BIOINFORMATICS IN R, THE QUICK AND EASY… AND THE NOT SO QUICK AND EASY

Chase Clark

PhD Candidate

Medicinal Chemistry and Pharmacognosy

University of Illinois at Chicago

@ChasingMicrobes

chasemc.github.io

Photos of slides OK

Main Takeaways

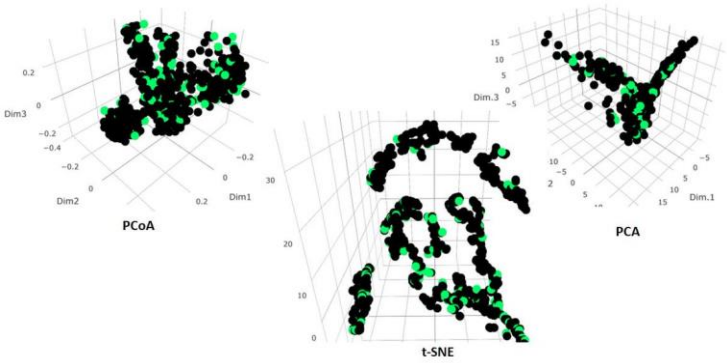
- Creating results is easy

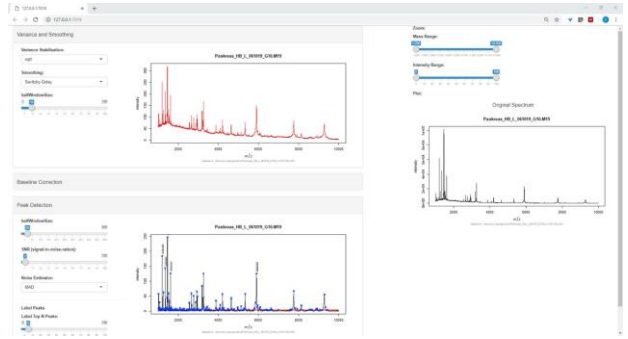
- Using R in production requires more thinking, more work
- Making reproducible results requires more thinking, more work

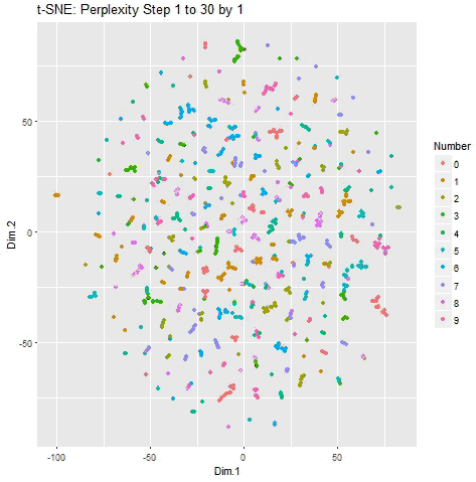
- You should use R in production
- You should make reproducible results

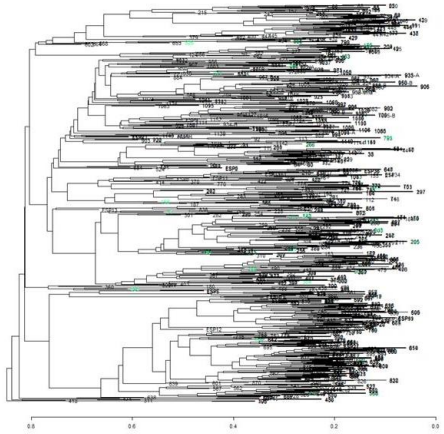
You should always be thinking about the people using your results

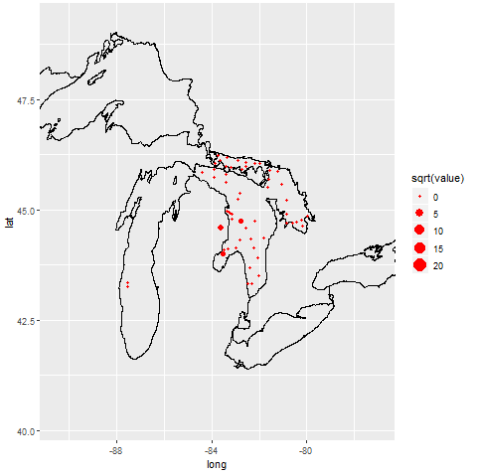
We’re All “Scientists”

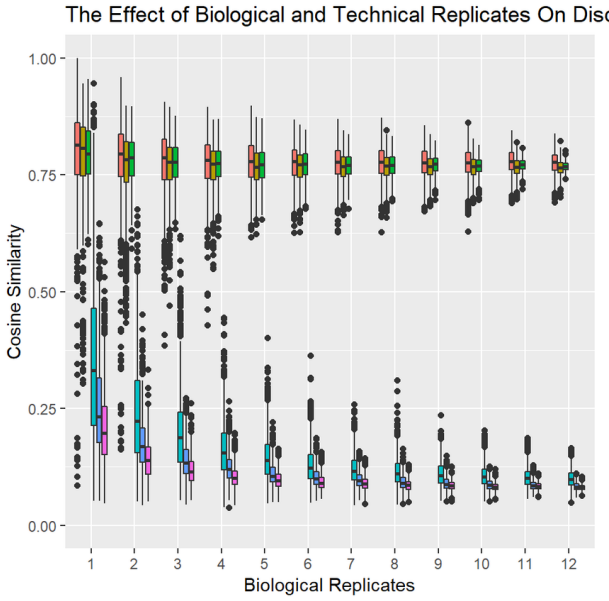


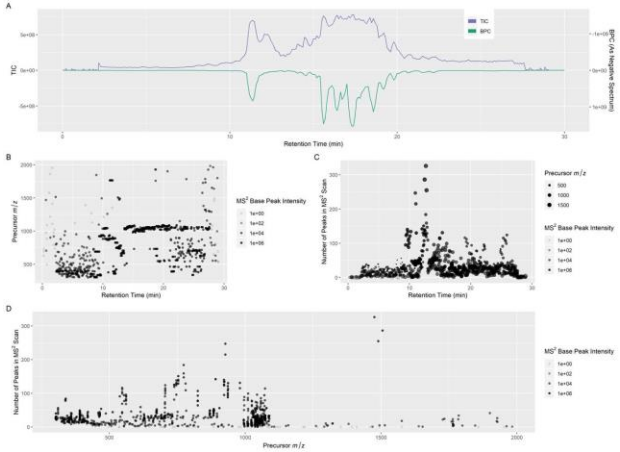












As Bioinformaticians, as Data Scientists We Have One Goal:

Provide insight regarding a question/hypothesis.

Outline

- Making Reproducible Insights Through:
  - Code
  - Reports
  - Applications/Software

- To think about:
  - How do users interact with your results?
    - Technical users (use R)
    - Technical users (don’t use R)
    - Non-technical users (nervous with Excel)

Providing Results as Code

How do users interact with your results?

- Technical users (use R)
  - Sensible variable names
  - Style conventions
  - Be verbose
  - Examples
- Technical users (don't use R)
  - Code Commenting

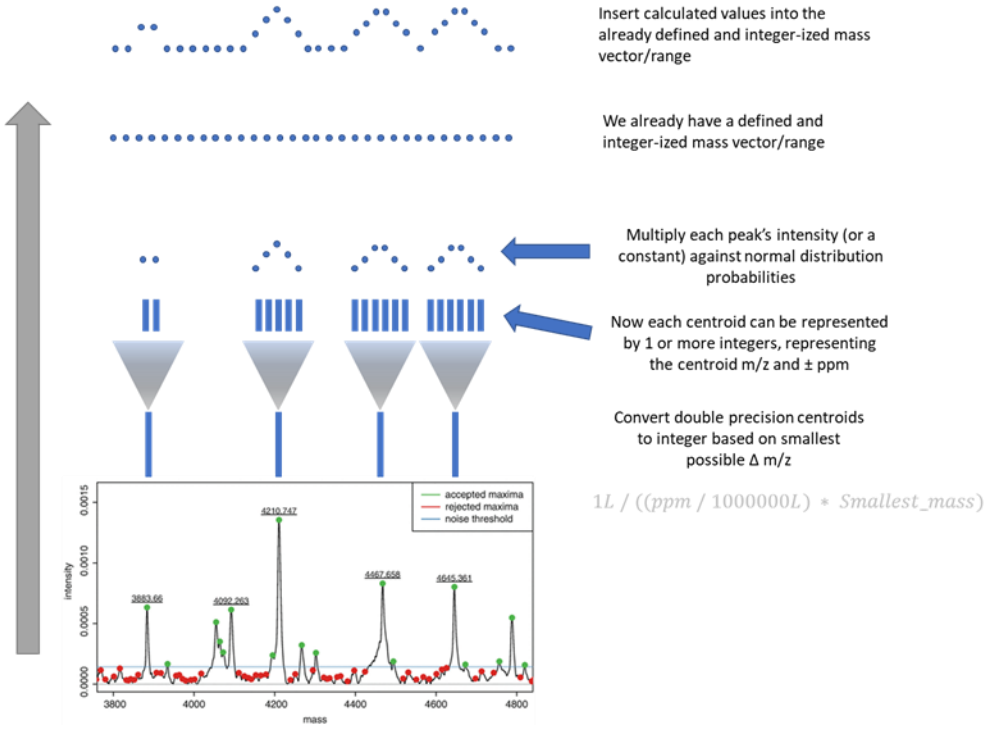
```
106
107 if (length(massList) * vecLength < 4e6) {
108   builtM <- base::matrix(0, nrow = length(massList), ncol = vecLength)
109 } else {
110   builtM <- Matrix::Matrix(0, nrow = length(massList), ncol = vecLength, sp
111 }
112
113 # mm returns a list of lists. each list element contains a list of length 3
114 # centroids are scaled to integer
115 # 1- centroid - ppm
116 # 2- centroid
117 # 3- centroid + ppm
118 scaledMass <- lapply(massList, function(x){
119   integerizedMass <- as.integer(round((x * scaler)))
120   integerizedPpm <- as.integer(round((x * ppm) / 1000000L))
121   list(integerizedMass - integerizedPpm,
122        integerizedMass,
123        integerizedMass + integerizedPpm)
124 })
125
126 # Create a distribution of "intensity" across each ppm range of each peak
127 # loop across all samples (spectra)
128 for (i in seq_along(scaledMass)) {
129
130   # For each centroid, create a sequence of integers from
131   # (centroid mass - ppm) to (centroid mass + ppm)
132   newprofile <- mapply(function(x, y) {x:y},
133                        scaledMass[[i]][[1]],
134                        scaledMass[[i]][[3]],
135                        SIMPLIFY = F)
136
137   # For each centroid we now have a sequence of integers representing uncer
138   # model normal distribution density across each centroid's uncertainty in
139   newIntensity <- mapply(
140     function(x, y, z){
141       round(dnorm(x,
142                   y,
143                   (z - y) / 4),
144             digits = 4)
145   }
```

Note: Some conventions broken here to fit to slide.

Code Isn’t Always Enough

How do users interact with your results?

- Non-technical users (definitely don’t use R)
  - Code diagrams, figures, etc



Reproducible Code Through Testing

- Makes you think about your code
- Prevents errors now, in 6 years
- DO IT!


```
context("test-binnr")

set.seed(42)

p <- c(MALDIquant::createMassPeaks(mass = c(2000, 3000, 4000),
                                   intensity = c(1, 1, 1)),
       MALDIquant::createMassPeaks(mass = c(2000, 3001, 4005),
                                   intensity = rep(1, 2, 3)),
       MALDIquant::createMassPeaks(mass = c(2000.01,3002,10000),
                                   intensity = rep(1, 2, 3)))


binned <- IDBacApp::peakBinner(peakList = p,
                               ppm = 300,
                               massStart = 1000,
                               massEnd = 15000)

test_that("binnR works", {
  expect_known_hash(binned, "f5d2c83a3f")
  expect_warning(IDBacApp::peakBinner(peakList = list("A"),
                                      ppm = 100000))
  expect_identical(suppressWarnings(IDBacApp::peakBinner(peakList = list("A"),
                                      ppm = 100000)),
                   list())
})
```

Making Reproducible Insights Through:

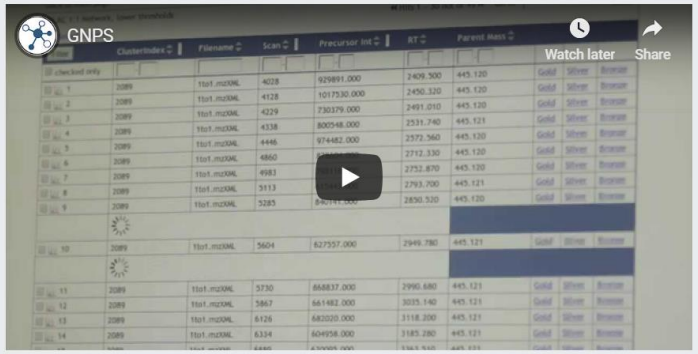
~~Code~~

Reports

Applications/Software

GNPS

GNPS is a web-based mass spectrometry ecosystem that aims to be an open-access knowledge base for community-wide organization and sharing of raw, processed or identified tandem mass (MS/MS) spectrometry data. GNPS aids in identification and discovery throughout the entire life cycle of data; from initial data acquisition/analysis to post publication.



Insight Through Reports

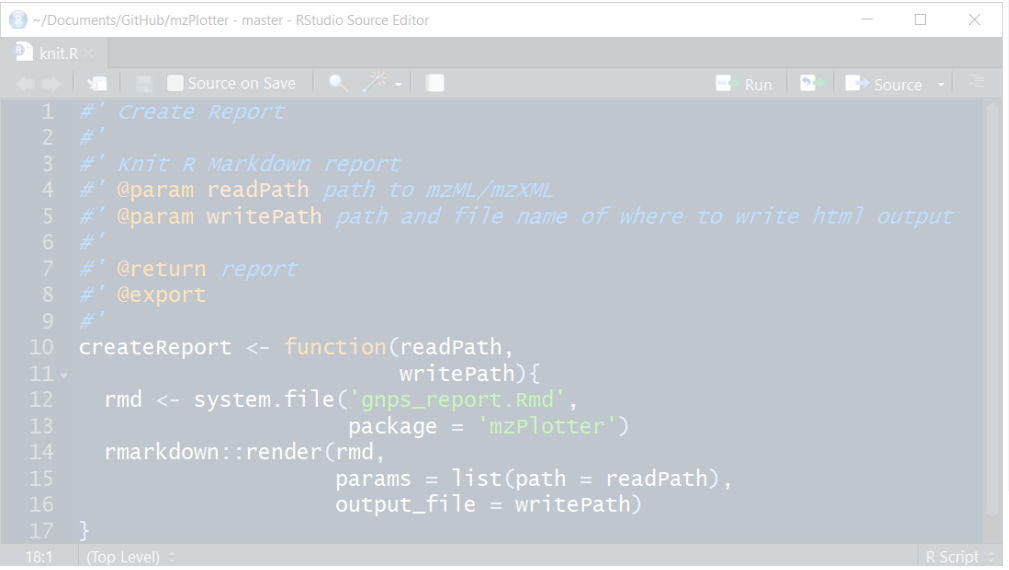
Had: Webservice with detailed analyses (python and bash)

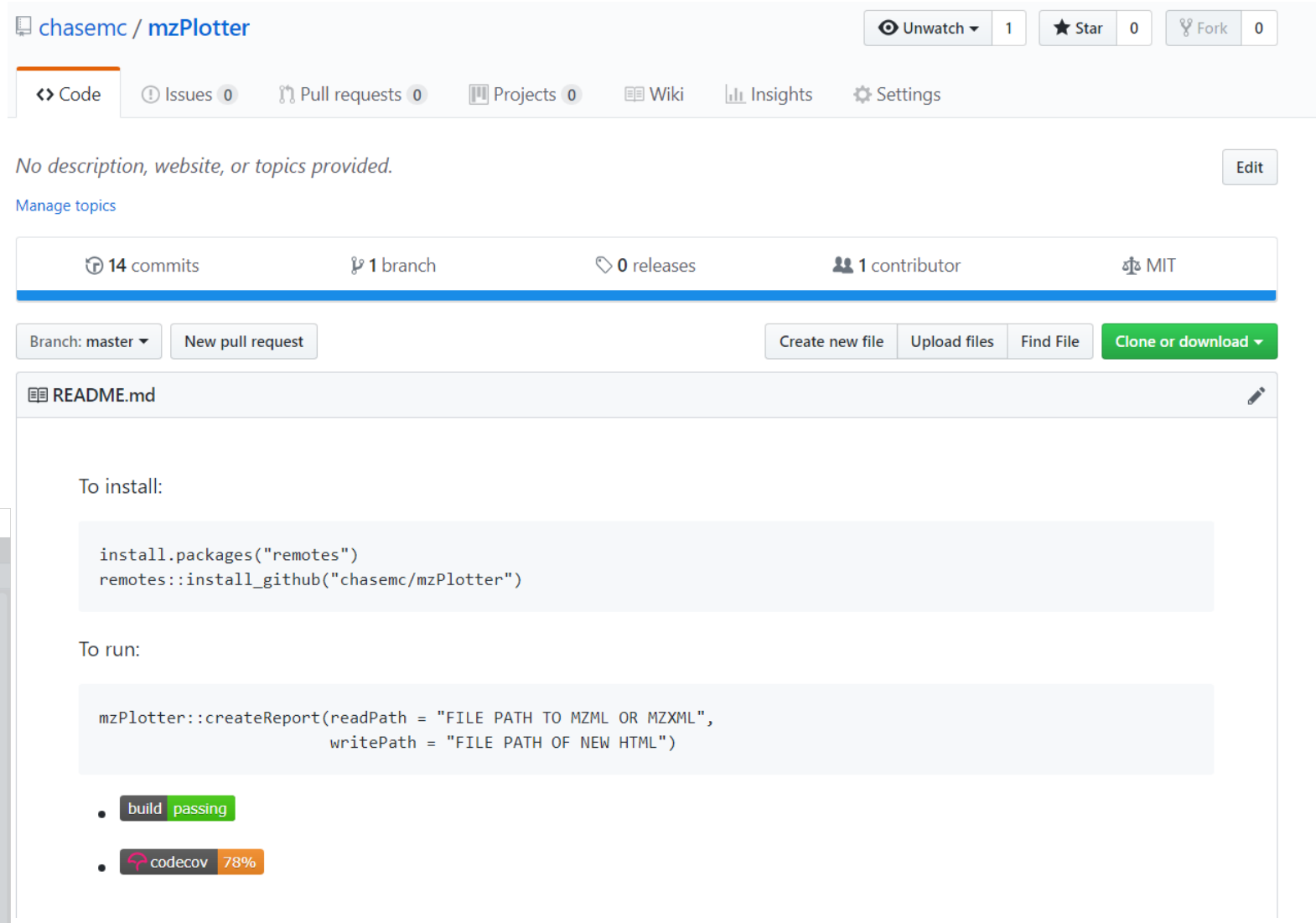
Wanted: Quick summary plots of uploaded raw data

How do users interact with your results?

Technical user that doesn't use R

Simple Functions





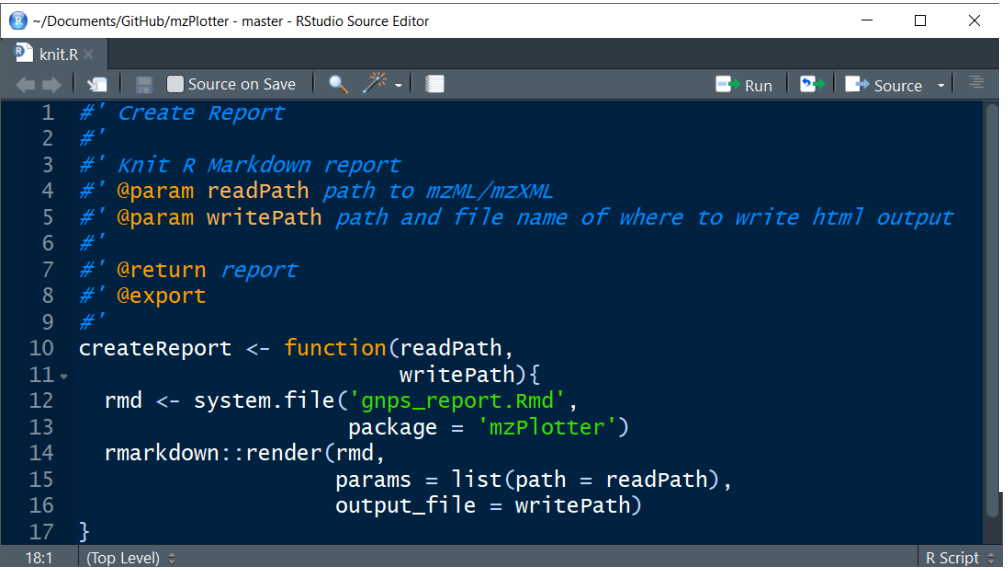
@ChasingMicrobes

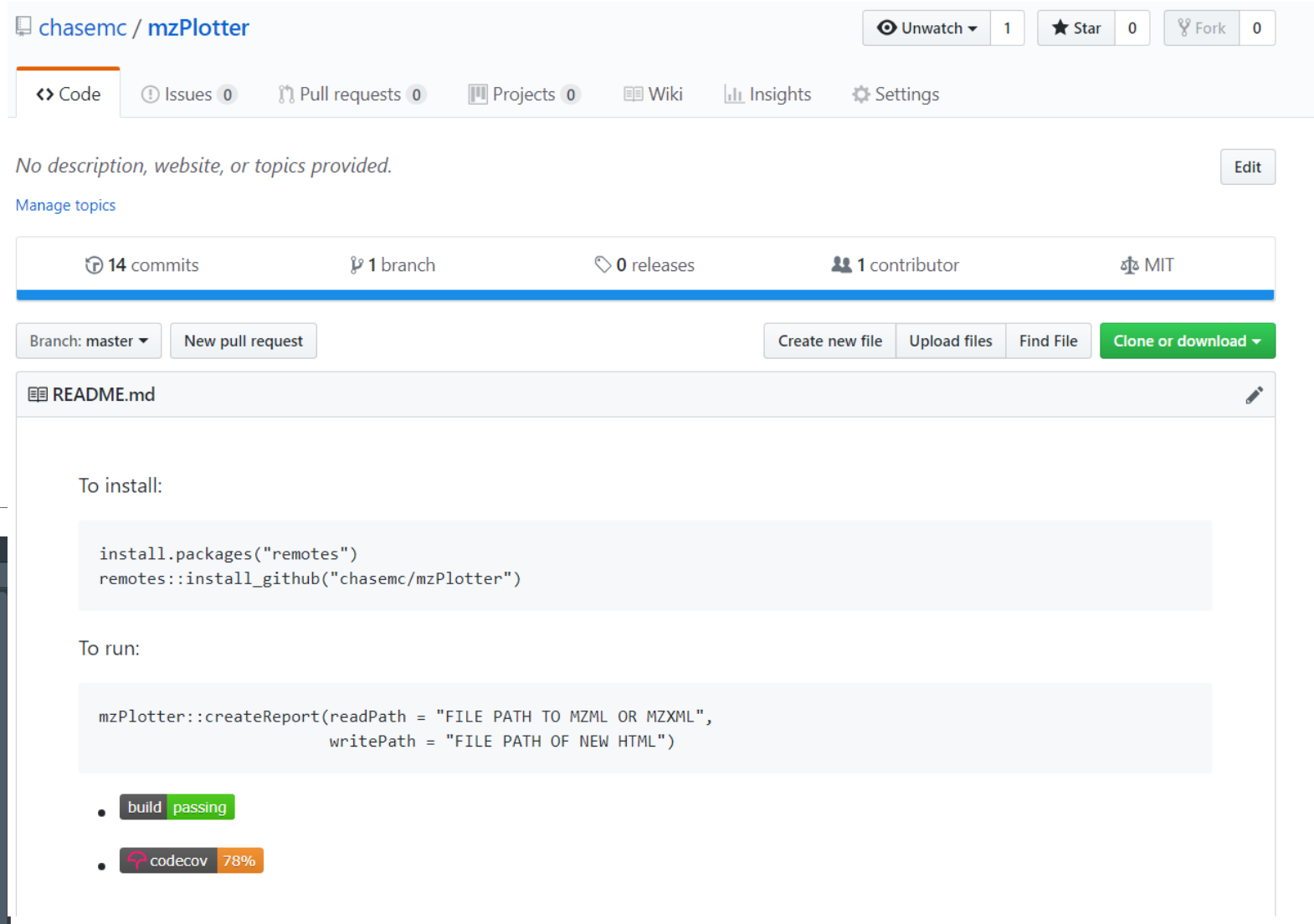
github.com/chasemc/mzPlotter

How do users interact with your results?

Technical user that doesn't use R

Simple Functions

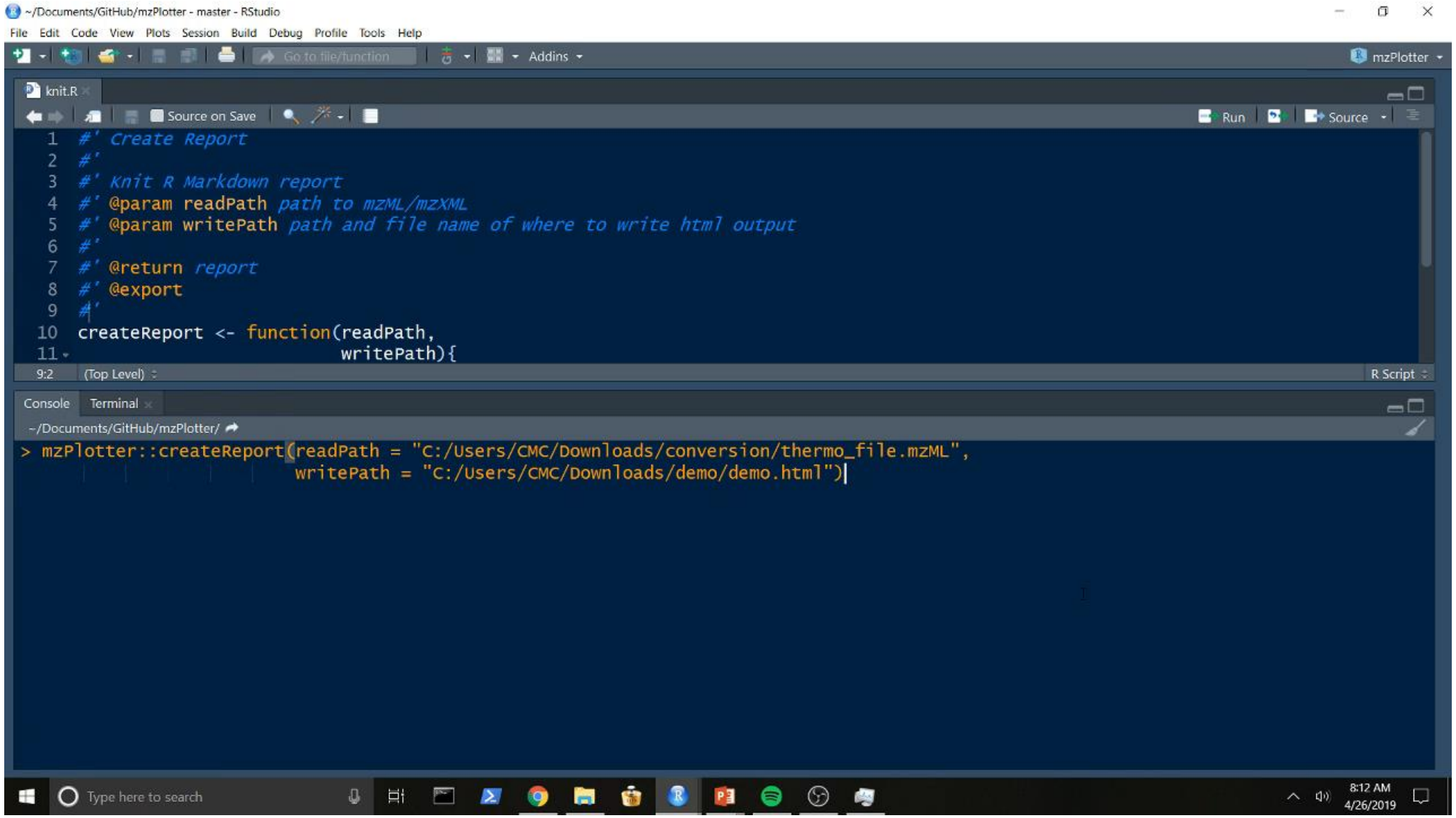




@ChasingMicrobes

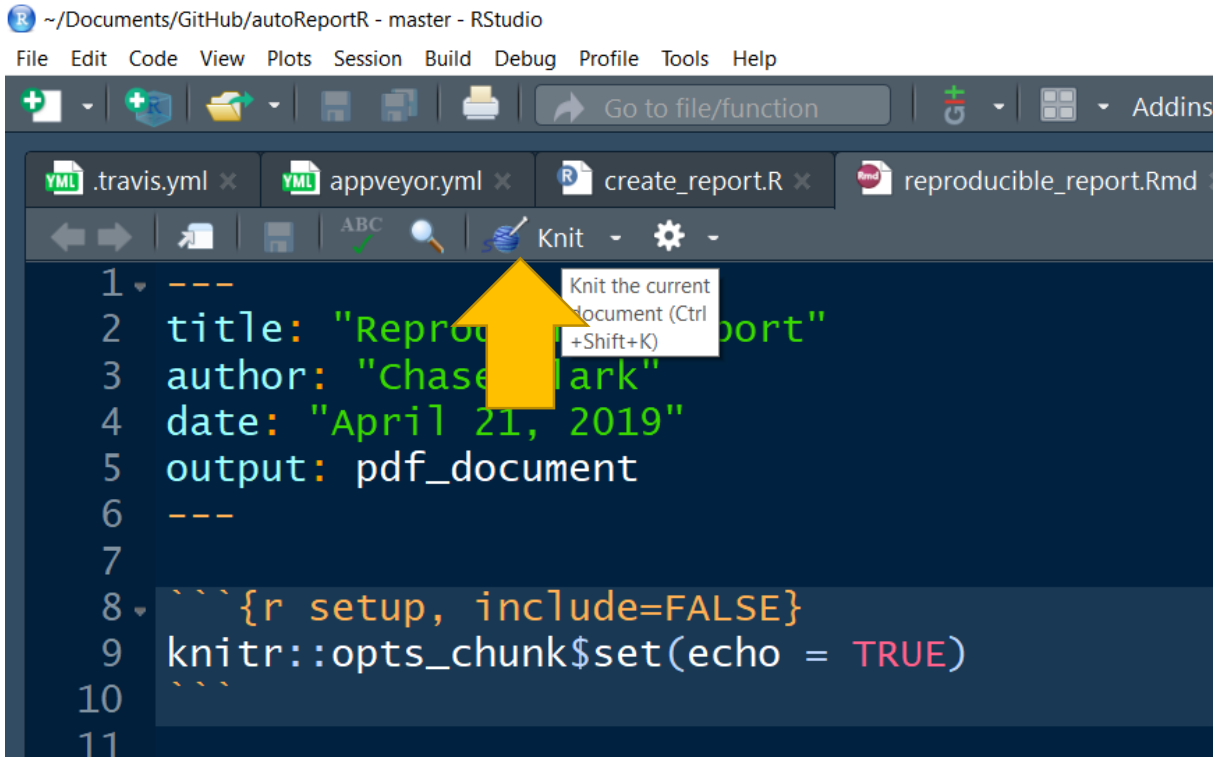
github.com/chasemc/mzPlotter

How do users interact with your results? Anyone



Insight Through Automated Reports

Insight Through Automated Reports



After knitting, how do you associate your report to its code version?

If you have lots of money:

RStudio Connect Pricing

RStudio Connect is the place to publish all the work your teams create in R. Publish Shiny applications, R Markdown reports, APIs, dashboards, plots, and more to one convenient place with a click of a button in your RStudio IDE. Get your developers and publishers started with Connect Base or Standard. Share their work with Named User Packs for only $5 per user per month.
Contact Sales for Named User volume pricing.

RStudio Connect Base	RStudio Connect Standard	RStudio Connect Enterprise*
$14,995 per year	$24,995 per year	$74,995 per year
$62 per user/month	$21 per user/month	$6.25 per user/month

Alternatively…

- You can add continuous deployment to your workflow
- Can be setup as online or offline workflow

Organize analysis as an R package

Report attached to GitHub release

Deploy

Create Release

Continuous Deployment

Travis CI

AppVeyor

Version Control with Git

GitHub

Commits

Continuous Integration

Travis CI

AppVeyor

Organize analysis as an R package

Version Control with Git

GitHub

Commits

Continuous Integration

Travis CI

AppVeyor

Report attached to GitHub release

Deploy

Create Release

Continuous Deployment

Travis CI

AppVeyor

Organize analysis as an R package

Version Control with Git

GitHub

Commits

Continuous Integration

Travis CI

AppVeyor

Report attached to GitHub release

Deploy

Create Release

Continuous Deployment

Travis CI

AppVeyor

Organize analysis as an R package

Report attached to GitHub release

Deploy

Create Release

Continuous Deployment

Travis CI

AppVeyor

Version Control with Git

GitHub

Commits

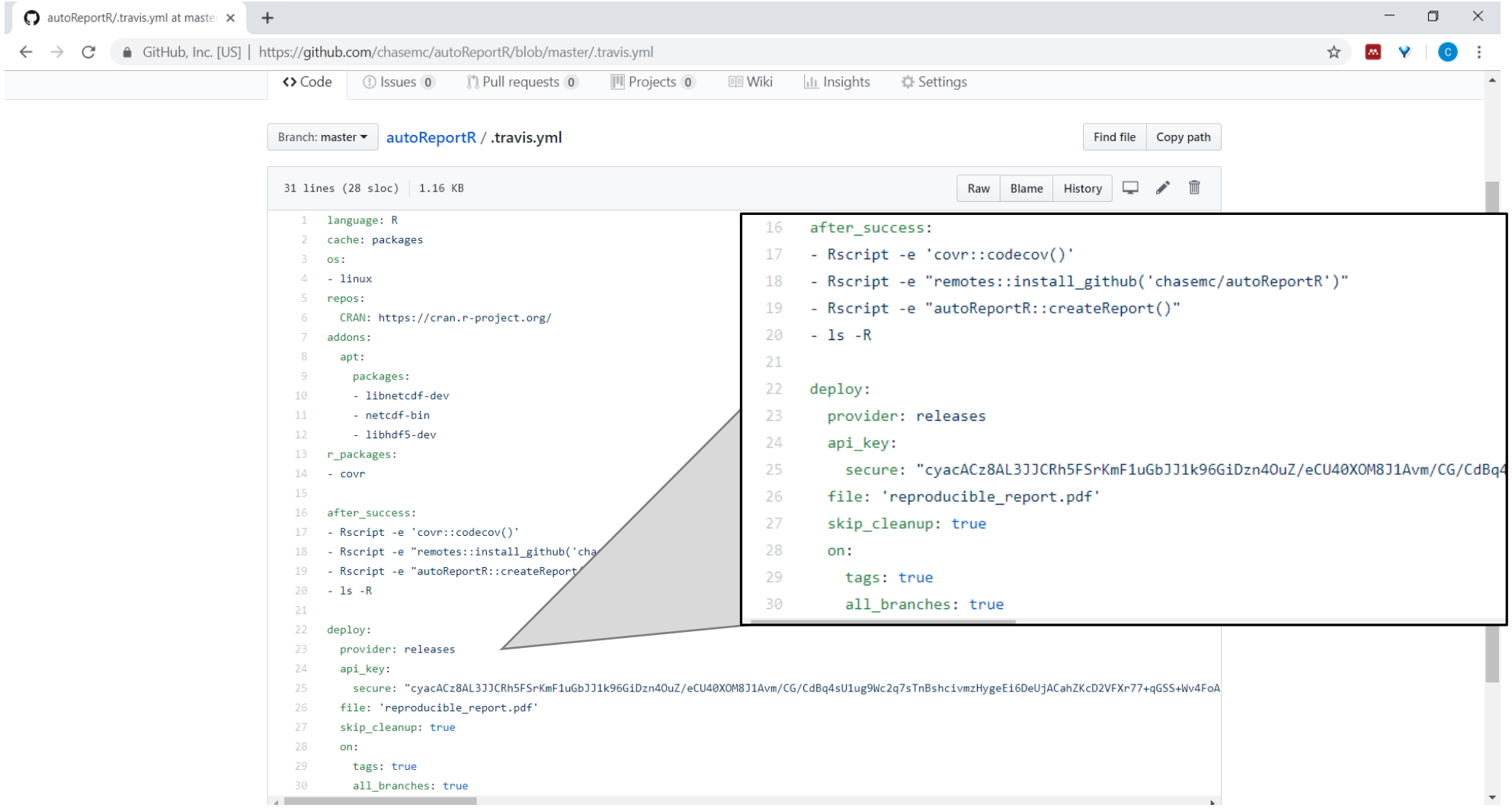
Continuous Integration

Travis CI

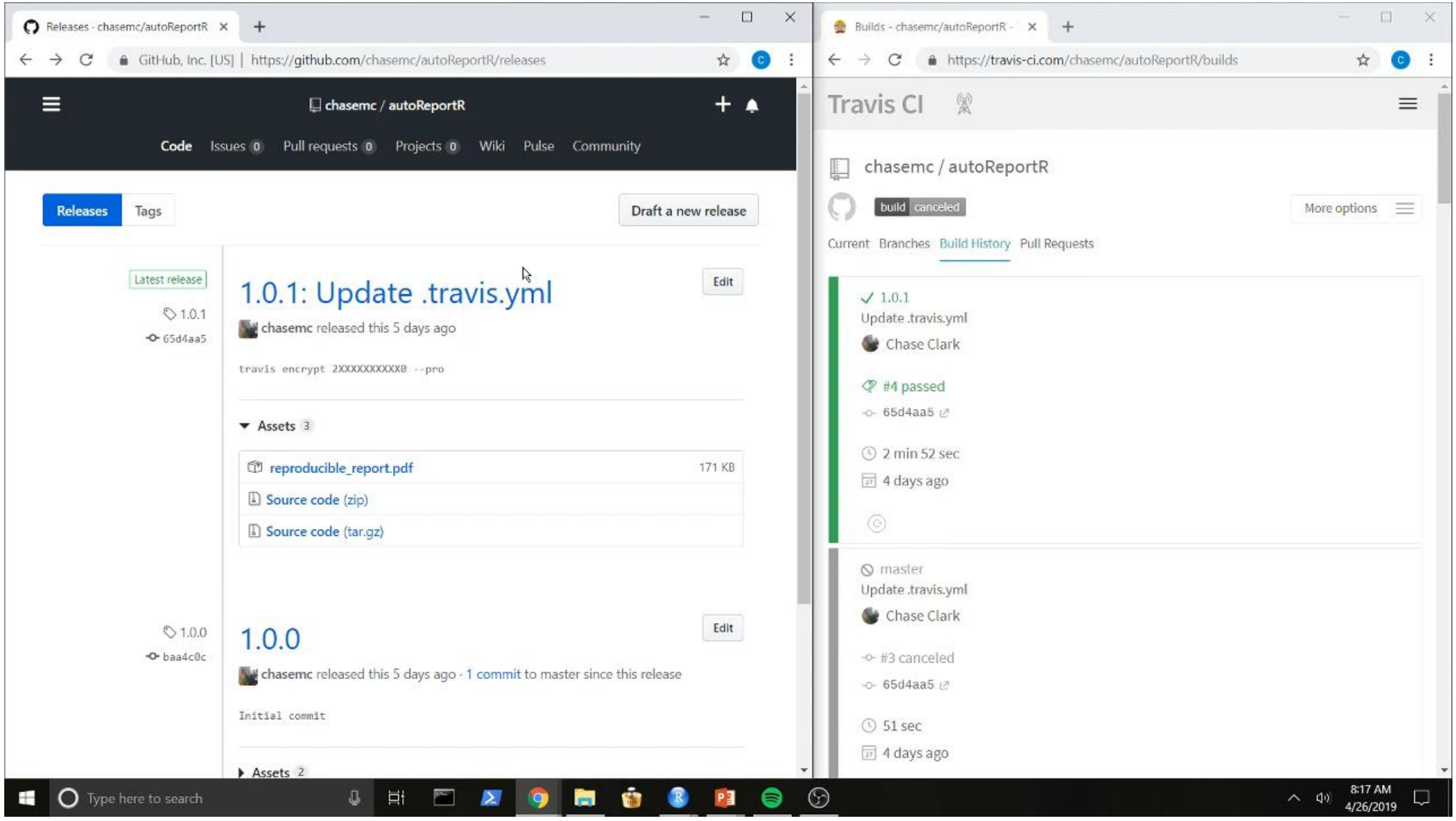
AppVeyor

@ChasingMicrobes

Example Setup Using Travis



https://github.com/chasemc/autoReportR



@ChasingMicrobes

Making Reproducible Insights Through:

~~Code~~

~~Reports~~

Applications/Software

ANTIBIOTIC HUNTERS

Words & Photos by Jennifer Yang

http://projects.thestar.com/antibiotics-resistance-and-the-race-for-new-bacteria/#one

@ChasingMicrobes

Pharmacognosy

The study of the physical, chemical, biochemical and biological properties of drugs, drug substances or potential drugs of natural origin as well as the search for new drugs from natural sources.

Pharmacognosy

The study of the physical, chemical, biochemical and biological properties of drugs, drug substances or potential drugs of natural origin as well as the search for new drugs from natural sources.

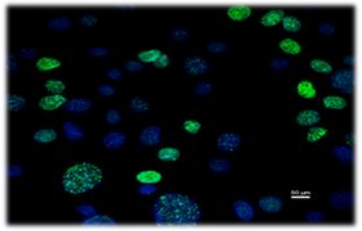
https://blog.sciencemuseum.org.uk/superbugs-at-the-science-museum

https://youtu.be/KoIDErEhaAQ

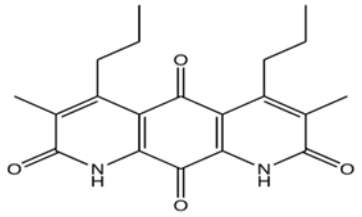
Discovery Pipeline (Simplified)

Collect Samples

Grow Bacteria



Antibiotic Activity?



Elucidate Antibiotic

Drug

Expensive!
Can take weeks to months

We didn’t have an easy way to prioritize bacteria

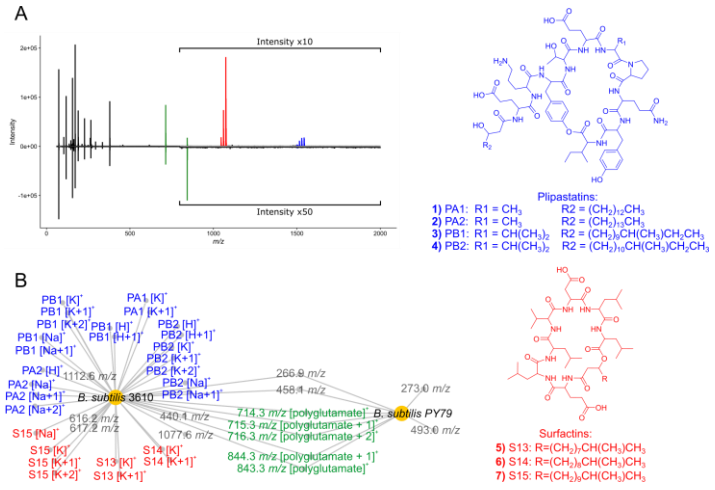

Shiny
www.rstudio.com
IDBac
A MALDI Protein and Small Molecule Bioinformatics Platform
GITHUB
TWITTER
@ChasingMicrobes

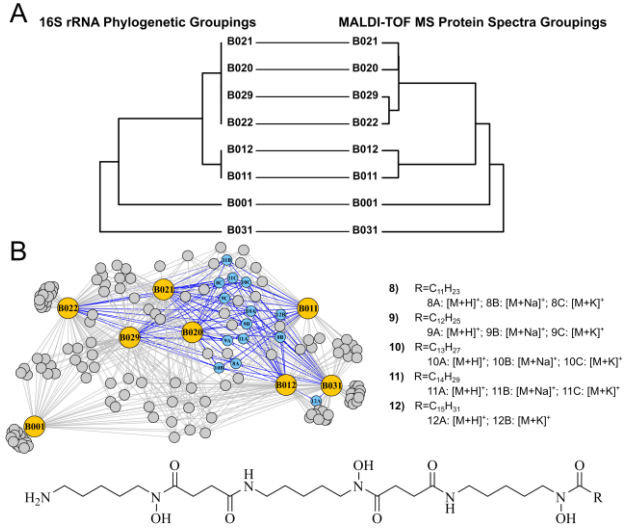
Coupling MALDI-TOF mass spectrometry protein and specialized metabolite analyses to rapidly discriminate bacterial function

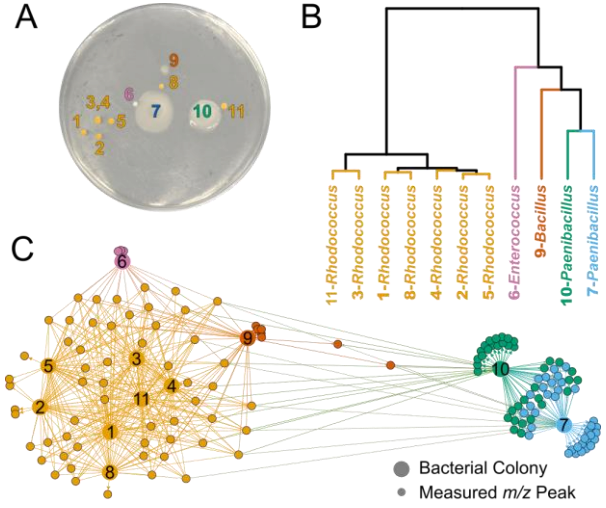
Chase M. Clark[a,1], Maria S. Costa[b,1], Laura M. Sanchez[a,2], and Brian T. Murphy[a,2,3]

[a]Department of Medicinal Chemistry and Pharmacognosy, College of Pharmacy, University of Illinois at Chicago, Chicago, IL; and [b]Faculty of Pharmaceutical Sciences, University of Iceland, Hagi, IS-107 Reykjavík, Iceland

Edited by Jerrold Meinwald, Cornell University, Ithaca, NY, and approved April 5, 2018 (received for review January 22, 2018)

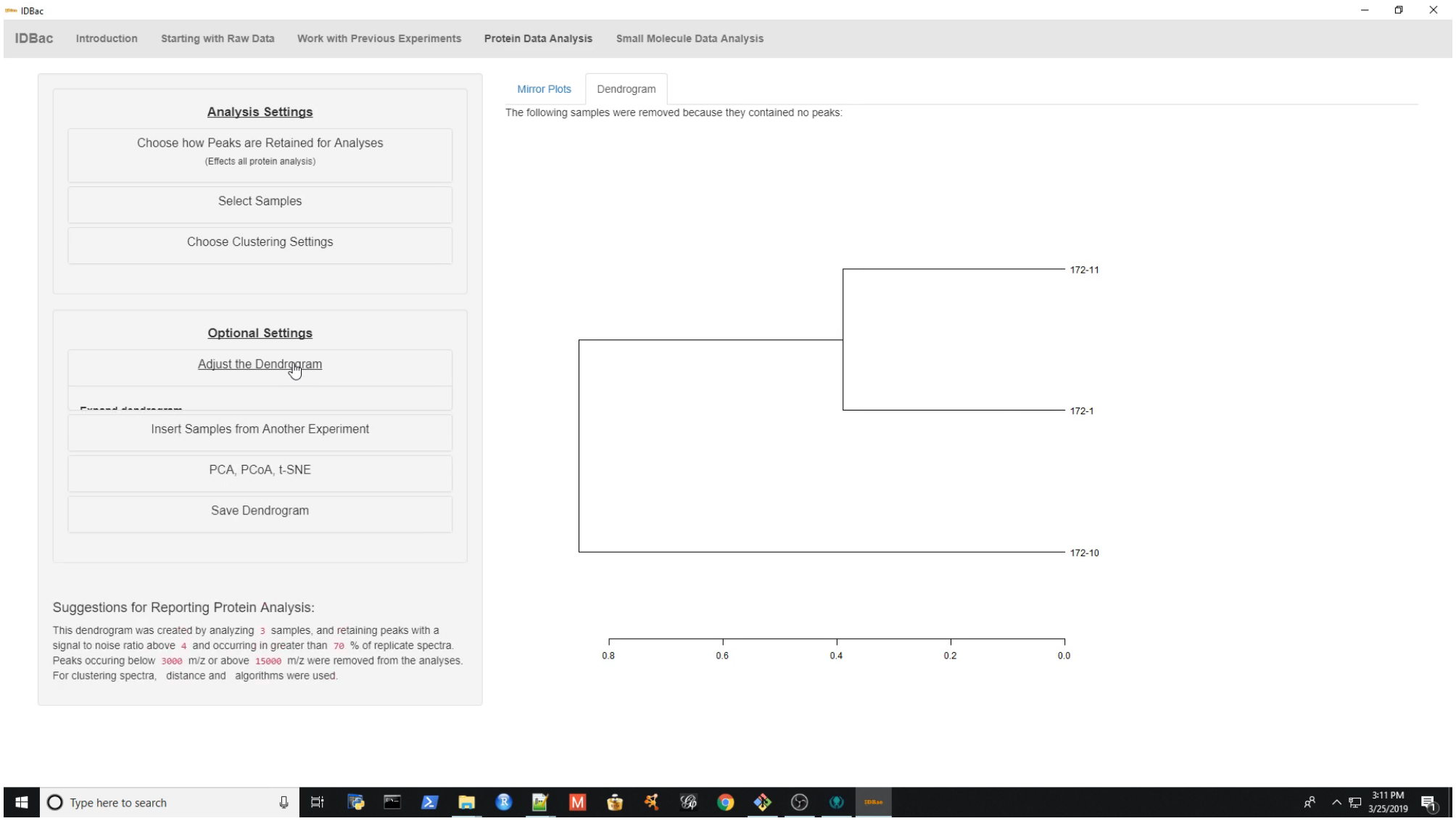






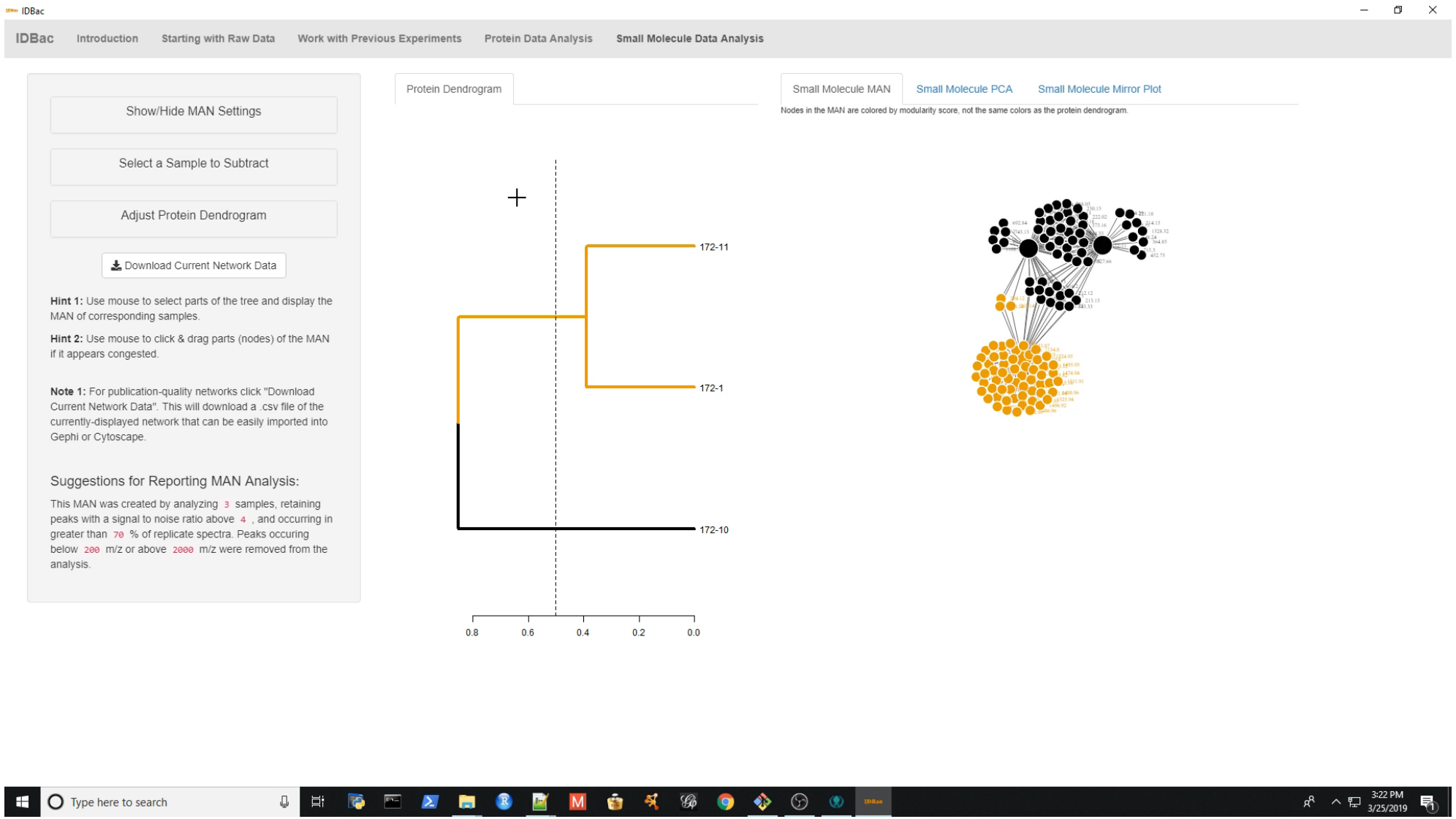
https://doi.org/10.1073/pnas.1801247115

Simple Interface to Complicated Data



@ChasingMicrobes

Interactively Working with Disparate Data Sets



@ChasingMicrobes

Can your intermediate or final data be read by other languages?

```
> DBI::dbListTables(examplePool)
[1] "IndividualSpectra" "XML"  "massTable"  "metaData"   "version"
```

```
> dplyr::tbl(examplePool, "IndividualSpectra") %>% dplyr::select(7:10)
# Source:   lazy query [?? x 4]
# Database: sqlite 3.22.0 [C:\Users\chase\Documents\GitHub\IDBacApp\example.sqlite]
   peakMatrix                                                                                                                                        spectrumIntensi~ maxMass minMass
   <chr>                                                                                                                                                       <blob>   <int>   <int>
 1 "{\"mass\":[61.28432,70.25831,72.2628,74.2751,75.27528,76.21592,81.20401,84.23639,85.19377,86.24006,90.20072,95.20732,97.1644,98.172~  <raw 375.28 kB>    2698      60
 2 "{\"mass\":[61.28972,70.25253,72.25693,74.2751,75.27528,76.20989,81.20712,84.2237,85.20653,86.24006,90.20072,95.19045,97.16781,98.17~  <raw 374.61 kB>    2698      60
 3 "{\"mass\":[1919.93905,1925.91941,1946.20174,1959.50804,1973.48944,1987.52339,2000.97804,2015.11476,2028.66757,2042.26833,2056.7717,~  <raw  82.38 kB>   20924    1919
 4 "{\"mass\":[1919.93905,1925.71302,1945.99421,1959.29976,1973.28037,1987.31354,2000.97804,2015.11476,2028.45543,2042.05545,2056.7717,~  <raw  80.59 kB>   20924    1919
 5 "{\"mass\":[61.30321,63.31429,69.29265,70.25831,70.48992,71.27737,72.27161,73.27579,74.2751,74.56997,75.0747,75.2663,75.96892,76.252~  <raw 308.50 kB>    2698      60
 6 "{\"mass\":[61.27083,69.26393,70.2612,71.26281,72.26867,73.26692,74.28105,75.27229,76.21592,79.21782,80.22017,81.22269,84.23005,85.2~  <raw 323.60 kB>    2698      60
 7 "{\"mass\":[1919.93905,1926.9515,1953.05681,1981.8616,2002.24187,2017.65226,2026.12276,2039.92729,2058.26777,2076.05058,2085.07985,2~  <raw  79.39 kB>   20924    1919
 8 "{\"mass\":[1919.93905,1953.47268,1977.04532,1987.94312,2002.45255,2018.07534,2025.91077,2039.92729,2076.26532,2085.07985,2099.74247~  <raw  78.46 kB>   20924    1919
 9 "{\"mass\":[70.24963,72.25987,74.27807,75.27828,76.22195,81.19778,84.22053,85.20334,86.24006,97.1644,98.16934,102.18194,106.16528,11~  <raw 389.08 kB>    2698      60
10 "{\"mass\":[61.29782,70.25831,71.24241,72.26574,74.28402,75.28127,76.21893,81.20712,84.23005,85.21929,86.24647,94.22459,97.17463,98.~  <raw 387.35 kB>    2698      60
# ... with more rows
```

Can your intermediate or final data be read by other languages?

```
> DBI::dbListTables(examplePool)
[1] "IndividualSpectra" "XML"  "massTable"  "metaData"   "version"
```

```
> dplyr::tbl(examplePool, "IndividualSpectra") %>% dplyr::select(7:10)
# Source:   lazy query [?? x 4]
# Database: sqlite 3.22.0 [C:\Users\chase\Documents\GitHub\IDBacApp\example.sqlite]
   peakMatrix                                                                                                                              spectrumIntensi~ maxMass minMass
   <chr>                                                                                                                                             <blob>   <int>   <int>
 1 "{\"mass\":[61.28432,70.25831,72.2628,74.2751,75.27528,76.21592,81.20401,84.23639,85.19377,86.24006,90.20072,95.20732,97.1644,98.172~ <raw 375.28 kB>    2698      60
 2 "{\"mass\":[61.28972,70.25253,72.25693,74.2751,75.27528,76.20989,81.20712,84.2237,85.20653,86.24006,90.20072,95.19045,97.16781,98.17~ <raw 374.61 kB>    2698      60
 3 "{\"mass\":[1919.93905,1925.91941,1946.20174,1959.50804,1973.48944,1987.52339,2000.97804,2015.11476,2028.66757,2042.26833,2056.7717,~ <raw  82.38 kB>   20924    1919
 4 "{\"mass\":[1919.93905,1925.71302,1945.99421,1959.29976,1973.28037,1987.31354,2000.97804,2015.11476,2028.45543,2042.05545,2056.7717,~ <raw  80.59 kB>   20924    1919
 5 "{\"mass\":[61.30321,63.31429,69.29265,70.25831,70.48992,71.27737,72.27161,73.27579,74.2751,74.56997,75.0747,75.2663,75.96892,76.252~ <raw 308.50 kB>    2698      60
 6 "{\"mass\":[61.27083,69.26393,70.2612,71.26281,72.26867,73.26692,74.28105,75.27229,76.21592,79.21782,80.22017,81.22269,84.23005,85.2~ <raw 323.60 kB>    2698      60
 7 "{\"mass\":[1919.93905,1926.9515,1953.05681,1981.8616,2002.24187,2017.65226,2026.12276,2039.92729,2058.26777,2076.05058,2085.07985,2~ <raw  79.39 kB>   20924    1919
 8 "{\"mass\":[1919.93905,1953.47268,1977.04532,1987.94312,2002.45255,2018.07534,2025.91077,2039.92729,2076.26532,2085.07985,2099.74247~ <raw  78.46 kB>   20924    1919
 9 "{\"mass\":[70.24963,72.25987,74.27807,75.27828,76.22195,81.19778,84.22053,85.20334,86.24006,97.1644,98.16934,102.18194,106.16528,11~ <raw 389.08 kB>    2698      60
10 "{\"mass\":[61.29782,70.25831,71.24241,72.26574,74.28402,75.28127,76.21893,81.20712,84.23005,85.21929,86.24647,94.22459,97.17463,98.~ <raw 387.35 kB>    2698      60
# ... with more rows
```

You shouldn’t just throw everything into a .rds

```
> serialize("hello", NULL)
 [1] 58 0a 00 00 00 02 00 03 05 03 00 02 03 00 00 00 00 10 00 00 00 01 00 04 00 09 00 00
00 05 68 65 6c 6c 6f
> charToRaw("hello")
[1] 68 65 6c 6c 6f
```

```
> serialize(data.frame(1,1), NULL)
  [1] 58 0a 00 00 00 02 00 03 05 03 00 02 03 00 00 00 03 13 00 00 00 02 00 00 00 0e 00 00 00 01 3f f0 00 00 00 00 00 00 00 00 0e 00
 [44] 00 00 01 3f f0 00 00 00 00 00 00 00 00 04 02 00 00 00 01 00 04 00 09 00 00 00 05 6e 61 6d 65 73 00 00 00 10 00 00 00 02 00 04 00
 [87] 09 00 00 00 02 58 31 00 04 00 09 00 00 00 04 58 31 2e 31 00 00 04 02 00 00 00 01 00 04 00 09 00 00 00 05 63 6c 61 73 73 00 00 00
[130] 10 00 00 00 01 00 04 00 09 00 00 00 0a 64 61 74 61 2e 66 72 61 6d 65 00 00 04 02 00 00 00 01 00 04 00 09 00 00 00 09 72 6f 77 2e
[173] 6e 61 6d 65 73 00 00 00 0d 00 00 00 02 80 00 00 00 ff ff ff ff 00 00 00 fe
> charToRaw(data.frame(1,1))
Error in charToRaw(data.frame(1, 1)) :
  argument must be a character vector of length 1
```

Storing compressed vectors in a SQL BLOB

```r
blob_me <- charToRaw("hello")

blob_me  <- base::memCompress(blob_me,
                              type = "gzip")
```

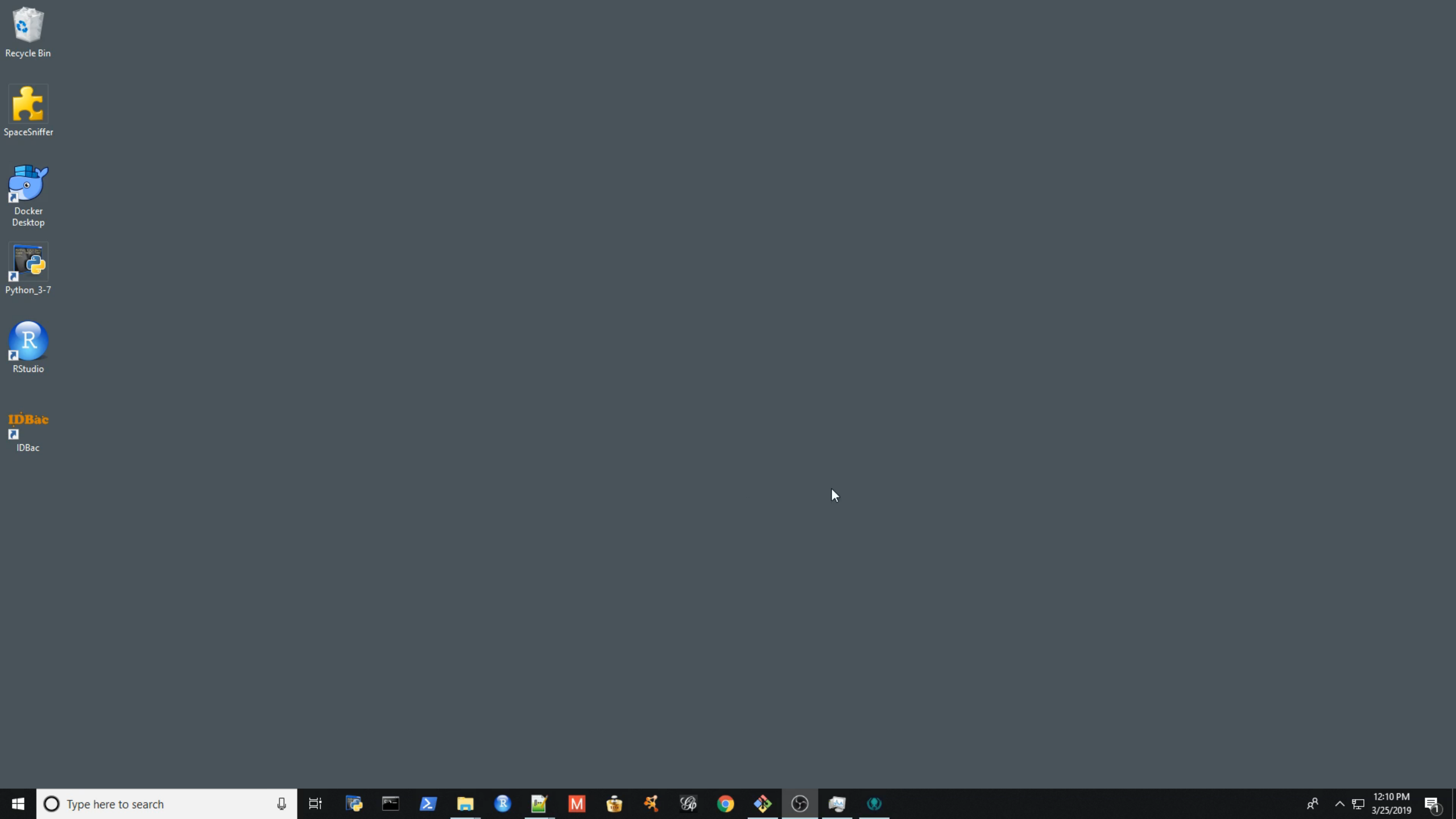
```r
blob_me <- charToRaw("hello")

blob_me <- fst::compress_fst(x,
                             compressor = "ZSTD",
                             compression = 0,
                             hash = FALSE)
```

How do users interact with your…

Shiny apps that are too impractical or prohibitively expensive to run in the cloud?

How do users interact with your results?

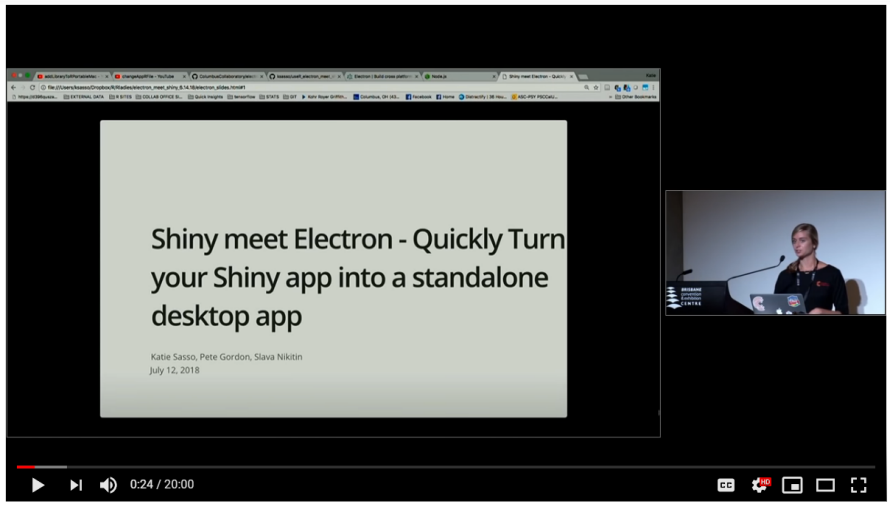
- Technical users (use R)
- Technical users (don’t use R)
- Non-technical users (definitely don’t use R)


Recycle Bin
SpaceSniffer
Docker Desktop
Python_3-7
RStudio
IDBac
Type here to search
12:10 PM
3/25/2019

Creating/Distributing Local Shiny Applications

Only Windows
Can be Complicated
Not Self-Sufficient

github.com/ficonsulting/RInno



Shiny meets Electron: Turn your Shiny app into a standalone desktop app in no time

4,057 views 81 1 SHARE SAVE

@KatieSasso

https://github.com/ColumbusCollaboratory/electron-quick-start

https://www.youtube.com/watch?v=ARrbbviGvjc

electricShine R Package

- lifecycle experimental

Windows CI:

- build passing

Mac and Linux CI

- build passing

Is easy! One meta function:

```
electricShine::buildElectricApp(
appName = "My_App",
description = "My demo application",
packageName = "demoApp",
semanticVersion = "1.0.0",
installTo = installTo,
MRANdate = MRANdate,
functionName = "run_app",
githubRepo = "chasemc/demoApp",
localPath  = NULL,
only64 = TRUE
)
```

Self-Sufficient

The electricShine package:

- Installs nodejs
- Sets up Electron build
- Compiles your app
- When complete, will work for Windows/Mac/Linux
- Compatible with continuous-deployment

An electricShine app:

- R- from specified MRAN date
- R packages- from specified MRAN date
- Your Shiny app!

https://github.com/chasemc/electricShine

Thanks for Listening

All packages talked about are open-source and welcome contributions.

Please empathize with your users.

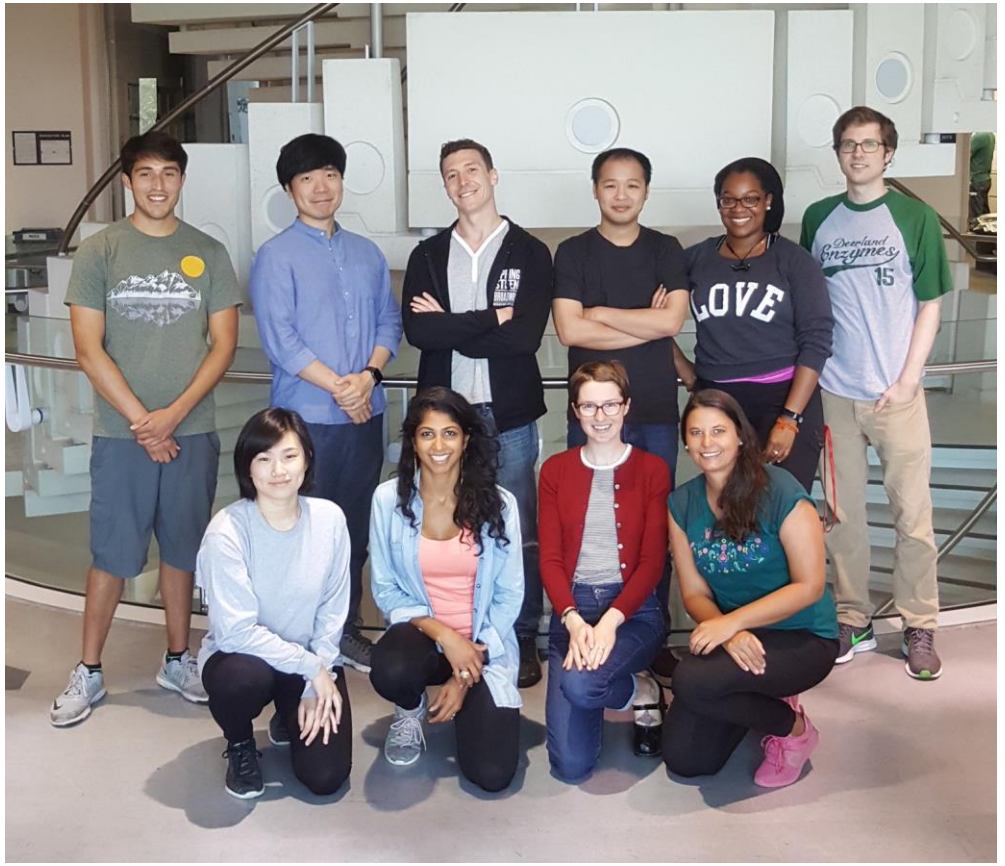
Please continue being a welcoming community.

Special Thanks

Dr. Laura Sanchez

Murphy Lab
Dr. Brian Murphy
Sofia Costa
Antonio Hernandez
Dr. Jeong Ho Lee
Dr. Tuan Anh Tran
Maryam Elfeki
Vanessa Nepomuceno
Emma Ward
Erin Conley

Past Undergraduate Researchers:
Antonio Hernandez
Milan Patel
Jhewelle Fitz-Henley
Adithyan Subramanian
David An



@ChasingMicrobes
chasemc.github.io
murphylabuic.com
@Murphylabuic

Funding:
National Center For Complementary & Integrative Health of the National Institutes of Health
Award Number: F31AT010419

Research reported in this presentation was supported by the National Center For Complementary & Integrative Health of the National Institutes of Health under Award Number F31AT010419. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.