

RevBayes has dozens of developers actively adding new features to the program. Usually, they do this as part of their own research, but other times they add functionality in a purely altruistic manner. How do you, as a new developer, add functionality to the program? This document will only describe the manner in which you can add functionality to the program through the pull request mechanism. This is the standard way in which you add your changes to the program.

RevBayes is hosted on github.com. If you haven't already become a member, do so. Ask to join the RevBayes group as a developer.

You will note that there are many branches in the RevBayes repository. There are really only a few you need to concern yourself with. **Master** is a branch you do not touch. This branch reflects the current release version of RevBayes. The **development** branch is where all of the action is. This branch reflects the bleeding edge of the program as your colleagues add functionality to the program. How do you add functionality to the **development** branch?

The screenshot shows the GitHub repository for RevBayes. At the top, the repository name "revbayes / revbayes" is displayed along with statistics: 37 Unwatch, 2 Stars, and 4 Forks. Below this, navigation tabs include Code, Issues (6), Pull requests (5), Actions, Projects (0), Security, Insights, and Settings. The main heading is "Bayesian Phylogenetic Inference Using Graphical Models and an Interactive Model-Specification Language" with a link to <http://revbayes.com> and an Edit button. A "Manage topics" link is also present. A summary bar shows 28 commits, 33 branches, 0 packages, 1 release, 3 contributors, and GPL-3.0 license. Below this, the current branch is "development" and buttons for "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download" are visible. The commit history shows the branch is 27 commits ahead and 9 commits behind master. A recent merge pull request #16 is highlighted. The commit list includes files like doxygen, help, projects, src, tests, validation, .gitignore, .travis.yml, LICENSE, README.md, meson.build, and meson_options.txt, with commit messages and timestamps.

File	Commit Message	Time Ago
doxygen	Oh, rapture!	3 months ago
help	Allow indentation in the help examples.	13 hours ago
projects	removed spaces	9 days ago
src	Allow indentation in the help examples.	13 hours ago
tests	Update expected test output.	19 days ago
validation	Oh, rapture!	3 months ago
.gitignore	raptured development	3 months ago
.travis.yml	fix script	20 days ago
LICENSE	Oh, rapture!	3 months ago
README.md	raptured development	3 months ago
meson.build	raptured development	3 months ago
meson_options.txt	Oh, rapture!	3 months ago

28 commits

33 branches

Branch: development ▾

New pull request

Switch branches/tags

jph_demo

Branches

Tags

Create branch: jph_demo from
'development'

projects

You will make your own branch to add your new features. On the RevBayes GitHub page, change the active branch to **development**, as shown above. To create your own branch from the **development** branch, click on **development** and type in the name of the new branch. Here, I called the new branch **jph_demo**. Then, click “Create branch.” You have now created your own branch!

On your own computer, you should switch the repo to your new branch and start coding! Implementing your ideas may take a while: hours, days, weeks, months. Make certain to pull from the **development** branch occasionally, thereby adding the functionality that your colleagues are adding to **development** to your own branch. It’s unlikely you will encounter conflicts as you do this, but if you do, it’s important to resolve them on your branch. Also, make certain to make commits of your changes to your machine, which you push to the GitHub repo.

For this demo, I simply fixed a minor typo in the **AbsoluteValueFunction.h** file.

```
class AbsoluteValueFunction : public ContinuousFunction {
public:
    AbsoluteValueFunction(const TypedDagNode<double> *a);

    AbsoluteValueFunction* clone(void) const; //!< Create a clone.
    void update(void); //!< Recompute the value

protected:
    void swapParameterInternal(const DagNode *oldP, const DagNode *newP); //!< Implementation of swapping parameters

private:
    const TypedDagNode<double>* a;
};
```

Note that the comment for the clone function says, “Create a clon.” I’m pretty certain that this should read, “Create a clone.” I made this change, committed my change, and pushed it. I’m ready to merge my change into the **development** branch.

Your ideas are likely to take longer to implement, but eventually you will be finished coding your ideas and ready to merge your changes back into the **development** branch, too. RevBayes has a process for doing this. You create a “pull request.” Back on the GitHub site, you will note a “New Pull Request” button on the main page. Push that button! You will be prompted with the following window:

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across](#):

base: development ← compare: jph_demo ✓ Able to merge. These branches can be automatically merged.

Fixed a typo as part of a demo for the RevBayes workflow practice.

Write Preview AA B i “ <> 🔗 ☰ ☷ ✓ @ 📌 ↶

This is a demo for new developers, demonstrating the nifty new workflow pattern (branches with pull requests).

Attach files by dragging & dropping, selecting or pasting them. 📎

Create pull request ▾

ⓘ Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Make certain that “base” is set to development. And, of course, you want “compare” to be your branch. Add a comment for the colleague who will be reviewing your code changes. Once you’ve done that, you are ready to push the “Create pull request” button at the bottom right.

What happens next? Well, your code changes will be reviewed by one or more of your colleagues. Once approved, your contribution must pass a compilation test. If it passes, then your changes will be merged into the development branch. All of your colleagues will then be able to see your contribution as they pull from **development** to their own branches.

Fixed a typo as part of a demo for the RevBayes workflow practice. #21

Edit

 Open jhuelsenbeck wants to merge 1 commit into `development` from `jph_demo`

Conversation 0 | Commits 1 | Checks 0 | Files changed 1 | +1 -1



jhuelsenbeck commented now

Member + 😊 ...

This is a demo for new developers, demonstrating the nifty new workflow pattern (branches with pull requests).

  Fixed a typo as part of a demo for the RevBayes workflow practice. 89d90ab

Add more commits by pushing to the `jph_demo` branch on `revbayes/revbayes`.



 **Review required**
At least 1 approving review is required by reviewers with write access. [Learn more.](#)

 **Some checks haven't completed yet** [Hide all checks](#)
2 pending checks

  `continuous-integration/travis-ci/pr` Pending — The Travis CI build is in ... **Required** [Details](#)


  `continuous-integration/travis-ci/push` Pending — The Travis CI build is i... **Required** [Details](#)

 **Merging is blocked**
Merging can be performed automatically with 1 approving review.

Merge pull request ▼ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Reviewers 

Suggestions

 **willpett** [Request](#)

At least 1 approving review is required to merge this pull request.

Assignees 


No one—assign yourself

Labels 

None yet

Projects 

None yet

Milestone 

No milestone

Linked issues 

Successfully merging this pull request may close these issues.

None yet

Notifications [Customize](#)

As an aside, after creating the pull request, you can request that the code be reviewed by one of the reviewers. Here, I requested that Will Pett review the changes. This is not necessary — someone will review the changes — but sends a notification to that person that you have submitted a pull request.