Création de rapports paramétrés avec {R Markdown}

Introduction

- **Importance de la paramétrisation des rapports :**
  - Représente une tâche cruciale dans le rapportage épidémiologique.
  - Offre une flexibilité pour générer dynamiquement du contenu basé sur des paramètres spécifiques.
- **Polyvalence et application inestimable :**
  - Inestimable dans des scénarios nécessitant la génération de rapports en fonction de divers facteurs (pays, périodes, incidences de maladies).
  - Permet l'utilisation d'un modèle unique et polyvalent pour produire une multitude de rapports.
- **Consistance, précision et efficacité :**
  - Assure la cohérence et la précision dans le rapportage des données épidémiologiques.
  - Améliore considérablement l'efficacité de la communication des données épidémiologiques.
- **Amélioration de la communication des données :**
  - Permet de présenter des données complexes dans un format plus accessible et interprétable.
  - Aide à la diffusion efficace d'informations sanitaires critiques.

Objectifs d'apprentissage

1. **Comprendre l'importance de la paramétrisation dans R Markdown** : Comprendre le concept fondamental de la paramétrisation, son importance dans le rapportage épidémiologique, et comment elle transforme la manière dont les données sont présentées et analysées.

2. **Apprendre à créer des rapports dynamiques basés sur des paramètres spécifiés** : Développer la compétence pour rédiger des rapports dans R Markdown qui ajustent automatiquement leur contenu en fonction des paramètres donnés, tels que les régions géographiques ou les cadres temporels.

3. **Développer des compétences dans la création de fonctions pour la paramétrisation de rapports** : Acquérir une maîtrise dans l'écriture de fonctions en R qui permettent la paramétrisation de rapports, simplifiant ainsi le processus de génération de rapports.

4. **Explorer la programmation fonctionnelle avec `map()` et `pwalk()`** : Se plonger dans les aspects de la programmation fonctionnelle de R, en se concentrant spécifiquement sur l'utilisation des fonctions `map()` et `pwalk()` du package {purrr}. Comprendre comment ces fonctions peuvent être utilisées pour gérer efficacement de multiples ensembles d'entrées de données pour la génération de rapports.

En atteignant ces objectifs, vous serez bien équipé pour gérer des ensembles de données épidémiologiques et les présenter de manière informative, organisée et percutante. La capacité à paramétrer des rapports simplifie non seulement le processus d'analyse des données, mais augmente également considérablement la polyvalence et l'applicabilité des résultats présentés.

Packages

Ce fragment de code ci-dessous montre comment charger les paquets nécessaires en utilisant `p_load()` du package {pacman}. S'il n'est pas installé, il installe le paquet avant de le charger.

```
pacman::p_load(readr, ggplot2, dplyr, knitr, purrr, gt, kableExtra)
```

```
## package 'kableExtra' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
```

```
##   C:\Users\Sabina
Rodriguez\AppData\Local\Temp\RtmpE54HPD\downloaded_packages
```

Jeu de données

```
hiv_data <- read_csv(here::here("data/clean/hiv_incidence.csv"))
```

Le jeu de données `hiv_incidence.csv` contient trois colonnes :

1. `country` : Le nom du pays.
2. `year` : L'année de l'enregistrement.
3. `new_cases` : Le nombre de nouveaux cas de VIH signalés dans ce pays pour l'année donnée.

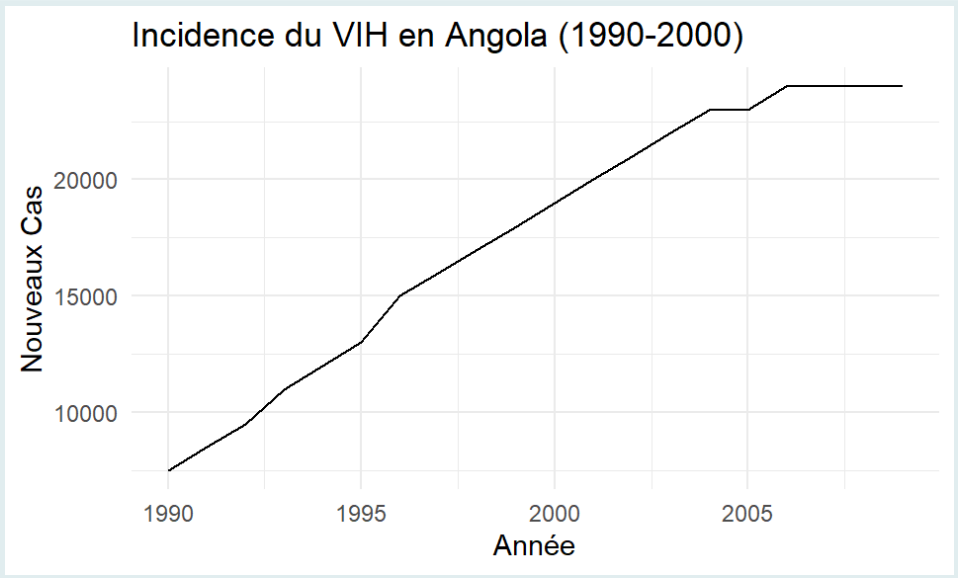
Avec cette compréhension, procédons à la création des rapports comme décrit précédemment. Nous adapterons les étapes à ce jeu de données spécifique. Le tutoriel démontrera comment construire des rapports se concentrant sur les données de prévalence du VIH pour différents pays au cours de diverses années.

~~Construire~~ un seul rapport pour un pays (par exemple, "Angola")

1. **Concentrez-vous sur la création de rapports visuels :**
   - Créer un rapport visuel pour l'Angola en utilisant le langage R.
2. **Analyser et visualiser les tendances :**
   - L'objectif est d'analyser et de visualiser les nouveaux cas de VIH en Angola.
3. **Analyse temporelle sur plusieurs années :**
   - Se concentrer sur la tendance des nouveaux cas de VIH sur une série d'années.

Sortie 1 : Graphique en ligne pour "Angola"

```
angola_data <- subset(hiv_data, country == "Angola")
ggplot(angola_data, aes(x = year, y = new_cases)) +
  geom_line() +
  theme_minimal() +
  labs(title = "Incidence du VIH en Angola (1990-2000)",
       x = "Année",
       y = "Nouveaux Cas")
```



Sortie 2 : Tableau Statistique pour “Angola”

Ce fragment de code ci-dessous est conçu pour générer un tableau récapitulatif pour l’Angola, en se concentrant sur les cas de VIH par année. Le tableau mettra en évidence les données de l’année la plus récente pour une identification plus facile. Décomposons et expliquons chaque partie du code.

```
# Filtrer pour les données de l'Angola
angola_data <- hiv_data %>% filter(country == "Angola")

# Résumer les données par année
angola_summary <- angola_data %>%
    group_by(year) %>%
    summarise(Total_Cases = sum(new_cases))

# Identifier l'année la plus récente
most_recent_year <- max(angola_summary$year)

# Afficher le tableau en utilisant kable et mettre en évidence l'année la plus
        récente
angola_summary %>%
    kable("html", caption = "Résumé des cas de VIH en Angola par année") %>%
    kable_styling(bootstrap_options = c("striped", "hover")) %>%
    row_spec(which(angola_summary$year == most_recent_year), background =
        "lightblue")
```

```
## Error in row_spec(., which(angola_summary$year == most_recent_year),
background = "lightblue"): could not find function "row_spec"
```

```r
angola_summary %>%
  gt() %>%
  tab_style(
    style = cell_fill(color = "lightblue"),
    locations = cells_body(
      columns = vars(year),
      rows = angola_summary$year == most_recent_year
    )
  ) %>%
  tab_header(title = "Résumé des cas de VIH en Angola par année")
```

Résumé des cas de VIH en Angola par année	
year	Total_Cases
1990	7500
1991	8500
1992	9500
1993	11000
1994	12000
1995	13000
1996	15000
1997	16000
1998	17000
1999	18000
2000	19000
2001	20000
2002	21000
2003	22000
2004	23000
2005	23000
2006	24000
2007	24000
2008	24000
2009	24000

Construire un rapport simple en utilisant un fichier R markdown

Maintenant que vous avez deux sorties bien préparées – un graphique en ligne et un tableau statistique pour les données sur le VIH en Angola – vous pouvez les compiler dans un rapport simple et cohérent en utilisant R Markdown. Ce rapport mettra non seulement en valeur vos compétences analytiques, mais aussi votre capacité à communiquer efficacement les résultats.

Étapes à suivre :

1. Créer un nouveau fichier RMarkdown
2. Rédiger l'en-tête YAML
3. Ajouter une section de préparation des données
4. Ajouter un graphique en ligne et un tableau statistique
5. Tricoter le fichier RMarkdown

Construire un rapport pour plusieurs pays (par exemple, "Angola", "Nigeria", "Mali")

En plus de l'Angola, nous souhaitons créer le même rapport pour le Nigeria et le Mali. Ce défi vous demande d'étendre l'approche analytique utilisée pour l'Angola à ces pays supplémentaires.

Y a-t-il une méthode plus rationalisée pour faire cela sans dupliquer nos lignes de code ?

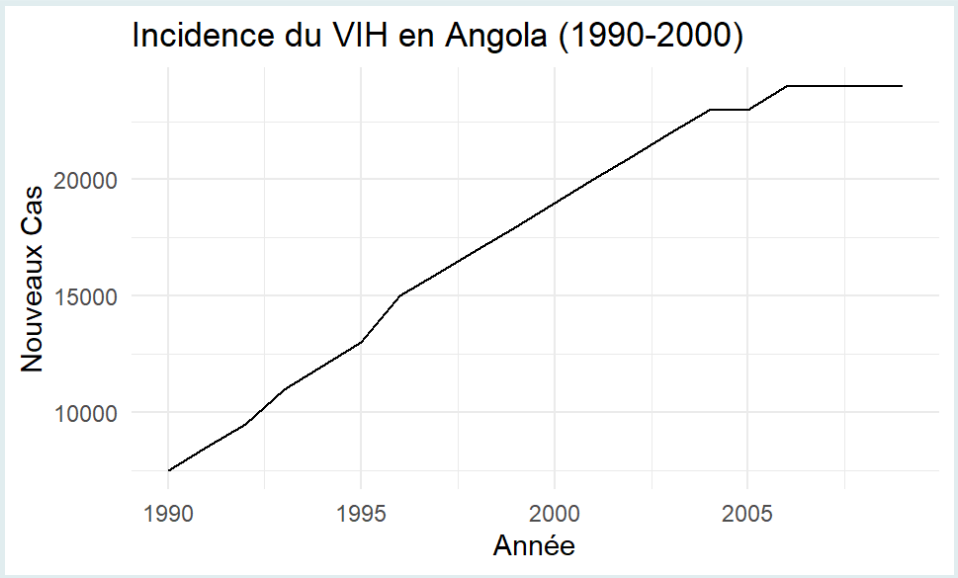
Principe DRY : Ne vous répétez pas !

Approche 1 : Produire une sortie grâce à une fonction personnalisée

- La fonction `generate_country_report()` est créée pour automatiser la génération de rapports pour un pays spécifié à l'aide d'un ensemble de données.
- Elle incarne le principe DRY (Don't Repeat Yourself).
- Cela est réalisé en centralisant les tâches répétitives au sein d'une seule fonction, évitant ainsi la redondance.

- **DRY (Don’t Repeat Yourself)** : Un principe fondamental de développement logiciel axé sur la minimisation de la répétition.
- Encourage le remplacement des schémas logiciels répétés par des abstractions ou une normalisation des données pour éliminer la redondance.
- Prône l’utilisation de fonctions ou de méthodes pour remplacer les blocs de code répétitifs.
- Le principe DRY est largement considéré comme essentiel dans de nombreux langages de programmation et méthodologies de conception.

```
generate_country_report <- function(data, country_name) {

  # Tracer les données
  country_data_plot <- subset(data, country == country_name)
  p <- ggplot(country_data_plot, aes(x = year, y = new_cases)) +
    geom_line() +
    theme_minimal() +
    labs(title = paste("Incidence du VIH en", country_name, "(1990-2000)"),
         x = "Année",
         y = "Nouveaux Cas")
  print(p)

  # Créer le tableau récapitulatif
  country_data_table <- data %>%
    filter(country == country_name) %>%
    group_by(year) %>%
    summarise(Total_Cases = sum(new_cases))

  # Identifier l'année la plus récente
  most_recent_year <- max(country_data_table$year)

  # Afficher le tableau en utilisant kable et mettre en évidence l'année la
        plus récente
  table_output <- country_data_table %>%
    gt() %>%
    tab_style(
      style = cell_fill(color = "lightblue"),
      locations = cells_body(
        columns = vars(year),
        rows = angola_summary$year == most_recent_year
      )
    ) %>%
    tab_header(title = "Résumé des cas de VIH en Angola par année")

  print(table_output)
}

# Exemple d'utilisation de la fonction
generate_country_report(hiv_data, "Angola")
```



Expliquons ce code :

1. **Déclaration de la fonction** : La fonction `generate_country_report()` prend deux paramètres : `data` (l'ensemble de données) et `country_name` (le nom du pays pour lequel le rapport doit être généré).

```
generate_country_report <- function(data, country_name) {
  ......
}
```

2. **Tracé des Données** : La fonction commence par isoler les données pour le pays spécifié. Elle crée ensuite un graphique en ligne des nouveaux cas de VIH au fil des années en utilisant {ggplot}.

```
# Tracé des données
  country_data_plot <- subset(data, country == country_name)
  p <- ggplot(country_data_plot, aes(x = year, y = new_cases)) +
    geom_line() +
    theme_minimal() +
    labs(title = paste("Incidence du VIH en", country_name, "(1990-2000)"),
         x = "Année",
         y = "Nouveaux Cas")
  print(p)
```

3. **Création de Tableau Récapitulatif** : Ensuite, elle filtre les données pour le pays spécifié, les regroupe par année et calcule le nombre total de nouveaux cas pour chaque année.

```r
# Création du tableau récapitulatif
country_data_table <- data %>%
  filter(country == country_name) %>%
  group_by(year) %>%
  summarise(Total_Cases = sum(new_cases))
```

4. **Mise en Évidence des Données Récentes** : La fonction identifie l'année la plus récente dans les données et la met en évidence dans le tableau récapitulatif.

```r
# Identifier l'année la plus récente
most_recent_year <- max(country_data_table$year)
```

5. **Affichage du Tableau** : Le tableau récapitulatif est affiché en utilisant `kable()` et `kable_styling()` pour une sortie HTML propre et interactive. La ligne correspondant à l'année la plus récente est mise en évidence.

```r
# Afficher le tableau en utilisant kable et mettre en évidence l'année la
      plus récente
table_output <- country_data_table %>%
  gt() %>%
  tab_style(
    style = cell_fill(color = "lightblue"),
    locations = cells_body(
      columns = vars(year),
      rows = angola_summary$year == most_recent_year
    )
  ) %>%
  tab_header(title = "Résumé des cas de VIH en Angola par année")
```

6. **Impression des Sorties** : Le graphique (`p`) et le tableau (`table_output`) sont imprimés dans la sortie.

```r
print(table_output)
```

- **Programmation Fonctionnelle Utilisant {purr}** :
  - Étend le processus de génération de rapports pour gérer plusieurs pays.
  - Utilise les paradigmes de programmation fonctionnelle en R, en particulier avec le package {purr}.
  - Facilite l'itération sur une liste de pays.
  - Applique la fonction `generate_country_report()` à chaque pays de la liste.
  - Rationalise le processus tout en respectant le principe DRY.

Cette approche basée sur les fonctions rend le code plus organisé et lisible et améliore également sa réutilisabilité et sa scalabilité, des aspects cruciaux des pratiques de programmation efficaces.

Utilisation de `map()`

- Vous avez une liste de pays pour lesquels des rapports doivent être générés.
- Utilisez la fonction `map()` du package {purrr} comme outil pour la tâche.
- La fonction `map()` permet d'appliquer la fonction `generate_country_report()` à chaque pays de la liste.
- Un exemple est fourni pour illustrer l'utilisation de `map()` en conjonction avec `generate_country_report()`.

```
library(purrr)
# Vecteur de pays
countries <- c("Angola", "Nigeria", "Mali")

# En supposant que 'hiv_data' soit votre jeu de données
# Appliquer 'generate_country_report' à chaque pays
map(countries, ~generate_country_report(hiv_data, .))
```

Explication

- `countries` est une liste de noms de pays pour la génération de rapports.
- `map()` applique `generate_country_report()` à chaque pays, en utilisant `~` pour créer une formule dans laquelle `.` est le pays actuel.
- Les rapports pour tous les pays de `countries` sont générés en appelant `generate_country_report(hiv_data, .)`.

Cette méthode rationalise la création de rapports cohérents pour plusieurs pays,

économisant du temps grâce à l'automatisation.

Construction d'un Rapport Paramétrisé

Configuration de l'En-tête YAML

- **Définition des Paramètres** : Les paramètres sont définis dans l'en-tête YAML. C'est la section au début de votre fichier R Markdown où vous spécifiez divers paramètres pour votre document, tels que son titre, son format de sortie et ses paramètres.

Exemple avec un Jeu de Données sur le VIH

Considérez un fichier R Markdown conçu pour générer un rapport sur les données du VIH pour différents pays. Vous pouvez définir un paramètre dans l'en-tête YAML pour changer dynamiquement le pays pour lequel le rapport est généré.

YAML dans le fichier R Markdown :

```
---
title: "Rapport VIH par Pays"
output: html_document
params:
  country: "Angola"
---
```

Explication

1. **Titre du Document et Sortie** : Dans l'en-tête YAML, le document est intitulé "Rapport VIH par Pays", et le format de sortie est spécifié comme un document HTML.

2. **Définition du Paramètre** :
   - `params` : Ce champ est utilisé pour définir des paramètres.
   - `country` : Sous `params`, un paramètre nommé `country` est déclaré. Ce paramètre contrôlera les données du pays utilisées dans le rapport.
   - `"Angola"` : La valeur par défaut pour le paramètre `country` est définie à "Angola". Cela signifie que lorsque le rapport est généré sans spécifier un autre pays, il utilisera automatiquement les données pour l'Angola.

Utilisation du Paramètre dans le Document

Dans le document R Markdown, vous pouvez référencer ce paramètre en utilisant `params$country`. Par exemple, si vous avez une fonction comme `generate_country_report()`, vous pouvez l'appeler dans un bloc de code comme `generate_country_report(data, params$country)`. Cela utilisera la valeur du paramètre `country` pour déterminer les données du pays sur lesquelles rapporter.

En utilisant des paramètres, votre document R Markdown devient plus adaptable. Si vous devez générer un rapport pour un autre pays, vous changez simplement la valeur du paramètre lors de la tricotage du document, plutôt que de modifier le code lui-même. Cela rend votre document plus efficace et polyvalent, en particulier lorsqu'il s'agit de rapports nécessitant des mises à jour fréquentes ou des modifications basées sur différents sous-ensembles de données.

Mise en Œuvre des Paramètres dans l'Analyse

- **Filtrage des Données Basé sur les Paramètres :** Utilisez les paramètres pour filtrer dynamiquement le jeu de données sur la tuberculose (TB). Par exemple, si le rapport concerne un pays et une année spécifiques, vous pouvez filtrer les données en conséquence en utilisant `dplyr`.

Exemple de fragment de code

```
filtered_data <- tb_data %>%
                  filter(country == params$country, year == params$year)
```

Explication de la Cartographie des Paramètres

- **Accès aux Paramètres** : Les paramètres définis dans l'en-tête YAML de votre document R Markdown peuvent être accessibles dans le code en utilisant `params$nom_du_paramètre`. Cela est similaire à la manière dont vous accédez aux colonnes dans un dataframe en utilisant `dataframe$nom_de_la_colonne`.
- **Rapport Dynamique** : Lorsque vous créez des visualisations ou des résumés, vous utilisez `params$country` et `params$year` pour garantir que la sortie reflète le pays et l'année spécifiques définis en tant que paramètres. Cela signifie que le contenu du rapport s'adaptera en fonction des valeurs fournies pour ces paramètres au moment du tricotage du document.

Dans le contexte de R Markdown avec des paramètres, `params$nom_du_paramètre` accède à la valeur d'un paramètre nommé `nom_du_paramètre` qui a été défini dans l'en-tête YAML. Cela permet de générer des rapports dynamiques et interactifs basés sur ces valeurs de paramètres.

Dans l'ensemble, l'opérateur `$` est une partie clé de la syntaxe R qui facilite la manipulation des éléments dans les objets R.

Tricotage avec Différents Paramètres

- **Tricotage depuis RStudio :** Expliquez comment tricoter le document avec différentes valeurs de paramètres directement à partir du bouton de tricotage de RStudio, où vous pouvez spécifier les valeurs des paramètres dans une boîte de dialogue.
- **Tricotage en Ligne de Commande :** Pour une génération de rapport automatisée, vous pouvez utiliser des commandes R pour tricoter le document avec différents paramètres. Cela est particulièrement utile pour le traitement par lots ou la génération de rapport automatisée.

- Exemple de commande :

```
rmarkdown::render("TB_Report.Rmd", params = list(country = "Angola"))
```

Tricotage avec Différents Paramètres

La création de rapports dynamiques et flexibles dans R Markdown implique souvent de tricoter le document avec des paramètres variables. Il existe deux principales manières de le faire : directement via l'interface de RStudio et via la ligne de commande pour l'automatisation.

Tricotage depuis RStudio

- **Utilisation du Bouton de Tricotage de RStudio** : Lorsque vous tricotez un document R Markdown dans RStudio qui contient des paramètres, une boîte de dialogue apparaîtra. Cette boîte de dialogue vous permet de spécifier les valeurs pour chaque paramètre avant de tricoter.
- **Sélection Interactive des Paramètres** : Cette méthode est conviviale et interactive, idéale pour la génération de rapport ad hoc où vous pouvez manuellement entrer ou modifier les valeurs des paramètres chaque fois que vous tricotez le document.
- **Personnalisation des Paramètres** : Dans la boîte de dialogue, vous pouvez personnaliser les paramètres selon vos besoins. Par exemple, si vous avez un paramètre pour le pays, vous pouvez changer sa valeur pour générer un rapport pour un autre pays.

Tricotage en Ligne de Commande

- **Génération de Rapport Automatisée** : Pour des scénarios où vous devez automatiser le processus de génération de rapport, comme dans le traitement par lots ou les rapports planifiés, vous pouvez tricoter des documents en utilisant des commandes R.
- **Exemple de Commande** :

```
rmarkdown::render("hiv_incidence.Rmd", params = list(country = "Angola"))
```

Cette commande utilise la fonction `render` du package `rmarkdown`. Elle spécifie le fichier à rendre (`"hiv_incidence.Rmd"`) et définit l'argument `params` à une liste de valeurs de paramètres souhaitées. Dans cet exemple, le paramètre pays est défini sur "Angola".

- **Flexibilité et Scripting** : Le tricotage en ligne de commande offre plus de flexibilité et peut être intégré dans des scripts pour des flux de travail automatisés. Cela est particulièrement utile lorsqu'on travaille avec de grands

ensembles de données ou qu'on doit générer plusieurs rapports périodiquement.

Les deux méthodes offrent les moyens d'exploiter la puissance des rapports paramétrés dans R Markdown, répondant à différents besoins — de la génération de rapport interactive et pilotée par l'utilisateur à des processus automatisés et dirigés par des scripts. Cette flexibilité est l'une des forces clés de l'utilisation de R Markdown pour l'analyse de données et la création de rapports.

Le Processus Complet

Créer un rapport paramétré en R implique plusieurs étapes clés, de la configuration de votre projet R à l'automatisation du processus de génération de rapport. Passons en revue ces étapes en détail :

Étape 0 : Créer un Projet R

- **Fichier .Rproj** : Commencez par créer un projet R avec un fichier `.Rproj`. Ce fichier aide à gérer les chemins et les paramètres, rendant plus facile de travailler avec des fichiers et des données dans le projet.

Étape 1 : En-tête YAML

Voici un exemple d'en-tête YAML pour votre document R Markdown :

```
---
title: "Rapport - `r stringr::str_to_title(params$state)`"
date: "`r format(Sys.time(), '%d %B %Y')`"
output: html_document
editor_options:
  chunk_output_type: console
params:
  state: "Alabama"
---
```

Cet en-tête configure le titre du document, la date, le format de sortie et définit un paramètre `state` avec une valeur par défaut de "Alabama".

Étape 2 : Configuration de l'Environnement du Document

Dans le premier bloc de code, configurez votre environnement en chargeant les packages requis et votre jeu de données :

```
# Configuration des options globales
knitr::opts_chunk$set(warning = FALSE, message = FALSE, echo = FALSE)

# Chargement des packages
pacman::p_load(ggplot2, dplyr, knitr, kableExtra)

# Chargement du jeu de données
hiv_data <- read.csv("hiv_incidence.csv")
```

Étape 3 : Le Cœur de Votre Rapport

Maintenant, ajoutez des sections avec des sorties, générées dynamiquement en fonction de l'état sélectionné.

Tendance de l'Incidence du VIH

```
hiv_incidence_data <- subset(hiv_data, country == params$country)
ggplot(hiv_incidence_data, aes(x = year, y = new_cases)) +
  geom_line() +
  theme_minimal() +
  labs(title = paste("Incidence du VIH en", params$country, "(1990-2000)"),
       x = "Année",
       y = "Nouveaux Cas")
```

Tableau Récapitulatif des Cas de VIH

```
## Tableau Récapitulatif des Cas de VIH

# Filtrer pour les données du pays sélectionné
country_data <- hiv_data %>% filter(country == params$country)

# Résumer les données par année
country_summary <- country_data %>%
    group_by(year) %>%
    summarise(Total_Cases = sum(new_cases))

# Identifier l'année la plus récente
most_recent_year <- max(country_summary$year)

# Afficher le tableau en utilisant kable et mettre en é

vidence l'année la plus récente
country_summary %>%
    kable("html", caption = paste("Résumé des cas de VIH en", params$country,
        "par année")) %>%
    kable_styling(bootstrap_options = c("striped", "hover")) %>%
    row_spec(which(country_summary$year == most_recent_year), background =
        "lightblue")
```

Étape 4 : Tricoter le Rapport

Le document peut maintenant être tricoté pour générer un rapport pour l'état par défaut ou spécifié.

Créez un nouveau R Markdown avec ces composants et tricotez-le pour voir la sortie.

Étape 5 : Créer un Script R pour Paramétrer Tout le Processus

Pour l'automatisation, utilisez un script R pour paramétrer le processus :

```r
# Chargement des packages nécessaires
pacman::p_load(rmarkdown, purrr, stringr)

# Définir un vecteur d'états
states <- c("Alabama", "Alaska", "Arizona")

# Générer une tibble pour les rapports
reports <- tibble(
  filename = str_c("state_report_", states, ".html"),
  params = map(states, ~list(state = .))
)

# Utiliser pwalk pour rendre chaque rapport
reports %>%
  select(output_file = filename, params) %>%
  pwalk(rmarkdown::render, input = "state_report.Rmd", output_dir = "output/")
```

Ce script utilise `map()` et `pwalk()` pour itérer sur chaque état, rendant un rapport individualisé pour chacun.

La fonction `map()` crée une liste de paramètres pour chaque état, et `pwalk()` applique la fonction `rmarkdown::render()` à chaque ensemble de paramètres, générant les rapports.

Avec ces étapes, vous pouvez efficacement générer des rapports personnalisés pour plusieurs états ou critères, rationalisant votre flux de travail d'analyse de données et de création de rapports en R.

CONCLUSION !

Pour conclure, nous avons exploré un flux de travail complet pour créer des rapports paramétrés en utilisant R Markdown. Ce processus permet de générer des rapports

dynamiques et efficaces adaptés à des critères spécifiques, tels que des régions géographiques ou des cadres temporels.

Résultats d'apprentissage

À la fin de cette leçon, vous devez être capable de :

- **Gérer Efficacement les Projets R** : Organiser et gérer des projets R pour diverses tâches d'analyse de données.
- **Produire des Documents R Markdown Dynamiques** : Générer des documents R Markdown qui s'adaptent à différents entrées de données.
- **Réaliser des Analyses de Données et des Visualisations** : Analyser des données et créer des visualisations, en se concentrant sur des jeux de données réels comme l'incidence du VIH.
- **Développer des Scripts d'Automatisation pour les Rapports** : Créer des scripts pour automatiser la génération de rapports basés sur des paramètres, améliorant la cohérence et l'efficacité.
- **Adopter les Meilleures Pratiques en Programmation R** : Mettre en œuvre les meilleures pratiques pour une programmation R plus efficace, lisible et maintenable.
- **Améliorer les Compétences de Création de Rapport** : Améliorer les capacités à élaborer des rapports complets, informatifs et visuellement attrayants pour divers publics.
- **Utiliser des Techniques de Programmation Fonctionnelle** : Appliquer des méthodes de programmation fonctionnelle en R pour un traitement de données et une création de rapports rationalisés.

Bon rendu !

Contributeurs

Les membres de l'équipe suivants ont contribué à cette leçon :

IMAD EL BADISY
Data Science Education Officer
Deeply interested in health data

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement

JOY VAZ

R Developer and Instructor, the GRAPH Network
Loves doing science and teaching science

Références

- Johnson, Paul. “R Markdown: Le Guide Définitif : Yihui Xie, JJ Allaire, et Garrett Grolemund. Boca Raton, FL : Chapman & Hall/CRC Press, 2018, xxxiv+ 303 pp., $38.95 (P), ISBN : 978-1-13-835933-8