1. Exercises

1.1 Running AutoBayes

1.1.1 Exercise 1

Run the norm.ab example and inspect generated code and derivation. If possible, generate the latex version of the derivation.

```plaintext
1 model normal_simple as 'NORMAL MODEL WITHOUT PRIORS'.
2 double mu.
3 double sigma_sq as 'SIGMA SQUARED'.
4   where 0 < sigma_sq.
5 const nat n as '# DATA POINTS'.
6   where 0 < n.
7 data double x(0..n-1) as 'KNOWN DATA POINTS'.
8 x(_) ~ gauss(mu, sqrt(sigma_sq)).
9 max pr(x | {mu, sigma_sq}) for {mu, sigma_sq}.
```

Listing 1.1: norm.ab

1.1.2 Exercise 2

Generate multiple versions for this problem. Note: use the appropriate flags to allow AutoBayes to generate numerical optimization algorithms:

(schema_control_arbitrary_init_values)

1.1.3 Exercise 3

Modify the “norm” example to use a different probability density function. Note that some of them do have a different number of parameters. Inspect the generated code and derivation. Can the problem be solved symbolically for all PDFs?

Hint: use vonmises1, poisson, weibull, cauchy
1.1.4 Exercise 4

Generate multiple versions for the mixture-of-gaussians example. What are the major differences between the different synthesized programs.

Note: the specification is `mog.ab` in the models_manual directory.

Generate a sampling data generator (autobayes -sample) for this specification.

In AutoBayes generate 1000 data points that go into 3 different classes. Then run the different programs and see how good they estimate the parameters.

Note: the generated functions require column-vectors, so, e.g., give the means as `[1,2,3]'`

```octave
 octave -3.4.0:1> sample_mog
 usage: [vector c, vector x] = sample_mog(vector mu, int n_points, vector phi, vector sigma)
 octave -3.4.0:2> [c, x] = sample_mog([1, 2, 4]', 1000, [0.3, 0.1, 0.6]', [0.1, 0.1, 0.2]');
```

Listing 1.2: calling the synthesized code in Octave

1.1.5 Exercise 5

Run a change-point detection model (e.g., `climb_transition.ab`) and look at generated code and derivation. How does AUTOBAYES find the maximum?

1.1.6 Exercise 6

Add the Pareto distribution to the built-in transitions. Get the formulas from wikipedia.

Try the following simple model:

```octave
 model pareto as 'NORMAL MODEL WITHOUT PRIORS'.
 double alpha.
 where 3 < alpha.
 const double xm.
 where 0 < xm.
 const nat n as '# DATA POINTS'.
 where 0 < n.
 data double x(0..n-1) as 'KNOWN DATA POINTS'.
 where 0 < x( ).
```
1.1 Running AutoBayes

where $x_m < x(\_)$.

$x(\_) \sim \text{pareto}(x_m, \alpha)$.

\[ \max \, p_r(x \mid \{\alpha\}) \quad \text{for} \quad \{\alpha\}. \]

Listing 1.3: Specification for Pareto distribution

```
octave-3.4.0:2> x_m=5;
octave-3.4.0:3> alpha=15;
octave-3.4.0:4> x=x_m*(1./(1-rand(10000,1)).^((1/alpha));
octave-3.4.0:5> alpha_est = pareto(x,5)
alpha_est = 15.081
```

Listing 1.4: Generate Pareto-distributed random numbers