First VPLanet Developers Workshop

Lesson 3
Support Scripts

Overview

We have developed 4 Python scripts to assist in VPLanet simulations and analysis

vplot: Plot results from a single simulation with one command, and use with matplotlib for publication-worthy figures (*Rodrigo Luger*)

vspace: Generate initial conditions for a parameter sweep (*Russell Deitrick*)

multi-planet: Run the parameter sweep across multiple cores (*Caitlyn Wilhelm*)

bigplanet: Compress parameter sweep output for fast analyses (*Caitlyn Wilhelm* and *David Fleming*)

VPLOT

vplot exists in a separate repository from VPLanet, but is installed automatically via either pip or setup.py

vplot's repo contains examples and documentation (in progress)

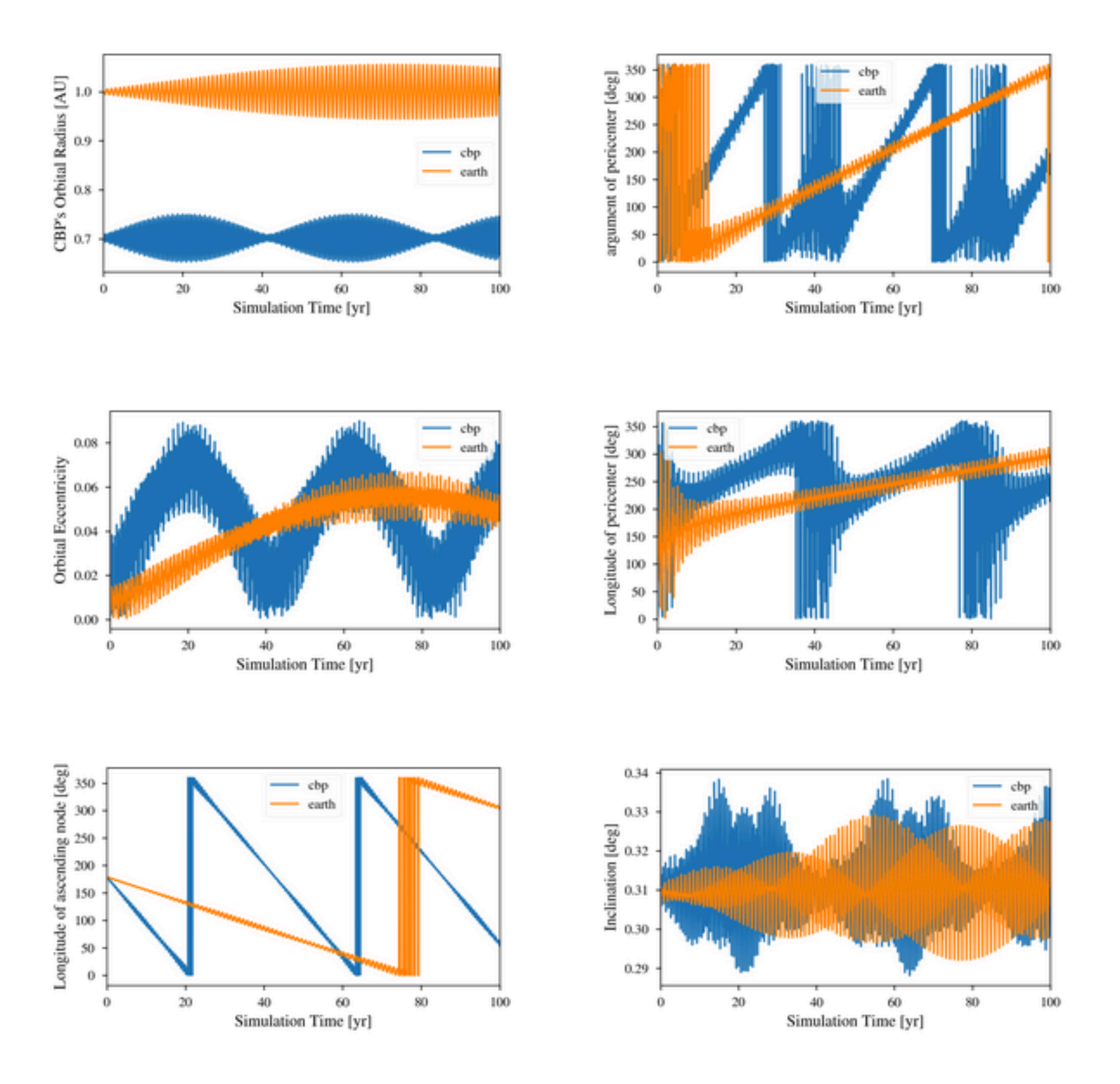
vplot also contains functionality for reading in infiles and forward files

Two ways to use: on the command line, or import into a script

VPLOT

For a quick visual inspection of a VPLanet run, just type > vplot

For example, after running examples/CircumbinaryOrbit:



vplot reads in the .forward files

Groups outputs

Plot each body w/unique color

VPLOT

Importing vplot and vplanet in Python makes plotting a breeze

You can read in and parse VPLanet output:
    output = vplanet.get_output()

Otherwise, just call VPLanet from inside your script:
    `output = vplanet.run(‘vpl.in’)`

output now contains all the log and forward data, which you can access:
    forward file data: `output.earth.Eccentricity` (array)
    log file data: `output.log.initial.earth.Eccentricity` (scalar)

With data read in, you can plot with matplotlib, but note vplot overrides matplotlib.figure.Figure to add axis and legend labels

All examples in the VPLanet repo use vplot

VPLOT

vplot also includes a set of colors chosen for accessibility:

🟪	vplot.colors.purple
🟦	vplot.colors.dark_blue
🟧	vplot.colors.orange
🟥	vplot.colors.red
🟦	vplot.colors.pale_blue

You can of course still use other colors, but these plus black represent about as many curves as you want to put in a figure

VSPACE

A common use case for VPLanet is to perform Monte Carlo simulations over a multi-dimensional parameter space

vspace is a simple code to generate the initial conditions

vspace exists in a separate repository from VPLanet, but is installed automatically via either pip or setup.py

vspace's repo contains examples and documentation

VSPACE

To generate the .in files, follow these steps:

1) Build and run a single case
2) Identify the options you want to vary
3) Write the `vspace.in' file
4) Run! > vspace vspace.in

vspace allows for multiple methods for sampling distributions:

- Grid mode, i.e. fixed interval between simulations
- Random mode, i.e. a Monte Carlo simulation
    - Uniform
    - Gaussian
    - Sine
    - Cosine

VSPACE

A typical vspace.in file looks like this:

```
srcfolder ~/vplanet/examples/EarthInterior
destfolder ParameterSweep
trialname test_

file   sun.in

file   earth.in

dTCore [5500, 6500, n10] tcore
d40KPowerCore [-1.5, -0.5, n10] K

file   vpl.in
```

This file tells vspace to look in the srcfolder for the template files, create a new directory called ParameterSweep with individual simulations in subdirs called test_*

In earth.in, add lines for dTCore and d40KPowerCore as a grid (n) and use “tcore” and “K” as prefixes, e.g. test_tcore01K08

MULTI-PLANET

After building your files with vspace, you run them with multi-planet:

```
> multi-planet [-c <num_cores>][-bp][-m <address>] <file>
```

This will run the simulations on num_cores CPUs

- note that multi-planet does *not* run across multiple nodes
- if you do not include -c, multi-planet will use *all* available cores

The -bp option will create the bigplanet file automatically (next slides)

The -m option send an e-mail to <address> when the jobs are finished

If your run stops for whatever reason, just run the command again and multi-planet will pick up where it stopped

You can also check progress with the command: mpstatus <file>

BIGPLANET

After running a large parameter sweep, you can create a lot of data

These data can be difficult/time-consuming to analyze and plot

bigplanet compresses vspace/multi-planet results into a single HDF5 file, with extension .bpl

This file contains all the options, log, and output data from your parameter sweep.

Building the bpl file can take time (~10 minutes for 1000 simulations) but uploading and extracting data is ~10x faster than working with files and folders

BIGPLANET

To run bigplanet, use the command:

```
> bigplanet [-c num_cores][-q][-m <address>] <file>
```

where -c is again the number of cores to use (default = all)
The -q option suppresses all output
And the -m flag send an e-mail to <address> when completed

If bigplanet is halted for any reason, it cannot restart from a checkpoint like multi-planet

bigplanet essentially creates a file in which each row corresponds to a single simulation, with columns corresponding to options, outputs, and other quantities

BIGPLANET

bigplanet can also be imported to a Python script, where additional functionality can make analyzing/plotting much easier

Most of bigplanet's functionality lies in importing it as a module into a Python script: import bigplanet as bp

bigplanet stores data as `key-value' pairs, i.e. a key (string) points to a value

Keys in bigplanet have the format: body_variable_aggregation

- body is the name (sName) of the object
- variable is name of an option or output
- aggregation is a description of the value
- e.g. earth_Eccentricity_initial

BIGPLANET

The following aggregations are available:

- initial
- final
- OutputOrder (list of names and units of the forward file)
- forward (nest list of forward file values)
- GridOutputOrder (list of names and units for climate sims)
- climate (list of climate values)
- option (value from a .in file)

Forward file variables also have statistical aggregations:

- min
- max
- mean
- geomean (geometric mean)
- stddev (standard deviation)

BIGPLANET

When using bigplanet in a script, the following functions are available:

“BPLFile”: open a bigplanet file for reading:

```
bpfile = bp.BPLFile(‘file.bpl’)
```

“ExtractColumn”: extract a single column from a bp file:

```
ecc = bp.ExtractColumn(bpfile,’earth_Eccentricity_initial’)
```

“ExtractUniqueValues”: create an array of unique values, which is useful for 2D plots

```
T_surf = bp.ExtractUniqueValues(bpfile,’earth_SurfaceTemp_final’)
```

CreateMatrix: Transform data into a 2D matrix for plotting:

```
z = CreateMatrix(xaxis,yaxis,zarray,orientation=1)
```

BIGPLANET

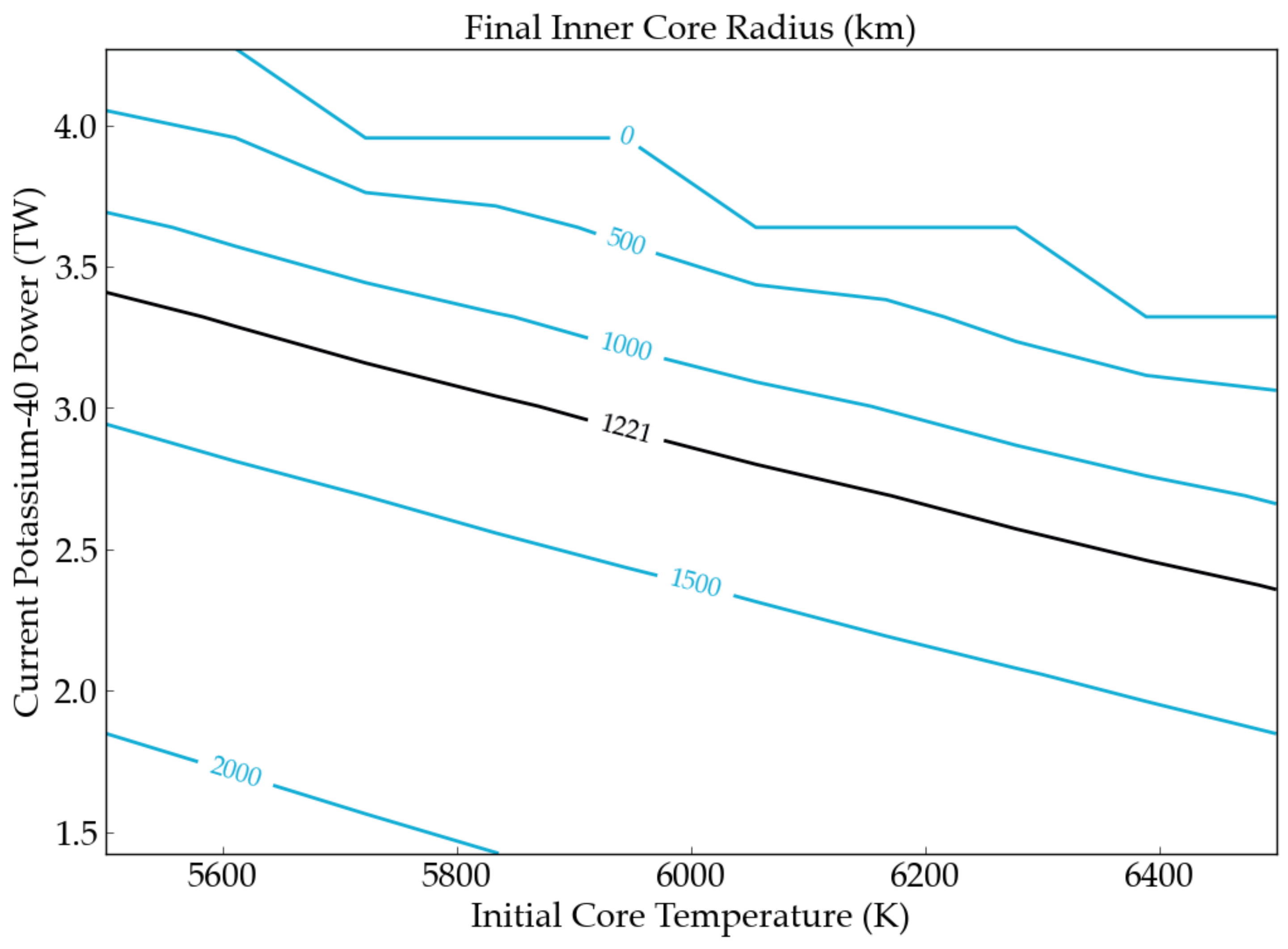
The “Parameter Sweep Guide” example shows how vspace, multi-planet, and bigplanet work together

The example computes Earth’s inner core radius for different core temperatures and potassium-40 abundances

```
data = bp.BPLFile(“ParameterSweep.bpl”)

RIC = bp.ExtractColumn(data, "earth_RIC_final")
RIC_units = bp.ExtractUnits(data, “earth_RIC_final")

TCore_uniq = bp.ExtractUniqueValues(data, "earth_TCore_initial")
TCore_units = bp.ExtractUnits(data, "earth_TCore_initial")

K40_uniq = bp.ExtractUniqueValues(data, "earth_40KPowerCore_final")
K40_units = bp.ExtractUnits(data, "earth_40KPowerCore_final")

RIC_Matrix = bp.CreateMatrix(TCore_uniq, K40_uniq, RIC)

plt.contour(TCore_uniq, K40_uniq, RIC_Matrix)
```

BIGPLANET

The "Parameter Sweep Guide" example shows how vspace, multi-planet, and bigplanet work together



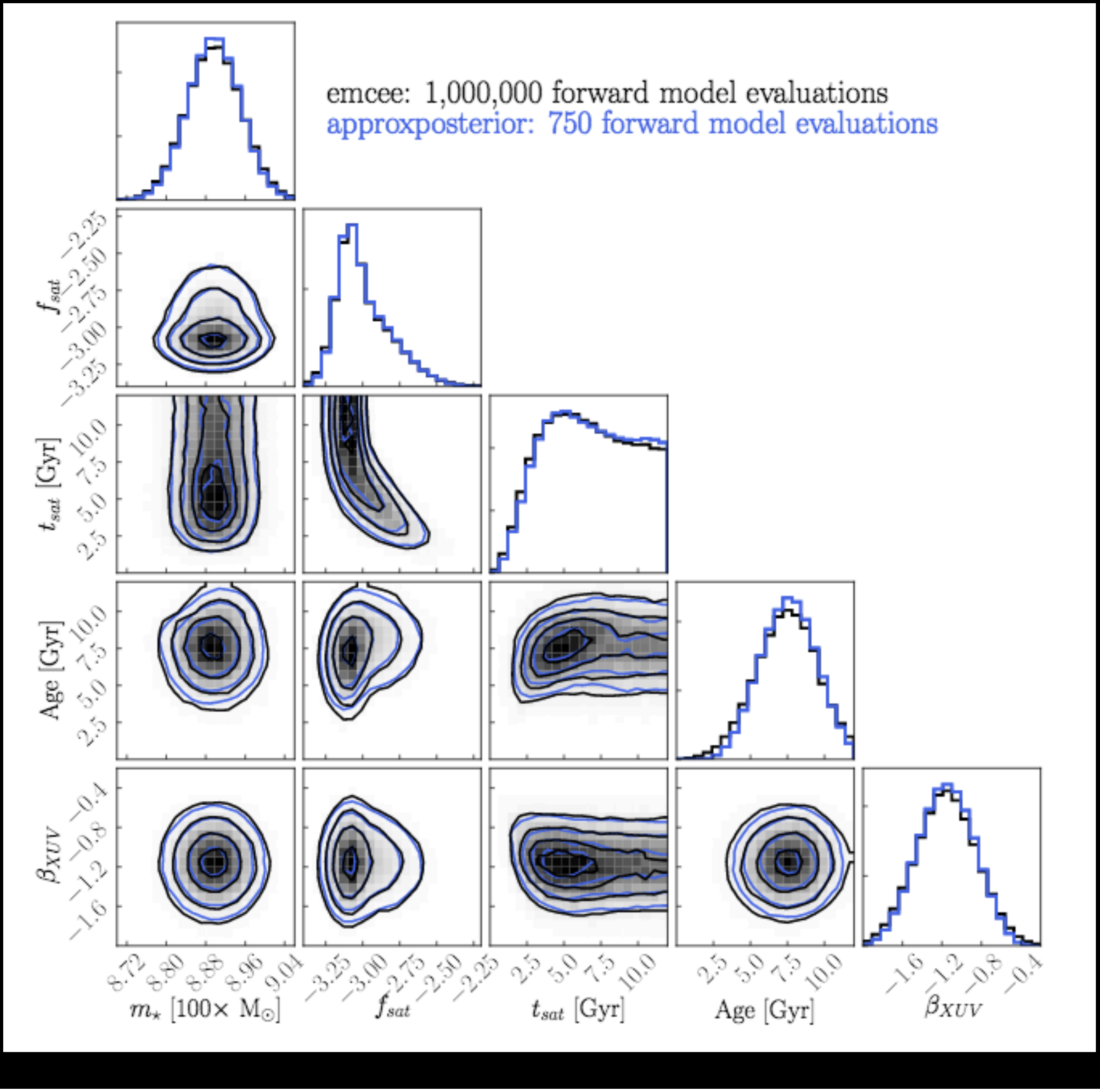
The exa nt core

temper

```
data = b

RIC = bp
RIC_uni

TCore_u l")
TCore_u

K40_un e_final")
K40_un )

RIC_Matrix = bp.CreateMatrix(TCore_uniq, K40_uniq, RIC)

plt.contour(TCore_uniq, K40_uniq, RIC_Matrix)
```

Additional and Planned Resources

approxposterior: Derive posteriors via machine learning up to 1000x faster than emcee



Stellar parameters for TRAPPIST-1

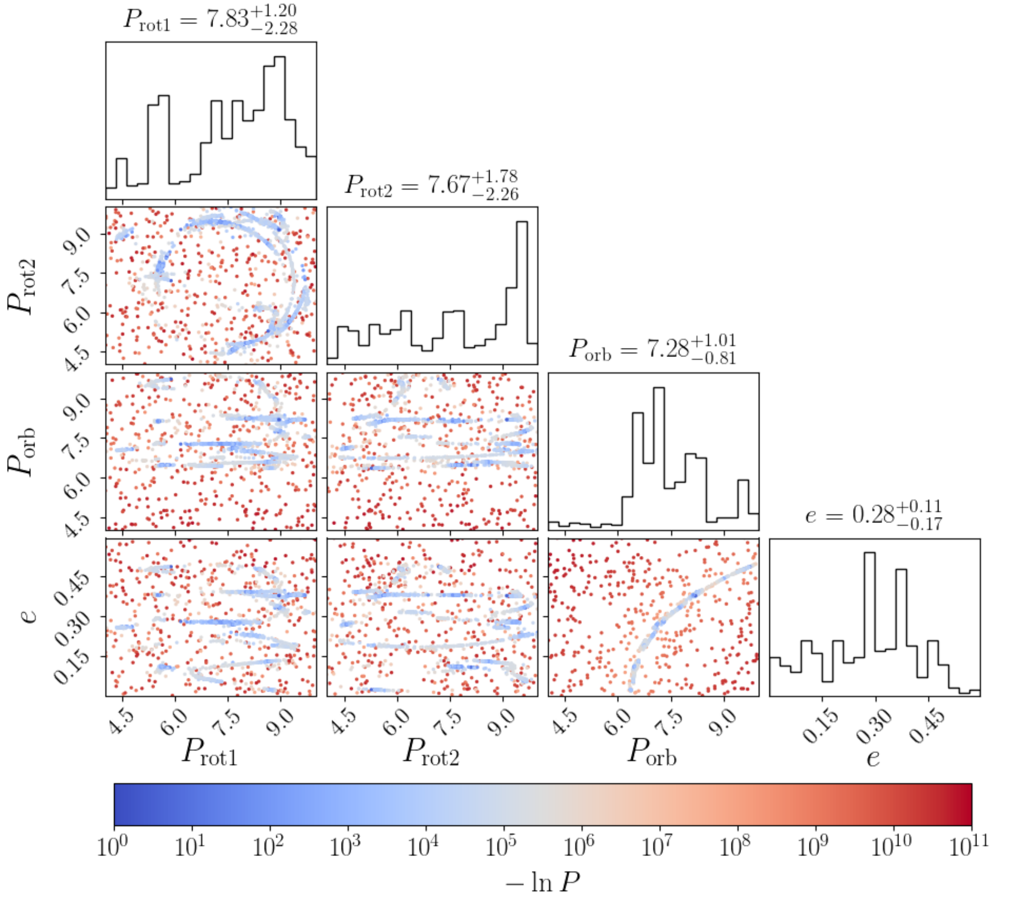
Posteriors differ by ~3% (within sampling error)

Code and examples available on GitHub

Fleming+ (2020)
But see update in Birky+ (2021)

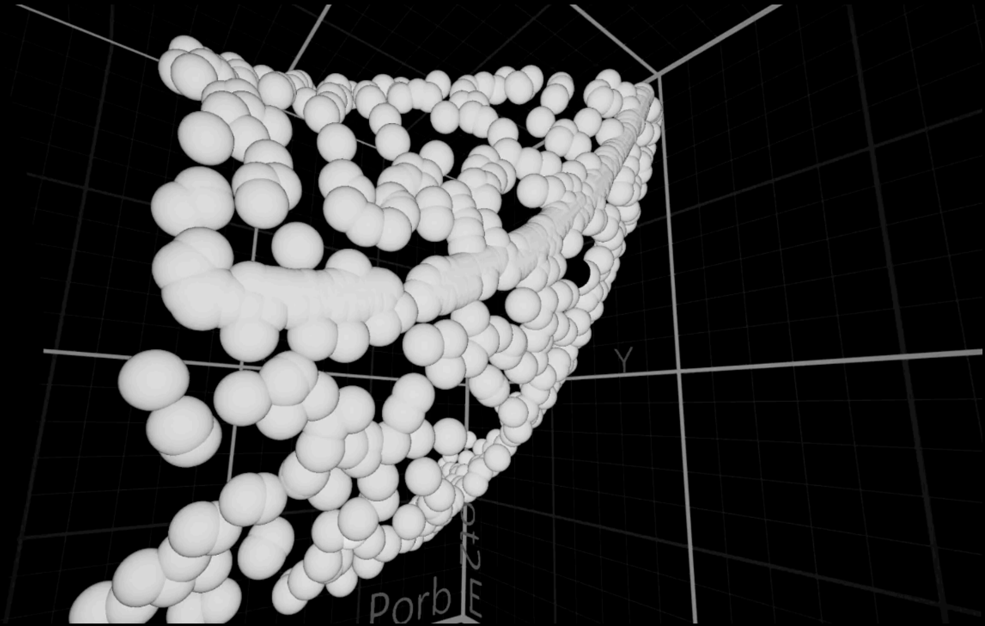
Additional and Planned Resources

VR Ulysses: Data exploration in Virtual Reality (UW spin-off)



Posteriors show strange trends in 2D

In 3D, there's clearly a surface



vrulysses.com

HTC Vive only (for now)

Using VR, we realized we had used incorrect bounds for one body

Thanks to Jessica Birky for figure and video!