gim tutorial

In this tutorial we will start a new NONMEM/PsN modeling project with gim from scratch, and explain the intended workflow. The tutorial assumes you have `gim` and `git` installed, please have a look at the installation guide if you have not done so yet.

Initialization

First, we'll create a folder for our new project and initialize gim:

```
cd ~/projects
mkdir proj_x
cd proj_x
```

Next, we will need to create the repository and hook it up to a git cloud service such as GitHub or BitBucket. For this, go to the respective website (e.g. github.com, bitbucket.com) and create a new repository, with the same name as you used for the folder name (e.g. `proj_x`). If you don't have an account at one of these services you will obviously need to create one first.

We can then initiate our git repository on the local machine and link it to the cloud. Assuming you use BitBucket for your project:

```
gim init bitbucket
```

or for GitHub:

```
gim init github
```

Note: Other git servers can be defined in your setup file (`~/.gim/settings.json`).

You should see this in your console:

```
gim: creating new git repository in current folder.
gim: done.
```

Now you are set to go!

A first commit

Before you start modeling you probably want to add some files to the folder, like datasets or pdf files. Let's do that, and then do a first synchronization to the cloud. So after adding some files, save your work to the cloud with:

```
gim sync
```

It may be that the first time you run this command, `gim` will have to do some extra work (save the git server as a known host), but in principle the output should look something like:

```
$ gim sync
gim: pulling changes from central (git@bitbucket.com:ronkeizer/proj_x)
gim: git added 4 files to repository (localhost_name)
gim: changes pushed to remote
```

What `gim` is doing is first to look for any new stuff on the central git repository (there is none now) and update the local repository if needed. Afterwards it will save the updates and new files to the central repository. If you now go to the GitHub or BitBucket website and look in your repository, you should see the commit you just made and the new files.

If we try to synchronize again afterwards, `gim` should report that there are no new changes (locally or remote).

```
$ gim sync
gim: pulling files from central (git@bitbucket.com:ronkeizer/proj_x)
gim: no new files or changes found
gim: everything up-to-date
```

Duplicate project

Let's say now we want to continue working on another computer. We can then just do basically the same steps, only now `gim` will automatically 'pull' in the files from the cloud:

```
mkdir proj_x
cd proj_x
gim init
gim link bitbucket proj_x
gim sync
```

which will give:

```
gim: pulling changes from central (git@bitbucket.com:ronkeizer/proj_x)
gim: no new files or changes found locally
gim: everything up-to-date
```

Running models (local)

When you have created your first model, you can run it on your local machine normally using:

```
execute run1.mod
```

Note: `gim` can currently only handle model execution using PsN, and not using `nmfe`.

You can also route the execution through `gim`:

```
gim execute run1.mod
```

This has the advantage that it will first look if there is a newer version of the model in the central repository. Then it will run the model, and afterwards it will push the new result files to the central repository. In effect, it is thus similar to:

```
gim sync
execute run1.mod
gim sync
```

If you don't want to check / update the central repository everytime you run a model, it is probably faster to only use `gim` when you stop working on the project for the day, or after you've made some important progress.

Running models (remote)

The big advantage of using `gim` is not in running models local, but in the ease in which you can run models on a cluster or in the cloud. After we have defined our cluster in the `settings.json` file, the only thing we have to do is specify the desired cluster to run on, and `gim` will take care of the rest.

```
gim amz execute run1.mod
```

will execute `run1.mod` in the cloud (`amz`, e.g. an Amazon EC2 instance). The output should look e.g. like this:

```
$ gim amz execute run1.mod
gim: changes pushed to remote
gim (@amz): git repository found
gim (@amz): starting PsN (execute run1.mod)
gim (@amz): done.
gim: done.
```

Checking run status