# Missing data

# Số liệu bị thiếu

This page will cover how to:

1. Assess missingness
2. Filter out rows by missingness
3. Plot missingness over time
4. Handle how NA is displayed in plots
5. Perform missing value imputation: MCAR, MAR, MNAR

Chương này sẽ bao gồm cách để:

1. Đánh giá số liệu bị thiếu
2. Lọc ra các hàng có số liệu bị thiếu
3. Vẽ biểu đồ số liệu bị thiếu theo thời gian
4. Thực hiện gán số liệu bị thiếu: MCAR, MAR, MNAR

## Preparation

### Load packages

This code chunk shows the loading of packages required for the analyses. In this handbook we emphasize p\_load() from **pacman**, which installs the package if necessary and loads it for use. You can also load installed packages with library() from **base** R. See the page on [R basics](#r-basics) for more information on R packages.

Đoạn mã này hiển thị cách tải các gói lệnh cần thiết cho phân tích. Trong cuốn sổ tay này, chúng tôi nhấn mạnh đến hàm p\_load() từ gói lệnh **pacman**, mà hàm này sẽ cài đặt gói lệnh khi cần thiết và tải nó để sử dụng. Có thể tải các gói lệnh đã cài đặt bằng hàm library() trong **base** R. Xem thêm thông tin về các gói lệnh của R trong chương [Cơ bản về R](#r-basics).

### Import data

### Nhập số liệu

We import the dataset of cases from a simulated Ebola epidemic. If you want to follow along, [click to download the “clean” linelist](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/case_linelists/linelist_cleaned.rds) (as .rds file). Import your data with the import() function from the **rio** package (it accepts many file types like .xlsx, .rds, .csv - see the [Import and export](#import-and-export) page for details).

The first 50 rows of the linelist are displayed below.

Chúng tôi nhập bộ số liệu của các ca bệnh được mô phỏng từ một vụ dịch Ebola. Để tiện làm theo, [nhấp tải số liệu linelist](https://github.com/epirhandbook/Epi_R_handbook/raw/master/data/case_linelists/linelist_cleaned.rds) “đã được làm sạch” (dưới dạng .rds). Nhập số liệu này bằng hàm import() trong gói lệnh **rio** (nó chấp nhận nhiều loại tập tin như .xlsx, .rds, .csv – xem chi tiết trong chương [Nhập và xuất số liệu](#import-and-export)).

Bên dưới là hiển thị của 50 hàng đầu tiên của bộ số liệu linelist.

### Convert missing on import

### Chuyển đổi số liệu bị thiếu khi nhập

When importing your data, be aware of values that should be classified as missing. For example, 99, 999, “Missing”, blank cells ("“), or cells with an empty space (” "). You can convert these to NA (R’s version of missing data) during the data import command.  
See the page on importing page section on [Missing data](#import_missing) for details, as the exact syntax varies by file type.

Khi nhập số liệu, hãy lưu ý các giá trị mà được phân loại như giá trị bị thiếu. Ví dụ: 99, 999, “Giá trị bị thiếu”, để trống (“”), hoặc các ô có khoảng trống (“ “). Chúng ta có thể chuyển đổi các giá trị này thành NA (số liệu bị thiếu trong R) trong khi thực hiện nhập số liệu.

Xem phần nhập số liệu trong chương [Số liệu bị thiếu](#import_missing) để biết thêm chi tiết, vì cú pháp chính xác sẽ khác nhau tùy theo loại tập tin.

## Missing values in R

## Giá trị bị thiếu trong R

Below we explore ways that missingness is presented and assessed in R, along with some adjacent values and functions.

Bên dưới, chúng ta sẽ khảo sát rõ cách số liệu bị thiếu được trình bày và đánh giá trong R cùng với các giá trị liền kề và các hàm.

### NA

In R, missing values are represented by a reserved (special) value - NA. Note that this is typed without quotes. “NA” is different and is just a normal character value (also a Beatles lyric from the song Hey Jude).

Trong R, giá trị bị thiếu được đại diện bởi một giá trị dành riêng (đặc biệt) - NA. Lưu ý rằng giá trị này được nhập không có dấu ngoặc kép. “NA” là khác biệt và chỉ là một giá trị ký tự bình thường (cũng như lời bài hát của Beatles trong bài Hey Jude).

Your data may have other ways of representing missingness, such as “99”, or “Missing”, or “Unknown” - you may even have empty character value "" which looks “blank”, or a single space " ". Be aware of these and consider whether to [convert them to NA during import](#import_missing) or during data cleaning with na\_if().

Số liệu của chúng ta có thể có các cách khác để biểu thị giá trị bị thiếu, như là “99”, hoặc “Giá trị bị thiếu / Missing”, hoặc “Không biết / Unknown”, thậm chí chúng ta có giá trị ký tự trống "" như “để trống / blank”, hoặc một khoảng trắng " ". Cần lưu ý những giá trị này và xem xét có nên [chuyển đổi chúng thành NA khi nhập vào R](#import_missing)  hoặc trong khi làm sạch số liệu bằng hàm na\_if().

In your data cleaning, you may also want to convert the other way - changing all NA to “Missing” or similar with replace\_na() or with fct\_explicit\_na() for factors.

Trong lúc làm sạch số liệu, chúng ta cũng có thể muốn chuyển đổi số liệu theo một cách khác – chuyển đổi tất cả NA thành “Giá trị bị thiếu / Missing” hoặc làm tương tự với hàm replace\_na() hoặc với hàm fct\_explicit\_na() cho các factors.

### Versions of NA

### Các phiên bản của NA

Most of the time, NA represents a missing value and everything works fine. However, in some circumstances you may encounter the need for variations of NA specific to an object class (character, numeric, etc). This will be rare, but you should be aware.

Trong hầu hết trường hợp, NA đại diện cho giá trị bị thiếu và mọi thứ đều vận hành tốt. Tuy nhiên, trong một số trường hợp, chúng ta có thể gặp phải những tình huống khó khăn đối với *các biến thể* của NA mà đặc trưng cho một lớp đối tượng (ký tự, số, .v.v.). Điều này rất hiếm, nhưng chúng ta cần phải lưu ý.

The typical scenario for this is when creating a new column with the **dplyr** function case\_when(). As described in the [Cleaning data and core functions](#clean_case_when) page, this function evaluates every row in the data frame, assess whether the rows meets specified logical criteria (right side of the code), and assigns the correct new value (left side of the code). Importantly: all values on the right side must be the same class.

Tình huống điển hình cho trường hợp này là khi tạo một cột mới bằng hàm case\_when() từ gói lệnh **dplyr.** Như được mô tả trong chương [Làm sạch số liệu và các hàm chính](#clean_case_when), hàm này lượng giá mọi hàng trong khung số liệu, đánh giá xem các hàng có đáp ứng với các tiêu chí logic cụ thể hay không (phía bên phải của mã), và gán chính xác giá trị mới (phía bên trái của mã). Quan trọng: tất cả các giá trị bên phải cùng một lớp biến số.

If you want NA on the right side, you may need to specify one of the special NA options listed below. If the other right side values are character, consider using “Missing” instead or otherwise use NA\_character\_. If they are all numeric, use NA\_real\_. If they are all dates or logical, you can use NA.

* NA - use for dates or logical TRUE/FALSE
* NA\_character\_ - use for characters
* NA\_real\_ - use for numeric

Nếu chúng ta muôn NA ở phía bên phải, chúng ta có thể cần phải chỉ định một trong các tùy chọn đặc biệt của NA được liệt kê bên dưới. Nếu các giá trị bên phải khác là ký tự, cân nhắc sử dụng “Missing” hoặc sử dụng NA\_character\_. Nếu các giá trị là số, sử dụng NA\_real\_. Nếu tất cả các giá trị là ngày tháng hoặc logic, chúng ta có thể sử dụng NA.

* NA – dùng cho biến ngày tháng hoặc biến số logic ĐÚNG/SAI
* NA\_character\_ - dùng cho biến số ký tự
* NA\_real\_ - dung cho biến dạng số

Một lần nữa, chúng ta sẽ không gặp các loại biến thể này trừ khi chúng ta sử dụng hàm case\_when() để tạo biến mới. Xem phần [Tài liệu về NA](https://stat.ethz.ch/R-manual/R-devel/library/base/html/NA.html) trong R để biết thêm thông tin.

Again, it is not likely you will encounter these variations unless you are using case\_when() to create a new column. See the [R documentation on NA](https://stat.ethz.ch/R-manual/R-devel/library/base/html/NA.html) for more information.

### NULL

NULL is another reserved value in R. It is the logical representation of a statement that is neither true nor false. It is returned by expressions or functions whose values are undefined. Generally do not assign NULL as a value, unless writing functions or perhaps writing a [**shiny** app](#dashboards-with-shiny) to return NULL in specific scenarios.

Null-ness can be assessed using is.null() and conversion can made with as.null().

See this [blog post](https://www.r-bloggers.com/2010/04/r-na-vs-null/) on the difference between NULL and NA.

NULL là một giá trị dành riêng trong R. Nó đại điện cho một câu lệnh không đúng cũng không sai. Nó là kết quả của các biểu thức hoặc hàm có giá trị không xác định. Nói chung, không gán NULL như một giá trị, trừ khi viết hàm khi viết một [**Một ứng dụng của shiny** app](#dashboards-with-shiny) để trả về NULL trong các tình huống cụ thể.

Null có thể được đánh giá bằng cach sử dụng hàm is.null() và có thể chuyển đổi bằng hàm as.null().

Xem [trang blog](https://www.r-bloggers.com/2010/04/r-na-vs-null/) này để biết về sự khác biệt giữa NULL và NA.

### NaN

Impossible values are represented by the special value NaN. An example of this is when you force R to divide 0 by 0. You can assess this with is.nan(). You may also encounter complementary functions including is.infinite() and is.finite().

Giá trị không phù hợp được đại diện bừng một giá trị đặc biệt NaN. Một ví dụ cho giá trị này là khi chúng ta bắt buộc R chia 0 cho 0. Chúng ta có thể đánh giá giá trị này bằng cách dùng hàm is.nan(). Chúng ta cũng có thể gặp các hàm bổ sung bao gồm is.infinite() và is.finite().

### Inf

Inf represents an infinite value, such as when you divide a number by 0.

Inf đại diện cho giá trị vô hạn như là khi chia một số nào đó cho 0.

As an example of how this might impact your work: let’s say you have a vector/column z that contains these values: z <- c(1, 22, NA, Inf, NaN, 5)

Một ví dụ về cách mà giá trị này ảnh hưởng đến số liệu của chúng ta: giả sử chúng ta có một véc tơ hay một cột z chứa các gái trị sau: z <- c(1, 22, NA, Inf, NaN, 5)

If you want to use max() on the column to find the highest value, you can use the na.rm = TRUE to remove the NA from the calculation, but the Inf and NaN remain and Inf will be returned. To resolve this, you can use brackets [ ] and is.finite() to subset such that only finite values are used for the calculation: max(z[is.finite(z)]).

Nếu chúng ta muốn sử dụng hàm max() của cột bên trên để tìm giá trị lớn nhất, chúng ta có thể sử dụng na.rm = TRUE để loại bỏ NA khỏi phép tính, nhưng Inf và NaN vẫn còn, và Inf sẽ là giá trị lớn nhất. Để giải quyết vấn đề này, chúng ta có thể sử dụng dấu ngoặc vuông [ ] và is.finite() để tạo tập hợp con mà chỉ có các giá trị hữu hạn được sử dụng để tính toán: max(z[is.finite(z)]).

### Examples

### Ví dụ

| **Lệnh tron R** | **Kết quả** |
| --- | --- |
| 5 / 0 | Inf |
| 0 / 0 | NaN |
| 5 / NA | NA |
| 5 / Inf |0NA - 5|NAInf / 5|Infclass(NA)| "logical"class(NaN)| "numeric"class(Inf)| "numeric"class(NULL)` | “NULL” |

“NAs introduced by coercion” is a common warning message. This can happen if you attempt to make an illegal conversion like inserting a character value into a vector that is otherwise numeric.

NULL is ignored in a vector.

Variance of one number results in NA.

“NAs được tạo ra do ép buộc” là một cảnh báo phổ biến. Điều này có thể xảy ra khi chúng ta cố gắng thực hiện một chuyển đổi không hợp lệ như chèn một giá trị ký tự vào trong một véc tơ dạng số.

NULL được bỏ qua trong một véc tơ.

Phương sai của một số cho kết quả là NA.

## Useful functions

## Các hàm hữu dụng

The following are useful **base** R functions when assessing or handling missing values:

### is.na() and !is.na()

Dưới đây là các hàm hữu dụng trong **base** R khi đánh giá hoặc xử lý các giá trị bị thiếu:

### is.na() và !is.na()

Use is.na()to identify missing values, or use its opposite (with ! in front) to identify non-missing values. These both return a logical value (TRUE or FALSE). Remember that you can sum() the resulting vector to count the number TRUE, e.g. sum(is.na(linelist$date\_outcome)).

Sử dụng hàm is.na()để nhận ra các giá trị bị thiếu hoặc sử dụng hàm đối của nó (với dấu ! ở phía trước) để nhận ra các giá trị không bị thiếu. Cả hai hàm trên đều trả kết quả về một giá trị logic (ĐÚNG hoặc SAI). Nhớ rằng chúng ta có thể dùng hàm sum() đối với véc tơ kết quả/đầu ra để đếm số lượng ĐÚNG, ví dụ sum(is.na(linelist$date\_outcome)).

### na.omit()

This function, if applied to a data frame, will remove rows with any missing values. It is also from **base** R.  
If applied to a vector, it will remove NA values from the vector it is applied to. For example:

Nếu được áp dụng cho một khung số liệu, hàm này sẽ loại bỏ các hàng có *bất kỳ* giá trị nào bị thiếu. Hàm này cũng từ **base** R.

Nếu áp dụng cho một véc tơ, hàm này sẽ loại bỏ các giá trị NA của véc tơ này. Ví dụ:

### drop\_na()

This is a **tidyr** function that is useful in a [data cleaning pipeline](#cleaning-data-and-core-functions). If run with the parentheses empty, it removes rows with any missing values. If column names are specified in the parentheses, rows with missing values in those columns will be dropped. You can also use “tidyselect” syntax to specify the columns.

Đây là một hàm trong gói lệnh **tidyr** mà hữu dụng trong [đường dẫn để làm sạch số liệu](#cleaning-data-and-core-functions). Nếu thực hiện hàm với dấu ngoặc trống, nó sẽ loại bỏ các hàng có *bất kỳ* giá trị nào bị thiếu. Nếu cụ thể tên cột trong dấu ngoặc, các hàng nào có giá trị bị thiếu trong những cột này sẽ bị loại bỏ. Chúng ta có thể sử dụng cú pháp “tidyselect” để chỉ định các cột này.

### na.rm = TRUE

When you run a mathematical function such as max(), min(), sum() or mean(), if there are any NA values present the returned value will be NA. This default behavior is intentional, so that you are alerted if any of your data are missing.

You can avoid this by removing missing values from the calculation. To do this, include the argument na.rm = TRUE (“na.rm” stands for “remove NA”).

Khi chúng ta thực hiện một hàm toán học như max(), min(), sum() or mean(), nếu có sự hiện diện của bất kỳ giá trị NA,kết quả sẽ là NA. Cách xử lý mặc định này là có chủ ý để cảnh báo chúng ta khi có số liệu bị thiếu.

Chúng ta có thể tránh việc này bằng cách loại bỏ các giá trị bị thiếu ra khỏi phép tính. Để thực hiện điều này, thêm đối số na.rm = TRUE vào trong phép tính (“na.rm” là viết tắt của “remove NA”).

## Assess missingness in a data frame

## Đánh giá giá trị bị thiếu trong một khung số liệu

You can use the package **naniar** to assess and visualize missingness in the data frame linelist.

Chúng ta có thể sử dụng gói lệnh **naniar** để đánh giá và hình ảnh hóa các giá trị bị thiếu trong bộ số liệu linelist.

### Quantifying missingness

### Định lượng số liệu bị thiếu

To find the percent of all values that are missing use pct\_miss(). Use n\_miss() to get the number of missing values.

Để tìm phần trăm của tất cả các giá trị bị thiếu, sử dụng hàm pct\_miss(). Để biết số lượng bị thiếu, sử dụng hàm n\_miss().

The two functions below return the percent of rows with any missing value, or that are entirely complete, respectively. Remember that NA means missing, and that `"" or " " will not be counted as missing.

Hai hàm bên dưới trả về phần trăm số hàng có bất kỳ giá trị bị thiếu hoặc hoàn toàn đầy đủ. Hãy nhớ rằng NA có nghĩ là giá trị bị thiếu, và `"" hoặc " " sẽ không được đếm là giá trị bị thiếu.

### Visualizing missingness

### Hình ảnh hóa số liệu bị thiếu

The gg\_miss\_var() function will show you the number (or %) of missing values in each column. A few nuances:

* You can add a column name (not in quote) to the argument facet = to see the plot by groups
* By default, counts are shown instead of percents, change this with show\_pct = TRUE
* You can add axis and title labels as for a normal ggplot() with + labs(...)

Hàm gg\_miss\_var() sẽ hiển thị cho chúng ta số lượng (hoặc %) của các giá trị bị thiếu trong mỗi cột. Một vài sắc thái như sau:

* Chúng ta có thể thêm tên cột (không ở trong dấu ngoặc kép) vào đối số facet = để xem biểu đồ theo nhóm
* Theo mặc định, số đếm sẽ được hiển thị thay vì con số phần trăm, để thay đổi điều này bằng cách sử dụng thêm show\_pct = TRUE
* Chúng ta có thể thêm nhãn các trục và tiêu đề như đối với các hàm ggplot() thông thường với + labs(...)

Here the data are piped %>% into the function. The facet = argument is also used to split the data.

You can use vis\_miss() to visualize the data frame as a heatmap, showing whether each value is missing or not. You can also select() certain columns from the data frame and provide only those columns to the function.

Ở đây số liệu được dẫn bằng ký hiệu %>% vào trong hàm. Đối số facet = cũng được sử dụng để phân chia số liệu.

Chúng ta có thể sử dụng hàm vis\_miss() để hình ảnh hóa số liệu dưới dạng một bản đồ nhiệt, mà cho biết từng giá trị có bị thiếu hay không. Chúng ta cũng có thể sử dụng hàm select() cho một số cột nhất định từ khung số liệu và chỉ cung cấp các cột đó vào trong hàm.

### Explore and visualize missingness relationships

### Khám phá và hình ảnh hóa các mối quan hệ của số liệu bị thiếu

How do you visualize something that is not there??? By default, ggplot() removes points with missing values from plots.

Làm thế nào chúng ta hình ảnh hóa một cái gì đó mà không có ở đó??? Theo mặc định, hàm ggplot() sẽ loại bỏ các giá trị bị thiếu ra khỏi biểu đồ.

**naniar** offers a solution via geom\_miss\_point(). When creating a scatterplot of two columns, records with one of the values missing and the other value present are shown by setting the missing values to 10% lower than the lowest value in the column, and coloring them distinctly.

**Gói lệnh naniar** cung cấp một giả pháp thông qua hàm geom\_miss\_point(). Khi tạo biểu đồ phân tán của hai cột, các bản ghi có một trong các giá trị bị thiếu và sự hiện diện của giá trị khác được hiển thị bằng cách đặt các giá trị bị thiếu thấp hơn 10% so với giá trị thấp nhất trong cột đó và và tô màu để dễ nhận ra.

In the scatterplot below, the red dots are records where the value for one column is present but the value for the other column is missing. This allows you to see the distribution of missing values in relation to the non-missing values.

Trong biểu đồ phân tán bên dưới, các chấm màu đỏ là các bản ghi khi có giá trị của một cột nhưng thiếu giá trị của cột còn lại. Điều này cho phép chúng ta thấy được sự phân bố của các giá trị bị thiếu tương quan với giá trị không bị thiếu.

To assess missingness in the data frame stratified by another column, consider gg\_miss\_fct(), which returns a heatmap of percent missingness in the data frame by a factor/categorical (or date) column:

Để đánh giá giá trị bị thiếu trong một khung số liệu *được phân tầng bởi một cột* khác, hãy lưu ý đến hàm gg\_miss\_fct(), mà sẽ trả về một bản đồ nhiệt về phần trăm giá trị bị thiếu trong một khung số liệu *bởi một cột của biến số factor/phân loại* *(hoặc ngày)*:

This function can also be used with a date column to see how missingness has changed over time:

Hàm này cũng có thể được sử dụng đối với cột dạng ngày tháng để xem mức độ của số liệu bị thiếu thay đổi theo thời gian:

### “Shadow” columns

### Các cột “hình bóng”

Another way to visualize missingness in one column by values in a second column is using the “shadow” that **naniar** can create. bind\_shadow() creates a binary NA/not NA column for every existing column, and binds all these new columns to the original dataset with the appendix "\_NA". This doubles the number of columns - see below:

Một cách khác để hình ảnh hóa số liệu bị thiếu trong một cột bởi các giá trị của cột thứ hai là sử dụng “hình bóng/shadow” mà gói lệnh **naniar** có thể tạo ra. Hàm bind\_shadow() tạo ra một cột biến số nhị phân NA/không NA cho mọi cột hiện có và liên kết tất cả các cột mới này với bộ số liệu gốc bằng phần phụ "\_NA". Điều này làm tăng gấp đôi số cột – xem bên dưới:

These “shadow” columns can be used to plot the proportion of values that are missing, by any another column.

Các cột “hình bóng” này có thể được sử dụng để vẽ biểu đồ tỷ lệ các giá trị bị thiếu theo bất kỳ cột nào khác.

For example, the plot below shows the proportion of records missing days\_onset\_hosp (number of days from symptom onset to hospitalisation), by that record’s value in date\_hospitalisation. Essentially, you are plotting the density of the x-axis column, but stratifying the results (color =) by a shadow column of interest. This analysis works best if the x-axis is a numeric or date column.

Ví dụ: biểu đồ bên dưới cho thấy tỷ lệ các giá trị bị thiếu của các bản ghi trong biến số days\_onset\_hosp (số ngày từ ngày khởi phát triệu chứng đến ngày nhập viện) theo giá trị của bản ghi đó trong biến số date\_hospitalisation. Về bản chất là chúng ta đang vẽ biểu độ mật độ của cột này trên trục x, nhưng phân tầng kết quả (color =) bằng cột hình bóng đang được quan tâm. Phân tích này thể hiện tốt nhất khi trục x là cột số hoặc cột ngày.

You can also use these “shadow” columns to stratify a statistical summary, as shown below:

Chúng ta cũng có thể sử dụng cột “hình bóng” để phân tầng một bảng tóm tắt thống kê, như được hiển thị bên dưới:

An alternative way to plot the proportion of a column’s values that are missing over time is shown below. It does not involve **naniar**. This example shows percent of weekly observations that are missing).

1. Aggregate the data into a useful time unit (days, weeks, etc.), summarizing the proportion of observations with NA (and any other values of interest)
2. Plot the proportion missing as a line using ggplot()

Bên dưới là một cách khác để vẽ biểu đồ tỷ lệ các giá trị bị thiếu của một cột theo thời gian. Nó *không* liên quan đến gói lệnh **naniar**. Ví dụ này cho thấy phần trăm số liệu bị thiếu theo tuần.

1. Tổng hợp số liệu thành một đơn vị thời gian muốn sử dụng (ngày, tuần, .v.v.), tóm tắt tỷ lệ quan sát theo NA (và bất kỳ các giá trị quan tâm khác)
2. Vẽ biểu đồ tỷ lệ giá trị bị thiếu dưới dạng một dòng bằng cách sử dụng hàm ggplot()

Below, we take the linelist, add a new column for week, group the data by week, and then calculate the percent of that week’s records where the value is missing. (note: if you want % of 7 days the calculation would be slightly different).

Bên dưới, chúng ta lấy bộ số liệu linelist, thêm một cột mới cho tuần, nhóm số liệu theo tuần và sau đó tính phần trăm của các bản ghi có số liệu bị thiếu theo tuần. (lưu ý: nếu chúng ta muốn % của 7 ngày, việc tính toán sẽ hơi khác một chút).

Then we plot the proportion missing as a line, by week. The [ggplot basics](#ggplot-basics) page if you are unfamiliar with the **ggplot2** plotting package.

Sau đó, chúng ta vẽ biểu đồ tỷ lệ các giá trị bị thiếu dưới dạng một dòng theo tuần. Xem chương [Cơ bản về ggplot](#ggplot-basics) nếu chưa quen với gói lệnh vẽ biểu đồ **ggplot2.**

## Using data with missing values

## Sử dụng số liệu có giá trị bị thiếu

### Filter out rows with missing values

### Lọc bỏ các hàng có giá trị bị thiếu

To quickly remove rows with missing values, use the **dplyr** function drop\_na().

Để loại bỏ nhanh các hàng có giá trị bị thiếu, sử dụng hàm drop\_na()trong gói lệnh **dplyr.**

The original linelist has nrow(linelist) rows. The adjusted number of rows is shown below:

Bộ số liệu gốc linelist có nrow(linelist) hàng. Số hàng sau khi hiệu chỉnh được hiển thị bên dưới:

You can specify to drop rows with missingness in certain columns:

Chúng ta có thể chỉ định loại bỏ các hàng có giá trị bị thiếu trong một số hàng nhất định:

You can list columns one after the other, or use [“tidyselect” helper functions](#clean_tidyselect):

Chúng ta có thể liệt kê lần lượt từng cột một, hoặc sử dụng [các hàm trợ giúp trong “tidyselect”](#clean_tidyselect):

### Handling NA in ggplot()

### Xử lý NA trong hàm ggplot()

It is often wise to report the number of values excluded from a plot in a caption. Below is an example:

Thường là khôn ngoan khi báo cáo số lượng các giá trị bị loại ra trong phần chú thích của biểu đồ . Bên dưới là một ví dụ:

In ggplot(), you can add labs() and within it a caption =. In the caption, you can use str\_glue() from **stringr** package to paste values together into a sentence dynamically so they will adjust to the data. An example is below:

* Note the use of \n for a new line.
* Note that if multiple column would contribute to values not being plotted (e.g. age or sex if those are reflected in the plot), then you must filter on those columns as well to correctly calculate the number not shown.

Trong hàm ggplot(), chúng ta có thể thêm labs() và bên trong nó là một chú thích caption =. Trong phần chú thích này, chúng ta có thể sử dụng hàm str\_glue() từ gói lệnh **stringr** để dán các giá trị lại với nhau thành một câu động, vì vậy nó sẽ điều chỉnh theo số liệu. Bên dưới là một ví dụ:

* Lưu ý sử dụng \n để tạo một hàng mới (ngắt chữ xuống hàng).
* Lưu ý rằng nếu nhiền cột góp phần tạo ra các giá trị, thì sẽ không vẽ được trên biểu đồ (ví dụ như tuổi hoặc giới nếu chúng được phản ánh trên biểu đồ, thì chúng ta cũng phải lọc trên các cột đó để tính toán chính xác số lượng không được hiển thị.

Sometimes, it can be easier to save the string as an object in commands prior to the ggplot() command, and simply reference the named string object within the str\_glue().

Thỉnh thoảng, có thể dễ dàng hơn để lưu chuỗi này dưới dạng một đối tượng trong các lệnh trước khi dùng lệnh ggplot() và chỉ cần tham chiếu thên của đối tượng chuỗi bên trong hàm str\_glue().

### NA in factors

### NA trong các factor

If your column of interest is a factor, use fct\_explicit\_na() from the **forcats** package to convert NA values to a character value. See more detail in the [Factors](#factors) page. By default, the new value is “(Missing)” but this can be adjusted via the na\_level = argument.

Nếu cột đang quan tâm là một factor, sử dùng hàm fct\_explicit\_na() từ gói lệnh **forcats**  để chuyển đổi các giá trị NA thành giá trị ký tự. Xem thêm chi tiết trong chương [Factors](#factors). Theo mặc định, giá trị mới là “(Giá trị bị thiếu)” nhưng điều này có thể được điều chỉnh thông qua đối số na\_level =.

## Imputation

## Gán số liệu

Thỉnh thoảng, khi phân tích số liệu, điều quan trọng là phải “làm đầy khoảng trống” và gán giá trị cho số liệu bị thiếu. Trong khi chúng ta luôn luôn có thể chỉ cần phân tích một bộ số liệu sau khi đã loại bỏ tất cả các giá trị bị thiếu, nhưng điều này có thể đưa đến các vấn đề theo nhiều cách. Đây là các ví dụ:

Sometimes, when analyzing your data, it will be important to “fill in the gaps” and impute missing data While you can always simply analyze a dataset after removing all missing values, this can cause problems in many ways. Here are two examples:

1. By removing all observations with missing values or variables with a large amount of missing data, you might reduce your power or ability to do some types of analysis. For example, as we discovered earlier, only a small fraction of the observations in our linelist dataset have no missing data across all of our variables. If we removed the majority of our dataset we’d be losing a lot of information! And, most of our variables have some amount of missing data–for most analysis it’s probably not reasonable to drop every variable that has a lot of missing data either.
2. Depending on why your data is missing, analysis of only non-missing data might lead to biased or misleading results. For example, as we learned earlier we are missing data for some patients about whether they’ve had some important symptoms like fever or cough. But, as one possibility, maybe that information wasn’t recorded for people that just obviously weren’t very sick. In that case, if we just removed these observations we’d be excluding some of the healthiest people in our dataset and that might really bias any results.
3. Bằng cách loại bỏ tất cả các quan sát có giá trị bị thiếu hoặc các biến số có một số lượng lớn giá trị bị thiếu, chúng ta có thể giảm lực thống kê hay khả năng thực hiện một số loại phân tích. Ví dụ như khi chúng ta đã phát hiện trước đó, chỉ có một phần nhỏ của các quan sát trong bộ số liệu linelist không có số liệu bị thiếu trên tất cả các biến số. Nếu chúng ta xóa phần lớn số liệu, chúng ta sẽ mất rất nhiều thông tin! Và hầu hết các biến của bộ số liệu trên có một số số liệu bị thiếu – đối với hầu hết các phân tích, hầu như là không hợp lý để loại bỏ mọi biến có nhiều số liệu bị thiếu.
4. Tùy thuộc vào lý do tại sao số liệu bị thiếu, nhưng chỉ phân tích số liệu không bị thiếu có thể dẫn đến kết quả sai lệch. Ví dụ như khi chúng ta đã biết trước đó, chúng ta đang thiếu số liệu của một số bệnh nhân về việc họ có một số triệu chứng quan trọng như sốt hoặc ho hay không. Nhưng có một khả năng là có thể thông tin đó không được ghi nhận đối với những người mà rõ ràng không bị bệnh nặng. Trong trường hợp đó, nếu chúng ta chỉ loại bỏ những quan sát này, chúng ta sẽ loại trừ một số người khỏe nhất trong bộ số liệu và điều đó thật sự có thể làm sai lệch bất kỳ kết quả nào.

It’s important to think about why your data might be missing in addition to seeing how much is missing. Doing this can help you decide how important it might be to impute missing data, and also which method of imputing missing data might be best in your situation.

Điều quan trọng là suy nghĩ tại sao số liệu có thể bị thiếu cũng như là thiếu bao nhiêu. Làm điều này có thể giúp chúng ta xác định mức độ quan trọng của việc gán giá trị bị thiếu và lựa chọn phương pháp tốt nhất để gán giá trị bị thiếu trong tình huống của chúng ta.

### Types of missing data

### Các loại số liệu bị thiếu

Here are three general types of missing data:

Đây là ba loại số liệu bị thiếu

1. **Missing Completely at Random** (MCAR). This means that there is no relationship between the probability of data being missing and any of the other variables in your data. The probability of being missing is the same for all cases This is a rare situation. But, if you have strong reason to believe your data is MCAR analyzing only non-missing data without imputing won’t bias your results (although you may lose some power). [TODO: consider discussing statistical tests for MCAR]

**Số liệu bị thiếu Hoàn toàn Ngẫu nhiên** (MCAR). Điều này có nghĩa là không có mối quan hệ giữa xác suất số liệu bị thiếu và bất kỳ biến số nào trong bộ số liệu. Xác suất số liệu bị thiếu là ngang nhau đối với tất cả các ca. Đây là một tình huống hiếm gặp. Nhưng nếu chúng ta có lý do chính đáng để tin rằng số liệu của chúng ta là MCAR, thì chỉ phân tích số liệu không bị thiếu (mà không gán giá trị bị thiếu) sẽ không làm sai lệch kết quả (mặc dù chúng ta có thể mất một ít lực thống kê). [CẦN LÀM: lưu ý đến các thảo luận về các kiểm định thống kê cho MCAR]

1. **Missing at Random** (MAR). This name is actually a bit misleading as MAR means that your data is missing in a systematic, predictable way based on the other information you have. For example, maybe every observation in our dataset with a missing value for fever was actually not recorded because every patient with chills and and aches was just assumed to have a fever so their temperature was never taken. If true, we could easily predict that every missing observation with chills and aches has a fever as well and use this information to impute our missing data. In practice, this is more of a spectrum. Maybe if a patient had both chills and aches they were more likely to have a fever as well if they didn’t have their temperature taken, but not always. This is still predictable even if it isn’t perfectly predictable. This is a common type of missing data

**Số liệu bị thiếu Ngẫu nhiên** (MAR). Tên của loại số liệu này thực sự gây ra một ít hiểu lầm vì MAR có nghĩa là số liệu bị thiếu một cách có hệ thống, có thể dự đoán dựa trên các thông tin khác mà chúng ta có. Ví dụ như có lẽ giá trị bị thiếu đối với biến số sốt của mọi quan sát trong bộ số liệu là thật sự không được ghi nhận vì mọi bệnh nhân có ớn lạnh và đau nhức được giả định là bị sốt, vì vậy thân nhiệt của họ không bao giờ được đo. Nếu đúng như vậy, chúng ta có thể dễ dàng dự đoán rằng, mọi quan sát bị thiếu khi có ớn lạnh và đau nhức cũng sẽ bị sốt và chúng ta sẽ sử dụng thông tin này để gán các giá trị bị thiếu. Trong thực tế, đây là loại nhiều hơn. Có lẽ nếu một bệnh nhân có cả ớn lạnh và đau nhức, thì hầu như họ cũng bị sốt nếu thân nhiệt không được đo, nhưng không phải luôn luôn như vậy. Điều này vẫn có thể dự đoán được ngay cả khi nó không thể dự đoán một cách hoàn toàn, nhưng không phải là luôn luôn. Đây là một loại số liệu bị thiếu phổ biến.

1. **Missing not at Random** (MNAR). Sometimes, this is also called **Not Missing at Random** (NMAR). This assumes that the probability of a value being missing is NOT systematic or predictable using the other information we have but also isn’t missing randomly. In this situation data is missing for unknown reasons or for reasons you don’t have any information about. For example, in our dataset maybe information on age is missing because some very elderly patients either don’t know or refuse to say how old they are. In this situation, missing data on age is related to the value itself (and thus isn’t random) and isn’t predictable based on the other information we have. MNAR is complex and often the best way of dealing with this is to try to collect more data or information about why the data is missing rather than attempt to impute it.

**Số liệu bị thiếu Không Ngẫu nhiên** (MNAR). Thỉnh thoảng, loại này được gọi là **Không bị Thiếu Ngẫu nhiên (NMAR)**. Loại này giả định rằng xác suất của một giá trị bị thiếu là KHÔNG có hệ thống hoặc không thể dự đoán được bằng cách sử dụng thông tin khác mà chúng ta có, nhưng nó cũng không là giá trị bị thiếu một cách ngẫu nhiên. Trong tình huống này, số liệu bị thiếu không rõ lý do hoặc vì các lý do mà chúng ta không có bất kỳ thông tin nào. Ví dụ: trong bộ số liệu có thể thiếu thông tin về tuổi bởi vì một số bệnh nhân rất cao tuổi hoặc không biết hoặc từ chối cho biết tuổi. Trong tình huống này, số liệu bị thiếu về tuổi có liên quan đến giá trị của chính nó (và vì thế nó không phải là ngẫu nhiên) và không thể dự đoán được dựa trên các thông tin khác mà chúng ta có. MNAR là phức tạp và cách tốt nhất để giải quyết loại số liệu này thường là cố gắng thu thập thêm số liệu hoặc thông tin tại sao số liệu bị thiếu thay vì cố gắng gán giá trị cho nó.

In general, imputing MCAR data is often fairly simple, while MNAR is very challenging if not impossible. Many of the common data imputation methods assume MAR.

Nhìn chung, việc gán số liệu cho MCAR thường khá đơn giản, trong khi cho MNAR là khá thách thức nếu không muốn nói là không thể. Có nhiều phương pháp gán số liệu phổ biến cho các số liệu được cho là MAR.

### Useful packages

### Các gói lệnh hữu dụng

Some useful packages for imputing missing data are Mmisc, missForest (which uses random forests to impute missing data), and mice (Multivariate Imputation by Chained Equations). For this section we’ll just use the mice package, which implements a variety of techniques. The maintainer of the mice package has published an online book about imputing missing data that goes into more detail here (<https://stefvanbuuren.name/fimd/>).

Một số gói lệnh hữu dụng để gán số liệu bị thiếu là Mmisc, missForest (mà sử dụng kỹ thuật random forests để gán số liệu bị thiếu), và gói lệnh mice (viết tắt của Multivariate Imputation by Chained Equations). Đối với phần này, chúng ta chỉ sử dụng gói lệnh mice để thực hiện nhiều kỹ thuật khác nhau. Người bảo trì gói lệnh mice xuất bản một cuốn sách trực tuyến về gán số liệu bị thiếu, xem chi tiết hơn tại đây ((<https://stefvanbuuren.name/fimd/>).

Here is the code to load the mice package:

Đây là mã lệnh để tải gói lệnh mice:

### Mean Imputation

### Gán bằng giá trị trung bình

Sometimes if you are doing a simple analysis or you have strong reason to think you can assume MCAR, you can simply set missing numerical values to the mean of that variable. Perhaps we can assume that missing temperature measurements in our dataset were either MCAR or were just normal values. Here is the code to create a new variable that replaces missing temperature values with the mean temperature value in our dataset. However, in many situations replacing data with the mean can lead to bias, so be careful.

Thỉnh thoảng, nếu chúng ta thực hiện một phân tích đơn giản hoặc chúng ta có một lý do chính đáng để nghĩ rằng chúng ta có thể giả định số liệu bị thiếu là hoàn toàn ngẫu nhiên (MCAR), chúng ta có thể chỉ cần gán các giá trị bị thiếu bằng giá trị trung bình của biến số đó. Có lẽ chúng ta có thể giả định rằng các giá trị bị thiếu của thân nhiệt trong bộ số liệu là MCAR hoặc chỉ là các giá trị bình thường. Đây là mã lệnh để tạo một biến số mới để thay thế các giá trị thân nhiệt bị thiếu bằng giá trị thân nhiệt trung bình trong bộ số liệu. Tuy nhiên, trong nhiều tình huống, việc thay thế số liệu bằng giá trị trung bình có thể dẫn đến sai lệch, vì thế hãy cẩn thận.

You could also do a similar process for replacing categorical data with a specific value. For our dataset, imagine you knew that all observations with a missing value for their outcome (which can be “Death” or “Recover”) were actually people that died (note: this is not actually true for this dataset):

Chúng ta cũng có thể thực hiện một quy trình tương tự để thay thế số liệu của biến phân loại bằng một giá trị cụ thể. Đối với bộ số liệu linelist, hãy hình dung là chúng ta đã biết tất cả các quan sát có giá trị bị thiếu của biến số kết cục (mà có thể là “Tử vong” hoặc “Hồi phục”) thực sự là những người đã chết (lưu ý: điều này không thực sự đúng đối với bộ số liệu này):

### Regression imputation

### Gán bằng hồi quy

A somewhat more advanced method is to use some sort of statistical model to predict what a missing value is likely to be and replace it with the predicted value. Here is an example of creating predicted values for all the observations where temperature is missing, but age and fever are not, using simple linear regression using fever status and age in years as predictors. In practice you’d want to use a better model than this sort of simple approach.

Một phương pháp nâng hơi cao hơn là sử dụng một số loại mô hình thống kê để dự đoán giá trị có lẽ bị thiếu và thay thế nó bằng giá trị dự đoán. Đây là một ví dụ về việc tạo ra các giá trị dự đoán cho tất cả các quan sát bị thiếu của biến số thân nhiệt, nhưng biến số tuổi và biến số sốt thì không bị thiếu, bằng cách sử dụng hồi quy tuyến tính đơn giản với biến số dự đoán là tình trạng sốt và tuổi theo năm. Trong thực tế, chúng ta muốn sử dụng một mô hình tốt hơn kiểu tiếp cận đơn giản này.

Or, using the same modeling approach through the mice package to create imputed values for the missing temperature observations:

Hoặc sử dụng cùng phương pháp mô hình hóa thông qua gói lệnh mice để tạo ra các giá trị gán cho các quan sát thân nhiệt bị thiếu:

This is the same type of approach by some more advanced methods like using the missForest package to replace missing data with predicted values. In that case, the prediction model is a random forest instead of a linear regression. You can use other types of models to do this as well. However, while this approach works well under MCAR you should be a bit careful if you believe MAR or MNAR more accurately describes your situation. The quality of your imputation will depend on how good your prediction model is and even with a very good model the variability of your imputed data may be underestimated.

Đây là cùng cách tiếp cận bằng một số phương pháp nâng cao hơn như sử dụng gói lệnh missForest để thay thế giá trị bị thiếu bằng các giá trị dự đoán. Trong trường hợp đó, mô hình dự đoán là một random forest thay vì là một hồi quy tuyến tính. Chúng ta cũng có thể sử dụng các loại mô hình khác để làm điều này. Tuy nhiên, phương pháp này thực hiện tốt với số liệu loại MCAR, nhưng chúng ta nên cẩn thận một chút nếu chúng ta tin rằng loại số liệu của chúng ta chính xác hơn là MAR hoặc MNAR. Chất lượng của việc gán giá trị bị thiếu sẽ phụ thuộc vào mô hình dự đoán tốt như thế nào, và ngay cả khi với một mô hình dự đoán rất tốt, tính biến thiên của số liệu được gán có thể đưa đến một ước lượng non.

### LOCF and BOCF

Last observation carried forward (LOCF) and baseline observation carried forward (BOCF) are imputation methods for time series/longitudinal data. The idea is to take the previous observed value as a replacement for the missing data. When multiple values are missing in succession, the method searches for the last observed value.

LOCF (viết tắt của Last observation carried forward) và BOCF (viết tắt của baseline observation carried forward) là các phương pháp gán số liệu bị thiếu đối với số liệu chuỗi thời gian/số liệu dọc. Ý tưởng là lấy giá trị được quan sát trước đó để thay thế cho số liệu bị thiếu. Khi có nhiều giá trị bị thiếu liên tiếp, phương pháp này sẽ tìm kiếm giá trị được quan sát cuối cùng.

The fill() function from the **tidyr** package can be used for both LOCF and BOCF imputation (however, other packages such as **HMISC**, **zoo**, and **data.table** also include methods for doing this). To show the fill() syntax we’ll make up a simple time series dataset containing the number of cases of a disease for each quarter of the years 2000 and 2001. However, the year value for subsequent quarters after Q1 are missing so we’ll need to impute them. The fill() junction is also demonstrated in the [Pivoting data](#pivoting-data) page.

Hàm fill() từ gói lệnh **tidyr** có thể được sử dụng cho cả phương pháp gán LOCF và BOCF (tuy nhiên, các gói lệnh khác như **HMISC**, **zoo**, và **data.table** cũng có các phương pháp để làm điều này). Để hiển thị cú pháp của hàm fill(), chúng ta sẽ tạo một bộ số liệu chuỗi thời gian đơn giản mà bao gồm một số ca của một bệnh trong các quý của năm 2000 và 2001. Tuy nhiên, giá trị năm cho các quý tiếp theo sau Q1 bị thiếu, vì thế chúng ta cần gán các số liệu bị thiếu này. Hàm fill() cũng được trình bày rõ trong chương [Chuyển đổi số liệu](#pivoting-data).

Note: make sure your data are sorted correctly before using the fill() function. fill() defaults to filling “down” but you can also impute values in different directions by changing the .direction parameter. We can make a similar dataset where the year value is recorded only at the end of the year and missing for earlier quarters:

Lưu ý: cần đảm bảo số liệu được sắp xếp chính xác trước khi sử dụng hàm fill(). Mặc định của hàm fill() là điền “xuống” nhưng chúng ta cũng có thể gán giá trị bị thiếu theo các hướng khác nhau bằng cách thay đổi thông số .direction. Chúng ta có thể tạo một tập số liệu tương tự mà giá trị của năm chỉ được ghi nhận vào cuối năm và bị thiếu ở các quý trước đó:

In this example, LOCF and BOCF are clearly the right things to do, but in more complicated situations it may be harder to decide if these methods are appropriate. For example, you may have missing laboratory values for a hospital patient after the first day. Sometimes, this can mean the lab values didn’t change…but it could also mean the patient recovered and their values would be very different after the first day! Use these methods with caution.

Trong ví dụ này, rõ ràng LOCF và BOCF thực hiện đúng các công việc, nhưng trong những tình huống phức tạp hơn, có thể là khó khăn hơn để quyết định những phương pháp này có phù hợp hay không. Ví dụ: chúng ta có thể thiếu các giá trị xét nghiệm của một bệnh nhân nằm viện sau ngày đầu tiên. Đôi lúc, điều này có thể có nghĩa là các giá trị xét nghiệm không thay đổi ... nhưng cũng có thể là bệnh nhân đã hồi phục và những giá trị xét nghiệm của bệnh nhân này có thể là rất khác sau ngày đầu tiên! Hãy sử dụng những phương pháp này một cách thận trọng.

### Multiple Imputation

### Gán số liệu Nhiều lần

The online book we mentioned earlier by the author of the mice package (<https://stefvanbuuren.name/fimd/>) contains a detailed explanation of multiple imputation and why you’d want to use it. But, here is a basic explanation of the method:

Cuốn sách trực tuyến mà chúng tôi đã đề cập trước đây bởi tác giả của gói lệnh mice (<https://stefvanbuuren.name/fimd/>) bao gồm một giải thích chi tiết về gán số liệu nhiều lần và lý do tại sao chúng ta muốn sử dụng nó. Dưới đây là một giải thích cơ bản của phương pháp này:

When you do multiple imputation, you create multiple datasets with the missing values imputed to plausible data values (depending on your research data you might want to create more or less of these imputed datasets, but the mice package sets the default number to 5). The difference is that rather than a single, specific value each imputed value is drawn from an estimated distribution (so it includes some randomness). As a result, each of these datasets will have slightly different different imputed values (however, the non-missing data will be the same in each of these imputed datasets). You still use some sort of predictive model to do the imputation in each of these new datasets (mice has many options for prediction methods including Predictive Mean Matching, logistic regression, and random forest) but the mice package can take care of many of the modeling details.

Khi chúng ta thực hiện gán nhiều lần, chúng ta tạo nhiều bộ số liệu với các giá trị bị thiếu mà được gán các giá trị hợp lý (tùy thuộc vào số liệu của chúng ta, chúng ta có thể muốn tạo nhiều hay ít các bộ số liệu được gán này, nhưng gói lệnh mice đặt con số mặc định là 5). Sự khác biệt là thay vì một giá trị đơn lẻ cụ thể, mỗi giá trị được gán được rút ra từ một phân phối được ước tính (vì vậy nó bao gồm một số đặc tính ngẫu nhiên). Kết quả là, mỗi bộ số liệu này sẽ có các giá trị được gán khác nhau một chút (tuy nhiên, số liệu không bị thiếu sẽ giống nhau trong mỗi bộ số liệu được gán này). Chúng ta vẫn sử dụng một số loại mô hình dự đoán để thực hiện việc gán số liệu bị thiếu cho mỗi bộ số liệu mới này (gói lệnh mice có nhiều tùy chọn cho các phương pháp dự đoán bao gồm Predictive Mean Matching, logistic regression, and random forest) nhưng gói lệnh mice có thể đảm nhận nhiều chi tiết của mô hình.

Then, once you have created these new imputed datasets, you can apply then apply whatever statistical model or analysis you were planning to do for each of these new imputed datasets and pool the results of these models together. This works very well to reduce bias in both MCAR and many MAR settings and often results in more accurate standard error estimates.

Sau đó, một khi chúng ta đã tạo các bộ số liệu mới được gán giá trị bị thiếu này, chúng ta có thể áp dụng bất kỳ mô hình thống kê hoặc phân tích nào mà chúng ta dự định thực hiện cho mỗi bộ số liệu mới này và gộp kết quả của các mô hình này lại với nhau. Thao tác này rất tốt để giảm sai lệch đối với cả MCAR và MAR và thường cho các ước tính sai số chuẩn chính xác hơn.

Here is an example of applying the Multiple Imputation process to predict temperature in our linelist dataset using a age and fever status (our simplified model\_dataset from above):

Đây là một ví dụ vể việc áp dụng quy trình Gán Nhiều lần để dự đoán thân nhiệt trong bộ số liệu linelist bằng cách sử dụng tuổi và sốt (bộ số liệu mô hình đơn giản hóa bên trên):

Here we used the mice default method of imputation, which is Predictive Mean Matching. We then used these imputed datasets to separately estimate and then pool results from simple linear regressions on each of these datasets. There are many details we’ve glossed over and many settings you can adjust during the Multiple Imputation process while using the mice package. For example, you won’t always have numerical data and might need to use other imputation methods (you can still use the mice package for many other types of data and methods). But, for a more robust analysis when missing data is a significant concern, Multiple Imputation is good solution that isn’t always much more work than doing a complete case analysis.

Ở đây, chúng tôi sử dụng phương pháp gán mặc định của gói lệnh mice, đó là Predictive Mean Matching. Sau đó, chúng tôi sử dụng các bộ số liệu đã được gán này để ước tính riêng rẽ và gộp kết quả từ các hồi quy tuyến tính đơn giản trên mỗi bộ số liệu này. Có nhiều chi tiết mà chúng tôi đã phớt lờ và nhiều chổ mà chúng ta có thể điều chỉnh trong suốt quá trình Gán Nhiều lần trong khi sử dụng gói lệnh mice. Ví dụ: chúng ta không phải lúc nào cũng có số liệu dạng số và có thể cần các phương pháp gán khác (chúng ta vẫn có thể sử dụng gói lệnh mice cho nhiều loại số liệu và phương pháp khác). Để một phân tích thiết thực hơn khi số liệu bị thiếu là một mối quan tâm đáng chú ý, Gán số liệu Nhiều lần là một giải pháp tốt, nhưng không phải lúc nào cũng hiệu quả hơn việc thực hiện một phân tích của các ca hoàn chỉnh.

## Resources

## Nguồn

Vignette on the [naniar package](https://cran.r-project.org/web/packages/naniar/vignettes/getting-started-w-naniar.html)

Gallery of [missing value visualizations](https://cran.r-project.org/web/packages/naniar/vignettes/naniar-visualisation.html)

[Online book](https://stefvanbuuren.name/fimd/) about multiple imputation in R by the maintainer of the **mice** package