Statistical analyses for long-term community studies for ITNs

Joseph D. Challenger, on behalf of the World Health Organisation

February 2025

Contents

Chapter 1

Introduction

This tutorial is designed to provide guidance on carrying out statistical analyses required for long-term community studies of insecticide-treated nets (ITNs). These studies are designed to test whether ITNs are "acceptable to the study community, and durable under routine conditions of use to demonstrate three years of functional ITN survival"[1]. As part of these studies, ITNs must demonstrate entomological efficacy *via* a non-inferiority semi-field study, using operationally-aged nets.

This tutorial builds upon previous materials, designed to provide guidance on non-inferiority assessments for vector control products [1]. That tutorial built on earlier work [2], which itself was influenced by the methodology presented in Ref. [3]. Here we strive to be as consistent as possible with the previous methodology. With operationally-aged ITNs, between-net heterogeneity has been shown to be considerable, and should be accounted for in the methodology. For experimental hut trials (EHTs), this factor represents the primary difference between Ref. [1] and the methodology presented here. For readers without much experience in using R, the earlier tutorial contains some information about commonly used R commands, and may be a useful reference.

The tutorial will be organised in the following way. Chapter 2 will contain a recap of the non-inferiority assessment used in Ref. [1], and Chapter 3 will outline the procedure for EHTs. Chapter 4 will outline the corresponding procedure for the Ifakara ambient chamber test (IACT), while Chapter 5 will outline the procedure for both the cone assay and the tunnel assay. In each of these chapters, we will also discuss the power calculations required to generate the guidance on sample size. This tutorial also provides guidance on calculating the median functional survival of ITNs in the field, using an established methodology from the literature (Chapter 6).

[1] Quotation taken from draft. We can update with a reference, when report is publically available

Chapter 2

Calculating non-inferiority

We shall begin this tutorial with a short recap of the non-inferiority assessment, as outlined in the earlier tutorial [1]. We note that in the earlier tutorial, we were tasked with comparing a candidate ITN against a pre-approved ITN (*i.e.* one that had already successful passed through the approval process). Here, we shall compare the field-aged ITNs with ITNs (of the same brand) that have been washed 20 times.

A key quantity in a non-inferiority trial is the non-inferiority margin (NIM). When assessing mosquito mortality, the NIM is chosen so that mosquito mortality measured for the candidate net (here, the field-aged net) must be no more than 7% lower than that measured for the active comparator (here the washed net), in order for non-inferiority to be achieved. This means that the value of the NIM will vary from trial to trial and must be calculated for each assessment.

We will now recap the non-inferiority assessment, which involves three steps: firstly, calculate the NIM; next, calculate the relevant OR (and its 95% confidence interval) generated by the regression model; thirdly, assess whether non-inferiority has been achieved. As outlined in the previous tutorial, to calculate the OR for a given scenario, we first calculate the mortality of the active comparator directly from the data set. The black curve in Figure 2.1 shows how the NIM varies with the mosquito mortality of the active comparator (y-axis). The OR, comparing the mortality induced by the field-aged nets with the washed nets, should then be calculated from the generalised linear model (GLM). This model will contain one or more fixed effects: as we will show in the tutorial, the fixed effects included will vary from one assay to another. Figure 2.1 shows an illustrative OR, along with its 95% confidence interval.

The extra complication with the analysis of field-aged nets is the introduction of between-net heterogeneity: in the community, individual nets will deteriorate at different rates. As we will show later, increasing between-net heterogeneity will lead to a decrease in study power. We can use data from previous studies to estimate a plausible range of values for this quantity, including a random effect for the individual net. However, difficulties can emerge if we try to include a random effect for the individual net *and* fixed effects for factors such as study day, hut, or sleeper. This is because for studies involving field-aged nets, each net will only be used a few times, before being swapped out for another net from the same trial arm. In the case of a EHT, this means that each individual net may not be tested in (for instance) each hut. This can lead to issues with model convergence. Therefore, when analysing datasets (both datasets from the field, and synthetic datasets, generated for purposes of power analysis), we only include the fixed effect terms (for day, hut, *etc.*). In this way, we are more consistent with the methodology presented in Ref. [1]. However, it is important that the between-net heterogeneity is accounted for during the data generating process used to create synthetic datasets. This means that, even though we analyse the synethic datasets *without*

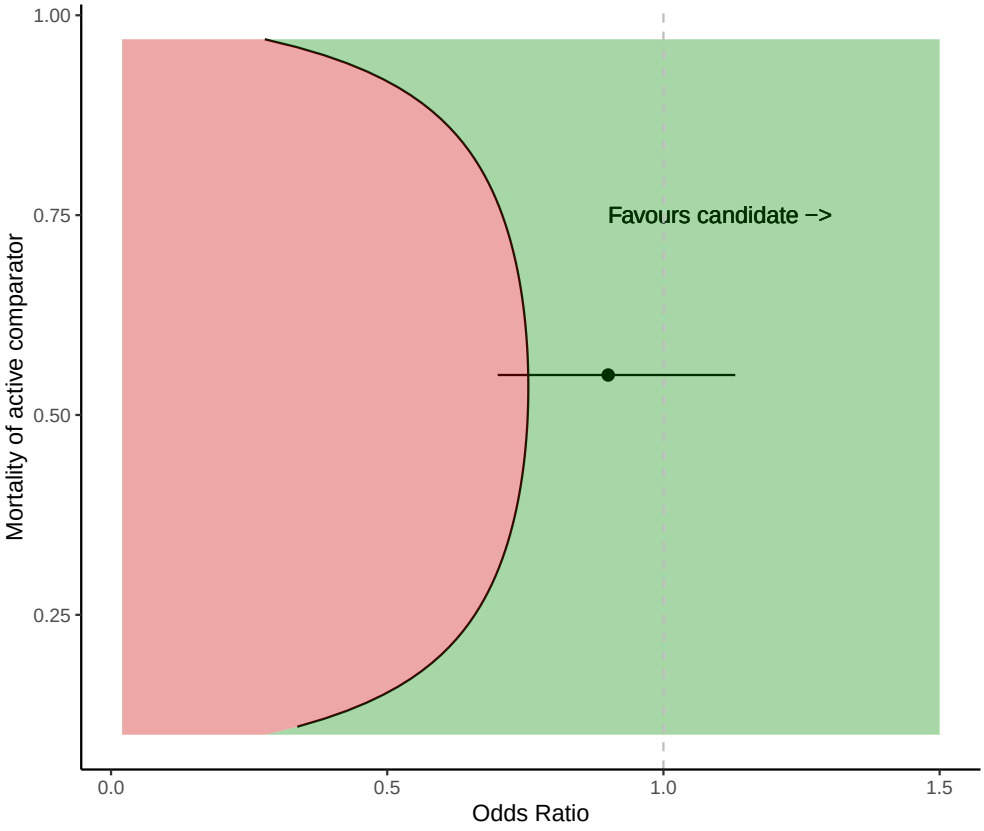

Figure 2.1: A visualisation of the non-inferiority assessment. The black curve indicates how the non-inferiority margin varies according to the mortality of the active comparator (in this case, the mortality of the washed nets). In this illustrative example, the mortality in the active comprator arm was 0.55 (NIM$\approx$ 0.755). Here we show an odds ratio of 0.90 [0.7,1.13], where the values in the square brackets are the 95% confidence interval. For non-inferiority to be satisfied, the entire 95% CI must be above the non-inferiority margin (the green region in the plot). As this is not the case here, we **cannot** conclude that the candidate is non-inferior to the active comparator, in this instance.

a random effect for net, we still see the influence that increased between-net heterogeneity has on study power. We shall show examples of this later in the tutorial. On occasion, we may wish to estimate the between-het heterogeneity itself. In which case, we would remove the fixed-effects (except for trial arm) and replace with a random effect for the individual net.

Chapter 3

Experimental hut trials

3.1 Data Analysis

We will start this chapter with an example of what typical dataset from an EHT study will look like, and give the names of the variables that we will use here.

```
> head(mosdata)
  hut net sleeper day n net_id tot_dead
   1   C       7   1  4    C_1        0
   2  E8       3   1 20   E8_4        6
   3  E7       2   1 10   E7_7        3
   4  E6       9   1 11  E6_10        8
   5  E5       4   1 18  E5_13        7
   6  E4       5   1  9  E4_16        1
```

Here `hut` and `sleeper` record the hut in which the mosquitoes were collected and the individual who slept in the hut that right, respectively. The trial arm is indicated by the `net` variable. Here, we have 9 trial arms, which we have labelled as `(C,E1,E2,E3,...,E8)`. We take these trial arms to be:

```
C: Untreated net (negative control)
E1: Unwashed Test ITN
E2: 20x washed Test ITN
E3: 12-months old Test ITN
E4: 24-months old Test ITN
E5: 36-months old Test ITN
E6: Unwashed positive control
E7: 20x washed positive control
E8: 36-months old positive control
```

The variable `n` is total number of mosquitoes collected (recall that wild, free-flying mosquitoes are used in this assay, so `n` will vary from night to night), and `tot_dead` records the number of mosquitoes that have died (normally this is recorded after 24 hours, but can vary, according to net chemistry). The `net_id` variable shows the individual net used: here we use the convention that this records the trial arm followed by a underscore then a number. So, if 30 ITNs are used for each arm, nets of brand `E1` will be labelled as `(E1_1,E1_2,E1_3,...E1_30)`.

Before fitting the regression models, we should make sure that `hut`, `day`, `sleeper`, and `net` are stored as factor variables in R, rather than numerical variables:

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

For the non-inferiority assessment, we wish to compare the field-aged (36-months-old) test net with the 20-times washed net of the same type. The assessment is simpler to carry out if the latter category is used for the model intercept:

```
> mosdata$net <- relevel(mosdata$net, 'E2') # make E2 the first level
> levels(mosdata$net) # print the levels, in order to check
[1] "E2" "C"  "E1" "E3" "E4" "E5" "E6" "E7" "E8"
```

Then we fit the regression model to the dataset, using the `glm()` function:

```
fit <- glm(
    cbind(tot_dead, n - tot_dead) ~ net + day + sleeper + hut,
    family = binomial, data = mosdata
)
```

The fitted model is stored in the object `fit`. Running the command `summary(fit)` generates a summary of the fitted parameters. For the non-inferiority assessment, we need to calculate the odds ratio (and 95% confidence interval) for the `E5` category:

```
OR <- exp(coef(summary(fit))['netE5',"Estimate"])
#lower limit of confidence interval
OR_lower <- exp(coef(summary(fit))['netE5',"Estimate"] -
                    1.96*coef(summary(fit))['netE5','Std. Error'])
#upper limit of confidence interval
OR_upper <- exp(coef(summary(fit))['netE5',"Estimate"] +
                    1.96*coef(summary(fit))['netE5','Std. Error'])
```

Now we have the OR, we need to compare it with the NIM. As discussed in Chapter 2, the NIM is calculated as follows:

```
#mortality of the active comparator
FIC_mortality <- sum(mosdata[mosdata$net=='E2',]$tot_dead)/
           sum(mosdata[mosdata$net=='E2',]$n)
# NIM selected so that mortality is at most 7% lower
non_inf_margin <- ((FIC_mortality - 0.07) / (1- (FIC_mortality - 0.07))) /
           (FIC_mortality / (1- FIC_mortality))
```

The OR and NIM can then be compared, as illustrated in Figure 2.1.

3.2 Estimating study power

In this section we will provide some guidance on powering an EHT study for non-inferiority. For the most part, the methodology is the same as outlined previously [1]. The primary difference is

that now it is important to introduce between-net heterogeneity into the data generating process. To make things easier for the user, an R function has been written to generate synthetic datasets. We quantify between-net heterogeneity in terms of the standard-deviation of a zero-mean, normally distributed random variable which is applied to the model on the log-odds scale (as are all the other model parameters).

The user-defined function used to generate synthetic EHT datasets is called `EHT_sim_CS()` (here the 'CS' denotes this is for a community study, rather than a hut trial using new and washed nets). The function takes a number of arguments, which allow the user to tailor the study to meet their needs. These arguments are:

- `n_arms`: The number of study arms. This will determine the number of huts
- `npr`: This stands for 'nights per round', and is the number of nights which a trial arm spends in a given hut, before nets are rotated
- `mos_det`: Determines whether the number of mosquitoes per hut per night is constant (`mos_det=1`). If not, mosquito counts follow a negative binomial distribution with parameters `meanMos` (mean) and `dispMos` (dispersion). The value selected for `meanMos` should be guided by mosquito densities in a given setting. For the dispersion, we use a default value of 1.5, based on previous EHTs [2].
- `verbose`: if this variable is set to 1, the function will print some useful information to the console in R. This can be useful when setting up a scenario for the first time, but you will probably want to turn this off when performing power calculations
- `rotations`: The number of rotations of the EHT to carry out. for `n_arms=9` and `npr=9`, the study will require 81 nights for one rotation (not including rest days).
- `sigma_net`: the standard deviation of the between-net heterogeneity (default value=0.9)
- `mortalities`: the vector of average mortalities expected for each arm. The length of this vector should be equal to `n_arms`, and should be ordered like this: `(C,E1,E2,E3,E4,E5,E6,E7,E8)`.
- `n_nets`: The number of individual nets to use for each arm. Following the WHO guidelines, we set this equal to 30. If you have selected `verbose=1`, running the function will produce a message that informs you of the number of times each net appears in the dataset.

Let's say that we wish to run a 9-arm EHT, for one rotation, using 30 nets per arm. In our study setting, we expect about 10 mosquitoes per hut per night. Choosing some estimated mosquito mortalities and a value of the between-net heterogeneity, we can generate a synthetic dataset like this:

```
mosdata <- EHT_sim_CS(n_arms = 9, npr = 9, mos_det = 0, meanMos = 10,
                      rotations = 1, verbose = T,
                      mortalities = c(0.05,0.6,0.25,0.5,0.35,0.25,0.60,0.25,0.25),
                      sigma_net = 0.5)
[1] "data pts per replicate: 2.7"
[1] "21 nets are tested 3 times; 9 nets are tested 2 times"
```

The printed message indicates that, for a study of this duration, some individual nets appear twice, and others appear three times. We have already outlined the non-inferiority assessment in this chapter. In the code, however, we have written the procedure as a function `EHT_NIM()`. The function takes the following arguments:

- `dataset`: the name of the data frame containing the EHT data
- `NIM_pc`: the proportion difference in mortality to use for the NIM (here we'll always use `NIM_pc=0.07` for a 7% difference)
- `verbose`: if `verbose=1`, the outcome of the non-inferiority assessment will be presented to the R console.
- `int_cat`: In the dataset, which net should be used as the active comparator (washed net, in this context)? As a default, we use `E2`.

To estimate study power, we simulate a large number of studies (here we use 1000), and test each for non-inferiority. If non-inferiority is achieved, the function returns a value of 1; if not, a value of 0 is returned.

```
n_sim <- 1000 # the number of simulations to carry out
store_power <- rep(NA, n_sim) # a container for the outcome of each simulated study
for(i in 1:n_sim){
    mosdata <- EHT_sim_CS(n_arms = 9, npr = 9, mos_det = 0,
    meanMos = 10, rotations = 1,
    mortalities = c(0.05,0.6,0.25,0.5,0.35,0.25,0.60,0.25,0.25),
    sigma_net = 0.5)

    store_power[i] <- EHT_NIM(dataset = mosdata, verbose = F)
}
```

Then the study power (as a percentage) is simply equal to `100*mean(store_power)`. As this estimate of power is based on a finite sample, we can also calculate the uncertainty associated with the estimate:

```
> binom.test(table(factor(store_power,c(1,0))))$conf.int
[1] 0.7700216 0.8296419
attr(,"conf.level")
[1] 0.95
> mean(store_power)
[1] 0.791
```

So here we have an estimated study power of 79.1% [95% CI: (77.0-83.0)].

As of yet, we haven't discussed what value we should use for `sigma_net`. Experience with historic datasets suggests that a value of `sigma_net=0.9` is a reasonably conservative choice. This value will guide the sample size recommendations used here.

We will finish this section with some guidance on increasing study power, if a single rotation of the study arms doesn't give adequate study power. One could increase the duration of the study by commencing a second rotation (things like `rotations=1.25` or `rotations=1.5` are valid options here). Another option is to consider a 12-hut study, if resources permit. With this study design, we duplicate the trial arms containing field-aged nets *i.e.* some trial arms appear in two huts each night, rather than one. One thing to note here is that our function `EHT_sim_CS()` expects the number of trial arms to match the number of huts. However, we can achieve the desired effect by relabelling the tenth, eleventh & twelfth trial arms in the following way:

```
n_sim <- 1000
store_power <- rep(NA, n_sim)
for(i in 1:n_sim){

    mosdata <- EHT_sim_CS(n_arms = 12, npr = 12, mos_det = 0,
    meanMos = 20, rotations = 1,
    mortalities = c(0.05,0.6,0.25,0.5,0.35,0.25,0.60,0.25,0.25,0.5,0.35,0.25),
    sigma_net = 0.9)

    #Re-label the extra trial arms
    mosdata[mosdata$net=="E9",]$net <- "E3"
    mosdata[mosdata$net=="E10",]$net <- "E4"
    mosdata[mosdata$net=="E11",]$net <- "E5"

    store_power[i] <- EHT_NIM(dataset = mosdata, verbose = F)
}
```

Note that the user must take care that the mortalities of the 'dummy' trial arms match that of the field-aged nets. Figure 3.1 presents some power estimates for a range of trial durations. The anticipated mosquito counts will guide the suitable choice in a given location.

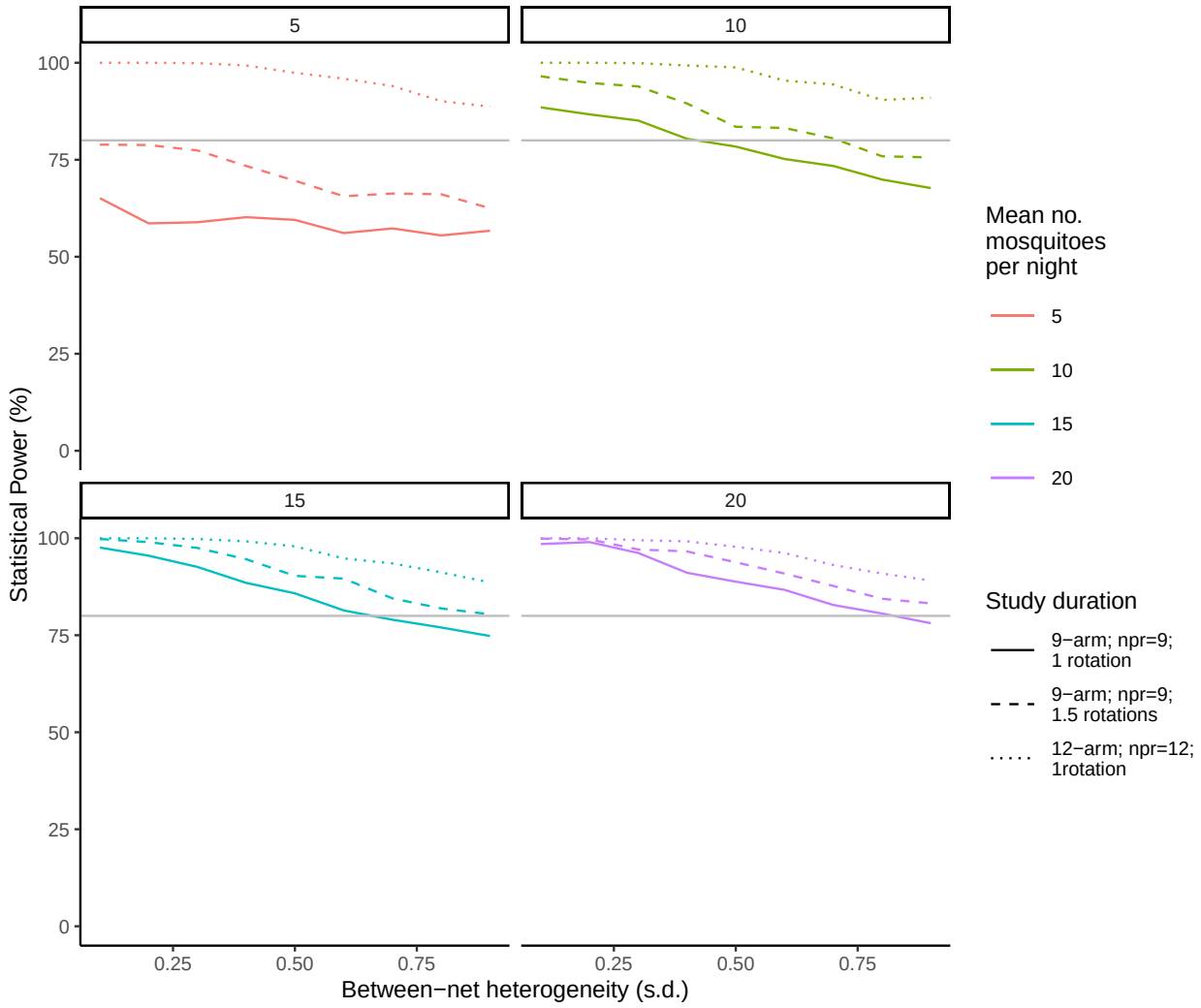

Figure 3.1: Statistical power to assess non-inferiority in an experimental hut trial with field-aged nets. Here we show how power varies with study duration and mosquito densities. For the 12-arm trial, the three arms containing field-aged nets are duplicated (see text for details). The horizontal grey line indicates 80% power.

Chapter 4

IACTs

4.1 Data analysis

Analysing data from IACTs is done in a very similar way to data from EHTs. Therefore, we shall focus mainly on where the two differ. For an IACT, trial arms remain in their designated compartments, rather than being rotated. This means we do not include a fixed effect for `compartment`. The other key difference is that the number of mosquitoes released into each compartment is controlled, rather than uncertain. Here we will consider IACTs with 18 compartments. So, for a 9-arm trial, two compartments are allocated to each arm. A typical dataset should contain the following quantities:

```
> head(data)
  compartment day sleeper net netcode total tot_dead
1           1   1       1  N1    N1_1    15        1
2           2   1       2  N1    N1_8    15        1
3           3   1       3  N2    N2_1    15       13
4           4   1       4  N2    N2_8    15       10
5           5   1       5  N3    N3_1    15        8
6           6   1       6  N3    N3_8    15        1
```

Ensuring that `compartment`, `day`, and `sleeper` are defined as factor variables, we can write the logistic regression model like this:

```
fit <- glm(
    cbind(tot_dead, total - tot_dead) ~ net + sleeper + day,
    family = binomial, data = mosdata
)
summary(fit)
```

And the desired OR can be calculated as per Section 3.1.

4.2 Estimating study power

As we saw in Chapter 3, study duration has a strong influence on statistical power. Since we do not rotate arms in the compartments in an IACT, we simply describe study duration in terms of the number of study days. As was the case for the EHTs, 30 net per arm are used. Therefore, the

duration of the study determines the number of times each individual net will appear in the dataset. To generate a synthetic dataset for an IACT, we introduce the user-defined function `IACT_sim()`. This works in a similar way to `EHT_sim()`, but we highlight the following arguments:

- `n_day`: Number of study days
- `n_mosq`: Number of mosquitoes released in each compartment. This is a constant value for an IACT
- `sigma_net`: the standard deviation of the between-net heterogeneity (default value=0.9)
- `n_comp`: Number of compartments used (here we use 18). This works well with `n_arms=9`

Selecting `verbose=T` will cause useful information to be printed to the R console, telling us the number of times each individual net is used in the study. Another user-defined function `IACT_NIM()` can be used to carry out the non-inferiority assessment. For a given trial design, we can use these two functions to estimate study power, as follows:

```
n_sim <- 1000
store_power <- rep(NA, n_sim)
for(i in 1:n_sim){
    mosdata <- IACT_sim(sigma_net = 0.9, n_day = 44, n_mosq = 20,
                        verbose = F, n_nets = 30)
    store_power[i] <- IACT_NIM(dataset = mosdata, verbose = F)
}
mean(store_power) # estimate of power
[1] 0.767
#95% confidence intervals
binom.test(table(factor(store_power,c(1,0))))$conf.int
[1] 0.739534 0.792884
attr(,"conf.level")
[1] 0.95
```

The above code calculates the power for a 44 day study, using 20 mosquitoes per compartment, and a value of 0.9 for the between-net heterogeneity. Thirty nets per arm are used: if we had selected the option `verbose=T`, we would have been informed that, for this study duration, 28 nets appear 3 times the simulated dataset, and the remaining two nets appear twice. Figure 4.1 shows how power varies with study duration, between-net heterogeneity, and mosquito numbers. Power increases quite slowly with study duration here: when using 25 mosquitoes per chamber, 44 study days should power the study, for the levels of between-net heterogeneity considered here. For 15 or 20 mosquitoes per chamber, at least 60 study days will be required.

As shown in Figure 4.1, a study of this duration is well powered, for 20 mosquitoes per compartment.

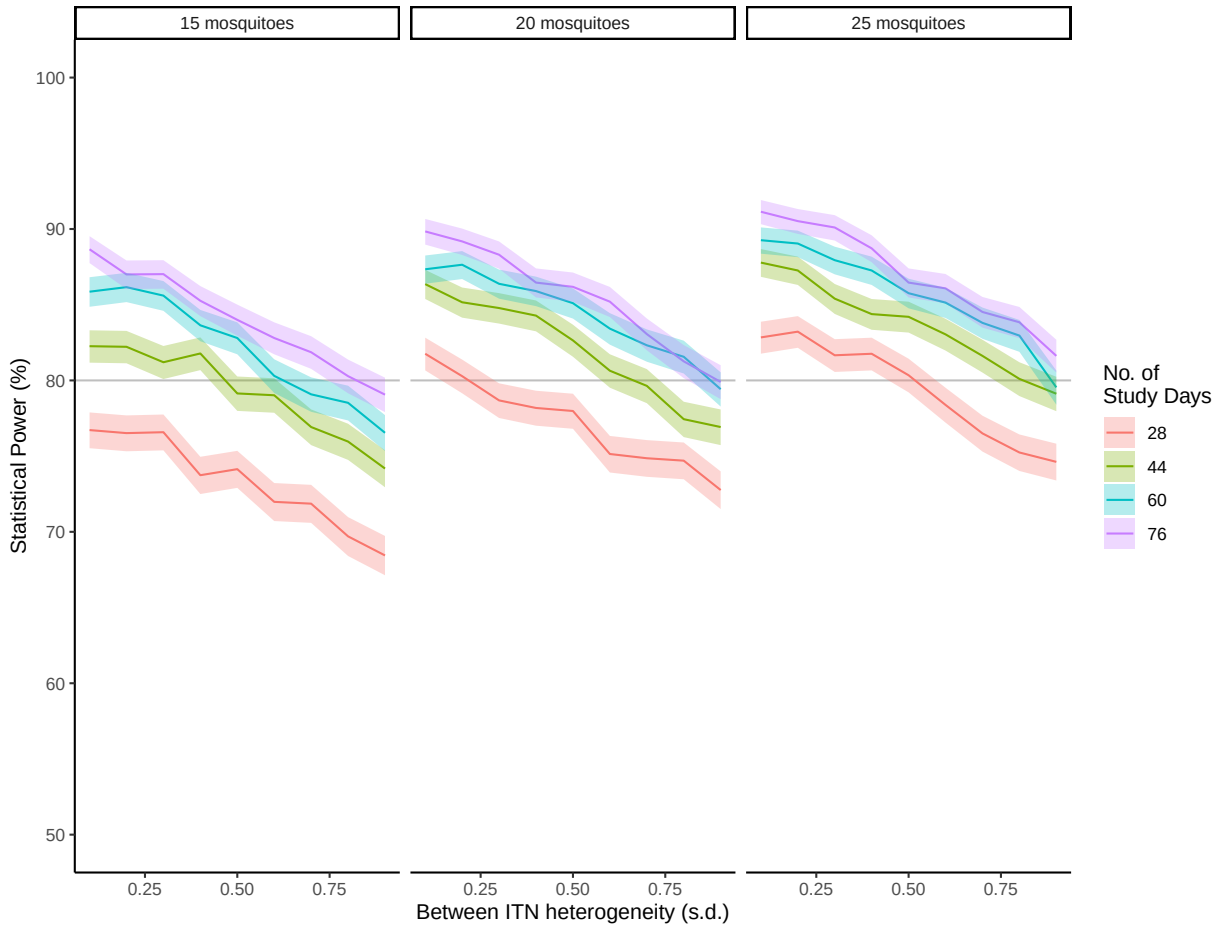

Figure 4.1: Statistical power to assess non-inferiority in an Ifakara ambient chamber test (IACT) with field-aged nets. Here we show how power varies with study duration (colours) & mosquito densities (panels). The shaded areas indicate the 95% confidence intervals. The horizontal grey line indicates 80% power. For this study, power increases quite slowly with study duration. For this reason we have used 5000 simulations for each scenario, rather than the usual 1000, to see the trend more clearly.

Chapter 5

Cones & Tunnels

5.1 Cones

For cone tests, pieces are cut from different locations on each net (variable `net_position` in the dataset). Each piece of net is assessed in multiple cone tests. A typical dataset will contain these variables:

```
    net_position replicates llin_code arm total day tot_dead
1              1          1      B_12   B     5   1        1
5              1          2      B_12   B     5   1        2
9              1          3      B_12   B     5   1        2
13             1          4      B_12   B     5   1        0
17             1          1      A_21   A     5   1        4
21             1          2      A_21   A     5   1        4
25             1          3      A_21   A     5   1        3
```

In this instance, each piece of net is tested 4 times (`replicates=4`).

When fitting a regression model to the data, we should include a fixed effect for `day`, to account for between-day variability:

```
fit <- glm(
    cbind(tot_dead, total - tot_dead) ~ arm + day,
    family = binomial, data = dataset
)
summary(fit)
```

To generate synthetic datasets for cone tests, we have written the function `cone_sim()`. Note that unlike the case for EHT, this function just generates synthetic data for the two trial arms required for the non-inferiority assessment (here we just label these arms as 'A' and 'B'). Most of the function's arguments will already be familiar to the reader, but we highlight these ones:

- `cone_mort`: Expected level of mortality in the cone test. For cone and tunnel tests, the mosquito mortality can be quite high. Therefore, we also look at the influence of mosquito mortality on study power for these tests.
- `npos`: The number of pieces to cut from each net (with each piece from a different position). We use 4 as a default here

- `rep`: The number of cone tests carried out for each net piece (the number of replicates).
- `nday`: The number of days to complete the study. Varying this number will change the number of cone tests carried out each day. As between-day variability is included in the data generating process, varying this can influence the amount of variability in the dataset.

The user-defined function `cone_NIM()` can be used to carry out the non-inferiority assessment. The impact of mosquito mortality and the number of replicates on study power is shown in Figure 5.1. When mosquito mortality is very low, three replicates will be sufficient to power the study. Four replicates should be enough for 80% power here, unless mosquito mortality is expected to be very high. The figure does also show results for 5 replicates, but we note that this is significant number of extra cone tests for a very small increase in power.

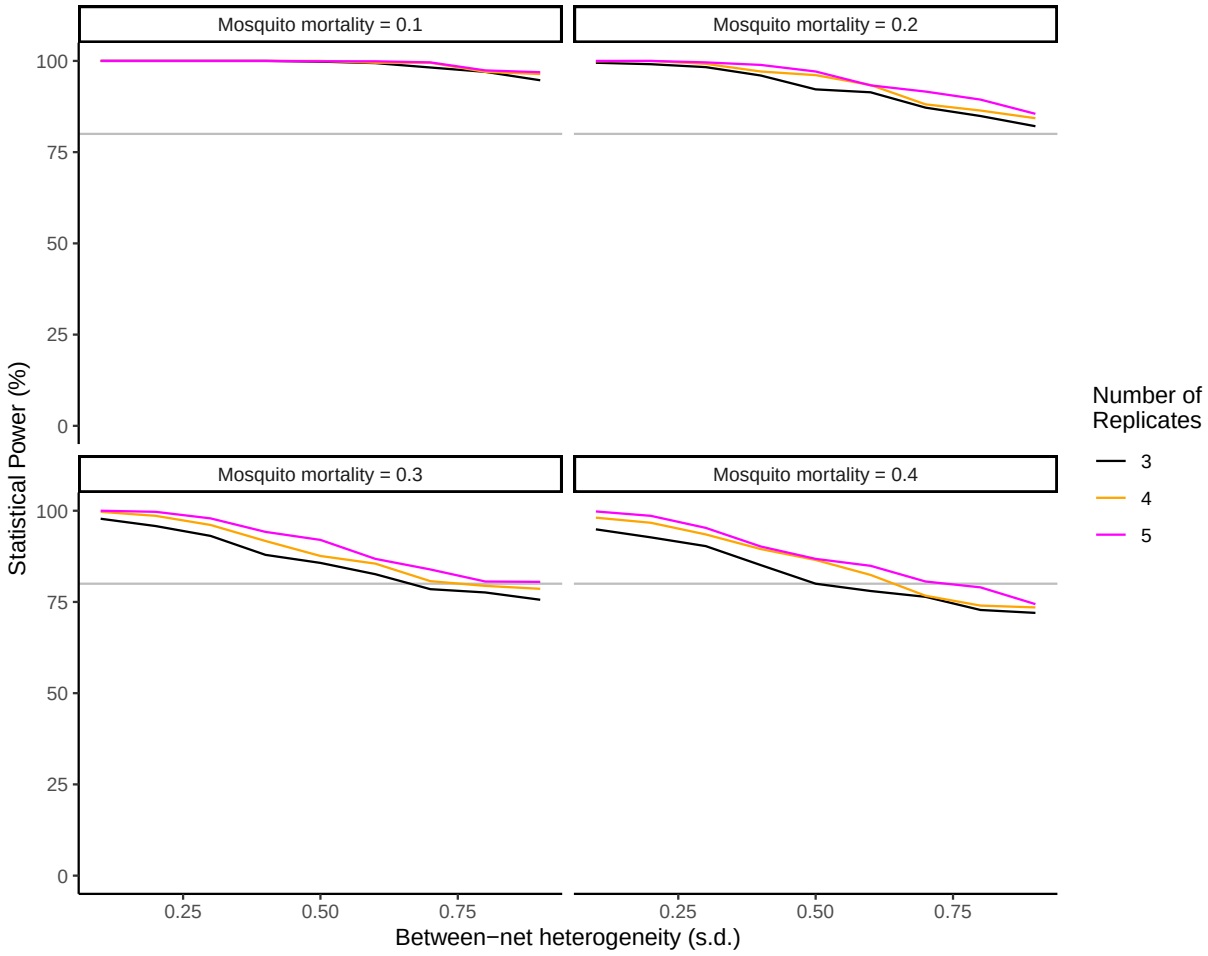

Figure 5.1: Statistical power to assess non-inferiority (comparing field-aged nets against nets washed 20 times) in a cone assay. Here we show how power varies with mosquito mortality and the number of replicates used. The horizontal grey line indicates 80% power. In these simulations, cone tests were carried out over 20 days, and four pieces were cut from each net (30 nets used per arm).

5.2 Tunnels

For the tunnel assay, as for the cone assay, we use pieces of the net, cut from different positions (here we use three pieces from each net, as recorded by the variable `net_position`). The variable `tunnel` records which tunnel is used: in this chapter we assume 5 tunnels are used. Here is an illustrative example of this type of dataset.

```
> head(tunnel_data)
    day tunnel llin_code net_position arm total tot_dead
1     1      1      B_15            1   B    48       11
37    1      2      B_25            1   B    50        2
73    1      3       B_1            2   B    51        2
109   1      4      A_14            2   A    50        3
145   1      5       A_8            3   A    49        3
2     2      1      B_17            1   B    46        5
```

Here `arm` is the net type used in each tunnel. With regard to mosqutio counts, datasets may record two values for this: the number of mosquitoes released, and the number recaptured. Mortality is assessed on the mosquitoes recaptured, which is the variable denoted `total` in the example dataset above. For the regression model, we include `day` and `tunnel` as fixed effects:

```
fit <- glm(
  cbind(tot_dead, total - tot_dead) ~ arm + day + tunnel,
  family = binomial, data = dataset
)
summary(fit)
```

As these mosquito counts are quite high, we can often power these studies using only one replicate (that is, using each net piece once). We use the function `tunnel_sim()` to generate synthetic datasets. As for cones, we here just simulate two arms (those required to make the non-inferiority assessment). In addition, the function `tunnel_NIM()` carries out the non-inferiority assessment. With these two functions, we can perform the power estimation. In the code snippet below, we carry out tunnel tests with 30 nets in each arm (cutting three net pieces from each net), assuming that in each tunnel 40 mosquitoes are successfully recaptured, and that the mean mortality measured is 0.4 (40%).

```
store_power <- rep(NA, nsim)
for(i in 1:nsim){
    tunnel_data <- tunnel_sim2(sigma_net = 0.9, n_nets = 30,
                         mort = 0.4, n_mosq = 40)
    store_power[i] <- tunnel_NIM(data = tunnel_data, verbose = F)
}
mean(store_power) # estimate of power
#95% confidence intervals for power estimation
binom.test(table(factor(store_power,c(1,0))))$conf.int
```

Figure 5.2 shows how power varies over the parameter space explored. For lower mosquito mortalities, 40 mosquitoes recaptured per compartment will be sufficient for 80% power. Fifty mosquitoes may be advisable for higher mosquito mortalities.

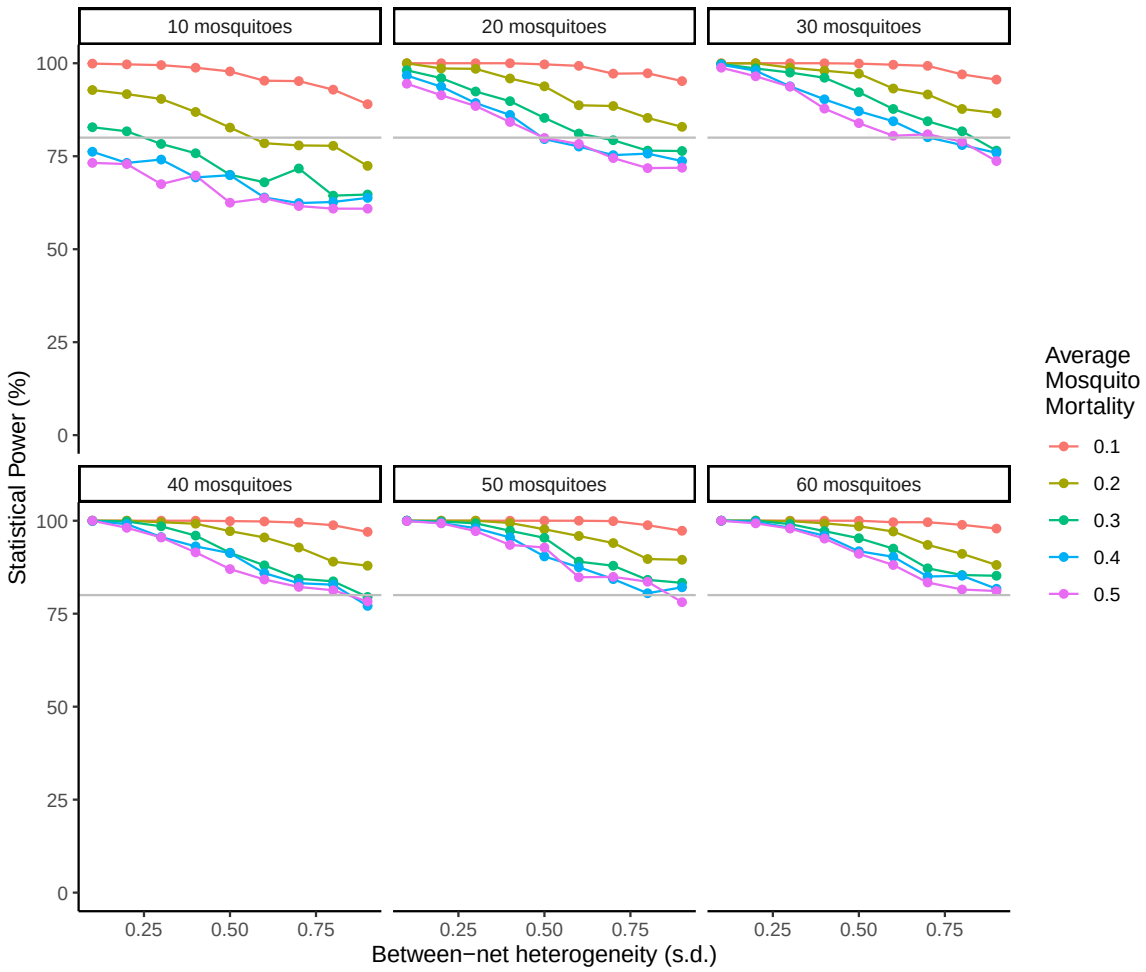

Figure 5.2: Statistical power to assess non-inferiority (comparing field-aged nets against nets washed 20 times) in a tunnel assay. Here we show how power varies with mosquito densities and mortality (colours). The horizontal grey line indicates 80% power. In each arm, 30 nets are used, with three pieces cut from each net.

Chapter 6

Median Functional Survival

The functional survival of ITNs is calculated using data from surveys on attrition and fabric integrity, carried out in the community. Functional survival is defined as 'the number of nets present and serviceable divided by the number of nets originally received and not given away or lost to follow up'. Multiple surveys should be carried out, to determine how functional survival changes over time.

The rate at which nets are given away or lost to follow up will play a role in the results obtained. We use a very simple model to estimate this quantity over time. We start with an estimate of the percentage of ITNs we expect to be given away or lost to follow up over the entire three-year period. We assume that, over this three-year period, nets are lost at a constant rate. This allows us to estimate the number of nets lost to follow up at intermediate times *i.e.* for surveys carried out at 6 months, or 1 year following the distribution of nets (Figure 6.1A).

The median net survival is the time at which the estimate of functional survival of an ITN product crosses 50%. To determine the median survival time, we first fit a S-shaped curve to the functional survival data We use a method that is well established in the literature [4, 5, 6, 7], where the curve has the following form:

$$f(k, L.t) = \exp\left(k - \frac{k}{1 - (\frac{t}{L})^2}\right). \tag{6.1}$$

Here, L and k are parameters and t is the time since the ITNs were distributed. The function $f(k, L, t)$, which is valid for $t < L$, gives the functional survival of the ITNs at time t. To aid parameter identifiability, we follow Ref. [7] and fix k to a value of 20. We use a Bayesian modelling framework to estimate the value of L, which is given a uniform prior distribution U(5.5,20.7) [7]. Below, we will write out the model in full, writing N_t to denote, at time t, the number of nets originally received and not given away or otherwise lost to follow up. Out of these N_t nets, we write S_t to denote the number of nets present and serviceable at time t. We write the full Bayesian model in the following way, where p_t represents the proportion of ITNs present and serviceable at each time point:

$$\begin{aligned} S_t &\sim \text{Binomial}(N_t, p_t) \\ p_t &= \exp\left(20 - \frac{20}{1 - (\frac{t}{L})^2}\right) \\ L &\sim \text{Uniform}(5.5, 20.7). \end{aligned} \tag{6.2}$$

Here the first line represents the binomial likelihood, the second sets $p_t = f(k = 20, L, t)$ (see Equation 6.1), and the third represents the choice of prior distribution for parameter L. To

Post-distribution time-point t (years)	Number of nets present & serviceable (S_t)	Number of nets originally received and not given away or lost to follow up (N_t)
0.5	86	96
1	66	93
2	38	86
3	22	80

Table 6.1: Illustrative example of survey data for functional survival of ITNs. In this example, we use a sample size of 100 households, assuming one net per household. These values are used to fit the curve shown in Figure 6.1B.

determine the median survival time, t_{50}, we set $f(k, L, t_{50}) = 0.5$, and rearrange to find t_{50}. In general, we find that:

$$t_{50} = L\sqrt{\frac{ln(2)}{ln(2)+k}}. \tag{6.3}$$

As demonstrated in the code that accompanies this tutorial, we use posterior samples from our fitted model to estimate t_{50}. To fit this Bayesian model, we will use the `rstan` package, which allows us to fit Stan models in R. To provide a more user-friendly interface to Stan, we will also use the `rethinking` package. The appendix to this tutorial contains some notes on installation, which can be a bit tricky for these packages.

We do not talk in terms of statistical power for this analysis, as there is no hypothesis testing to carry out. However, repeated simulation can allow us to assess the *precision* associated with measuring the MSF, for a given sample size. For a sample size of 300 households (assuming at least one ITN per household), and assuming a loss rate (over the course of 3 years) of 20%, our simulations suggest that we should be able to measure t_{50} with a precision of about 60 days, defining precision as the width of the 95% credible interval of t_{50}.

Below we show the implementation of the model outlined above, using the `ulam` function from the `rethinking package`:

```
n1 <- ulam(
  alist(
    S ~ dbinom(N, p), # model likelihood
    p <- exp(20 - 20/(1 - (t/L)*(t/L))),
    L ~ dunif(5.5,20.7) #uniform prior
  ),
  data = dxh, chains = 4, cores = 4, iter = 5000
)
precis(n1)
stancode(n1)
es1 <- extract.samples(n1)
```

The collected data, as summarised in Table 6.1, is stored in the data frame `dxh`. More experienced users of `rstan` may prefer to work with the raw stan code, which can be found in the code repository. The function `stancode()` can be used to convert models written for the `rethinking` package to stan code. Those using the simpler version of the `\rethinking` package should replace the `ulam()` function with the `quap()` function (see Appendix A and tutorial code).

As demonstrated in the code snippet above, once the model has been fitted, we can extract samples from the posterior distribution for L (we only have one parameter to fit here, so the posterior distribution is one dimensional). We can use these samples to estimate t_{50}, using the uncertainty in the fitted model to tell us about the precision of our t_{50} estimate.

```
> store <- rep(0,10000) # Using 10000 samples from the posterior distribution
> for(i in 1:10000){
+     store[i] <- es1$L[i]*sqrt(log(2)/(log(2) + 20)) # See Eq. 6.3
+ }
```

We can then estimate the mean value of t_{50}, or work with the median and quantiles:

```
> mean(store)
[1] 1.556522
> quantile(store, prob = c(0.025,0.5,0.975))
    2.5%      50%    97.5%
1.439570 1.554108 1.686406
```

Figure 6.1C shows how the precision in the estimate varies with sample size and the rate of net loss.

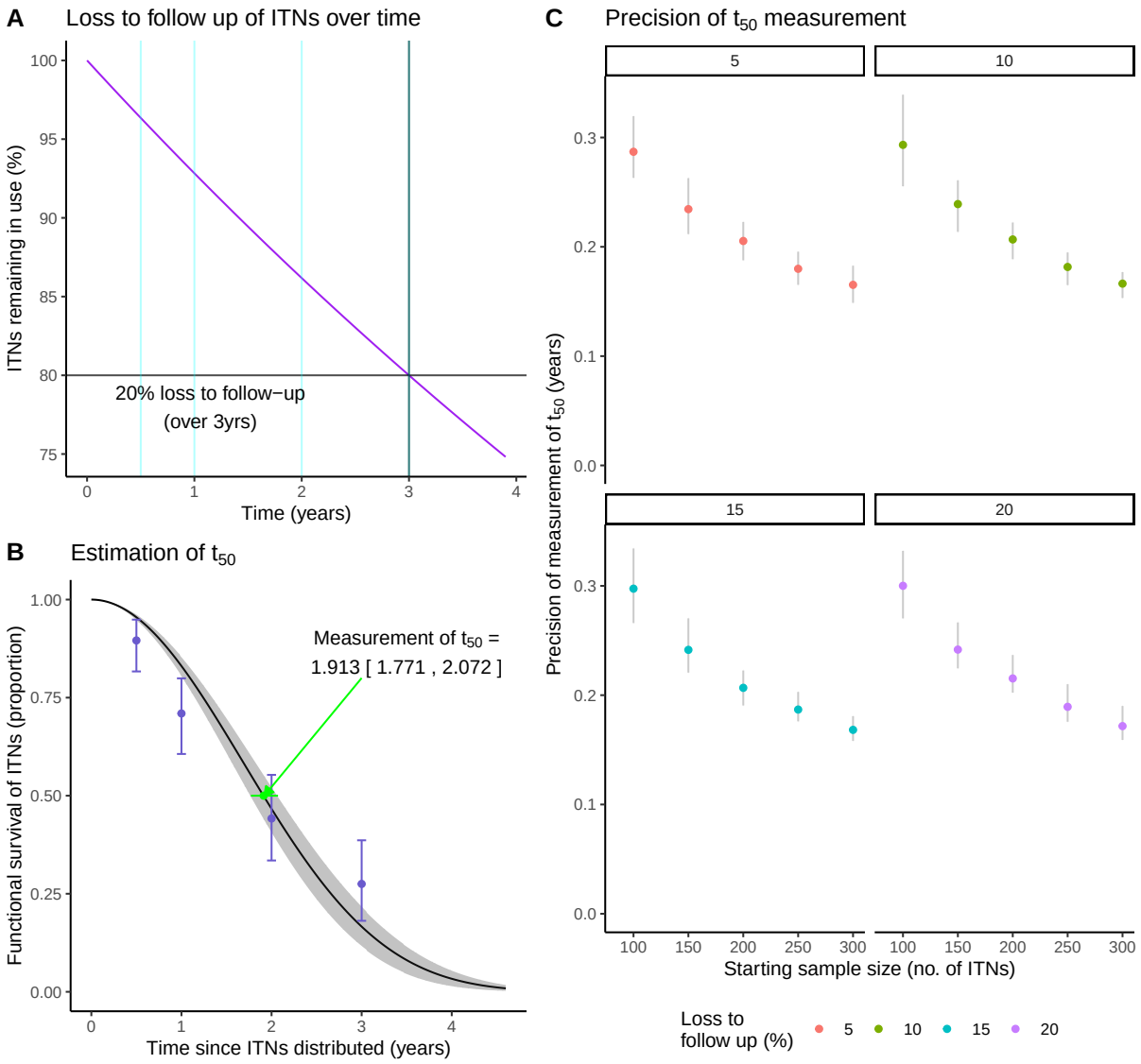

Figure 6.1: Calculating the median functional survival. Panel A: based on the anticipated percentage of nets lost over a three-year period, we can estimate the number of nets lost for earlier surveys. Panel B: fitting an s-shaped curve to the survey data. Here we fit a curve to an illustrative dataset, shown in Table 1. Samples from the posterior distribution for parameter L are used to generate the uncertainty in the curve (shaded grey area shows the 95% credible interval). The horizontal light-green line indicates the precision in the estimate of t_{50}. Panel C: how the precision of the t_{50} estimate changes with sample size and the number of nets lost to follow up.

Bibliography

[1] Joseph D. Challenger. A tutorial on non-inferiority assessments for experimental hut trials. `https://github.com/JDChallenger/WHO_NI_Tutorial`, 2023. Produced for the World Health Organisation.

[2] Joseph D. Challenger, Rebecca K. Nash, Corine Ngufor, Antoine Sanou, K. Hyacinthe Toé, Sarah Moore, Patrick K. Tungu, Mark Rowland, Geraldine M. Foster, Raphael N'Guessan, Ellie Sherrard-Smith, and Thomas S. Churcher. Assessing the variability in experimental hut trials evaluating insecticide-treated nets against malaria vectors. *Current Research in Parasitology & Vector-Borne Diseases*, 3:100115, 2023.

[3] Paul C. D. Johnson, Sarah J. E. Barry, Heather M. Ferguson, and Pie Müller. Power analysis for generalized linear mixed models in ecology and evolution. *Methods in Ecology and Evolution*, 6(2):133–142, 2015.

[4] Olivier JT Briët, Diggory Hardy, and Thomas A Smith. Importance of factors determining the effective lifetime of a mass, long-lasting, insecticidal net distribution: a sensitivity analysis. *Malaria Journal*, 11:20, 12 2012.

[5] Hannah M Koenker, Joshua O Yukich, Alex Mkindi, Renata Mandike, Nick Brown, Albert Kilian, and Christian Lengeler. Analysing and recommending options for maintaining universal coverage with long-lasting insecticidal nets: the case of Tanzania in 2011. *Malaria Journal*, 12:150, 12 2013.

[6] Samir Bhatt, Daniel J Weiss, Bonnie Mappin, Ursula Dalrymple, Ewan Cameron, Donal Bisanzio, David L Smith, Catherine L Moyes, Andrew J Tatem, Michael Lynch, Cristin A Fergus, Joshua Yukich, Adam Bennett, Thomas P Eisele, Jan Kolaczinski, Richard E Cibulskis, Simon I Hay, and Peter W Gething. Coverage and system efficiencies of insecticide-treated nets in Africa from 2000 to 2017. *eLife*, 4, 12 2015.

[7] Amelia Bertozzi-Villa, Caitlin A. Bever, Hannah Koenker, Daniel J. Weiss, Camilo Vargas-Ruiz, Anita K. Nandi, Harry S. Gibson, Joseph Harris, Katherine E. Battle, Susan F. Rumisha, Suzanne Keddie, Punam Amratia, Rohan Arambepola, Ewan Cameron, Elisabeth G. Chestnutt, Emma L. Collins, Justin Millar, Swapnil Mishra, Jennifer Rozier, Tasmin Symons, Katherine A. Twohig, T. Deirdre Hollingsworth, Peter W. Gething, and Samir Bhatt. Maps and metrics of insecticide-treated net access, use, and nets-per-capita in Africa from 2000-2020. *Nature Communications*, 12:3589, 6 2021.

Appendix A

Guidance on installing packages for Bayesian models

This appendix contains some brief comments on installing the `rstan` package, which is used in Chapter 6. As installing and using `rstan` can be quite tricky, we recommend following the simpler method, unless you have experience of using `rstan` previously. This simpler method should still estimate t_{50} to a good level of accuracy. In each case, we recommend that you are using a fairly recent version of R (e.g. 4.4).

Simpler installation

You can install a simpler version of the `rethinking` package using the following commands (if you've tried to install the full package without success, it may be worth removing the `rethinking` package from your computer before doing this, using the `remove.packages()` function):

```
install.packages(c("coda","mvtnorm","devtools","loo","dagitty"))
devtools::install_github("rmcelreath/rethinking@slim")
```

This installs a version of the package with reduced functionality (one cannot carry out Markov Chain Monte Carlo). Then we run the Bayesian model using the `quap()` function in the `rethinking` package. This approximates the posterior distribution, by assuming that the distribution is quadratic in shape, around its mode. This should estimate the posterior distribution to a good level of accuracy. The code that accompanies this tutorial shows how to compare the two results (although this comparison can only be made with the full version of the `rethinking` package).

Full installation installation

To install `rstan`, there are essentially three steps: (i) install a C++ compiler (RTools44 for Windows users, guidance for all operating systems here: https://mc-stan.org/docs/cmdstan-guide/installation.html) (ii) installing `cmdstanr` by following the instructions here: https://mc-stan.org/cmdstanr/ [Note that the first time you install cmdstanr, you will need to compile the libraries by running the command `cmdstanr::install_cmdstan()`] (iii) install the `rethinking` package by running the following commands in R:

```
install.packages(c("coda","mvtnorm","devtools","loo","dagitty","shape"))
devtools::install_github("rmcelreath/rethinking")
```

If you are using the full version of `rstan`, you can run the Bayesian model described in Chapter 6 using the `ulam()` function (see also the tutorial code).