Code ocean calculation capsule
An injectable meta-biomaterial
for the manuscript
"An injectable meta-biomaterial: From Design and Simulation to In-vivo Shaping and Tissue induction"
(https://doi.org/10.1002/adma.202102350)

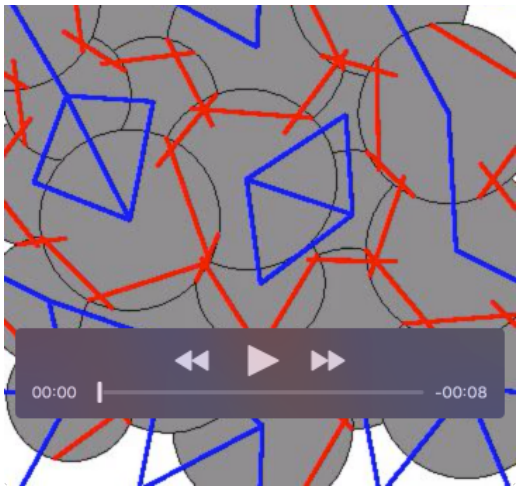
1. Aim and scope of the Code Ocean capsule "An injectable meta-biomaterial" 2
   1.1. Python Simulation "particleShear" 2
   1.2. Simulation figures 2
   1.3. Physical measurements and *in-vivo* data 3
2. Python simulation particleShear 3
   2.1. Source code 4
   2.2. System requirements 4
   2.3. Installation Guide 5
   2.4. Demo 6
      2.4.1. Run instructions 6
      2.4.2. Expected output 7
      2.4.3. Short demo 8
   2.5. Instructions for use and reproduction of the results presented in the main manuscript 10
3. Structure of the CodeOcean capsule 11
   3.1. The run file 11
   3.2. Demo simulations 12
   3.3. Data analysis 12
   3.4. Main scripts of the CodeOcean capsule 12
      3.4.1. Short simulation demo 13
      3.4.2. Full simulation demo 13
      3.4.3. Data import 13
      3.4.4. Plotting and statistical analysis 14
   3.5. Details on the python simulation in a capsule without graphical display 14
   3.6. Details on data analysis: import from raw files, figure plotting and statistics 15
      3.6.1. Data analysis process flow 15
      3.6.2. Reproducibility of the data evaluation 17
4. Documentation 18
   4.1. Python simulation 18
      4.1.1. Quick install guide 18
      4.1.2. Manual 18
      4.1.3. API documentation 19
   4.2. Custom R libraries 19
   4.3. R-Data files 20
   4.4. CodeOceanOnly figures 20
   4.5. Statistical reporting 20
5. Datasets in this CodeOcean Capsule 20
6. Bibliography 21

1. Aim and scope of the Code Ocean capsule “An injectable meta-biomaterial”

The aim of this Code Ocean capsule is to provide reproducible calculation support to the associated manuscript “An Injectable Meta-biomaterial: From Design and Simulation to In-vivo Shaping and Tissue induction”[1]. It aims at providing reproduction of the Python “particleShear”[2] simulation, and also of the quantitative figures of the manuscript.

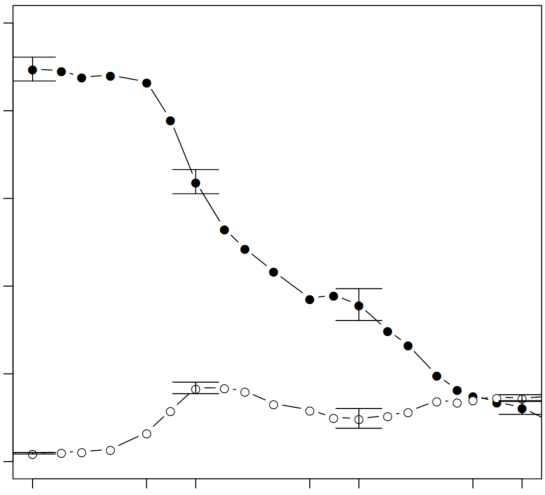
The associated manuscript[1] indeed relies in part on a new software: a Python simulation (“particleShear”[2]) of microgel particles, either as spherical individual particles or crosslinked together to form various overall irregular particles. This CodeOcean capsule provides a reproducible environment where this software can be run, in addition to reproducible generation of the figures.

This document describes the high-level structure and usage of the calculation capsule associated with the manuscript indicated above.



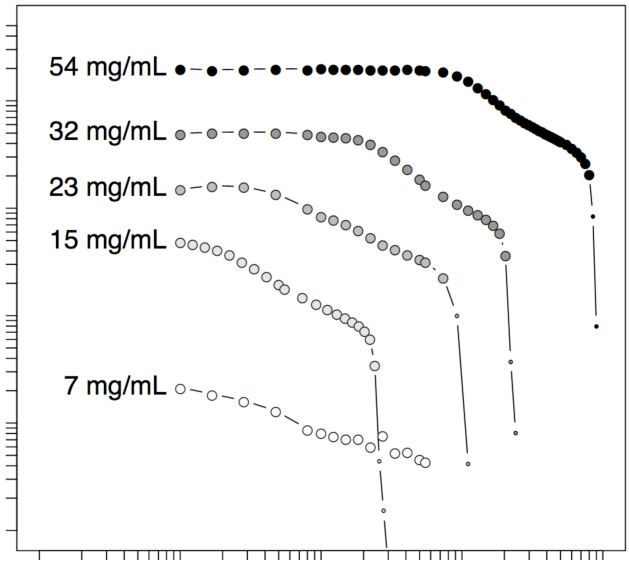
1.1. Python Simulation “particleShear”

The capsule provides demos and unit tests of the Python particleShear microgel simulation[2]. This includes representative simulations (Supplemental Videos S1-S4) as used for the main manuscript[1]. These remain however demos (a few hours of calculation time). Data from ca. 30’000 hours worth of calculation time ran un the Baobab cluster of the University of Geneva is however evaluated in this CodeOcean Capsule.



1.2. Simulation figures

Fig. 2 in the associated manuscript[1] draws on the Python simulation. The capsule contains the overview data from a cluster-based Python particleShear[2] simulation (Baobab, University of Geneva, Switzerland). It allows recreating the quantitative figures related to simulation, but also to explore the rich dataset without an immediate need for cluster.



1.3. Physical measurements and *in-vivo* data

In the associated manuscript[1], we used numerical simulation to design a novel type of biocompatible injectable meta-material that obtains a softening transition combined with large yield strains by frictional interaction between irregular, porous particles. We confront this theoretical design with physical measurements on our novel material, termed EPI biomaterial for "Elastic Porous Injectable". This Code Ocean capsule also contains our quantitative raw data on rheology, uniaxial compression, porosity, injectability, *in-vivo* shape-ability, *in-vivo* survival and weight gain, as well as hematology data. The Code Ocean capsule therefore enables replotting of the quantitative physical measurements (Fig. 3 of the manuscript), *in-vivo* shape performance and biocompatibility (Fig. 4 of the manuscript) as well as meta-material generalization including cell adhesion data (Fig. 5 of the manuscript). It also enables replotting of all the supplementary figures (reported in the supporting information to the main manuscript[1]) as well as a series of additional CodeOcean-only figures (documented in /code/Documentation/CodeOceanOnlyResults/CodeOceanOnlyResults.pdf). Detailed information about the statistical testing and description are collected from the output of the R scripts in this Capsule and reported in the Excel table available at /code/Documentation/Statistical Reporting.xlsx and also summarized in Table S7 in the supporting information of the main manuscript[1].

This data evaluation in R is in part based on custom R libraries automatically loaded in the Capsule, but also available for separate download: textureAnalyzerGels[3] for analysing gel compression data, rheologyEvaluation[4] for importing and analyzing rheological sweeps and particleShearEvaluation[5] for importing and analyzing digital rheological sweeps generated by the particleShear Python simulation[2]. For graphing and validation of numerical precision we use further custom R packages[6, 7].

2. Python simulation particleShear

The simulation "particleShear" is an installable Python module. It enables simulation of elastically and frictionally interacting particles. It is inspired by the simulations from Otsuki et a.l.[8], but provides the added possibility to crosslink neighboring particles. Regarding evaluation, it is mainly aimed at performing simulated shear rheology, including at large amplitudes, and special care was taken to ensure strictly symmetrical stress tensor evaluation (using the framework of Nicot et al.[9], with correction for some

calculation mistakes, see "/code/Documentation/Simulation particleShear/Manual particleShear.pdf", part 1).

2.1. Source code

The source code of the Python module particleShear is located on the Python distribution server (webpage at https://pypi.org/project/particleShear/), and thus installable automatically via the Python installer pip (e.g. via pip install particleShear or variants of it; the exact install command on the present CodeOcean capsule in the Dockerfile is (corresponding to graphical view in screenshot 1):

```
pip3 install -U --no-cache-dir \
    particleshear==1.0.2
```

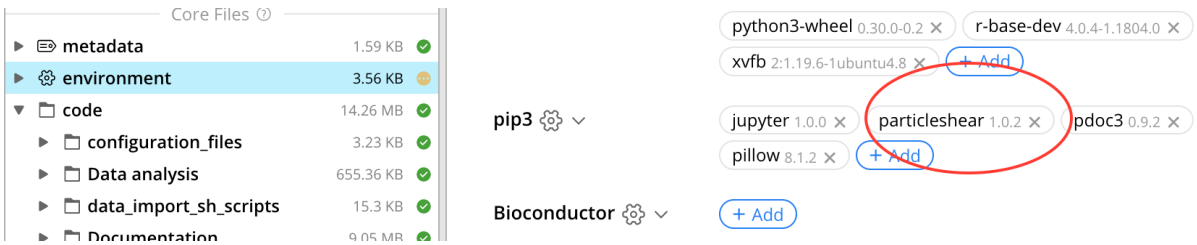
Releases of particleShear are archived permanently on Zenodo, release 1.0.2 in particular is available at https://doi.org/10.5281/zenodo.4589212.[2] The current development version is hosted on Github: https://github.com/tbgitoo/particleShear, for development purposes, it is best to clone the repository from there.

2.2. System requirements

- To run the particleShear[2] simulations as preconfigured on this CodeOcean capsule, no particular external System requirements need to be satisfied. All the dependencies are pre-installed within this Capsule.

- To run the simulation locally (after download and installation, see below), a processor capable of running Python 3.5 or higher is required; at the time of writing, this includes all common personal computer and server architectures (Examples : Intel Atom® processor or Intel® Core™ i3 processor), but also some embeddable systems like the Raspberry Pi. There is no requirement for non-standard hardware.

- Local installation have successfully been tested on Windows 7, Windows 10, MacOSX 10.11.1, and Linux (CentOS 6.1)



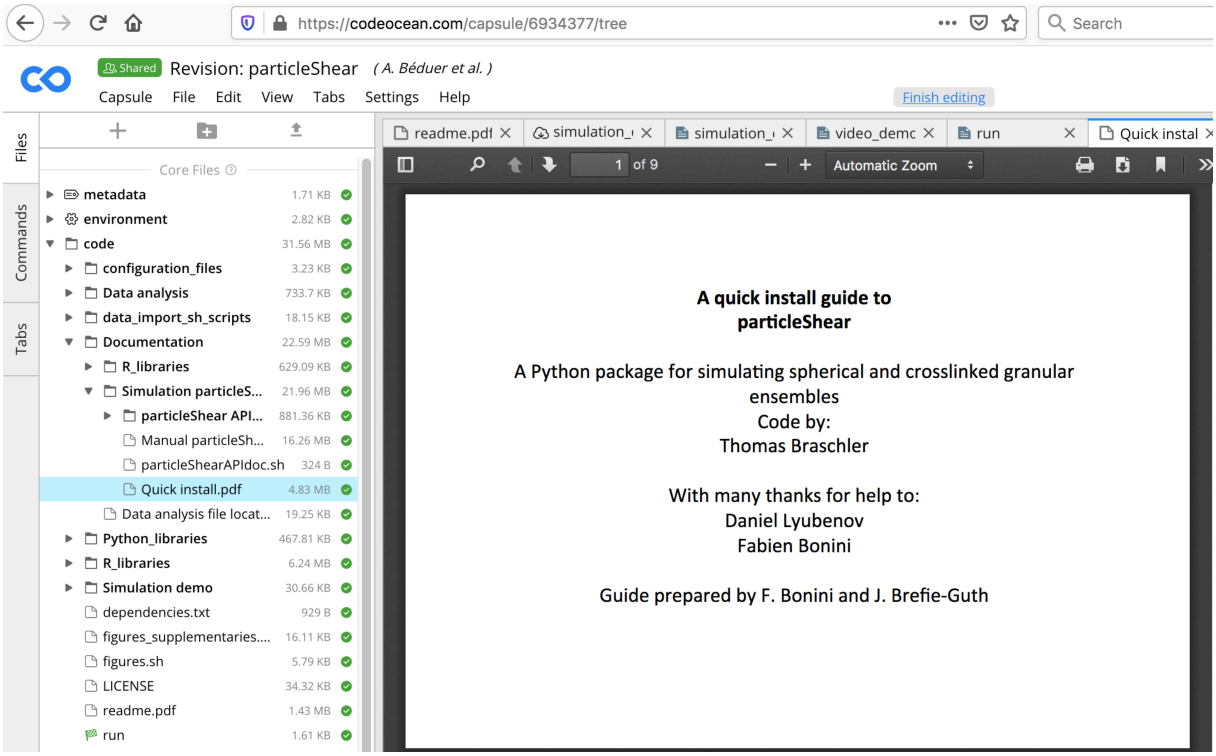
Screenshot 1. Configuration of the particleShear package in the CodeOcean capsule. Following configuration in the environment section, the installation is automated from the Python distribution server (pypi.org).

2.3. Installation Guide

As the particleShear simulation is automatically installed in the CodeOcean capsule, no local installation is required. In the CodeOcean capsule, the installation is handled via the standard Python installer (pip3, see above) from the standard Python server where a copy of the particleShear Python module is hosted (https://pypi.org/project/particleShear/)

The simulations can however also be run locally after download and install. Detailed installation instructions are provided as a separate Quick Install Guide (at "/code/Documentation/Simulation particleShear/Quick install.pdf", see Screenshot 2). In the nominal case, the local installation takes about 10 minutes. Running the small demo described in the Quick install guide takes another 20-30 minutes, the unit test a few minutes.

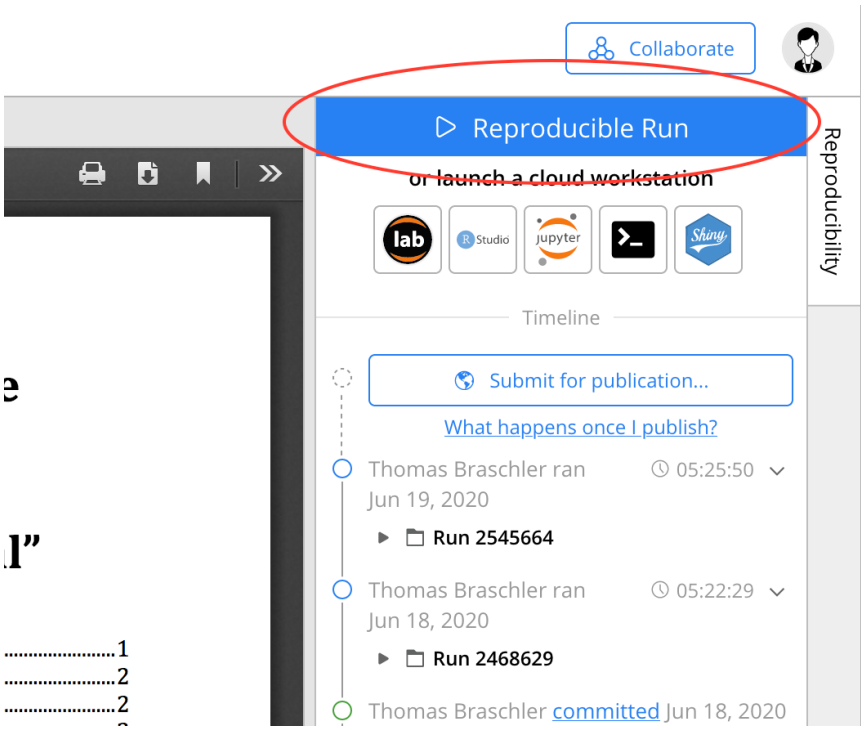
This being said, in our experience, while supposedly trivial, the installation of Python itself can be tricky. We therefore recommend to at least initially use the CodeOcean environment. Issues with Python installation are not directly related to particleShear, and possible unfortunate users are referred to the many relevant forum discussions regarding all sorts of problems ranging from system path configuration, concurrent Python versions and pip package installers, to missing or inappropriately configured graphical display devices that may arise.



Screenshot 2. Quick install guide. The Quick install guide can be viewed directly on this CodeOcean capsule, at "/code/Documentation/Simulation particleShear/Quick install.pdf"

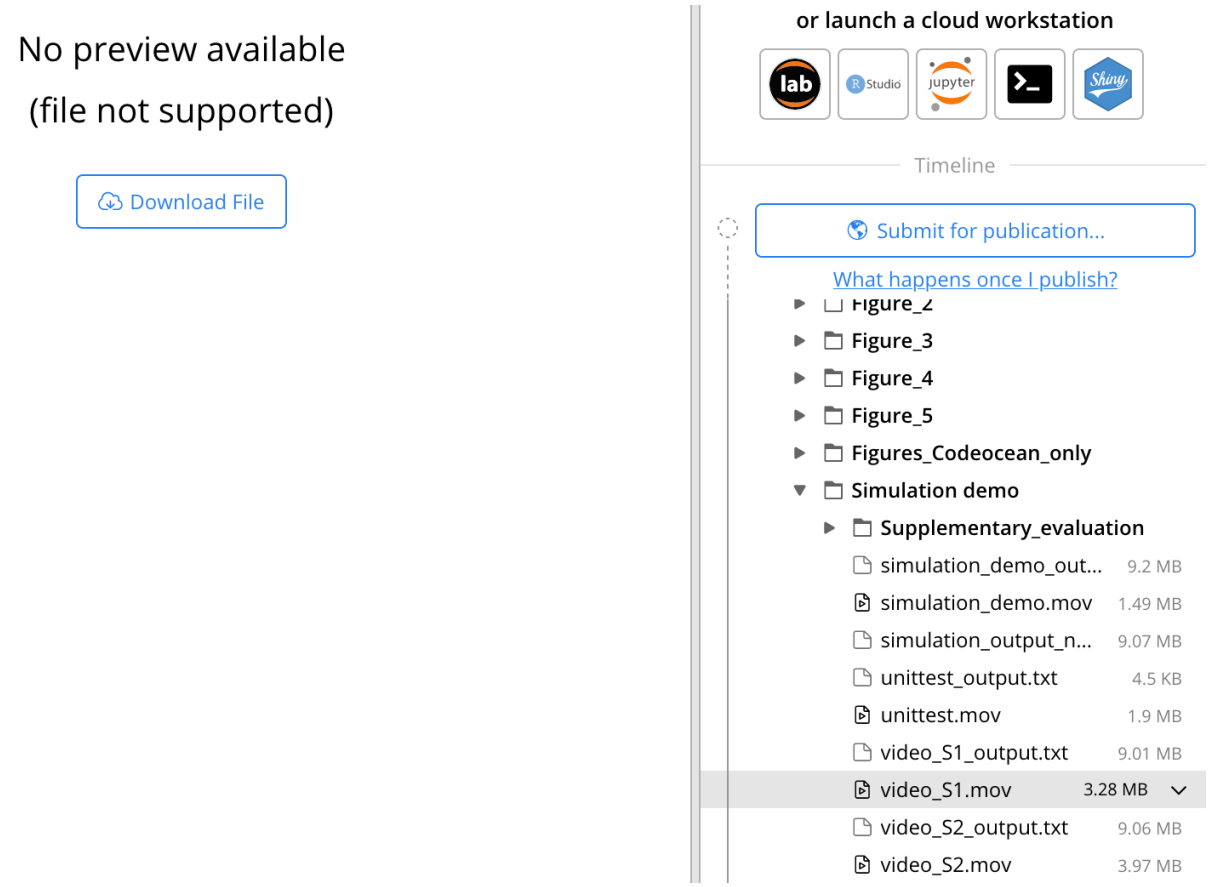
2.4. Demo

2.4.1. Run instructions



Screenshot 3. Starting a "Reproducible Run". This is common to all CodeOcean capsules: in the upper right corner of the web interface, a reproducible run can be launched by clicking the "Reproducible Run" button. Since reproducible runs add their output data to the capsule, it may be necessary to duplicate the published capsule first.

particleShear simulations are preconfigured in this CodeOcean capsule and are thus run with each "Reproducible Run" (run time about 5h, for shorter demos see below; this will also generate the other output such as the figures and supplementary figures associated with the manuscript). This generates time-lapse videos of the format of the supplemental videos S1-S4 to the associated manuscript[1], each time this CodeOcean capsule is run. They are available after completion of the reproducible runs, in the /results section (see Screenshot 3 for starting a reproducible run, and Screenshot 4 for the location of the demo output in the /results section). Of note, the particleShear simulations not only provide graphical output to illustrate the particle movements and interactions, but also quantitative output (simulation description, primary G' and G'' output values, as well as detailed stress tensor evaluation). The location of these quantitative text output files is shown in Screenshot 5. The structure of these output files is documented in the particleShear simulation manual (at "/code/Documentation/Simulation particleShear/Manual particleShear.pdf", Fig. S2-2, Tables S2-2 and S2-3).

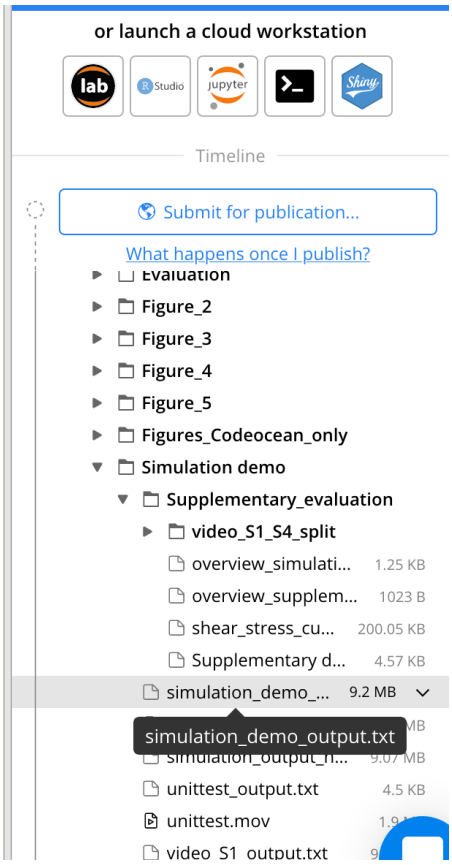


Screenshot 4. Output of the demo simulations. The output is re-generated at each run, and stored in the results area (internal path in CodeOcean "/results"). In the example, the output movie of the demo in the conditions of supplemental Video S1 (spherical particles) is shown. At present (2021), preview of movie files within the CodeOcean capsule is not supported, and so it is necessary to download the movies (.mov) to view them.

2.4.2. Expected output

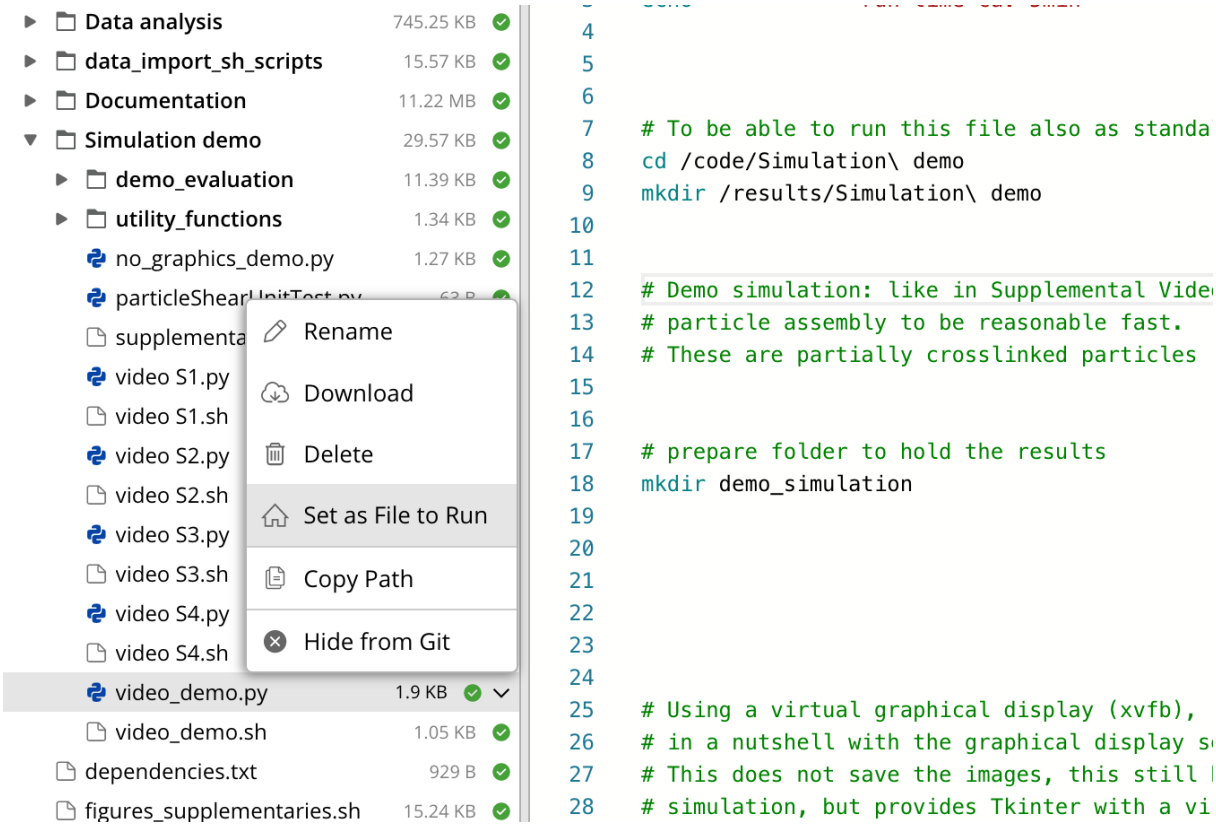
Full-scale demo simulations (with N=150 particles) are provided as supplemental videos S1-S4 to the associated publication[1]. These demos show the expected output in different configurations (Supplemental Video S1: non-crosslinked spheres; Supplemental Video S2: bulk from fully crosslinked spheres; Supplemental Video S3: Spheres crosslinked into irregular compact particles; Supplemental Video S4: Spheres crosslinked into irregular loose particles to emulate porosity). Each simulation also produces a quantitative text output file (Screenshot 5).

```
rmal filler Ensemble:
   Cutting lines used to generate the particles5
   N=20 spheres, nominal packing density = 1.5
   Young modulus of the individual spheres = 8000 Pa
   Friction coefficient = 0.01



nction call :
mulation_interlocking_rheology(root_folder=demo_simulation,
_permanent_links=True,cut_lines=5,N=20,packing_fraction=1.5,
modal_factor=1.4,amplitude=0.2,Young_modulus_spheres=8000,density=50,
=0.01,relative_viscosity=0.1,relative_y_scale_force=100000000.0,
lative_frequency=0.025,relative_transversal_link_strength=1,
ntral_repulsion_coefficient=1,avoid_horizontal_angle_degree=15,
oid_height_spanning_particles=True,interface_reenforcement_central=1,
terface_reenforcement_tangential=1,
ep_viscosity_coefficients_constant=False,cut_top_bottom=False,
CutByTriangulation=True,remove_link_fraction=0,edge_fuzziness=0,
Drawing=True)
   theSimulation.baseline_pre_periods = 2
   theSimulation.baseline_post_periods = 1
   theSimulation.saveOutputImages = True
   theSimulation.imageFileType = jpg
   theSimulation.runSimulation(periods=3, cool_factor=0.5)

cillatory shear experiment

del parameters
dth of simulation area size_x =500micrometers
ight of simulation area size_y =500micrometers
minal packing fraction = 1.5
ung modulus spheres = 8000Pa
```



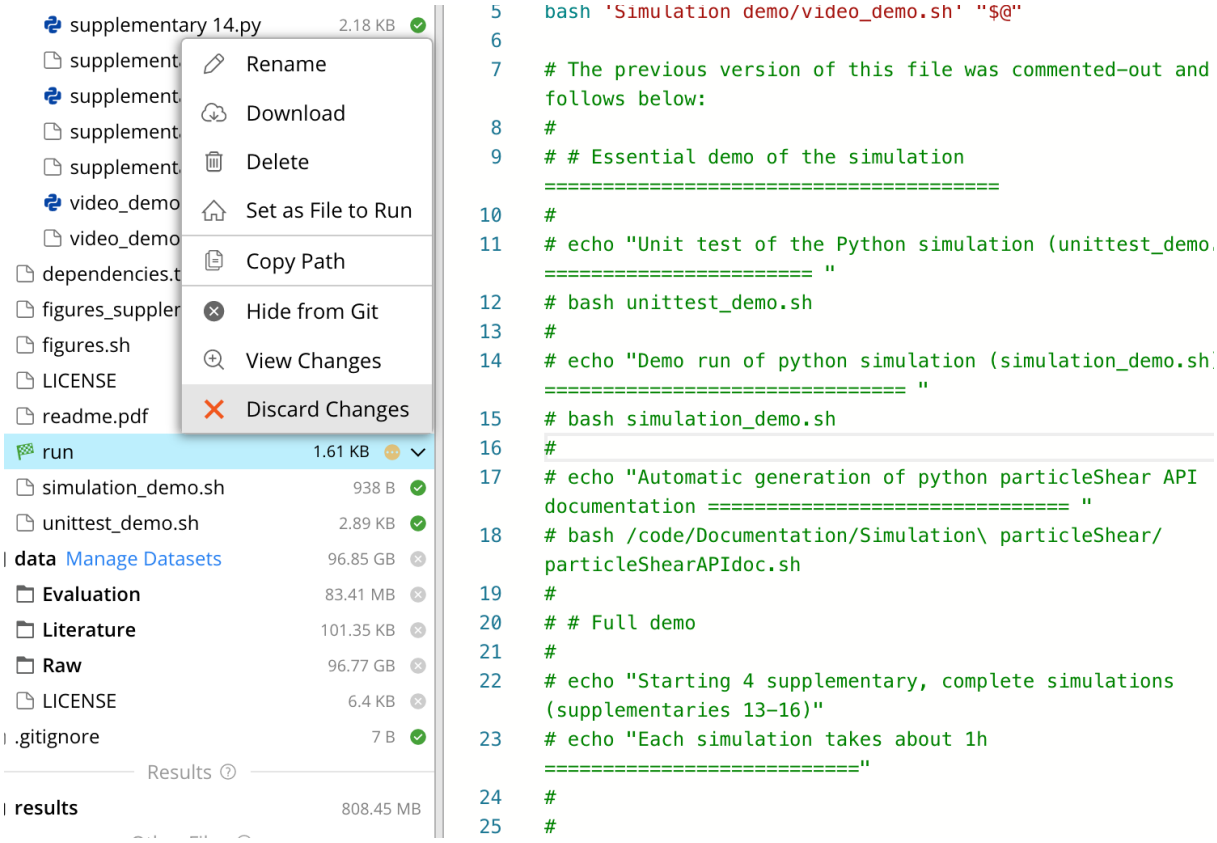
Screenshot 5. Textual output of the demo simulations. The text output is provided along with the visual output (Screenshot 4), each .mov file is associated with a .txt file. The content of the text files can be viewed in the CodeOcean viewer, although it tends to be rather lengthy and thus more appropriate for scripted analysis.

2.4.3. Short demo



Screenshot 6. Running a shorter demo. This is most easily obtained by setting the file "/code/Simulation demo/video_demo.sh" as the main file to run (Click on the V sign appearing by hovering over the file). The next "reproducible run" (see screenshot 3) will run specifically the video_demo.sh script, completing in about 10 minutes instead of 5h as for the complete run. Of note, setting the "video_demo.sh" script as the main file to run automatically modifies the main run file ("/code/run"), see screenshot 7 for resetting the capsule to its original state.

The particleShear simulations are rather calculation-intensive in the standard configuration used in the manuscript[1]; and so in addition to the "Reproducible run" with its present run time of about 5h, we provide also a quicker demo version. This can be run by setting the bash file "video_demo.sh" at "/code/Simulation demo" as the main run file (screenshot 6). As this modifies the file "/code/run", it may be advantageous to be able to undo these changes afterwards, they are provided along with screenshot 7.



Screenshot 7. Discarding changes due to using the video_demo.sh (i.e. screenshot 6). Running the short demo as a standalone has the side effect of modifying the main run file at /code/run. As shown in the screenshot, this leads to the main content being commented out and replaced by a line launching "video_demo.sh". To undo these changes, use the "Discard Changes" option as shown in the screenshot, appearing when clicking on the V sign next to the run file. Click "Confirm" in the ensuing dialog (not shown).

2.5. Instructions for use and reproduction of the results presented in the main manuscript

The particleShear simulation was used to generate the simulation data presented in the associated manuscript[1]. The actual simulations were run on a cluster (Baobab, University of Geneva), and represent about 30'000 CPU hours. Therefore, they cannot easily be reproduced on this CodeOcean capsule or on a present (2021) desktop station. This CodeOcean capsule however reproduces the complete data evaluation process flow for these data in each reproducible run.

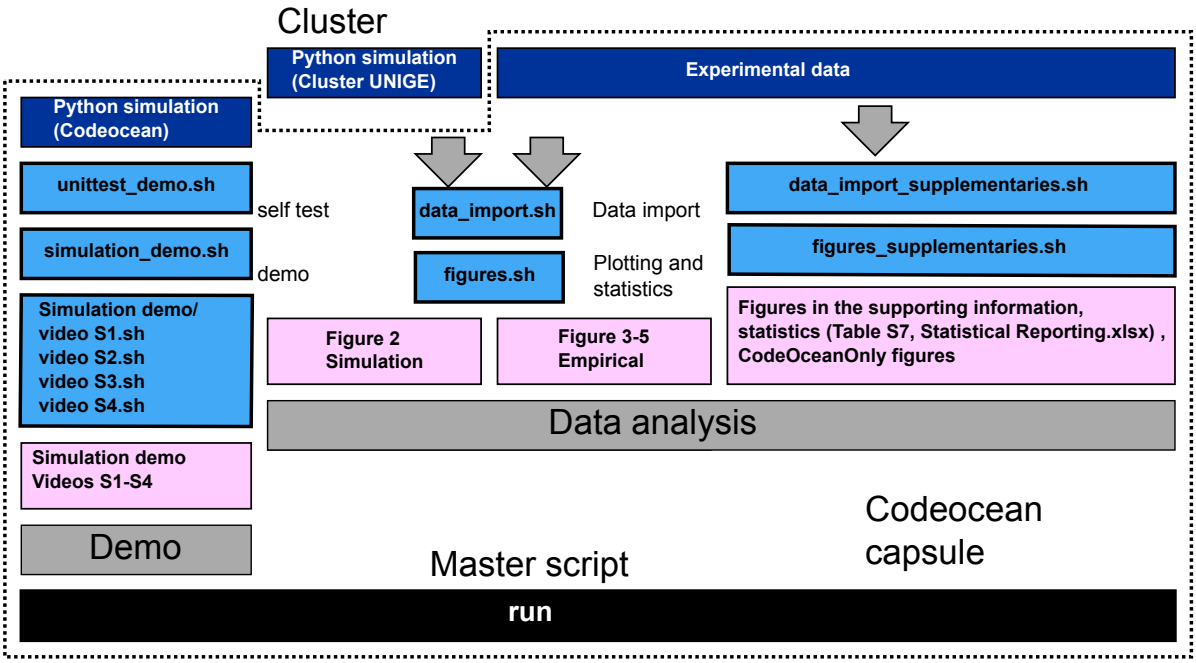
Detailed instructions for the reproduction of the simulations on an appropriate cluster are nevertheless provided in the manual available on this CodeOcean capsule (at "/code/Documentation/Simulation particleShear/Manual particleShear.pdf"). In particular, Table S2-4 provides the detailed instructions for complete re-running. At present, this is slightly outdated since we ran some more simulations in the mean time, the complete list of Python scripts run is at /data/Raw/Simulation/python_scripts_Baobab on this CodeOcean capsule.

3. Structure of the CodeOcean capsule

This CodeOcean capsule provides the simulation particleShear, in a reproducible runnable format along with shorter demos (see section 2 above).

It also provides the reproducible evaluation of all quantitative main and supplementary figures of the associated manuscript An Injectable Meta-biomaterial: From Design and Simulation to In-vivo Shaping and Tissue induction"[1] and also a set of CodeOcean only figures (see section 4.4).

Section 3 describes the overall structure of the CodeOcean capsule, with a focus on the CodeOcean-specific approach of completely reproducible runs.



Scheme 1. Content of this CodeOcean calculation capsule. The figure, supplemental video and table numbering refer to the main manuscript and its supplementaries,[1] with added CodeOceanOnly figure reported at /results/Figures_Codeocean_only. The file "Statistical Reporting.xlsx" is located at /code/Documentation/Statistical Reporting.xlsx and was generated manually by collecting the statistical output from the different R scripts in this CodeOcean capsule.

3.1. The run file

As for any other CodeOcean capsule, the main entry point for reproducible "re-runs" is the "run" file located in /code. Thus, the "run" file constitutes the master script of the computation capsule (Scheme 1).

The "run" file invokes various additional bash script files. For data import from various raw data file formats, it calls Data "data_import.sh", located in the "data_import_sh_scripts" folder, which in turn invokes various data import scripts. The run file also launches "unittest_demo.sh", "simulation_demo.sh" (c.f. section 2.4.3) , "figures.sh", and "figures_supplementaries.sh", which are all located at the same level as

"run" in /code. Further, the "run" file starts the scripts for the supplementary videos 3-6 in the folder "Simulation demo". Details on these files are given below. The purpose of these scripts is to provide a demo of the Python simulation on the one hand, and to reproduce all main figures and supplementary figures of the paper "An injectable meta-biomaterial", as outlined in scheme 1. Additionally, a set of CodeOcean only figures is generated; these CodeOcean only figures are part of the additiona technical documentation provided at /code/Documentation/CodeOceanOnlyResults.

All further bash scripts invoked by the "run" file are designed as standalone files and can be run individually, for example by commenting out the others in the run file.

3.2. Demo simulations

For practical purposes, it is not possible to re-simulate the integrality of the Python simulations underpinning this CodeOcean capsule, this amounts to about 30'000 CPU hours.

To nevertheless provide an environment where small-scale simulations can be run and evaluated, we provide various demonstrations with visual (video) output.

A first set of small-scale demos is run by "simulation_demo.sh", illustrating the basic use of the particleShear python package (c.f. section 2.4.3). The "unittest_demo.sh" invokes a small series of more specific simulations, designed as self-test for the Python package.

In addition, the "run" file also contains longer simulations, corresponding to the videos provided as supplemental videos S1-S4 to the associated manuscript[1]. These longer simulations replicate the format of the simulations run at the Baobab cluster at University of Geneva, Switzerland, to generate the simulation data analyzed in this capsule. For demonstration purposes and to provide re-usable code, we also carry out evaluation of these four simulations up to the constitution of overview files analogous to the ones obtained from the cluster (c.f. section 2.4.2).

Screenshots 4 and 5 indicate the location of the output files produced by these simulation demos.

3.3. Data analysis

In addition to demonstration of the Python simulation, the capsule also provides reproducible evaluation of the raw data, including plotting of the quantitative sub-figures. All quantitative data in the main text of the associated paper[1] is covered by the scripts "data_import.sh" and "figures.sh" (see scheme 1). The scripts "data_import_supplementaries.sh" and "figures_supplementaries.sh" cover data import, figure plotting for the supporting information and CodeOcean only figure plotting.

3.4. Main scripts of the CodeOcean capsule

Scheme 1 shows a more detailed view on the role of the main scripts in the CodeOcean capsule. The global "run" script is the main entry point and invokes all other scripts in a reproducible run.

As shown in scheme 1, this CodeOcean calculation capsule pursues two related goals:

1) Demo: This part provides a small demo of the Python simulation "particleShear" underpinning the theoretical part of the manuscript[1], along with self-testing by predefined unit tests, and production of the demo videos for supplemental videos S1-S4. See section 2.

2) Data analysis and plotting: This part provides the statistical analysis and plotting of the raw data, both originating from the Python simulation run on an external cluster (Baobab at the University of Geneva, Switzerland) and various physical measurements. The data arising from the physical experiments is fully treated in the calculation capsule . The simulations were performed on an external cluster to handle the heavy calculation workload, but the data treatment of the simulation output files is provided here in a reproducible manner (screenshot 5). Along with the figures, the CodeOcean capsule also provides all relevant statistical evaluation in an automated fashion.

Eight main bash scripts, all invoked by the main script "run", are the main work horses of this CodeOcean calculation capsule (scheme 1):

3.4.1. Short simulation demo

unittest_demo.sh: Bash script launches elementary test cases for the particleShear Python simulation. These test cases as well as their recorded output are described in the particleShear manual (see section 4.1.2). Completes in a few minutes.

simulation_demo.sh: Bash script launching a minimalistic simulation (with only 20 microspheres) for illustration. Completes in about 20 minutes. See section 2.4.3.

3.4.2. Full simulation demo

"video S1.sh", "video S2.sh", "video S3.sh", "video S4.sh": Located in folder /Simulation demo, these scripts launch complete simulations corresponding to the video supplemental videos S1-S4. These scripts take each about 1h to complete, and produce output as shown in screenshots 4 and 5.

In addition to providing video examples of the simulation, the text file output of the scripts "video S1.sh" to "video S4.sh" also serves to demonstrate the initial evaluation of the simulation output files. This is done by the script "supplementary_demo_evaluation.sh", located in the folder "/code/Simulation demo". This evaluation can only be run together and immediately following "video S1.sh" to "video S4.sh". Therefore, contrary to the other main .sh scripts described here, "supplementary_demo_evaluation.sh" is not designed as a standalone script.

3.4.3. Data import

data_import.sh, data_import_supplementaries.sh: These two script import various raw data file formats into R Data files (.rda) that can then be used to plot the figures in the main text and supplementaries. For this, various R scripts are invoked by "data_import.sh" and "data_import_supplementaries.sh". See also the file "/code/Documentation/Data analysis file locations.xlsx" (and also section 4.3 and section 5 in this document) for details on the various intermediate .rda files and the R scripts used to produce them.

3.4.4. Plotting and statistical analysis

figures.sh, figures_supplementaries.sh: These scripts plot the figures and evaluate the associated statistics. Along with the plotting, all relevant statistical parameters and tests are evaluated and reported, generally in dedicated text files produced along with the figures. In addition to the detailed statistical results reported in the text files along with the figure output, they are also gathered in an aggregated manner in Table S7 in the supporting information of the associated manuscript[1], and in deeper detail at /code/Documentation/Statistical Reporting.xlsx.

3.5. Details on the python simulation in a capsule without graphical display

Running of a demo of a fundamentally visual simulation is slightly complicated in an environment without a graphical display.

Three options are demonstrated in the calculation capsule (Table 1):
1. Running the simulation without graphical display by passing suitable arguments to the high-level function doParticleShearSimulation;
2. Using the virtual frame buffer xvfb to provide a virtual graphical display, along with image file saving from within the simulation; and
3. Running parallel xvfb and simulation processes with periodical screenshot taking by the main script.

Option	Details	Example scripts
1) No graphics	Use doDrawing=False and plotStress=False as arguments to doParticleShearSimulation (see Manual of particleShear package at "/code/Documentation/Simulation particleShear/Manual particleShear.pdf", part 2)	/code/simulation_demo.sh , first part, along with /code/Simulation demo/no_graphics_demo.py
2) xvfb to buffer graphics, **internal saving** from python simulation	Use the utility xvfb-run to supply a local graphical environment	/code/Simulation demo/video_demo.sh along with /code/Simulation demo/video_demo.py Supplemental videos S1-S4.
3) xvfb to buffer graphics, external **screenshots**	xvfb and the python simulation are launched as separate, connected processes (here, using DISPLAY=:1). Periodically, the main script takes screenshots (import function from imagemagick)	/code/unittest_demo.sh along with /code/Simulation demo/particleShearUnitTest.py

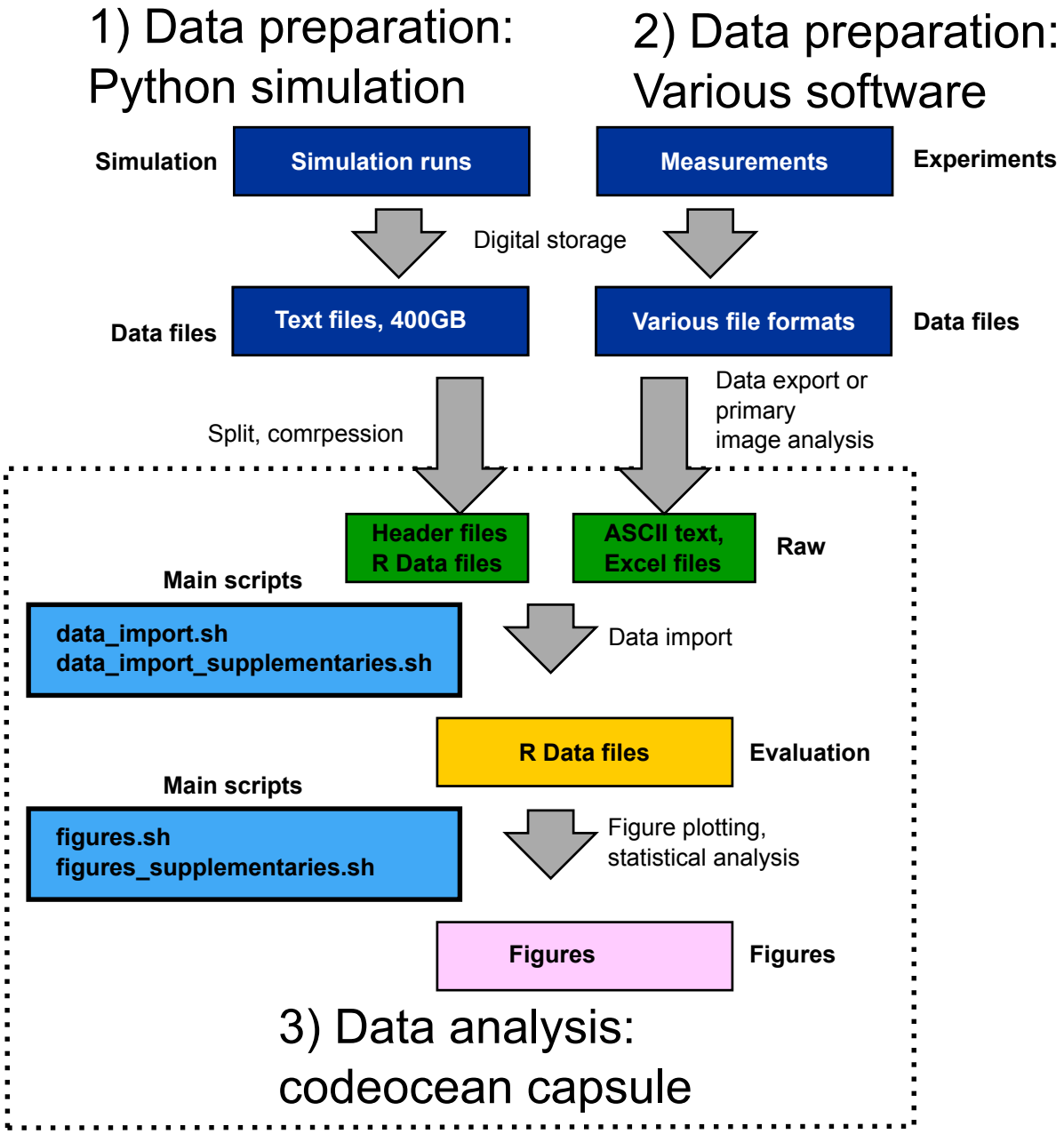
Table 1. Options to run the Python simulation in a "headless" environment (without physical graphical display).

On a technical level, we use the virtual graphical display Xvfb for providing the simulations with a simulated screen display (thus enabling Python's Tkinter), and imagemagick + ffmpeg for handling screenshots and output images, and finally provide them as videos (.mov files, see for example screenshot 4).

In a local installation with a graphical display attached to the computer, it is easier to directly follow the quick install instructions (located at /code/Documentation/Simulation particleShear/Quick install.pdf, see section 4.1.1) to obtain graphical output instead.

3.6. Details on data analysis: import from raw files, figure plotting and statistics

3.6.1. Data analysis process flow



Scheme 2. The data evaluation for the manuscript "An injectable meta-biomaterial" comprises two major steps: The first step is data preparation (data acquisition, and export to text or Excel files in proprietary software), the second is data analysis of these machine-readable files. This CodeOcean capsule provides the possibility to reproduce the data analysis part in a controlled environment

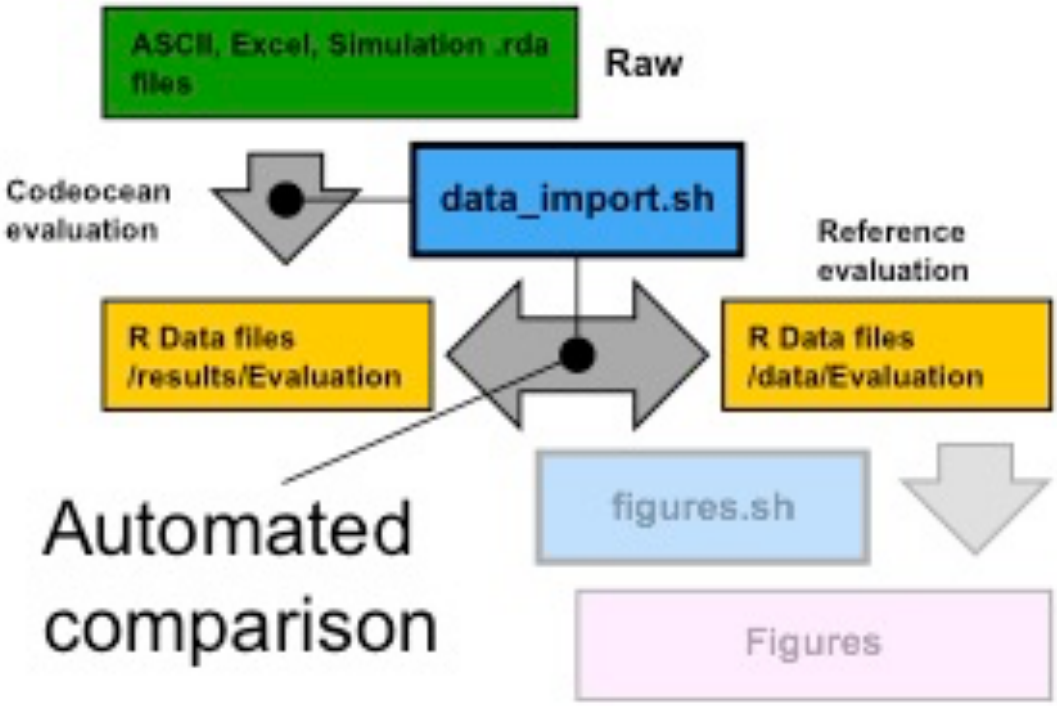
The CodeOcean capsule also includes the data treatment from primary data to figure plotting. The process is shown in Scheme 2. The CodeOcean capsule implements most, although not all of the data treatment. Upstream of the CodeOcean capsule are two elements: for the simulation, acquisition of the simulation raw data on a cluster (Baobab, University of Geneva), with conversion to r-data (.rda) and header files (.txt) to save storage space. For the physical measurement, the primary raw data are sometimes in proprietary formats (rheology, uniaxial compression), from which we extracted ASCII

text files or Excel files for data treatment. Likewise, image treatment of confocal images for porosity and particle shape analysis was performed outside the CodeOcean capsule, generating Excel files with the results uploaded into the raw data section of this CodeOcean capsule.

The remaining data treatment is reproducibly implemented into the CodeOcean capsule. It takes place in two consecutive steps, implemented generally by two consecutive scripts (Scheme 2):

1) Importation of raw data into R data files. This is done by the bash script **data_import.sh** (respectively **data_import_supplementaries.sh** for data used solely in supplementaries of the manuscript "An injectable meta-biomaterial"), which invokes a number of R scripts for reading the various text and Excel files. A detailed list of R data (.rda) files with the R scripts used to generate them can be found in /code/Documentation/Data analysis file locations.xlsx.
2) Analysis of the data and figure plotting. This is done by the bash script **figures.sh**, and for the supplementary figures **figures_supplementaries.sh**. These scripts invoke the R scripts for plotting of the individual sub-figures; there is one R script per subfigure.

3.6.2. Reproducibility of the data evaluation



Scheme 3. Comparison of the CodeOcean evaluation to the reference evaluation.

In the CodeOcean capsule, we evaluate the raw data by using "data_import.sh". This yields a series of .rda files (see /code/Documentation/Data analysis file locations.xlsx for details on these). To ensure matching with the R data files used for the associated manuscript[1], we set up and run automated comparison between the reference evaluation (uploaded to the /data/Evaluation section of the capsule) and the CodeOcean

evaluation (generated at each run of data_import.sh in the "Evaluation" folder of the output section, noted here as "/results/Evaluation"). This is done at the end of the data_import.sh script, by invoking the R script "/code/Data analysis/Data import/compare_to_local_evaluation.R".

The comparison between reference evaluation and CodeOcean evaluation is done at the level of the .rda files by one-by-one comparison, as implemented in the custom R package "reproducibleCalculationTools".[7] For each of the rda files, we assess whether it contains the same variables and whether values that should correspond do correspond. The corresponding entries should indeed be identical, with a few exceptions such as local paths. In cases involving sophisticated numerical evaluation such as least squares fitting, we find near exact matches (precision better than 10^{-6} for more involved algorithm such as least squares fitting, and typically better than 10^{-10} for simple data input), probably reflecting slightly different implementations of similar algorithms on different platforms. The comparison is reported in the results, as "validation_evaluation.txt" in the output section.

Providing the reference evaluation as a permanent set of .rda files in the /data section also has the advantage that the analysis and figure plotting scripts "figures.sh" and "figures_supplementaries.sh" can be run independently of the "data_import.sh" script, allowing substantial gain of time if desired (this can be done by setting them as main run files, analogously to screenshot 6). The automated comparison between prior and current evaluation of the data also provides a sensitive means to detect inadvertent changes arising through updating of internal or external libraries and scripts.

4. Documentation

In addition to this readme file, this CodeOcean capsule contains detailed documentation on both the Python simulation particleShear and custom R libraries used.

4.1. Python simulation

Extensive documentation for the Python simulation is available.

4.1.1. Quick install guide

A very brief quick install guide is provided at "/code/Documentation/Simulation particleShear/**Quick install.pdf**". The aim of this document is to provide for rapid instructions for a local installation and usage of the Python simulations.

4.1.2. Manual

The manual "Manual particleShear.pdf", located at "/code/Documentation/Simulation particleShear/Manual particleShear.pdf" provides extensive documentation on:

- The **mathematical framework** of the simulation (Part I in Manual particleShear.pdf; this namely contains developments over [8] and [9])
- **Usage instructions**, including local installation (Part II)
- **Implementation details**, including sub-modules, class hierarchy and pseudo-code for a typical simulation run (Part III)
- Description of the **unit tests** performed as part of each run of this CodeOcean capsule (Part IV)

4.1.3. API documentation

An automatically generated API documentation of the particleShear Python simulations is available at "/code/Documentation/Simulation particleShear/particleShear API Documentation/index.html". This file is best used when looking for details on a particle function or class in particleShear. Also, with limited javascript functionality within the CodeOcean preview, it is better to download the folder and view it locally.

4.2. Custom R libraries

This R capsule uses a series of custom R functions. These functions are organized in custom R libraries. The libraries themselves are hosted at github for incremental development, with an archive copy generated at each release on Zenodo. The custom libraries used here are:

- **particleShearEvaluation**[5]: Utility functions to read and analyze output from the Python package "particleShear". The latest source version of this library is publicly available at https://github.com/tbgitoo/particleShearEvaluation; the release used here is archived at Zenodo (https://doi.org/10.5281/zenodo.4594649).
- **plot.counts**[6]: A collection of plotting functions, particularly for plotting both aggregated (average and standard deviation) and individual data. The latest source of this library is available at https://github.com/tbgitoo/plot.counts; the release used here is archived at Zenodo (https://doi.org/10.5281/zenodo.4589498)
- **rheologyEvaluation**[4]: Utility functions to read and analyze rheology data (from Rheowin, exported as Ascii data). The latest source of this library is available at https://github.com/tbgitoo/rheologyEvaluation; the release used here is archived at Zenodo (https://doi.org/10.5281/zenodo.4594353).
- **textureAnalyzerGels**[3]: Utility function to read and analyze output from the textureAnalyzerXT machine (after conversion to ASCII files). The latest source of this library is available at https://github.com/tbgitoo/textureAnalyzerGels. The release used here is archived at Zenodo (https://doi.org/10.5281/zenodo.4589276).
- **reproducibleCalculationTools**[7]: Comparison of successive evaluations in R with definition of numerical tolerance. The latest source of this library is available at https://github.com/tbgitoo/reproducibleCalculationTools. The release used here is archived at https://doi.org/10.5281/zenodo.4594515.

In the CodeOcean capsule, these libraries are automatically installed from the Github repositories. Locally, this installation carried out by the commands:

```
library(devtools)
install_github("tbgitoo/plot.counts")
```

and analogously for the other R packages. This may require installation of devtools, via the usual R package installer.

4.3. R-Data files

A detailed technical overview over the R-data files (.rda files) of this CodeOcean capsule can be found at "/code/Documentation/ Data analysis file locations.xlsx", including the scripts used to produce them. A global overview is given below (section 5).

4.4. CodeOceanOnly figures

In addition to the figures reported in the associated manuscript[1] and its supplementaries, there are a number of figure evaluations reported only in this CodeOcean capsule. They are reported in the folder "Figures_Codeocean_only" in the Results section. This folder also comprises an extensive description of these figures: CodeOceanOnlyResults.pdf (copied from /code/Documentation/CodeOceanOnlyResults during capsule evaluation).

4.5. Statistical reporting

Most R scripts used for the production of main figures, supplementary figures or CodeOceanOnly figures report their statistical output in text files produced along with the graphical files in the corresponding section in the /results. From these text files, we compiled an in-depth overview document, stored at /code/Documentation/Statistical Reporting.xlsx. In the supporting information of the main manuscript,[1] table S7 summarizes the most important points of this information.

5. Datasets in this CodeOcean Capsule

In addition to the automated and reproducible evaluation, this CodeOcean capsule also contains some datasets that may have an interest on their own. As a direct result of result of the two-step evaluation with intermediate data files(Scheme 2), the datasets are available in the R-loadable .rda format. A full overview along with the relations to the scripts and figures is provided at "/code/Documentation/ Data analysis file locations.xlsx"

Regrouped by area of interest, the main datasets are summarized in Table 2, below.

Area of interest	Datasets (.rda)	Path in /data/Evaluation	Content
In vivo	=>in_vivo_data.rda =>in_vivo_colonization_vascularization.rda =>in_vivo_inflammation_encapsulation.rda =>in_vivo_degradation.rda	/In vivo	-Shape data -Implant colonization -Peri-implant inflammation - Degradation kinetics in-vivo
Physical properties	=> various rheology datasets (EPI / Juvéderm / Sephacryl S200 / Cultisphere / synthesized control materials)	/Rheology, various sub folders /Ejection Force	- Stress sweeps - Stress recovery - Self healing - Injectability (large delivery cannula, Thiebaud

	=> ejection_force_data.rda, ejection_force_data_aggregated.rda => various ejection force datasets in "from_additional_experiments" => stress_sweeps_comparison.rda => stress_sweeps_adipose_tissue_mouse.rda	/Ejection Force/ from_additional_experiments /For Supplementaries/ Supplementary 9 Material comparison	F9020100) - Datasets acquired during manuscript review, partially used - CMC control materials (non-porous irregular and spherical) - Rheology subcutaneous and visceral fat mouse
Geometry	=> various pore size datasets in "Porosity" => particle_size_overview_EPI.rda => Particle shape: particle_shape.rda	/Porosity /Particle Geometry /Particle Geometry/Particle Shape	- Pore size as a function of polymer concentration - EPI particle size - Particle shape, EPI and EPI variants, Juvéderm Voluma, Sephacryl S200, Cytodex, spherical and irregular control materials. Cultisphere S from literature[10].
Simulation	=>General simulations: overview_simulation_jumps_corrected.rda => Modified force law: plateau_contact_law_fig_5e.rda	/Simulation/Overview_data_files /Simulation/ plateau_law	- Overview results simulation (Baobab Cluster UNIGE) - Specific adaption of the contact force law to match experimental EPI data

Table 2. Main datasets in this CodeOcean capsule

6. Bibliography

[1] A. Beduer, F. Bonini, C. Verheyen, M. Genta, M. Martins, J. Brefie-Guth, J. Tratwal, A. Filippova, P. Burch, O. Naveiras, T. Braschler, Adv Mater 2021, DOI: 10.1002/adma.202102350.
[2] T. Braschler, particleShear: Discrete Python particle simulation with digital rheology and stress tensor evaluation (Version v1.0.2), Zenodo, https://doi.org/10.5281/zenodo.4589212, 2021.
[3] P. Burch, M. Braschler, T. Braschler, textureAnalyzerGels: R package for importing and analyzing hydrogel compression data (Version v1.0), Zenodo, https://doi.org/10.5281/zenodo.4589276, 2001.
[4] T. Braschler, rheologyEvaluation: R package to read and analyze rheowin text export files v1.1 (Version v1.1), Zenodo, https://doi.org/10.5281/zenodo.4594353, 2021.
[5] T. Braschler, particleShearEvaluation: Import and analysis of output files generated by the discrete particle simulation Python module particleShear (Version v1.0), Zenodo, https://doi.org/10.5281/zenodo.4594649, 2021.
[6] T. Braschler, plot.counts: R package with convenience functions for plotting count data (v1.0), Zenodo, https://doi.org/10.5281/zenodo.4589498 , 2021.
[7] T. Braschler, reproducibleCalculationTools: R package for comparing numeric output from successive evaluations, Zenodo, https://doi.org/10.5281/zenodo.4594515 , 2021.
[8] M. Otsuki, H. Hayakawa, Phys Rev E 2017, 95, 062902.
[9] F. Nicot, N. Hadda, M. Guessasma, J. Fortin, O. Millet, Int J Solids Struct 2013, 50, 2508.
[10] S. de Bournonville, L. Geris, G. Kerckhofs, Sci Rep 2021, 11, 2819; R. Alfred, J. T. Taiani, R. J. Krawetz, A. Yamashita, D. E. Rancourt, M. S. Kallos, Biomaterials 2011, 32, 6006; C. E. Nweke, J. P. Stegemann, J Mater Chem B 2020, 8, 3972; A. A. Akasha, "Attachment of Embryonic Stem Cells-derived Cardiomyocytes in Cultispher-S Microcarriers by using Spinner flask", 2012.