Let’s Move to GitHub!

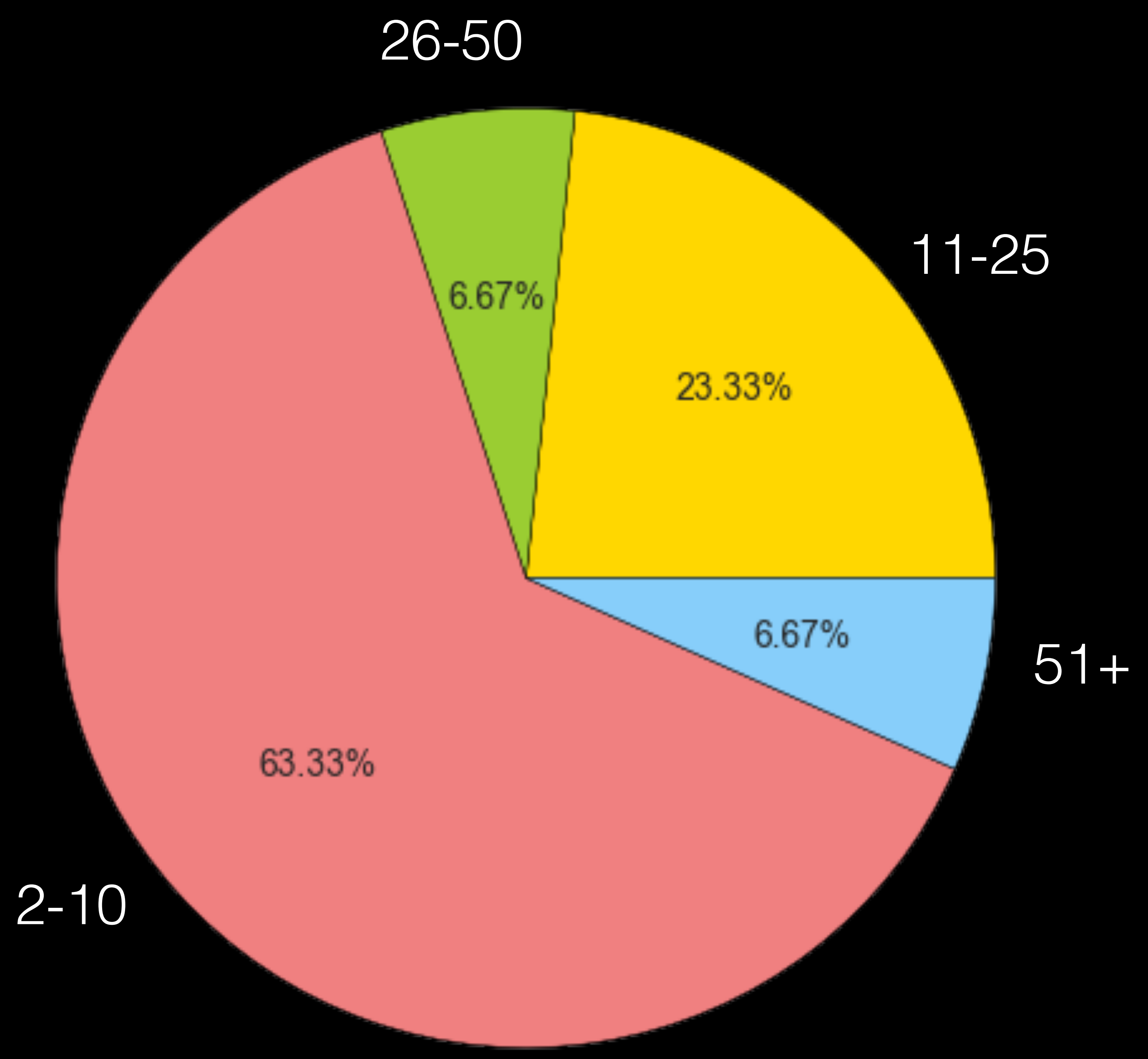
Survey Results

The Survey

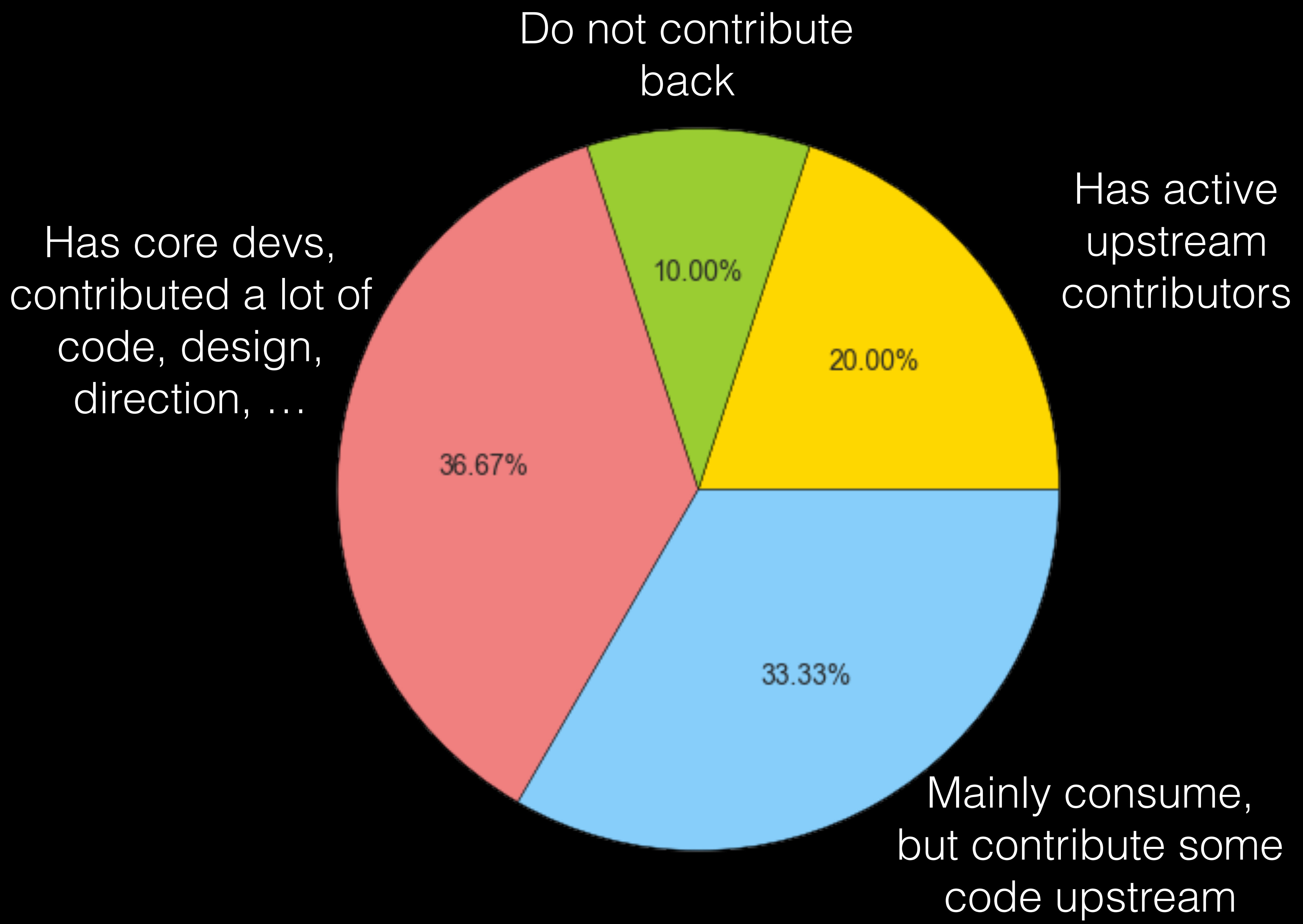
$$\begin{tikzcd}
 & \textbf{235 Answers} \arrow[dl] \arrow[dr] & \\
\text{205 individuals} \arrow[d] & & \text{30 Orgs} \\
\text{73 \textit{Active} Contributors} & & \\[-1.5em]
\text{(\textbf{>10} commits over the last 2 years)} & &
\end{tikzcd}$$

24 Organizations

How Many devs do you have in your team?

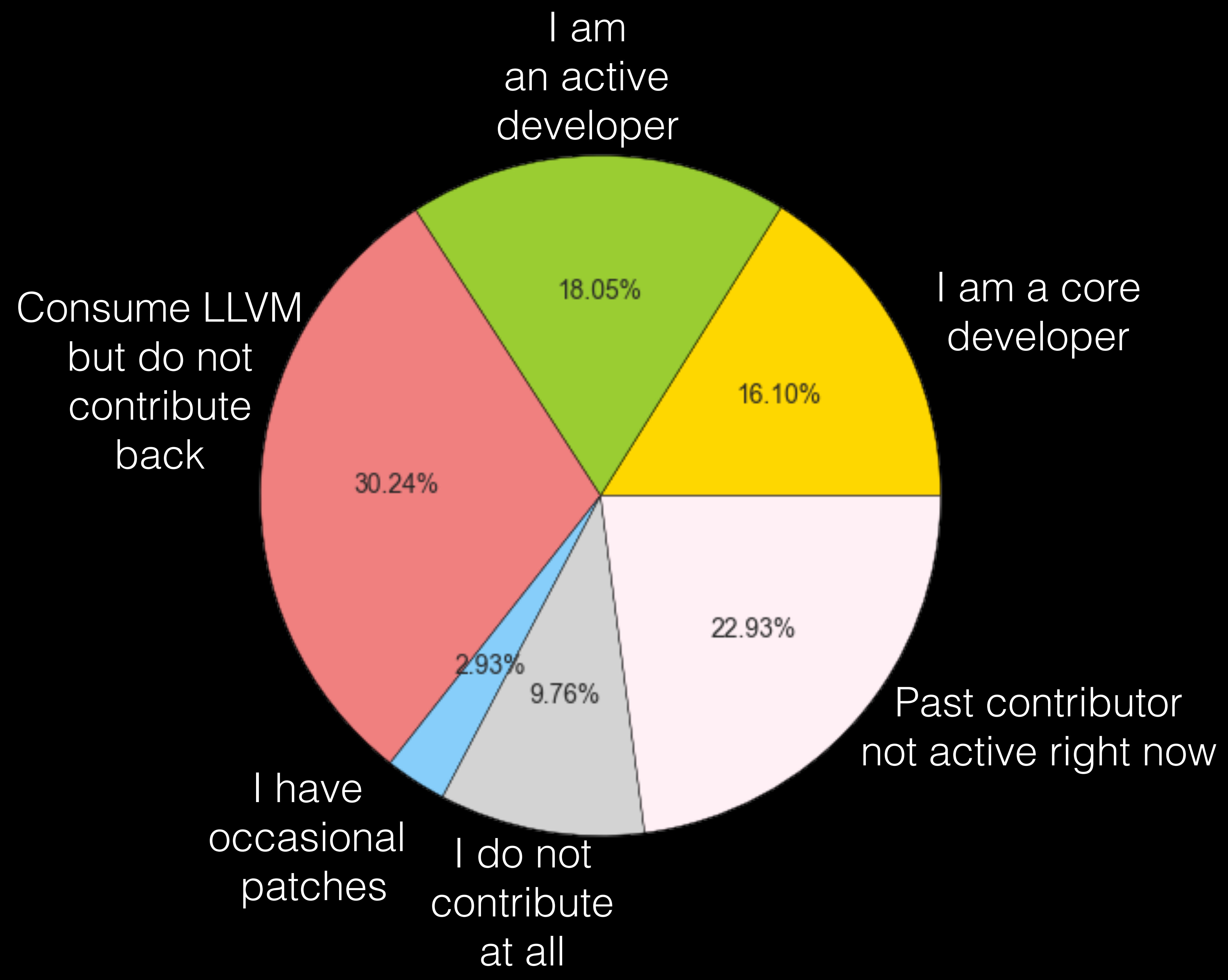


What is your contribution to LLVM?

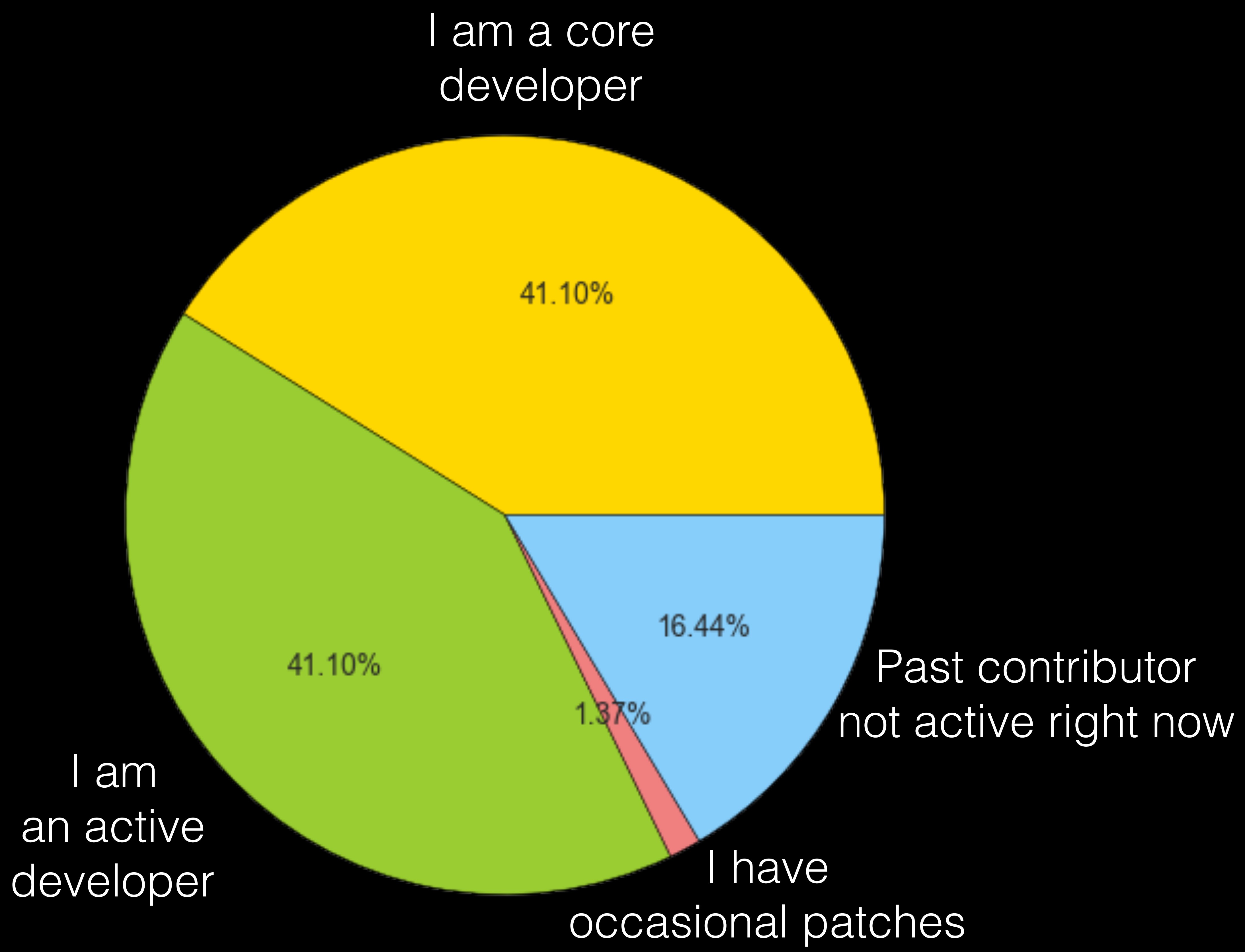


What is your contribution to LLVM?

All Individuals

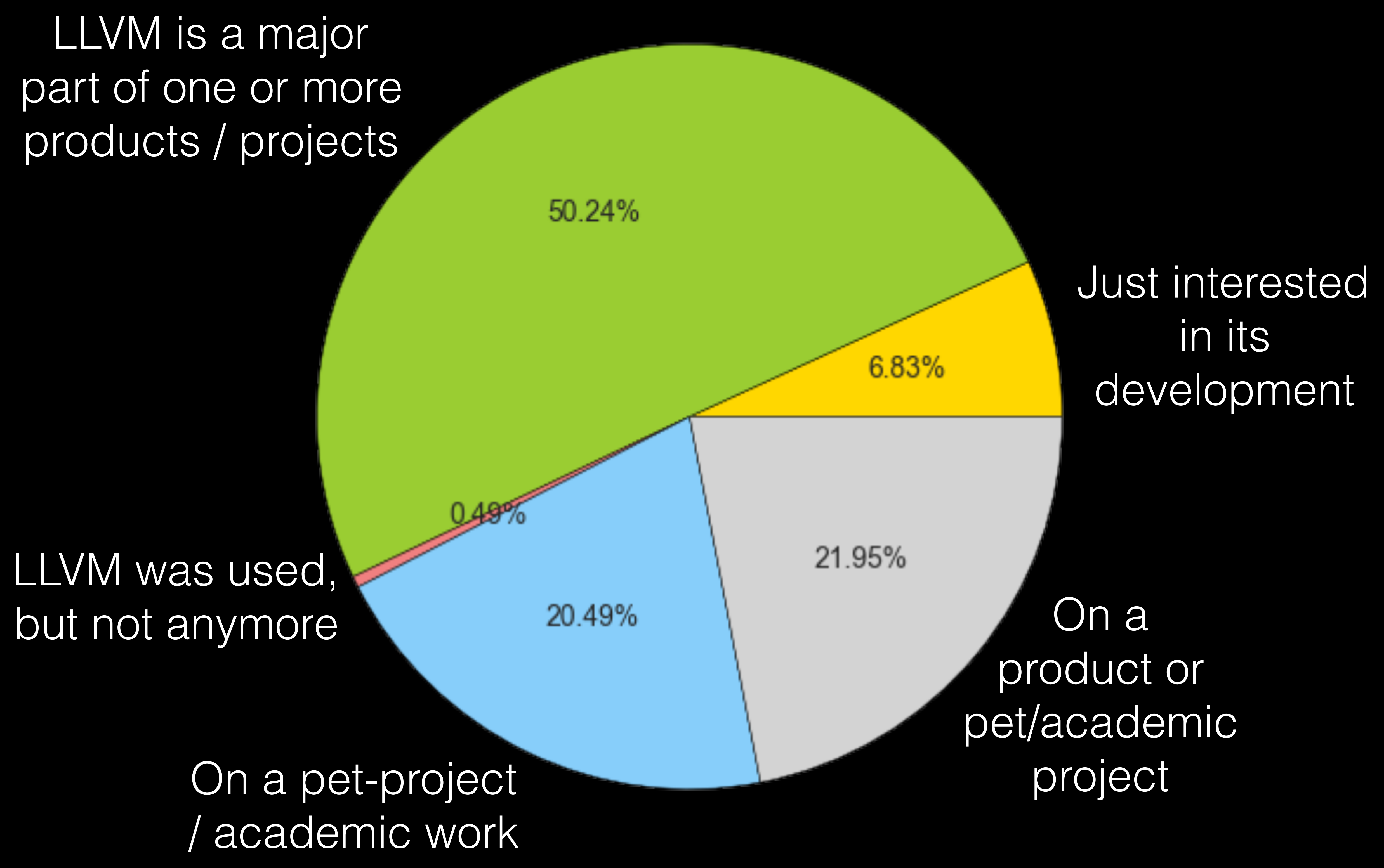


Active Contributors

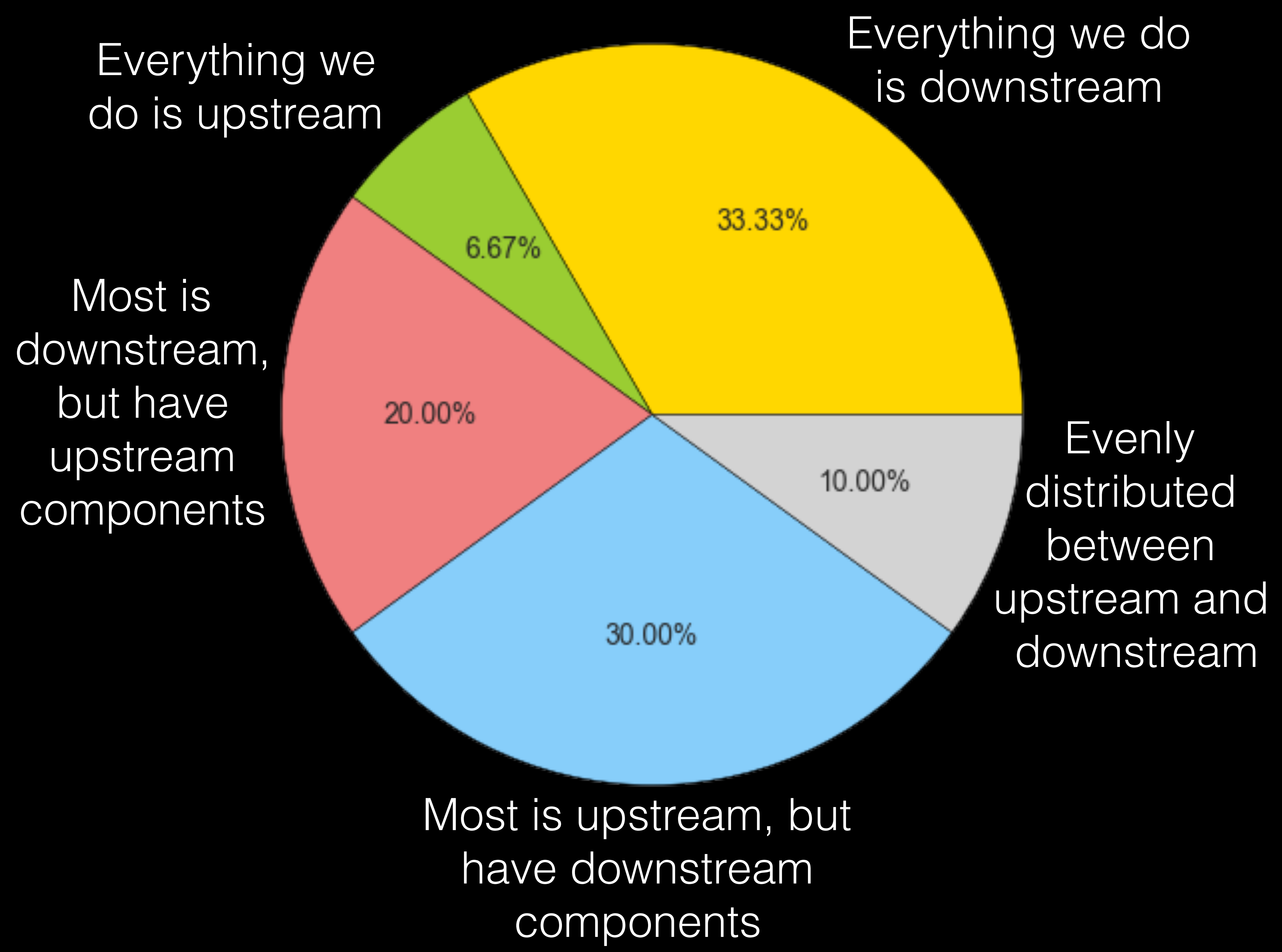


How do you use LLVM?

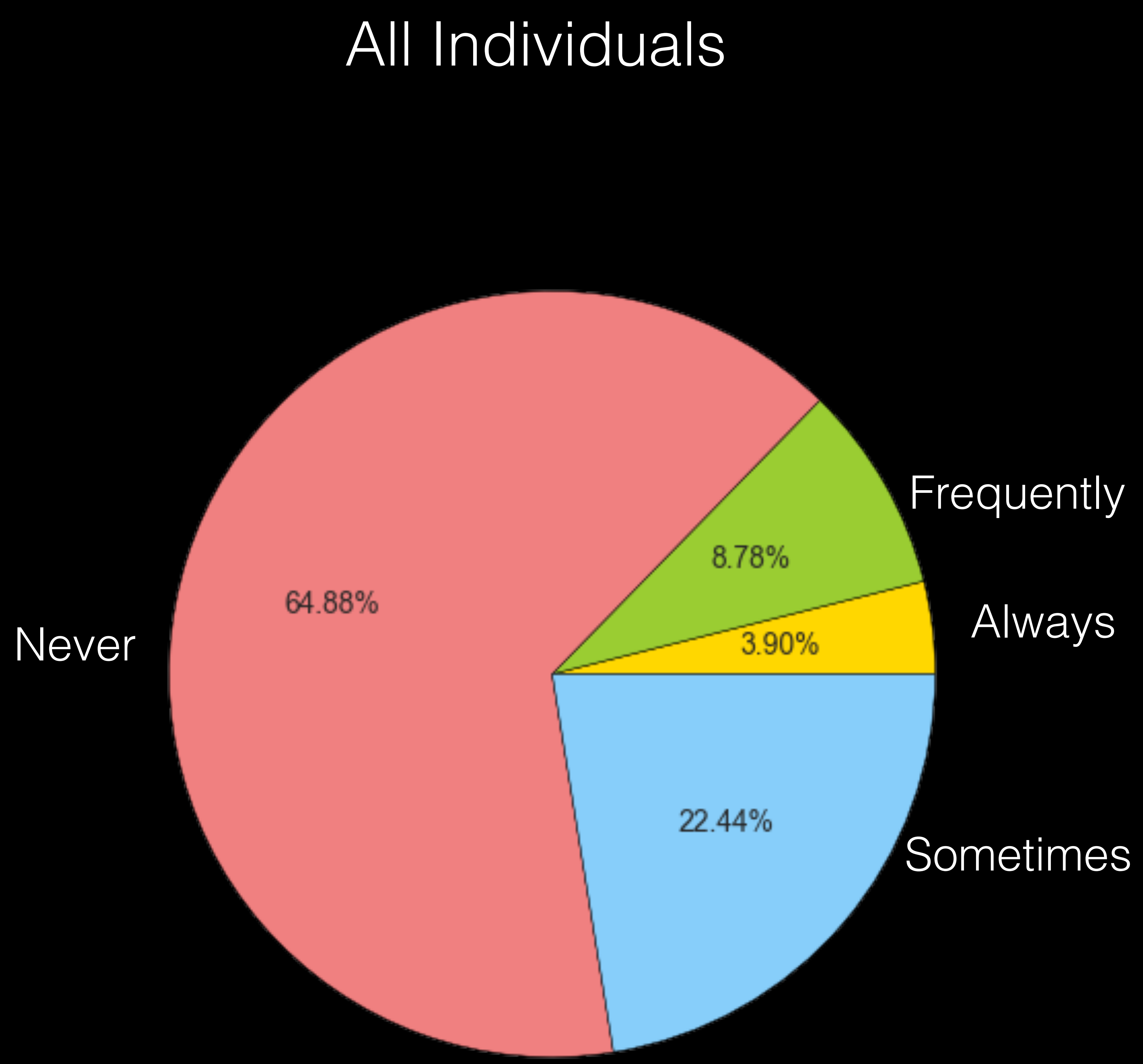
All Individuals

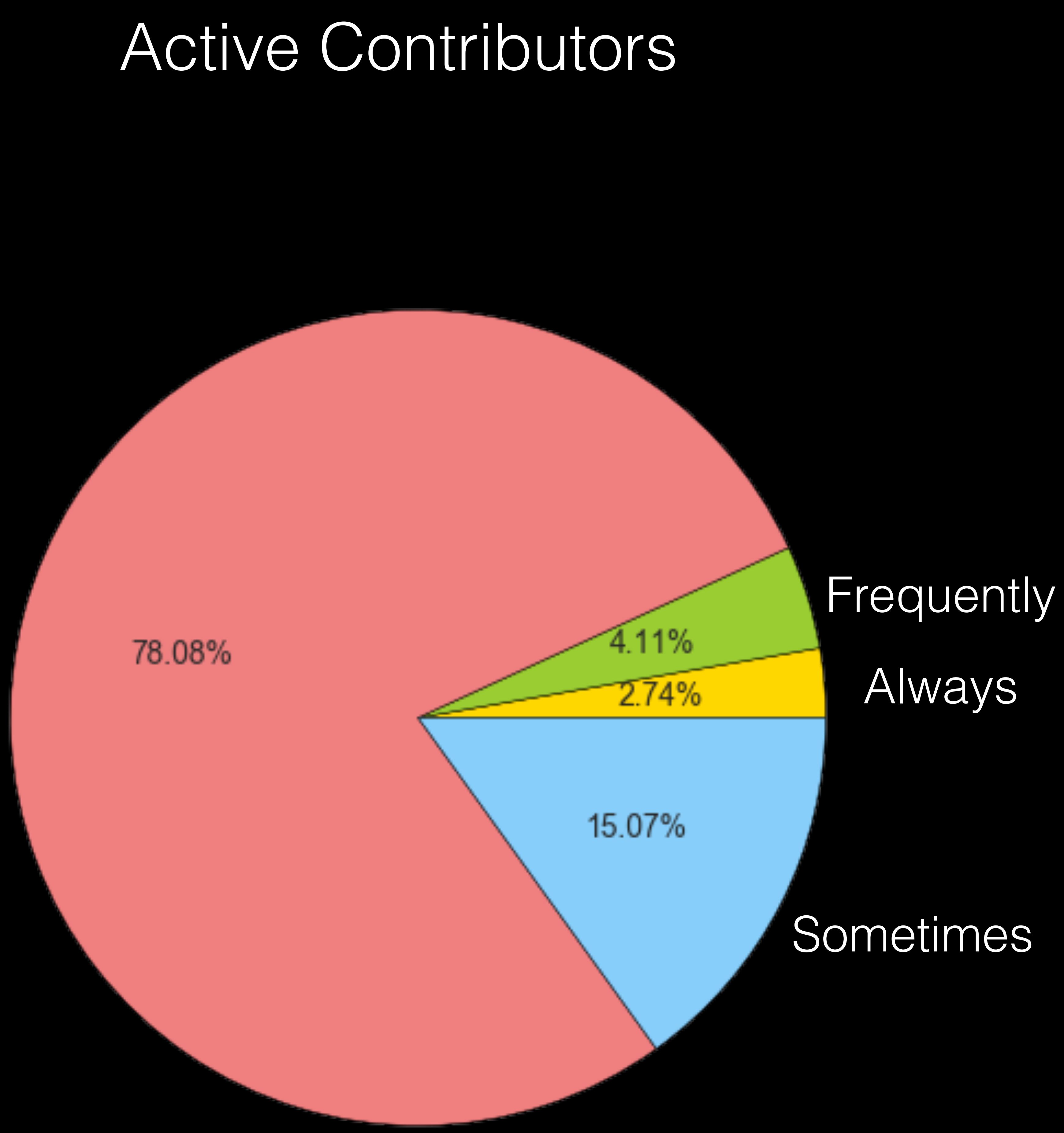


Orgs



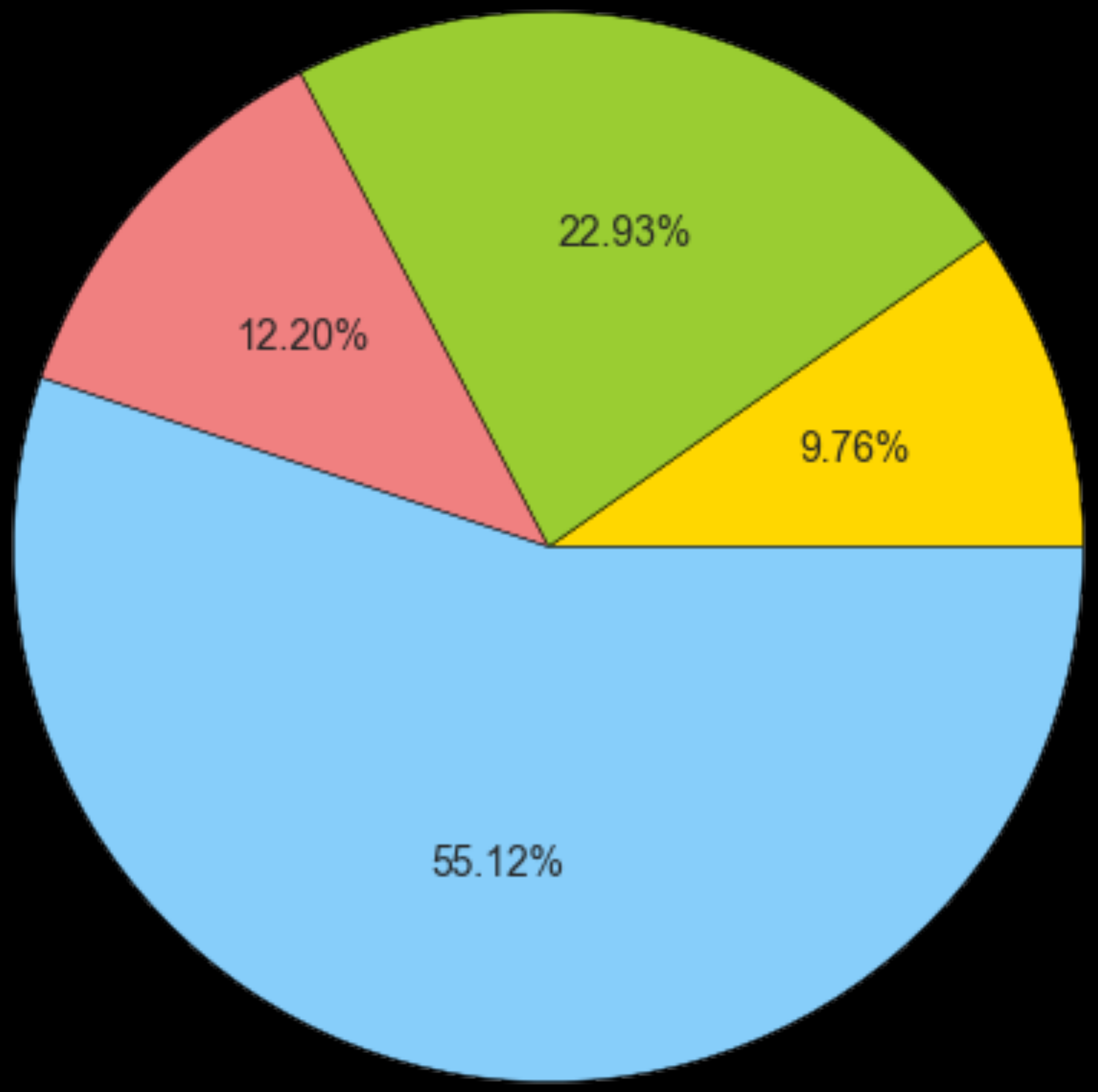
How often do you work on a small LLVM sub-project without LLVM itself?



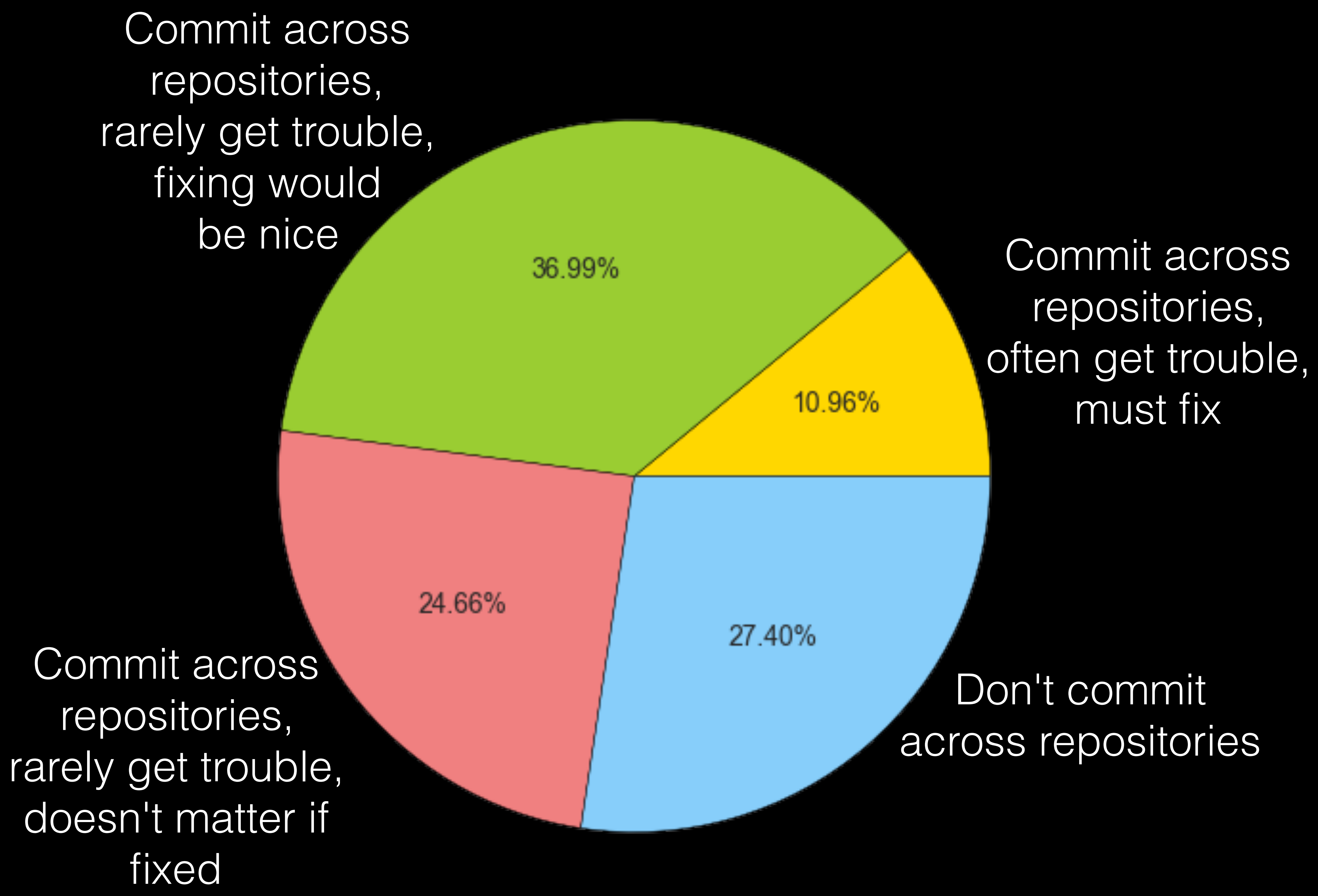


How often do you to commit across repo?

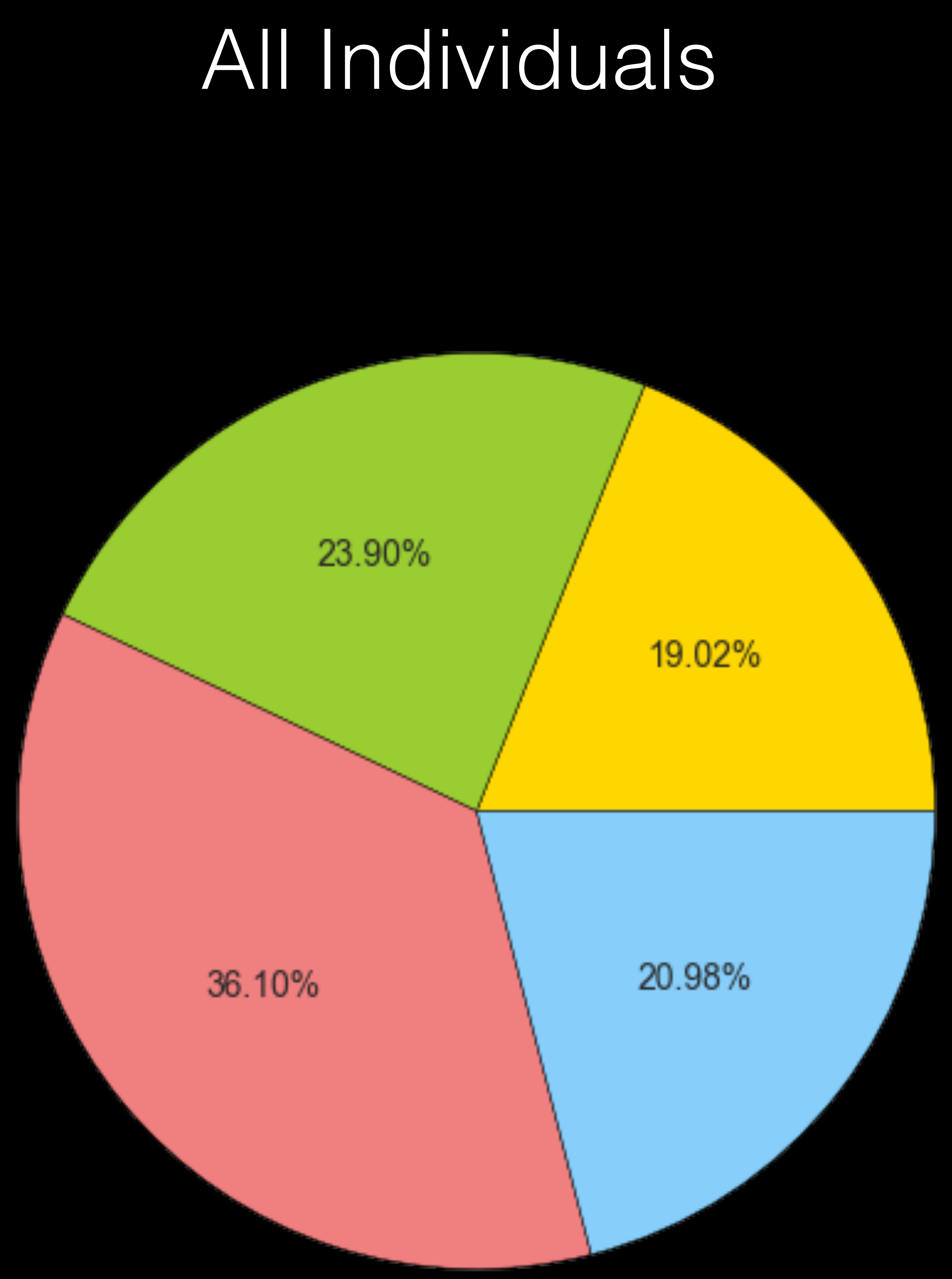
All Individuals

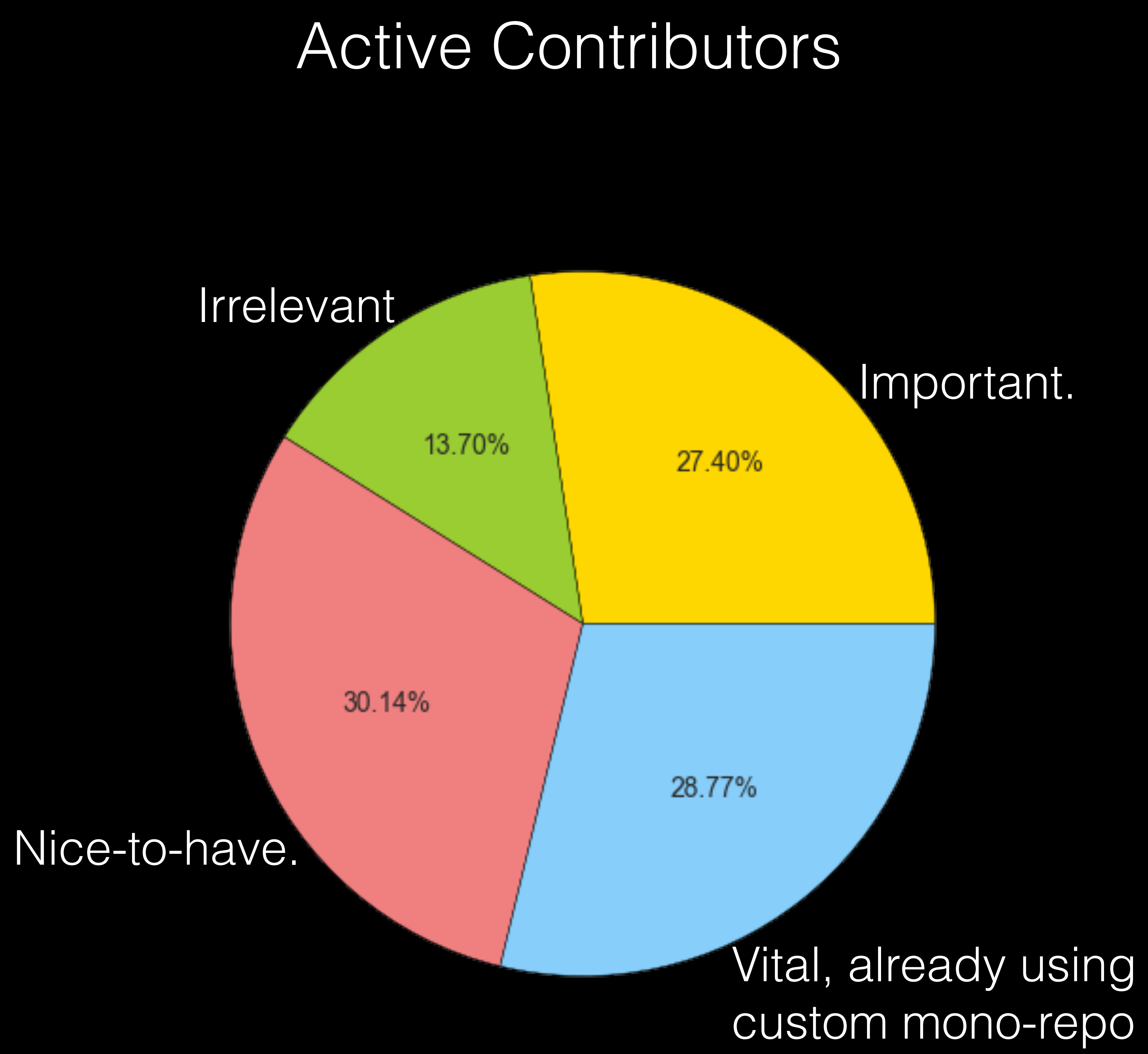


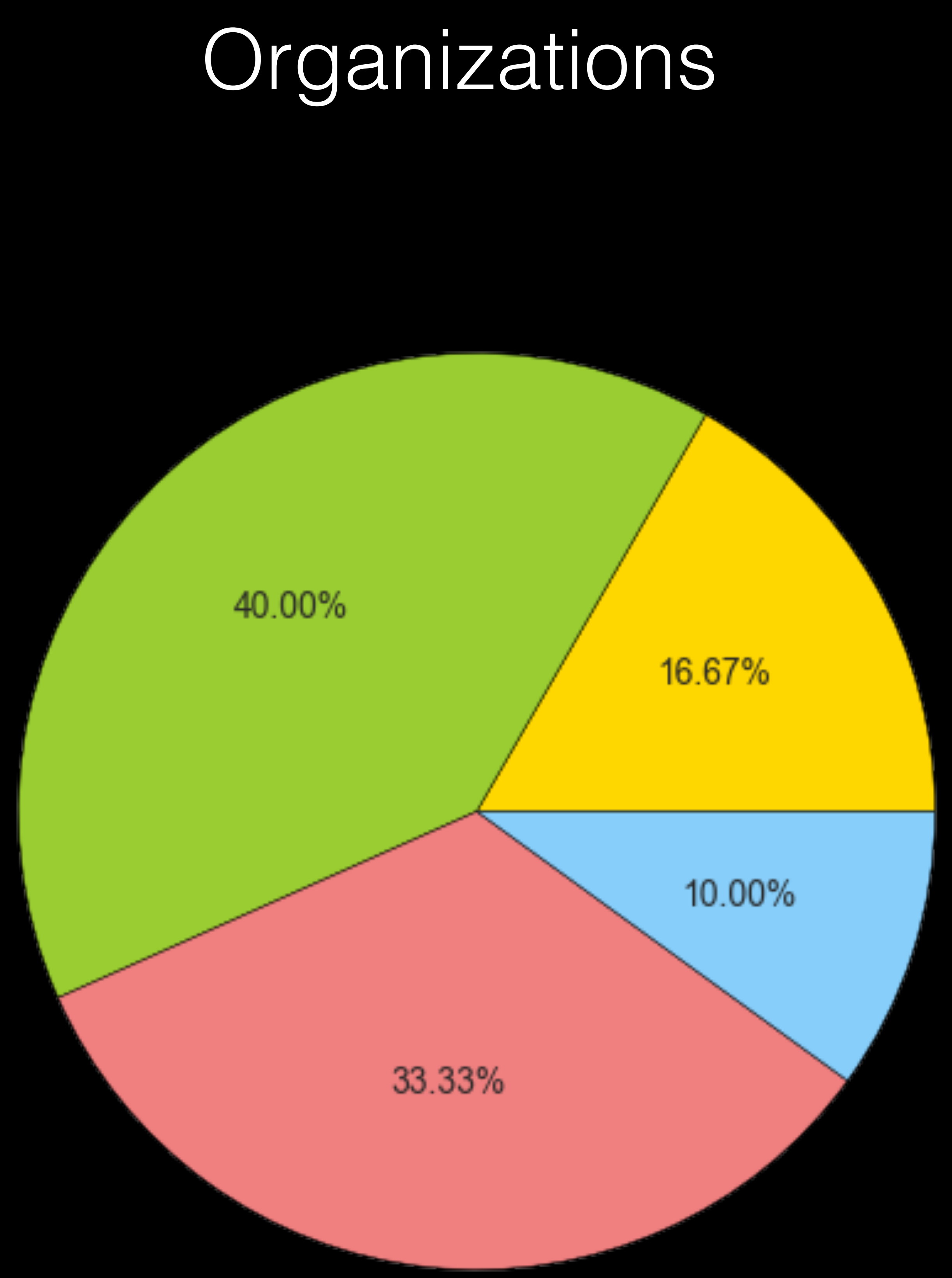
Active Contributors



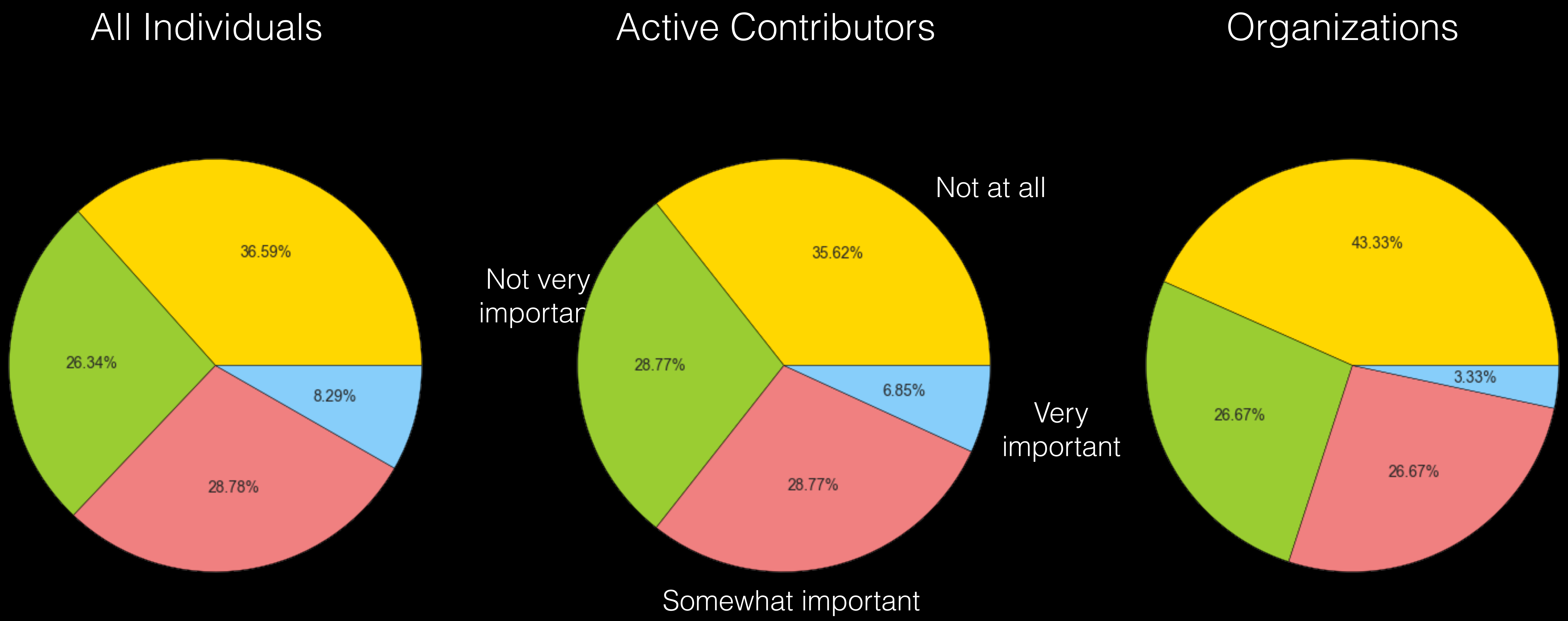
How important is it to have multiple projects in one repo?



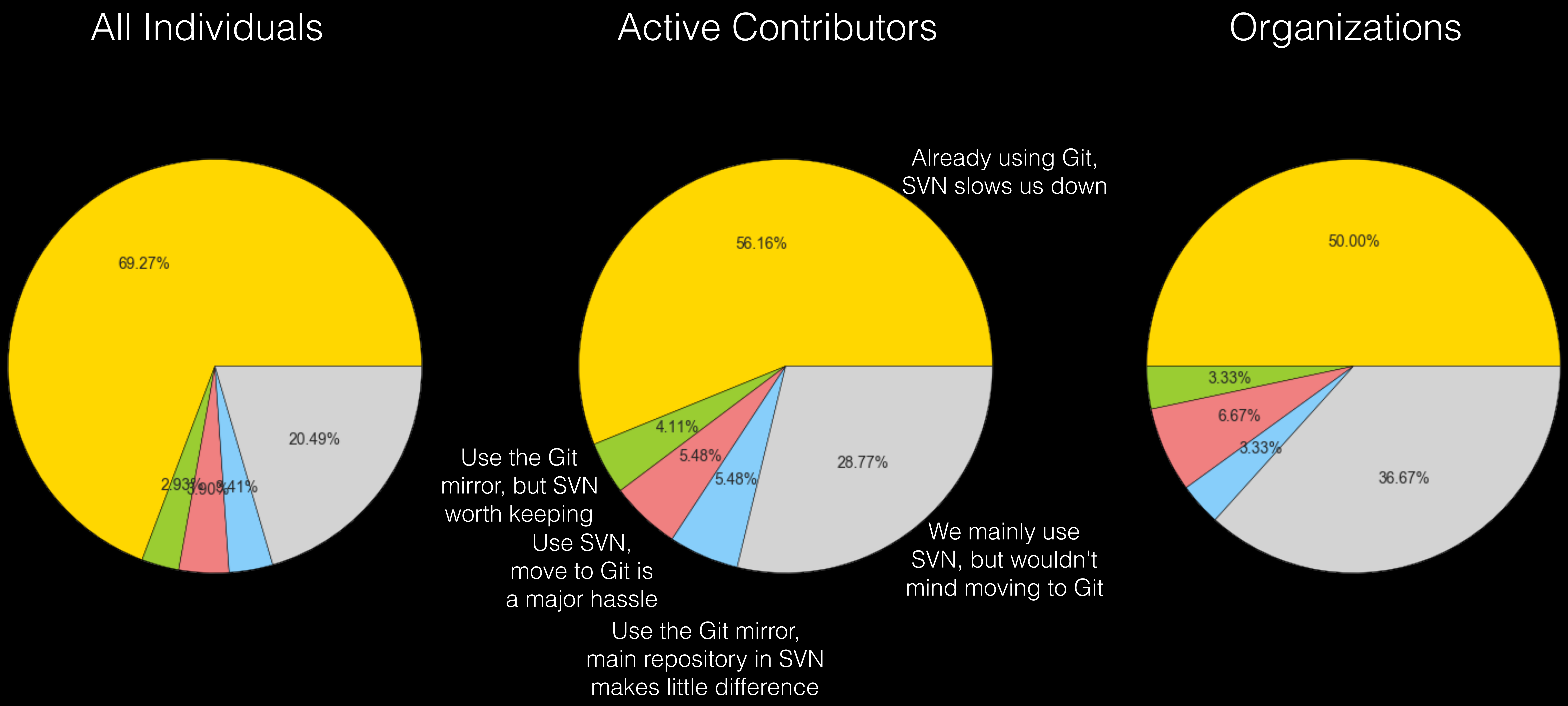




How important is it that we save as much space as possible? (clone/checkout)

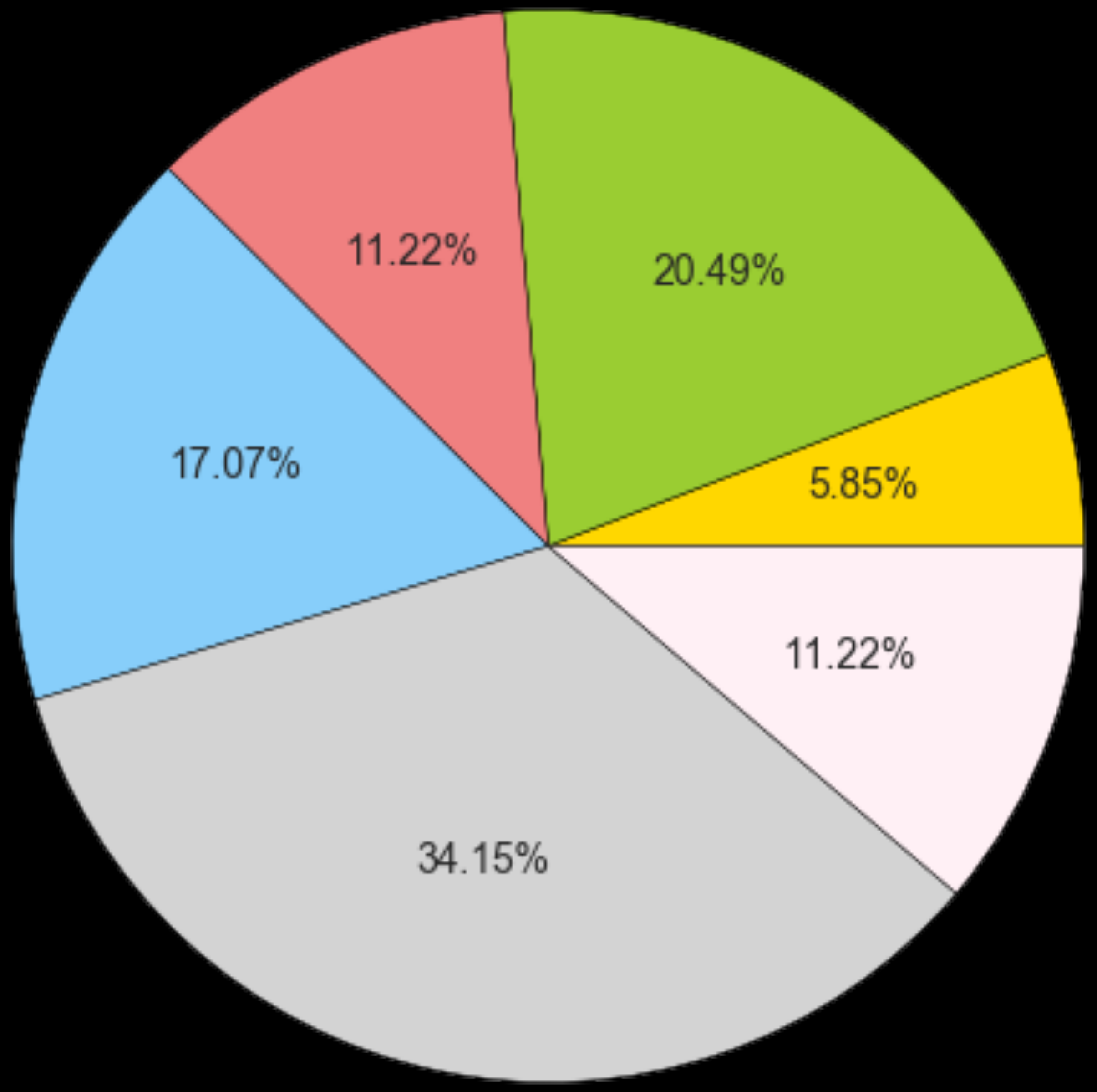


Do you believe the LLVM project would be better with a main Git repository, or keeping it in SVN would still be better?

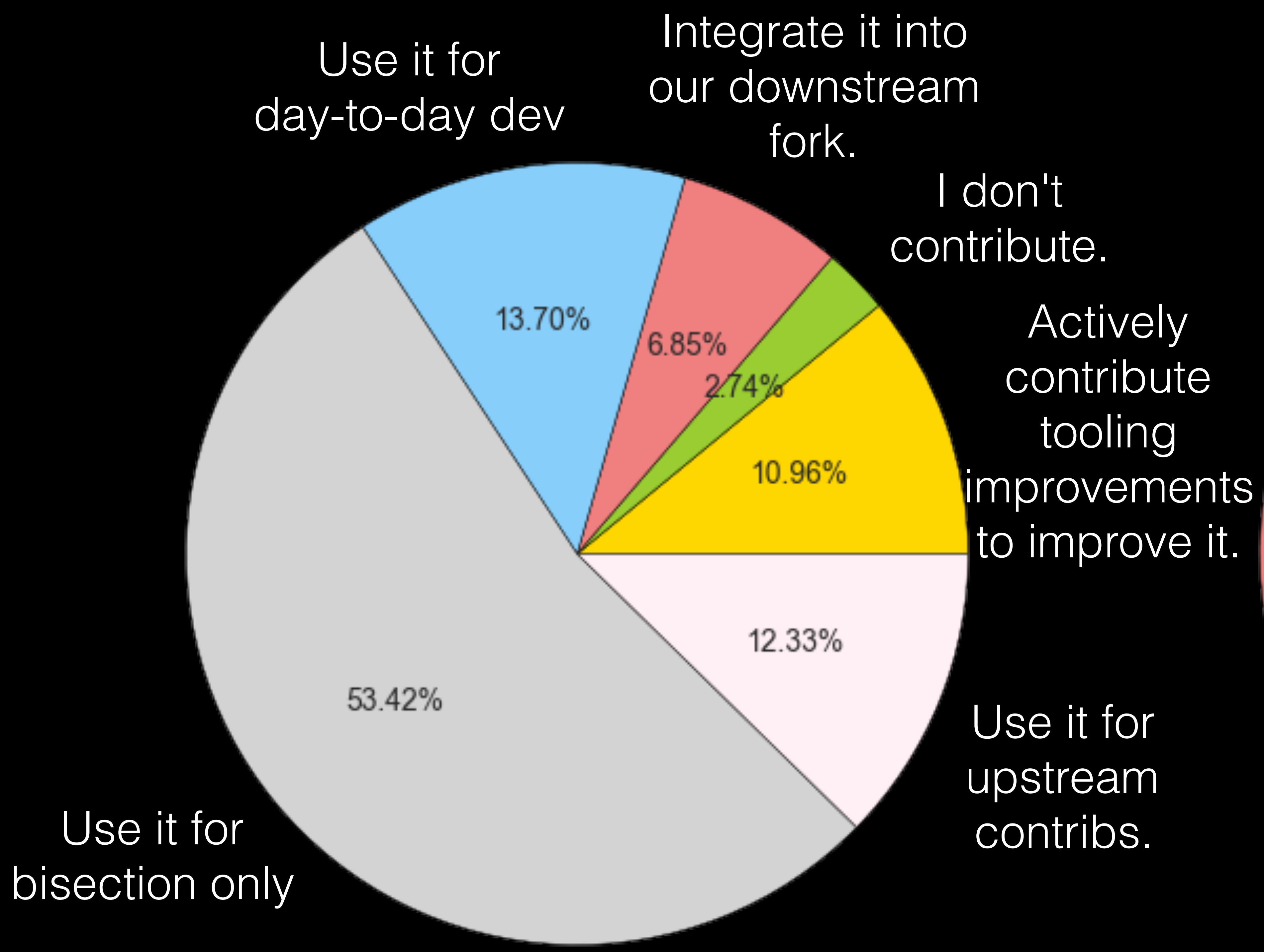


The multi-repo variant provides read-only umbrella repository to coordinate commits between the split sub-project repositories using Git sub-modules. Assuming multi-repo gets adopted, how do you expect to use the umbrella?

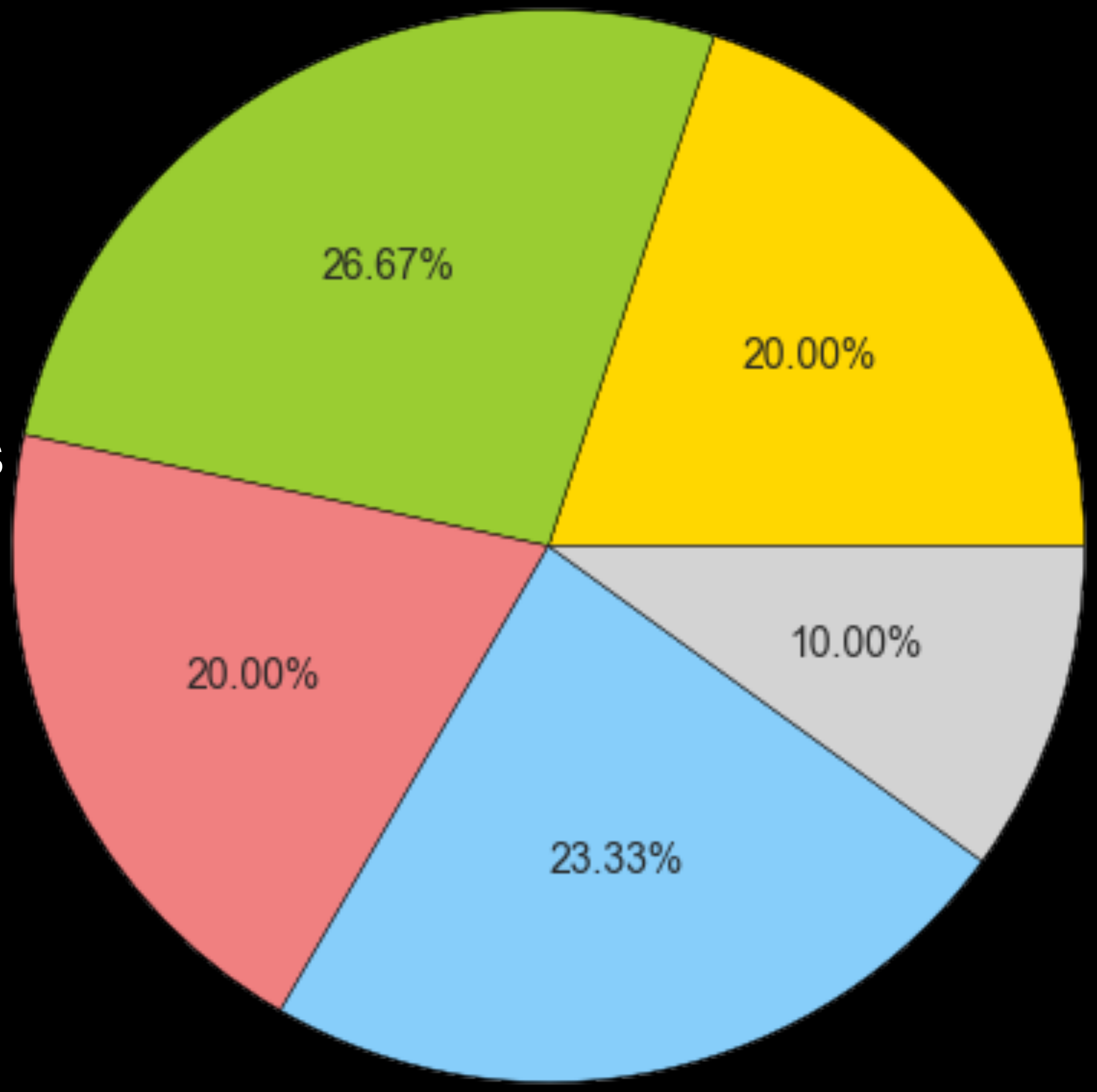
All Individuals



Active Contributors

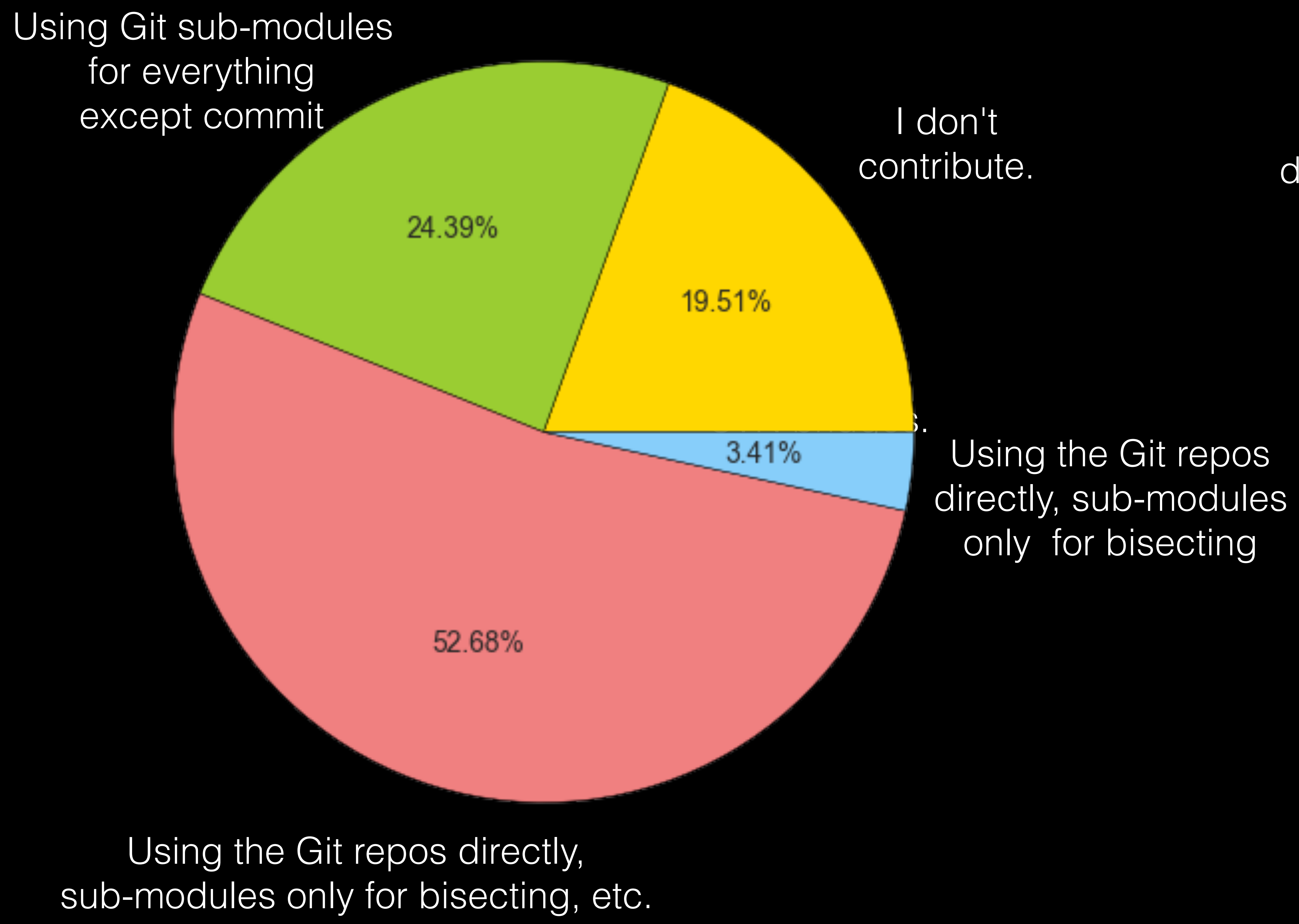


Organizations

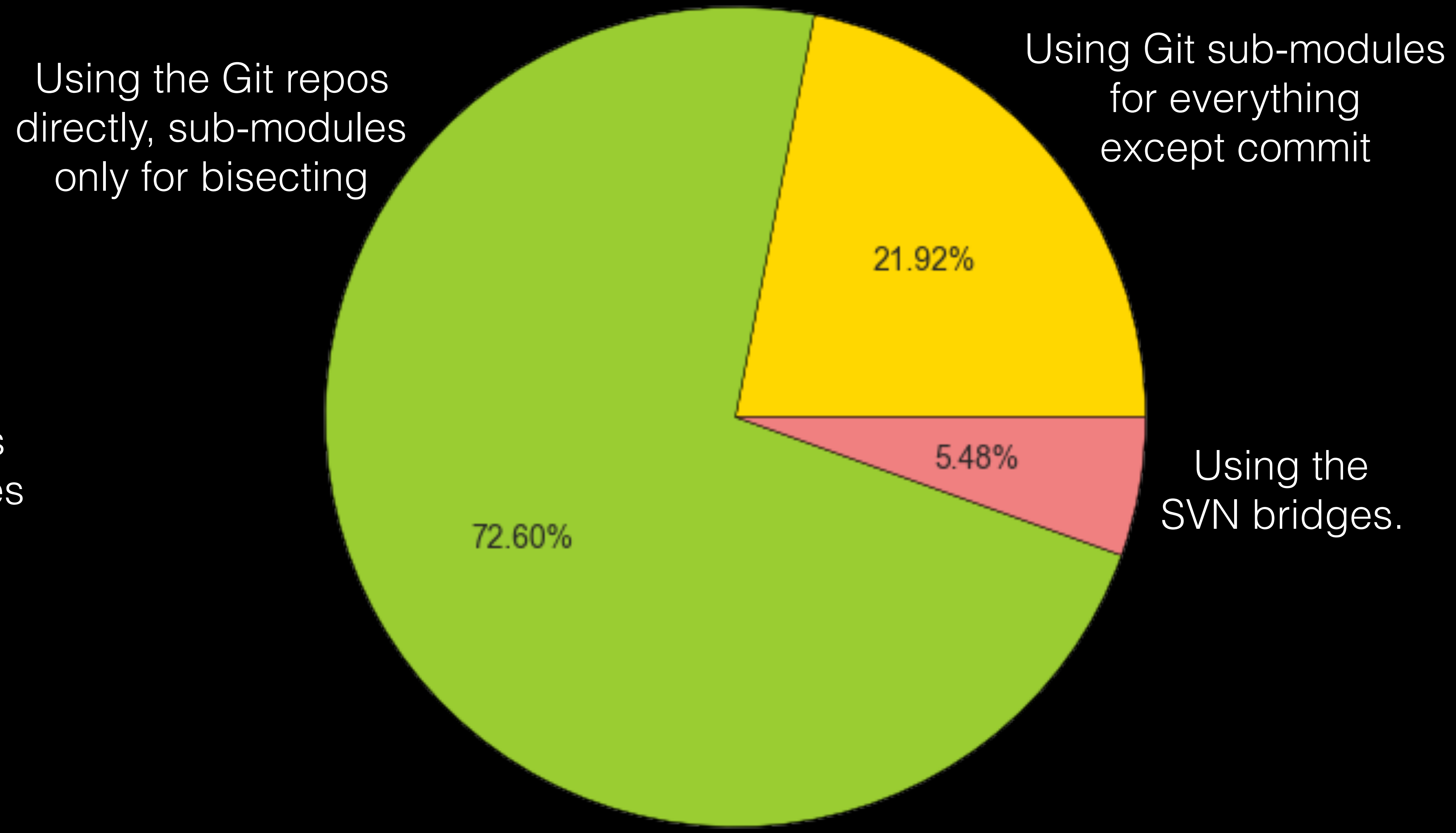


If multi-repo is adopted, how do you plan to contribute to upstream?

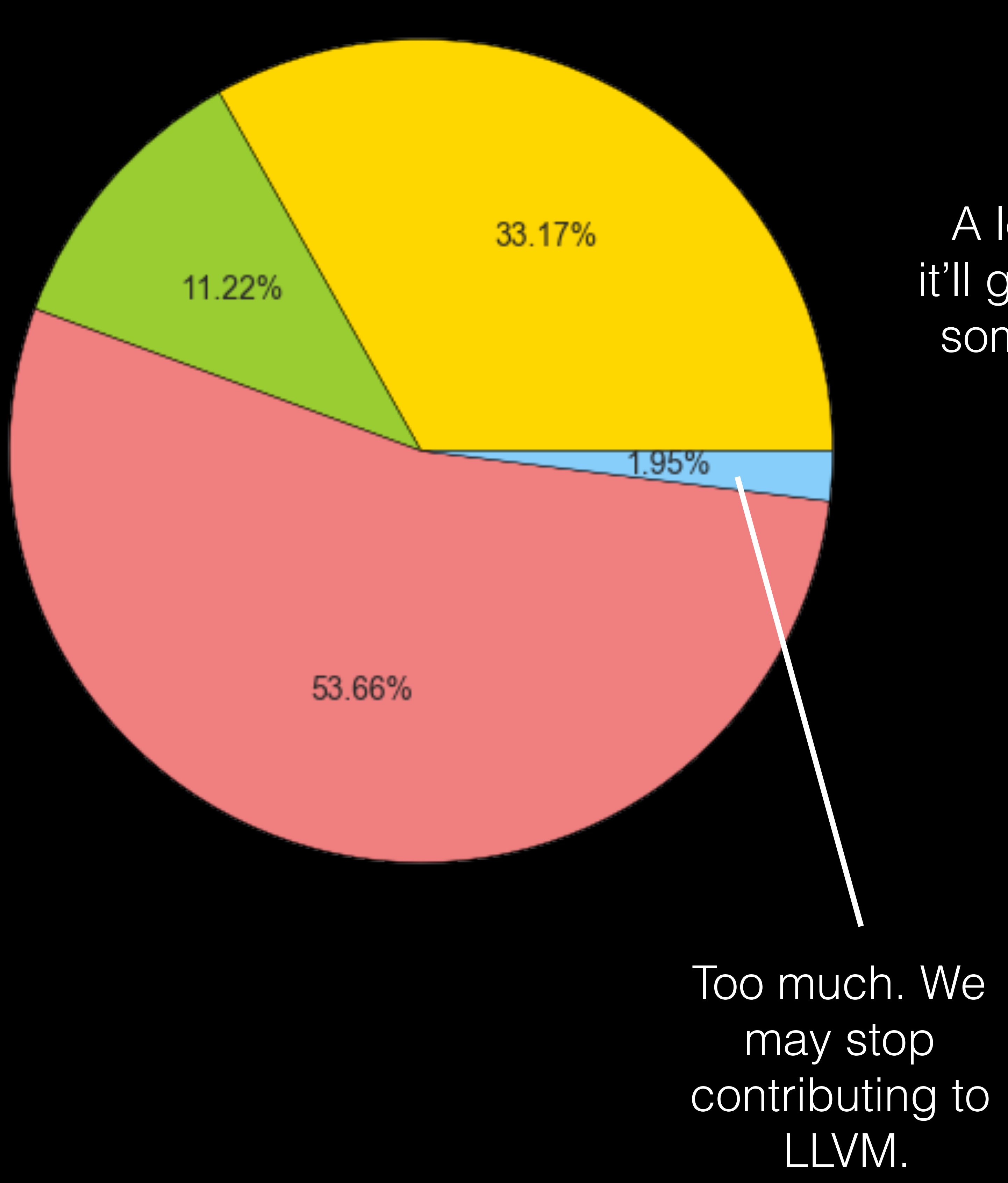
All Individuals

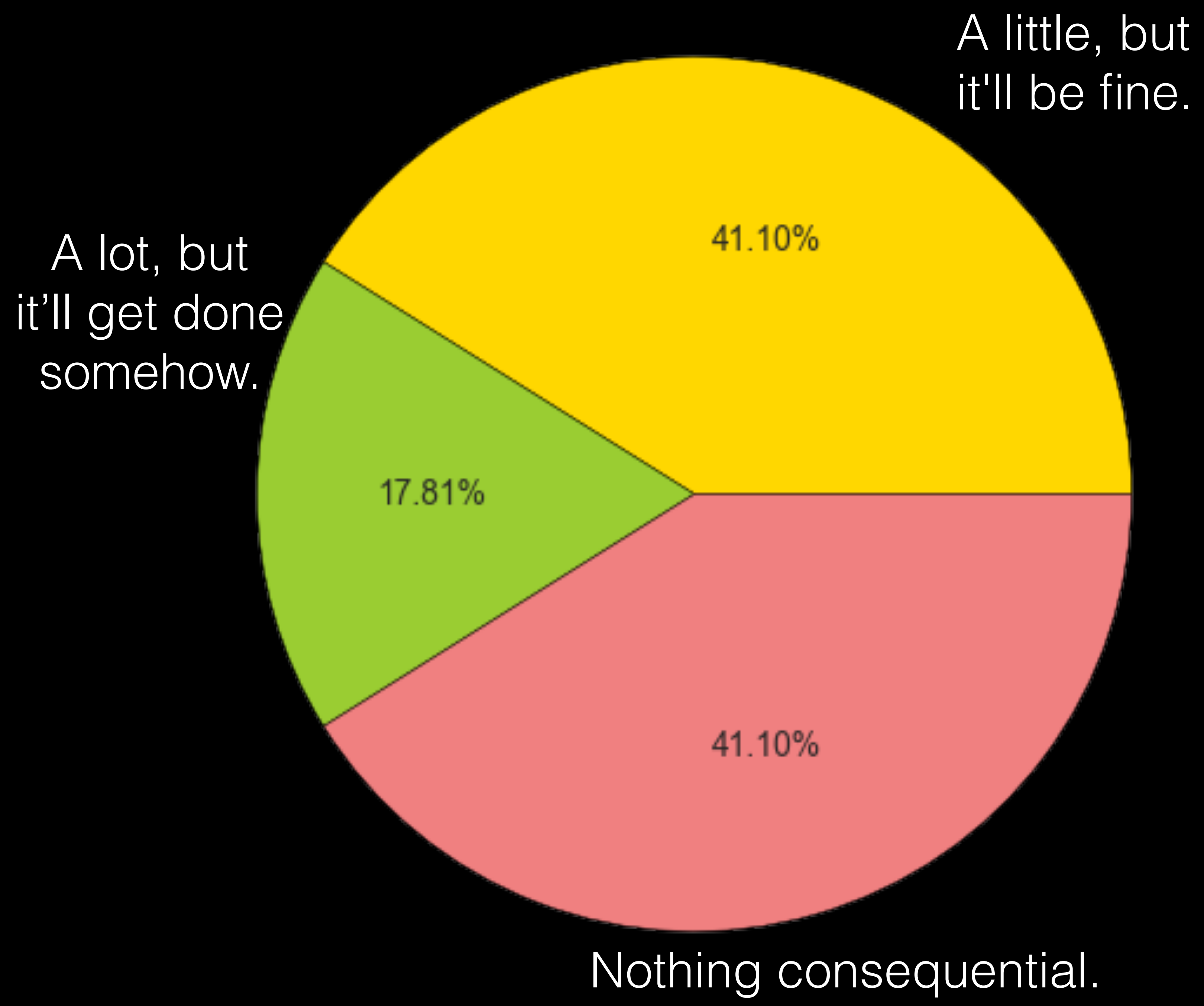


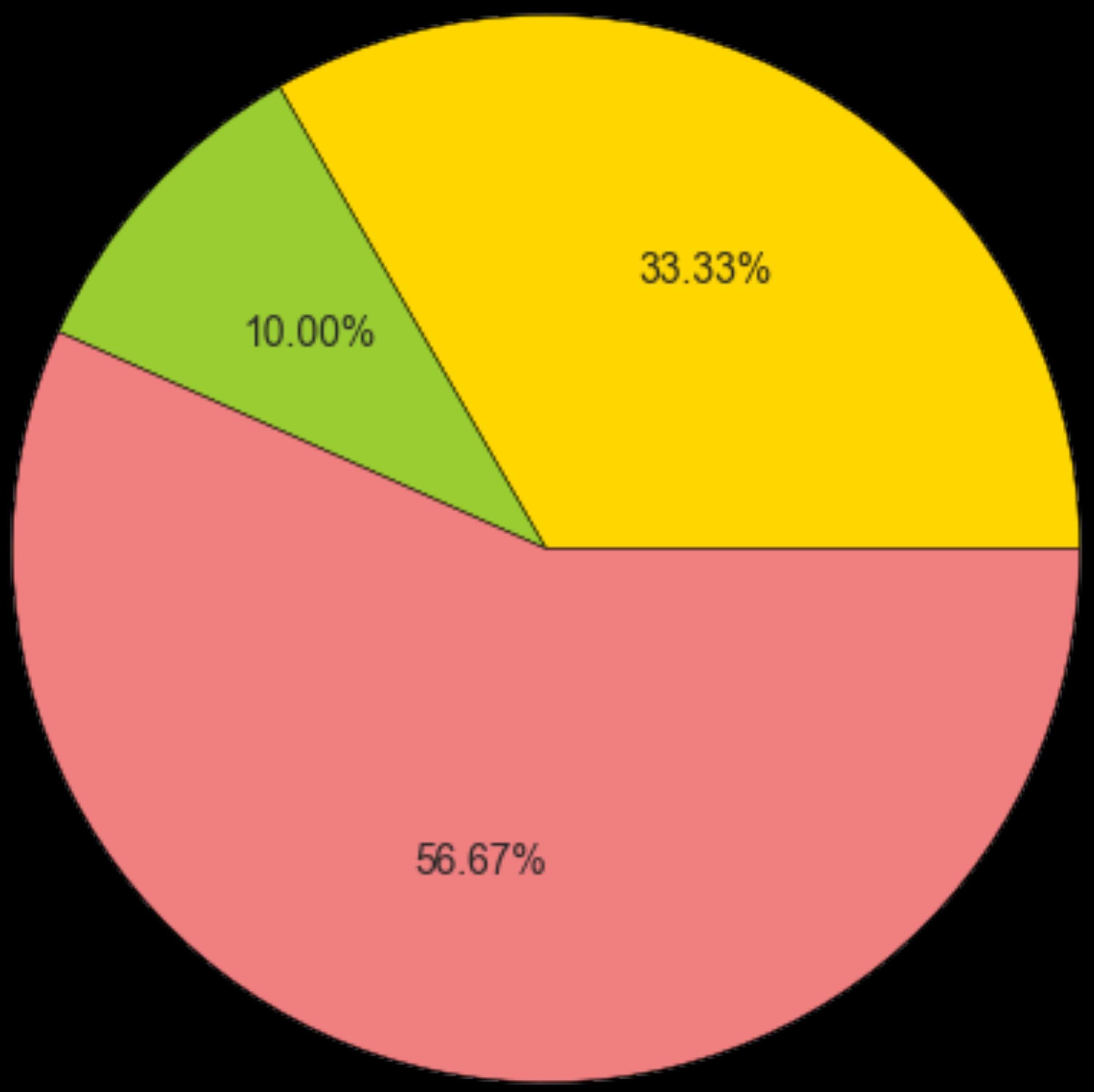
Active Contributors



If multi-repo is adopted, how much pain will there be in your transition?

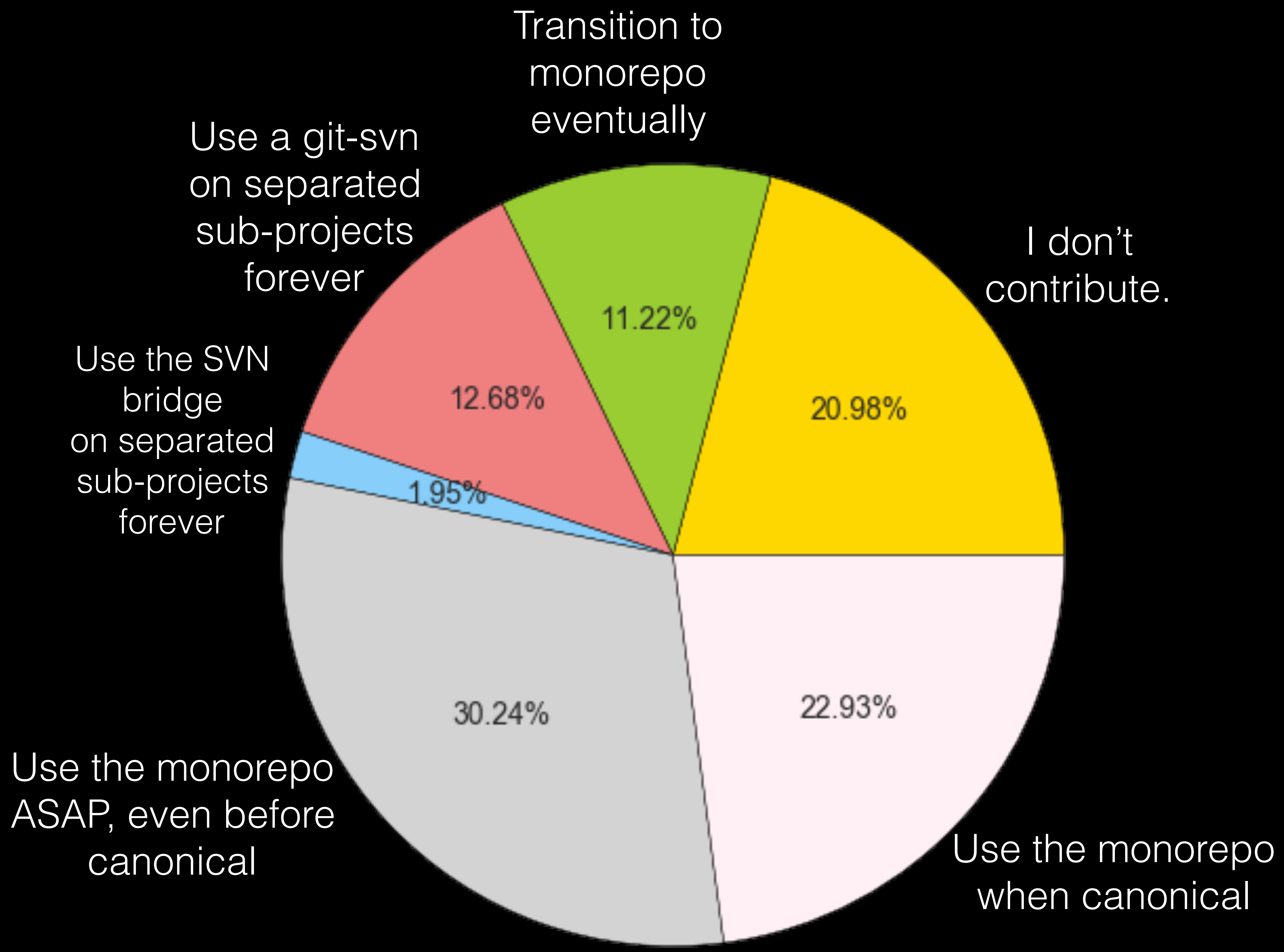




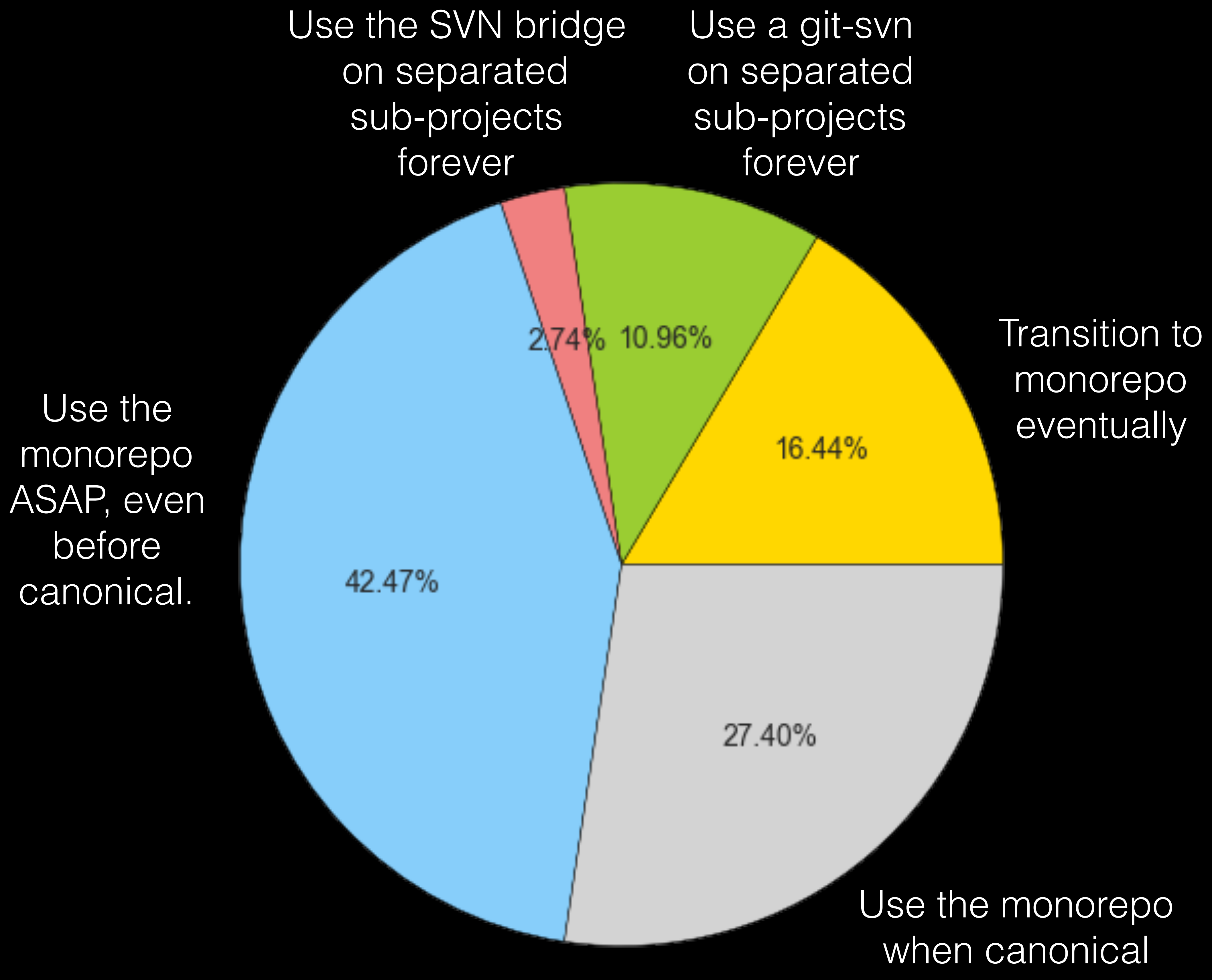


The mono-repo variant provides read/write access to sub-projects via an SVN bridge and git-svn. Contributors will have the option to continue using repositories split on project boundaries. Assuming mono-repo gets adopted, how do you plan to contribute?

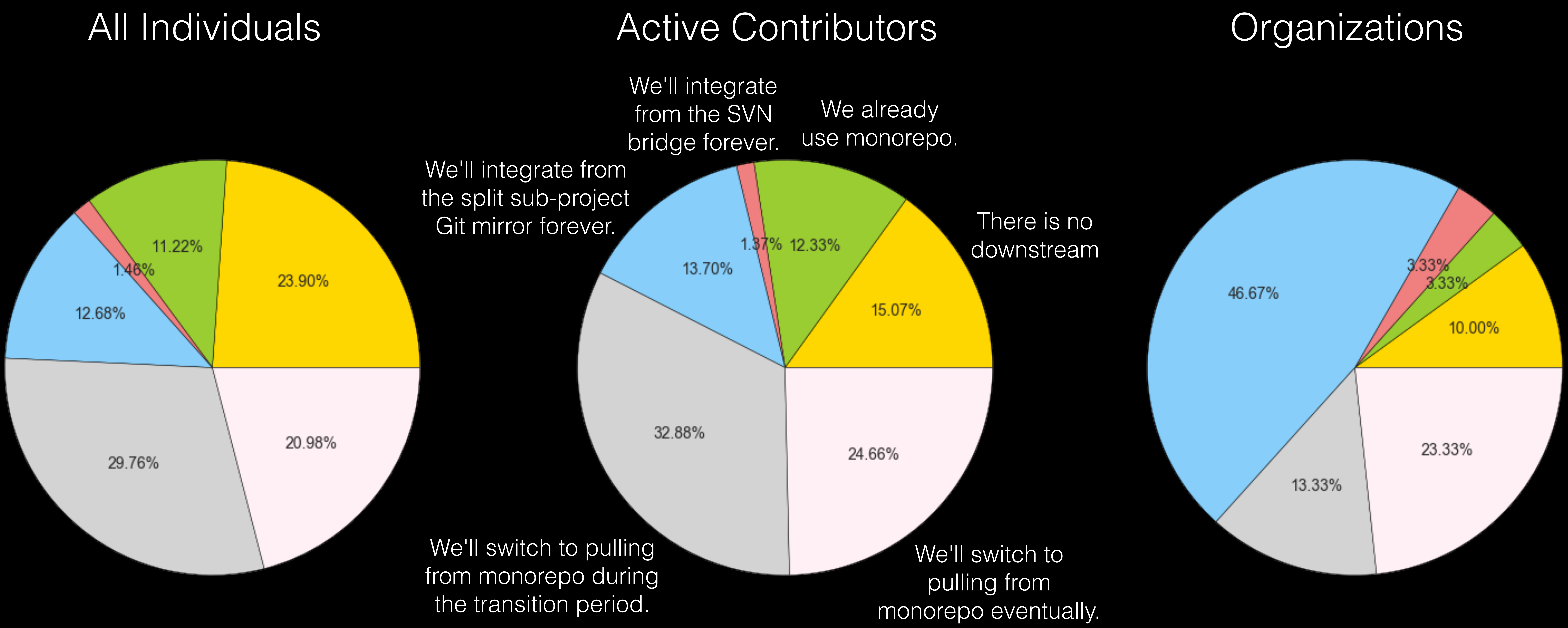
All Individuals



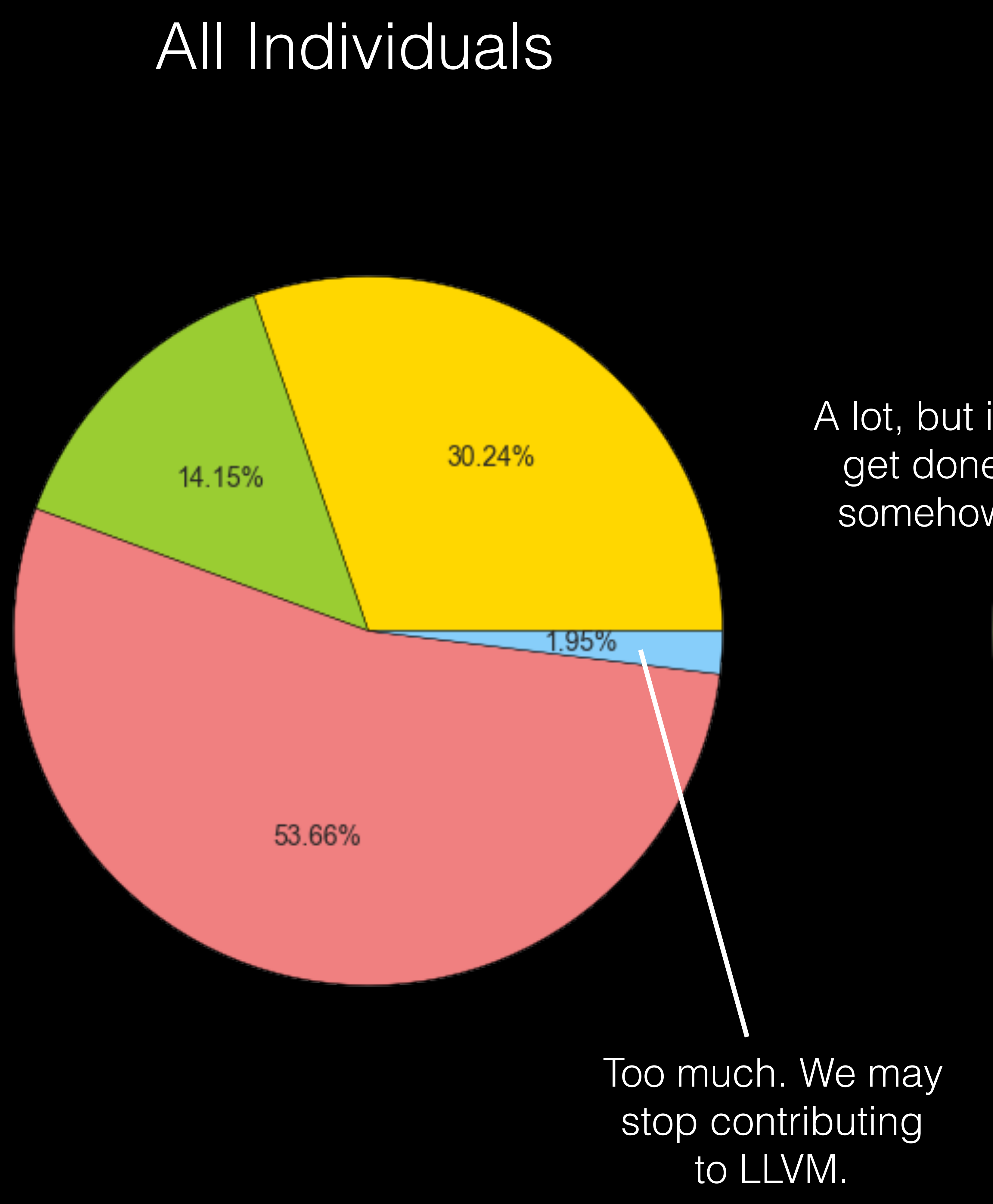
Active Contributors

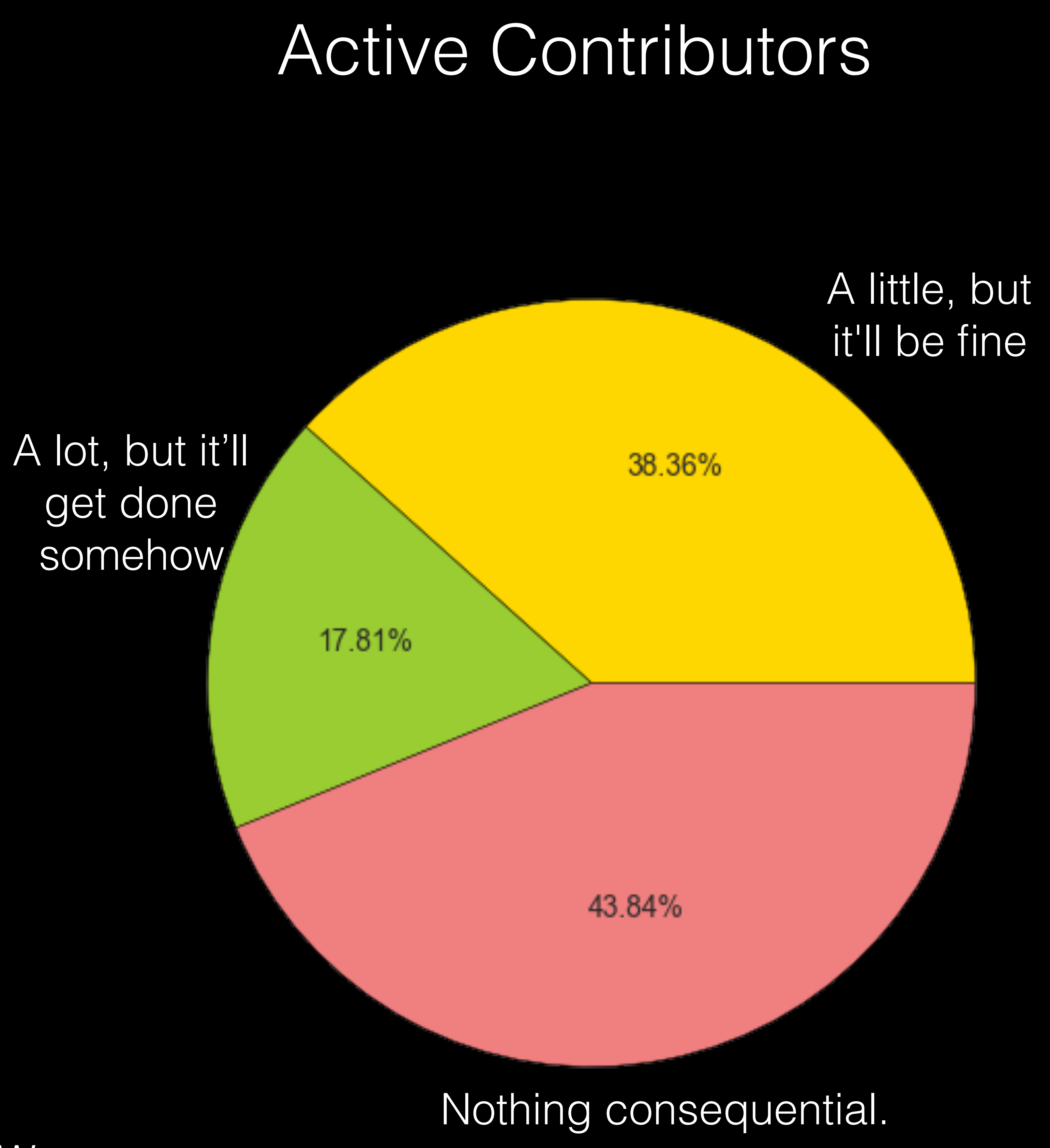


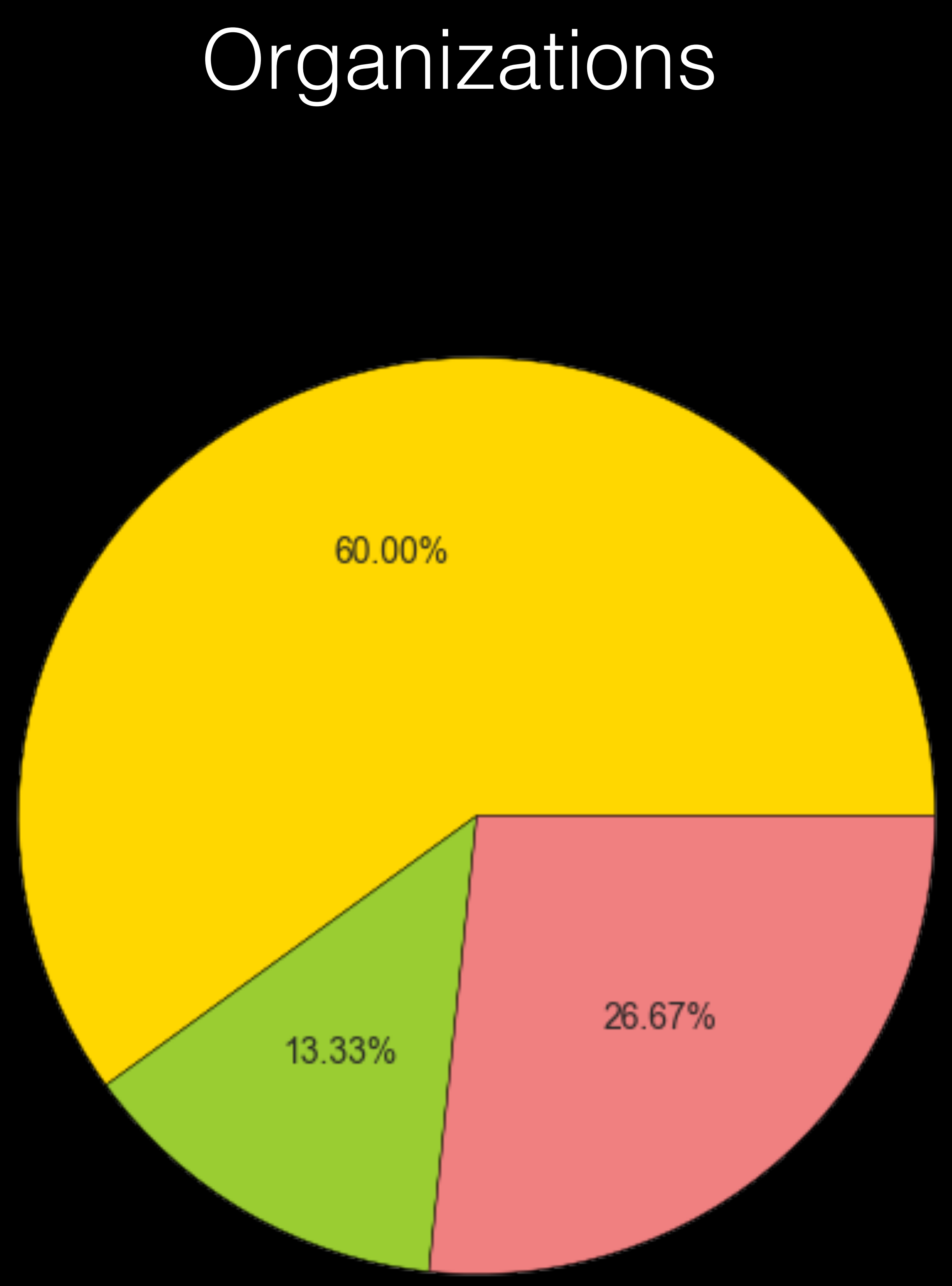
If mono-repo is adopted, how do you plan to integrate it downstream?



If mono-repo is adopted, how much pain will there be in your transition?

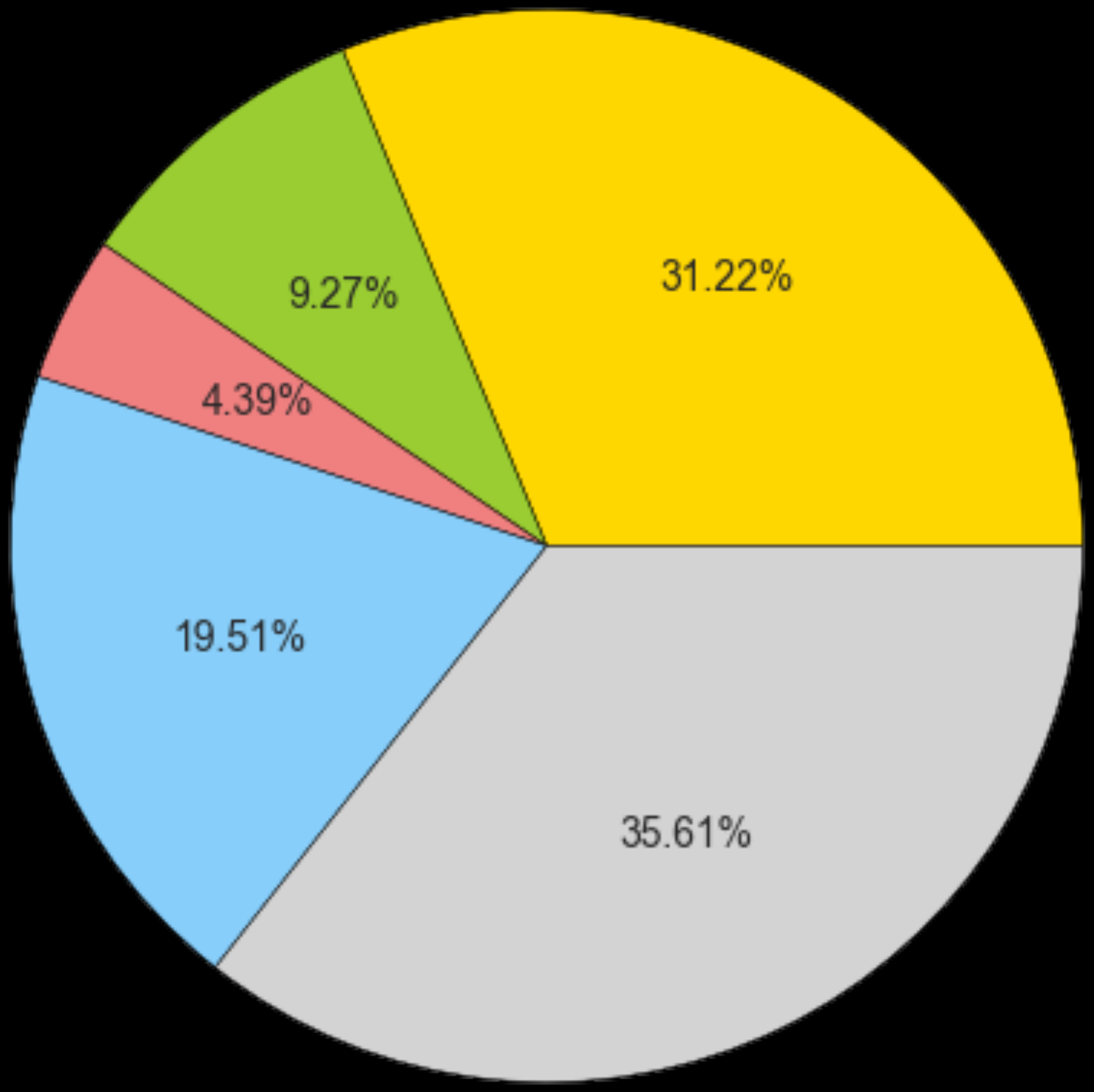




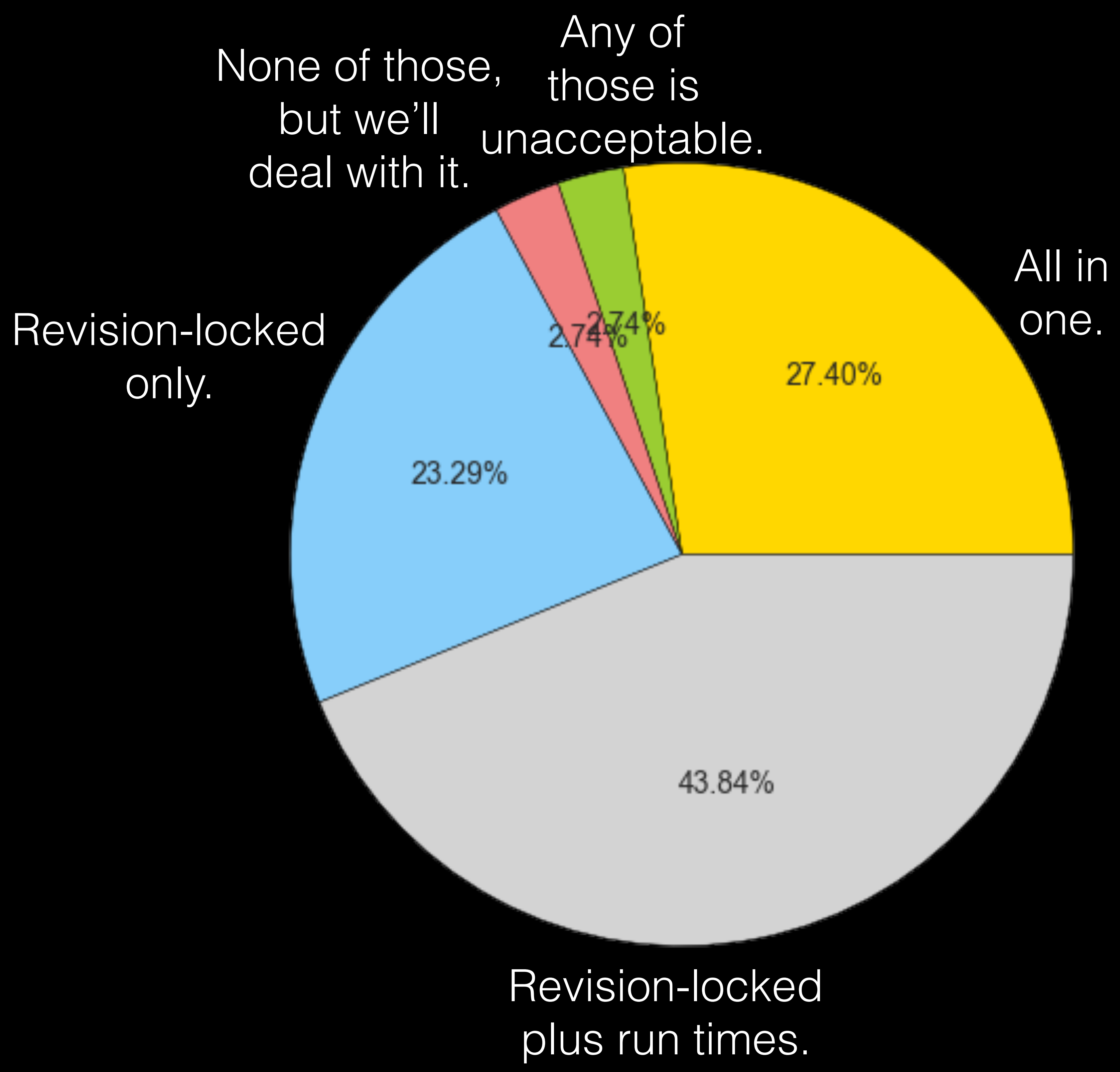


There were concerns over which projects could stay out of the mono-repo to make it more malleable. What would be the best ratio for you?

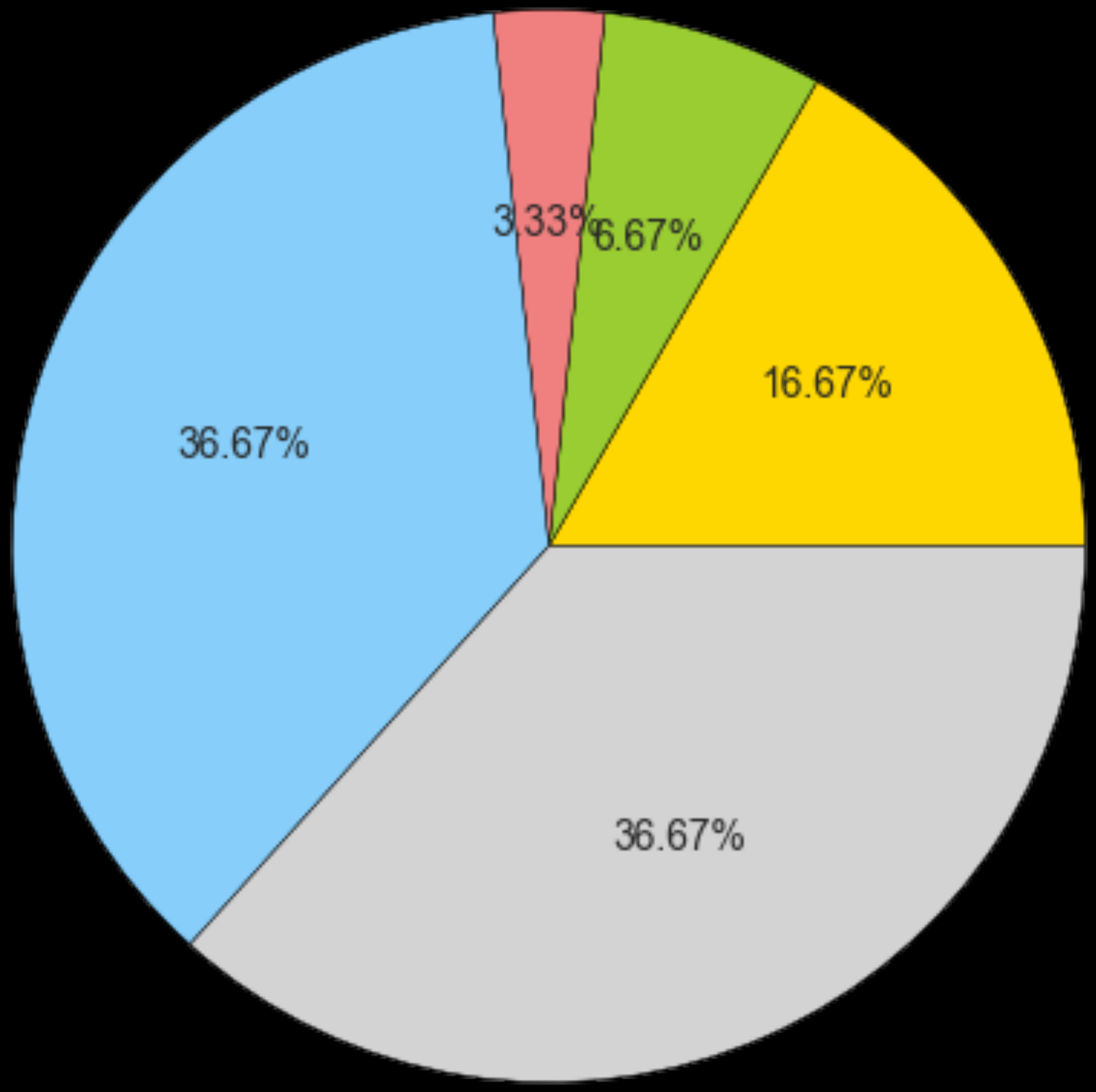
All Individuals



Active Contributors

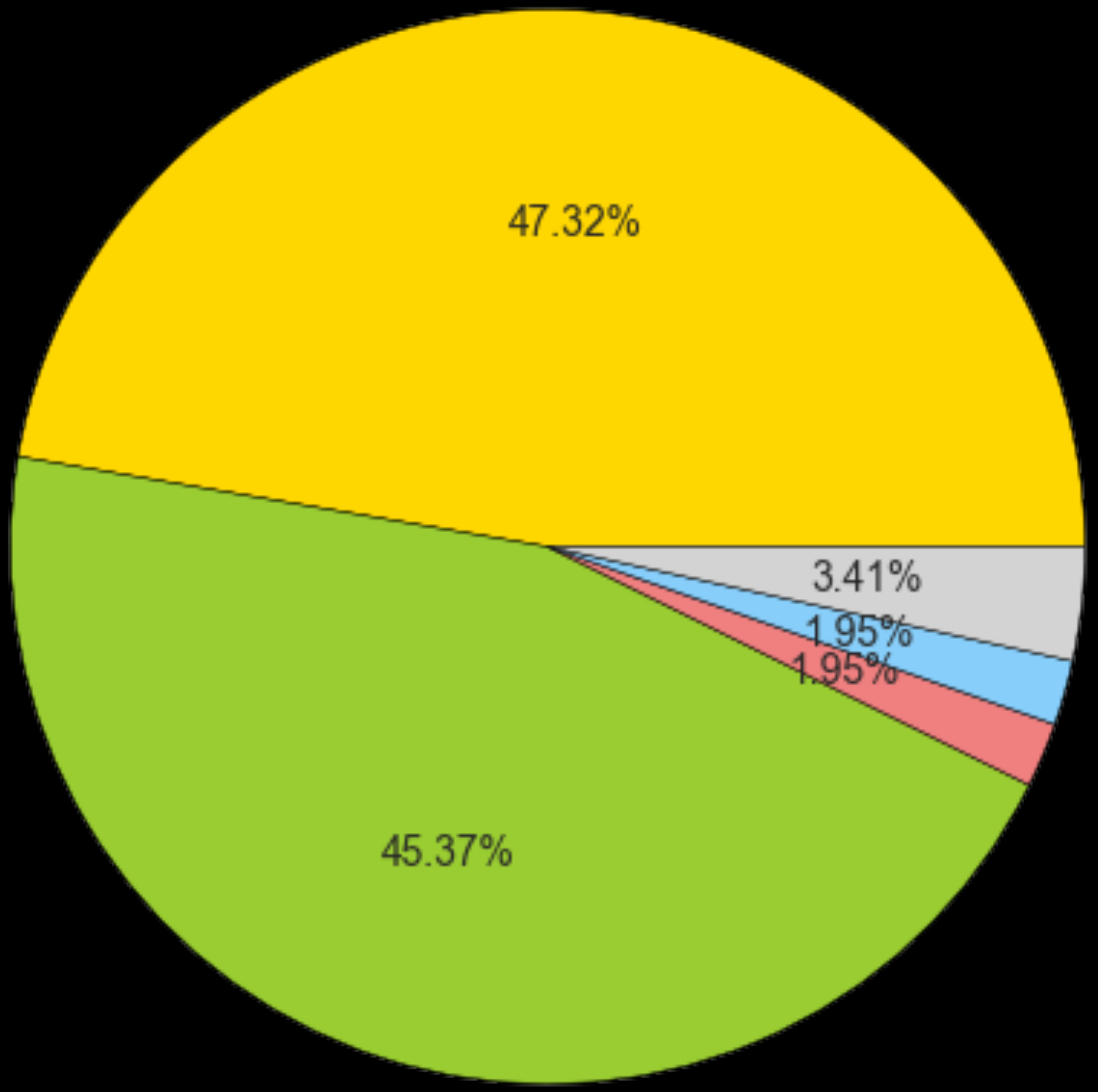


Organizations

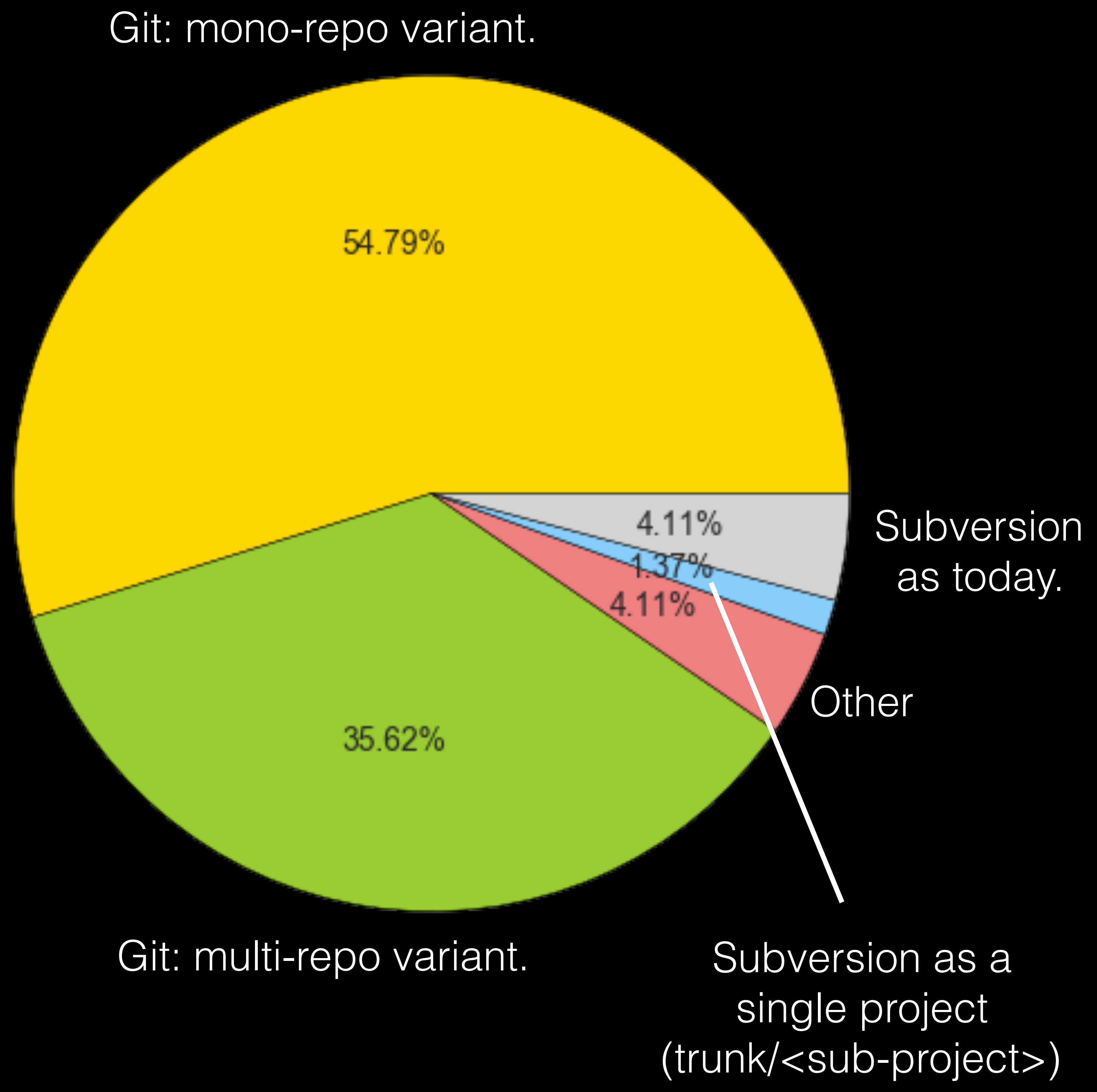


If we could go back in time and restart the project with today's technologies, which repository scheme would be best for the LLVM project?

All Individuals



Active Contributors



Quotes

About Git itself

The linear history that SVN provides is important. Git with a nice linear history would be fine.

We will lose integer revision numbers. Sad, but tolerable

The existing linear history is extremely convenient, and I hope it stays that way
(perhaps enforced by a pre-commit check, which can be done through GH integration).
Note: GitHub does not allow standard pre-commit hook

I have to learn to use Git.

I will not see any personal benefit from a switch, only more busy-work.

if version locked repositories aren't well handled, I'll be adamantly opposed to it. I cannot effectively work without that. Even when changes can be cleanly split between repositories today, the ability to monotonically apply them such that bisection and regression analysis works is *essential*."

As long as checkout + commit works via this git svn bridge idea I guess everything will be fine.

The most popular Version Control System on the planet will help people get interested in [LLVM] more

Quotes

About Git itself: is it worth it?

All of the tooling and CI infrastructure would have to be redesigned.
We'll need to re-write a some infrastructure to keep bisection over internal compiler artifacts working.
I want to avoid this. We would need to modify our daily update, build, and tests scripts. We really appreciate the ability to pin down changes to a specific numbered change/checkin.

All of our custom tooling will need to be revisited. That's acceptable.

Major changes to all of our systems will be needed to make this happen. Major changes to LNT, lab.llvm.org, Green Dragon, cloud bisection, plus many of Apple's internal systems. This proposal does not even enumerate the scope and depth of the changes needed to those systems. It won't "just work". There is months of man hours need to make this change, for what is minor benefit. Further, there is no concrete proposal for where the effort to do all these change would come from.

Quotes

About GitHub

I have not found the existing infrastructure's stability to be a pain point,
GitLab seems to open up more opportunities in the future

GitHub itself is not open source, which makes it less trustworthy,

Github checkouts are slower then the current git mirror hosted by llvm.

I would prefer self-hosted. it's damaging to Free software generally to have
so many projects cluster around a single provider.

GitHub is not "free" if you actually want responsibility for your data. If you cannot afford
to lose the repo, then you may need to pay for a commercial QoS.

I look forward towards the move to github however it might be nice to have
a daily-synced backup git repo on llvm servers just in case.

would like to keep URLs pointing at llvm.org (or a subdomain) so that they're in llvm's control so
we can minimize disruption in case we need to switch providers again.

It'll be great to get the official sources from Github.
No more looking around where to clone what and what is supported.

Good experience of GitHub with Boost repository.

GitHub is great! :)

Quotes

Mono-repo

For a long time these have been ""almost"" separate because of licensing issues. We're making great strides on fixing that and I'd like to avoid us inserting a new barrier in the form of repo boundaries. "

All of the mainstream stuff should go in the mono-repo, and the "special" stuff like test-suite, www, www-pubs, llvm-www-releases etc. should remain separate.

I feel that the most important question in deciding whether to go with mono-repo or multi-repo is: "Are these projects organizationally independent?" In LLVM's case, the core projects (revision-locked plus runtimes) clearly are not, so it does not make sense to put them into separate repositories either.

The main benefit for our workflow is cross-project code reviews in a downstream gerrit repository.

Quotes

Mono-repo: transition

We'll have to change the scripts around, treating the mono-repo as a single repository.

It invalidates all my clones. I'd have to dump any WIP to patches and rebuild it in the new repo.
That's a pain but it's only a one-time thing.

Mono repo will certainly take more time to migrate to but it is the better way in my opinion.
We can eliminate all the tooling for state-keeping
(one less thing to break, which I believe it may take some to stabilize).

This would require larger changes to our helper scripts, but in the end it will probably make them simpler.

Quotes

Mono-repo: simplifies day-to-day development

It greatly simplifies my work […], and should accelerate the deployment/evolution of [my work] out in the open if we have a single repository where synchronised changes across the runtimes happens.

I favour mono-repo: already all too often I screw up and have incoherent projects, or "ag" from the root forgetting that it ignores tools/ sub-projects. For sure I can work this way but it's a source of paper cuts that I'd rather live without. Disks are cheap, networks are fast, but sadly my brain is ever slower so I want to optimize for that.

Monorepo makes my life easier and saves me time where it matters (frequent checkout/log/blame/commit/bisect/branch switch) and I'm willing to pay with extra space and checkout time for that.

I am so sick of setting up everything to work across 5-7 repos.

It allows easy bisection and blame without the faff of an umbrella repository.

Quotes

Mono-repo: better cross-project integration

I think that the mono-repo proposal better suits our current and future requirements. Disk space is not a big issue […]. I think that with the mono-repo solution we will be able to better track live LLVM changes in our out-of-tree implementation than we can currently with SVN or with the multi-repo Git proposal.

Mono repo provides an unified environment for all the LLVM developers and it is easier to maintain. It also encourages code sharing, encourages improving API to benefit other LLVM projects and helps better integration between projects.

"I believe mono-repo will be a boon to the LLVM project for a variety of reasons. There's one part in particular which I think is not emphasized enough:
I believe putting all the projects into a mono-repo will allow the more-independent subprojects to depend on ""LLVM"" itself *less* than they do today, by making it feasible to factor out common infrastructure into a separate subdirectory, without it being a major pain to do so.

Quotes

Mono-repo: other consequences

We have some workflows that would be much more efficient in a mono-repo scenario.

Monorepo would make maintaining our current multi-repo automation *much* simpler and less error prone.

Apple downstream internal CI will be simplified a lot [with monorepo]

The cost of a mono-repo checkout only needs to be paid once; any other checkouts can use git clone --reference.

100% in agreement, this matches my existing workflow very closely and I don't foresee any major disruption at all.

I'm already using monorepo, and it simplifies my day-to-day development.

I could get rid of my "create mono repo script" \o/

Quotes

Mono-repo: concerns

Mono-repo would discourage contributions and work on projects that use LLVM by student and hobby developers because it focuses on too many things in one main repository.

I don't think there is an acceptable combination of projects.
Putting all of them in there makes the repo too large.

We have a hard dependency on LLVM and Clang, but nothing else (~compiler-rt),
so the size of checkouts could become a problem here.

As you accrete code and projects, the mono-repo approach won't scale well across a number of dimensions which is who I'd go with the multi-repo approach.

Quotes

Mono-repo: concerns

It's not clear how the mono-repo direction naturally scales for the inclusion of more projects into LLVM. It does nicely solve some revision-lock issues with LLVM and Clang, but if another sizable project like Clang came up that was not originally part of the mono-repo it's not clear what is the path forward with integrating that project into the mono-repo. LLVM existed for a while before Clang showed up, and it's not clear if the mono-repo is meant to include everything, especially if the "all in one" approach is not taken.

The mono-repo forces all the projects to be handled as a single block. This introduces unnecessary dependencies which currently are not as strictly enforced.

I'm worried about the single-repo git, where I won't be able to update (git pull) compiler-rt only.

if you have the foresight to see a project as long-living it makes sense to modularize because Git has known scalability issues with large projects. There are several algorithms in git that are O(n), and as the number of commits increases the performance dramatically decreases. One such algorithm is "git blame."

Quotes

Multi-repo

This proposal seems to be exactly like what we have now with git-svn (except committing is slightly simpler, and the tooling to create useful revision numbers exist is somewhat harder) and have been working with successfully for at least five years

Note: multiple comments not repeated here were along the same line of "close to what we have now with git-svn"

I belief single [GitHub] org[, and] multi repo is the best approach for us.

I prefer the multi-repo proposal over the mon-repo because it scales better in several ways.

I would probably go as far as to split the multi-repo further, as some LLVM components seem useful elsewhere

Quotes

Multi-repo: Umbrella

Wary of the bot merging the commits into the umbrella repo. Not having guarantees on the merge + possibility of the bot going down and making a mess when catching up.

My main concern is that Clang would now be versioned by a tuple of sub-project versions. I guess the umbrella repo might address this, but it seems a bit convoluted.

The umbrella repository seems something that few would bother to use. It only helps people doing local investigation

I consider the umbrella repository a non-solution - I highly doubt anyone would bother to change their workflow to incorporate it.

the multi-repo seems too complex. It's better not to invent custom tooling for a problem that isn't unique to LLVM.

Quotes

Multi-repo: day-to-day development

I'd need the umbrella, if only to get a consistent full-project checkout.

I mainly know that I'll mess up submodule state from time to time (using it on other projects), which is a bit of a productivity waste. If it provides enough value for others, I'm fine, though.

Multi-repo is just more moving parts. I find that I wind up forgetting that I did something in one repo, and sometimes screw myself up for hours (the next week, generally) before I recall what I did.

There is no easy way to browse combined history from multiple submodules that I'm aware of.

I fear the difficulties in backing out committed changes that later turn out to have problems.

Multiple repos add another mental dimension making it harder to set up llvm and do small contributions

Quotes

Multi-repo: submodules

I'd really rather not have to deal with git submodules during local development.

After having tried to use it for a while on a previous project, I don't consider git submodules to provide a usable developer workflow,

Experience with Boost is that submodules work OK (are a steepish learning curve).

Do not fear submodules; they work. Just set up the necessary tooling to make them work smoothly.

Tooling support for submodules is generally poor, both in core git, and git integrations.

Submodules are not ideal. A tool such as Google's 'repo' would be highly preferred.

Similar to what you've proposed, but instead of submodules,
I'd consider using subtrees instead (see git-subtree(1)).

Quotes

Multi-repo: submodules

In my experience, multi-repo / submodules are only really useful for disk space and bandwidth saving reasons

Submodules make atomic commits unnecessarily failure-prone. In fact, it is hard for me to imagine a robust workflow that permits them,

Quotes

Multi-repo: concerns

I think this is strictly worse than the status quo and inferior to the mono-repo proposal.

I am extremely concerned about whether LLVM can be effectively developed in this world. I mean, I'll find a way to get by, but without monotonicity, I think this will be a source of constant and painful problems for me.

The multi-repo is closest to the current situation so very likely to have less pain in transition for most. But I still think it's a worse position to be in.

I would hate to be in a situation where we'd have to describe a Clang version as (LLVM hash1 + Clang hash2 + compiler-rt hash3 + ...)

I never understood how to manage multi repo stuff in git.

We can make it work, but the multi-repo perpetuates longstanding problems for us.

would prefer a simple solution, generally.

Quotes

Multi-repo: regression for existing “monorepo” users

It seems like a productivity regression to me. I've already gained a significant productivity boost by adopting the mono-repo mirror (https://github.com/llvm-project/llvm-project) and jlebar's scripts for committing to SVN and the multi-repo proposal seems like a step back to the individual sub-project mirrors.

In order to maintain the same level of productivity I would probably need tooling to transform the umbrella repo into a mono-repo for local development. In the end it seems we will either be stuck with the productivity regression or will have to maintain two sets of bespoke tooling when the whole point of the move is to avoid having to maintain custom tools/services."

If the multirepo is adopted, I will write a script to make a non-canonical monorepo mirror, like we have today. It will be a pain, and it's likely that the trend of continuing to break the tree with cross-cutting changes

Quotes

Multi-repo: submodules layout (nested alternative)

I'd actually prefer proper hierarchical submodules with clang in llvm/tools, but that's not an option here. If we need to contribute to git to improve this, we should do that.

We would be using git submodules properly if it would be up to me (not in the meta-repository way).

I'd prefer proper hierarchical submodules.