Vectorized general-purpose utilities: how to handle?

= gathering opinions and ideas for a more general discussion =

The context

- Library-driven vectorization more and more popular, proliferating into several packages
    - ROOT, VecGeom, GeantV now, soon maybe many more + user code
- Community still in doubt of what to adopt (Vc, VecCore, …) while waiting for more general solutions (C++20)
    - VecCore has the advantage of being type-less (typedef to the desired backend)
    - Other benefits (e.g. supporting scalar+vector kernels, not discussed here)
- Every package developing their own solutions for vector-aware utils but also ancillary data structures
    - Math, basic algorithms, RNG, physics types/operastions, SOA types, …
    - The seed for work duplication and/or spaghetti dependencies

The problems

- We’ve started developing algorithms depending on such utilities
  - Physics vectorization in GeantV needs vectorized/scalar RNG, in future Lorentz vectors and boosts, rotations, vector algebra, …
  - RNG hosted temporary in a branch of VecCore, cannot merge to master since the place might not be appropriate
    - Result: the vectorized physics branch not yet mergeable to GeantV master, making further developments harder.
- While including optimized Log/Exp headers originated from VDT, but from Geant4, we realized they were under LGPL license
  - How do we handle code imported and optimized from third-party libraries and having a different license?
  - Should we handle externals at this level?
- This is just the tip of the iceberg…
  - Utility code (“kernels”) migrated to VecCore may have have interfaces based on custom containers, e.g. vecgeom::Vector3D/SOA3D
  - Scalar vs. vector implementation support: these utils will need a compatible scalar version as well
- Sounds familiar? What should we do?
  - Doing good-old in-house cooking and push the problem to future generations???

Discussion

- A common namespace (e.g. VUtils::)?
- Which git repo?
- Header-only?
- Externals?
- Content? Granularity (many small libs against a bigger one)?
- Basic vector types & containers?
- Scalar & vector support?
- Precision-related issues (approximations, float vs. double)
- Evolution (migration towards C++20), maintenance
- …