




Lab 2
RStudio et langage R



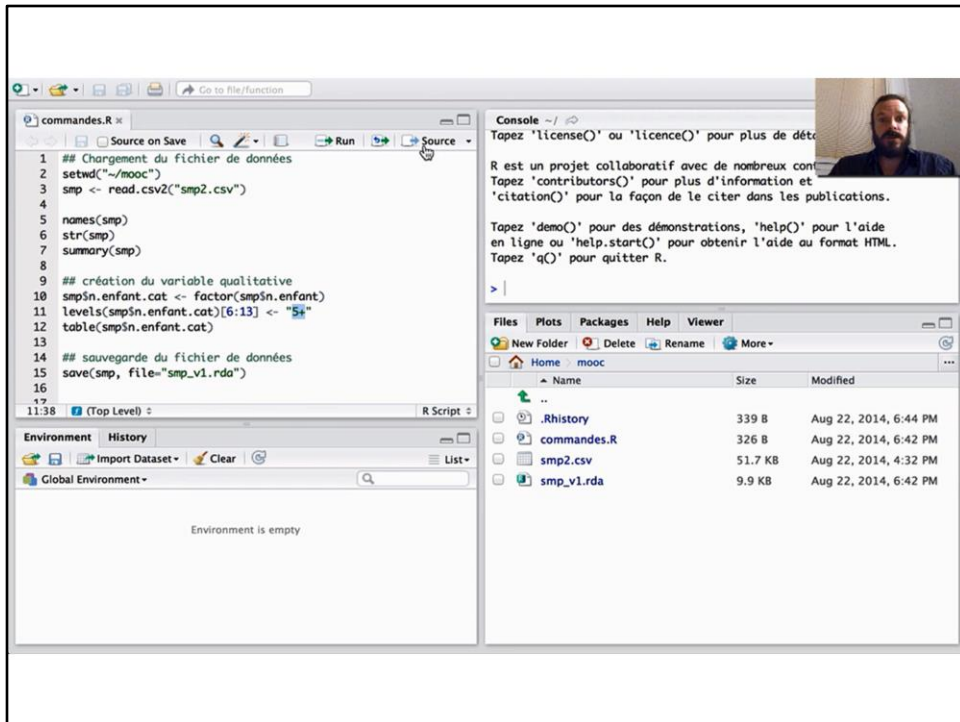
Indexation critériée d'observations, sélection de variables • graphiques univariés



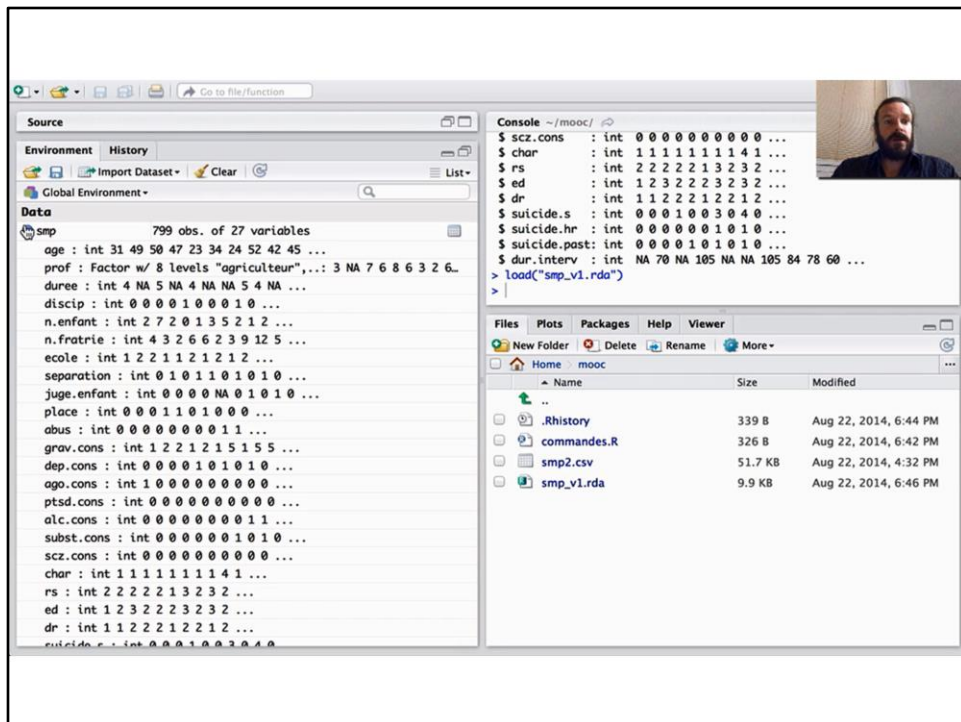
Pr. Bruno Falissard



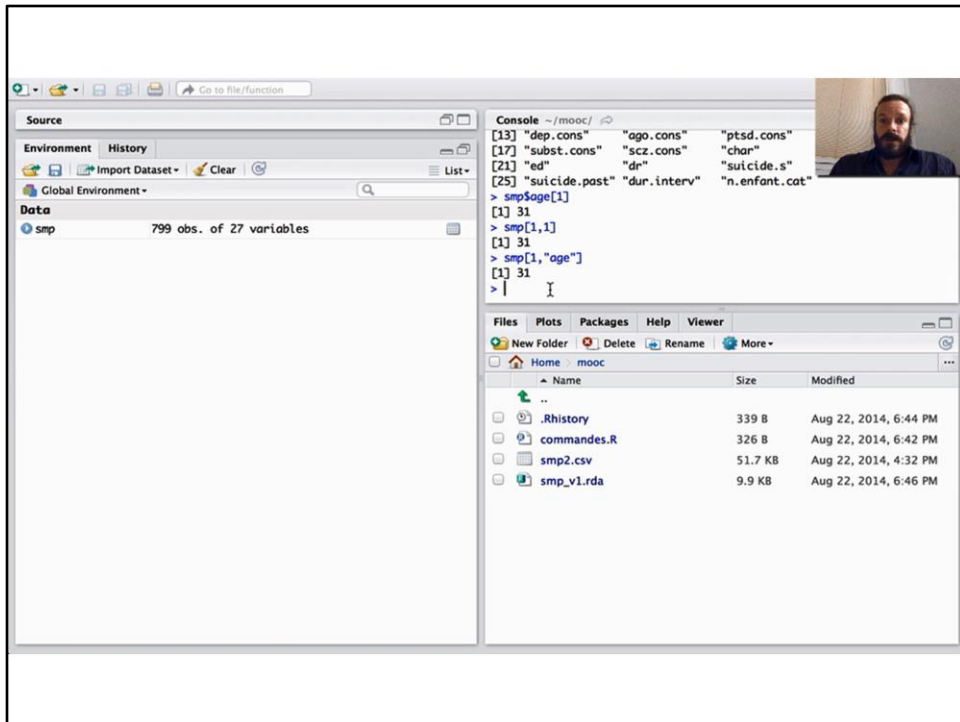
[00:01] Dans cette deuxième session, on va s'intéresser à la sélection indexée d'observations ou à la restriction d'un tableau de données à un certain nombre de variables, ce qui est souvent plus pratique, soit pour faire des analyses statistiques, soit pour faire des représentations graphiques.



[00:14] Dans la session précédente on avait sauvegardé toutes nos commandes dans un fichier historique qu'on avait appelé commandes.R. Donc voilà le même fichier qui a été légèrement nettoyé, dans lequel on n'a gardé que les informations essentielles, les commandes essentielles. On remarquera que pour afficher les commentaires on utilise le signe dièse (#) (1 ou plusieurs signes) et ensuite on a la séquence d'instructions qui nous ont permis de générer par exemple notre fichier final smp_v1 qui inclut une variable catégorielle supplémentaire n.enfant.cat qui est une variable nombre d'enfants pour laquelle on a recodé ou agrégé les dernières modalités et en désignant par « 5+ » pour 5 enfants ou plus.



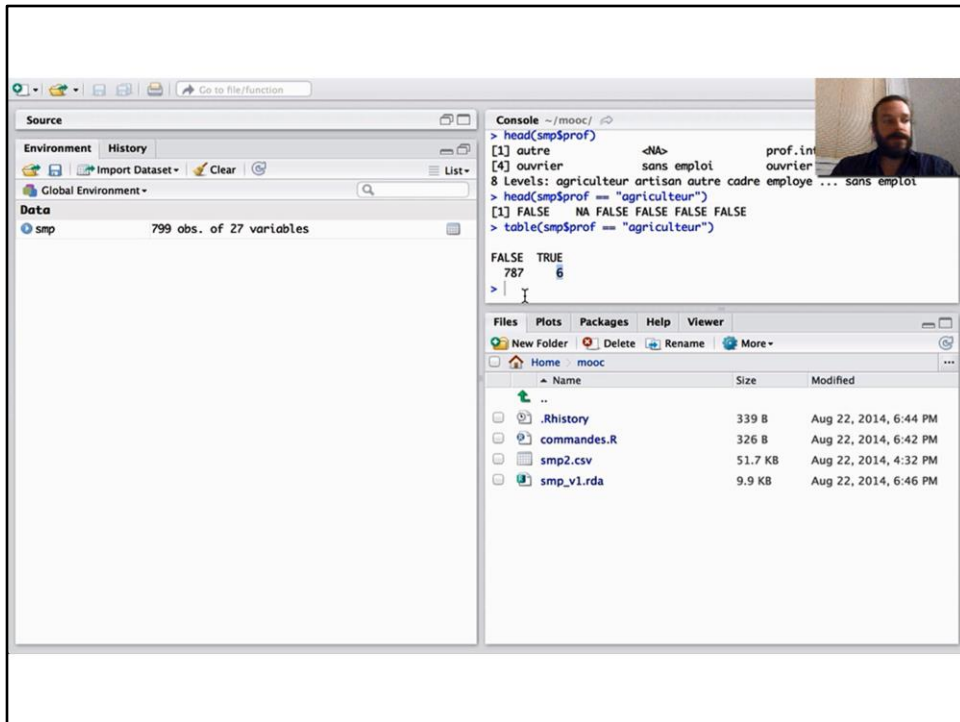
[01:00] Si on tape sur le bouton source, en fait, tout le script R est joué directement dans la console et on se retrouve ici avec notre dataframe smp qui apparaît dans le workspace et qui inclut la variable n.enfant.cat. Evidemment, on aurait très bien pu tout supprimer et simplement charger le fichier smp_v1.rda (`load(smp_v1.rda)`) pour obtenir le même résultat. L'avantage c'est que ça permet d'archiver les étapes de l'analyse en particulier de la préparation de données et de pouvoir rejouer à n'importe quel moment les scripts R ou les commandes R qu'on a utilisés.



[01:43] On a vu dans la session précédente qu'on pouvait adresser directement les observations dans une variable en notant le nom de la variable et entre crochets le numéro d'observation qui nous intéresse (`smp$age[1]`).

On peut faire la même chose à partir du dataframe lui-même en indexant le dataframe lui-même donc la première observation de la première variable (`smp[1,1]`) puisque l'âge est la première variable du dataframe `smp`.

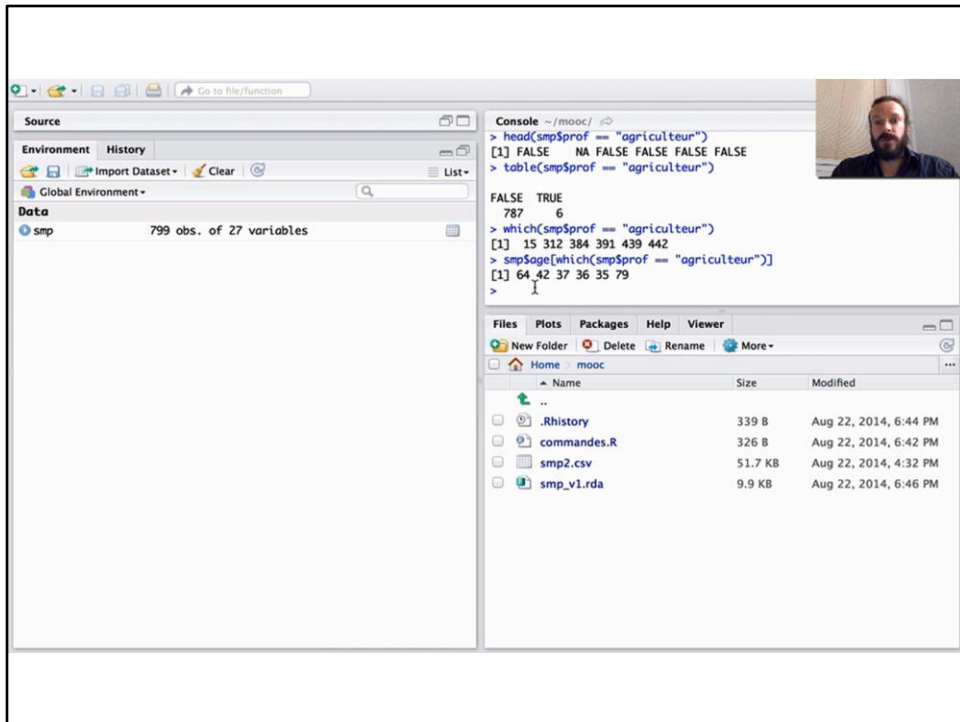
Souvent il est plus commode de nommer directement le nom de la variable donc ici `smp` entre crochets la 1^{re} observation pour la variable `age` (`smp[1, "age"]`).



[02:20] Ça c'est pour les variables numériques mais on peut aussi regarder par exemple les variables catégorielles. En particulier on peut s'intéresser pour la variable profession dont on affiche les 6 premières valeurs ici (`head(smp$prof)`) aux seules valeurs pour lesquelles la profession vaut agriculteur par exemple (`head(smp$prof == "agriculteur")`). Donc on notera que pour réaliser une restriction sur une variable à une seule modalité, on utilise le double signe égal (`(head(smp$prof == "agriculteur"))`), donc les valeurs de `smp` pour lesquelles la variable `prof` vaut agriculteur.

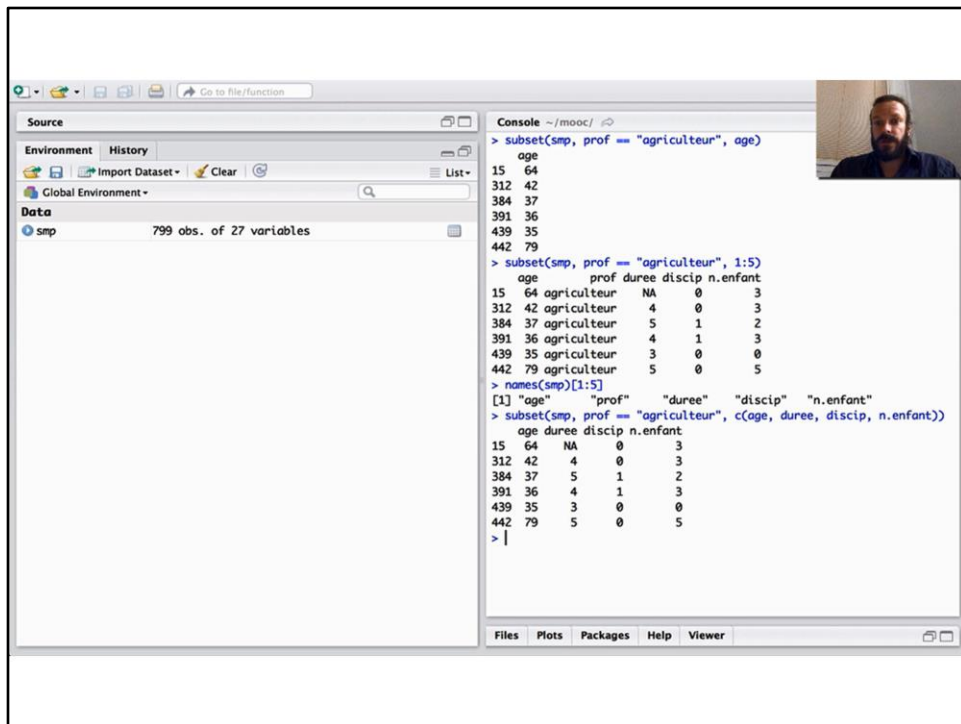
Donc, qu'est-ce qui se passe ici ? En fait R est en train de tester chacune des modalités de la variable à la valeur agriculteur. Donc la 1^{re} ne réalise pas cette condition, il nous renvoie FALSE, la 2^e est une valeur manquante, la 3^e ne remplit pas non plus la condition.

En fait ça ne nous apporte pas beaucoup d'information, sauf que on peut très simplement construire un tableau des valeurs de la variable `prof` qui remplissent cette condition et ici on s'aperçoit que finalement on a 6 valeurs qui remplissent la condition « profession est égale à agriculteur ».



[03:31] On peut même vérifier quelles sont les valeurs qui remplissent la condition en utilisant la commande `which()` (`which(smp$prof == "agriculteur")`). Et ici, R va nous renvoyer les numéros d'observation pour lesquelles la valeur de `prof` vaut bien `agriculteur`.

Ce système d'indexation qui est en gros le principe d'un dictionnaire où on associe à une position une valeur particulière nous permet par exemple d'indexer directement les valeurs de la variable `age` pour lesquelles la profession vaut `agriculteur`. Donc ici il suffit simplement de copier cette expression, de la coller ici, donc en fait ce qu'on fait ici c'est qu'on prend la variable `age` et entre crochets on indique la liste des numéros d'observation qui nous intéressent et cette liste d'observations est obtenue directement à partir d'un test d'égalité logique (`smp$age[which(smp$prof == "agriculteur")]`). Et ici on a donc l'âge des individus pour lesquels la profession est `agriculteur`.



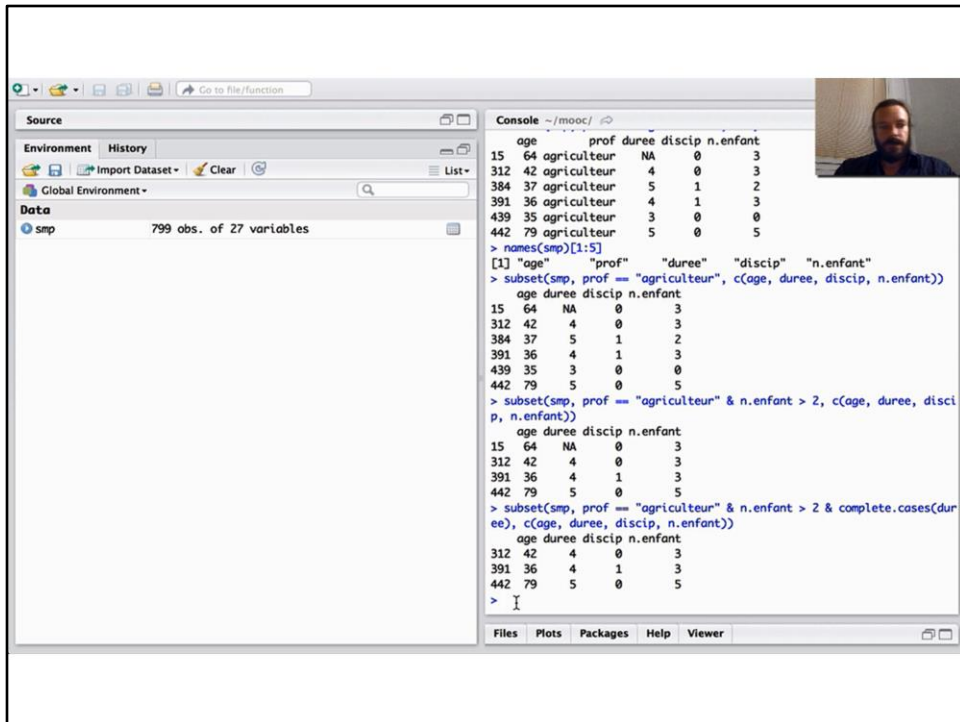
[04:30] En fait on peut faire exactement la même chose sans utiliser le mécanisme d'indexation entre crochets à l'aide de la commande `subset()`.

La commande `subset()` prend en 1^{er} argument le nom du dataframe, en 2^e argument le filtre qu'on applique sur les lignes – ici ça va être `prof == "agriculteur"` – et en 3^e argument la variable qui nous intéresse, ici c'est l'âge (`subset(smp, smp$prof == "agriculteur", age)`).

L'avantage c'est qu'on n'est plus obligé de préfixer le nom des variables par le nom du dataframe et ici on voit qu'on retrouve bien nos 6 valeurs d'âge qui remplissent la condition « la profession de l'individu est de type agriculteur ».

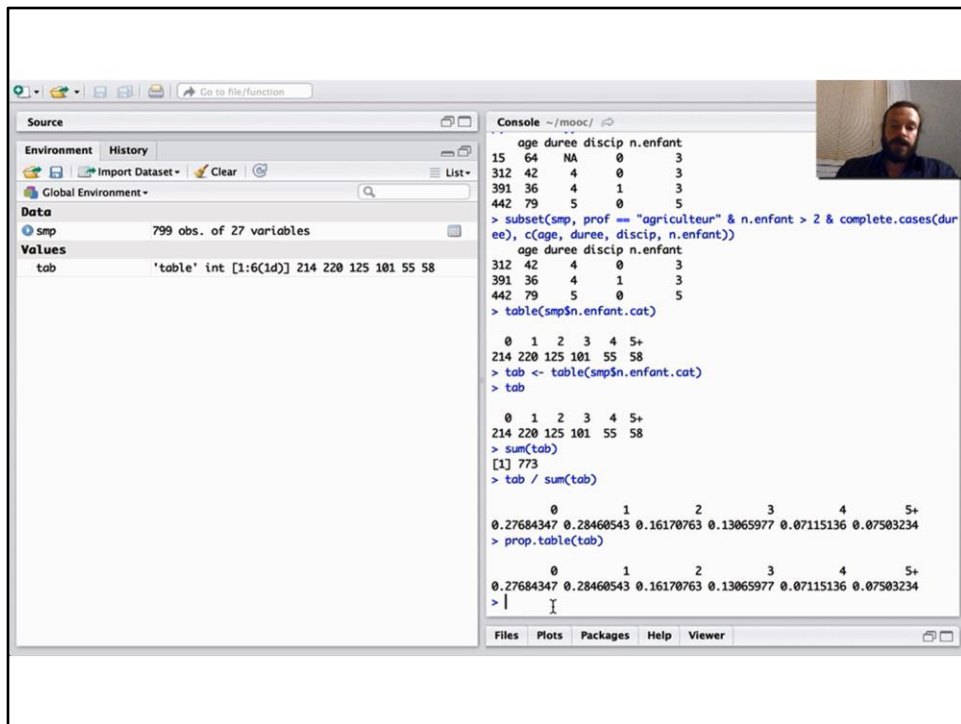
Si on veut étendre la sélection à plus d'une variable, on peut par exemple sélectionner les variables 1 à 5 (`subset(smp, smp$prof == "agriculteur", 1:5)`) et cette fois-ci on aura une partie de notre dataframe qui vérifie bien la condition « la profession vaut agriculteur ». Et on a les variable `duree`, `discip` et `n.enfant` qui sont associées à l'âge.

En fait, ces variables-là, on pourrait directement les utiliser dans la commande `subset()` au lieu des données 1 à 5, en particulier lorsque les variables ne sont pas contiguës. Donc, on va réécrire exactement le même filtre logique mais ce qu'on va faire, c'est sélectionner le nom des variables qui nous intéressent, donc par exemple l'âge, la profession à la rigueur on n'en a pas besoin, la durée, `discip` et le nombre d'enfants (`subset(smp, smp$prof == "agriculteur", c(age, duree, discip, n.enfant))`). On notera qu'ici on ne met pas de quote ou d'apostrophe anglo-saxonne pour encadrer les variables. Alors ici on a exactement le même résultat sans la colonne `prof`.



[06:12] On peut rajouter des filtre sur les lignes, par exemple on peut s'intéresser aux individus dont la profession est agriculteur ET – c'est le ET logique – dont le nombre d'enfants est supérieur à 2 (`subset(smp, smp$prof == "agriculteur" & n.enfant > 2, c(age, duree, discip, n.enfant))`). Donc ici on a combiné 2 filtres, ce sont toujours des filtres sur les lignes, le 1^{er} porte sur la profession, le 2^e porte sur le nombre d'enfants et l'opérateur logique pour la conjonction est le ET esperluette (&).

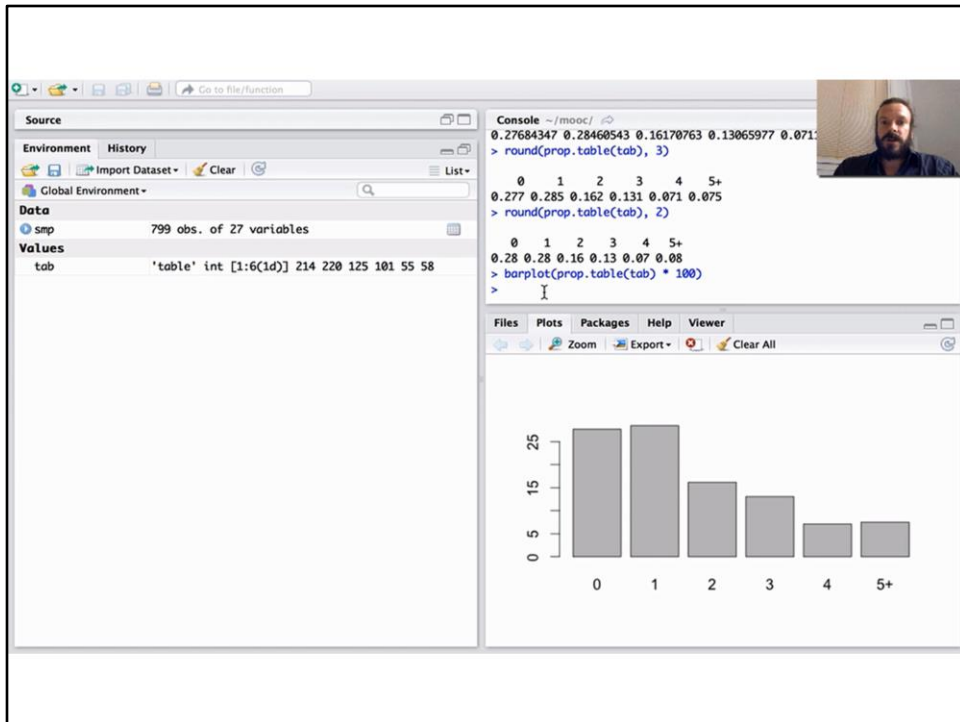
Alors évidemment on peut faire des filtres beaucoup plus complexes. On pourrait par exemple demander, puisqu'on voit que par exemple ici pour la durée on a une valeur manquante, que les résultats ne comportent que les cas complets c'est-à-dire les individus sans valeur manquante sur la variable `duree` (`subset(smp, smp$prof == "agriculteur" & n.enfant > 2 & complete.cases(duree), c(age, duree, discip, n.enfant))`). Et on voit qu'ici on a réduit notre sélection à un ensemble donc de 4 variables et de 3 individus.



[07:14] Si on se rappelle la variable `smp$n.enfant.cat` qu'on avait créée, donc qui était dérivée de la variable `n.enfant` d'origine ici et pour laquelle on avait agrégé les dernières modalités, on peut très bien stocker le résultat de cette instruction `table()` qui fournit un tableau d'effectif, dans une variable qu'on va appeler `tab` (`tab <- table(smp$n.enfant.cat)`). Donc la variable `tab` est créée directement dans le workspace et on voit que c'est un objet de type `table`. Donc on a simplement un tableau d'effectif.

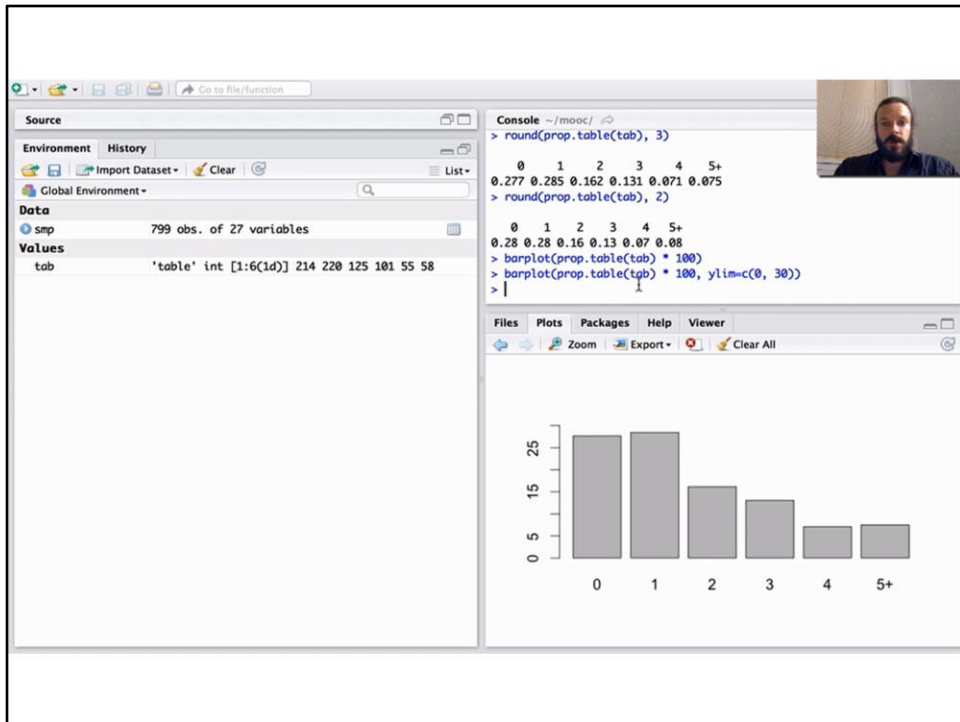
L'intérêt, c'est qu'on peut effectuer toutes sortes d'opérations dessus, par exemple calculer l'effectif total à l'aide de la commande `sum()` (`sum(tab)`) ou alors diviser ce tableau-là par l'effectif total (`tab / sum(tab)`) donc on verra ici qu'R procède élément par élément donc il va prendre chaque élément et diviser par le total. Donc 0.277 correspond à 214 divisé par 773.

En fait il existe une commande qui s'appelle `prop.table()` et qui fait exactement la même chose (`prop.table(tab)`). Il suffit de lui donner un tableau, ici c'est un tableau qu'on a stocké dans la variable `tab` mais on pourrait très bien utiliser directement `table(smp$n.enfant.cat)` et qui nous fournit en gros un résumé de fréquences relatives donc les effectifs rapportés à l'effectif total.

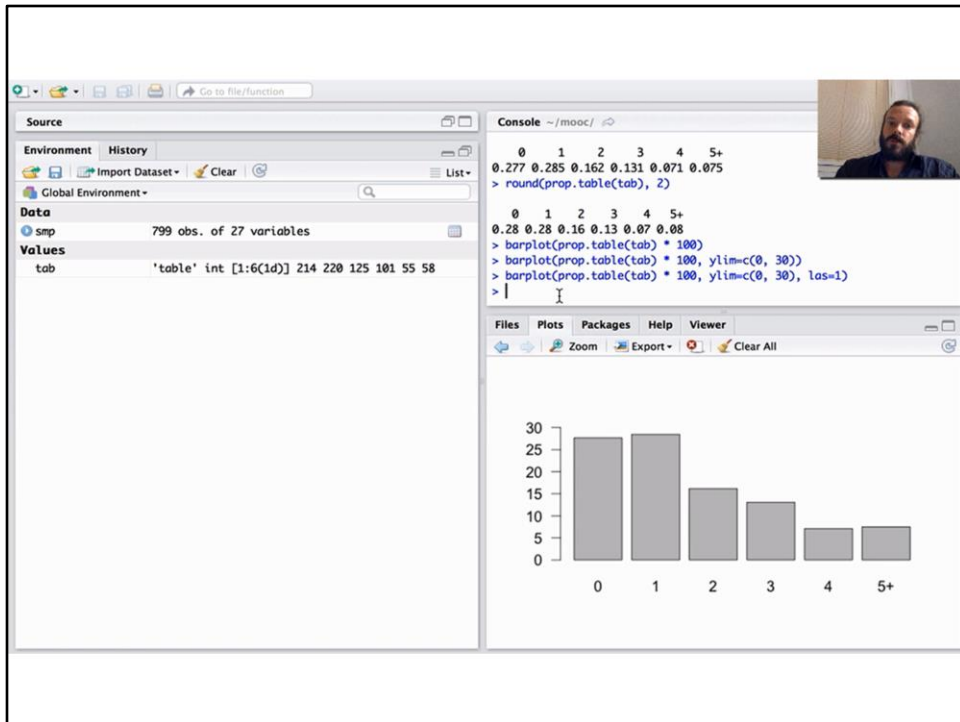


[08:35] Souvent, on peut vouloir présenter les résultats avec 2 ou 3 chiffres, ici on utilisera la commande `round()` qui permet d'arrondir le résultat, ici à 3 décimales (`prop.table(tab), 3`). On peut changer et mettre 2 décimales si on veut (`prop.table(tab), 2`).

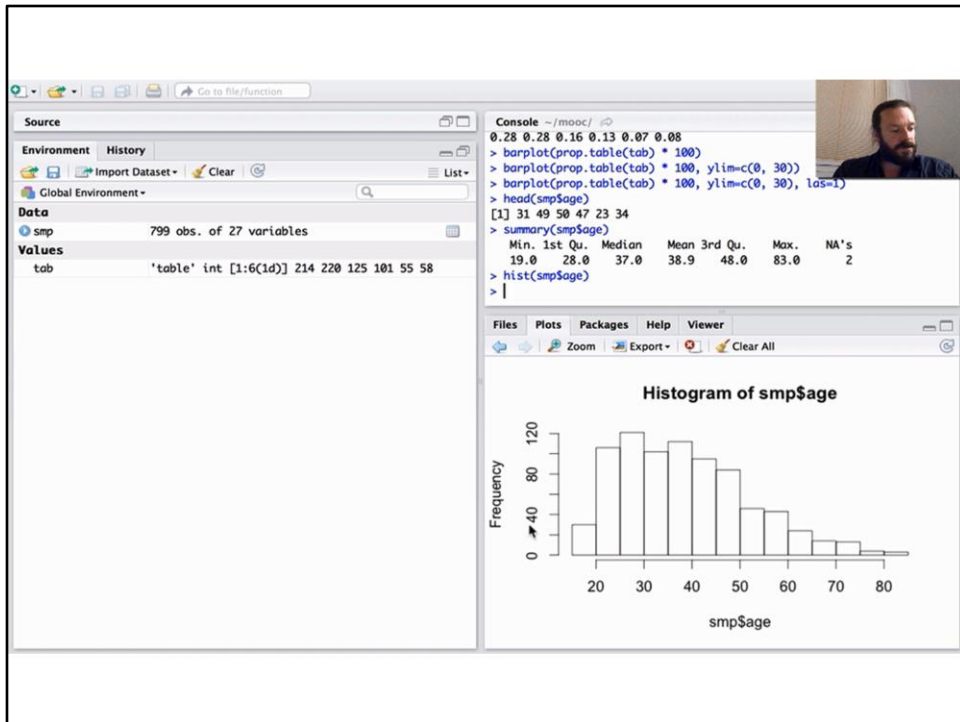
[08:50] On est toujours en train de travailler avec un tableau. Finalement on peut faire une représentation graphique relativement simple qui consiste en un diagramme en barres et pour cela on utilise la commande `barplot()` et on va lui donner en fait le résultat de notre tableau qu'on va exprimer sous forme de pourcentage, donc ce sont les fréquences relatives obtenues avec `prop.table()` que l'on va simplement multiplier par 100 pour avoir un pourcentage (`barplot(prop.table(tab) * 100)`). Et R va afficher automatiquement dans le panneau des graphiques le résultat de la commande graphique.



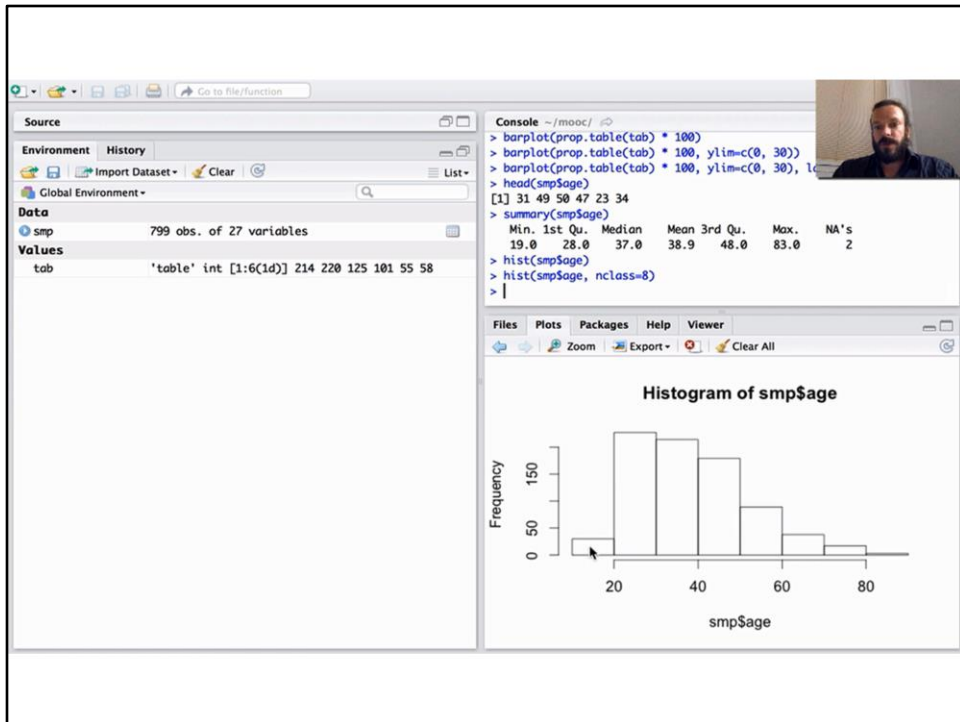
[09:22] Ici on voit que l'axe n'est pas tout à fait correct et on pourrait vouloir augmenter par exemple les coordonnées sur l'axe vertical (`barplot(prop.table(tab) * 100, ylim=c(0, 30))`).



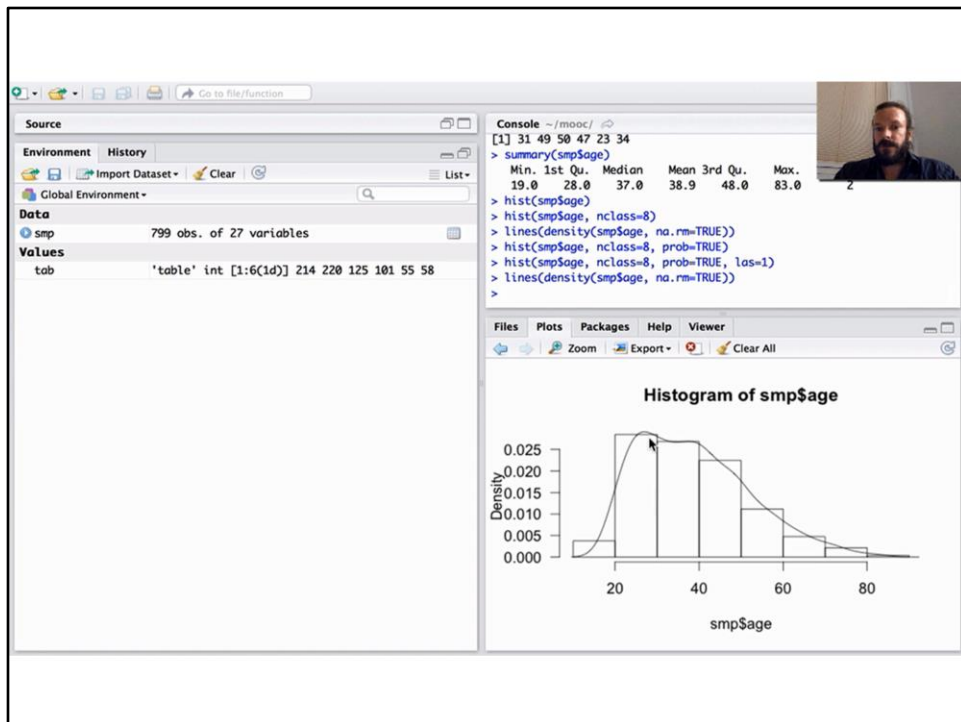
[09:40] On pourrait également simplement vouloir aligner différemment les labels donc les orienter horizontalement (`barplot(prop.table(tab) * 100, ylim=c(0, 30), las=1)`) et on voit qu'ici, on a bien des proportions de 0 à 30% avec la répartition de notre échantillon selon les modalités de la variable nombre d'enfants.



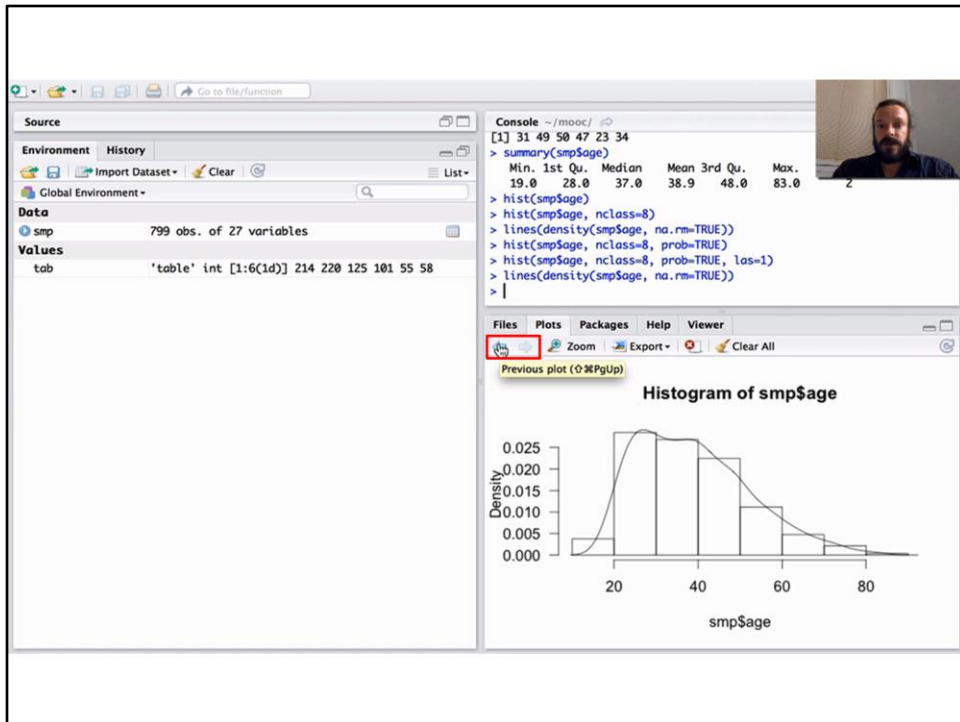
[10:00] Pour les variables numériques, on pourra souvent utiliser des histogrammes, donc si on reprend la variable `smp$age` (`head(smp$age)`), pour laquelle on avait fourni un résumé numérique avec la commande `summary()` (`summary(smp$age)`), on utilisera cette fois la commande `hist()` pour histogramme, suivie du nom de la variable (`hist(smp$age)`). Cette fois-ci, on a un histogramme, les effectifs, que R appelle des fréquences,



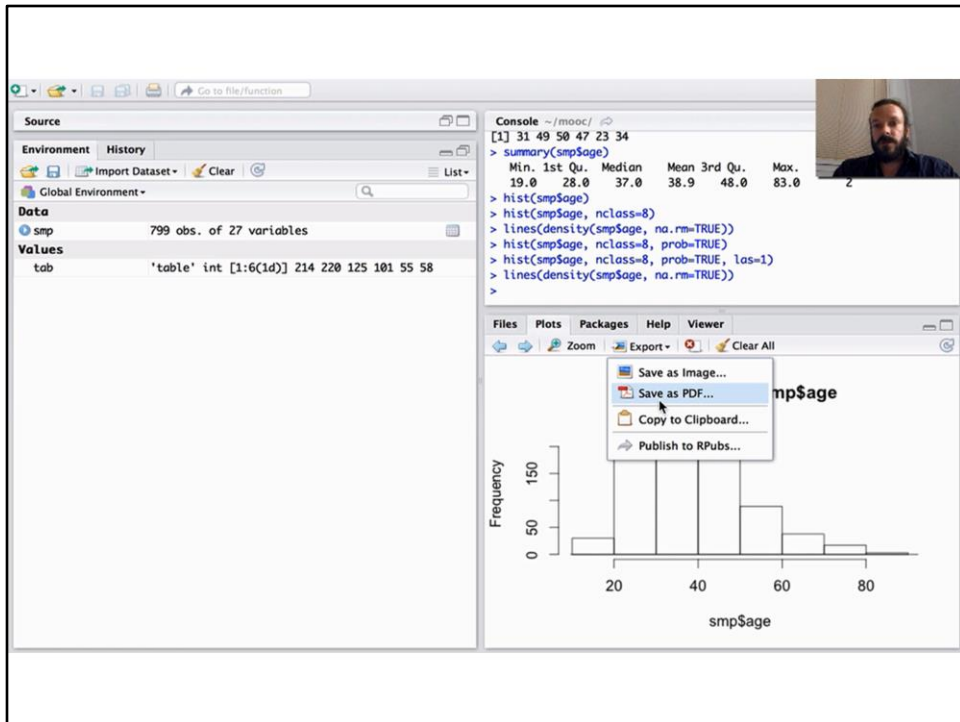
[10:25] et qu'on peut aussi personnaliser légèrement en fournissant approximativement 8 classes (`hist(smp$age, nclass=8)`), ce qui nous permet de réduire le nombre d'intervalles de classes et on voit que les âges se concentrent majoritairement autour de 20 à 45 ans ;



[10:42] et on peut également utiliser une densité non paramétrique (`lines(density(smp$age, na.rm=TRUE))`) mais pour cela, il faudra en fait convertir l'histogramme en histogramme de densité (`hist(smp$age, nclass=8, prob=TRUE)`), selon le même principe on peut réorienter les labels (`hist(smp$age, nclass=8, prob=TRUE, las=1)`) et ensuite afficher notre fonction de densité non paramétrique (`lines(density(smp$age, na.rm=TRUE))`) qui souvent évite d'avoir à se poser des questions sur le nombre d'intervalles de classes ou l'amplitude des intervalles dans le cas des histogrammes.



[11:19] Sous R en fait on peut naviguer entre les graphiques à l'aide directement des flèches droite et gauche. Ça nous permet de voir tous les graphiques qu'on a générés.



[11:28] On peut également les exporter au format PDF ou au format PNG et on peut également zoomer sur les graphiques.