

RHUL Psychology Statistical modelling notebook

Matteo Lisi

2022-04-25

Contents

1	About	5
2	Departmental survey about statistical methods	7
2.1	March 2022	7
3	Introduction to R	13
3.1	Installing R	13
3.2	First steps	13
3.3	Resources for learning R	21
4	Linear models	23
4.1	Simple linear regression	23
5	Meta-analyses	29
6	Missing data	31
6.1	Types of missing data	31
6.2	Deciding whether the data are MCAR	32
6.3	Causal analysis and Bayesian imputation	33

Chapter 1

About

This online book is created and maintained by Matteo Lisi and is meant to be a shared resource for staff and students at the Department of Psychology of Royal Holloway, University of London. It will contain a miscellaneous set of tutorial, examples, case studies, workshops materials and any other useful material related to data analysis and modelling. These will be added and revised over time, based on the most common questions and requests that I receive.

This is a work in progress and may contain imprecisions and typos. If you spot any please let me know at [matteo.lisi \[at\] rhul.ac.uk](mailto:matteo.lisi@rhul.ac.uk). The materials that will be included builds upon and draw from existing literature on statistics and modelling. I will endeavor to properly cite existing books and papers; but if any author feels that I have not given them fair acknowledgement, please let me know and I will make amend.

Chapter 2

Departmental survey about statistical methods

I used an anonymous survey to ask colleagues some questions about which topics may be more interesting or useful in their research.

2.1 March 2022

2.1.1 Question 1

In the first question people indicated topics of interests. The winner are multi-level models, followed closely by Bayesian statistics.

8CHAPTER 2. DEPARTMENTAL SURVEY ABOUT STATISTICAL METHODS



There were some additional suggestions.

```
#> [1] "power analyses using Shiny apps"
#> [2] "agent-based models"
#> [3] "this may be covered in the above, but approaches to analysing experience sampl."
#> [4] "Methods for longitudinal analyses"
#> [5] "Network modelling"
#> [6] "Neural networks, Markov processes"
```



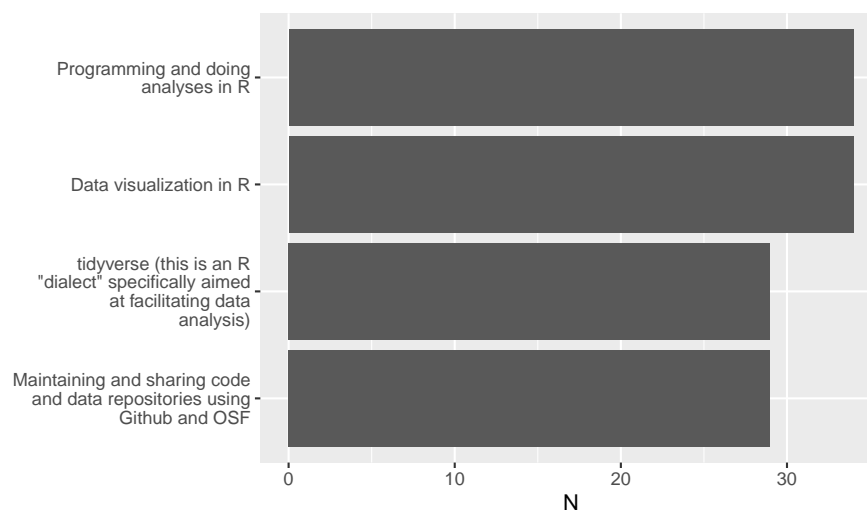
```
#> [7] "Random forests and related"  
#> [8] "causal modelling using regression models - path models etc"  
#> [9] "prediction modelling"
```

A few other topics were mentioned in the comment section:

- Shiny apps
- Network modelling
- Longitudinal analyses
- Random forests
- Neural network

2.1.2 Question 2

Here people indicated their interest for topics related to data analysis.



Other things mentioned in the comments were:

- SPM
- Docker
- Python

2.1.3 Question 4

This question was about likelihood of using different formats of support



2.1.4 Respondents' status

The final questions asked about the status / career level.



Chapter 3

Introduction to R

Most of the practical statistical tutorials and recipes in this book use the software R, so this section provides some introduction to R for the uninitiated.

3.1 Installing R

The base R system can be downloaded at the following link, which provides installers for both Windows, Mac and Linux:

<https://cran.rstudio.com/>

In addition to the base R system, it is useful to have also R-studio, which is an IDE (Integrated Development Environment) for R, and provides both an editor, a graphical interface and much more. It can be downloaded from:

<https://www.rstudio.com/products/rstudio/download/>

3.2 First steps

R is a programming language and free software environment for statistical computing and graphics. It is an *interpreted language*, which means that to give

instructions to the computer you do not have to compile it first in machine language, everything is done ‘on the fly’ through a command line interpreter, e.g. if you type `2+2` in the command line R, the computer will reply with the answer (try this on your computer):

```
2+2
#> [1] 4
```

Typically the normal workflow involve writing and saving a series of instructions in a *script* file (usually saved with the `.R` extension), which can be executed (either step by step or all at once). Since all steps of the analyses are documented in the script, this makes them transparent and reproducible.

In an R script you can use the `#` sign to add comments, so that you and others can understand what the R code is about. Commented lines are ignored by R, so they will not influence your result. See the next example:

```
# calculate 3 + 4
3 + 4
#> [1] 7
```

3.2.1 Arithmetic with R

In its most basic form, R can be used as a simple calculator. Consider the following arithmetic operators:

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Division: `/`
- Exponentiation: `^`
- Modulo: `%%`

The last two might need some explaining:

The `^` operator raises the number to its left to the power of the number to its right: for example `3^2` is 9.

The modulo returns the remainder of the division of the number to the left by the number on its right, for example 5 modulo 3 (or `5 %% 3`) is 2.

3.2.2 Variable assignment

A basic concept in programming (statistical or not) is called a *variable*.

A variable allows you to store a value (e.g. 2) or an object (e.g. a function description) in R. You can then later use this variable’s name to easily access the value or the object that is stored within this variable.

You can assign a value 2 to a variable `my_var` with the command

```
my_var <- 2
```

Note that you would have obtained the same result using:

```
2 -> my_var
```

that is, the *assignment operator* works in both directions `<-` and `->`.

The variable can then be used in any computation, for example:

```
my_var + 2  
#> [1] 4
```

3.2.3 Basic data types in R

Variables can be of many types, not just numerical values. For example, they can contain *text* values (e.g. a string of characters). Arithmetic operators such as `+` do not work with these. If you tried to apply them characters R will give you an error message.

```
# Assign a value to the variable apples  
apples <- 5  
  
# Assign a text value  
oranges <- "six"  
  
#  
apples + oranges  
#> Error in apples + oranges: non-numeric argument to binary operator
```

In fact R works with numerous data types, and some of these are not numerical (so they can't be added, subtracted, etc.). Some of the most basic types to get started are:

- Decimal values like 4.5 are called numerics.
- Natural numbers like 4 are called integers. Integers are also numerics.
- Boolean values (`TRUE` or `FALSE`, abbreviated `T` and `F`) are called logical¹.
- Text (or string) values are called characters.

3.2.4 Vectors and other data types

Additionally, the simple data types listed above can be combined in more complex 'objects' that can comprise several values. For example, we can obtain a *vector* by concatenating values using the function `c()`. This can be applied both on numerical or character data types, e.g.

¹Note that you can add or multiply logical Boolean values: internally `FALSE` are treated as zeroes, and `TRUE` as ones.

```
some_numbers <- c(4,87,10, 0.5, -6)
some_numbers
#> [1]  4.0 87.0 10.0  0.5 -6.0

my_modules <- c("PS115", "PS509", "PS300", "PS938", "PS9457")
my_modules
#> [1] "PS115" "PS509" "PS300" "PS938" "PS9457"
```

There are some special handy functions to create specific types of vectors, such as *sequences* (using the function `seq()` or the operator `:`)

```
x <- seq(from = -10, to = 10, by = 2)
x
#> [1] -10 -8 -6 -4 -2  0  2  4  6  8 10

y <- seq(-0, 1, 0.1)
y
#> [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

z <- 1:5
z
#> [1] 1 2 3 4 5
```

Another useful type of vector can be obtained by *repetition* of elements, and this can be numerical, character, or even applied to other vectors

```
rep(3, 5)
#> [1] 3 3 3 3 3

x <- 1:3
rep(x, 4)
#> [1] 1 2 3 1 2 3 1 2 3 1 2 3

rep(c("leo the cat", "daisy the dog"), 2)
#> [1] "leo the cat" "daisy the dog" "leo the cat"
#> [4] "daisy the dog"
```

We can combine vectors of different types into a *data frame*, one of the most useful ways of storing data in R. Let's say we have 3 vectors:

```
# create a numeric vector
a <- c(0, NA, 2:4) # NA means not available

# create a character vector
b <- c("PS115", "PS509", "PS300", "PS938", "PS9457")

# create a logical vector
```



```
c <- c(TRUE, FALSE, TRUE, FALSE, FALSE) # must all be caps!
```

we can combine them into a data.frame using:

```
# create a data frame with the vectors a, b, and c that we just created
my_dataframe <- data.frame(a,b,c)

# we could also change the column names (currently they are a, b, c)
colnames(my_dataframe) <- c("some_numbers", "my_modules", "logical_values")

# now let's have a look at it
my_dataframe
#>   some_numbers my_modules logical_values
#> 1           0     PS115           TRUE
#> 2          NA     PS509           FALSE
#> 3           2     PS300           TRUE
#> 4           3     PS938           FALSE
#> 5           4    PS9457           FALSE
```

Although note that in most cases we would probably import a dataframe from an external data file, for example using the functions `read.table` or `read.csv`.

3.2.5 Basic plotting in R

We can create plots using the function `plot()`. For example:

```
x = 1:10
y = 3*x - 5
plot(x, y)
```

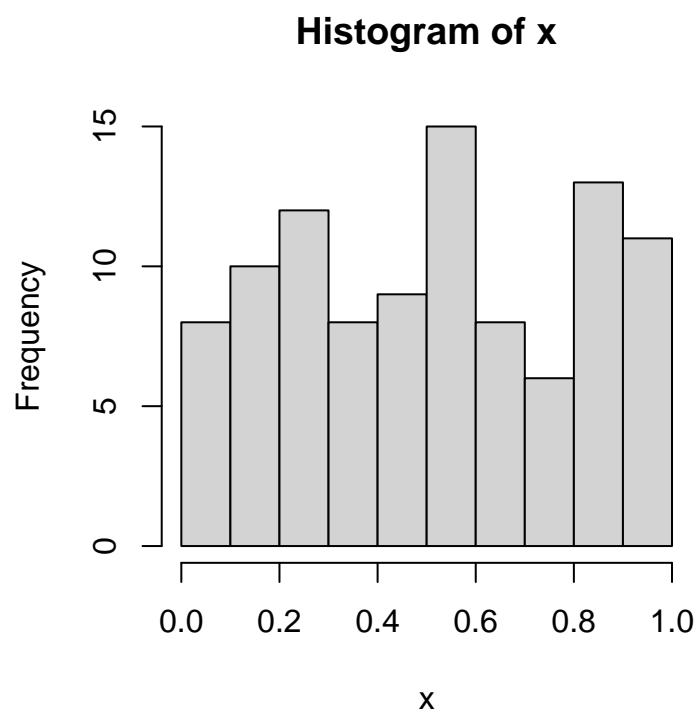


3.2.6 Other operations

3.2.6.1 Random number generation

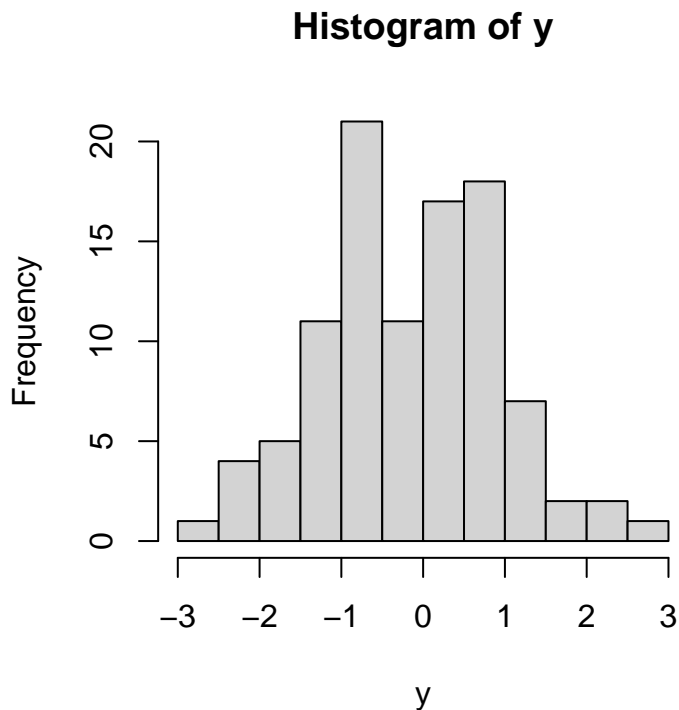
Generate uniformly distributed random numbers (function `runif()`)

```
x <- runif(100, min = 0, max = 1)
hist(x)
```



Generate numbers from a normal distribution

```
y <- rnorm(100, mean = 0, sd = 1)
hist(y)
```



3.2.7 Getting help

R has a lot of functions, and extra packages that can provides even more. It may seem a bit overwhelming, but it is very easy to get help about how to use a function: just type in a question mark, followed by the name of the function. For example, to see the help of the function we used above to generate the histogram, type

```
?hist
```

3.3 Resources for learning R

There is plenty of resources on the web to learn R. I will recommend a couple that I think are particularly well-done and useful:

- Software Carpentry tutorials on R for Reproducible Scientific Analysis
- The free book *Learning Statistics with R* by Danielle Navarro

Chapter 4

Linear models

This section will provide some worked examples of how to do analyses in R.

4.1 Simple linear regression

In this example¹ we will see how to import data into R and perform a simple linear regression analysis.

According to the standard big-bang model, the universe expands uniformly and locally, according to Hubble’s law(Hubble, 1929)

$$\text{velocity} = \beta \times \text{distance}$$

where velocity and distance are the relative velocity and distance of a galaxy, respectively; and β is “Hubble’s constant”². Note that this is a simple linear equation, in which β indicate how much the variable velocity changes for each unitary increase in the variable distance.

According to this model β^{-1} gives the approximate age of the universe, but β is unknown and must somehow be estimated from observations of velocity and distance, made for a variety of galaxies at different distances from us. Luckily we have available data from the Hubble Space Telescope. Velocities are assessed by measuring the Doppler effect red shift in the spectrum of light that we receive from the Galaxies. Distance is estimated more indirectly, by using the discovery that in certain class of stars (Cepheids), which display fluctuations in diameter and temperature over a stable period, there is a systematic relationship between the period and their luminosity.

¹Taken from Simon Wood’s book on GAM(Wood, 2017).

²Note the Hubble “constant” is a constant only in space, not in time

We can load a dataset of measurements from the Hubble Space Telescope in R using the following code

```
d <- read.table(file="https://raw.githubusercontent.com/mattelsi/RHUL-stats/main/data.csv",
                header=T)
```

`read.table` is a generic function to import dataset in text files (e.g. .csv files) into R. We use the argument `header=T` to specify that the first line of the dataset gives the names of the columns. Note that the argument `file` here is a URL, but it could be also a path to a file in our local folder. To see the help of this function, and what other arguments and features are available type `?read.table` in the R command line.

We can use the command `str()` to examine what we have imported

```
str(d)
#> 'data.frame':   24 obs. of  3 variables:
#> $ Galaxy : chr  "NGC0300" "NGC0925" "NGC1326A" "NGC1365" ...
#> $ velocity: int  133 664 1794 1594 1473 278 714 882 80 772 ...
#> $ distance: num  2 9.16 16.14 17.95 21.88 ...
```

This tells us that our data frame has 3 variables:

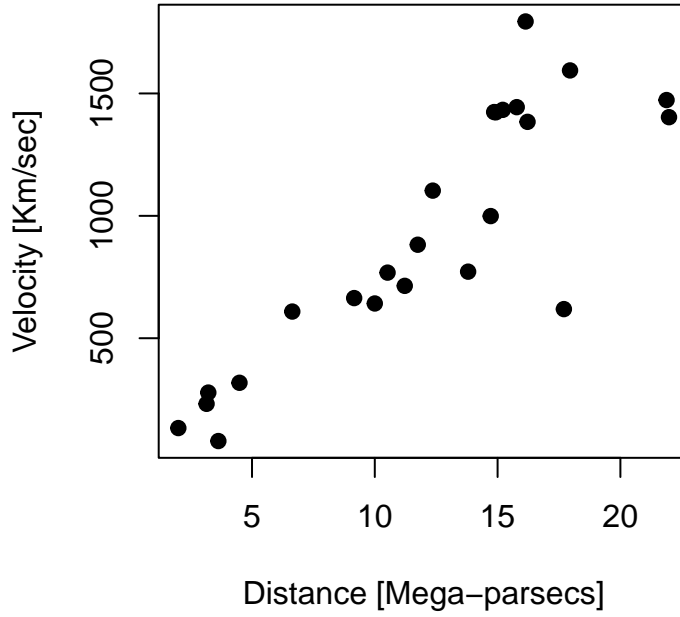
- `Galaxy`, the ‘names’ of the galaxies in the dataset
- `velocity`, their relative velocity in Km/sec
- `distance`, their distance expressed in Mega-parsecs³

We can plot⁴ them using the following code:

```
plot(d$distance, # indicate which variable on X axis
     d$velocity, # indicate which variable on Y axis
     xlab="Distance [Mega-parsecs]",
     ylab="Velocity [Km/sec]",
     pch=19) # set the type of point
```

³1Mega-parsec = 3.09×10^{19} Km

⁴See `?plot` for more info about how to customize plots in R.



It is clear, from the figure, that the observed data do not follow Hubble's law exactly, but given the how these measurements were obtained (there is uncertainty about the true values of the distance and velocities) it would be surprising if they did. Given the apparent variability, what can be inferred from these data? In particular:

1. what value of β is most consistent with the data?
2. what range of β values is consistent with the data?

In order to make inferences we make some assumptions about the nature of the measurement noise. Specifically, we assume that measurements errors are well-characterized by a Gaussian distribution. This results in the following model:

$$y = \beta x + \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

which is essentially a linear regression but without the intercept: that is, whereas normally a linear regression model includes an additive term that is not multiplied with the predictor (as in $y = \beta_0 + \beta_1 x + \epsilon$), which gives the expected value of the dependent variable when all predictors are set to zero, in this case the theory tells us we can assume the intercept (the term β_0) is zero and we can

ignore it.

We can fit the model with the function `lm` in R. Note that to tell R that I don't want to fit the intercept, I include in the formula the term `0 +` - this essentially tells R that the intercept term is set to zero⁵

```
hub.m <- lm(velocity ~ 0 + distance, d)
summary(hub.m)
#>
#> Call:
#> lm(formula = velocity ~ 0 + distance, data = d)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -736.5 -132.5  -19.0   172.2   558.0
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> distance    76.581      3.965    19.32 1.03e-15 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 258.9 on 23 degrees of freedom
#> Multiple R-squared:  0.9419, Adjusted R-squared:  0.9394
#> F-statistic: 373.1 on 1 and 23 DF,  p-value: 1.032e-15
```

So, based on this data, **our estimate of the Hubble constant is 76.58 with a standard error of 3.96**. The standard error - which is the standard deviation of the sampling distribution of our estimates - gives an idea of the range of values that is compatible with our data and could be used to compute a confidence interval (roughly, we would expect that the 'true' values of the parameters lies in the interval defined by ± 2 standard errors 95% of the times).

So, how old is the universe?

The Hubble constant estimate has units of $\frac{\text{Km/sec}}{\text{Mega-parsecs}}$. A Mega-parsecs is 3.09×10^{19} Km, so we divide our estimate of $\hat{\beta}$ by this amount. The reciprocal of $\hat{\beta}$ then gives the approximate age of the universe (in seconds). In R we can calculate it (in years) as follow

```
# transform in Km
hubble.const <- coef(hub.m)/(3.09 * 10^(19))

# invert to get age in seconds
age <- 1/hubble.const
```

⁵A similar results would have been obtained using the notation `velocity ~ -1 + distance`.

```
# use unname() to avoid carrying over  
# the label "distance" from the model  
age <- unname(age)  
  
# transform age in years  
age <- age/(602 * 24 * 365)  
  
# age in billion years  
age/109  
#> [1] 12.79469
```

giving an estimate of about 13 billion years.

Chapter 5

Meta-analyses

For running meta-analyses, we recommend the `metafor` package for R (see [link 1](#), [link 2](#)).

A comprehensive, hands-on guide on how to use this package is provided in the book by Harrer and colleagues (Harrer et al., 2021), freely available at [this link](#).

An alternative to the `metafor` package is to Bayesian multilevel modelling (also discussed in the book by Harrer and colleagues). A more technical discussion of Bayesian multilevel modelling for meta-analyses is provided in this paper by Williams, Rast and Bürkner (Williams et al., 2018).

Chapter 6

Missing data

6.1 Types of missing data

Following the work of Rubin (RUBIN, 1976), missing data are typically grouped in 3 categories:

- Missing completely at random (**MCAR**). This assumes that the probability of being missing is the same for all cases; this implies that the mechanisms that causes missingness is not related in any way to the data. For example, say, there's a known unpredictable error on the server side that prevented recording some responses for some respondents to a survey. As the missingness is entirely independent on the respondents' characteristics, this would be MCAR. When the data are MCAR we can ignore a lot of the complexities and just do a *complete-case* analysis (that is, simply exclude incomplete observations from the dataset). A part from possible loss of information, doing a complete case analysis should not introduce bias in the results. In practice, however, it is difficult to establish whether the data are truly MCAR. Ideally, to argue that data are MCAR, one should have a good idea of the mechanisms that caused missingness (more on this below). Formally, data is MCAR if

$$\Pr(R = 0 | Y_{\text{obs}}, Y_{\text{mis}}, \psi) = \Pr(R = 0 | \psi)$$

where R is an indicator variable that is set to 0 for missing data and 1 otherwise; $Y_{\text{obs}}, Y_{\text{mis}}$ indicate observed and missing data, respectively; and ψ is simply a parameter that determine the overall (fixed) probability of being missing.

- Missing at random (**MAR**). A less strong assumption about missingness is that it is systematically related to the observed but not the unobserved data. For example, data are MAR if in a study male respondents are less likely to complete a survey on depression severity than female respondents

- that is, the probability of reaching the end of the survey is related to their sex (fully observed) but not the severity of their symptoms. Formally, data is MAR if

$$\Pr(R = 0|Y_{\text{obs}}, Y_{\text{mis}}, \psi) = \Pr(R = 0|Y_{\text{obs}}, \psi)$$

When data are missing at random (MAR) the results of complete case analyses may be biased and a common approach to deal with this is to use imputation. Stef van Buuren has a freely available online book on this topic (van Buuren, 2018). Among other things, it illustrates how to do multiple imputation in R with examples.

- Missing not at random (**MNAR**). This means that the probability of being missing varies for reasons that are unknown to us, and may depend on the missing values themselves. Formally this means that $\Pr(R = 0|Y_{\text{obs}}, Y_{\text{mis}}, \psi)$ does not simplify in any way. This case is the most hard to handle: a complete case analyses may or may not be biased, but there is no way of knowing it and we may have to find more information about what caused missingness.

6.2 Deciding whether the data are MCAR

As MCAR is the only scenario in which it is safe to do a complete case analysis, it would seem useful to have way to test this assumption. Some approaches have been proposed to test whether the data are MCAR, but they are not widely used and it's not clear how useful they are in practice. For example one could run a logistic regression with “missingness” as dependent variable (e.g. an indicator variable set to 1 if data is missing and 0 otherwise), and all other variables as predictors - if the data are MCAR then none of the predictors should predict missingness. A popular alternative, implemented in several software packages is Little's test (Little, 1988).

Technically, these approaches can help determine whether the missingness depends on some observed variables (that is, if they are MAR), but strictly speaking cannot exclude missingness due to unobserved variables (MNAR scenario). Nevertheless, if one has good reasons to believe that the data are MCAR, and want to add some statistical test that corroborate this assumption, these could be reasonable tests to do. However, it remains important to also discuss openly possible reasons and mechanisms of missingness, and explain why we deem it a priori plausible that the data are MCAR. In fact, *statistical tests alone cannot tell whether data are missing completely at random*. The terms MCAR, MAR and MNAR refers to the *causal* mechanisms that is responsible for missing data and, strictly speaking, causal claims cannot be decided uniquely on the basis of a simple statistical test. If the data “pass” the test it would provide some additional support to the assumption that they are MCAR, but in and of itself the test alone does not fully satisfy the assumptions of MCAR. To see why note that MCAR (as formally defined above) assumes also that there should be no relationship between the missingness on a particular variable and the values of

that same variable: but since this is a question about what is missing from the data, it cannot be tested with any quantitative analysis of the available data. Finally, it should be added that as these are null-hypothesis significance test, a failure to reject the null hypothesis does not, in and of itself, provide evidence for the null hypothesis (that the data are MCAR). It may be also that we don't have enough power to reliably detect the pattern in the missingness.

6.3 Causal analysis and Bayesian imputation

The best and most principled approach to deal with missingness (at least in my opinion) is to think hard about the causal mechanisms that may determine missingness, and use our assumption about the causal mechanisms to perform a full Bayesian imputation (that is , treating the missing data as parameter and estimating them).

I plan to create and include here a worked example of how to do this; in the meantime interested readers are referred to Chapter 15 (in particular section 15.2) of the excellent book by Richard McElreath *Statistical Rethinking*(McElreath, 2020) which present a very accessible worked example of how to do this in R.

Bibliography

- Harrer, M., Cuijpers, P., A, F. T., and Ebert, D. D. (2021). *Doing Meta-Analysis With R: A Hands-On Guide*. Chapman & Hall/CRC Press, Boca Raton, FL and London, 1st edition.
- Hubble, E. (1929). A relation between distance and radial velocity among extragalactic nebulae. *Proceedings of the National Academy of Sciences*, 15(3):168–173.
- Little, R. J. A. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association*, 83(404):1198–1202.
- McElreath, R. (2020). *Statistical Rethinking, A Course in R and Stan*. Chapman & Hall/CRC Press, 2nd edition.
- RUBIN, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.
- van Buuren, S. (2018). *Flexible imputation of missing data*. Chapman & Hall/CRC Press, 2nd edition.
- Williams, D. R., Rast, P., and Bürkner, P. C. (2018). Bayesian meta-analysis with weakly informative prior distributions.
- Wood, S. (2017). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC Press, 2nd edition.