```
(def N 20000)
(def predicts (map get-predicts (take N (drop N samples))))
```

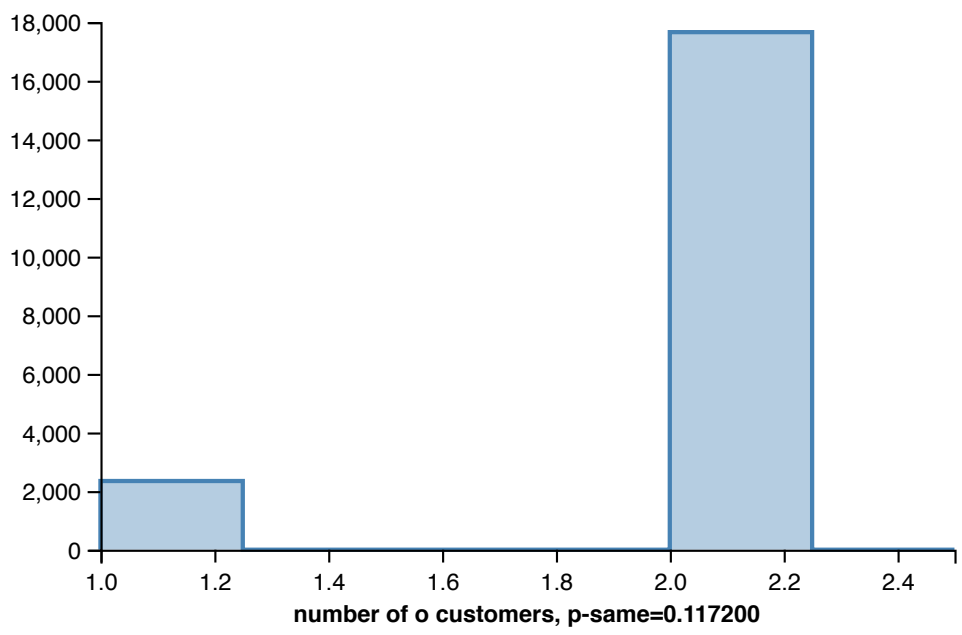
```
#'deli/N

#'deli/predicts
```

Let's compute the probability that this is the same customer, and arrival times for each case:

```
(def p-same+ (/ (count (filter :same-customer predicts)) (double N)))


;; single customer
(def time-to-arrive+ (map :time-to-arrive (filter :same-customer predicts)))
(def mean-to-arrive+ (mean time-to-arrive+))
(def sd-to-arrive+ (std time-to-arrive+))

;; two customers
(def times-to-arrive+ (map :times-to-arrive
                           (filter (complement :same-customer) predicts)))
(def mean-1-to-arrive+ (mean (map first times-to-arrive+)))
(def sd-1-to-arrive+ (std (map first times-to-arrive+)))
(def mean-2-to-arrive+ (mean (map second times-to-arrive+)))
(def sd-2-to-arrive+ (std (map second times-to-arrive+)))
```

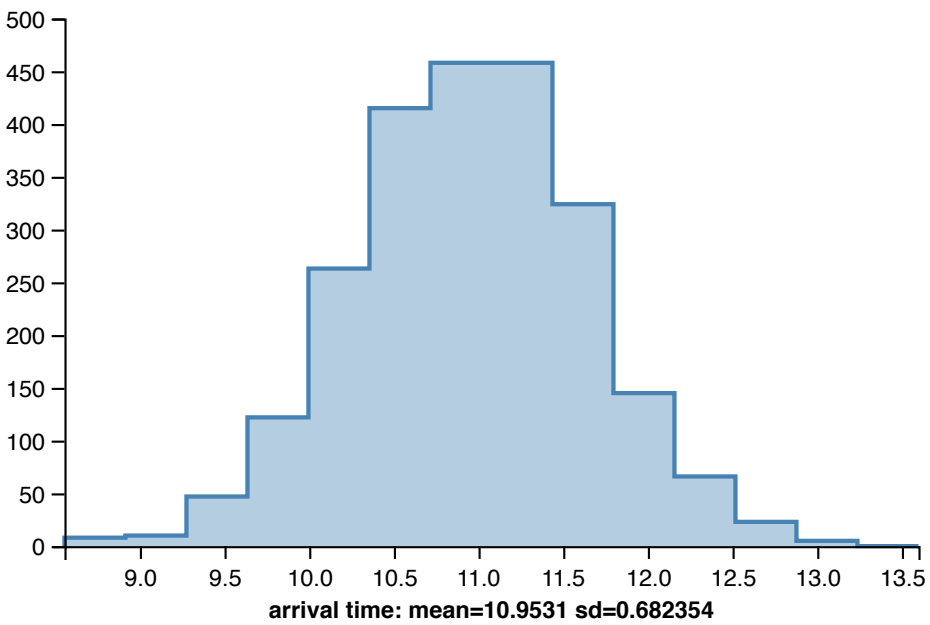
```
#'deli/p-same+

#'deli/time-to-arrive+

#'deli/mean-to-arrive+

#'deli/sd-to-arrive+

#'deli/times-to-arrive+

#'deli/mean-1-to-arrive+

#'deli/sd-1-to-arrive+

#'deli/mean-2-to-arrive+

#'deli/sd-2-to-arrive+
```

```
(plot/histogram (map #(if (:same-customer %) 1 2) predicts)
                :bins 4 :x-title (format "number of o customers, p-same=%6g" p-
same+))
```



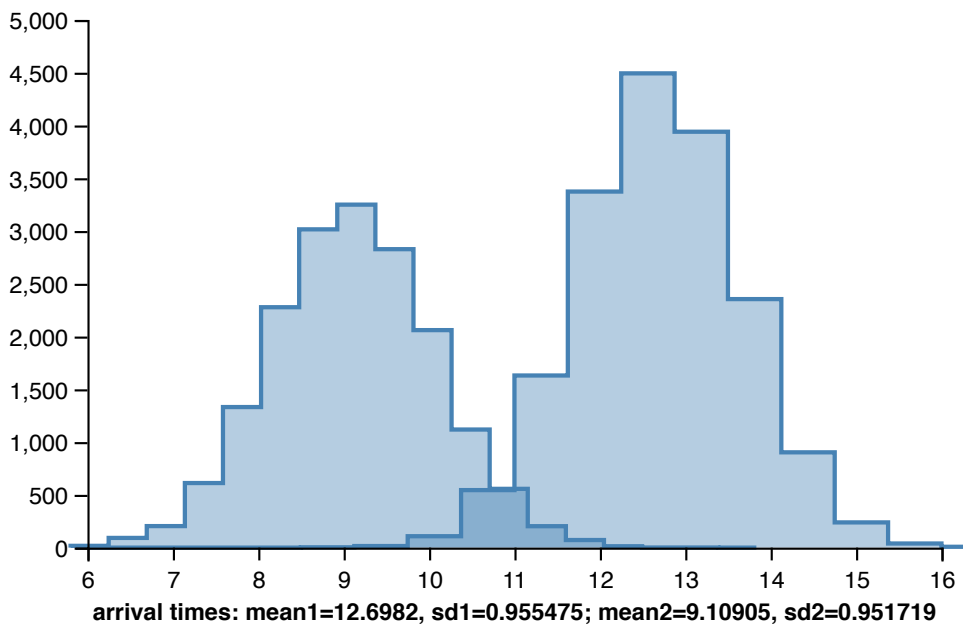
If there is a single customer, there is one arrival time, let's see how it is distributed:

```
(plot/histogram  time-to-arrive+
                 :x-title (format "arrival time: mean=%6g sd=%6g"
                                  mean-to-arrive+
                                  sd-to-arrive+))
```



For two customers there are two different time distributions, let's compare them.

```
(plot/compose
  (plot/histogram (map first times-to-arrive+)
                  :x-title (format
                             "arrival times: mean1=%6g, sd1=%6g; mean2=%6g, sd2=%6g"
                             mean-1-to-arrive+ sd-1-to-arrive+
                             mean-2-to-arrive+ sd-2-to-arrive+)
                  :plot-range [[6 16] :all])
  (plot/histogram (map second times-to-arrive+)))
```



What if we had an algorithm that constructs the posterior? Let's rewrite the `deli` query with posterior distributions and without observations.

```
(defquery deli+
  (let [same-customer (sample (flip p-same+))]
    (predict :same-customer same-customer)
    (if same-customer
      ;; One customer
      (let [time-to-arrive (sample (normal mean-to-arrive+ sd-to-arrive+))]
         (predict :time-to-arrive time-to-arrive))
      ;; Two customers
      (let [time-to-arrive-1 (sample (normal mean-1-to-arrive+ sd-1-to-arrive+))
            time-to-arrive-2 (sample (normal mean-2-to-arrive+ sd-2-to-arrive+))]
        (predict :times-to-arrive [time-to-arrive-1 time-to-arrive-2])))))
```

```
#'deli/deli+
```

This is what **Variational Inference** algorithm does **AUTOMATICALLY**.