MSI data science manual

A review of analytical approaches followed by Management Systems International

2023-09-21

Table of contents

Preface

This manual introduces a variety of analytical approaches MSI adopts in order to learn, teach, and meet its client deliverables.

1 Introduction

We used to do data science in spreadsheets, while a more select realm of demi-gods used expensive statistical analysis packages or wrote code. Now we have access to open source software that puts immense computing power in the hands of the people. The easy accessibility of such power is both a blessing and a curse. This manual seeks to bestow the blessings while avoiding the curse.

This manual is already out of date. Tomorrow, we will be able to tell our AI assistants what we want, and the AI instance will provide it to us through some opaque combination of computation and creation. Aspiring analysts are still encouraged to learn this manual as a way to welcome our new overlords.

2 Summary

First and foremost, MSI is client driven. We provide what is asked for, following client guidance. Within this framework, we use a variety of analytical approaches and tools that follow best practice while satisfying the guidance we are under.

This manual provides an idealized approach of a data analysis. MSI is agnostic about the tools used to conduct an analysis, but the majority of explanation and examples are provided in the R programming language.

In unit one, we introduce R, explain how it works, and provide guidance as to how to get set up and start analyzing.

In unit two, we go through the steps of a data analysis.

In unit three, we review different ways to report your analyses.

Throughout the journey, we will provide examples of MSI analyses that were used for internal learning, or to generate a client deliverable.

Let's start!

Part I

Entering the R ecosystem

3 How R works

Most of us are familiar with Excel or used a software like SPSS, SAS, or STATA in school. Some of us use these regularly at work.

A programming environment, such as R, offers some cool stuff.

- R is open-source and free. It has a huge support community that is constantly de-bugging and creating new functionalities.
- If you find an analysis or cool example, the code is almost always included. The R community is all about sharing.
- R was developed specifically for statistical programming.
- If you can imagine an analytic task, you can implement it in R.
- Analyses in R are transparent, easily shareable, and reproducible. Can you remember every step you did to create a data visualization in Excel so that someone else could add to it?
- R integration with Github allows a team to work together.
- R can read and write in virtually any data format.
- R can be used for any data science task: scraping websites, developing websites, making static or interactive charts, automating repetitive tasks, statistical computations, querying databases, and many others.
- R has a lot of inter-operability with other platforms.

To realize these benefits, however, requires an understanding of how R works. This chapter will walk through some foundations of using R and its data structures

3.1 Basic use

In R, you create objects and then use those objects for your analysis. Objects are made up of whatever you assign them to. They can be a direct value, a file, a graphic, etc. Here's an example:

```
a <- 5
```

We have assigned the object, `a`, the value of 5. The assignment operator `<-` is what tells R to assign the value of 5 to `a`.

Now we can use the object `a`. As in `a + a.` We use the `#` to annotate our code for human readers. R will not compute any text to the right of a `#`. Annotating code is very helpful for code review and for remembering what you were doing when you open up a script that you have not worked on for 6 months.

```
# Assign a the value of 5
a <- 5

# Add a + a (or 5 +5)
a+a
```

```
[1] 10
```

Notice that R understands to output the value of `a+a` without any additional instructions. Or, you could store the value of `a + a` as a new object.

```
a <- 5

# Assign the value of a + a to b
b <- a + a

#print value of b
b
```

```
[1] 10
```

3.2 Data Structures

The primary data structures in R are vectors, matrices, lists, and data frames. They all basically begin as a vector. The idea here is not to master what the data structures are, but to understand how R handles each one as it will affect more advanced coding operations. Knowledge of data structures is also helpful when debugging code because error messages will reference the different data structures.

Naturally, we will start with the most "atomic" of the data structures, the (atomic) vector.

3.2.1 Vectors

A vector is the most basic data structure in R. A vector can only contain a single data type. It can be any of logical, integer, double, character, complex or raw, but it cannot mix and match types.

Here's a vector

```
# Create vectors
vector <- 10
vector1 <- c(10, 14, 27, 99)
vector2 <- c("purple", "blue", "red")

# Print the value of each vector
vector
```

```
[1] 10
```

```
vector1
```

```
[1] 10 14 27 99
```

```
vector2
```

```
[1] "purple" "blue"   "red"
```

3.3 Matrices

A matrix is a vector with dimensions - it has rows and columns. As with a vector, the elements of a matrix must be of the same data type. Here are a few examples.

```
# Create a 2 x 2 matrix with the numbers 1 through 4
m <- matrix(1:4, nrow = 2, ncol = 2)

# Note that the matrix is filled column-wise. (e.g., it completes # the left column with 1
# and entering 3 and 4
m
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

3.4 Lists

A list is a vector that can have multiple data types. You can call `class()` on any object in R to display the type of object that it is.

```r
# Make a list a
a <- list(10, "red", 74, "blue")

# What is the class, or type, of a?
class(a)
```

```
[1] "list"
```

3.5 Dataframes

You can think of a dataframe as your Excel Spreadheet. At MSI, this is the most common form of dataset. We read a .xlsx or .csv file into R, and we get a dataframe. At its core, a dataframe is a list of lists where each list (column) is the same length (i.e., it is a "rectangular list"). A data frame can contain many types of data because it is a collection of lists, and lists, as you remember, can consist of multiple data types.

```r
# Create a dataframe called df

df <- data.frame(a = c(10,20,30,40)
                 , b = c('book', 'pen', 'textbook', 'pencil_case')
                 , c = c(TRUE,FALSE,TRUE,FALSE)
                 , d = c(TRUE,FALSE,TRUE,FALSE))

# Print df
df
```

```
   a           b     c     d
1 10        book  TRUE  TRUE
2 20         pen FALSE FALSE
3 30    textbook  TRUE  TRUE
```

```
4 40 pencil_case FALSE FALSE
```

Now that we have a dataframe, we want to look at some of its details using `glimpse()`.

```r
# Create a dataframe called df

df <- data.frame(a = c(10,20,30,40)
                 , b = c('book', 'pen', 'textbook', 'pencil_case')
                 , c = c(TRUE,FALSE,TRUE,FALSE)
                 , d = c(TRUE,FALSE,TRUE,FALSE))

# Look at structure of df
dplyr::glimpse(df)
```

```
Rows: 4
Columns: 4
$ a <dbl> 10, 20, 30, 40
$ b <chr> "book", "pen", "textbook", "pencil_case"
$ c <lgl> TRUE, FALSE, TRUE, FALSE
$ d <lgl> TRUE, FALSE, TRUE, FALSE
```

4 How R programs

5 Happy Git with R

6 R and Data Science resources

Part II

Data Analysis Workflow

7 Designing an analytical activity

8 Sampling

9 Data cleaning and preparation

10 Reproducibility

11 Data exploration

12 Introduction

In a well-defined study, exploratory analysis may be largely unnecessary. Consider a scenario where the client has identified questions / hypotheses of interest and the additional variables that may predict or mediate the identified outcomes. An activity partner such as MSI was provided the time and resources to consult with stakeholders, scope out the target environment to identify and map the data generating process, and develop an inferential design by which the collected data and analytical routines would address the questions posed by the client. In this scenario, exploratory analysis may be entirely eliminated, and the time that may be spent on exploration is shifted to sensitivity analysis of the pre-identified analytical routines.

On the other hand, consider a scenario where data has been collected for one purpose, and later realized to have possible use with other, not yet identified, purposes. In this case, the initial analysis is entirely exploratory.

13 Data visualization

13.1 Data visualization tactics

13.1.1 geomtextpath

There are common situations where MSI is tasked with ongoing data collection and evaluation of a client activity. For example, the MENA MELS activity (2020-2024) was tasked with ongoing monitoring and evaluation of the Middle East Partnership for Peace Activity (MEPPA). MEPPA comprised grants to several local partners organized around the common motif of building bonds between different demographic groups. The primary outcome of interest was whether or not a participant in the grant activity reported a perceived increase in understanding the political, social, and economic situation and viewpoints of another group. Data were collected on a rolling basis across several implementing partners, across baseline and endline. That data are summarized as follows:

```
df1 <- read_csv("data/short demo series/meppa response items.csv",
                show_col_types=F)
df1 %>%
  flextable() %>%
  autofit()
```

item	responselab	endline	n	perc
Political views	1Basic understanding	0	22	0.286
Political views	2Fair understanding	0	38	0.494
Political views	3High understanding	0	17	0.221
Political views	1Basic understanding	1	18	0.173
Political views	2Fair understanding	1	50	0.481
Political views	3High understanding	1	36	0.346
Social views	1Basic understanding	0	30	0.390
Social views	2Fair understanding	0	25	0.325

item	responselab	endline	n	perc
Social views	3High understanding	0	22	0.286
Social views	1Basic understanding	1	15	0.147
Social views	2Fair understanding	1	52	0.510
Social views	3High understanding	1	35	0.343
Economic views	1Basic understanding	0	32	0.416
Economic views	2Fair understanding	0	30	0.390
Economic views	3High understanding	0	15	0.195
Economic views	1Basic understanding	1	20	0.196
Economic views	2Fair understanding	1	55	0.539
Economic views	3High understanding	1	27	0.265

For purposes of client reporting, there are three ordinal responses across baseline and endline, for each of three types of viewpoint. There is insufficient data to conduct inferential tests at this level of granularity, but this data may be visualized descriptively.

The geomtextpath package offers the functionality to directly label line-based plots with text that is able to follow a curved path. Simply replace 'geom_line' with 'geom_textpath' and assign the variable to use as the label. The following figures illustrate.

```
pol <- ggplot(filter(df1, item=="Political views"), aes(endline, perc, color=as.factor(res
  geom_textpath(aes(label=lab),
                size=4) +
  geom_label(aes(label=paste(round(perc*100,0), "%", sep="")),
             size=4,
             label.padding = unit(.14, "lines")) +
  scale_color_viridis_d(option="D") +
  scale_x_continuous(limits=c(-.1, 1.1),
                     breaks=c(0,1),
                     labels=c("Baseline","Endline")) +
  scale_y_continuous(labels=percent_format(accuracy=1)) +
  theme(legend.position="none",
        axis.text.y=element_blank()) +
  labs(x="",
       y="",
       title="Political situation")
```

```
pol
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

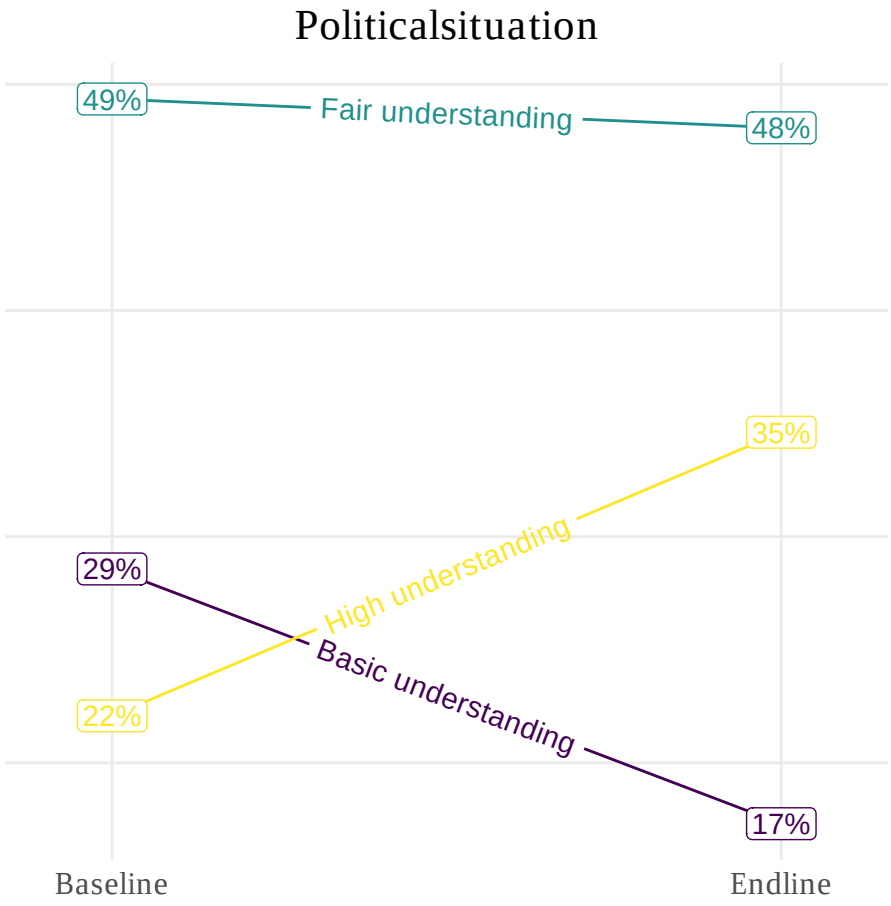
```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

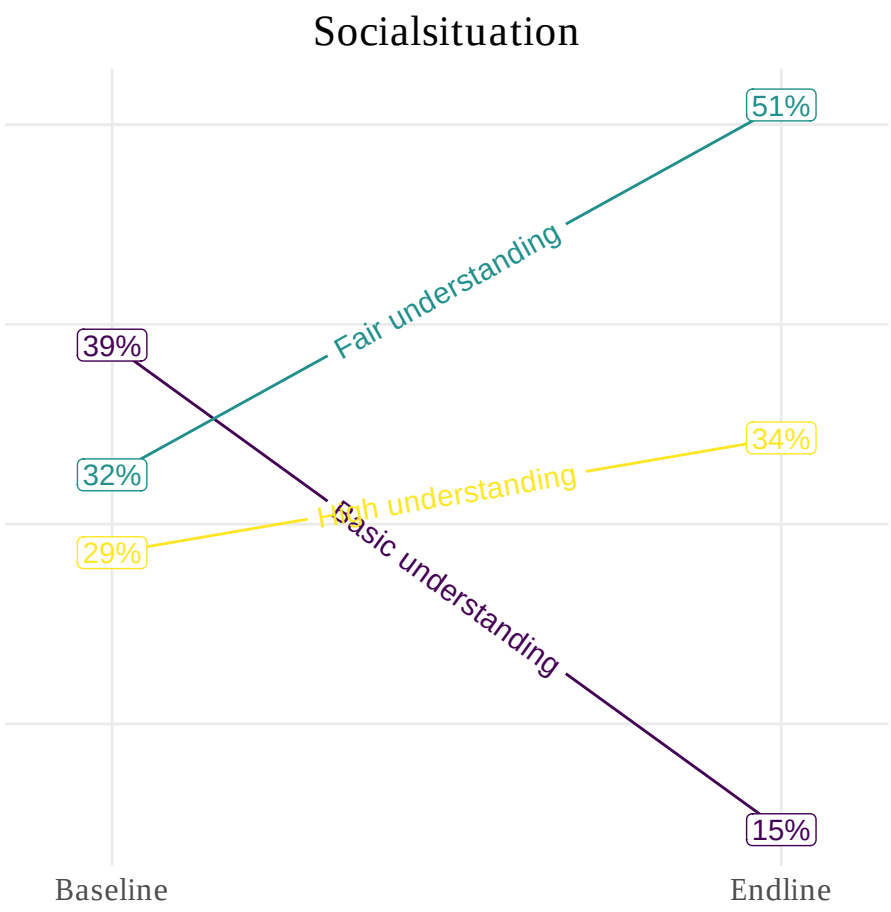
```
Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```




```
soc <- ggplot(filter(df1, item=="Social views"), aes(endline, perc, color=as.factor(respon
  geom_textpath(aes(label=lab),
                size=4) +
  geom_label(aes(label=paste(round(perc*100,0), "%", sep="")),
             size=4,
             label.padding = unit(.14, "lines")) +
  scale_color_viridis_d(option="D") +
  scale_x_continuous(limits=c(-.1, 1.1),
                     breaks=c(0,1),
                     labels=c("Baseline","Endline")) +
  scale_y_continuous(labels=percent_format(accuracy=1)) +
  theme(legend.position="none",
        axis.text.y=element_blank()) +
  labs(x="",
```

```
        y="",
        title="Social situation")

soc
```

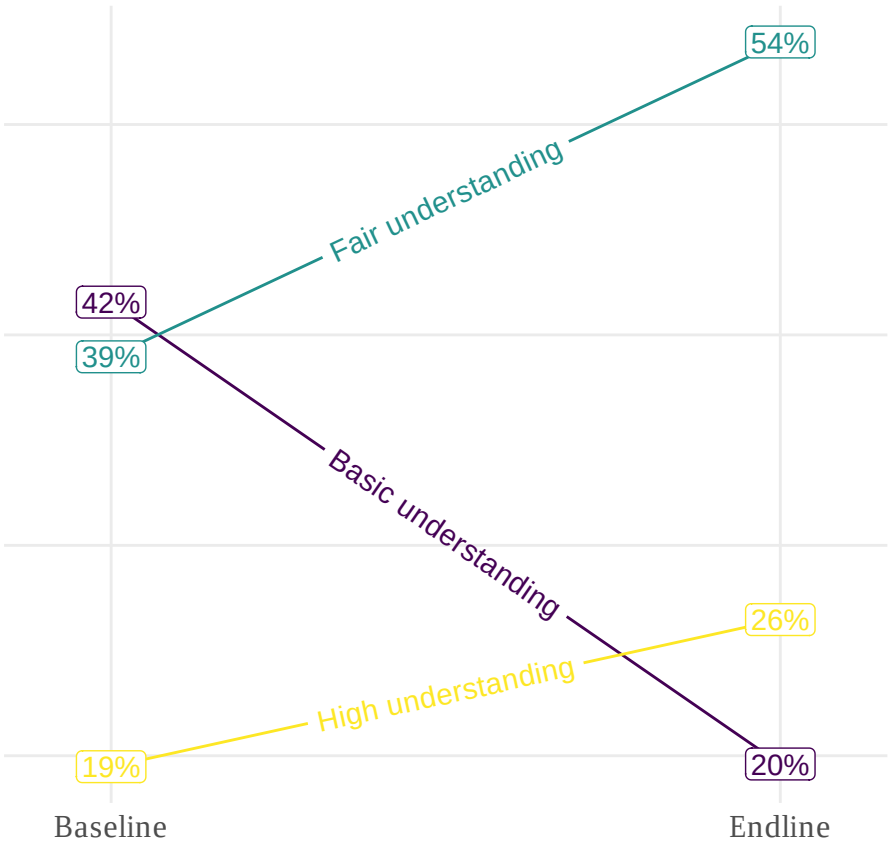
```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```




```
ec <- ggplot(filter(df1, item=="Economic views"), aes(endline, perc, color=as.factor(respo
  geom_textpath(aes(label=lab),
                size=4) +
  geom_label(aes(label=paste(round(perc*100,0), "%", sep="")),
             size=4,
             label.padding = unit(.14, "lines")) +
  scale_color_viridis_d(option="D") +
  scale_x_continuous(limits=c(-.1, 1.1),
                     breaks=c(0,1),
                     labels=c("Baseline","Endline")) +
  scale_y_continuous(labels=percent_format(accuracy=1)) +
  theme(legend.position="none",
        axis.text.y=element_blank()) +
  labs(x="",
```

```
       y="",
       title="Economic situation")

ec
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```



Given that the three types of understanding of others' situation are highly correlated, it makes sense to present these measures compactly as aspects of a deeper underlying construct. The patchwork library allows multiple ggplots to be assembled together as a single plot. The following figure illustrates.

```
pol + soc + ec +
  plot_annotation(title="How well do you understand the situation of others?")
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
```

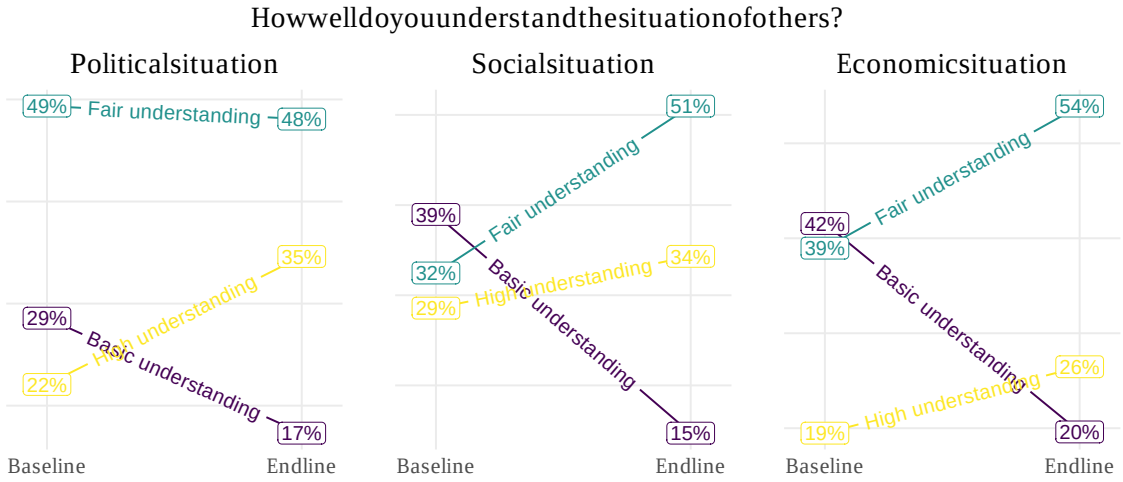
```
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
```

```
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```



A final use of presenting the data more compactly is to collapse the ordinal responses to binary, and collect the three measures as lines in a single plot. The following data captures each type of understanding as either fair or high understanding as one category, and basic understanding as the other category.

```
dat <- read_csv("data/short demo series/meppa item ladder.csv",
                show_col_types=F)

dat_flx <- dat %>%
  flextable() %>%
  autofit()

dat_flx
```

endline	n	percitem
0	55	0.714Political situation
1	86	0.827Political situation
0	47	0.610Social situation
1	87	0.853Social situation
0	45	0.584Economic situation
1	82	0.804Economic situation

With this simplified data summary, the trendline for each type of understanding may now be

collected in a single plot.

```r
ggplot(dat, aes(endline, perc, color=as.factor(item))) +
  geom_textpath(aes(label=item),
                size=4) +
  geom_label(aes(label=paste(round(perc*100,0), "%", sep="")),
             size=4,
             label.padding = unit(.14, "lines")) +
  scale_color_viridis_d(option="D") +
  scale_x_continuous(limits=c(-.1, 1.1),
                     breaks=c(0,1),
                     labels=c("Baseline","Endline")) +
  scale_y_continuous(labels=percent_format(accuracy=1),
                     breaks=c(.5,1),
                     sec.axis=dup_axis(breaks=c(.804,.827,.853),
                                       labels=c("+22","+12","+24"))) +
  theme(legend.position="none",
        axis.text.y.left=element_blank()) +
  labs(x="",
       y="",
       caption="Proportion reporting fair or high\nunderstanding of others' situation")
```

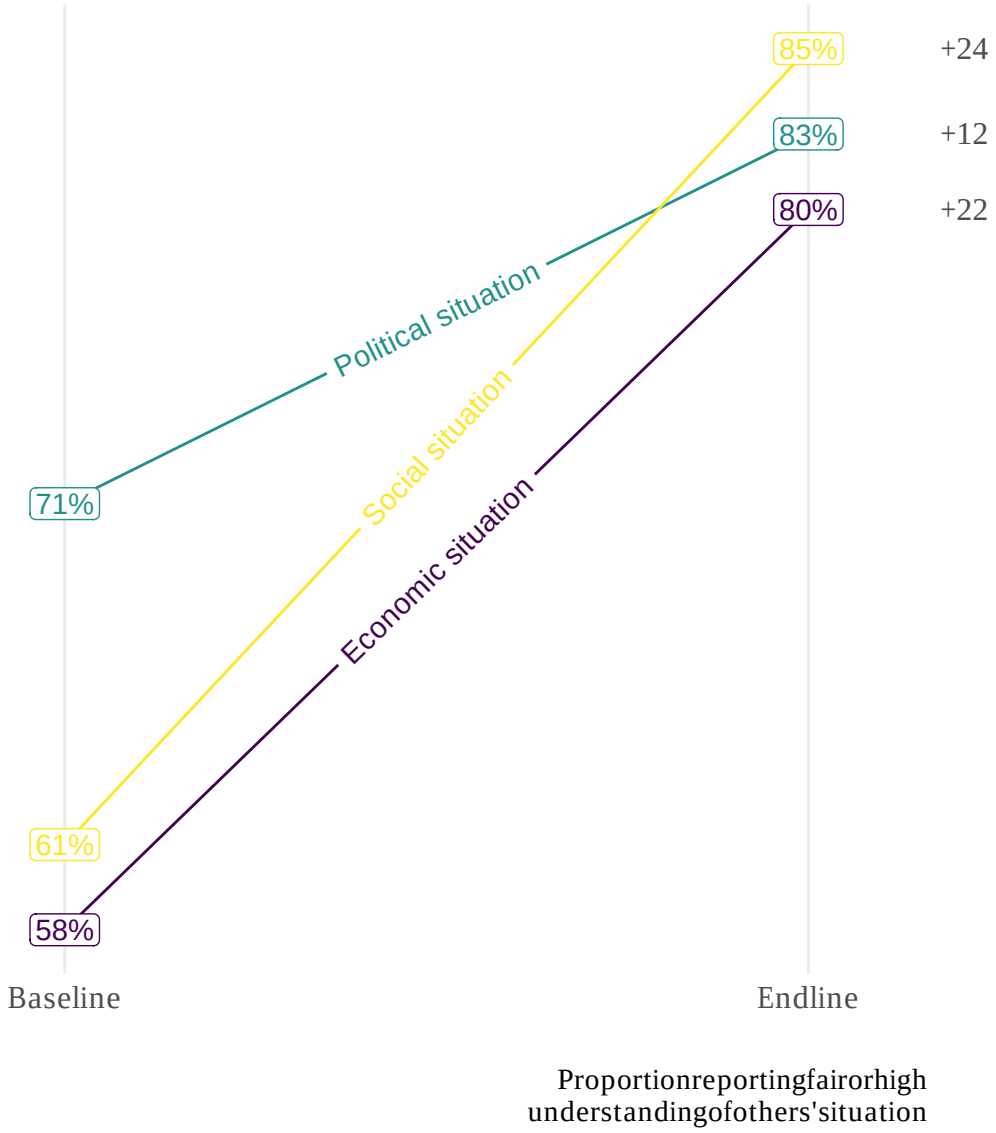
```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
```

```
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
```

```
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x20
```



Note further the use of secondary axis breaks to illustrate the change score for each trendline from baseline to endline.

The R computing language allows for several ways to customize the use of labels in statistical or descriptive graphics. This short demo has illustrated MSI's use of the geomtextpath package to place labels directly along the line or curve of a plot. This illustration used only straight lines between two points in time. For additional use cases of the geomtextpath package, see the package vignette.

13.1.2 Mapping

Much of our data is collected from surveys and has geographic coordinates associated with a point of collection, a house, or a city, etc. It can be useful to generate a map to get a sense of where the data is coming from. Mapping is a part of MSI's exploratory data analysis process and is also used in developing a sampling plan and in producing high quality data vizualizations for clients. This section provides a brief introduction into mapping in R.

The most commonly used packages to handle spatial data are `sf` for vectors, `terra` for vectors and rasters, and `raster` for rasters. To visualize the data, two frequently used packages are tmap and ggplot2 packages.

To get started, we need to load our packages.

```
#run this line first if you have never used these packages before
#install.packages(c("tidyverse", "sf", "tmap", "readr", "here"))

library(tidyverse) #install the core tidyverse packages including ggplot2
library(sf) #provides tools to work with vector data
library(tmap) #for visualizing spatial data
library(readr) #functions for reading external datasets
library(here) #to better locate files not in working directory
library(geodata) #to download administrative boundaries
```

13.2 Read in the data

```
#It is a csv file so I use the read_csv function and provide the file path
cities <- read_csv(here::here("../methods corner/Map demo/data/Madagascar_Cities.csv")
                   , show_col_types = FALSE)

#Observe the first few rows of data
DT::datatable(head(cities))
```

```
Google Chrome was not found. Try setting the `CHROMOTE_CHROME` environment variable to the ex
```

Name Population Region Province Captial Latitude Longitude Number of Activities

Now, for the administrative boundaries. Each of these are being read in using `gadm()` from the `geodata` package and then converted to an sf object with the `st_as_sf()` function. In the example, we download only the country boundary, but if we wanted regions or departments, we would simply change the level argument inside the `gadm()` function call to a 1 or a 2.

13.3 Country boundary

```
#This is only the country boundary.
mdg <- geodata::gadm(country = "MDG"
                     , level = 0
                     , path = tempdir()) |>
  st_as_sf()
```

13.4 Convert the cities to an sf object

Remember that the cities object is a standard .csv with longitude and latitude columns, but it is not yet recognized as an sf object that has geographic properties. Here is how to convert

it to an sf object with a single geometry column and a crs.

```
cities_sf <- cities |>
  st_as_sf(coords = c("Longitude", "Latitude")
           , crs = 4326)

#observe the first few rows of data
DT::datatable(head(cities_sf))
```

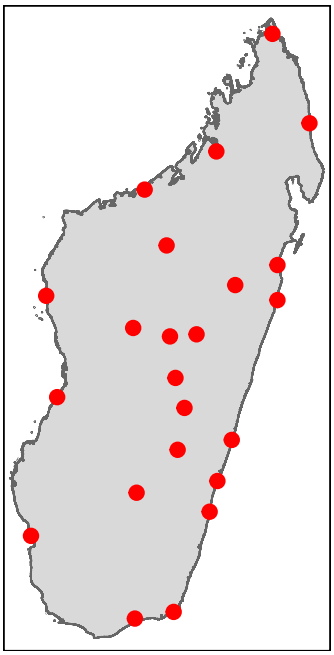
Name Population Region Province Captial Number of Activities geometry

13.5 Make the map

The following code chunks and tabs walk through the process of making and improving a map in both tmap and ggplot2. In the example, cities are what we are plotting, but we could be plotting any variable of a dataset.

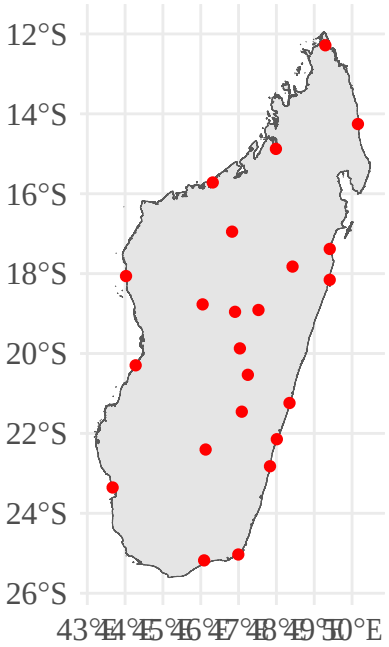
13.6 tmap

```
tmap_mode("plot") +
  tm_shape(mdg) +
  tm_polygons() + #for only the borders, use tm_borders()
  tm_shape(cities_sf) +
  tm_dots(size = .25, col = "red")
```



13.7 ggplot2

```
ggplot2::ggplot(mdg) +
  geom_sf() +
  geom_sf(data = cities_sf, color = "red")
```

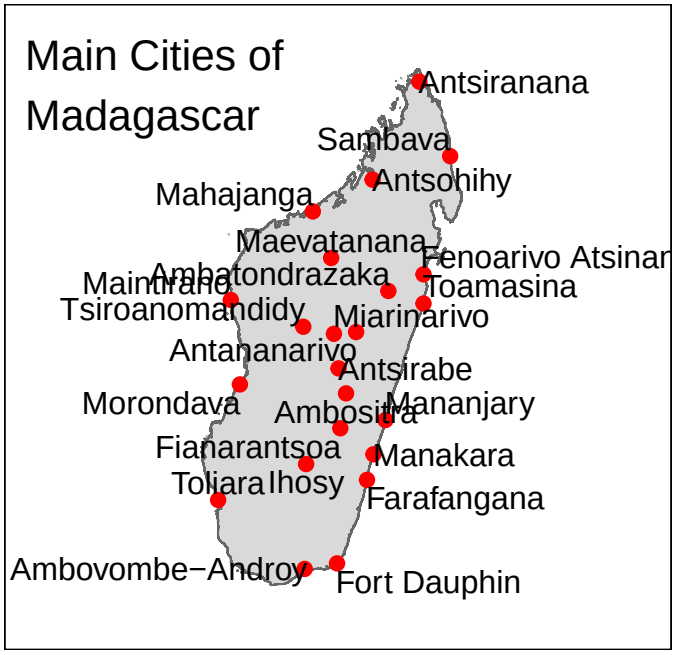


13.8 Make the map better

13.9 tmap

```
#the city names are long so we have to
# make a bigger window to fit them. This isn't part of the normal process
#make an object with the current bounding box
bbox_new <- st_bbox(mdg)

#calculate the x and y ranges of the bbox
xrange <- bbox_new$xmax - bbox_new$xmin # range of x values
yrange <- bbox_new$ymax - bbox_new$ymin # range of y values

#provide the new values for the 4 corners of the bbox
  bbox_new[1] <- bbox_new[1] - (0.7 * xrange) # xmin - left
  bbox_new[3] <- bbox_new[3] + (0.75 * xrange) # xmax - right
  bbox_new[2] <- bbox_new[2] - (0.1 * yrange) # ymin - bottom
  bbox_new[4] <- bbox_new[4] + (0.1 * yrange) # ymax - top

#convert the bbox to a sf collection (sfc)
```

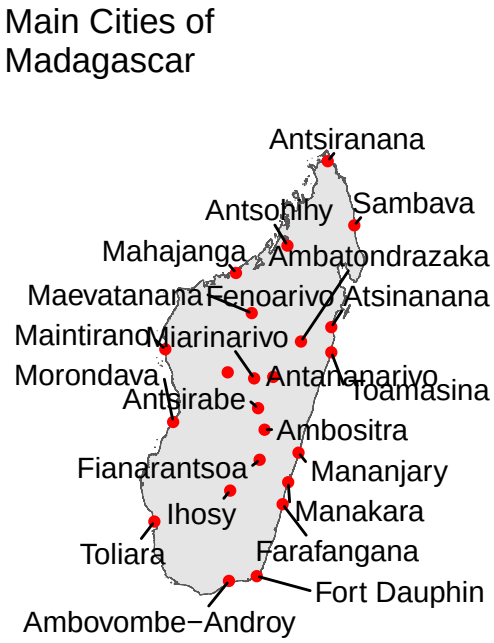
```r
bbox_new <- bbox_new |>  # take the bounding box ...
  st_as_sfc() # ... and make it a sf polygon

#now plot the map
tmap_mode("plot") +
  tm_shape(mdg, bbox = bbox_new) +
  tm_polygons() +
  tm_shape(cities_sf) +
  tm_dots(size = .25, col = "red") +
  tm_text(text = "Name", auto.placement = T) +
  tm_layout(title = "Main Cities of\nMadagascar")
```



13.10 ggplot2

```r
#the city names are long so we have to
# make a bigger window to fit them. This isn't part of the normal process
#make an object with the current bounding box
bbox_new <- st_bbox(mdg)
```

```r
#calculate the x and y ranges of the bbox
xrange <- bbox_new$xmax - bbox_new$xmin # range of x values
yrange <- bbox_new$ymax - bbox_new$ymin # range of y values

#provide the new values for the 4 corners of the bbox
  bbox_new[1] <- bbox_new[1] - (0.5 * xrange) # xmin - left
  bbox_new[3] <- bbox_new[3] + (0.5 * xrange) # xmax - right
  bbox_new[2] <- bbox_new[2] - (0.1 * yrange) # ymin - bottom
  bbox_new[4] <- bbox_new[4] + (0.1 * yrange) # ymax - top

#convert the bbox to a sf collection (sfc)
bbox_new <- bbox_new |>  # take the bounding box
  st_as_sfc() # ... and make it a sf polygon


ggplot2::ggplot() +
  geom_sf(data = mdg) +
  geom_sf(data = cities_sf, color = "red") +
  ggrepel::geom_text_repel(data = cities_sf
                , aes(label = Name
                      , geometry = geometry)
                , stat = "sf_coordinates"
                , min.segment.length = 0) +
  coord_sf(xlim = st_coordinates(bbox_new)[c(1,2),1], # min & max of x values
           ylim = st_coordinates(bbox_new)[c(2,3),2]) + # min & max of y values +
  labs(title = "Main Cities of\nMadagascar") +
  theme_void()
```



13.11 Final touches

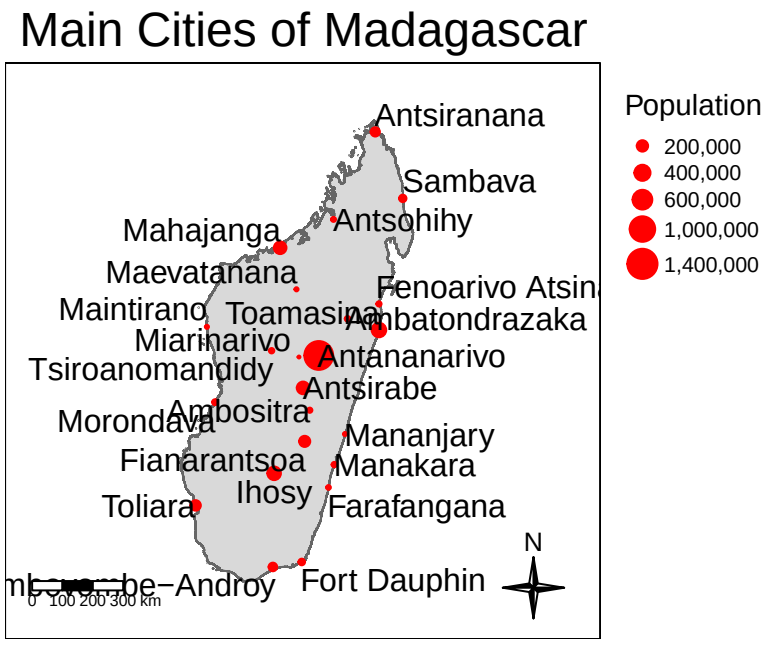
Now that we have a map with cities plotted (we achieved our goal!), we will add a few finishing touches and set the size of the city points to the `population` variable in the original dataset.

Additionally, tmap provides a simple interface to go from a static map to an interative map simply by changing `tmap_mode("plot")` to `tmap_mode("view")`.

13.12 tmap

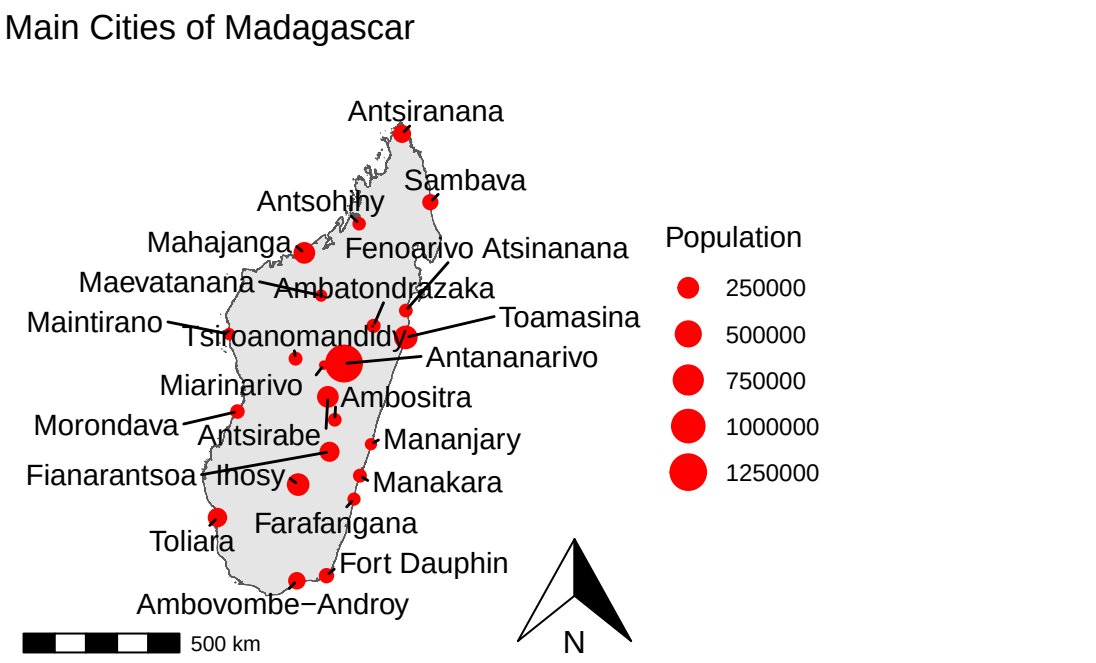
```
#the city names are long so we have to
# make a bigger window to fit them. This isn't part of the normal process
#make an object with the current bounding box
bbox_new <- st_bbox(mdg)

#calculate the x and y ranges of the bbox
xrange <- bbox_new$xmax - bbox_new$xmin # range of x values
yrange <- bbox_new$ymax - bbox_new$ymin # range of y values
```

```r
#provide the new values for the 4 corners of the bbox
  bbox_new[1] <- bbox_new[1] - (0.7 * xrange) # xmin - left
  bbox_new[3] <- bbox_new[3] + (0.75 * xrange) # xmax - right
  bbox_new[2] <- bbox_new[2] - (0.1 * yrange) # ymin - bottom
  bbox_new[4] <- bbox_new[4] + (0.1 * yrange) # ymax - top

#convert the bbox to a sf collection (sfc)
bbox_new <- bbox_new |>  # take the bounding box ...
  st_as_sfc() # ... and make it a sf polygon
tmap_mode("plot") +
  tm_shape(mdg, bbox = bbox_new) +
  tm_polygons() +
  tm_shape(cities_sf) +
  tm_dots(size = "Population", col = "red"
          , legend.size.is.portrait = TRUE) +
  tm_text(text = "Name", auto.placement = T
          , along.lines = T) +
  tm_scale_bar(position = c("left", "bottom"), width = 0.15) +
  tm_compass(type = "4star"
             , position = c("right", "bottom")
             , size = 2) +
  tm_layout(main.title = "Main Cities of Madagascar"                                    , legend.outs
```



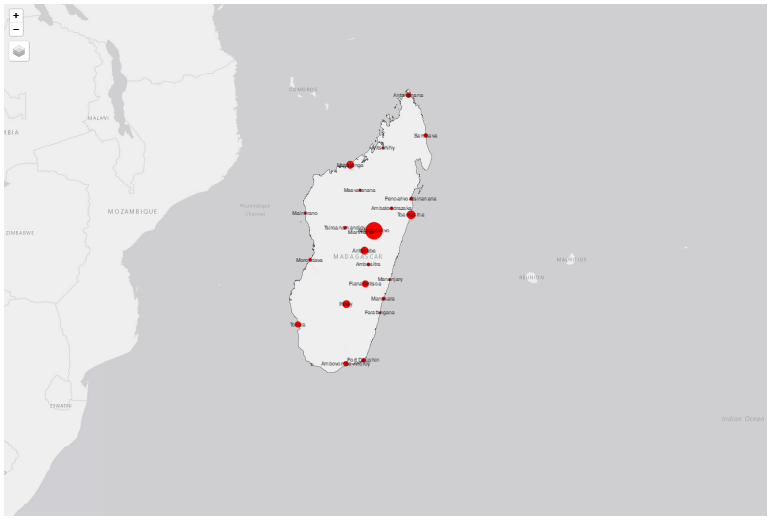
13.13 ggplot2

```
ggplot2::ggplot() +
  geom_sf(data = mdg) +
  geom_sf(data = cities_sf, aes(size = Population)
          , color = "red") +
  ggrepel::geom_text_repel(data = cities_sf
                , aes(label = Name
                      , geometry = geometry)
                , stat = "sf_coordinates"
                , min.segment.length = 0) +
  coord_sf(xlim = st_coordinates(bbox_new)[c(1,2),1], # min & max of x values
           ylim = st_coordinates(bbox_new)[c(2,3),2]) + # min & max of y values +
  ggspatial::annotation_scale(location = "bl") +
  ggspatial::annotation_north_arrow(location = "br"
                                    , which_north = "true"
                                    , size = 1)+
  labs(title = "Main Cities of Madagascar") +
  theme_void()
```



13.14 tmap interactive

```
tmap_mode("view") +
  tm_shape(mdg) +
  tm_borders() +
  tm_shape(cities_sf) +
  tm_dots(size = "Population", col = "red"
          , legend.size.is.portrait = TRUE) +
  tm_text(text = "Name", auto.placement = T
          , along.lines = T) +
  tm_scale_bar(position = c("left", "bottom"), width = 0.15) +
  tm_compass(type = "4star"
             , position = c("right", "bottom")
             , size = 2) +
  tm_layout(main.title = "Main Cities of Madagascar"                                , legend.outs
```



13.15 Additional Resources

For those interested in mapping in R (or QGIS) there are many free resources available online. A great starting point for R is the online text book, Geocomputation with R. If you would rather learn more in Python, Geocomputation with Python is a great resource.

14 Data modeling

15 Generating client deliverables

Part III

Reporting

16 Reports

17 Presentations

18 Dashboards

Part IV

Annexes

References