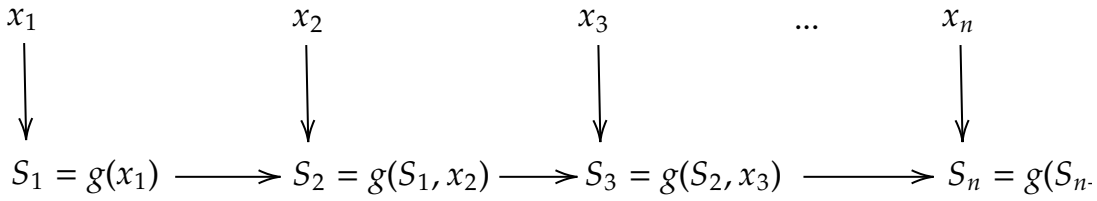
ONE-PASS ALGORITHMS

Llorenç Lledó (BSC) - March 2022



Let $X_i := \{x_1,\ x_2,\ \ldots,\ x_i\}$, then $S_i = f(X_i) = g(S_{i-1}, x_i)\ \forall i$
While f computes the summary S_i at once with all values, g updates a previous summary $S_{i-1}$with a new value x_i.
We also need lightweight summaries, i.e. $mem(S_i) \ll mem(X_i)$

Algorithm for min/max

$g(S_{n-1}, x_n) = g(min_{n-1}, x_n) = ifelse(x_n < min_{n-1},\ x_n,\ min_{n-1})$
$g(S_{n-1}, x_n) = g(max_{n-1}, x_n) = ifelse(x_n > max_{n-1},\ x_n,\ max_{n-1})$
memory needs:

- 1 double to store min_{n-1} or max_{n-1}

Algorithm for mean

$$\overline{x}_n = g(S_{n-1}, x_n) = g(\overline{x}_{n-1}, x_n) = \frac{n-1}{n}\,\overline{x}_{n-1} + \frac{x_n}{n} = \overline{x}_{n-1} + \frac{x_n - \overline{x}_{n-1}}{n}$$

memory needs:

- 1 double to store $\overline{x}_{n-1}$
- 1 int to store $n-1$ or n

Algorithm for variance (shifted Welford's algorithm for the unbiased sample variance)

Requires updating two estimates iteratively. Let $M_{2,n} = \sum_{i=1}^{n}(x_i - \overline{x}_n)^2$, then:
$M_{2,n} = g(S_{n-1}, x_n) = g(M_{2,n-1}, x_n) = M_{2,n-1} + (x_i - \overline{x}_{n-1})(x_i - \overline{x}_n)$
and $\overline{x}_n$ is updated from $\overline{x}_{n-1}$ using the algorithm for the mean before.
At the end of the iterative process (i.e. when the last value is provided):

$$var(X_n) = \frac{M_{2,n}}{n-1}$$

If a rough estimate of the mean is available, subtract it at each step.
$Var(X_n - k) = Var(X_n)$. No need for accuracy, just reduce large values that produce numerical instability. E.g. for sea level pressure, data is around 100000Pa, subtract this value.
Do $x'_n = x_n - k$ at the beginning of each step and proceed as above.

Numerical stability warning! For large n, $M_{2,n}$ can grow larger than the max. float value

and produce inaccurate results. In such cases, one could divide by the current n and store an intermediate variance value, to be cobined later with other chunks using Chan's formula for the general case of combining the variance of two chunks of data. An alternative is to explore Kahan summation.

Memory needs:

- 1 double to store $\overline{x}_{n-1}$
- 1 int to store $n-1$
- 1 double to store $M_{2,n-1}$

Algorithm for standard deviation

Compute variance with algorithm above, and at the end take square root.
Storage needs: same as variance.

Algorithm for threshold exceedance

$$g(S_{n-1}, x_n) = g(Exc_{n-1}, x_n) = \ ifelse(x_n > threshold, Exc_{n-1} + 1, \ Exc_{n-1})$$

Memory needs:

- 1 int to store Exc_{n-1}

Algorithm for histogram

Decide the bin size of the histogram based on the precision of your variable. e.g. for wind speed, storing more than one figure is meaningless for users. Considering a range of wind speed from 0 to 100 m/s every 0.1 ms results in storing counts for a maximum of 1000 values.

Memory needs:

- k integers to store occurrences for each bin
- k doubles to store the bin centers (this could be derived from bin width as well saving a bit of space)

Memory savings: this method starts to save space whenever values repeat themselves, therefore as soon as counts for each bin start to grow larger than 1 we save space.

Algorithm for wind power CF

Compute a histogram with previous algorithm. For each bin, look in the power curve for the power output or CF value. Weight all CF values by the frequency of each bin.

Algorithm for percentiles

Compute the histogram above. Ensure a minimum number of bins in the histogram to get accurate results. Derive percentiles from the histogram by accumulating the histogram frequencies.

Algorithm for wind direction standard deviation

Difficulty: values close to 0/360 should not average to 180.Use Yamartino method.

Algorithm for covariance

Use a method similar to the variance. Let $C_n = \sum_{i=1}^{n} (x_i - \overline{x}_n)(y_i - \overline{y}_n)$, then:
$C_n = C_{n-1} + (x_n - \overline{x}_n)(y_n - \overline{y}_{n-1}) = C_{n-1} + (x_n - \overline{x}_{n-1})(y_n - \overline{y}_n)$

Algorithm for correlation

Compute correlation and the variances of the two variables.