HDFS 523: Strategies for Data Analysis in Developmental Research

Zachary F. Fisher

2023-01-30

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.

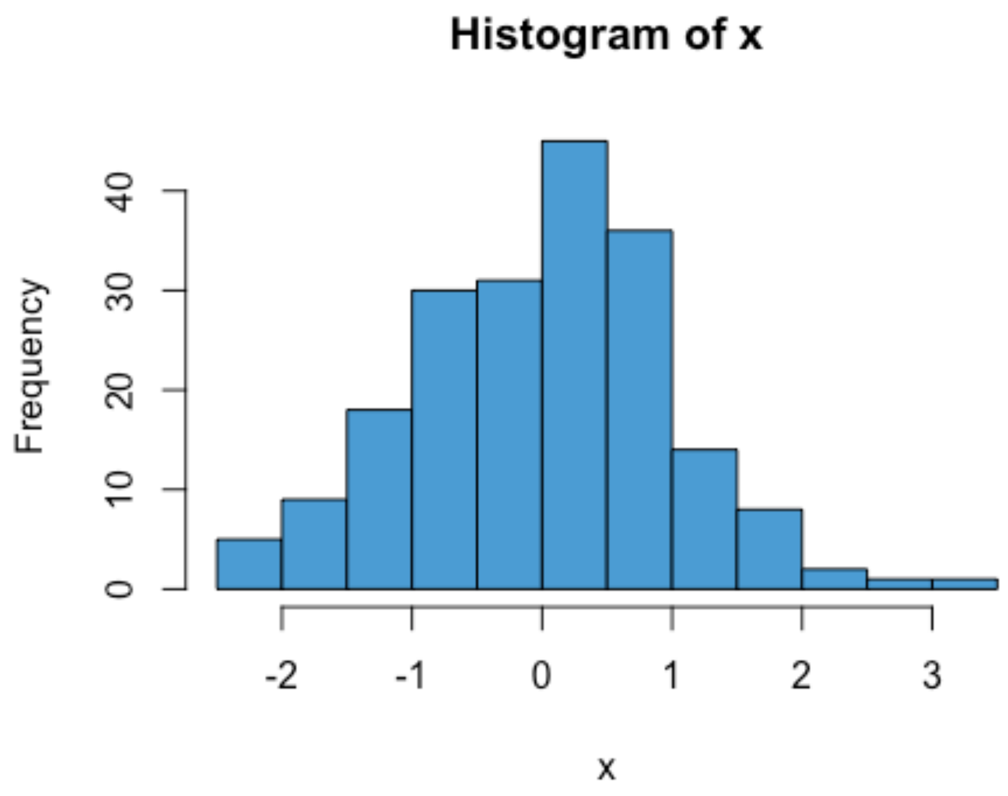
1.1 Why this book?

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.

```r
# 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)] dataset described by Osborne and Suddick (1972). These data have been detailed extensively in a number of papers (McArdle and Epstein 1987; McArdle 1988; Mcardle and Aber 1990; McArdle and Nesselroade 1994) 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 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) 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.

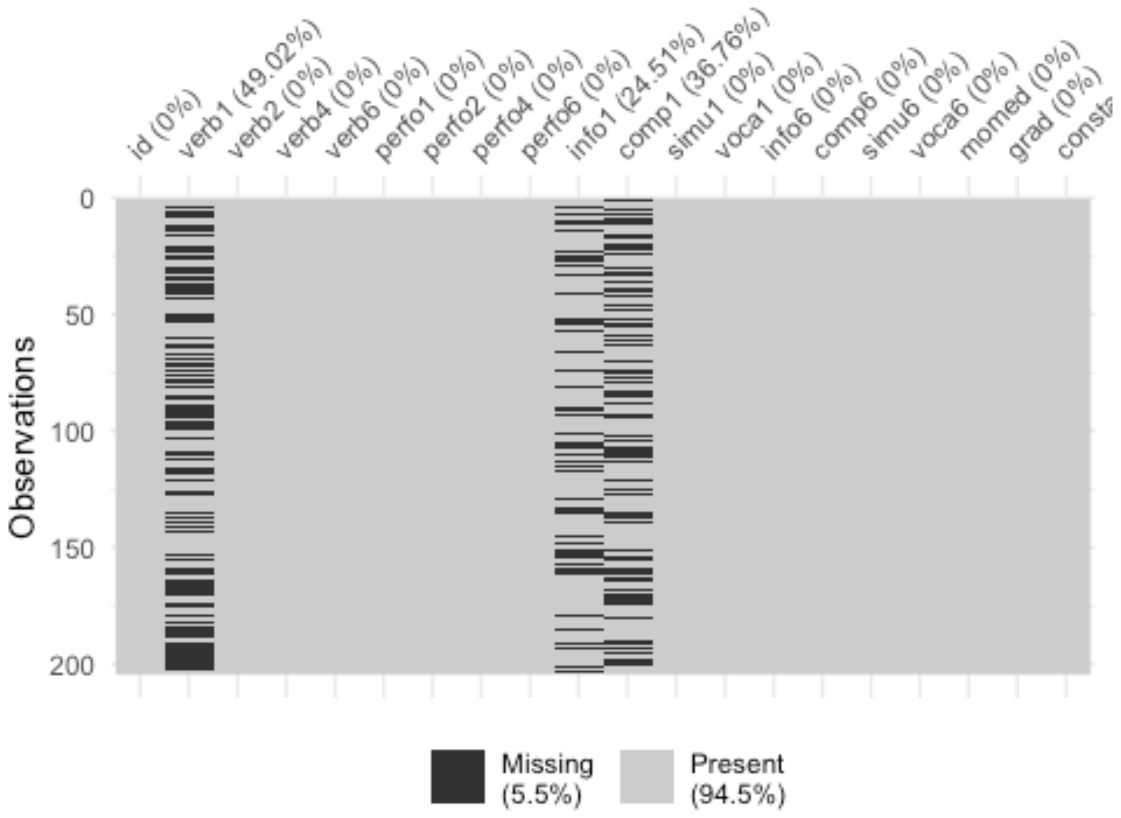
```
na_strings <- c("NA", "N/A", -99)

wisc_miss <- naniar::replace_with_na_all(
  wisc_miss, condition = ~.x %in% na_strings
)
```

See the naniar vignette on recoding NA values for more detailed information on the package functionality.

2.6.3 Missing Data Visualization

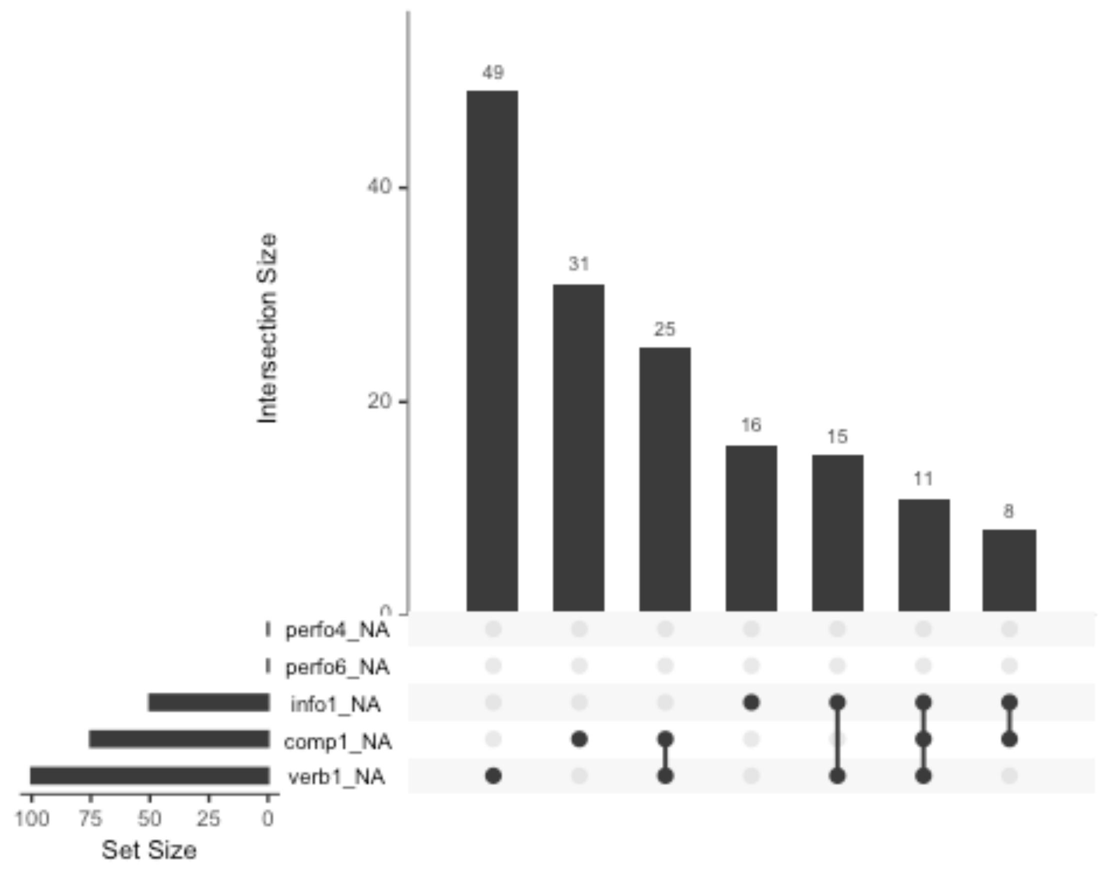
Once we have recoded our data in a consistent manner we can use visualizations to explore the missing data. The vis_miss() function from naniar is a good starting point for visualizing the amount of missing data in our dataset. The plots shows the missing values in black and non-missing values in gray.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

```
naniar::vis_miss(wisc_miss)

## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## ℹ Please use `gather()` instead.
## ℹ The deprecated feature was likely used in the visdat package.
##   Please report the issue at
<]8;;https://github.com/ropensci/visdat/issueshttps://github.com/ropensci/vis
dat/issues]8;;>.
```



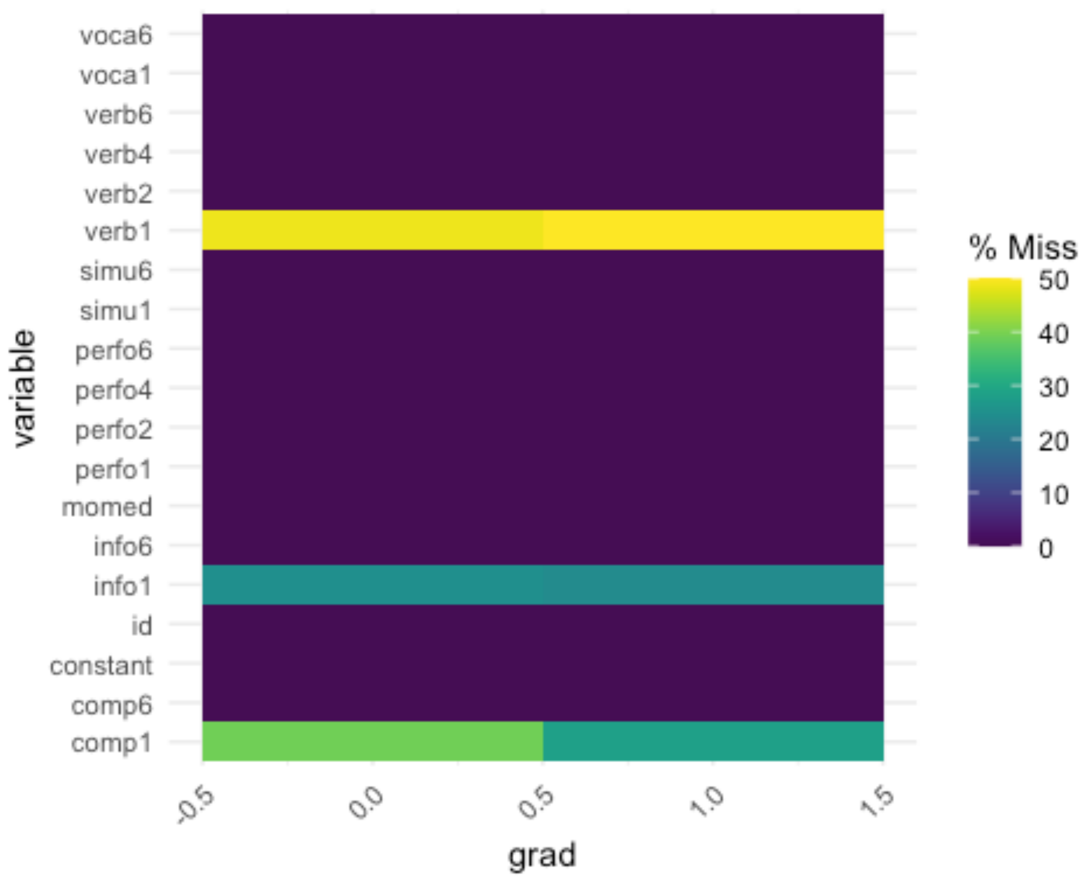
It is often useful to look at combinations of missingness among different variables.

```
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 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) 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.

3 Describing Longitudinal Data

In Chapter 3 we will look at some option for describing and visualizing longitudinal data.

3.1 Example Data

Again we will make use of the WISC data described in Chapter 2. The following commands recreate the wide and long data we will use throughout this chapter.

```
filepath <- "https://quantdev.ssri.psu.edu/sites/qdev/files/wisc3raw.csv"

wisc3raw <- read.csv(file=url(filepath),header=TRUE)

var_names_sub <- c(
  "id", "verb1", "verb2", "verb4", "verb6",
  "perfo1", "perfo2", "perfo4", "perfo6",
  "momed", "grad"
)

wiscraw <- wisc3raw[,var_names_sub]

# reshaping 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), ]

#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"
)]
```

3.2 Describing Means and Variances

Once the wide and long data sets are in place, we can begin describing and plotting the data. Descriptive statistics and visualization are one of the most important aspects of data analysis.

Descriptives and plots will be produced from wide data and long data to show the information that can be gleaned from each construction. Having both in place facilitates learning about the data. Continually keep in mind what portions of the data-box are being described (e.g., persons, variables, occasions).

We can do a quick look at descriptives using the `describe()` function from the `psych` (Revelle 2021) package. Note the `n` in both outputs.

```
psych::describe(wiscwide)

##          vars   n   mean    sd median trimmed   mad   min    max  range
skew
## id          1 204 102.50 59.03 102.50  102.50 75.61  1.00 204.00 203.00
0.00
## verb1       2 204  19.59  5.81  19.34   19.50  5.41  3.33  35.15  31.82
0.13
## verb2       3 204  25.42  6.11  25.98   25.40  6.57  5.95  39.85  33.90 -
0.06
## verb4       4 204  32.61  7.32  32.82   32.42  7.18 12.60  52.84  40.24
0.23
## verb6       5 204  43.75 10.67  42.55   43.46 11.30 17.35  72.59  55.24
0.24
## perfo1      6 204  17.98  8.35  17.66   17.69  8.30  0.00  46.58  46.58
0.35
## perfo2      7 204  27.69  9.99  26.57   27.34 10.51  7.83  59.58  51.75
0.39
## perfo4      8 204  39.36 10.27  39.09   39.28 10.04  7.81  75.61  67.80
0.15
## perfo6      9 204  50.93 12.48  51.76   51.07 13.27 10.26  89.01  78.75 -
0.06
```

```
## momed    10 204  10.81  2.70  11.50   11.00  2.97  5.50  18.00  12.50 -
0.36
## grad     11 204   0.23  0.42   0.00    0.16  0.00  0.00   1.00   1.00  
1.30
##        kurtosis   se
## id        -1.22 4.13
## verb1     -0.05 0.41
## verb2     -0.34 0.43
## verb4     -0.08 0.51
## verb6     -0.36 0.75
## perfo1    -0.11 0.58
## perfo2    -0.21 0.70
## perfo4     0.59 0.72
## perfo6     0.18 0.87
## momed      0.01 0.19
## grad      -0.30 0.03
```

```
psych::describe(wisclong)
```

```
##       vars   n   mean    sd median trimmed   mad  min    max  range  skew
## id       1 816 102.50 58.93 102.50  102.50 75.61 1.00 204.00 203.00  0.00
## momed    2 816  10.81  2.69  11.50   11.00  2.97 5.50  18.00  12.50 -0.36
## grad     3 816   0.23  0.42   0.00    0.16  0.00 0.00   1.00   1.00  1.31
## grade    4 816   3.25  1.92   3.00    3.19  2.22 1.00   6.00   5.00  0.28
## verb     5 816  30.34 11.86  28.46   29.39 11.33 3.33  72.59  69.26  0.71
## perfo    6 816  33.99 16.14  33.14   33.34 18.14 0.00  89.01  89.01  0.34
##       kurtosis   se
## id       -1.20 2.06
## momed     0.03 0.09
## grad     -0.28 0.01
## grade    -1.43 0.07
## verb      0.33 0.42
## perfo    -0.43 0.56
```

3.2.1 Verbal Ability (All Persons and Occasions)

Let's focus on the repeated measures of verbal ability. This step is useful to get a general view of what verbal ability scores look like across persons and occasions, but note that we are *ignoring Time*. In doing so we are not considering how the repeated measures are nested within individuals.

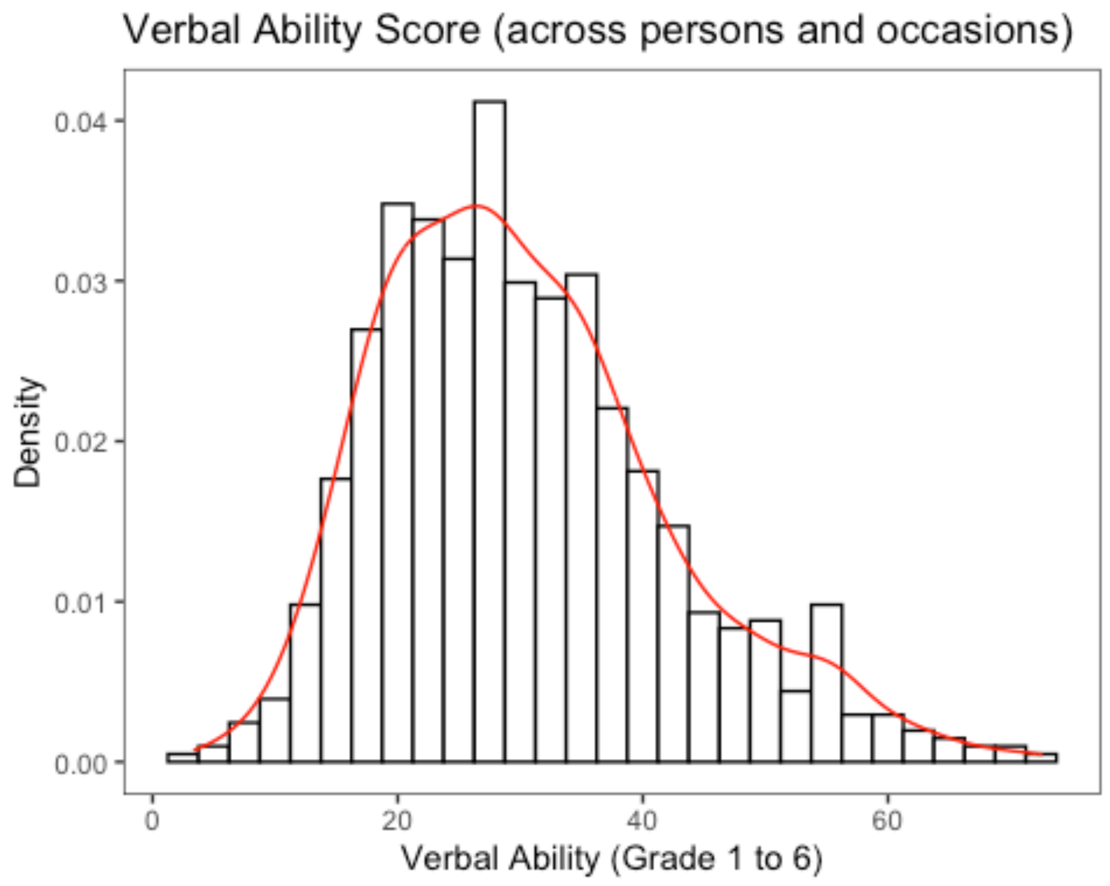
```
psych::describe(wisclong$verb)
```

```
##    vars   n  mean    sd median trimmed   mad  min   max range skew
kurtosis
## X1    1 816 30.34 11.86  28.46   29.39 11.33 3.33 72.59 69.26 0.71
0.33
##      se
## X1 0.42
```

In addition to the descriptive statistics we can look at a boxplot of verbal ability scores across persons and occasions. Here we will start to use the `ggplot2` (Wickham 2016) package.

```
library("ggplot2")
ggplot(data = wisclong, aes(x=verb, y=..density..)) +
  geom_histogram(binwidth=2.5, fill = "white", color = "black") +
  geom_density(color = "red") +
  ggtitle("Verbal Ability Score (across persons and occasions)") +
  xlab("Verbal Ability (Grade 1 to 6)") +
  ylab("Density") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  )
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2
3.4.0.
## ℹ Please use `after_stat(density)` instead.
```



3.2.2 Verbal Ability (Across Time)

Note that our variable is actually “multivariate” because we have repeated measures. We should really consider the time-dependence when we are looking at descriptive statistics and plots.

Let’s now look at **verbal ability scores across time collapsed across individuals**. This can be done using either the `describe()` function and the wide data or the `describeBy()` function and the long data.

Let’s look at descriptives using the wide data.

```
psych::describe(wiscwide[,c("verb1","verb2","verb4","verb6")])

##       vars   n  mean    sd median trimmed   mad   min   max range  skew
## verb1    1 204 19.59  5.81  19.34   19.50  5.41  3.33 35.15 31.82  0.13
## verb2    2 204 25.42  6.11  25.98   25.40  6.57  5.95 39.85 33.90 -0.06
## verb4    3 204 32.61  7.32  32.82   32.42  7.18 12.60 52.84 40.24  0.23
## verb6    4 204 43.75 10.67  42.55   43.46 11.30 17.35 72.59 55.24  0.24
##       kurtosis   se
## verb1    -0.05 0.41
## verb2    -0.34 0.43
## verb4    -0.08 0.51
## verb6    -0.36 0.75
```

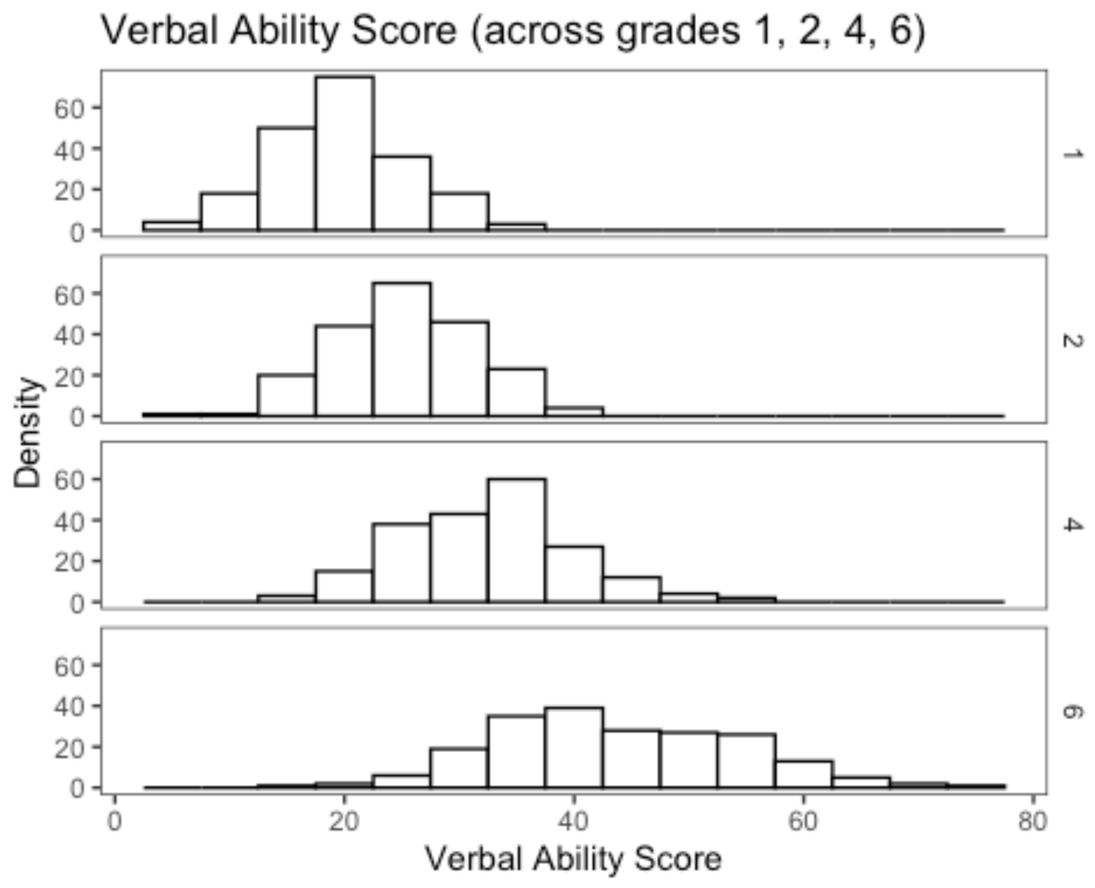
Identical results can be obtained using the long data.

```
psych::describeBy(wisclong[,c("verb")], group = wisclong$grade)

##
##  Descriptive statistics by group
## group: 1
##    vars   n  mean   sd median trimmed  mad  min   max range skew kurtosis
se
## X1    1 204 19.59 5.81  19.34    19.5 5.41 3.33 35.15 31.82 0.13    -0.05
0.41
## ------------------------------------------------------------
## group: 2
##    vars   n  mean   sd median trimmed  mad  min   max range  skew kurtosis
se
## X1    1 204 25.42 6.11  25.98    25.4 6.57 5.95 39.85  33.9 -0.06    -0.34
0.43
## ------------------------------------------------------------
## group: 4
##    vars   n  mean   sd median trimmed  mad  min   max range skew kurtosis
se
## X1    1 204 32.61 7.32  32.82   32.42 7.18 12.6 52.84 40.24 0.23    -0.08
0.51
## ------------------------------------------------------------
## group: 6
##    vars   n  mean    sd median trimmed  mad   min   max range skew
```

```
kurtosis
## X1    1 204 43.75 10.67  42.55   43.46 11.3 17.35 72.59 55.24 0.24    -
0.36
##       se
## X1 0.75
```

We can visualize the distribution of verbal scores across grades in a number of different ways. Here we have a histogram.

```
ggplot(data=wisclong, aes(x=verb)) +
  geom_histogram(binwidth=5, pad = TRUE, fill="white", color="black") +
  facet_grid(grade ~ .) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Verbal Ability Score") +
  ylab("Density") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank()
  )

## Warning: Duplicated aesthetics after name standardisation: pad
```

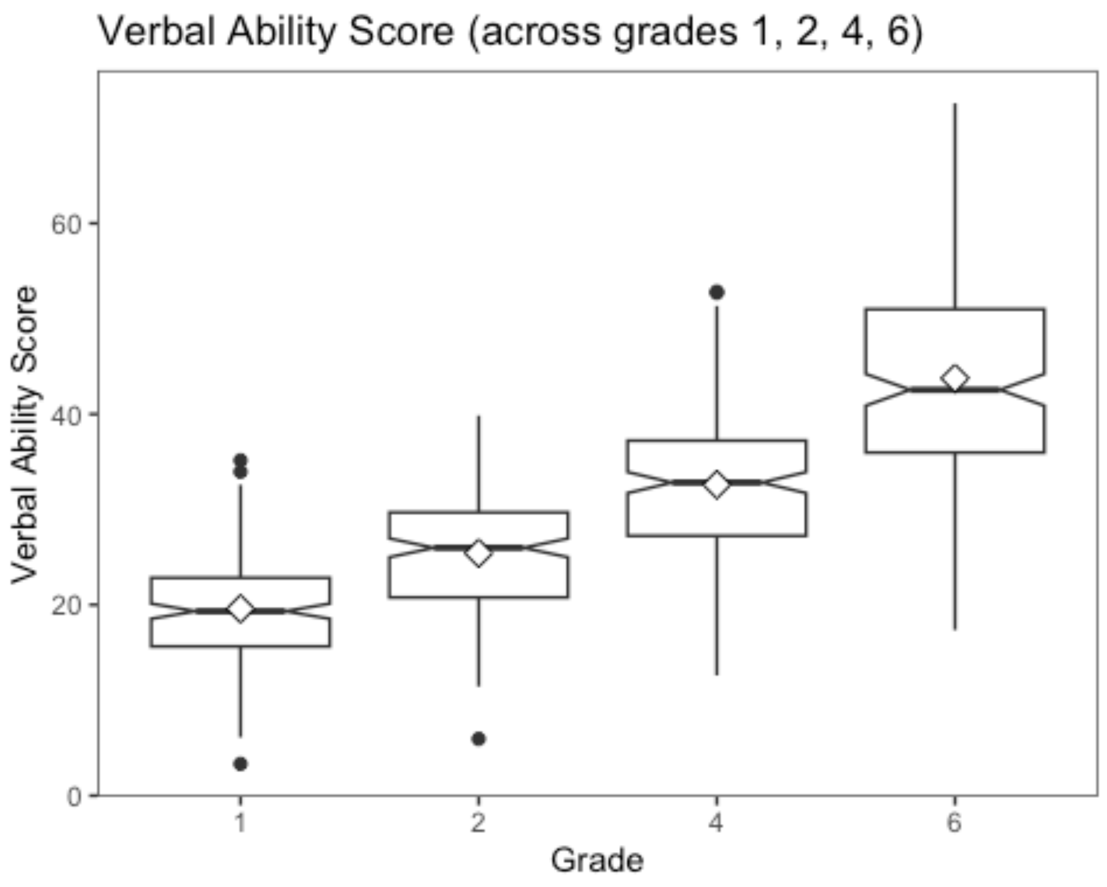


We can also create notched boxplots of the within-grade distributions (across individuals).

From Wikipedia: *Notched box plots apply a notch or narrowing of the box around the median. Notches are useful in offering a rough guide of the significance of the difference of medians; if the notches of two boxes do not overlap, this can provide evidence of a statistically significant difference between the medians.*

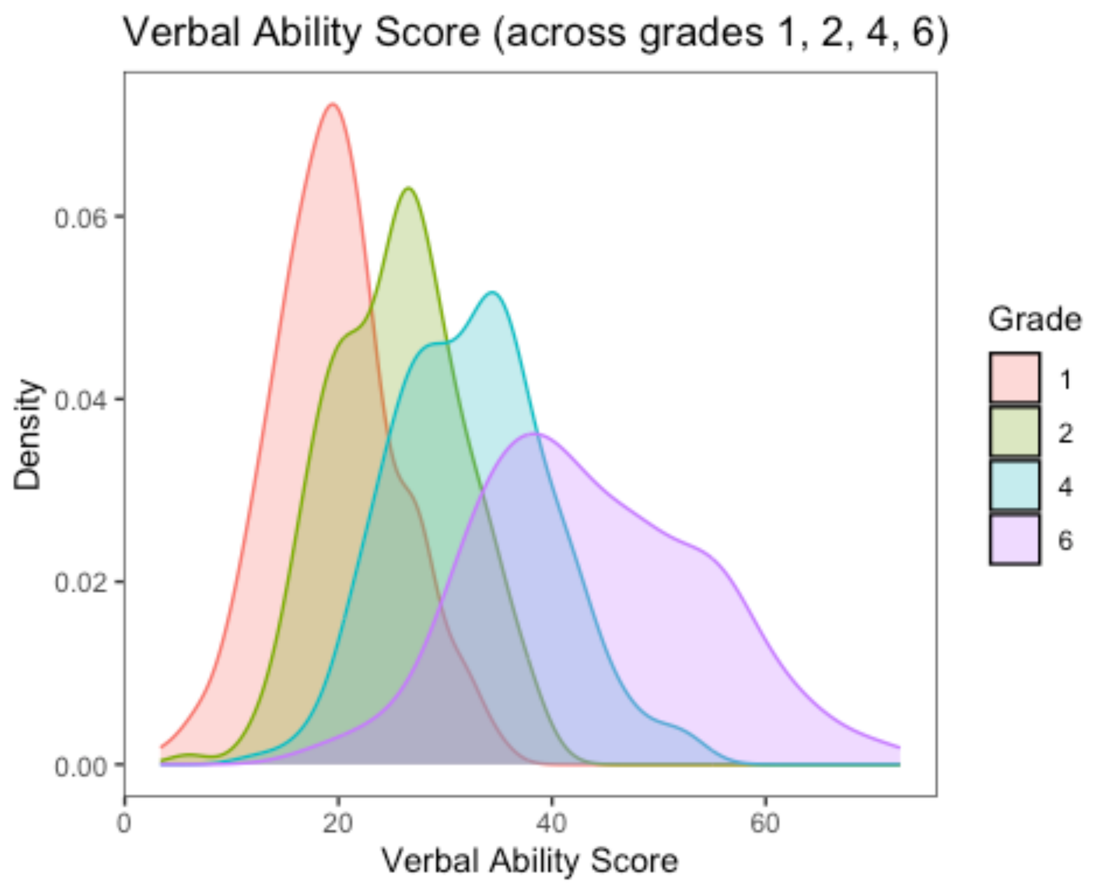
Adding the mean value to the plot gives us additonal information about central tendency and skew of the distribution.

```
#boxplot by grade
ggplot(data=wisclong, aes(x=factor(grade), y=verb)) +
  geom_boxplot(notch = TRUE) +
  stat_summary(fun="mean", geom="point", shape=23, size=3, fill="white") +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  ylab("Verbal Ability Score") +
  xlab("Grade") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank()
  )
```



Finally, we can view overlapping densities of the within-grade distributions of verbal ability scores.

```
ggplot(data=wisclong, aes(x=verb)) +
  geom_density(aes(group=factor(grade), colour=factor(grade),
fill=factor(grade)), alpha=0.3) +
  guides(colour="none",  fill=guide_legend(title="Grade")) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  ylab("Density") +
  xlab("Verbal Ability Score") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank()
  )
```



Notice in these plots how much “change” there is at the sample level across grades. Is that expected?

3.3 Describing Covariances

In the previous section we looked at the means and variances. Because these are repeated measures, we can also look at covariances and correlations over time. A simple covariance and correlation matrix of the verbal scores across grades can be produced using the `cov()` and `cor()` function.

```
cov(wiscwide[,c("verb1","verb2","verb4","verb6")], use="complete.obs")
```

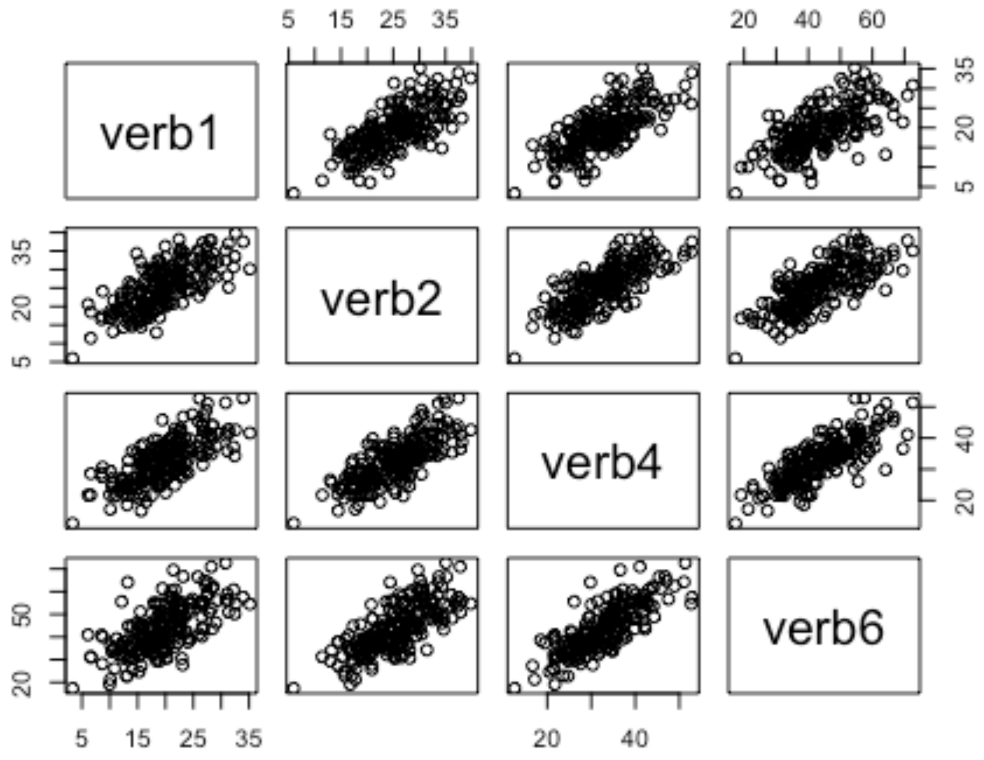
```
##          verb1    verb2    verb4     verb6
## verb1 33.72932 25.46388 30.88886  40.51478
## verb2 25.46388 37.28784 33.81957  47.40488
## verb4 30.88886 33.81957 53.58070  62.25489
## verb6 40.51478 47.40488 62.25489 113.74332
```

```
cor(wiscwide[,c("verb1","verb2","verb4","verb6")], use="complete.obs")
```

```
##           verb1     verb2     verb4     verb6
## verb1 1.0000000 0.7180209 0.7265974 0.6541040
## verb2 0.7180209 1.0000000 0.7566242 0.7279080
## verb4 0.7265974 0.7566242 1.0000000 0.7974552
## verb6 0.6541040 0.7279080 0.7974552 1.0000000
```

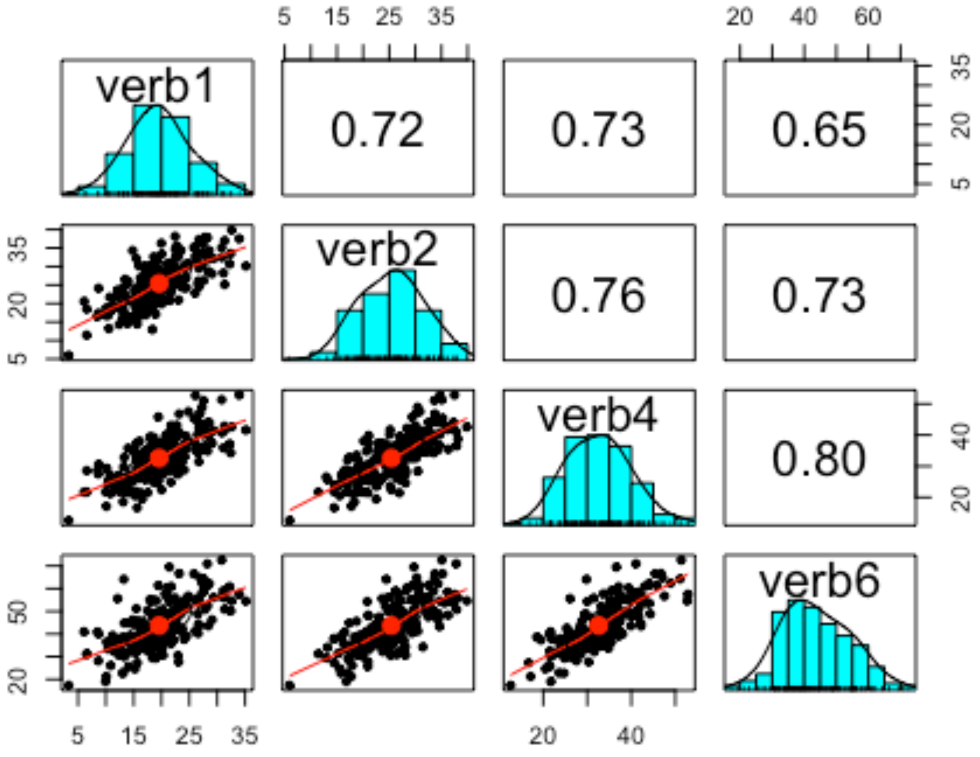
A plot corresponding to the correlation matrix can be obtained in a number of different ways. First, using the `pairs()` function from base R.

```
pairs(wiscwide[,c("verb1","verb2","verb4","verb6")])
```



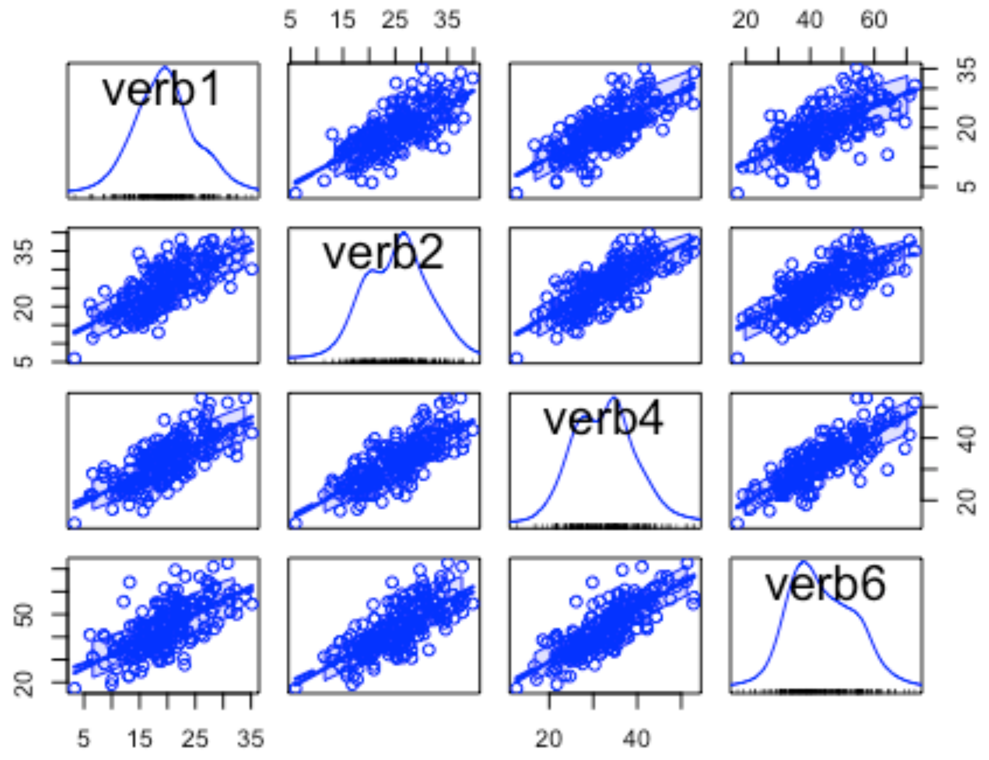
There is also a `pairs.panel()` function in the `psych` package. Here we see a LOESS smoothed fit line in red.

```
psych::pairs.panels(wiscwide[,c("verb1","verb2","verb4","verb6")])
```



Finally, the`scatterplotMatrix()` from the `car` (Fox and Weisberg 2019) package can be used to create scatterplot matrices with confidence bands around the line of best fit.

```
car::scatterplotMatrix(~ verb1 + verb2 + verb4 + verb6, data=wiscwide)
```

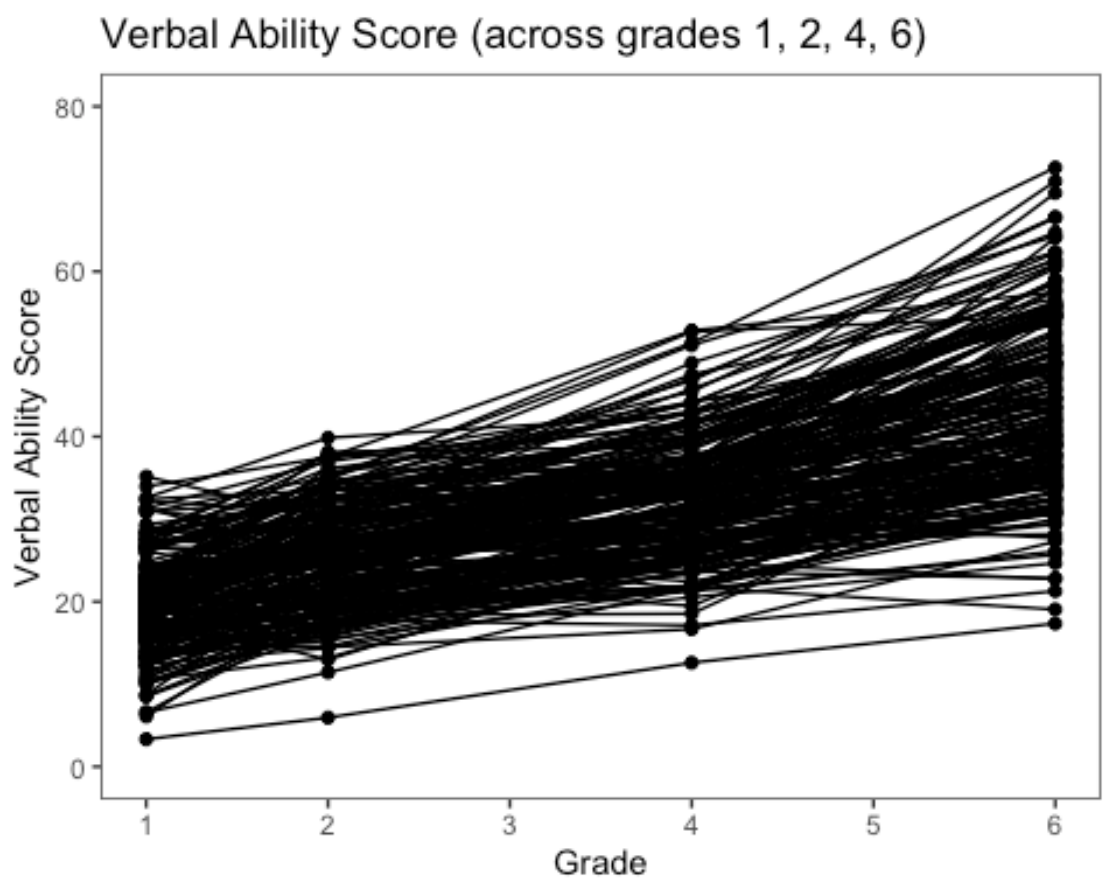


Each of these functions can be customized with additional features. Those interested in specifics should consult the help documentation for each function (e.g. `?car::scatterplotMatrix`). It is also worth noting the default behavior of these functions is to provide automatic, data-based ranges for each pair of variables separately.

3.4 Individual-Level Descriptives

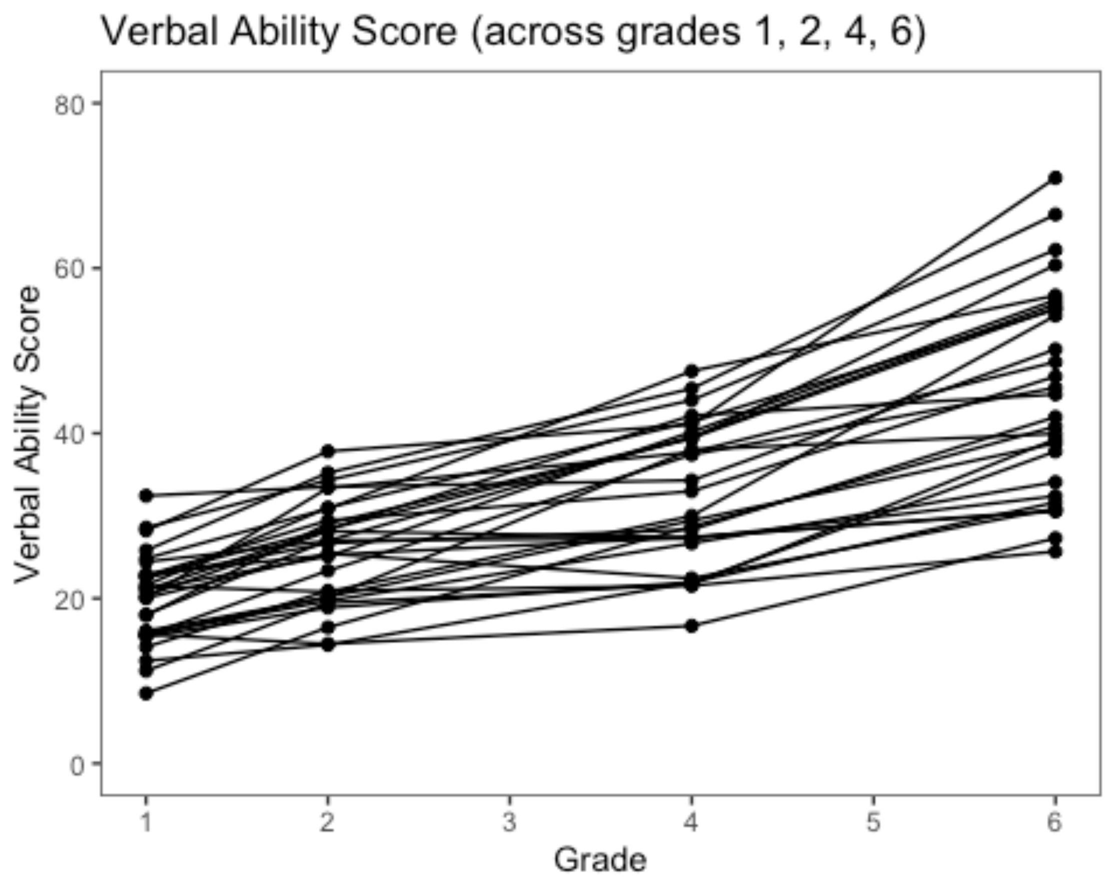
Note that our interest is often in *individual* development, rather than sample development. We need to consider how each individual is changing over time. Thus, we are interested in verbal ability across Time for each individual person. Visualization is typically our best tool for synthesizing the large amounts of information in individual-level data.

```
ggplot(data = wisclong, aes(x = grade, y = verb, group = id)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ylim(0,80) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Grade") +
  ylab("Verbal Ability Score") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank()
  )
```



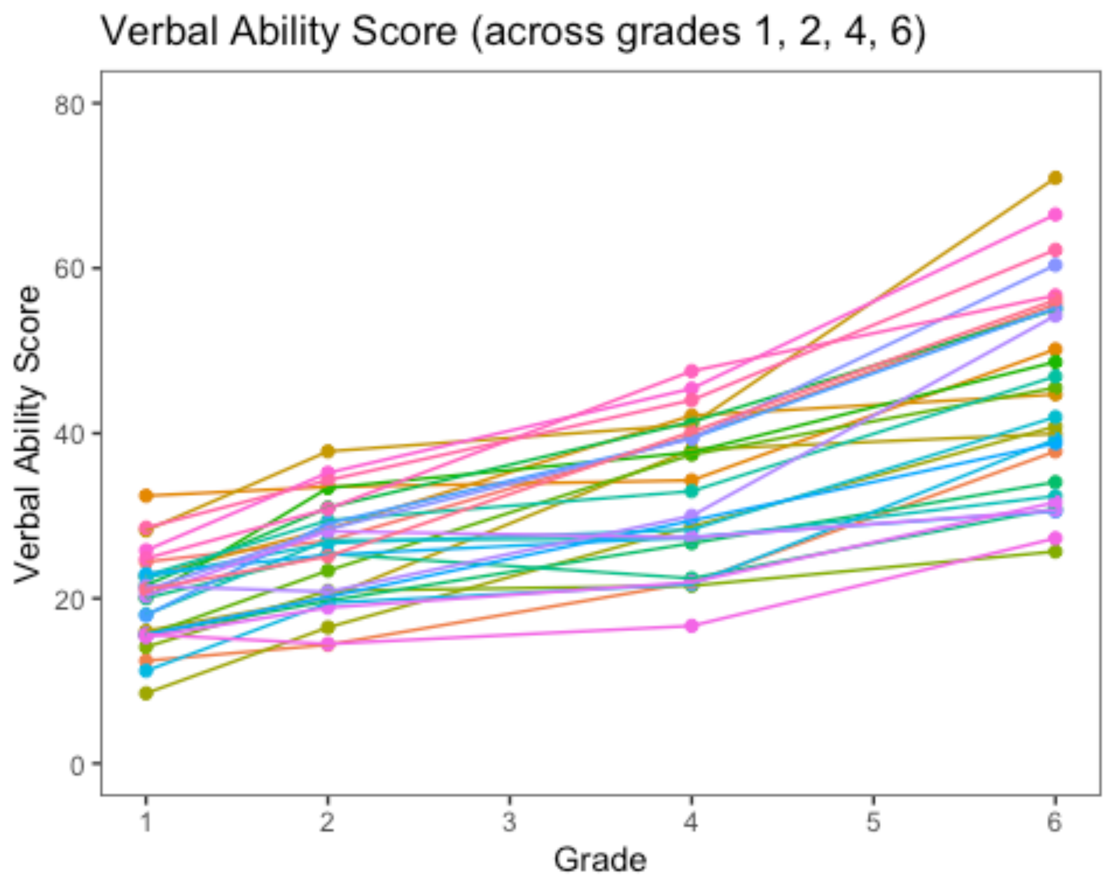
Sometimes the “blob” gets too dense. This can be fixed by selecting a subset of persons to visualize.

```
ggplot(subset(wisclong, id < 30), aes(x = grade, y = verb, group = id)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ylim(0,80) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Grade") +
  ylab("Verbal Ability Score") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank()
  )
```



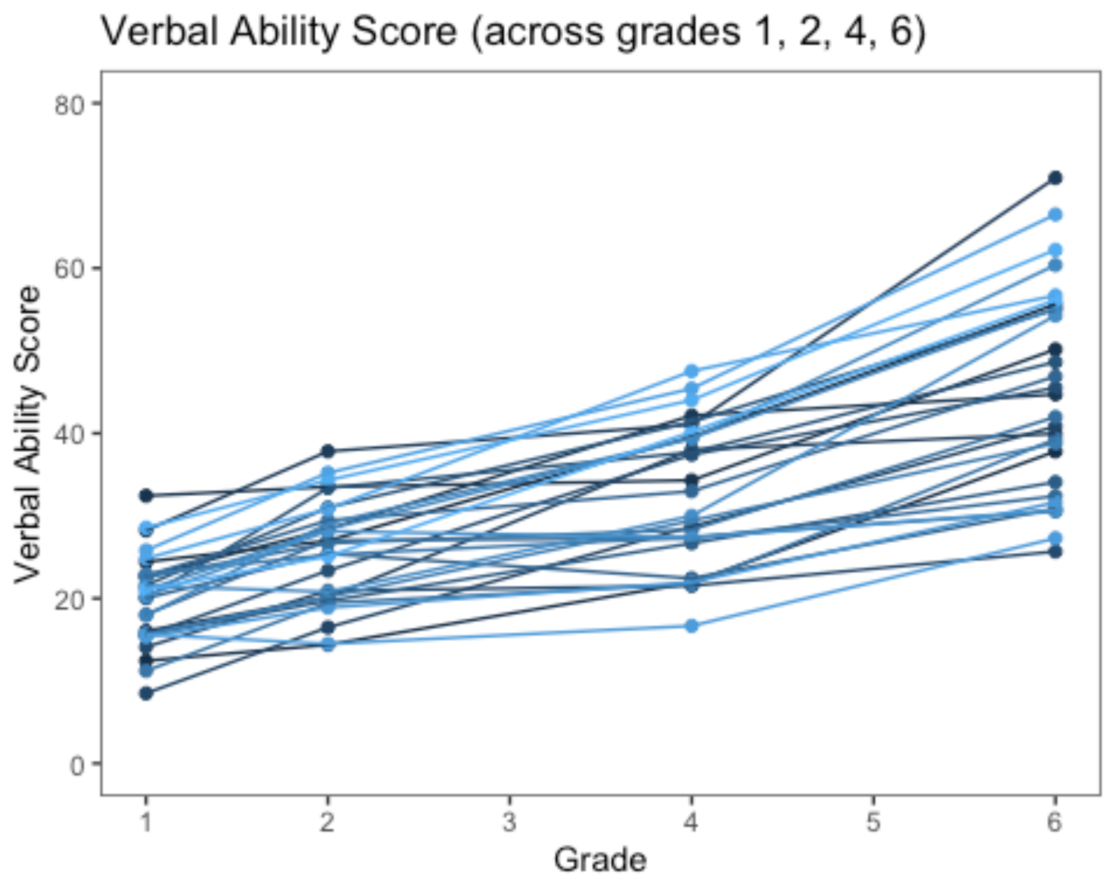
We can add some color to our plot using the `color` argument and treating `id` as a factor.

```
ggplot(subset(wisclong, id < 30), aes(x = grade, y = verb, group = id, color
= factor(id))) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ylim(0,80) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Grade") +
  ylab("Verbal Ability Score") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    legend.position = "none"
  )
```



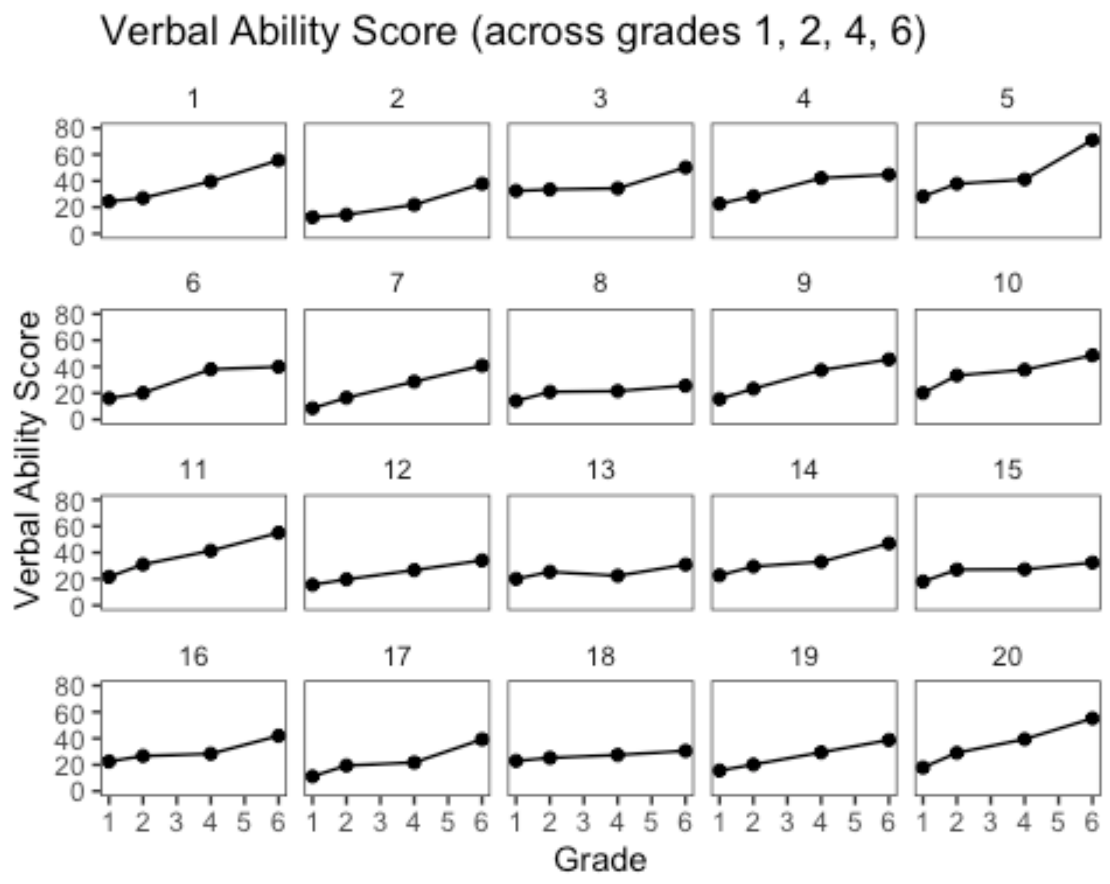
We can also get a gradient of colors by treatingid as continuous.

```
ggplot(subset(wisclong, id < 30), aes(x = grade, y = verb, group = id, color
= id)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ylim(0,80) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Grade") +
  ylab("Verbal Ability Score") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    legend.position = "none"
  )
```



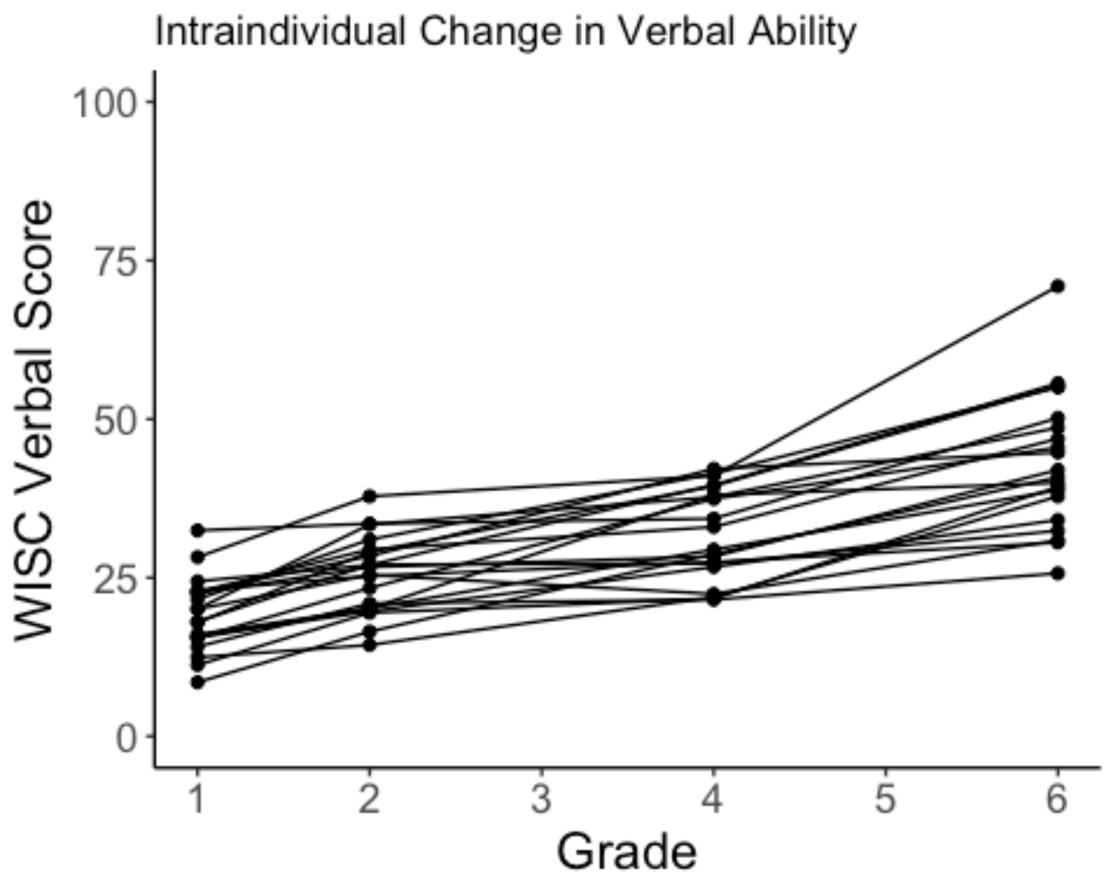
It is also sometimes useful to look at the collection of individual-level plots.

```
ggplot(subset(wisclong, id <= 20), aes(x = grade, y = verb)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ylim(0,80) +
  ggtitle("Verbal Ability Score (across grades 1, 2, 4, 6)") +
  xlab("Grade") +
  ylab("Verbal Ability Score") +
  theme_bw() +
  facet_wrap( ~ id) +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    strip.background = element_blank(),
    legend.position = "none"
  )
```



Some other aesthetics to get to the formal APA style.

```
#ggplot version .. see also http://ggplot.yhathq.com/docs/index.html
ggplot(subset(wisclong, id <= 20), aes(x = grade, y = verb, group = id)) +
  geom_point() +
  geom_line() +
  xlab("Grade") +
  ylab("WISC Verbal Score") +
  ylim(0,100) +
  scale_x_continuous(breaks=seq(1,6,by=1)) +
  ggtitle("Intraindividual Change in Verbal Ability") +
  theme_classic() +
  #increase font size of axis and point labels
  theme(axis.title = element_text(size = rel(1.5)),
        axis.text = element_text(size = rel(1.2)),
        legend.position = "none")
```



Saving the plot file. See also outputting plots to a file.

```
ggsave(filename = "wiscverbal.png", width = 5, height = 5, dpi=300)
```

Now we have a good set of strategies to apply when looking at new longitudinal data.

3.5 References

4 Matrix Algebra

In Chapter 4 we will briefly review some basic algebra results useful for this course. Those needing a reliable reference for basic results in matrix algebra should consult the *The Matrix Cookbook* at https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf.

4.1 Types of matrices

Remember that matrices are defined by rows (the first dimension) and columns (the second dimension):

$$\underset{m\times n}{\mathbf{A}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

You can refer to a specific element in matrix using a subscript of the row and column index (e.g. a_{31}). For our purposes there are a few *special* matrices worth mentioning,

4.1.1 Square

A square matrix has the same number of rows and columns. Covariance and correlation matrices are square.

$$\underset{n\times n}{\mathbf{A}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

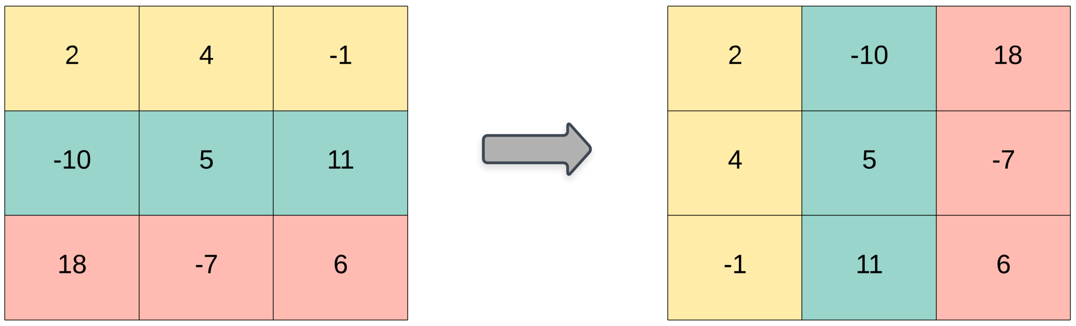
4.1.2 Symmetric

A symmetric matrix is a square matrix that equals its transpose. This means that corresponding entries on either side of the main diagonal are equal.

Matrix Transpose

The transpose of a matrix is an operator which flips a matrix over its diagonal. That is, it switches the row and column indices of the matrix A by producing another matrix, often denoted by A' (or A^T).

Graphical Depiction of a Matrix Transpose



2	4	-1
-10	5	11
18	-7	6

2	-10	18
4	5	-7
-1	11	6

https://leetcode.com/problems/transpose-matrix/

4.1.3 Diagonal

A diagonal matrix is a special case of a square symmetric matrix in which there are values along the diagonal, but zeros elsewhere:

4.1.4 Identity

An identity matrix is a special case of a diagonal matrix in which the elements of the diagonal are all 1:

$$\underset{n\times n}{\mathbf{I}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Any matrix multiplied by an identity matrix is unchanged.

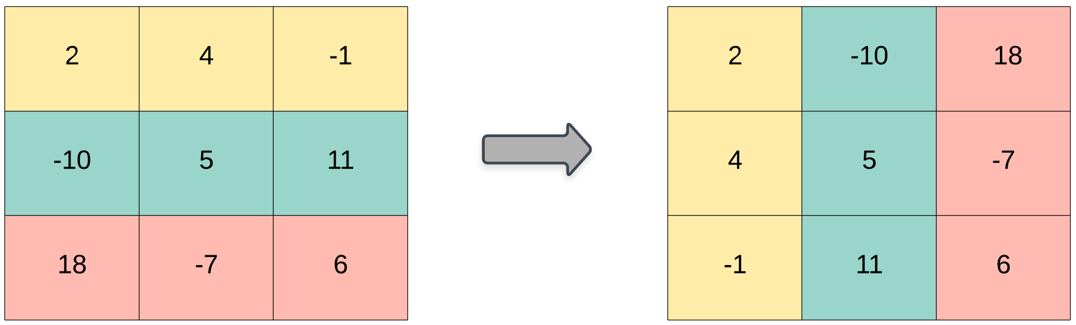
4.2 Operations on Matrices

4.2.1 Matrix Transpose

As stated earlier the transpose of a matrix is an operator which flips a matrix over its diagonal. That is, it switches the row and column indices of the matrix A by producing another matrix, often denoted by A' (or A^T). Some useful properties of the matrix transpose include:

$$ (\mathbf{A + B})' = \mathbf{A' + B'}\\ (c\mathbf{A'}) = c(\mathbf{A'}) = (\mathbf{A'})c \\ (\mathbf{A'B}) = \mathbf{B'A}\\ (\mathbf{AB})' = \mathbf{B'A'}\\ (\mathbf{A'})' = \mathbf{A} $$

Graphical Depiction of a Matrix Transpose



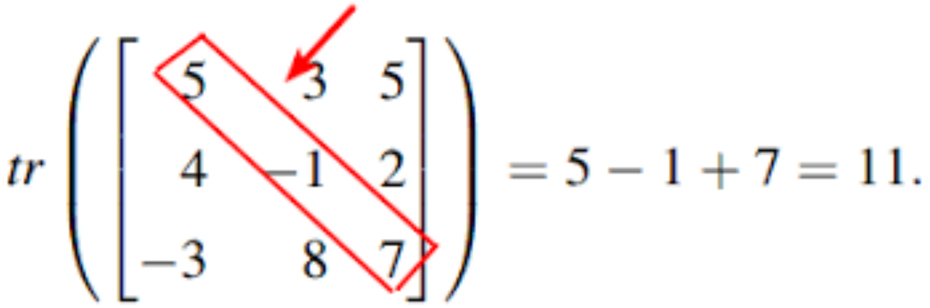
https://leetcode.com/problems/transpose-matrix/

4.2.2 Matrix Trace

The *trace* of a square matrix is the sum of elements along the diagonal. The trace is only defined for a square matrix. For an $n \times n$ matrix the trace is defined as follows:

$$tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + \ldots + a_{nn}$$

Graphical Depiction of a Matrix Trace


$$tr\left(\begin{bmatrix} 5 & 3 & 5 \\ 4 & -1 & 2 \\ -3 & 8 & 7 \end{bmatrix}\right) = 5 - 1 + 7 = 11.$$

Some useful properties of the matrix trace include:

$$ tr(\mathbf{A + B}) = tr(\mathbf{A}) + tr(\mathbf{B})\\ tr(c\mathbf{A}) = c(tr(\mathbf{A})) \\ tr(\mathbf{A}) = tr(\mathbf{A'})\\ tr(\mathbf{AB}) = tr(\mathbf{BA})\\ tr(\mathbf{ABC}) = tr(\mathbf{CAB})=tr(\mathbf{BCA}) $$

4.2.3 Addition

For addition, matrices must be of the same order. Addition of two matrices is accomplished by adding corresponding elements, $c_{ij} = a_{ij} + b_{ij}$

$$ \mathbf{A} = \begin{bmatrix} 10 & 5 \\ 9 & 1 \end{bmatrix} , \enspace \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 20 & 0 \end{bmatrix}, \enspace \textrm{then } \mathbf{A}+\mathbf{B}= \begin{bmatrix} 12 & 6 \\ 29 & 1 \end{bmatrix} $$

Matrix addition is *commutative* (gives the same result whatever the order of the quantities involved),

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$$

and *associative* (gives the same result whatever grouping their is, as long as order remains the same),

$$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$$

and

$$\mathbf{A} + (-\mathbf{B}) = (\mathbf{A} - \mathbf{B}).$$

4.2.4 Subtraction

Like addition, subtraction requires matrices of the same order. Elements in the difference matrix are given by the algebraic difference between corresponding elements in matrices being subtracted:

$$ \mathbf{A} = \begin{bmatrix} 10 & 5 \\ 9 & 1 \end{bmatrix} , \enspace \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 20 & 0 \end{bmatrix}, \enspace \textrm{then } \mathbf{A}-\mathbf{B}= \begin{bmatrix} 8 & 4 \\ -11 & 1 \end{bmatrix} $$

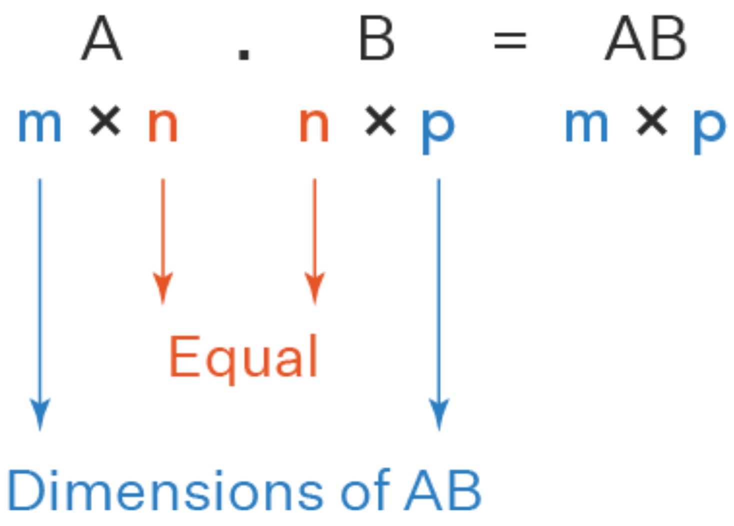
4.2.5 Matrix Multiplication

Three useful *rules* to keep in mind regarding matrix multiplication:

1. Only matrices of the form $(m \times n) * (n \times p)$ are conformable for multiplication. The number of columns in the premultiplier must equal the number of rows in the post multiplier.

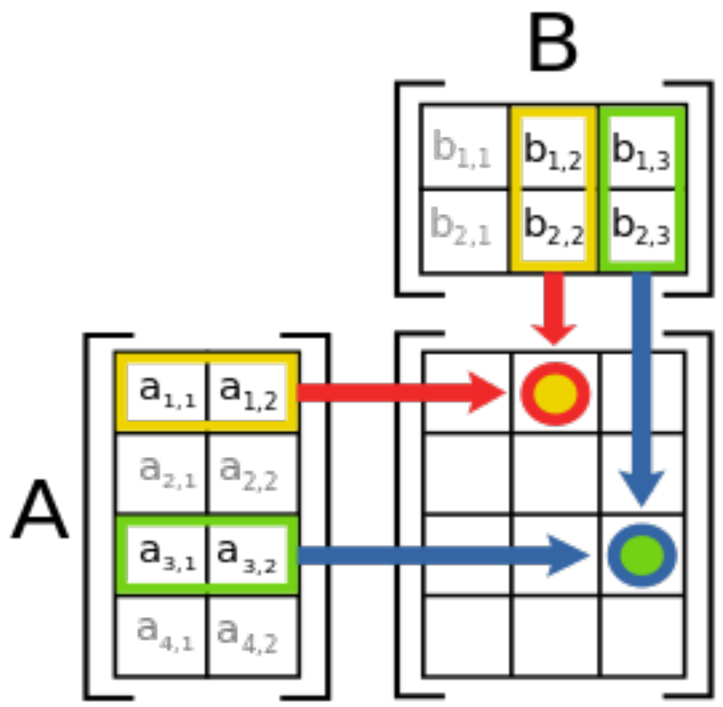
2. The product matrix will have the following order: $\mathbf{A}_{m\times n}\mathbf{B}_{n\times p} = \mathbf{C}_{m\times p}$.

Graphical Depiction of Rules 1 and 2



3. The element c_{ij} in the product matrix is the result of multiplying row i of the premultiplier matrix, and row j of the post multiplier matrix (e.g. $(c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j})$).

Graphical Depiction of Rule 3



https://code.kx.com/q/ref/mmu/

Matrix multiplication is associative (i.e. rearranging the parentheses in an expression will not change the result). That is,

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

and is distributive with respect to addition,

$$ \mathbf{A(B+C)} = \mathbf{AB + AC} \\ \mathbf{(B+C)A} = \mathbf{BA + CA} \\ $$

If *c* is a scalar, then

$$c(\mathbf{AB}) = c(\mathbf{A})\mathbf{B} = \mathbf{A}(c\mathbf{B}) = (\mathbf{AB})c$$

or equivalently,

$$ \mathbf{A} = \begin{bmatrix} 10 & 5 \\ 9 & 1 \end{bmatrix}, \enspace k=2, \enspace k\mathbf{A} = \begin{bmatrix} 20 & 10 \\ 18 & 2 \end{bmatrix}. $$

In general, matrices that can be multiplied are called ‘compatible’ or ‘comformable.’ Matrices in which the *inner dimensions* (i.e., columns of **A**, rows of **B**) do not match are called ‘incompatible’ or ‘non-conformable.’ These cannot be multiplied.

4.2.6 Matrix Division

Division is not defined for matrix operations, but may be accomplished by multiplication by the inverse matrix. In algebra, the reciprocal of a scalar is, by definition, the scalar raised to the minus one power (e.g. $5^{-1} = 1/5$), and equations may be solved by multiplication by reciprocals.

For example:

$$ 5^{-1} = 1/5\\ 5x=35\\ 5^{-1}(5x)=5^{-1}(35)\\ x = 7 $$

Now consider the following equation where the vector **x** is unknown,

$$\mathbf{A}_{p\times p}\mathbf{x}_{p\times 1} = \mathbf{b}_{p\times 1}$$

Each element in the column vector **x** is unknown and the solution involves solving a set of simultaneous equations for the unknown element of **x**,

$$ a_{11}x_{1} + a_{12}x_{2} + \dots + a_{1p}x_{p} = b1 \\ a_{21}x_{1} + a_{22}x_{2} + \dots + a_{2p}x_{p} = b2 \\ \vdots \\ a_{p1}x_{1} + a_{p2}x_{2} + \dots + a_{pp}x_{p} = bp $$

A solution analogous to the scalar equations above would give the following solution for the elements of the vector **x**:

$$ \mathbf{A}_{p \times p} \mathbf{x}_{p \times 1} = \mathbf{b}_{p \times 1} \\ \mathbf{A}^{-1}_{p \times p}\mathbf{A}_{p \times p} \mathbf{x}_{p \times 1} = \mathbf{A}^{-1}_{p \times p}\mathbf{b}_{p \times 1} \\ \mathbf{I}_{p \times p}\mathbf{x}_{p \times 1} = \mathbf{A}^{-1}_{p \times p}\mathbf{b}_{p \times 1} \\ \mathbf{x}_{p \times 1} = \mathbf{A}^{-1}_{p \times p}\mathbf{b}_{p \times 1} $$

The inverse of a matrix must satisfy the following properties:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

where I is the identity matrix with 1's along the diagonal and 0's elsewhere.

So, why is division undefined for matrices. Here is a quick example. Suppose, **A** is a matrix and **B** is the inverse of **A**, such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

Now, let

$$ \mathbf{A} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} , \enspace \mathbf{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, $$

Then,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This means that $a + b = 1$ and $2a + 2b = 0$, which is a contradiction, suggesting A does not have an inverse.

5 Ordinary Least Squares

In Chapter 5 we will briefly review the assumptions and properties of Ordinary Least Squares (OLS) regression, a cornerstone method that supports many of the other methods we will consider. We will present the regression model in both scalar and matrix forms to facilitate the material to follow.

5.1 Linear Regression Model

Generally, the regression model is written as

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_q x_{qi} + \epsilon_i$$

where

- y_i is the value of the outcome variable for individual i
- β_0 is an *intercept parameter*, the expected value of y_i when the predictor variables are all 0
- β_q is a regression parameter indicating the relation between x_{qi} and the outcome variable, y_i
- ϵ_i are *errors or disturbances*

5.2 Ordinary Least Squares (OLS)

The Ordinary Least Squares (OLS) is one of the most common estimators of the linear regression model. What assumptions do we make with OLS? Why should we care?

How well me meet the assumptions of OLS determines (1) the accuracy of OLS coefficients, and (2) the accuracy of our inferences and substantive hypothesis tests.

5.3 Assumptions of OLS

The assumptions of OLS are as follows:

1. $\mathbb{E}(\epsilon_i) = 0$
2. $\mathbb{E}(\epsilon_i^2) = \sigma^2$ for all i (homoscedasticity)
3. $\mathbb{E}(\epsilon_i \epsilon_j) = 0$ for all $i \neq j$
4. No perfect collinearity among x variables
5. $\mathbb{C}(\epsilon_i, x_{qi}) = 0$ for all i and k

Let's discuss each assumption in more detail.

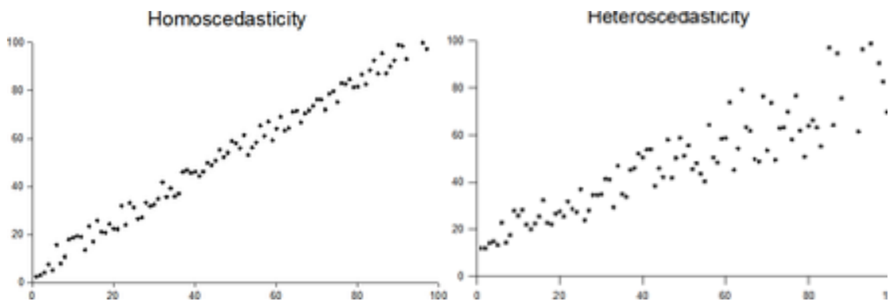
5.3.1 Assumption 1. $\mathbb{E}(\epsilon_i) = 0$

Note that $\mathbb{E}(\quad)$ is the expectation operator. The expected value is an "average" of whatever is inside the parentheses. This assumption states that, on average, the error for the *ith* observation is zero. Here "for all i" means the same is true for all cases.

5.3.2 Assumption 2. Homoscedasticity

In statistics, a vector of random variables is heteroscedastic if the variability of the random disturbance is different across elements of the vector, here our **X**s. The errors or disturbances in our model are homoskedastic if the variance of ϵ_i is a constant (e.g. σ^2), otherwise, they are heteroskedastic.

Graphical Depiction of Homoskedasticity and Heteroskedasticity

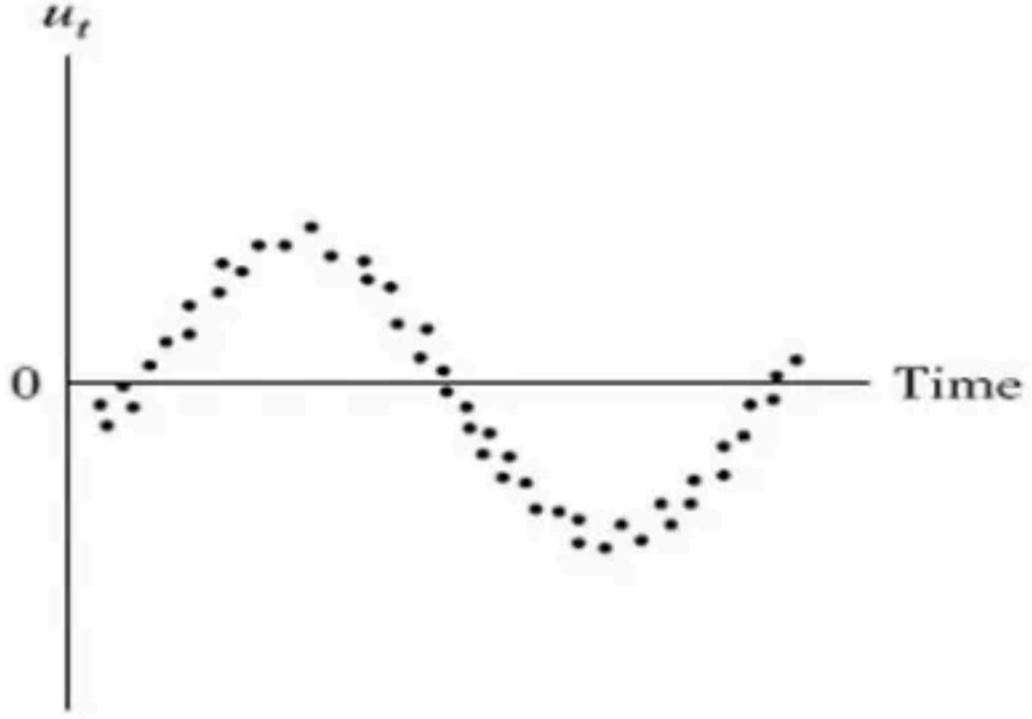


https://en.wikipedia.org/wiki/Homoscedasticity

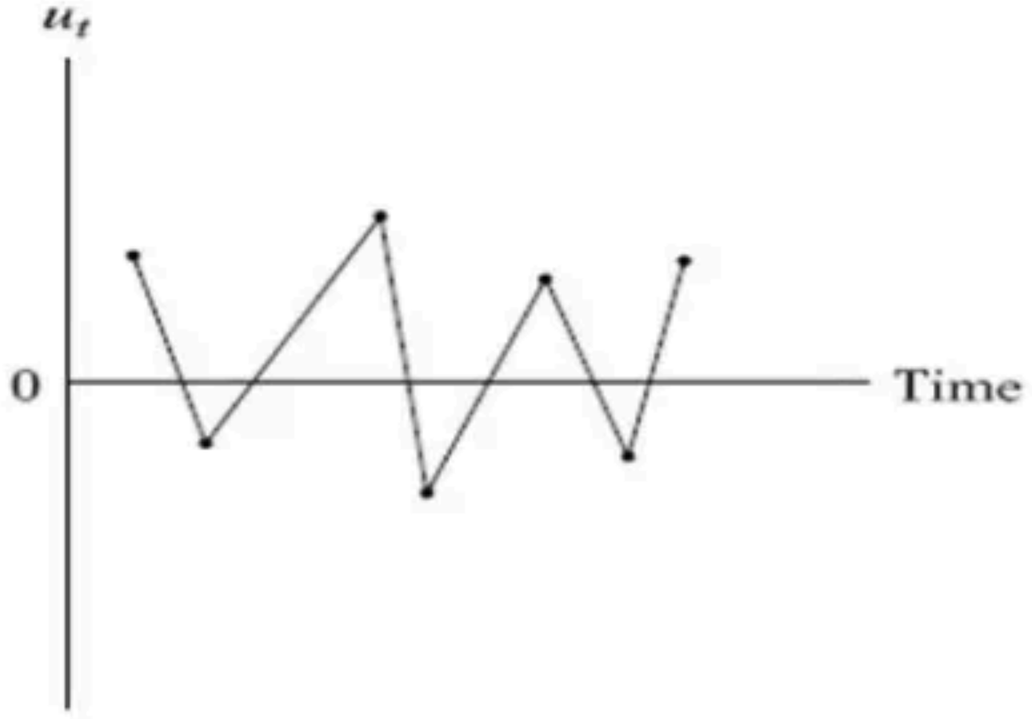
5.3.3 3. $\mathbb{E}(\epsilon_i \epsilon_j) = 0$

Assumption 3 is sometimes referred to as the *autocorrelation assumption*. This assumption states that the error terms of different observations should not be correlated with each other. For example, when we have time series data and use lagged variables we may want to examine residuals for the possibility of autocorrelation.

Graphical Depiction of Positive and Negative Autocorrelation



(*a*)



(*b*)

5.3.4 4. No Perfect Collinearity

Perfect collinearity occurs when one variable is a perfect linear function of any other explanatory variable. If perfect collinearity is found among the **X**s then $(\mathbf{X}'\mathbf{X})$ has no inverse and OLS estimation fails. Perfect collinearity is unlikely except for programming mistakes such as dummy coding all the values in a nominal variable.

5.3.5 5. $\mathbb{C}(\epsilon_i, x_{ki}) = 0$

Note that $\mathbb{C}(\quad)$ is the covariance operator. Assumption five states that that the error of our equation is uncorrelated with all the **X**s. This is often referred to as an *endogeneity assumption*.

This can be a confusing assumption because by definition the residuals $\widehat{e_i}$ are uncorrelated with the **X**s. Here, however, we are concerned with the true errors ϵ_i. Unfortunately, there are a variety of conditions that lead to $\mathbb{C}(\epsilon_i, x_{qi}) \neq 0$ in applied contexts.

Graphical Depiction of Sources of Endogeneity

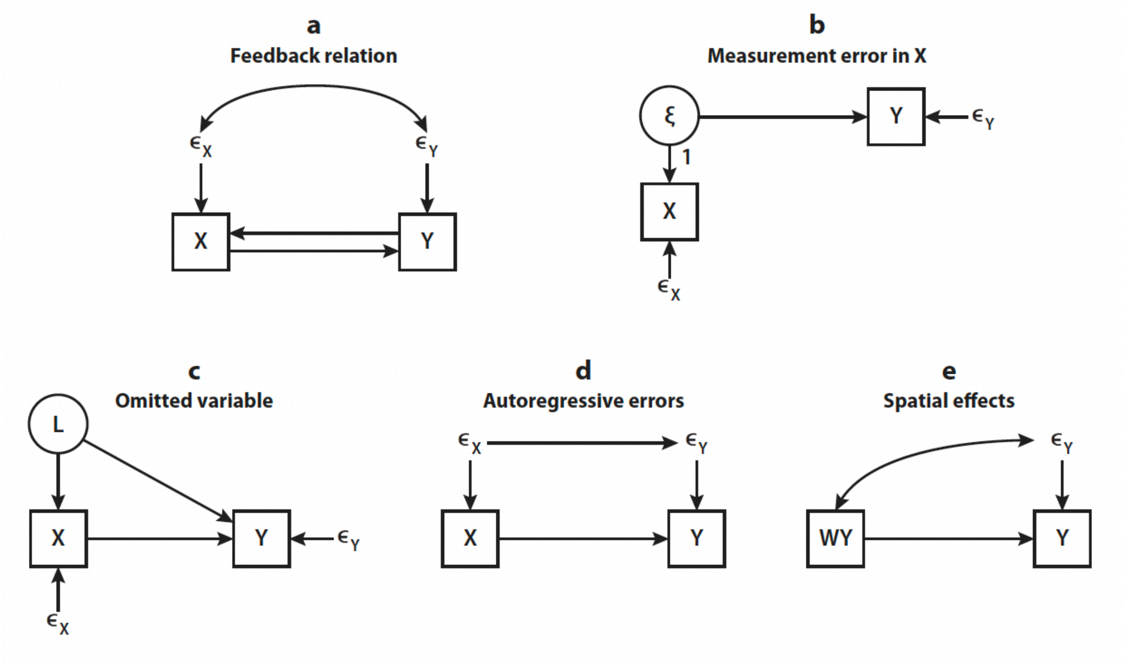


Figure 2
Possible causes of error-covariate (X) correlation.

Figure 2 from Bollen (2012)

If we meet these assumptions what large sample properties can we expect?

5.4 Properties of the OLS Estimator

If assumptions (1) to (5) hold, then the OLS estimator $\widehat{\boldsymbol{\beta}}$ is:

1. A consistent estimator of $\boldsymbol{\beta}$
2. Asymptotically normally distributed

3. Having a variance of $\mathbb{V}(\widehat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$

Notice that we did not assume normality of ϵ_i, y_i *or* x_i.

Let's discuss each of these properties in a little bit more detail.

5.4.1 1. Consistentcy of $\boldsymbol{\beta}$

$\widehat{\boldsymbol{\beta}}$ is the OLS estimator of $\boldsymbol{\beta}$. A consistent estimator is one for which, as the sample size (n) increases, the estimate converges in probability to the value that the estimator is designed to estimate. This is often stated as $plim(\widehat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$. Stated differently, as the sample size grows, the OLS coefficients converge to the true coefficients.

5.4.2 2. Asymptotic Normality

Asymptotic normality is another property of the OLS estimator when all assumptions are met. "Asymptotic" refers to how an estimator behaves as the sample size tends to infinity. "Normality" refers to the normal distribution, so an estimator that is asymptotically normal will have an approximately normal distribution as the sample size gets larger.

5.4.3 Variance of $\hat{\beta}$

Having a variance of $\mathbb{V}(\widehat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$ is another property of the OLS estimator when the previously stated assumptions are met. This means, for example, we can estimate the standard errors from the main diagonal of $\sigma^2(\mathbf{X}'\mathbf{X})^{-1}$ and perform significance testing based on this variance.

5.5 Failure to Meet Assumptions

It is worth thinking about the consequences of not meeting these assumptions.

5.5.1 Failure of Assumption 1.

Assumption 1 states $\mathbb{E}(\epsilon_i) = 0$. This assumption states that, on average, the error for the ith observation is zero. If instead, $\mathbb{E}(\epsilon_i) = c$ and $c \neq 0$, and all other assumptions hold, then only the intercept term is biased. Other coefficients OK.

5.5.2 Failure of Assumption 2 or 3.

Assumptions 2 and 3 are the *homoskedasticity* and *no autocorrelation* assumption, respectively. If we violate (2) or (3), but all other assumptions hold, (1) variance of $\hat{\beta}$ is no longer dependable, (2) SEs possibly inaccurate, and (3) significance tests are possibly inaccurate. However, importantly, $\hat{\beta}$ is still an unbiased and consistent estimator.

5.5.3 Failure of Assumption 5.

Assumption (5) states that that the error of our equation is uncorrelated with all the **X**s. If this assumption fails, while others hold, OLS is no longer a consistent estimator.

5.6 Regression and Matrix Notation

Now that we have reviewed the assumptions of OLS, let's return to the linear regression model and translate it into a matrix form.

5.6.1 An Intercept-Only Model

First, let's take a simpler form of the model, an intercept-only model where

$$y_i = \beta_0 1_i + \epsilon_i.$$

Note that we have made the "silent" 1 explicit. This will become important later (e.g., when fitting growth models). It is worthwhile to look at regression model without predictors to understand what it can tell us about the nature of the intercept (or constant).

So here we have no predictors, what is β_0?

Here, β_0 is the mean of the response variable, and we can show this with some algebra,

$$\mathbb{E}(y_i) = \mathbb{E}(\beta_0 1_i + \epsilon_i) = \beta_0 1_i + \mathbb{E}(\epsilon_i) = \beta_0$$

where $\mathbb{E}(\epsilon_i) = 0$ (Assumption 1).

5.6.2 Intercept-Only Model in Matrix Form

Translating into matrix form, y_i can be written as an N x 1 matrix (a column vector). More specifically, for $i = 1$ to N individuals,

$$y_i = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \mathbf{Y}$$

.

(Remember, matrices are often designated as bold capital letters)

Doing the same for all the other parts of the model, we get

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} [\beta_0] + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

Note that we have taken care that each matrix is of an order that will allow for matrix multiplication.

5.6.3 Simple Regression in Matrix Form

Now, let's expand our regression model by adding a predictor x_{1i}. Our model becomes

$$y_i = \beta_0 + \beta_1 x_{1i} + \epsilon_i$$

Written out explicitly in matrix form, the model is

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1, x_{11} \\ 1, x_{12} \\ \vdots \\ 1, x_{1N} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

5.6.4 Multiple Regression in Matrix Form

Finally, extending the model to the general case with q predictor variables, we have

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_q x_{qi} + \epsilon_i$$

which is written out in matrix form as

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{N\times 1} = \underbrace{\begin{bmatrix} 1, x_{11}, \ldots, x_{q1} \\ 1, x_{12}, \ldots, x_{q2} \\ \vdots \\ 1, x_{1N}, \ldots, x_{qN} \end{bmatrix}}_{N\times(q+1)} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix}}_{(q+1)\times 1} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}}_{N\times 1}$$

Where we have the following elements:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}; \mathbf{X} = \begin{bmatrix} 1, x_{11}, \ldots, x_{q1} \\ 1, x_{12}, \ldots, x_{q2} \\ \vdots \\ 1, x_{1N}, \ldots, x_{qN} \end{bmatrix}; \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix}; \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

Observe the order of the matrices/vectors. On the right hand side you are matrix multiplying a $N \times (q+1)$ matrix with a $(q+1) \times 1$ vector. This yields an $N \times 1$ vector, to which another $N \times 1$ vector $\boldsymbol{\epsilon}$ is added, and this is equal to our outcome vector $\mathbf{Y}$ which is also $N \times 1$.

When we implement this model in R, it will be important to know the portions of the model that are in our data frame, y_i and $x_1, \ldots, x_q$, and to have them structured properly. This will become clear in the examples below.

Now that we have the model written out explicitly as matrices, we can easily simplify the notation.

In compact matrix notation, the regression model then can be written as

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

5.7 Solving the Regression Equation

In practice, we would like to know the contents of (i.e., solve for) $\boldsymbol{\beta}$.

Assuming the model is correct, the expected value of $\boldsymbol{\epsilon}$ is 0, therefore,

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta}$$

Then we just need to solve for $\boldsymbol{\beta}$. We can think back about some of the matrix operations we discussed earlier.

5.7.1 Matrix Multiplication and Transpose

Our goal is to isolate $\boldsymbol{\beta}$. One initial idea might be to multiple each side of the equation by $\mathbf{X}^{-1}$ in an attempt to remove $\mathbf{X}$ from the right hand side, and isolate $\boldsymbol{\beta}$. Why won't this work?

Instead, let's pre-multiply each side of the equation by $\mathbf{X}'$. This would give us

$$\mathbf{X}'\mathbf{Y} = \mathbf{X}'\mathbf{X}\boldsymbol{\beta}$$

This gets us a quantity, $(\mathbf{X}'\mathbf{X})$, a square matrix containing information about the relations among the $\mathbf{x}$s.

5.7.2 Matrix Inverse

Now, since $\mathbf{X}'\mathbf{X}$ is a square matrix and presumabely has an inverse (e.g. no perfect collinearity), we can premultiply both sides by $(\mathbf{X}'\mathbf{X})^{-1}$, to obtain

$$(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{Y}) = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X})\boldsymbol{\beta}$$

Remembering our assumptions that a matrix multiplied by its inverse equals the identity matrix, $(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X}) = \mathbf{I}$ the equation simplifies to

$$(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{Y}) = \mathbf{I}\boldsymbol{\beta}$$

or more succinctly

$$(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{Y}) = \boldsymbol{\beta}$$

We've now isolated the unknowns, $\boldsymbol{\beta}$ onto one side of the equation and figured out how to use matrix algebra to obtain the regression coefficients. Quite literally, this algebra is what allows for estimation of the parameters when fitting a regression model to data.

We will now work through some practical examples - staying aware that this kind of matrix algebra is being done in the background.

5.8 The Linear Probability Model

While we are discussing the assumptions of OLS it is worth pausing to consider a model for dichotomous outcomes: the *linear probability model* (LPM).

In the LPM we don't do anything fancy with a binary outcome variable. Instead, we simply apply OLS as we would with a continuous out come variable. Since we aren't considering the normality of our outcome you might be curious how our assumptions would hold.

Remembering the assumptions of OLS:

1. $\mathbb{E}(\epsilon_i) = 0$
2. $\mathbb{E}(\epsilon_i^2) = \sigma^2$ for all i (homoscedasticity)
3. $\mathbb{E}(\epsilon_i \epsilon_j) = 0$ for all $i \neq j$
4. No perfect collinearity among x variables
5. $\mathbb{C}(\epsilon_i, x_{qi}) = 0$ for all i and k

Which assumptions are needed for consistency and asymptotic unbiasedness?

4. No perfect collinearity among x variables
5. Errors uncorrelated with all x variables.

In regard to (4) having a dependent variable valued at 0,1 does not cause any problems. In regard to (5), again no, nothing about a dichotomous outcome violates this assumption.

Therefore, in this model $\widehat{\boldsymbol{\beta}}$ is still consistent and asymptotically unbiased.

What about the remaining assumptions?

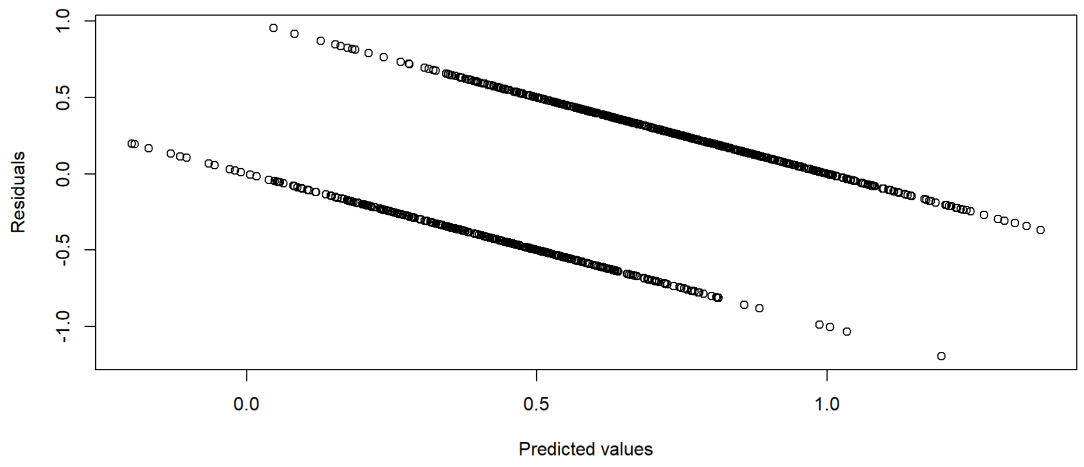
2. Homoscedasticity

Here, *a dichotomous outcome does inherently violate the assumption of homoskedasticity.*

Why is this case? It can be shown that $\mathbb{V}(\epsilon_i)$ now directly depends on the value of x that is taken. If the ith person has one set of values for the x variables, and another individual j has another set, the estimates of $\mathbb{V}(\epsilon_i)$ and $\mathbb{V}(\epsilon_j)$ will differ. This can be seem by looking at the variance of a Bernoulli random variable.

This means estimate of variance of $\widehat{\boldsymbol{\beta}}$ is no longer reliable, SEs and significance tests possibly inaccurate. However, this could be addressed using robust standard errors.

It is common to look at plots of predicted values vs residuals to diagnose heteroskedasticity. Generally one would like to see a random blob of points without any discernible pattern. Here is an example of what that plot might look like for an LPM model. Each line represents a different outcome, $y = 0$, or $y = 1$.



5.8.1 Advantages of the LPM

1. Simplicity.
2. Regression coefficients give impact of x on $P(y_i = 1|x)$.
3. Effect same regardless of value of x or values of other xs.
4. Can extend with traditional methods easily (interactions, quadratic terms).

5.8.2 Disadvantages of the LPM

1. Functional form unlikely accurate at extreme lows & highs
2. Binary dependent variable creates heteroscedasticity
3. Some regression diagnostics assume homoscedastic error
4. Predicted probabilities not restricted to 0 to 1 range

alright grandma let's get you to bed
We used to say that linear regression on binary outcomes was cool. We called it: linear probability model. People believed it.
KAPWING

Fox, John, and Sanford Weisberg. 2019. *An R Companion to Applied Regression*. Third. Thousand Oaks CA: Sage. https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

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.

Revelle, William. 2021. *Psych: Procedures for Psychological, Psychometric, and Personality Research*. Evanston, Illinois: Northwestern University. https://CRAN.R-project.org/package=psych.

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. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2021. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.