CSV to SAV Instructions

Dina Sinclair

January 16, 2018

Si vous voulez convertir un fichier de CSV (un format que vous pouvez obtenir du serveur) en SAV (un format que vous pouvez charger dans SPSS), ce document vous aidera à démarrer.

Format de code général

Lorsque vous modifiez un format de fichier de CSV à sav dans R, votre code suivra les étapes de base suivantes:

1. Importez les données dans R à partir de votre fichier csv
2. Nettoyez les données. Cela peut signifier la fixation des noms de variables ou la modification du type (numérique, chaîne, facteur, etc.) des colonnes souhaitées.
3. Exportez les données dans un fichier sav. Ce qui suit est un exemple de code. Les lignes commençant par '#' sont des commentaires, ce qui signifie que R les ignore.

```
# STEP ONE: read (import) the data into R. Here we use the function read.csv, and the
# first entry '18 01 08 donnee.csv' is the name of the csv file we want to use.
# More on the import step in a later section.
d <- read.csv("18 01 08 donnee.csv", na.strings = c("---",""), check.names=FALSE)

# STEP TWO: clean the data before saving it as an sav file. Here that means shortening
# the variable names and making sure the number column of the data is saved as an integer.
# More on the cleaning step in a later section.
names(d) <- gsub(".*\\.", "",names(d))
names(d) <- make.names(names(d))
names(d) <- gsub("\\.\\.\\.",'_',names(d))
d$Number <- as.integer(d$Number)

# STEP THREE: export (save) the data as an sav file using the write_sav function from
# the haven library.
# More on the export step in a later section.
library(haven)
write_sav(d,"exported_donnee.sav")
```

Lecture / Importation des données

Pour lire avec succès les données d'un CSV, il y a trois questions importantes à poser.

1. Comment les données sont-elles séparées dans ce fichier csv? Par des virgules (par défaut) ou par d'autres moyens (points-virgules, espaces, etc.)?
2. Comment les AN sont-elles représentées dans ce fichier csv?
3. Est-ce que je veux que R garde les noms de variables d'origine, ou peut-il fixer les noms de variables pour qu'ils soient lisibles dans R?

Séparation des entrées de données

Pour commencer à répondre à ces questions, une bonne première étape consiste à importer les données dans R à l'aide de la commande read_csv, puis à regarder les six premières lignes de données à l'aide de la commande head.

```
ex1 <- read.csv("Example.csv")
head(ex1)
```

```
##   ex.pays ex.interviewer ex.temps ex.registrer ex.nombre_denfants
## 1    Mali            ---       EB         0.82                 1
## 2    Mali        adoptee       EF         0.56                 2
## 3      GB    participant       EF         0.44               ---
## 4    Mali         simple       EB         0.55                 5
## 5     SEN        adoptee       EF         0.29               ---
## 6     SEN    participant       EF         0.45                 2
```

Ci-dessus, le résultat de la commande head vous donne le nombre correct de colonnes et les colonnes sont remplies de données comme prévu. Génial - le fichier utilise probablement la virgule pour séparer les valeurs de données. Si vous ouvrez "Example.csv" dans un éditeur de texte, vous verrez en effet que toutes les entrées sont séparées par des virgules.

Mais que se passe-t-il si les données sont séparées par un caractère différent? Ci-dessous, nous voyons un tel exemple.

```
ex2 <- read.csv("Example - Semicolons.csv")
head(ex2)
```

```
##   ex.pays.ex.interviewer.ex.temps.ex.registrer.ex.nombre_denfants
## 1                                                 Mali;---;EB;0.82;1
## 2                                             Mali;adoptee;EF;0.56;2
## 3                                         GB;participant;EF;0.44;---
## 4                                              Mali;simple;EB;0.55;5
## 5                                            SEN;adoptee;EF;0.29;---
## 6                                         SEN;participant;EF;0.45;2
```

Si vous voyez que les données ne sont chargées que dans une colonne ou une ligne (ou si aucune donnée ne s'affiche), ouvrez le fichier csv dans un éditeur de texte. Comment les entrées de données sont-elles séparées? Si un caractère autre qu'une virgule est utilisé, nous devrons dire à R qu'il sait lire correctement les données. Si vous ouvrez "Example - Semicolons.csv" dans un éditeur de texte, vous verrez qu'il est séparé par des points-virgules. Nous pouvons utiliser l'argument 'sep' dans la fonction read.csv pour indiquer à R que nous devons utiliser des points-virgules pour séparer les entrées de données, puis utiliser de nouveau head pour regarder les données et voir si le problème a été corrigé.

```
ex3 <- read.csv("Example - Semicolons.csv", sep = ';')
head(ex3)
```

```
##   ex.pays ex.interviewer ex.temps ex.registrer ex.nombre_denfants
## 1    Mali            ---       EB         0.82                 1
## 2    Mali        adoptee       EF         0.56                 2
## 3      GB    participant       EF         0.44               ---
## 4    Mali         simple       EB         0.55                 5
## 5     SEN        adoptee       EF         0.29               ---
## 6     SEN    participant       EF         0.45                 2
```

Effectivement, maintenant les données se chargent correctement.

Non Applicables

Après chargement dans les données en utilisant la séparation d'entrée de données correcte, nous devons également vérifier que R a identifié les entrées NA dans les données. La méthode par défaut pour écrire NA dans R est 'NA', et si vos données utilisent un ensemble différent de caractères pour représenter les entrées NA, vous devez demander à R de rechercher cet ensemble de caractères différent. Trouvez un élément NA dans l'entrée principale de votre jeu de données. Le voyez-vous représenté comme le texte NA, ou autre chose?

```
ex4 <- read.csv("Example.csv")
head(ex4)
```

```
##   ex.pays ex.interviewer ex.temps ex.registrer ex.nombre_denfants
## 1    Mali            ---       EB         0.82                 1
## 2    Mali        adoptee       EF         0.56                 2
## 3      GB    participant       EF         0.44               ---
## 4    Mali         simple       EB         0.55                 5
## 5     SEN        adoptee       EF         0.29               ---
## 6     SEN    participant       EF         0.45                 2
```

Dans cet exemple, nous voyons que les éléments NA sont représentés par '—'. Puisque R ne sait pas que '—' est NA, il verra '—' comme une chaîne, et donc lira n'importe quelle colonne avec '—' comme chaînes ou facteurs, même si le reste des éléments dans la colonne sont numériques. Pour résoudre ce problème, nous pouvons utiliser l'argument na.strings dans la fonction read.csv.

```
ex5 <- read.csv("Example.csv", na.strings = '---')
head(ex5)
```

```
##   ex.pays ex.interviewer ex.temps ex.registrer ex.nombre_denfants
## 1    Mali          <NA>       EB         0.82                 1
## 2    Mali        adoptee       EF         0.56                 2
## 3      GB    participant       EF         0.44                NA
## 4    Mali         simple       EB         0.55                 5
## 5     SEN        adoptee       EF         0.29                NA
## 6     SEN    participant       EF         0.45                 2
```

Maintenant, les entrées NA apparaissent comme NA, comme nous le voulons.

Conservation ou correction de noms de variables

Parfois, les noms de variables (en-têtes de colonne) dans le fichier csv utiliseront des caractères que R ne peut pas utiliser comme noms de variables. Les caractères d'exemple qui ne sont pas valides dans les noms de variables R sont -, *, $, + et les espaces. C'est parce que ces caractères représentent des opérations dans R. Par exemple, si vous aviez les variables a, b et a-b, comment R sait-il si a-b signifie la variable 'a-b' ou la variable 'a' moins la variable 'b'?

Si vous ne demandez pas à R de conserver les noms de variables d'origine, tous les noms de variables seront corrigés en changeant tous les caractères que R ne peut pas utiliser pour les points. Parfois, cependant, vous voudrez conserver les noms de variables d'origine (nous parlerons de quand dans la section de données de nettoyage). Si vous souhaitez conserver les noms de variables d'origine, vous pouvez le faire en utilisant l'argument check.names dans la fonction read.csv.

```
ex6 <- read.csv("Example.csv", check.names = FALSE)
head(ex6)
```

```
##   ex-pays ex-interviewer ex-temps ex-registrer ex-nombre_denfants
## 1    Mali            ---       EB         0.82                 1
```

```
## 2    Mali        adoptee       EF         0.56                2
## 3      GB    participant       EF         0.44              ---
## 4    Mali         simple       EB         0.55                5
## 5     SEN        adoptee       EF         0.29              ---
## 6     SEN    participant       EF         0.45                2
```

Ici, nous pouvons voir que les noms de variables qui apparaissaient auparavant comme «ex.pays» et «ex.interviewer» sont maintenant «ex-pay» et «ex-interviewer», comme ils étaient à l'origine dans le fichier csv.

Nettoyage des données

Avant d'exporter vers SPSS, nous voulons souvent rendre les données plus faciles à utiliser. Cela peut signifier changer les noms des variables ou le type (numérique, chaîne, facteur, etc.) d'une colonne de données.

Changement de noms de variables

Pour changer les noms des variables, nous devons d'abord savoir quels sont les noms des variables. Nous pouvons comprendre cela en utilisant la commande names ().

```
ex7 <- read.csv("Example.csv")
names(ex7)
```

```
## [1] "ex.pays"            "ex.interviewer"     "ex.temps"
## [4] "ex.registrer"       "ex.nombre_denfants"
```

Lisez les noms et décidez: êtes-vous d'accord avec les noms de variables, ou voulez-vous les reformater? Les reformater pourrait signifier les raccourcir, en les changeant de majuscules en minuscules, ou toute autre manipulation de chaîne que vous pouvez trouver. Si vous êtes satisfait des noms de variables tels qu'ils sont, génial - vous pouvez passer cette section! Si vous n'êtes pas satisfait des noms de variable, demandez-vous: puis-je écrire des instructions claires sur la façon de changer les noms de ces variables? Imaginez que vous deviez donner ces instructions à un étranger, avec la liste des noms de variables. S'ils peuvent utiliser les instructions pour changer correctement toutes les variables, alors les instructions sont bonnes.

Exemples de bonnes instructions (claires):

- Supprimer tout le texte jusqu'à et y compris la dernière période, gardez tout à droite de la dernière période.
- Changer toutes les lettres en minuscules et changer tout '. . . ' en '_'

Si dans ce cas nous pouvons décider de supprimer tout le texte jusqu'à et y compris la dernière période. Pour ce faire, nous pouvons utiliser la fonction gsub ().

```
ex8 <- read.csv("Example.csv")
names(ex8) <- gsub('.*\\.','',names(ex8))
names(ex8)
```

```
## [1] "pays"            "interviewer"     "temps"           "registrer"
## [5] "nombre_denfants"
```

Notez que nous n'appelons pas simplement gsub, nous affectons le résultat du gsub aux noms (ex8) pour mettre à jour les noms des variables. Assurez-vous de mettre à jour les noms des variables en attribuant une nouvelle valeur à names () chaque fois que vous voulez faire une modification, sinon votre travail ne sera pas sauvegardé!

Si vous ne voyez aucun motif dans vos noms de variables qui vous laisse penser à de bonnes instructions de changement, il peut être plus facile de regarder les noms de variables d'origine (souvenez-vous, sauf si

vous indiquez R sinon, il remplace les caractères lire dans les noms de variables aux périodes). Pour ce faire, utilisez l'option check.names mentionnée dans la section d'importation:

```
ex9 <- read.csv("Example.csv", check.names = FALSE)
names(ex9)
```

```
## [1] "ex-pays"            "ex-interviewer"     "ex-temps"
## [4] "ex-registrer"       "ex-nombre_denfants"
```

Si vous pouvez voir une bonne règle ou un ensemble d'instructions à utiliser maintenant, génial. Sinon, R peut ne pas être en mesure de vous aider, car vous ne pouvez pas dire à R ce qu'il faut faire si vous n'avez pas d'instructions à lui donner!

Ici, nous pouvons décider de supprimer tout le contenu des noms de variables dans le dernier tiret '-', ou de supprimer simplement toutes les versions de la phrase 'ex-'. Soit fonctionne très bien, et le code pour les deux sont ci-dessous:

```
# Removing everything up through the last dash
 gsub('.*-','',names(ex9))
```

```
## [1] "pays"            "interviewer"     "temps"           "registrer"
## [5] "nombre_denfants"
```

```
# Removing all instances of the phrase 'ex-'
gsub('ex-','',names(ex9))
```

```
## [1] "pays"            "interviewer"     "temps"           "registrer"
## [5] "nombre_denfants"
```

Notez que ce sont des exemples d'essais de code, mais ils n'ont pas affecté la sortie à names (), donc aucun travail n'est sauvegardé. Si nous voyons qu'ils fonctionnent tous les deux, nous pouvons choisir l'un ou l'autre. Disons que nous choisissons la première voie. Ensuite, le code que nous aurions besoin d'écrire pour enregistrer les résultats de notre gsub dans les noms de variables est

```
names(ex9) <- gsub('.*-','',names(ex9))
```

Si vous utilisez les noms de variables d'origine, après les avoir manipulés, vous devez vous assurer que R peut les lire. Pour ce faire, nous utilisons la fonction make.names (). Si vous oubliez cette étape, vous risquez d'obtenir des erreurs dans votre code R ou des variables bizarres (v1, v2, etc.) dans votre fichier SPSS.

```
names(ex9) <- make.names(ex9)
```

Vous pouvez en savoir plus sur la [fonction gsub ici] (http://www.endmemo.com/program/R/gsub.php) et plus sur [manipulations de chaînes en général ici] (http://www.gastonsanchez.com /Handling_and_Processing_Strings_in_R.pdf). Il y a une grande variété de commandes que vous pouvez utiliser, mais un résumé des commandes clés susceptibles de venir pour les données de Tostan est

```
ex10 <- read.csv("Example - Names.csv", check.names = FALSE)
# The original names
names(ex10)
```

```
## [1] "EXAMPLE-DATA.COUNTRY"     "EXAMPLE-DATA.INTERVIEWER"
## [3] "EXAMPLE-DATA.PERIOD"
```

```
# Removing all characters up through the last period (here, \\. represents a period)
gsub('.*\\.','',names(ex10))
```

```
## [1] "COUNTRY"     "INTERVIEWER" "PERIOD"
```

```
# Removing all characters up through the last dash
gsub('.*-','',names(ex10))
```

```
## [1] "DATA.COUNTRY"     "DATA.INTERVIEWER" "DATA.PERIOD"
```

```
# Removing a specific set of characters
gsub('EXAMPLE-DATA.','',names(ex10))
```

```
## [1] "COUNTRY"     "INTERVIEWER" "PERIOD"
```

```
# Changing a set of characters to another set of characters
gsub('EXAMPLE-DATA','data', names(ex10))
```

```
## [1] "data.COUNTRY"     "data.INTERVIEWER" "data.PERIOD"
```

```
# Changing the words to lowercase (to change to uppercase, use toupper())
tolower(names(ex10))
```

```
## [1] "example-data.country"     "example-data.interviewer"
## [3] "example-data.period"
```

```
# Make the names readible by R
make.names(names(ex10))
```

```
## [1] "EXAMPLE.DATA.COUNTRY"     "EXAMPLE.DATA.INTERVIEWER"
## [3] "EXAMPLE.DATA.PERIOD"
```

Vous pouvez également faire plusieurs de ces commandes dans une rangée. Notez que cet ordre est important! Et n'oubliez pas d'attribuer la nouvelle valeur à names () chaque fois que vous utilisez une commande. Par exemple, voici les mêmes commandes de manipulation de chaînes que celles utilisées dans l'exemple de script en haut de ce fichier.

```
# Read in the data, using the original variable names
ex11 <- read.csv("18 01 08 donnee.csv", check.names=FALSE)
# Look at the first 15 variable names
head(names(ex11),15)
```

```
##  [1] "Number"
##  [2] "Pays"
##  [3] "localization.departement"
##  [4] "localization.commune"
##  [5] "localization.village"
##  [6] "type_de_communaut"
##  [7] "type_evaluation"
##  [8] "interviewer"
##  [9] "ethnie.nbre_groupe_ethnique"
## [10] "demog_enquete.CE1_caracteristiques_enquete.CE2_sexe"
## [11] "demog_enquete.CE1_caracteristiques_enquete.CE3_groupe_ethnique"
## [12] "demog_enquete.CE1_caracteristiques_enquete.CE4_age_enquete.CE4x1_connait_age"
## [13] "demog_enquete.CE1_caracteristiques_enquete.CE4_age_enquete.CE4x1a_age_exact"
## [14] "demog_enquete.CE1_caracteristiques_enquete.CE4_age_enquete.CE4x1b_tranche_age"
## [15] "demog_enquete.CE1_caracteristiques_enquete.CE5_etat_matrimonial"
```

```
# Remove the text up through the last period
names(ex11) <- gsub(".*\\.", "",names(ex11))
# Change the variable names so that R can read them (this will create some '...')
names(ex11) <- make.names(names(ex11))
# Change instances of '...' to '_' to increase readability
names(ex11) <- gsub("\\.\\.\\.",'_',names(ex11))
# Look at the first 15 resulting variable names
head(names(ex11),15)
```

```
##  [1] "Number"               "Pays"                 "departement"
```

```
##  [4] "commune"              "village"              "type_de_communaut"
##  [7] "type_evaluation"      "interviewer"          "nbre_groupe_ethnique"
## [10] "CE2_sexe"             "CE3_groupe_ethnique"  "CE4x1_connait_age"
## [13] "CE4x1a_age_exact"     "CE4x1b_tranche_age"   "CE5_etat_matrimonial"
```

Modification des types de colonnes

Parfois, R ne peut pas deviner le type (entier, numérique, chaîne, facteur, etc) de chaque colonne correctement. Vous pouvez corriger cela avant d'exporter vers SPSS. Si vous remarquez dans votre fichier SPSS qu'une colonne n'est pas du bon type, vous pouvez le changer en utilisant les commandes

```
ex12 <- read.csv("18 01 08 donnee.csv", na.strings = '---', check.names=FALSE)
# Convert to integer
ex12$Number <- as.integer(ex12$Number)
# Convert to factor
ex12$localization.commune <- as.factor(ex12$localization.commune)
# Convert to decimal
ex12$ethnie.nbre_groupe_ethnique <- as.numeric(ex12$ethnie.nbre_groupe_ethnique)
# Convert to character
ex12$interviewer <- as.character(ex12$interviewer)
```

Ecriture / Exportation des données

Une fois que vos données sont dans un format qui vous convient, vous pouvez les exporter vers un fichier sav que vous pouvez ouvrir dans SPSS. Pour ce faire, nous devons utiliser une bibliothèque (également appelée package) appelée haven. Si vous n'avez jamais installé de refuge auparavant, allez-y et cliquez sur Outils> Installer les paquets, tapez le mot «havre» dans la barre des paquets, cliquez sur Installer et attendez que l'installation soit terminée. Si vous avez déjà installé un havre de paix, ou pensez y être, allez-y et passez à l'étape suivante - s'il s'avère que le havre n'est pas installé, vous obtiendrez une erreur qui se lit

Error in library(haven) : there is no package called 'haven'

Dans ce cas, vous devez installer le haven en suivant les instructions ci-dessus.

Une fois le havre installé, nous devons le charger dans un fichier si nous voulons l'utiliser. Pour charger le havre, nous utilisons la bibliothèque de commandes (haven). Après cette commande, nous pouvons utiliser le fichier write_sav () de la bibliothèque haven pour créer un fichier sav, avec le nom de la donnée et le nouveau nom de fichier comme arguments. Le fichier apparaîtra alors dans le même répertoire / dossier que partout où le fichier exécutant la commande write_sav () est sauvegardé.

```
library(haven)
ex13 <- read.csv("18 01 08 donnee.csv", na.strings = '---', check.names=FALSE)
write_sav(ex13,"new_sav_file.sav")
```

Si la commande write_sav () s'exécute sans erreur, vous devriez pouvoir ouvrir votre nouveau fichier sav dans SPSS maintenant! Vous pouvez obtenir une erreur ou deux la première fois que vous exécutez votre fichier, et c'est correct, car il existe des moyens de corriger ces erreurs.

Erreurs dans le code R

Voici quelques-unes des erreurs les plus probables que vous rencontrerez:

Error: SPSS only supports levels with <= 120 characters Problems: `Number`

Cela signifie que R pense que la colonne Nombre devrait être un facteur, mais il y a plus de 120 éléments différents dans cette colonne, ce qui représente un trop grand nombre d'options pour un facteur dans SPSS. Nombre doit en fait être un nombre entier, donc pour résoudre ce problème, vous pouvez utiliser la fonction as.integer dans la colonne problématique

```
library(haven)
ex13 <- read.csv("18 01 08 donnee.csv", na.strings = '---', check.names=FALSE)
ex13$Number <- as.integer(ex13$Number)
write_sav(ex13,"new_sav_file.sav")
```

comme nous l'avons vu dans la section de nettoyage des données. Si la colonne avec l'erreur doit être une chaîne, changez-la en caractères en utilisant la commande

```
ex13$Number <- as.character(ex13$Number)
```

Après avoir changé le format en caractère (ou parfois avec des colonnes qui sont déjà des caractères plutôt que des facteurs), vous pourriez voir l'erreur

Error in write_sav_(data, normalizePath(path, mustWork = FALSE)) : Writing failure: A provided string value was longer than the available storage size of the specified column.

Cela signifie que l'une des entrées de données est trop longue et prend trop de place (il s'agit probablement d'une longue réponse textuelle à une question ouverte). Une façon potentielle de résoudre ceci est de réencoder la colonne problématique d'une manière qui occupe moins d'espace, ce que vous pouvez faire en utilisant la fonction enc2utf8 ():

```
ex13$Number <- enc2utf8(ex13$Number)
```

Erreurs dans le fichier SAV après l'ouverture de SPSS

Vous pouvez également voir des problèmes dans le fichier SAV lorsque vous l'ouvrez dans SPSS. Le plus commun de ces problèmes est que certains ou tous les noms de variables seront remplacés par v1, v2, v3, etc. Si tous les noms de variables sont perdus, vous avez probablement oublié de vous assurer que les noms des variables sont lisibles dans R. Ajouter la ligne

```
names(data) <- make.names(data)
```

avant la commande write_sav () dans votre code, remplacez les données de mot ci-dessus par le nom de votre variable de données.

Si, au contraire, seules certaines variables de votre fichier SPSS sont manquantes, vous avez probablement des noms de variables en double dans votre schéma de nommage. Essayez d'ajouter la ligne

```
names(data)[duplicated(names(data))]
```

à votre code, juste avant la commande write_sav (), en remplaçant à nouveau le mot data par le nom de votre variable de données. Cette ligne imprimera tous les noms en double dans votre code, donc si quelque chose imprime, vous avez des doublons! Repensez votre schéma de nommage afin qu'il n'y ait plus de variables en double.