Reactive Online Incremental Statistics

Brian Beckman

Improved version of 13 April 2014

In[4]:=
```
<< (NotebookDirectory[] <> "InterfacesAndPolymorphism004.m")
```

State Monad

A value in the state monad has type $s \to \{v, s\}$, that is, function from *state* to pair of *value* and *state*; it's a `fs2vs`.

StateReturn is of type $v \to (s \to \{v, s\})$, that is, it's a `fv2fs2vs`. Call *StateReturn* with a value *v*, and you get a `fs2vs`, that is, a value in the state monad. Members of the state monad are functions of this form: `fv2fs2vs` because they form a monoid under composition via "bind".

In[5]:=
```
ClearAll[StateReturn];
StateReturn[value_] :=
 state ↦ {"value" → value, "state" → state}
```

In[7]:=
```
ClearAll[StateBind];
StateBind[fs2vs_, fv2fs2vs_] :=
 state ↦
  With[{vs0 = fs2vs[state]},
   With[
    {v0 = "value" /. vs0,
     s0 = "state" /. vs0},
    fv2fs2vs[v0][s0]]]
```

Unit Tests

In[9]:=
```
{"value" → 42, "state" → 1036} ===
 StateBind[
   StateReturn[42],
   value ↦
    state ↦
     With[{r = 1},
      {"value" → value,
       "state" → state + value + r}]][993]
```

Out[9]=
```
True
```

Statistics Without Binding

The pattern is to *Fold* a state extractor over a sequence of monadic values. This is still pull model, requiring a sequence, either lazy or fully realized in memory.

The ***state extractor*** is a function of a current state, *s*, and a monadic value, *m*. Our statistics constructors, namely *bumper*, *summer*, and *welford*, construct monadic values from ordinary values. Each statistics constructor also contains code for some accumulation task on the state arguments: *bumper*'s monadic value bumps a count; *summer*'s monadic value sums the values into the state; *welford*'s monadic value applies Welford's algorithm to the current value and state.

Pairs of state and value are packaged in ***objects***, that is, *Dispatch* lists of replacement rules for the names of the object's members, namely *value* and *state* as strings. Thus, our state extractor is:

In[10]:=
```
ClearAll[stateExtractor];
stateExtractor[currentState_, monadicValueProducingAPair_] :=
  "state" /. monadicValueProducingAPair[currentState];
```

Running Count

Let the state of a running-count calculation be the number of items seen in a collection. Here's some sample data:

In[12]:=
```
$collection = RandomInteger[{-100, 100}, 42]
```

Out[12]=
```
{-1, 66, -78, 59, -50, 44, 11, 34, 63, -1, 7, -92, -68, 75,
 32, -77, 5, -41, 60, -80, -20, -16, -8, 4, -63, -37, -64, -48,
 25, -65, -3, 30, -43, -21, -86, -46, -38, 100, -64, 100, -74, 22}
```

The following keeps a running aggregate without any global, mutated variables.

Bumper

In[13]:=
```
ClearAll[bumper];
bumper =
  value ↦
   state ↦
    {"value" → value, "state" → state + 1};
```

Unit Test

In[15]:=
```
Fold[stateExtractor, 0, bumper /@ $collection]
```

Out[15]=
```
42
```

Running Sum

The following keeps a running sum:

Summer

In[16]:=
```
ClearAll[summer];
summer =
  value ↦
   state ↦
    {"value" → value, "state" → state + value};
```

Unit Test

In[18]:=
```
Fold[stateExtractor, 0, summer /@ $collection]
```

Out[18]=
```
-447
```

Welford's Sum Of Squared Residuals (SSR)

This method avoids catastrophic cancellation and underflow when calculating variance and standard deviation.

Suppose we have the SSR of the first $n-1$ data:

$$S_{n-1} = \sum_{i=1}^{n-1} \left(x_i - \frac{1}{n-1} \sum_{i=1}^{n-1} x_i \right)^2 = \sum_{i=1}^{n-1} (x_i - \overline{x_{n-1}})^2$$

defining $\overline{x_{n-1}}$ as the mean of the first $n-1$ data. We now write the same for the SSR of the first n data, then solve for a recurrence.

$$S_n = \sum_{i=1}^{n} (x_i - \overline{x_n})^2 = \sum_{i=1}^{n-1} (x_i - \overline{x_n})^2 + (x_n - \overline{x_n})^2$$

$$= \sum_{i=1}^{n-1} (x_i - \overline{x_{n-1}} + \gamma)^2 + (x_n - \overline{x_n})^2$$

defining $\gamma = \overline{x_n} - \overline{x_{n-1}}$, a correction term. Expanding the square reveals S_{n-1} and some more correction terms :

$$S_n = \sum_{i=1}^{n-1} (x_i - \overline{x_{n-1}})^2 + 2 \sum_{i=1}^{n-1} (x_i - \overline{x_{n-1}})\, \gamma + (n-1)\, \gamma^2 + (x_n - \overline{x_n})^2$$

$$= S_{n-1} + (n-1)\, \gamma^2 + (x_n - \overline{x_n})^2$$

The middle term vanishes because the sum of residuals about the old mean vanishes. A bit of play reveals an easy-to-remember formula:

$$x_n - \overline{x_n} = n\, \overline{x_n} - (n-1)\, \overline{x_{n-1}} - \overline{x_n} = (n-1)\, (\overline{x_n} - \overline{x_{n-1}}) = (n-1)\, \gamma$$

$$S_n = S_{n-1} + (n-1)\, \gamma^2 + (n-1)^2\, \gamma^2$$
$$= S_{n-1} + \left(n^2 - n\right) \gamma^2$$
$$= S_{n-1} + n\, \gamma\, (n-1)\, \gamma$$
$$= S_{n-1} + (x_n - \overline{x_n})\, (x_n - \overline{x_{n-1}})$$

because

$$n \gamma = n \overline{x_n} - n \overline{x_{n-1}} = x_n - \overline{x_{n-1}}$$

Naive SSR

We shall also include the naive SSR:

$$NS_n = \sum_{i=1}^{n} x_i^2 - n \left(\sum_{i=1}^{n} x_i \right)^2$$

Welford

In[19]:=
```
ClearAll[welford];
welford =
  value ↦
   state ↦
    {"value" → value,
     "state" →
      With[
       {count = 1 + state."count",
        sum = value + state."sum",
        sum2 = value^2 + state."sum2"},
       With[
        {mean = sum / count,
         oldMean = state."mean",
         oldSsr = state."ssr"},
        {"count" → count,
         "sum" → sum,
         "sum2" → sum2,
         "mean" → mean,
         "ssr" → (value - oldMean) (value - mean) + oldSsr,
         "nssr" → sum2 - count * mean^2}]]};
```

In[21]:=
```
ClearAll[welfordZero];
welfordZero = {
   "count" → 0,
   "sum" → 0,
   "sum2" → 0,
   "mean" → 0,
   "ssr" → 0,
   "nssr" → 0};
```

In[23]:=
```
Fold[stateExtractor,
 welfordZero,
 welford /@ $collection]
```

Out[23]=
$$\left\{\text{count} \to 42,\ \text{sum} \to -447,\ \text{sum2} \to 122\,561,\ \text{mean} \to -\frac{149}{14},\ \text{ssr} \to \frac{1\,649\,251}{14},\ \text{nssr} \to \frac{1\,649\,251}{14}\right\}$$

Mathematica's `Variance` computes the unbiased estimate, so should be $1/(N-1)$ times the sum of

square residuals.

In[24]:=
```
(Length@$collection - 1) * Variance@$collection
```

Out[24]=
$$\frac{1\,649\,251}{14}$$

Sidebar: Combining Stats

Iterating Instead of Folding

Fearing Not the Mutable

In[29]:=
```
Module[{count = 0},
 (v ↦ count++) /@ $collection]
```

Out[29]=
```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41}
```

In[30]:=
```
Module[{sum = 0},
 (v ↦ sum += v) /@ $collection]
```

Out[30]=
```
{-1, 65, -13, 46, -4, 40, 51, 85, 148, 147, 154, 62, -6, 69, 101, 24, 29,
 -12, 48, -32, -52, -68, -76, -72, -135, -172, -236, -284, -259, -324,
 -327, -297, -340, -361, -447, -493, -531, -431, -495, -395, -469, -447}
```

In[31]:=
```
Module[{welfordState = welfordZero},
  (v ↦
    With[{o = welfordState},
     With[{
       count = 1 + o."count",
       sum = v + o."sum",
       sum2 = v^2 + o."sum2"},
      With[{
        mean = sum / count,
        oldMean = o."mean",
        oldSsr = o."ssr"},
       welfordState = {
         "count" → count,
         "sum" → sum,
         "sum2" → sum2,
         "mean" → mean,
         "ssr" → (v - oldMean) (v - mean) + oldSsr,
         "nssr" → sum2 - count * mean^2}]]]) /@
  $collection] // TableForm
```

Out[31]//TableForm=

count → 1	sum → -1	sum2 → 1	mean → -1	ssr → 0	nssr → 0
count → 2	sum → 65	sum2 → 4357	mean → $\frac{65}{2}$	ssr → $\frac{4489}{2}$	nssr → $\frac{4489}{2}$

count → 3	sum → −13	sum2 → 10 441	mean → $-\frac{13}{3}$	ssr → $\frac{31\,154}{3}$	nssr → $\frac{31\,1}{3}$
count → 4	sum → 46	sum2 → 13 922	mean → $\frac{23}{2}$	ssr → 13 393	nssr → 13
count → 5	sum → −4	sum2 → 16 422	mean → $-\frac{4}{5}$	ssr → $\frac{82\,094}{5}$	nssr → $\frac{82\,0}{5}$
count → 6	sum → 40	sum2 → 18 358	mean → $\frac{20}{3}$	ssr → $\frac{54\,274}{3}$	nssr → $\frac{54\,2}{3}$
count → 7	sum → 51	sum2 → 18 479	mean → $\frac{51}{7}$	ssr → $\frac{126\,752}{7}$	nssr → $\frac{126}{7}$
count → 8	sum → 85	sum2 → 19 635	mean → $\frac{85}{8}$	ssr → $\frac{149\,855}{8}$	nssr → $\frac{149}{8}$
count → 9	sum → 148	sum2 → 23 604	mean → $\frac{148}{9}$	ssr → $\frac{190\,532}{9}$	nssr → $\frac{190}{9}$
count → 10	sum → 147	sum2 → 23 605	mean → $\frac{147}{10}$	ssr → $\frac{214\,441}{10}$	nssr → $\frac{214}{1}$
count → 11	sum → 154	sum2 → 23 654	mean → 14	ssr → 21 498	nssr → 21
count → 12	sum → 62	sum2 → 32 118	mean → $\frac{31}{6}$	ssr → $\frac{95\,393}{3}$	nssr → $\frac{95\,3}{3}$
count → 13	sum → −6	sum2 → 36 742	mean → $-\frac{6}{13}$	ssr → $\frac{477\,610}{13}$	nssr → $\frac{477}{1}$
count → 14	sum → 69	sum2 → 42 367	mean → $\frac{69}{14}$	ssr → $\frac{588\,377}{14}$	nssr → $\frac{588}{1}$
count → 15	sum → 101	sum2 → 43 391	mean → $\frac{101}{15}$	ssr → $\frac{640\,664}{15}$	nssr → $\frac{640}{1}$
count → 16	sum → 24	sum2 → 49 320	mean → $\frac{3}{2}$	ssr → 49 284	nssr → 49
count → 17	sum → 29	sum2 → 49 345	mean → $\frac{29}{17}$	ssr → $\frac{838\,024}{17}$	nssr → $\frac{838}{1}$
count → 18	sum → −12	sum2 → 51 026	mean → $-\frac{2}{3}$	ssr → 51 018	nssr → 51
count → 19	sum → 48	sum2 → 54 626	mean → $\frac{48}{19}$	ssr → $\frac{1\,035\,590}{19}$	nssr → $\frac{1\,03}{1}$
count → 20	sum → −32	sum2 → 61 026	mean → $-\frac{8}{5}$	ssr → $\frac{304\,874}{5}$	nssr → $\frac{304}{5}$
count → 21	sum → −52	sum2 → 61 426	mean → $-\frac{52}{21}$	ssr → $\frac{1\,287\,242}{21}$	nssr → $\frac{1\,28}{2}$
count → 22	sum → −68	sum2 → 61 682	mean → $-\frac{34}{11}$	ssr → $\frac{676\,190}{11}$	nssr → $\frac{676}{1}$
count → 23	sum → −76	sum2 → 61 746	mean → $-\frac{76}{23}$	ssr → $\frac{1\,414\,382}{23}$	nssr → $\frac{1\,41}{2}$
count → 24	sum → −72	sum2 → 61 762	mean → −3	ssr → 61 546	nssr → 61
count → 25	sum → −135	sum2 → 65 731	mean → $-\frac{27}{5}$	ssr → 65 002	nssr → 65
count → 26	sum → −172	sum2 → 67 100	mean → $-\frac{86}{13}$	ssr → $\frac{857\,508}{13}$	nssr → $\frac{857}{1}$
count → 27	sum → −236	sum2 → 71 196	mean → $-\frac{236}{27}$	ssr → $\frac{1\,866\,596}{27}$	nssr → $\frac{1\,86}{2}$
count → 28	sum → −284	sum2 → 73 500	mean → $-\frac{71}{7}$	ssr → $\frac{494\,336}{7}$	nssr → $\frac{494}{7}$
count → 29	sum → −259	sum2 → 74 125	mean → $-\frac{259}{29}$	ssr → $\frac{2\,082\,544}{29}$	nssr → $\frac{2\,08}{2}$
count → 30	sum → −324	sum2 → 78 350	mean → $-\frac{54}{5}$	ssr → $\frac{374\,254}{5}$	nssr → $\frac{374}{5}$
count → 31	sum → −327	sum2 → 78 359	mean → $-\frac{327}{31}$	ssr → $\frac{2\,322\,200}{31}$	nssr → $\frac{2\,32}{3}$
count → 32	sum → −297	sum2 → 79 259	mean → $-\frac{297}{32}$	ssr → $\frac{2\,448\,079}{32}$	nssr → $\frac{2\,44}{3}$
count → 33	sum → −340	sum2 → 81 108	mean → $-\frac{340}{33}$	ssr → $\frac{2\,560\,964}{33}$	nssr → $\frac{2\,56}{3}$
count → 34	sum → −361	sum2 → 81 549	mean → $-\frac{361}{34}$	ssr → $\frac{2\,642\,345}{34}$	nssr → $\frac{2\,64}{3}$
count → 35	sum → −447	sum2 → 88 945	mean → $-\frac{447}{35}$	ssr → $\frac{2\,913\,266}{35}$	nssr → $\frac{2\,91}{3}$
count → 36	sum → −493	sum2 → 91 061	mean → $-\frac{493}{36}$	ssr → $\frac{3\,035\,147}{36}$	nssr → $\frac{3\,03}{3}$
count → 37	sum → −531	sum2 → 92 505	mean → $-\frac{531}{37}$	ssr → $\frac{3\,140\,724}{37}$	nssr → $\frac{3\,14}{3}$

count → 38	sum → -431	sum2 → 102 505	mean → $-\frac{431}{38}$	ssr → $\frac{3\,709\,429}{38}$	nssr → $\frac{370}{3}$
count → 39	sum → -495	sum2 → 106 601	mean → $-\frac{165}{13}$	ssr → $\frac{1\,304\,138}{13}$	nssr → $\frac{130}{1}$
count → 40	sum → -395	sum2 → 116 601	mean → $-\frac{79}{8}$	ssr → $\frac{901\,603}{8}$	nssr → $\frac{901}{8}$
count → 41	sum → -469	sum2 → 122 077	mean → $-\frac{469}{41}$	ssr → $\frac{4\,785\,196}{41}$	nssr → $\frac{478}{4}$
count → 42	sum → -447	sum2 → 122 561	mean → $-\frac{149}{14}$	ssr → $\frac{1\,649\,251}{14}$	nssr → $\frac{164}{1}$

Observing Instead of Iterating

In[32]:=

```
$observer[subscriptionId_] :=
 Module[{welfordState = welfordZero}, {
     SubscriptionId :> subscriptionId,
     OnNext[v_] :> (
       With[{o = welfordState},
        With[{
          count = 1 + o."count",
          sum = v + o."sum",
          sum2 = v^2 + o."sum2"},
         With[{
           mean = sum / count,
           oldMean = o."mean",
           oldSsr = o."ssr"},
          welfordState = {
            "count" -> count,
            "sum" -> sum,
            "sum2" -> sum2,
            "mean" -> mean,
            "ssr" -> (v - oldMean) (v - mean) + oldSsr,
            "nssr" -> sum2 - count * mean * mean}]]];
       $currentStats = {"subId" -> subscriptionId, "OnNext" -> welfordState}),
     OnError[exc_] :>
      ($currentStats =
        {"subId" -> subscriptionId, "exception" -> exc, "OnErrr" -> welfordState}),
     OnCompleted[] :> (
       $currentStats = {"subId" -> subscriptionId, "OnCmpl" -> welfordState})} //
    Sort // Dispatch]
```

In[33]:=

```
Dynamic[$currentStats]
```

Out[33]=

$$\{\text{subId} \to \$12,\ \text{OnCmpl} \to \{\text{count} \to 50,\ \text{sum} \to 5.\times 10^{11},\ \text{sum2} \to 5.\times 10^{21},\ \text{mean} \to 1.\times 10^{10},\ \text{ssr} \to 48\,309.8,\ \text{nssr} \to -1.04858\times 10^{6}\}\}$$

In[34]:=
```
GenerateWithTime[
  1, (* initial state *)
  # ≤ Length[$collection] &, (* condition for continuing *)
  $collection[[#]] &, (* value sent to OnNext *)
  0.0625 &, (* time to wait between values *)
  # + 1 & (* function to increment state *)
 ].Subscribe[$observer[Unique[]]]
```

Catastrophic Cancellation in Naive Computation of Variance

We may also compute SSR by the naive method, $S^{(2)} - \left(S^{(1)}\right)^2 \Big/ S^{(0)}$ and investigate the conditions under which it diverges.

In[35]:=
```
Dynamic[$currentStats]
```

Out[35]=
```
{subId → $12, OnCmpl → {count → 50, sum → 5. × 10^11,
    sum2 → 5. × 10^21, mean → 1. × 10^10, ssr → 48309.8, nssr → -1.04858 × 10^6}}
```

In[36]:=
```
With[{collection = RandomReal[{1*^10, 1*^10 + 100}, 50]},
 GenerateWithTime[
   1, (* initial state *)
   # ≤ Length[collection] &, (* condition for continuing *)
   collection[[#]] &, (* value sent to OnNext *)
   0.0625 &, (* time to wait between values *)
   # + 1 & (* function to increment state *)
  ].Subscribe[$observer[Unique[]]]]
```

In[37]:=
```
$collF = RandomReal[{1*^9, 1*^9 + 100}, 5000];
```

In[38]:=
```
Fold[stateExtractor,
 welfordZero,
 welford /@ $collF]
```

Out[38]=
```
{count → 5000, sum → 5. × 10^12, sum2 → 5. × 10^21,
 mean → 1. × 10^9, ssr → 4.11604 × 10^6, nssr → 2.09715 × 10^7}
```

Compare against built-in Variance:

In[39]:=
```
(Length@$collF - 1) * Variance[$collF]
```

Out[39]=
```
4.11604 × 10^6
```