A tutorial on non-inferiority assessments for experimental hut trials

Joseph D. Challenger & The Global Malaria Programme, World Health Organisation

August 2023

Contents

Chapter 1

Introduction

This tutorial is designed to provide guidance on carrying out non-inferiority assessments for insecticide-treated products in experimental huts trials (EHTs). The methodology used here matches that used by the Global Malaria Programme for assessing the comparative effectiveness of vector control products[1]. In these trials, volunteers sleep inside the huts with the vector control products, and mosquitoes are collected early the next morning, to gather information on mosquito mortality and mosquito blood feeding. Both the products and the volunteers are rotated around the huts, to avoid potential biases.

Once the data have been collected, they are analysed using logistic regression models, with separate models fitted for mosquito mortality and mosquito blood feeding. Fixed effects should be included for the treatment arm, hut, sleeper, and the day of the trial. For insecticide-treated nets (ITNs), it is common to include unwashed and washed nets of the same brand in the same trial. Here, washing the nets is designed to reproduced the effects of ageing. In this way one can assess the longevity of the product. For ITNs, therefore, one can make separate non-inferiority assessments for unwashed and washed nets, as well as a combined assessment, where unwashed and washed nets of the same brand are assessed together. For this latter analysis, the fixed effects included in the logistic regression are: brand of net, hut, sleeper, the day of the trial, and whether the net has been washed or not.

The dataset used in this tutorial to illustrate the methodology, is a synthetic dataset (that is, one generated from computer simulation), rather than one from a real-world EHT. The treatments arms are different types of bed net, but the same methodology can be used for *e.g.* indoor residual spraying (IRS). The treatment arms in the dataset are named for their role in the non-inferiority trial. The treatment arms are: an untreated control net, a standard comparator (unwashed and washed), an active comparator (unwashed and washed), and a candidate net (unwashed and washed). Hence, in total there are seven treatment arms. The candidate net should be of the same chemistry as the active comparator. For example, in the context of ITNs, this could be a dual-chemistry, such as pyrethroid and piperonyl-butoxide (PBO). The active comparator (sometimes referred to as the 'first in class product') should be a pre-approved product (*e.g.* one that has shown significant efficacy in a cluster randomised trial). In these trials, we wish to assess the non-inferiority of the candidate net compared to the active comparator. As this assessment does not provide information on the products efficacy in absolute terms, we will also test whether the candidate net is superior to the standard comparator. In the context of ITNs, the standard comparator is usually a pyrethroid-only net.

[1]See *Technical consultation on determining non-inferiority of vector control products within an established class* Report of a virtual meeting 31 August–2 September 2021, World Health Organisation

Finally, we should discuss the choice of the non-inferiority margin (NIM) used here. The efficacy of the candidate net and the active comparator (be it for mosquito mortality or blood feeding inhibition) are compared by constructing an odds ratio (OR). This OR and its 95% confidence interval should then be compared to the NIM. For mosquito mortality, the entire 95% confidence interval must lie *above* the NIM for the candidate product to be non-inferior to the active comparator. If this is not the case, the candidate is said to be 'not non-inferior' to the active comparator. In contrast, for blood feeding the entire 95% confidence interval must lie *below* the NIM for non-inferiority to be achieved.

In this work, the NIM is set so that the candidate net efficacy can be no more than 7% lower than that of the active comparator. So, when assessing mosquito mortality, the NIM is chosen so that mosquito mortality measured for the candidate net must be no more than 7% lower than that measured for the active comparator, in order for non-inferiority to be achieved. This means that the OR for the NIM will vary from trial to trial, and must be calculated for each assessment. We will show examples of this in the following chapters. Chapter 2 will outline the procedure in R; Chapter 3 will follow the same procedure using STATA.

Chapter 2

Analysis in R

A brief introduction to R

The code included in this chapter is written in R. We do not include a comprehensive introduction to R, as many others are available elsewhere. However, we will include a few brief comments on the syntax of R. Variables (be they numbers, or character strings) can be stored in the internal memory using either `=` or `<-`. For example, writing `x<-5` assigns a value of 5 to `x`. The symbol `#` is used to indicate a comment. That is, anything that follows a `#` will not be read as R code.

R contains a number of core functions, which carry out commonly used operations. Additional functions can be found within *packages*. A package can be loaded using the `library()` function. For example, we can load the ggplot2 package, which is a versatile library for making graphs, by running the following command:

```
> library(ggplot2)
```

If you have not used this package before, you may need to download it. You can do this by running the command `install.packages('ggplot2')`. Alternatively, if you're using Rstudio, you can click on the 'Tools' menu, then click on 'Install packages...', and search for the desired package. For users who have not installed R before, R can be downloaded from `https://cran.r-project.org`. When installing the packages, Windows users may find that they are prompted to install Rtools, which can be downloaded from the same website. The results presented here were generated using R version 4.2.1, using RStudio Desktop.

2.1 Loading & summarising a dataset

To demonstrate the statistical analyses to be carried out, we will use a simulated dataset. This dataset has been uploaded with these materials, along with an R script containing the work outlined in this tutorial. To run the R script you should download R & RStudio. If you wish to run the script, you should download the ZIP file, and extract the folder. Then, double-click on the project file `WHO_NI_tutorial.Rproj` to open it in RStudio. We recommend doing this, as this should mean that the R session begins with the current working directory matching the location of our files. This will make it easier for R to find all our code. You can check the current working directory at any time by running the command `getwd()`.

Once the R project file is open, you can then open the R script `R_tutorial.R`. First we will load the packages that we will use:

```
library(ggplot2)
library(lme4)
library(cowplot)
...
```

Running the command `source('useful_functions_tutorial.R')` will load some user-defined functions that are stored in another file within this project. If you wish, you can open this file to inspect the functions.

The dataset has been stored as an `.csv` file, and can be loaded using the following command:

```
> df <- read.csv('example_dataset.csv')
```

This stores the data in the variable `df`. Now we can see the contents of this dataset, using the `str()` function.

```
> str(df)
'data.frame': 343 obs. of  14 variables:
 $ day      : int  1 1 1 1 1 1 1 2 2 2 ...
 $ hut      : int  1 2 3 4 5 6 7 1 2 3 ...
 $ sleeper  : int  2 3 4 5 6 7 1 3 4 5 ...
 $ treatment: chr  "Control" "Standard_comparator_unwashed" ...
 $ ITN      : chr  "Control" "Standard_comparator" ...
 $ wash     : int  0 0 1 0 1 0 1 0 0 1 ...
 $ unf_live : int  7 6 1 13 12 3 9 4 11 7 ...
 $ unf_dead : int  0 1 1 8 3 1 2 0 0 0 ...
 $ bf_live  : int  6 1 1 7 3 3 4 1 1 2 ...
 $ bf_dead  : int  0 1 0 2 2 1 1 0 0 0 ...
 $ tot_dead : int  0 2 1 10 5 2 3 0 0 0 ...
 $ tot_bf   : int  6 2 1 9 5 4 5 1 1 2 ...
 $ total    : int  13 9 3 30 20 8 16 5 12 9 ...
```

Here we see that the dataset contains 343 data points. This trial has 7 arms and runs over 7 weeks (49 days). Let's look at this in more detail. The variable `treatment` defines which trial arm each data point relates to. We can summarise the trial arms like this:

```
> table(df$treatment)

  Active_comparator_unwashed     Active_comparator_washed
                          49                           49
          Candidate_unwashed             Candidate_washed
                          49                           49
                     Control Standard_comparator_unwashed
                          49                           49
  Standard_comparator_washed
                          49
```

Hence we can see that the trial contains 3 ITNs: a standard comparator, an active comparator, and the candidate net. For each ITN, there are 2 trial arms, containing unwashed and washed nets. Additionally, the trial contains an arm with an untreated control net. The dataset also contains

an alternative way to describe the trial arms, using the variables `ITN` and `wash`. The latter variable takes a value 0 (unwashed) or 1 (washed). You can look at a summary of these variables running `table(df$ITN)` and `table(df$wash)`.

The variable `total` records the total number of mosquitoes collected in a given hut on a given night. These mosquito counts are broken down to indicate whether or not each mosquito has died or blood fed: `unf_live` = unfed & alive; `unf_dead` = unfed & dead; `bf_live` = blood fed & live; `bf_dead` = blood fed & dead. The dataset also summarises the total number of dead mosquitoes (`tot_dead`), and the total number of blood-fed mosquitoes (`tot_bf`).

We can also see that the study contains 7 huts and 7 sleepers:

```
> table(df$hut)
 1  2  3  4  5  6  7
49 49 49 49 49 49 49

> table(df$sleeper)
 1  2  3  4  5  6  7
49 49 49 49 49 49 49
```

It will be useful to change the data types of some of the variables. For the terms that we will include as fixed effects in the model, we will make these *factor variables* in R:

```
df$hut <- as.factor(df$hut)
df$sleeper <- as.factor(df$sleeper)
df$day <- as.factor(df$day)
df$treatment <- as.factor(df$treatment)
df$ITN <- as.factor(df$ITN)
```

This is particularly important for variables like `hut` and `sleeper`, where the numbering is purely to label the huts. We want our regression model to recognise that each number indicates a distinct category: it does not represent a quantitative measurement. Finally for this section, we will look at a summary of the mosquito mortality and blood-feeding across the 7 trial arms. These summaries are taken directly from the dataset (*i.e.* they are not adjusted estimates derived from a fitted regression model), using the user-defined function `summm()`. See Appendix A for a description of this function.

```
> tab_mortality <- summm(df, vec = df$treatment, td = 'tot_dead',
                           tot = 'total', table = 1)
> tab_mortality
                          Arm Percentage
1                     Control       5.99
2 Standard_comparator_unwashed      11.61
3   Standard_comparator_washed       8.75
4   Active_comparator_unwashed      34.09
5     Active_comparator_washed      21.45
6           Candidate_unwashed      28.99
7             Candidate_washed      22.54
> tab_bf <- summm(df, vec = df$treatment, td = 'tot_bf', tot = 'total', table = 1)
> tab_bf
                          Arm Percentage
```

```
1                       Control      33.81
2 Standard_comparator_unwashed      17.41
3   Standard_comparator_washed      23.96
4   Active_comparator_unwashed      17.05
5     Active_comparator_washed      25.81
6           Candidate_unwashed      16.77
7             Candidate_washed      26.16
```

2.2 Making the non-inferiority assessment

For each hut trial of ITNs, we will make 6 separate non-inferiority assessments. Note that, typically, the blood-feeding assessments are not carried out for trials of IRS. For ITNs, the 6 assessments are as follows:

1. Mosquito mortality: comparing the unwashed candidate to the unwashed active comparator
2. Mosquito mortality: comparing the washed candidate to the washed active comparator
3. Mosquito mortality: comparing the candidate to the active comparator (unwashed and washed combined)
4. Blood feeding: comparing the unwashed candidate to the unwashed active comparator
5. Blood feeding: comparing the washed candidate to the washed active comparator
6. Blood feeding: comparing the candidate to the active comparator (unwashed and washed combined)

For each individual assessment, we can break it down into the following steps:

- Calculate the unadjusted mosquito mortality (or blood feeding proportion) directly from the data (*i.e.* without fitting a regression model)
- Choose which trial arm should be used as the baseline category in the regression model. Then fit the regression model, adjusting for the relevant factors.
- Calculate the odds ratio for mosquito mortality (blood feeding) in the candidate arm, compared to that observed in the active comparator arm.
- Calculate the non-inferiority margin to use, based on the unadjusted mosquito mortality observed in the active comparator arm. Then make the non-inferiority assessment.
- Now check that the candidate net is superior to the standard comparator. First we change the baseline category of the regression model, then we fit the same regression model as before to the data. The p-value for the fixed effect will be used to test for superiority.

In the R code, we also provide code to make visualisations of the non-inferiority assessment, which we shall also outline here. We shall now make a brief comment about the type of regression models we will use here– the logistic regression model. This type of model is used to model data that can be described as proportions (*e.g.* proportion of mosquitoes killed, or proportion of mosquitoes

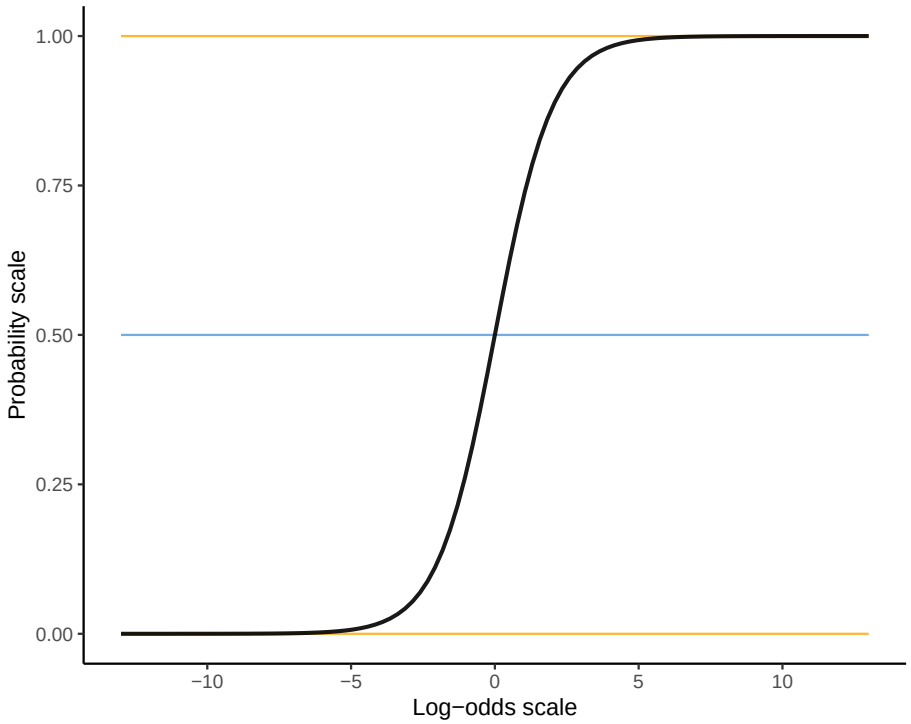

Figure 2.1: Relationship between the log-odds and probability scales. The function is symmetric about 0, which corresponds to a probability of 0.5. The probability approaches 1 as the log-odds value becomes very large (technically, we say 'as it tends to infinity'). Similarly, the probability goes to 0 as the log-odds value tends to minus infinity. The blue horizontal line locates $p = 0.5$, which corresponds to a value of 0 on the log-odds scale.

blood fed). To fit this model, the proportion in question, often denoted as p, is transformed onto the log-odds scale. If we define the transformed value as $X(p)$, we can write:

$$X(p) = \log\left(\frac{p}{1-p}\right).$$

Note that, on the log-odds scale, values can be positive or negative. A value of 0 corresponds to $p = 0.5$. We can write the inverse function like this:

$$p = \text{InvLogit}(X) = \frac{\exp(X)}{\exp(X)+1}.$$

Whilst a proportion is restricted to the interval [0,1], $X(p)$, can take any value, positive or negative. This facilitates the fitting of the regression model: the model is not hampered by a lower limit as p approaches 0, or an upper limit as p approaches 1. Figure 2.1 shows this transformation.

2.2.1 Mosquito mortality

Let's start with the first non-inferiority assessment listed above (mosquito mortality, unwashed nets). We will fit a regression model with the following form:

```
fit1 <-
```

```
glm(
  cbind(tot_dead, total - tot_dead) ~
    treatment + hut + sleeper + day,
  family = binomial, data = df)
```

Let's define the terms mentioned here. The function `glm()`, which comes from the `lme4` package will fit a generalised linear regression model to the data (we use the argument `data` to tell the function which data to use). Setting `family = binomial` indicates that we wish to fit a logistic regression model. We have asked that the model output be stored in the container `fit1`. Note that the numbering of the fitted models in this tutorial (`fit1`, `fit2`, *etc.*) matches with the 6 non-inferiority assessments listed at the beginning of this section.

The first argument included inside the function `glm()`, `cbind(tot_dead, total - tot_dead)`, indicates that the mosquito counts can be split into two categories: dead (`tot_dead`), or alive (the number of alive mosquitoes can be written as the total number of mosquitoes minus the number of dead mosquitoes). All terms after the tilde symbol (`~`) are terms we wish to include in the model. Here we include fixed effects for the treatment arm, hut, sleeper and day of the trial. Before proceeding further, we should think about how the model will interpret these fixed effects: we will use `treatment` as an example, but the same principles hold to the other variables too. As we saw earlier, `treatment` can take 7 different values- one for each arm in the trial. When fitting the model, the function `glm()` will choose one of these 7 values as a 'baseline' category (by default, the first category in alphabetical or numerical value will be chosen). Then the value of 6 parameters will be estimated; these can be thought of as *offsets*, differences between a given trial arm and the baseline trial arm. The choice of which trial arm is an arbitrary one, and does not affect the non-inferiority assessment. However, the assessment is made simpler if the active comparator is used as the baseline category. Then, one can simply read off the parameter value associated with the candidate net: this tells us the difference in performance of the candidate net, compared to the active comparator. If the treatment variable is a factor variable in R, `glm()` will set the lowest factor to be the baseline category. We can check the ordering of factors using the `levels()` function:

```
> levels(df$treatment)
[1] "Active_comparator_unwashed"   "Active_comparator_washed"
[3] "Candidate_unwashed"           "Candidate_washed"
[5] "Control"                      "Standard_comparator_unwashed"
[7] "Standard_comparator_washed"
```

Here we can see that `"Active_comparator_unwashed"` is the lowest level in the factor (this is because the levels are ordered alphabetically by default). So, this means we don't have to reorder the levels, but later we'll meet an example where we do need to do this. We can now fit the model defined above (`fit1`). When we have done this, we can use the `summary()` function to view the fitted model

```
> summary(fit1)

Coefficients:
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -0.0690133  0.3184980  -0.217 0.828455
treatmentActive_comparator_washed -0.5350687  0.1545003  -3.463 0.000534 ***
treatmentCandidate_unwashed       -0.0286659  0.1567682  -0.183 0.854911
```

```
treatmentCandidate_washed             -0.5543215  0.1655243  -3.349 0.000811 ***
treatmentControl                      -2.0162343  0.2021793  -9.973  < 2e-16 ***
treatmentStandard_comparator_unwashed -1.2311743  0.1855108  -6.637 3.21e-11 ***
treatmentStandard_comparator_washed   -1.6727303  0.2036198  -8.215  < 2e-16 ***
hut2                                  -0.3197885  0.1578667  -2.026 0.042797 *
hut3                                  -0.4449210  0.1603844  -2.774 0.005536 **
hut4                                  -1.0380668  0.1867768  -5.558 2.73e-08 ***
hut5                                  -0.9908420  0.1783093  -5.557 2.75e-08 ***
hut6                                  -1.1492457  0.1921662  -5.980 2.22e-09 ***
hut7                                  -1.4119923  0.1844370  -7.656 1.92e-14 ***
sleeper2                              -0.1441138  0.1894002  -0.761 0.446719
sleeper3                               0.1593957  0.1906211   0.836 0.403047
sleeper4                               0.0880827  0.1966041   0.448 0.654138
sleeper5                               0.5207049  0.1795870   2.899 0.003738 **
sleeper6                               0.3354179  0.1861006   1.802 0.071491 .
sleeper7                               0.5837013  0.1763917   3.309 0.000936 ***
day2                                  -0.7682702  0.5383765  -1.427 0.153576
day3                                  -0.2172527  0.4005852  -0.542 0.587585
day4                                   0.3238629  0.4541270   0.713 0.475750
day5                                  -0.3809665  0.4612852  -0.826 0.408872
day6                                   0.3210389  0.4113972   0.780 0.435178
...
day47                                 -0.4420176  0.4107280  -1.076 0.281846
day48                                  0.3312319  0.3657527   0.906 0.365139
day49                                 -0.5292627  0.4886953  -1.083 0.278803
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here, the output has been truncated: in particular, we've dropped the fixed-effect parameters for days 4-46 (inclusive), for conciseness. The intercept category gives the log-odds-transformed proportion of mosquitoes killed in the trial arm with the unwashed active comparator nets, used in hut 1, with sleeper 1, on day 1. This value, -0.069, corresponds to a proportion of ≈ 0.485 (*i.e.* nearly half of the mosquitoes killed). Let's now calculate the odds-ratio between the unwashed candidate and the unwashed active comparator nets. The OR is simply the exponent of the offset parameter for the unwashed candidate net (the estimate of this parameter is given above as -0.0286...). To calculate this, we need to extract the desired parameter from the fitted model (`fit1`). This can be done using the row & column numbers of the table of coefficents. Here, we shall chose the slightly more verbose option of identifying the parameters by name, like this:

```
>coef(summary(fit1))[‘treatmentCandidate_unwashed’,"Estimate"]
-0.02866594
```

Check that you can see where this value has come from in the model output. To calculate the OR, we simply take the exponent of this value. We'll store this as `OR1`, as it is associated with model 1.

```
>OR1 <- exp(coef(summary(fit1))[‘treatmentCandidate_unwashed’,"Estimate"])
```

In this example, `OR1=0.9717...`. We can use the standard error of this parameter estimate to form the 95% confidence interval like this:

```
>OR1_lower <- exp(coef(summary(fit1))[‘treatmentCandidate_unwashed’,‘Estimate’] -
             1.96*coef(summary(fit1))[‘treatmentCandidate_unwashed’,‘Std. Error’])
>OR1_upper <- exp(coef(summary(fit1))[‘treatmentCandidate_unwashed’,‘Estimate’] +
             1.96*coef(summary(fit1))[‘treatmentCandidate_unwashed’,‘Std. Error’])
```

In order to carry out the non-inferiority assessment, we need to set the non-inferiority margin (NIM). In these guidelines, we use a variable NIM, which depends on the level of mosquito mortality (or blood-feeding) observed in the first-in-class product (the active comparator). This is because using a fixed NIM can make it difficult to show non-inferiority when the first-in-class product is highly efficacious (*e.g.* if the mosquito mortality is >80%). Here the NIM is defined so that the proportion of mosquitoes killed by the candidate net should be no more than 7% less than the proportion killed by the active comparator. To determine the NIM, we will use the unadjusted estimates of mosquito mortality, that we calculated in Section 1:

```
FIC_mortality1 <- tab_mortality[tab_mortality$Arm==‘Active_comparator_unwashed’,
    ]$Percentage / 100
# Calculate the odds ratio (OR) for a mortality 7% lower than this one:
non_inf_margin1 <- ((FIC_mortality1 - 0.07) / (1- (FIC_mortality1 - 0.07))) /
    (FIC_mortality1 / (1- FIC_mortality1))
```

This gives a NIM of 0.7183... . We have developed a user-defined function to make the non-inferiority assessment and produce a visualisation of it. The function is called `plot_NI_OR()`, and must be provided with the OR and the NIM (see Appendix A for full function definition):

```
plot_NI_OR(OR = OR1, ORl = OR1_lower, ORu = OR1_upper, mortality = 1,
        NIM = non_inf_margin1, precision = 3)
```

Calling this function, will also lead to the outcome of the non-inferiority assessment being printed to the console. In this case, it produces this:

```
[1] "OR=0.972 [0.715, 1.321]"
[1] "NIM: 0.718"
[1] "NOT non-inferior"
```

In this case, the lower confidence interval (0.715...) is slightly less than the NIM (0.7183). So we cannot say that the unwashed candidate net is non-inferior to the unwashed active comparator. This assessment is summarised in Figure 2.2B: however, as the 95% CI only just passes below the NIM, the outcome is not so clear in the figure. Therefore, we must use the numerical values for the 95% CI of the OR and the NIM to make the assessment.

In addition to the non-inferiority assessment, we must also check that the unwashed candidate net is superior to the unwashed standard comparator. We can do this using the same regression model used above, but it will be more straightforward if we change the baseline category first, using the `relevel()` function:

```
df$treatment <- relevel(df$treatment, ‘Standard_comparator_unwashed’)
> levels(df$treatment) # check the levels of the factor
[1] "Standard_comparator_unwashed" "Active_comparator_unwashed"
[3] "Active_comparator_washed"     "Candidate_unwashed"
[5] "Candidate_washed"             "Control"
[7] "Standard_comparator_washed"
```

Now it is easier to see how the other treatment arms compare to the unwashed standard comparator. Superiority can be assessed using the p-value for the fixed-effect parameter associated with the unwashed candidate net. Running the `summary()` function on the re-fitted model (here called `fit1a`, to distinguish from the original model) gives the following output (shortened for conciseness):

```
> fit1a <-
+   glm(
+     cbind(tot_dead, total - tot_dead) ~
+       treatment + hut + sleeper + day,
+     family = binomial, data = df)
> summary(fit1a)
Coefficients:
                                     Estimate Std. Error z value Pr(>|z|)
(Intercept)                        -1.3001876  0.3456476  -3.762 0.000169 ***
treatmentActive_comparator_unwashed  1.2311743  0.1855108   6.637 3.21e-11 ***
treatmentActive_comparator_washed    0.6961056  0.1808125   3.850 0.000118 ***
treatmentCandidate_unwashed          1.2025083  0.1841820   6.529 6.62e-11 ***
treatmentCandidate_washed            0.6768528  0.1901508   3.560 0.000371 ***
treatmentControl                   -0.7850601  0.2210803  -3.551 0.000384 ***
treatmentStandard_comparator_washed -0.4415561  0.2280484  -1.936 0.052838 .
hut2                               -0.3197885  0.1578667  -2.026 0.042797 *
hut3                               -0.4449210  0.1603844  -2.774 0.005536 **
...
```

For superiority, we now need to check the p-value for the parameter for the unwashed candidate net:

```
> coef(summary(fit1a))[‘treatmentCandidate_unwashed’,"Pr(>|z|)"]
[1] 6.624834e-11
```

However, the alternative hypothesis that `glm()` uses is two-sided. Therefore, we should also check the **sign** of the coefficient for the unwashed candidate net: if it is positive then the candidate net is *superior* to the unwashed standard comparator, if it is negative then it is *inferior*.

```
> coef(summary(fit1a))[‘treatmentCandidate_unwashed’,"Estimate"]
[1] 1.202508
```

In this instance, we can conclude that the unwashed candidate is superior to the unwashed standard comparator, in terms of mosquito mortality.

The assessment for the washed candidate net is very similar to that carried out for the unwashed candidate. We shall not go through it here, but it is covered in the R code. The combined analysis is slightly different, so we will discuss it in detail. Let's look at the regression model we shall use here:

```
> fit3 <-
+   glm(
+     cbind(tot_dead, total - tot_dead) ~
+       ITN + hut + sleeper + wash + day,
+     family = binomial, data = df)
```

Now, the details of the trial arm (`treatment`) are provided by a combination of two variables: `ITN` and `wash`. As usual, before we run the model we should check which value of `ITN` will be used as the baseline case:

```
> levels(df$ITN)
[1] "Active_comparator"   "Candidate"       "Control"    "Standard_comparator"
```

This is fine for this non-inferiority assessment. Now run model `fit3` and look at the output (here truncated):

```
> summary(fit3)
...
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -0.075366   0.314884  -0.239 0.810838
ITNCandidate            -0.021888   0.111285  -0.197 0.844078
ITNControl              -2.003670   0.193250 -10.368  < 2e-16 ***
ITNStandard_comparator -1.189590   0.135565  -8.775  < 2e-16 ***
hut2                    -0.323268   0.157175  -2.057 0.039711 *
hut3                    -0.446850   0.160158  -2.790 0.005270 **
...
wash                    -0.513052   0.100605  -5.100 3.40e-07 ***
```

And, as before, we construct the OR and its 95%

```
OR3 <- exp(coef(summary(fit3))['ITNCandidate',"Estimate"])
OR3_lower <- exp(coef(summary(fit3))['ITNCandidate',"Estimate"] -
                   1.96*coef(summary(fit3))['ITNCandidate','Std. Error'])
OR3_upper <- exp(coef(summary(fit3))['ITNCandidate',"Estimate"] +
                   1.96*coef(summary(fit3))['ITNCandidate','Std. Error'])
```

And calculate the NIM (note that we use the mortality table for net type, not trial arm):

```
tab_mortality_ITN <- summm(df, vec = df$ITN, td = 'tot_dead',
                           tot = 'total', table = 1)
FIC_mortality3 <- tab_mortality_ITN[tab_mortality_ITN$Arm=='Active_comparator',
                        ]$Percentage / 100
non_inf_margin3 <- ((FIC_mortality3 - 0.07) / (1- (FIC_mortality3 - 0.07))) /
                        (FIC_mortality3 / (1- FIC_mortality3))
> non_inf_margin3
[1] 0.67349
```

Combining these things together, we find that the candidate net is non-inferior to the active comparator, for mosquito mortality (combining unwashed and washed nets together).

```
> plot_NI_OR(OR = OR3, ORl = OR3_lower, ORu = OR3_upper, mortality = 1,
+            NIM = non_inf_margin3, precision = 3)
[1] "OR=0.978 [0.787, 1.217]"
[1] "NIM: 0.673"
[1] "Non-inferior"
```

2.2.2 Blood Feeding

The procedure for assessing non-inferiority for blood feeding is extremely similar to that illustrated above for mosquito mortality. The only substantive difference stems from the fact that the effect of the vector control products is to reduce blood feeding, compared to increasing mosquito mortality. Reducing blood feeding is sometimes described as increasing blood-feeding inhibition. We will try to keep our descriptions clear in what follows.

To illustrate the process, we will go through Assessment 4 (comparing the unwashed candidate net to the unwashed active comparator). Here we write down the regression model to be used, first checking that the model has the correct baseline treatment arm:

```
df$treatment <- relevel(df$treatment, 'Active_comparator_unwashed')

fit4 <-
  glm(
    cbind(tot_bf, total - tot_bf) ~
      treatment + hut + sleeper + day,
    family = binomial, data = df)
summary(fit4)
```

Notice that now we are looking at the proportion of collected mosquitoes that have blood fed. Let's take a look at some of the model output:

```
summary(fit4)
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                          -1.121312   0.284960  -3.935 8.32e-05 ***
treatmentStandard_comparator_unwashed  0.084849   0.180408   0.470 0.638128
treatmentActive_comparator_washed      0.587605   0.164835   3.565 0.000364 ***
treatmentCandidate_unwashed           -0.009009   0.182133  -0.049 0.960548
treatmentCandidate_washed              0.643385   0.172984   3.719 0.000200 ***
treatmentControl                       0.956698   0.160061   5.977 2.27e-09 ***
treatmentStandard_comparator_washed    0.473101   0.175575   2.695 0.007048 **
hut2                                  -0.183247   0.153882  -1.191 0.233723
hut3                                  -0.259986   0.155708  -1.670 0.094978 .
...
```

We calculate the odds ratio and 95% CIs in exactly the same way as for mosquito mortality:

```
OR4 <- exp(coef(summary(fit4))[‘treatmentCandidate_unwashed’,"Estimate"])
OR4_lower <- exp(coef(summary(fit4))[‘treatmentCandidate_unwashed’,"Estimate"] -
            1.96*coef(summary(fit4))[‘treatmentCandidate_unwashed’,’Std. Error’])
OR4_upper <- exp(coef(summary(fit4))[‘treatmentCandidate_unwashed’,"Estimate"] +
            1.96*coef(summary(fit4))[‘treatmentCandidate_unwashed’,’Std. Error’])
```

We now use the proportion of mosquitoes blood fed in the unwashed active comparator arm to set the NIM. Recall that we use the estimate taken directly from the dataset for this, rather than the regression model:

```
FIC_bf4 <- tab_bf[tab_bf$Arm==’Active_comparator_unwashed’,]$Percentage / 100
non_inf_margin4 <-
        ((FIC_bf4 + 0.07) / (1- (FIC_bf4 + 0.07))) / (FIC_bf4 / (1- FIC_bf4))
```

Note that now the OR for the NIM is selected based on blood feeding being no more than 7% higher for the unwashed candidate net. For non-inferiority, the entire 95% CI for the OR must lie *below* the NIM. The user can assess this either by inspection of the values calculated above, or by using the `plot_NI_OR()` function. We set the argument `mortality` equal to 0, to indicate that this assessment is for blood-feeding inhibition:

```
plot_NI_OR(OR = OR4, ORl = OR4_lower, ORu = OR4_upper, mortality = 0,
           NIM = non_inf_margin4, precision = 3)
[1] "OR=0.991 [0.694, 1.416]"
[1] "NIM: 1.541"
[1] "Non-inferior"
```

Hence we have demonstrated non-inferiority of the unwashed candidate net compared to the unwashed active comparator, for blood feeding.

We should also check that the unwashed candidate net is superior to the unwashed standard comparator:

```
#Change baseline treatment arm
df$treatment <- relevel(df$treatment, 'Standard_comparator_unwashed')
levels(df$treatment)
[1] "Standard_comparator_unwashed" "Active_comparator_unwashed"
[3] "Active_comparator_washed"     "Candidate_unwashed"
[5] "Candidate_washed"     "Control"
[7] "Standard_comparator_washed"
fit4a <-
  glm(
    cbind(tot_dead, total - tot_dead) ~
      treatment + hut + sleeper + day,
    family = binomial, data = df)
summary(fit4a)
#Check the p-value
coef(summary(fit4a))[`treatmentCandidate_unwashed`,"Pr(>|z|)"]
```

Here we see that the relevant p-value is much less than 0.05. Finally we should check that the coefficient is negative (*i.e.* the candidate net is superior, not inferior):

```
coef(summary(fit4a))[`treatmentCandidate_unwashed`,"Estimate"] < 0
[1] TRUE
```

Although we have not gone through all 6 non-inferiority assessments listed at the start of this Section, the examples given should allow the user to generate the remaining assessments (they are fully specified in the R code which accompanies this tutorial). Table 2.1 summarises the assessments, for completeness.

2.3 Summarising and plotting the data

Often when carrying out a non-inferiority (or superiority) assessment, it is useful to also provide a summary of the trial data (total number of mosquitoes collected in each arm, model-adjusted mosquito mortality, *etc.*). We have developed some functions to tabulate and visualise the data.

However, there are some subtleties around interpreting the output of the regression models, which we will discuss.

Let's first look at the mosquito mortality estimated directly from the dataset. For each trial arm, this is simply the total number of dead mosquitoes divided by the total number of mosquitoes. For this dataset, we find:

```
> summm(df, vec = df$treatment, td = 'tot_dead', tot = 'total')
[1] "Control: 5.99%"
[1] "Standard_comparator_unwashed: 11.61%"
[1] "Standard_comparator_washed: 8.75%"
[1] "Active_comparator_unwashed: 34.09%"
[1] "Active_comparator_washed: 21.45%"
[1] "Candidate_unwashed: 28.99%"
[1] "Candidate_washed: 22.54%"
```

Now let's compare this to the estimates generated from the regression model. The function `mFE()` has been designed to do this for all trial arms (see Appendix A for full details):

```
> mFE(model = fit1, vec = df$treatment, intercept = 'Active_comparator_unwashed',
    bfi = 0, name = 'treatment')
                          Arm Mortality Lower_95pc_CI Upper_95pc_CI
2                     Control     0.111         0.059         0.197
1   Active comparator unwashed     0.483         0.333         0.635
3 Standard comparator unwashed     0.214         0.127         0.338
4   Standard comparator washed     0.149         0.084         0.250
5     Active comparator washed     0.353         0.223         0.510
6           Candidate unwashed     0.476         0.316         0.640
7             Candidate washed     0.349         0.214         0.514
```

These results are quite different to the data-derived estimates. This is because we have not considered the role of the other fixed effects- day, hut and sleeper. To be precise, the estimates above are the model-estimated mortalities in each treatment arm in hut 1, day 1 and with sleeper 1. If the observed mortalities for this combination of fixed effects is not typical of that observed across the whole trial, then the summarised mortalities may look at bit strange. We could instead present all the model parameters, which would provide a comprehensive overview of the fitted model. However, in this case we have 6 parameters for hut, 6 parameters for sleeper and 48 parameters for day. So it is a lot of information to present- and a lot of information for the reader to absorb. Let's think a bit more carefully about how we set up the model, and how we choose the baseline categories for the fixed effects in our model. From a logical point of view, it should make no difference which categories are chosen as the baseline categories for the model. As we have 7 huts, 7 sleepers and 49 trial days, there are $7 \times 7 \times 49 = 2401$ ways of setting up the baseline category for the model. We have developed a bespoke function that looks at all these permutations for the baseline category for the model and calculates the estimates (on the log-odds scale) for the mosquito mortality for the baseline category, across all possible 2401 estimates. The function returns an 'offset', which adjusts the model output to return the *median* value for the mosquito mortality. We demonstrate the procedure as follows:

```
> ofs1 <- new_median_FE(model = fit1, FE = c('hut','sleeper','day'))
> ofs1
[1] -0.9085774
```

The argument `FE` must list all the fixed effects we wish to consider. We now re-run the function `mFE()`, with this offset as one of the arguments:

```
> mFE(model = fit1, vec = df$treatment, intercept = 'Active_comparator_unwashed',
            bfi = 0, name = "treatment", offset = ofs1)
                           Arm Mortality Lower_95pc_CI Upper_95pc_CI
2                      Control     0.048         0.025         0.090
1   Active comparator unwashed     0.273         0.168         0.413
3 Standard comparator unwashed     0.099         0.055         0.170
4   Standard comparator washed     0.066         0.036         0.118
5     Active comparator washed     0.181         0.104         0.295
6           Candidate unwashed     0.268         0.157         0.417
7             Candidate washed     0.178         0.099         0.299
```

These values are closer to the values directly from the dataset. The output from this function could be used to generate a table to save to file *e.g.*

```
mk1 <- mFE(model = fit1, vec = df$treatment,
        intercept = 'Active_comparator_unwashed',
        bfi = 0, name = "treatment", offset = ofs1)
#save as a .csv file
write.csv('Table_mosquito_mortality.csv', mk1)
```

The R code that accompanies this tutorial contains an additional function `tidy_blf_FE()`, which produces a more comprehensive summary of the trial, including mosquito mortality, blood feeding inhibition and mosquito counts in each arm.

Now let's prepare the data for a visualisation. If we only wish to show the data from unwashed ITNs, we can remove the washed arms:

```
mk1a <- mk1[-grep(" washed", mk1$Arm),]
#Determine the plotting order
mk1a$ord <- c(1,3,2,4) #Should match the order of the labels in the legend
p1 <- ggplot(data = mk1a) +
  geom_errorbarh(aes(y = ord, xmin = Lower_95pc_CI,
    xmax = Upper_95pc_CI), height = 0) +
  geom_point(aes(y=ord, x=Mortality, colour = Arm), size = 3) +
  xlim(c(0,1)) + xlab('Proportion of mosquitoes blood fed') +
  theme_classic() + ylab('') +
  theme(axis.line.y = element_blank(),
    axis.ticks.y = element_blank(), axis.text.y = element_blank()) +
  scale_color_discrete(breaks = c('Candidate unwashed',
    'Active comparator unwashed', 'Standard comparator unwashed','Control')) +
  theme(legend.position = c(0.8,0.3)) + labs(color = '') +
  ggtitle('Mosquito mortality (unwashed ITNs)')
```

Here we've saved the plot as `p1`. If we save the non-inferiority assessment as `NI_1`, we can display the plots together:

```
NI_1 <- plot_NI_OR(OR = OR1, ORl = OR1_lower, ORu = OR1_upper, mortality = 1,
        NIM = non_inf_margin1, precision = 3)
plot_grid(p1,NI_1,nrow = 1, rel_widths = c(0.6,0.4), labels = c('A','B'))
```

These plots are shown in Figure 2.2.

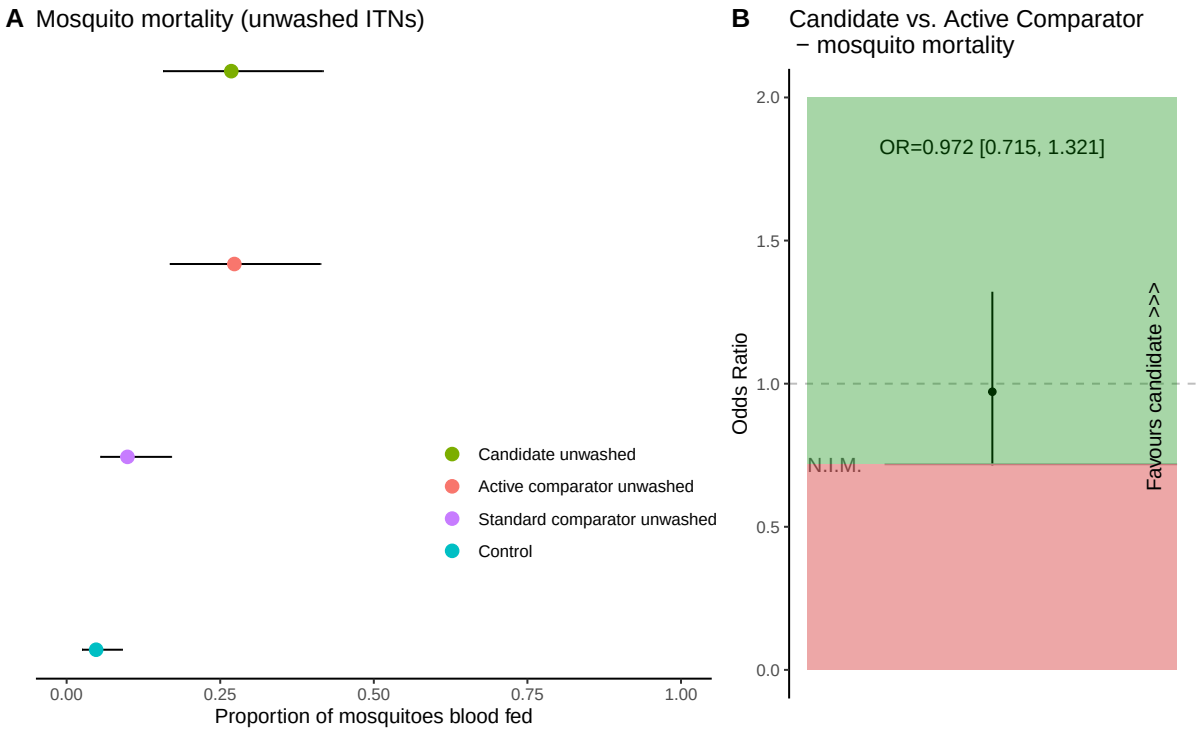

Figure 2.2: A visualisation of the non-inferiority assessment: comparing the unwashed candidate net to the unwashed active comparator. Panel A: mortality estimates (and 95% confidence intervals) in the arms containing unwashed nets, as well as the untreated control. Panel B: A summary of the non-inferiority assessment between the unwashed candidate net and the unwashed active comparator (for mosquito mortality). In this instance, the non-inferiority margin (N.I.M., horizontal grey line) is very close to the 95% CI: therefore, the numerical values of these quantities must be inspected, to make the non-inferiority assessment (see Table 2.1).

	Assessment	N.I.M.	Odds Ratio (95% CI)	Superior to the standard comparator?
1.	Mortality: unwashed candidate *vs.* unwashed active comparator	0.718	0.972 [0.715, 1.321]	Yes
2.	Mortality: washed candidate *vs.* washed active comparator	0.619	0.981 [0.717, 1.341]	Yes
3.	Mortality: candidate *vs.* active comparator (combined)	0.673	0.978 [0.787, 1.217]	Yes
4.	Blood feeding: unwashed candidate *vs.* unwashed active comparator	1.541	0.991 [0.694, 1.416]	No
5.	Blood feeding: washed candidate *vs.* washed active comparator	1.404	1.057 [0.795, 1.406]	No
6.	Blood feeding: candidate *vs.* active comparator (combined)	1.446	1.024 [0.821, 1.278]	No

Table 2.1: A summary of the six non-inferiority assessments carried out on the dataset used for this tutorial. We show the non-inferiority margin (N.I.M.) used in each case. For mosquito mortality, the entire 95% confidence interval for the odds ratio must lie above the N.I.M. for non-inferiority to be shown. For blood-feeding, the entire 95% confidence interval must lie below the N.I.M. for non-inferiority to be shown. We also check that the candidate net is superior to the standard comparator (rightmost column).

Chapter 3

Analysis in STATA

3.1 Loading & summarising the dataset

In this chapter, we will repeat the same non-inferiority assessments that we carried out in R, for the same dataset but this time in STATA[1]. We first load the data (saved as a `.csv` file) using the `import` command:

```
import delimited example_dataset.csv
```

Note that you may need to change the current working directory before you do this, *e.g.*

```
cd C:\Users\username\Documents
```

Here the file path should match the location of the materials for this tutorial. You can run the command `pwd` if you need to check the current working directory. Once the data is loaded, we can take a look at it. The command `tab` gives the frequency counts for a numerical variable:

```
. tab tot_dead
```

tot_dead	Freq.	Percent	Cum.
0	117	34.11	34.11
1	81	23.62	57.73
2	48	13.99	71.72
3	32	9.33	81.05
4	17	4.96	86.01
5	16	4.66	90.67
6	10	2.92	93.59
7	7	2.04	95.63
8	3	0.87	96.50
9	4	1.17	97.67
10	3	0.87	98.54
11	2	0.58	99.13
14	1	0.29	99.42

[1]These analysis have been run in both STATA version 13 and STATA version 19.

```
        15 |          2        0.58      100.00
-----------+-----------------------------------
     Total |        343      100.00
```

Whereas the command `summarize` gives the summary statistics for the variable:

```
. summarize tot_dead

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    tot_dead |       343           2    2.575185          0         15
```

The command `levelsof` returns a list of the values of a variable, *e.g.*

```
. levelsof(treatment)
`"Active_comparator_unwashed"' `"Active_comparator_washed"'
`"Candidate_unwashed"' `"Candidate_washed"' `"Control"'
`"Standard_comparator_unwashed"' `"Standard_comparator_washed"
```

3.2 Making the non-inferiority assessment

3.2.1 Mosquito mortality

Now we will reproduce the non-inferiority assessments we carried out in R in the previous chapter, and check that we can obtain the same results. We will start by comparing the unwashed candidate to the unwashed active comparator in terms of mosquito mortality (this was Assessment 1 in the previous chapter). Our first step will be to calculate the NIM. We use the `collapse` function to sum together the mosquito counts (both total mosquitoes and dead mosquitoes) for the various treatment arms. We then calculate the (unadjusted) mosquito mortality observed in each arm:

```
collapse (sum) sum1=tot_dead sum2=total, by(treatment)
gen prop_dead = sum1/sum2
list
```

```
     +-------------------------------------------------------+
     |                    treatment   sum1   sum2   prop_d~d |
     |-------------------------------------------------------|
  1. |   Active_comparator_unwashed    150    440   .3409091 |
  2. |     Active_comparator_washed    133    620   .2145161 |
  3. |           Candidate_unwashed    147    507   .2899408 |
  4. |             Candidate_washed    112    497   .2253521 |
  5. |                      Control     42    701   .0599144 |
     |-------------------------------------------------------|
  6. | Standard_comparator_unwashed     60    517   .1160542 |
  7. |   Standard_comparator_washed     42    480      .0875 |
     +-------------------------------------------------------+
```

Recall that the NIM here should be set so that the proportion of mosquitoes killed by the unwashed candidate net should be no more than 7% less than the proportion killed by the unwashed active comparator. We construct the NIM as follows:

```
gen or1 = (prop_dead - 0.07)/(1-prop_dead + 0.07)
gen or2 = (prop_dead)/(1-prop_dead)
*Calculate the odds-ratio (OR) for the NIM
gen nim = or1/or2
list
```

treatment	sum1	sum2	prop_d~d	or1	or2	nim
Active_co~unwashed	150	440	.3409091	.3715711	.5172414	.7183707
Active_co~washed	133	620	.2145161	.1689291	.2731006	.61856
Candidate_unwashed	147	507	.2899408	.2819541	.4083333	.6904997
Candidate_washed	112	497	.2253521	.1839253	.2909091	.6322432
Control	42	701	.0599144	-.0099849	.0637329	-.1566677
Standard_~unwashed	60	517	.1160542	.0482775	.131291	.3677139
Standard_~washed	42	480	.0875	.0178117	.0958904	.1857506

Note that, in this case, it is the NIM for the unwashed active comparator that we need to extract from this table (`0.718307...`). We will now reload the original dataset, to fit the regression model. We will first save the information generated above, and then append it to the full dataset:

```
% save in the current working directory
save "aggregated_mortality.dta"
clear
import delimited example_dataset.csv
append using aggregated_mortality.dta
% we can drop the variables that we won't use again
drop sum1 sum2 prop_dead or1 or2
```

Note that we will also need to make an equivalent table for blood-feeding (this can be found in the STATA code that accompanies this tutorial). We are nearly ready to fit the regression model. However, STATA has imported `treatment` as a 'string' variable. We will need to generate an equivalent 'factor' variable to include in the regression model. We will call this `treatment2`:

```
encode(treatment), generate(treatment2)
\*We only want treatment2 values for the original dataset
(not the values we've appended beneath):*/
replace treatment2=. if day==.
```

Here we can see how the levels of `treatment2` correspond to `treatment`:

```
. label list treatment2
treatment2:
           1 Active_comparator_unwashed
           2 Active_comparator_washed
           3 Candidate_unwashed
           4 Candidate_washed
           5 Control
           6 Standard_comparator_unwashed
           7 Standard_comparator_washed
```

We will use the `blogit` command to fit the regression model, as this fits a logistic regression model to aggregated count data. Including fixed-effects for treatment, hut, sleeper, and day, the model has the following form:

```
blogit tot_dead total i.treatment2 i.hut i.sleeper i.day
```

Remember that for these models it is important to consider which `treatment2` category is used as the baseline. By default, category 1 is used, which in this case is the best one for us to use. We will show an example later, where we will manually change the baseline category. Once the model is run, the results will be stored in STATA's memory. We won't explore the whole model contents here, but it can be viewed by running the command `ereturn list`. Now we will construct the ORs for the non-inferiority assessment. We need to extract the parameter estimate for the unwashed candidate net category for `treatment`. This is category 3 of `treatment2`. STATA stores the estimates for the fixed effects in the container `_b[]`, whilst the standard errors for these parameters are stored in `_se[]`. We extract the OR and its 95% CI like this:

```
gen or_model = exp(_b[_outcome:3.treatment2])
gen or_model_lower = exp(_b[_outcome:3.treatment2] - 1.96* _se[_outcome:3.treatment2])
gen or_model_upper = exp(_b[_outcome:3.treatment2] + 1.96* _se[_outcome:3.treatment2])
```

The `display` command will print these values to the screen:

```
. display or_model
.97174102
. display or_model_lower
.71467191
. display or_model_upper
1.3212785
```

An alternative way to extract the OR is to request the ORs when fitting the model (output truncated):

```
. blogit tot_dead total i.treatment2 i.hut i.sleeper i.day, or

------------------------------------------------------------------------------
         _outcome | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
------------------------------+-----------------------------------------------
       treatment2 |
  Active~_washed  |   .5856291   .0904798    -3.46   0.001     .4326251     .792745
Candid~_unwashed  |    .971741   .1523381    -0.18   0.855      .714676    1.321271
Candidate_washed  |   .5744619   .0950874    -3.35   0.001     .4153045    .7946132
         Control  |   .1331559   .0269214    -9.97   0.000     .0895912    .1979046
Standard~unwashed |   .2919496   .0541598    -6.64   0.000     .2029555    .4199667
  Standard~washed |   .1877338   .0382263    -8.21   0.000     .1259566    .2798104
                  |
              hut |
               2  |   .7263026    .114659    -2.03   0.043     .5330172    .9896783
               3  |   .6408749   .1027864    -2.77   0.006     .4680086    .8775922
...
```

Regardless of which method we use, we can see that the 95% CI `[.714676-1.321271]` is not entirely above the NIM we calculated above (`0.718307...`). Therefore, we cannot conclude that the unwashed candidate is non-inferior to the unwashed active comparator, in terms of mosquito mortality. This is consistent with what we found in R (Table 2.1). For completeness, however, we will check whether the unwashed candidate is superior to the unwashed standard comparator. We can use the same regression model again: but it will be simpler if we change the `treatment2` baseline category to be the unwashed standard comparator. This is level 6 of `treatment2` (output truncated):

```
. blogit tot_dead total ib6.treatment2 i.hut i.sleeper i.day
------------------------------------------------------------------------------
         _outcome |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------------------+---------------------------------------------------
       treatment2 |
Active~_unwashed  |  1.231174   .1855108     6.64   0.000     .8675798    1.594769
Active_c~_washed  |  .6961056   .1808125     3.85   0.000     .3417195    1.050492
Candidate_unwashed|  1.202508    .184182     6.53   0.000     .8415183    1.563498
 Candidate_washed |  .6768528   .1901508     3.56   0.000     .3041641    1.049541
         Control  | -.7850601   .2210804    -3.55   0.000     -1.21837   -.3517505
Standard_~_washed | -.4415561   .2280484    -1.94   0.053    -.8885226    .0054105
                  |
              hut |
                2 | -.3197885   .1578667    -2.03   0.043    -.6292016   -.0103754
                3 |  -.444921   .1603844    -2.77   0.006    -.7592687   -.1305733
```

Here we see that the coefficient for the unwashed candidate is positive, and that the p-value is <0.001. So we can conclude that the unwashed candidate net is superior to the unwashed standard comparator, in terms of mosquito mortality. We won't generate the arm-level mortality estimates here, but we will show an example of how to generate these from the fitted regression model. STATA denotes the intercept for the regression model with the label `_cons`. We can calculate the mosquito mortality for the intercept category in the most-recently fitted regression model using the `invlogit` command:

```
. display invlogit(_b[_cons])
.21413345
```

So this means that the estimated mosquito mortality in the unwashed standard comparator arm, on day 1 in hut 1 with sleeper 1 is about 21.4%.

3.2.2 Blood feeding

The procedure for assessing non-inferiority for mosquito blood feeding is extremely similar to that for mosquito mortality. However, we shall go through the process here, for completeness. As we assessed unwashed nets for mosquito mortality, we will perform the combined analysis here (this is Assessment 6 in the code). As before, our first step is to calculate the NIM. Note that we now group the data by the variable `itn`:

```
clear
import delimited "example_dataset.csv"
collapse (sum) sum1=tot_bf sum2=total, by(itn)
```

```
gen prop_fed = sum1/sum2
/*Note: NIM chosen by considering slightly higher blood feeding for the
candidate net (compared to slightly lower mosquito mortality)*/
gen or1 = (prop_fed + 0.07)/(1-prop_fed - 0.07)
gen or2 = (prop_fed)/(1-prop_fed)
*Calculate the odds-ratio (OR) for the NIM
gen nim = or1/or2
list
save "aggregated_bf_itn.dta"
```

Now we reload the data, and append the NIM, as before:

```
clear
import delimited "example_dataset.csv"
append using "aggregated_bf_itn.dta"
*Remove variables we don't need anymore
drop sum1 sum2 prop_fed or1 or2
```

Now we generate a new factor variable, `itn2`, to use in the regression model:

```
encode(itn), generate(itn2)
replace itn2=. if day==.
```

For the regression model, we again use the command `blogit`. This time we use `tot_bf` in the model, which is the number of mosquitoes that were blood fed:

```
blogit tot_bf total i.itn2 i.hut i.sleeper i.day i.wash
```

Note that we have included an additional fixed effect for the washed status of the net. Now we calculate the OR and its 95% confidence intervals

```
gen or_model = exp(_b[_outcome:2.itn2])
gen or_model_lower = exp(_b[_outcome:2.itn2] - 1.96* _se[_outcome:2.itn2])
gen or_model_upper = exp(_b[_outcome:2.itn2] + 1.96* _se[_outcome:2.itn2])
```

Finally, we extract the relevant NIM from the dataset with the following command:

```
list if itn=="Active_comparator" & missing(day)
```

This yields an NIM of 1.44578. Comparing it to the 95% CI for the OR:

```
. display or_model_lower
.82099819
. display or_model_upper
1.2784339
```

we can see that we have demonstrated non-inferiority for blood feeding. We should also check for superiority to the standard comparator. We can do this with the same regression model, changing the baseline category for `itn2` (output truncated):

```
. blogit tot_bf total ib4.itn2 i.hut i.sleeper i.day i.wash
------------------------------------------------------------------------------
     _outcome |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
         itn2 |
Active_com~r  |   .0264738   .1149307     0.23   0.818    -.1987863    .2517339
   Candidate  |   .0506745   .1182441     0.43   0.668    -.1810796    .2824287
     Control  |   .9533925   .1320353     7.22   0.000     .6946081    1.212177
              |
          hut |
            2 |  -.1709663   .1532829    -1.12   0.265    -.4713952    .1294627
```

Here we find that the candidate is not superior to the standard comparator for blood feeding. This matches our findings in the R analysis (Table 2.1).

Appendix A

A description of the R functions developed for this work

- `plot_NI_OR()`
  - Description: Carries out a non-inferiority assessment, and produces the corresponding visualisation
  - Usage: `plot_NI_OR(OR, ORl, ORu, mortality, NIM, precision, title)`
  - Arguments:
    - `OR, ORl, ORu`: These are: the odds ratio estimate, and the lower and upper estimates of its 95% CI, respectively
    - `mortaltiy`: Set equal to 1 if the non-inferiority assessment is for mosquito mortality; set to zero for blood feeding. The default option is 1
    - `NIM`: The non-inferiority margin. This should be in the form of an odds ratio. For blood-feeding, a warning will be issued if `NIM<1`. Similarly, for mosquito mortality a warning will be issued if `NIM>1`
    - `precision`: the number of decimal places to be used for the non-inferiority summary that is printed on the visualisation. Note: unrounded values are used for the odds-ratios in the actual non-inferiority assessment.
    - The title to be used for the visualisation
- `summm()`
  - Description: A simple function to summarise the mosquito mortality or blood feeding observed in each trial arm
  - Usage: `summm(data, vec, td, tot, table, precision)`
  - Arguments:
    - `data`: Dataset to be analysed
    - `vec`: Variable in the dataset we wish to use to stratify the data. If this variable is `treatment` and the dataset is `df`, we write: `vec=df$treatment`
    - `td`: This is the numerator for the summary. For mosquito mortality, this is `'tot_dead'`; for blood-feeding it is: `'tot_bf'`
    - `tot`: This is the denominator for the summary. Here this will be `total`

    * `table`: This variable determines the form of the output. For `table=0`, sentences will be printed in the R console; otherwise the function returns a data frame
    * `precision`: number of decimal places in the output. Note that this function reteruns percentages, rather than proportions

- `mFE()`
  - Description: A function that takes a fitted regression model and returns the mortality (blood feeding) estimates, along with their 95% CIs.
  - Usage: `mFE(model, vec, intercept, bfi, name, offset, precision)`
  - Arguments:
    * `model`: Fitted regression model
    * `vec`: Variable in the dataset we wish to use to stratify the data. If this variable is `treatment` and the dataset is `df`, we write: `vec=df$treatment`
    * `intercept`: In the regression model output, the treatment arm that pertains to the baseline category is not named. So we must enter it here
    * `bfi`: If `bfi=0`, the function will assume the model is estimating mosquito mortality. For `bfi=1`, it will assume the model is estimating blood feeding. The default value is 0.
    * `name`: Should match the variable entered in the argument `vec`. So if `vec=df$treatment`, we write `name = 'treatment'`
    * `offset`: Adjustment factor (on the log-odds scale) to the mortality (or blood-feeding) measured in the baseline category (Default value is 0)
    * `precision`: Number of decimal places to use for the output (the default is 3)

- `new_median_FE()`
  - Description: A function which looks at the permutations of fixed-effects that gives a representative mortality (or blood feeding) estimate for the baseline category. This is done by calculating the median value, across all permutations
  - Usage: `new_median_FE(model, FE)`
  - Arguments
    * `model`: Fitted regression model
    * `FE`: At the moment, this argument can only take lists of containing 1,2, or 3 fixed effects. The default list is `FE = c('hut','sleeper','day')`

- `tidy_blf_FE()`
  - Description: Once regression models for mortality and blood feeding have been fitted (`fit1` and `fit4` in this analysis), this function can generate a summary table for the overall trial
  - Usage: `tidy_blf_FE(data, model_fit, vec, intercept, name, first_cat, model_fit_blf, offset)`
  - Arguments:
    * `data`: Dataset to be analysed
    * `model_fit`: Fitted regression model for mosquito mortality

* `vec`: Variable in the dataset we wish to use to stratify the data. If this variable is `treatment` and the dataset is `df`, we write: `vec=df$treatment`
* `intercept`: In the regression model output, the treatment arm that pertains to the baseline category is not named. So we must enter it here
* `name`: Should match the variable entered in the argument `vec`. So if `vec=df$treatment`, we write `name = 'treatment'`
* `first_cat`: Treatment arm to be listed first in the output. This will also be the treatment arm against which blood-feeding inhibition is measured. Hence, it is ususally the untreated control
* `model_fit_blf`: Fitted regression model for mosquito blood feeding
* `offset`: A vector of length two, giving the offsets to use to calculate the mortality and blood feeding from the respective regression models. The default is `offset=c(0,0)`