Gov 50: 2. R, RStudio, and Rmarkdown

Matthew Blackwell

Harvard University

Roadmap

1. Working in Plain Text
2. Let's take a touR
3. Using Rmarkdown
4. Getting R bearings
5. Our first visualizations

1/ Working in Plain Text

The two computer revolutions

The frontier of computing

- Touch-based interfaces

Where statistical computing lives

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time

Where statistical computing lives

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps

Where statistical computing lives

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

- Windows and pointers

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

- Windows and pointers
- Multi-tasking, multiple windows

The two computer revolutions

The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

- Windows and pointers
- Multi-tasking, multiple windows
- Works heavily with the file system

The two computer revolutions

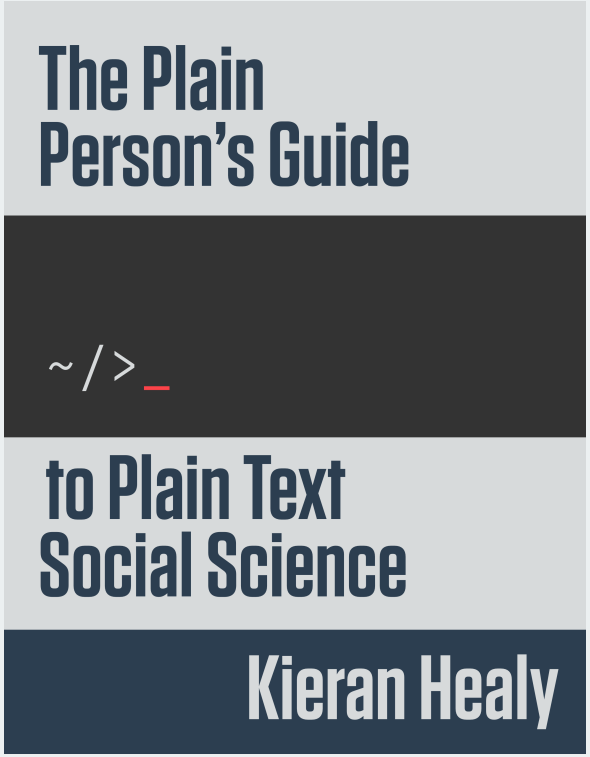
The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

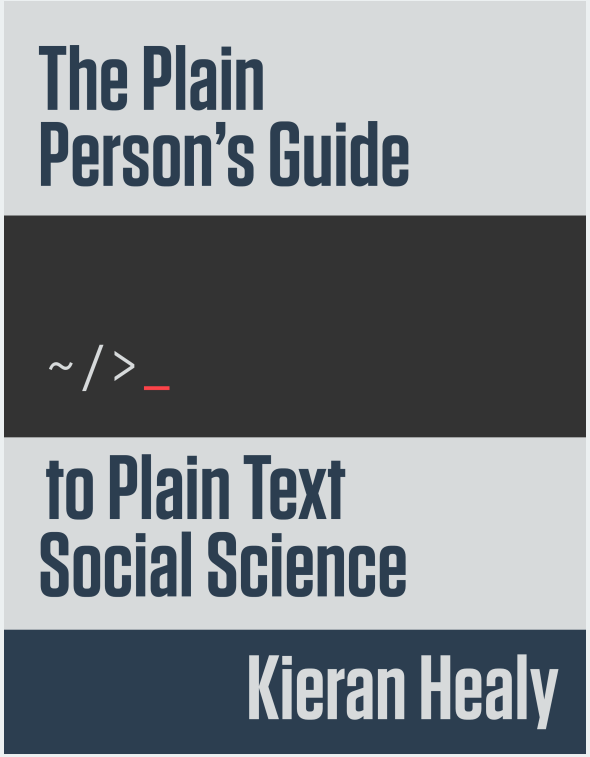
- Windows and pointers
- Multi-tasking, multiple windows
- Works heavily with the file system
- Underneath it's UNIX and the command line

Plain-text tools for data analysis



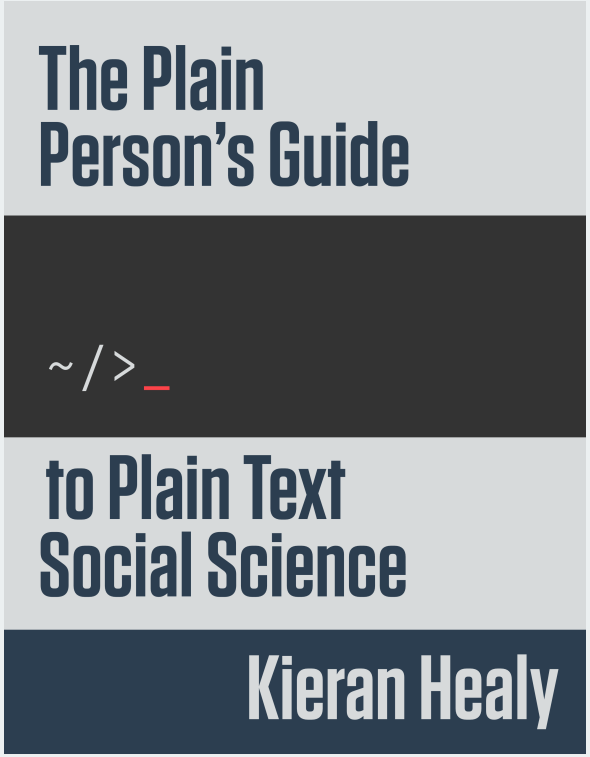
- Often free, open-sourced, and powerful.

Plain-text tools for data analysis



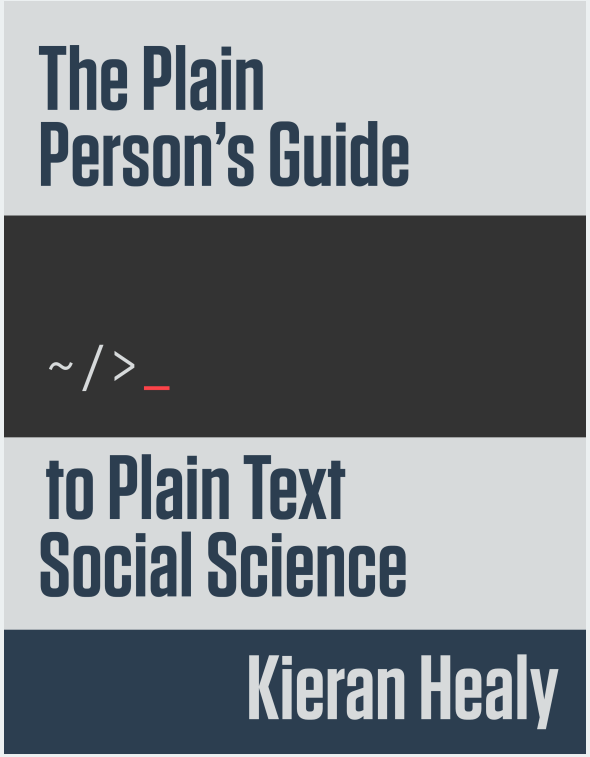
- Often free, open-sourced, and powerful.
- Large, friendly communities around them.

Plain-text tools for data analysis



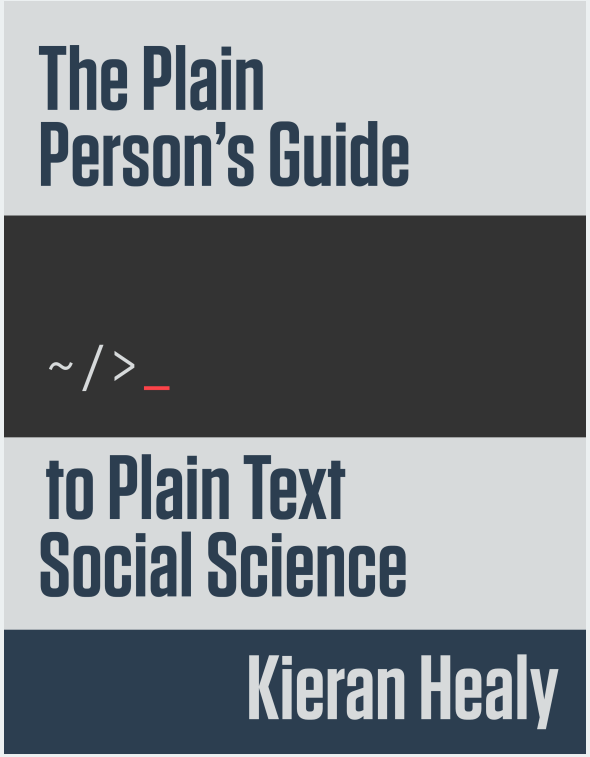
- Often free, open-sourced, and powerful.
- Large, friendly communities around them.
- Tons of resources

Plain-text tools for data analysis



- Often free, open-sourced, and powerful.
- Large, friendly communities around them.
- Tons of resources
- But... far from the touch-based paradigm of modern computing

Plain-text tools for data analysis



- Often free, open-sourced, and powerful.
- Large, friendly communities around them.
- Tons of resources
- But... far from the touch-based paradigm of modern computing
- So why use them?

The process of data science is instrinsically messy

Office vs engineering model of computing

What's real in the project? How are changes managed?

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.

In the Engineering model

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.

In the Engineering model

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.

In the Engineering model

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

- Plain-text files are real.

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

- Plain-text files are real.
- Intermediate outputs are produced via code, often inside documents.

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

- Plain-text files are real.
- Intermediate outputs are produced via code, often inside documents.
- Changes are tracked outside files.

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

- Plain-text files are real.
- Intermediate outputs are produced via code, often inside documents.
- Changes are tracked outside files.
- Final outputs are assembled programatically and converted to desired output format.

Pros and cons to each approach

- Office model:

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.

Pros and cons to each approach

- Office model:
    - Everyone knows Word, Excel, Google Docs.
    - “Track changes” is powerful and easy.

Pros and cons to each approach

- Office model:
    - Everyone knows Word, Excel, Google Docs.
    - “Track changes” is powerful and easy.
    - Wait, how did I make this figure?

Pros and cons to each approach

- Office model:
    - Everyone knows Word, Excel, Google Docs.
    - “Track changes” is powerful and easy.
    - Wait, how did I make this figure?
    - Which version of my code made this table?

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!
  - Version control is a pain.

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!
  - Version control is a pain.
  - `Object of type 'closure' is not subsettable`

Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!
  - Version control is a pain.
  - `Object of type 'closure' is not subsettable`

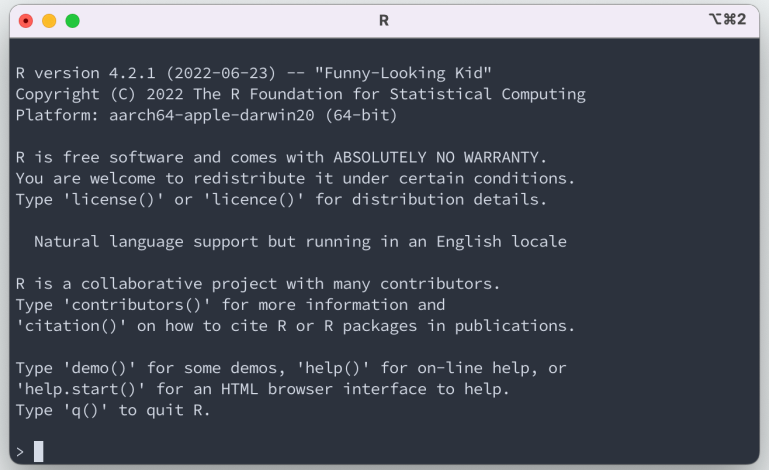
Pros and cons to each approach

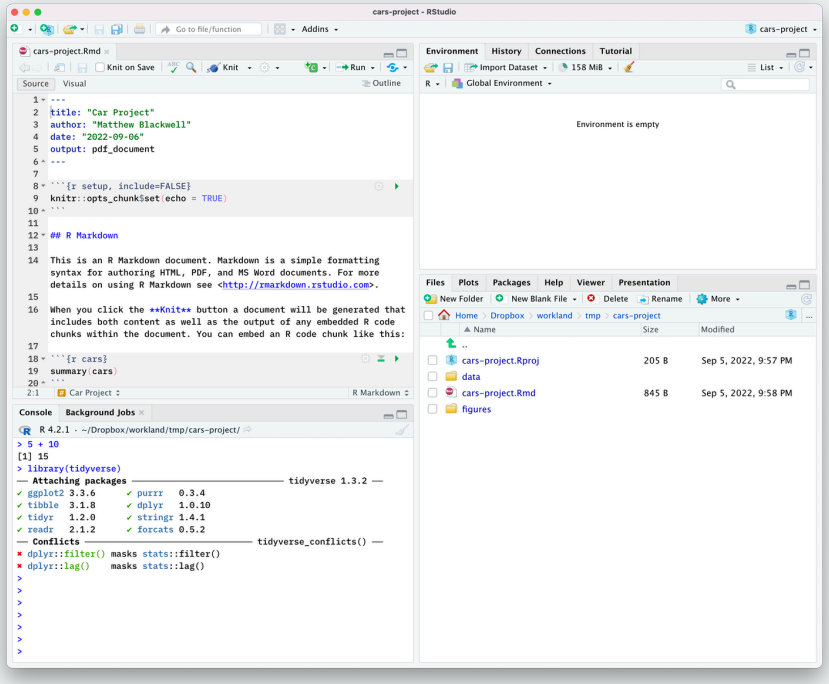
- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!
  - Version control is a pain.
  - `Object of type 'closure' is not subsettable`

We’ll tend toward the Engineering model because it’s better suited to keep the mess in check.

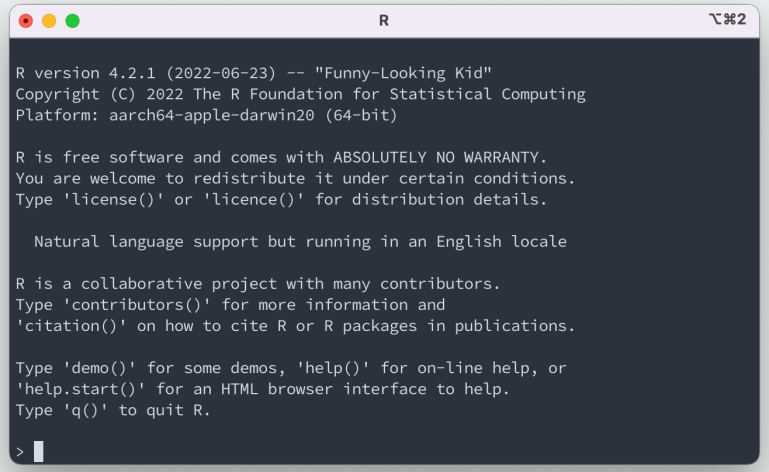
2/ Let’s take a touR

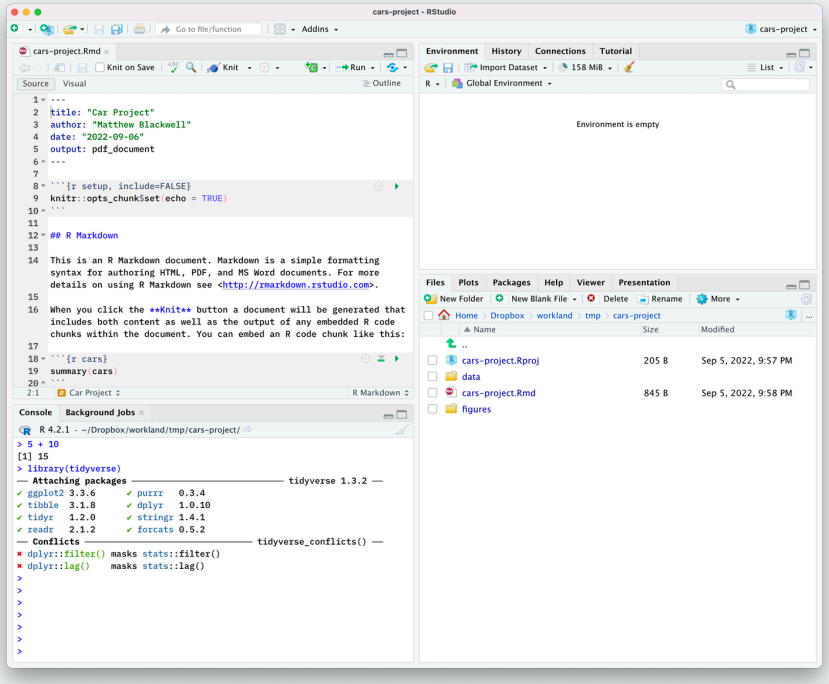
R versus RStudio





R versus RStudio





cars-project - RStudio

Go to file/function | Addins | cars-project

cars-project.Rmd

Knit on Save | Knit | Run

Source | Visual | Outline

```
1  ---
2  title: "Car Project"
3  author: "Matthew Blackwell"
4  date: "2022-09-06"
5  output: pdf_document
6  ---
7
8  ```{r setup, include=FALSE}
9  knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting
   syntax for authoring HTML, PDF, and MS Word documents. For more
   details on using R Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the **Knit** button a document will be generated that
   includes both content as well as the output of any embedded R code
   chunks within the document. You can embed an R code chunk like this:
17
18 ```{r cars}
19 summary(cars)
20 ```
```

2:1 | Car Project | R Markdown

Console | Background Jobs

R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✓ ggplot2 3.3.6     ✓ purrr   0.3.4
✓ tibble  3.1.8     ✓ dplyr   1.0.10
✓ tidyr   1.2.0     ✓ stringr 1.4.1
✓ readr   2.1.2     ✓ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Environment | History | Connections | Tutorial

Import Dataset | 158 MiB | List

R | Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer | Presentation

New Folder | New Blank File | Delete | Rename | More

Home > Dropbox > workland > tmp > cars-project

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

cars-project - RStudio

Write notes, paper in Rmarkdown

cars-project.Rmd

```
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```
```

Console — R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6     ✔ purrr   0.3.4
✔ tibble  3.1.8     ✔ dplyr   1.0.10
✔ tidyr   1.2.0     ✔ stringr 1.4.1
✔ readr   2.1.2     ✔ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Environment: Environment is empty

Files: Home > Dropbox > workland > tmp > cars-project

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

cars-project - RStudio

cars-project.Rmd

```
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

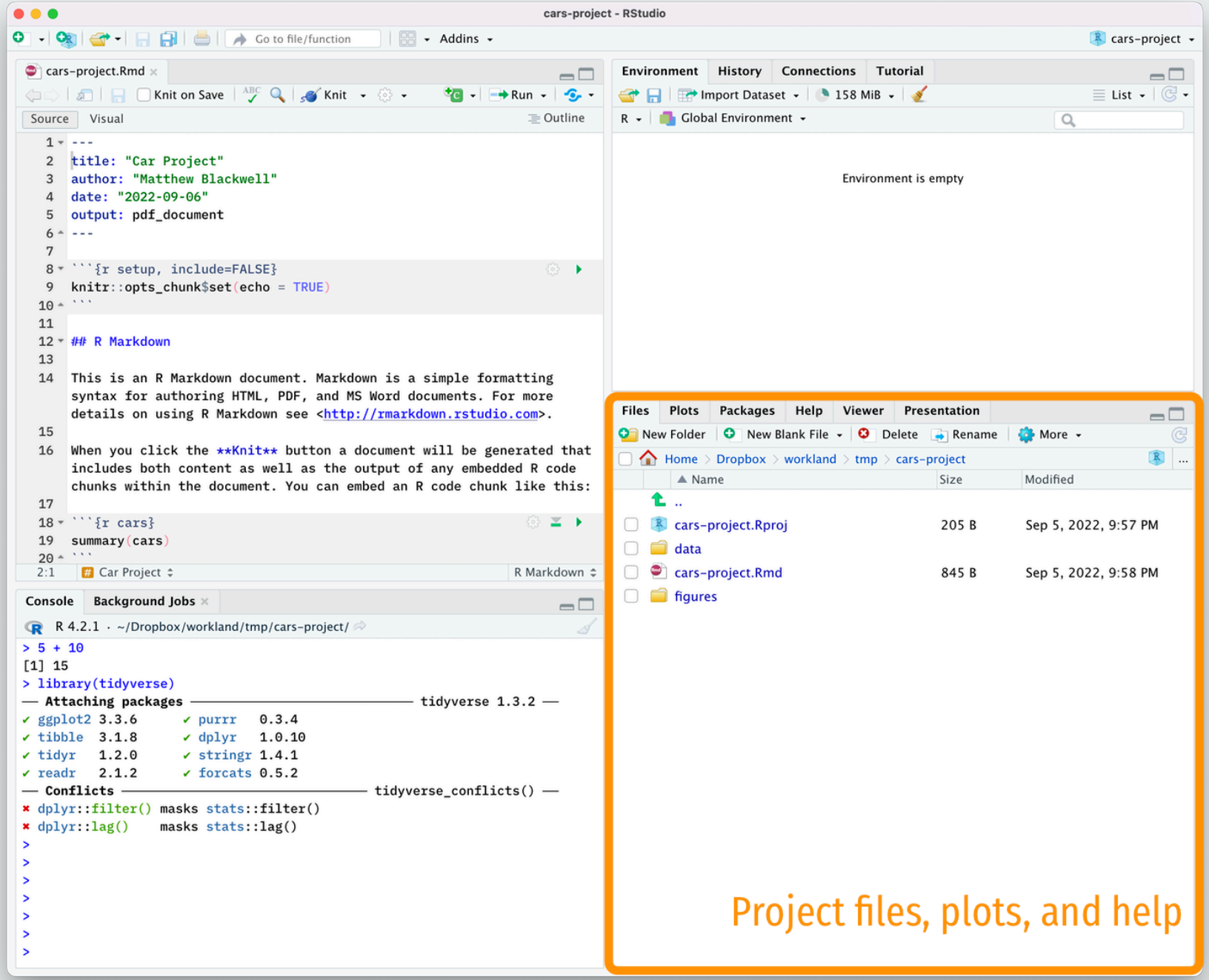
```{r cars}
summary(cars)
```
```

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✓ ggplot2 3.3.6     ✓ purrr   0.3.4
✓ tibble  3.1.8     ✓ dplyr   1.0.10
✓ tidyr   1.2.0     ✓ stringr 1.4.1
✓ readr   2.1.2     ✓ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
```

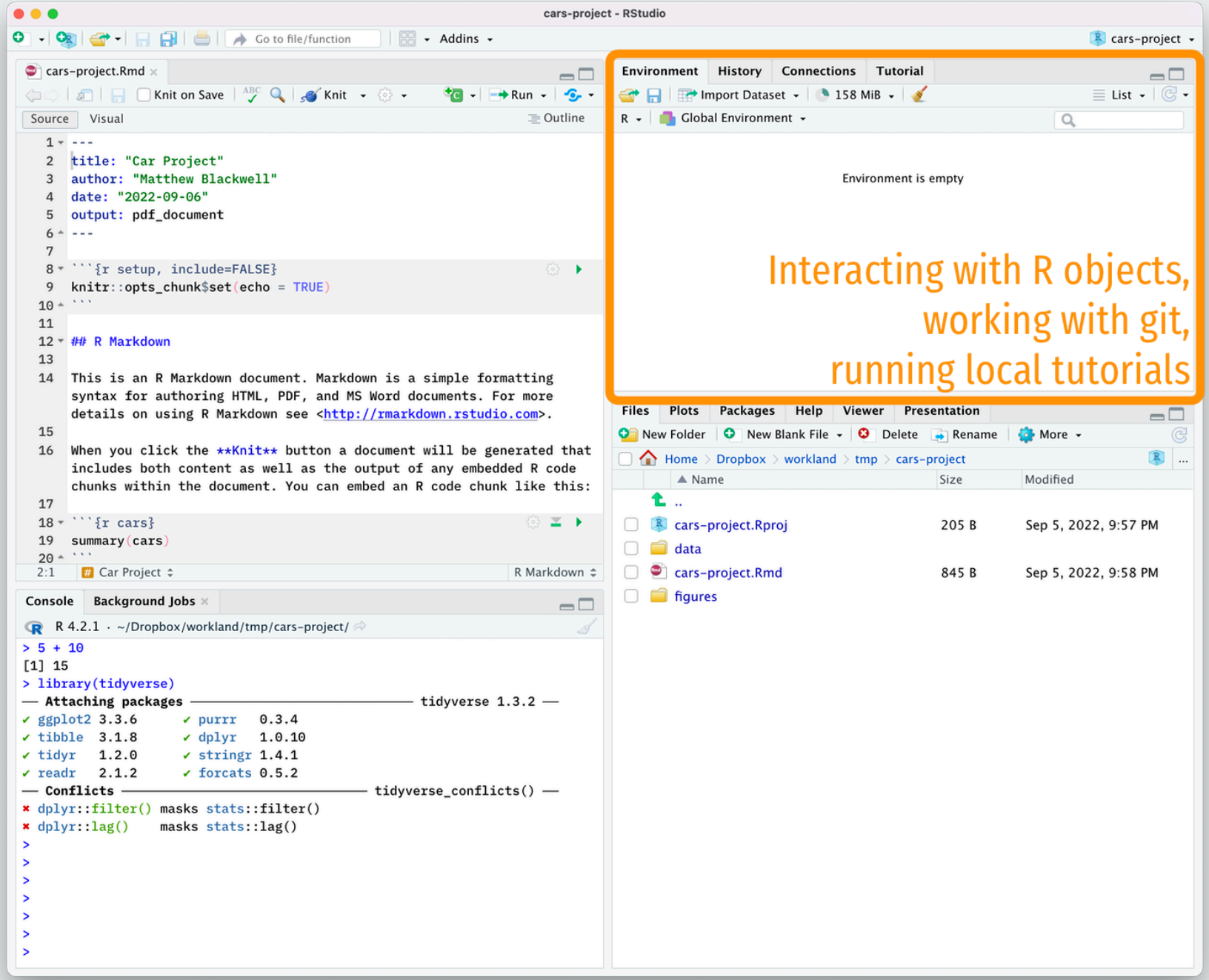
Console: run code, send code to here, inspect output

Environment is empty

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		


cars-project - RStudio
Go to file/function
Addins
cars-project
cars-project.Rmd
Knit on Save
Knit
Run
Source
Visual
Outline
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
```
2:1
Car Project
R Markdown
Console
Background Jobs
R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6 ✔ purrr 0.3.4
✔ tibble 3.1.8 ✔ dplyr 1.0.10
✔ tidyr 1.2.0 ✔ stringr 1.4.1
✔ readr 2.1.2 ✔ forcats 0.5.2
── Conflicts ─── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag() masks stats::lag()
Environment
History
Connections
Tutorial
Import Dataset
158 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New Blank File
Delete
Rename
More
Home > Dropbox > workland > tmp > cars-project
Name
Size
Modified
..
cars-project.Rproj 205 B Sep 5, 2022, 9:57 PM
data
cars-project.Rmd 845 B Sep 5, 2022, 9:58 PM
figures
Project files, plots, and help


cars-project - RStudio
Go to file/function
Addins
cars-project
cars-project.Rmd
Knit on Save
Knit
Run
Source
Visual
Outline
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
```
2:1
Car Project
R Markdown
Console
Background Jobs
R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6 ✔ purrr 0.3.4
✔ tibble 3.1.8 ✔ dplyr 1.0.10
✔ tidyr 1.2.0 ✔ stringr 1.4.1
✔ readr 2.1.2 ✔ forcats 0.5.2
── Conflicts ─── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag() masks stats::lag()
Environment
History
Connections
Tutorial
Import Dataset
158 MiB
List
R
Global Environment
Environment is empty
Interacting with R objects,
working with git,
running local tutorials
Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New Blank File
Delete
Rename
More
Home > Dropbox > workland > tmp > cars-project
Name
Size
Modified
..
cars-project.Rproj 205 B Sep 5, 2022, 9:57 PM
data
cars-project.Rmd 845 B Sep 5, 2022, 9:58 PM
figures

3/ Using Rmarkdown

The acts of coding

```r
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Figure: 1. Writing code

The acts of coding

```r
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Figure: 1. Writing code

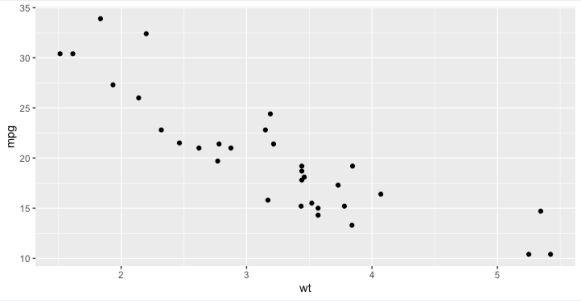


Figure: 2. Looking at output

The acts of coding

```r
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Figure: 1. Writing code

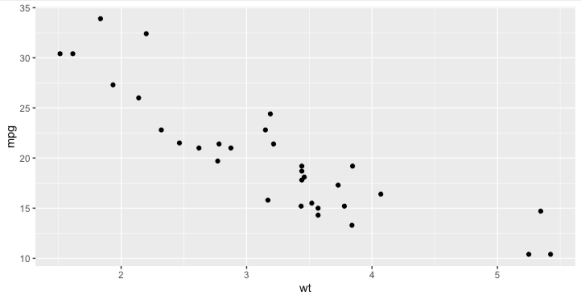


Figure: 2. Looking at output

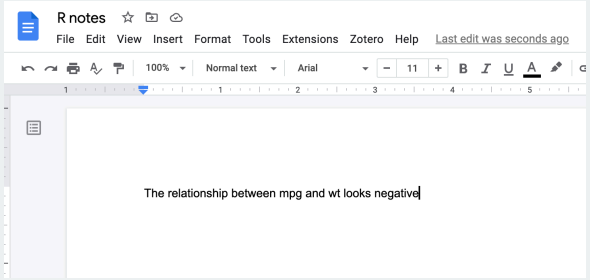


Figure: 3. Taking notes

The acts of coding

```r
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Figure: 1. Writing code

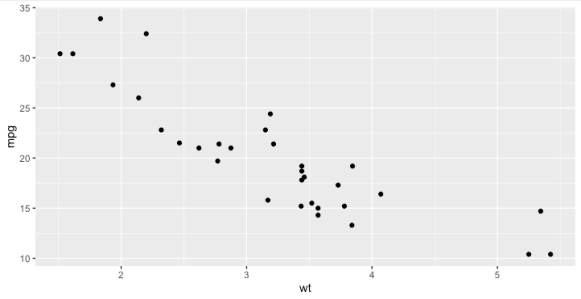


Figure: 2. Looking at output

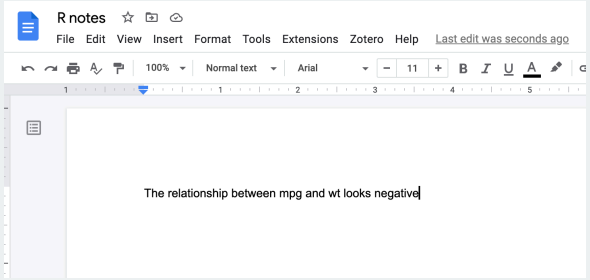


Figure: 3. Taking notes

How to do all of these efficiently?

Rmarkdown files to the rescue

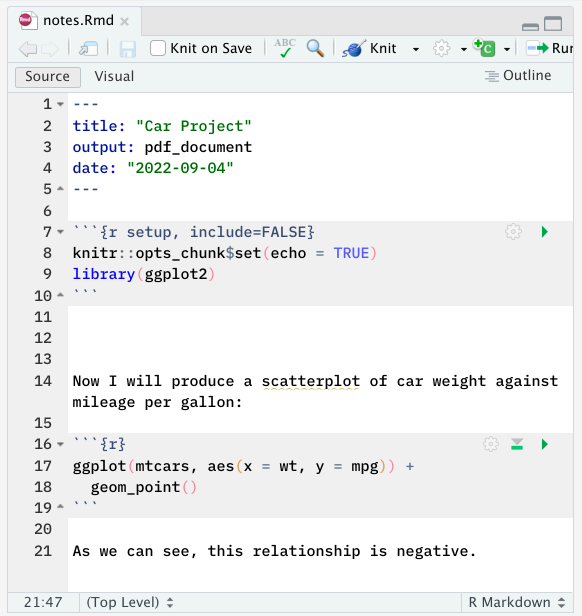


Figure: Rmarkdown file

Keep code and notes together in plain text

Rmarkdown files to the rescue

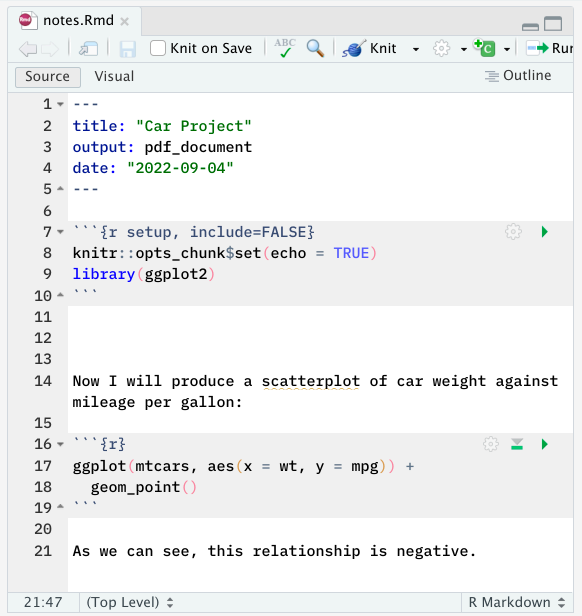


Figure: Rmarkdown file

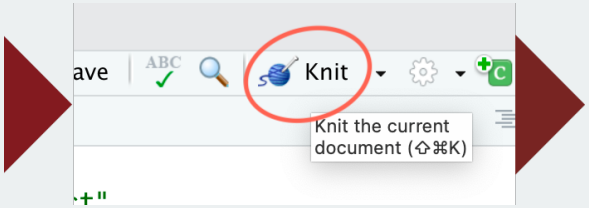


Figure: Knit in R

Keep code and notes together in plain text

Rmarkdown files to the rescue

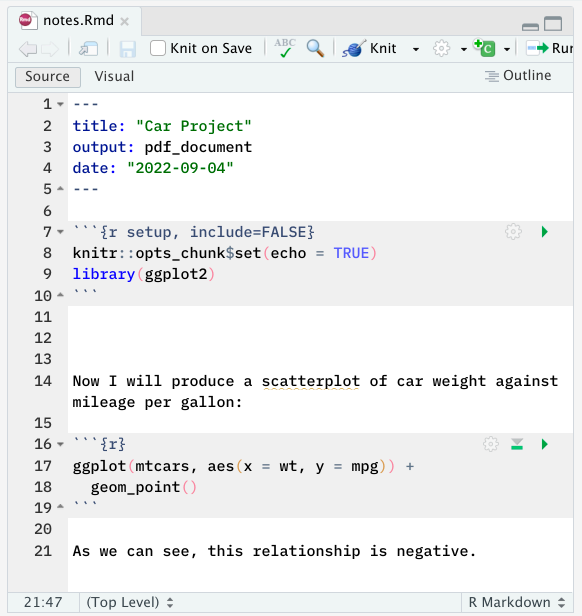


Figure: Rmarkdown file

Keep code and notes together in plain text

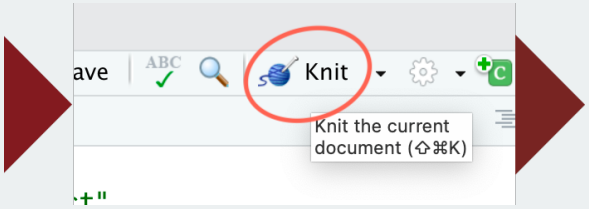


Figure: Knit in R

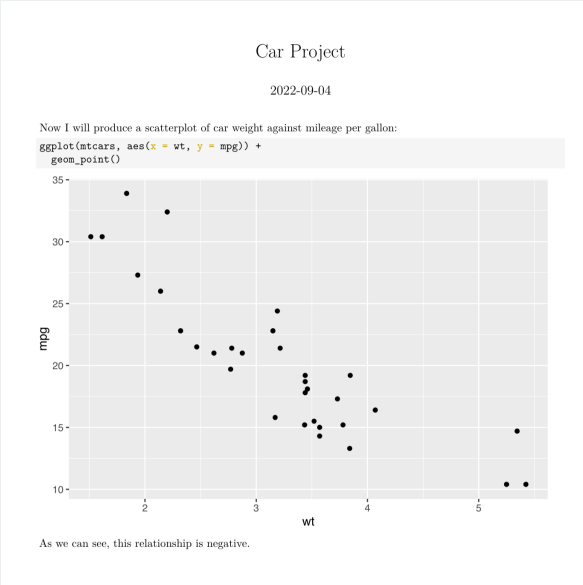


Figure: PDF output

Produce nice-looking outputs in different formats

Markdown: formatting in plain text

Non-code text in Rmd files is plain text with formatting instructions

syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: –
emdash: —
ellipsis: …
inline equation: $A = \pi * r^2$
image: 
horizontal rule (or slide break):

> block quote

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

```
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---
```

Header contains metadata and sets options about the whole document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Code Chunk

```
## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for
authoring HTML, PDF, and MS Word documents. For more details on using R
Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes
both content as well as the output of any embedded R code chunks within the
document. You can embed an R code chunk like this:
```

Plain text with markdown formatting

```{r cars}
summary(cars)
```

Can "play" chunks interactively

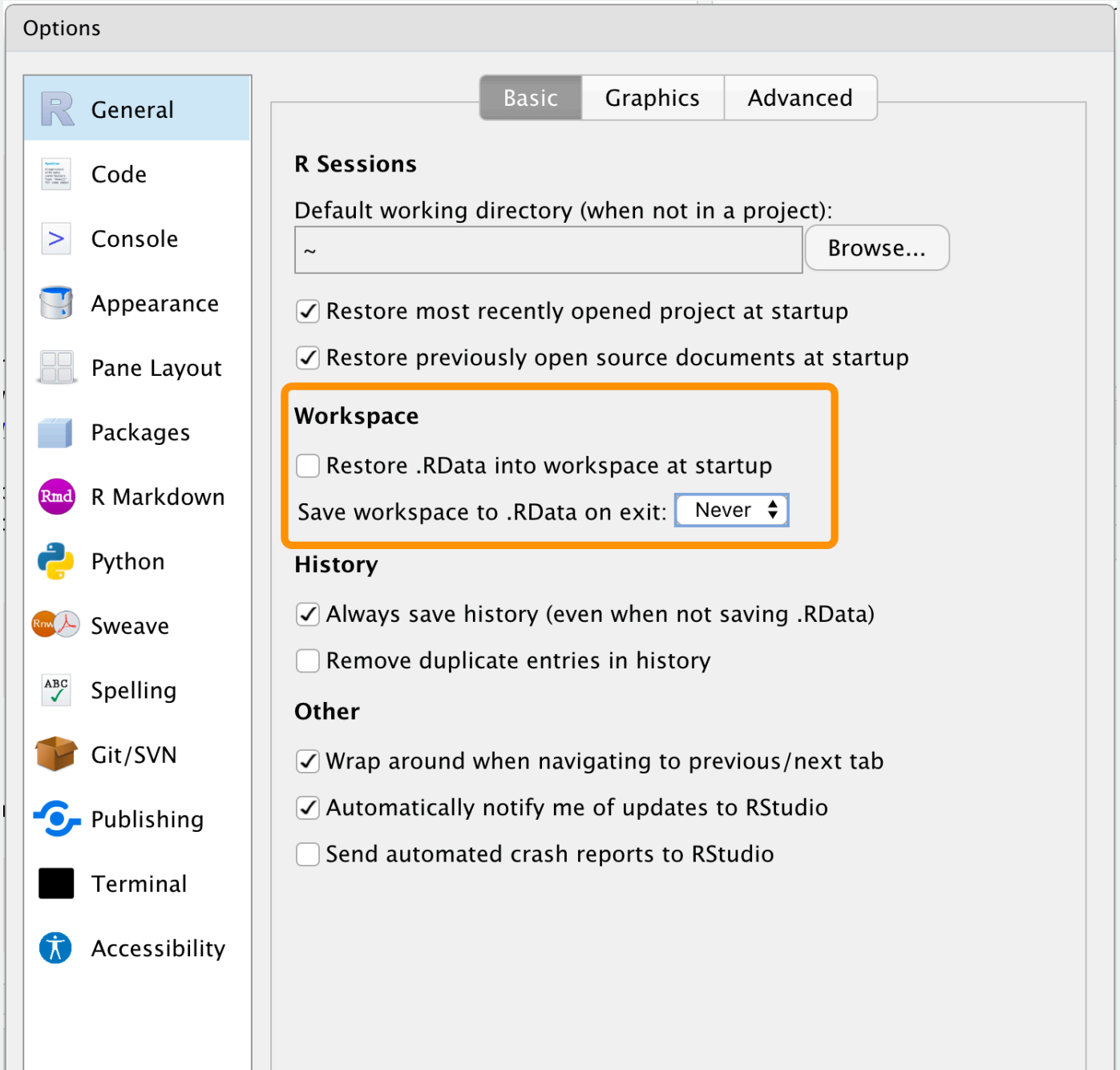
```
## Including Plots

You can also embed plots, for example:
```

Chunks can have names and options

```{r pressure, echo=FALSE}
plot(pressure)
```

Code chunks replaced with output when Knitted

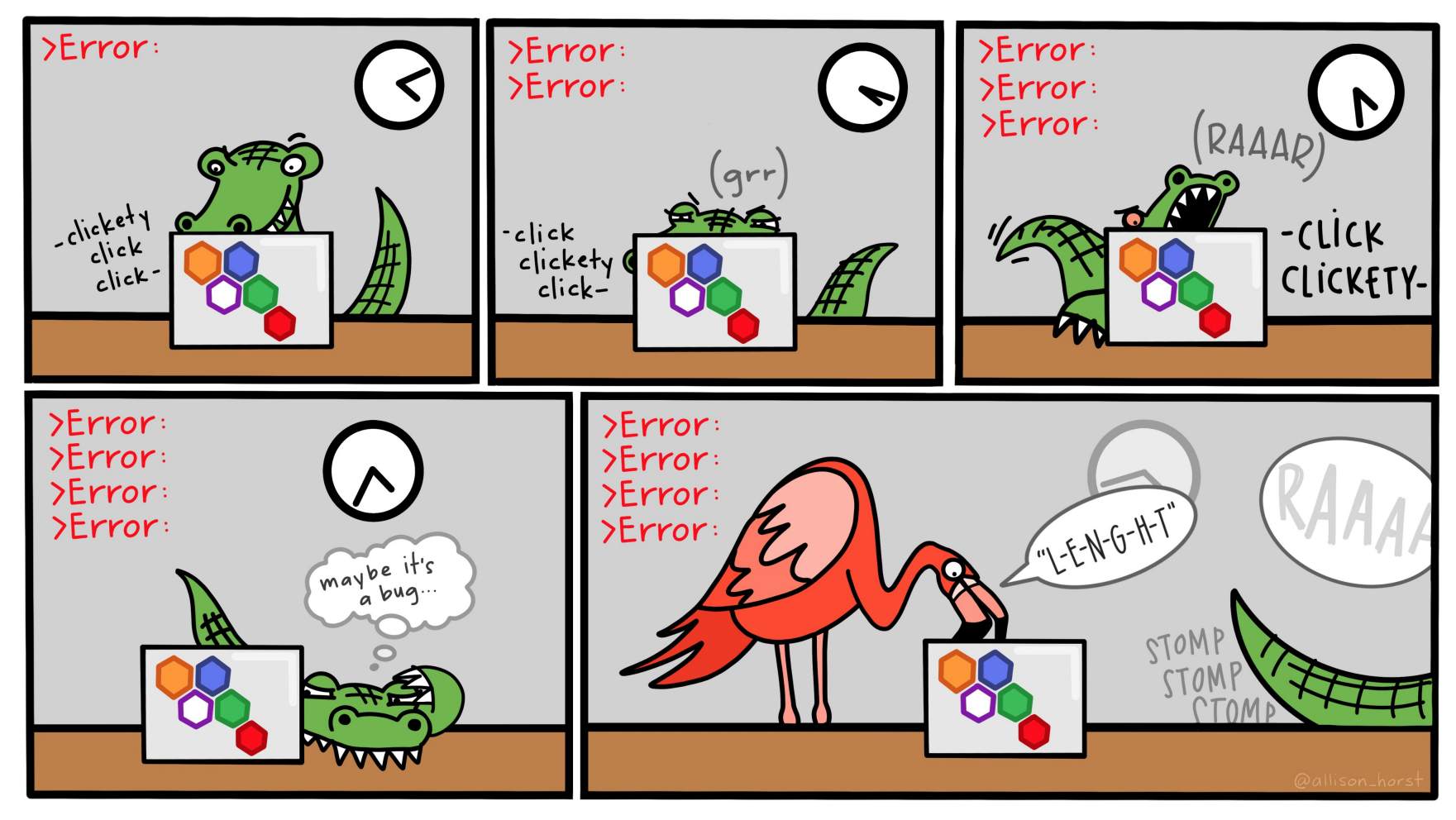
Remember what's real



4/ Getting R bearings

Try to type your code by hand

Typing speeds up the try-fail cycle



Physically typing the code is best way to familiarize yourself with R and the try-fail-try-fail-try-succeed cycle

Credit: Allison Horst

What R looks like

Code that you can type and run:

```
## Any R code that begins with the # character is a comment
## Comments are ignored by R

my_numbers <- c(4, 8, 15, 16, 23, 42) # Anything after # is also a comment
```

What R looks like

Code that you can type and run:

```
## Any R code that begins with the # character is a comment
## Comments are ignored by R

my_numbers <- c(4, 8, 15, 16, 23, 42) # Anything after # is also a comment
```

Output from code prefixed by ## by convention:

```
my_numbers
```

```
## [1]  4  8 15 16 23 42
```

What R looks like

Code that you can type and run:

```
## Any R code that begins with the # character is a comment
## Comments are ignored by R

my_numbers <- c(4, 8, 15, 16, 23, 42) # Anything after # is also a comment
```

Output from code prefixed by ## by convention:

```
my_numbers
```

```
## [1]  4  8 15 16 23 42
```

Output also has a counter in brackets when over one line:

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
## [15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

Everything in R has a name

```r
my_numbers # just created this
```

```
## [1]  4  8 15 16 23 42
```

```r
letters # this is built into R
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
## [15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
pi # also built in
```

```
## [1] 3.14
```

Everything in R has a name

```r
my_numbers # just created this
```

```
## [1]  4  8 15 16 23 42
```

```r
letters # this is built into R
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
## [15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
pi # also built in
```

```
## [1] 3.14
```

Some names are forbidden (`NA`, `TRUE`, `FALSE`, etc) or strongly not recommended (`c`, `mean`, `table`)

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```
mean(x = my_numbers)
```

```
## [1] 18
```

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```
mean(x = my_numbers)
```

```
## [1] 18
```

- x is the argument name,

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```
mean(x = my_numbers)
```

```
## [1] 18
```

- `x` is the argument name,
- `my_numbers` is what we're passing to the that argument

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```
mean(x = my_numbers)
```

```
## [1] 18
```

- `x` is the argument name,
- `my_numbers` is what we're passing to the that argument

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```
mean(x = my_numbers)
```

```
## [1] 18
```

- x is the argument name,
- `my_numbers` is what we're passing to the that argument

If you omit the argument name, R will assume the default order:

```
mean(my_numbers)
```

```
## [1] 18
```

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.
  - Ask on Ed or on class Slack.

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.
  - Ask on Ed or on class Slack.
  - Come to section, office hours, study hall.

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.
  - Ask on Ed or on class Slack.
  - Come to section, office hours, study hall.
- Get help **early** before becoming too frustrated!

Getting help with R

How do we know the default argument order? Look to help files:

```
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.
  - Ask on Ed or on class Slack.
  - Come to section, office hours, study hall.
- Get help **early** before becoming too frustrated!
  - Easy to overlook small issues like missing commas, etc.

Functions live in packages

Packages are bundles of functions written by other users that we can use.

Functions live in packages

Packages are bundles of functions written by other users that we can use.

Install packages using `install.packages()` to have them on your machine:

```
install.packages("ggplot2")
```

Functions live in packages

Packages are bundles of functions written by other users that we can use.

Install packages using `install.packages()` to have them on your machine:

```
install.packages("ggplot2")
```

Load them into your R session with `library()`:

```
library(ggplot2)
```

Functions live in packages

Packages are bundles of functions written by other users that we can use.

Install packages using `install.packages()` to have them on your machine:

```
install.packages("ggplot2")
```

Load them into your R session with `library()`:

```
library(ggplot2)
```

Now we can use any function provided by `ggplot2`.

Functions live in packages

We can also use the `mypackage::` prefix to access package functions without loading:

```
knitr::kable(head(mtcars))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.5	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.0	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.6	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.4	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.0	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.2	1	0	3	1

5/ Our first visualizations

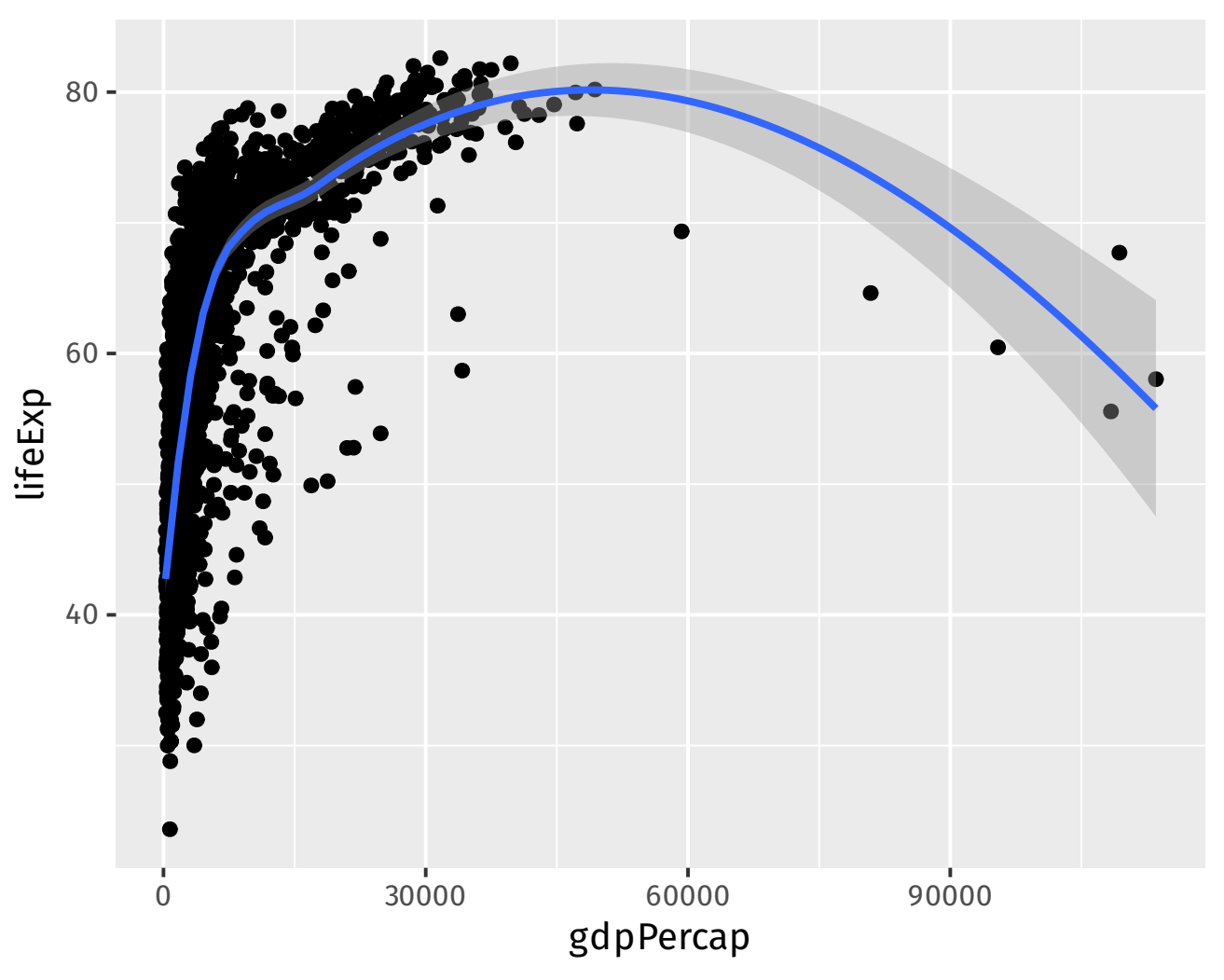
Gapminder data

```r
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # i 1,694 more rows
```

Plotting life expectancy over time

```
ggplot(gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point() + geom_smooth(method = "loess")
```



A histogram of GDP per capita

```
ggplot(gapminder, mapping = aes(x = gdpPercap)) +
  geom_histogram()
```