DART Tutorial Section 2: The DART Directory Tree
Much of DART is implemented as Fortran-90 modules and programs.

DART also contains:
• Documentation (really!),
• Namelist control files,
• Compilation tools,
• Shell scripts for managing large applications, and
• Diagnostic tools.
DART Top-level directory structure

DART/

- models/
- observations/
- assimilation_code/
- documentation/
- diagnostics/
- build_templates/
- README/
- CHANGELOG/
- Others ...

Peruse your DART subdirectories!
DART models directory:

- lorenz_63/
- lorenz_96/
- utilities/
- model_mod_tools/
- cam-fv/
- POP/
- wrf/
- template/

Many others models

Contains a code ‘template’ for adding new models.
Example DART model directory detail:

- lorenz_96/
  - model_mod.f90
  - model_mod.nml
  - model_mod.html
  - shell_scripts/
    - Model-specific scripts. None ‘required’.
  - tests/
    - Optional. Not required.
  - matlab/
    - Model-specific functions to aid analysis, if any.
  - work
    - *Where all the action happens!*

Various input files...

Anything with this background is optional.
DART model/work directory details:

Executables are built and run in model work directories. Makefiles and compiler output files reside here. Input and output files generally reside here. Lots of other junk files tend to accumulate here. Check out contents of `models/lorenz_63/work`.

- `mkmf_xxxxxx`: files that control what compiler is used, compiler options, etc. – for program `xxxxxx`
- `path_names_xxxxxx`: files that control what source code files are needed for program `xxxxxx`
- `input.nml`: file used by all DART programs for user control
- `workshop_setup.csh`: script used to run ‘set’ experiments for some workshop exercises. Not all models run workshop experiments.
- `quickbuild.csh`: script used to compile ALL applicable DART programs for this model. Feel free to take a peek, but no need to understand the details.
- `obs_seq.out.xxxxxx`: Sequence of observations to be assimilated for case `xxxxxx`
DART Fortran-90 code comes as code, documentation, and run-time control files. For instance, the directory `assimilation_code/modules/assimilation/` contains the following three files that implement localization (more on this later).

- `cov_cutoff_mod.f90`
- `cov_cutoff_mod.html`
- `cov_cutoff_mod.nml`

**Code for module** `cov_cutoff_mod`

**Documentation for module**

**Run-time control for module**
DART observations directory:

- **forward_operators/**
  - Code to compute forward operators for many types of instruments and for some idealized models.

- **obs_converters/**
  - Directories containing code and build tools for programs that create observation sequence files from many data sources.

- **utilities/**
  - Code and build tools for utilities that manipulate observation sequence files. Available for low-order models (oned/) and large models (threed_sphere/).
DART documentation directory:

- DART/documenta'on/
- DART_LAB/
- html/
- tutorial/
- index.html

Interactive matlab intro to ensemble assimilation.

Discussions of aspects of DART design.

DART tutorial. What you’re looking at now.

Header for html documentation.

Many others...
DART assimilation_code/ directory:

- **programs/**: Code for all DART programs including filter that does ensemble assimilation.
- **modules/**: All DART assimilation code that is not a main program.
- **scripts/**: Some scripts for specialized tasks.
- **location/**: Modules that define a geometry for an assimilation. Most big problems use `threed_sphere/`. Simple models use `oned/`.
DART assimilation_code/modules/ directory:

- assimilation/
  - Modules associated with ensemble solver algorithms, includes filter_mod.f90.

- observations/
  - Modules that do generic forward operators and manage observations.

- io/
  - Modules for getting data in/out of DART filters.

- utilities/
  - Modules that manage DART data structures, parallel processing, time and calendars, etc.
Look at ensemble adjustment filter observation increment subroutine.

In `assimilation_code/modules/assimilation/assim_tools_mod.f90` search for the string
`subroutine obs_increment_eakf`.

`obs_increment_eakf()` computes updated mean in a temporary variable named `new_mean`.

Computes ratio of updated standard deviation to prior. Compare to tutorial slides in section 1.
1. Filtering For a One Variable System
2. The DART Directory Tree
3. DART Runtime Control and Documentation
5. Comprehensive Filtering Theory: Non-Identity Observations and the Joint Phase Space
6. Other Updates for An Observed Variable
7. Some Additional Low-Order Models
8. Dealing with Sampling Error
9. More on Dealing with Error; Inflation
10. Regression and Nonlinear Effects
11. Creating DART Executables
12. Adaptive Inflation
13. Hierarchical Group Filters and Localization
14. Quality Control
15. DART Experiments: Control and Design
16. Diagnostic Output
17. Creating Observation Sequences
18. Lost in Phase Space: The Challenge of Not Knowing the Truth
19. DART-Compliant Models and Making Models Compliant
20. Model Parameter Estimation
21. Observation Types and Observing System Design
22. Parallel Algorithm Implementation
23. Location module design (not available)
24. Fixed lag smoother (not available)
25. A simple 1D advection model: Tracer Data Assimilation