Solution 4: the joy of probabilistic inference

Set the seed for reproducibility

```
set.seed(1)
```

Q.1: logistic rocket improvement

Let us load the data

```
success_indicators = c(1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1)
```

Simulator

```
logistic_regression = function() {
  intercept = simulate(Norm(0, 1))
  slope     = simulate(Norm(0, 1))
  for (i in seq_along(success_indicators)){
    success_probability = plogis(intercept + i*slope)
    observe(success_indicators[i], Bern(success_probability))
  }
  # predict
  i_next = length(success_indicators) + 1
  predictive_probability = plogis(intercept + i_next*slope)
  predictive_indicator = simulate(Bern(predictive_probability))
  #
  return(c(intercept, slope, predictive_indicator))
}
```

Reproduce plot

Load the code for simPPLe and utilities

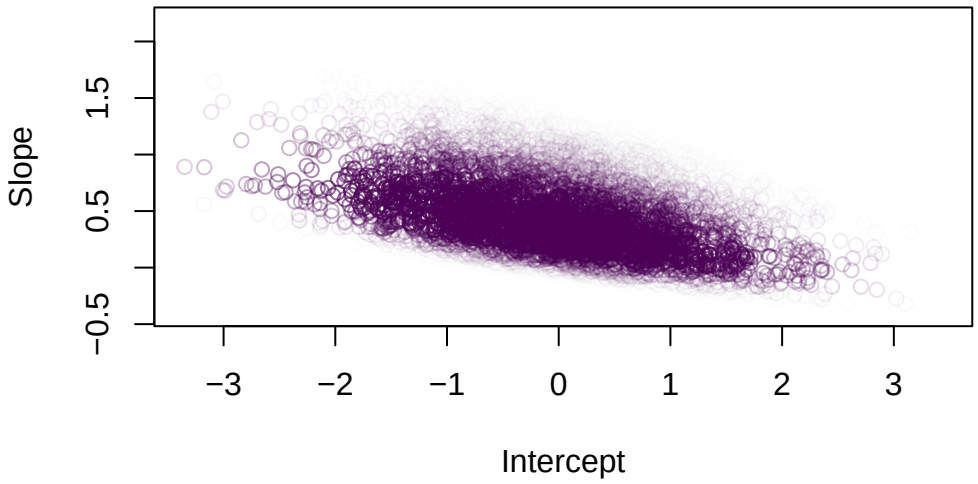
```
source("simple.R")
source("simple_utils.R")
```

Now we can run simPPLe on the `logistic_regression` simulator

```
particles = posterior_particles(logistic_regression, 10000)
```

Finally, we use the sampled particles to create the weighted scatter plot

```
weighted_scatter_plot(particles, plot_options = list(xlab="Intercept", ylab="Slope"))
```



Estimate next launch success

Using the fact that the predictions for the next launch correspond to the third column in the matrix `particles$samples`, we have

```
probability_next_success = sum(particles$samples[,3]*particles$weights) /
  sum(particles$weights)
cat(paste("Estimated probability of success:", probability_next_success))
```

```
Estimated probability of success: 0.948918882615765
```

Estimate next launch success with zero slope

As before, we need to define a simulator for the model without slope

```
logistic_regression_no_slope = function() {
  intercept = simulate(Norm(0, 1))
  slope     = 0
  for (i in seq_along(success_indicators)){
    success_probability = plogis(intercept + i*slope)
    observe(success_indicators[i], Bern(success_probability))
  }
  # predict
  i_next = length(success_indicators) + 1
  predictive_probability = plogis(intercept + i_next*slope)
  predictive_indicator = simulate(Bern(predictive_probability))
  #
  return(c(intercept, slope, predictive_indicator))
}
```

Now we run the model using simPPLe

```
particles_no_slope = posterior_particles(logistic_regression_no_slope, 10000)
```

Finally, estimate the success probability using

```
probability_next_success_ns =
  sum(particles_no_slope$samples[,3]*particles_no_slope$weights) /
  sum(particles_no_slope$weights)
cat(paste("Estimated probability of success (no slope):", probability_next_success_ns))
```

```
Estimated probability of success (no slope): 0.717993413719777
```

Q.2: choosing a model

Since the two logistic regression simulators have the same return type and both look at the same dataset, we can define the unified model simply using

```
logistic_regression_unified = function() {
  with_slope = simulate(Bern(0.5))
  sample = if(with_slope){
    logistic_regression()
  }else{
    logistic_regression_no_slope()
  }
  return(with_slope) # return the query
}
```

Now we run the model using simPPLe

```
set.seed(1)
probability_with_slope = posterior(logistic_regression_unified, 10000)
cat(paste("Estimated posterior probability of the model with slope:", probability_with_slope)
```

```
Estimated posterior probability of the model with slope: 0.539175614310968
```

Alternative: use the `posterior_particles` approach

We write `logistic_regression_unified` so that it returns all the unobserved variables—including `with_slope`

```
logistic_regression_unified = function() {
  with_slope = simulate(Bern(0.5))
  sample = if(with_slope){
    logistic_regression()
  }else{
    logistic_regression_no_slope()
  }
  return(c(sample, with_slope)) # add the model indicator to the output
}
```

Run it

```
set.seed(1)
particles_unified = posterior_particles(logistic_regression_unified, 10000)
```

Finally, estimate the probability that the model with slope is preferred under the posterior

```r
probability_with_slope =
  sum(particles_unified$samples[,4]*particles_unified$weights) /
  sum(particles_unified$weights)
cat(paste("Estimated posterior probability of the model with slope:", probability_with_slope)
```

```
Estimated posterior probability of the model with slope: 0.53928881346008
```