Source: <https://github.com/PsyTeachR/msc-data-skills/blob/master/09-glm.Rmd>

Date: 2020-04-03

Authors:

Lisa De Bruine [aut]

Cédric Batailler [aut]

Nicolas Verger [relecture]

# Introduction to GLM {#glm}

# Introduction aux GLM {#glm}

<img src="images/memes/linear\_regression.jpg" class="meme right">

<img src="https://github.com/PsyTeachR/msc-data-skills/blob/master/images/memes/linear\_regression.jpg?raw=true" class="meme right">

## Learning Objectives

## Objectifs du cours

### Basic

### Débutant

1. Define the [components](#glm-components) of the GLM

2. [Simulate data](#sim-glm) using GLM equations

3. Identify the model parameters that correspond to the data-generation parameters

4. Understand and plot [residuals](#residuals)

5. [Predict new values](#predict) using the model

6. Explain the differences among [coding schemes](#coding-schemes)

1. Définir les [paramètres](#glm-components) d’un GLM

2 [Simuler des données](#sim-glm) en utilisant les équations d’un GLM

3. Identifier les paramètres d’un modèle correspondant aux paramètres sous-tendant la génération de données

4. Comprendre et faire un graphique de [résidus](#residuals)

5. [Prédire de nouvelles observations](#predict) à partir du modèle GLM

6. Expliquer les différences dues au [codage des variables](#coding-schemes)

### Intermediate

### Intermédiaire

7. Demonstrate the [relationships](#test-rels) among two-sample t-test, one-way ANOVA, and linear regression

8. Given data and a GLM, [generate a decomposition matrix](#decomp) and calculate sums of squares, mean squares, and F ratios for a one-way ANOVA

7. Démontrer les [relations](#test-rels) entre test \_t\_ pour échantillons indépendant, ANOVA, et régression linéaire

8. A partir de données et d’un modèle GLM, [générer la matrice de décomposition](decomp) et calculer la somme des carrés, les carrés moyens et les rapports de F pour une ANOVA à un facteur contrôlé

## Resources

## Ressources

\* [Stub for this lesson](stubs/9\_glm.Rmd)

\* [Jeff Miller and Patricia Haden, Statistical Analysis with the Linear Model (free online textbook)](http://www.otago.ac.nz/psychology/otago039309.pdf)

\* [lecture slides introducing the General Linear Model](slides/08\_glm\_slides.pdf)

\* [GLM shiny app](http://rstudio2.psy.gla.ac.uk/Dale/GLM)

\* [F distribution](<http://rstudio2.psy.gla.ac.uk/Dale/fdist>)

\* [Fichier source original pour cette leçon](<https://github.com/PsyTeachR/msc-data-skills/blob/master/stubs/9_glm.Rmd>)

\* [Statistical Analysis with the Linear Model de Jeff Miller et Patricia Haden (manuel en ligne gratuit)](http://www.otago.ac.nz/psychology/otago039309.pdf)

\* [Diapositives du cours sur le modèle linéaire général](<https://github.com/PsyTeachR/msc-data-skills/blob/master/slides/08_glm_slides.pdf>)

\* [shiny app GLM](http://rstudio2.psy.gla.ac.uk/Dale/GLM)

\* [distribution du F](http://rstudio2.psy.gla.ac.uk/Dale/fdist)

## Setup

## Configuration

You'll need the following packages.

Vous aurez besoin des packages suivants.

```{r, results = 'hide', warning = FALSE, message = FALSE}

# libraries needed for these examples

library(tidyverse)

library(broom)

set.seed(30250) # makes sure random numbers are reproducible

# packages nécessaires pour les exemples

library(tidyverse)

library(broom)

set.seed(30250) # afin que les données aléatoires générées soient reproductibles

```

## GLM

## GLM

### What is the GLM?

### Qu’est-ce que le GLM ?

The General Linear Model (GLM) a general mathematical framework for expressing relationships among variables that can express or test linear relationships between a numerical \*\*dependent variable\*\* and any combination of categorical or continuous \*\*independent variables\*\*.

Le Modèle Linéaire Généralisé (General Linear Model ; GLM) est un outil mathématique dans lequel il est possible d’exprimer des relations entre variables. Ces relations permettent de tester les relations linéaires existantes entre une \*\*variable dépendante\*\* continue et une combinaison de \*\*variables dépendantes\*\* (qu’elles soient continues ou catégorielles).

### Components {#glm-components}

### Paramètres {#glm-components}

There are some mathematical conventions that you need to learn to understand the equations representing linear models. Once you understand those, learning about the GLM will get much easier.

Certaines conventions mathématiques permettent de mieux comprendre les équations représentant les modèles linéaires. Une fois ces conventions assimilées, il sera plus facile d'apprendre le GLM.

| Component of GLM | Notation |

|------------------|-------------------------------|

| Dependent Variable (DV) | $Y$ |

| Grand Average | $\mu$ (the Greek letter "mu") |

| Main Effects | $A, B, C, \ldots$ |

| Interactions | $AB, AC, BC, ABC, \ldots$ |

| Random Error | $S(Group)$ |

| Paramètres du GLM | Paramètres [EN] | Notation |

|--------------------------|-------------------------|---------------------------------|

| Variable dépendante (VD) | Dependent Variable (DV) | $Y$ |

| Moyenne générale | Grand Average | $\mu$ (la lettre grecque "mu") |

| Effets principaux | Main Effects | $A, B, C, \ldots$ |

| Interactions | Interactions | $AB, AC, BC, ABC, \ldots$ |

| Erreur aléatoire | Random Error | $S(Group)$ |

<img src="images/memes/confused-math-lady.png" class="meme right">

<img src="images/memes/confused-math-lady.png" class="meme right">

The linear equation predicts the dependent variable ($Y$) as the sum of the grand average value of $Y$ ($\mu$, also called the intercept), the main effects of all the predictor variables ($A+B+C+ \ldots$), the interactions among all the predictor variables ($AB, AC, BC, ABC, \ldots$), and some random error ($S(Group)$). The equation for a model with two predictor variables ($A$ and $B$) and their interaction ($AB$) is written like this:

L’équation linéaire permet de prédire la variable dépendante ($Y$) à partir de la somme de la moyenne générale, des effets principaux de chacune des variables ($A, B, C, \ldots$), de l’interaction entre chacune des variables ($AB, AC, BC, ABC, \ldots$) et de l’erreur résiduelle ($S(Group)$). Pour un modèle utilisant deux prédicteurs et l’interaction entre les deux, l’équation linéaire s’écrira ainsi :

$Y$ ~ $\mu+A+B+AB+S(Group)$

$Y$ ~ $\mu+A+B+AB+S(Group)$

Don't worry if this doesn't make sense until we walk through a concrete example.

Ne vous inquiétez pas si tout n’est pas parfaitement clair pour l’instant. Nou vous guiderons à travers un exemple plus concret.

### Simulating data from GLM {#sim-glm}

### Simuler les données d’un GLM {#sim-glm}

A good way to learn about linear models is to simulate data where you know exactly how the variables are related, and then analyse this simulated data to see where the parameters show up in the analysis.

Les simulations de données sont une bonne manière pour apprendre les modèles linéaire. L’idée est de générer des données où nous choisirons nous même comment nos différentes variables seront liées les unes aux autres. Ensuite, nous analyserons des données générées (simulées) afin de comprendre comment les paramètres de la simulations se retranscrivent dans notre analyse.

We'll start with a very simple linear model that just has a single categorical factor with two levels. Let's say we're predicting reaction times for congruent and incongruent trials in a Stroop task for a single participant. Average reaction time (`mu`) is 800ms, and is 50ms faster for congruent than incongruent trials (`effect`).

Nous commencerons avec un modèle linéaire dans lequel nous avons un facteur à deux modalités. Imaginons que nous cherchions à prédire les temps de réponses pour des essais congruents et des essais congruents à une tâche de Stroop, et ce pour une seule participante. Le temps de réponse moyen (`mu`) sera de 800ms, notre participante sera plus rapide de 50ms pour les essais congruents plutôt que les essais incongruents (`effect`).

<div class="warning">

<div class="warning">

A \*\*factor\*\* is a categorical variable that is used to divide subjects into groups, usually to draw some comparison. Factors are composed of different \*\*levels\*\*. Do not confuse factors with levels!

Un facteur (dans R, un `factor`) est une variable catégorielle utilisée pour répartir les participant·es dans différents groupe, le plus souvent dans le but de comparer ces groupes. Les facteurs sont composés de différentes modalités (dans R, un `level`). Attention à ne pas confondre les facteurs et les modalités !

In the example above, trial type is a `r mcq(c(answer = "factor", "level"))`, incongruent is a `r mcq(c("factor", answer = "level"))`, and congruent is a `r mcq(c("factor", answer = "level"))`.

</div>

Dans notre exemple, l’essai sera `r mcq(c(answer = "un facteur", "une modalité"))`, un essai congruent sera `r mcq(c("un facteur", answer = "une modalité")` et un essais congruent sera `r mcq(c("un facteur", answer = "une modalité")`.

</div>

You need to represent categorical factors with numbers. The numbers, or \*\*coding\*\* you choose will affect the numbers you get out of the analysis and how you need to interpret them. Here, we will \*\*effect code\*\* the trial types so that congruent trials are coded as +0.5, and incongruent trials are coded as -0.5.

Pour représenter une variable catégorielle, ils nous faudra utiliser des nombres. Ces nombres, également appelé \*\*codage\*\*, auront une influence sur les résultats de l’analyse, et comment interpréter ces résultats. Pour notre exemple, nous choisirons d’utiliser un codage “effet” pour représenter les différents type d’essais. Avec un tel codage, les essais congruents seront représentés par le code +0.5 et les essais incongruents seront représentés par le code -0.5.

A person won't always respond exactly the same way. They might be a little faster on some trials than others, due to random fluctuations in attention, learning about the task, or fatigue. So we can add an \*\*error term\*\* to each trial. We can't know how much any specific trial will differ, but we can characterise the distribution of how much trials differ from average and then sample from this distribution.

A la tâche de Stroop, les participant·es ne répondent pas toujours exactement de la même manière. Parfois, ils seront plus rapide et parfois, plus lent, et ce à cause de fluctuation de leur attention, à cause d’un apprentissage de la tâche ou même à cause de la fatigue. Pour cette raison, dans notre simulation, nous allons ajouter un \*\*terme d’erreur\*\*. Nous ne pouvons pas prévoir de combien précisément chacun des essais différera des autres, mais nous pouvons décrire la distribution représentant les différences entre un essai en particulier et la moyenne des essais.

Here, we'll assume the error term is sampled from a normal distribution with a standard deviation of 100 ms (the mean of the error term distribution is always 0). We'll also sample 100 trials of each type, so we can see a range of variation.

Nous ferons ici l’hypothèse que ce terme d’erreur est tiré d’une distribution normale avec un écart type de 100ms. A noter que la moyenne de la distribution des termes d’erreur vaut toujours 0. Nous échantillonerons 100 essais de chacun des types (congruents et incongruents) afin de pouvoir observer les fluctuations.

So first create variables for all of the parameters that describe your data.

A présent, créons chacune des variables qui nous permettront de décrire nos données.

```{r}

n\_per\_grp <- 100

mu <- 800 # average RT

effect <- 50 # average difference between congruent and incongruent trials

error\_sd <- 100 # standard deviation of the error term

trial\_types <- c("congruent" = 0.5, "incongruent" = -0.5) # effect code

n\_per\_grp <- 100

mu <- 800 # Temps de réponse (RT) moyen

effect <- 50 # Différence moyenne entre les essais congruents et les essais incongruents

error\_sd <- 100 # Ecart-type du terme d’erreur

trial\_types <- c("congruent" = 0.5, "incongruent" = -0.5) # codage “effet”

```

Then simulate the data by creating a data table with a row for each trial and columns for the trial type and the error term (random numbers samples from a normal distribution with the SD specified by `error\_sd`). For categorical variables, include both a column with the text labels (`trial\_type`) and another column with the coded version (`trial\_type.e`) to make it easier to check what the codings mean and to use for graphing. Calculate the dependent variable (`RT`) as the sum of the grand mean (`mu`), the coefficient (`effect`) multiplied by the effect-coded predictor variable (`trial\_type.e`), and the error term.

Ensuite, simulons les donnons en créant un jeu de données avec une ligne par essai, et des colonnes indiquant le type d’essai ainsi que le terme d’erreur (`error`, un nombre tiré d’une distribution normale ayant un écart type spécifié via la variable `error\_sd`). Pour notre variable catégorielle, incluons à la fois les colonnes contenant la modalité de la variable sous son format texte (`trial\_type`) et le codage “effet” qui y est associé (`trial\_type.e`). Cela nous permettra de repérer plus facilement le codage utilisé lorsque nous réaliserons des graphiques. Calculons les valeurs de notre VD (`RT`) comme la somme de la moyenne générale (`mu`), du produit du coefficient (`effet`) et du codage “effet” associé à notre variable (`trial\_type.e`) et du terme d’erreur (`error`).

```{r}

dat <- data.frame(

trial\_type = rep(names(trial\_types), each = n\_per\_grp)

) %>%

mutate(

trial\_type.e = recode(trial\_type, !!!trial\_types),

error = rnorm(nrow(.), 0, error\_sd),

RT = mu + effect\*trial\_type.e + error

)

dat <- data.frame(

trial\_type = rep(names(trial\_types), each = n\_per\_grp)

) %>%

mutate(

trial\_type.e = recode(trial\_type, !!!trial\_types),

error = rnorm(nrow(.), 0, error\_sd),

RT = mu + (effect \* trial\_type.e) + error

)

```

```{block type="info"}

The `!!!` (triple bang) in the code `recode(trial\_type, !!!trial\_types)` is a way to expand the vector `trial\_types <- c("congruent" = 0.5, "incongruent" = -0.5)`. It's equivalent to `recode(trial\_type, "congruent" = 0.5, "incongruent" = -0.5)`. This pattern avoids making mistakes with recoding because there is only one place where you set up the category to code mapping (in the `trial\_types` vector).

Le `!!!` (triple bang) dans le code `recode(trial\_type, !!!trial\_types)` sera évalué comme `trial\_types <- c("congruent" = 0.5, "incongruent" = -0.5)`. Ce code est équivalent au code `recode(trial\_type, "congruent" = 0.5, "incongruent" = -0.5)`. Cependant, le `!!!`nous permet d’éviter de faire des erreurs de recodage en ne définissant quel catégorie est associée à quel code qu’une seule fois (dans le vecteur `trial\_types`).

```

Last but not least, always plot simulated data to make sure it looks like you expect.

Enfin, dernière chose, mais non des moindre, nous devons toujours penser toujours à faire un graphique des données simulées. Ainsi, nous pourrons vérifier que nos données ressemble bien à ce à quoi elles sont censées ressembler.

```{r plot-sim, fig.cap = "Simulated Data"}

ggplot(dat, aes(trial\_type, RT)) +

geom\_violin() +

geom\_boxplot(aes(fill = trial\_type),

width = 0.25, show.legend = FALSE)

```

### Linear Regression

### La régression linéaire

Now we can analyse the data we simulated using the function `lm()`. It takes the formula as the first argument. This is the same as the data-generating equation, but you can omit the error term (this is implied), and takes the data table as the second argument. Use the `summary()` function to see the statistical summary.

Maintenant, nous allons pouvoir analyser les données que nous venons de simuler grâce à la fonction `lm()`. Cette fonction prend une formule comme premier argument et cette formule est équivalente à comment nous nous y somme pris pour générer nos données (si ce n’est que nous n’y spécifierons pas le terme d’erreur). Nous utiliserons notre jeu de données comme second argument. Afin d’obtenir un résumé des informations pertinentes concernant notre modèle, nous utiliserons la fonction `summary()`.

```{r}

my\_lm <- lm(RT ~ trial\_type.e, data = dat)

summary(my\_lm)

```

Notice how the \*\*estimate\*\* for the `(Intercept)` is close to the value we set for `mu` and the estimate for `trial\_type.e` is close to the value we set for `effect`.

Nous pouvons remarquer à quel point la valeur \*\*estimate\*\* (i.e., la valeure estimée par la régression) pour l’`(Intercept)` est proche de la valeur que nous avions défini pour `mu`. Concernant la valeur estimée pour `trial\_type.e`, nous pouvons remarquer qu’elle est très proche de notre variable `effect`.

```{block, type="try"}

Change the values of `mu` and `effect`, resimulate the data, and re-run the linear model. What happens to the estimates?

Changez les valeurs pour `mu` et `effet`, simulez à nouveau les données et relancez la régression linéaire. Qu’advient-il des valeurs estimées ?

```

### Residuals {#residuals}

### Les résidus {#residuals}

You can use the `residuals()` function to extract the error term for each each data point. This is the DV values, minus the estimates for the intercept and trial type. We'll make a density plot of the residuals below and compare it to the normal distribution we used for the error term.

Afin d’extraire le terme d’erreur de chacune des observations dont nous nous sommes servis (i.e., les résidus), nous pouvons utiliser la fonction `residuals()`. Cette fonction calcule les résidus en soustrayant à notre VD les valeurs estimées pour l’intercept et le produit entre le type d’essai et le code de contraste. Générons un graphique de densité afin de comparer cette distribution de résidus à la distribution que nous avons utilisée pour définir notre terme d’erreur.

```{r res-density-plot, fig.cap="Model residuals should be approximately normally distributed for each group"}

```{r res-density-plot, fig.cap="Les résidus du modèle devraient être plus ou moins normalement distribué pour chacun des groupes"}

res <- residuals(my\_lm)

ggplot(dat) +

stat\_function(aes(0), color = "grey60",

fun = dnorm, n = 101,

args = list(mean = 0, sd = error\_sd)) +

geom\_density(aes(res, color = trial\_type))

```

You can also compare the model residuals to the simulated error values. If the model is accurate, they should be almost identical. If the intercept estimate is slightly off, the points will be slightly above or below the black line. If the estimate for the effect of trial type is slightly off, there will be a small, systematic difference between residuals for congruent and incongruent trials.

Il est également possible de comparer les résidus de notre modèle à l’erreur que nous avons simulé. Si notre modèle fonctionne correctement, les deux devraient être quasi-identique. Si jamais la valeur estimée de l’`(Intercept) est légèrement différente de celle définie, alors chacun des points de notre graphique devrait se situer au dessus ou en dessous de la droite en noir. Si jamais la valeur estimée au coefficient lié au type d’essai `trial\_type.e`est légèrement différente de celle définie, nous devrions retrouver des différences systématiques entre les essais congruents et les essais incongruents.

```{r res-err-plot, fig.cap="Model residuals should be very similar to the simulated error"}

ggplot(dat) +

geom\_abline(slope = 1) +

geom\_point(aes(error, res,color = trial\_type)) +

ylab("Model Residuals") +

xlab("Simulated Error")

```

```{r res-err-plot, fig.cap="Les résidus du modèles devraient être très proche de l’erreur simulée"}

ggplot(dat) +

geom\_abline(slope = 1) +

geom\_point(aes(error, res, color = trial\_type)) +

ylab("Résidus du modèle") +

xlab("Erreur simulée")

```

```{block, type="try"}

What happens to the residuals if you fit a model that ignores trial type (e.g., `lm(Y ~ 1, data = dat)`)?

```

```{block, type="try"}

Que se passerait-il au niveau des résidus si nous décidions d'utiliser un modèle ne rendant pas compte du type d’essai (i.e., `lm(Y ~ 1, data = dat)`) ?

```

### Predict New Values {#predict}

### Prédire de nouvelles valeurs {#predict}

You can use the estimates from your model to predict new data points, given values for the model parameters. For this simple example, we just need to now the trial type to make a prediction.

Il est possible d’utiliser les valeurs estimées par notre modèle pour prédire notre VD pour de nouvelles observations. Pour notre exemple, nous avons simplement besoin à quel type d’essai correspond notre observation pour faire une prédiction.

For congruent trials, you would predict that a new data point would be equal to the intercept estimate plus the trial type estimate multiplied by 0.5 (the effect code for congruent trials).

Pour les essais congruents, nous pourrions prédire qu’une nouvelle observation correspondrait à la valeur estimée pour l’intercept auquel nous ajoutons la valeur estimée pour le coefficient lié au type d’essai multiplié par 0.5 (le codage “effet” pour les essais congruents).

```{r}

int\_est <- my\_lm$coefficients[["(Intercept)"]]

tt\_est <- my\_lm$coefficients[["trial\_type.e"]]

tt\_code <- trial\_types[["congruent"]]

new\_congruent\_RT <- int\_est + tt\_est \* tt\_code

new\_congruent\_RT

```

You can also use the `predict()` function to do this more easily. The second argument is a data table with columns for the factors in the model and rows with the values that you want to use for the prediction.

Il est également possible d’utiliser la fonction `predict()` afin d’obtenir au même résultat plus facilement. Le premier argument de cette fonction est notre régression linéaire et le second un jeu de données dont les colonnes sont les prédicteurs utilisés dans notre modèle. Les lignes doivent correspondre aux valeurs pour lesquels nous souhaitons faire de nouvelles prédictions.

```{r}

predict(my\_lm, newdata = tibble(trial\_type.e = 0.5))

```

```{block type="info"}

If you look up this function using `?predict`, you will see that "The function invokes particular methods which depend on the class of the first argument."

What this means is that `predict()` works differently depending on whether you're predicting from the output of `lm()` or other analysis functions. You can search for help on the lm version with `?predict.lm`.

```

```{block type="info"}

En cherchant de l’aide sur cette fonction en utilisant `?predict`, il est possible que cette fonction “invokes particular methods which depend on the class of the first argument”, c’est à dire qu’elle invoque une \_méthode\_ particulière en fonction de la classe du premier argument.

Cela signifie que la fonction `predict()` fonctionnera différemment en fonction de si l’on cherche à faire une prédiction à partir d’une régression linéaire `lm()`ou d’un autre type d’analyse. Pour obtenir de l’aide concernant la version liée à la classe `lm`, il faudrait utiliser `?predict.lm`.

```

### Coding Categorical Variables {#coding-schemes}

### Le codage des variable catégorielles {#coding-schemes}

In the example above, we used \*\*effect coding\*\* for trial type. You can also use \*\*sum coding\*\*, which assigns +1 and -1 to the levels instead of +0.5 and -0.5. More commonly, you might want to use \*\*treatment coding\*\*, which assigns 0 to one level (usually a baseline or control condition) and 1 to the other level (usually a treatment or experimental condition).

Dans l’exemple ci-dessus, nous avons utilisé un \*\*codage “effet”\*\* pour coder le type d’essai. Nous aurions également pu utiliser un \*\*codage “somme”\*\* qui attribuent des valeurs de +1 et -1 aux modalités de la variable, plutôt que des valeurs de +0.5 et -0.5. Il aurait également été possible d’utiliser un codage “traitement” qui assigne une valeur de 0 à l’un des facteurs (le plus souvent, à la condition contrôle) et une valeur de 1 à l’autre facteur (le plus souvent, à la condition expérimentale).

Here we will add sum-coded and treatment-coded versions of `trial\_type` to the dataset using the `recode()` function.

Ajoutons les codes de type somme et traitement pour notre variable `trial\_type`dans notre jeu de données avec la fonction `recode()`.

```{r}

dat <- dat %>% mutate(

trial\_type.sum = recode(trial\_type, "congruent" = +1, "incongruent" = -1),

trial\_type.tr = recode(trial\_type, "congruent" = 1, "incongruent" = 0)

)

```

If you define named vectors with your levels and coding, you can use them with the `recode()` function if you expand them using `!!!`.

En utilisant un vecteur nommé contenant les modalités et le code associé, nous aurions pu utiliser la fonction `recode()` en combinaison du `!!!`.

```{r}

tt\_sum <- c("congruent" = +1,

"incongruent" = -1)

tt\_tr <- c("congruent" = 1,

"incongruent" = 0)

dat <- dat %>% mutate(

trial\_type.sum = recode(trial\_type, !!!tt\_sum),

trial\_type.tr = recode(trial\_type, !!!tt\_tr)

)

```

Here are the coefficients for the effect-coded version. They should be the same as those from the last analysis.

Voici les coefficients de notre version de l’analyse utilisant le codage “effet”. Ils devraient être identiques à ceux obtenus lors de notre dernière analyse.

```{r}

lm(RT ~ trial\_type.e, data = dat)$coefficients

```

Here are the coefficients for the sum-coded version. This give the same results as effect coding, except the estimate for the categorical factor will be exactly half as large, as it represents the difference between each trial type and the hypothetical condition of 0 (the overall mean RT), rather than the difference between the two trial types.

A présent, voici les coefficients pour la version utilisant un codage “somme”. Cette fois, les résultats sont identiques à ceux du codage “effet”, à l’exception de la valeur estimée pour le coefficient associé à notre facteur catégoriel. Celui-ci est deux fois plus petit. En effet, plutôt que la différence de temps de réponses entre les deux types d’essais, il représente désormais la différence entre la moyenne des temps de réponses pour un type d’essai et la condition virtuelle codée 0 (notre moyenne générale des temps de réponses).

```{r}

lm(RT ~ trial\_type.sum, data = dat)$coefficients

```

Here are the coefficients for the treatment-coded version. The estimate for the categorical factor will be the same as in the effect-coded version, but the intercept will decrease. It will be equal to the intercept minus the estimate for trial type from the sum-coded version.

Enfin, voici les coefficients pour la version utilisant un codage “traitement”. La valeur estimée pour le coefficient associé à notre variable catégorielle sera la même que pour la version de notre analyse utilisant un codage “effet”. En revanche, la valeur estimée pour l’intercept diminuera. Celle-ci sera égale à la valeur de l’intercept auquel nous soustrayons la valeurs estimée du coefficient associé au type d’essais provenant de notre analyse utilisant le codage “somme”.

```{r}

lm(RT ~ trial\_type.tr, data = dat)$coefficients

```

## Relationships among tests {#test-rels}

## Relations entre les tests statistiques {#test-rels}

### T-test

### Test t

The t-test is just a special, limited example of a general linear model.

Le test t est un cas particulier du modèle linéaire.

```{r}

t.test(RT ~ trial\_type.e, data = dat, var.equal = TRUE)

```

```{block, type="try"}

What happens when you use other codings for trial type in the t-test above? Which coding maps onto the results of the t-test best?

Que se passe-t-il si nous utilisons un autre codage pour le type d’essai dans le test t ci-dessus ? Lequel des codages correspond au mieux à un test t ?

```

### ANOVA

### L’ANOVA

ANOVA is also a special, limited version of the linear model.

L’ANOVA est également un cas particulier du modèle linéaire

```{r}

my\_aov <- aov(RT ~ trial\_type.e, data = dat)

summary(my\_aov, intercept = TRUE)

```

The easiest way to get parameters out of an analysis is to use the `broom::tidy()` function. This returns a tidy table that you can extract numbers of interest from. Here, we just want to get the F-value for the effect of trial\_type. Compare the square root of this value to the t-value from the t-tests above.

La manière la plus simple d’extraire les paramètres d’une analyse est d'utiliser la fonction `broom::tidy()`. Celle-ci renvoie un jeu de données ordonnée il est possible d’extraire les valeurs d’intérêt.

Ici, essayons d’extraire la valeur du F de Fisher correspondant à l’effet du type d’essais. Comparons la racine de cette valeur à celle du test t ci-dessus.

```{r}

f <- broom::tidy(my\_aov)$statistic[1]

sqrt(f)

```

## Understanding ANOVA

## Comprendre l’ANOVA

We'll walk through an example of a one-way ANOVA with the following equation:

Décrivons étape par étape l’exemple d’une ANOVA à un facteur contrôlé en commençant par cette équation :

$Y\_{ij} = \mu + A\_i + S(A)\_{ij}$

This means that each data point ($Y\_{ij}$) is predicted to be the sum of the grand mean ($\mu$), plus the effect of factor A ($A\_i$), plus some residual error ($S(A)\_{ij}$).

Cette équation indique que chaque observation ($Y\_{ij}$) peut être décomposée en une moyenne générale, plus l’effet d’un facteur A ($A\_i$) et enfin de l’erreur résiduelle ($S(A)\_{ij}$).

### Means, Variability, and Deviation Scores

### Moyennes, variabilité, et scores de déviation

Let's create a simple simulation function so you can quickly create a two-sample dataset with specified Ns, means, and SDs.

Pour commencer, créons une simple fonction qui nous permettra de simuler un jeu de données comprenant deux échantillons avec des N, moyennes et écart types spécifiques.

```{r}

two\_sample <- function(n = 10, m1 = 0, m2 = 0, sd1 = 1, sd2 = 1) {

s1 <- rnorm(n, m1, sd1)

s2 <- rnorm(n, m2, sd2)

data.frame(

Y = c(s1, s2),

grp = rep(c("A", "B"), each = n)

)

}

```

Now we will use `two\_sample()` to create a dataset `dat` with N=5 per group, means of -2 and +2, and SDs of 1 and 1 (yes, this is an effect size of d = 4).

A présent, utilisons cette fonction `two\_sample()` pour générer un jeu de données avec un nombre de participants par groupe de N = 5, des moyennes de -2 et +2 et des écart types de 1 et 1 (yup, cela nous donne une modeste taille d’effet de d = 4).

```{r}

dat <- two\_sample(5, -2, +2, 1, 1)

```

You can calculate how each data point (`Y`) deviates from the overall sample mean ($\hat{\mu}$), which is represented by the horizontal grey line below and the deviations are the vertical grey lines. You can also calculate how different each point is from its group-specific mean ($\hat{A\_i}$), which are represented by the horizontal coloured lines below and the deviations are the coloured vertical lines.

Il est possible de calculer à quel point chaque observation (`Y`) dévie de la moyenne générale ($\hat{\mu}$). Cette moyenne générale est représentée par la ligne grise horizontale ci-dessous, et les déviations sont représentée par les lignes grises verticale. Il est également possible de calculer de combien chaque observation diffère de la moyenne de son groupe ($\hat{A\_i}$). La moyenne des groupes est représentée par les lignes horizontales colorées et les déviations par les lignes verticales colorées.

```{r deviation, echo = FALSE, fig.cap="Deviations of each data point (Y) from the overall and group means"}

```{r deviation, echo = FALSE, fig.cap="Déviations entre chaque observation (Y) et les moyennes générales et des groupes"}

group\_by(dat, grp) %>%

mutate(Ymean = mean(Y)) %>%

ungroup() %>%

mutate(i = row\_number()) %>%

ggplot(aes(i, Y, color = grp)) +

geom\_hline(yintercept = mean(dat$Y), color = "grey40") +

geom\_linerange(aes(x = i-0.025, ymin = Y, ymax = mean(dat$Y)), color = "grey40") +

geom\_linerange(aes(x = i-0.025, ymin = mean(dat$Y), ymax = Y), color = "grey40") +

geom\_hline(aes(yintercept = Ymean, color = grp)) +

geom\_linerange(aes(x = i+0.025, ymin = Y, ymax = Ymean, group = i, color = grp)) +

geom\_linerange(aes(x = i+0.025, ymin = Ymean, ymax = Y, group = i, color = grp)) +

geom\_point(size = 3) +

theme(axis.text.x = element\_blank(),

axis.ticks.x = element\_blank()) +

xlab("")

```

You can use these deviations to calculate variability between groups and within groups. ANOVA tests whether the variability between groups is larger than that within groups, accounting for the number of groups and observations.

A présent, il est possible d’utiliser ces déviations pour calculer les variabilités inter- et intra-groupes. L’ANOVA teste si la variabilité inter-groupe est importante comparativement à la variabilité intra-groupe, et ce en rendant compte du nombre de groupes et du nombre d’observations.

### Decomposition matrices {#decomp}

### Matrices de décompositions{#decomp}

We can use the estimation equations for a one-factor ANOVA to calculate the model components.

Il est possible d’utiliser les équations d’estimation pour l’ANOVA à un facteur contrôlé afin de déterminer les paramètres de notre modèle.

\* `mu` is the overall mean

\* `a` is how different each group mean is from the overall mean

\* `err` is residual error, calculated by subtracting `mu` and `a` from `Y`

\* `mu` désigne la moyenne générale

\* `a` désigne à quel point chaque groupe dévie de la moyenne générale

\* `err` désigne l’erreur résiduelle obtenu en soustrayant `mu`et `a`d’une observation `Y`

This produces a \*decomposition matrix\*, a table with columns for `Y`, `mu`, `a`, and `err`.

Il est maintenant possible de réaliser une matrice de décomposition, un tableau ayant pour colonnes `Y`, `mu`, `a` et`err`.

```{r}

decomp <- dat %>%

select(Y, grp) %>%

mutate(mu = mean(Y)) %>% # calculate mu\_hat

group\_by(grp) %>%

mutate(a = mean(Y) - mu) %>% # calculate a\_hat for each grp

ungroup() %>%

mutate(err = Y - mu - a) # calculate residual error

decomp <- dat %>%

select(Y, grp) %>%

mutate(mu = mean(Y)) %>% # calcul du mu\_hat

group\_by(grp) %>%

mutate(a = mean(Y) - mu) %>% # calcul du a\_hat pour chacun des groupes

ungroup() %>%

mutate(err = Y - mu - a) # calcul de l’erreur résiduelles

```

`r knitr::kable(decomp)`

Calculate sums of squares for `mu`, `a`, and `err`.

Calculons la somme des carrés pour `mu`, `a` et `err`.

```{r}

SS <- decomp %>%

summarise(mu = sum(mu\*mu),

a = sum(a\*a),

err = sum(err\*err))

SS <- decomp %>%

summarise(mu = sum(mu \* mu),

a = sum(a \* a),

err = sum(err \* err))

```

`r knitr::kable(SS)`

If you've done everything right, `SS$mu + SS$a + SS$err` should equal the sum of squares for Y.

Si chacune des étapes a été suivies correctement, `SS$mu + SS$a + SS$err` devrait être égale à la somme des carré de Y.

```{r}

SS\_Y <- sum(decomp$Y^2)

all.equal(SS\_Y, SS$mu + SS$a + SS$err)

```

Divide each sum of squares by its corresponding degrees of freedom (df) to calculate mean squares. The df for `mu` is 1, the df for factor `a` is `K-1` (K is the number of groups), and the df for `err` is `N - K` (N is the number of observations).

Divisons chaque somme des carrées par leur degré de liberté (df) afin de calculer les carrés moyens. Le degré de liberté pour `mu` est de 1, pour notre facteur `a`, il est de `K - 1` (avec `K` le nombre de groupes) et pour `err`, il est de `N - K`(avec `N`le nombre total d’observation).

```{r}

K <- n\_distinct(dat$grp)

N <- nrow(dat)

df <- c(mu = 1, a = K - 1, err = N - K)

MS <- SS / df

```

`r knitr::kable(MS)`

Then calculate an F-ratio for `mu` and `a` by dividing their mean squares by the error term mean square. Get the p-values that correspond to these F-values using the `pf()` function.

Nous pouvons à présent calculer un F-ratio pour `mu` et `a` en divisant leurs carrés moyens par le carré de l’erreur. Il nous suffit de calculer les valeurs de \_p\_ correspondant à ces valeurs de \_F\_ en utilisant la fonction `pf()`.

```{r}

F\_mu <- MS$mu / MS$err

F\_a <- MS$a / MS$err

p\_mu <- pf(F\_mu, df1 = df['mu'], df2 = df['err'], lower.tail = FALSE)

p\_a <- pf(F\_a, df1 = df['a'], df2 = df['err'], lower.tail = FALSE)

```

Put everything into a data frame to display it in the same way as the ANOVA summary function.

Créons à présent un jeu de données qui comprend l’ensemble de ces termes afin de les afficher comme nous l’aurions fait avec la fonction `summary()` de l’ANOVA.

```{r}

my\_calcs <- data.frame(

term = c("Intercept", "grp", "Residuals"),

Df = df,

SS = c(SS$mu, SS$a, SS$err),

MS = c(MS$mu, MS$a, MS$err),

F = c(F\_mu, F\_a, NA),

p = c(p\_mu, p\_a, NA)

)

```

`r knitr::kable(my\_calcs , digits = 3)`

Now run a one-way ANOVA on your results and compare it to what you obtained in your calculations.

A présent, lancez l’ANOVA à un facteur contrôlé et comparez les résultats à ceux que vous aviez obtenus précédemment.

```{r}

aov(Y ~ grp, data = dat) %>% summary(intercept = TRUE)

```

```{block, type="try"}

Using the code above, write your own function that takes a table of data and returns the ANOVA results table like above.

En utilisant le code ci-dessus, écrivez votre propre fonction qui prendra en argument un jeu de données, et en sortira une table d’ANOVA comme celle ci-dessus.

```

## Exercises

## Exercices

Download the [exercises](exercises/09\_glm\_exercise.Rmd). See the [answers](exercises/09\_glm\_answers.Rmd) only after you've attempted all the questions.

Téléchargez les exercices. Ne regardez les réponses qu’après avoir essayé de répondre à l’ensemble des questions.