How to Write Unit Tests with GoogleTest

Krzysztof Sakrejda edited this page on 4 Apr 2015 · 2 revisions

What to Test

1. We're trying to make sure all the boundary conditions get called
    - zero-size inputs for containers
    - 0, NaN, inf, and -inf inputs for floating point
2. We want to make sure all tests get tickled and the appropriate exception types are thrown and documented.
3. We want to test that autodiff works
    - reverse-mode
    - forward mode: double (1st order), var (2nd order), fvar *(3rd order)*

There should be a finite difference testing framework soon. An alternative is to define a second, simpler templated versio of a function and test against that.

Using GoogleTest

The place to start for doc for GoogleTest (other than looking at our existing examples) are:

- Primer: an overview
- Advance Guide: good description of available tests and best API guide I know

Common test types

The documentation (above) is excellent so check there for details and the complete list of tests. The following is a small sample of commonly used test types in Stan and links to the relevant sections of the GoogleTest documentation.

- EXPECT/ASSERT EXPECT records the result but allows further tests in the same function, ASSERT exits the current function without cleanup. Stan unit tests use both forms.
- EXPECT_TRUE EXPECT_TRUE/FALSE test that the condition is TRUE/FALSE.
- EXPECT_EQ Tests that a calculated value equals (==) an expected result exactly, not appropriate for floating point comparison.
- EXPECT_FLOAT_EQ Tests that a floating point calculation results to known output values.
- EXPECT_NEAR Like EXPECT_FLOAT_EQ but a third argument sets the threshold used.
- ASSERT_THROW Tests that a specific exception type is thrown.
- ASSERT_NO_THROW Tests that the statement does not throw. Some parts of the code are not supposed to throw... not sure which.

~

Pages 56

Find a Page…

Home

Adding a Gaussian Process Covariance Function

CloudStan functional specification

Code Quality

Coding Style and Idioms

Compiler and OS Detection Macros for Cpp

Complex ODE Based Models

Continuous Integration Testing

Contributing New Functions to Stan

Cpp11 Upgrade

Design: Consolidated Output for Sample, Optimize, ADVI, and Diagnose

Dev: Git Process

Dev: Tricks

Developer process overview

Development Testing: Tools and Procedures

Show 41 more pages…

Clone this wiki locally

`https://github.com/stan-c`