Some improvements so far

- Converted to Python 3
- Code refactoring in workflow/qcore
- Centralizing of constants and simulation folder structure
- Unit tests
- Initial End to End test
- Environments

HPC Environments

- Usage:
  - activate_env /nesi/project/nesi00213/Environments/cbs51
    - Updates $gmsim, $PYTHONPATH, $CUR_ENV
    - Activates python virtual env associated with environment
  - deactivate_env
- Creation
  - Simple, /worfklow/install_workflow/create_env env_name config
- Updating
  - Update git repo as ususal
  - If it's a package, install with pip install (−I) (--no-deps) ./package
  - Use pip −e option to not have to reinstall during development (haven't actually tested this..), https://pip-python3.readthedocs.io/en/latest/reference/pip_install.html#editable-installs
- Note: Path modifications while in a python virtual env are lost when deactivating

Packages

- Convert repos to packages, as done with qcore and Empirical Engine
- Reduces requirement on $PYTHONPATH and other environment variables ($gmsim, $impath)
- Scripts can be specified in the setup.py, which means upon install they will be added to the $PATH
    - Removes the need for $gmsim/bla/script.py
    - Do we want to use this?
    - If this is done sufficiently, removes the need for $gmsim in slurm scripts etc.
- Versioning
    - Follow Brendon’s guidelines, i.e. year.month.minor with year.month getting updated for major versions?
        - When do we update minor/major version?
    - Changelog − start updating changelog.md?
        - Also allows noting environment changes required for updating to specific version?
- Dependencies across packages, requirements in setup.py

Branching & EndToEnd test

- Feature branches for development -> E2E test feature branch in developer HPC environment -> PR -> unittests -> merge master -> E2E in master environment on HPC (either manually by developer or once every x hours?), ensures that changes work with changes from other developers.
- Have a stable branch, to which master is released when it is deemed stable/good version? Or use tags on master (and create pip package for it)?
- Discuss? Other ideas?

Misc

- Licenses on open source projects?
- Logging of time spent assisting researchers on misc stuff (to get an idea how much time spent on this, impacts on sprint etc)?
    - Single task to track time (for under 1 hour)