# Convention d'écriture pour la bibliothèque ARCH

24 novembre 2006

## Généralités

1. les noms doivent avoir une signification.
2. Tout doit être écrit en anglais (donc IsOpen() et pas EstOuvert())
3. On met une majuscule à chaque fois qu'on change de mot et des minuscules après.
4. Pas d'"underscore" `_` sauf dans les `#define`
5. On crée un fichier ".h" et un fichier ".cpp" par classe.
6. le fichier "toto.h" commence par
   `#ifndef _TOTO_H`
   `#define _TOTO_H_`
   et se termine par
   `#endif // _TOTO_H_`
7. Les variables locales commence par `my` sauf pour les variables de boucles (i, j, k, ...)
8. Les paramètres des fonctions par `the` et on met les noms des paramètres dans les headers (pas de `void toto(int);`)
9. le nom d'un nouveau type commence par une minuscule (`typedef unsigned int uint`)

On résume ...

```
#include "kmeans.h"

void MkMeans(double *theYt, uint theT, uint theQ,
               uint theN, int* theSeq, gsl_rng* theR)
{
double** myKmeans ;
   AllocMatrix(myKmeans, theQ, theN) ;
double *myMoy,
       *myVar   ;
   AllocVector(myMean, theN) ;
   AllocVector(myVar, theN) ;
```

```
register uint i ;
   for (i = 0 ; i < theN ; i++)
       myMean[i] = myVar[i] = 0.0 ;

   for (register uint t = 0 ; t < theT ; t++)
       for (i = 0 ; i < theN ; i++)
       {   double myDt = (double)t ;
           myMean[i]=(myDt*myMean[i]+theYt[t+theT*i])/(myDt+1.0L) ;
           myVar[i]=(myDt*myVar[i]+theYt[t+theT*i]*theYt[t+theT*i])/(myDt+1.0L) ;
       }
   for(i = 0 ; i < theN ; i++)
       myVar[i] -= myMean[i]*myMean[i] ;

       ....
```

## Les typedef

### structures

le nom commence par `s` puis une majuscule. Un type pointeur sur une structure commence par `sp`. Les champs des structures commencent par `s` puis une majuscule.

### union

le nom commence par `u` puis une majuscule.

### enum

le nom commence par `e` puis une majuscule. Les différent éléments de l'énumération commencent aussi par `e` puis une majuscule.

```
 typedef enum eArch
 {  eNone = -1,
    eArch = 0,
    eGarch = 1,
    eAparch = 2
 }eArch ;

typedef struct sResArch
{  eArch           sType ;
   vector<double>  sYt   ;
   vector<double>  sEpst ;
   union uTypeArch
```

```
   {   vector<double>  sSigmat     ;
       vector<double>  sHt         ;
   };
}sResArch, *spResArch ;
```

## Classes

- Le nom de la classe commence par “c” puis une majuscule.
- Les conventions pour l'écriture des membres et des méthodes est résumée dans ce tableau :

| Préfixe | Membres | Méthodes |
|---|---|---|
| `public` | `m` | *sans préfixe* |
| `protected` | `mt` | t |
| `private` | `mv` | v |

```
#ifndef _HMM_H_
#define _HMM_H_
#include <math.h>


typedef unsigned int uint ;
typedef enum distrDefEnum
{  eUnknown = -1,
   eNormal = 0,
   eMultivariateNormal = 1,
   eNormalMixture = 2,
   eDiscrete=3
}distrDefEnum ;

class cBaumWelch
{  private :
           uint        mvNbSample   ;
           uint*       mvT          ;
           uint        mvQ          ;
   protected :
           double***   mtXsi        ;
           double*     mtLLH        ;
   public :
           double***   mAlpha       ;
           double***   mBeta        ;
           double**    mRho        ;
           double***   mGamma      ;
```

```
   public :
       cBaumWelch(uint theNbSample, uint* theT, uint theQ) ;
       cBaumWelch(const cInParameter &theInParameter) ;
       virtual ~cBaumWelch() ;
   protected :
       void tForwardBackward(double*** theProbaCond, const cHMM &theHMM) ;
} ;

class cViterbi
{  private :
       uint    mvNbSample  ;
   public :
       uint**   mSeq     ;
       double*  mLogProb  ;
   private :
       void vViterbiPath(const cInParameter &theInParameter, const cHMM &theHMM) ;
   public :
       cViterbi(const cInParameter &theInParameter) ;
       virtual ~cViterbi() ;
} ;
#endif // _HMM_H_
```