



|  |  |  |
| --- | --- | --- |
|  | REPUBLIQUE DU SENEGAL  \*\*\*\*\*\*\*  Un peuple-Un but-Une foi  \*\*\*\*\*\* |  |

Cours de R

ECOLE NATIONALE DE LA STATISTIQUE ET DE L’ANALYSE ECONOMIQUE

(ENSAE-Pierre NDIAYE)

\*\*\*\*\*\*

RESUME DES EXPOSES DE R

**Réalisé par :**

1. Fatou DIOP (ISE-Math)
2. Laurine ADOGOUN (ISE-Eco)
3. Daniel KPEKPASSI (ISEP3)

**Sous la direction de :**

Mr Aboubacar HEMA

Research analyst

Table of Contents

[I. Présentation 1 : Janitor 1](#_Toc139490124)

[II. Presentation 2 : Gtsummary 2](#_Toc139490125)

[III. Présentation 3 : RMarkdown 4](#_Toc139490126)

[IV. Presentation 4 : R-Quarto 5](#_Toc139490127)

[V. Présentation 5 : R vers Excel 8](#_Toc139490128)

[VI. Présentation 6 : Text Mining 10](#_Toc139490129)

[VII. Présentation 7 : Systèmes d’équations non linéaires sur R 11](#_Toc139490130)

[VIII. Présentation 8 : Python dans R 12](#_Toc139490131)

[IX. Présentation 9 : Calcul parallèle avec R 13](#_Toc139490132)

[X. Presentation 10 : Cartographie sur R 15](#_Toc139490133)

[XI. Présentation 11 : R-Shiny 17](#_Toc139490134)

# 

# Présentation 1 : Janitor

## Utilité

Le package janitor permet de nettoyer et explorer la base de données. Il offre principalement des options de formatage des données et de gestion des doublons

## Packages

Avant de l’utiliser, il est essentiel d’installer le package **janitor** ainsi que d’autres packages nécessaires pour une manipulation efficace des données tels que **tidyverse, dplyr, gt, knitr, tinytex, readxl, flextable**

## Fonctions principales

La fonction **clean\_names()** de Janitor : permet de nettoyer les noms de colonnes dans un ensemble de données en remplaçant les espaces et les caractères spéciaux par des underscores (tiret du huit). Elle peut également corriger les noms de colonnes en double en ajoutant des suffixes et effectuer une modification générale des noms de variables en prenant en compte des cas spécifiques tels que la casse des lettres, les noms de variables vides et les espaces dans les noms. On a aussi la fonction **make\_clean\_names()** qui est une variante de clean\_names() qui permet une utilisation plus large.

La fonction **compare\_df\_cols() :**  facilite l’identification des différences entre les colonnes de deux dataframes, ce qui est utile notamment lors de la fusion de données.

La fonction **remove\_empty()** de Janitor permet de supprimer les colonnes vides d’une base de données , contribuant ainsi à nettoyer les données. De plus, il existe d’autres fonctions telles que **remove\_constant()** qui supprime les colonnes ne contenant qu’une seule valeur constante, et **get\_dupes()** qui permet de d’identifier les enregistrements en doublon (lignes, par rapport à des variables au besoin).

Le package Janitor nous propose plusieurs fonctions pour réaliser des tableaux (la fonctions **tabyl()** ) et les formater avec les fonctions **adorn\_\*()** ( *adorn\_pct\_formatting()*, *adorn\_ns()*, *adorn\_title()*,…)

## Limites

Il n'est pas possible de sortir des tableaux de plus de deux variables ni de combiner plusieurs tableaux.

# Présentation 2 : Gtsummary

## Utilité

Gtsummary fait référence à un package conçu pour les études d’impacts, l’analyse des différences entre 2 groupes (il permet cependant de comparer d’une certaine manière plusieurs groupes). Ce package permet de produire des tableaux statistiques de résumé d’une base de données ; de combiner plusieurs tableaux de résumé ; de produire des tableaux près à la présentation, à l’inclusion dans des rapports et même à la publication.

## Packages

***gtsummary*** : fourni toutes les fonctions nécessaires à la production des tableaux statistiques de résumé

***dplyr*** : fourni l’essentiel des fonctions de traitement des bases de données

***gt, Kabble, flextable…*** : utilisé surtout pour formater les sorties

## Quelques fonctions à retenir

Il sera exposé dans cette partie quelques fonctions de base, très utiles liées au package gtsummary

* **tbl\_summary()** : il s’agit de la fonction de base qui permet de faire des tables de résumé. Elle est hautement personnalisable de manière à obtenir un tableau comme que nous désirons. Voici quelques uns de ses paramètres :
  + by = : on y défini la variable qui identifie les groupes à comparer
  + include = : permet de définir les variables à prendre en compte dans le tableau
  + label = : on y explicite l’affichage des variables dans le tableau.
  + percent = : détermine le type de pourcentage affiché dans le tableau (ligne, colonne ou proportion sur le total)
  + digits = : sert à définir l’affichage des chiffres dans le tableau (proportion, pourcentage, nombre de chiffres après la virgule, unité…). On peut le définir par variable ou par groupe de variables
  + statistic = : définit le type de statistiques calculées (par variable ou par groupe de variables)
  + type = : précise le type des variables pour éviter de travailler avec la considération par défaut de R
  + missing = ; missing\_text = : pour gérer l’affichage des valeurs manquantes dans le tableau.
* **tbl\_cross()** : permet de faire les tableaux croisés
* **add\_difference()** : uniquement lorsqu’il y a 2 groupes. Ajoute une colonne de différence entre les moyennes, une pour le test de significativité de cette différence et une pour l’intervalle de confiance.
* **add\_overall()** : affiche les statistiques (voules) relatives à l’ensemble des données.
* **tbl\_continuous(), tbl\_continuous\_summary()** : sert à présenter les statistiques d’une variable continue par rapport à chaque groupe et à chaque variable qualitative incluse dans le tableau.
* **tbl\_regression()** : sert à faire le résumé (sous format tableau) d’un modèle implémenté.
* **tbl\_stack(), tbl\_merge()** : permettent d’aggréger plusieurs tableaux. La première les empile l’un au-dessus de l’autre et la seconde l’un à côté de l’autre en s’assurant que les mêmes variables soient sur la même ligne
* **tbl\_strata()** : permet de faire un tableau gtsummary pour chaque modalité d’une variable ou d’un vecteur de variable catégorielle définie via le paramètre strata =, puis de combiner les tableaux entre eux via le paramètre .combine\_with = soit par tbl\_stack soit par tbl\_merge
* **tbl\_spit()** : permet de couper un géant tableau en plusieurs parties
* **theme\_gtsummary\_\*()** : un ensemble de fonction qui permettent de définir au préalable les paramètres de sortie des tableaux.
* **tbl\_svysummary()** : pour les bases de type *survey*

**NB :** Pour en savoir plus sur ce package, veuillez consulter ces sites : [site\_fonction](https://larmarange.github.io/analyse-R/gtsummary.html#type_variables) et [site\_de\_base](https://www.danieldsjoberg.com/gtsummary/articles/tbl_summary.html)

## Limites

Les limites de ce package résident essentiellement vers les sorties. En effet, les sorties sont très peu personnalisables au sens des mises en forme telles que les couleurs… pour montrer visuellement les résultats critiques.

# Présentation 3 : RMarkdown

## Utilité

Rmarkdown est une extension de R qui peut être utilisé lorsque l'on souhaite produire avec R des documents contenant à la fois du texte, des extraits de code R et les résultats de l'exécution de programmes R (et même d’autre langages) qui se présente sous forme de package. Les documents Rmarkdown sont entièrement ***reproductibles*** et prennent en charge beaucoup de formats de sortie, tels que les fichiers PDF, Word, html, beamer (pour les présentations)…etc.

## Que doit-on retenir ?

Les documents Rmarkdown ont pour extension .Rmd. Ils comportent généralement trois parties :

* Le **YAML** : il représente l'entête du document et est délimité par ---. Il contient les métadonnées (type du document, titre, auteur …). On y définie aussi la table des matières toc : yes. Notons qu'il est important de respecter l'indentation au niveau des meta-données.
* La **partie texte** : on y écrit du texte normal. On peut également y inclure des codes html, latex ; des images ; des liens hypertextes ; des tableaux (*à dessiner manuellement*) et aussi des codes R (entre ` `) pour rendre le document reproductible. Par exemple automatiser des changements numériques.
* **Chunk** : sert à écrire des codes de plusieurs langages tels python, R, SQL… Dans le cas de R, le Chunk se comporte comme R-script. Il existe un chunk où on définit les paramètres généraux de tous les autres chunks du document : il s'agit du ***tout premier chunk du ficher .Rmd, le Chunk directeur*** !

Pour améliorer les sorties du document, on peut jouer sur le paramétrage des chunks :

* On peut choisir d'afficher les résultats des chunks ou non
* Afficher les codes contenus dans les chunks ou non
* Afficher les avertissements (warnings=FALSE) ou non…

**NB :** **On peut appliquer des mis à formes à un document sur la base d'un document de référence !**

# Presentation 4 : R-Quarto

## Utilité

R-Quarto fait référence à un package R mais aussi à une extension R développée depuis peu mais qui aujourd’hui est multi-tâches. En effet, elle est aujourd’hui utilisée pour écrire des documents (comme RMarkdown) ; développer de site web, des blogs, des livres, des présentations dynamiques et même pour écrire des articles avec R.

Quarto aspire aussi à une certaine indépendance. Il possède une application propre à lui suffisamment similaire à RStudio.

## Packages

On peut citer le package **Quarto** qui n’est pas vraiment indispensable.

## Quelques fonctions de quarto à retenir

Il sera développé dans cette partie quelques fonctions relatives aux documents et d’autres sorties proposées par R-Quarto.

### Documents Quarto

Ils se présentent comme du Rmarkdown évolué et simplifié. Le document Quarto se distingue par son extension .qmd. En général, il donne accès à 3 sorties de documents (pdf, word et html). Le fichier Quarto, sur R, à 2 interfaces :

* **la partie Source :** c’est du Rmarkdown pure mais surtout la retranscription en format *.Rmd* de tout ce qui se fait dans la seconde partie.
* **la partie Visual:** c’est l’interface qui se présente comme celle d’un document *Word*. Là, la majeure partie du formatage peu se faire par des cliques. Mais en plus de cela, cette interface offre la possibilité d’y écrire du code comme dans un fichier *.Rmd*.

L’interface Visual possède aussi une partie YAML. Elle nous offre de nombreuses fonctionnalités parmi lesquelles :

* ***le niveau de titre et le format du texte*** que l’on configure dans l’onglet Normal
* ***le formatage du texte*** (mise en gras, en italique, souligné, barré, exposant, indice, mode coding…) que l’on configure dans l’onglet Format
* ***l’insertion des tableaux*** par de simple clic (comparé à RMarkdown) que l’on fait à partir de l’onglet Table spécifiquement et de l’onglet Insert aussi.
* ***l’insertion d’image*** (en réglant directement les paramètres), ***de notes de bas de page*, *de sauts de page, de lien hypertexte, de citation, de code latex***… que l’on fait à partir de l’onglet Insert principalement.
* Notons que la mise en gras, en italique ; l’insertion de zone de code R, d’image, de liens, de puce numérotée ou non peut se faire directement sans avoir recours aux onglets “Format” et “Insert”.

**NB :** **Il est possible d’écrire un fichier Quarto sur la base d’un document de référence !**

Pour plus de détails, [cliquez](https://quarto.org/docs/output-formats/pdf-basics.html) !

### Présentation Quarto

Très similaires aux présentations à base de Canvas, les présentations Quarto offrent des possibilités de personnalisations très poussées. Il existe 3 types de sorties : MS PowerPoint -> pptx, reveal.js -> revealjs, Beamer (PDF) ->.

Pour la présentation reveal.js (comme pour la plupart des présentations sur R), les slides sont délimitées par ## ou par ---. Elles peuvent comporter :

* une première page contenant le titre, l’auteur, le logo et même des notes de bas de page
* les slides, de façon générale, peuvent être animées (ordre d’affichage, un peu de shiny,…) ; interactive ; personnalisées (arrière-plan, style ou thème de présentation,…)
* les slides peuvent comporter des graphiques et tableaux (personnalisés, plus d’un tout en s’assurant de leurs emplacement,…) ; les caractères spéciaux et symboles ; notes notes de bas de page ; les codes (R, latex,…) ; les équations…

Vous trouverez plus d’informations sur ce type de présentations ainsi que sur d’autres sur [le site quarto resevé](https://quarto.org/docs/presentations/revealjs/).

### Livre Quarto ou Quarto book

Il s’agit, en effet, du présent document !!

**Pour en créer, il faudrait :**

(créer) un nouveau projet [new project] -> nouveau repository [new directory] -> livre quarto [Quarto book] -> (configurer).

Un ensemble de 5 fichiers est créé. Il représente le squelette du livre. Il s’agit de :

* \_quarto.yml : il est comme la partie <head> <\head> d’un fichier .html. Il connecte toutes les parties (fichiers quarto utilisés) et comporte l’ensemble des métadonnés concernant le livre. C’est le YAML du livre quarto.
* index.qmd, intro.qmd, references.qmd, summary.qmd : ils représentent respectivement le contenu de la *préface, l’introduction, des références et du résumé* du livre.

**Ainsi pour ajouter un chapitre, il faudrait juste :**

1. créer un nouveau document quarto
2. l’enregistrer dans le dossier du projet
3. le spécifier dans le fichier .yml ; dans la partie chapters (souhaitée) ; suivant l’ordre que vous désirez (avant ou après summary.qmd, par exemple).

Pour plus de détails, [cliquez](https://quarto.org/docs/books/) !

**NB :**

* l’ensemble des documents Quarto peuvent être mis en forme en étant liés à des ***fichiers CSS.***
* Pour en savoir plus sur d’autres fonctionnalités Quarto, [visitez le site](https://quarto.org/docs/guide/) !

# Présentation 5 : R vers Excel

## Utilité

R est un langage de programmation largement utilisé pour l’analyse statistique et la visualisation des données. Excel aussi est un logiciel populaire utilisé pour l’organisation et la manipulation de plusieurs types de données. Il est alors intéressant d’observer le lien entre ces 2 logiciels afin de profiter à la fois des facilités de R et de celles d’Excel. La création de tableaux ou plus généralement le traitement préalable dans R (par grâce aux fontions de gtsummary, par exemple), l’exportation des tableaux de/et/ou résultats vers Excel et la mise en forme Excel à partir de R sont entre autres des approches qui permettent de gagner du temps et de rendre nos travaux plus reproductibles.

## Packages

**readxl** : ce package permet d’importer des données à partir de fichiers Excel

**writexl** : ce package permet d’exporter des données depuis R vers des fichiers Excel (.xlsx) ; créer des fichiers Excel avec différentes feuilles de calcul et formats.

**openxlsx** : ce package permet de lire, d’écrire et de manipuler des fichiers Excel (.xlsx) (création de feuilles de calcul, ajout de graphiques, gestion des styles, des formats de cellules, etc)

**xlsx** : similaire au précédent (créer des feuilles de calcul, gérer les formats de cellules et de fusionner des cellules…)

**r2excel** : ce package permet aux utilisateurs de manipuler des données, de créer des rapports, de générer des graphiques et d’effectuer d’autres analyses en utilisant la puissance de R tout en conservant la flexibilité et la popularité du format Excel.

## Quelques fonctions à retenir

* Créer des éléments Excel
  + r2excel::createWorkbook() : pour créer un classeur Excel
  + xlsx::createSheet() : pour créer une feuille Excel
  + add\_sheet() : ajouter des feuilles
* Importer des données depuis Excel : xlsx.openFile() ; readxl::read\_excel()
* Écrire dans Excel
  + openxlsx::write.xlsx() ; writexl::writeData() : écrire des données dans fichier Excel
  + write\_xlsx\_from\_dataframes() : écrire plusieurs dataframe sur des feuilles différentes
  + write\_xlsx\_from\_lists() : écrire une liste d’éléments
  + add\_table() ; xlsx.addTable() : ajouter un tableau formaté
  + xlsx.addPlot() : ajouter un graphique
* Mise en forme
  + xlsx.addHeader() : ajouter un entête de titre et/ou sous-titre
  + xlsx.addParagraph() : ajouter un paragraphe
  + setColWidths() : largeur des colonnes
  + createStyle() ; addStyle() : pour les styles
* Enregistrer : xlsx::saveWorkbook()

# Présentation 6 : Text Mining

## Utilité

Le text mining est une méthode d’analyse de données qui vise à extraire des informations significatives à partir d’une base de données ou d’un corpus de texte.

## Packages

**tm** : d’effectuer un prétraitement du texte inclut la suppression des nombres, des ponctuations et des espaces…

**wordcloud** : extraire la matrice des termes et créer un nuage de mots…

**tidytext** : l’équivalent de tidyverse en analyse de données textuelles

## Quelques fonctions

tm:: tm\_map() : prétraitement

TermDocumentMatrix() : transformation en matrice

wordcloud() : nuage des mots

findFreqTerms() : les mots les plus fréquents

findAssocs() : recherche d’association

tidytext::unnest\_tokens() : séparer les mots. En ajoutant l’argument token=ngrams et n=2 on obtient les **bigrammes** ( couple de mots)

# Présentation 7 : Systèmes d’équations non linéaires sur R

## Définition et utilité

Un système d’équations non linéaire est un système d’équations dans lequel au moins l’un d’entre eux est non linéaire, c’est-à-dire ne peut pas être exprimé comme une combinaison linéaire des inconnues et présente une constante. La résolution d’un système d’équations non linéaire est plus complexe que celui d’un système d’équation linéaire. Cependant le logiciel R nous offre des fonctionnalités pour résoudre aisément ce type de système. Deux catégories de méthodes de résolutions ont été présentées par le groupe chargé des systèmes d’équations non linéaires. Il s’agit des ***méthodes directes et des méthodes indirectes***.

## Packages

* méthodes directes : package **rootSolve** ou package **nleqslv**
* méthodes indirectes : package **stats** ou package **optimx**

## À retenir !

* **Méthodes directes :**
  + À partir du package rootSolve, la résolution du système se fait grâce à la fonction **multiroot()** qui prend en argument un vecteur de fonctions d’équation multivariée, ainsi qu’un vecteur initial de valeurs approchées pour les variables et le type de méthode souhaitée. Ce dernier argument est optionnel !! Si aucune méthode n’est spécifiée, la méthode par défaut est appliquée.
  + La fonction **multiroot()** retourne les solutions du système d’équation et le nombre d’itérations avant l’obtention des solutions.
  + La fonction **nleqslv()** du package nleqslv permet de résoudre numériquement des systèmes d’équations non linéaires multivariées en utilisant les méthode de Newton et de Broyden. Elle prend les mêmes arguments que la fonction précédente.
* **Méthodes indirectes :**
* Elles consistent à traduire le système d’équation en un problème d’optimisation en définissant une fonction objective à minimiser ou à maximiser. Cette fonction objective est généralement obtenue en prenant l’écart entre les valeurs réelles des équations et les valeurs en fonctions des inconnues. Ensuite on applique les méthodes d’optimisation pour l’obtention des solutions.
* Sous R, on dispose des fonctions comme **optim()** du package stats et **optimx()** du package optimxpour résoudre les problèmes d’optimisation

# Présentation 8 : Python dans R

## Utilité

R et Python sont des langages de programmation très utilisés par les statisticiens dans l’exécuion de leurs tâches. C’est alors un avantage pour eux de pouvoir bénéficier simultanément des fonctionnalités et des bibliothèques offertes par les deux langages.

## Packages

Avant de pouvoir utiliser cette interaction, il est nécessaire d’installer le package **reticulate**.

Le package reticulate est une bibliothèque de R qui facilite l’interaction avec Python à l’intérieur de l’environnement R. Celui-ci permet d’appeler directement des fonctions Python, d’utiliser les bibliothèques et modules Python dans R.

**NB :** **Il est important d’installer Python sur votre machine, avec une version supérieure ou égale à 2.7**

## Quelques fonctions importantes

reticulate::install\_python() : pour installer python en cas d’absence

reticulate::use\_python() : configuration de l’environnement python

reticulate::py\_install() : installer un module python

reticulate::py\_import() : importer un module python. module$nom\_function pour accéder à une fonction du module

reticulate::py\_run\_file() ; reticulate::py\_run\_string() : exécuter un script python dans R

reticulate::repl\_python() : accéder temporairement à la console python dans R

reticulate::r\_to\_py() : convertir un objet de R en python

reticulate::py\_to\_r() : convertir un objet python en R

**NB :**

* pour accéder aux objets python dans la section r, on écrit : py$nom\_objet\_python
* pour accéder aux objets r dans la section python, on écrit : r.nom\_objet\_R

# Présentation 9 : Calcul parallèle avec R

## Le calcul parallèle

Le calcul parallèle est une méthode d'optimisation de tâches sur R qui permet de pallier aux problèmes de manipulation de jeux de données relativement grands à partir d'un calcul séquentiel. L'avantage de cette méthode se retrouve dans le fait qu'il permet d'effectuer plus rapidement et de façon indépendante des tâches réparties entre les unités de calcul sur la base de données volumineuses. Le calcul parallèle consiste à :

* répartir un calcul informatique en des blocs de calculs indépendants ;
* exécuter simultanément (en parallèle) les blocs de calcul sur plusieurs unités de calcul ;
* rassembler les résultats obtenus par chaque bloc pour obtenir le résultat final.

## Principe

Le calcul parallèle est basé sur **le principe MapReduce**. Le MapReduce est un principe qui consiste à passer des calculs sur chaque blocs au résultat sur l'ensemble des données. La procédure ou étape du MapReduce :

1. Démarrer n processus "travailleurs" (i.e.cœurs de calcul) et les initialiser ;
2. Envoyer les fonctions et données nécessaires pour chaque tache aux travailleurs ;
3. Séparer les taches en n opérations d'envergure similaire ou différente et les envoyer aux travailleurs ;
4. Attendre que tous les travailleurs aient terminer leurs calculs et obtenir leurs résultats ;
5. Rassembler les résultats des différents travailleurs ;
6. Arrêter les processus travailleurs.

## Dans R

### Packages

Les principaux packages R utilisés pour le calcul parallèle sont **parallel, doParallel, foreach** et **rmr2**

### Implémentation

Le package parallel permet de démarrer et d'arrêter un cluster de plusieurs processus travailleur (étape 1). En plus du package parallel, le package doParallel est exploité et permet de gérer les processus travailleurs et la communication (étape 1).

Enfin, on gère l'articulation avec le package foreach qui permet, lui, de gérer le dialogue avec les travailleurs (envois, réception et rassemblement des résultats - étapes 2, 3, 4 et 5).

Le package rmr2 quant à lui permet l'application du MapReduce.

**NB :** Pour l'importation des bases de données volumineuses sous R, on se sert de la fonction **read\_fs()** du package **fs**.

# Presentation 10 : Cartographie sur R

## Utilité

L’information imagée et l’analyse tenant compte de la zone d’observation s’avèrent importantes. R nous offre, à cet effet, une flexibilité d’analyse et de traitement de ce type de données qu’il sera utile pour nous d’apprendre et de maîtriser.

## Packages

Pour une excellente analyse, un bon traitement et une bonne représentation des données cartographiques, il est important de s’assurer de la présence d’un certain nombre de package.

## Que doit-on retenir ?

### Fichiers et notions théoriques

* Types de données :
  + raster = image ~ matrice composée de pixels de même taille.
  + *Formats de fichiers* : .jpg ou .png ~image, visualisation ; .asc ~ coordonnées spaciales des cellules et limites du raster ; .db ~ table d’attributs : informations non spaciales associées aux cellules.
  + vecteurs = points, lignes et polygones
  + *Formats de fichiers* :
    - shapfile (.shp) ~ informations liées à la géométrie des unités spaciales. Il est accompagné de **.dbf** ~ table d’attribut : informations liées aux unités spaciales du *.shp* ; de **.shx** ~ définit la relation entre les coordonnées des polygones du *.shp* et de **.proj** ~ contient les coordonnées de projections des éléments du *.shp* .
    - GeoJSON (GJSON) ~ une sorte de dictionnaire contenant à la fois les informations géographiques (coordonnées des vecteurs), les informations statistiques liées à ces vecteurs et le système de projection utilisé.
* Fichier Geopackage (.gpkg) : combine rasters et vecteurs ; données spaciales et informations statistiques associées.

### Fonctions

* Passer à des données statistiques simples à des données spaciales :
  + utiliser coordinates() pour la classe sp
  + utiliser st\_as\_sf() pour la classe sf
  + st\_drop\_geometry() ~ éliminer les coordonnées spaciales et repasser en informations statistiques brutes
* Importer des données spaciales :
  + utiliser rgdal::readOGR() pour les vecteurs avec la classe sp
  + utiliser st\_read() pour lire les vecteurs en format .shp ou .gpkg aveec la classe sf
  + utiliser raster::raster() pour lire les rasters
  + st\_write() ~ exporter les données spaciales sous format .json ou .shp (par exemple)
* Représenter des cartes :
  + plot() , title() , legend() : pour des représentations basiques
  + mapsf::mf\_map() , mapsf::mf\_label() , mapsf::mf\_scale() : pour représenter la carte et ajouter des étiquètes et une échelle.
  + plot.igraph() : pour représenter les flux
  + ggplot()

# Présentation 11 : R-Shiny

## Utilité

Shiny est un package de R qui permet de créer des applications web interactives. Il fournit un moyen simple et rapide de construire des tableaux de bord, des visualisations de données et des interfaces utilisateur dynamiques. Shiny offre une panoplie de possibilités pour présenter nos données et résultats d'analyse sur une sortie web. Les utilisateurs peuvent interagir avec l’application web en modifiant les paramètres, en sélectionnant des données spécifiques et en visualisant les résultats mis à jour en temps réel.

## À retenir !

La création d’une application Shiny implique généralement la définition d’une **interface utilisateur** dans un fichier R Shiny, qui spécifie les éléments interactifs et leur mise en page. Ensuite, l'on peut définir les comportements réactifs dans une fonction R qui manipule les entrées et génère les sorties en fonction de ces entrées. Cette fonction est celle qui attribue les capacités de réactivité (mise à jour automatique) à l'application ainsi créée. Une fois l’application terminée, elle peut être déployée sur un **serveur Shiny** pour être accessible via un navigateur web.

Les principales fonctions du **package Shiny** sont :

* **shinyUI()** : Cette fonction est utilisée pour définir l’interface utilisateur (UI) de l’application Shiny. On peut y spécifier les éléments interactifs tels que les boutons, les formulaires, les graphiques et les tableaux, ainsi que leur disposition et leur apparence.
* **shinyServer()** : Cette fonction est utilisée pour définir le serveur de l’application Shiny. C’est ici qu'est définit le comportement réactif de l’application en spécifiant comment les entrées des utilisateurs sont traitées et comment les sorties sont mises à jour en fonction de ces entrées.