# Speeding up Python with Rust

## Summer school on modelling and complex systems 2021

Metodi Nikolov

10 July 2021

# Motivating Example Model

Say that you have come up with a very interesting Bayesian model for the data you have. The model is not supported by the standard packages for inference (e.g. `PyMCMC`, `Stan`), so you will have to code it yourself.

# Scaled Student T Distribution

Throughout this we will use a known model, with likelihood function based around the Scaled Student T Distribution, with priors as straightforward as they can be:

$$
\begin{aligned}
x &\sim t_\nu(\mu, \sigma^2) \\
\mu &\propto \mathit{const.} \\
\sigma^2 &\propto \mathit{const.}
\end{aligned}
$$

# Scaled Student T Distribution

Throughout this we will use a known model, with likelihood function based around the Scaled Student T Distribution, with priors as straightforward as they can be:

$$
\begin{aligned}
x &\sim t_\nu(\mu, \sigma^2) \\
\mu &\propto const. \\
\sigma^2 &\propto const.
\end{aligned}
$$

**Subordinate Representation**

$$x \sim \mu + N\sqrt{\frac{\nu\sigma^2}{\chi^2_\nu}}$$

# Scaled Student T Distribution

Throughout this we will use a known model, with likelihood function based around the Scaled Student T Distribution, with priors as straightforward as they can be:

$$
\begin{aligned}
x &\sim t_\nu(\mu, \sigma^2) \\
\mu &\propto const. \\
\sigma^2 &\propto const.
\end{aligned}
$$

**Subordinate Representation**

$$x \sim \mu + N\sqrt{\frac{\nu\sigma^2}{\chi^2_\nu}}$$

**Another Representation**

$$
\begin{aligned}
x &\sim N(\mu, V) \\
V &\sim \text{Inv-}\chi^2(\nu, \sigma^2)
\end{aligned}
$$

# Scaled Student T Distribution

Throughout this we will use a known model, with likelihood function based around the Scaled Student T Distribution, with priors as straightforward as they can be:

$$
\begin{aligned}
x &\sim t_\nu(\mu, \sigma^2) \\
\mu &\propto const. \\
\sigma^2 &\propto const.
\end{aligned}
$$

**Subordinate Representation**

$$x \sim \mu + N\sqrt{\frac{\nu\sigma^2}{\chi^2_\nu}}$$

**Another Representation**

$$
\begin{aligned}
x &\sim N(\mu, V) \\
V &\sim \text{Inv-}\chi^2(\nu, \sigma^2)
\end{aligned}
$$

NB: There is a *V* for each data point!

We will cheat a bit and extend the above model to allow better mixing:

$$X \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

We will cheat a bit and extend the above model to allow better mixing:

$$X \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

Here, $\alpha^2 U$ plays the role of $V$, and $\alpha\tau = \sigma - \alpha$ is a 'mixing' parameter, helps with MCMC. The full conditional for Gibbs are as follows:

We will cheat a bit and extend the above model to allow better mixing:

$$x \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

Here, $\alpha^2 U$ plays the role of $V$, and $\alpha\tau = \sigma$ - $\alpha$ is a 'mixing' parameter, helps with MCMC. The full conditional for Gibbs are as follows:

- $$U_i|\alpha, \mu, \tau^2, \nu, \sim \text{Inv-}\chi^2\left(\nu+1, \frac{\nu\tau^2 + ((x_i-\mu)/\alpha)^2}{\nu+1}\right)$$

We will cheat a bit and extend the above model to allow better mixing:

$$x \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

Here, $\alpha^2 U$ plays the role of $V$, and $\alpha\tau = \sigma - \alpha$ is a 'mixing' parameter, helps with MCMC. The full conditional for Gibbs are as follows:

- $$U_i|\alpha, \mu, \tau^2, \nu, \sim \text{Inv-}\chi^2\left(\nu + 1, \frac{\nu\tau^2 + ((x_i - \mu)/\alpha)^2}{\nu + 1}\right)$$
- $$\mu|\alpha, \tau^2, U, \nu, x \sim N\left(\frac{\sum \frac{1}{\alpha^2 U_i} x_i}{\sum \frac{1}{\alpha^2 U_i}}, \frac{1}{\sum \frac{1}{\alpha^2 U_i}}\right)$$

We will cheat a bit and extend the above model to allow better mixing:

$$x \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

Here, $\alpha^2 U$ plays the role of $V$, and $\alpha\tau = \sigma$ – $\alpha$ is a 'mixing' parameter, helps with MCMC. The full conditional for Gibbs are as follows:

- $$U_i|\alpha, \mu, \tau^2, \nu, \sim \text{Inv-}\chi^2\left(\nu+1, \frac{\nu\tau^2 + ((x_i - \mu)/\alpha)^2}{\nu+1}\right)$$
- $$\mu|\alpha, \tau^2, U, \nu, x \sim N\left(\frac{\sum \frac{1}{\alpha^2 U_i} x_i}{\sum \frac{1}{\alpha^2 U_i}}, \frac{1}{\sum \frac{1}{\alpha^2 U_i}}\right)$$
- $$\tau^2|\alpha, \mu, U, \nu, x \sim \text{Gamma}(\frac{\nu n}{2}, \frac{\nu}{2}\sum\frac{1}{U_i})$$

We will cheat a bit and extend the above model to allow better mixing:

$$x \sim N(\mu, \alpha^2 U)$$
$$U \sim \text{Inv-}\chi^2(\nu, \tau^2)$$

Here, $\alpha^2 U$ plays the role of $V$, and $\alpha\tau = \sigma$ – $\alpha$ is a 'mixing' parameter, helps with MCMC. The full conditional for Gibbs are as follows:

- $$U_i|\alpha, \mu, \tau^2, \nu, \sim \text{Inv-}\chi^2\left(\nu + 1, \frac{\nu\tau^2 + ((x_i - \mu)/\alpha)^2}{\nu + 1}\right)$$
- $$\mu|\alpha, \tau^2, U, \nu, x \sim N\left(\frac{\sum \frac{1}{\alpha^2 U_i} x_i}{\sum \frac{1}{\alpha^2 U_i}}, \frac{1}{\sum \frac{1}{\alpha^2 U_i}}\right)$$
- $$\tau^2|\alpha, \mu, U, \nu, x \sim \text{Gamma}(\frac{\nu n}{2}, \frac{\nu}{2}\sum \frac{1}{U_i})$$
- $$\alpha^2|\mu, \tau^2, U, \nu, x \sim \text{Inv-}\chi^2\left(n, \frac{1}{n}\sum \frac{(x_i - \mu)^2}{U_i}\right)$$

```python
class ScaledTModel(object):

        __slots__ = ['_data', '_data_size', '_nu',...]

        def __init__(self, data, nu):
                pass

        def run(self, burnin, sample_size):
                pass

        def get_mu(self):
                pass

        def get_sigma2(self):
                pass

        def _update_mu(self):
                pass

        def _update_tau2(self):
                pass

        def _update_alpha2(self):
                pass

        def _update_extended_vars(self):
                pass

        def _sampleScaledInvChiSquare(self, ni, scale):
                pass
```

```python
def __init__(self, data, nu):
        print("making a model")
        self._data = np.asarray(data)
        self._data_size = len(data)
        self._nu = nu

        self._rng = default_rng()
        # Some starting values
        self._extended_vars = np.zeros(self._data_size)
        self._tau2 = 1
        self._mu = sum(data) / self._data_size
        self._alpha2 = 1

        self._update_extended_vars()

        # temporary data holders, so that we reuse memory
        self._tmp_with_data_size = np.zeros(self._data_size)
        self._tmp_with_data_size2 = np.zeros(self._data_size)
```

```python
def _update_mu(self):
        np.reciprocal(self._extended_vars, out=self._tmp_with_data_size)
        self._tmp_with_data_size2 = self._data * self._tmp_with_data_size

        variance = self._tmp_with_data_size.sum()
        expected_value = self._tmp_with_data_size2.sum()

        variance /= self._alpha2
        expected_value /= self._alpha2
        variance = 1.0 / variance
        expected_value = expected_value * variance
        self._mu = self._rng.normal(expected_value, math.sqrt(variance))
```

```python
def _update_tau2(self):
        np.reciprocal(self._extended_vars, out=self._tmp_with_data_size)
        x = self._tmp_with_data_size.sum()
        self._tau2 = self._rng.gamma(self._data_size * self._nu / 2.0, 2.0 / (self._nu * x))


def _update_alpha2(self):
        x = 0.0
        self._tmp_with_data_size = self._data - self._mu
        self._tmp_with_data_size = (self._tmp_with_data_size *
                self._tmp_with_data_size) / self._extended_vars
        x = self._tmp_with_data_size.sum()
        x /= self._data_size
        self._alpha2 = self._sampleScaledInvChiSquare(self._data_size, x)
```

```
def _update_extended_vars(self):
        for i in range(self._data_size):
                x = (self._data[i] - self._mu) * (self._data[i] - self._mu) / self._alpha2
                self._extended_vars[i] = self._sampleScaledInvChiSquare(self._nu + 1,
                        (self._nu * self._tau2 + x) / (self._nu + 1))

@property
def get_mu(self):
        if hasattr(self, '_results_mu'):
                return self._results_mu

@property
def get_sigma2(self):
        if hasattr(self, '_results_sigma2'):
                return self._results_sigma2
```

```python
def run(self, burn_in = 1000, sample_size = 2000):
        self._results_mu = np.zeros(sample_size)
        self._results_sigma2 = np.zeros(sample_size)
        print("Starting Burn-in")
        for _ in range(burn_in):
                self._update_extended_vars()
                self._update_alpha2()
                self._update_mu()
                self._update_tau2

        print("Starting Data run")
        for i in range(sample_size):
                self._update_extended_vars()
                self._update_alpha2()
                self._update_mu()
                self._update_tau2
                self._results_mu[i] = self._mu
                self._results_sigma2[i] = self._alpha2 * self._tau2
```

```python
from great_model.python_great_model import ScaledTModel as Model
import numpy as np

nu = 6
mu = -3.14
sigma2 = 30
data_size = 500
z = np.random.randn(data_size)
x = np.random.chisquare(nu, data_size)

t_data = mu + z * np.sqrt(sigma2 * nu / x)

g_model = Model(t_data, nu)

%timeit g_model.run(2000, 2000)
```

On my local machine %timeit reports 8 seconds of execution.

# Crash Course in Rust

# PyO3

# Motivating Example Model Revisited

# Next Steps

- Gelman, A.; Carlin, J. B.; Stern, H. S. & Rubin, D. B. (2014), *Bayesian Data Analysis*, Chapman and Hall/CRC