# Univariate and multivariable regression

# Hồi quy đơn biến và đa biến

This page demonstrates the use of **base** R regression functions such as glm() and the **gtsummary** package to look at associations between variables (e.g. odds ratios, risk ratios and hazard ratios). It also uses functions like tidy() from the **broom** package to clean-up regression outputs.

1. Univariate: two-by-two tables
2. Stratified: mantel-haenszel estimates
3. Multivariable: variable selection, model selection, final table
4. Forest plots

For Cox proportional hazard regression, see the [Survival analysis](#survival-analysis) page.

**NOTE:** We use the term multivariable to refer to a regression with multiple explanatory variables. In this sense a multivariate model would be a regression with several outcomes - see this [editorial](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3518362/) for detail

Trong chương này, chúng tôi trình bày cách sử dụng các hàm hồi quy trong **base** R như hàm glm()và gói lệnh **gtsummary** để xem xét các mối liên quan giữa các biến (ví dụ như tỷ số chênh, tỷ số nguy cơ, tỷ số rủi ro). Chúng tôi cũng trình bày cách sử dụng các hàm như tidy()trong gói lệnh **broom** để sắp xếp các kết quả hồi quy.

1. Phân tích đơn biến: bảng 2 x 2
2. Phân tích phân tầng: ước lượng của mantel-haenszel
3. Phân tích đa biến: lựa chọn biến số, lựa chọn mô hình, bảng cuối
4. Tạo biểu đồ chứa nhiều hệ số

Đối với hồi quy Cox, xem chương [Phân tích sống còn](#survival-analysis).

**CHÚ Ý:** Chúng tôi sử dụng thuật ngữ đa biến (multivariable) để nói đến một hồi quy có nhiều biến giải thích. Mô hình đa biến (multivariate) có nghĩa là một mô hình đa biến có nhiều biến kết cục – xem chi tiết trong [bài xã luận](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3518362/) này.

## Preparation

## Các bước chuẩn bị

### Load packages

### Tải các gói lệnh

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 hàm p\_load() trong 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.

### Clean data

### Làm sạch số liệu

#### Store explanatory variables

We store the names of the explanatory columns as a character vector. This will be referenced later.

Tên của các biến giải thích sẽ được lưu trữ dưới dạng một véc tơ ký tự. Véc tơ này sẽ được đề cập về sau.

#### Convert to 1’s and 0’s

#### Chuyển đổi sang số 1 và số 0

Below we convert the explanatory columns from “yes”/“no”, “m”/“f”, and “dead”/“alive” to **1 / 0**, to cooperate with the expectations of logistic regression models. To do this efficiently, used across() from **dplyr** to transform multiple columns at one time. The function we apply to each column is case\_when() (also **dplyr**) which applies logic to convert specified values to 1’s and 0’s. See sections on across() and case\_when() in the [Cleaning data and core functions page](#clean_across)).

Note: the “.” below represents the column that is being processed by across() at that moment.

Bên dưới, giá trị của các biến giải thích được chuyển đổi từ “có”/“không”, “nam”/“nữ” và “chết”/“sống” thành **1 / 0**, để hợp với các đặc tính của mô hình hồi quy logistic. Để thực hiện việc này một cách hiệu quả, sử dụng hàm across() từ gói lệnh **dplyr** để chuyển đổi nhiều biến cùng một lúc. Để áp dụng cho mỗi biến, dùng hàm case\_when() (cũng trong gói lệnh **dplyr**) để chuyển đổi các giá trị cụ thể thành 1 và 0. Xem các mục về across() và case\_when()trong chương [Làm sạch số liệu và các hàm quan trọng](#clean_across).

Chú ý: dấu “.” bên dưới đại diện cho biến đang được xử lý trong hàm across()tại thời điểm đó.

#### Drop rows with missing values

#### Bỏ các hàng có giá trị bị thiếu

To drop rows with missing values, can use the **tidyr** function drop\_na(). However, we only want to do this for rows that are missing values in the columns of interest.

The first thing we must to is make sure our explanatory\_vars vector includes the column age (age would have produced an error in the previous case\_when() operation, which was only for dichotomous variables). Then we pipe the linelist to drop\_na() to remove any rows with missing values in the outcome column or any of the explanatory\_vars columns.

Before running the code, the number of rows in the linelist is nrow(linelist).

The number of rows remaining in linelist is nrow(linelist).

Để bỏ các hàng có giá trị bị thiếu, dùng hàm drop\_na()trong gói lệnh **tidyr**. Tuy nhiên, chúng ta chỉ muốn thực hiện điều này cho các hàng có giá trị bị thiếu đối với các cột đang được quan tâm.

Trước hết, chúng ta phải đảm bảo rằng véc tơ explanatory\_vars bao gồm biến tuổi (biến tuổi này có thể tạo ra một lỗi trong thao tác của hàm case\_when()trước đó, mà chỉ dành cho biến nhị phân). Sau đó, đặt dấu dẫn (%>%) từ bộ số liệu linelist đến drop\_na()để bỏ các hàng có giá trị bị thiếu trong biến kết cục outcome hoặc bất kỳ cột nào của véc tơ explanatory\_vars.

Trước khi thực hiện các lệnh này, xem số hàng trong bộ số liệu linelist bằng hàm nrow(linelist).

Dùng trở lại hàm nrow(linelist)để xem số hàng còn lại trong bộ số liệu linelist.

## Univariate

## Phân tích đơn biến

Just like in the page on [Descriptive tables](https://epirhandbook.com/descriptive-tables.html), your use case will determine which R package you use. We present two options for doing univariate analysis:

* Use functions available in **base** R to quickly print results to the console. Use the **broom** package to tidy up the outputs.
* Use the **gtsummary** package to model and get publication-ready outputs

Cũng giống như chương [Bảng mô tả số liệu](https://epirhandbook.com/descriptive-tables.html), chúng ta cần xác định gói lệnh nào trong R mà chúng ta muốn sử dụng. Chúng tôi trình bày hai chọn lựa để thực hiện các phân tích đơn biến:

* Dùng hàm có sẵn trong **base** R để in nhanh kết quả ra bảng điều kiển. Dùng gói lệnh **broom** để làm gọn kết quả.
* Dùng gói lệnh **gtsummary** để lập mô hình và nhận được kết quả đầu ra đã sẵn sàng để công bố

### **base** R

#### Linear regression

#### Hồi quy tuyến tính

The **base** R function lm() perform linear regression, assessing the relationship between numeric response and explanatory variables that are assumed to have a linear relationship.

Provide the equation as a formula, with the response and explanatory column names separated by a tilde ~. Also, specify the dataset to data =. Define the model results as an R object, to use later.

You can then run summary() on the model results to see the coefficients (Estimates), P-value, residuals, and other measures.

Alternatively you can use the tidy() function from the **broom** package to pull the results in to a table. What the results tell us is that for each year increase in age the height increases by 3.5 cm and this is statistically significant.

You can then also use this regression to add it to a **ggplot**, to do this we first pull the points for the observed data and the fitted line in to one data frame using the augment() function from **broom**.

It is also possible to add a simple linear regression straight straight in **ggplot** using the geom\_smooth() function.

See the Resource section at the end of this chapter for more detailed tutorials.

Hàm lm()trong **base** R thực hiện hồi quy tuyến tính để đánh giá mối quan hệ giữa biến đáp ứng (biến liên tục) và các biến giải thích mà được giả định có mối quan hệ tuyến tính.

Cung cấp phương trình dưới dạng công thức với tên của biến đáp ứng và các biến giải thích được phân tách bằng dấu ngã ~. Ngoài ra, cụ thể bộ số liệu với data =. Kết quả của mô hình được nhận biết dưới dạng đối tượng của R để sử dụng về sau.

Sau đó tóm tắt kết quả của mô hình bằng hàm summary()để xem các hệ số (các ước tính), giá trị p, phần dư và các đo lường khác.

Ngoài ra, có thể dùng hàm tidy()trong gói lệnh **broom** để xuất kết quả vào trong một bảng. Kết quả bên dưới cho chúng ta biết khi tăng một tuổi thì thì chiều cao tăng 3,5 cm và mối quan hệ này có ý nghĩa thống kê.

Sau đó, có thể sử dụng kết quả hồi quy này để đưa vào **ggplot.** Để thực hiện điều này, trước tiên chúng ta đưa các giá trị quan sát và đường hiệu chỉnh vào một khung số liệu bằng cách dùng hàm augment()trong gói lệnh **broom**.

Cũng có thể vẽ đường hồi quy tuyến tính đơn bằng hàm geom\_smooth()trong gói lệnh **ggplot**.

Xem thêm các hướng dẫn chi tiết trong mục Nguồn ở cuối chương này.

#### Logistic regression

#### Hồi quy logistic

The function glm() from the **stats** package (part of **base** R) is used to fit Generalized Linear Models (GLM).

Hàm glm()trong gói lệnh **stats** (một phần của **base** R) được sử dụng để fit (chọn mô hình dự đoán tối ưu dựa trên số liệu quan sát) đối với Mô hình Tuyến tính Tổng quát (GLM).

glm() can be used for univariate and multivariable logistic regression (e.g. to get Odds Ratios). Here are the core parts:

* formula = The model is provided to glm() as an equation, with the outcome on the left and explanatory variables on the right of a tilde ~.
* family = This determines the type of model to run. For logistic regression, use family = "binomial", for poisson use family = "poisson". Other examples are in the table below.
* data = Specify your data frame

glm() có thể được sử dụng cho cả hồi quy logistic đơn biến và đa biến (ví dụ như để lấy tỷ số chênh). Đây là những phần chính:

* formula = Mô hình được cung cấp cho glm() dưới dạng một phương trình với biến kết cục ở bên trái và biến giải thích ở bên phải dấu ngã ~.
* family = Xác định loại mô hình sẽ thực hiện. Đối với hồi quy logistic, sử dụng family = "binomial", đối với hồi quy poisson sử dụng family = "poisson". Các ví dụ khác có trong bảng bên dưới.
* data = Cụ thể bộ số liệu

If necessary, you can also specify the link function via the syntax family = familytype(link = "linkfunction")). You can read more in the documentation about other families and optional arguments such as weights = and subset = (?glm).

Nếu cần, có thể cụ thể hàm liên kết bằng cú pháp family = familytype(link = "linkfunction")). Đọc thêm tài liệu về các họ khác và các đối số tùy chọn như weights = và subset = (?glm).

| **Họ** | **Hàm liên kết mặc định** |
| --- | --- |
| "binomial" | (link = "logit") |
| "gaussian" | (link = "identity") |
| "Gamma" | (link = "inverse") |
| "inverse.gaussian" | (link = "1/mu^2") |
| "poisson" | (link = "log") |
| "quasi" | (link = "identity", variance = "constant") |
| "quasibinomial" | (link = "logit") |
| "quasipoisson" | (link = "log") |

When running glm() it is most common to save the results as a named R object. Then you can print the results to your console using summary() as shown below, or perform other operations on the results (e.g. exponentiate).

Khi thực hiện glm(), phổ biến nhất là lưu kết quả dưới dạng một đối tượng của R được đặt tên. Sau đó, có thể xuất kết quả ra bảng điều khiển bằng cách sử dụng hàm summary() như được trình bày bên dưới, hoặc thực hiện các thao tác khác từ kết quả (ví dụ như lấy lũy thừa).

If you need to run a negative binomial regression you can use the **MASS** package; the glm.nb() uses the same syntax as glm(). For a walk-through of different regressions, see the [UCLA stats page](https://stats.idre.ucla.edu/other/dae/).

Nếu cần thực hiện một hồi quy nhị thức nghịch/âm, có thể sử dụng gói lệnh **MASS**. Hàm glm.nb()sử dụng cùng cú pháp như glm(). Để xem qua các hồi quy khác, xem trên trang thống kê của [UCLA](https://stats.idre.ucla.edu/other/dae/).

#### Univariate glm()

#### Phân tích đơn biến sử dụng glm()

In this example we are assessing the association between different age categories and the outcome of death (coded as 1 in the Preparation section). Below is a univariate model of outcome by age\_cat. We save the model output as model and then print it with summary() to the console. Note the estimates provided are the log odds and that the baseline level is the first factor level of age\_cat (“0-4”).

Trong ví dụ này, chúng tôi đánh giá mối liên quan giữa nhóm tuổi và biến kết cục tử vong (được mã hóa là 1 trong phần chuẩn bị). Bên dưới là một mô hình đơn biến của biến kết cục outcome theo age\_cat. Chúng tôi lưu kết quả đầu ra được đặt tên model và sau đó xuất kết quả đến bảng điều kiển bằng hàm summary(). Lưu ý, các ước tính được tạo ra là các giá trị lôgarít của tỷ số chênh và giá trị tham chiếu là giá trị đầu tiên của biến age\_cat (“0-4”).

To alter the baseline level of a given variable, ensure the column is class Factor and move the desired level to the first position with fct\_relevel() (see page on [Factors](#factors)). For example, below we take column age\_cat and set “20-29” as the baseline before piping the modified data frame into glm().

Để thay đổi giá trị tham chiếu của một biến Factor và chuyển giá trị mong muốn lên vị trí đầu tiên, dùng hàm fct\_relevel() (xem chương [Factors](#factors)). Ở ví dụ bên dưới, chúng tôi lấy biến age\_cat và đặt nhóm tuổi “20-29” làm giá trị tham chiếu trước khi chuyển số liệu đã sửa đổi vào hàm glm().

#### Printing results

#### Xuất kết quả

For most uses, several modifications must be made to the above outputs. The function tidy() from the package **broom** is convenient for making the model results presentable.

Đối với hầu hết các mục đích sử dụng, kết quả đầu ra cần phải có một số sửa đổi. Hàm làm gọn tidy() trong gói lệnh **broom** có những tiện lợi để hiển thị kết quả của mô hình.

Here we demonstrate how to combine model outputs with a table of counts.

1. Get the exponentiated log odds ratio estimates and confidence intervals by passing the model to tidy() and setting exponentiate = TRUE and conf.int = TRUE.

Below is the outputted tibble model:

Ở đây, chúng tôi trình bày cách để kết hợp các kết quả đầu ra của mô hình vào trong một bảng

1. Lấy *lũy thừa* lôgarít của các tỷ số chênh và khoảng tin cậy bằng cách đưa mô hình vào hàm tidy() và thiết lập lũy thừa exponentiate = TRUE và khoảng tin cậy conf.int = TRUE.

Bên dưới là bảng kết quả đầu ra của model:

1. Combine these model results with a table of counts. Below, we create the a counts cross-table with the tabyl() function from **janitor**, as covered in the [Descriptive tables](#descriptive-tables) page.

Here is what this counts\_table data frame looks like:

2. Kết hợp các kết quả của mô hình vào trong một bảng. Bên dưới, chúng tôi tạo một bảng đếm bằng hàm tabyl() từ **Janitor** như được đề cập trong chương [Các bảng mô tả số liệu](#descriptive-tables).

Đây là hiển thị cấu trúc số liệu của counts\_table:

Now we can bind the counts\_table and the model results together horizontally with bind\_cols() (**dplyr**). Remember that with bind\_cols() the rows in the two data frames must be aligned perfectly. In this code, because we are binding within a pipe chain, we use . to represent the piped object counts\_table as we bind it to model. To finish the process, we use select() to pick the desired columns and their order, and finally apply the **base** R round() function across all numeric columns to specify 2 decimal places.

Bây giờ chúng ta có thể nối bảng counts\_table và kết quả của mô hình model lại với nhau theo chiều ngang bằng hàm nối cột bind\_cols() (**dplyr**). Hãy nhớ rằng đối với hàm bind\_cols(), các hàng trong hai cấu trúc dữ liệu trên phải được căn chỉnh hoàn hảo. Trong đoạn mã này, bởi vì chúng tôi kết nối vào trong một chuổi, chúng tôi sử dụng dấu ‘.’ để đại diện cho đối tượng được nối trong bảng đếm counts\_table khi chúng tôi nối nó với kết quả mô hình model. Để kết thúc kết nối này, chúng tôi sử dụng hàm select()để chọn các cột mong muốn và thứ tự của nó, và cuối cùng sử dụng hàm làm tròn số round() trong **base** R để làm tròn với hai số thập phân cho tất cả các cột.

Here is what the combined data frame looks like, printed nicely as an image with a function from **flextable**. The [Tables for presentation](#tables-for-presentation) explains how to customize such tables with **flextable**, or or you can use numerous other packages such as **knitr** or **GT**.

Đây là hiển thị của cấu trúc đã được kết hợp, nó được xuất gọn gẽ như một hình bằng một hàm trong gói lệnh **flextable**. Trong chương [Trình bày bảng](#tables-for-presentation) giải thích cách tùy chỉnh các bảng như vậy bằng **flextable,** hoặc có thể sử dụng các gói lệnh khác như **knitr** hoặc **GT.**

#### Looping multiple univariate models

#### Vòng lặp của các mô hình đơn biến

Below we present a method using glm() and tidy() for a more simple approach, see the section on **gtsummary**.

Bên dưới chúng tôi trình bày một phương pháp sử dụng glm() và tidy()để có một cách tiếp cận đơn giản hơn, xem thêm ở phần **gtsummary**.

To run the models on several exposure variables to produce univariate odds ratios (i.e. not controlling for each other), you can use the approach below. It uses str\_c() from **stringr** to create univariate formulas (see [Characters and strings](#characters-and-strings)), runs the glm() regression on each formula, passes each glm() output to tidy() and finally collapses all the model outputs together with bind\_rows() from **tidyr**. This approach uses map() from the package **purrr** to iterate - see the page on [Iteration, loops, and lists](#iteration-loops-and-lists) for more information on this tool.

1. Create a vector of column names of the explanatory variables. We already have this as explanatory\_vars from the Preparation section of this page.
2. Use str\_c() to create multiple string formulas, with outcome on the left, and a column name from explanatory\_vars on the right. The period . substitutes for the column name in explanatory\_vars.
3. Pass these string formulas to map() and set ~glm() as the function to apply to each input. Within glm(), set the regression formula as as.formula(.x) where .x will be replaced by the string formula defined in the step above. map() will loop over each of the string formulas, running regressions for each one.
4. The outputs of this first map() are passed to a second map() command, which applies tidy() to the regression outputs.
5. Finally the output of the second map() (a list of tidied data frames) is condensed with bind\_rows(), resulting in one data frame with all the univariate results.

Để thực hiện các mô hình cho một số biến giải thích và cho ra các tỷ số chênh trong phân tích đơn biến (nghĩa là không có kiểm soát lẫn nhau), chúng ta có thể sử dụng phương pháp bên dưới. Sử dụng hàm str\_c() từ gói lệnh **stringr** để tạo ra các công thức cho phân tích đơn biến (xem chương [Ký tự và chuổi](#characters-and-strings)), thực hiện hàm glm() cho mỗi công thức, chuyển mỗi kết quả đầu ra của glm() đến hàm tidy(), và cuối cùng thu gọn lại tất các kết quả đầu ra của mô hình bằng hàm nối dòng bind\_rows() từ **tidyr**. Phương pháp này sử dụng hàm map() từ gói lệnh **purrr** để lặp – xem chương [Lặp, vòng lặp và danh sách](#iteration-loops-and-lists) để biết thêm thông tin về công cụ này.

1. Tạo một véc tơ tên các cột của biến giải thích. Chúng ta đã tạo biến này explanatory\_vars trong phần chuẩn bị của chương này.
2. Sử dụng hàm str\_c() để tạo các công thức chuỗi với biến kết cục outcome ở bên trái và tên một cột của véc tơ explanatory\_vars ở bên phải. Dấu chấm ‘.’ trong hàm này thay thế cho tên cột trong véc tơ explanatory\_vars.
3. Đưa các công thức chuỗi này vào hàm map() và đặt ~glm() làm hàm ứng dụng cho mỗi đầu vào. Bên trong hàm glm(), thiết lập công thức hồi quy as.formula(.x), trong đó ‘.x’ sẽ được thay thế bằng các công thức chuỗi đã được tạo bên trên. Hàm map() sẽ lặp từng công thức chuỗi và thực hiện hồi quy cho từng công thức.
4. Kết quả đầu ra của hàm map() đầu tiên sẽ được chuyển đến hàm map() thứ hai mà sử dụng hàm tidy() để làm gọn các kết quả đầu ra.
5. Cuối cùng, kết quả đầu ra của hàm map() thứ hai (một danh sách của các cấu trúc số liệu đã được làm gọn) được tóm tắt bằng hàm nối dòng bind\_rows() để cho một khung số liệu với tất cả các kết quả đơn biến.

This time, the end object models is longer because it now represents the combined results of several univariate regressions. Click through to see all the rows of model.

Lúc này, kết quả xuất ra của models dài hơn bởi vì kết quả bây giờ bao gồm các kết quả đầu ra của một số hồi quy đơn biến. Nhấp nút tiếp theo để xem tất cả các hàng của models.

As before, we can create a counts table from the linelist for each explanatory variable, bind it to models, and make a nice table. We begin with the variables, and iterate through them with map(). We iterate through a user-defined function which involves creating a counts table with **dplyr** functions. Then the results are combined and bound with the models model results.

Như lúc trước, chúng ta có thể tạo một bảng đếm từ bộ số liệu linelist cho mỗi biến giải thích, kết nối với models, và tạo một bảng đẹp. Chúng ta bắt đầu với các biến giải thích này, và lặp lại các biến này thông qua hàm map(). Chúng ta lặp lại qua một hàm do người dùng tạo ra mà liên quan đến việc tạo ra một bảng đếm bằng cách dùng các hàm trong gói lệnh **dplyr**. Sau đó, kết quả được kết nối trình tự với kết quả của mô hình models.

Below is what the data frame looks like. See the page on [Tables for presentation](https://epirhandbook.com/tables-for-presentation.html" \l "tables-for-presentation) for ideas on how to further convert this table to pretty HTML output (e.g. with **flextable**).

Bên dưới là cấu trúc số liệu kết nối được tạo ra. Xem chương [Trình bày bảng](#tables-for-presentation) để có thêm ý tưởng về cách chuyển đổi bảng số liệu này thành một bảng đẹp trên HTML (ví dụ như với gói lệnh **flextable**).

### **gtsummary** package

### Gói lệnh gtsummary

Below we present the use of tbl\_uvregression() from the **gtsummary** package. Just like in the page on [Descriptive tables](https://epirhandbook.com/descriptive-tables.html), **gtsummary** functions do a good job of running statistics and producing professional-looking outputs. This function produces a table of univariate regression results.

Bên dưới chúng tôi trình bày cách sử dụng hàm tbl\_uvregression() trong gói lệnh **gtsummary**. Cũng giống như trong chương [Bảng mô tả số liệu](https://epirhandbook.com/descriptive-tables.html), các hàm trong **gtsummary** thực hiện tốt các thống kê và xuất ra các kết quả khá chuyên nghiệp. Hàm này xuất ra một bảng kết quả của hồi quy đơn biến.

We select only the necessary columns from the linelist (explanatory variables and the outcome variable) and pipe them into tbl\_uvregression(). We are going to run univariate regression on each of the columns we defined as explanatory\_vars in the data Preparation section (gender, fever, chills, cough, aches, vomit, and age\_cat).

Chúng ta chỉ chọn các cột cần thiết từ bộ số liệu linelist (các biến giải thích và biến kết cục) và đưa các biến này vào hàm tbl\_uvregression(). Chúng ta sẽ thực hiện hồi quy đơn biến cho mỗi cột như được xác định trong véc tơ explanatory\_vars trong phần Chuẩn bị (giới tính/gender, sốt/fever, ớn lạnh/chills, ho/cough, đau nhức/aches, nôn ói/vomit, và nhóm tuổi/ age\_cat).

Within the function itself, we provide the method = as glm (no quotes), the y = outcome column (outcome), specify to method.args = that we want to run logistic regression via family = binomial, and we tell it to exponentiate the results.

Trong hàm này, chúng ta cung cấp thêm phương pháp thực hiện method = là glm (không có dấu ngoặc kép), biến kết cục y = cột kết quả (biến outcome), cụ thể method.args = mà chúng ta muốn thực hiện hồi quy logistic qua family = binomial, và lấy lũy thừa của kết quả.

The output is HTML and contains the counts

Kết quả đầu ra dưới dạng HTML và chứa cột đếm

There are many modifications you can make to this table output, such as adjusting the text labels, bolding rows by their p-value, etc. See tutorials [here](http://www.danieldsjoberg.com/gtsummary/articles/tbl_regression.html) and elsewhere online.

Chúng ta có thể sửa đổi đối với kết quả đầu ra của bảng này, ví dụ như điều chỉnh các nhãn, tô đậm các hàng theo giá trị p, .v.v. Xem hướng dẫn [tại đây](http://www.danieldsjoberg.com/gtsummary/articles/tbl_regression.html) và các tài liệu trực tuyến khác.

## Stratified

## Phân tích phân tầng

Stratified analysis is currently still being worked on for **gtsummary**, this page will be updated in due course.

Hiện tại, phân tích phần tầng đang được xây dựng cho gói lênh **gtsummary**, phần này sẽ được cập nhật trong thời gian thích hợp.

## Multivariable

## Phân tích đa biến

For multivariable analysis, we again present two approaches:

* glm() and tidy()
* **gtsummary** package

The workflow is similar for each and only the last step of pulling together a final table is different.

Đối với phân tích đa biến, chúng tôi trình bày hai cách tiếp cận:

* Dùng hàm glm() và tidy()
* Gói lệnh **gtsummary**

Quy trình thực hiện khá tương tự và chỉ khác ở bước cuối cùng để kết nối kết quả lại với nhau.

### Conduct multivariable

### Thực hiện phân tích đa biến

Here we use glm() but add more variables to the right side of the equation, separated by plus symbols (+).

Ở đây chúng tôi sử dụng hàm glm() nhưng thêm nhiều biến hơn vào bên phải của phương trình và được phân tách với nhau bằng dấu cộng (+).

To run the model with all of our explanatory variables we would run:

If you want to include two variables and an interaction between them you can separate them with an asterisk \* instead of a +. Separate them with a colon : if you are only specifying the interaction. For example:

Optionally, you can use this code to leverage the pre-defined vector of column names and re-create the above command using str\_c(). This might be useful if your explanatory variable names are changing, or you don’t want to type them all out again.

Để thực hiện mô hình với tất cả cac biến giải thích, chúng ta thực hiện lệnh sau:

Nếu muốn bao gồm hai biến và tương tác của hai biến này, chúng ta có thể phân tách chúng bằng dấu hoa thị ‘\*’ thay cho dấu ‘+’. Nếu chúng ta chỉ muốn cụ thể sự tương tác, phân tách chúng bằng dấu hai chấm ‘:’. Ví dụ:

Một tùy chọn khác, chúng ta có thể sử dụng đoạn mã này để sử dụng một véc tơ đã được định nghĩa trước của các cột và tạo lại lệnh trên bằng cách sử dụng hàm str\_c(). Điều này có thể hữu ích nếu chúng ta thay đổi tên các biến giải thích hoặc không muốn gõ lại tất cả.

#### Building the model

#### Xây dựng mô hình

You can build your model step-by-step, saving various models that include certain explanatory variables. You can compare these models with likelihood-ratio tests using lrtest() from the package **lmtest**, as below:

**NOTE:** Using **base** anova(model1, model2, test = "Chisq) produces the same results

Chúng ta có thể xây dựng mô hình theo từng bước, lưu các mô hình khác nhau với một số biến giải thích. Chúng ta có thể sử dụng kiểm định tỷ số khả dĩ để so sánh các mô hình này bằng cách sử dụng hàm lrtest() từ gói lệnh **lmtest** như bên dưới:

CHÚ Ý: sử dụng hàm anova(model1, model2, test = "Chisq) trong **base** cũng cho kết quả tương tự.

Another option is to take the model object and apply the step() function from the **stats** package. Specify which variable selection direction you want use when building the model.

Một tùy chọn khác là lấy đối tượng của mô hình và sử dụng hàm step() từ gói lệnh **stats**. Chỉ rõ hướng lựa chọn biến mà chúng ta muốn sử dụng khi xây dựng mô hình.

You can also turn off scientific notation in your R session, for clarity:

As described in the section on univariate analysis, pass the model output to tidy() to exponentiate the log odds and CIs. Finally we round all numeric columns to two decimal places. Scroll through to see all the rows.

Here is what the resulting data frame looks like:

Để hiển thị rõ số, chúng ta có thể tắt ký hiệu khoa học trong R bằng lệnh sau:

Như được mô tả trong phần phân tích đơn biến, chuyển kết quả đầu ra của mô hình vào hàm tidy()để lấy lũy thừa cho các hệ số và khoảng tin cậy (CIs). Cuối cùng, làm tròn tất cả các cột số với hai số thập phân. Kéo qua để xem tất cả các hàng.

Đây là hiển thị kết quả dưới dạng khung số liệu:

### Combine univariate and multivariable

### Kết nối kết quả phân tích đơn biến và đa biến

#### Combine with **gtsummary**

#### ****Kết nối bằng gói lệnh** gtsummary**

The **gtsummary** package provides the tbl\_regression() function, which will take the outputs from a regression (glm() in this case) and produce an nice summary table.

Let’s see the table:

Hàm tbl\_regression()trong gói lệnh gtsummary sẽ lấy kết quả đầu tra từ một hồi quy (hàm glm()trong trường hợp này) và tạo ra một bảng tóm tắt đẹp.

Hãy xem bảng sau:

You can also combine several different output tables produced by **gtsummary** with the tbl\_merge() function. We now combine the multivariable results with the **gtsummary** univariate results that we created [above](#reg_gt_uni):

Chúng ta cũng có thể kết hợp một số bảng kết quả đầu ra bằng cách dùng hàm tbl\_merge() trong gói lệnh **gtsummary**. Bây giờ chúng ta kết nối các kết quả đa biến với kết quả đơn biến đã được tạo [bên trên](#reg_gt_uni) **bằng gói lệnh gtsummary:**

#### Combine with **dplyr**

#### Kết nối bằng gói lệnh dplyr

An alternative way of combining the glm()/tidy() univariate and multivariable outputs is with the **dplyr** join functions.

* Join the univariate results from earlier (univ\_tab\_base, which contains counts) with the tidied multivariable results mv\_tab\_base
* Use select() to keep only the columns we want, specify their order, and re-name them
* Use round() with two decimal places on all the column that are class Double

Một cách khác để kết nối các kết quả đơn biến và đa biến từ các hàm glm()/tidy() bằng cách sử dụng các hàm kết nối từ gói lệnh **dplyr**.

* Kết nối kết quả đơn biến trước đó (univ\_tab\_base, chứa được các cột đếm) với kết quả đa biến đã được làm gọn mv\_tab\_base
* Sử dụng hàm lựa chọn biến select() để giữ lại, sắp xếp lại thứ tự và đặt lại tên các cột mà chúng ta muốn.
* Dùng hàm round() để làm tròn tất cả các cột với hai số thập phân

## Forest plot

## Tạo biểu đồ chứa nhiều hệ số

This section shows how to produce a plot with the outputs of your regression. There are two options, you can build a plot yourself using **ggplot2** or use a meta-package called **easystats** (a package that includes many packages).

Phần này hướng dẫn cách tạo ra một biểu đồ của các kết quả hồi quy. Có hai lựa chọn để tạo biểu đồ, chúng ta có thể tự tạo một biểu đồ bằng cách sử dụng gói lệnh **ggplot2** hoặc sử dụng một gói meta gọi là **easystats** (một gói lệnh bao gồm nhiều gói lệnh)

See the page on [ggplot basics](#ggplot-basics) if you are unfamiliar with the **ggplot2** plotting package.

Nếu chưa quen thuộc với gói lệnh tạo biểu đồ **ggplot2**, xem thêm chương [ggplot cơ bản](#ggplot-basics).

### **ggplot2** package

### Gói lệnh ggplot2

You can build a forest plot with ggplot() by plotting elements of the multivariable regression results. Add the layers of the plots using these “geoms”:

* estimates with geom\_point()
* confidence intervals with geom\_errorbar()
* a vertical line at OR = 1 with geom\_vline()

Before plotting, you may want to use fct\_relevel() from the **forcats** package to set the order of the variables/levels on the y-axis. ggplot() may display them in alpha-numeric order which would not work well for these age category values (“30” would appear before “5”). See the page on [Factors](#factors) for more details.

Với hàm ggplot(), chúng ta có thể tạo một biểu đồ chứa nhiều hệ số bằng cách vẽ các thành phần của kết quả hồi quy đa biến. Thêm các lớp của biều đồ bằng cách sử dụng “geoms”:

* Các ước lượng bằng hàm geom\_point()
* Các khoảng tin cậy bằng hàm geom\_errorbar()
* Đường thẳng đứng ở vị trí OR = 1 bằng hàm geom\_vline()

Trước khi tạo biểu đồ, chúng ta sử dụng hàm fct\_relevel() từ gói lệnh **forcats** để đặt thứ tự các biến trên trục tung. Hàm ggplot() cho phép hiển thị theo thứ tự chữ-số mà có thể không hiển thị tốt cho các giá trị của biến tuổi (“30” có thể hiển thị trước “5”). Xem chương [Factors](#factors) để biết thêm chi tiết.

### **easystats** packages

### Gói lệnh easystats

An alternative, if you do not want to the fine level of control that **ggplot2** provides, is to use a combination of **easystats** packages.

The function model\_parameters() from the **parameters** package does the equivalent of the **broom** package function tidy(). The **see** package then accepts those outputs and creates a default forest plot as a ggplot() object.

Lựa chọn thứ hai là sử dụng một sự kết hợp của các gói lệnh trong **easystats**, nếu chúng ta không muốn mức độ kiểm soát chặc chẽ mà gói lệnh **ggplot2** cung cấp.

Hàm model\_parameters() từ gói lệnh **parameters** thực hiện tương đương với hàm tidy()từ gói lệnh **broom**. |Sau đó, gói lệnh **see** chấp nhận các kết quả đầu ra và tạo một biểu đồ mặc định chứa nhiều hệ số giống như cho một đối tượng trong hàm ggplot().

## Resources

## Nguồn

The content of this page was informed by these resources and vignettes online:

Nội dung của chương này thu thập từ các nguồn sau:

[Linear regression in R](https://www.datacamp.com/community/tutorials/linear-regression-R)

[gtsummary](http://www.danieldsjoberg.com/gtsummary/articles/tbl_regression.html)

[UCLA stats page](https://stats.idre.ucla.edu/other/dae/)

[sthda stepwise regression](http://www.sthda.com/english/articles/36-classification-methods-essentials/150-stepwise-logistic-regression-essentials-in-r/)