**RABDAM** Handbook

**RABDAM** (**Rad**iation **BDam**age calculator) is a program written in Python 2.7.10 which calculates ***B*Damage** values for the atoms in biomolecular structures, when atomic information is input in PDB format.

NOTE: An historic version of the program written in MATLAB is also available from <https://github.com/td93/B_Damage>

**Please cite:** M. Gerstel, C. M. Deane and E.F. Garman. (2015). J. Synchrotron Radiation. **22**, 201-212 <http://dx.doi.org/doi:10.1107/S1600577515002131>

Table of Contents

[Introduction 3](#_Toc440619548)

[Usage - Inputs 4](#_Toc440619549)

[Required Inputs: 4](#_Toc440619550)

[PDB code 4](#_Toc440619551)

[File Path 4](#_Toc440619552)

[Optional Inputs: 4](#_Toc440619553)

[PDT 4](#_Toc440619554)

[binSize 4](#_Toc440619555)

[createAUCpdb 4](#_Toc440619556)

[createTApdb 4](#_Toc440619557)

[Sample Input 5](#_Toc440619558)

[Location of Input 5](#_Toc440619559)

[Outputs 6](#_Toc440619560)

[<name>Bdamage.txt: 6](#_Toc440619561)

[<name>UnitCell.pdb: 6](#_Toc440619562)

[Description of Exceptions Output by RABDAM 7](#_Toc440619563)

[For Developers 8](#_Toc440619564)

# Introduction

For molecular structures determined using X-ray crystallography each atom in the structure is assigned (amongst a range of atom specific metrics) an ***isotropic atomic B-factor*** value. This value is correlated to the distribution of the atom across all unit cells throughout time (from the beginning to the end of the diffraction collection protocol) and space (throughout all unit cells which the X-ray beam intercepts). Atoms with the larger distributions will have higher *B*-factor values. Atomic *B-*factors can increase as a result of a number of different factors: increased dynamic disorder (increased thermal motion - correlated to temperature), increased static disorder (e.g. multiple conformations of residues or the atom being located on mobile loops and chain termini), incorrectly refining the correct number of electrons into a region of electron density (i.e. the incorrect element or **elemental charge**), incorrectly refining atomic occupancy. The degree of *crystallographic packing* will also affect an atom’s *B-*factor: if the atom is surrounded by a greater number of other atoms, it will be sterically restricted, and thus have a lower *B*-factor. ***B*Damage** is a metric that attempts to **deconvolute** the ***crystallographic packing density*** to give an "effective *B-*factor value” that is correlated to radiation damage for each atom (provided that the crystal structure has been successfully and well refined).

|  |  |  |
| --- | --- | --- |
| ***Factor*** | ***Correlated to:*** | |
| ***Radiation Damage?*** | ***B*Damage?** |
| Dynamic disorder | Yes[1] | No |
| Static disorder | Yes – e.g. reduction of disulphide bond | Yes |
| Element/elemental charge | Yes – e.g. reduction of metal ions | Yes |
| Atomic occupancy | No | No |
| Crystallographic Packing | No | No |

[1] During irradiation with the beam, there are local increases in temperature. This increases the dynamic disorder (thermal motion) of the atoms, but it does so **uniformly**, so will also be deconvoluted during the calculation of ***B*Damage.**

***B*Damage** for a given atom is the ratio of the atom's *B-*factor and the average *B-*factor of atoms within a *similar packing density*. There are many ways to measure packing density, but the metric used by **RABDAM** is the ‘**Atomic Contact Number**’ (**ACN**). This metric simply counts the number of atoms contained in a sphere of a given radius (which can be input into the program as the **Packing Density Threshold**, or **PDT**, in Å). All the atoms contained in the crystal structure are grouped into ‘**bins**’ where all atoms in that bin have a ‘similar’ ACN. Similarity is defined as width (which can be input into the program as the **bin size**, or **binSize**, in atoms) where the minimum value is an integer multiple of the bin size, and the maximum value is 1 fewer than the next integer multiple of the bin size.

# Usage - Inputs

## Required Inputs:

(**RABDAM** takes a minimum of one input, which directs the program to where the atomic information can be found.)

### PDB code

There are two options here, if the required input is a structure publicly available on the PDB (<http://www.rcsb.org>) then the four digit PDB accession code is sufficient to run the program. Provided the machine on which the program is running has access to an internet connection, the program will automatically retrieve the required data from the PDB, and download a local copy of the PDB file. The final input in this case should look something like **####** .

### File Path

The other option is to provide the program with a file path pointing to a local file in PDB format (allowed file extensions are .pdb and .txt), regardless of whether this file has been deposited in the PDB or not. No internet connection is required to run the program with this kind of input. Once again, this should be enclosed in inverted commas. The final input in this case should look something like **path/to/PDB/file/foo.pdb** .

## Optional Inputs:

(**RABDAM** can in addition take a number of optional inputs to vary the output of the program. Unless you have a specific reason for changing these outputs, leaving them as the default is recommended.)

### PDT

If the **Packing Density Threshold** used by **RABDAM** is to be varied, **PDT=##** (where ## is a number in ångströms) can be used. The program uses a default value of 14 (Å). **This number need not be an integer, but MUST be positive.**

### binSize

If the **Bin Size** used by **RABDAM** is to be varied, **binSize=##** (where ## is a number of atoms) can be used. The program uses a default value of 10 (atoms). **This number MUST be a positive integer!**

The program can write PDB files for intermediate stages of PDB manipulation. This is a useful tool for debugging, but as a default these are not produced. (Note that the program will always produce a PDB file of the Unit Cell.) There are two options to produce further PDB files: **createAUCpdb** (create ‘all unit cells’ PDB) and **createTApdb** (create ‘trimmed atoms’ PDB).

createAUCpdb**=True** will make the program write a PDB file which contains all the modelled atoms in a 3x3x3 Unit Cells ‘mini-crystal’.

createTApdb**=True** will make the program write a PDB file which contains all the modelled atoms in a cube defined as the orthogonal limits of the asymmetric unit, with an additional margin (equal to the **PDT**).

If a user does wish to produce these PDB files it should be noted that COOT will rarely – if ever – be able to open and view the set of atomic coordinates. PyMOL is the best program to use if these files are to be viewed and used, or alternatively CCP4MG if the filesize is not too great.

### Sample Input

Optional inputs can be included using comma delimiters after the required input. An example final input with all optional inputs included would look like: **####, PDT=##, binSize=##, createAUCpdb=bool, createTApdb=bool** .

## Location of Input

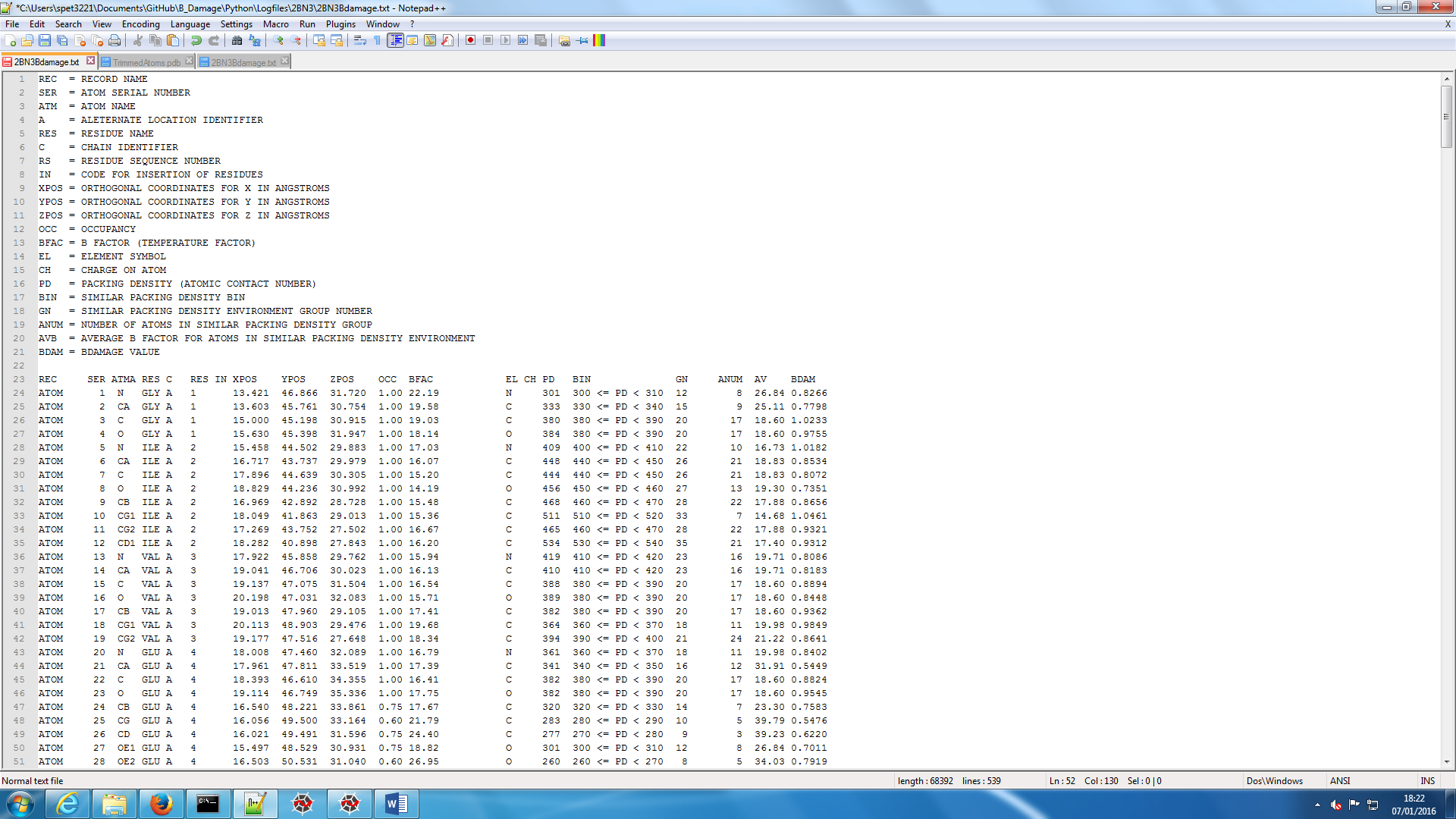
It is currently the case that the input should be constructed on the line at the bottom of the file ‘**INPUT.txt**’, which is found in the working directory. There is currently a placeholder there in the default program which reads **2BN3, PDT=14, binSize=10, createAUCpdb=True, createTApdb=True** . This line can be edited using a text editor program (such as Notepad). The program can then be run with this via the command line by navigating to the directory containing the downloaded Python script (e.g. C:\User\GitHub\B\_Damage\Python) then typing **python RABDAM.py INPUT.txt** in a single lineand pressing the **return** key. In order for the program to function, Python needs to be downloaded onto the machine. If you wish the log to be printed to an output file, then enter **python RABDAM.py INPUT.txt > *output*.txt**. If ***output*.txt** already exists, this file will be overwritten, so be careful! (In this example, ‘*output*’ can be replaced by whatever you want to call the file.)

# Outputs

(**RABDAM** sends all output files to a folder called ‘Logfiles’**.** This parent folder will then contain subfolders for and named after each non-unique structure that the program has processed. In this folder (‘<*name>*’) will be a local copy of the input file in PDB format (‘*<name>.*pdb), ‘<*name>*UnitCell.pdb’ and ‘<*name>*Bdamage.txt’. There may also be optional files ‘AllUnitCells.pdb’ and ‘TrimmedAtoms.pdb’ depending on the inputs used when originally running the program.)

## <name>Bdamage.txt:

This file is the main output of the program. It contains a short header describing what each of the column headers means, then a table containing the output data for all the atoms in the asymmetric unit. These data are produced from a run of the program with the PDB structure ‘2BN3’. The data for each atom is stored in a similar manner as it is in a PDB file (in rows) and each column is defined at the top of the file. The column titled **BDAM** contains the B\_Damage value for each atom.



(In addition, the MATLAB version of the program would output a histogram showing the numbers of atoms that are in each packing density environment. No such functionality currently exists in the Python release. For the MATLAB release and relevant README, please visit <https://github.com/td93/B_Damage>)

## <name>UnitCell.pdb:

This file is generated by PDBCUR. It contains the atomic information for all atoms from the input asymmetric unit, and all other asymmetric units which make up the Unit Cell. For the PDB structure ‘2BN3’, which has space group I2­13, this contains a total of 24 asymmetric units.

# Description of Exceptions Output by RABDAM

During execution of **RABDAM**, a number of exceptions may arise. Here is a list of known errors, and potential reasons that they may occur. If you cannot find an exception you are experiencing in this list, please contact the author at <https://github.com/td93>.

HTTP error: Likely that PDB code does not exist

1. Input file is not formatted correctly, read the handbook for guidelines on input
2. Supplied filepath to PDB is neither a .pdb nor .txt file nor a PDB accession code
3. Supplied filepath does not exist
4. Failed to download and save PDB - cause unknown (check internet connection)
5. Failed to copy PDB to a local version. Check that supplied PDB is not in use by another program
6. Failed to generate Unit Cell PDB file. It is likely that PDBCUR failed to run to completion, or experienced an error

# For Developers

Developers are welcome to write their own Python plug-ins and attach them to the script. The source code is available from <https://github.com/td93/B_Damage/tree/master/Python>

A short script exists in the *Subroutines* subdirectory named ‘**devRun.py**’ which can be edited and run in your development environment. This is the fastest way to run the code, in my experience. The last line of this file may be edited to your specific inputs, according to the details in this handbook. The only addition is that the input variable relating to the input file is a string, and must therefore be enclosed in inverted commas, regardless of whether it is a filepath or a PDB code that is supplied. There is a placeholder there demonstrating this that currently reads **RABDAM(‘2BN3’)**.