Solução resumida do 1º trabalho de Física Computacional

MEFT/1º semestre 2020-21 (IST)

Fernando Barão, Miguel Orcinha, Jorge Vieira

Solução resumida do 1º trabalho

a) Leitura dos dados

> A classe `DataReader` deve possuir como membro **`protected`** os dados lidos. A leitura dos dados é "naturalmente" feita no constructor, ou seja, realizada aquando da instanciação da classe `DataReader` ou da classe `DataManip` que herda da primeira.

b) Representação gráfica dos dados

> O gráfico **sunspots .vs. tempo em anos**, deve possuir legendas que identifiquem o que está representado no eixo dos ***x*** e ***y***.

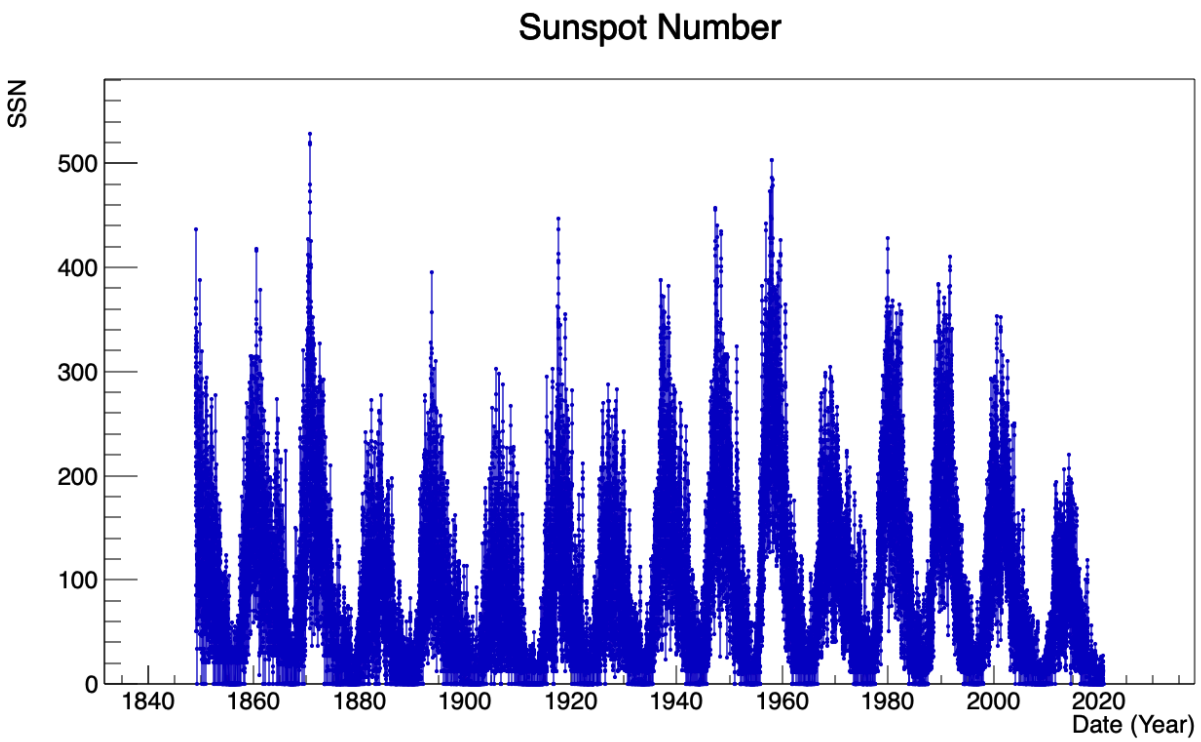


Figura 1: representação gráfica dos dados

c) O vector de dados ordenado de forma decrescente.

> Podemos utilizar a biblioteca STL e a sua função `sort` em que fornecemos como argumento a função lambda `binary predicate`, algo que retorna um booleano. Veja a documentação do `sort`[a].

[a] http://www.cplusplus.com/reference/algorithm/sort/

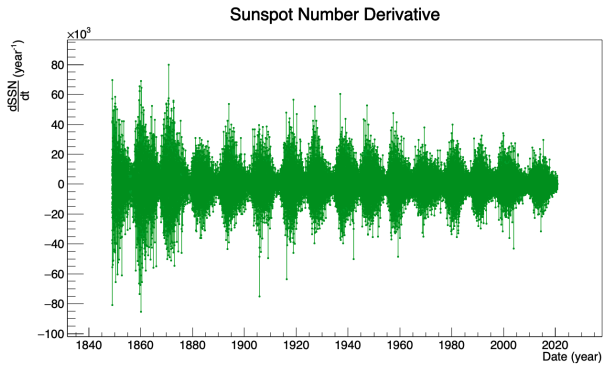
> Doc: "Binary function that accepts two elements in the range as arguments, and returns a value convertible to bool. The value returned indicates whether the element passed as first argument is considered to go before the second in the specific strict weak ordering it defines. The function shall not modify any of its arguments."

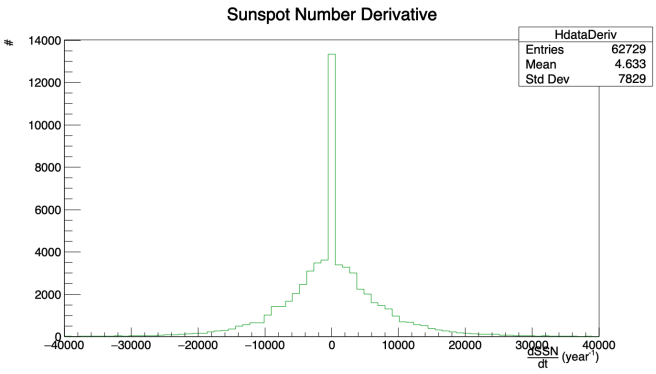
```
1 vector<pair<double,double>> DataManip::GetDataVectorSorted(int order) {
2   // x the data vector
3   auto x = GetDataVector();
4
5   // define lambda function for sorting
```

```
  auto lambda = [order](pair<double, double> before, pair<double, double> after) -> bool {
    if (order == 0) { // descending order
        return before.first > after.first;
    } else {
        return before.first < after.first;
    }
  };

  // sorting data vector<pair<double,double>>
  std::sort(x.begin(), x.end(), lambda);
```

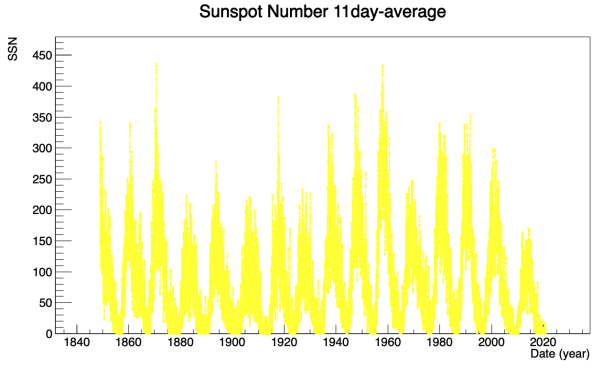
d) Cálculo das derivadas.

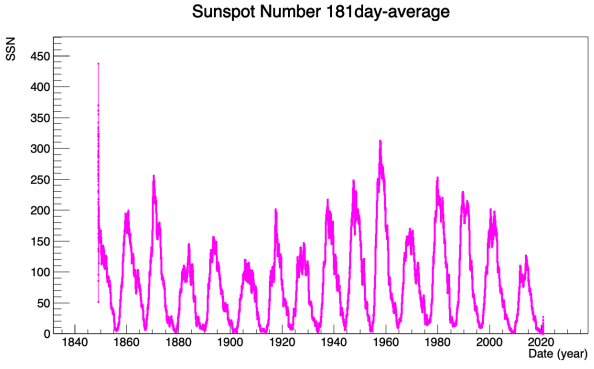
```
std::vector<std::pair<double, double>> DataManip::GetDataDerivativeVector() {

  // data
  auto x = GetDataVector();

  // lambda function returning pair with (time, dval/dt)
  auto lambda = [](pair<double,double> after, pair<double,double> before) ->pair<double,double> {
    double dt = after.first - before.first;
    double dval = after.second - before.second;

    // backward derivative: d(i) = [s(i+1) - s(i)] / Delta t_i
    return make_pair(before.first, dval/dt);
  };

  // allocate the vector of results
  vector<pair<double, double>> deriv(x.size());

  // note: unmodified copy of first pair is written to first difference pair
  std::adjacent_difference(x.begin(), x.end(), deriv.begin(), lambda);
  derive[0].second=derive[1].second; // without much knowledge make equal derivatives
  return deriv;
}
```

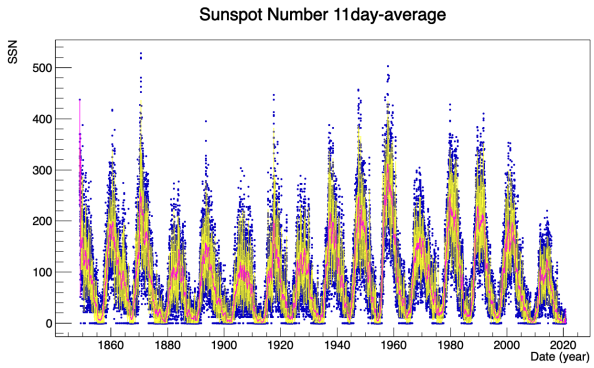


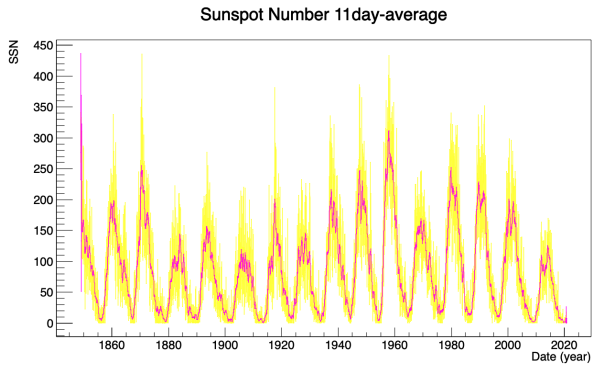


e) Média deslizante.

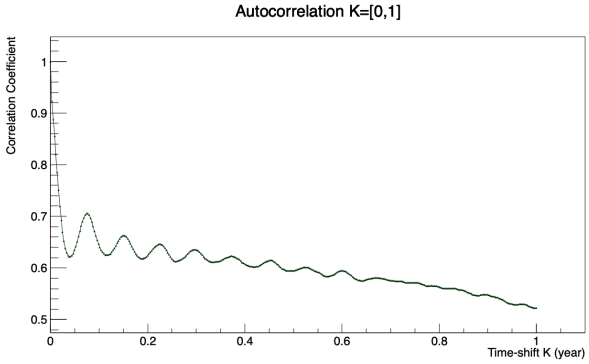


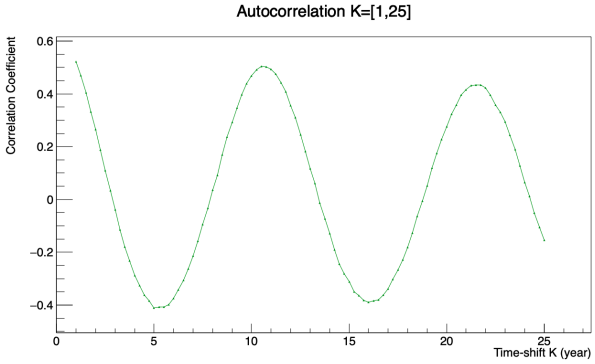






f) Auto-correlação.





> A periodicidade temporal do sinal está bem patente no valor do coeficiente de auto-correlação que atinge valores máximos para determinados valores de tempo. Como se tem acesso a vários valores de máximo de correlação que correspondem ao mesmo período pode-se fazer uma análise mais precisa do valor médio. A determinação dos máximos pode ser feita com a elaboração de uma função algoritmo para determinação do máximo,

```
1    vector<double> localmax;
2    for (int i=1; i<v.size()-1; ++i) {
3      if (x[i-1] < x[i] && x[i] > x[i+1] ) localmax.push_back(x[i]);
4    }
```

> A periodicidade que se observa resulta da existência de fenómenos cíclicos no sol:
> - o período de rotação solar é de cerca de 27 dias
> - a actividade solar que resulta na observação de uma variação temporal no número de manchas solares, tem um período de cerca de 11 anos. Veja no site da NASA[a] mais informação.
>
> [a] https://www.nasa.gov/content/goddard/does-all-solar-activity-impact-earth