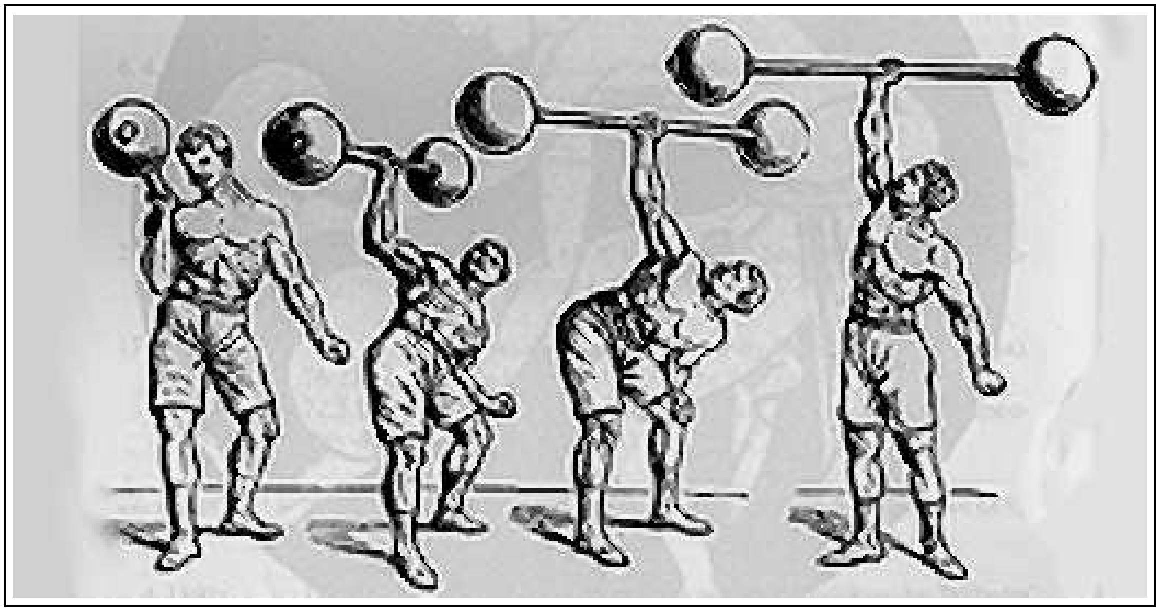
# WeightWatcher
## $v1.2$
## User's guide

E. BERTIN
Institut d'Astrophysique de Paris *



*present address: European Southern Observatory, Karl-Schwarzschild-Str. 2, D-85748 Garching bei München, Germany

## Contents

# 1 What is WeightWatcher?

WeightWatcher (or simply WW) is a program that combines weight-maps, flag-maps, and vector data in order to produce control maps which can directly be used by astronomical image-processing packages like Drizzle[1] or SExtractor2[2]. Weight-thresholding and/or specific flag selection are applied by WW through a configuration file: this alleviates other programs from such interpretation work. WeightWatcher will mostly be useful as part of an imaging survey pipeline. Its main features are:

- Processing speed: limited by the I/O performances of the machine (typically $\approx 1$ Mpixel/s on a workstation),
- Ability to work with very large images (up to, say, $10^8 \times 10^9$ pixels on a 64 bits workstation),
- FITS format is used for input and output. Output flag-map format selection is automatic (8, 16 or 32bits),
- Up to 30 weight-maps, 30 flag-maps, and thousands of polygons can be handled simultaneously.

# 2 Overview of the software

The layout of WeightWatcher is displayed in Fig. 1. One can distinguish 5 main operations which are controlled by sets of configuration parameters: `WEIGHT_MIN` and `WEIGHT_MAX` for threshold selection; `FLAG_MASKS` and `WEIGHT_MASKS` for bit masking; `WEIGHT_OUTFLAGS`, `FLAG_OUTFLAGS` and `POLY_OUTFLAGS` for flag assignments; all value assignments are defaulted to 0.0 in the present version.

The assignment of output flags is left to the user, who should take care of possibly overlapping flag-bits. Both the "Assign-values" and "Assign-flags" operations in Fig. 1 *act as the logical "OR"*. For instance, one "bad" pixel in one of the input weight-maps is enough for having the output weight-map set to zero at the same pixel location. Likewise, one flag surviving the "Bit-masking" operation is enough for triggering some bit in the output flag-map.

# 3 Using WeightWatcher

WeightWatcher is run from the shell with the following syntax:

% `ww -c` *configuration-file* `[ -`*Parameter1 Value1* `]` `[ -`*Parameter2 Value2* `]` ...

The part enclosed within brackets is optional. Any "`-`*Parameter Value*" statement in the command-line overrides the corresponding definition in the configuration-file or any default value (see below).

## 3.1 The Configuration file

Each time WeightWatcher is run, it looks for a configuration file. If no filename is given in the command line, a warning message is printed.

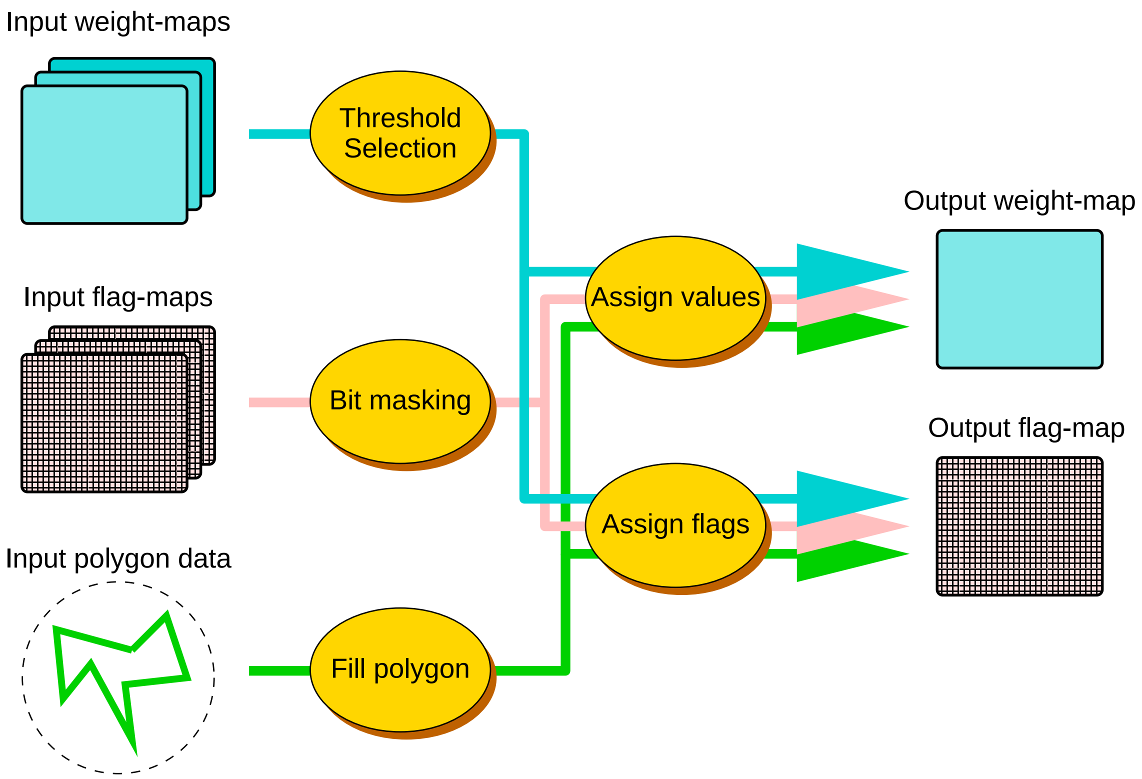



Figure 1: Global organization of WeightWatcher.

### 3.1.1 Format of the configuration file

The format is ASCII. There must be only one parameter set per line, following the form:

*Config-parameter* *Value(s)*

Extra spaces or linefeeds are ignored. Comments must begin with a "#" and end with a linefeed. Values can be of different types: strings (can be enclosed between double quotes), floats, integers, keywords or boolean (`Y/y` or `N/n`). Some parameters accept zero or several values, which must then be separated by commas. Integers can be given as decimals, in octal form (preceded by digit `0`), or in hexadecimal (preceded by `0x`). The hexadecimal format is particularly convenient for writing multiplexed bit values such as binary masks. Environment variables, written as `$HOME` or `${HOME}` are expanded, and not only for string parameters.

### 3.1.2 Parameter list

Here is a list of all the parameters known to WeightWatcher. Please refer to §3.2 for a detailed description of their meaning.

| Parameter | type / keyword | Description |
|---|---|---|
| WEIGHT_NAMES | *string(s)* | Filename(s) of the input weight-map(s). |
| WEIGHT_MIN | *float(s)* | Weight-pixels below these thresholds will be flagged. |
| WEIGHT_MAX | *float(s)* | Weight-pixels above these thresholds will be flagged. |
| WEIGHT_OUTFLAGS | *integer(s)* | Flag values given to thresholded pixels. |
| FLAG_NAMES | *string(s)* | Filename(s) of the input flag-map(s). |
| FLAG_WMASKS | *integer(s)* | Flag-bits which will affect the weight-map pixels. |
| FLAG_MASKS | *integer(s)* | Bits which will be converted as output flags. |
| FLAG_OUTFLAGS | *integer(s)* | Translation of the FLAG_MASKS bits. |
| POLY_NAMES | *string(s)* | Filename(s) of the input polygon descriptions. |
| POLY_OUTFLAGS | *integer(s)* | Flag values given to pixel *inside* polygons. |
| OUTWEIGHT_NAME | *string* | Filename of the output weight-map. |
| OUTFLAG_NAME | *string* | Filename of the output flag-map. |
| MEMORY_BUFSIZE[1] | *integer* | Buffer size, in image lines. |
| VERBOSE_TYPE[1] | *keyword* | How much WeightWatcher comments its operations: |
| | QUIET | – run silently, |
| | NORMAL | – display warnings and limited info concerning the work in progress, |
| | FULL | – display more complete information. |

## 3.2 Detailed description of the configuration parameters

### 3.2.1 Processing of the input weight-map(s)

WEIGHT_NAMES - Filename(s) of the weight-maps provided as input to WW. Any 2-dimensional FITS image may be used as a weight-map; integer images are automatically converted to floating-point format. If several images are requested, they must all have the same width and the same height. A null string ("") can be provided, in which case no input weight-map will be used. *Note that only the 1st weight-map will have its pixel values copied to the output weight-map.*

WEIGHT_MIN - Lower threshold(s) applied to the input weight-map(s). There must be one value per WEIGHT_NAMES filename. Pixels below these values will trigger flagging in the output flag-map and will set to zero the corresponding output weight-map pixel.

WEIGHT_MAX - Same as WEIGHT_MIN but for upper threshold(s).

WEIGHT_OUTFLAGS - Flag(s) that will be OR'ed in the output flag-map in case of threshold overtaking. There must be one value per WEIGHT_NAMES filename.

### 3.2.2 Processing of the input flag-map(s)

FLAG_NAMES - Filename(s) of the flag-maps provided as input to WW. Any 2-dimensional FITS image with pixels coded as integers[2] (BITPIX = 8, 16 or 32) may be used as a weight-map. If

[1]Optional parameter

[2]According to the FITS convention, integers should be *signed* integers. The behaviour of the sign bit in bitwise operations is notoriously ambiguous. Therefore it is strongly advised to always use positive flag values and to avoid using the sign bit for storing flag information.

several images are requested, they must all have the same width and the same height. A null string ("") can be provided, in which case no input flag-map will be used.

`FLAG_WMASKS` - Mask(s) that will be applied to the input flag-map(s). If the result is non-zero, the corresponding pixel in the output weight-map will be set to zero. There must be one mask per `FLAG_NAMES` filename.

`FLAG_MASKS` - Mask(s) that will be applied to the input flag-map(s). For each bit which passes the masking, a flag (specified by the `OUTPUT_FLAGS` parameter below) will be OR'ed with the corresponding pixel in the output flag-map. There must be one mask per `FLAG_NAMES` filename.

`FLAG_OUTFLAGS` - Flag value(s) that will be OR'ed in the output flag-map for input flag-map pixels which pass the `FLAG_MASKS` masking. There must be one flag value per `FLAG_MASKS` bit set.

### 3.2.3 Processing of the polygon vector files

`POLY_NAMES` - Filename(s) of the files containing descriptions of polygon shapes, provided as input to WW. Contrarily to all other files handled by WW, polygon data are in `ASCII` format. The adopted syntax is very simple and fully compatible with that of "region" data produced by SAOimage. They are traditionally given the "`.reg`" filename extension. The rules are as follows:

- The first line contains a "`#`" as a first character, followed by a space character and some identificator (for instance the FITS image filename to which the polygon description refers). Example: "`# toto.fits`".
- Lines containing the polygon description must be of the form "`-POLYGON(`$x_1$`,`$y_1$`,`$x_2$`,`$y_2$`,...)`"[3], where pairs of coordinates $x_n$,$y_n$ (in decimal pixel units) refer to consecutive vertices of the polygon. WW assumes that the polygon is a closed shape, and therefore it is useless to repeat the first point at the end of the list. Line segments defined by these points must not intersect each other, and a polygon must have at least 3 vertices. One `ASCII` line only shall be used per polygon. Currently, overlapping polygon or folded polygon regions are combined according to an "Exclusive-OR" rule; therefore two concentric, circle-like polygons will lead to an annulus.
- `ASCII` lines with a different content will be ignored.

When polygon informations have been loaded from one or several files (each file possibly containing a very large number of polygons), WW will use them to define enclosed areas which will be flagged in the output flag-map, and/or set to zero in the output weight-map (for exclusion). A null string ("") can be provided for `POLY_NAMES`, in which case no polygons will be drawn.

`POLY_OUTFLAGS` - Flag(s) that will be OR'ed in the output flag-map for pixels "inside" polygons. There must be one value per `POLY_NAMES` filename.

[3]The "-" preceding `POLYGON` is not mandatory: *included* and *excluded* SAOImage "regions" are processed the same way.

#### 3.2.4 Output files

OUTWEIGHT_NAME - Filename for the weight-map produced by WW. A null string ("") can be provided if no weight-map output is desired.

OUTFLAG_NAME - Filename for the flag-map produced by WW. A null string ("") can be provided if no flag-map output is desired.

#### 3.2.5 Optional parameters

MEMORY_BUFSIZE - Number of image lines that should be processed at once in memory. Numbers as small as 1 allow huge images to be processed on modest systems. The default value is 32, and can be increased to accelerate the processing of small images (less than 1024 pixels wide).

### 3.3 Example of configuration

Let's suppose that we want to produce a weight-map `weight.fits` which takes into account the gain-map of the detector if the gain is better than 0.2 (from the normalized flat-field `ff.fits`), saturated pixels (pixel values above 65000 in the raw science image `sc.fits`), cosmetic defects (flagged by bit 4 in the flag-map `cd.fits`), cosmic rays (identified by pixel values above 12.0 in the filtered image `cr.fits`) and finally some "bad" regions described as polygonal shapes in `bad.reg`. In the same time, we want to produce a flag-map `flag.fits` in which saturated pixel are flagged by bit 0, cosmetic defects by bit 1, gain drops by bit 2, cosmic rays by bit 3, and bad regions by bit 4. Then the configuration file will be:

```
# Default configuration file for WeightWatcher V1.2
# EB 06/11/97

#---------------------------------- Weights -----------------------------------

WEIGHT_NAMES  ff.fits, sc.fits, cr.fits # filenames of the input WEIGHT maps

WEIGHT_MIN      0.2,  -1e30, -1e30      # pixel values < -1e9 are unlikely!
WEIGHT_MAX      1e30, 65000,  12.0      # pixel values > 1e9 are unlikely!
WEIGHT_OUTFLAGS 0x04,  0x01,  0x08      # hexadecimal coding of bits 2, 0 and 3

#----------------------------------- Flags ------------------------------------

FLAG_NAMES      cd.fits                 # filename of the input FLAG map

FLAG_WMASKS     0x10                    # hexadecimal coding of bit 4
FLAG_MASKS      0x10                    # hexadecimal coding of bit 4
FLAG_OUTFLAGS   0x02                    # hexadecimal coding of bit 1

#---------------------------------- Polygons ----------------------------------

POLY_NAMES      bad.reg                 # name of the file containing polygons

POLY_OUTFLAGS   0xa0                    # hexadecimal coding of bit 4
```

```
#---------------------------------- Output -----------------------------------

OUTWEIGHT_NAME  weight.fits            # output WEIGHT-map filename
OUTFLAG_NAME    flag.fits              # output FLAG-map filename
```

**Acknowledgments** Thanks to Erik Deul for proposing the WeightWatcher name and comments on this text.

## References

[1] Fruchter A.S., Hook R.N., http://www.stsci.edu/ fruchter/dither/dither.html

[2] Bertin E., SExtractor 2.0, User's manual, 1997, ESO