UGA - L3 STE - Mathématiques appliquées 2018-2019

TP1 - correction

Exercice :

2.1 Les formats de données correspondent à la manière dont sont codées les données du fichier.

A chaque codage (format) correspond une application qui décode le format et le redirige vers l'utilisateur, sous la forme d'une page à l'écran, d'un fichier imprimable, d'un son, d'une vidéo…

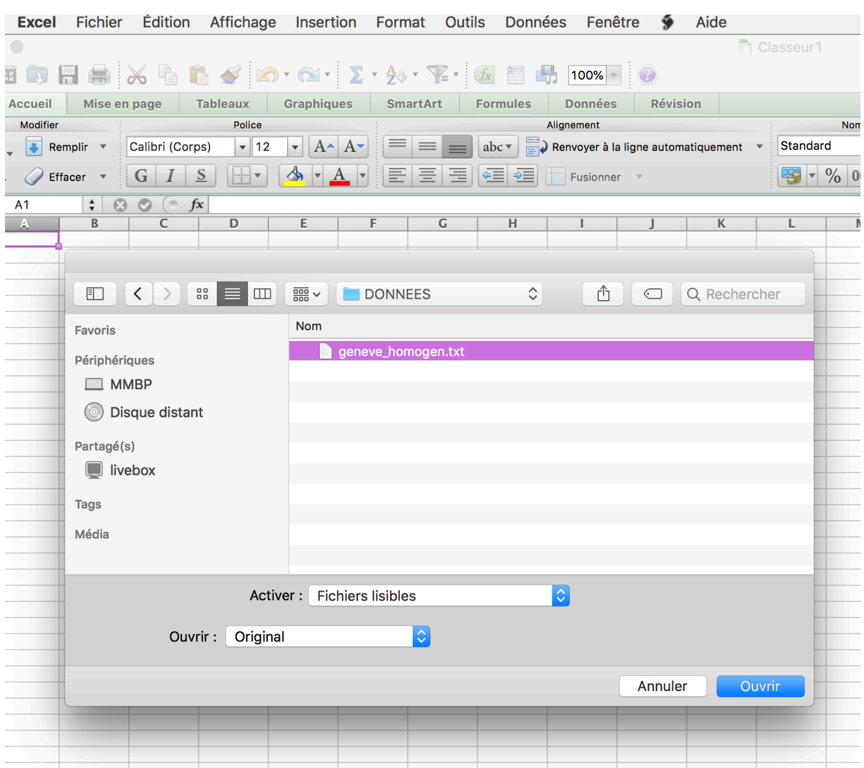
Les fichiers codés en ASCII sont visualisables grâce à plusieurs applications dont les tableurs (excel, libreoffice, openoffice,..) et les éditeurs de texte tels que bloc-notes, notepad, word, gedit, xemacs, openoffice, libreoffice,…

2.2 Importations des données de températures de la stations de Genève Cointrin.

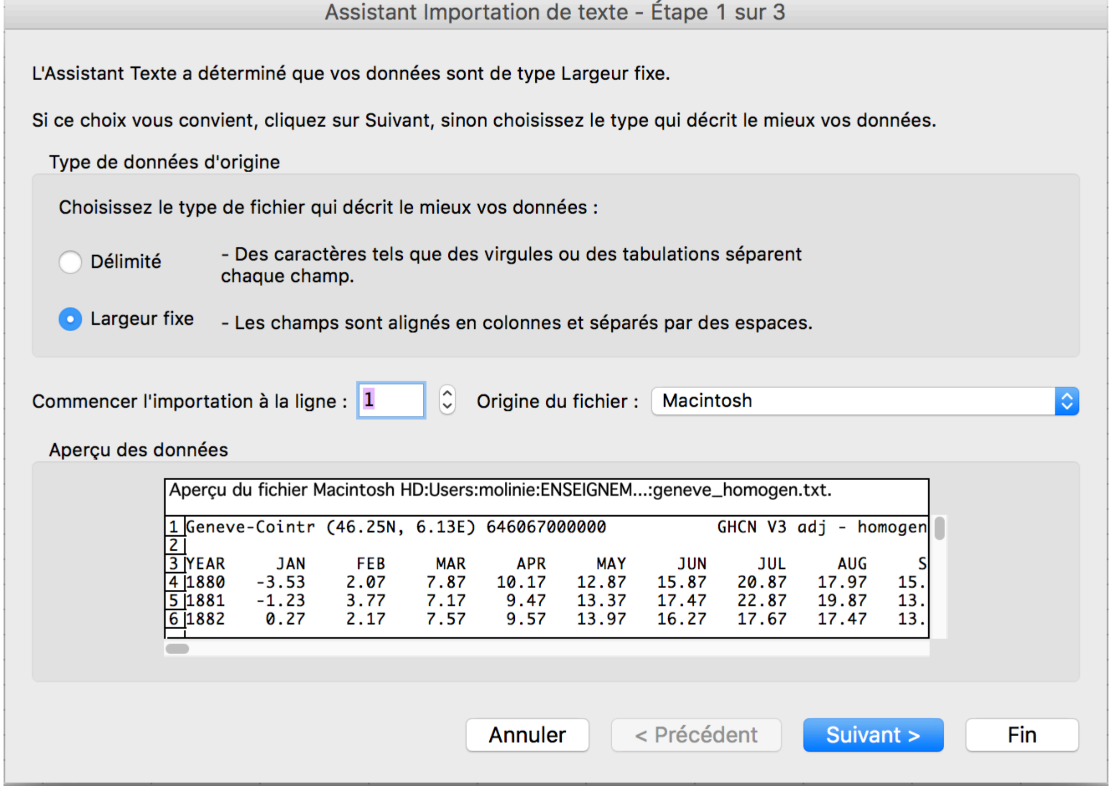
Dans un tableur :

Ouvrir un fichier -> sélectionné geneve_homogen.txt

Choisir des champs de données de largeur fixe comme c'est le cas pour ce fichier (cas particulier).



Dans un logiciel



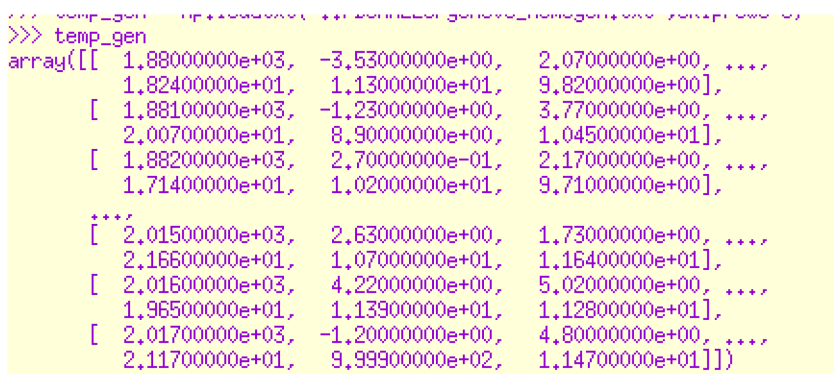
programmable :

Python :

```
# bibliotheques
import numpy as np
import matplotlib.pyplot as plt
#-----------------------------------------------------------------------------
```

```
#Lecture des donnees dans un fichier.
temp_gen = np.loadtxt('../DONNEES/geneve_homogen.txt',skiprows=3);
```

Une fois ce script exécuté, les données se trouvent stockées dans la variable temp_gen.
On peut les visualiser :


```
>>> temp_gen
array([[  1.88000000e+03,  -3.53000000e+00,   2.07000000e+00, ...,
          1.82400000e+01,   1.13000000e+01,   9.82000000e+00],
       [  1.88100000e+03,  -1.23000000e+00,   3.77000000e+00, ...,
          2.00700000e+01,   8.90000000e+00,   1.04500000e+01],
       [  1.88200000e+03,   2.70000000e-01,   2.17000000e+00, ...,
          1.71400000e+01,   1.02000000e+01,   9.71000000e+00],
       ...,
       [  2.01500000e+03,   2.63000000e+00,   1.73000000e+00, ...,
          2.16600000e+01,   1.07000000e+01,   1.16400000e+01],
       [  2.01600000e+03,   4.22000000e+00,   5.02000000e+00, ...,
          1.96500000e+01,   1.13900000e+01,   1.12800000e+01],
       [  2.01700000e+03,  -1.20000000e+00,   4.80000000e+00, ...,
          2.11700000e+01,   9.99900000e+02,   1.14700000e+01]])
```

On peut visualiser la première colonne d'indice 0 qui indique les années ou une autre:

```
>>> temp_gen[:,0]
array([ 1880.,  1881.,  1882.,  1883.,  1884.,  1885.,  1886.,  1887.,
        1888.,  1889.,  1890.,  1891.,  1892.,  1893.,  1894.,  1895.,
        1896.,  1897.,  1898.,  1899.,  1900.,  1901.,  1902.,  1903.,
        1904.,  1905.,  1906.,  1907.,  1908.,  1909.,  1910.,  1911.,
        1912.,  1913.,  1914.,  1915.,  1916.,  1917.,  1918.,  1919.,
        1920.,  1921.,  1922.,  1923.,  1924.,  1925.,  1926.,  1927.,
        1928.,  1929.,  1930.,  1931.,  1932.,  1933.,  1934.,  1935.,
        1936.,  1937.,  1938.,  1939.,  1940.,  1941.,  1942.,  1943.,
        1944.,  1945.,  1946.,  1947.,  1948.,  1949.,  1950.,  1951.,
        1952.,  1953.,  1954.,  1955.,  1956.,  1957.,  1958.,  1959.,
        1960.,  1961.,  1962.,  1963.,  1964.,  1965.,  1966.,  1967.,
        1968.,  1969.,  1970.,  1971.,  1972.,  1973.,  1974.,  1975.,
        1976.,  1977.,  1978.,  1979.,  1980.,  1981.,  1982.,  1983.,
        1984.,  1985.,  1986.,  1987.,  1988.,  1989.,  1990.,  1991.,
        1992.,  1993.,  1994.,  1995.,  1996.,  1997.,  1998.,  1999.,
        2000.,  2001.,  2002.,  2003.,  2004.,  2005.,  2006.,  2007.,
        2008.,  2009.,  2010.,  2011.,  2012.,  2013.,  2014.,  2015.,
        2016.,  2017.])
```

En R :

```
# Read data file

temp_gen = read.table("../DONNEES/geneve_homogen.txt",
  header = TRUE,
  skip=2)
```

On peut aussi visualiser les données :

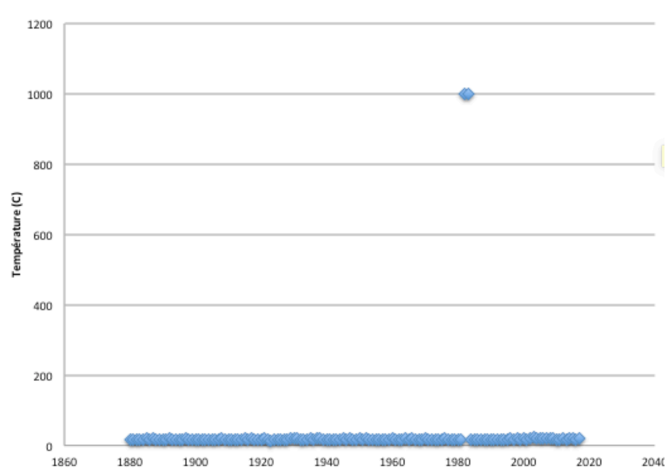
On voit sur la copie d'écran ci-dessous qu'il y a plusieurs manières d'explorer une même colonne avec R. Soit en évoquant son nom (repéré par le logiciel dans l'entête du fichier) soit par son indice (7 pour le mois de juin).

```
> temp_gen[['YEAR']]
  [1] 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894
 [16] 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909
 [31] 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924
 [46] 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939
 [61] 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954
 [76] 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969
 [91] 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
[106] 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
[121] 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014
[136] 2015 2016 2017
> temp_gen[['JUN']]
  [1]  15.87  17.47  16.27  16.67  15.27  18.57  16.57  18.57  18.07  18.37
 [11]  17.17  17.27  18.47  18.27  16.87  17.67  16.76  19.06  16.16  17.26
 [21]  18.16  18.16  16.36  15.66  18.06  18.06  17.46  16.86  18.76  15.46
 [31]  17.46  16.76  17.05  17.15  16.25  18.85  15.25  19.15  15.65  17.75
 [41]  17.45  18.64  18.24  14.64  17.44  18.34  15.04  17.44  17.93  18.53
 [51]  19.83  20.03  16.83  15.23  18.33  19.33  17.23  18.53  18.53  17.33
 [61]  16.83  17.63  18.03  17.13  17.43  20.52  17.22  19.52  16.42  17.72
 [71]  20.12  16.82  19.42  16.02  18.22  17.32  16.02  17.19  16.49  17.99
 [81]  18.79  18.14  16.84  16.74  18.44  17.84  19.04  16.84  17.24  15.22
 [91]  18.61  15.99  15.97  17.75  15.94  15.92  20.10  16.19  16.27  17.55
[101]  15.23  17.12 999.90 999.90  16.67  15.85  17.73  15.81  17.40  17.78
[111]  17.06  17.05  16.93  18.31  18.39  16.58  18.86  17.14  18.63  17.31
[121]  20.09  17.47  20.76  24.34  19.02  20.71  19.69  18.77  18.65  18.44
[131]  17.72  17.90  19.19  17.07  19.15  19.93  17.72  20.90
>  temp_gen[:,7]
Error: unexpected ':' in " temp_gen[:"
>  temp_gen[,7]
  [1]  15.87  17.47  16.27  16.67  15.27  18.57  16.57  18.57  18.07  18.37
 [11]  17.17  17.27  18.47  18.27  16.87  17.67  16.76  19.06  16.16  17.26
 [21]  18.16  18.16  16.36  15.66  18.06  18.06  17.46  16.86  18.76  15.46
 [31]  17.46  16.76  17.05  17.15  16.25  18.85  15.25  19.15  15.65  17.75
 [41]  17.45  18.64  18.24  14.64  17.44  18.34  15.04  17.44  17.93  18.53
 [51]  19.83  20.03  16.83  15.23  18.33  19.33  17.23  18.53  18.53  17.33
 [61]  16.83  17.63  18.03  17.13  17.43  20.52  17.22  19.52  16.42  17.72
 [71]  20.12  16.82  19.42  16.02  18.22  17.32  16.02  17.19  16.49  17.99
 [81]  18.79  18.14  16.84  16.74  18.44  17.84  19.04  16.84  17.24  15.22
 [91]  18.61  15.99  15.97  17.75  15.94  15.92  20.10  16.19  16.27  17.55
[101]  15.23  17.12 999.90 999.90  16.67  15.85  17.73  15.81  17.40  17.78
[111]  17.06  17.05  16.93  18.31  18.39  16.58  18.86  17.14  18.63  17.31
[121]  20.09  17.47  20.76  24.34  19.02  20.71  19.69  18.77  18.65  18.44
[131]  17.72  17.90  19.19  17.07  19.15  19.93  17.72  20.90
```

2.4 Graphe des températures :

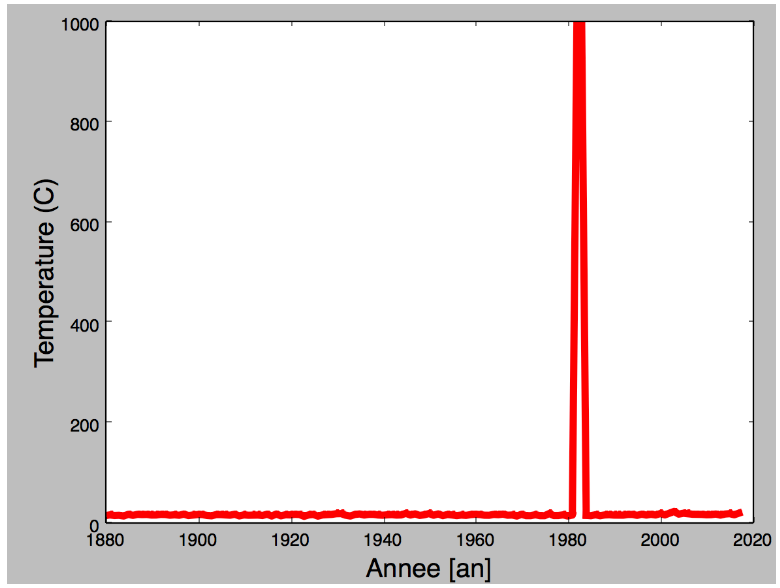
Tableur :

Sélectionner les 2 colonnes à tracer et insérer un graphe.



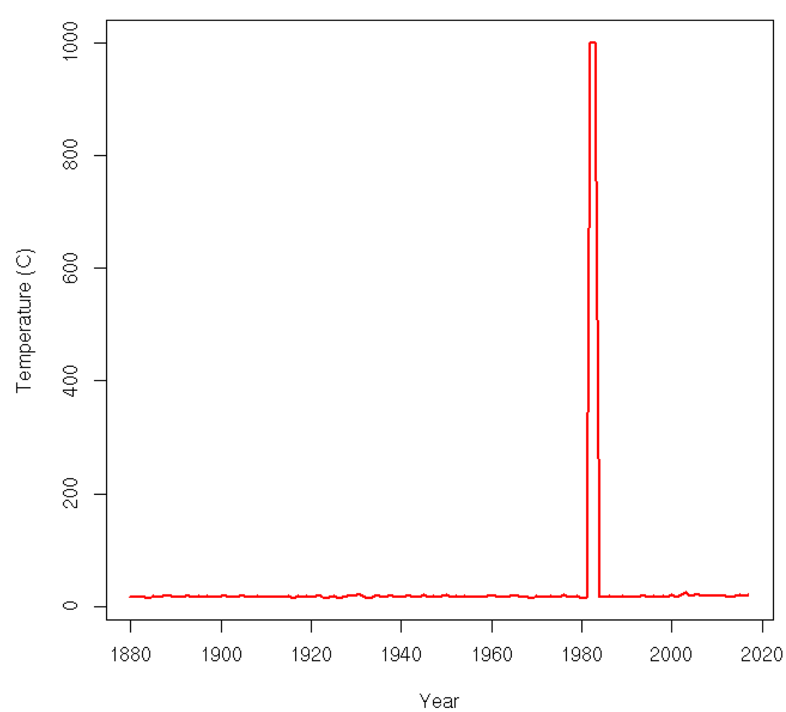
Python :

```
# bibliothÃ¨ques
import numpy as np
import matplotlib.pyplot as plt
#-----------------------------------------------------------------------------

#Lecture des donnees dans un fichier.
temp_gen = np.loadtxt('../DONNEES/
geneve_homogen.txt',skiprows=3);

print(temp_gen[:,0] );
# Temperature mois du juin

year = temp_gen[:,0];
tjuin = temp_gen[:,6];

plt.plot(year,
         tjuin,
         marker='+',
         color='r',
         ms=3,mec='r',
         mew=2,
         linestyle='-',
         linewidth=5,
         label='Temperature Juin');
plt.xlabel("Annee [an]", fontsize=18);
plt.ylabel("Temperature (C)", fontsize=18);
plt.show(block=False)
```



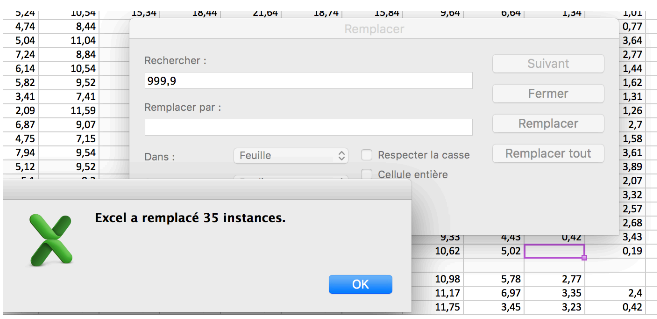
En R :

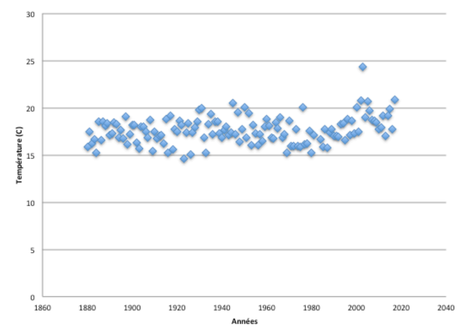
```
# Read data file
temp_gen = read.table("../DONNEES/
geneve_homogen.txt",
  header = TRUE,
  skip=2)

temp_gen[['YEAR']]

# Temperature mois du juin
temp_gen[['JUN']]
year = temp_gen[['YEAR']]
tjuin = temp_gen[,7]

plot(year,tjuin,
     xlab='Year',
     ylab ='Temperature (C)',
     type='l',
     lwd=2,
     col=2)
```



2.4 Les données non valides sont signalées soit par des valeurs improbables, soit par marqueurs non numériques.

Ici 999.9 est une valeur improbables. Elle signifie qu'à l'instant de cette mesure la donnée est invalide. Il faut l'ignorée lors du traitement des données.
Dans un tableur on peut la remplacer par une case vide avec la fonction remplacer pour être sur de n'oublier aucune données erronées :

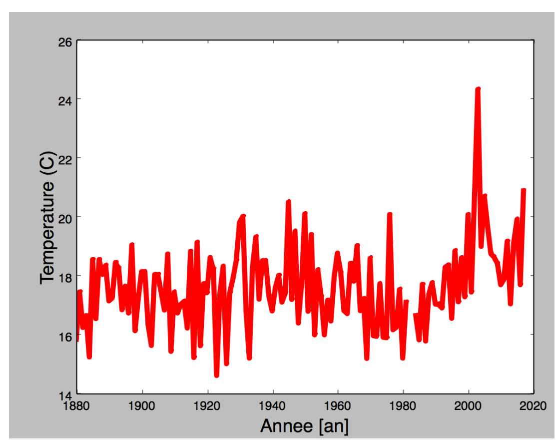




Dans un logiciel de programmation, on remplace ces valeurs par le drapeau non numérique NaN.

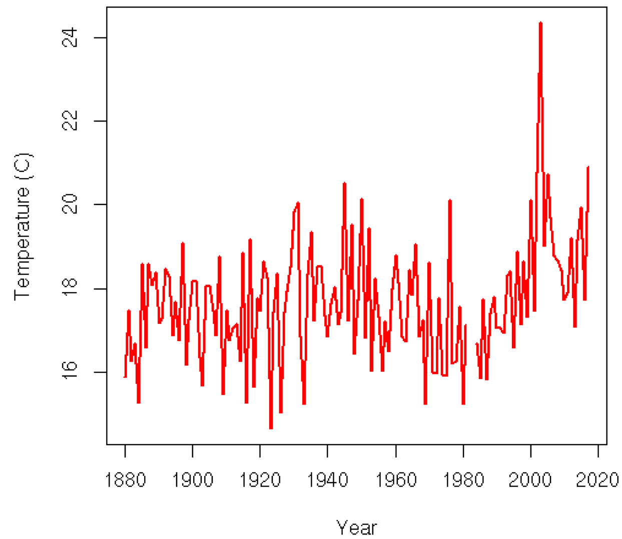
Python :

tjuin[tjuin > 999.] = np.nan



R :

tjuin[tjuin >=999] = NaN



2.6 Les donnés affichées ont des valeurs de températures probables. Il faut les caractériser.

Comme indiqué dans le cours, la moyenne est une statistique centrale qui donne un ordre de grandeur des données.
Les fonctions donnant les statistiques usuelles sont programmées dans les logiciels qu'on utilise. Toutefois, pour prendre en main ces statistiques, on compare la valeur calculé en appliquant la formule brute et en utilisant une fonction prédéfinie :

$$\bar{x} = \frac{\sum_{i=1}^{c} f_i \cdot x_i}{n} = \sum_{i=1}^{c} \frac{f_i}{n} \cdot x_i \qquad s = \sqrt{\frac{\sum_{i=1}^{c} f_i (x_i - \bar{x})^2}{n-1}}$$

Tableur

Nombre de donnees valides	
136	
moyenne	
=SOMME(G2:G138)/136	

Nombre de donnees valides	
136	
moyenne	
17,5133824	

Pour calculer l'écart-type, on fait une colonne égale au carré de la colonne G moins la moyenne (17.51 C). C'est dans mon tableur la colonne X:

Nombre de donnees valides				3,93272968
135				0,17379635
				0,17379635
moyenne		ecart type		0,03352968
17,6431111		=RACINE((1/T26)*SOMME(X2:X138))		

On obtient un écart-type de 2.53 qui signifie que l'écart moyen entre les mesures et leur moyenne est de 2.56 C.

Ecart-type
=ECARTYPE(G2:G138)

Ecart-type
1,4164189

Python :

nbvalue = tjuin.size - np.count_nonzero(np.isnan(tjuin))
mtjformule = np.nansum(tjuin) /nbvalue

```
>>> mtjformule
17.667058823529416
```

R :

```
> # Nombre de donnees valide
> nbvalue = length(which(!is.nan(tjuin)))
> mtjformule = sum(tjuin,na.rm=TRUE) /nbvalue
> mtjformule
[1] 17.66706
```

Calculs similaires avec les fonctions intégrées :

Python

```
>>> mtj = np.nanmean(tjuin)
>>> sdtjuin = np.nanstd(tjuin)
>>> mintjuin = np.nanmin(tjuin)
>>> maxtjuin = np.nanmax(tjuin)
>>> mtj
17.667058823529416
>>> sdtjuin
1.4332342680553056
>>> mintjuin
14.640000000000001
>>> maxtjuin
24.34
```

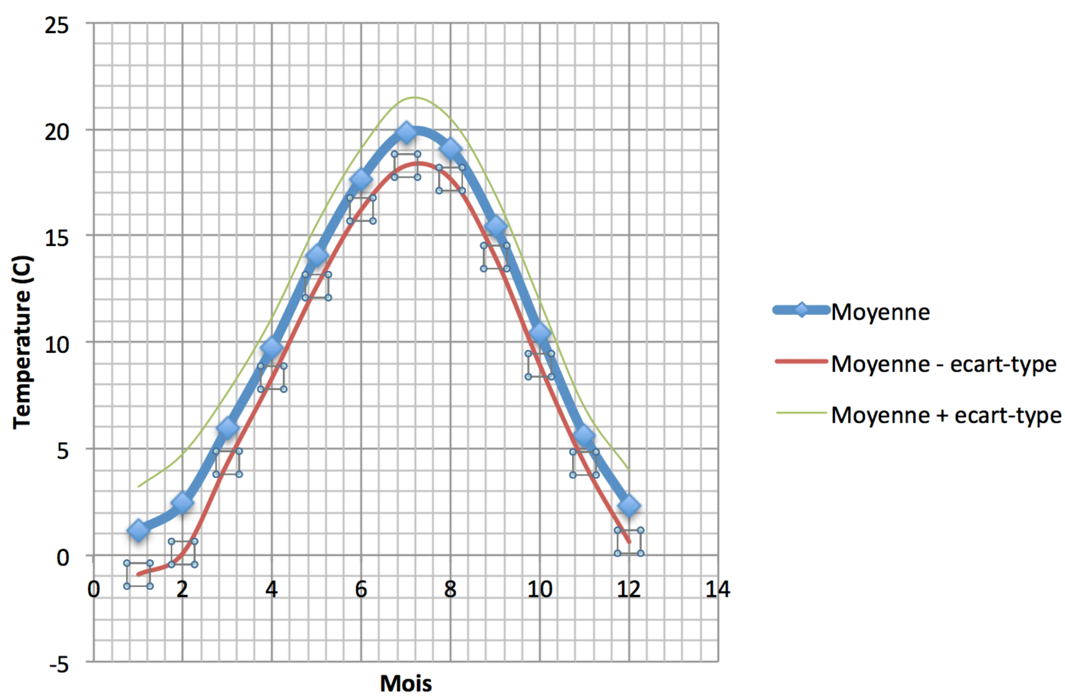
R

```
>
> mtj = mean(tjuin, na.rm=TRUE)
> sdtjuin = sd(tjuin, na.rm=TRUE)
> mintjuin = min(tjuin, na.rm=TRUE)
> maxjuin = max(tjuin, na.rm=TRUE)
> mtj
[1] 17.66706
> sdtjuin
[1] 1.438533
> mintjuin
[1] 14.64
> maxjuin
[1] 24.34
>
```

2.6 Normales saisonnières

Tableur :

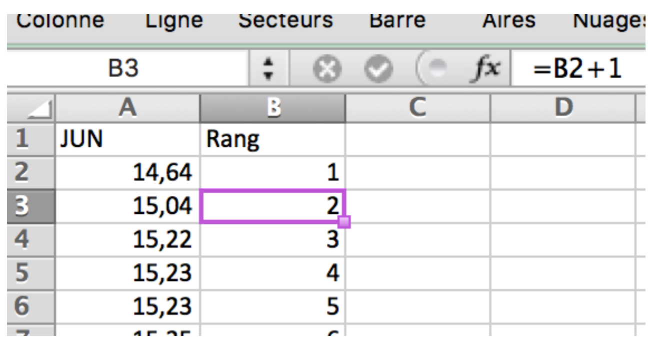
Voir la feuille de calcul statistiques de base



2.7 Simulation d’une distribution Gaussienne.

$$p(x)=\frac{1}{\sqrt{2\pi}\cdot\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Tableur



Colonne Ligne Secteurs Barre Aires Nuages

B3 =B2+1

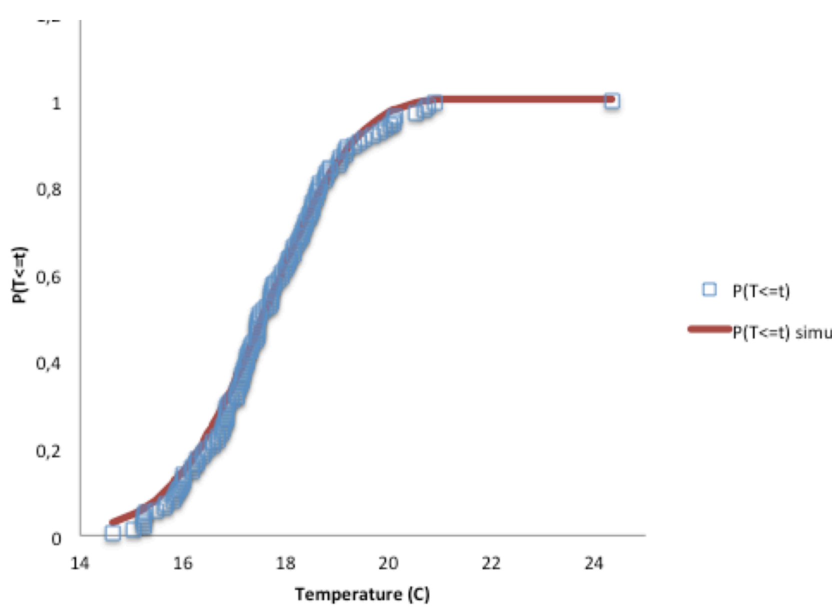
	A	B	C	D
1	JUN	Rang		
2	14,64	1		
3	15,04	2		
4	15,22	3		
5	15,23	4		
6	15,23	5		

Ici x est la température. p(x) est une fonction de densité. p(x)dx est la probabilité que x appartienne à l’intervalle [x;x+dx]. Comme les tableurs ne permettent pas de construire des histogrammes automatiquement, on comparera la fonction de répartition P(X<= x) calculée avec la formule ci-dessus et celle empirique déterminée avec les données.

Calcul de la fonction de répartition empirique : Nouvelle feuille de calcul, dans une colonne copier les données de température de juin. Les trier par ordre croissant. Dans la colonne voisine, calculer le rang de chaque valeur. Le rang d’une ligne est l’incrément de la précédente.

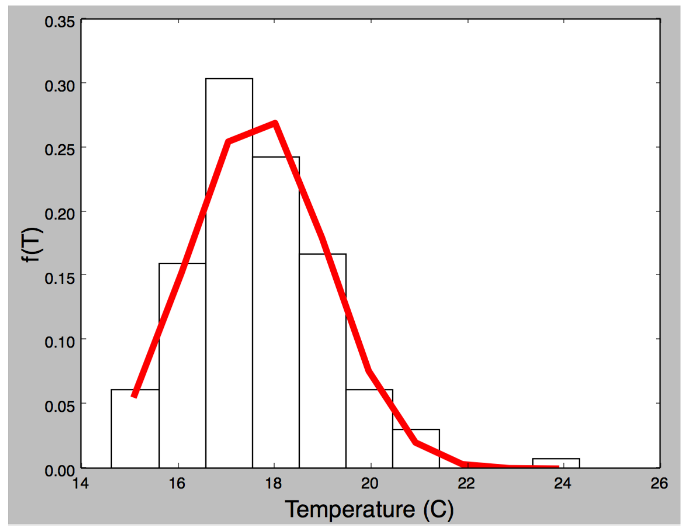
Convertir le rang en fonction de répartition P(T<=t). La probabilité que la température soit inférieure ou égale à 14.64 est 1/136. Elle est de 2/136 pour 15.04 et ainsi de suite.

On obtient le graphe de la fonction de répartition.
Sur ce type de logiciel, il est difficile de faire un histogramme, on trace la fonction de répartition simulée comme la somme cumulée des fonctions de densité multiplié par l’écart de température entre 2 valeurs consécutives.



Python

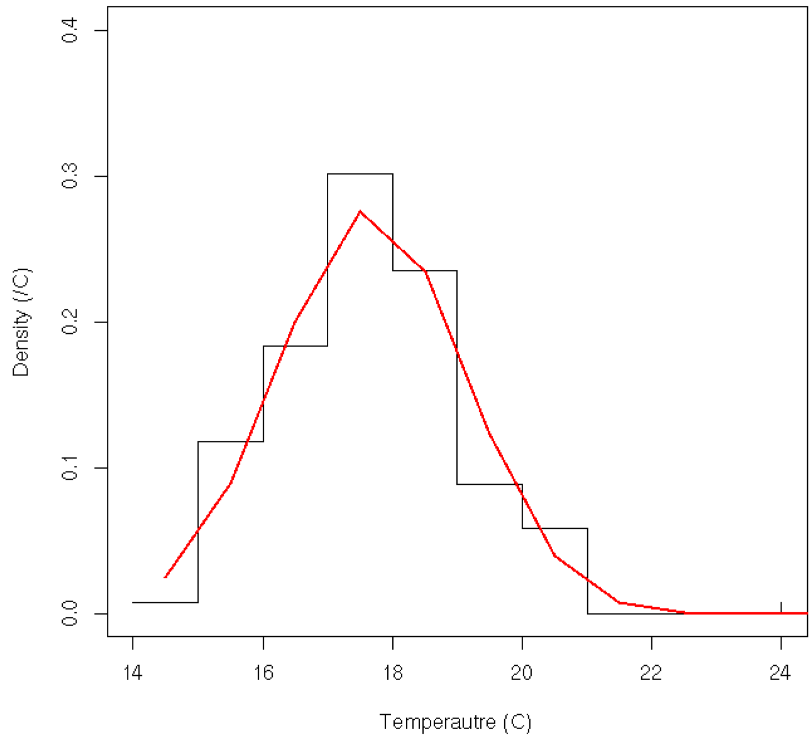
```
tjuin = [x for x in tjuin if (np.isnan(x) == False)]

# Calcul de la densite simulee
hist, bin_edges = np.histogram(tjuin,
density=True)
Inbbox = np.count_nonzero(bin_edges)
Fdeltatemp =  np.diff(bin_edges)
Ftempboxmids =  bin_edges[0:Inbbox-1] +
np.diff(bin_edges)/2

Fsim_normal_dens = (1/
(2*np.pi*Fsdtjuin**2))**0.5 *
np.exp( ((Ftempboxmids - Fmtj)**2 /
(-2*Fsdtjuin**2)))

plt.hist(tjuin,
        normed=True,
        color='w',
        label='Histogramme empirique')

plt.plot(Ftempboxmids,
        Fsim_normal_dens,
        color='r',
        ms=3,
        mec='r',
        mew=2,
        linewidth=5,
        label='fonction de densite');



plt.xlabel("Temperature (C)", fontsize=18)
plt.ylabel("f(T)", fontsize=18);

plt.show()
```



R

```
Fhist = hist(tjuin
  ,plot = FALSE)

# Calcul de la densite simulee
Fdeltatemp = diff(Fhist$breaks)
```

```
Ftempboxmids = Fhist$mids
lnbbox = length(Fdeltatemp)

plot(Ftempboxmids - Fdeltatemp/2
     ,Fhist$density
     ,type ='s'
     ,xlab ='Temperautre (C)'
     ,ylab ='Density (/C)'
     ,ylim=c(0,0.4))


Fsim_normal_dens =
  (1/(2*pi*Fsdtjuin^2))^0.5 *
  exp(
      ((Ftempboxmids - Fmtj)^2 /
(-2*Fsdtjuin^2))
      )
#*  Fdeltatemp


lines(Ftempboxmids
     ,Fsim_normal_dens
     ,lwd =2
     ,col=2)
```



2.8 Température du mois de juin 2003

Le maximum des mois de juin se produit en 2003 :

PYTHON

```
lind = np.where(np.isnan(tjuin) == False)
tjuin = tjuin[lind]
year = year[lind]

lind = np.where(tjuin >= Fmaxtjuin)
print(year[lind])
[ 2003.]
```

Variable centrée réduite.

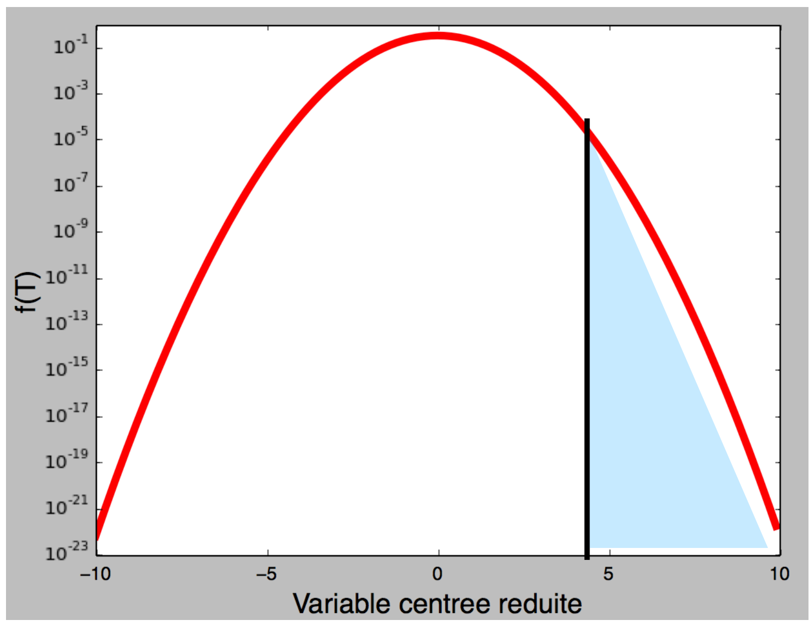
Rapport (T2003 - Tjuin)/σ. = 4.66

Le calcul de ce rapport suggère qu'on va chercher une probabilité dans un tableau standard de la loi normale. Plutôt que de chercher dans le tableau, on trace la fonction de densité pour la loi normale centré réduite, c'est à dire de moyenne 0 et d'écart-type 1.

$$P(t < T < t + dt) = \frac{1}{\sqrt{2\pi}} e^{-t^2} dt$$

PYTHON

```
# Fonction de densite de la loi normale centree reduite.

Fx = [x/10.-10 for x in range(200)]
Fsim_normal_dens = (1/(2*np.pi))**0.5 *
np.exp( [(x**2)/(-2) for x in Fx])

plt.plot(Fx,
        Fsim_normal_dens,
        color='r',
        ms=3,
        mec='r',
        mew=2,
        linewidth=5,
        label='fonction de densite');
plt.yscale('log')

plt.xlabel("Variable centree reduite",
fontsize=18)
plt.ylabel("f(T)", fontsize=18);

plt.show()
```



On cherche l'aire sous la courbe pour t > 4.66;

$$P(T > t2003) = \int_{4.66}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2} dt$$

Discrétisé en :

$$t = t0 + i\Delta t$$

$$P(T > t2003) = \frac{1}{\sqrt{2\pi}} \sum_i e^{-(to+i\Delta t)^2} \Delta t$$

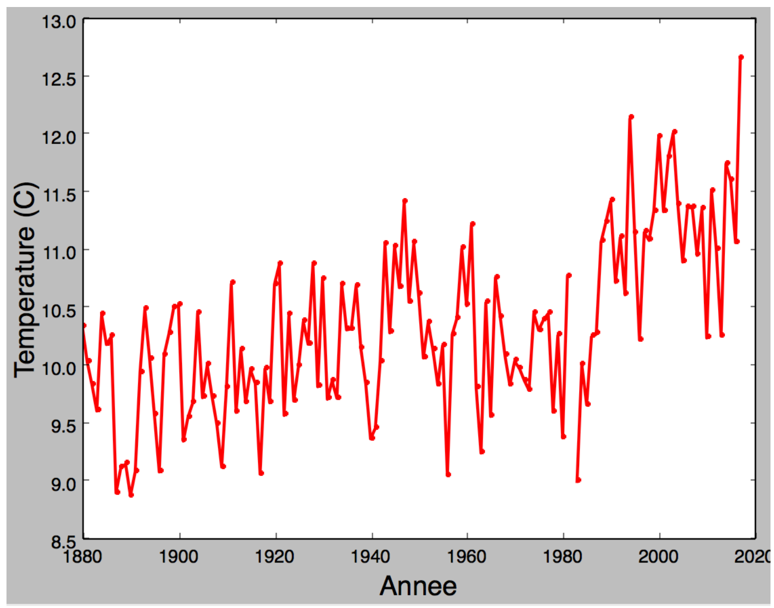
PYTHON

```
# P(X > 4.66)
Fdeltat= 0.01
Ft0 = 4.66
Ft = [Ft0 + Fdeltat*i for i in range(200)]
Fproba = (1/(2*np.pi))**0.5 * np.exp( [(x**2)/(-2) for x in Ft])*Fdeltat
print 'La probabilite que T> T2003 est ', np.sum(Fproba)
```

La probabilite que T> T2003 est 1.6197368884e-06

2.9 Evolution de la temperature moyenne annuelle

PYTHON



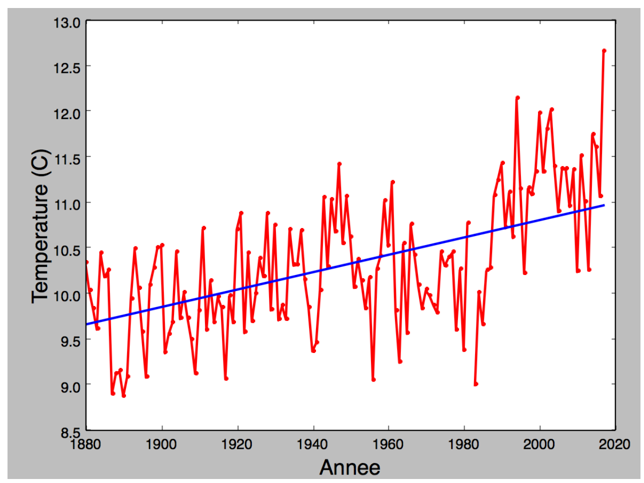

```
#2.9 Evolution des temperatures
moyennes annuelles
Ftemp = temp_gen[:,1:13]
Iind = np.where(Ftemp >= 999.)
Ftemp[Iind] = np.nan

Ftempmoyan =
np.nanmean(Ftemp
                        ,axis=1)
year = temp_gen[:,0]

plt.plot(year,
        Ftempmoyan,
        color='r',
        marker='+',
        ms=3,
        mec='r',
        mew=2,
        linewidth=2,
        label='temperature moyenne annuelle');


plt.ylabel("Temperature (C)", fontsize=18)
plt.xlabel("Annee", fontsize=18);
plt.show()
```

Modele linéaire T = a * annnee + b

$$\boxed{a=\frac{\overline{x\,y}-\bar{x}\,\bar{y}}{\overline{x^2}-\bar{x}^2}} \quad \text{et} \quad \boxed{b=\bar{y}-a\,\bar{x}}$$

PYTHON

```
# Determination du modele lineaire

Fybar = np.nanmean(Ftempmoyan)
Fxbar = np.nanmean(year)
Fxybar = np.nanmean(Ftempmoyan[:] * year[:])
Fxbarybar = Fxbar * Fybar
Fx2bar = np.nanmean(year[:] * year[:])
Fxbar2 = Fxbar * Fxbar

Fa = (Fxybar - Fxbarybar) / (Fx2bar-Fxbar2)
Fb = Fybar - Fa*Fxbar

plt.plot(year,
        Fa * year[:] + Fb,
        color='b',
        linewidth=2,
        label='Regression lineaire');
```




```
plt.show()

print 'Pente  = ' , Fa
print 'Ordonnee a l origine =' , Fb


Pente  =  0.0095434507334
>>
Ordonnee a l origine = -8.27367741251
>>
```

La température moyenne annuelle à augmenté de 0.01 C par an depuis 1880 soit 1.3C.

Qualité de la prédiction [modifier | modifier le code]

Pour évaluer la qualité de la prédiction, on peut utiliser différents critères.

Dans un premier temps rappelons que :

- $\mathrm{SSR} = \sum_{i=1}^{n} (\hat{y_i} - \bar{y})^2$ est la variation expliquée par la régression (Sum of Squares Regression, en français SCE Somme des Carrés Expliquée [par la régression]).
- $\mathrm{SSE} = \mathrm{SCR} = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$ est la variation expliquée par les résidus (Sum of Squares Errors, en français SCR Somme des Carrés Résiduelle).
- $\mathrm{SST} = \mathrm{SSE} + \mathrm{SSR} = \sum_{i=1}^{n} (y_i - \bar{y})^2$ est la variation totale (Sum of Squares Total, en français SCT Somme des Carrés Totale).

Nous pouvons alors définir le coefficient de détermination (R^2) comme le ratio entre la somme des carrés des écarts à la moyenne des valeurs prédites par la régression et la somme des carrés des écarts à la moyenne totale :

$$R^2 = \frac{\mathrm{SSR}}{\mathrm{SST}} = \frac{\mathrm{SST} - \mathrm{SSE}}{\mathrm{SST}} = \frac{\sum_{i=1}^{n} (\hat{y_i} - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Le coefficient de détermination varie entre 0 et 1. Lorsqu'il est proche de 0, le pouvoir prédictif du modèle est faible et lorsqu'il est proche de 1, le pouvoir prédictif du modèle est fort.

PYTHON

```
 Fr = np.nansum((Fa * year[:] + Fb-Fybar)**2) /np.nansum((Ftempmoyan - Fybar)**2)
>>> Fr
0.26650951506077153
```

25% des écarts de chaque température annuelle avec la température moyenne est expliqué par une augmentation linéaire de 0.01C/an depuis 1880.

L’ajustement des données de température n’améliorerait certainement par la qualité de la régréssion. Il semble y avoir une cassure entre les années 60 et 90.