GETTING STARTED WITH THE VOID FINDER
VIDE: The Void IDEntifier pipeline

Draft guidelines by Alice Pisani

Here is a more detailed list of guidelines to download, install and run the void finder pipeline. It is just a draft for now, but it can still give some help. All terminal commands are written in blue.

I) Download

- Open terminal
- Install git (on a mac you can write: brew install git)
- Go to bickbucket site and enter with your account and password. Next to "download" is the git link, copy it.
- Create a directory where you want to install the void finder: mkdir void_finder (for example)
- With the terminal cd to that directory: cd void_finder
- Paste the git link in the following terminal line: git clone ≪pasted link for void_finder≫
- This will create a directory called *void_identification* with all files of void finder. Do not change the name *void_identification*.
- Terminal: create a outputs directory in void_finder: mkdir outputs You will need space to put results in this directory (almost same size of simulation or data you use).

II) Install

- Into the directory where git downloaded the void finder go into *void_identification*: cd void_identification
- Terminal: ccmake CmakeList.txt
- Terminal: check if everything is ok (paths!). If you need to change something, press t to see full configuration and then modify link pressing enter.
- Terminal: press c (means configure) press g (means generate)
- Terminal: make

- You have to create a file to give the paths needed by the void finder. You can find an example for the file at this path:
  void_identification/python_tools/pipeline_source/defaults.py
  Once you copy your simulation (or data) to a folder you choose, insert the paths in this file. This file, that we now call *name_sim.py*, must be saved in *void_identification/pipeline/datasets*

III) Run!

The run must be divided in two parts: generate the catalogs from your file of simulation/data *name_sim.py*. This will create python files needed for the second part: run the void finder on those catalogs.

- Generate catalogs: you can create a bash script in the folder void_identification: *run.sh* (but if you prefer you can directly run the command line for the initialization of the void finder).

  - Copy these lines in your bash script file
    #!/bin/sh
    echo "Initializing void finder..."
    bash /path_to_your_void_finder_folder/void_finder/void_identification/run_python.sh
    /path_to_your_void_finder_folder/void_finder/void_identification/pipeline/prepareCatalogs.py
    –parm='/path_to_your_void_finder_folder/ void_finder/void_identification/pipeline
    /datasets/name_sim.py' –scripts –halos –subsamples
    exit 0
    where you have to change the *path_to_your_void_finder_folder* and *name_sim.py* with the one you used.
  - Run the bash file: bash run.sh This generates catalogs. It will produce a script for each chosen subsampling.
  - Now the catalogs are ready to run the void finder. Go to the folder scripts at the path you gave in your *name_sim.py* file. It is probably in *void_finder/output/scripts/* . You now have there a file with the name of your simulation: *sim_name_redshift_value_subsampling.py*.
    (If you asked for halos you will have also the file to run with halos: *sim_name_halos_subsampling.py*. You can run the void finder on both files alternatively.) These are the files created when generating catalogs, the void finder will use them.
    Copy the file name that you want to use, you will need it to run the void finder.

- Run the void finder.

  - You can use the following bash file, where you copy the file name you have in scripts.
    #!/bin/sh
    echo "Running void finder..."
    bash /path_to_your_void_finder_folder/void_finder/void_identification/run_python.sh
    /path_to_your_void_finder_folder/void_finder/void_identification/pipeline/generateCatalog.py

/path_to_your_void_finder_folder/void_finder/outputs/scripts/sim_name_redshift_value_subsampling.py

exit 0

- So now it's running, let's check this:
  In the folder */path_to_your_void_finder_folder/void_finder/outputs/logs/* a new folder is created with the name of the simulation/data you used. Inside this folder the run will create three files:

  generate_name_sim.out: builds the mock catalog for the void finder

  zobov_name_sim.out: shows details about the procedure of the void finder (at the end of the file is the number of zones)

  pruneVoids_name_sim.out: selects the accepted voids

IV)And then? Where are the results?

In the folder */path_to_your_void_finder_folder/void_finder/outputs/* is a folder with the name of simulation containing a *sample_name_simulation* folder. Here you can find all the results from the void finder.
A —very brief—description of the output (to be continued):

- sample_info.txt : contains informations about the simulation
- centers_ (no prefix): central density cut, just top-level (which means files where children are removed, it only contains parent voids)
- untrimmed_centers_: no central density cut, all voids in tree
- untrimmed_dencut_centers_: central density cut, all voids in tree
- trimmed_nodencut_centers_: no central density cut, just top-level (which means files where children are removed, it only contains parent voids)

In this files you have the position of the center of the void, the normalized volume, the radius, the volume, the void ID, the density contrast, the number of particles, the parent ID, the tree level (smaller level corresponds to parents), the number of children and the central density.