

# A Short Introduction into Data Science with R

Robert Ladwig  
University of Wisconsin-Madison



rladwig2@wisc.edu



@hydrobert

# Disclaimer

- material is based on the AWESOME workshop by Rachel Pilla (now postdoc at Oak Ridge National Lab, @rmpilla) “Introduction to R” (*Computer Science in Modern Biology* at Miami University, Ohio)
- with help by Andrew Cannizzaro, Alva Strand and Nicole Berry



# About myself

$$\frac{\partial DO}{\partial t} = \frac{1}{A} \frac{\partial}{\partial z} (AK_z^{DO} \frac{\partial DO}{\partial z}) + S_{DO}$$

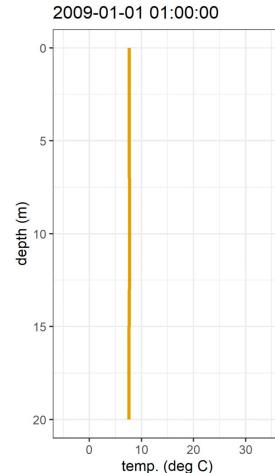


# About myself

- **Physical limnologist**
  - use mathematical models to explore mixing dynamics and water quality in lakes
- developing and applying open-source and open-access software in **R**, **Python** and **Matlab**
- background in civil engineering and geology



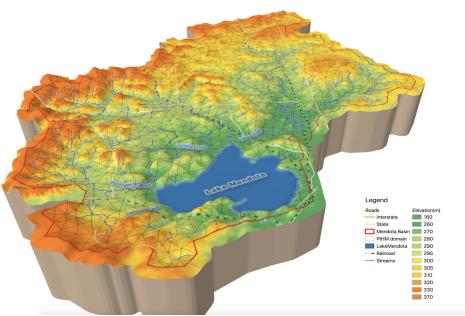
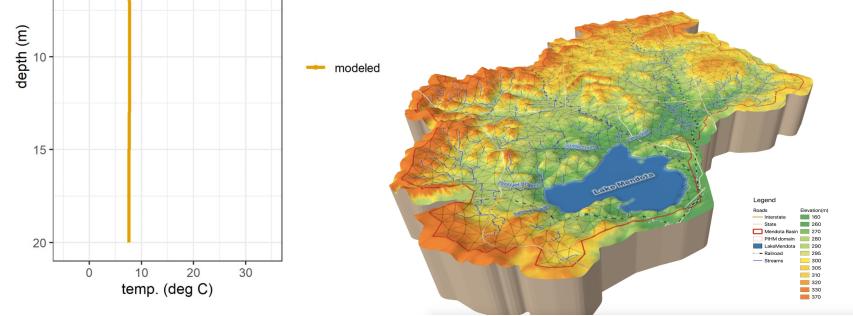
**Aquatic ecosystem modeling**  
mechanistic understanding of cause and effects



$$\frac{\partial T}{\partial t} = \frac{1}{A} \frac{\partial}{\partial z} (A(v_t + v') \frac{\partial T}{\partial z}) + \frac{1}{\rho_0 c_p} \frac{\partial H_{sol}}{\partial z} + \frac{dA}{dz} \frac{H_{geo}}{A \rho_0 c_p}$$

$$\frac{\partial [O_2]}{\partial t} = NEP = GPP - ER \pm D$$

$$\frac{\partial P}{\partial t} = D \frac{\partial^2 P}{\partial z^2} + \lambda P(1 - \frac{P}{K})$$

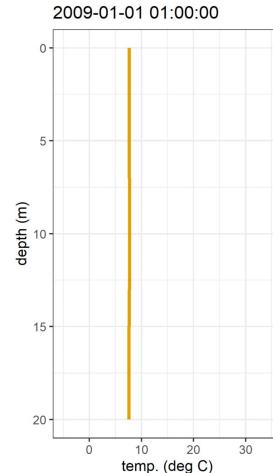


# About myself

- **Physical limnologist**
  - use mathematical models to explore mixing dynamics and water quality in lakes
- developing and applying open-source and open-access software in **R**, **Python** and **Matlab**
- background in civil engineering and geology



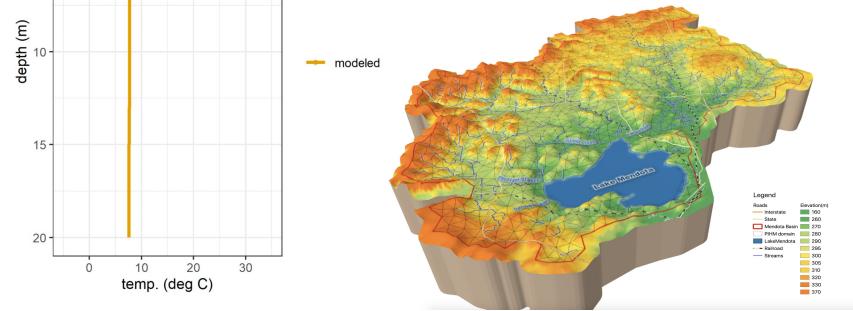
**Aquatic ecosystem modeling**  
mechanistic understanding of cause and effects



$$\frac{\partial T}{\partial t} = \frac{1}{A} \frac{\partial}{\partial z} (A(v_t + v^*) \frac{\partial T}{\partial z}) + \frac{1}{\rho_0 c_p} \frac{\partial H_{sol}}{\partial z} + \frac{dA}{dz} \frac{H_{geo}}{A \rho_0 c_p}$$

$$\frac{\partial [O_2]}{\partial t} = NEP = GPP - ER \pm D$$

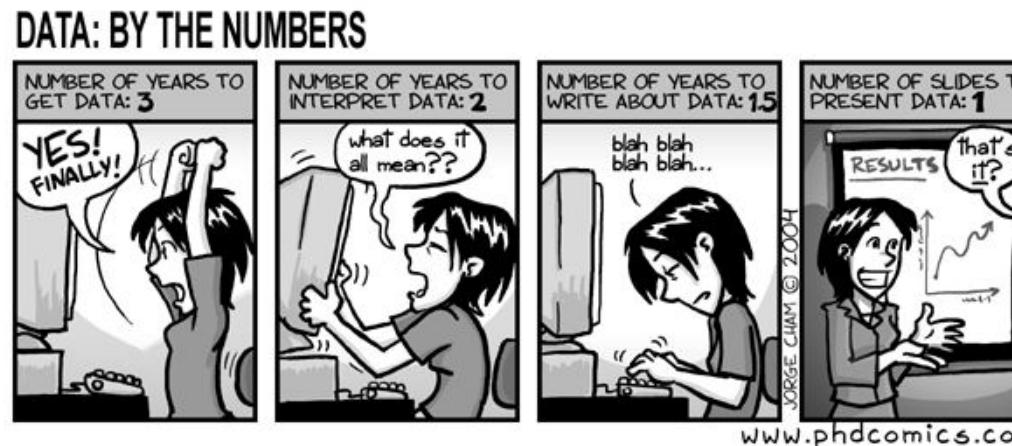
$$\frac{\partial P}{\partial t} = D \frac{\partial^2 P}{\partial z^2} + \lambda P(1 - \frac{P}{K})$$



What's your name & data science (or R) experience?

# What's data science?

- combining math & statistics, programming and machine learning to explore any kind of data
- data is often noisy or unstructured → cleaning is big part of the job
- buzzwords like big data



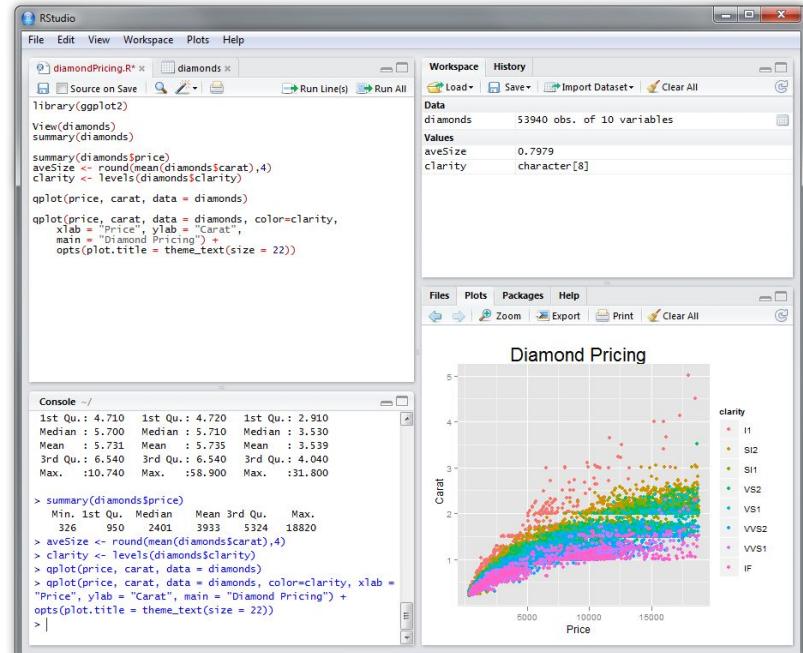
# What's the plan for today?

- short introduction into the R scripting language
  - all material on github:  
<https://github.com/robertladwig/introdatascience>
  - GitHub: uses Git (version control), fantastic for coding and collaborations
- hands-on coding to introduce you to data analysis
- **at the end:** able to know basic R commands, load data, analyze data, plot data



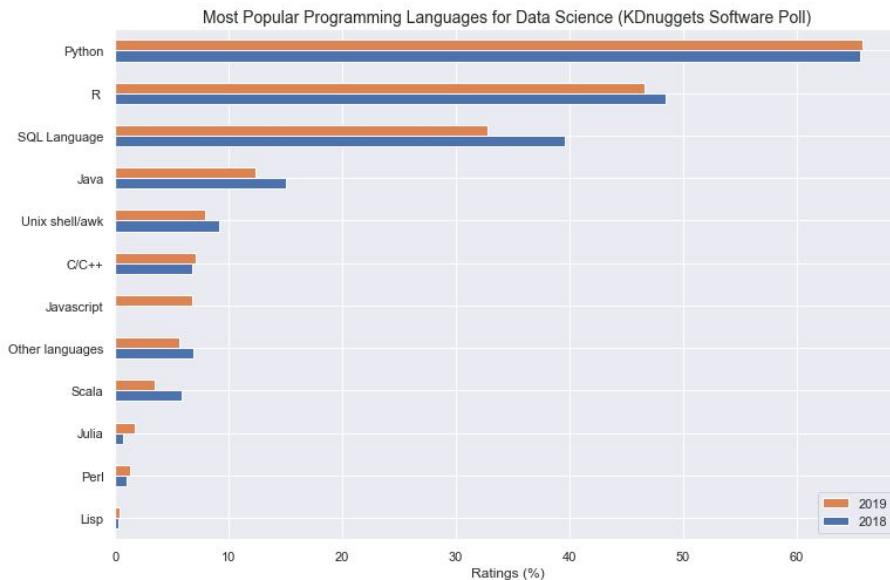
# Install R and RStudio

- Install R from here (language):  
<https://www.r-project.org/>
- Install RStudio (editor and GUI):  
<https://rstudio.com/products/rstudio/download/#download>



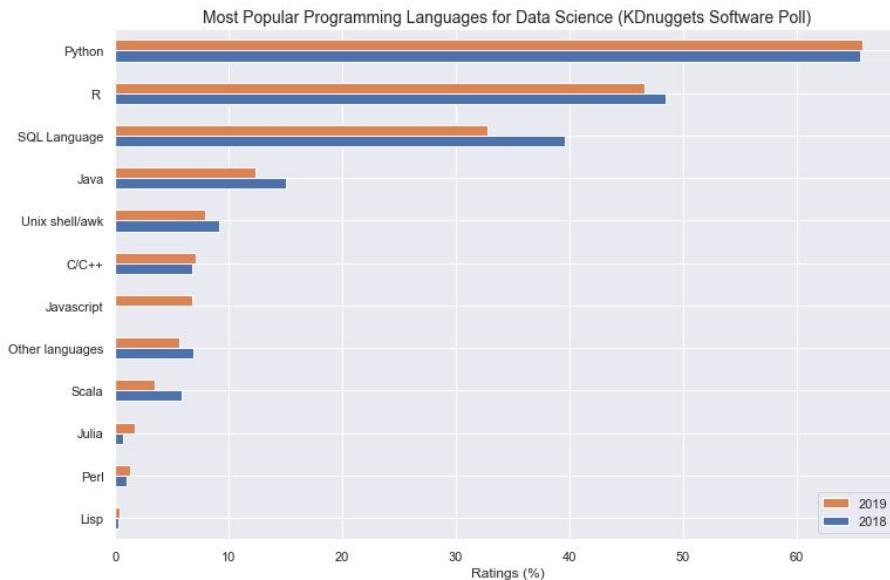
# What's R?

- FREE and OPEN SOURCE statistical and computational software



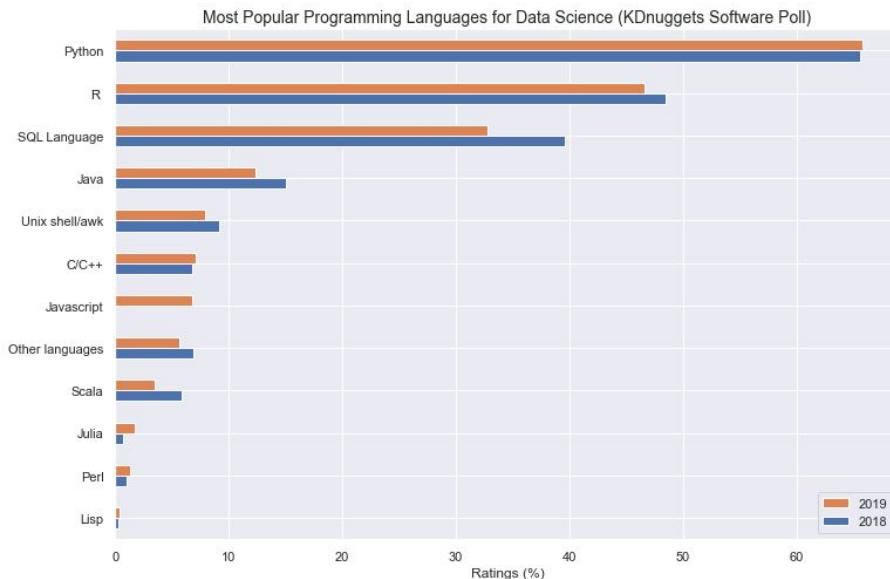
# What's R?

- FREE and OPEN SOURCE statistical and computational software
- big community: easy to find solutions and troubleshooting online



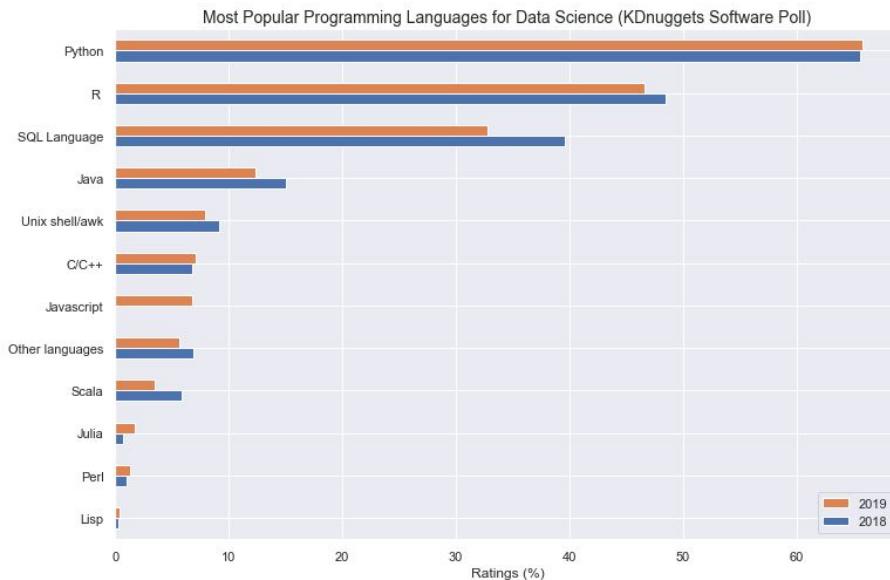
# What's R?

- FREE and OPEN SOURCE statistical and computational software
- big community: easy to find solutions and troubleshooting online
- widely used in sciences (esp. statistics and visualization) and rapidly growing in popularity



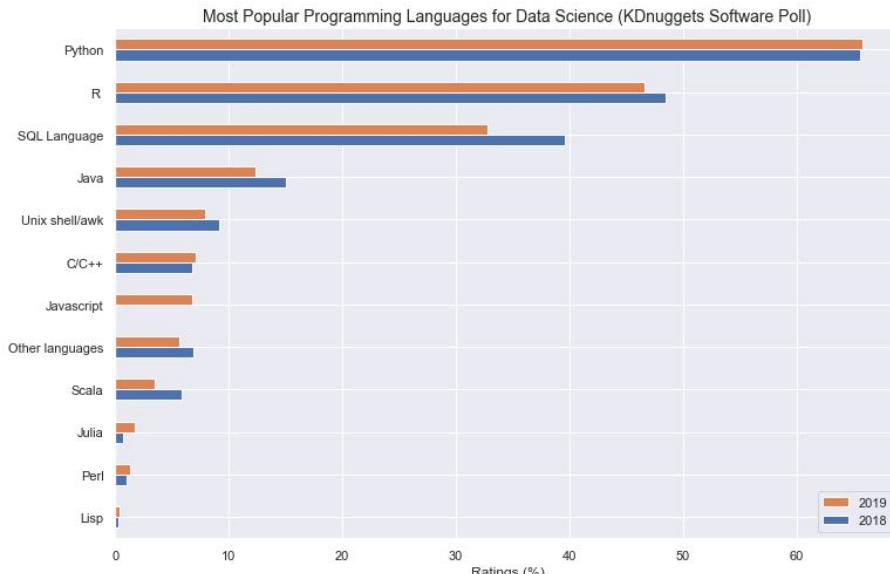
# What's R?

- FREE and OPEN SOURCE statistical and computational software
- big community: easy to find solutions and troubleshooting online
- widely used in sciences (esp. statistics and visualization) and rapidly growing in popularity
- can handle more advanced computations, statistical analyses and bigger data files than Excel

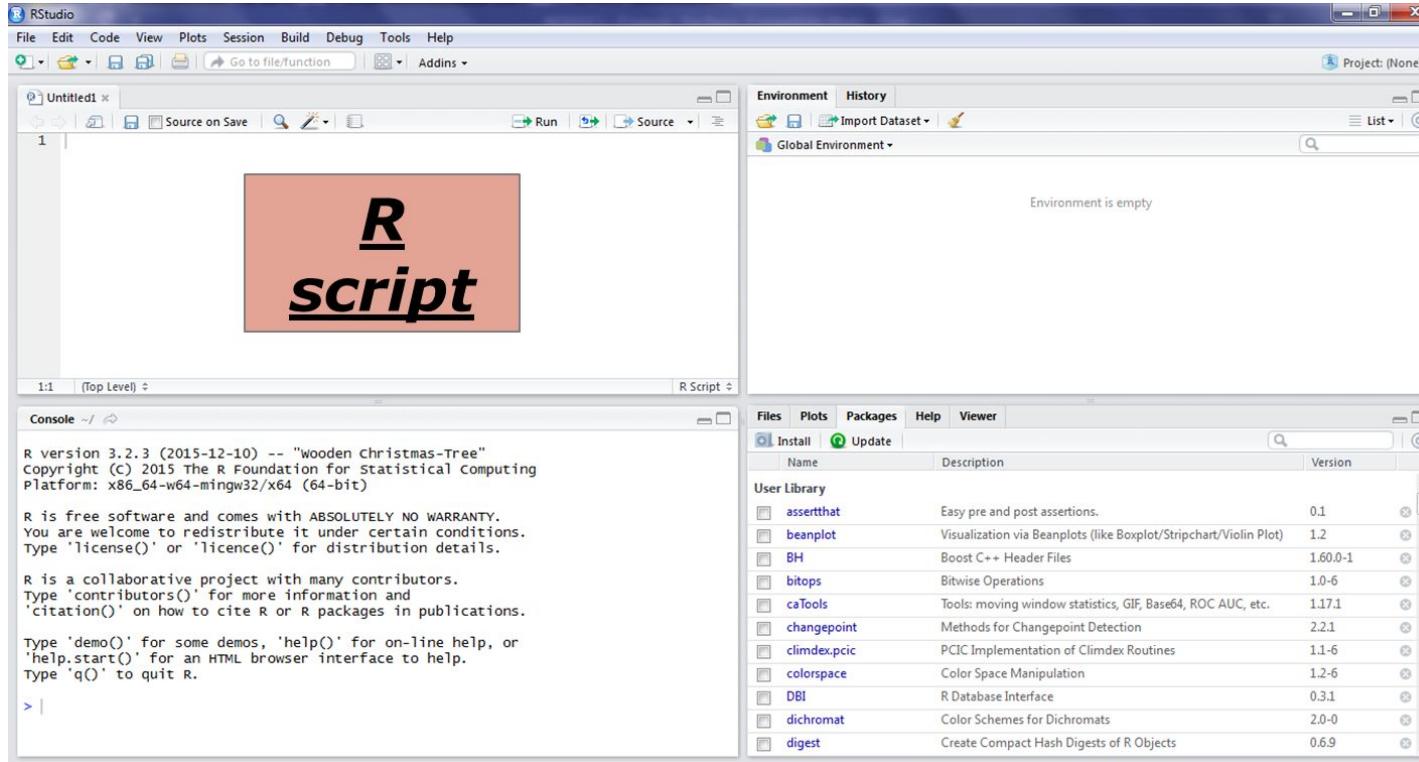


# What's R?

- FREE and OPEN SOURCE statistical and computational software
- big community: easy to find solutions and troubleshooting online
- widely used in sciences (esp. statistics and visualization) and rapidly growing in popularity
- can handle more advanced computations, statistical analyses and bigger data files than Excel
- lots of styles for coding



# Open RStudio on your laptop now



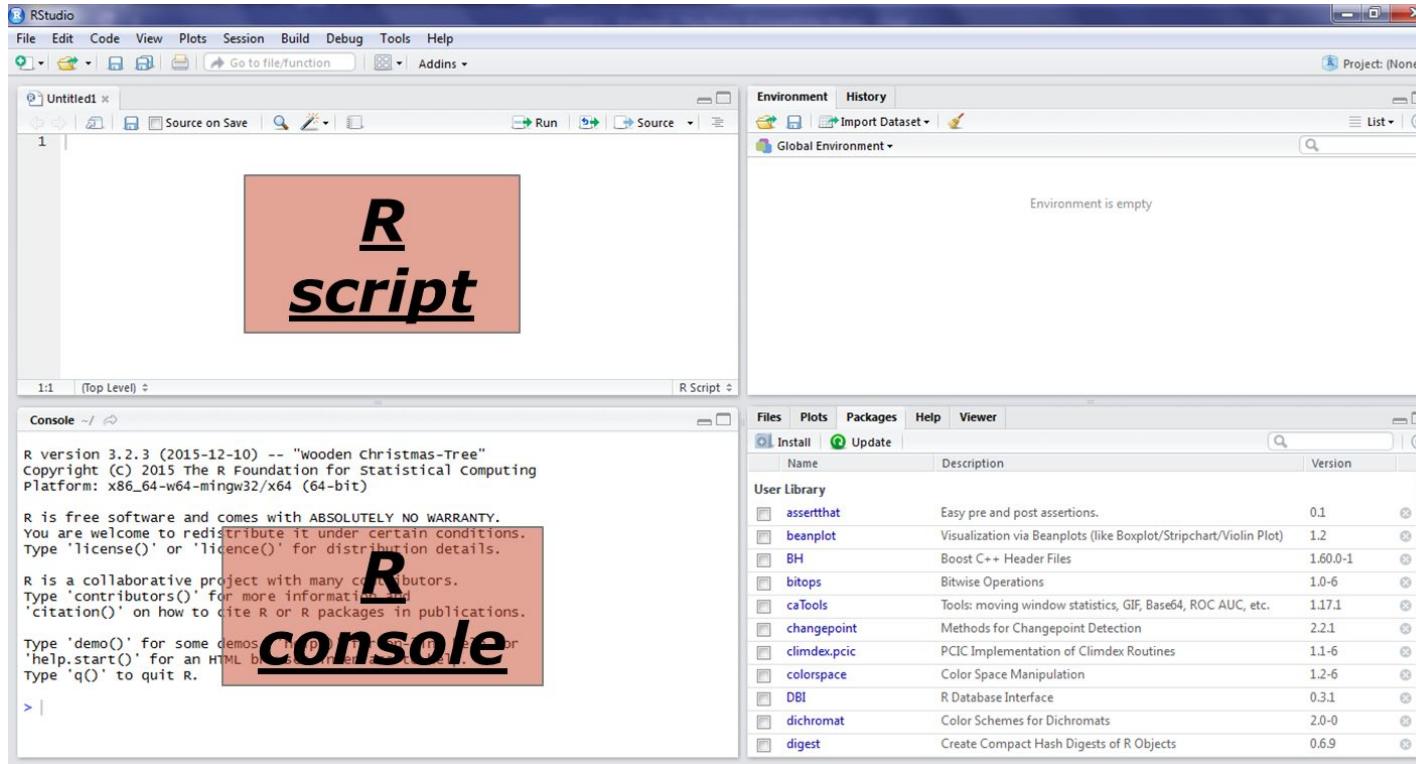
R script:

write and edit code

color-coding for easier identification

automatically fills in parentheses and quotations

# Open RStudio on your laptop now

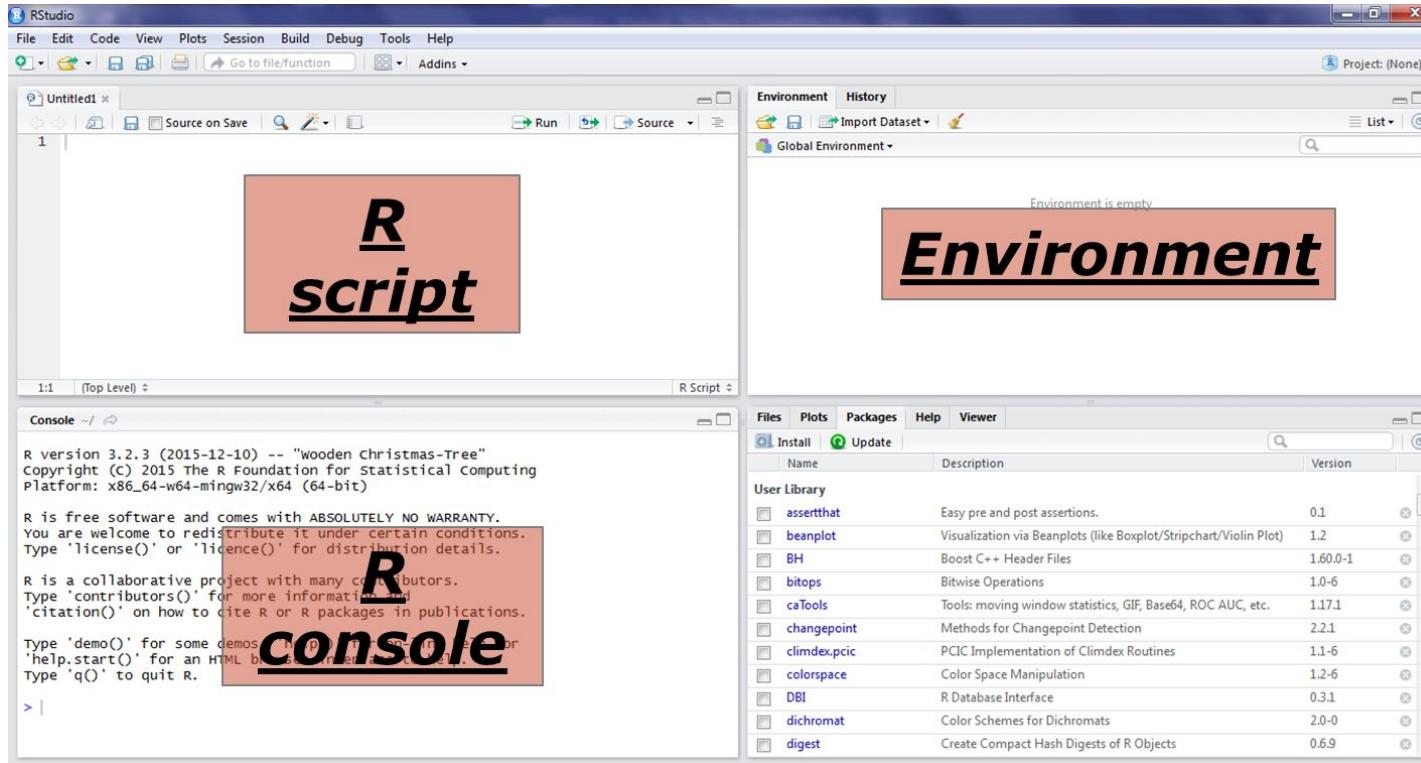


**R console:**

commands are run and results appear

> means 'ready to work'

# Open RStudio on your laptop now

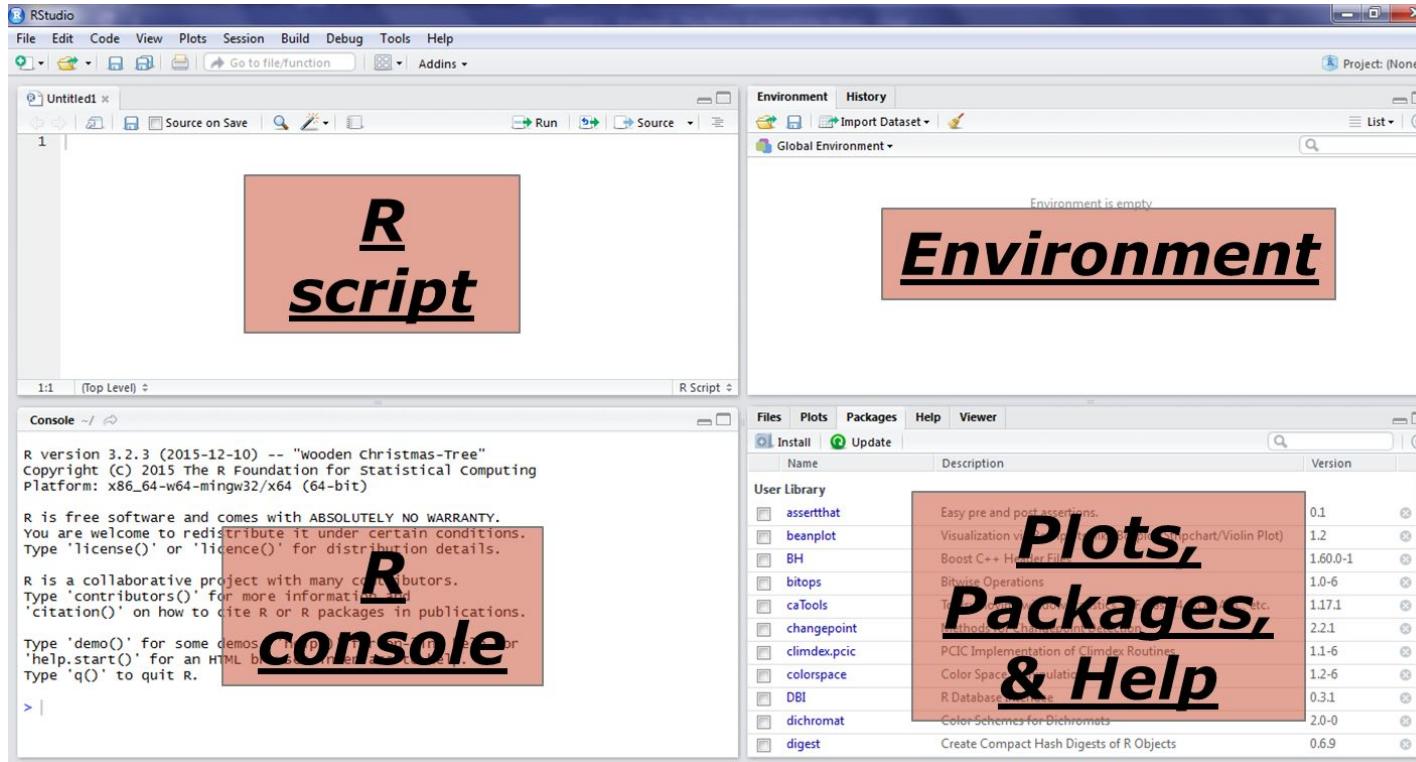


**Environment:**

see which objects were created

type of objects (double, integers, etc.), size and dimensions

# Open RStudio on your laptop now



Plots/Packages/  
Help:

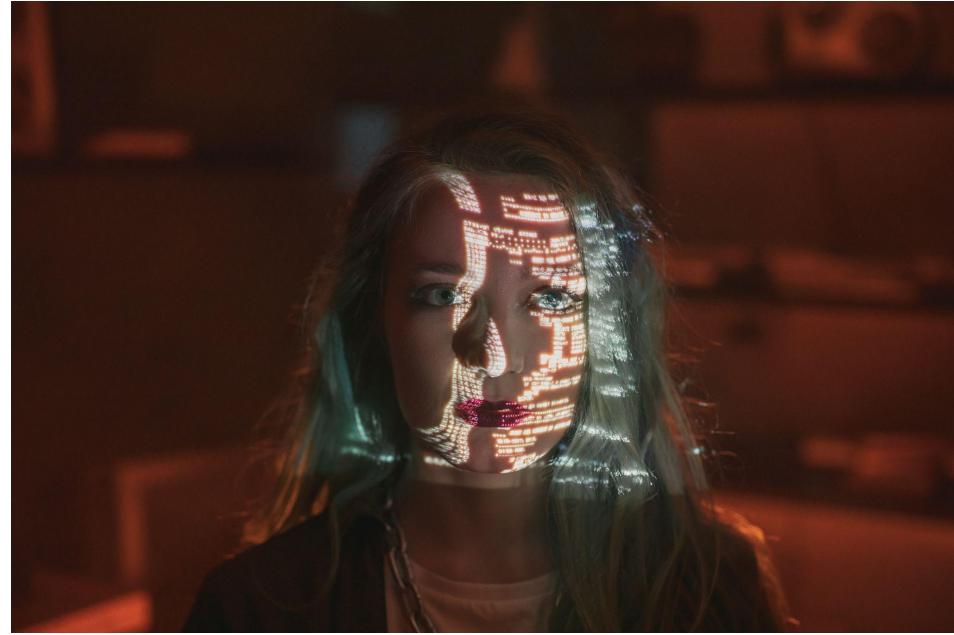
plots will appear

see which  
packages are  
installed

help functions

# What are packages?

- bundles of tools and functions that others have developed to be used in R



# What are packages?

- bundles of tools and functions that others have developed to be used in R
- grouped to specific types of functions, analyses or datasets



# What are packages?

- bundles of tools and functions that others have developed to be used in R
- grouped to specific types of functions, analyses or datasets
- you can also create your own packages for specific tasks!



# What are packages?

- bundles of tools and functions that others have developed to be used in R
- grouped to specific types of functions, analyses or datasets
- you can also create your own packages for specific tasks!
- currently, above 16,000 packages are available



# Install one package

The screenshot shows the RStudio interface. The 'Tools' menu is open, with 'Install Packages...' highlighted. The main workspace shows a script named 'Final\_Script\_NE-GLEON.R' containing R code for creating barplots and calculating mean temperatures. The 'Environment' tab of the sidebar is selected, showing the 'User Library' with a list of installed packages and their versions.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Import Dataset
Install Packages...
Check for Package Updates...
Version Control
Shell...
Addins
Keyboard Shortcuts Help Alt+Shift+K
Modify Keyboard Shortcuts...
Project Options...
Global Options...

Final_Script_NE-GLEON.R*
160 barplot(surface$Temp[1:5])
161
162 mean.temp <- aggregate(lake.temp$Temp, by = list(lake = lake$lake), FUN = mean)
163 barplot(mean.temp[,2], names=mean.t
164
165 stdev.temp <- aggregate(lake.temp$Temp, by = list(lake = lake$lake), FUN = sd)
166 help(arrows)
167
168 barplot(mean.temp[,2], names=mean.t
169   ylim=c(0,25))
170 centers=barplot(mean.temp[,2], names=mean.t
171   ylim=c(0,25))
172 arrows(x0=centers,
173   y0=mean.temp[,2]+stdev.temp[,2],
174   y1=mean.temp[,2]-stdev.temp[,2],
175   code=3, angle=90, length=0.05)
176
177 #####
178 ## PART 3 - Limnology in R ##
180 < [     ] R Script
184:1 ~/Miami/R Workshop/ 
> install.packages("rLakeAnalyzer")
Installing package into 'C:/Users/Rachel Pilla/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/rLakeAnalyzer_1.7.6
.zip'
Content type 'application/zip' length 516163 bytes (504 KB)
downloaded 504 KB

package 'rLakeAnalyzer' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:/Users/Rachel Pilla/AppData/Local/Temp/RtmpmmKDw7/downloaded_packages
> |

```

we need:  
tidyverse

# Install one package

The screenshot shows the RStudio interface with the 'Final-Script\_NE-GLEON.R' script open in the code editor. The script contains R code for creating bar plots of lake temperature data across different depths. A modal dialog box titled 'Install Packages' is overlaid on the interface, prompting the user to install packages from CRAN. The 'Install from:' dropdown is set to 'Repository (CRAN, CRANExtra)'. The 'Packages' input field contains 'rLake' and 'rLakeAnalyzer'. The 'Install dependencies' checkbox is checked. In the background, the 'Environment' tab of the tools panel is visible, showing a list of installed packages like assertthat, hexbin, and gridExtra.

```
Final-Script_NE-GLEON.R
160 barplot(surface$Temp[1:5])
161
162 mean.temp <- aggregate(lake.temp$temp, by=list(lake.temp$Depth), FUN=mean)
163 barplot(mean.temp[,2], names=mean.temp[,1], xlab="Depth (m)", ylab="Mean Temperature (C)", ylim=c(0,25))
164
165 stdev.temp <- aggregate(lake.temp$temp, by=list(lake.temp$Depth), FUN=sd)
166 help(arrows)
167
168 barplot(mean.temp[,2], names=mean.temp[,1], xlab="Depth (m)", ylab="Mean Temperature (C)", ylim=c(0,25))
169 centers<-barplot(mean.temp[,2], names=mean.temp[,1], xlab="Depth (m)", ylab="Mean Temperature (C)", ylim=c(0,25))
170
171 arrows(x0=centers,
172         y0=mean.temp[,2]+stdev.temp[,2],
173         y1=mean.temp[,2]-stdev.temp[,2],
174         code=3, angle=90, length=0.05)
175
176 #####
177 ## PART 3 - Limnology in R ##
178 #####
179
180
181
182
183 < [Top Level] <
184:1 ~/Miami/R Workshop/
```

Console ~/Miami/R Workshop/

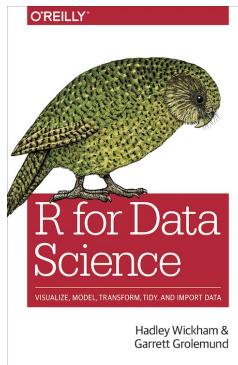
```
> barplot(mean.temp[,2], names=mean.temp[,1], xlab="Depth (m)", ylab="Mean Temperature (C)", ylim=c(0,25))
> centers<-barplot(mean.temp[,2], names=mean.temp[,1], xlab="Depth (m)", ylab="Mean Temperature (C)", ylim=c(0,25))
> arrows(x0=centers,
+         y0=mean.temp[,2]+stdev.temp[,2],
+         y1=mean.temp[,2]-stdev.temp[,2],
+         code=3, angle=90, length=0.05)
> |
```

we need:  
tidyverse

# Tidyverse

R for Data Science  
(Wickham &  
Grolemund):

[https://r4ds.had.co  
.nz](https://r4ds.had.co.nz)



A screenshot of the tidyverse.org website. The header reads 'Tidyverse' and 'tidyverse.org'. The main content area features a large graphic of the tidyverse hexagonal logo, which is a cluster of hexagons containing various R package icons like dplyr, ggplot2, and purrr. To the right of the logo, the text reads 'R packages for data science' and 'The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.' Below this is a section titled 'Install the complete tidyverse with:' containing the code 'install.packages("tidyverse")'.

# Installing a package

- packages only need to be installed ONCE
- but, if you want to use it you'll need to load it:

```
library(package name)
```



Hadley Wickham @hadleywickham

Replying to [@ijlyttle](#)

[@ijlyttle](#) a package is like a book, a library is like a library; you use library() to check a package out of the library [#rsats](#)

8:34 AM · Dec 8, 2014 · Echofon

# Package information

The screenshot shows the RStudio interface. On the left, the code editor displays R script code for data manipulation. On the right, the Environment pane is open, showing the 'Packages' tab of the Packages panel. The 'dplyr' package is selected in the list, which is highlighted in the console below.

Code in the script editor:

```
197 # reformatting from wide to long using "gather"
198
199 TempGather <- temp %>%
200   gather(key = "Depth", value = "Temperature", SurfaceWaterTemp, Bot
201
202
203 # grouping data and reformatting data with dplyr
204
205 ?InsectSprays
206 head(InsectSprays)
207 str(InsectSprays)
208
209 InsectSummary <- InsectSprays %>%
210   group_by(spray) %>%
211   summarize(MeanCount = mean(count),
212             StDevCount = sd(count))
213
214
215
216
217
218
219
220 < [REDACTED] >
221
222 < [REDACTED] >
223
224 < [REDACTED] >
225
226 < [REDACTED] >
227
228 < [REDACTED] >
229
230 < [REDACTED] >
231
232 < [REDACTED] >
233
234 < [REDACTED] >
235
236 < [REDACTED] >
237
238 < [REDACTED] >
239
240 < [REDACTED] >
241
242 < [REDACTED] >
243
244 < [REDACTED] >
245
246 < [REDACTED] >
247
248 < [REDACTED] >
249
250 < [REDACTED] >
251
252 < [REDACTED] >
253
254 < [REDACTED] >
255
256 < [REDACTED] >
257
258 < [REDACTED] >
259
260 < [REDACTED] >
261
262 < [REDACTED] >
263
264 < [REDACTED] >
265
266 < [REDACTED] >
267
268 < [REDACTED] >
269
270 < [REDACTED] >
271
272 < [REDACTED] >
273
274 < [REDACTED] >
275
276 < [REDACTED] >
277
278 < [REDACTED] >
279
280 < [REDACTED] >
281
282 < [REDACTED] >
283
284 < [REDACTED] >
285
286 < [REDACTED] >
287
288 < [REDACTED] >
289
290 < [REDACTED] >
291
292 < [REDACTED] >
293
294 < [REDACTED] >
295
296 < [REDACTED] >
297
298 < [REDACTED] >
299
300 < [REDACTED] >
301
302 < [REDACTED] >
303
304 < [REDACTED] >
305
306 < [REDACTED] >
307
308 < [REDACTED] >
309
310 < [REDACTED] >
311
312 < [REDACTED] >
313
314 < [REDACTED] >
315
316 < [REDACTED] >
317
318 < [REDACTED] >
319
320 < [REDACTED] >
321
322 < [REDACTED] >
323
324 < [REDACTED] >
325
326 < [REDACTED] >
327
328 < [REDACTED] >
329
330 < [REDACTED] >
331
332 < [REDACTED] >
333
334 < [REDACTED] >
335
336 < [REDACTED] >
337
338 < [REDACTED] >
339
340 < [REDACTED] >
341
342 < [REDACTED] >
343
344 < [REDACTED] >
345
346 < [REDACTED] >
347
348 < [REDACTED] >
349
350 < [REDACTED] >
351
352 < [REDACTED] >
353
354 < [REDACTED] >
355
356 < [REDACTED] >
357
358 < [REDACTED] >
359
360 < [REDACTED] >
361
362 < [REDACTED] >
363
364 < [REDACTED] >
365
366 < [REDACTED] >
367
368 < [REDACTED] >
369
370 < [REDACTED] >
371
372 < [REDACTED] >
373
374 < [REDACTED] >
375
376 < [REDACTED] >
377
378 < [REDACTED] >
379
380 < [REDACTED] >
381
382 < [REDACTED] >
383
384 < [REDACTED] >
385
386 < [REDACTED] >
387
388 < [REDACTED] >
389
390 < [REDACTED] >
391
392 < [REDACTED] >
393
394 < [REDACTED] >
395
396 < [REDACTED] >
397
398 < [REDACTED] >
399
400 < [REDACTED] >
401
402 < [REDACTED] >
403
404 < [REDACTED] >
405
406 < [REDACTED] >
407
408 < [REDACTED] >
409
410 < [REDACTED] >
411
412 < [REDACTED] >
413
414 < [REDACTED] >
415
416 < [REDACTED] >
417
418 < [REDACTED] >
419
420 < [REDACTED] >
421
422 < [REDACTED] >
423
424 < [REDACTED] >
425
426 < [REDACTED] >
427
428 < [REDACTED] >
429
430 < [REDACTED] >
431
432 < [REDACTED] >
433
434 < [REDACTED] >
435
436 < [REDACTED] >
437
438 < [REDACTED] >
439
440 < [REDACTED] >
441
442 < [REDACTED] >
443
444 < [REDACTED] >
445
446 < [REDACTED] >
447
448 < [REDACTED] >
449
450 < [REDACTED] >
451
452 < [REDACTED] >
453
454 < [REDACTED] >
455
456 < [REDACTED] >
457
458 < [REDACTED] >
459
460 < [REDACTED] >
461
462 < [REDACTED] >
463
464 < [REDACTED] >
465
466 < [REDACTED] >
467
468 < [REDACTED] >
469
470 < [REDACTED] >
471
472 < [REDACTED] >
473
474 < [REDACTED] >
475
476 < [REDACTED] >
477
478 < [REDACTED] >
479
480 < [REDACTED] >
481
482 < [REDACTED] >
483
484 < [REDACTED] >
485
486 < [REDACTED] >
487
488 < [REDACTED] >
489
490 < [REDACTED] >
491
492 < [REDACTED] >
493
494 < [REDACTED] >
495
496 < [REDACTED] >
497
498 < [REDACTED] >
499
500 < [REDACTED] >
501
502 < [REDACTED] >
503
504 < [REDACTED] >
505
506 < [REDACTED] >
507
508 < [REDACTED] >
509
510 < [REDACTED] >
511
512 < [REDACTED] >
513
514 < [REDACTED] >
515
516 < [REDACTED] >
517
518 < [REDACTED] >
519
520 < [REDACTED] >
521
522 < [REDACTED] >
523
524 < [REDACTED] >
525
526 < [REDACTED] >
527
528 < [REDACTED] >
529
530 < [REDACTED] >
531
532 < [REDACTED] >
533
534 < [REDACTED] >
535
536 < [REDACTED] >
537
538 < [REDACTED] >
539
540 < [REDACTED] >
541
542 < [REDACTED] >
543
544 < [REDACTED] >
545
546 < [REDACTED] >
547
548 < [REDACTED] >
549
550 < [REDACTED] >
551
552 < [REDACTED] >
553
554 < [REDACTED] >
555
556 < [REDACTED] >
557
558 < [REDACTED] >
559
560 < [REDACTED] >
561
562 < [REDACTED] >
563
564 < [REDACTED] >
565
566 < [REDACTED] >
567
568 < [REDACTED] >
569
570 < [REDACTED] >
571
572 < [REDACTED] >
573
574 < [REDACTED] >
575
576 < [REDACTED] >
577
578 < [REDACTED] >
579
580 < [REDACTED] >
581
582 < [REDACTED] >
583
584 < [REDACTED] >
585
586 < [REDACTED] >
587
588 < [REDACTED] >
589
590 < [REDACTED] >
591
592 < [REDACTED] >
593
594 < [REDACTED] >
595
596 < [REDACTED] >
597
598 < [REDACTED] >
599
600 < [REDACTED] >
601
602 < [REDACTED] >
603
604 < [REDACTED] >
605
606 < [REDACTED] >
607
608 < [REDACTED] >
609
610 < [REDACTED] >
611
612 < [REDACTED] >
613
614 < [REDACTED] >
615
616 < [REDACTED] >
617
618 < [REDACTED] >
619
620 < [REDACTED] >
621
622 < [REDACTED] >
623
624 < [REDACTED] >
625
626 < [REDACTED] >
627
628 < [REDACTED] >
629
630 < [REDACTED] >
631
632 < [REDACTED] >
633
634 < [REDACTED] >
635
636 < [REDACTED] >
637
638 < [REDACTED] >
639
640 < [REDACTED] >
641
642 < [REDACTED] >
643
644 < [REDACTED] >
645
646 < [REDACTED] >
647
648 < [REDACTED] >
649
650 < [REDACTED] >
651
652 < [REDACTED] >
653
654 < [REDACTED] >
655
656 < [REDACTED] >
657
658 < [REDACTED] >
659
660 < [REDACTED] >
661
662 < [REDACTED] >
663
664 < [REDACTED] >
665
666 < [REDACTED] >
667
668 < [REDACTED] >
669
670 < [REDACTED] >
671
672 < [REDACTED] >
673
674 < [REDACTED] >
675
676 < [REDACTED] >
677
678 < [REDACTED] >
679
680 < [REDACTED] >
681
682 < [REDACTED] >
683
684 < [REDACTED] >
685
686 < [REDACTED] >
687
688 < [REDACTED] >
689
690 < [REDACTED] >
691
692 < [REDACTED] >
693
694 < [REDACTED] >
695
696 < [REDACTED] >
697
698 < [REDACTED] >
699
700 < [REDACTED] >
701
702 < [REDACTED] >
703
704 < [REDACTED] >
705
706 < [REDACTED] >
707
708 < [REDACTED] >
709
710 < [REDACTED] >
711
712 < [REDACTED] >
713
714 < [REDACTED] >
715
716 < [REDACTED] >
717
718 < [REDACTED] >
719
720 < [REDACTED] >
721
722 < [REDACTED] >
723
724 < [REDACTED] >
725
726 < [REDACTED] >
727
728 < [REDACTED] >
729
730 < [REDACTED] >
731
732 < [REDACTED] >
733
734 < [REDACTED] >
735
736 < [REDACTED] >
737
738 < [REDACTED] >
739
740 < [REDACTED] >
741
742 < [REDACTED] >
743
744 < [REDACTED] >
745
746 < [REDACTED] >
747
748 < [REDACTED] >
749
750 < [REDACTED] >
751
752 < [REDACTED] >
753
754 < [REDACTED] >
755
756 < [REDACTED] >
757
758 < [REDACTED] >
759
760 < [REDACTED] >
761
762 < [REDACTED] >
763
764 < [REDACTED] >
765
766 < [REDACTED] >
767
768 < [REDACTED] >
769
770 < [REDACTED] >
771
772 < [REDACTED] >
773
774 < [REDACTED] >
775
776 < [REDACTED] >
777
778 < [REDACTED] >
779
780 < [REDACTED] >
781
782 < [REDACTED] >
783
784 < [REDACTED] >
785
786 < [REDACTED] >
787
788 < [REDACTED] >
789
790 < [REDACTED] >
791
792 < [REDACTED] >
793
794 < [REDACTED] >
795
796 < [REDACTED] >
797
798 < [REDACTED] >
799
800 < [REDACTED] >
801
802 < [REDACTED] >
803
804 < [REDACTED] >
805
806 < [REDACTED] >
807
808 < [REDACTED] >
809
810 < [REDACTED] >
811
812 < [REDACTED] >
813
814 < [REDACTED] >
815
816 < [REDACTED] >
817
818 < [REDACTED] >
819
820 < [REDACTED] >
821
822 < [REDACTED] >
823
824 < [REDACTED] >
825
826 < [REDACTED] >
827
828 < [REDACTED] >
829
830 < [REDACTED] >
831
832 < [REDACTED] >
833
834 < [REDACTED] >
835
836 < [REDACTED] >
837
838 < [REDACTED] >
839
840 < [REDACTED] >
841
842 < [REDACTED] >
843
844 < [REDACTED] >
845
846 < [REDACTED] >
847
848 < [REDACTED] >
849
850 < [REDACTED] >
851
852 < [REDACTED] >
853
854 < [REDACTED] >
855
856 < [REDACTED] >
857
858 < [REDACTED] >
859
860 < [REDACTED] >
861
862 < [REDACTED] >
863
864 < [REDACTED] >
865
866 < [REDACTED] >
867
868 < [REDACTED] >
869
870 < [REDACTED] >
871
872 < [REDACTED] >
873
874 < [REDACTED] >
875
876 < [REDACTED] >
877
878 < [REDACTED] >
879
880 < [REDACTED] >
881
882 < [REDACTED] >
883
884 < [REDACTED] >
885
886 < [REDACTED] >
887
888 < [REDACTED] >
889
890 < [REDACTED] >
891
892 < [REDACTED] >
893
894 < [REDACTED] >
895
896 < [REDACTED] >
897
898 < [REDACTED] >
899
900 < [REDACTED] >
```

Console output:

```
6 12 A
> str(InsectSprays)
'data.frame': 72 obs. of 2 variables:
 $ count: num 10 7 20 14 14 12 10 23 17 20 ...
 $ spray: Factor w/ 6 levels "A","B","C","D",...: 1 1 1 1 1 1 1 1 1 1 ...
> 
> InsectSummary <- InsectSprays %>%
+   group_by(spray) %>%
+   summarize(MeanCount = mean(count),
+             StDevCount = sd(count))
> |
```

under packages,  
click on **dplyr**

# Package information

The screenshot shows the RStudio interface. On the left, the code editor displays R code for manipulating data frames. In the center, the Help Viewer pane is open, showing the documentation for the 'dplyr' package. The title is 'A Grammar of Data Manipulation'. It includes links to the DESCRIPTION file and user guides. Below this, a section titled 'Help Pages' lists functions starting with 'A', such as 'add\_count', 'add\_tally', 'all\_equal.tbl\_df', 'all\_equal', and 'all\_vars'. The right side of the interface shows the Environment, History, and Connections panes.

this lists all available functions

click on any function to see the help file

# Coding in R

- write commands in the script (upper left panel)
  - save it, edit it, revisit it later, etc.



# Coding in R

- write commands in the script (upper left panel)
  - save it, edit it, revisit it later, etc.
- code NOT automatically run when you hit Enter



# Coding in R

- write commands in the script (upper left panel)
  - save it, edit it, revisit it later, etc.
- code NOT automatically run when you hit Enter
- to run it:
  - “Run” button in upper right corner
  - “CTRL + Enter” (Windows)
  - “Command + Enter” (Mac)



# Coding in R

```
if(result !== false) {  
    $distArray = array();  
    $row = mysqli_fetch_assoc($result);  
    $correctAnswer = $row['Correct'];  
    $distArray['A'] = $row['Anum'];  
    $distArray['B'] = $row['Bnum'];  
    $distArray['C'] = $row['Cnum'];  
    $distArray['D'] = $row['Dnum'];  
    $distArray['Correct'] = $correctAnswer;  
    $distArray['Answer'] = rtrim($row[$correctAnswer], ".");  
    $distArray['Query'] = "SELECT * FROM TechTerms WHERE Date='".date("Y-m-d")."';  
    return $distArray;  
}  
else {  
    $distArray['Error'] = 'Quiz load query failed';  
    return $distArray;  
}
```



- add comments using #
  - additional information
  - will be ignored

# Coding in R



- add comments using #
  - additional information
  - will be ignored
- R is case sensitive
  - “Mean” ≠ “mean”

# Coding in R



- add comments using #
  - additional information
  - will be ignored
- R is case sensitive
  - “Mean” ≠ “mean”
- R doesn't care about spaces and tabs

# Coding in R



- add comments using #
  - additional information
  - will be ignored
- R is case sensitive
  - “Mean” ≠ “mean”
- R doesn't care about spaces and tabs
  - need to close all parentheses and quotations

# Key components

- **functions**: allow you to manipulate data, apply calculations, run statistical analysis, much more!

## Key Components of Code

```
x <- seq(1,10)
```

name of the **function**  
to create a sequence

# Key components

- **functions:** allow you to manipulate data, apply calculations, run statistical analysis, much more!
- **arguments:** defining information for functions, “customize” it

## Key Components of Code

```
x <- seq(1,10)
```

the **arguments** to define the function, to create a sequence from 1 through 10

# Key components

- **functions**: allow you to manipulate data, apply calculations, run statistical analysis, much more!
- **arguments**: defining information for functions, “customize” it
- **objects**: pieces of data saved in R, can be called up, reused and manipulated

## Key Components of Code

```
x <- seq(1,10)
```

name of the **object**  
that saves the results  
of the function in R

# Key components

- **functions**: allow you to manipulate data, apply calculations, run statistical analysis, much more!
- **arguments**: defining information for functions, “customize” it
- **objects**: pieces of data saved in R, can be called up, reused and manipulated

## Key Components of Code

x `<-` seq(1,10)

**assignment operator**  
tells R to save the result  
of the function as the  
named object

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")
  - logical (TRUE/FALSE)

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")
  - logical (TRUE/FALSE)
  - complex (1+4*i*)

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")
  - logical (TRUE/FALSE)
  - complex (1+4i)
- **data structures:**
  - vector (1-D object with same data type)

```
> c(3, 4, 5, 1, 2, 3)
[1] 3 4 5 1 2 3
```

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")
  - logical (TRUE/FALSE)
  - complex (1+4i)
- **data structures:**
  - vector (1-D object with same data type)
  - matrix (2-D object with same data type)

```
> matrix(data = c(3, 4, 5, 1, 2,  
3), nrow = 3, ncol = 2)  
 [,1] [,2]  
 [1,]    3    1  
 [2,]    4    2  
 [3,]    5    3
```

# Types of data

- **objects (data types):**
  - numeric (2.334, 3.14159)
  - integer (2, 2405, 54)
  - characters ("hello world")
  - logical (TRUE/FALSE)
  - complex (1+4i)
- **data structures:**
  - vector (1-D object with same data type)
  - matrix (2-D object with same data type)
  - data frame (2-D object with different data types per column if needed, very useful!)

```
> data.frame('Numbers' = c(3, 4, 5, 1, 2, 3), "Strings" =  
c("Hi", "I", "am", "collecting", "all", "characters"))  
  Numbers      Strings  
1       3          Hi  
2       4            I  
3       5          am  
4       1  collecting  
5       2            all  
6       3 characters
```

# Types of brackets

## round brackets:

- are used for functions
- include arguments

```
seq( . . . )
```

# Types of brackets

## round brackets:

- are used for functions
- include arguments

seq(...)

## squared brackets:

- are used to subset data

```
> x <- data.frame("name" = c("Thing 1",  
"Thing 2"), "Chaos" = c(10, 9))  
> x[1, 1]  
[1] "Thing 1"  
> x[2, 2]  
[1] 9  
> x$name  
[1] "Thing 1" "Thing 2"  
> x$name[2]  
[1] "Thing 2"
```

# Types of brackets

## round brackets:

- are used for functions
- include arguments

seq(...)

## squared brackets:

- are used to subset data

data frame column 1 stores strings

```
> x <- data.frame("name" = c("Thing 1",  
"Thing 2"), "Chaos" = c(10, 9))
```

```
> x[1, 1]  
[1] "Thing 1"
```

data frame column 2 stores  
doubles

```
> x[2, 2]  
[1] 9  
> x$name  
[1] "Thing 1" "Thing 2"  
> x$name[2]  
[1] "Thing 2"
```

# Types of brackets

## round brackets:

- are used for functions
- include arguments

seq(...)

## squared brackets:

- are used to subset data

data frame column 1 stores strings

```
> x <- data.frame("name" = c("Thing 1",  
"Thing 2"), "Chaos" = c(10, 9))
```

```
> x[1, 1] ←  
[1] "Thing 1" ← column 1
```

data frame column 2 stores  
doubles

```
> x[2, 2]  
[1] 9  
> x$name  
[1] "Thing 1" "Thing 2"  
> x$name[2]  
[1] "Thing 2"
```

# Types of brackets

## round brackets:

- are used for functions
- include arguments

seq(...)

## squared brackets:

- are used to subset data

data frame column 1 stores strings

```
> x <- data.frame("name" = c("Thing 1",  
"Thing 2"), "Chaos" = c(10, 9))
```

```
> x[1, 1]
```

```
[1] "Thing 1"
```

column 1

data frame column 2 stores  
doubles

```
> x[2, 2]
```

```
[1] 9
```

dollar sign for indexing columns

```
> x$name
```

```
[1] "Thing 1" "Thing 2"
```

```
> x$name[2]
```

```
[1] "Thing 2"
```

# The wonder of pipes

**pipes %>% are powerful tools for making data manipulation clearer**

- originate from `magrittr` package (included in `tidyverse`)

```
> x
      name Chaos
1 Thing 1     10
2 Thing 2     9
> filter(x, name == "Thing 1")
      name Chaos
1 Thing 1     10
> x %>%
      filter(name == "Thing 1")
      name Chaos
1 Thing 1     10
```

# The wonder of pipes

pipes `%>%` are powerful tools for making data manipulation clearer

- originate from `magrittr` package (included in `tidyverse`)

same code but easier to read

```
> x  
      name Chaos  
1 Thing 1      10  
2 Thing 2      9  
> filter(x, name == "Thing 1")  
      name Chaos  
1 Thing 1      10  
> x %>%  
    filter(name == "Thing 1")  
      name Chaos  
1 Thing 1      10
```

# Live Coding time!

Bird banding data provided by Dr. Dave Russell at Miami University in collaboration with Hueston Woods State Park

