1 Question 1

Statement 1

Why is the denominator $M-1$ instead of M?

$$\sigma_M^2 := \frac{1}{M-1}\sum_{i=1}^{M}(\xi_i-\mu_M)^2$$

简单说是因为共线性，n 个样本的自由度为 n，但是样本都减去样本均值后自由度就变为 $n-1$，所以分母上应为 $n-1$。证明思路为

$$\begin{aligned}
E\left[\frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})^2\right] &= E\left[\frac{1}{n}\sum_{i=1}^{n}\left((X_i-\mu)-(\bar{X}-\mu)\right)^2\right] \\
&= E\left[\frac{1}{n}\sum_{i=1}^{n}(X_i-\mu)^2\right] - E\left[\frac{1}{n}\sum_{i=1}^{n}(\bar{X}-\mu)^2\right] \\
&= \sigma^2 - \frac{1}{n}\sigma^2 = \frac{n-1}{n}\sigma^2
\end{aligned}$$

因此我们得到满足 $E[S^2]=\sigma^2$ 的无偏估计

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i-\bar{X})^2 .$$

2 Question 2 and 3

Statement 2

Try both cases in computer simulations and what is your observation? Prove that the $M-1$ case is better in the statistical sense.

运行程序如下，其中共进行 65536 次实验，每次生成服从正态分布 $N(0,1)$ 的 1024 组数据进行运算。无偏估计较好的实验次数为 33347，超过实验总数的一半以上。且多次运行程序结果相差不大，无一例外地无偏估计所得结果较好，所以使用 $M-1$ 作为分母在统计学上是更好的选择。

Simulations

```
/*
 * author: Iydon Liang
 * time: 2019/10/01 21:35:00
 * compile: `g++ -fconcepts 4.cpp -o 4.out`
 */
#include <cstdio>
#include <math.h>
#include <time.h>

#define _rand_ (double)rand()/RAND_MAX
#define _self_ [](double x)->double{return x;}
#define _len_(x) (sizeof(x)/sizeof(x[0]))
#define _abs_(x) (x>0 ? x:-x)
#define _tic_ int tic = time(0)
#define _toc_ int toc = time(0)
#define _debug_ true


double randn(double mu=0., double sigma=1.) {
    double V1, V2, S;
    do {
        V1 = 2*_rand_ - 1;
        V2 = 2*_rand_ - 1;
        S = V1*V1 + V2*V2;
    } while(S>=1 || S==0);
    double _randn = V1 * sqrt(-2*log(S)/S);

    return sigma*_randn + mu;
}

double E(double array[], int length, auto key) {
    double result = 0.;
    for (int ith=0; ith<length; ith++) {
        result += key(array[ith]);
    }
    return result / length;
}

double S2(double array[], int length, bool unbiased=true) {
    double mean = E(array, length, _self_);
    double result = 0.;
    double delta;
    for (int ith=0; ith<length; ith++) {
        delta = array[ith] - mean;
        result += delta * delta;
    }
    if (unbiased) return result / (length-1);
    else return result / length;
}

// main function
int main() {
    // set the random seed
    srand((int)time(0));
    double mu = 0.;
    double sigma  = 1.;
    double sigma2 = sigma * sigma;

    // new sample array
    int sample_length = 1024;
    double sample[sample_length];
    double unbiased, biased;

    // new experiment array
    int experiment_length = 65536;
    int experiment[2] = {0, 0}; // {unbiased, biased}
    int record[2] = {0, 0}; // {exceed, behind}

    // experiment
    _tic_;
    for (int ith=0; ith<experiment_length; ith++) {
        // initialize sample
        for (int jth=0; jth<sample_length; jth++)
            sample[jth] = randn(mu, sigma);

        // calculate S^2
        unbiased = S2(sample, sample_length, true);
        biased   = S2(sample, sample_length, false);

        if (_abs_(unbiased-sigma2) < _abs_(biased-sigma2))
            experiment[0] += 1;
        else experiment[1] += 1;
        if (experiment[0] > experiment[1])
            record[0] += 1;
        else if (experiment[0] < experiment[1])
            record[1] += 1;
    }
    _toc_;

    // print result
    printf("unbiased:␣%d\n", experiment[0]);
    printf("bias:␣␣␣␣␣%d\n", experiment[1]);
    printf("exceed:␣␣␣%d\n", record[0]);
```

```
        printf(”behind:␣␣␣%d\n”, record[1]);
        printf(”elapsed␣time(s):␣%d\n”, toc-tic);
}
```