

1. Projet, session et fichiers

Ouvrir le fichier `.Rproj` du dossier de travail. Écrire les commandes dans un script et l'enregistrer. Un chemin relatif est interprété à partir du répertoire de travail courant : vérifier ce répertoire avant de lire ou d'écrire un fichier.

Besoin	Repère
Voir le répertoire courant	<code>getwd()</code>
Construire un chemin	<code>file.path("data", "raw", "mesures.csv")</code>
Vérifier un fichier	<code>file.exists(chemin)</code>
Charger un package installé	<code>library(readr)</code>
Ouvrir l'aide locale	<code>help("read_delim", package = "readr")</code>
Exécuter un script enregistré	<code>source("R/analyse.R", encoding = "UTF-8")</code>

Sous Windows, / convient dans les chemins écrits en R. Les packages sont installés sur le poste : les charger au début du script. L'exemple suivant utilise des données fictives écrites dans un fichier temporaire ; il s'exécute sans téléchargement.

```
library(readr)
fichier <- tempfile(fileext = ".csv")
writeLines(c("produit;quantite", "pomme;4", "poire;NA"), fichier)
mesures <- read_delim(
  fichier, delim = ";", na = c("", "NA"),
  locale = locale(encoding = "UTF-8"),
  col_types = cols(produit = col_character(), quantite = col_double())
)
mesures
```

```
# A tibble: 2 x 2
  produit quantite
  <chr>      <dbl>
1 pomme         4
2 poire        NA
```

`read_csv()` utilise la virgule comme séparateur ; `read_csv2()` utilise le point-virgule et la virgule décimale. `read_delim()` permet de préciser le séparateur. `cols(.default = col_character())` permet de commencer par une lecture en texte, puis d'effectuer des conversions choisies.

Pour exporter, utiliser par exemple `write_csv(objet, chemin_sortie, na = "")`. Choisir un fichier de sortie différent du fichier brut et vérifier que son dossier existe. Exemple autonome :

```
library(readr)
sortie <- tempfile(fileext = ".csv")
write_csv(data.frame(produit = "pomme", quantite = 4), sortie, na = "")
stopifnot(file.exists(sortie))
```

Pour vérifier la reproduction, enregistrer le script, redémarrer R dans RStudio avec Session > Restart R, puis exécuter tout le script depuis le projet. Le travail ne doit pas dépendre d'objets créés seulement dans la console ou restaurés d'une ancienne session.

2. Objets, valeurs manquantes et logique

Question	Fonctions ou opérateurs
Dimensions et noms	<code>dim()</code> , <code>nrow()</code> , <code>ncol()</code> , <code>names()</code>
Aperçu des valeurs et des types	<code>head()</code> , <code>str()</code> , <code>dplyr::glimpse()</code>
Extraire une colonne	<code>tableau\$variable</code> , <code>tableau[["variable"]]</code>
Valeurs manquantes	<code>is.na()</code> , <code>anyNA()</code>
Conversion choisie	<code>as.integer()</code> , <code>as.numeric()</code> , <code>as.character()</code>
Comparer	<code>==</code> , <code>!=</code> , <code><</code> , <code><=</code> , <code>></code> , <code>>=</code> , <code>%in%</code>
Combiner des conditions élément par élément	<code>&</code> : et ; <code> </code> : ou ; <code>!</code> : négation

Une chaîne vide "", le texte "NA" et la valeur manquante NA sont trois choses différentes. Pour tester une valeur manquante, utiliser `is.na(x)` et non `x == NA`. Certaines conversions produisent des valeurs manquantes lorsqu'un texte ne peut pas être interprété comme un nombre : inspecter le résultat et les avertissements.

```
valeurs <- c(8, NA, 12, 5)
is.na(valeurs)
```

```
[1] FALSE TRUE FALSE FALSE
```

```
valeurs >= 8
```

```
[1] TRUE NA TRUE FALSE
```

Une condition impliquant une valeur inconnue peut elle-même être inconnue. `TRUE | NA` vaut `TRUE`, mais `FALSE | NA` vaut `NA`. Décider du sens d'une valeur manquante avant de la remplacer ou d'omettre une observation.

```
library(dplyr)
texte <- c("pomme", "", NA_character_)
na_if(texte, "") # Convertit une valeur précise en NA.
```

```
[1] "pomme" NA NA
```

```
coalesce(c(NA, 8, 12), 0) # Remplacement explicite par une valeur choisie.
```

```
[1] 0 8 12
```

`na_if()` remplace les valeurs égales à une valeur donnée par NA. `coalesce()` choisit la première valeur non manquante aux positions correspondantes. Le remplacement par zéro de l'exemple est un choix de démonstration, pas une règle générale de nettoyage.

Pour compter les valeurs manquantes par colonne d'un tableau, `colSums(is.na(tableau))` fournit un résultat par colonne. Supprimer toutes les lignes incomplètes peut changer la population sur laquelle on calcule un résultat.

3. Manipuler des textes

Les fonctions de `stringr` travaillent sur un vecteur de chaînes. La fonction renvoie un résultat : l'affecter à un objet ou à une colonne pour le conserver.

Besoin	stringr	R de base
Casse	<code>str_to_upper()</code> , <code>str_to_lower()</code>	<code>toupper()</code> , <code>tolower()</code>
Espaces aux extrémités	<code>str_trim()</code>	<code>trimws()</code>
Espaces superflus, y compris internes	<code>str_squish()</code>	Combiner plusieurs opérations
Détecter un motif	<code>str_detect()</code>	<code>grep()</code>
Remplacer la première occurrence	<code>str_replace()</code>	<code>sub()</code>
Remplacer toutes les occurrences	<code>str_replace_all()</code>	<code>gsub()</code>
Extraire par position	<code>str_sub()</code>	<code>substr()</code>
Séparer en morceaux	<code>str_split()</code>	<code>strsplit()</code>
Extraire un motif	<code>str_extract()</code>	<code>regexpr()</code> et <code>regmatches()</code>

```
library(stringr)
libelles <- c(" pomme   verte ", "POIRE", NA_character_)
str_squish(libelles)
```

```
[1] "pomme verte" "POIRE"      NA
```

```
str_to_lower(libelles)
```

```
[1] " pomme   verte " "poire"      NA
```

Avec `fixed()`, un motif est recherché littéralement. Sans `fixed()`, il est généralement interprété comme une expression régulière. En R de base, l'argument correspondant est `fixed = TRUE`.

```
library(stringr)
etiquettes <- c("lot.bleu", "lotXbleu", "LOT.bleu")
str_detect(etiquettes, fixed("lot."))
```

```
[1] TRUE FALSE FALSE
```

```
str_replace_all("a-b-c", fixed("-"), " ")
```

```
[1] "a b c"
```

Pour ignorer la casse lors d'une recherche avec `stringr`, utiliser `regex(motif, ignore_case = TRUE)` ou `fixed(motif, ignore_case = TRUE)`. Avec `grep()`, utiliser `ignore.case = TRUE`.

Une position dans une chaîne n'est pas une position dans un vecteur : `x[2]` extrait le deuxième élément du vecteur ; `str_sub(x, 2, 2)` extrait le deuxième caractère de chaque chaîne. `str_split()` renvoie normalement une liste de morceaux.

4. Lire un motif simple

Les motifs ci-dessous sont écrits comme des chaînes R, avec leurs guillemets. Les repères suivent le moteur de **stringr**. Les exemples portent sur des étiquettes fictives.

Chaîne écrite en R	Signification
"^mot" / "mot\$"	Début / fin de la chaîne
"[0-9]"	Un chiffre de 0 à 9
"[A-Z]" / "[a-z]"	Une lettre dans l'intervalle indiqué
"[:space:]"	Un caractère d'espacement
"[ab]" / "[^ab]"	Un caractère parmi a et b / autre que a et b
"x?", "x*", "x+"	Zéro ou un x ; zéro ou plusieurs ; un ou plusieurs
"[0-9]{2}"	Exactement deux chiffres
"(rouge bleu)"	Une alternative regroupée
". " / "\\."	Un caractère, sauf saut de ligne par défaut / un point littéral

Une barre oblique inverse qui appartient au motif doit être doublée dans une chaîne R. Pour rechercher simplement un texte littéral, `fixed()` évite de gérer les métacaractères.

```
library(stringr)
lots <- c("lot-7", "lot-12", "lot-A", "ancien-lot-7")
str_detect(lots, "^lot-[0-9]+$")
```

```
[1] TRUE TRUE FALSE FALSE
```

```
str_extract(lots, "[0-9]+$")
```

```
[1] "7" "12" NA "7"
```

Les ancres imposent où la correspondance doit se trouver. Sans ancres, une détection peut réussir sur une partie seulement de la chaîne.

```
library(stringr)
str_detect(c("ab", "a.b", "a..b"), "^a\\.?.b$")
```

```
[1] TRUE TRUE FALSE
```

Dans ce dernier motif, le point est littéral et facultatif. Pour tester une règle, essayer au moins une chaîne qui doit être reconnue, une qui doit être rejetée et une valeur manquante. Les valeurs manquantes sont généralement propagées par les fonctions de **stringr**.

Les expressions régulières de R de base et de **stringr** n'utilisent pas le même moteur. Le comportement peut différer : par exemple, le point reconnaît un saut de ligne avec `grepl()` par défaut, mais pas avec `str_detect()`. Consulter l'aide du moteur utilisé.

5. Transformer, regrouper et calculer

Opération sur un tableau	dplyr
Créer ou modifier une colonne	<code>mutate()</code>
Garder certaines colonnes / lignes	<code>select()</code> / <code>filter()</code>
Garder les lignes distinctes	<code>distinct()</code>
Ordonner	<code>arrange()</code>
Définir des groupes	<code>group_by()</code>
Calculer un résumé	<code>summarise()</code>
Compter les lignes / valeurs distinctes	<code>n()</code> / <code>n_distinct()</code>

`distinct()` sans colonnes précisées compare les colonnes du tableau. `uplicated()` et `unique()` sont des alternatives de base pour repérer et traiter les répétitions. Une observation répétée et deux observations partageant seulement certaines valeurs ne sont pas nécessairement la même chose.

Exemple autonome de résumé par catégorie, avec des mesures fictives :

```
library(dplyr)
mesures <- tibble(
  instrument = c("A", "A", "B"),
  temperature = c(18, 20, 22)
)
mesures |>
  group_by(instrument) |>
  summarise(moyenne = mean(temperature), .groups = "drop")
```

```
# A tibble: 2 x 2
  instrument moyenne
  <chr>         <dbl>
1 A             19
2 B             22
```

`|>` transmet le résultat à la fonction suivante. En R de base, `aggregate()` permet aussi un résumé par groupe ; avec `data.table`, ce rôle est rempli par l'argument `by`.

Une proportion est un nombre de cas divisé par l'effectif de référence. La moyenne d'un indicateur sans valeur manquante est la proportion de TRUE : TRUE compte pour 1, FALSE pour 0.

```
reponses <- c(TRUE, FALSE, TRUE, NA)
sum(reponses, na.rm = TRUE) / length(reponses) # Positives observées / toutes.
```

```
[1] 0.5
```

```
mean(reponses, na.rm = TRUE) # Positives / réponses connues.
```

```
[1] 0.6666667
```

Les deux résultats, $1/2$ et $2/3$, répondent à des questions différentes. `na.rm = TRUE` ne fixe pas le dénominateur à votre place : dans une moyenne, il retire les valeurs manquantes du calcul. Un dénominateur nul ne définit pas une proportion.

6. Vérifier et documenter

Une assertion arrête le script lorsqu'une condition n'est pas vraie. Elle sert à vérifier une propriété des données ou du résultat.

```
mesures <- data.frame(
  instrument = c("A", "B", "C"),
  temperature = c(18, 20, 22)
)
stopifnot(nrow(mesures) > 0)
stopifnot(all(mesures$temperature >= -40 & mesures$temperature <= 50))
```

Si une valeur sort de la plage choisie, la deuxième assertion échoue. Cette page est propre à cet exemple fictif : une vérification doit correspondre au sens des variables étudiées.

Fonction	Lecture
<code>all(condition)</code>	Toutes les conditions sont vraies.
<code>any(condition)</code>	Au moins une condition est vraie.
<code>anyNA(x)</code>	Au moins une valeur est manquante.
<code>stopifnot(condition)</code>	Arrêter si la condition ne vaut pas entièrement TRUE.

Avec des conditions inconnues, `all()` et `any()` peuvent renvoyer NA. `stopifnot()` n'accepte pas cette incertitude. Retirer les NA peut convenir pour vérifier uniquement les valeurs observées, mais cela ne prouve pas que les valeurs requises sont présentes.

Attention aussi aux objets vides : `all(logical(0))` vaut TRUE. Vérifier séparément l'existence d'observations lorsque c'est nécessaire.

Le README est un court document texte. Il indique la provenance des données, le script à lancer et les packages, les décisions prises, les vérifications effectuées et une limite. Décrire le travail réellement réalisé ; ne pas copier une conclusion générique.

Avant la fin du temps de travail : enregistrer les fichiers, relancer le script depuis une session propre et ouvrir les résultats produits. Les consignes du défi déterminent les fichiers à remettre et les vérifications exigées.

Sources et usage

Fiche préparée pour le cours le 22 septembre 2026. Exemples fictifs, indépendants du jeu de données évalué. Les 19 PDF de la galerie du cours et les extraits de Sophie Baillargeon sont accessibles depuis [ACCUEIL.html](#) dans le même dossier de références. L'aide locale complète les rappels : `help("stopifnot")`, `help("summarise", package = "dplyr")` et `help("str_detect", package = "stringr")`.

Références : notes STT-4230 / STT-6230 de Sophie Baillargeon (2021) ; documentation officielle de R, readr, dplyr et stringr. Adresses et attributions détaillées dans [SOURCES.html](#). Les liens vers Internet de ces documents servent de références bibliographiques ; ils ne sont pas nécessaires à la consultation locale.