**Instructions for RDI**

Creating the reference library:

1. Log into analysis

**ssh -XY [username e.g. abc123]@analysis.astro.ex.ac.uk**

1. Navigate to your champion stars folder in the DangerZoner

**cd ../../../data/shinkley/Keck\_Data/completed\_stars/DangerZone/HR8799**

1. List all the files in this folder

**ls**

There should be folders with the names of their epochs e.g. 2012aug28. Make a note of your chosen epoch

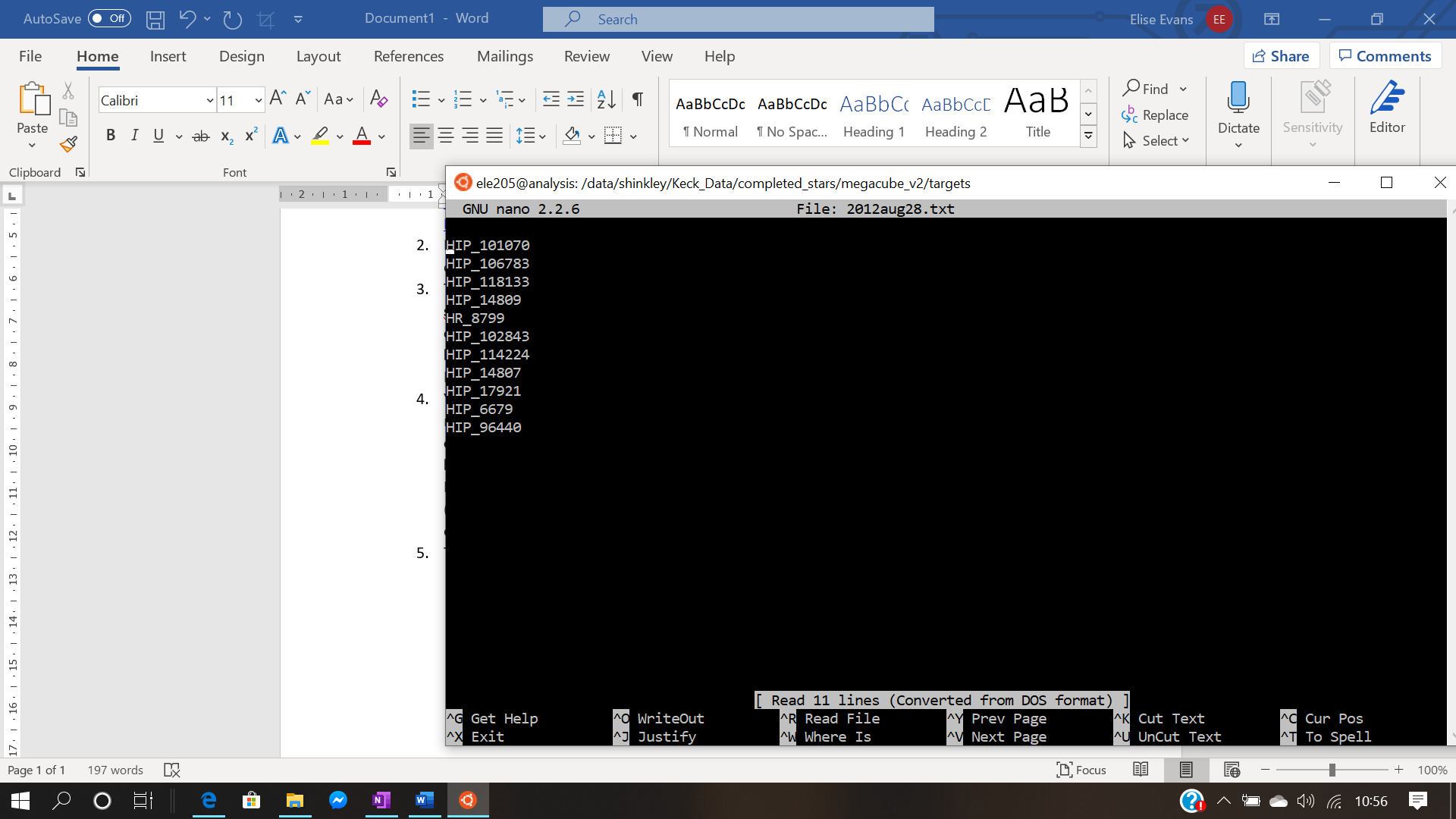
1. To find a list of all the stars with this same epoch, navigate back to the DangerZone folder

**cd ../**

**find -name "2012aug28"**

where 2012aug28 is your epoch – this should produce a list of stars that you need to copy into a text file.

1. The text file of the list of targets needs to be in this form, including the underscore in the star names:



To create this file navigate to the targets folder in megacube\_v2, which is in completed\_stars. If you’re in the folder completed\_stars:

**cd megacube\_v2/targets**

**umask 0000** (This is so the files you create will have rw permissions for all so that others can use the same files for the epoch)

**nano <epoch>.txt** where epoch is in the form 2012aug28

and then write the list of stars, and save and exit using Ctrl X.

You could also create this file on your laptop and copy it to the targets folder using:

**scp <epoch>.txt abc123@analysis.astro.ex.ac.uk:/data/shinkley/Keck\_Data/**

**completed\_stars/megacube\_v2/targets**

1. Navigate to the megacube\_v2 folder

**cd ..** (if you’re in the targets folder)

1. Run the cubestitch code using:

**python cubestitch5.py -cad <epoch>** where epoch is in the form 2012aug28

1. Outputs:

* <epoch>\_megacube.fits (in output folder) --- The reference cube
* <epoch>\_\_target\_frames\_data\_megacube.csv (in stitch\_data folder) --- A dictionary of each star in the reference cube and their corresponding image numbers
* <epoch>\_\_angles\_megacube.csv (in stitch\_data folder) --- A dictionary of each star and their parallactic angles

1. Open ds9 and the <epoch>\_megacube.fits file (ds9 is found in analysis in **usr/local/ds9** then **./ds9**) and check that all the stars align with 512,512 at the centre.

If they don’t:

There is a manual align function

**python cubestitch5.py -align <epoch>** where epoch is in the form 2012aug28

which asks for the frame number and the coordinate of the star in this frame

Running the RDI code:

1. Navigate to your stars folder in DangerZone and create a new folder called RDI and a folder called correlation in RDI.

For example, from megacube\_v2:

**cd ../DangerZone/<Star>**

**mkdir RDI**

**cd RDI**

**mkdir correlation**

1. In this folder, you’ll need the RDI script, the correlation script and the script combining them as well as the psf.fits file

**cp ../../RDI/\*.py .**

If the psf is in the epoch’s folder:

**cp ../<epoch>/psf.fits .**

1. To run RDI:

**python -W ignore RDIcorrelationv5.py <epoch> <star> <Maximum PCs>**

e.g python -W ignore RDI\_correlation.py 2012aug28 HR\_8799 10

* For each RDI run it iterates through the max PCs number so too high can take a long time (It also can’t be more than the number of target images so RDIv5.py changes it to the max if it’s too big)
* You need to enter the minimum, maximum and step size for the percentage. This is the percentage of the reference cube that will contain target images, with the rest of the images being the best correlated images.
* It’ll ask you for a coordinate of a bright spot, If you’re doing this on a star you’ve already run the project script for then you might have some coordinates written down. If you don’t you’ll need to open the megacube (found in megacube\_v2/output) in ds9 and note some coordinates

1. Outputs:

* It’ll first create the correlation reference cube, saved in the correlation folder along with the target frames and angles files. If not all the target images were used then the actual percentage and the input percentage might be very different so it tries again to produce another cube with (hopefully) a closer percentage to the input.
* It loops through RDI for each of the correlation cubes, printing the optimal number of PCs in the console
* It’ll print out the value for the contrast at 1 arcsecond and say which percentage gave the best results and saves the files for just this RDI run (apart from the contrast curve data)
* <star>\_RDI\_<epoch>\_PCAS.fits --- A cube containing the processed images for each PC (the first image is PC=1, the second is PC = 2 etc, so the image with the optimal number of PCs can be chosen)
* RDI\_contrast\_curve\_<epoch>.png --- A picture of the contrast curve for the optimal number of PCs for the best percentage
* RDI\_contrast\_curve\_<epoch>\_<percentage>.txt --- A text file of all the data from the contrast curve – this will be saved for each percentage entered.
* RDI\_snr\_<epoch>.txt --- A text file of the signal to noise ratio for each PC, this can be mostly ignored it just demonstrates the optimal number of PCs has the best S/N ratio

1. To run ADI:

**python -W ignore RDIv5.py <epoch> <star> <Maximum PCs> ADI**

e.g python -W ignore RDIv5.py 2012aug28 HR\_8799 1 ADI

* This produces the same outputs as RDI (but with ADI instead of RDI in the title)

Contrast curves:

RDIv5 has fixed some issues with the contrast curve but does not calibrate them. The calibration is needed from assuming the same exposure time (and hence the same starphot) and because psf.fits is an image of BD+45598 so the flux needs to be calibrates

In DangerZone/RDI/RDIv6 there are the codes to perform RDI and produce a calibrated curve – it needs the k band magnitude of the target star as well (can be found in the Keck spreadsheet)

In DangerZone/Contrast\_Curves:

* Contrast.py: remakes the calibrated contrast curves for RDI and ADI from a csv file list of stars – usage: python -W ignore Contrast.py file\_name.csv
* Average.py: Creates an average of all the contrast curves – usage: python -W ignore Average.py file\_name.csv
* ADI.py: Runs ADI on a csv file of stars – usage: python -W ignore ADI.py file\_name.csv Max\_PCs

Possible errors:

1. Originally I got an error that the vip module was not installed. I fixed this by installing miniconda on my home directory

**wget https://repo.continuum.io/miniconda/**[**Miniconda3-py37\_4.8.3-Linux-x86\_64.sh**](https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh)

**bash** [**Miniconda3-py37\_4.8.3-Linux-x86\_64.sh**](https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.3-Linux-x86_64.sh)

**pip install vip-hci**

* VIP will only work on python version 3.7 (not 3.8 currently)

1. I found that some of the stars didn’t have a centred cube in their dangerzone folder so the cubestitch code came up with an error saying the file didn’t exist. You can check you’ve typed the name of the star correctly and then just remove this star from the list of targets. This also applies if the centred cube is not in the normal location as the code won’t be able to find it.