of being able to select subset(s) of volumes for copying, calculation, etc. from a 4D set.

  This works by putting the index or index range in square brackets ***and*** quotation marks "[ ]" ("" keep the terminal from interpreting the square brackets specially).

  A comma separates indices, and two dots .. specifies an (inclusive) range; $ means final volume. *Ex.*:

```
DSET"[0]"           # initial subbrick (NB: count from 0!)
DSET"[0..5]"        # subbricks 0,1,2,3,4,5
DSET"[3,5..8,19]"   # subbricks 3,5,6,7,8,19
DSET"[1,14,29..$]"  # subbricks 1,14,29-to-the-last
DSET[0,4,5,15]      # ERROR in tcsh (no quotes); OK in bash
```

  Ex. application, to copy out subset:

```
3dcalc -a DSET "[3,5..8,19]" -expr 'a' -prefix DSET_NEW
```

  Fun fact: there are other forms of subbrick selection (brik label, voxel value...).

Grids

- What are the grid's properties?

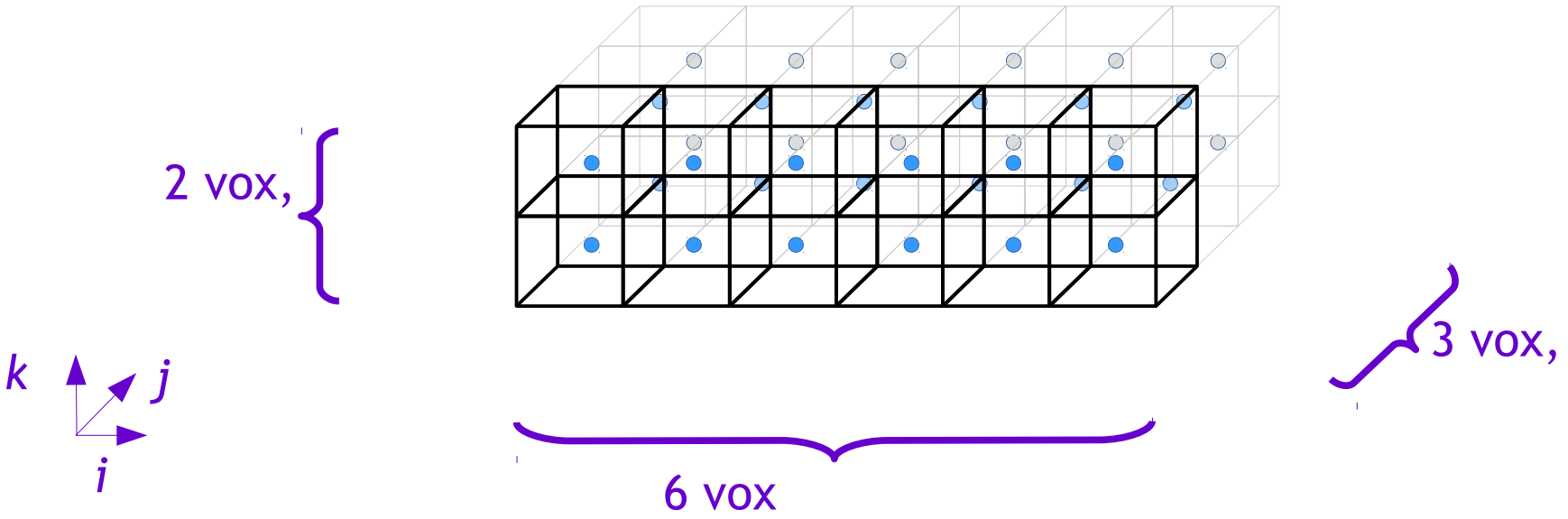
Grids

- What are the grid's properties?

  → *One is "size." Three ways to describe it:*

1) *matrix size*: count voxels in each dimension (n_i rows, n_j cols, n_k slices)

   - independent of voxel size

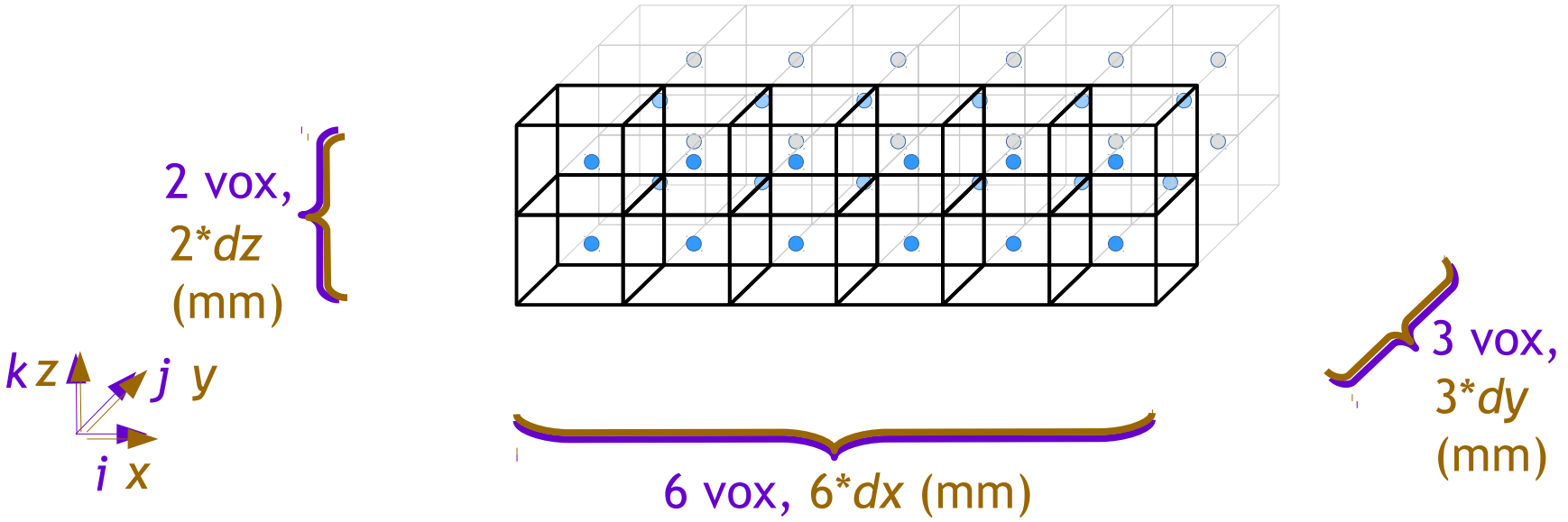


Grids

- What are the grid's properties?

  → *One is "size." Three ways to describe it:*

1) *matrix size*: count voxels in each dimension (n_i rows, n_j cols, n_k slices)
   - independent of voxel size
2) *field of view (FOV)*: units of mm, 3D phys vol of all voxels
   - depends on voxel size *(dx, dy, dz)*

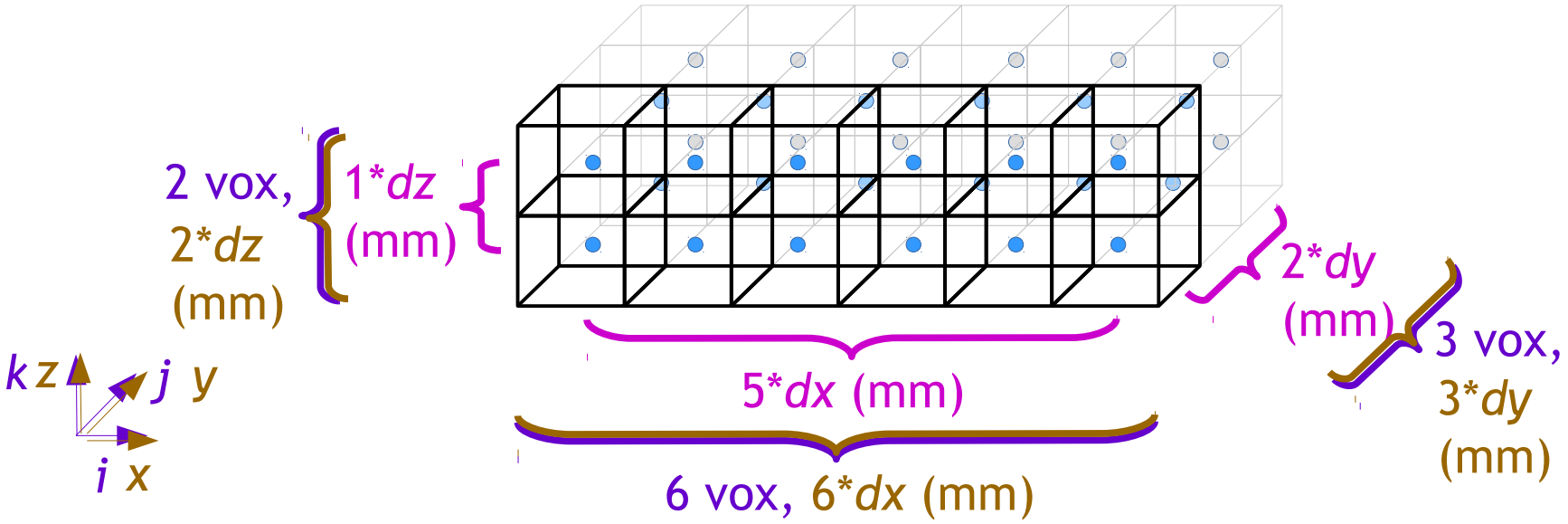


Grids

- What are the grid's properties?

  → *One is "size." Three ways to describe it:*

1) *matrix size*: count voxels in each dimension (n_i rows, n_j cols, n_k slices)
   - independent of voxel size
2) *field of view (FOV)*: units of mm, 3D phys vol of all voxels
   - depends on voxel size *(dx, dy, dz)*
3) *slab*: *units of mm, phys dist between first & last centroids*
   - e.g., dist between [0]th and [n_i-1]th centroid

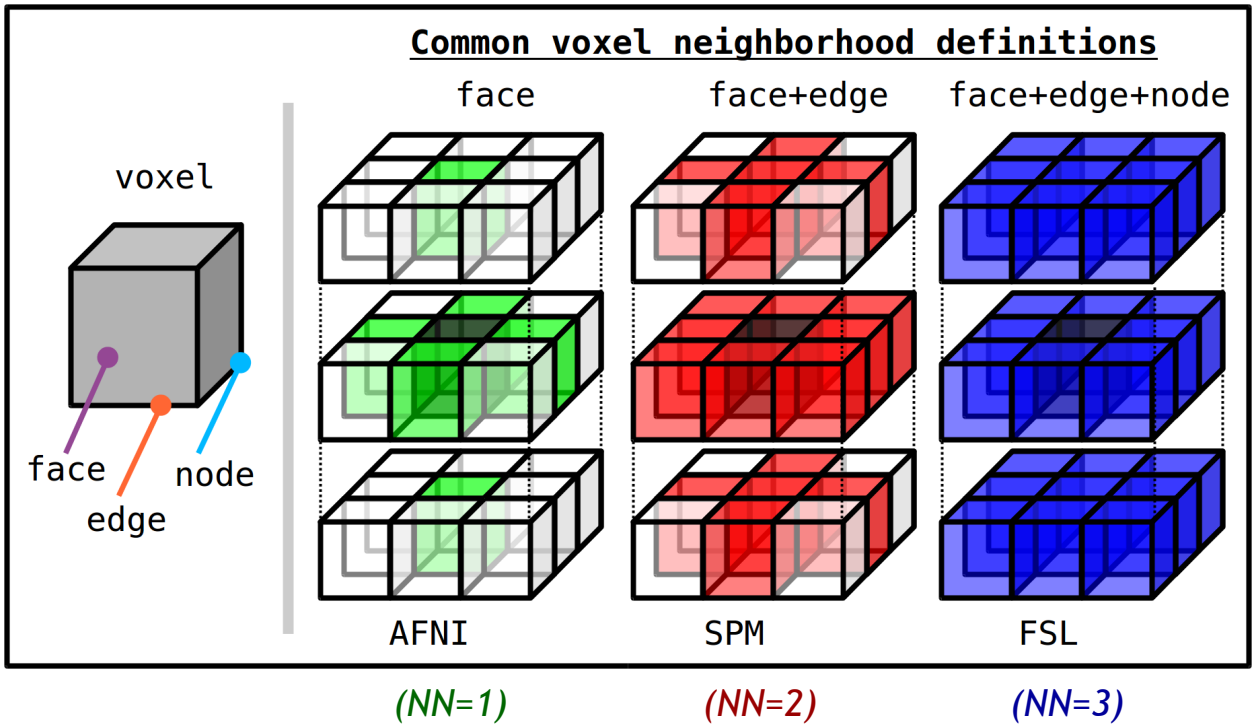


Sidenote: Getting to know your neighbors

- Who are a voxel's neighbors?

Sidenote: Getting to know your neighbors

- Who are a voxel's neighbors?
  → *This is used for blurring, clustering and several other steps.*
  → *Different softwares define this differently by default (o-o-o-of course…).*



NB: One can choose any of these three definitions in AFNI, typically with the "NN" specification.

Dataset storage: origin and orientation

- How is a 3D vol stored on the computer?

Dataset storage: origin and orientation

- How is a 3D vol stored on the computer?

  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*

  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*

  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0)*,* ***probably not (0, 0, 0)!***

Dataset storage: origin and orientation

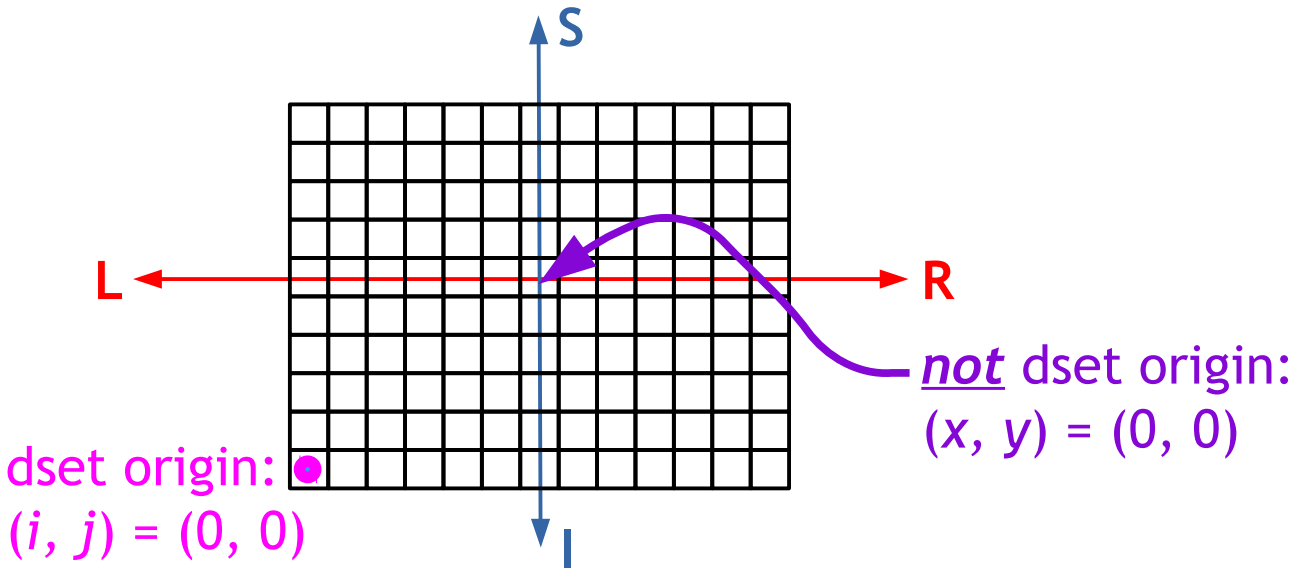
- How is a 3D vol stored on the computer?

  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*

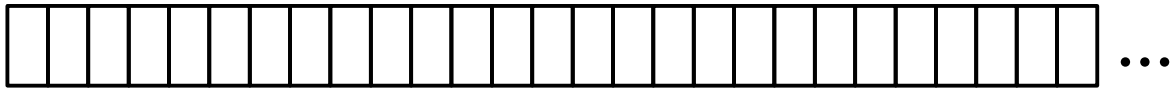
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*

  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Example,
2D grid



→ stored on comp:



Dataset storage: origin and orientation

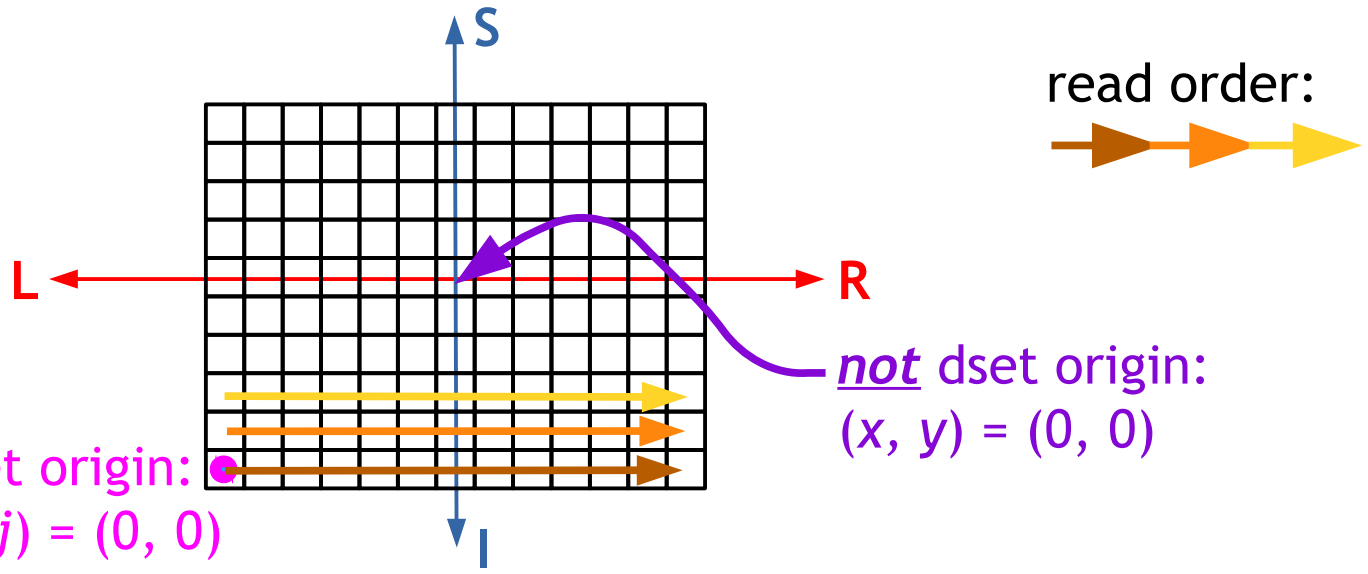
- How is a 3D vol stored on the computer?

  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*

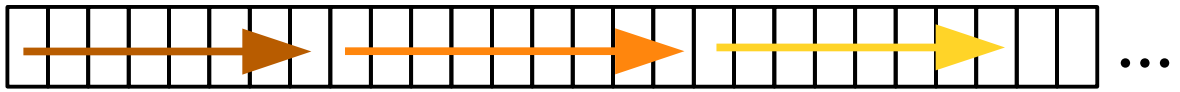
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*

  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Example,
2D grid



→ stored on comp:



Dataset storage: origin and orientation

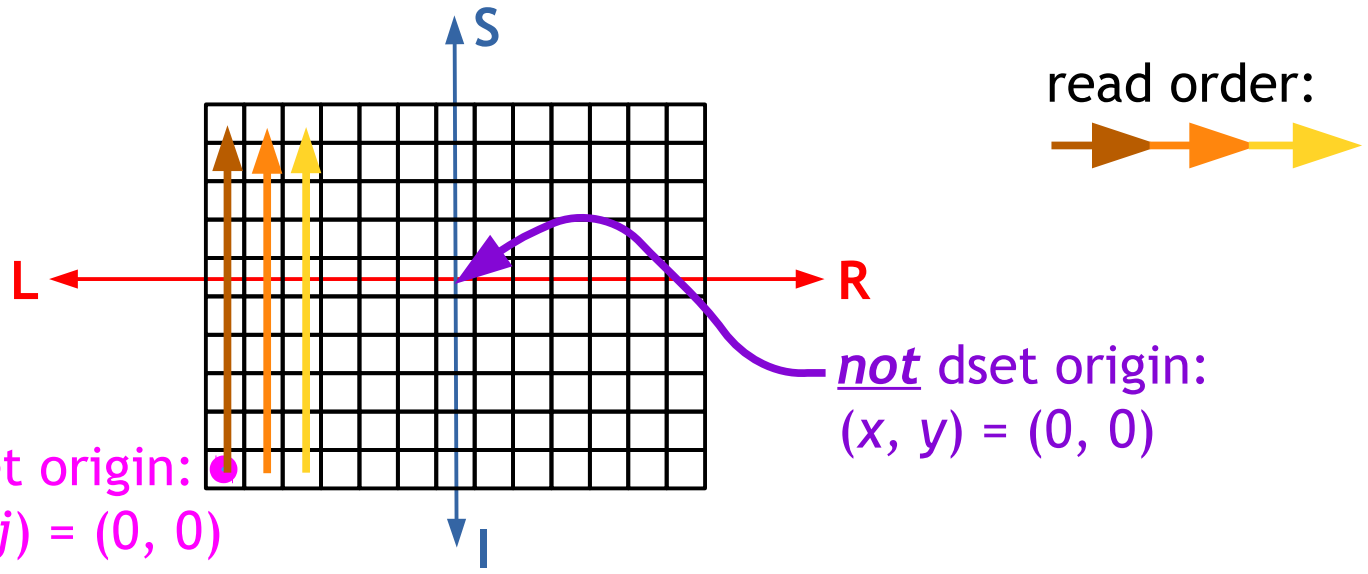
- How is a 3D vol stored on the computer?

  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*

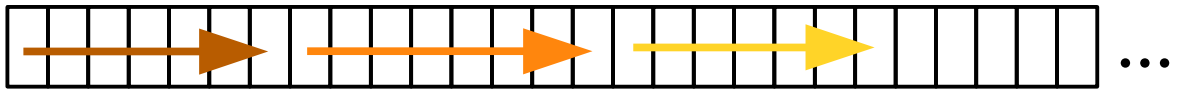
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*

  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Example,
2D grid



→ stored on comp:



Dataset storage: origin and orientation

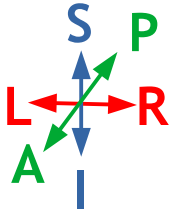
- How is a 3D vol stored on the computer?
  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*
  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

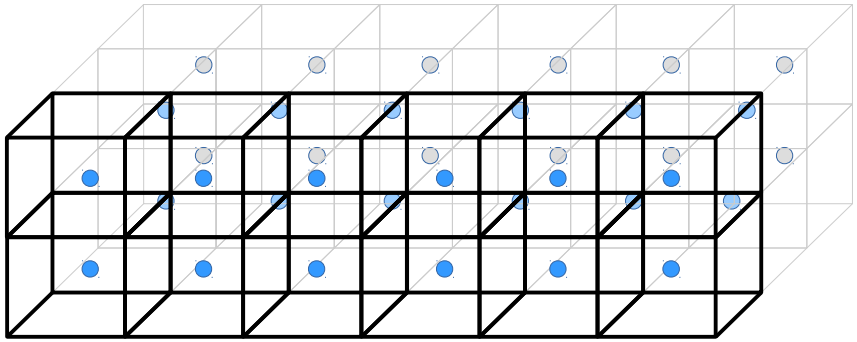
Ex: orientation = LAI

Q1: So where is the origin here?

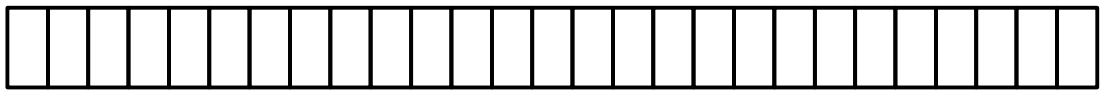
read order:

3D vol grid:





→ stored on comp:



...

See this in a vol with: *3dinfo -orient -origin DSET*

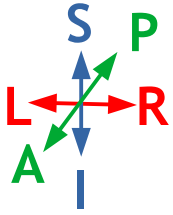
Dataset storage: origin and orientation

- How is a 3D vol stored on the computer?
  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*
  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

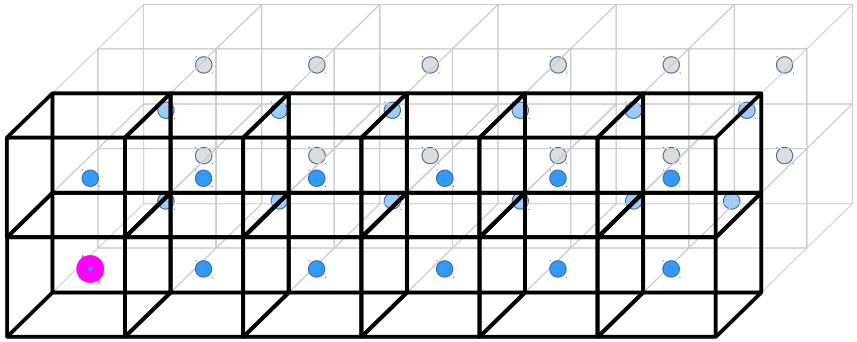
Ex: orientation = LAI

read order:

3D vol grid:



origin



Q2: So how is the data read into storage?

→ stored on comp:

...

See this in a vol with: *3dinfo -orient -origin DSET*

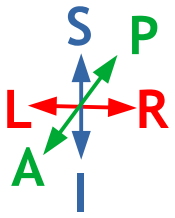
Dataset storage: origin and orientation

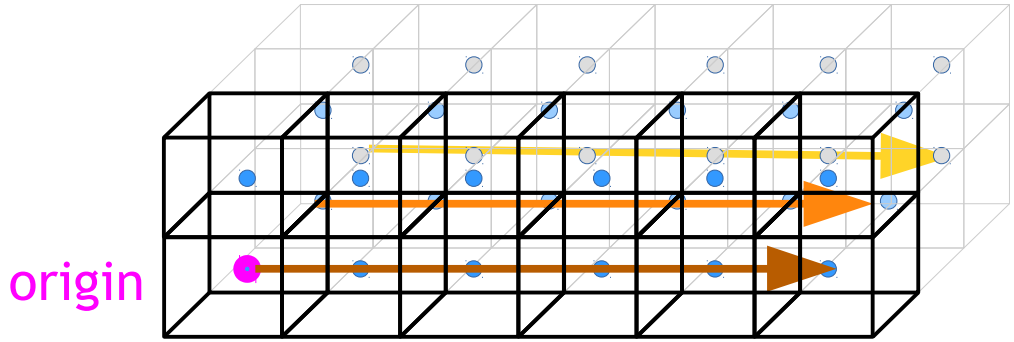
- How is a 3D vol stored on the computer?
  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*
  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Ex: orientation = LAI

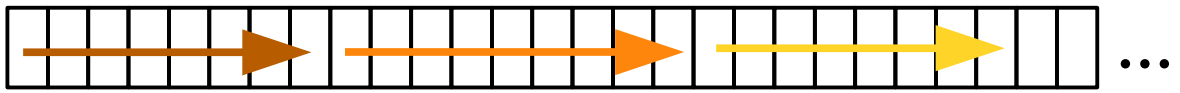
read order:

3D vol grid:





→ stored on comp:



See this in a vol with: *3dinfo -orient -origin DSET*

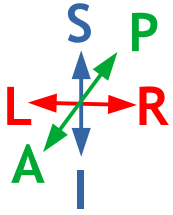
Dataset storage: origin and orientation

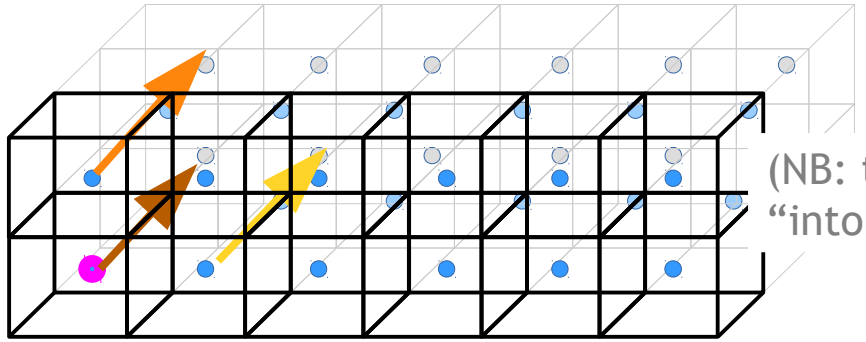
- How is a 3D vol stored on the computer?
  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*
  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Ex: orientation = AIL

read order:

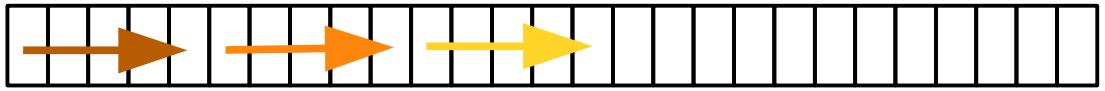
3D vol grid:





(NB: the arrows point "into" slide here...)

→ stored on comp:



...

See this in a vol with: *3dinfo -orient -origin DSET*

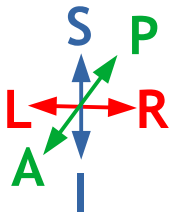
Dataset storage: origin and orientation

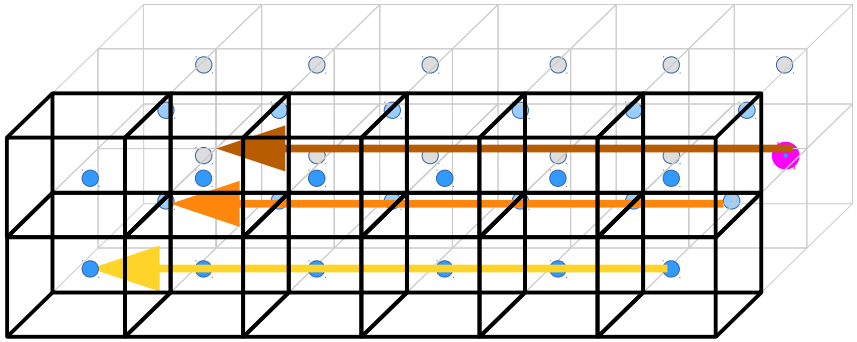
- How is a 3D vol stored on the computer?
  → *Row by row (as a flattened matrix), starting from one corner called the* ***origin****.*
  Orientation *states* which *corner, and* in which order *the rows are read (e.g., RPI).*
  At the origin: (i, j, k) = (0, 0, 0); and (x, y, z) = (x_0, y_0, z_0), ***probably not (0, 0, 0)!***

Ex: orientation = RPI

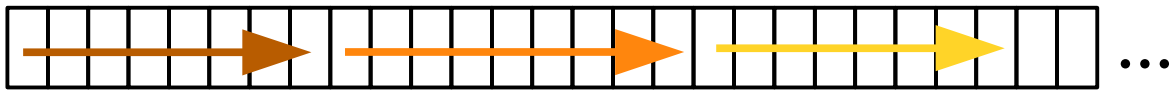
read order:

3D vol grid:





→ stored on comp:



See this in a vol with: *3dinfo -orient -origin DSET*

Volumetric data sets: files

- Where is information stored in files?

  → *A file contains two categories of information:*

  - data block: the numbers stored at each voxel
  - header: organizational information about the dset, like origin, orient, dimensions, voxel size, TR, labeltables, etc.

Volumetric data sets: files

- Where is information stored in files?

  → *A file contains two categories of information:*

  - data block: the numbers stored at each voxel
  - header: organizational information about the dset, like origin, orient, dimensions, voxel size, TR, labeltables, etc.

  → *There are multiple volumetric file formats. In AFNI, we mostly use two:*

  - BRIK-HEAD: pair of files, e.g., DSET+orig.HEAD and DSET+orig.BRIK
    - *.BRIK file contains data block (only)
    - *.HEAD file contains header info (only)
  - NIFTI: single file, e.g., DSET.nii, or (compressed) DSET.nii.gz
    - both header and data block in the same file

AFNI terminology sidenote

- SPACE and VIEW properties in a header

  Longtime AFNI users will notice that there is extra info in BRIK/HEAD file names:

  - DSET+**orig**.HEAD and DSET+**orig**.BRIK, or

    DSET+**tlrc**.HEAD and DSET+**tlrc**.BRIK, etc.

- This is the ("AFNI") "view" and describes if the dset is in original/native/acquired coordinates, or has been aligned to a template space (or AC-PC aligned).

AFNI terminology sidenote

- SPACE and VIEW properties in a header

  Longtime AFNI users will notice that there is extra info in BRIK/HEAD file names:

  - DSET+**orig**.HEAD and DSET+**orig**.BRIK, or

    DSET+**tlrc**.HEAD and DSET+**tlrc**.BRIK, etc.

- This is the ("AFNI") "view" and describes if the dset is in original/native/acquired coordinates, or has been aligned to a template space (or AC-PC aligned).
- The "space" property carries the specific name of *which* space (MNI, etc.) it is in.

```
3dinfo -space -av_space DSET   # 'av_space' = AFNI view space
```

AFNI terminology sidenote

- SPACE and VIEW properties in a header

  Longtime AFNI users will notice that there is extra info in BRIK/HEAD file names:
  - DSET+**orig**.HEAD and DSET+**orig**.BRIK, or

    DSET+**tlrc**.HEAD and DSET+**tlrc**.BRIK, etc.
- This is the ("AFNI") "view" and describes if the dset is in original/native/acquired coordinates, or has been aligned to a template space (or AC-PC aligned).
- The "space" property carries the specific name of *which* space (MNI, etc.) it is in.

```
3dinfo -space -av_space DSET   # 'av_space' = AFNI view space
```

- In the GUI, tlrc dsets have special features like "whereami" and atlas access.
- Dsets of different spaces cannot be overlayed in the GUI.

AFNI terminology sidenote

- SPACE and VIEW properties in a header

  Longtime AFNI users will notice that there is extra info in BRIK/HEAD file names:

  - DSET+**orig**.HEAD and DSET+**orig**.BRIK, or

    DSET+**tlrc**.HEAD and DSET+**tlrc**.BRIK, etc.

- This is the ("AFNI") "view" and describes if the dset is in original/native/acquired coordinates, or has been aligned to a template space (or AC-PC aligned).
- The "space" property carries the specific name of *which* space (MNI, etc.) it is in.

```
3dinfo -space -av_space DSET   # 'av_space' = AFNI view space
```

- In the GUI, tlrc dsets have special features like "whereami" and atlas access.
- Dsets of different spaces cannot be overlayed in the GUI.
- *Ex.*:

VIEW	SPACE	
orig	*ORIG*	# some original space
tlrc	*TLRC*	# mapped to a template space, called TLRC
tlrc	*MNI*	# mapped to a template space, called MNI
tlrc	*HaskinsPeds*	# mapped to a template space, called HaskinsPeds

Note: "tlrc" view label is generic, while "TLRC" space name is specific to a template.

Fun fact: these properties also map onto NIFTI sform and qform codes directly.

Other data set formats

- AFNI can read/transform other data set formats
  + ANALYZE (.hdr/.img file pairs), such as from SPM, FSL; e.g., *3dcopy*
  + MINC-1 (.mnc), such as from mnitools [but not MINC-2]; e.g., *3dMINCtoAFNI*
  + CTF (.mri, .svl), from MEG analysis volumes
  + BrainVoyager (.vmr), from BrainVoyager; e.g., *3dBRAIN_VOYAGERtoAFNI*
  + ASCII text (.1D): just numbers arranged into columns; e.g., *3dUndump*
- Note that these other formats may be missing some standard header information, which may need to be borrowed/used from other known files in NII or BRIK/HEAD format (e.g., 3dUndump to get grid)

- AFNI has some conversion programs to write out MINC-1, ANALYZE (3dAFNItoMINC, 3dAFNItoANALYZE, 3dmaskdump, etc.)

- For fuller related program list, see:

  https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/educational/classified_progs.html#copy-convert-manipulate-dsets

- ***Always check your results carefully when converting to other format/software!***

Creating dsets from DICOM files

- Data are often aquired as DICOM files
- AFNI has several programs for creating BRIK-HEAD and NIFTI files from DICOMs
- One has to be careful with DICOMs- not really standardized (booo!), fields/structure can change across scanner vendor, across version numbers, across acquisition sequences, and on the 3rd Tuesday after a blue moon.
- Some AFNI programs:
  + dcm2niix_afni: Chris Rorden's popular program, distributed in AFNI (thanks, Chris!)
    - very general use, can create whole collection of dsets
    - NB: NIFTI does *not* store complicated slice timings, so even if dcm2niix_afni can find it, it can't be stored
    - AFNI's 3drefit can be used to add slice timing info to the AFNI header extension
  + Dimon: R Reynold's creation, originally for sending "realtime FMRI" direct to AFNI
  + fat_proc_convert_dcm_{anat,dwis}: wrappers of dcm2niix_afni for DWI proc
  + and: https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/educational/classified_progs.html#dicom-info-and-conversion

- ***Always* check your results carefully (left-right flips!) when converting from DICOM!***

AFNI program note on dset and grid properties

To simply find out what the dset's grid, orientation, origin, etc. properties are, ***3dinfo*** is the way to go. *Ex.:*

```
3dinfo -orient -o3 DSET
```

AFNI program note on dset and grid properties

To simply find out what the dset's grid, orientation, origin, etc. properties are, ***3dinfo*** is the way to go. *Ex.:*

```
3dinfo -orient -o3 DSET
```

To alter dset/grid properties:

In AFNI, the program ***3dresample*** is useful for starting with one input and making a dset with a new grid, orientation, origin, etc. The program assumes that the starting information (both header **and** brick info) are *correct*. *Ex.:*

```
3dresample -orient RAI -prefix DSET_NEW -inset DSET
```

To change grid, orientation, origin, etc. properties when the header information is *incorrect*, then the program ***3drefit*** is useful. *Ex.:*

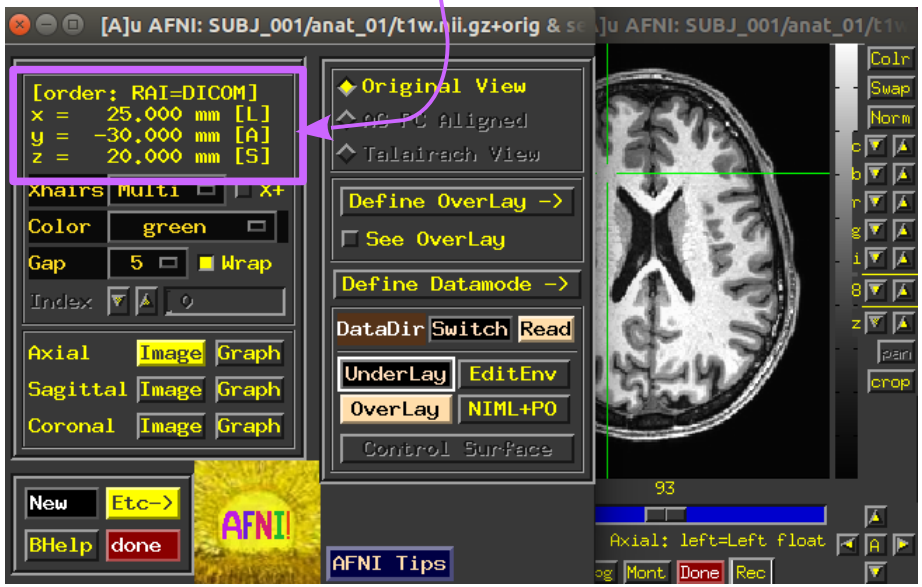
```
3drefit -orient RAI -inset DSET
```

Note the different purposes of ***3dresample*** and ***3drefit***.

Dataset orientation

Note: there are (at least) two uses of dset "orientation"

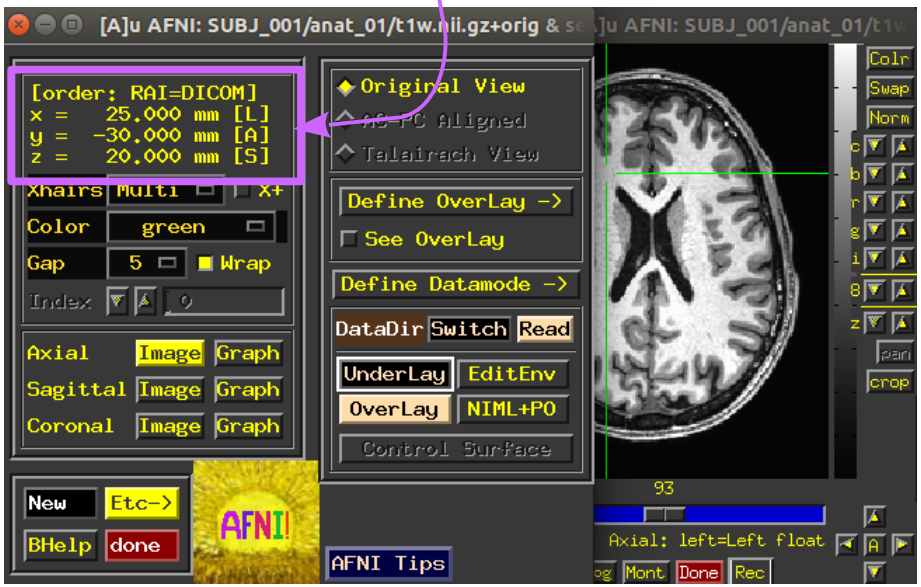
1) **Storage order**: how data is **stored** on disk (*3dinfo -orient DSET*).
2) **Reporting order:** describing the physical location coordinates.
   Ex: "We found a cluster with peak at (25, -30, 20)."
   - common cases: RAI (DICOM, and AFNI default), LPI (SPM)
   - in AFNI GUI:



Dataset orientation

Note: there are (at least) two uses of dset "orientation"

1) **Storage order**: how data is **stored** on disk (*3dinfo -orient DSET*).
2) **Reporting order:** describing the physical location coordinates.
   Ex: "We found a cluster with peak at (25, -30, 20)."
   - common cases: RAI (DICOM, and AFNI default), LPI (SPM)
   - in AFNI GUI:



When reporting:

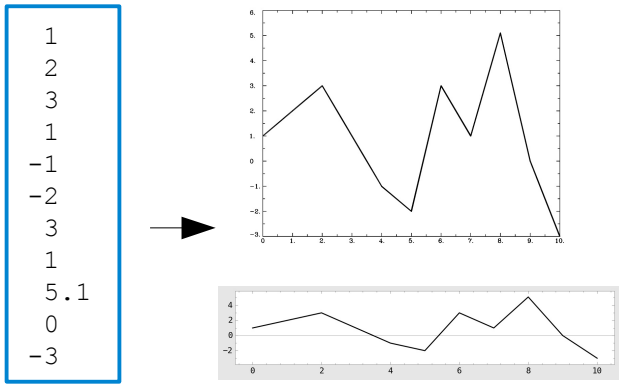
+ *Must* also report orient, because numbers change:
→ "... (25, -30, 20) in RAI-DICOM"
= "... (-25, 30, 20) in LPI coords"
+ **But better/best** is to report locations unambiguously:
→ "... (25L, 30A, 20S)", which applied to either RAI, LPS, or other!

Sidenote: non-volumetric files

*.1D files

1D files: text file, columns and/or rows of numbers
+ can represent time series, alignment/motion parameters, voxel locations, etc.
+ might include "# commented regions" at the top

Ways to view
+ open in text editor: "afni_open -t FILE.1D"
+ view in terminal: less, cat
+ plot: AFNI's 1dplot or 1dplot.py
  - each column is one time series
  - *Ex.:* `1dplot file.1D`




```
1dplot.py -infiles file.1D -prefix OUT.jpg
```

Useful programs for these types of dsets
+ 1dcat, 1dtranspose, 1d_tool.py, cat_matvec, 1deval, …
+ see: https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/educational/classified_progs.html#deal-with-1d-time-series

*.txt and/or *.dat files

TXT/DAT files: text file, could be numbers, could be words/strings
+ e.g., stimulus timing files with numbers and symbols
+ might include "# commented regions" at the top

Ways to view
+ open in text editor: "afni_open -t FILE.1D"
+ view in terminal: less, cat

*.json files

JSON files: text file, stores dictionaries and lists of information
+ general/standard file format, stands for **JavaScript Object Notation**
+ increasingly commonly used in neuroimaging to include extra/meta information about datasets

Ways to view
+ open in text editor: "afni_open -t FILE.1D"
+ view in terminal: less, cat
++ but to read/write/use: very common to use Python functionality

*.niml.dset and *.gii files

GII (GifTI): surface equivalent of NifTI files; standard format
NIML-DSET: surface data file format used in AFNI
→ for both, will deal more with these in the SUMA talks
(SUMA also has other intermediate/useful files *.niml*)

Useful programs for these types of dsets
+ See:

https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/educational/classified_progs.html#suma-surface-calculations-formats-and-viewing

*.niml.lt files

"labeltable" files made by and used in AFNI
+ files to associate a label (text string) with an ROI value (integer)
+ e.g., store names of ROIs in an atlas, such as FS parcellation
+ discussed more in the ROI talks

<u>If you can't wait to read more</u>
+ See *@MakeLabelTable*'s help:

  https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/programs/@MakeLabelTable_sphx.html#ahelp-makelabeltable

+ See ROI demo examples in AFNI doc tutorials:

  https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/tutorials/rois_corr_vis/main_toc.html