Chronique 16

Diagrammes

16.1 Matriochkas

Quand on charge une extension, il se peut que celle-ci charge aussi une ou plusieurs autres extensions qui elles-mêmes peuvent charger d'autres extensions, etc.
Par exemple, l'extension `pst-all` appelle différentes extensions dont on a déjà parlé (ou pas !) : `pstricks`, `pst-plot`, `pst-node`, `pst-tree`, `pst-grad`, `pst-coil`, `pst-text`, `pst-3d`, `pst-eps`, `pst-fill`, `pstricks-add` et `multido`.
Quand on charge `pstricks-add`, on charge en plus `pstricks` et `pst-math`, puis on recharge `pst-plot`, `pst-node`, `pst-3d` et `multido`.
Et quand on charge...
Bref, vous avez compris.
Donc en appelant l'extension `pst-all` par `\usepackage{pst-all}`, on charge la plupart des extensions dont on a besoin pour travailler en `psTricks` ; à partir de maintenant, on pourra donc se contenter de charger cette extension en laissant tomber les `\usepackage{pstricks-add}` et autres `\usepackage{pst-tree}`.
Au passage, l'extension `xcolor` est chargée par `pstricks` ; on peut donc également supprimer `\usepackage{xcolor}`.

Comment vérifier tout ça ?
Un « package » est un fichier ayant pour extension `sty`. En éditant le fichier `pst-all.sty`, par exemple avec l'éditeur LaTeX que vous utilisez, vous verrez au début du fichier :

- la ligne `\RequirePackage{pstricks}` qui charge le package `pstricks` ;
- la ligne `\RequirePackage{pst-plot}` qui charge le package `pst-plot` ;
- etc.

16.2 Diagramme à barres

Voici un tableau donnant la répartition des notes à un test :

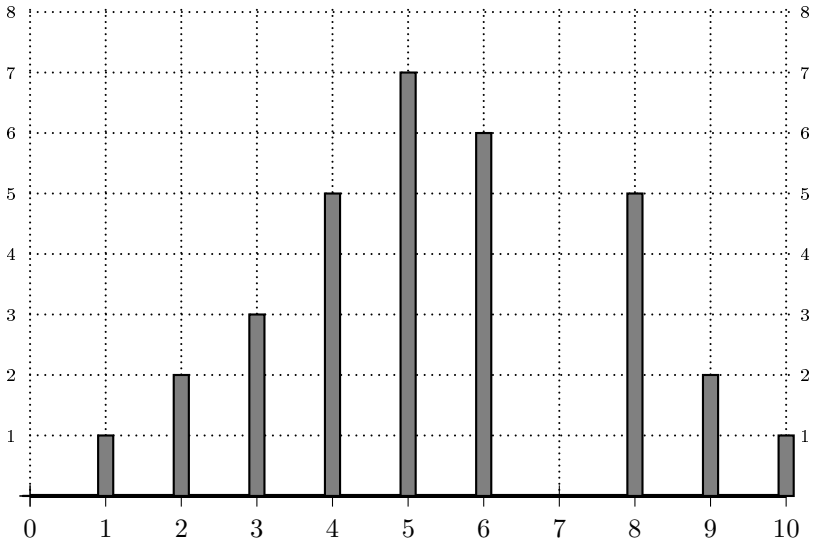
Notes x_i	0	1	2	3	4	5	6	7	8	9	10
Effectifs y_i	0	1	2	3	5	7	6	0	5	2	1

On veut représenter cette série par un diagramme en barres.
Pour cela on va utiliser l'instruction `\listplot` dont on va définir les paramètres :

`plotstyle = bar`	pour tracer des barres ;
`barwidth = 0.2cm`	pour définir la largeur des barres ;
`fillcolor = black`	pour définir la couleur de remplissage des barres ;
`fillstyle = solid`	pour définir le style de remplissage des barres.

À part `solid`, les valeurs qu'on peut donner à `fillstyle` sont `crosshatch`, `vlines` et `hlines`, et leurs versions étoilées `crosshatch*`, `vlines*` et `hlines*`. À essayer !
Ensuite il faut définir les données, le plus simple étant de les placer dans une variable. Les données sont entrées sous la forme d'une liste x_1 y_1 x_2 y_2 etc. (abscisses et ordonnées séparées par un espace).
On peut entrer les notes dans un ordre quelconque (ce que je déconseille) et il vaut mieux entrer les notes ayant pour effectif 0. Et pour une meilleure lisibilité, on entrera les données en colonnes.
Voici comment on définit les données appelées `\data` et le graphique obtenu :

```
\def\data
{
0 0
1 1
2 2
3 3
4 5
5 7
6 6
7 0
8 5
9 2
10 1
}
```



Le code complet du graphique est :

```
\psset{xunit=1cm,yunit=0.8cm}
\begin{pspicture}(-1,-1)(10,8)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0](0,0)(10,8)% quadrillage
\psaxes[linewidth=1.5pt](0,0)(10,0)%       axe horizontal seulement
\multido{\n=1+1}
        {8}%                               nombre de graduations
        {
        \uput[l](0,\n){\scriptsize \n}%    graduations côté gauche
        \uput[r](10,\n){\scriptsize \n}%   graduations côté droit
        }
\listplot[plotstyle=bar,barwidth=0.2cm,%
          fillcolor=gray,fillstyle=solid]{\data}%  tracés des barres
\end{pspicture}
```

En changeant les paramètres de `\listplot`, on peut avoir des effets plus originaux.
Voici deux diagrammes basés sur la même série de données ; seule la ligne `\listplot` a été modifiée.
Pour le premier diagramme, on a entré :

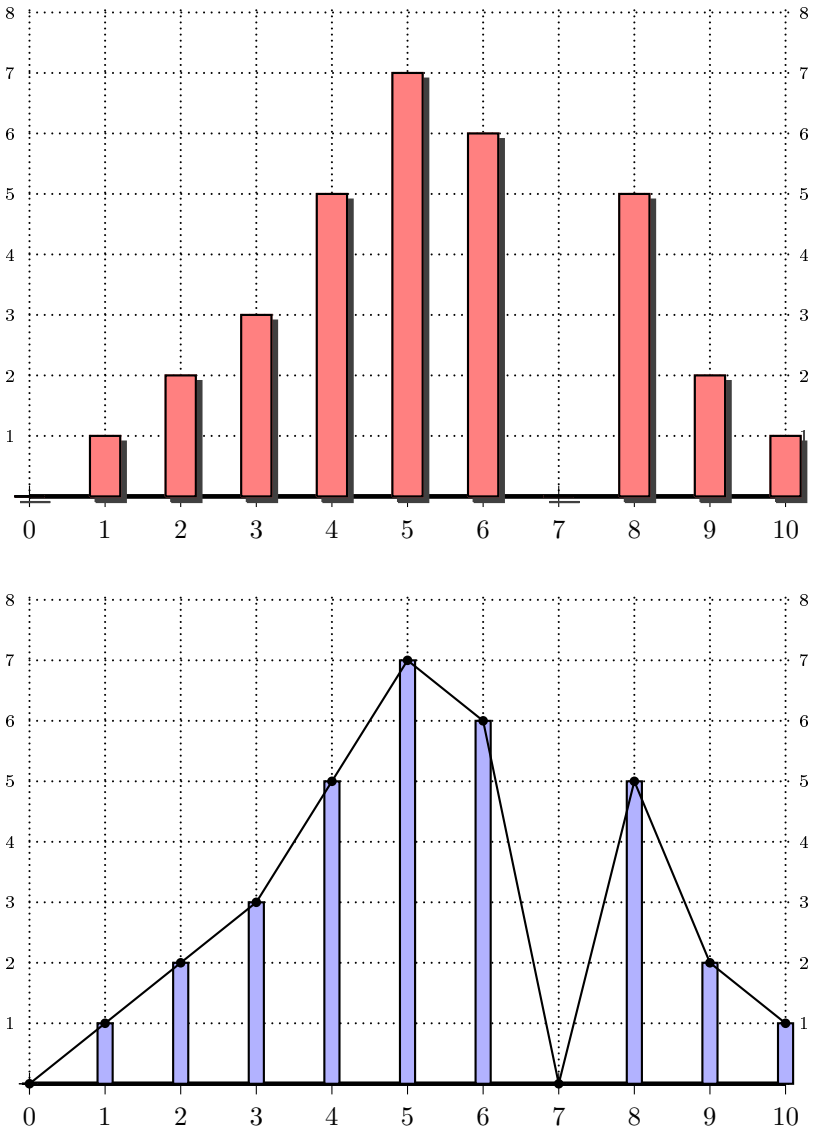
```
\listplot[shadow=true,plotstyle=bar,%
          barwidth=0.4cm,fillcolor=red!50,fillstyle=solid]{\data}
```

Deux explications sur les paramètres : `shadow=true` trace une ombre derrière chaque rectangle, et `fillcolor=red!50` remplit de rouge à 50 %.

Pour le deuxième diagramme, on a entré :

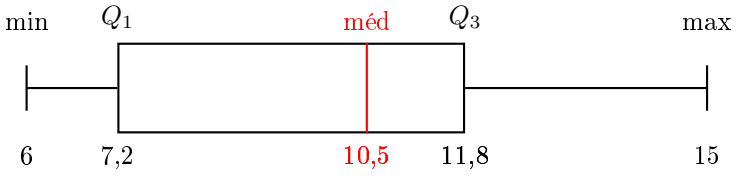
```
\listplot[plotstyle=bar,barwidth=0.2cm,%
          fillcolor=blue!30,fillstyle=solid]{\data}
\listplot[showpoints=true]{\data}
```

Le deuxième `\listplot` permet de tracer une courbe qui relie les sommets des rectangles, avec affichage des points défini par `showpoints=true`.
Refaites ce même graphique en intervertissant les données dans la définion de `\data` pour voir !



16.3 Diagramme en boîte

Voilà un diagramme en boîte qui représente une série dans laquelle le minimum est 6, le premier quartile 7,2, la médiane 10,5, le troisième quartile 11,8 et le maximum 15 :



Ce diagramme est obtenu par l'instruction :

```
\diagboite{6}{7.2}{10.5}{11.8}{15}{1cm}
```

dans laquelle `\diagboite` est une nouvelle commande que l'on va décrire.

Cette commande `\diagboite` a besoin de six paramètres : le minimum de la série, son premier quartile, sa médiane, son troisième quartile et son maximum, qu'il faut entrer impérativement dans cet ordre sous peine de catastrophe ! Le dernier paramètre est l'unité en x qu'il faut ajuster en fonction des données que l'on veut représenter.
L'unité en y est fixée à 0,3 cm.

On remarque que l'on rentre comme paramètre le nombre 10.5 avec un point comme séparateur décimal, et qu'à l'affichage on a 10,5 avec une virgule : c'est le package `numprint` qui fait ce travail. Il faut donc avoir entré dans le préambule : `\usepackage[np]{numprint}`.

Le code de cette commande est :

```
\newcommand{\diagboite}[6]
{
%%%%%%%%%%%      \diagboite{min}{Q1}{méd}{Q3}{max}{xunit}
\psset{xunit=#6,yunit=0.3cm}%           unités
\begin{pspicture}(#1,-5)(#5,5)%         zone de dessin
%%%%  dessin du diagramme
\psline(#1,-1)(#1,1)%                   petit trait vertical de début
\psline(#1,0)(#2,0)%                    moustache gauche de min à Q1
\psframe(#2,-2)(#4,2)%                  boîte de Q1 à Q3
\psline[linecolor=red](#3,-2)(#3,2)%    trait vertical rouge de médiane
\psline(#4,0)(#5,0)%                    moustache droite de Q3 à max
\psline(#5,-1)(#5,1)%                   petit trait vertical de fin
%%%% affichage des valeurs
\uput[d](#1,-2){\np{#1}}%               minimum
\uput[d](#2,-2){\np{#2}}%               Q1
\uput[d](#3,-2){\red \np{#3}}%          médiane
\uput[d](#4,-2){\np{#4}}%               Q3
\uput[d](#5,-2){\np{#5}}%               maximum
%%%% labels
\uput[u](#1,2){min}%                    min
\uput[u](#2,2){$Q_1$}%                  Q1
\uput[u](#3,2){\red méd}%               méd
\uput[u](#4,2){$Q_3$}%                  Q3
\uput[u](#5,2){max}%                    max
\end{pspicture}
}
```

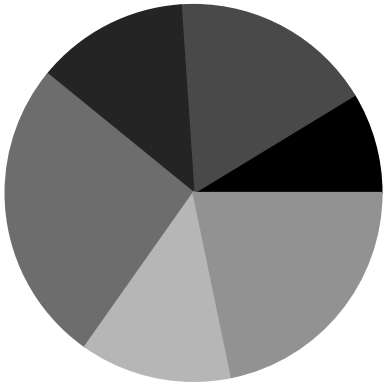
16.4 Diagramme circulaire

Autre diagramme qu'il est très facile de construire en LaTeX, le diagramme circulaire ; pas besoin de créer une nouvelle instruction, celle qu'il faut existe : c'est `\psChart` (attention au C majuscule).

Admettons que l'on ait à représenter par un diagramme circulaire la série suivante :

A	B	C	D	E	F
10	20	15	30	15	25

Voici un diagramme circulaire (obtenu sans aucune précaution) et son code (assez simple) :



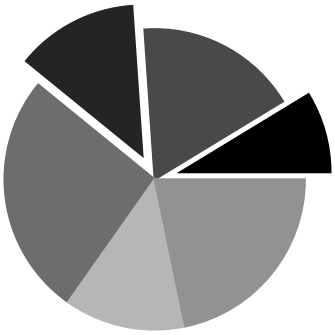
```
\psset{unit=1cm}
\begin{pspicture*}(-2,-2)(2,3)
\psChart{10,20,15,30,15,25}{}{2.5}
\end{pspicture*}
```

On voit que `\psChart` a besoin de trois paramètres :

- le premier est la liste des valeurs, séparées par une virgule ;
- le troisième est le rayon du cercle qui sera tracé ;
- le deuxième est la liste des secteurs qui vont être éclatés.

Exemple de diagramme avec des secteurs éclatés (1 et 3) :

```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psChart{10,20,15,30,15,25}{1,3}{2}
\end{pspicture*}
```



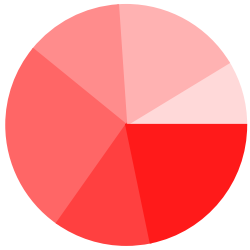
L'éclatement peut être modifié avec la variable `chartSep` qui, normalement, est à 10 points.

Par défaut, le diagramme circulaire est en niveaux de gris ; on peut naturellement demander des couleurs avec l'option `chartColor=color`.

```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psChart[chartColor=color]{10,20,15,30,15,25}{}{2}
\end{pspicture*}
```

Les couleurs sont prédéfinies, mais on peut les changer si on le souhaite ; il suffit d'utiliser la variable `userColor`.

Pour tracer des secteurs en niveaux de rouge, on entrera :



```
\psset{unit=1cm}
\begin{pspicture*}(-2,-2)(2,2)
\psChart[chartColor=color,%
   userColor={red!15,red!30,red!45,red!60,red!75,red!90}]%
   {10,20,15,30,15,25}{}{1.6}
\end{pspicture*}
```

Il suffit d'entrer la liste des couleurs souhaitées, une par secteur.

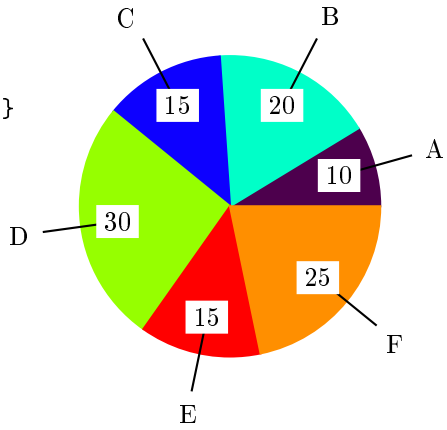
Enfin, il est très facile de marquer des labels correspondant aux secteurs circulaires ; la commande `\psChart` définit automatiquement des points pour chaque secteur.

Dans le diagramme ci-contre de rayon r, on a défini trois points pour le secteur 2, situés sur la bissectrice de l'angle :

- le point intérieur est situé à une distance de $0{,}75\,r$ du centre et a pour coordonnées `psChartI2`, où `I` signifie `In` et où 2 est le numéro du secteur ;
- le point situé sur le cercle a pour coordonnées `psChart2` ;
- le point extérieur est situé à une distance de $1{,}5\,r$ du centre et a pour coordonnées `psChartO2`, la lettre `O` signifiant `Out`.

Ces trois points sont tracés par l'instruction
`\psdots(psChartI2)(psChart2)(psChartO2)`.

Ce qui permet de réaliser un diagramme de ce type :




```
\psset{unit=1cm}
\begin{pspicture*}(-3,-3)(3,3)
\psset{chartNodeO=1.25}% 1.5 par défaut
\psChart[chartColor=color]{10,20,15,30,15,25}{}{2}
\multido{\i=1+1}{6}
        {\psline(psChartI\i)(psChartO\i)}
\uput[15](psChartO1){A}  \rput*(psChartI1){10}
\uput[60](psChartO2){B}  \rput*(psChartI2){20}
\uput[130](psChartO3){C} \rput*(psChartI3){15}
\uput[190](psChartO4){D} \rput*(psChartI4){30}
\uput[260](psChartO5){E} \rput*(psChartI5){15}
\uput[-45](psChartO6){F} \rput*(psChartI6){25}
\end{pspicture*}
```

Au passage, on peut voir comment rapprocher le point extérieur en modifiant la variable `chartNodeO` qui détermine la distance par rapport au centre ; par défaut cette distance est de 1,5 fois le rayon et on l'a définie ici à 1,25 fois le rayon. On pourrait de même modifier la distance entre le point intérieur et le centre en modifiant la variable `chartNodeI`.

C'est `\rput*` – employé à la place de `\rput` – qui permet d'écrire les nombres sur fond blanc.