Modeling calculation kernels

Hoël Jalmin
Tutored by Arnaud Legrand and Tom Cornebize

The 21th of June, 2019

Context

With the current need for high performance computing, and the hardware complexity:

- How to predict the duration of calculations?
- How to check if the performance is normal?

For this talk:

1. Brief presentation of the context
2. Introduction to Bayesian sampling
3. Examples of application

Background on HPC and Polaris research

Modern context

- HPC systems use thousands of nodes, multiple levels of cache, hyperthreading, etc -> makes it difficult to predict performance
- Some functions are used everywhere, and called thousands of times

Polaris research

- Simulating HPL on smaller supercomputers to optimize it at a lesser cost
- Elaborated complex models but needed to evaluate and confirm the models

Examples

- The blas library is used by thousands of programs and constitutes most of HPL calculations.
- Performance variability is also caused by network communications
- Needs to check the models with bayesian sampling.

Bayesian Statistics

Bayes model

Model Let's say $y \sim \mathcal{N}(\mu, \sigma)$

- μ: Model parameters
- y: Dependent data (posterior)
- σ: Independent data (prior)

We observe some data and need to find model parameters

The vocabulary

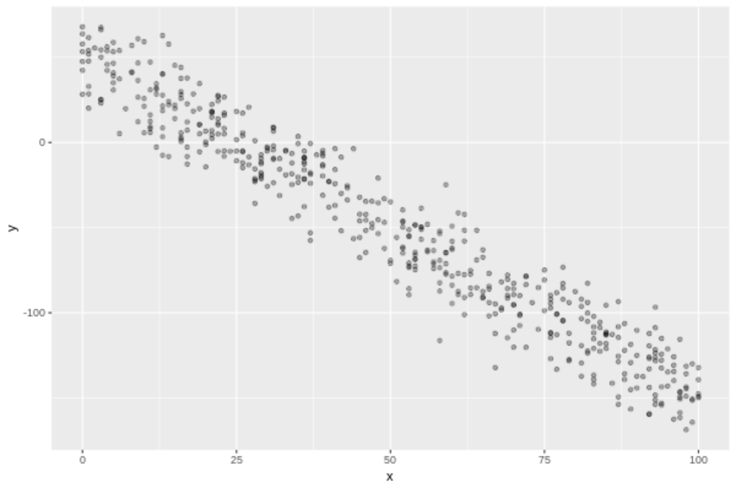
- Posterior: The distribution of the parameters
- Likelihood: A function of the parameters, the model
- Prior: Existing knowledge of the system, guesses on the parameters values

$$\boxed{\underbrace{p(\mu|y,\sigma)}_{\text{Posterior}} \propto \underbrace{p(y|\mu,\sigma)}_{\text{Likelihood}}\,\underbrace{p(\mu,\sigma)}_{\text{Prior}}} \text{ assuming } y \sim \mathcal{M}(\mu,\sigma)$$

Bayesian Sampling

A Bayesian Sampler, Stan

With a simple example



Using this data, we'll try to find the parameters that were used to generate it.

The Stan model

```
library(rstan)

modelString = "data { // the observations
    int<lower=1> N; // number of points
    vector[N] x;
    vector[N] y;
}
parameters { // what we want to find
    real intercept;
    real coefficient;
    real<lower=0> sigma; // indication: sigma cannot be negative
}
model {
    // We define our priors
    intercept   ~ normal(0, 10); // We know that all the parameters follow a nor
    coefficient ~ normal(0, 10);
    sigma       ~ normal(0, 10);

    // Then, our likelihood function
    y ~ normal(coefficient*x + intercept, sigma);
}
"
sm = stan_model(model_code = modelString)
```

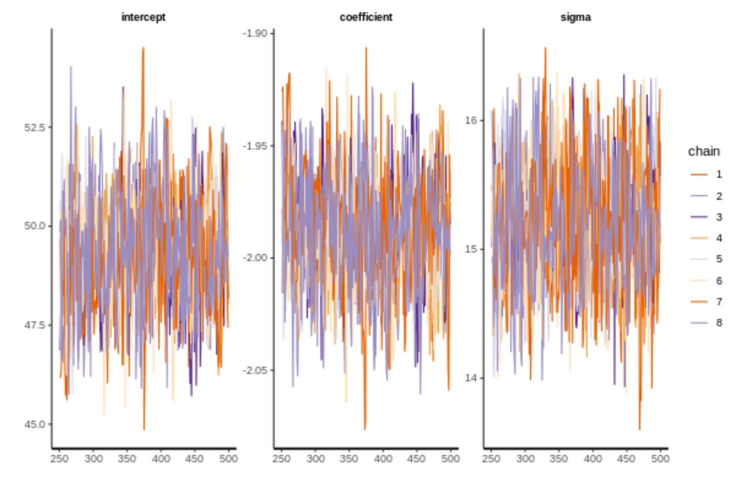
```
data = list(N=nrow(df),x=df$x,y=df$y)
fit = sampling(sm,data=data, iter=500, chains=8)

print(fit)

Inference for Stan model: ea4b5a288cf5f1d87215860103a9026e.
8 chains, each with iter=500; warmup=250; thin=1;
post-warmup draws per chain=250, total post-warmup draws=2000.

               mean se_mean   sd     2.5%      25%      50%      75%    97.5%
intercept     49.86    0.04 1.36    47.14    48.94    49.87    50.82    52.40
coefficient   -2.00    0.00 0.02    -2.04    -2.01    -2.00    -1.98    -
1.95
sigma         15.03    0.01 0.47    14.18    14.70    15.02    15.35    15.99
lp__       -1615.90    0.04 1.12 -1618.80 -1616.45 -1615.62 -1615.05 -1614.58
           n_eff Rhat
intercept    1070 1.00
coefficient  1042 1.00
sigma        1042 1.01
lp__          871 1.00

Samples were drawn using NUTS(diag_e) at Wed Jun 19 17:07:18 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```


intercept
coefficient
sigma
52.5
50.0
47.5
45.0
-1.90
-1.95
-2.00
-2.05
16
15
14
250
300
350
400
450
500
chain
1
2
3
4
5
6
7
8

A polynomial model with specific noise

A model depending on the host

A hierarchical model

The importance of the priors

The follow up

Possibilities

- Modeling other calculation kernels
- Modeling the network communications
- Parsing and converting Stan code to C, to generate new data more efficiently
- Anomaly detection