Appunti Lezione 04

Lorenzo Visca

1 Controllo aggiuntivo sulla formattazione dei file

Codice di riferimento: `ioexample3.C`

Può accadere che un file di dati in input contenga delle stringhe di commento o delle righe vuote. La libreria `sstream` permette di gestire questi casi in modo semplice:

1. Il file viene letto riga per riga con la funzione `getline()`.
2. La riga viene convertita in uno stream di stringa `iss` con la funzione `istringstream`: in questo modo ogni riga permette di estrarre i dati.
3. La sezione di codice che gestisce l'estrazione dei dati viene condizionata a `(iss >> x)`: in questo modo quando si incontra un valore non numerico (come un commento) la condizione non è più rispettata e si passa alla riga successiva.

NOTA: se un commento è seguito da altri dati validi nella stessa riga questi ultimi non vengono letti; tuttavia i file di dati sono generalmente formattati in modo da non avere questo problema, evitando così controlli ulteriori che rallenterebbero la lettura del file.

2 Esercizio: verifica del teorema del limite centrale

Codice di riferimento: `central.C`

Dato un parametro $w \in [0, 0.5]$, definiamo la seguente densità di probabilità:

$$f(x) = \begin{cases} \frac{1}{2w} & \text{if } x \in [0, w] \cup [1-w, 1] \\ 0 & \text{elsewhere} \end{cases} \tag{1}$$

I momenti di questa distribuzione sono:

- $E[x] = \int_{-\infty}^{\infty} x\, f(x)\, dx = \int_0^w \frac{x}{2w}\, dx + \int_{1-w}^1 \frac{x}{2w}\, dx = \frac{1}{2}$
- $E[x^2] = \int_{-\infty}^{\infty} x^2\, f(x)\, dx = \int_0^w \frac{x^2}{2w}\, dx + \int_{1-w}^1 \frac{x^2}{2w}\, dx = \frac{1}{6}(2w^2 - 3w + 3)$
- $V[x] = E[x^2] - E[x]^2 = \frac{1}{6}(2w^2 - 3w + 3) - \frac{1}{4}$

L'esercizio consiste nell'estrarre campioni della somma di N valori di x estratti da questa distribuzione, e verificare che la distribuzione della somma tende a una gaussiana al crescere di N (teorema del limite centrale). Per ogni valore di N si utilizza un campione di 2×10^6 estrazioni.
Viene inoltre richiesto di verificare che la media e la varianza della distribuzione della somma siano coerenti con i valori attesi:

- $E[S] = N \cdot E[x]$
- $V[S] = N \cdot V[x]$

3 Generazione di numeri casuali

Codice di riferimento: `central.C`

le funzioni generatrici di numeri casuali sono di fatto funzioni che estraggono un valore da una lista di numeri pre-calcolati. In questo senso i numeri generati sono detti pseudo-casuali: l'unica garanzia è data da una distribuzione sufficientemente caotica dei numeri in questa lista, e una lunghezza sufficiente a non avere pattern ripetuti.
Alla prima estrazione viene scelto un punto di partenza nella lista, detto seed, e da questo punto in poi i numeri vengono estratti in sequenza: se il seed è lo stesso, la sequenza di numeri estratti sarà la stessa.
La sintassi per la generazione di numeri casuali in ROOT, gestita dalla libreria `TRandom3`, è la seguente:

```
TRandom3 *myptr = new TRandom3(seed);
double x = myptr->Rndm();
```

Dove `Rndm()` estrae un numero uniformemente distribuito nell'intervallo $[0, 1]$.
Se si devono generare più sequenze di numeri casuali, inizializzando generatori distinti c'è il rischio che i seed siano troppo vicini e le sequenze risultino quindi correlate.
Per evitare questo problema si può utilizzare un unico generatore globale in modo che tutte le estrazioni vengano fatte sequenzialmente; in ROOT questo generatore è accessibile tramite il puntatore globale `gRandom` (in generale i puntatori globali in ROOT iniziano con la lettera `g`):

```
double x = gRandom->Rndm();
```

3.1 Estrazione dalla distribuzione scelta

Dalla teoria delle distribuzioni di probabilità sappiamo che data una variabile x con distribuzione $f(x)$ e distribuzione cumulativa $F(x)$, la variabile $r = F(x)$ è distribuita uniformemente nell'intervallo $[0, 1]$.
Di conseguenza, estraendo una variabile r uniformemente distribuita in $[0, 1]$ si può campionare una variabile x da una distribuzione arbitraria calcolando $x = F^{-1}(r)$, quando la funzione F è invertibile.
Nel caso della distribuzione definita in (1), la funzione cumulativa è:

$$F(x) = \begin{cases} \frac{x}{2w} & \text{se } x \in [0, w] \\ \frac{1}{2} + \frac{x-(1-w)}{2w} & \text{se } x \in [1-w, 1] \end{cases}$$

La funzione inversa con cui si estrae x da r è quindi:

$$x = F^{-1}(r) = \begin{cases} 2wr & \text{se } r \in [0, \frac{1}{2}] \\ 1 - 2w(1-r) & \text{se } r \in [\frac{1}{2}, 1] \end{cases}$$

La funzione `randomeff()` implementa una versione più efficiente di questa estrazione con il cambio di variabile $r' = 2wr$, in modo da evitare una moltiplicazione risparmiando tempo di calcolo.

4 Codici

4.1 ioexample3.C

```
#include <Riostream.h>
#include <TH1D.h>
#include <TFile.h>
#include <TCanvas.h>
#include <sstream>
using namespace std;

void ioexample3(const string& fimpName, const string& histName, const unsigned int limit = 100000)
{
    ifstream in(fimpName);
    if (!in)
    {
        cout << "Il file " << fimpName << " non esiste." << endl;
        return;
    }

    double x, min, max;
    vector<double> data;

    string line;

    bool first = true;

    while(getline(in, line))
    {
        istringstream iss(line);
        while(iss >> x)
        {
            data.push_back(x);
            if (first)
            {
                min = x;
                max = x;
                first = false;
            }
            else
            {
                if(x < min) min = x;
                if(x > max) max = x;
            }

        }
    }
    if(first)
    {
        cout << "Non sono stati trovati dati validi." << endl;
        return;
    }

    in.close();

    if(data.size() == limit)
        cout << "WARNING: e' stato raggiunto il limite massimo di " << limit << " dati letti." << endl;

    cout << "\nDati letti: " << data.size() << "\nValori estremi: (" << min << ", " << max << ")" << "\n\n";

    TH1D* hist;
    hist = new TH1D("hist", "Istogramma", 100, min-1, max+1);

    for(auto y:data) hist->Fill(y);

    hist->Draw();
```

```
63
64      TFile file(histName.c_str(), "recreate");
65      hist->Write();
66      file.Close();
67  }
```

4.2 central.C

```
#include <Riostream.h>
#include <TH1D.h>
#include <TFile.h>
#include <TCanvas.h>
#include <TRandom3.h>
#include <TF1.h>
using namespace std;

double random1(const double& w)
{
    double r = gRandom->Rndm();
    return (r<0.5) ? (2.*w*r) : (1. - 2.*w*(1.-r));
}

double randomeff(const double& w)
{
    double r = 2.*w*gRandom->Rndm();
    return (r<w) ? r : (1.-2.*w+r);
}



void createhisto(const double& w, const int& entries, const int& Nsum)
{
    string name = "Sum of " + to_string(Nsum);
    string cname = "canvas" + to_string(Nsum);
    string hname = "hist" + to_string(Nsum);

    TCanvas* canvas = new TCanvas(cname.c_str(), "Teorema Limite Centrale", 800, 600);
    canvas->SetFrameFillColor(17);

    double mean = 0.5 * Nsum;
    double stdev = sqrt(((1./6.)*(2.*w*w - 3.*w + 3) - 0.25) * (double)Nsum);
    if(Nsum==1) stdev = 0.1;

    TH1D* hist = new TH1D(hname.c_str(), name.c_str(), 100, mean-5.*stdev, mean+5.*stdev)
    ;
    hist->SetLineColor(kGray+3);
    hist->SetLineWidth(3);
    hist->SetFillColorAlpha(kOrange+5, 0.75);
    for(int i=0; i<entries; i++)
    {
        double sum = 0.;
        for(int j=0; j<Nsum; j++)    sum += randomeff(w);
        hist->Fill(sum);
    }

    TF1* fit = new TF1("gaus", "gaus", mean-5.*stdev, mean+5.*stdev);
    fit->SetLineColor(kBlue+2);
    hist->Fit(fit,"RQ");

    cout << "N = " << Nsum
         << "\t mu error = " << fabs(fit->GetParameter(1)/mean -1.)
         << "\t sigma error = " << fabs(fit->GetParameter(2)/stdev -1.)
         << "\t prob = " << fit->GetProb() << endl;

    hist->Draw();
}


void central(const double& w = 0.2, const int& nHisto = 7, const int& entries = 2000000,
    unsigned int seed = 98765)
{
    gRandom->SetSeed(seed);

    if(w<0 || w>0.5)
    {
```

```
66          cout << "w deve essere compreso tra 0 e 0.5" << endl;
67          return;
68      }
69
70      cout << setprecision(5) << setiosflags(ios::scientific)
71           << "CENTRAL LIMIT THEOREM SIMULATION" << endl;
72
73      int N[] = {1, 2, 5, 10, 50, 100, 300, 900, 2000};
74
75      for(int i=0; i<nHisto; i++) createhisto(w, entries, N[i]);
76  }
```