An open source software tool to assign the material properties of bone for ABAQUS finite element simulations

Elise C Pegg[1*], Harinderjit S Gill[1]

[1] Department of Mechanical Engineering, University of Bath, Bath, UK

* Corresponding Author:

Dr Elise C Pegg

Department of Mechanical Engineering

University of Bath, Bath

BA2 7AY

Tel: 01225 383375

Email: e.c.pegg@bath.ac.uk

Keywords: Material properties; Bone; Finite Element; Python.

Word Count: 1,989

Abstract

A new software tool to assign the material properties of bone to an ABAQUS finite element mesh was created and compared with Bonemat, a similar tool originally designed to work with Ansys finite element models. Our software tool (py_bonemat_abaqus) was written in Python, which is the chosen scripting language for ABAQUS. The purpose of this study was to compare the software packages in terms of the material assignment calculation and processing speed. Three element types were compared (linear hexahedral (C3D8), linear tetrahedral (C3D4) and quadratic tetrahedral elements (C3D10)), both individually and as part of a mesh.

Comparisons were made using a CT scan of a hemi-pelvis as a test case. A small difference, of -0.05 kPa on average, was found between Bonemat version 3.1 (the current version) and our python package. Errors were found in the previous release of Bonemat (version 3.0 downloaded from www.biomedtown.org) during calculation of the quadratic tetrahedron Jacobian, and conversion of the apparent density to modulus when integrating over the Young's modulus field. These issues caused up to 2 GPa error in the modulus assignment. For these reasons, we recommend users upgrade to the most recent release of Bonemat.

Processing speeds were assessed for the three different element types. Our Python package took significantly longer (110 s on average) to perform the calculations compared with the Bonemat software (10 s). Nevertheless, the workflow advantages of the package and added functionality makes 'py_bonemat_abaqus' a useful tool for ABAQUS users.

1 Introduction

Bone has heterogeneous material properties, and so in order to create representative finite element models it is essential that the properties are correctly described in order to draw useful results. It is possible to calculate the apparent density of bone from a Computed Tomography (CT) scan. Once the density is known, it can be used to calculate the Young's modulus value, typically using a power equation. Opinion varies as to which mathematical relationships are the most appropriate to use. Helgason *et al.* have published a comprehensive review on the area (Helgason, Perilli et al., 2008) and it is not the purpose of this study to go into further detail on the subject.

What we are concerned with is the practical implementation of such mathematical relationships when using the finite element software ABAQUS (Simulia, Dassault Systèmes, Paris, France). It is possible to make such material assignments for an ABAQUS input file using commercial segmentation software, such as Simpleware (Simpleware Ltd., Exeter, UK) or Mimics (Materialise, Leuven, Belgium), but the cost of such software can be prohibitive. Researchers at the Istituto Ortopedico Rizzoli in Bologna, Italy, have created a software program called Bonemat to help with accurate material assignment for a finite element mesh. Bonemat is publically available and can be freely downloaded, making it a useful research tool. However, it has three main limitations:

1. Bonemat will only work on Microsoft Windows, which can be a problem for users of other operating systems.
2. Bonemat is not currently compatible with ABAQUS models, so users have to convert their finite element meshes into a compatible format, using an intermediate software such as Hypermesh (Altair Hyperworks, MI, USA) or custom scripting, before material properties can be assigned.
3. All model information (aside from the mesh) is lost after material assignment; such as, boundary conditions, element sets, or contact definitions. Redefining the element sets and model parameters can add significant time to a project involving multiple models.

The algorithm used by Bonemat to calculate the material properties of bone from a CT scan has evolved over the years. The first iteration (V1) calculated the average Hounsfield Unit (HU) value of all voxels found within an element; these were then converted to the modulus of bone to use for the element material (Zannoni, Mantovani et al., 1999). The second iteration (V2) used trilinear interpolation of the CT scan to more accurately estimate the HU value for each element node, and then

used numerical integration to compute the HU for the element volume, which was then converted to modulus (Taddei, Pancanti et al., 2004). The third iteration (V3) first converted the CT scan voxels to modulus values, and then performed the linear interpolation and numerical integration to find the modulus for each element volume. Taddei *et al.* demonstrated V2 and V3 algorithms do result in different finite element model results. Strain calculated from models of a femur processed with V2 and V3 algorithms were compared to strains measured experimentally. V3 results had a higher regression coefficient with the experimentally measured strains (0.79) compared with V2 results (0.69) (Taddei, Schileo et al., 2007).

We have created an open-source software which has been written in Python (Python Software Foundation, NH, USA) and aims to address the limitations of Bonemat for ABAQUS users. Python was an obvious choice for the program as it is the scripting language used by ABAQUS; furthermore Python works with all operating systems, so the package is multi-platform. It applies the same calculations as Bonemat, allows the same user options, but the mesh data input format is an ABAQUS input file. Furthermore, the program does not remove or change any model parameters already defined in the input file, such as boundary conditions, element sets, or contact definitions. The program also has the added functionality that it can cope with multiple parts, and can 'ignore' parts which are not bone and so do not need material assignment.

The aims of this study were to: (1) assess equivalence between material assignment for individual elements using Bonemat3.1 software (current version, Crimi, 2015), Bonemat3.0 software (previous version, Chiarini, 2006) and our Python package ('py_bonemat_abaqus'), (2) assess equivalence for the modulus grouping and element assignment for a whole bone, and (3) compare the speed of the two software packages.

2 Methodology

Material assignment is applied to a finite element mesh in two stages: (1) the software calculates the modulus for each individual element, then (2) the modulus values for the entire mesh are grouped into bins to reduce computation time for the finite element model. Equivalence was assessed separately for these two stages of the material assignment. First, the numerical integration algorithm was checked on individual elements of controlled mesh quality. Second, modulus values were calculated for a CT scan of a hemi-pelvis using the different software packages; this enabled the grouping algorithms to be compared, and was representative of a typical study case. Lastly, the time taken for each software package to perform the calculations on the hemi-pelvis meshes was assessed.

2.1 Creation of the 'py_bonemat_abaqus' Python package

All scripts were written to be compatible with Python version 2.6 and 2.7 (Python Software Foundation, www.python.org) and were dependent on two other open-source packages: 'numpy' and 'pydicom'. The package can be installed from the Python Package Index using 'pip' or 'easy-install'', or can be downloaded and manually installed using the setup.py script (Pegg, 2015). Once installed, the user can either run the script from the command line, or include the statement 'import py_bonemat_abaqus' in their python scripts, to process their ABAQUS input files.

2.2 Numerical integration accuracy and modulus assignment

The CT scan used was that of a pelvis downloaded from the VAKHUM database, created as part of a project funded by the European Commission under the Information Society Technologies Programme. The dataset was provided by the Laboratory of Human Anatomy and Embryology, University of Brussles (UBL), Belgium (Jan, 2005).

Ten single element input files were created with linear tetrahedral (C3D4), quadratic tetrahedral (C3D10), and linear hexahedral (C3D8) elements, and randomly assigned nodal co-ordinates within the CT scan volume. All elements had a Jacobian determinant greater than 0.2; this limit was recommended by Burkhart *et al.* for biomechanical studies of bone (Burkhart, Andrews et al., 2013). The parameters used for the equivalence tests are summarised in Table 1, and described schematically in Figure 1. The parameters used to convert HU to apparent density (ρ_{app}) (Equation 1) and from ρ_{app} to elastic modulus (Equation 2) were based on reported values for the pelvis (Anderson, Peters et al., 2005). Numerical integration across both the HU field (V2) and the modulus field (V3) were compared.

$$\rho_{app} = -0.021075 + 0.000786\ HU \qquad \text{Equation 1}$$

$$E = 2.0173\ {\rho_{app}}^{2.46} \qquad \text{Equation 2}$$

The single element meshes were analysed using our 'py_bonemat_abaqus' Python package, Bonemat3.0 downloaded from the BiomedTown website (Chiarini, 2006), and Bonemat3.1 from www.bonemat.org (Crimi, 2015).

2.3 Modulus grouping and assignment of multiple elements

The hexahedral finite element mesh of the left hemi-pelvis from the VAKHUM database was used for the material assignment of the bone. The linear and tetrahedral meshes were created from the hexahedral mesh by dividing each hexahedron into five tetrahedral elements. The hexahedral element mesh had 5,929 elements, the linear tetrahedral mesh had 29,645 elements, and the quadratic tetrahedral mesh had 29,645 elements. Each model was analysed using the parameters detailed in Table 1. Results were processed using both the V2 and the V3 algorithms.

2.4 Speed comparison

Python is an interpreted programming language, and therefore scripts written in Python are typically slower compared with pre-compiled programs. For this reason, it was important to quantify any difference in speed. The time taken to analyse the models described in Section 2.3 using the different software packages was assessed, each model was measured 5 times. Tests were performed on a Windows 7 PC with a 64-bit operating system, with four CPUs, 8 GB of RAM, and an Intel Core i5-3470 processor.

3 Results

The assigned modulus values for individual elements were very similar between the newest release of Bonemat (Crimi, 2015) and our python package (Figures 2a-c). The mean absolute difference was -0.47 kPa (standard deviation: 13.55 kPa, range: -35.00 to 45.28 kPa). Modulus values from the older release of Bonemat were also similar for the linear hexahedral and tetrahedral elements, with a mean absolute difference of -6.00 kPa (standard deviation: 16.67 kPa, range: -49.00 to 14.33 kPa). However, for the quadratic tetrahedral elements, the mean absolute difference was -0.19 GPa (standard deviation: 1.43 GPa, range: -3.01 to 1.57 GPa) (Figure 2c).

Modulus assignment of the hemi-pelvis mesh (Figure 3) using the py_bonemat_abaqus software had a good agreement with that calculated by the newest release of Bonemat (Figure 4, Table2). The mean absolute difference was -0.05 kPa (range:-19.48 to 10.23 kPa, standard deviation: 0.62 kPa) (Table 2). The difference in modulus assignment was approximately the same for both numerical integration methods (V2 or V3).

Conversely, the modulus assignment with the old version of Bonemat had a mean absolute difference 38.62 MPa (range: -2193.52 to 1907.02 MPa) (Figure 5, Table 3). Furthermore, when using the V3 algorithm the error linearly increased

with modulus, which for the CT dataset investigated in this study introduced a maximum error of 2.19 GPa. Examination of the results revealed that the equation to convert apparent density to modulus was not correctly applied when the integration over modulus was selected in the software, which introduced the systematic error.

The Python package took longer to perform the calculations compared to both the releases of Bonemat (Figure 6). The python package took a similar time to run for all element types, and this was between 109 and 126 s. The speed of the old version of Bonemat (3.0) was dependent on the element type, ranging from 78s for a hexahedral mesh, to 102s for a quadratic tetrahedral mesh. The newest version of Bonemat (3.1) was the fastest for all element types; the maximum time taken was 20 s to process the quadratic tetrahedral mesh.

4 Discussion

The modulus values calculated by our python package closely matched those from the newest release of Bonemat (Crimi, 2015), and any differences were likely to be due to rounding effects. The average difference calculated for the whole CT volume was -0.05 kPa, which to put into context is 0.00000002% of the typical modulus of cortical bone (20.7 GPa), and 0.00000003% of the modulus of trabecular bone (14.8 GPa) (Rho, Ashman et al., 1993).

Bonemat3.0 had an error in the quadratic tetrahedron modulus calculation which resulted in large differences in modulus assignment. It was possible to reproduce erroneous results by modifying the Jacobian calculation; the two different Jacobian equations are provided as supplementary information. When the representative study case of the hemi-pelvis was analysed for quadratic tetrahedral elements, this calculation error was found to introduce an error of up to 2 GPa, which could have a significant impact on study results. A second bug was observed in the old version of Bonemat software for the V3 calculation, where the equation to convert apparent density to modulus was not correctly applied and introduced additional errors. These results call into question the accuracy of published studies which have used Bonemat3.0 to assign materials to quadratic tetrahedral elements. We recommend that users update their Bonemat software to the newest version which has addressed these issues.

The python package ‘py_bonemat_abaqus’ took 6 to 24 times longer to process the models, depending on the element type, compared to the newest release of Bonemat. This increased run time may be an issue for extremely large models or high resolution CT datasets, however, the research time that will be saved by users not having to convert the file format of

the mesh, or perform manual edits to the input file to re-assign element sets prior to solving the model, may offset the increased time to assign the material properties.

In summary, the 'py_bonemat_abaqus' Python package produced equivalent results to Bonemat3.1 (the current release), but not to Bonemat3.0 (the previous version from the BiomedTown website) which contained some calculation errors. The 'py_bonemat_abaqus' Python package was slower to process the models, but will provide ABAQUS users with a useful method to assign material properties of bone to finite element models, with an easier workflow, fewer limitations, and as it is written in Python, it can be directly incorporated into scripts written to interact with ABAQUS. Through collaboration with the Instituto Ortopedico Rizzoli in Bologna, work is underway to incorporate the ABAQUS input and output parts of the 'py_bonemat_abaqus' script into Bonemat.

Acknowledgements

The methodology and general algorithm for the Python package was based upon the original Bonemat software and we would like to thank the Istituto Ortopedico Rizzoli in Bologna, Italy for their work. Furthermore, the CT scan and finite element mesh used for the testing was from the VAKHUM database, which was kindly provided by the Laboratory of Human Anatomy and Embryology, University of Brussels (ULB) in Belgium and the Istituto Ortopaedico Rizzoli. We would also like to thank the Python Software Foundation, and the Python Package Index, for enabling the source code to be hosted online so other researchers can download and use the 'py_bonemat_abaqus' package.

Conflict of Interest

The authors did not receive payment or services from any third party for any aspect of the submitted work. The authors have received support for research projects which were unrelated to the present study.

References

Anderson, A. E., Peters, C. L., Tuttle, B. D., Weiss, J. A., 2005. Subject-Specific Finite Element Model of the Pelvis: Development, Validation and Sensitivity Studies. Journal of Biomechanical Engineering 127, 364-373.

Burkhart, T. A., Andrews, D. M.,Dunning, C. E., 2013. Finite element modeling mesh quality, energy balance and validation methods: A review with recommendations associated with the modeling of bone tissue. Journal of Biomechanics 46, 1477-1488.

Chiarini, A., 2006. Download Bonemat Software. Retrieved 6/8/2015, 2015, from https://www.biomedtown.org/biomed_town/B3C_Building/products/bonemat/download_html/.

Crimi, G., 2015. Bonemat. Retrieved 6/8/2015, 2015, from http://www.bonemat.org/downloads.php.

Helgason, B., Perilli, E., Schileo, E., Taddei, F., Brynjólfsson, S., Viceconti, M., 2008. Mathematical relationships between bone density and mechanical properties: A literature review. Clinical Biomechanics 23, 135-146.

Jan, S. V. S., 2005. The VAKHUM Project: Virtual Animation of the Kinematics of the Human. Theoretical Issues in Ergonomics Science 6.3-4, 277-279.

Pegg, E. ,2016. py_bonemat_abaqus.. Retreved 7/1/2016, 2016, from https://pypi.python.org/pypi/py_bonemat_abaqus/

Rho, J. Y., Ashman, R. B., Turner, C. .H., 1993. Young's modulus of trabecular and cortical bone material: ultrasonic and microtensile measurements. Journal of Biomechanics 26, 111-119.

Taddei, F., Pancanti, A., Viceconti, M., 2004. An improved method for the automatic mapping of computed tomography numbers onto finite element models. Medical Engineering and Physics 26, 61-69.

Taddei, F., Schileo, E., Helgason, B., Christofolini, L., Viceconti, M., 2007. The material mapping strategy influences the accuracy of CT-based finite element models of bones: An evaluation against experimental measurements. Medical Engineering and Physics 29, 973-979.

Zannoni, C., Mantovani, R., Viceconti, M., 1999. Material properties assignment to finite element models of bone structures: a new method. Medical Engineering and Physics 20, 735-740.

Table and Figure Captions

Table 1. Parameters used when assessing equivalence between material assignments of the software packages.

Table 2. Summary of the modulus difference (py_bonemat_abaqus – Bonemat3.1) of the hemi-pelvis when analysed using different elements and numerical integration algorithms. Results are shown in kPa.

Table 3. Summary of the modulus difference (py_bonemat_abaqus – Bonemat3.0) of the hemi-pelvis when analysed using different elements and numerical integration algorithms. Results are shown in MPa.

Figure 1. Schematic illustration of the parameters detailed in Table 1 and how they influence the grouping of the modulus results.

Figure 2. Comparison of the modulus values output by Bonemat3.1 and Bonemat3.0 to the py_bonemat_abaqus python package, for the (a) linear hexahedron, (b) linear tetrahedron, and (c) quadratic tetrahedron (n=10) with the V2 algorithm.

Figure 3. Comparison of the modulus values assigned to the hexahedral mesh of the hemi-pelvis by py_bonemat_abaqus and Bonemat 3.1 using the V3 algorithm. An anterior-posterior view of the whole mesh is shown, and a cross-sectional view through the volume centroid.

Figure 4. Bland Altman plots summarising the difference in assigned modulus (in GPa) for the hemi-pelvis mesh for all element types combined. Differences shown compare the py_bonemat_abaqus to Bonemat 3.1 analysed with (a) the V2 algorithm, (b) the V3 algorithm. Bland-Altman plots with kPa y-axis scales are provided as supplementary information

Figure 5. Bland Altman plots summarising the difference in assigned modulus (in GPa) for the hemi-pelvis mesh for all element types combined. Differences shown compare the py_bonemat_abaqus to Bonemat 3.0 analysed with (a) the V2 algorithm, (b) the V3 algorithm.

Figure 6. Summary of the time taken for each software to perform the calculations and output the modified mesh file for the three element types. Error bars represent the standard deviation in the time measurement (n=5). Tests were performed on a Windows 7 PC with a 64-bit operating system, with four CPUs, 8 GB of RAM, and an Intel Core i5-3470 processor.

Tables and Figures

Parameter	Description	Set value
Gap Value	Modulus interval used when grouping the modulus values	0.01 GPa
Integration order	Order of the numerical integration used across the element	4
CT calibration coefficients [ρ_{app}=a+b HU]	Calibration parameters *a* and *b* of the equation used to calculate the apparent density of bone (ρ_{app}) from the CT Hounsfield Unit (HU)	a = -0.021075, b = 0.000786
Calibration correction	Option which allows correction of the calibration with up to three linear correlation.	Not applied
Modulus calculation parameters [E=a+b ρ_{app}^{c}]	Parameters *a*, *b* and *c* of the power equation used to convert the apparent density of bone to Young's modulus	a = 0, b = 2.0173, c = 2.46
Minimum modulus value	Any modulus values below the minimum are changed to the minimum value	0.000001

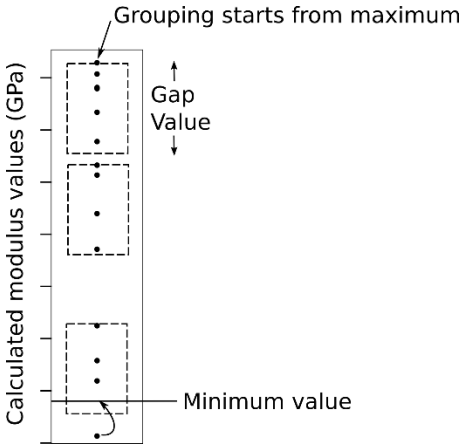
Table 1. Parameters used when assessing equivalence between material assignments of the software packages.

Element Type	Algorithm	Mean Absolute Difference	Standard Deviation	Minimum	Maximum
Linear Hex	V2	-0.11	0.96	-18.38	6.91
Linear Tet	V2	-0.18	0.09	-0.24	0.05
Quad Tet	V2	0.00	0.13	-6.15	4.27
All Elements	V2	-0.10	0.39	-18.38	6.91
Linear Hex	V3	-0.07	0.92	-19.48	10.23
Linear Tet	V3	0.05	1.47	-10.01	0.45
Quad Tet	V3	0.00	0.16	-6.90	5.11
All Elements	V3	-0.01	0.85	-9.48	10.23
All results combined		**-0.05**	**0.62**	**-19.48**	**10.23**

Table 2. Summary of the modulus difference (py_bonemat_abaqus – Bonemat3.1) of the hemi-pelvis when analysed using different elements and numerical integration algorithms. Results are shown in kPa.

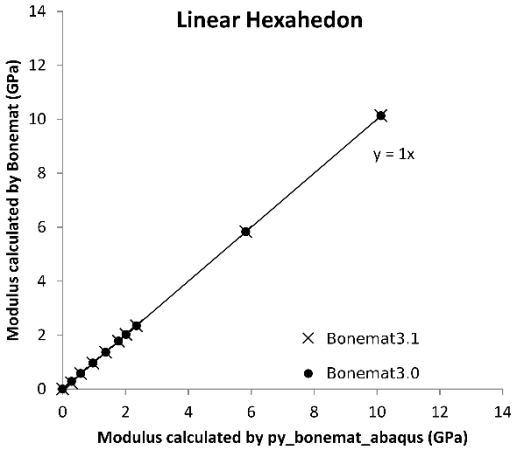
Element Type	Algorithm	Mean Absolute Difference	Standard Deviation	Minimum	Maximum
Linear Hex	V2	0.00	0.00	-0.04	0.02
Linear Tet	V2	0.00	0.08	-0.14	10.00
Quad Tet	V2	-0.21	36.58	-2193.52	1242.79
All Elements	V2	-0.07	12.22	-2193.52	1242.79
Linear Hex	V3	81.06	108.60	5.70	672.33
Linear Tet	V3	75.68	121.16	-2.38	812.07
Quad Tet	V3	75.18	124.04	-1226.69	1907.02
All Elements	V3	77.31	117.93	-1226.69	1907.02
All results combined		**38.62**	**65.08**	**-2193.52**	**1907.02**

Table 3. Summary of the modulus difference (py_bonemat_abaqus – Bonemat3.0) of the hemi-pelvis when analysed using different elements and numerical integration algorithms. Results are shown in MPa.

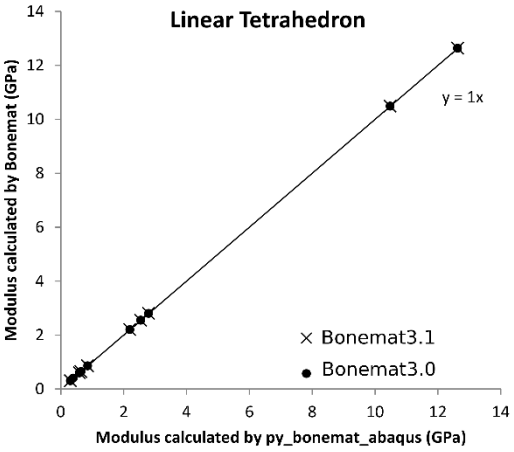


228

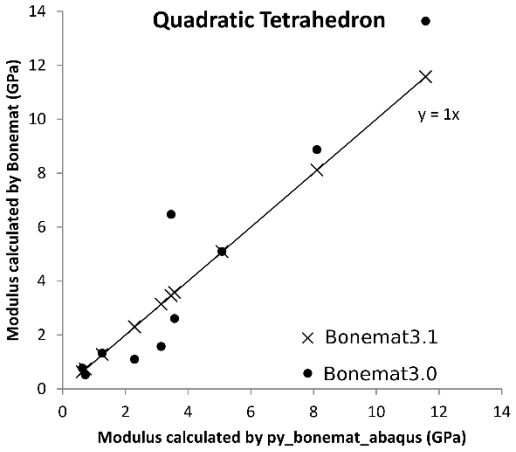
229 Figure 1. Schematic illustration of the parameters detailed in Table 1 and how they influence the grouping of the modulus
230 results.



231 a.



b.



232 c.

233 Figure 2. Comparison of the modulus values output by Bonemat3.1 and Bonemat3.0 to the py_bonemat_abaqus python
234 package, for the (a) linear hexahedron, (b) linear tetrahedron, and (c) quadratic tetrahedron (n=10) with the V2 algorithm.

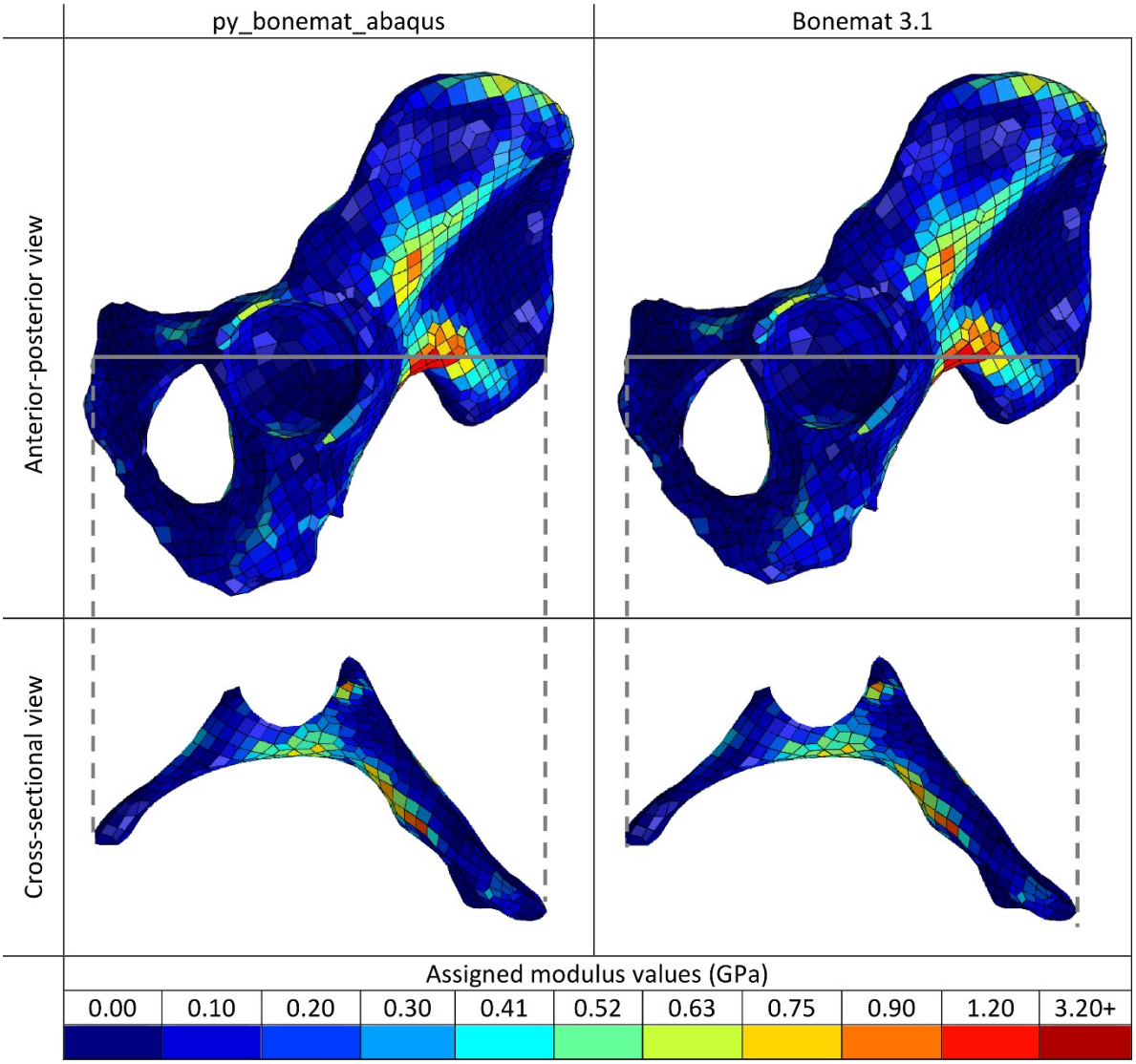


Figure 3. Comparison of the modulus values assigned to the hexahedral mesh of the hemi-pelvis by py_bonemat_abaqus and Bonemat 3.1 using the V3 algorithm. An anterior-posterior view of the whole mesh is shown, and a cross-sectional view through the volume centroid.

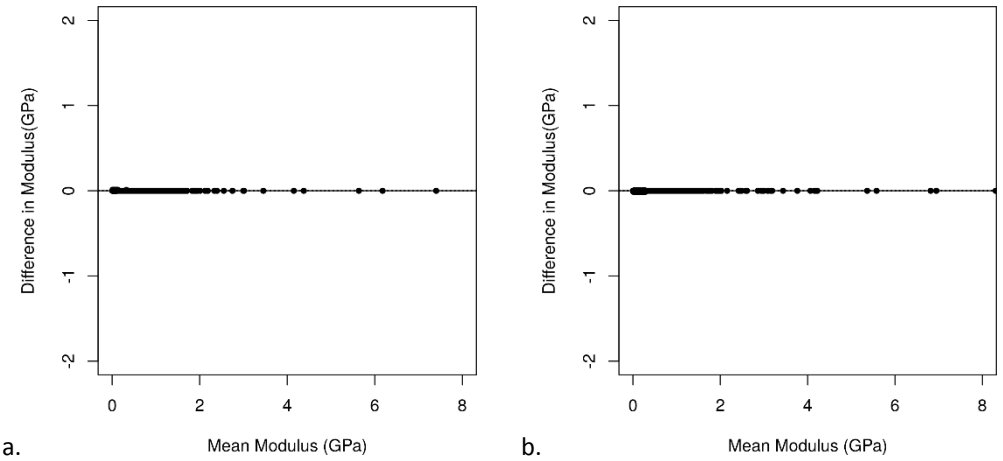


Figure 4. Bland Altman plots summarising the difference in assigned modulus (in GPa) for the hemi-pelvis mesh for all element types combined. Differences shown compare the py_bonemat_abaqus to Bonemat 3.1 analysed with (a) the V2 algorithm, (b) the V3 algorithm. Bland-Altman plots with kPa y-axis scales are provided as supplementary information

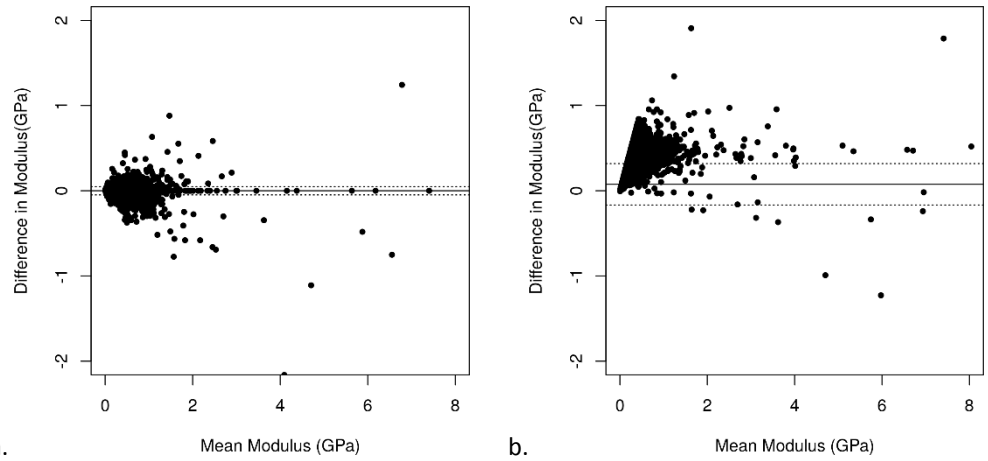


Figure 5. Bland Altman plots summarising the difference in assigned modulus (in GPa) for the hemi-pelvis mesh for all element types combined. Differences shown compare the py_bonemat_abaqus to Bonemat 3.0 analysed with (a) the V2 algorithm, (b) the V3 algorithm.

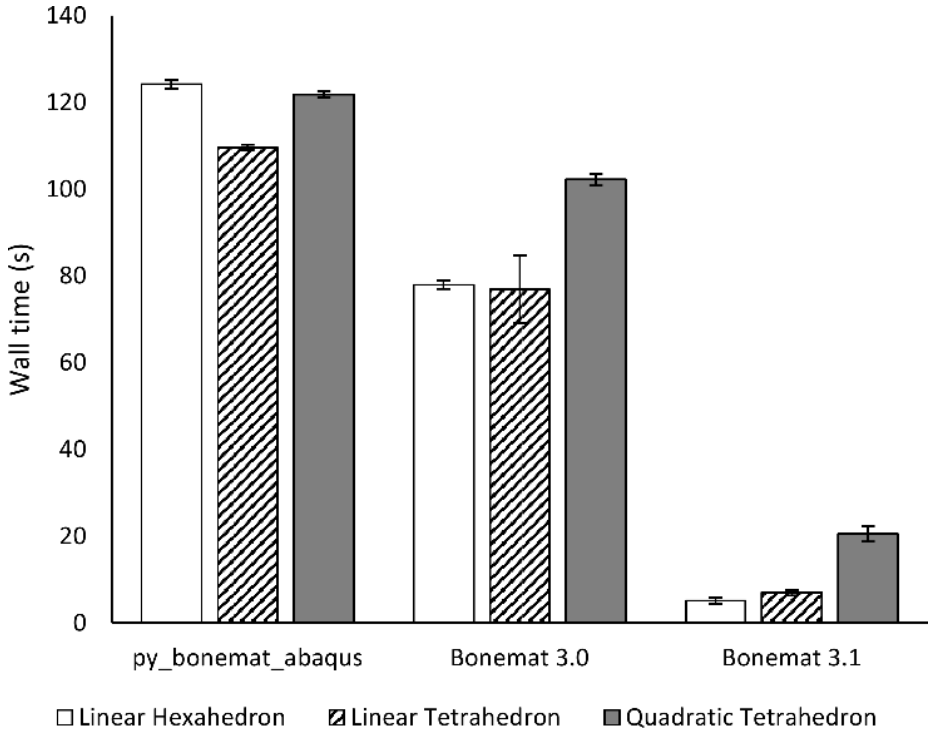


Figure 6. Summary of the time taken for each software to perform the calculations and output the modified mesh file for the three element types. Error bars represent the standard deviation in the time measurement (n=5). Tests were performed on a Windows 7 PC with a 64-bit operating system, with four CPUs, 8 GB of RAM, and an Intel Core i5-3470 processor.

Supplementary Information 1

Jacobian calculation for a quadratic tetrahedron by the 'py_bonemat_abaqus' Python package.

Assuming the shape functions defined in Equation 1, where l, r, s, and t represent the natural co-ordinate system of a tetrahedron, the differentiation with respect to each of the natural co-ordinates is shown in Equations 2-5. Based on these differentiated shape functions, the Jacobian can be calculated using the natural co-ordinates and the nodal co-ordinates $[(x^1, y^1, z^1), (x^2, y^2, z^2) \ldots (x^{10}, y^{10}, z^{10})]$ using Equation 6 (the more compact transpose matrix is shown).

$$N = \begin{bmatrix} (2l-1)l \\ (2r-1)r \\ (2s-1)s \\ (2t-1)t \\ 4lr \\ 4rs \\ 4ls \\ 4lt \\ 4rt \\ 4st \end{bmatrix} \quad (1)$$

$$\frac{\delta N}{\delta l} = \begin{bmatrix} 4l-1 \\ 0 \\ 0 \\ 0 \\ 4r \\ 0 \\ 4s \\ 4t \\ 0 \\ 0 \end{bmatrix} \quad \frac{\delta N}{\delta r} = \begin{bmatrix} 0 \\ 4r-1 \\ 0 \\ 0 \\ 4l \\ 4s \\ 0 \\ 0 \\ 4t \\ 0 \end{bmatrix} \quad \frac{\delta N}{\delta s} = \begin{bmatrix} 0 \\ 0 \\ 4s-1 \\ 0 \\ 0 \\ 4r \\ 4l \\ 0 \\ 0 \\ 4t \end{bmatrix} \quad \frac{\delta N}{\delta t} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 4t-1 \\ 0 \\ 0 \\ 0 \\ 4l \\ 4r \\ 4s \end{bmatrix} \quad (2), (3), (4), (5)$$

$$J^T = 4\begin{bmatrix} \frac{1}{4} & x^1\left(l-\frac{1}{4}\right)+x^5r+x^7s+x^8t & y^1\left(l-\frac{1}{4}\right)+y^5r+y^7s+y^8t & z^1\left(l-\frac{1}{4}\right)+z^5r+z^7s+z^8t \\ \frac{1}{4} & x^2\left(r-\frac{1}{4}\right)+x^5l+x^6s+x^9t & y^2\left(r-\frac{1}{4}\right)+y^5l+y^6s+y^9t & z^2\left(r-\frac{1}{4}\right)+z^5l+z^6s+z^9t \\ \frac{1}{4} & x^3\left(s-\frac{1}{4}\right)+x^6r+x^7l+x^{10}t & y^3\left(s-\frac{1}{4}\right)+y^6r+y^7l+y^{10}t & z^3\left(s-\frac{1}{4}\right)+z^6r+z^7l+z^{10}t \\ \frac{1}{4} & x^4\left(t-\frac{1}{4}\right)+x^8l+x^9r+x^{10}s & y^4\left(t-\frac{1}{4}\right)+y^8l+y^9r+y^{10}s & z^4\left(t-\frac{1}{4}\right)+z^8l+z^9r+z^{10}s \end{bmatrix}^T \quad (6)$$

Jacobian calculation used by the old release of the Bonemat software

The shape functions used for the old release of Bonemat software (Chiarini, 2006) were the same, but l was defined as equal *to 1-r-s-t,* and then a 3x3 Jacobian matrix was used (Equation 7). This Jacobian calculation was erroneous.

$$J = \begin{bmatrix} x^1(1-4l)+x^2(4r-1)+x^5(4l-4r)+(x^6-x^7)4s+(x^9-x^8)4t & \ldots same\ for\ y & \ldots same\ for\ z \\ x^1(1-4l)+x^3(4s-1)+x^7(4l-4s)+(x^4-x^5)4r+(x^{10}-x^8)4t & \ldots same\ for\ y & \ldots same\ for\ z \\ x^1(1-4l)+x^4(4t-1)+x^8(4l-4t)+(x^9-x^5)4r+(x^{10}-x^7)4s & \ldots same\ for\ y & \ldots same\ for\ z \end{bmatrix} \quad (7)$$

Supplementary Information 2 – Individual Bland Altman Plots

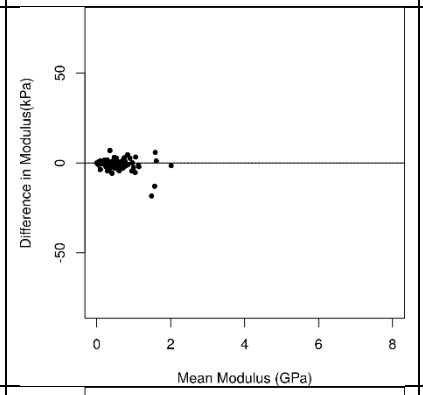
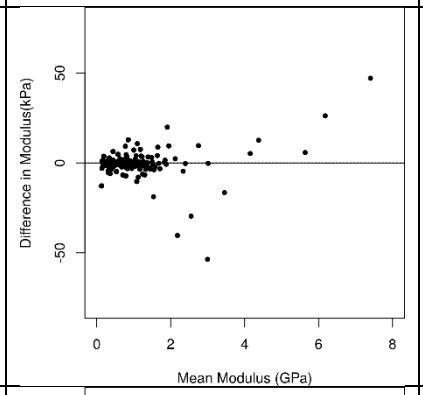
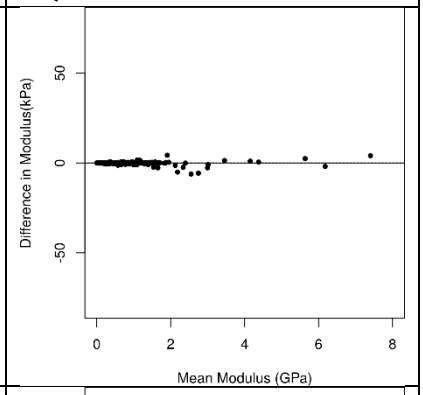
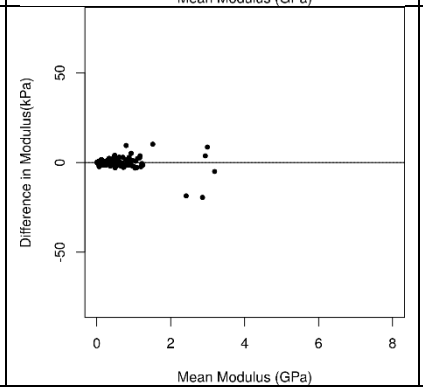
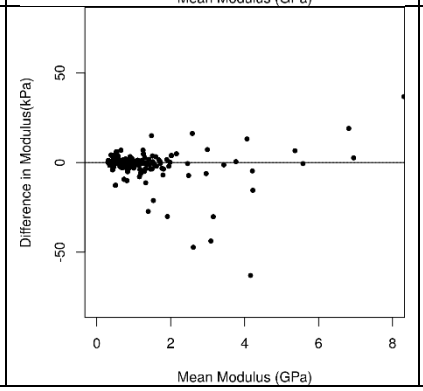
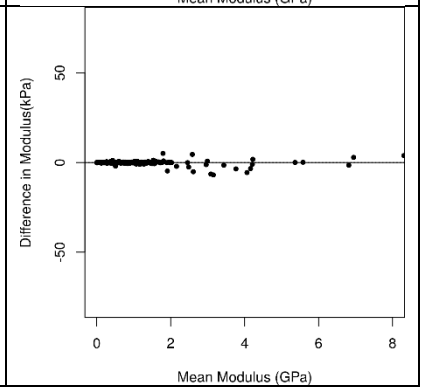
Algorithm	Linear Hexahedral Elements	Linear Tetrahedral Elements	Quadratic Tetrahedral Elements
V2			
V3			

Table S1. Bland Altman plots comparing the difference in modulus assignment with Bonemat3.1 and the py_bonemat_abaqus package (py_bonemat_abaqus - Bonemat3.1) of an element mesh of the hemi-pelvis mesh. Difference shown is in kPa.

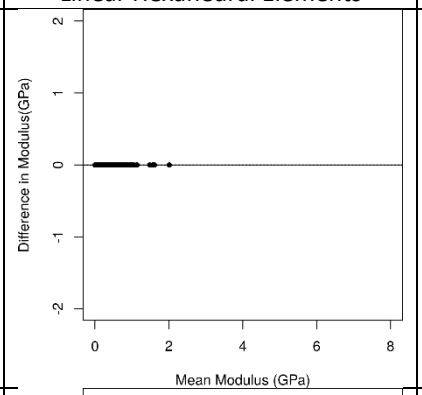
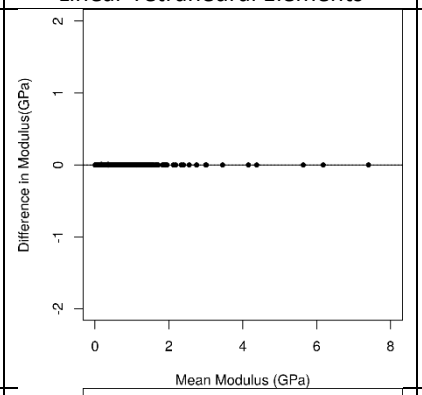
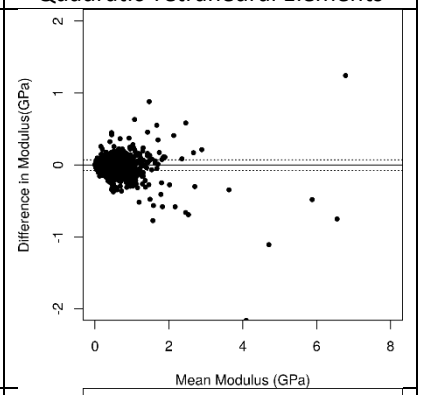
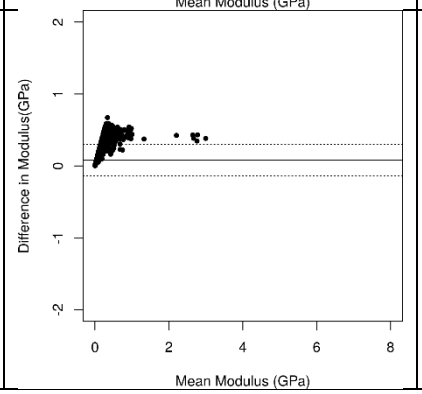
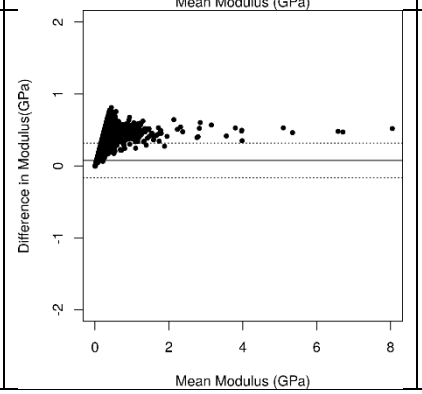
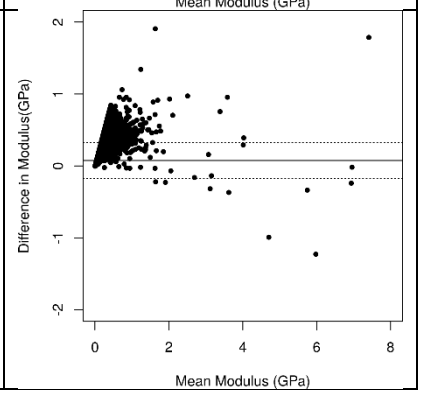
Algorithm	Linear Hexahedral Elements	Linear Tetrahedral Elements	Quadratic Tetrahedral Elements
V2			
V3			

Table S2. Bland Altman plots comparing the difference in modulus assignment with Bonemat3.0 and the py_bonemat_abaqus package (py_bonemat_abaqus - Bonemat3.0) of an element mesh of the hemi-pelvis mesh. Difference shown is in GPa.