Avoiding recompilation of Stan models

Compiling models takes time. It is in our interest to avoid recompiling models whenever possible. If the same model is going to be used repeatedly, we would like to compile it just once. The following demonstrates how to reuse a model in different scripts and between interactive Python sessions.

Within sessions you can avoid recompiling a model in two ways. The simplest method is to reuse a fit object in the call to `stan`. For example,

```
from pystan import stan

# bernoulli model
model_code = """
    data {
      int<lower=0> N;
      int<lower=0,upper=1> y[N];
    }
    parameters {
      real<lower=0,upper=1> theta;
    }
    model {
      theta ~ beta(0.5, 0.5);  // Jeffreys' prior
      for (n in 1:N)
        y[n] ~ bernoulli(theta);
    }
"""

data = dict(N=10, y=[0, 1, 0, 1, 0, 1, 0, 1, 1, 1])
```

```
fit = stan(model_code=model_code, data=data)
print(fit)

new_data = dict(N=6, y=[0, 0, 0, 0, 0, 1])
fit2 = stan(fit=fit, data=new_data)
print(fit2)
```

Alternatively, we can compile the model once using the `StanModel` class and then sample from the model using the `sampling` method.

```
from pystan import StanModel

# using the same model as before
data = dict(N=10, y=[0, 1, 0, 1, 0, 1, 0, 1, 1, 1])
sm = StanModel(model_code=model_code)
fit = sm.sampling(data=data)
print(fit)

new_data = dict(N=6, y=[0, 0, 0, 0, 0, 1])
fit2 = sm.sampling(data=new_data)
print(fit2)
```

It is also possible to share models **between sessions** (or between different Python scripts). We do this by saving compiled models (`StanModel` instances) in a file and then reloading it when we need it later. (In short, `StanModel` instances are picklable.)

The following two code blocks illustrate how a model may be compiled in one session and reloaded in a subsequent one using `pickle` (part of the Python standard library).

```
import pickle
from pystan import StanModel

# using the same model as before
data = dict(N=10, y=[0, 1, 0, 1, 0, 1, 0, 1, 1, 1])
sm = StanModel(model_code=model_code)
fit = sm.sampling(data=data)
print(fit)

# save it to the file 'model.pkl' for later use
with open('model.pkl', 'wb') as f:
    pickle.dump(sm, f)
```

The following block of code might appear in a different script (in the same directory).

```
import pickle

sm = pickle.load(open('model.pkl', 'rb'))

new_data = dict(N=6, y=[0, 0, 0, 0, 0, 1])
fit2 = sm.sampling(data=new_data)
print(fit2)
```

Automatically reusing models

For those who miss using variables across sessions in R, it is not difficult to write a function that automatically saves a copy of every model that gets compiled using `stan` and opportunistically loads a copy of a model if one is available.

```
import pystan
import pickle
from hashlib import md5

def stan_cache(model_code, model_name=None, **kwargs):
    """Use just as you would `stan`"""
    code_hash = md5(model_code.encode('ascii')).hexdigest()
    if model_name is None:
        cache_fn = 'cached-model-{}.pkl'.format(code_hash)
    else:
        cache_fn = 'cached-{}-{}.pkl'.format(model_name, code_hash)
    try:
        sm = pickle.load(open(cache_fn, 'rb'))
    except:
        sm = pystan.StanModel(model_code=model_code)
        with open(cache_fn, 'wb') as f:
            pickle.dump(sm, f)
    else:
        print("Using cached StanModel")
    return sm.sampling(**kwargs)

# with same model_code as before
data = dict(N=10, y=[0, 1, 0, 0, 0, 0, 0, 0, 0, 1])
fit = stan_cache(model_code=model_code, data=data)
print(fit)

new_data = dict(N=6, y=[0, 0, 0, 0, 0, 1])
# the cached copy of the model will be used
fit2 = stan_cache(model_code=model_code, data=new_data)
print(fit2)
```