HDFS 523: Strategies for Data Analysis in Developmental Research

Zachary F. Fisher

2024-01-21

# 1 About This Book

This book provides the course notes for HDFS 523. It is currently under development, so any feedback is appreciated (e.g., during class, via email, or the edit link in the header). This first chapter is just about how to use the book – the course content starts in Chapter 2.

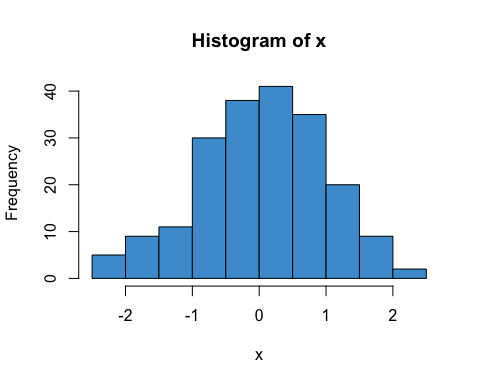
## 1.1 Why this book?

There are a few goals of moving from “textbook + slides + exercises” to an ebook. For now, the main goal is to update and integrate code contents from the course into one consistent format, rather than having multiple files to sort through on Canvas.

## 1.2 Code Folding

The book combines lecture slides and R coding examples. It is often convenient to hide code when introducing new material. This is accomplished using code folding. An example of code folding is given on this page. Below, a histogram integrated into the text. By clicking on the button called “Show Code” on the top of the page, the R code that produced the histogram will also be visible. Notice that you may need to scroll horizontally to see all of the text in the code window. Also notice that when you hover your mouse over the code window, an icon appears in the top right corner – this lets you copy the block of code with one click.

# Here is some R code. You don't have to look at it when reading the book, but it is here when you need it  
x <- rnorm(200)  
hist(x, col = "#4B9CD3")



## 1.3 Acknowledgements

Many people have contributed to the course materials for HDFS 523. Most importantly, many of the original R markdown files for the course were developed by Nilam Ram and Zita Oravecz.

# 2 Data Cleaning

In Chapter 2 we will work through some basic data cleaning operations useful in longitudinal data analysis. The basic idea is provide a set of scripts to use for exploring new repeated measures data sets.

## 2.1 Example Data

For Chapter 2 we will make use of the longitudinal Wechsler Intelligence Scale for Children [WISC; Wechsler, -Wechsler ([1949](#ref-wechsler1949))] dataset described by Osborne and Suddick ([1972](#ref-osborne1972)). These data have been detailed extensively in a number of papers ([McArdle and Epstein 1987](#ref-mcardle1987); [McArdle 1988](#ref-mcardle1988); [Mcardle and Aber 1990](#ref-mcardle1990); [McArdle and Nesselroade 1994](#ref-mcardle1994)) and are used here with with permission.

The WISC data contains repeated measures data from 204 children between the ages of 6 and 11 years old (during grades 6, 7, 9 and 11). Thee repeated measures include component scores for the verbal tests and performance subtests at all four occasions, along with verbal subtest scores for the information, comprehension, similarities, and vocabulary domains at the first and last measurement occasion. The demographics variables mother’s education (continuous in years) and mother graduated high school (dichotomous) are also included.

## 2.2 Reading in Repeated Measures Data

We can read in the WISC data directly from the [QuantDev](https://quantdev.ssri.psu.edu/) website.

filepath <- "https://quantdev.ssri.psu.edu/sites/qdev/files/wisc3raw.csv"  
wisc3raw <- read.csv(file=url(filepath), header=TRUE)

Additional details on importing different data types into R can be found here: <http://www.statmethods.net/input/importingdata.html>.

## 2.3 Familiarize Yourself with the Data

Let’s take an initial look at the structure of our data object using str()

str(wisc3raw)

## 'data.frame': 204 obs. of 20 variables:  
## $ id : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ verb1 : num 24.4 12.4 32.4 22.7 28.2 ...  
## $ verb2 : num 27 14.4 33.5 28.4 37.8 ...  
## $ verb4 : num 39.6 21.9 34.3 42.2 41.1 ...  
## $ verb6 : num 55.6 37.8 50.2 44.7 71 ...  
## $ perfo1 : num 19.8 5.9 27.6 33.2 27.6 ...  
## $ perfo2 : num 23 13.4 45 29.7 44.4 ...  
## $ perfo4 : num 43.9 18.3 47 46 65.5 ...  
## $ perfo6 : num 44.2 40.4 77.7 61.7 64.2 ...  
## $ info1 : num 31.3 13.8 35 24.8 25.3 ...  
## $ comp1 : num 25.6 14.8 34.7 31.4 30.3 ...  
## $ simu1 : num 22.93 7.58 28.05 8.21 15.98 ...  
## $ voca1 : num 22.2 15.4 26.8 20.2 35.4 ...  
## $ info6 : num 69.9 41.9 60.4 52.9 67.4 ...  
## $ comp6 : num 44.4 44.9 50.3 42.7 86.7 ...  
## $ simu6 : num 68 33.9 35.8 45.8 72.4 ...  
## $ voca6 : num 51.2 37.7 55.5 36 60.4 ...  
## $ momed : num 9.5 5.5 14 14 11.5 14 9.5 5.5 9.5 11.5 ...  
## $ grad : int 0 0 1 1 0 1 0 0 0 0 ...  
## $ constant: int 1 1 1 1 1 1 1 1 1 1 ...

From the output, we can also see that the data frame consists of 204 observations (rows) and 20 variables (columns). Each variable’s name and data type is also listed. Methods like the ones above can be an effective way to initially familiarize yourself with the main features of a dataset.

## 2.4 Look for Duplicated IDs

It is always worth looking for non-unique ID numbers when ID labels are included in a dataset. Here we have an id variable indicating the subject number. Since our data is in a long format (more on that later) duplicate IDs may indicate a potential problem with the data source or clues on how the data is structured.

any(duplicated(wisc3raw$id))

## [1] FALSE

## 2.5 Using table() to Spot Irregularities

When a variable takes on a limited range of values it is often useful to screen for irregularities or invalid values. This is common across all variable types and can occur for character strings, numeric, integer and factor types. For example, we would expect the grad variable to only take the values of zero or one. We can use the table() function to quickly confirm this.

By default table() simply omits any values coded as NA. To include a count of the NA values use the useNA argument of table() as follows:

table(wisc3raw$grad, useNA = "always")

##   
## 0 1 <NA>   
## 158 46 0

## 2.6 Missing Data

Dealing with missing data in a consistent manner is one of the most important aspects of data cleaning. When data are imported into R it is common to discover missing values are coded according to a variety of conventions.

Often a first step in handling missing data involves recoding missing values as NA. Writing bespoke code to handle the different types of missing data one might encounter is tedious and unnecessary.

naniar ([Tierney et al. 2021](#ref-naniar)) is a useful package with many convenience functions for managing missing data in R. Here we demonstrate some of this functionality.

### 2.6.1 Generating Example Data

Since the WISC data does not contain missing values it is helpful to generate a synthetic dataset containing some commonly encountered missing data codes.

set.seed(123)  
wisc\_miss <- wisc3raw  
wisc\_miss$verb1[sample(nrow(wisc\_miss),100)] <- -99  
wisc\_miss$comp1[sample(nrow(wisc\_miss),75)] <- "N/A"  
wisc\_miss$info1[sample(nrow(wisc\_miss),50)] <- "NA"

### 2.6.2 Recoding Values with NA

Now that we have a dataset with missing values we can use naniar to recode these values to NA.

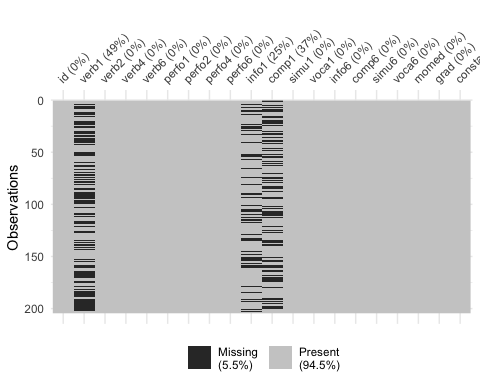
na\_strings <- c("NA", "N/A", -99)  
   
wisc\_miss <- naniar::replace\_with\_na\_all(  
 wisc\_miss, condition = ~.x %in% na\_strings  
)

See the [naniar vignette on recoding NA values](https://cran.r-project.org/web/packages/naniar/vignettes/replace-with-na.html) for more detailed information on the package functionality.

### 2.6.3 Missing Data Visualization

Once we have recoded our data in a consistent manner we can use visualizations to explore the missing data. The vis\_miss() function from naniar is a good starting point for visualizing the amount of missing data in our dataset. The plots shows the missing values in black and non-missing values in gray. In addition, percentages of missing data in both the dataset and individual variables are provided.

naniar::vis\_miss(wisc\_miss)



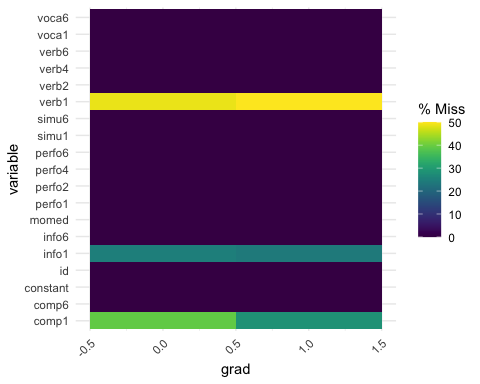
It is often useful to look at combinations of missingness among different variables.

naniar::gg\_miss\_upset(wisc\_miss)



We can also look at the percentage of missing data across a factor variable.

naniar::gg\_miss\_fct(x = wisc\_miss, fct = grad)



Many missing data visualizations are described in the [naniar vignette on missing data visualization](https://cran.r-project.org/web/packages/naniar/vignettes/naniar-visualisation.html) including plots for exploring missing data mechanisms.

## 2.7 Exporting Data

Depending on work-flow, you may need to export your dataset for use in another statistical software program. The write.csv() function is a convenient method for outputting comma delimited files.

write.csv(wisc3raw, file = "wisc3raw.csv", row.names = FALSE, na = "-99")

Note that by default the write.csv() function will include an extra column of row numbers and will notate missing data with an NA. More information on exporting data is available at <http://www.statmethods.net/input/exportingdata.html>.

## 2.8 Reshaping Repeated Measures Data

Behavioral science tends to use relational data structures - in basic form, spreadsheets. Typically, the data are stored in a data frame (a “fancy” matrix) with multiple rows and columns. Two common schemata used to accommodate repeated measures data are *wide format* and *long format*. Different analysis and plotting functions require different kinds of data input. Thus, it is imperative that one can convert the data back and forth between wide and long formats.

There are lots of ways to do this. We illustrate one way.

*Sidebar*: The dput() function provides a convenient method to get the variable names (or any R object) into a format that can be read back into R. For example, this can be helpful when working with a long vector of strings.

dput(colnames(wisc3raw))

## c("id", "verb1", "verb2", "verb4", "verb6", "perfo1", "perfo2",   
## "perfo4", "perfo6", "info1", "comp1", "simu1", "voca1", "info6",   
## "comp6", "simu6", "voca6", "momed", "grad", "constant")

First, let’s subset our data to only include the variables we need for this analysis.

var\_names\_sub <- c(  
 "id", "verb1", "verb2", "verb4", "verb6",  
 "perfo1", "perfo2", "perfo4", "perfo6",  
 "momed", "grad"  
)  
  
wiscraw <- wisc3raw[,var\_names\_sub]  
head(wiscraw)

## id verb1 verb2 verb4 verb6 perfo1 perfo2 perfo4 perfo6 momed grad  
## 1 1 24.42 26.98 39.61 55.64 19.84 22.97 43.90 44.19 9.5 0  
## 2 2 12.44 14.38 21.92 37.81 5.90 13.44 18.29 40.38 5.5 0  
## 3 3 32.43 33.51 34.30 50.18 27.64 45.02 46.99 77.72 14.0 1  
## 4 4 22.69 28.39 42.16 44.72 33.16 29.68 45.97 61.66 14.0 1  
## 5 5 28.23 37.81 41.06 70.95 27.64 44.42 65.48 64.22 11.5 0  
## 6 6 16.06 20.12 38.02 39.94 8.45 15.78 26.99 39.08 14.0 1

### 2.8.1 Reshape Wide to Long

One way to go *from wide to long* is using the reshape() function from base R.

Notice, the varying argument contains the repeated measures columns we want to stack and the timevar is a new variable containing the grade level information previosuly appended at the end of the colnames listed in varying.

# reshape data from wide to long  
wisclong <- reshape(  
 data = wiscraw,  
 varying = c("verb1", "verb2", "verb4","verb6", "perfo1","perfo2","perfo4","perfo6"),  
 timevar = c("grade"),   
 idvar = c("id"),  
 direction = "long",   
 sep = ""  
)  
  
# reorder by id and day   
wisclong <- wisclong[ order(wisclong$id, wisclong$grade), ]  
  
head(wisclong, 8)

## id momed grad grade verb perfo  
## 1.1 1 9.5 0 1 24.42 19.84  
## 1.2 1 9.5 0 2 26.98 22.97  
## 1.4 1 9.5 0 4 39.61 43.90  
## 1.6 1 9.5 0 6 55.64 44.19  
## 2.1 2 5.5 0 1 12.44 5.90  
## 2.2 2 5.5 0 2 14.38 13.44  
## 2.4 2 5.5 0 4 21.92 18.29  
## 2.6 2 5.5 0 6 37.81 40.38

Again, notice how reshape automatically split verb1, verb2, etc. into a string name and a grade variable.

### 2.8.2 Reshape Long to Wide

Now we go *from long to wide*, again using the reshape() function. The v.names argument specifies the variables to be expanded column wise based on the repeated measure specified in timevar.

#reshaping long to wide  
wiscwide <- reshape(  
 data = wisclong,   
 timevar = c("grade"),   
 idvar = c("id"),  
 v.names = c("verb","perfo"),  
 direction = "wide",   
 sep = ""  
)  
  
# reordering columns   
wiscwide <- wiscwide[, c(  
 "id", "verb1", "verb2", "verb4", "verb6",  
 "perfo1", "perfo2", "perfo4", "perfo6",  
 "momed","grad"   
)]  
  
head(wiscwide)

## id verb1 verb2 verb4 verb6 perfo1 perfo2 perfo4 perfo6 momed grad  
## 1.1 1 24.42 26.98 39.61 55.64 19.84 22.97 43.90 44.19 9.5 0  
## 2.1 2 12.44 14.38 21.92 37.81 5.90 13.44 18.29 40.38 5.5 0  
## 3.1 3 32.43 33.51 34.30 50.18 27.64 45.02 46.99 77.72 14.0 1  
## 4.1 4 22.69 28.39 42.16 44.72 33.16 29.68 45.97 61.66 14.0 1  
## 5.1 5 28.23 37.81 41.06 70.95 27.64 44.42 65.48 64.22 11.5 0  
## 6.1 6 16.06 20.12 38.02 39.94 8.45 15.78 26.99 39.08 14.0 1

Using functions included in base R can be useful in a number of situations. One example is package development where one may wants to limit dependencies.

That said, many people find reshape to be unnecessarily complicated. A similar, and potentially more convenient, set of functions have been developed for reshaping data in the tidyr ([Wickham 2021](#ref-tidyr)) package. For those interested take a look at the pivot\_longer() and pivot\_wider() functions.

For examples using tidyr to reshape data see the [tidyr vignette on pivoting](https://cran.r-project.org/web/packages/tidyr/vignettes/pivot.html).

McArdle, J. J. 1988. “Dynamic but Structural Equation Modeling of Repeated Measures Data.” In *Handbook of Multivariate Experimental Psychology, 2nd Ed*, 561–614. Perspectives on Individual Differences. New York, NY, US: Plenum Press. <https://doi.org/10.1007/978-1-4613-0893-5_17>.

Mcardle, J. J., and MARK S. Aber. 1990. “Chapter 5 - Patterns of Change Within Latent Variable Structural Equation Models.” In *Statistical Methods in Longitudinal Research*, edited by Alexander von Eye, 1:151–224. Statistical Modeling and Decision Science. San Diego: Academic Press. <https://doi.org/10.1016/B978-0-12-724960-5.50010-X>.

McArdle, J. J., and David Epstein. 1987. “Latent Growth Curves Within Developmental Structural Equation Models.” *Child Development* 58 (1): 110–33. <https://doi.org/10.2307/1130295>.

McArdle, J. J., and John R. Nesselroade. 1994. “Using Multivariate Data to Structure Developmental Change.” In *Life-Span Developmental Psychology: Methodological Contributions*, 223–67. The West Virginia University Conferences on Life-Span Developmental Psychology. Hillsdale, NJ, US: Lawrence Erlbaum Associates, Inc.

Osborne, R. T., and D. E. Suddick. 1972. “A Longitudinal Investigation of the Intellectual Differentiation Hypothesis.” *The Journal of Genetic Psychology: Research and Theory on Human Development* 121 (1): 83–89. <https://doi.org/10.1080/00221325.1972.10533131>.

Tierney, Nicholas, Di Cook, Miles McBain, and Colin Fay. 2021. *Naniar: Data Structures, Summaries, and Visualisations for Missing Data*. <https://CRAN.R-project.org/package=naniar>.

Wechsler, David. 1949. *Wechsler Intelligence Scale for Children*. Wechsler Intelligence Scale for Children. San Antonio, TX, US: Psychological Corporation.

Wickham, Hadley. 2021. *Tidyr: Tidy Messy Data*. <https://CRAN.R-project.org/package=tidyr>.