Objectif

Dans le tutoriel précédent, vous avez appris les bases du langage R. Vous avez entre autres effectué des calculs, utilisé des fonctions ou encore assigné des résultats. Ce tutoriel va vous permettre d'utiliser le logiciel R pour résoudre une question biologique sur l'obésité. Vous serez guidé tout au long de ce travail.

Introduction

Dans ce tutoriel, nous utiliserons ensemble R pour résoudre une question biologique à travers l'analyse d'un jeu de données concernant la biométrie humaine.

Manipuler

En partant d'un tableau de données relatif à de la biométrie humaine contenant 395 observations et sept variables :

```
# Chargement de SciViews::R
SciViews::R(lang = "fr")
# Importation du jeu de données
biometry <- read("biometry", package = "BioDataScience")
# Affichage des 10 premières lignes du tableau
tabularise(biometry, max.rows = 10)
```

Genre	Date de naissance	Masse [kg]	Taille [cm]	Tour de poignet [mm]	Année de la mesure	Age [année]
M	1995-03-11	69	182	15.0	2013	18
M	1998-04-03	74	190	16.0	2013	15
M	1967-04-04	83	185	17.5	2013	46
M	1994-02-10	60	175	15.0	2013	19
W	1990-12-02	48	167	14.0	2013	23
W	1994-07-15	52	179	14.0	2013	19
W	1971-03-03	72	167	15.5	2013	42
W	1997-06-24	74	180	16.0	2013	16
M	1972-10-26	110	189	19.0	2013	41
M	1945-03-15	82	160	18.0	2013	68

385 more rows

Notez ceci :

- Les encadrés gris dénotent des instructions entrées dans R. Si ce dernier renvoie des résultats, ils sont présentés dans des encadrés blancs juste en dessous.

- Toujours commencer par l'instruction `SciViews::R()` pour installer les différents outils (rassemblés dans des extensions du programme, appelées "packages" R) dont nous aurons besoin. Cette instruction peut comprendre des arguments comme `lang = "fr"` . Cet

argument va avoir pour effet de définir le français comme langue par défaut (là où cela est supporté).

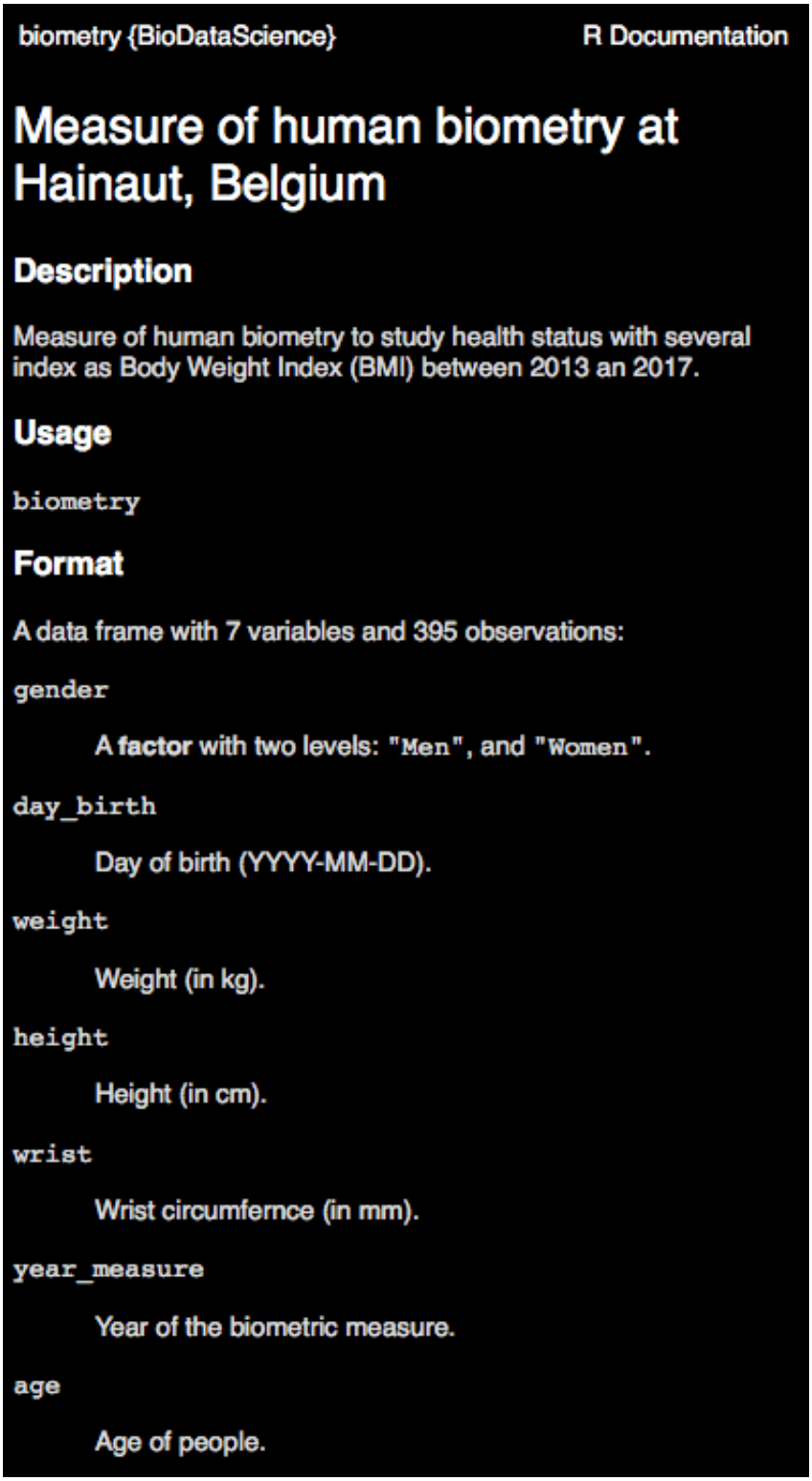
- L'anglais est la langue la plus employée en science des données. Les jeux de données employés sont encodés en anglais. En définissant le français comme langue par défaut, si une traduction est disponible, certains graphiques et tableaux que l'on va réaliser auront automatique des labels en français.

- Le jeu de données est disponible dans un "package" R (un package est une sorte d'"addin" pour R qui apporte de nouvelles fonctions et de nouveaux jeux de données à votre environnement de travail) : {BioDataScience}, spécialement préparé pour ce cours. Notez que le nom des packages est présenté toujours entre accolades.

- La fonction `read()` permet de lire les données issues du package {BioDataScience}. La fonction `tabularise()` a pour objectif de formater correctement un tableau (et bien plus encore. Vous le découvrirez tout à long de ce cours).

Le point d'interrogation devant notre jeu de données renvoie vers une page d'aide, tout comme pour les fonctions.

```
?biometry
```



biometry {BioDataScience} R Documentation

Measure of human biometry at Hainaut, Belgium

Description

Measure of human biometry to study health status with several index as Body Weight Index (BMI) between 2013 an 2017.

Usage

```
biometry
```

Format

A data frame with 7 variables and 395 observations:

`gender`

A **factor** with two levels: `"Men"`, and `"Women"`.

`day_birth`

Day of birth (YYYY-MM-DD).

`weight`

Weight (in kg).

`height`

Height (in cm).

`wrist`

Wrist circumfernce (in mm).

`year_measure`

Year of the biometric measure.

`age`

Age of people.

Nous commençons par sélectionner des colonnes d'intérêt du tableau initial `biometry` .

```
bio <- select_(biometry, ~-day_birth, ~-wrist, ~-year_measure)
```

Notez ceci :

Nous réalisons cette sélection avec la fonction `select_()` en éliminant des colonnes du tableau que nous n'utiliserons pas (les variables `day_birth` , `wrist` et `year_measure`). Pour indiquer que nous voulons *éliminer* ces variables, nous les précédons d'un signe `-` , et enfin, nous utilisons

l'interface formule, donc nous précédons encore ces instructions d'un `~`. Le résultat de cette fonction est assigné à `bio` grâce à l'opérateur d'assignation `<-` déjà vu dans le tutoriel précédent.

Utilisons la fonction `tabularise()` du package {tabularise} maintenant pour présenter les huit premières lignes de notre jeu de données.

```
tabularise(bio, max.rows = 8)
```

Genre	Masse [kg]	Taille [cm]	Age [année]
M	69	182	18
M	74	190	15
M	83	185	46
M	60	175	19
W	48	167	23
W	52	179	19
W	72	167	42
W	74	180	16

387 more rows

Notez ceci :

Le "pipe" `%>.%` (prononcez "païpe" comme en anglais) et le chaînage que nous avons également découvert à la fin du précédent tutoriel sont employés dans les instructions ci-dessus.

Considérons maintenant uniquement les femmes. On peut filtrer les lignes d'un tableau de données en utilisant comme critère `gender == "W"` (**attention :** notez bien que dans un *test de condition*, l'égalité s'écrit avec **deux** signes égaux dans R). Les autres options sont : `!=` pour différent de, `>` pour plus grand que, `<` pour plus petit que, ainsi que `>=` ou `<=` pour plus grand ou égale et plus petit ou égal.

```
bio %>.%
  filter_(., ~gender == "W") %>.%
  tabularise(., max.rows = 8)
```

Genre	Masse [kg]	Taille [cm]	Age [année]
W	48	167	23
W	52	179	19
W	72	167	42

W	74	180	16
W	61	154	47
W	57	164	49
W	58	162	76
W	61	168	53

189 more rows

Continuons à manipuler notre tableau en sélectionnant des colonnes et en filtrant les lignes en une seule opération. Pour sélectionner les femmes `gender == "W"` et retirer la colonne `age`, nous utiliserons :

```
bio %>.%
  filter_(., ~gender == "W") %>.% # Sélectionne les femmes
  select_(., ~-age) %>.% # Retire la colonne âge
  tabularise(., max.rows = 8)
```

Genre	Masse [kg]	Taille [cm]
W	48	167
W	52	179
W	72	167
W	74	180
W	61	154
W	57	164
W	58	162
W	61	168

189 more rows

Lorsque la succession des opérations à réaliser devient longue, il est difficile de débusquer les erreurs dans le “pipeline” parce qu’en fait, toutes les opérations sont regroupées en une seule longue et même instruction R. C’est un peu comme si vous écriviez un roman en une seule phrase avec juste un point à la fin du livre ! Pour éviter cela, tout en conservant la clarté du code, agencé de telle manière que l’on comprenne bien qu’il s’agit d’étapes successives d’un même traitement, vous pouvez aussi utiliser la forme “bullet-point” avec le pseudo-opérateur `.=`. Le code précédent s’écrira alors comme ceci (comparez attentivement les deux formes) :

```
.= bio
.= filter_(., ~gender == "W") # Sélectionne les femmes
.= select_(., ~-age)          # Retire la colonne âge
.= tabularise(., max.rows = 8)
```

Les `.=` en début de ligne forment des espèces de puces (“bullet points” en anglais), ce qui permet donc de présenter vos étapes successives comme un ensemble d’éléments d’une liste. Cela se rapproche plus de ce que vous écririez naturellement pour présenter une recette (de cuisine, par exemple) sous forme abrégée. Vous n’écrirez, en effet, jamais “éplucher les pommes de terre *et puis* remplir une casserole d’eau *et puis* y placer les pommes de terre *et puis* faire bouillir l’eau…” (le *et puis* représentant votre opérateur de pipe). Vous écrirez plutôt :

- éplucher les pommes de terre
- remplir une casserole d’eau
- y placer les pommes de terre
- faire bouillir l’eau
- …

Cela se rapproche plus de la forme “bullet-point”, non ? De plus, dans cette dernière forme, chaque ligne est une instruction indépendante et il est possible de suivre facilement le remaniement des données en utilisant `View(.)`, puisqu’en réalité `.=` ne fait rien d’autre qu’assigner à l’objet `.` le résultat du calcul à chaque étape.

Maintenant que nous avons vu comment lire, remanier et présenter des tableaux de données dans R (nous reviendrons sur ces notions plus tard), nous pouvons explorer ses potentialités pour réaliser des graphiques à la section suivante…

Continue

L’obésité

Plusieurs médias publient ou ont publié récemment des articles avec des titres accrocheurs comme obésité, le mal du siècle, 13% de la population adulte mondiale est obèse, 20% pourraient l’être en 2025 (https://www.lemonde.fr/sante/article/2016/04/01/13-de-la-population-adulte-mondiale-est-obese-20-pourrait-bientot-l-etre_4893671_1651302.html) ou encore obésité et malnutrition, fléaux du XXI^{e} siècle (http://www.natura-sciences.com/sante/obesite-malnutrition.html). Ils se basent sur plusieurs déclarations de l’Organisation Mondiale de la Santé (OMS) indiquant que la lutte contre l’obésité sera l’un des défis majeurs pour la santé publique au 21^{e} siècle. L’OMS estime que 1,5 milliard de personnes sont en surpoids actuellement et ce chiffre augmentera si rien ne change.

Une multitude d’indicateurs pour quantifier l’excédent de poids ont été employés au cours du temps (formule de Lorentz (https://www.calculersonimc.fr/autres-calculs/poids-ideal-lorentz.html), formule de Creff (https://www.calculersonimc.fr/autres-calculs/poids-ideal-creff.html) ou encore formule de Broca (https://www.calculersonimc.fr/autres-calculs/poids-ideal-broca.html)). Actuellement, c’est l’indice de masse corporelle (**IMC**, ou encore *BMI* en anglais) qui est l’indicateur le plus employé. La formule est la suivante :

$$imc\ [kg/m^2] = \frac{masse\ [kg]}{taille[m]^2}$$

ou encore en anglais (puisque le nom des variables est en anglais dans notre jeu de données) :

$$bmi\ [kg/m^2] = \frac{weight\ [kg]}{height\ [m]^2}$$

Une fois la valeur de l’IMC obtenue, il faut la comparer au tableau ci-dessous pour connaitre son état de santé.

IMC (kg/m2)	Interprétation (selon l’OMS)
Inférieur 18.5	Sous-poids (en anglais `underweight`)
Entre 18.5 et 25	Corpulence normale (en anglais `normal weight`)

Entre 25 et 30	Surpoids (en anglais `overweight`)
Supérieur à 30	Obésité (en anglais `obese`)

Nous allons maintenant avancer pas à pas dans cette première analyse avec R. Le but est de calculer l'IMC (qui sera dans la variable `bmi`), et puis de visualiser comment cet indice se répartit dans la population étudiée.

Calcul de l'IMC

Les observations relatives à la première personne mesurée se présentent comme ceci dans le jeu de données :

id	gender	weight [kg]	height [cm]
1	W	50	170

Rappelez-vous de la formule qui est :

$$bmi\ [kg/m^2] = \frac{weight\ [kg]}{height\ [m]^2}$$

Rappelez-vous aussi que vous pouvez employer les opérations mathématiques de base dans R. Elles respectent l'ordre de priorité des opérateurs mathématiques. Au besoin, il est possible d'indiquer explicitement, ou de modifier les priorités avec des parenthèses comme `3 * (2 + 1)`.

Opérations de base	Symboles
addition	+
soustraction	–
division	/
multiplication	*
puissance	^

La formule mathématique se traduit donc comme suit en une instruction que R peut utiliser :

```
50 / (170/100)^2
```

```
[1] 17.30104
```

Nous avons dû diviser la taille par 100 car elle est donnée en centimètres alors que nous devons l'avoir en mètres. Rappelez-vous également que R indique un `[1]` devant la réponse. En fait, R travaille avec des vecteurs (même si ici, le vecteur ne contient qu'un seul élément). Ainsi, le nombre entre crochets devant indique la position dans le vecteur. Ce calcul sur vecteurs nous sera très utile lorsque nous traiterons l'ensemble du tableau. En effet, le même calcul sera *automatiquement distribué* sur tous les individus !

L'IMC de cette femme indique qu'elle est en **sous-poids** selon l'échelle de l'OMS.

Réalisez maintenant par vous-mêmes le calcul sur notre deuxième individu :

id	gender	weight [kg]	height [cm]
2	M	93	191

L'espace ci-dessous est une zone où vous pouvez entrer du code R. Le bouton `Run Code` permet ensuite de l'exécuter et de visualiser le résultat. Vous pouvez modifier autant de fois qu'il faut l'expression, et utiliser plusieurs fois `Run Code` . Lorsque vous êtes satisfait du résultat, cliquez sur `Submit Answer` . Dans les tutoriels, la `Solution` est également accessible, mais faites l'exercice par vous-même d'abord ! Dans les tests, vous n'y aurez pas accès, évidemment.

Calculez l'IMC de l'homme ci-dessus de 191 cm et de 93 kg.

Code R | Start Over | Solution | Run Code | Submit Answer

```
1
2
3
```

Calcul de l'IMC sur plusieurs individus

Vous vous retrouvez rapidement avec cinq nouveaux individus, femmes et hommes.

id	gender	weight [kg]	height [cm]
3	W	69	174
4	W	49	155
5	W	75	169
6	W	66	179
7	W	54	168

Le calcul un à un de l'IMC de chaque individu deviendra très rapidement fastidieux. Nous allons créer des **vecteurs** contenant plusieurs nombres en les **concaténant** avec la fonction `c()`. Nous utiliserons aussi l'assignation `<-` pour conserver le résultat du calcul dans un objet nommé (comme `weight_w` ci-dessous) :

```
# Assignation des valeurs de masses dans un vecteur nommé `weight_w`
weight_w <- c(69, 49, 75, 66, 54)
# Assignation des valeurs de tailles dans un vecteur nommé `height_w`
height_w <- c(174, 155, 169, 179, 168)
# Transformation du vecteur `height_w` de centimètre en mètre
height_w <- height_w / 100
# Calcul de l'IMC/BMI
weight_w / height_w^2
```

```
[1] 22.79033 20.39542 26.25958 20.59861 19.13265
```

Important :

- Choisissez bien les noms de vos objets. Ces noms doivent être courts, mais informatifs sur leur contenu.
- Rappelez-vous que des noms acceptables commencent par une lettre, et comportent ensuite des lettres, des chiffres, le trait souligné `_` ou le point `.`, mais évitez, si possible, le point qui a une signification particulière dans divers contextes.
- Comme il est difficile de mémoriser la casse d'un nom, il est conseillé d'utiliser uniquement des lettres minuscules.
- Si le nom est constitué de plusieurs mots, il est préférable de séparer ces mots par un trait souligné. Pour rappel, l'espace n'est pas utilisable. Par exemple, `tour_de_poignet`.

- Éviter d'utiliser des caractères accentués.
- Si possible, utilisez des noms en anglais. Certainement si votre travail sera échangé avec d'autres scientifiques en international… mais c'est une bonne habitude à prendre même sur votre propre code. Pour reprendre l'exemple précédent `wrist_circumference`

Réalisez les mêmes opérations sur les individus 8 à 12 qui sont des hommes.

id	gender	weight [kg]	height [cm]
8	M	82	174
9	M	73	186
10	M	105	203
11	M	61	172
12	M	95	190

Code R | Start Over | Hints | Run Code | Submit Answer

```r
# Assignation des valeurs de masses dans un vecteur nommé `weight_m`
weight_m <- c(___)
# Assignation des valeurs de tailles dans un vecteur nommé `height_m`
height_m <- ___
# Transformation du vecteur `height_m` de centimètre en mètre
height_m <- height_m / 100
# Calcul de IMC
___
```

Encodage d'un tableau de données

Il devient rapidement évident qu'il est plus simple que nos observations de terrain soient rassemblées en un jeu de données structuré. Pour cela vous allez créer ce qu'on appelle un objet **data.trame** (qui se traduit en français par "tableau de données") dans R. La fonction qui permet de le créer dans `SciViews::R` est `dtx()`. Cette dernière fonction permet de combiner vos différents vecteurs colonne par colonne dans un tableau.

Dans `dtx()`, vous entrerez vos différents vecteurs comme autant d'**arguments** de la fonction, séparés par une virgule `,`. De plus, vous pouvez nommer vos colonnes en donnant des noms aux arguments de type `nom = valeur`. Analysez avec attention l'exemple ci-dessous.

```
# Création du tableau de données (data.table) avec dtx()
woman <- dtx(
  id     = 3:7,                 # Valeurs numériques
  gender = rep("W", times = 5), # Chaines de caractères (! guillemets)
  weight = weight_w,            # Vecteur de masses créé précédemment
  height = height_w             # Vecteur de tailles créé précédemment
)
# Afficher le tableau
woman
```

id <int>	gender <chr>	weight <dbl>	height <dbl>
3	W	69	1.74
4	W	49	1.55
5	W	75	1.69
6	W	66	1.79
7	W	54	1.68

5 rows

Avez-vous remarqué la différence dans la façon d'encoder des valeurs numériques et des chaînes de caractères ?

Réalisez les mêmes opérations sur les individus de 8 à 12 (inspirez-vous des instructions ci-dessus et du tableau ci-dessous) :

id	gender	weight [kg]	height [cm]
8	M	82	174
9	M	73	186
10	M	105	203

11	M	61	172
12	M	95	190

Utilisez les vecteurs `weight_m` et `height_m` que vous avez créés à l'exercice précédent.

Code R | Start Over | Solution | Run Code | Submit Answer

```
# Encoder le jeu de données
man <- dtx(
  id     = ___,
  gender = ___,
  weight = ___,
  height = ___
)
# Afficher le tableau (simplement sans utiliser tabularise)
___
```

Calculez de nouveau l'IMC et ajoutez vos résultats dans le tableau de données. Vous avez à votre disposition la fonction `mutate_()` qui requiert comme argument le jeu de données et le nom de la nouvelle variable, suivie du signe égal `=` , puis de l'instruction qui calcule son contenu sous forme d'une formule (donc, précédée d'un `~` .

```
# Calculer l'IMC pour les femmes
woman <- mutate_(woman, bmi = ~weight / height^2)
# Afficher le tableau de données
woman
```

id <int>	gender <chr>	weight <dbl>	height <dbl>	bmi <dbl>
3	W	69	1.74	22.79033
4	W	49	1.55	20.39542
5	W	75	1.69	26.25958
6	W	66	1.79	20.59861
7	W	54	1.68	19.13265

5 rows

À retenir :

- Vous pouvez vous référer à d'autres colonnes du tableau (= autres variables) en utilisant leurs noms directement dans la formule,
- La ou les nouvelles colonnes sont ajoutées à la fin du tableau et sont directement utilisables.

Réalisez par vous-mêmes les mêmes opérations sur le jeu de données `man` .

Code R | Start Over | Solution | Run Code | Submit Answer

```
# Calculer l'IMC pour les hommes
man <- mutate_(man, bmi = ___)
# Afficher le tableau de données (sans tabularise)
___
```

Vous pouvez observer que tout comme le tableau de données portant sur les femmes, vous obtenez une nouvelle colonne au sein de votre tableau de données portant le nom `bmi` (pour “Body Mass Index”, soit l’IMC en français).

Fraction obèse de la population

Le monde titre que 13% de la population mondiale est obèse. Vérifiez cette affirmation avec le jeu de données `biometry` qui regroupe les masses et les tailles de 395 personnes adultes vivant sur le territoire belge (Hainaut, Belgique). Nous allons créer un tableau `bio_100` qui contient les 100 premières lignes de `biometry` (l'opérateur `[ ]` extrait un sous-tableau). Nous vous montrerons ensuite comment le travailler, et vous ferez de même sur le tableau complet `biometry` par vous-mêmes.

```
biometry <- read("biometry", package = "BioDataScience")
# Récupération des 100 premières lignes du tableau
bio_100 <- biometry[1:100, ]
```

Vous pouvez observer que la taille est ici exprimée en centimètres, il faut en tenir compte lors du calcul de l'IMC qui attend la taille exprimée en mètre. Un jeu de données réduit est employé pour expliciter les suites d'instructions `bio_100` qui ne reprend que 100 observations du jeu de données complet `biometry` .

Pour calculer l'IMC sur le jeu de données `bio_100` , nous employons à nouveau la fonction `mutate_()` .

```
bio_100 <- mutate_(bio_100, bmi = ~weight / (height / 100)^2)
# Afficher les premières lignes du tableau de données
head(bio_100, n = 5)
```

gender <fct>	day_birth <date>	weight <dbl>	height <dbl>	wrist <dbl>	year_measure <dbl>	a… <dbl>	bmi <dbl>
M	1995-03-11	69	182	15.0	2013	18	20.83082
M	1998-04-03	74	190	16.0	2013	15	20.49861
M	1967-04-04	83	185	17.5	2013	46	24.25128
M	1994-02-10	60	175	15.0	2013	19	19.59184
W	1990-12-02	48	167	14.0	2013	23	17.21109

5 rows

Calculez maintenant l'IMC sur le jeu de données `biometry` tout entier. Utilisez la fonction `mutate_()` . Nommez cette nouvelle variable `bmi` et assignez votre résultat de nouveau à `biometry`

Code R | Start Over | Hints | Run Code | Submit Answer

```
1 # Calcul de l'IMC
2 ___ <- ___
3 # Affichage des 5 premières lignes du tableau de données
```

```
3 # Affichage des 5 premières lignes du tableau de données
4 ___
```

Signification de l'IMC

Une fois la valeur obtenue de l'IMC, il faut lui attribuer son interprétation pour connaitre son état de santé. Rappel du tableau de l'OMS :

IMC (kg/m2)	Interprétation (selon l'OMS)
Inférieur 18.5	Sous-poids (en anglais `underweight`)
Entre 18.5 et 25	Corpulence normale (en anglais `normal weight`)
Entre 25 et 30	Surpoids (en anglais `overweight`)
Supérieur à 30	Obésité (en anglais `obese`)

Vous avez à votre disposition la fonction `case_when()` qui permet d'attribuer l'interprétation de l'OMS à la valeur d'IMC. Vous devez lui indiquer d'une part la condition (ex. : `bmi < 18.5`), et d'autre part son interprétation (ex. : `underweight`), le tout séparé par un `~` (donc, dans une formule). Vous pouvez retrouver les opérateurs de comparaison dans R ci-dessous.

Comparaison	Représentation
Égal à	==
Différent de	!=
Supérieur à	>
Inférieur à	<
Supérieur ou égal à	>=
Inférieur ou égal à	<=
Et (combinaison de tests)	&
Ou (idem)	|

Ajoutez une nouvelle variable qui tient compte de l'échelle de l'OMS dans le jeu de données `bio_100` dans la variable `bmi_cat`. Analysez la structuration de la suite d'instructions, les conditions employées, la position des guillemets …

```
# Ajouter la nouvelle variable
bio_100 <- mutate_(bio_100, bmi_cat = ~case_when(
  bmi < 18.5 ~ "underweight",
  bmi >= 18.5 & bmi < 25 ~ "normal weight",
  bmi >= 25 & bmi < 30 ~ "overweight",
  bmi >= 30 ~ "obese"
))
# Afficher les 5 premières lignes du tableau
head(bio_100)
```

gen… <fct>	day_birth <date>	wei… <dbl>	height <dbl>	wrist <dbl>	year_measure <dbl>	a.. <dbl>	bmi <dbl>	bmi_cat <chr>
M	1995-03-11	69	182	15.0	2013	18	20.83082	normal weight
M	1998-04-03	74	190	16.0	2013	15	20.49861	normal weight
M	1967-04-04	83	185	17.5	2013	46	24.25128	normal weight
M	1994-02-10	60	175	15.0	2013	19	19.59184	normal weight
W	1990-12-02	48	167	14.0	2013	23	17.21109	underweight
W	1994-07-15	52	179	14.0	2013	19	16.22921	underweight

6 rows

Ajoutez maintenant une nouvelle variable `bmi_cat` au jeu de données `biometry` en complétant les informations manquantes. Affichez les premières lignes du tableau avec la fonction `head()`.

Code R | Start Over | Solution | Run Code | Submit Answer

```
# Ajouter la nouvelle variable
___ <- mutate_(___, bmi_cat = ~case_when(
  bmi  ___ ___~ "underweight",
  bmi  ___ ___~ "normal weight",
  bmi  ___ ___~"overweight",
  bmi  ___ ___~ "obese"
))
# Afficher les 5 premières lignes du tableau
head(___)
```

Votre nouvelle variable contient des chaînes de caractères (du texte). Elle est de classe **character** dans R. Cependant c'est une variable non numérique. On parle encore de variable qualitative ou variable "facteur" en statistiques. Si nous voulons qu'elle soit comprise comme tel dans R, nous pouvons la convertir en un objet **factor** avec la fonction `factor()`. Avec l'argument `levels =`, nous spécifions l'ordre des différents niveaux de notre variable. Nous utilisons la fonction `mutate_()` comme précédemment.

Pour convertir en **factor** notre nouvelle variable `bmi_cat` dans `bio_100`, nous faisons donc :

```
bio_100 <- mutate_(bio_100, bmi_cat = ~factor(bmi_cat,
  levels = c("underweight", "normal weight", "overweight", "obese")))
bio_100
```

gen… <fct>	day_birth <date>	wei… <dbl>	height <dbl>	wrist <dbl>	year_measure <dbl>	a.. <dbl>	bmi <dbl>	bmi_cat <fct>
M	1995-03-11	69.0	182	15.0	2013	18	20.83082	normal weight
M	1998-04-03	74.0	190	16.0	2013	15	20.49861	normal weight
M	1967-04-04	83.0	185	17.5	2013	46	24.25128	normal weight
M	1994-02-10	60.0	175	15.0	2013	19	19.59184	normal weight
W	1990-12-02	48.0	167	14.0	2013	23	17.21109	underweight
W	1994-07-15	52.0	179	14.0	2013	19	16.22921	underweight
W	1971-03-03	72.0	167	15.5	2013	42	25.81663	overweight
W	1997-06-24	74.0	180	16.0	2013	16	22.83951	normal weight
M	1972-10-26	110.0	189	19.0	2013	41	30.79421	obese
M	1945-03-15	82.0	160	18.0	2013	68	32.03125	obese

1-10 of 100 rows Previous **1** 2 3 4 5 6 … 10 Next

En apparence, pas grand changement, mais maintenant, nous avons indiqué un ordre logique de progression dans les différents **niveaux** (`levels =`) de la variable qui est maintenant comprise comme **factor**. Ici, nous pourrions faire encore mieux. Si nous utilisons la fonction `ordered()` à la place de `factor()` , avec les mêmes arguments, nous indiquons à R qu'en plus d'être une variable qualitative, les différents niveaux sont classés du plus petit au plus grand (`underweight < normal weight < overweight < obese`). Nous pouvons voir la façon dont les différents niveaux sont encodés à l'aide de la fonction `levels()` .

Classer votre nouvelle variable avec le jeu de données `biometry` tout entier, et en utilisant `ordered()` ici.

Code R | Start Over | Solution | Run Code | Submit Answer

```
biometry <- mutate_(biometry, bmi_cat = ~ordered(___,
  levels = c(___)))
# Visualiser le vecteur
biometry$bmi_cat
# Extraire les niveaux d'encodage des niveaux
___(biometry$bmi_cat)
```

Vérification

La fonction `summary()` permet d'obtenir un résumé complet d'un tableau de données. La fonction `table()` renvoie ce que l'on appelle un **tableau de contingence**, soit un tableau qui dénombre les occurrences de chaque catégorie d'une variable telle que `bmi_cat` du tableau `bio_100`, ce qui se note `bio_100$bmi_cat` :

```
# Résumé des données
summary(bio_100)
```

```
 gender   day_birth             weight           height          wrist      
 M:56   Min.   :1937-04-13   Min.   : 48.00   Min.   :154.0   Min.   :10.00  
 W:44   1st Qu.:1965-10-06   1st Qu.: 60.00   1st Qu.:165.0   1st Qu.:16.00  
        Median :1990-01-21   Median : 70.00   Median :171.5   Median :16.65  
        Mean   :1981-03-01   Mean   : 72.18   Mean   :171.6   Mean   :16.65  
        3rd Qu.:1994-07-12   3rd Qu.: 81.00   3rd Qu.:177.2   3rd Qu.:17.70  
        Max.   :1998-04-03   Max.   :116.00   Max.   :193.0   Max.   :21.00  
                                                              NA's   :2      
 year_measure       age              bmi                bmi_cat    
 Min.   :2013   Min.   :15.00   Min.   :16.23   underweight  : 7  
 1st Qu.:2013   1st Qu.:19.00   1st Qu.:20.79   normal weight:54  
 Median :2013   Median :23.50   Median :23.78   overweight   :25  
 Mean   :2013   Mean   :32.48   Mean   :24.48   obese        :14  
 3rd Qu.:2013   3rd Qu.:48.25   3rd Qu.:26.60                     
 Max.   :2014   Max.   :76.00   Max.   :42.68                     
```

```
# Tableau de contingence pour bmi
table(bio_100$bmi_cat)
```

```
  underweight normal weight    overweight         obese 
            7            54            25            14 
```

Utilisez la fonction `summary()` pour obtenir une description du tableau de données complet `biometry`.

Code R | Start Over | Solution | Run Code | Submit Answer

```
1 # Résumé des données
2 summary(___)
3 
```

Pour déterminer le pourcentage d'obèses, nous pouvons générer un tableau de contingence sur l'ensemble des données dans `biometry`. Faites-le maintenant et assignez cette table au nom `bmi_tab`.

Code R | Start Over | Solution | Run Code | Submit Answer

```
# Calculer le pourcentage d'obèses via un tableau de contingence
bmi_tab <- ___(___$___)
bmi_tab
```

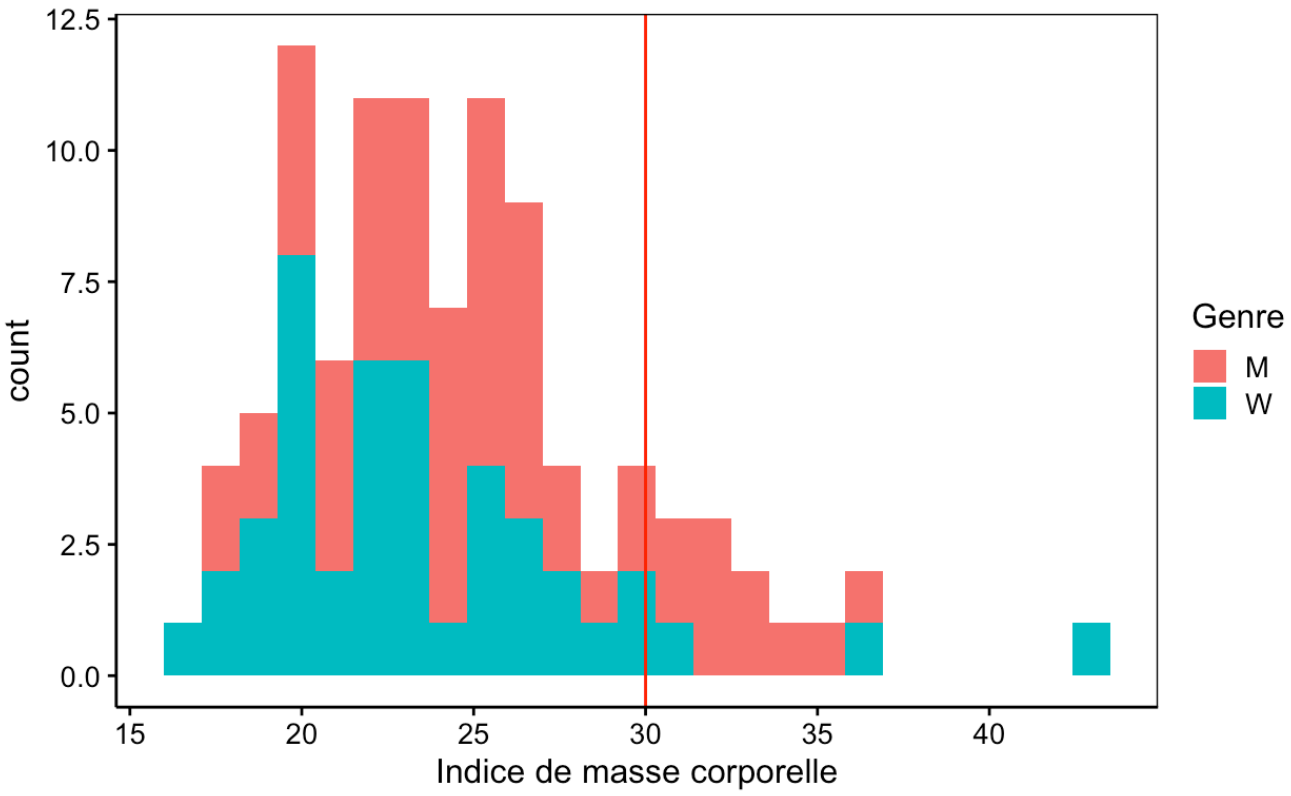
Vos résultats concordent-ils avec les valeurs avancées dans les médias ? Pour le savoir, vous divisez le nombre d'obèses observés par le nombre total d'observations du tableau que vous pouvez obtenir à l'aide de `nrow()` et vous multipliez le tout par 100 pour l'avoir en pourcentage. Notez bien que vous vectorisez l'opération sur tout le tableau de contingence pour avoir les pourcentages de toutes les catégories d'un seul coup :

```
bmi_tab / nrow(biometry) * 100
```

```
numeric(0)
```

Représentons graphiquement la distribution du `bmi` de nos individus sondés.

```
chart(data = bio_100, ~ bmi %fill=% gender) +
  geom_histogram(bins = 25) +
  geom_vline(xintercept = 30, color = "red")
```



Utilisez la fonction `chart()` pour représenter graphiquement la distribution des données par rapport au `bmi` de tous les individus de `biometry` .

Code R | Start Over | Solution | Run Code | Submit Answer

```
chart(data = ___, ~ ___ %fill=% gender) + # instruction du graphique
  geom_histogram(bins = 25) + # réaliser un histogramme
  geom_vline(xintercept = 30, color = "red") # ajouter une ligne à 30 de bmi
```

Conclusion

La fraction d'obèses dans notre jeu de données est de 12,7%, ce qui est très proche des 13% annoncés dans l'article paru dans le Monde.

Bravo ! Vous venez de terminer cette petite analyse dans un tutoriel learnr. Durant cette séance, vous avez découvert comment :

- Approfondir vos calculs dans R
- Vectoriser vos calculs
- Rassembler des données dans un tableau "data.table"
- Remanier un tableau de données
- Créer des variables qualitatives ordonnées (ordered) ou non (factor)
- Réaliser vos premiers graphiques dans R pour visualiser les données

Tout cela a été très vite et la quantité de nouveautés est importante. Rassurez-vous, vous ne devez pas retenir tout le contenu de ce tutoriel pour l'instant. Il s'agit essentiellement d'un tour rapide de ce que nous apprendrons à faire ensemble pendant le premier quadrimestre.

Laissez-nous vos impressions sur ce learnr

Entrez vos commentaires ici...

Soumettre une réponse