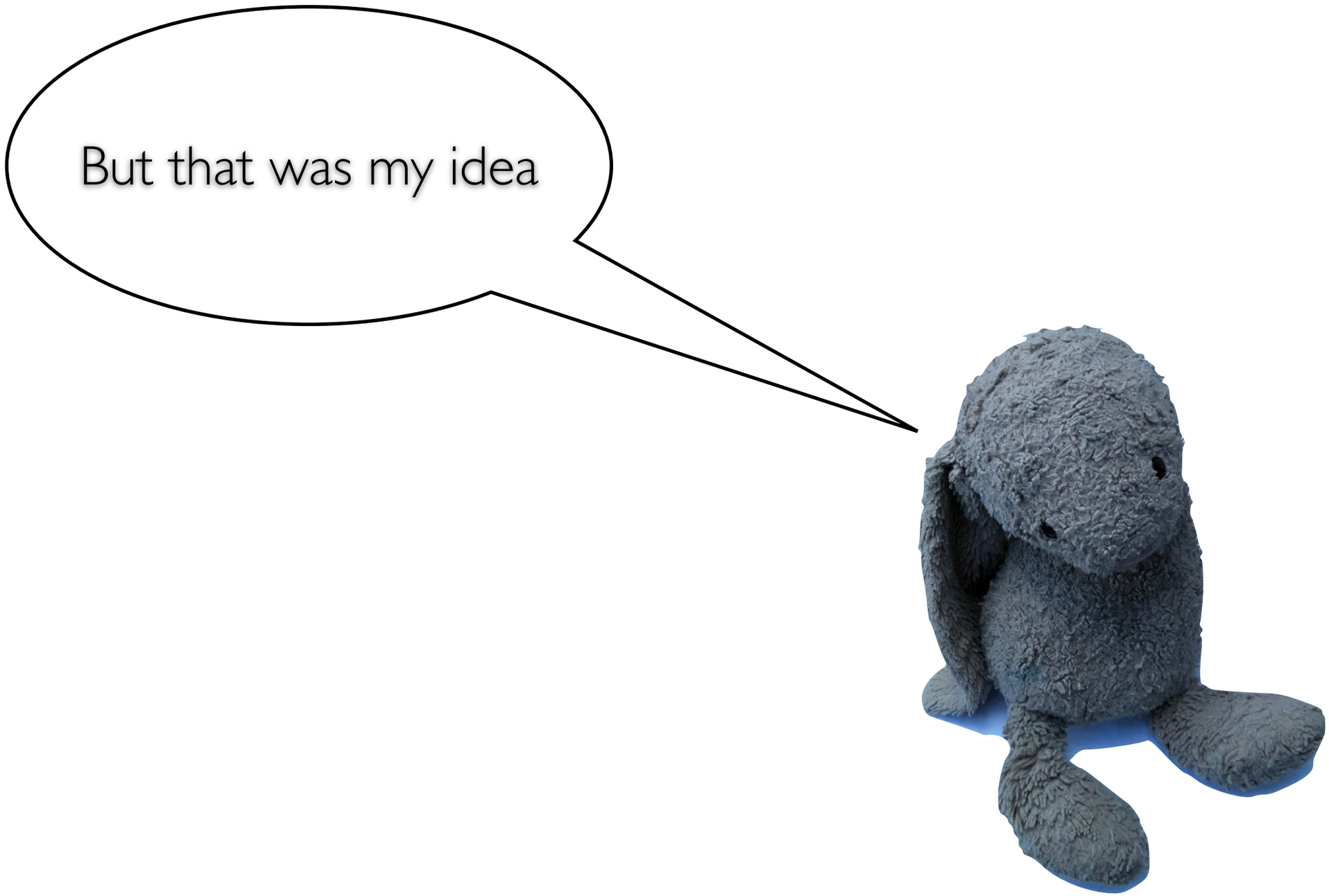
Basics of R: Markdown

Research Methods for Human Inquiry

Andrew Perfors

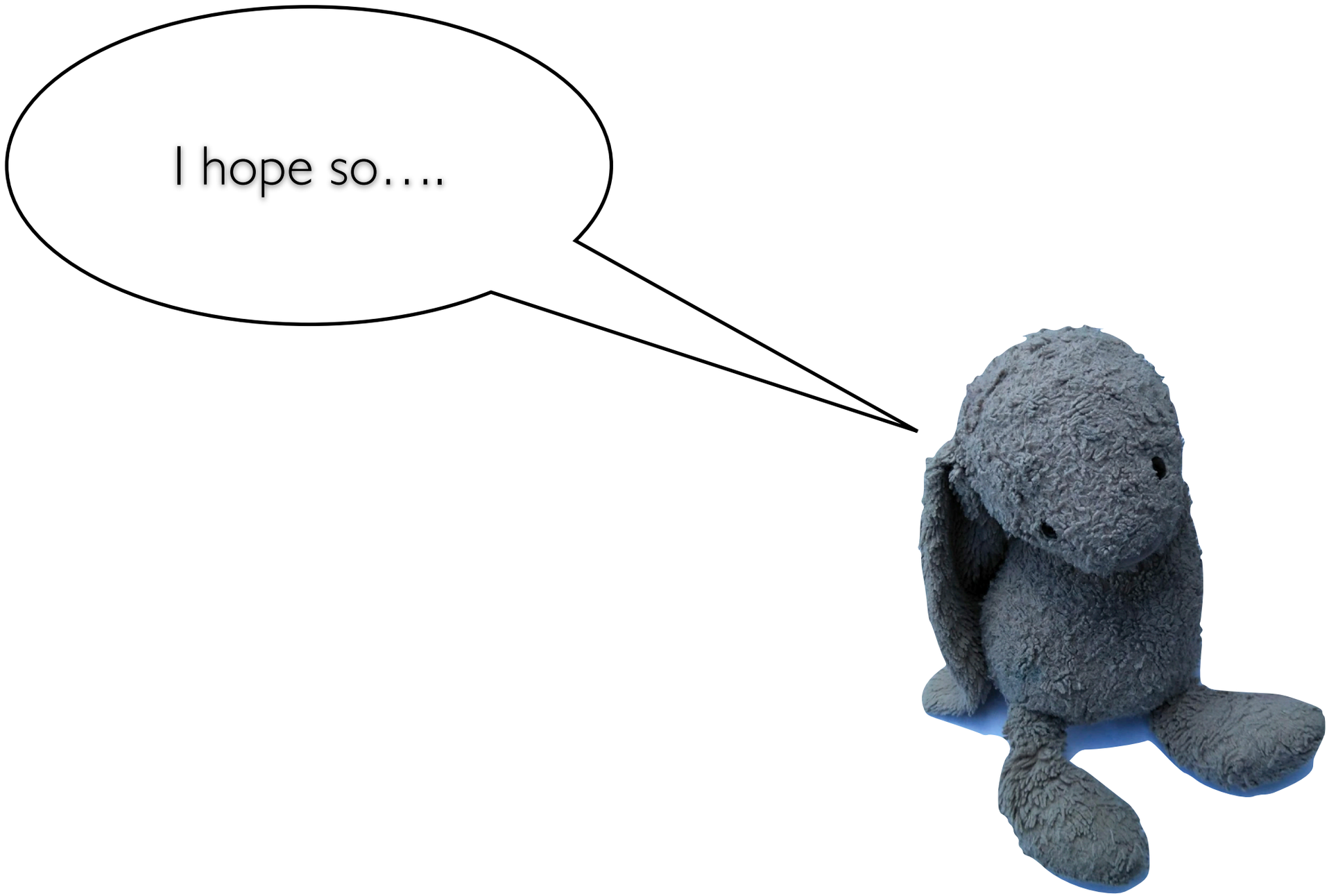
Today's story…

Today's story…



Today's story…

Today's story…



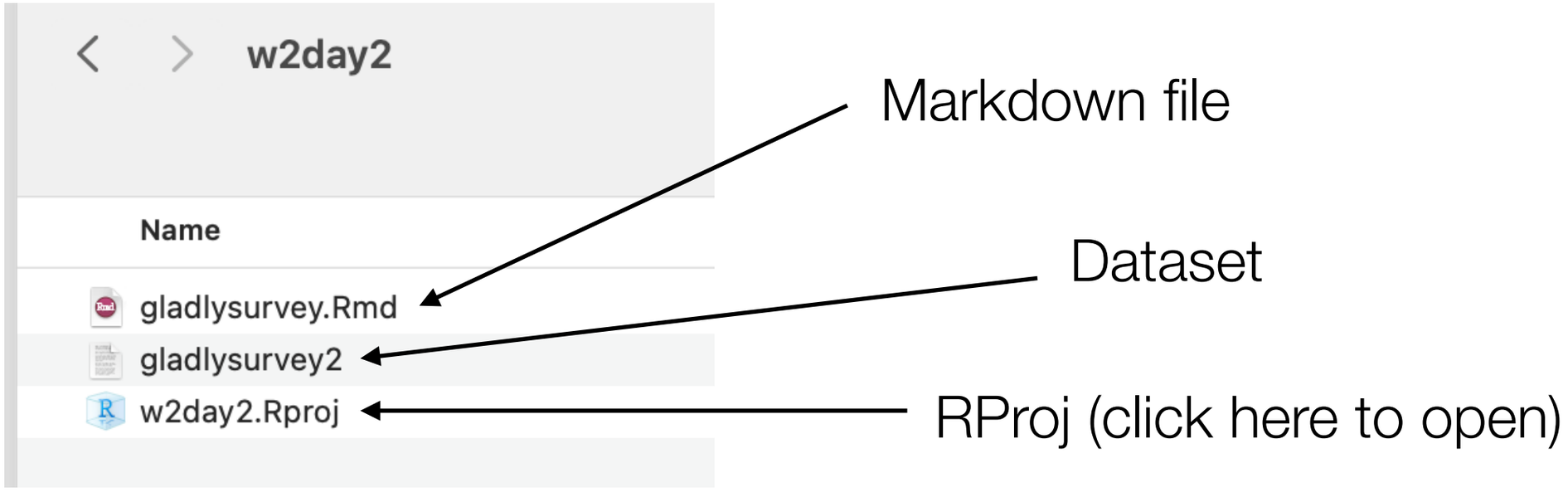
Art by @alison_horst

R Markdown

R Markdown lets you save and easily re-run your R commands, while also combining them with text. From now on we'll be doing all of our exercises, as well as the assignment, using R Markdown.

First, install and load: `install.packages("rmarkdown")`
`library(rmarkdown)`

Download the zipped file called **`w2day2`** from Canvas (it's the link to the exercises for today). Put it in your **`rmhi/exercises/week2`** folder and unzip.



Gladly's survey

Those of you who already have had the tutes know that Gladly did a survey as well.

1. What year were you born?

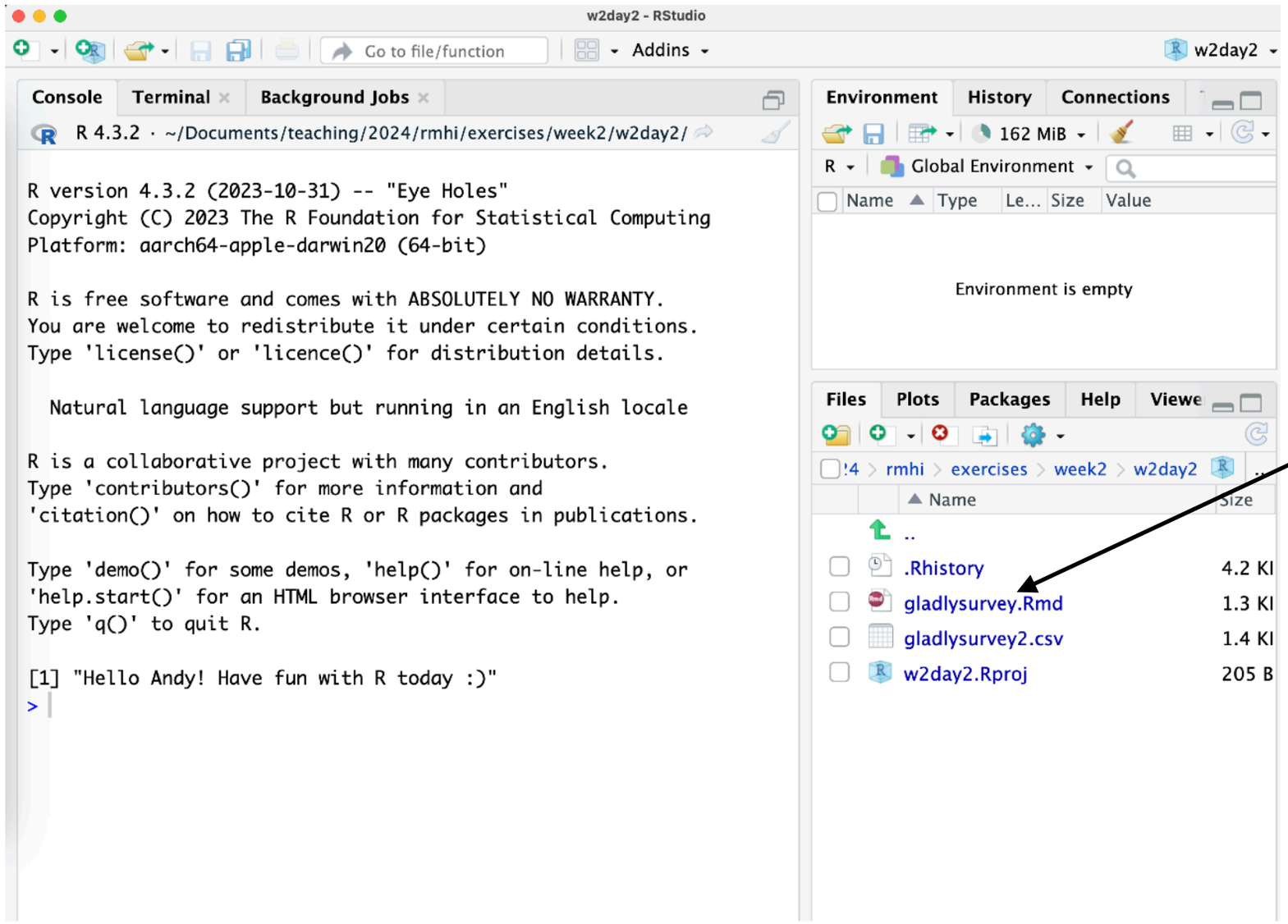
2. What is your favourite food?

3. On a scale of 1-10, how much do you like carrots?

4. On a scale of 1-10, how much do you like cake?

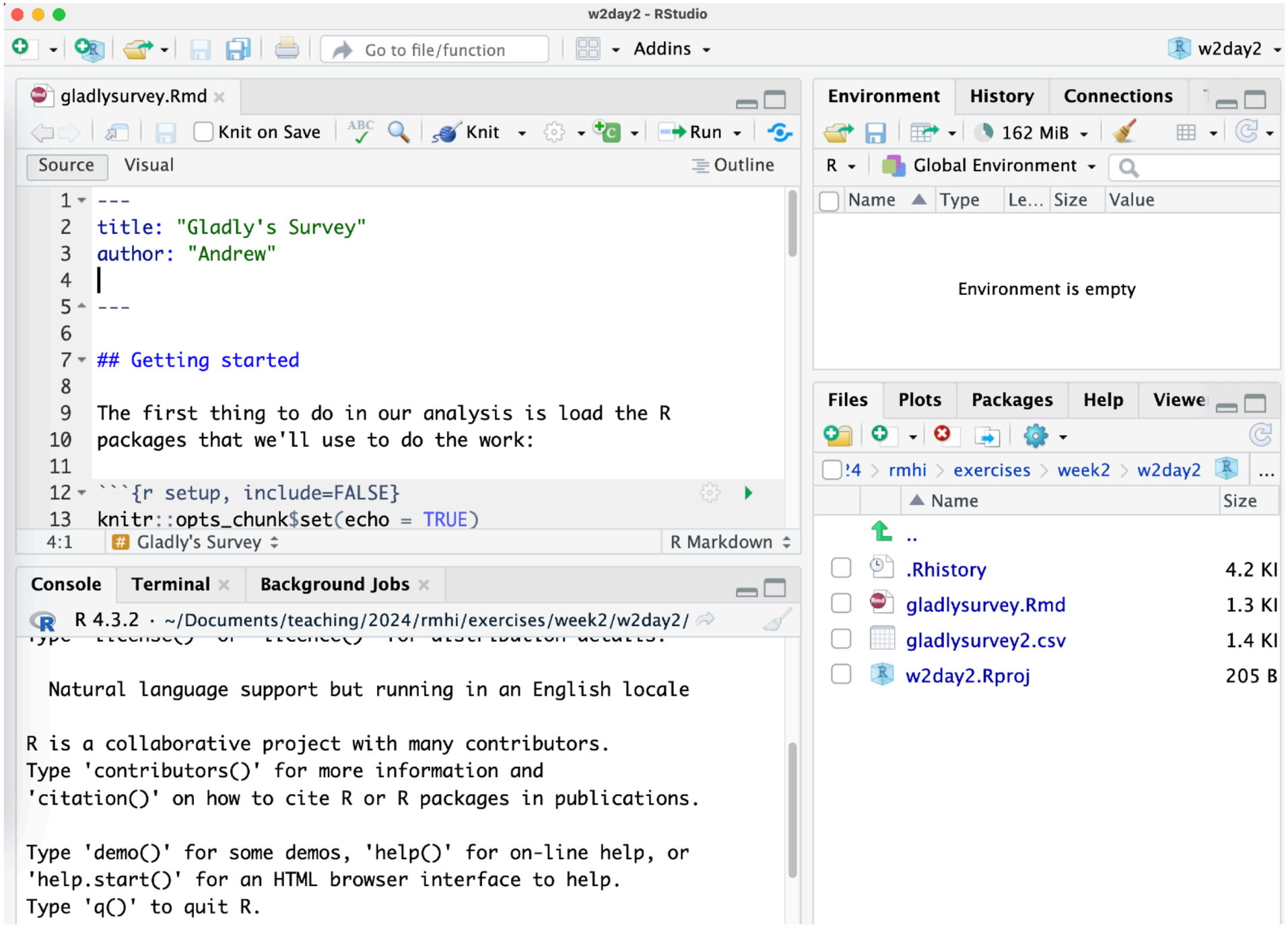
Gladly's survey

He since added a bit to it. Let's open the R Markdown document associated with it, called **`gladlysurvey.Rmd`**. Open it from RStudio, by clicking on it from the panel.



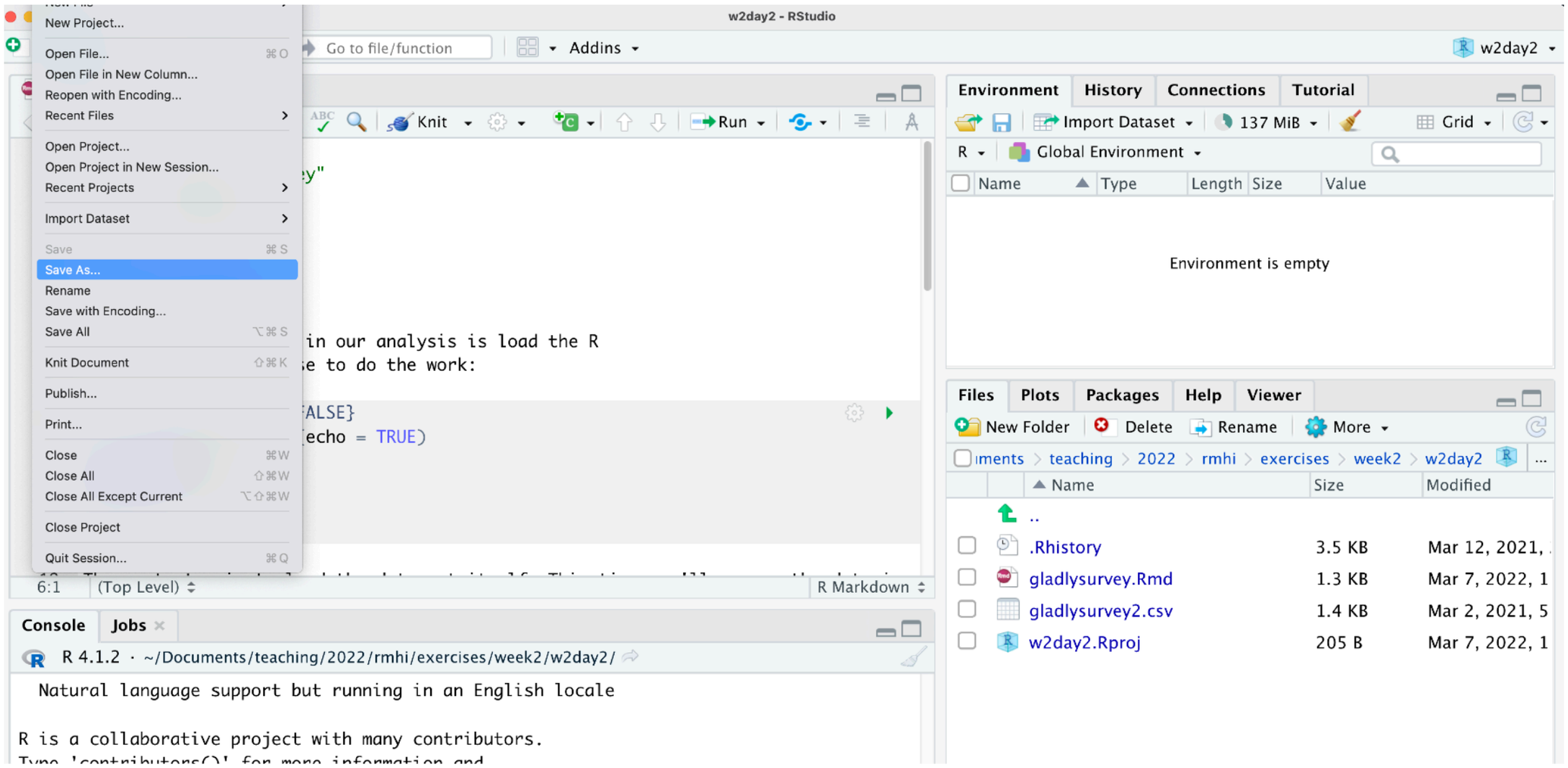
R Markdown

You should see something that looks like this show up in the top left:



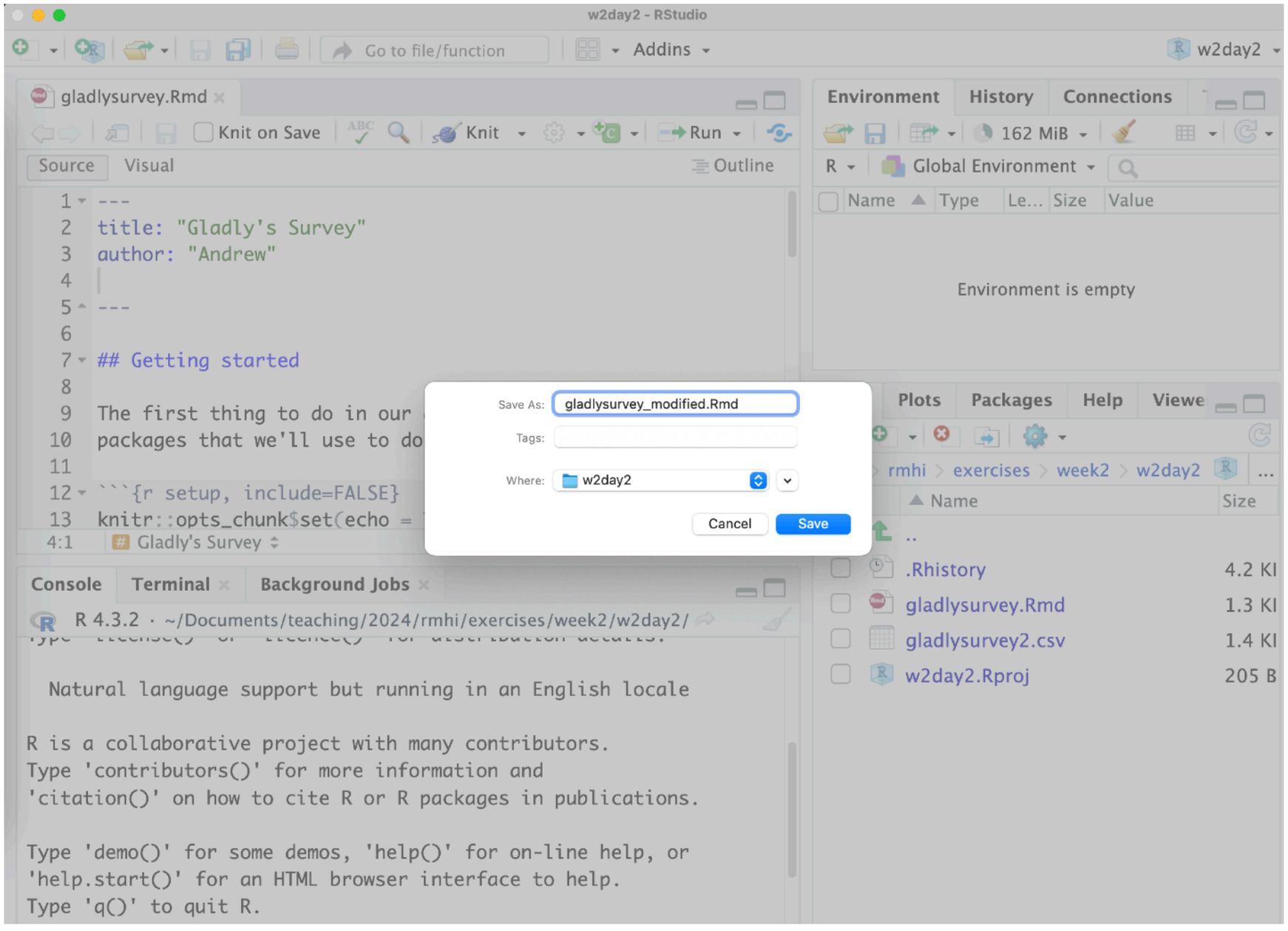
R Markdown

We'll go into the parts of it in a bit, but let's first save it as a new thing so that if we make changes, they don't overwrite the original. (I always try to save right away because I'm paranoid that way - it's a good habit to get into).

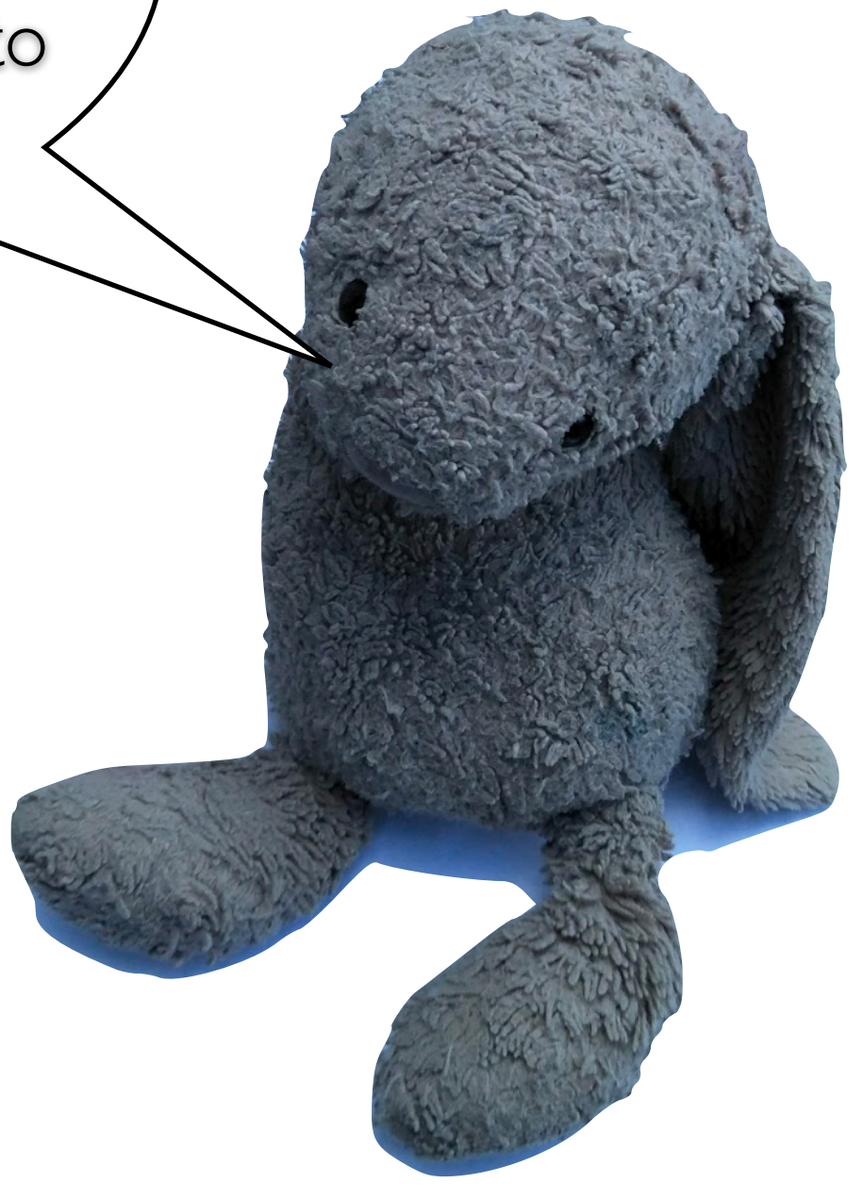


R Markdown

We'll go into the parts of it in a bit, but let's first save it as a new thing so that if we make changes, they don't overwrite the original. (I always try to save right away because I'm paranoid that way - it's a good habit to get into).

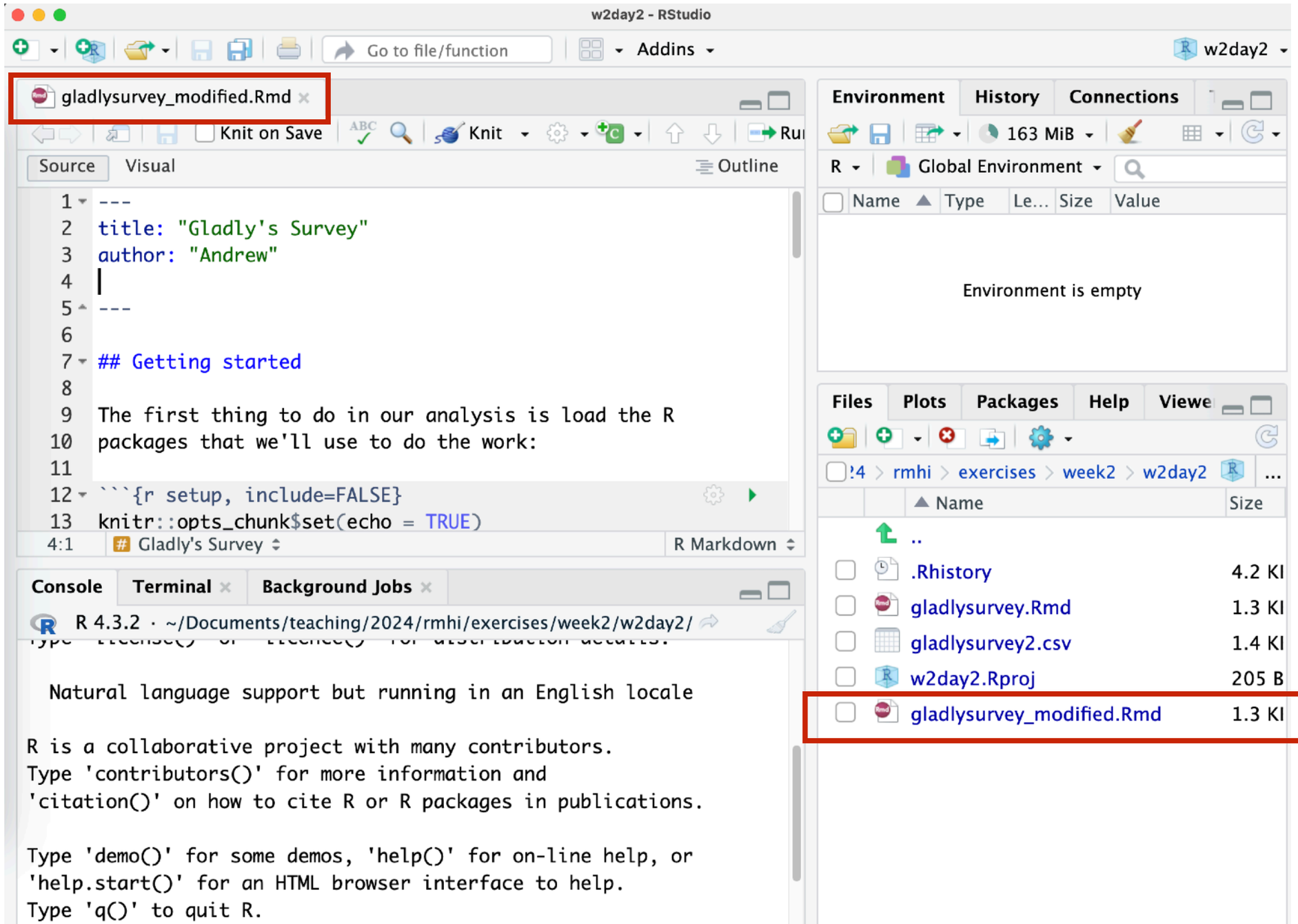


Ooh please tell me
about your paranoid habits
because I have a lot of them myself
and it makes me feel better to
know others do too



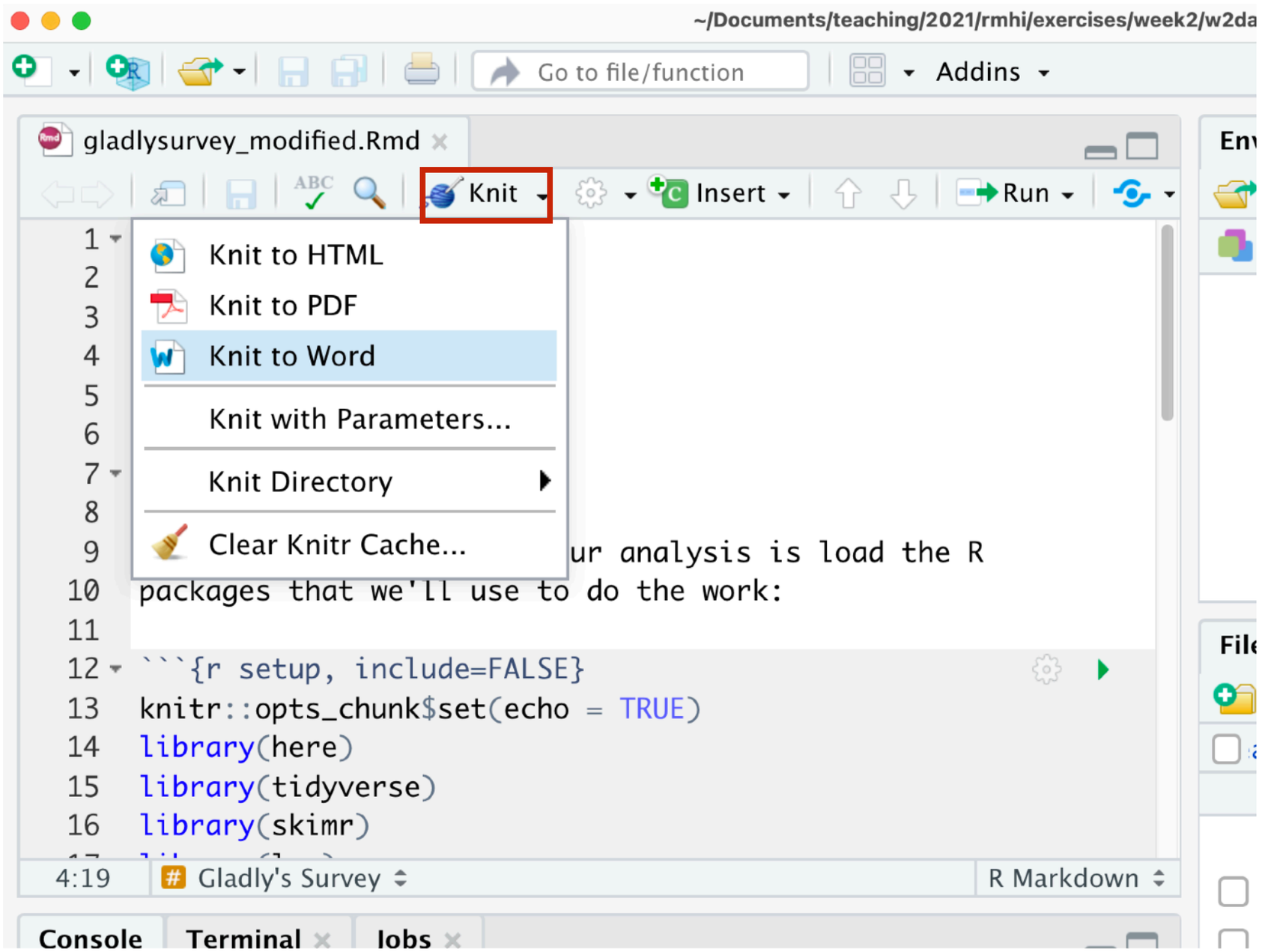
R Markdown

You know it's saved correctly if gladlysurvey_modified shows up on the lower right in your directory, and if the name on the upper left changes



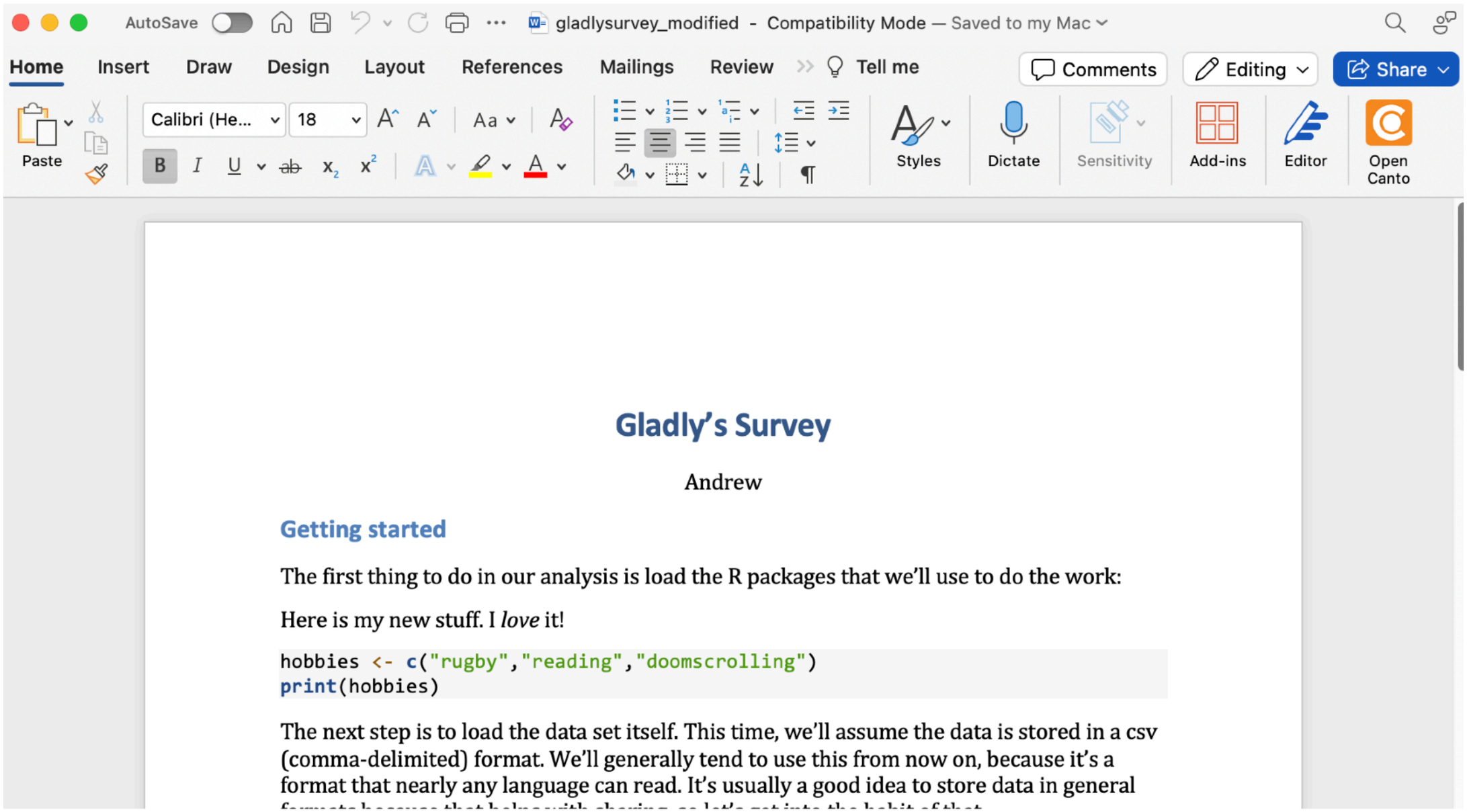
R Markdown

Before we take a look at the parts, let's try creating some output… Go to "Knit" and choose Knit to Word.



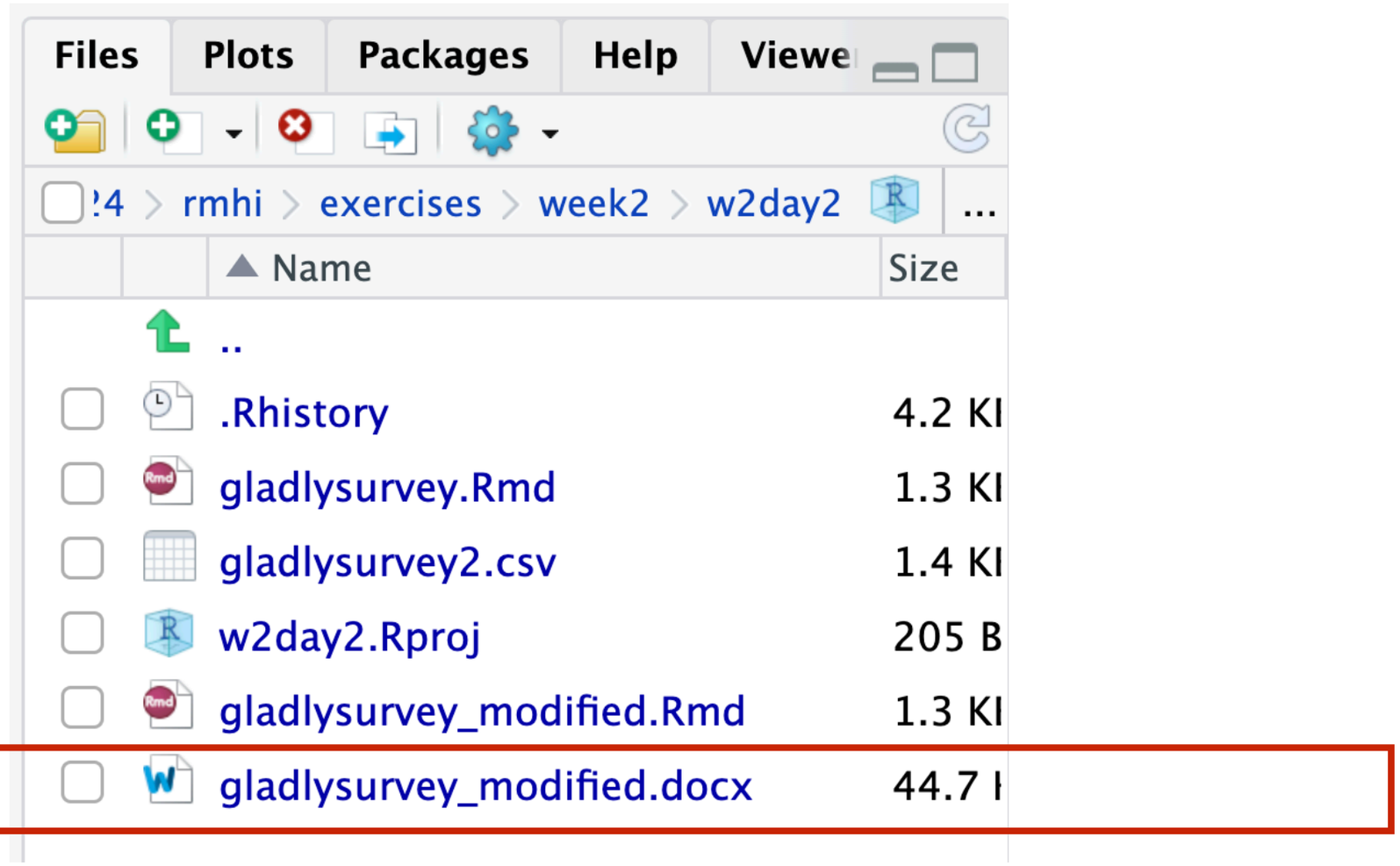
R Markdown

After a moment you should see something that looks like this. It's the Word version of our Markdown document!



R Markdown

You'll also notice that a docx shows up in the panel. This Word version is the *output* of running all of the R commands embedded in the Markdown. You can do assignments and analyses by making R Markdown documents with R commands in them, and then "knit" to create readable documents with all of the outputs of all of your commands

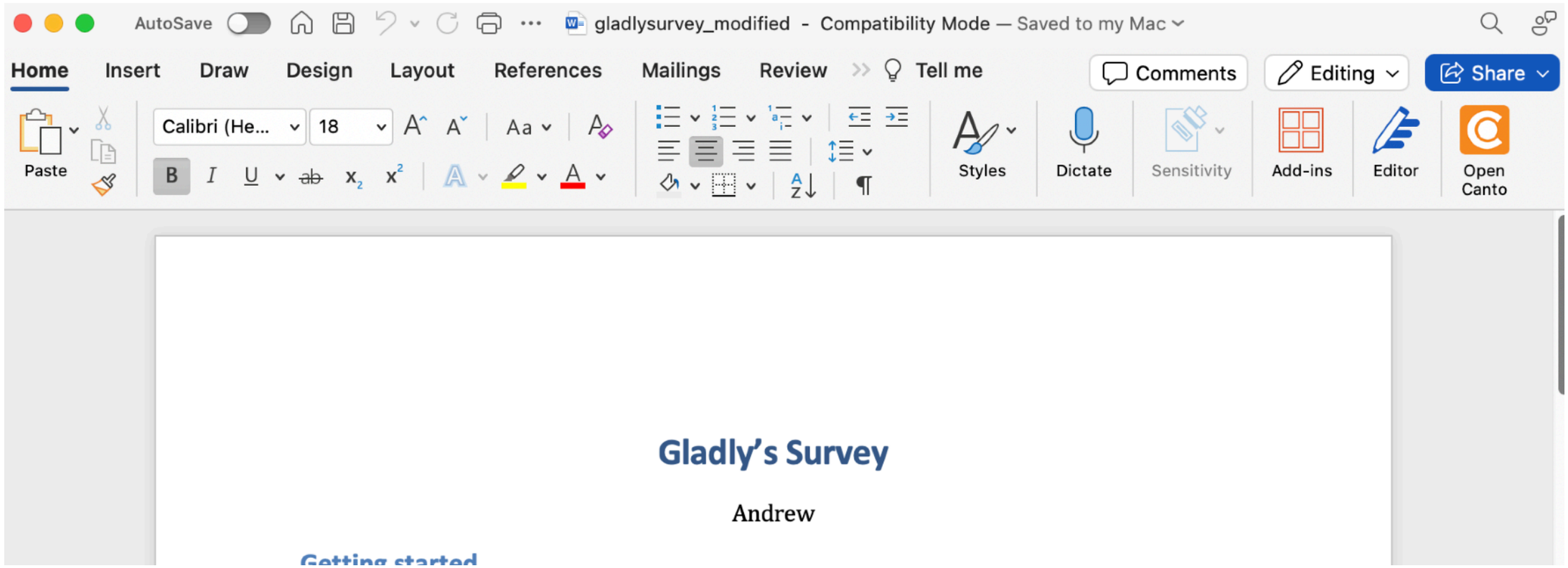


R Markdown

Alright, close the Word document. It's just the output, the end bit. Let's go back to our Markdown file and see what we can do with it

```
---
title: "Gladly's Survey"
author: "Andrew"
output: word_document
---
```

This clearly corresponds to the header of our document, and the values are what we gave it when we created it.



R Markdown

```
## Getting started

The first thing to do in our analysis is load the R
packages that we'll use to do the work:
```

These first bits (and anything not grey and between ``` chunks)
is your text and will show up in your output file:

Getting started

The first thing to do in our analysis is load the R packages that we'll use to do the work:

R Markdown

```
## Getting started

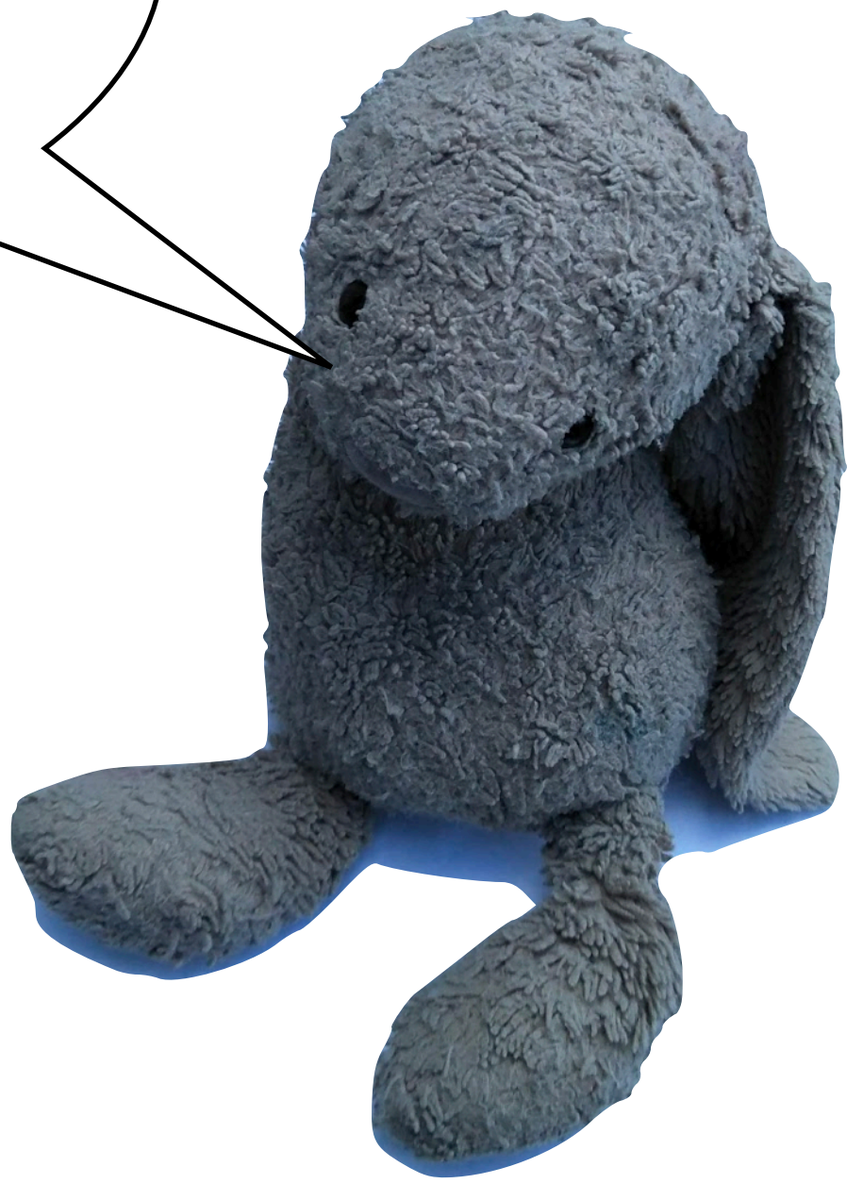
The first thing to do in our analysis is load the R
packages that we'll use to do the work:
```

You can format the text using R Markdown commands. I've included a little "cheatsheet" of such commands in the materials for Week 2 on Canvas, but the main ones you'll find useful are:

R Markdown Formatting

Format	Description	Example
Italic	Italicised text in between the *	*Italic*
Bold	Bolds text in between the **	**Bold**
Pi rounded is `round(3.14)`	Lets you run the R code inline in between the 1	Pi rounded is 3
# Header 1	Makes a really big header	Header 1
## Header 2 (down to 4)	Makes progressively smaller headers	etc.
[Link](https://google.com)	Lets you embed links	Link

This is… a lot of stuff to remember. Really?




Yeah there is no way I'm remembering all of this.


We don't have to! The stuff we use all the time, we'll remember, and for the other stuff the thing to remember is that it exists so you can look it up when you need it.

Code chunks

```
11
12 ```{r setup, include=FALSE}
13 knitr::opts_chunk$set(echo = TRUE)
14 library(here)
15 library(tidyverse)
16 library(lsr)
17 ```
```

This is called a **code chunk**. Code chunks must be set off with the ```{r } and end with another ```. In between, you can put R commands.

Code chunks, like functions, take arguments.

The first argument, before the comma, is the name of the chunk. You can name it anything. These are useful for referring to later, but you don't have to have one.

The argument include tells RStudio whether to show the code in the output. Since we set it as false, the code isn't showing up.

Code chunks

````
11
12 ```{r setup, include=FALSE}
13 knitr::opts_chunk$set(echo = TRUE)
14 library(here)
15 library(tidyverse)
16 library(lsr)
17 ```
````

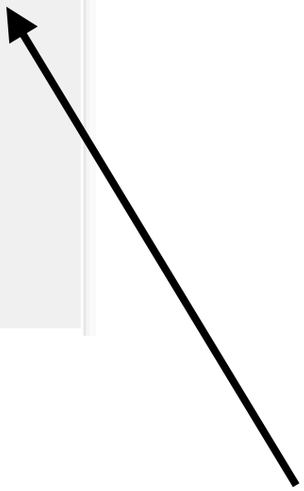
This is called a **code chunk**. Code chunks must be set off with the ```{r } and end with another ```. In between, you can put R commands.

This is an R command. It's in there to knit the document together. Just leave it there.

Here I've loaded three packages we'll need to analyse our data. (If you haven't already installed `lsr`, you should pause the video and do so now using `install.packages()`).

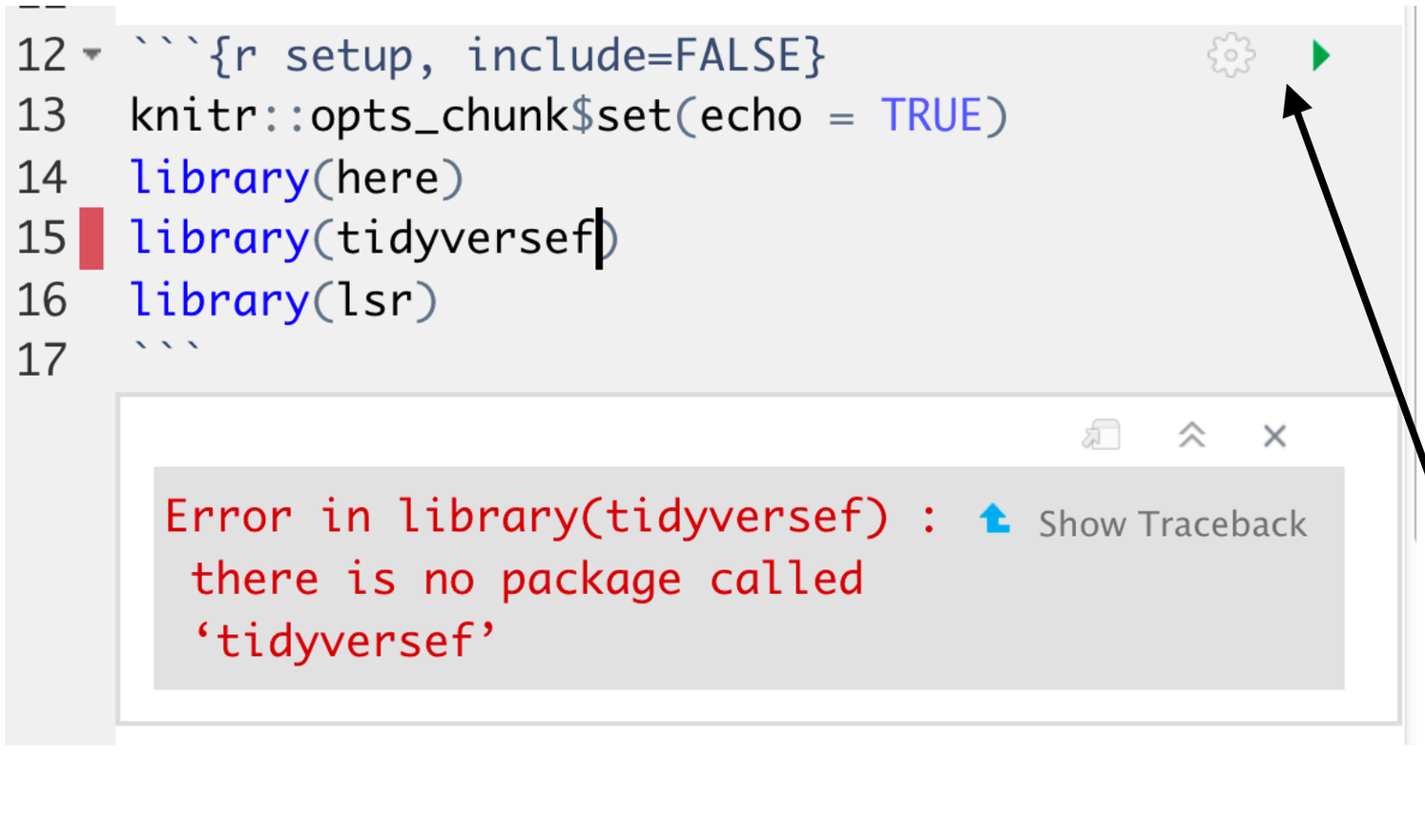
Code chunks

```
11
12 ```{r setup, include=FALSE}
13 knitr::opts_chunk$set(echo = TRUE)
14 library(here)
15 library(tidyverse)
16 library(lsr)
17 ```
```



Note this little green ‘play’ button. If you press it, it will execute the commands in this chunk. (You can see them show up in the console!) It is very handy for debugging as you go!

Code chunks



If you get an error when you press 'Play' it often gives you helpful information about what went wrong. It is red on the line where it stopped, and the error message contains details

I *strongly suggest* that you press Play for each code chunk you create to make sure it's working one by one, and only knit once you've done that!

Code chunks

Next we'll load the data:

````
The next step is to load the data set itself. This time, we'll assume the data
is stored in a csv (comma-delimited) format. We'll generally tend to use this
from now on, because it's a format that nearly any language can read. It's
usually a good idea to store data in general formats because that helps with
sharing, so let's get into the habit of that.

```{r}
loc <- here("gladlysurvey2.csv")
gdata <- read_csv(file = loc)
```
````

Remember that `here()` tells us where to find the file, and is putting that information in the variable we'll call `loc`

Text that shows up in the knitted file (not the code chunk)

Code chunks

Next we'll load the data:

```
The next step is to load the data set itself. This time, we'll assume the data
is stored in a csv (comma-delimited) format. We'll generally tend to use this
from now on, because it's a format that nearly any language can read. It's
usually a good idea to store data in general formats because that helps with
sharing, so let's get into the habit of that.

```{r}
loc <- here("gladlysurvey2.csv")
gdata <- read_csv(file = loc)
```
```

Remember that `here()` tells us where to find the file, and is putting that information in the variable we'll call `loc`

Reads the csv and puts it into a variable called `gdata`, which is a tibble

Code chunks

Next we'll load the data:

```
The next step is to load the data set itself. This time, we'll assume the data
is stored in a csv (comma-delimited) format. We'll generally tend to use this
from now on, because it's a format that nearly any language can read. It's
usually a good idea to store data in general formats because that helps with
sharing, so let's get into the habit of that.

```{r}
loc <- here("gladlysurvey2.csv")
gdata <- read_csv(file = loc)
```
```

```
── Column specification ────────────────────────
cols(
  name = col_character(),
  species = col_character(),
  year = col_double(),
  food = col_character(),
  carrot = col_double(),
  cake = col_double(),
  mud = col_double(),
  age = col_double(),
  gender = col_character()
)
```

When you press play you can see what it's loaded…

Important note

When you press play, it basically just puts the code that was in your code chunk onto the console, and then runs it from there. You can see it happen!

When you knit, nothing goes to the console — all the variables and figures being created stay *in the document*.

It can be easy to get confused between these things.

* If you press play and it doesn't seem to work, that may be because you haven't run the code chunks beforehand that create useful variables.

* If you knit and it doesn't work (but worked when you pressed play) that's often because you may have created a variable via the console in the environment that you need (but is not there in the Markdown)

Exercises

1. Make the following changes to `gladlysurvey_modified.Rmd`:
    1. Change the title to “My first R Markdown document”.
    2. Change my name to yours
    3. Right after the setup code chunk and before loading the data, add a line of text that says “I am learning Markdown! I *love* it.” and then follow it by a chunk that creates a vector called `hobbies` that contains three of your favourite hobbies. Then use the `print()` function to print `hobbies`. Name that code chunk something new, press play to run it, and once that all works, knit the entire document.
2. After doing #2, see if you can figure out the difference between the code chunk arguments `echo`, `include`, and `eval`, as well as what other arguments there are. (Hint: you can do this by experimenting with it yourself and also googling! Google is your friend).
3. On your computer (not R), make a copy of your datafile called gladlystudy3.csv and put it in the folder called `mydata`. See if you can figure out how to load it from within R. (Hint: you’ll have to change the arguments to your `here()` function).
4. Knit your markdown file to html instead of Word.