Guide to Variance and Data Quality Extensions and to Pixel Data Units

Emma Hogan

Data Processing Software Group

V1.0 – 22 June 2012

Revision History

V1.0 – 22 June 2009 Emma Hogan

Document ID: DPSG-STD-104_VarianceDQPixelUnits

Document Purpose

The purpose of the this document is to clearly define variance and data quality, and how they are calculated and propagated in the Gemini data reduction software, and to present the rules applies to the pixel data and how to keep track of the units of the values stored in those pixels.

Intended Audience

This document is primarily intended for the developers of the Gemini data reduction software suite. The authors of the user's manual and the programmer's manual are expected to use elements of this document to explain how the software handles variance, data quality, and units.

Table of Contents

1. Introduction

In this document we explain how the variance, data quality flags, and pixel data units are being defined and handled in the Gemini data reduction software.

2. Variance

2.1 Introduction to Variance

For each pixel in a science extension, a statistical, uncorrelated (random) error can be estimated and stored in a corresponding pixel in a variance extension. During each data processing step, the variance extension is processed in parallel with the science extension to reflect how the noise changes during the processing. Other (systematic) errors are not included in the variance extension.

STANDARD DEVIATION: The statistical, uncorrelated (random) errors for the pixels in a science extension can be described by the standard deviation, which is a measure of the spread of a distribution. The standard deviation has the same units as the pixels in the science extension and in the case of a Gaussian distribution, the standard deviation is the "sigma". The initial standard deviation for raw pixel data can be estimated using two main components; an estimated read noise component and a Poisson noise component.

READ NOISE: The read noise is a consequence of the conversion of electrons in a pixel to Analogue-to-Digital Units (ADU) and is a fixed property of the instrument. Detector electronics usually do not report one ADU for each electron that is read. The number of ADU corresponding to a single electron is known as the detector gain. This is set to a value low enough for the read noise to dominate digitisation errors, but high enough to avoid reaching the maximum value of the Analogue-to-Digital Converter (ADC) at too small a fraction of the detector full well capacity. The estimated read noise affects every pixel independently of the number of electrons in that pixel and can be described using the standard deviation of a Gaussian distribution that has a mean of zero:

$$N(mean, stddev) = N(0, readnoise)$$

where the read noise is measured in electrons.

POISSON NOISE: The Poisson noise is the standard deviation of a Poisson distribution that describes the uncertainty in the counting of electrons in a pixel. Poisson statistics apply to electrons, rather than ADU. The Poisson distribution has a distribution parameter that is equal to the mean (the expected number of electrons) and the variance. The standard deviation from the mean is the square root of the mean. In practice, the standard deviation is estimated using the square root of the *measured* number of electrons in that pixel, N_e, since that is what is known:

$$Po(stddev) = Po\left(\sqrt{N_e}\right)$$

CALCULATIONS: Since the read noise and Poisson noise components of the standard deviation are uncorrelated (random) noise (i.e., the noise in each pixel is different), they are added in quadrature. The central limit theorem is invoked, which allows the the Poisson distribution to be approximated to a Gaussian distribution (which is true when the expected number of electrons is large). The final product is a Gaussian distribution with the mean and the variance equal to the number of electrons in that pixel, N_e:

$$Po\left(\sqrt{N_e}\right) \rightarrow N(mean, stddev) = N\left(N_e, \sqrt{N_e}\right)$$

The standard deviations (of the Gaussian distributions) can then be added in quadrature:

$$N(0, readnoise) + N\left(N_e, \sqrt{N_e}\right) \rightarrow$$
$$N\left(N_e, \sqrt{readnoise^2 + \sqrt{N_e}^2}\right) \rightarrow$$
$$N\left(N_e, \sqrt{readnoise^2 + N_e}\right)$$

$$stddev[electrons] = \sqrt{(readnoise[electrons])^2 + N_e}$$

$$stddev[ADU] = \frac{stddev[electrons]}{gain} = \sqrt{\left(\frac{readnoise[electrons]}{gain}\right)^2 + \frac{N_{ADU}}{gain}}$$

where the gain is measured in ADU per electron.

A variance extension is stored rather than a standard deviation extension simply because for most arithmetic operations on statistical distributions (i.e., they add in quadrature), the equations are simpler if the variance is used rather than the standard deviation (since the variance is defined as the standard deviation squared, repeated square and root operations are avoided).

$$variance[ADU] = \left(\frac{readnoise[electrons]}{gain}\right)^2 + \frac{N_{ADU}}{gain}$$

$$variance[electrons] = (readnoise[electrons])^2 + N_{ADU}$$

2.2 Calculating Initial Variance

The initial variance should be calculated and added to a dataset as near to the start of the processing as possible. The following list contains specific details related to variance coding:

1) The pixel data in the variance extension should always have the same size as the pixel data in the science extension.
2) The read noise component of the variance can be calculated and added to the variance extension at any time, but should be done before performing operations with other datasets.
3) The Poisson noise component of the variance can be calculated and added to the variance extension only after any bias levels have been subtracted from the pixel data in the science extension (due to the large DC bias component on the CCDs).
4) The variance of a raw bias frame contains only a read noise component (which represents the uncertainty in the bias level of each pixel), since the Poisson noise component of a bias frame is meaningless.
5) When the variance is calculated, the units of the pixel data in the science extension should be checked so that the variance is calculated in the same units. The variance can be calculated either before or after the pixel data in the science extension is converted to electrons. For more information, see 'Introduction to Pixel Data Units' section below.

2.3 Propagating Variance

At each reduction step, the variance extension is manipulated according to the operation being performed on the science extension. When applying calibrations, the statistical errors from the science frame and the statistical errors from the calibration are added in quadrature. Since the

science frame and the calibration have a variance extension that describe the statistical, uncorrelated errors, the variance extensions can simply be added together. The general equation for this is:

$$var\big(f(a,b)\big) = \left(var(a) * \left(\frac{\partial f}{\partial a}\right)^2\right) + \left(var(b) * \left(\frac{\partial f}{\partial b}\right)^2\right) + covariance\ term$$

where $\partial f/\partial a$ and $\partial f/\partial b$ are partial derivatives. More specific equations are:

$$var(\ a + b\ ) = var(a) + var(b) + covariance\ term$$
$$var(\ a - b\ ) = var(a) + var(b) - covariance\ term$$

$$var(\ a * b\ ) = (var(a) * b^2) + (var(b) * a^2) + covariance\ term$$
$$var(\ a \div b\ ) = \left(\frac{var(a)}{b^2}\right) + \left(\frac{var(b) * a^2}{b^4}\right) + covariance\ term$$

Since the variance extensions contain only uncorrelated noise, the covariance terms above are zero.

When the variable b above is a single value that does not have its own variance, the equations can still be used (the variance for a is scaled accordingly).

When average combining frames that have variance extensions, the equations can still be used (the variance of the average combined frame is reduced and is equal to the sum of the variance of the individual frames divided by N^2, where N is the number of frames contributing to the combined frame). If the variance extensions just contain the read noise, then for the case of N average combined frames, the read noise will decrease by a factor of sqrt(N) (assuming that the read noise in all frames are the same):

$$var(a) = readnoise^2 = var(b) = var(c)\ ...$$

$$var(\bar{N}) = \frac{N * readnoise^2}{N^2} = \frac{readnoise^2}{N}$$

$$stddev(\bar{N}) = \sqrt{var(\bar{N})} = \frac{readnoise}{\sqrt{N}}$$

When median combining frames that have variance extensions, the equations cannot be used. The median of a Poisson distribution is ~ [mean + 1/3 - 0.02/mean] (http://en.wikipedia.org/wiki/Poisson_distribution). Therefore, the variance of the median combined frame is TBD.

The following list contains specific details related to variance propagation coding:

1) The uncertainty in a measurement that is equal for all pixels in a dataset is defined as correlated noise and should not be included when propagating the variance.
2) If either the calibration frame or the science frame does not have a variance extension, no variance is propagated.

At the end of the complete data reduction process, the variance propagation allows the final noise values to be estimated just by taking the square root of the final variance extension.

3. Data Quality

3.1 Introduction to Data Quality

A data quality extension provides a way to flag and track information about the quality of the pixels that are present in the pixel data of the associated science extension. The adopted encoding for the pixel data in the data quality extension are as follows:

0 Good pixel
1 Detector defect causing a bad pixel (hot or dead)
2 Non-linear regime (may not be used for all instruments)
4 Saturated
8 Cosmic-ray-hit. This is typically identified as part of a co-adding procedure and will mostly be used by tasks like imcoadd.
16 No data. Pixels that contain no data include those that are in the gaps between the arrays in GMOS and GSAOI. This bit value will also be used by, e.g., imcoadd to flag areas of the final image for which no data are available. This will be the case if the dithering pattern does not give full coverage of the output image.
32 Contaminated by overlap. Pixels that are contaminated by overlap are those that contain data from other MOS or IFU slits.
64 Unilluminated. Pixels that are unilluminated include those that are off the edge of the imaging field in data from instruments such as GMOS and FLAMINGOS-2, and between the slits along the spatial axis.

[The first six values and descriptions were taken from http://internal.gemini.edu/science/dataProc/Docs/gemini_DQ.txt. The last two values and descriptions were added in June 2012 by the DPSG group].

Additional numeric bit values will be allocated in bit-order as they are required. Therefore, for a 16-bit data quality extension, a further 9 numeric bit values are available. If more bit values are required, the data quality extensions can be seamlessly expanded to 32-bit.

In the Gemini Python package, names are associated to each numeric bit value so that the values can be referred to by name rather than the numeric bit value directly. These names are defined in a single file and allows the numeric bit values to be changed conveniently, if desired (and also enables the possibility to accommodate multiple conventions):

0 DQ_good
1 DQ_bad_pixel
2 DQ_non_linear
4 DQ_saturated
8 DQ_cosmic_ray
16 DQ_no_data
32 DQ_overlap
64 DQ_unilluminated

The data quality encodings will be grouped depending on the severity of the reason the pixel was flagged. These data quality group names will also be defined in the single file mentioned previously and will have numeric values equal to the bitwise OR of the bit values in that group:

DQ_pass = DQ_good == 0
DQ_usable = (DQ_non_linear | DQ_overlap | DQ_unilluminated)
DQ_fail = (DQ_bad_pixel | DQ_saturated | DQ_cosmic_ray | DQ_no_data)

The data quality encodings that belong in the DQ_fail group refer to pixels in the science

extensions that are almost certainly completely useless and should never be used during processing, while those that belong in the DQ_usable group refer to pixels in the science extensions that can be used for most purposes, but carry warnings. For example, unilluminated pixels can give an indication of the background level, but it is unlikely that those pixels would be used during processing.

3.2 Calculating Initial Data Quality

The initial data quality should be calculated and added to the dataset as near to the start of the processing as possible. The following list contains specific details related to data quality coding:

1) The data quality extension should be a 16-bit fits extension.
2) The pixel data in the data quality extension should always have the same size as the pixel data in the science extension.
3) Bad pixels are flagged in the data quality extension using a bad pixel mask (BPM). Generally, BPMs do not include any overscan regions.
4) Non-linear and saturated pixels are flagged in the data quality extension using the non_linear_level and saturated_level descriptors. These pixels should be flagged before any bias levels are subtracted (since the hard saturation level for GMOS is the ADC cut off, not full well). Non-linear and saturated pixels are flagged in any overscan regions.
5) A description of the numeric bit values, the names and the data quality groups used as described above should be written to the COMMENT headers of the data quality extensions. This ensures that the values used in a given data quality extension are fully defined and associated with that data quality extension, providing a user all the information they need to interpret the data quality extension.

3.3 Propagating Data Quality

When processing frames that contain data quality extensions (e.g., when applying a calibration frame that contains a data quality extension or when combining science frames that contain data quality extensions), the pixel data in the data quality extensions should be propagated by combining the pixels with a bitwise OR, except in the special cases as described below. This allows a pixel in the data quality extension to contain information about multiple problems with the associated pixels in any frame that contributed to that pixel. For example:

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
16	10000

0001 OR 0010 = 0011
0001 OR 0001 = 0001

The following list contains specific details related to data quality propagation coding:

1) If either the calibration frame or the science frame does not have a data quality extension, no data quality information is propagated.
2) The data quality extension should be used whenever possible to mask bad pixels when

processing the science extension.
    - When performing certain processing steps on the science extension, data with specific numeric bit values may be used.
    - The option to include data with certain numeric bit values should be made available to the user (in principle, it should be possible to choose for each operation what is considered good and what is considered bad).
3) When combining data, pixels in the DQ_fail group should never be propagated.
    - If all input pixels for a given output pixel belong in the DQ_fail group, the output pixel in the output data quality extension should be flagged as DQ_no_data.
    - If all input pixels for a given output pixel are flagged in their respective data quality extensions, but some of the pixels are flagged DQ_usable, the DQ_usable pixels should be combined to produce the output pixel in the output science extension and the corresponding output pixel in the output data quality extension should be flagged with the appropriate bit value (following the philosophy that it is better to have usable data than no data).
    - If one input pixel for a given output pixel is flagged DQ_good and more than one input pixel is flagged DQ_usable, the single good pixel will be used as the output pixel in the output science extension (pixels should only be combined with other pixels that have a data quality flag no worse than theirs) [should there be an option to use the multiple DQ_usable pixels instead of the single good pixel?]

4. Pixel Data Units

4.1 Introduction to Pixel Data Units

Pixel data can have units of either ADU or electrons. The BUNIT keyword is used to store this unit information. The following list contains specific details related to pixel data unit coding:

1) The BUNIT keyword in the science extension should have a value of either "adu" or "electron", the BUNIT keyword in the variance extension should have a value of either "adu*adu" or "electron*electron" and there should be no BUNIT keyword in the DQ extensions (since a bit mask does not have units).
    - STScI use "electrons", so the recipe system should recognise both "electron" and "electrons" as "electron".
2) The BUNIT keyword is added to the science extension by standardizeHeaders (it is assumed that raw pixel data are in ADU) and is updated when the pixel data is converted to electrons.
    - The GAIN keyword should *not* be updated or changed when the pixel data is converted to electrons (changing the GAIN to be equal to 1 after converting the pixel data to electrons just so that the variance equation in ADU still works is not the "right thing to do").
3) Pixel data should not be converted to electrons until after any bias levels have been subtracted (nod-and-shuffle TBD).
4) The user level functions check the units of the pixel data in the science extension (via the BUNIT keyword) and whether any processing has (or hasn't) been done that would invalidate the variance calculation.
    - The user level functions should print a warning if the Poisson noise component is added to the variance extension of a bias frame.
    - The user level functions should print a warning if the Poisson noise component is added to the variance extension of a science frame that still contains a bias level.

4.2 Calibrations

1) Calibrations should only be retrieved if the pixel data in the calibration has the same units as the pixel data in the science frame.
    - The user level functions should print a warning if a user tries to apply a calibration that has different units to the pixel data in the science frame.
2) Processed bias frames are stored in units of ADU.
    - The processing of bias frames should be done in ADU.
3) Processed dark frames are stored in units of electrons.
4) Processed, normalized flat frames are unitless. Therefore BUNIT should be set to "" in the science and variance extensions in the dataset during the normalize step.

5. Detailed Revision History

v1.0 22 June, 2012 Emma Hogan

Initial revision. Ported to Word for configuration management by Kathleen Labrie on 10 April 2013.