THE NIFTI-1 DATA FORMAT

Data Format Working Group (DFWG)

Contents

Chapter 1

Introduction

1.1 About NIfTI

This document describes a data format that originated from the Data Format Working Group (DFWG, chair: Prof. S.C. Strother) in the Neuroimaging Informatics Technology Initiative (NIfTI). The development of this data format has been supported by the National Institutes for Health (NIH).
The NIfTI-1 data format has been presented at the Human Brain Mapping Conference in 2004 [1]. Most of the information in the current document was made available by R.W. Cox in the `Annotated nifti.1 header`. The chapter "On q-form and s-form" was written by M. Jenkinson.
These chapters has been provided in 2004. Further developments on the format can be found on the NIfTI-1 website `http://nifti.nimh.nih.gov/` by the Data Format Working Group (DFWG).

1.2 Introduction to NIfTI-1

Robert W. Cox (NIH)
nifti1.h

The twin (and somewhat conflicting) goals of this modified ANALYZE 7.5 format are:

1. To add information to the header that will be useful for functional neuroimaging data analysis and display. These additions include:
   - More basic data types
   - Two affine transformations to specify voxel coordinates
   - "Intent" codes and parameters to describe the meaning of the data
   - Affine scaling of the stored data values to their "true" values
   - Optional storage of the header and image data in one file (*.nii)(see figure **??**).
2. To maintain compatibility with non-NIFTI-aware ANALYZE 7.5 compatible software (i.e., such a program should be able to do something useful with a NIFTI-1 dataset – at least, with one stored in a traditional .img/.hdr file pair).

Most of the unused fields in the ANALYZE 7.5 header have been taken, and some of the lesser-used fields have been co-opted for other purposes. Notably, most of the `data_history` substructure has been co-opted for other purposes, since the ANALYZE 7.5 format describes this substructure as "not required".

NIfTI-1 flag (magic strings)

To flag such a struct as being conformant to the NIFTI-1 spec, the last 4 bytes of the header must be either the C String "ni1" or "n+1"; in hexadecimal, the 4 bytes *6E 69 31 00* or *6E 2B 31 00* (in any future version of this format, the '1' will be upgraded to '2', etc.). Normally, such a "magic number" or flag goes at the start of the file, but trying to avoid clobbering widely-used ANALYZE 7.5 fields led to putting this marker last. However, recall that "the last shall be first" (Matthew 20:16).

If a NIFTI-aware program reads a header file that is NOT marked with a NIFTI magic string, then it should treat the header as an ANALYZE 7.5 structure.

NIfTI-1 file storage

"ni1" means that the image data is stored in the ".img" file corresponding to the header file (starting at file offset 0).

"n+1" means that the image data is stored in the same file as the header information. We recommend that the combined header+data filename suffix be ".nii". When the dataset is stored in one file, the first byte of image data is stored at byte location `(int)vox_offset` in this combined file.

Grace under fire

Most NIFTI-aware programs will only be able to handle a subset of the full range of datasets possible with this format. All NIFTI-aware programs should take care to check if an input dataset conforms to the program's needs and expectations (e.g., check datatype, `intent_code`, etc.). If the input dataset can't be handled by the program, the program should fail gracefully (e.g., print a useful warning; not crash).

Chapter 2

Interpretation of voxel data

Robert W. Cox (NIH)

2.1 Introduction

The `intent_code` field can be used to indicate that the voxel data has some particular meaning. In particular, a large number of codes is given to indicate that the the voxel data should be interpreted as being drawn from a given probability distribution.

2.2 Vector-valued datasets

The 5th dimension of the dataset, if present (i.e., $dim[0] = 5$ and $dim[5] > 1$), contains multiple values (e.g., a vector) to be stored at each spatiotemporal location. For example, the header values

- $dim[0] = 5$
- $dim[1] = 64$
- $dim[2] = 64$
- $dim[3] = 20$
- $dim[4] = 1$ (indicates no time axis)
- $dim[5] = 3$
- $datatype =$ `DT_FLOAT`
- $intent_code =$ `NIFTI_INTENT_VECTOR`

mean that this dataset should be interpreted as a 3D volume (64x64x20), with a 3-vector of floats defined at each point in the 3D grid.

A program reading a dataset with a 5th dimension may want to reformat the image data to store each voxels' set of values together in a struct or array. This programming detail, however, is beyond the scope of the NIFTI-1 file specification! Uses of dimensions 6 and 7 are also not specified here.

2.3 Statistical parametric datasets (i.e., SPMs)

Values of `intent_code` from `NIFTI_FIRST_STATCODE` to `NIFTI_LAST_STATCODE` (inclusive) indicate that the numbers in the dataset should be interpreted as being drawn from a given distribution. Most such distributions have auxiliary parameters (e.g., `NIFTI_INTENT_TTEST` has 1 DOF parameter).

If the dataset DOES NOT have a 5th dimension, then the auxiliary parameters are the same for each voxel, and are given in header fields `intent_p1`, `intent_p2`, and `intent_p3`.

If the dataset DOES have a 5th dimension, then the auxiliary parameters are different for each voxel. For example, the header values

- $dim[0] = 5$
- $dim[1] = 128$
- $dim[2] = 128$
- $dim[3] = 1$ (indicates a single slice)
- $dim[4] = 1$ (indicates no time axis)
- $dim[5] = 2$
- datatype = `DT_FLOAT`
- `intent_code` = `NIFTI_INTENT_TTEST`

mean that this is a 2D dataset (128x128) of t-statistics, with the t-statistic being in the first "plane" of data and the degrees-of-freedom parameter being in the second "plane" of data.

If the dataset 5th dimension is used to store the voxel-wise statistical parameters, then `dim[5]` must be 1 plus the number of parameters required by that distribution (e.g., `intent_code`=`NIFTI_INTENT_TTEST` implies dim[5] must be 2, as in the example just above).

Note: `intent_code` values 2..10 are compatible with AFNI 1.5x (which is why there is no code with value=1, which is obsolescent in AFNI).

2.4 Other intentions

The purpose of the `intent_*` fields is to help interpret the values stored in the dataset. Some non-statistical values for `intent_code` and conventions are provided for storing other complex data types.

The `intent_name` field provides space for a 15 character (plus 0 byte) 'name' string for the type of data stored. Examples:

- `intent_code = NIFTI_INTENT_ESTIMATE`; `intent_name` = "T1"; could be used to signify that the voxel values are estimates of the NMR parameter T1.

- `intent_code = NIFTI_INTENT_TTEST`; `intent_name` = "House"; could be used to signify that the voxel values are t-statistics for the significance of 'activation' response to a House stimulus.
- `intent_code = NIFTI_INTENT_DISPVECT`; `intent_name` = "ToMNI152"; could be used to signify that the voxel values are a displacement vector that transforms each voxel (x,y,z) location to the corresponding location in the MNI152 standard brain.
- `intent_code = NIFTI_INTENT_SYMMATRIX`; `intent_name` = "DTI"; could be used to signify that the voxel values comprise a diffusion tensor image.

If no data name is implied or needed, `intent_name[0]` should be set to 0.

default: no intention is indicated in the header.

Chapter 3

MRI-specific spatial and temporal information

Robert W. Cox (NIH)

3.1 Introduction

A few fields are provided to store some extra information that is sometimes important when storing the image data from an FMRI time series experiment. (After processing such data into statistical images, these fields are not likely to be useful.)

- `freq_dim` = These fields encode which spatial dimension (1,2, or 3)
- `phase_dim` = corresponds to which acquisition dimension for MRI data.
- `slice_dim` =

Examples:

Rectangular scan multi-slice EPI: `freq_dim = 1` `phase_dim = 2` `slice_dim = 3` (or some permutation)

Spiral scan multi-slice EPI: `freq_dim` = `phase_dim` = 0 `slice_dim` = 3

since the concepts of frequency- and phase-encoding directions don't apply to spiral scan.

`slice_duration` = If this is positive, AND if `slice_dim` is nonzero, indicates the amount of time used to acquire 1 slice. `slice_duration`*dim[`slice_dim`] can be less than pixdim[4] with a clustered acquisition method, for example.

`slice_code` = If this is nonzero, AND if `slice_dim` is nonzero, AND if `slice_duration` is positive, indicates the timing pattern of the slice acquisition. The following codes are defined:

- `NIFTI_SLICE_SEQ_INC`

- NIFTI_SLICE_SEQ_DEC
- NIFTI_SLICE_ALT_INC
- NIFTI_SLICE_ALT_DEC
- slice_start = indicates the start and end of the slice acquisition
- slice_end = pattern, when slice_code is nonzero.

These values are present to allow for the possible addition of"padded" slices at either end of the volume, which don't fit into the slice timing pattern. If there are no padding slices, then slice_start=0 and slice_end=dim[slice_dim]-1 are the correct values. For these values to be meaningful, slice_start must be non-negative and slice_end must be greater than slice_start.

The following table indicates the slice timing pattern, relative to time=0 for the first slice acquired, for some sample cases. Here, dim[slice_dim]=7 (there are 7 slices, labeled 0..6), slice_duration=0.1, and slice_start=1, slice_end=5 (1 padded slice on each end).

slice index	SEQ_INC	SEQ_DEC	ALT_INC	ALT_DEC
6	n/a	n/a	n/a	n/a
5	0.4	0.0	0.2	0.0
4	0.3	0.1	0.4	0.3
3	0.2	0.2	0.1	0.1
2	0.1	0.3	0.3	0.4
1	0.0	0.4	0.0	0.2
0	n/a	n/a	n/a	n/a

The fields freq_dim, phase_dim, slice_dim are all squished into the single byte field dim_info (2 bits each, since the values for each field are limited to the range $0 \ldots 3$). This unpleasantness is due to lack of space in the 348 byte allowance.

The macros DIM_INFO_TO_FREQ_dDIM, DIM_INFO_TO_PHASE_DIM, and DIM_INFO_TO_SLICE_DIM can be used to extract these values from the dim_info byte.

The macro FPS_INTO_DIM_INFO can be used to put these 3 values into the dim_info byte.

Chapter 4

3D Image (volume) orientation and location in space

Robert W. Cox (NIH)

4.1 Introduction

There are 3 different methods by which continuous coordinates can attached to voxels. The discussion below emphasizes 3D volumes, and the continuous coordinates are referred to as (x,y,z). The voxel index coordinates (i.e., the array indexes) are referred to as (i,j,k), with valid ranges:

- i = 0 .. dim[1]-1
- j = 0 .. dim[2]-1 (if dim[0] ¿= 2)
- k = 0 .. dim[3]-1 (if dim[0] ¿= 3)

The (x,y,z) coordinates refer to the CENTER of a voxel. In methods 2 and 3, the (x,y,z) axes refer to a subject-based coordinate system, with +x = Right +y = Anterior +z = Superior. This is a right-handed coordinate system. However, the exact direction these axes point with respect to the subject depends on `qform_code` (Method 2) and `sform_code` (Method 3).

N.B.: The i index varies most rapidly, j index next, k index slowest. Thus, voxel (i,j,k) is stored starting at location $(i+j*dim[1]+k*dim[1]*dim[2])*(bitpix/8)$ into the dataset array.

N.B.: The ANALYZE 7.5 coordinate system is +x = Left +y = Anterior +z = Superior which is a left-handed coordinate system. This backwardness is too difficult to tolerate, so this NIFTI-1 standard specifies the coordinate order which is most common in functional neuroimaging.

N.B.: The 3 methods below all give the locations of the voxel centers in the (x,y,z) coordinate system. In many cases, programs will wish to display image

data on some other grid. In such a case, the program will need to convert its desired (x,y,z) values into (i,j,k) values in order to extract (or interpolate) the image data. This operation would be done with the inverse transformation to those described below.

N.B.: Method 2 uses a factor 'qfac' which is either -1 or 1; qfac is stored in the otherwise unused pixdim[0]. If pixdim[0]=0.0 (which should not occur), we take qfac=1. Of course, pixdim[0] is only used when reading a NIFTI-1 header, not when reading an ANALYZE 7.5 header.

N.B.: The units of (x,y,z) can be specified using the `xyzt_units` field.

4.2 Methods

4.2.1 Method 1

(the "old" way, used only when `qform_code` = 0):

The coordinate mapping from (i,j,k) to (x,y,z) is the ANALYZE 7.5 way. This is a simple scaling relationship:

- x = pixdim[1] * i
- y = pixdim[2] * j
- z = pixdim[3] * k

No particular spatial orientation is attached to these (x,y,z) coordinates. (NIFTI-1 does not have the ANALYZE 7.5 orient field, which is not general and is often not set properly.) This method is not recommended, and is present mainly for compatibility with ANALYZE 7.5 files.

4.2.2 Method 2

(used when `qform_code` > 0, which should be the "normal" case): The (x,y,z) coordinates are given by the pixdim[] scales, a rotation matrix, and a shift. This method is intended to represent "scanner-anatomical" coordinates, which are often embedded in the image header (e.g., DICOM fields (0020,0032), (0020,0037), (0028,0030), and (0018,0050)), and represent the nominal orientation and location of the data. This method can also be used to represent "aligned" coordinates, which would typically result from some post-acquisition alignment of the volume to a standard orientation (e.g., the same subject on another day, or a rigid rotation to true anatomical orientation from the tilted position of the subject in the scanner). The formula for (x,y,z) in terms of header parameters and (i,j,k) is:

x [R11 R12 R13] [pixdim[1] * i] [`qoffset_x`]

y = [R21 R22 R23] [pixdim[2] * j] + [`qoffset_y`]

z [R31 R32 R33] [qfac * pixdim[3] * k] [`qoffset_z`]

The `qoffset_*` shifts are in the NIFTI-1 header. Note that the center of the (i,j,k)=(0,0,0) voxel (first value in the dataset array) is just (x,y,z)=(`qoffset_x`, `qoffset_y`, `qoffset_z`).

The rotation matrix R is calculated from the `quatern_* parameters`. This calculation is described below.

The scaling factor qfac is either 1 or -1. The rotation matrix R defined by the quaternion parameters is "proper" (has determinant 1). This may not fit the needs of the data; for example, if the image grid is

- i increases from Left-to-Right
- j increases from Anterior-to-Posterior
- k increases from Inferior-to-Superior

Then (i,j,k) is a left-handed triple. In this example, if qfac=1, the R matrix would have to be

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

which is "improper" (determinant = -1).

If we set $qfac = -1$, then the R matrix would be

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

which is proper.

This R matrix is represented by quaternion $[a, b, c, d] = [0, 1, 0, 0]$ (which encodes a 180 degree rotation about the x-axis).

4.2.3 Method 3

(used when `sform_code` > 0):

The (x,y,z) coordinates are given by a general affine transformation of the (i,j,k) indexes:

- x = `srow_x[0] * i + srow_x[1] * j + srow_x[2] * k + srow_x[3]`
- y = `srow_y[0] * i + srow_y[1] * j + srow_y[2] * k + srow_y[3]`
- z = `srow_z[0] * i + srow_z[1] * j + srow_z[2] * k + srow_z[3]`

The `srow_*` vectors are in the NIFTI-1 header. Note that no use is made of pixdim[] in this method.

4.2.4 Why 3 methods?

Method 1 is provided only for backwards compatibility. The intention is that Method 2 (`qform_code > 0`) represents the nominal voxel locations as reported by the scanner, or as rotated to some fiducial orientation and location. Method 3, if present (`sform_code` ¿ 0), is to be used to give the location of the voxels in some standard space. The `sform_code` indicates which standard space is present. Both methods 2 and 3 can be present, and be useful in different contexts (method 2 for displaying the data on its original grid; method 3 for displaying it on a standard grid).

In this scheme, a dataset would originally be set up so that the Method 2 coordinates represent what the scanner reported. Later, a registration to some standard space can be computed and inserted in the header. Image display software can use either transform, depending on its purposes and needs.

In Method 2, the origin of coordinates would generally be whatever the scanner origin is; for example, in MRI, (0,0,0) is the center of the gradient coil.

In Method 3, the origin of coordinates would depend on the value of `sform_code`; for example, for the Talairach coordinate system, (0,0,0) corresponds to the Anterior Commissure.

4.3 Quaternion representation of rotation matrix (method 2)

The orientation of the (x,y,z) axes relative to the (i,j,k) axes in 3D space is specified using a unit quaternion [a,b,c,d], where $a*a+b*b+c*c+d*d=1$. The (b,c,d) values are all that is needed, since we require that $a = sqrt(1.0-b*b+c*c+d*d)$ be nonnegative. The (b,c,d) values are stored in the (`quatern_b`,`quatern_c`,`quatern_d`) fields.

The quaternion representation is chosen for its compactness in representing rotations. The (proper) 3x3 rotation matrix that corresponds to [a,b,c,d] is

$$R = \begin{pmatrix} a*a+b*b-c*c-d*d & 2*b*c-2*a*d & 2*b*d+2*a*c \\ 2*b*c+2*a*d & a*a+c*c-b*b-d*d & 2*c*d-2*a*b \\ 2*b*d-2*a*c & 2*c*d+2*a*b & a*a+d*d-c*c-b*b \end{pmatrix}$$

$$= \begin{pmatrix} R11 & R12 & R13 \\ R21 & R22 & R23 \\ R31 & R32 & R33 \end{pmatrix}$$

If (p,q,r) is a unit 3-vector, then rotation of angle h about that direction is represented by the quaternion

$$[a,b,c,d] = [cos(h/2), p*sin(h/2), q*sin(h/2), r*sin(h/2)]. \tag{4.1}$$

Requiring $a >= 0$ is equivalent to requiring $-Pi <= h <= Pi$. (Note that [-a,-b,-c,-d] represents the same rotation as [a,b,c,d]; there are 2 quaternions that can be used to represent a given rotation matrix R.) To rotate a 3-vector (x,y,z) using quaternions, we compute the quaternion product

$$[0,x',y',z'] = [a,b,c,d]*[0,x,y,z]*[a,-b,-c,-d] \tag{4.2}$$

which is equivalent to the matrix-vector multiply

$$\begin{pmatrix} x' & x \\ y' & R[y] \\ z' & z \end{pmatrix}$$

(equivalence depends on a*a+b*b+c*c+d*d=1).

Multiplication of 2 quaternions is defined by the following:

$$[a, b, c, d] = a * 1 + b * I + c * J + d * K \tag{4.3}$$

where

- $I * I = J * J = K * K = -1$
- $I * J = KJ * K = IK * I = J$
- $J * I = -KK * J = -II * K = -J$

I,J,K are square roots of -1; not commutative!

For example

$$[a, b, 0, 0] * [0, 0, 0, 1] = [0, -b, 0, a] \tag{4.4}$$

since this expands to

$$(a + b * I) * (K) = (a * K + b * I * K) = (a * K - b * J). \tag{4.5}$$

The above formula shows how to go from quaternion (b,c,d) to rotation matrix and direction cosines. Conversely, given R, we can compute the fields for the NIFTI-1 header by

- $a = 0.5 * sqrt(1 + R11 + R22 + R33)$ (not stored)
- $b = 0.25 * (R32 - R23)/a \Rightarrow$ `quatern_b`
- $c = 0.25 * (R13 - R31)/a \Rightarrow$ `quatern_c`d = 0.25 * (R21-R12) / a $\Rightarrow$ `quatern_d`

  If a=0 (a 180 degree rotation), alternative formulas are needed. See the `nifti1_io.c` function `mat44_to_quatern()` for an implementation of the various cases in converting R to [a,b,c,d].

  Note that R-transpose (= R-inverse) would lead to the quaternion [a,-b,-c,-d].

  The choice to specify the `qoffset_x` (etc.) values in the final coordinate system is partly to make it easy to convert DICOM images to this format. The DICOM attribute "Image Position (Patient)" (0020,0032) stores the (Xd,Yd,Zd) coordinates of the center of the first voxel. Here, (Xd,Yd,Zd) refer to DICOM coordinates, and $Xd = -x$, $Yd = -y$, $Zd = z$, where (x,y,z) refers to the NIFTI coordinate system discussed above. (i.e., DICOM +Xd is Left, +Yd is Posterior, +Zd is Superior, whereas +x is Right, +y is Anterior , +z is Superior.) Thus, if the (0020,0032) DICOM attribute is extracted into (px,py,pz), then `qoffset_x` $= -px$, `qoffset_y` $= -py$, `qoffset_z`$= pz$ is a reasonable setting when `qform_code=NIFTI_XFORM_SCANNER_ANAT`.

  That is, DICOM's coordinate system is 180 degrees rotated about the z-axis from the neuroscience/NIFTI coordinate system. To transform between DICOM and NIFTI, you just have to negate the x- and y-coordinates.

The DICOM attribute (0020,0037) "Image Orientation (Patient)" gives the orientation of the x- and y-axes of the image data in terms of 2 3-vectors. The first vector is a unit vector along the x-axis, and the second is along the y-axis. If the (0020,0037) attribute is extracted into the value (xa, xb, xc, ya, yb, yc), then the first two columns of the R matrix would be $\begin{pmatrix} -xa & -ya \\ -xb & -yb \\ xc & yc \end{pmatrix}$

The negations are because DICOM's x- and y-axes are reversed relative to NIFTI's. The third column of the R matrix gives the direction of displacement (relative to the subject) along the slice-wise direction. This orientation is not encoded in the DICOM standard in a simple way; DICOM is mostly concerned with 2D images. The third column of R will be either the cross-product of the first 2 columns or its negative. It is possible to infer the sign of the 3rd column by examining the coordinates in DICOM attribute (0020,0032) "Image Position (Patient)" for successive slices. However, this method occasionally fails for reasons that I (RW Cox) do not understand.

Chapter 5

On q-form and s-form

- Mark Jenkinson

This page outlines how to use the qform and sform in applications that modify the image - particularly reslicing and registration. When processing images that contain qforms and sforms, the following general rules are recommended:

qform is always related to the input image in the processing and is appropriately transformed when padding, cropping or applying some affine spatial transformation sform is transformed in the same way as the qform (when it is set) in processing operations that deal with single images

sform is copied from the reference or target image when an image is being registered/resampled to this reference space.

The reason for the last rule is that if you are using the intensity information in the reference image then you are trying to align these images in this case and want to have the registered image with the same coordinates as the reference (e.g. when registering to standard space, your final transformed image will want to be "in" standard space and so have the same voxel → mm lookup as the standard image).

The transformation that needs to be applied to a qform or sform matrix is most easily seen by considering the original operation (e.g. cropping) as a transformation of the original voxel coordinates. The new (or transformed) matrix (either qform or sform) is just the concatenation of the inverse voxel transform and the original matrix (qform or sform). This ensures that the same location in the image (which changes voxel coordinates) still has the same mm coordinate as specified by the qform or sform. For example, consider a sform matrix S0 (which is 4x4 for homogeneous coordinates). Also let the original voxel coordinate be V0 and the new voxel coordinate be V1, related by an affine transformation, A. That is, $V1 = A * V0$. The location V0 has a mm coordinate given by S0 * V0. The new sform matrix should give the same mm coordinate at the same location - that is, $S1 * V1 = S0 * V0$ which implies $S1 * A = S0$, or $S1 = S0 * inv(A)$, where inv(A) is the inverse of A. This is the new, transformed sform matrix. The same rule holds for qform matrices; that is, $Q1 = Q0 * inv(A)$, but only when A is a rigid-body transformation matrix.

Chapter 6

NIfTI-1 units

Robert W. Cox (NIH)

6.1 Units of spatial and temporal dimensions

The codes below can be used in `xyzt_units` to indicate the units of `pixdim`. As noted earlier, dimensions 1,2,3 are for x,y,z; dimension 4 is for time (t).

- If $dim[4] = 1$ or $dim[0] < 4$, there is no time axis.
- A single time series (no space) would be specified with
  - $dim[0] = 4$ (for scalar data) or $dim[0] = 5$ (for vector data)
  - $dim[1] = dim[2] = dim[3] = 1$
  - $dim[4]$ =number of time points
  - $pixdim[4]$ = time step
  - `xyzt_units` indicates units of $pixdim[4]$
  - $dim[5]$ = number of values stored at each time point

Bits 0..2 of `xyzt_units` specify the units of `pixdim[1..3]` (e.g., spatial units are values 1..7). Bits 3..5 of `xyzt_units` specify the units of `pixdim[4]` (e.g., temporal units are multiples of 8).

This compression of 2 distinct concepts into 1 byte is due to the limited space available in the 348 byte ANALYZE 7.5 header. The macros `XYZT_TO_SPACE` and `XYZT_TO_TIME` can be used to mask off the undesired bits from the `xyzt_units` fields, leaving "pure" space and time codes. Inversely, the macro `SPACE_TIME_TO_XYZT` can be used to assemble a space code (0,1,2,...,7) with a time code (0,8,16,32,...,56) into the combined value for `xyzt_units`.

Note that codes are provided to indicate the "time" axis units are actually frequency in Hertz (`_HZ`) or in part-per-million (`_PPM`).

The toffset field can be used to indicate a nonzero start point for the time axis. That is, time point m is at $t = toffset + m * pixdim[4] for m = 0..dim[4] - 1$.

Chapter 7

Unused fields

Robert W. Cox (NIH)

7.1 Introduction

Some of the ANALYZE 7.5 fields marked as ++UNUSED++ may need to be set to particular values for compatibility with other programs. The issue of interoperability of ANALYZE 7.5 files is a murky one – not all programs require exactly the same set of fields. (Unobscuring this murkiness is a principal motivation behind NIFTI-1.)

Some of the fields that may need to be set for other (non-NIFTI aware) software to be happy are:

extents dbh.h says this should be 16384

regular dbh.h says this should be the character 'r'

glmin dbh.h says these values should be the min and max voxel values for the entire dataset

glmax dbh.h says these values should be the min and max voxel values for the entire dataset

It is best to initialize ALL fields in the NIFTI-1 header to 0 (e.g., with calloc()), then fill in what is needed.

Chapter 8

Probability distributions

Robert W. Cox (NIH)

8.1 General info about codes for probability distributions

Most distributions have a number of parameters, below denoted by p1, p2, and p3, and stored in

- `intent_p1`, `intent_p2`, `intent_p3` if dataset doesn't have 5th dimension
- image data array if dataset does have 5th dimension

Functions to compute with many of the distributions below can be found in the CDF library from U Texas.

Formulas for and discussions of these distributions can be found in the following books:

[4] [2] [3]

Chapter 9

Extending the data in a NIfTI-1 header

Robert W Cox
SSCC/DIRP/NIMH/NIH/DHHS/USA/Earth
07 October 2004

9.1 Clarifying voxel offset

float `vox_offset`;

In a .nii file, the `vox_offset` field value is interpreted as the start location of the image data bytes in that file. In a .hdr/.img file pair, the `vox_offset` field value is the start location of the image data bytes in the .img file. If `vox_offset` is less than 352 in a .nii file, it is equivalent to 352 (i.e., image data never starts before byte `#352` in a .nii file). The default value for `vox_offset` in a .nii file is 352; in a .hdr file it is 0. `vox_offset` should be an integer multiple of 16; otherwise, some programs may not work properly (e.g., SPM). This is to allow memory-mapped input to be properly byte-aligned. Note that since `vox_offset` is an IEEE-754 32 bit float (for compatibility with the ANALYZE-7.5 format), it effectively has a 24 bit mantissa. All integers from 0 to 224 can be represented exactly in this format, but not all larger integers are exactly storable as IEEE-754 32 bit floats. However, unless you plan to have `vox_offset` be potentially larger than 16 MB, this should not be an issue. (Actually, any integral multiple of 16 up to 227 can be represented exactly in this format, which allows for up to 128 MB of random information before the image data.)

In a .img file (i.e., image data stored separately from the NIfTI-1 header), data bytes between `#0` and `#vox_offset-1` (inclusive) are completely undefined and unregulated by the NIfTI-1 standard. One potential use of having `vox_offset > 0` in the .hdr/.img file pair storage method is to make the .img file be a copy of (or link to) a pre-existing image file in some other format, such as DICOM; then `vox_offset` would be set to the offset of the image data in this file. (It may not be possible to follow the multiple-of-16 rule with an arbitrary

external file; using the NIfTI-1 format in such a case may lead to a file that is incompatible with software that relies on `vox_offset` being a multiple of 16.)

In a .nii file, data bytes between `#348` and `#vox_offset-1` (inclusive) may be used to store user-defined extra information; similarly, in a .hdr file, any data bytes after byte `#347` are available for user-defined extra information. The (very weak) regulation of this extra header data is described next.

9.2 Extended Header Data Section(s)

byte extension[4] ;

After the end of the 348 byte header (e.g., after the magic field), the next 4 bytes are an byte array field named "extension". By default, all 4 bytes of this array should be set to zero. In a .nii file, these 4 bytes will always be present, since the earliest start point for the image data is byte `#352`. In a separate .hdr file, these bytes may or may not be present. If not present (i.e., if the length of the .hdr file is 348 bytes), then a NIfTI-1 compliant program should use the default value of extension=0,0,0,0. The first byte (extension[0]) is the only value of this array that is specified at present. The other 3 bytes are reserved for future use.

If extension[0] is nonzero, it indicates that extended header information is present in the bytes following the extension array. In a .nii file, this extended header data is before the image data (and `vox_offset` must be set correctly to allow for this). In a .hdr file, this extended data follows extension and proceeds (potentially) to the end of the file.

The format of extended header data is weakly specified. Each extension must be an integer multiple of 16 bytes long. The first 8 bytes of each extension comprise 2 integers:

int esize , ecode ;

These values may need to be byte-swapped, as indicated by dim[0]for the rest of the header.

- esize is the number of bytes that form the extended header data
  - esize must be a positive integral multiple of 16
  - this length includes the 8 bytes of esize and ecode themselves

- ecode is a non-negative integer that indicates the format of the extended header data that follows
  - different ecode values are assigned to different developer groups
  - at present, the registered values for code are
    - 0 = unknown private format (not recommended!)
    - 2 = DICOM format (i.e., attribute tags and values)
    - 4 = AFNI group (i.e., ASCII XML-ish elements)

In the interests of interoperability (a primary rationale for NIfTI), groups developing software that uses this extension mechanism are encouraged to document and publicize the format of their extensions. To this end, the NIfTI DFWG will assign even numbered codes upon request to groups submitting at least rudimentary documentation for the format of their extension; at present, the contact is `rwcox@nih.gov`. The assigned codes and documentation will be posted on the NIfTI website. All odd values of ecode (and 0) will remain unassigned; at least, until the even ones are used up, when we get to 2,147,483,646.

Note that the other contents of the extended header data section are totally unspecified by the NIfTI-1 standard. In particular, if binary data is stored in such a section, its byte order is not necessarily the same as that given by examining dim[0]; it is incumbent on the programs dealing with such data to determine the byte order of binary extended header data.

Multiple extended header sections are allowed, each starting with an esize,ecode value pair. The first esize value, as described above, is at bytes `#352-355` in the .hdr or .nii file (files start at byte `#0`). If this value is positive, then the second (esize2) will be found starting at byte `#352+esize1`, the third (esize3) at byte `#352+esize1+ esize2` , et cetera. Of course, in a .nii file, the value of `vox_offset` must be compatible with these extensions. If a malformed file indicates that an extended header data section would run past `vox_offset`, then the entire extended header section should be ignored. In a .hdr file, if an extended header data section would run past the end-of-file, that extended header data should also be ignored.

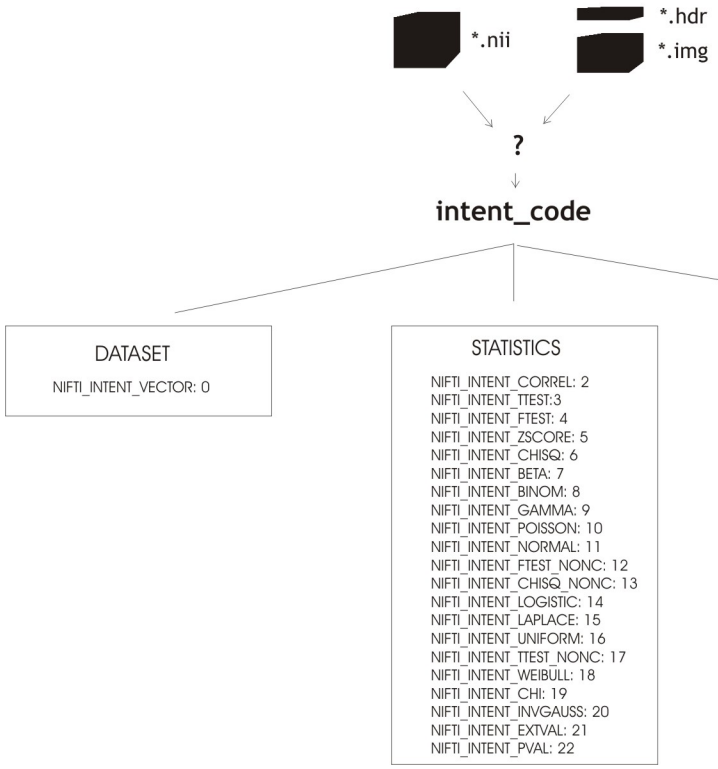
With the above scheme, a program can successively examine the esize and ecode values, and skip over each extended header section if the program doesnt know how to interpret the data within. Of course, any program can simply ignore all extended header sections simply by jumping straight to the image data using `vox_offset`.

Appendix A

NIfTI-1 intent diagrams

Hester Breman

THE NIFTI1 DATA FORMAT

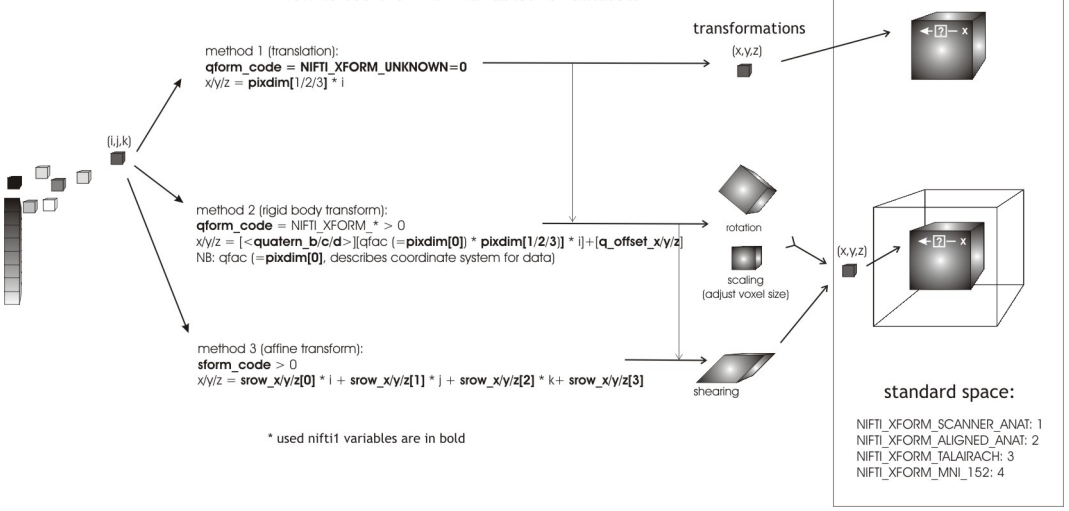


NIFTI1 can store data with different meanings. Imaging data, statistical values and other data (any vector, matrix, label set or mesh). can be saved in a nifti1 *.nii or *.hdr/*.img file. Once a data intent is chosen, the use of the nifti1 format is unambigious since the use of particular fields for a certain intent is predetermined.

To determine the position of the voxel in the dataset, method 1 is used (translation). Methods 2 and 3 serve also for reconstructing rigid body and affine transformations so that the positions of the voxels withinthe dataset in a stereotactic space can be determined.

NIFTI_INTENT: DATASETS

Locating position of voxel in dataset: how to use the nifti1 variables for datasets



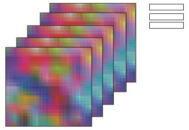
* used nifti1 variables are in bold

Nifti1 can also be used to store values drawn from a given distribution. For this purpose, many intent_types are dedicated to describe statistical tests Univariate and multivariate tests can be stored, including the used parameters. In nifti1, it is possible to save more than one values per voxel (even a matrix per voxel).

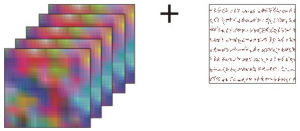
NIFTI_INTENT: STATISTICS

How to use the nifti1 variables for statistical values

dim[3] = 1: single slice
dim[5] = 1: statistical parameters stored in ***intent_p1/2/3*** (parameters applied to whole dataset)



dim[3] > 1: several slices
dim[5] = 1: statistical parameters stored in ***intent_p1/2/3*** (parameters applied to whole dataset)



dim[5] > 1: voxel-wise statistical parameters stored in data planes after stat value plane, for example the degrees of freedom

For more information about the nifti-1 data format, see http://nifti.nimh.nih.gov/dfwg/.
Latest update of this documentation: 2004.

Bibliography

[1] R.W. Cox et al. A (sort of) new image data format standard: Nifti-1. *Human Brain Mapping*, 25(x):xxx, 2004.

[2] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 1. xx, xxxx.

[3] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. xx, xxxx.

[4] N.L. Johnson, S. Kotz, and A.W. Kemp. *Univariate Discrete Distributions*. xx, xxxx.