

Otimização de Algoritmos de Análise de Séries Temporais Utilizando a Interface .Call()


Roger Almeida¹ Alejandro C. Frery²

¹UFAL - Bacharelado em Engenharia da Computação

²UFAL - Laboratório de Computação Científica e Análise Numérica

02 de abril de 2019

Sumário

- 1 Séries Temporais e Teoria da Informação
- 2 Imputação de dados
- 3 .Ca  a otimização dos algoritmos de imputação de dados

Sumário

- 1 Séries Temporais e Teoria da Informação
- 2 Imputação de dados
- 3 .Call() na otimização dos algoritmos de imputação de dados

Séries Temporais

~~Tratam-se de~~ conjuntos de dados, obtidos por meio de um processo observacional ao longo de um ~~determinado~~ período de tempo.

Etapas do processo de análise

~~Simbolização~~

~~Processo de simbolização de Bandt e Pompe~~

Histograma de padrões



Extração de informações

Entropias

Distâncias estocásticas

Complexidade estatística

A Simbolização de Bandt & Pompe

Dada uma série temporal $X = (x_1, x_2, \dots, x_N)$ e uma dimensão D .
Dividimos a série temporal em padrões de dimensão D , onde seus valores serão ordenados e simbolizados ~~de acordo com seu índice~~.

Por exemplo, seja $X = (4, 3, 9, 10, 6)$ uma série temporal, e considerando $D = 4$ dimensão, então obteremos dois padrões:

$$\pi_1 = 2134, \text{ pois } x_2 < x_1 < x_3 < x_4,$$

$$\pi_2 = 1423, \text{ pois } x_1 < x_4 < x_2 < x_3.$$

Distribuição de probabilidade e Teoria da Informação

Através da simbolização de Bandt & Pompe podemos extrair a distribuição de probabilidade $p(\pi)$ das $D!$ permutações π . $p(\pi)$ é dada por:

$$p(\pi) = \frac{\#\{t | t \leq N - D, (x_1, \dots, x_N) \text{ has type } \pi\}}{N - D + 1}.$$

Com isso podemos utilizar os descritores da teoria da informação para extrair informações sobre nosso sistema, como a entropia, a distância estocástica e a complexidade estatística.

Sumário

- 1 Séries Temporais e Teoria da Informação
- 2 Imputação de dados
- 3 .Call() na otimização dos algoritmos de imputação de dados

Qual é o problema?

Ao introduzirem a ideia de entropia de permutação e o método de simbolização, Bandt & Pompe assumiram como condição ~~inicial~~ que ~~os dados analisados consistiam em valores contínuos, ou seja, que na~~ ~~série temporal a ser analisada não existe~~ $x_i = x_j$, onde $i \neq j$. Mas e se ~~trabalharmos com dados discretos o que acontece se houverem~~ repetições?

Solução

Diversas estratégias ~~eficazes~~ foram apresentadas para contornar o problema da repetição de dados, trabalhamos com quatro delas:

Complete Case Imputation

Time Ordered Imputation

Random Imputation

Data Driven Imputation[5]

Algoritmos de Imputação de Dados

Complete Case Imputation

Elimina todos os padrões que contém elementos repetidos do cálculo da probabilidade.

Time Ordered Imputation

Se $x_{t1} = x_{t2}$ e $t_1 < t_2$ então $x_{t1} < x_{t2}$.

Random Imputation

Adiciona-se à probabilidade de cada padrão um peso probabilístico baseado na quantidade de elementos repetidos do padrão.

Data Driven Imputation

Adiciona-se à probabilidade de cada padrão uma perturbação extraída de uma distribuição de probabilidade calculada previamente através do método Complete Case.

Sumário

- 1 Séries Temporais e Teoria da Informação
- 2 Imputação de dados
- 3 .Call() na otimização dos algoritmos de imputação de dados

Qual o problema?

Os algoritmos em R anteriormente criados para a imputação de padrões de séries temporais se mostraram eficazes e precisos em seus resultados. Porém, tendo em vista que era necessária a análise de diversas séries temporais de tamanho muito grande, os algoritmos possuíam um tempo de execução inviável. Com isso veio a necessidade de um meio de incrementar a velocidade desses algoritmos.

Solução

Uma solução encontrada para o problema da velocidade foi a utilização da interface `.Call()`, que funciona como uma ponte entre as duas linguagens, possibilitando “chamar” funções escritas em C dentro de um código em R, e assim transmitindo dados entre as duas plataformas.

Implementação

As seguintes bibliotecas devem ser incluídas: *R.h*, *Rinternals.h*, *Rmath.h*.

```
#include<stdio.h>
#include <stdlib.h>
#include <math.h>
```



A função em C é declarada como retornando um tipo **SEXP**, que significa *Simple EXpression*, todos os dados divididos entre as duas linguagens precisam ser deste tipo, ou seja, todos os dados recebidos pela função em C são desta forma e o array de distribuição de probabilidade criado pela função também assume esse tipo.[3, 4]

```
SEXP CompleteCaseImputation(SEXP Rseries, SEXP Rdimension, SEXP Rdelay, SEXP Rpatterns, SEXP Relements, SEXP Rsymbols){
```

Implementação

O arquivo `.c` deve ser compilado com o comando *R CMD SHLIB* `arquivo.c`, isso gerará um novo arquivo do tipo `.so`.

```
roger@roger-Aspire-E5-473:~$ R CMD SHLIB BandtPompe.c
gcc -std=gnu99 -I/usr/share/R/include -DNDEBUG -fpic -g -O2 -fdebug-prefix
-map=/build/r-base-AitvI6/r-base-3.4.4=. -fstack-protector-strong -Wformat -Werr
or=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -g -c BandtPompe.c -o BandtP
ompe.o
g++ -shared -L/usr/lib/R/lib -Wl,-Bsymbolic-functions -Wl,-z,relro -o BandtPompe
.so BandtPompe.o -L/usr/lib/R/lib -lR
```



Band&Pom
pe.R



BandtPom
pe.c



BandtPom
pe.o



BandtPom
pe.so

Implementação

No programa em R, esse arquivo `.so` deve ser carregado através do comando `dyn.load(arquivo.so)`, em seguida se pode utilizar a função em C através da interface utilizada com `.Call(função, ...)`, onde após o nome da função são inseridos todos os parâmetros necessários.

```
SEXP Rprobability;  
  
Rdimension = coerceVector(Rdimension, INTSXP);  
Rdelay = coerceVector(Rdelay, INTSXP);  
Rpatterns = coerceVector(Rpatterns, INTSXP);  
Rsymbols = coerceVector(Rsymbols, INTSXP);  
Relements = coerceVector(Relements, REALSXP);  
  
dimension = INTEGER(Rdimension)[0];  
delay = INTEGER(Rdelay)[0];  
seriesize = length(Rseries);  
n = dimension;  
  
elements_aux = REAL(Relements);  
symbols_aux = INTEGER(Rsymbols);  
patterns_aux = INTEGER(Rpatterns);
```

```
1 ~ Bandt.Pompe <- function(elements, dimension, size){  
2  
3   dyn.load("BandtPompe.so")  
4  
5   probability <- .Call("BandtPompe", elements, dimension, size)  
6  
7   return (probability)  
8 }
```


Resultados

A velocidade do algoritmo original em R foi comparada com a do algoritmo utilizando C. Os algoritmos implementados e analisados até o momento foram o Complete Case e o Time Ordered. Para calcular o tempo de execução de cada uma foi utilizado o seguinte algoritmo:

```
start.time <- Sys.time()  
— função a ser analisada —  
end.time <- Sys.time()  
time.taken <- end.time - start.time
```



Onde a variável time.taken nos diz exatamente o tempo levado para executar a função testada.

Resultados

Abaixo segue uma tabela, baseada numa amostra obtida a partir de dez distintas execuções dos programas, comparando os tempos de execução das implementações do Complete Case em cada linguagem:

	Valor mínimo	Valor máximo	Média
R	31.677 78 s	34.970 58 s	33.053 12 s
C	0.321 858 s	0.365 160 5 s	0.336 052 1 s



Ou seja, tendo como base a média de tempo, é fácil ver que a versão em C é aproximadamente cem vezes mais veloz que a versão em R.

Resultados

Quanto à implementação da Time Ordered, analisando também sua velocidade em ambas as versões, com o mesmo número de amostragem, obtém-se a seguinte tabela:

	Valor mínimo	Valor máximo	Média
R	129.3993 s	142.9892 s	135.4226 s
C	1.806 375 s	1.903 321 s	1.849 567 s

Tal qual o Complete Case, vemos que a velocidade da versão em C do Time Ordered supera em muito a versão em R.

Conclusão

Com isto, temos que os algoritmos sendo desenvolvidos em C serão mais vantajosos, pois possuem a mesma precisão mas com superior velocidade, o que facilitará suas aplicações no futuro.

Referências



Avi

The need for speed part 1: Building an R package with Fortran (or C), 2018.

<http://www.r-bloggers.com/the-need-for-speed-part-1-building-an-r-package-with-fortran-or-c/>, Last accessed on 08-Feb-2019.



C. Band and B. Pompe.

Permutation entropy: a natural complexity measure for time series.

Physical review letters, 88(17):174102, 2002.



Caffo, B.

Using .call in `r`, 2018.

<http://www.biostat.jhsph.edu/~bcaffo/statcomp/files/dotCall.pdf>, Last accessed on 10-Feb-2019.



R Core Team.

Writing `r` extensions, 2018.

<https://cran.r-project.org/doc/manuals/r-release/R-exts.html>, Last accessed on 10-Feb-2019.



F. Traversaro, F. O. Redelico, M. R. Risk, A. C. Frery, and O. A. Rosso.

Bandt-pompe symbolization dynamics for time series with tied values: A data-driven approach.

Chaos: An Interdisciplinary Journal of Nonlinear Science, 28(7):075502, 2018.