TP2 : Analyse de données quantitatives avec le logiciel R

1 Données quantitatives discrètes

Dans le cas d'une variable quantitative discrète, le nombre de valeurs possibles (ou modalités) est fini et vous pouvez résumer ces données sous la forme d'un tableau de fréquences.

Exemple 1.1 Le nombre d'arbres plantés sur les parcelles d'un lotissement a été compté. Les données obtenues sont les suivantes :

1,2,4,1,6,3,2,1,2,0,1,2,2,1,3,0,3,2,1,2,2,3,2,3.

1. Rentrez ces données sous la forme d'un vecteur nommé `arbres` et affichez ce vecteur.
2. Triez les valeurs de ce vecteur par ordre croissant.
3. Donnez la taille de l'échantillon (c'est-à-dire le nombre de composantes de ce vecteur) en la notant `n` et affichez sa valeur.

1.1 Effectifs et effectifs cumulés

La fonction `unique()` permet d'afficher les modalités (ou valeurs possibles) de la valeur étudiée. Retournons à l'exemple 1.1 :

```
>effectif<-table(arbres)
>effectif
arbres
 0  1  2  3  4  6
 2  6  9  5  1  1
```

Vous obtenez la séquence des modalités et la séquence des effectifs correspondants : 2 parcelles n'ont aucun arbre, 6 parcelles ont un arbre... Si vous n'aviez pas tapé la deuxième ligne de commande `effectif`, que se serait-il passé ?

Le tableau des fréquences (en pourcentage) s'obtient en divisant les effectifs par la taille n de l'échantillon :

```
>frequence<-effectif*100/n
>frequence
```

Retour à l'exemple 1.1 :

```
>frequence<-effectif*100/24
>frequence
arbres
        0         1         2         3         4         6
 8.333333 25.000000 37.500000 20.833333  4.166667  4.166667
```

Vous pouvez obtenir ce tableau des effectifs par les effectifs cumulés que vous pouvez obtenir avec la fonction `cumsum()`.

Retour à l'exemple 1.1 :

```
>effcum<-cumsum(effectif)
>effcum
 0  1   2   3   4   6
 2  8  17  22  23  24
```

Vous obtenez la séquence des effectifs cumulés de chaque modalité : 8 parcelles ont un arbre ou moins, 17 parcelles ont au maximum 2 arbres,...

De la même façon que pour les fréquences, vous pouvez obtenir les fréquences cumulées (en %) :

```
>effcum*100/n
```

Retour à l'exemple 1.1 :

```
>effcum*100/24
        0         1         2         3         4         6
 8.333333  33.333333  70.833333  91.666667  95.833333 100.000000
```

La fonction `sum()` calcule la somme des valeurs.
Retour à l'exemple 1.1 :

```
>sum(arbres)
[1] 49
```

Dans l'exemple, cette fonction calcule le nombre total d'arbres plantés sur les 24 parcelles.

1.2 Indicateurs de tendance centrale

Vous pouvez obtenir quelques indicateurs de tendance tels que la moyenne, le maximum, le minimum ou le range (minimum, maximum), la médiane par les fonction `mean(), max(), min(), range(), median()` que vous avez déjà rencontrés.

Retour à l'exemple 1.1 :

```
> mean(arbres) # nombre moyen d'arbres par parcelle
[1] 2.041667
> max(arbres) # nombre maximum d'arbres sur une parcelle
[1] 6
> min(arbres) # nombre minimum d'arbres sur une parcelle
[1] 0
> range(arbres) # intervalle des valeurs possibles
[1] 0 6
> median(arbres) # nombre médian d'arbres par parcelle
[1] 2
```

Conclusion sommaire sur les résultats produits : vous observez donc près de 2 arbres en moyenne par parcelle, pouvant aller de 0 à 6 arbres sur une parcelle. Le nombre médian d'arbres est égal à 2, c'est-à-dire que la moitié des parcelles ont 2 arbres ou plus, et l'autre moitié des parcelles ont 2 arbres ou moins.
La fonction `summary()` permet d'obtenir un tableau récapitulatif des indicateurs avec en complément les premier et troisième quartiles.

Retour à l'exemple 1.1 :

```
> summary(arbres)
  Min.   1st Qu.  Median   Mean   3rd Qu.   Max.
 0.000    1.000    2.000   2.042   3.000    6.000
```

1.3 Indicateurs de dispersion

Vous pouvez calculer la variance et l'écart-type avec les fonctions `var()` et `sd()`.
Retour à l'exemple 1.1 :

```
> var(arbres) # variance
[1] 1.693841
```

Calculez la variance non-corrigée vous-même :

```
> sum((arbres-mean(arbres))^2)/length(arbres)
[1] 1.623264
```

La variance obtenue est différente de la précédente, pourquoi ?

```
>?var
```

Vous constatez que le logiciel R utilise $n-1$ pour le dénominateur dans la définition de la variance, c'est-à dire $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2$ (d'écart-type noté σ_{n-1} sur les calculettes. Cette quantité est souvent préférée dans les applications numériques pour des questions d'estimation). Vérifiez-le :

```
> (n-1)*var(arbres)/n
[1] 1.623264
```

Calculez maintenant l'écart-type et vérifiez que l'écart-type est la racine carrée de la variance :

```
> sd(arbres)
[1] 1.301476
> (sd(arbres))^2
[1] 1.693841
```

Vous retrouvez la variance donnée par la fonction `var()`. Une alternative robuste pour l'estimation de la dispersion est la fonction `mad()` :

```
> mad(arbres)
[1] 1.4826
```

1.4 Représentations graphiques

Pour toutes les fonctions graphiques que vous allez voir, vous pouvez donner des titres au graphique, à l'axe des abscisses ou l'axe des ordonnées en utilisant les options `main="...", xlab="..." ou ylab="..."`.
Quand vous exécutez une fonction graphique avec le logiciel R, une nouvelle fenêtre graphique s'ouvre (que vous pouvez réduire ou agrandir) dans laquelle les figures sont affichées. Plusieurs représentations graphiques sont possibles sur ce type de données :

- un nuage de points ou
- un diagramme en bâtons.

La fontion `plot()` affiche un nuage de points avec en abscisse le numéro de l'observation (ici de 1 à 24) et en ordonnée le nombre d'arbres :

```
> plot(arbres)
```

Vous pouvez aussi demander la courbe des effectifs cumulés, avec en abscisse le nombre d'arbres par parcelles et en ordonnée les effectifs cumulés :

```
> plot(effcum)
```

Vous pouvez également tracer un diagramme en bâtons par la fonction `barplot` à partir du tableau des effectifs ou des fréquences :

```
> barplot(effectif, xlab="nombre d'arbres", ylab="effectif")
> barplot(frequence, xlab="nombre d'arbres", ylab="frequence")
```

À noter que l'allure du diagramme n'est pas modifiée, seul change l'axe des ordonnées.

2 Données issues d'un caractère quantitatif continu

Une des principales caractéristiques des données continues réside dans le fait qu'elles sont pratiquement toutes différentes (les égalités sont dues à la nécessité d'arrondir et/ou au fait que les instruments de mesure sont gradués) ; les effectifs des modalités sont alors pratiquement tous égaux à 1. Pour tracer un histogramme de ces données, nous procédons à un regroupement de ces données en classes. Les raisons du choix du nombre de classes, de leurs amplitudes ou de leurs effectifs ne seront pas abordées ici.

2.1 Un exemple

Nous avons relevé les poids (en grammes) de souris soumises à une expérience de supplémentation en vitamines :

74, 85, 95, 84, 68, 93, 84, 87, 78, 72, 81, 91, 80, 65, 76, 81, 97, 69, 70, 98.

1. Créez la séquence souris et l'afficher.
2. Vérifiez que les effectifs des modalités sont pratiquement tous égaux à 1 en affichant le tableau des effectifs.
3. Combien de souris ont subi l'expérience ?
4. Donnez les indicateurs de tendance centrale de ce jeu de données.

2.2 Représentations graphiques

Sur des données quantitatives, il est conseillé de représenter la « boîte à moustaches » que vous obtenez avec la fonction `boxplot` :

```
> boxplot(souris)
```

La boîte à moustaches permet de représenter la distribution d'une variable avec les éléments suivants (de bas en haut) : le minimum, le 1er quartile, la médiane, le 3ème quartile et le maximum. Plus la boîte est étirée en hauteur, plus les valeurs de la variable sont dispersées. Vous pouvez également représenter les données continues en traçant un histogramme. Le problème de l'histogramme est la définition des classes. Voyons ce que le logiciel `R` nous trace si vous ne spécifiez aucune option :

```
> hist(souris)
```

Il y a ici plusieurs remarques à faire. D'abord, il faut noter que le logiciel `R` a choisi, par défaut, de regrouper les données par classes d'amplitude 5, soit 7 classes : la première étant $[65, 70]$ et ensuite du type $]a, b]$. Si on souhaite obtenir des classes ouvertes à droite, il suffit de le préciser :

```
> hist(souris,right=FALSE)
```

Une autre remarque : `R` a marqué en ordonnée « Frequency » mais en fait, ce qu'il représente correspond plutôt aux effectifs. Ceci vient d'une légère différence de vocabulaire entre pays. Ce que nous appelons nous « effectif » s'appelle en fait « frequency » en anglais et nos « fréquences »françaises s'appellent « relative frequencies ». De plus, puisque les classes sont de même amplitude, les densités d'effectif sont proportionnelles aux effectifs, et `R` donne alors directement les effectifs en ordonnée. Vous pouvez modifier le nombre de classes par l'option `nclass(...)` de la façon suivante (en 4 classes par exemple) :

```
> hist(souris, nclass=4)
```

Les classes restent d'amplitude égale. Si on veut maintenant définir d'autres classes, il suffit de spécifier dans la fonction `hist`, les bornes des classes par l'option `br=c(...)` sous la forme :

```
> hist(souris, br=c(65,70,90,100))
```

Dans le cas où les classes ont des amplitudes différentes, il faut noter que le logiciel `R` choisit par défaut de représenter les fréquences (et non les effectifs). Notez qu'ici, comme les classes n'ont pas la même amplitude, `R` a bien représenté en ordonnée les densités de fréquence.

Pour résumer :
- lorsque les classes ont même amplitude, `R` choisit de représenter les effectifs.
- Lorsque les classes ont des amplitudes différentes, `R` choisit de représenter les fréquences (à l'aide des densités de fréquence).

Vous pourrez, si vous le souhaitez, consulter la notice de la commande `hist` en tapant :

```
>?hist
```

2.3 Décomposition d'un vecteur selon des groupes

L'expérience à laquelle on s'intéresse porte sur des souris soumises à une supplémentation en vitamines. On sait finalement que les 10 premières données concernent des souris effectivement soumises à cette supplémentation et

les 10 dernières données correspondent aux souris non supplémentées en vitamines. Il peut donc être intéressant de décomposer les données en deux groupes selon la présence ou l'absence de cette supplémentation.
Dans un premier temps, on crée un vecteur vitamine de 20 composantes : les 10 premières prenant la valeur s (comme supplémentée) et les 10 suivantes, la valeur ns (comme non supplémentée).
Pour pouvoir comparer les résultats de l'expérience sur les souris supplémentées et sur les souris non supplémentées, on décompose le vecteur souris en deux listes par la fonction `split` :

```
> split(souris,vitamine)
$ns
[1] 81 91 80 65 76 81 97 69 70 98
$s
[1] 74 85 95 84 68 93 84 87 78 72
```

Pour analyser les données en fonction de la supplémentation, vous conservez les résultats de la fonction `split` sous le nom `souris.sup` :

```
> souris.sup <- split(souris,vitamine)
```

`R` crée alors automatiquement deux nouveaux vecteurs :
- l'un pour les souris supplémentées, `souris.sup$s`,
- l'autre pour les souris non supplémentées, `souris.sup$ns`.

Vous pouvez étudier ces deux vecteurs :

```
> summary(souris.sup$s)
 Min.   1st Qu.   Median   Mean   3rd Qu.   Max.
 68.0    75.0     84.0    82.0    86.5    95.0
> summary(souris.sup$ns)
 Min.   1st Qu.   Median   Mean   3rd Qu.   Max.
 65.0    71.5     80.5    80.8    88.5    98.0
```

Vous observez qu'en moyenne et en médiane, le poids des souris supplémentées en vitamines est légèrement plus élevé que le poids des souris non supplémentées.
Vous pouvez aussi appliquer toutes les autres fonctions vues précédemment. Si vous demandez seulement une analyse sur `souris.sup`, `R` vous affiche des informations la composition de cet objet.

Représenter l'histogramme du poids des souris supplémentées et non supplémentées :

```
> hist(souris.sup$s, main="Histogramme du poids des souris supplementees en vitamine",
+ xlab="poids", br=c(60,70,80,90,100))
> hist(souris.sup$ns, main="Histogramme du poids des souris non supplementees",
+ xlab="poids", br=c(60,70,80,90,100))
```

À noter qu'il est préférable de définir des classes pour la construction des 2 histogrammes pour être sûr que les histogrammes soient comparables. En effet, si on ne spécifie pas de classes pour cet exemple, l'échelle de l'axe des abscisses n'est pas la même sur les deux histogrammes.

2.4 Épilogue

Vous pouvez également utiliser la commande `split` pour fragmenter une table.
Chargez la table CO2 avec la commande `data(CO2)` puis copiez la dans un objet appelé `tab`.

```
> data(CO2)
> tab<-(CO2)
```

Affichez le contenu de `tab` avec `summary`.

```
>summary(tab)
```

Identifiez les vecteurs qui contiennent des données qualitatives, par exemple `Type, Treatment`.
Fragmentez le vecteur `uptake`selon ces deux vecteurs.
ATTENTION ! uptake tout seul n'existe pas.

```
> uptake
Error : Object "uptake" not found
```

Il faut le voir comme élément de notre table `tab`.
Exemple :

```
> split(tab$uptake,tab$Type)
> split(tab$uptake,tab$Treatment)
```

Commentez les intructions suivantes :

```
> (fragments <- lapply(split(tab,tab$Type),split,tab$Treatment))
> str(fragments)
```

3 Exercice 1 : le fichier Forbes2000

Le fichier de données que nous allons utiliser dans ce TD est constitué d'un ensemble de 2000 lignes qui représentent les 2000 premières entreprises au monde, suivant les critères du classement « Forbes 2000 » de l'année 2004.
Téléchargez ce fichier :

```
>data("Forbes2000", package="HSAUR")
```

1. Imprimez-le à l'écran.
2. Quelle est la structure de `Forbes2000` ?
3. Quelle est la classe ou le type de `Forbes2000` ?
4. Combien de lignes comporte ce fichier ?
5. Combien de colonnes comporte ce fichier ?
6. Quels sont les noms des colonnes et les classes des objets qui les composent ?
7. Quelle est la longueur d'une colonne ?
8. Quelle est le nom de la première entreprise du jeu de données ?
9. Combien il y a-t-il de catégories d'entreprises différentes ?
10. Quels sont les noms des différentes catégories d'entreprises ?
11. Donnez le tableau de contingence de ces catégories.
12. Quelle est la classe de la colonne sales ?
13. Donnez la médiane, la moyenne, l'étendue puis un ensemble de statistiques descriptives de la variable « Sales ».
14. Donnez un ensemble de statistiques descriptives pour l'ensemble du jeu de données.
    Nous allons maintenant passer à la représentation graphique du jeu de données.
15. Que font les lignes suivantes ?

    ```
    > layout(matrix(1 :2,nrow=2))
    > hist(Forbes2000$marketvalue)
    > hist(log(Forbes2000$marketvalue))
    ```
16. Avez-vous compris la commande `layout` ? Si oui, que se passera-t-il si vous la supprimez des lignes de commande ?
17. Tracer les log(marketvalue) en fonction des log(sales).
18. Tracer les boxplot de la variable `log(marketvalue)` des quatre pays suivants :
    - Germany,
    - India,
    - Turkey,
    - United Kingdom.