

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

The Plain Person's Guide

~/>_

to Plain Text
Social Science

Kieran Healy

- Often free, open-sourced, and powerful.

Plain-text tools for data analysis

The Plain Person's Guide

~/>_

to Plain Text
Social Science

Kieran Healy

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

Plain-text tools for data analysis

The Plain Person's Guide

~/>_

to Plain Text
Social Science

Kieran Healy

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

Plain-text tools for data analysis

The Plain Person's Guide

~/>_

to Plain Text Social Science

Kieran Healy

- 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

The Plain Person's Guide

~/>_

to Plain Text Social Science

Kieran Healy

- 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 intrinsically
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 outputs
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 outputs
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 outputs 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 outputs 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 outputs 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 outputs 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 outputs 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
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

```
cars-project.Rmd
1 title: "Car Project"
2 author: "Matthew Eckhoff"
3 date: "2022-09-09"
4 output: pdf_document
5 ---
6
7
8 """" (to view: include("01.Rmd"))
9 knitr: apply_chunked_code = TRUE
10
11 ## R Notebook
12
13 This is an R Notebook document. R Markdown is a simple formatting
14 system for authoring HTML, PDF, and MS Word documents. For more
15 details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 When you click the refresh button a document will be generated that
18 includes both content as well as the output of any embedded R code
19 chunks within the document. You can embed an R code chunk like this:
20
21 ```{r}
22 #> library(cars)
23 #> summary(cars)
24
25 ##> Car Project
26
27 ##> R Markdown
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



cars-project - RStudio

cars-project

cars-project.Rmd

```
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
15 syntax for authoring HTML, PDF, and MS Word documents. For more
16 details on using R Markdown see http://rmarkdown.rstudio.com.
17
18 When you click the Knit button a document will be generated that
19 includes both content as well as the output of any embedded R code
20 chunks within the document. You can embed an R code chunk like this:
21
22 '''[r cars]
23 summary(cars)
24 '''
25
```

Environment History Connections Tutorial

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		

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()
>
>
>
>
>
```

cars-project - RStudio

cars-project

Environment History Connections Tutorial

Import Dataset 158 MiB

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		

```
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
15 syntax for authoring HTML, PDF, and MS Word documents. For more
16 details on using R Markdown see http://rmarkdown.rstudio.com.
17
18 When you click the Knit button a document will be generated that
19 includes both content as well as the output of any embedded R code
20 chunks within the document. You can embed an R code chunk like this:
21
22 '''[r cars]
23 summary(cars)
24 '''
25
26 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()
>
>
>
>
>
```

Write notes,
paper in
Rmarkdown

The screenshot shows the RStudio interface with a Knit session in progress. The source editor displays a markdown document with R code chunks. The console window, highlighted with an orange border, shows the following output:

```
R 4.2.1 - ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
Attaching packages:
  ggplot2 3.3.6      purrr  0.3.4
  tidbale 3.1.8      dplyr  1.0.10
  tidyr  1.2.0       stringr 1.4.1
  readr  2.1.2       forcats 0.5.2
tidyverse 1.3.2
Conflicts:
  dplyr::filter() masks stats::filter()
  dplyr::lag()    masks stats::lag()
>
>
>
>
>
```

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

The image shows the RStudio interface with an R Markdown document open. The document content is as follows:

```
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
15 syntax for authoring HTML, PDF, and MS Word documents. For more
16 details on using R Markdown see http://rmarkdown.rstudio.com.
17
18 When you click the Knit button a document will be generated that
19 includes both content as well as the output of any embedded R code
20 chunks within the document. You can embed an R code chunk like this:
21
22 '''[r cars]
23 summary(cars)
24 '''
```

The console output shows the following R commands and their results:

```
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()
>
>
>
>
>
```

An orange-bordered window is overlaid on the bottom right, showing a file explorer for the project directory. The window title is "cars-project" and it contains the following files:

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

The image shows the RStudio interface with a project named 'cars-project'. The editor window displays an R Markdown file 'cars-project.Rmd' with the following content:

```
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_chunkset(echo = TRUE)
10 ''
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting
15 syntax for authoring HTML, PDF, and MS Word documents. For more
16 details on using R Markdown see http://rmarkdown.rstudio.com.
17
18 When you click the Knit button a document will be generated that
19 includes both content as well as the output of any embedded R code
20 chunks within the document. You can embed an R code chunk like this:
21
22 ''[r cars]
23 summary(cars)
24 ''
25
```

The console window shows the following R session output:

```
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()
>
>
>
>
>
```

The file explorer on the right shows the project structure:

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		

An orange box highlights the Environment pane, which contains the text: "Environment is empty". Overlaid on this box is the text: "Interacting with R objects, working with git, running local tutorials".

3/ Using Rmarkdown

The acts of coding

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

Figure: 1. Writing code

The acts of coding

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

Figure: 1. Writing code

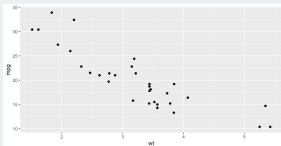


Figure: 2. Looking at output

The acts of coding

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

Figure: 1. Writing code

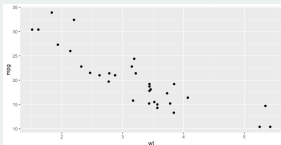


Figure: 2. Looking at output

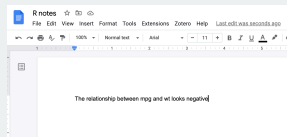


Figure: 3. Taking notes

The acts of coding

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

Figure: 1. Writing code

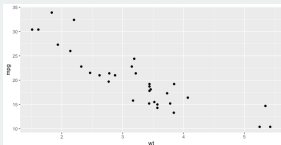


Figure: 2. Looking at output

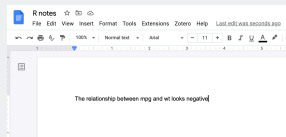
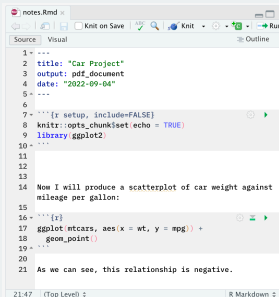


Figure: 3. Taking notes

How to do all of these efficiently?

Rmarkdown files to the rescue

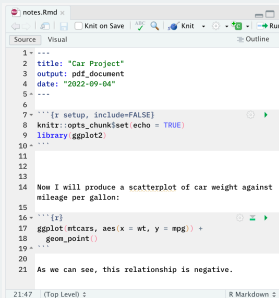


```
1 - ---
2   title: "Car Project"
3   output: pdf_document
4   date: "2022-09-04"
5 - ---
6
7 - {r setup, include=FALSE}
8   knitr::opts_chunk$set(echo = TRUE)
9   library(ggplot2)
10 - {r}
11
12
13
14 Now I will produce a scatterplot of car weight against
15 mileage per gallon:
16
17 {r}
18   ggplot(mtcars, aes(x = wt, y = mpg)) +
19     geom_point()
20 - {r}
21 As we can see, this relationship is negative.
```

Figure: Rmarkdown file

Keep code and notes
together in plain text

Rmarkdown files to the rescue



```
1 ---
2 title: "Car Project"
3 output: pdf_document
4 date: "2022-09-04"
5 ---
6
7 ```{r setup, include=FALSE}
8 knitr::opts_chunk$set(echo = TRUE)
9 library(ggplot2)
10 ```
11
12
13
14 Now I will produce a scatterplot of car weight against
15 mileage per gallon:
16
17 ```{r}
18 ggplot(mtcars, aes(x = wt, y = mpg)) +
19   geom_point()
20 ```
21 As we can see, this relationship is negative.
```

Figure: Rmarkdown file

Keep code and notes
together in plain text

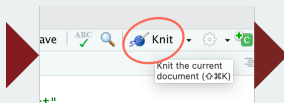
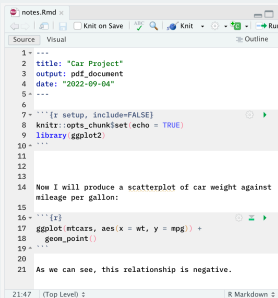


Figure: Knit in R

Rmarkdown files to the rescue



```
1 ---
2 title: "Car Project"
3 output: pdf_document
4 date: "2022-09-04"
5 ---
6
7 {r setup, include=FALSE}
8 knitr::opts_chunk$set(echo = TRUE)
9 library(ggplot2)
10
11
12
13
14 Now I will produce a scatterplot of car weight against
15 mileage per gallon:
16
17 {r}
18 ggplot(mtcars, aes(x = wt, y = mpg)) +
19   geom_point()
20
21 As we can see, this relationship is negative.
```

Figure: Rmarkdown file

Keep code and notes together in plain text

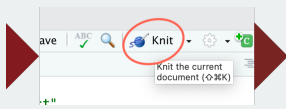


Figure: Knit in R

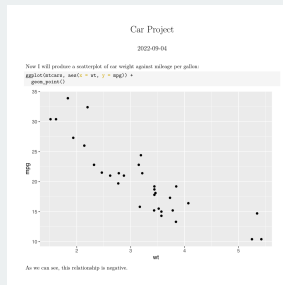


Figure: PDF output

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: 

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
superscript2
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

Plain text with markdown formatting

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)
```
```

Can "play" chunks interactively

Including Plots

Chunks can have names and options

You can also embed plots, for example:

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

Code chunks replaced with output when Knitted

Remember what's real

Options

Basic Graphics Advanced

R Sessions

Default working directory (when not in a project):
~ Browse...

- Restore most recently opened project at startup
- Restore previously open source documents at startup

Workspace

- Restore .RData into workspace at startup

Save workspace to .RData on exit: Never ▾

History

- Always save history (even when not saving .RData)
- Remove duplicate entries in history

Other

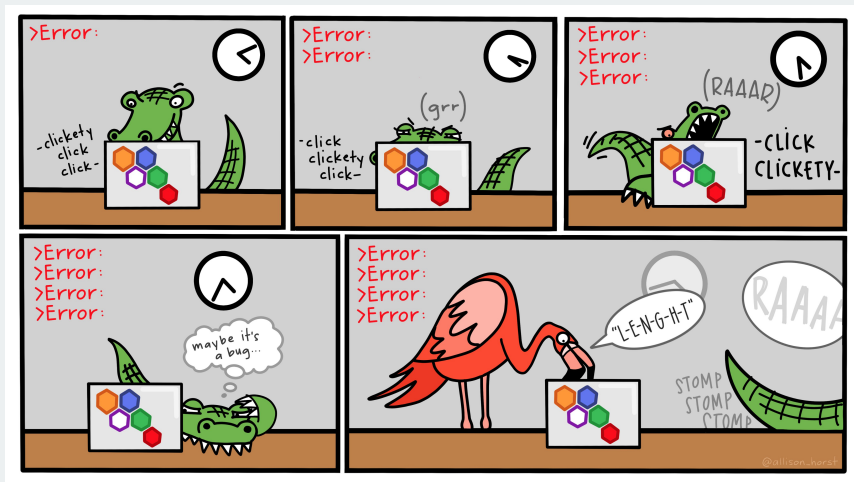
- Wrap around when navigating to previous/next tab
- Automatically notify me of updates to RStudio
- Send automated crash reports to RStudio

General
Code
Console
Appearance
Pane Layout
Packages
R Markdown
Python
Sweave
Spelling
Git/SVN
Publishing
Terminal
Accessibility

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

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

```
my_numbers # just created this
```

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

```
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"
```

```
pi # also built in
```

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

Everything in R has a name

```
my_numbers # just created this
```

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

```
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"
```

```
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 | 21.0 | 6 | 160 | 110 | 3.90 | 2.88 | 17.0 | 0 | 1 | 4 | 4 |
| Wag | | | | | | | | | | | |
| 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 | 18.7 | 8 | 360 | 175 | 3.15 | 3.44 | 17.0 | 0 | 0 | 3 | 2 |
| Sportabout | | | | | | | | | | | |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.46 | 20.2 | 1 | 0 | 3 | 1 |

5/ Our first visualizations

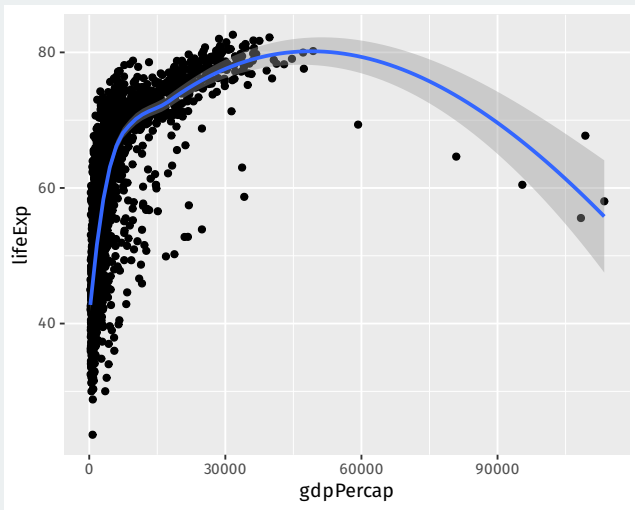
Gapminder data

```
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()
```

