gim

`gim` (GIt for Modeling) is a wrapper script that facilitates synchronization of pharmacometric models and results between the local computer and remote servers/clouds. It is initially aimed at PsN/NONMEM, but might be extended to other modeling tools as well.

The problem that `gim` solves

Many pharmacometric modelers find it a pain to constantly move files between the local computer and a cluster or a cloud server. Modeling only on a local computer is usually not an option due to limited CPU capacity. On the other hand, modeling on a cluster/cloud is not always convenient either: although some excellent cloud tools are available (like RStudio Server, PiranaJS), many modelers find it more convenient to parse results and datasets on their local machine. Also, cluster administrators may not allow installation of specific custom tools or libraries, and remote modeling of course always requires a connection to the internet. `gim` solves these problems by performing 'on-the-fly' transfer of models and results between your local computer and the cloud / cluster using Git. At the same time it provides automatic backup and version control of your work in the cloud.

The mechanism

`gim` is called from the console on your local machine, followed by a location indicator, and the PsN command you want to execute:

```
gim [location] [PsN command]
```

The location indicator specifies where you want to run the PsN command. With every call to `gim`, your model and results are first saved to the local Git repository and synchronized to the cloud (`commit` + `push`), before any run is started. If the location indicator is not `local`, then `gim` will login to the specified server, lookup the project folder you are working in locally, and synchronize to the latest version of your models (`git pull`). Afer that it will start the actual PsN command on the server, and return to your local console.

With the `status` and `sync` commands you can check run progress and synchronize results between your local computer and your server(s).

Some example commands

```
# Setup gim on machine and customize settings. This command needs to
```

```
# be run on the local machine and on the remote server(s).
gim setup

# run a model locally
# (same as 'execute run1.mod', but first saves/syncs changes)
gim local execute run1.mod

# 'local' may be dropped, as it is default
gim execute run1.mod

# Run the same model but now on the 'serv1' cluster
gim serv1 execute run1.mod

# Check if any results are available on any of the specified servers
gim status

# Check if any results are available on server 'serv1'
gim status serv1

# Look for new models/results on all servers, and synchronize
gim sync

# Look for new models/results on 'serv1' and synchronize
gim sync serv1

# Set up the git repository on your local machine, as well as
# on all specified servers
gim init
gim sync

# set/change link to repository at GitHub or BitBucket
gim link github project1
gim link bitbucket project1
```

A short tutorial is available in `/doc`.

Advantages / use cases

- Run your simple models locally, dispatch heavier computations to a cluster
- Run models locally when not connected to the internet, and work in the cloud when you are
- Always have a backup in the cloud
- Automatic version control on all models and results
- No wait when cluster is down, just sync your project to a different cloud/cluster

- Easily share projects with colleagues
- Integrates with Pirana (desktop version). Fully compatible with PiranaJS.

Requirements

- PsN, NONMEM, and `gim` and `git` need to be installed both locally and on the servers.
- GitHub account or custom Git server account

Installation guide

Follows soon....

License

Open source MIT license.