## Accessibility

* **Be accessible** - it's easy to be blind to the barriers (and there can be many different types of barriers) to joining that community. So be sure to document all the different ways someone can get involved, the roles that can be played and matching these to different types of "phenotype" of community member. Write this down: CONTRIBUTIONS.md for example.
* {Practical tip} But there are many ways of doing this. In general having different levels of contribution can suite many different types of contributors: checking for spelling, writing documentation, doing small-scale programming projects or even large rewrites and new features. Modular design of the codebase is very helpful for this (ruby gems, bionode, CRAN, pypi ...). As an example, shifting from "central control" of a main repository which would consider only high-quality code that fit a specific philosophy to a modular design where anyone could create anything was a revitalising game changer for the bioruby community. Indeed, the biogems initiative provided a means (template), encouragement, and active indexing of all contributions (on<http://biogems.info>); this rapidly and dramatically increased the number of bioruby contributors.

## Communicativeness

* {Practical tip} BioJS uses a "Gitter channel (<https://gitter.im/biojs>), which currently includes 47 members, is a public chat-room that is directly linked to our GitHub repository. Developers use the Gitter channel to announce new commits, bug fixes, patches and forks. Questions and answers about source code are posted to the room on a daily basis and, as a matter of principle, the core development team attempts to provide immediate responses on the Gitter channel."
* {Positivity} **Make friends and code** Invest time and energy in cultivating positive relationships with others in the community. The productivity and long-term success of the project depends upon it.
* **Seek - and then follow - advice** This can include advice from technical experts beyond your community, political advice from senior players, as well as scientific advice to ensure the communities aims are relevant. Importantly (at the risk of sounding daft) don't ask for advice, and then blithely ignore it. People will (rightly) assume that you are being arrogant or stupid.
* If the response to questions, contributions, is quick, then people are more likely to be able to keep track of discussions, feel that they are active, have them more likely to deliver useful answers/solutions, and to reach these more quickly
* {Attentiveness, Openness} enough, enthusiastic, positive people need to be listening to people comments, acknowledging their actions, and responding to them, for people to feel that they are being listened too, and to build active discussions. If you start as a small group, this is a challenge, as it requires more contribution/time commitment from each member than when the group is larger
* Communication and discussion is the life-blood of community; it's how the interactions that make up the community happen. If this is positive and constructive, most people are encouraged by, and enjoy the interaction, and thus are more likely to do more of it

### Diversity

* Diverse by default: bio-informatics is interdisciplinary and thus already brings lots of different professional backgrounds -> see and acknowledge the value of that
* additionally communities are diverse in terms of demographics, perspectives, and experiences
  + offers novel strategies to problem-solving
  + communities learn new skills
  + makes sure barriers to contribute are kept low

## Practical tips

* A profile page for each member where the contributions can be recorded
* Motivating academics by publishing papers using models like BioJS: Professional credit. BioJS published an 'Application Note' in *Bioinformatics* with all contributing members as authors. BioJS has organised that application-note-type-articles are publishable in F1000 for individual BioJS modules, giving contributors an easy way of getting their contributions acknowledged within the scientific literature
* A common update news with the topic "Who's doing what?"
* Updates of contribution in the documents, release notes, monthly newsletters
* Has a mailing or similar channel (e.g. Slack, gitter) list to keep users up-to-date about the projects they are involved in, or provides syndication feed.
* Provides a short glossary/vocabulary to help users quickly understand basic things, for example, types of open-source licenses, version control systems (VCS), & other relevant components of open-source – Optional.
  + **Rich in resources (e.g. infrastructure, project leads, learning materials and help from experienced members for new members etc)**
  + {Supportiveness} Enough members who take on a mentoring role - helps keep active contributors motivated, guiding projects, avoiding conflicts etc.
* **Self-sustaining/perpetuating**
* **Misc. features of the community itself and people involved**
  + {Openness} **Open Spirit** It's not enough to just make the source code available. The project's key movers and shakers must genuinely be open to others people's thoughts, suggestions, leadership and contributions.
  + {Openness} **Open Hearted** Be nice to people. Likely as not they're not paid to work on the project either, and will not want to work with awkward or unpleasant people.
  + {Transparency} **Sustainability is important** Individuals and institutes are more likely to buy into your product if they can have some confidence it is stable, and will be supported in the long-term. To convey stability, make sure that the origin, history and future intents are public and transparent.
  + {Supportiveness} **Make supporting others your no.1 priority** It is always fun to cherry pick, but supporting others in their usage (of the products of your community) must always be the number 1 priority. It is primarily they, not you, who will determine the success of the project in the long term.
  + {Practicality} **Promise less, deliver more** It can be difficult managing ambitions and the excitement of new ideas, especially in a community that is not bound by hard-nosed financials strictures. Brainstorm for sure, but then focus on the achievable practical things, with the bigger picture in mind. Importantly, be seen to deliver on the things you promise.
  + {Transparency, Practical tip} **Have a nice Web site** Advertise your community, it's mission, products, aims etc. in a nice clean Web site. This may be the first port of call for newcomers to the project, so it should function as a staging post for getting involved.

### Governance

All long-lasting communities are governed, even if this is ad-hoc de-facto governance. It may pay to to have a formal governance structure and be clear about the tiers of governance (regardless of whether there is formal management or not).

* **Effective organisational structure:** The organisational structure needs to be flexible so that it can be adapted through time to respond to changing needs
* **Roles and responsibilities** For people to join a community in a productive way, it can help to clarify or even formalise what sort of roles can be played and what are the expectations of that role. It is OK to demand formal responsibilities for certain roles (e.g., release coordinator), while other roles can be very open with no formal responsibilities at all. This helps build professionalism.
* **Leadership:** In particular, it is important to have effective, efficient and inspiring leadership.
* **Delegate**: With most on-going community projects, there tend to be an on-going To Do list to deal with updates and bug fixes. As such, senior/ experienced members of the community should attempt document any tasks that need to undertaken clearly and concisely, and then seek to delegate these task to other community members. This can be particularly helpful to new-comers, who wish to contribute in a meaningful way but are unclear where to start.
  + One useful tools for this includes waffle.io - <https://waffle.io>
* **Openness**
  + Structure: It should be possible for anyone to take up any leadership or other role independent of professional affiliation, geography or other factors not related to their involvement in this community. The base-level of any governance model should always be the general (scientific) public: anyone should be free to pass comment and have their voice heard (and this should be acknowledged and with a mechanism for it).
  + Information: Any vital information with regards to how the community functions should be clearly documented and should be transparent. For example with regards to contributions, there should be clear, concise documentation on exactly how to contribute and how any contributions would recognised.
* Community should encourage and support associated projects and communities.
* **Make friends in high places** Your community is more likely to thrive if it is seen to have the backing of "big beasts" including senior individuals, respected institutes or industry partners.

### Professionalism

It might be a hobby, but that does not mean you should not apply the same professionalism as you would apply to the day job. Most of all, ensure that the basic stuff is done right (and that might mean announcing new releases, making change logs, making sure things do not ship broken). Be transparent about what you would like to do, but cannot because of limited resources.

* **Endorse and actively encourage (even mandate) the development of high quality source code**
  + *Adhere to the language community accepted coding styles*, and thereby ensure that the source code is readable and more accessible to a wider range of developers.
  + *In-Line Documentation* - the average developer should be able to understand each line of code without having to dig into the code. Thus, in-line documentation is helpful in guiding a potential contributor in understanding a section of code.
  + *Simple Logic* is desirable in contrast to complex one-liners that tend to take time to decode.
  + '*Continuous Integration (CI) driven development*' should be preferred. This ensures that the project works as expected under a number of different case scenarios. Moreover, if in place properly and correctly, this provides a quick test to see whether a PR should be merged. Code *lint* tools can also be incorporated into the unit tests to ensure that the source code quality does not deteriorate over time.
  + '*Version Control*' - The type and style of versioning should be *clearly* documented. [Semantic versioning](http://semver.org) is commonly used and if so, it should be used correctly (breaking changes *must* result in a major version update). If these rules are not adhered to, this can result in breaking upstream software that depend on that code.
  + '*Change Log*' - Explain (in detail) what has been updated between minor, and more importantly, between major versions. If necessary provide guides on how to update from previous versions.
  + '*Use easy to install dependencies*' - when developing tools for the scientific community, it is important to remember that the target market (the average biologist) will not have installed all developer packages (as a typical programmer would). And moreover, a number would find it difficult to simply install a package. Therefore, wherever possible dependencies that are easier to install should be preferred.
  + '*Root-less Installation*' should be catered for. If necessary, provide detailed documentation on how to install the software *and* dependencies without root access.
  + '*DRY*' - Follow the *don't repeat yourself* principle to ensure that single responsibility is maintained throughout the code.
* **Excellent software and related infrastructure** (from SSI’s ["Software Evaluation: Criteria-based Assessment"](http://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf))
  + Usability
    - Understandability
    - Documentation
    - Buildability
    - Installability
    - Learnability
  + Sustainability and maintainability
    - Identity
    - Copyright
    - Licencing
    - Accessibility
    - Testability
    - Portability
    - Analysability
    - Changeability
    - Interoperability
* **Encourage open & reproducible research**
  + There are many aspects to openness: to ideas, to new people, to publication, to code.
  + Values reproducibility

### Positivity

**Have fun** Likely as not, a lot of the time you spend working on your OS project will be in your own time. If you're in it for the long haul, try to spend more time on the things you enjoy, than the things that must be done but which you don't enjoy.

### Practicality

**Meet (not too) regularly** Momentum is built by regular meetings, but getting the frequency right is important: respect the fact that everyone is busy. It is better to have well attended, relatively infrequent meetings than regular meetings to which no-one goes. This goes for virtual as well as face-to-face meetings. And aim for at least one "all hands" meeting per year - over beer!

### 

## 

## 

## Konrad, Malvika

### Purposefulness

* Define and communicate a clear vision and mission.
  + **Have a (common) sense of purpose** Everyone in a community will (and should) have their own agenda; it is is important to be clear about it on a personal level, but even more important to forge a common sense of purpose. Try to document, for your community, it's mission and aims - and a vision for the future - and be clear that everyone should support that vision (or help you change it). Well designed strategy to increase its visibility to the community (use of social media, conferences etc.)

### Respectfulness (to include stuff previously tagged as "recognition")

* Promote the friendly, respectful, welcoming tone
  + Respect each others ideas, opinions and point of views
* Contributions acknowledged and rewarded
  + personal
    - Weekly shoutout
    - Direct appreciation
  + professional
    - mention contributions in the changelog
    - include in publications
* Once you have a larger group of contributors to discussions, it's important to actively support and **Credit is infinitely divisible** It benefits everything and harms no one to properly acknowledge the contributions of others, not many how small. Everyone who matters will, sooner or later, realise who are the major players are, so there is nothing to lose by this.
  + **reasons why this is important for a great community of this kind**:
    - Its one of the important factors that motivates members to keep on being an active contributor
    - Helps in self sustaining and increasing visibility within a community

### Supportiveness

* once you have a larger group of contributors to discussions, it's important to actively support and promote the friendly, respectful, welcoming tone
* Enough members who take on a mentoring role - helps keep active contributors motivated, guiding projects, avoiding conflicts etc.
* Project provides support and encouragement for associated sub-projects/sub-communities
* Make supporting others as one of your priorities.

### Transparency

* Individuals and institutes are more likely to buy into your product if they can have some confidence it is stable, and will be supported in the long-term. To convey stability, make sure that the origin, history and future intents are public and transparent (Sustainability)
  + **Have a nice Web site** Advertise your community, it's mission, products, aims etc. in a nice clean Web site. This may be the first port of call for newcomers to the project, so it should function as a staging post for getting involved.
* And be transparent about what you'd like to do, but can't because of limited resources. Be professional. It might be a hobby, but that doesn't mean you shouldn't apply the same professionalism as you would apply to the day job. Most of all, ensure that the basic stuff is done right (and that might mean announcing new releases, making change logs, making sure things don't go out broken etc.).

### Openness

### (define openness, may be put it on the top of the article as everything can be put under this)

* Open/inclusive for new people - make it easy to enter the community (documentation, tips, see respect)
* Open for new ideas, technology
* **Open Spirit** It's not enough to just make the source code available. The project's key movers and shakers must genuinely be open to others people's thoughts, suggestions, leadership and contributions.
* **Open Hearted** Be nice to people. Likely as not they're not paid to work on the project either, and will not want to work with awkward or unpleasant people. = Respect (see above - mayemove)
* enough, enthusiastic, positive people need to be listening to people comments, acknowledging their actions, and responding to them, for people to feel that they are being listened too, and to build active discussions. If you start as a small group, this is a challenge, as it requires more contribution/time commitment from each member than when the group is larger
* Values openness (this is implied, therefore should be a part of introduction rather than a feature; although, there are many aspects to openness, [to ideas, to new people, to publication, to code], perhaps it's worth having it as an extra 'feature' given that [I think] great communities of this kind also have these general, other kinds of openness as a feature...?)