Thème 3 : Analyse de données
Statistique exploratoire et inférentielle

Les données météorologiques sont par essence aléatoires et nécessitent une description statistique. Dans ce TE, nous nous intéressons aux températures mensuelles moyennes mesurées à Paris depuis 1757...

1 Préliminaire

1.1 Fichier de données

Le fichier texte `tpparis.dat`, fourni dans le répertoire `~lefrere/M1/2013-2014/etu/mncs/te/te3/`, contient la série des températures mensuelles moyennes à Paris de 1757 à 1995 (239 années) avec une ligne par année ; chaque ligne comporte l'année (entier), suivie des 12 températures mensuelles (en °C, au dixième près) associées.

1.2 Visualisation des données sous scilab

Écrire un script qui permette de lire l'ensemble des données sous SCILAB dans une seule matrice pour simplifier [1], puis de visualiser :

a) d'une part la température au cours de l'ensemble de la période (une courbe comportant 12×239 points) : il faudra alors (a) restructurer (avec la fonction `matrix`) la matrice des températures en un vecteur colonne et (b) construire (avec la fonction `linspace`) un axe des temps gradué en années avec un point par mois. Consulter l'aide sur ces fonctions par `help linspace` par exemple. Mettre en évidence la variation saisonnière en zoomant sur quelques décennies.

b) d'autre part la température en fonction du mois de l'année pour toutes les années (superposition de 239 courbes de 12 points) : on pourra choisir de ne pas relier les points en précisant un symbole commun pour toutes les courbes via l'option `style` et la fonction `ones`. Repérer le cycle saisonnier et la dispersion mois par mois.

1.3 Consignes générales de programmation

En fortran, les tableaux seront alloués dynamiquement *dans le programme principal*. Ne pas oublier, en fin de programme, de libérer la mémoire ainsi réservée.

En C99, on préférera utiliser des tableaux automatiques quitte à les déclarer tardivement.

Les procédures à écrire par la suite comportent des tableaux en argument, qui auront été réservés par l'appelant. Les interfaces de ces procédures dépendent bien entendu du langage utilisé : il est nécessaire de transmettre explicitement les dimensions des tableaux en C alors qu'en fortran cela demeure déconseillé. Ces interfaces sont précisées dans la section 4. La compilation et l'édition de liens sont gérés par l'outil `make`.

Le programme complet comprendra les fichiers source suivants :

— `temper.f90`/`temper.c` : programme principal

— `inout.f90`/`inout.c` : contient la procédure de lecture des données et celles d'écriture des résultats

— `statistique.f90`/`statistique.c` : contient les procédures générales d'analyse statistique : `calc_moments`, `histogramme`, `gauss_pdf`, `calc_chi2` et `test_chi2`

— `anomalie.f90`/`anomalie.c` : contient la procédure `extrait_anomalie`

2 Partie 1 : Description statistique d'une série temporelle

2.1 Lecture des données en C ou fortran

Le programme principal appelle dans un premier temps la procédure `lecture` du module `inout` pour lire séquentiellement les données du fichier `tpparis.dat`. Elle les range dans 2 tableaux :

— `annees` : tableau 1D de 239 **entiers** ($1^{\text{ère}}$ colonne du fichier `tpparis.dat`)

— `temperature` : tableau 2D de 12×239 **réels** (simple précision), dont **le premier indice correspond au mois et le second à l'année**. Cette structure de tableau facilitera la programmation ultérieure en C [2]

1. En revanche, en fortran et en C, les années codées en entier et les températures en flottant, seront stockées dans deux tableaux différents.

2. Par la suite, on souhaite étudier l'évolution de la température d'un mois particulier au cours des années. En fortran, cela n'impose aucune contrainte, car on peut sélectionner des sections de tableaux 2D en fixant un quelconque des indices. Mais on C, où il s'agit de tableaux de tableaux, cela nécessite de stocker de manière contigüe les températures d'un mois donné.

Vérifier que la lecture est correcte en affichant la température d'un mois et d'une année donnés.

2.2 Calcul de moments

a) Écrire une procédure `calc_moments` qui calcule la moyenne d'un échantillon statistique mono-dimensionnel quelconque, ainsi que les moments centrés d'ordre supérieur jusqu'à l'ordre spécifié par l'utilisateur.

Les données échangées avec le programme appelant sont les suivantes :

— Données d'entrée : `nm`, entier, nombre total de moments à calculer
`tab1d`, tableau 1D de réels, contient les données échantillonnées

— Données de sortie : `moments`, tableau 1D de réels, contient les `nm` moments calculés

Le premier élément de `moments` contient la valeur moyenne de `tab1d` (moment *non centré* d'ordre 1). Si `nm` est supérieur à 1, les éléments suivants contiennent les moments *centrés* d'ordre 2 à `nm`.

b) Dans le programme principal, utiliser la procédure `calc_moments` pour estimer la moyenne mensuelle des températures, l'écart-type pour chaque mois ainsi que les paramètres d'asymétrie γ_1 et d'aplatissement γ_2.

Stocker ces valeurs dans un tableau, puis les écrire dans le fichier `tempmoy.dat`. Représenter le cycle saisonnier moyen (température du mois, son écart-type puis γ_1 et γ_2) à l'aide de SCILAB.

2.3 Calcul des « anomalies »

Les températures relevées fluctuent de manière aléatoire autour d'une valeur moyenne qui évolue elle-même au cours du temps (variation saisonnière). Afin de s'affranchir de l'effet de saison, on s'intéresse à la différence entre les températures relevées et leur valeur moyenne pour le mois considéré. On construit ainsi ce qu'on appelle la série des « anomalies » de température.

a) Écrire la procédure `extrait_anomalie`, qui calcule les anomalies mensuelles de température.

Les données échangées avec le programme appelant sont :

— Données d'entrée : `temperature`, tableau 2D de réels (12, nombre d'années)
`moyenne`, tableau 1D de réels, contient les moyennes mensuelles

— Données de sortie : `anom`, tableau 1D de réels, contient les anomalies calculées

Puisque les variations mensuelles sont éliminées, on peut modifier la structure de rangement des données et passer d'un tableau 2D de `12` fois `n_annees` à un tableau 1D de `12*n_annees` éléments, en respectant l'ordre chronologique. Cette procédure sera codée dans le fichier `anomalie.f90` ou `anomalie.c`

b) Dans le programme principal, appeler la procédure `extrait_anomalie`. Enregistrer ces anomalies dans le fichier `tempanom.dat` à raison d'une par ligne. Les représenter à l'aide de SCILAB.

2.4 Moments et histogramme des anomalies de température

a) Utiliser `calc_moments` pour estimer la moyenne, l'écart-type ainsi que les coefficients d'asymétrie et d'aplatissement de la série d'anomalies. Qu'attend-on pour la moyenne ?

b) L'étape suivante consiste à construire l'histogramme des fréquences d'occurrence empiriques des anomalies de température. On écrira donc une procédure générale `histogramme` pour compter le nombre d'échantillons d'une série quelconque dans des classes pré-établies. Noter qu'un algorithme extrêmement simple peut être utilisé si on suppose constante la largeur des classes.

Les données échangées avec le programme appelant sont :

— Données d'entrée : `tab1d` : tableau 1D de réels, données à répartir en classes
`classes` : tableau 1D de réels, classes définies par leurs bornes

— Données de sortie : `histo` : tableau 1D de réels, contenant les effectifs de chaque classe

NB 1 : si tous les intervalles sont fermés, le tableau `classes` des bornes possède un élément de plus que le tableau `histo` des effectifs.

NB 2 : les effectifs, bien que de nature entière, sont codés en réels pour faciliter la suite de la programmation.

c) Dans le programme principal, construire le tableau décrivant les bornes des classes à partir des informations fournies par l'utilisateur : bornes de l'histogramme et largeur (constante) des différentes classes. Puis appeler la procédure `histogramme` pour calculer les effectifs.

Enregistrer l'histogramme dans le fichier `histo1.dat`, puis représenter l'histogramme des anomalies de températures. Quelle est l'allure de la loi suivie par les anomalies ?

3 Partie 2 : Tests sur la loi de probabilité

L'histogramme construit suggère que la distribution des anomalies de température suit une loi normale. Pour décider dans quelle mesure cette distribution peut être représentée par la loi normale de même moyenne et de même écart-type, on utilise le test du χ^2.

3.1 Loi normale de même moyenne et écart-type

a) Écrire la procédure `gauss_pdf`, permettant de calculer, pour une variable gaussienne de moyenne et écart-type donnés, les probabilités d'une suite de classes fournies.

Les données échangées avec le programme appelant sont :

— Données d'entrée : `mu`, `sigma` : réels, valeur moyenne et écart-type de la gaussienne
`classes` : tableau réel 1D des classes, définies par leurs bornes

— Données de sortie : `proba` : tableau réel 1D des probabilités d'intervalle de chaque classe

En C, on utilisera la fonction d'erreur (`erf`) de la bibliothèque mathématique. En fortran `erf` relève du standard 2008 : avec `g95`, l'option `-fintrinsic-extensions=erf` permet d'accéder à cette fonction intrinsèque non-standard ; avec `gfortran`, on est obligé d'autoriser toutes les fonctions non-standard par l'option `-fall-intrinsics`.

b) Utiliser la procédure `gauss_pdf` pour construire l'histogramme d'une variable aléatoire gaussienne de mêmes paramètres que la série d'anomalies étudiée (effectif, moyenne et écart-type identiques). Enregistrer les deux histogrammes dans le fichier `histo2.dat`. Tracer l'histogramme des anomalies de températures, ainsi que celui de la variable gaussienne associée. Commenter.

3.2 Test du χ^2

Il faut donc quantifier l'écart entre les deux distributions, puis tester dans quelle mesure il est acceptable.

a) Programmer la procédure `calc_chi2`, qui calcule la quantité D caractérisant l'écart entre ces distributions :

$$D = \sum_{i=1}^{k} \frac{\left(N_i - \widetilde{N_i}\right)^2}{\widetilde{N_i}} \quad \text{où la somme est restreinte aux classes d'effectif supérieur à 5,}$$

N_i est la population expérimentale de la classe i et $\widetilde{N_i} = Np_i$ l'effectif attendu déduit de la probabilité p_i et de l'effectif total N. Elle calculera aussi le nombre de degrés de liberté en retranchant au nombre de classes effectivement prises en compte le nombre de contraintes imposées en choisissant les paramètres de la loi.

Les données échangées avec le programme appelant sont les suivantes :

Données d'entrée : `histo` : tableau réel 1D, effectifs mesurés
`eff_att` : tableau réel 1D, effectifs théoriques (pour la loi testée)
`n_contraintes` : entier,

Données de sortie : `chi2` : réel, écart D entre les deux histogrammes `histo` et `eff_att`
`n_dl` : entier, nombre de degrés de liberté

en fonction du nombre de classes retenues (de 5 à 30 par exemple).

b) On cherche donc à établir si l'hypothèse H0 : « l'échantillon des anomalies de température est une réalisation d'une variable aléatoire gaussienne de même moyenne et de même écart-type » est statistiquement acceptable. On se fixe un risque (faible) de rejeter abusivement l'hypothèse H0 ; on en déduit le seuil au-delà duquel l'écart D conduit à rejeter H0. La détermination du seuil en fonction du risque et du nombre de degrés de liberté peut se faire à l'aide d'une table de distribution cumulée du χ^2. On peut aussi utiliser la fonction `cdfchi` de SCILAB avec le mot clef `"X"` : `cdf("X", n_dl, p, 1-p)`.

Fixer un risque de 5 %, en déduire le seuil associé, puis la décision. Reprendre le test avec un risque de 1 %.

Mais on peut aussi calculer le seuil en inversant par une méthode itérative simple la fonction de répartition cumulée F du χ^2 à n degrés de liberté, qui s'exprime à l'aide de la fonction gamma incomplète Γ :

$$F(x) = P\left(\sum_{k=1}^{n} X_k^2 < x\right) = \Gamma\left(\frac{n}{2}, \frac{x}{2}\right) \quad \text{où} \quad \Gamma(a, x) = \frac{\int_0^x e^{-t}\, t^{a-1}\, dt}{\int_0^\infty e^{-t}\, t^{a-1}\, dt}$$

Le code de calcul de la fonction gamma **incomplète**, est fourni dans les fichiers `gamma.h`/`gamma.c` et `gamma.f90` des sous-répertoires appropriés de `~lefrere/M1/2013-2014/etu/mncs/te/te3/`

`function gammap(a, x)` `a` et `x` sont des réels, `gammap` renvoie une valeur réelle.

Appeler la procédure `test_chi2` pour décider si l'hypothèse H0 doit être rejetée.

4 Interfaces des procédures à écrire

4.1 Langage fortran

inout.f90

```
subroutine lecture (nomfic, annees, temperature)
  character(len=*), intent(in) :: nomfic
  integer, dimension(:), intent(out) :: annees
  real, dimension(:,:), intent(out) :: temperature
```

statistique.f90

```
subroutine calc_moments (tab1d, moments)
  real, dimension(:), intent(in) :: tab1d
  real, dimension(:), intent(out) :: moments

subroutine histogramme (tab1d, classes, histo)
  real, dimension(:), intent(in) :: tab1d, classes
  real, dimension(:), intent(out) :: histo

subroutine gauss_pdf (mu, sigma, classes, proba)
  real, intent(in) :: mu, sigma
  real, dimension(:), intent(in) :: classes
  real, dimension(:), intent(out) :: proba

subroutine calc_chi2 (histo, eff_att, n_contraintes, n_dl, chi2)
  real, dimension(:), intent(in) :: histo, eff_att
  integer, intent(in) :: n_contraintes
  integer, intent(out) :: n_dl
  real, intent(out) :: chi2

subroutine test_chi2 (risque, n_dl, seuil)
  real, intent(in) :: risque
  integer, intent(in) :: n_dl
  real, intent(out) :: seuil
```

anomalie.f90

```
subroutine extrait_anomalie (temperature, moyenne, anom)
  real, dimension(:,:), intent(in) :: temperature
  real, dimension(:), intent(in) :: moyenne
  real, dimension(:), intent(out) :: anom
```

4.2 Langage C99

inout.c

```
void lecture (char * nomfic, int n_mois, int n_annees, int annees[n_annees],
              float temperature[n_mois][n_annees]) ;
```

statistique.c

```
void calc_moments (int n1d, float tab1d[n1d], int nm, float moments[nm]) ;

void histogramme (int n1d, float tab1d[n1d], int n_classes,
                  float classes[n_classes+1], float histo[n_classes]) ;

void gauss_pdf (float mu, float sigma, int n_classes,
                float classes[n_classes+1], float proba[n_classes]) ;

void calc_chi2 (int n_classes, float histo[n_classes], float eff_att[n_classes],
                int n_contraintes, float *ptr_chi2, int *ptr_n_dl) ;

void test_chi2 (float risque,  int n_dl, float * seuil);
```

anomalie.c

```
void extrait_anomalie (int n_annees, int n_mois,
                       float temperature[n_mois][n_annees],
                       float moyenne[nmois], float anom[n_mois*n_annees]) ;
```

N.-B. : `n_classes` représente le nombre de classes, donc le tableau `classes` est de dimension `n_classes+1` puisqu'il contient les bornes des classes...