

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](https://github.com/chasemc)

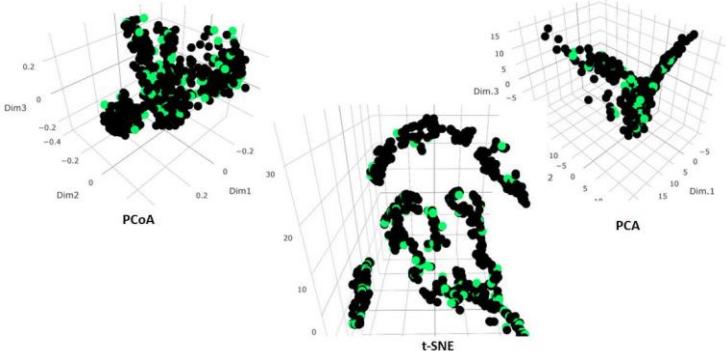


Photos of slides OK

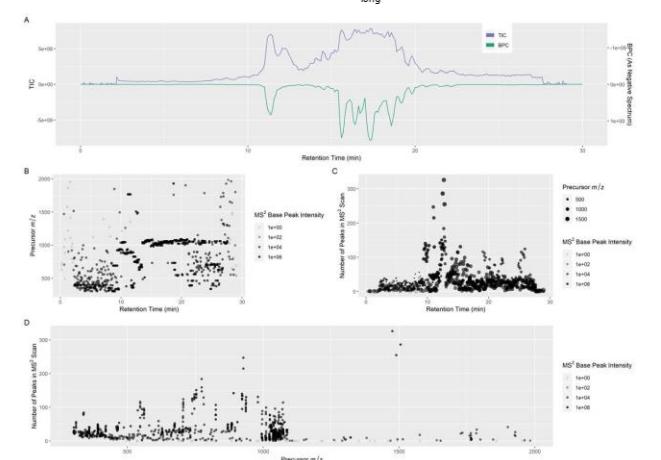
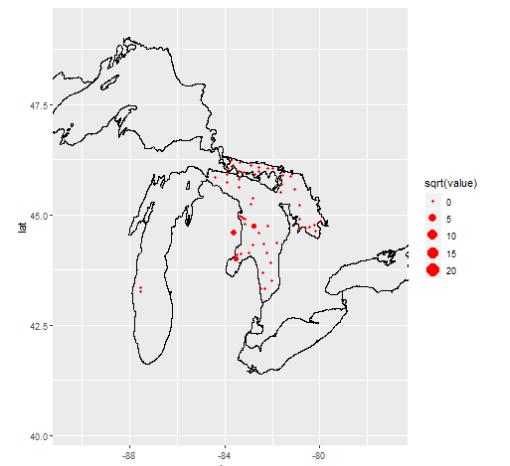
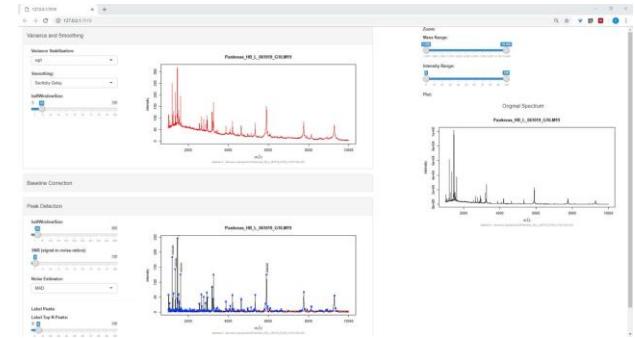
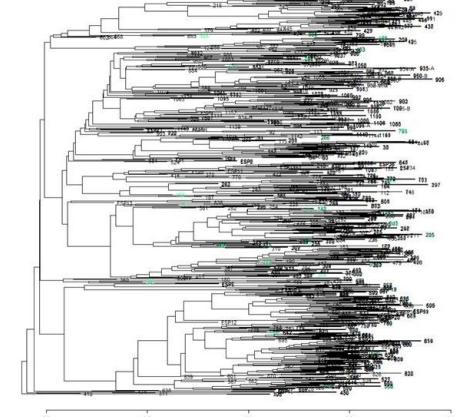
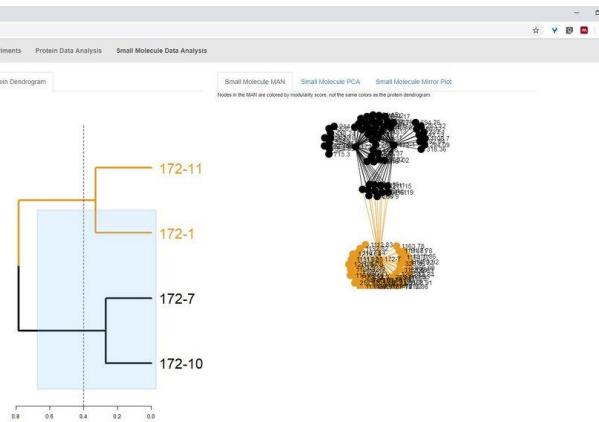
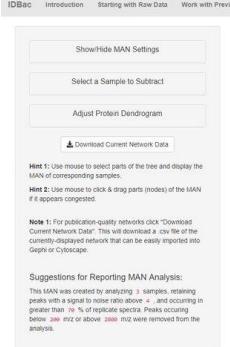
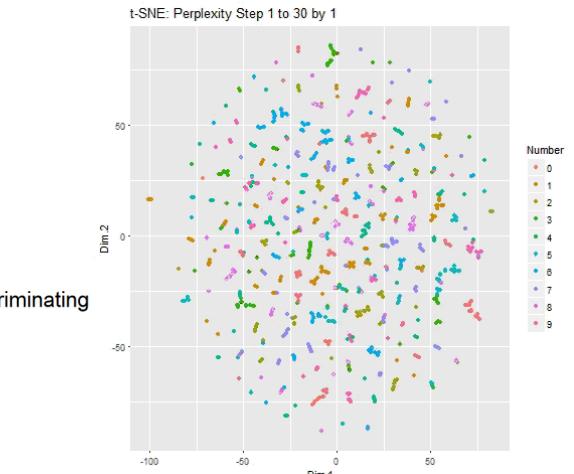
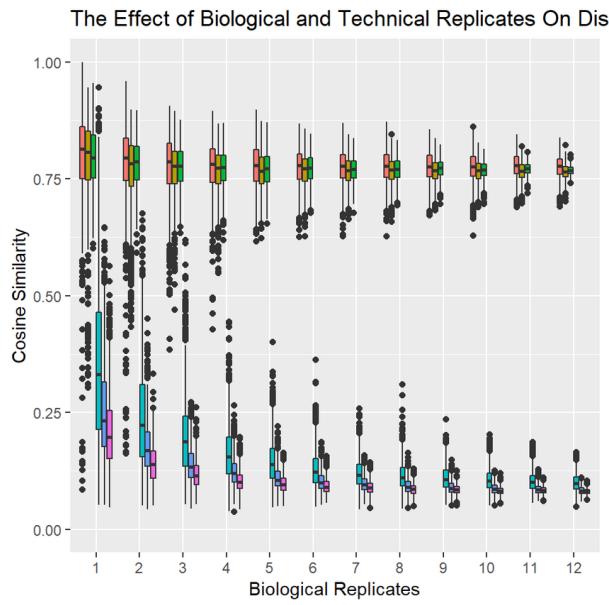
Main Takeaways

- 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 * scalar)))
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   # tainty in the 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   },
```

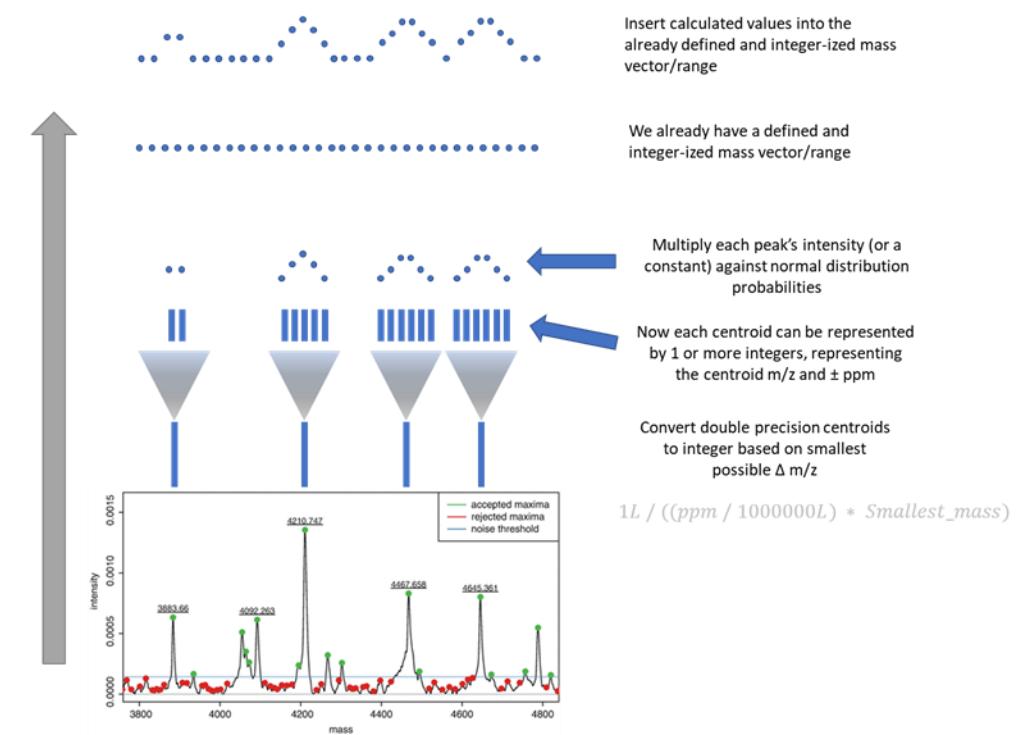
Note: Some conventions
broken here to fit to slide.



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.

The screenshot shows a table of mass spectra entries. The columns include ClusterIndex, Filename, Scan, Precursor Int, RT, and Parent Mass. The table has 15 rows, each representing a different mass spectrum entry. A play button icon is overlaid on the first row.

ClusterIndex	Filename	Scan	Precursor Int	RT	Parent Mass
1	1001.ms2XML	4028	929891.000	2408.500	445.120
2	1001.ms2XML	4128	101730.000	2405.310	445.120
3	1001.ms2XML	4229	730379.000	2491.010	445.120
4	1001.ms2XML	4338	802948.000	2531.740	445.121
5	1001.ms2XML	4446	974482.000	2572.560	445.120
6	1001.ms2XML	4850	5000.000	2712.330	445.120
7	1001.ms2XML	4983	5000.000	2732.870	445.120
8	1001.ms2XML	5113	2793.700	445.121	Gold Silver Bronze
9	1001.ms2XML	5285	856711.000	2850.510	445.120
10	1001.ms2XML	5604	627557.000	2949.780	445.121
11	1001.ms2XML	5730	666837.000	2990.680	445.121
12	1001.ms2XML	5867	561482.000	3035.140	445.121
13	1001.ms2XML	6126	683200.000	3118.200	445.121
14	1001.ms2XML	6314	604958.000	3185.280	445.121

Insight Through Reports

Had: Webservice with detailed analyses (python and bash)

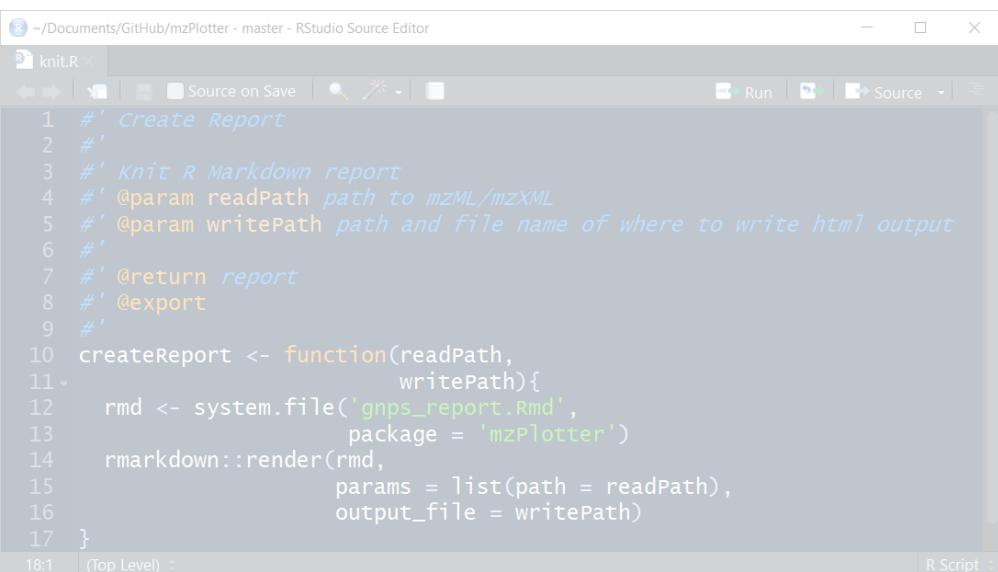
Wanted: Quick summary plots of uploaded raw data



@ChasingMicrobes

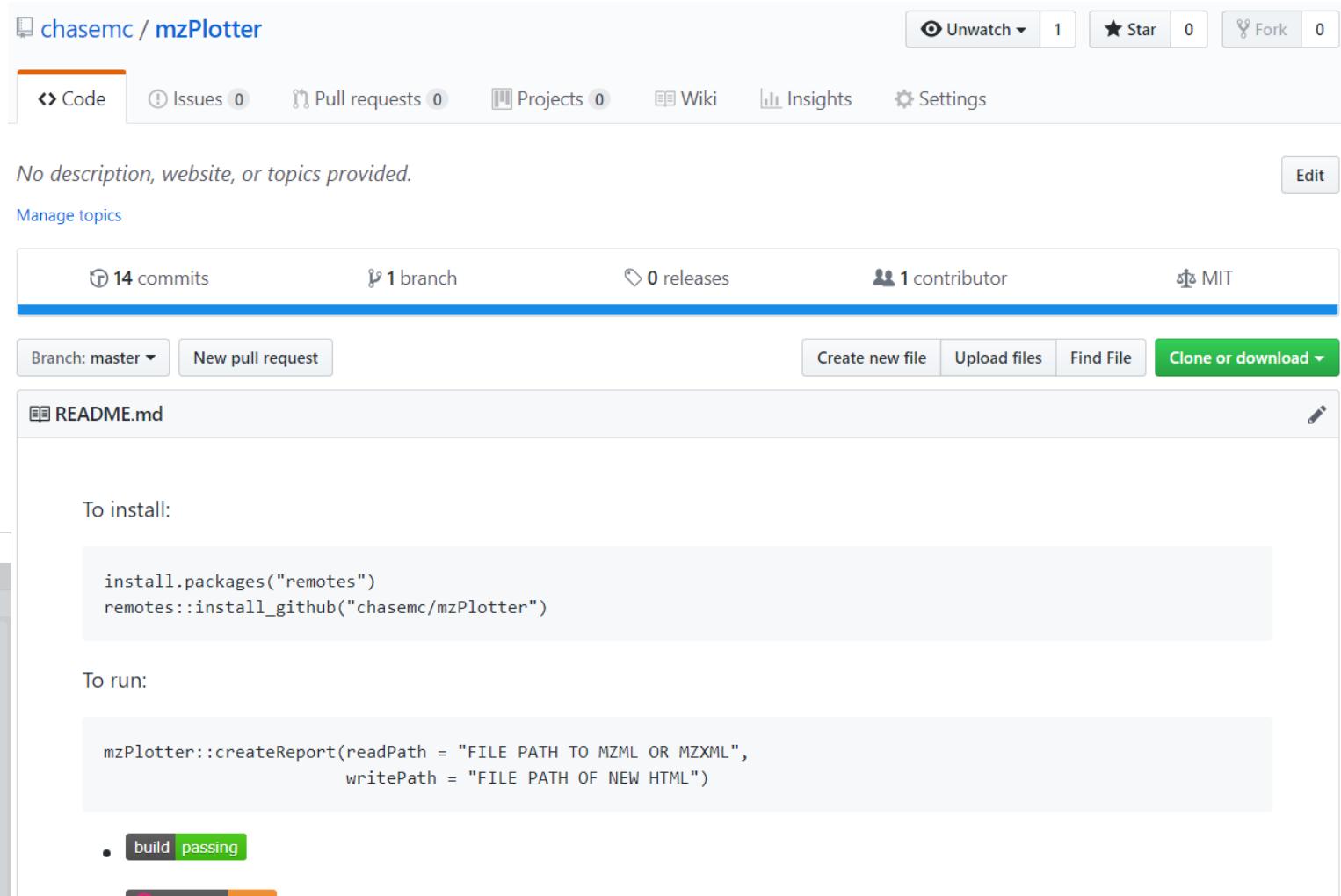
How do users interact with your results?
Technical user that doesn't use R

Simple Functions



The screenshot shows the RStudio Source Editor window with the file 'knit.R' open. The code defines a function 'createReport' that takes 'readPath' and 'writePath' parameters, creates an R Markdown report ('gnps_report.Rmd'), and renders it using rmarkdown. The code is annotated with comments explaining its purpose.

```
1 #' Create Report
2 #
3 #' Knit R Markdown report
4 #' @param readPath path to mzML/mzXML
5 #' @param writePath path and file name of where to write html output
6 #
7 #' @return report
8 #' @export
9 #
10 createReport <- function(readPath,
11                         writePath){
12   rmd <- system.file('gnps_report.Rmd',
13                      package = 'mzPlotter')
14   rmarkdown::render(rmd,
15                   params = list(path = readPath),
16                   output_file = writePath)
17 }
```



The screenshot shows the GitHub repository page for 'mzPlotter' owned by 'chasemc'. The repository has 14 commits, 1 branch, 0 releases, and 1 contributor. It uses the MIT license. The README.md file contains instructions for installation and running the code.

No description, website, or topics provided.

Manage topics

14 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find File Clone or download

README.md

To install:

```
install.packages("remotes")
remotes::install_github("chasemc/mzPlotter")
```

To run:

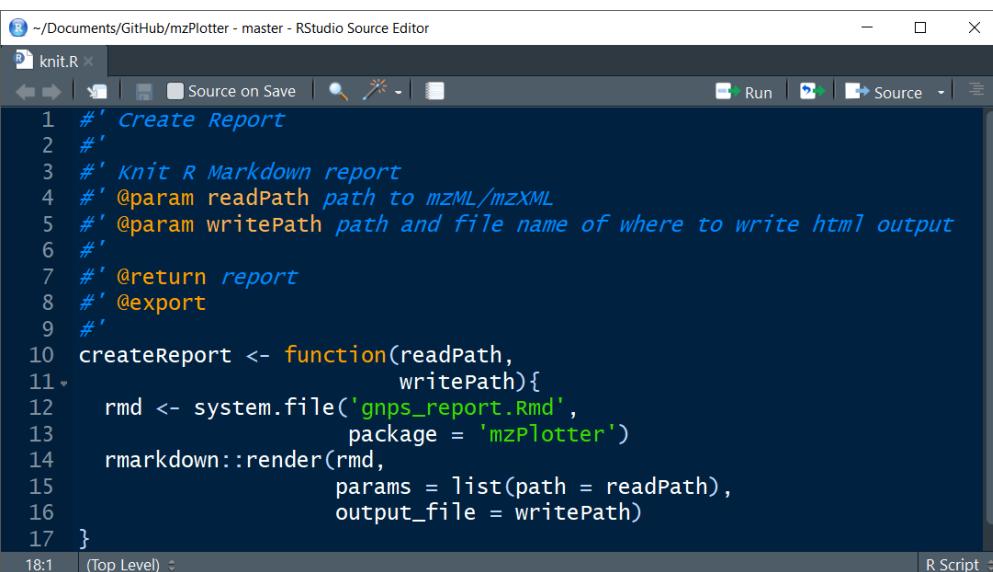
```
mzPlotter::createReport(readPath = "FILE PATH TO MZML OR MZXML",
                        writePath = "FILE PATH OF NEW HTML")
```

- build passing
- codecov 78%



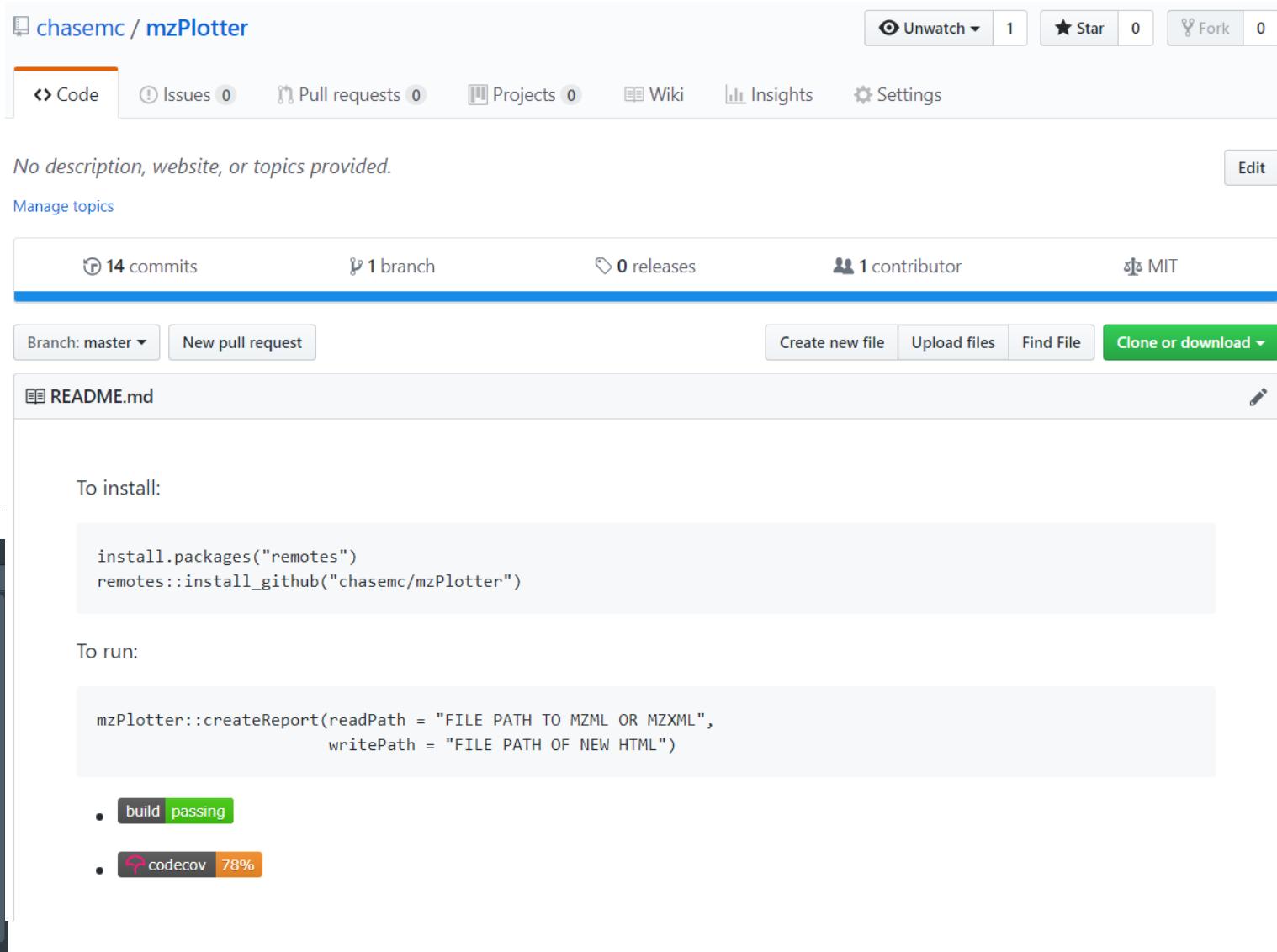
How do users interact with your results?
Technical user that doesn't use R

Simple Functions



The screenshot shows the RStudio Source Editor with the file 'knit.R' open. The code defines a function 'createReport' that creates an R Markdown report ('.Rmd') from an mzML or mzXML file ('readPath') and saves it as an HTML file ('writePath'). The code uses the 'knitr' and 'rmarkdown' packages.

```
1 #' Create Report
2 #
3 #' Knit R Markdown report
4 #' @param readPath path to mzML/mzXML
5 #' @param writePath path and file name of where to write html output
6 #
7 #' @return report
8 #' @export
9 #
10 createReport <- function(readPath,
11   writePath){
12   rmd <- system.file('gnps_report.Rmd',
13     package = 'mzPlotter')
14   rmarkdown::render(rmd,
15     params = list(path = readPath),
16     output_file = writePath)
17 }
```



The screenshot shows the GitHub repository page for 'mzPlotter' owned by 'chasemc'. The repository has 14 commits, 1 branch, 0 releases, and 1 contributor. It is licensed under MIT. The README.md file contains instructions for installation and running the code.

No description, website, or topics provided.

Manage topics

14 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request Create new file Upload files Find File Clone or download

README.md

To install:

```
install.packages("remotes")
remotes::install_github("chasemc/mzPlotter")
```

To run:

```
mzPlotter::createReport(readPath = "FILE PATH TO MZML OR MZXML",
                        writePath = "FILE PATH OF NEW HTML")
```

• build passing
• codecov 78%



How do users interact with your results? Anyone

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The title bar indicates the current project is 'mzPlotter' and the file is 'knit.R'. The main workspace shows the following R code:

```
1 #' Create Report
2 #
3 #' Knit R Markdown report
4 #' @param readPath path to mzML/mzXML
5 #' @param writePath path and file name of where to write html output
6 #
7 #' @return report
8 #' @export
9 #
10 createReport <- function(readPath,
11                         writePath){
```

The status bar at the bottom shows the file is at line 9:2, level Top Level, and the script type is R Script.

In the bottom-left corner, there is a terminal window with the following command and output:

```
Console Terminal ×
-/Documents/GitHub/mzPlotter/
> mzPlotter:::createReport(readPath = "C:/Users/CMC/Downloads/conversion/thermo_file.mzML",
   writePath = "C:/Users/CMC/Downloads/demo/demo.html")|
```

The taskbar at the bottom of the screen includes icons for File Explorer, Task View, Start, Search, Taskbar settings, and several pinned applications: File Explorer, Task View, Start, Search, Taskbar settings, Microsoft Edge, File Explorer, Spotify, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, Microsoft OneDrive, Microsoft Teams, and Microsoft Edge.



Insight Through Automated Reports



Insight Through Automated Reports

A screenshot of the RStudio interface. The title bar shows the path: ~/Documents/GitHub/autoReportR - master - RStudio. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar has various icons. The main workspace shows several files: .travis.yml, appveyor.yml, create_report.R, and reproducible_report.Rmd. The code editor displays R Markdown code. A yellow arrow points to a tooltip for the 'Knit' button in the toolbar. The tooltip text is: 'Knit the current document (Ctrl + Shift + K)'.

```
1 ---  
2 title: "Reproducible Report"  
3 author: "Chase Lark"  
4 date: "April 21, 2019"  
5 output: pdf_document  
6 ---  
7  
8 ``{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ...  
11
```

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

\$14,995 per year

\$62 per user/month

RStudio Connect Standard

\$24,995 per year

\$21 per user/month

RStudio Connect Enterprise*

\$74,995 per year

\$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



Version Control with Git



Commits

Continuous Deployment



Travis CI



Continuous Integration



Travis CI



Organize analysis as an R package



Report attached to GitHub release



Deploy

Create Release

Commits

Continuous Deployment



Travis CI



Continuous Integration



Travis CI



Organize analysis as an R package



Report attached to GitHub release



Deploy

Create Release

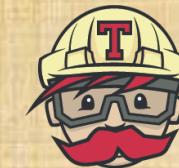
Continuous Deployment



Travis CI



Continuous Integration



Travis CI



@ChasingMicrobes

Organize analysis as an R package



Report attached to GitHub release



Continuous Deployment



Travis CI



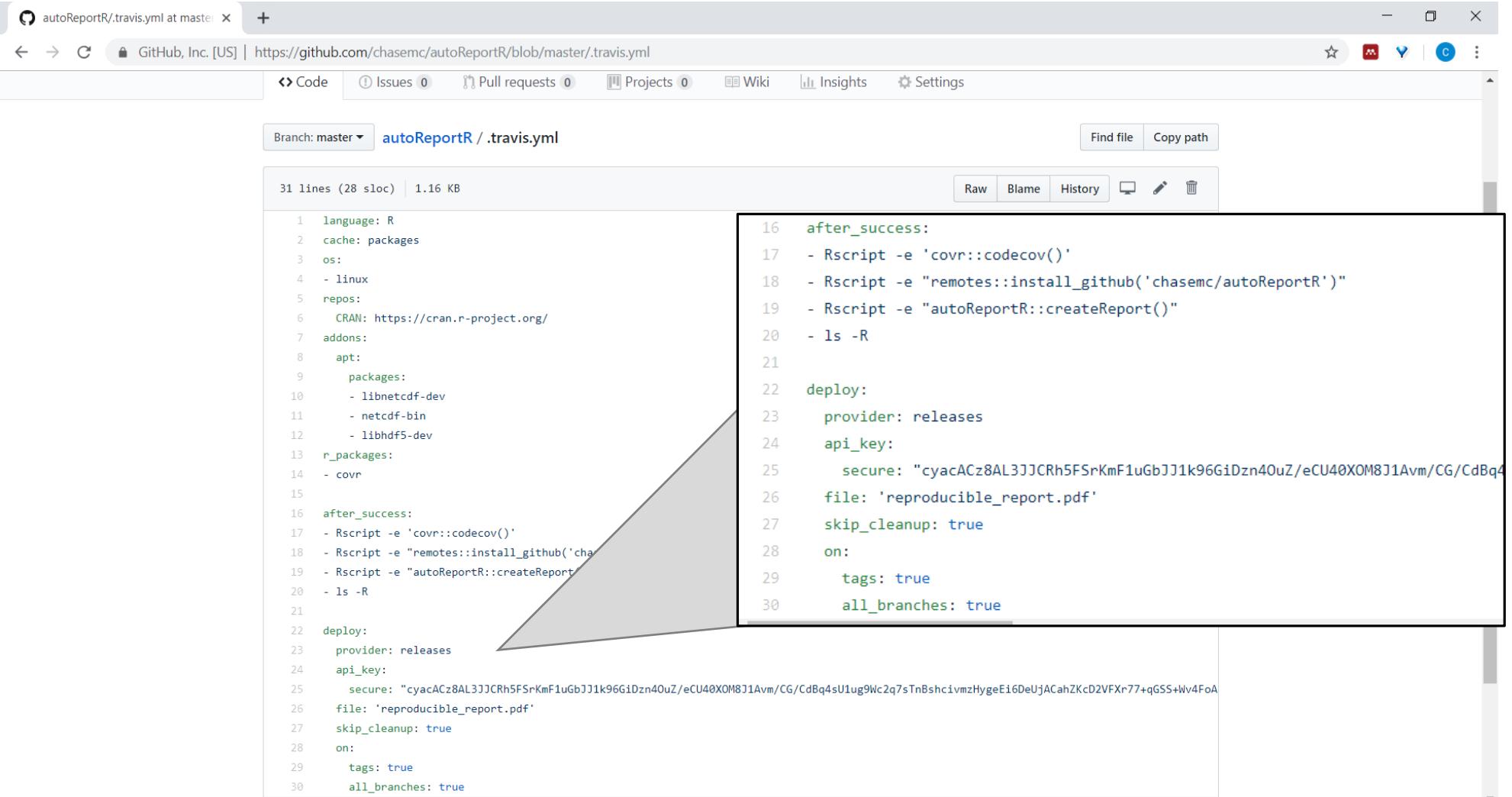
Continuous Integration



Travis CI



Example Setup Using Travis



The screenshot shows a GitHub browser window displaying the `.travis.yml` configuration file for the `autoReportR` repository. The file contains YAML code for Travis CI, specifying build steps, dependencies, and deployment settings.

```
language: R
cache: packages
os:
- linux
repos:
  CRAN: https://cran.r-project.org/
addons:
  apt:
    packages:
      - libnetcdf-dev
      - netcdf-bin
      - libhdf5-dev
  r_packages:
    - covr

after_success:
- Rscript -e 'covr::codecov()'
- Rscript -e "remotes::install_github('chasemc/autoReportR')"
- Rscript -e "autoReportR::createReport()"
- ls -R

deploy:
  provider: releases
  api_key:
    secure: "cyacACz8AL3JJCRh5FSrKmF1uGbJJ1k96GiDzn40uZ/eCU40XOM8J1Avm/CG/CdBq4sU1ug9Wc2q7sTnBshcivmzHygeEi6DeUjACahZKcD2VFXr77+qGSS+Wv4FoA"
  file: 'reproducible_report.pdf'
  skip_cleanup: true
  on:
    tags: true
    all_branches: true
```



The image shows two side-by-side browser windows. The left window is a GitHub release page for the repository "chasemc/autoReportR". It displays two releases: "1.0.1: Update .travis.yml" (released 5 days ago) and "1.0.0" (released 5 days ago). The "1.0.1" release includes a PDF asset named "reproducible_report.pdf" (171 KB) and two source code assets: "Source code (zip)" and "Source code (tar.gz)". The right window is a Travis CI build history page for the same repository. It shows two builds: a successful build #4 (4 days ago) and a canceled build #3 (4 days ago). Both builds were triggered by commit 65d4aa5 and lasted 51 seconds. The Travis CI interface includes navigation links for Current, Branches, Build History, and Pull Requests.

Releases - chasemc/autoReportR

GitHub, Inc. [US] | https://github.com/chasemc/autoReportR/releases

chasemc / autoReportR

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Community

Releases Tags

Draft a new release

Latest release

1.0.1: Update .travis.yml

chasemc released this 5 days ago

travis encrypt 2XXXXXXXXXX8 --pro

Assets 3

- reproducible_report.pdf (171 KB)
- Source code (zip)
- Source code (tar.gz)

1.0.0

chasemc released this 5 days ago · 1 commit to master since this release

Initial commit

Assets 2

Builds - chasemc/autoReportR

Travis CI

chasemc / autoReportR

build canceled

More options

Current Branches Build History Pull Requests

#4 passed

Chase Clark

2 min 52 sec

4 days ago

master

Update .travis.yml

Chase Clark

#3 canceled

65d4aa5

51 sec

4 days ago

Type here to search

8:17 AM 4/26/2019



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.



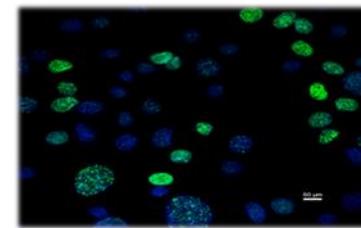
Discovery Pipeline (Simplified)



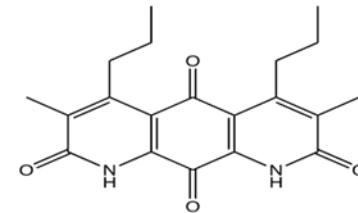
Collect
Samples



Grow
Bacteria



Antibiotic
Activity?



Elucidate
Antibiotic



Drug



Expensive!
Can take weeks to months

We didn't have an easy way to prioritize bacteria





IDBac

A MALDI Protein and Small Molecule Bioinformatics Platform

GITHUB

TWITTER



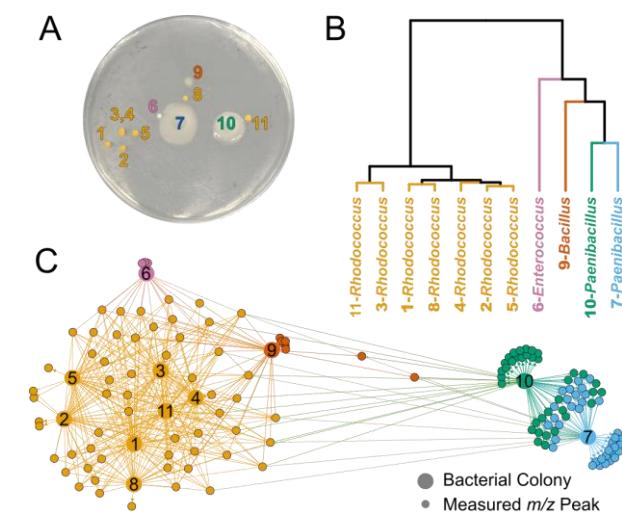
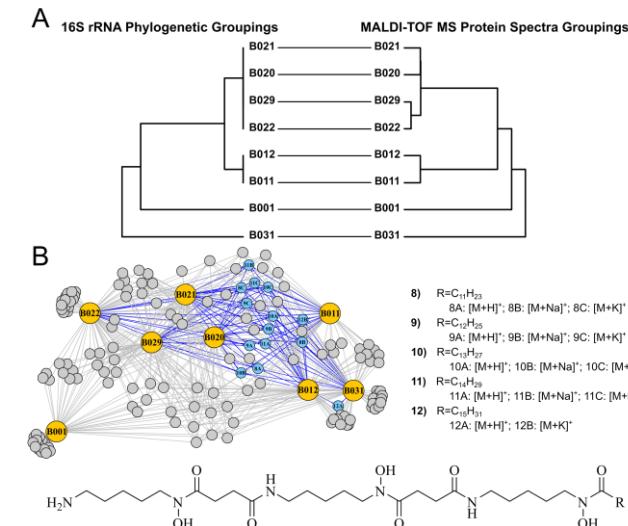
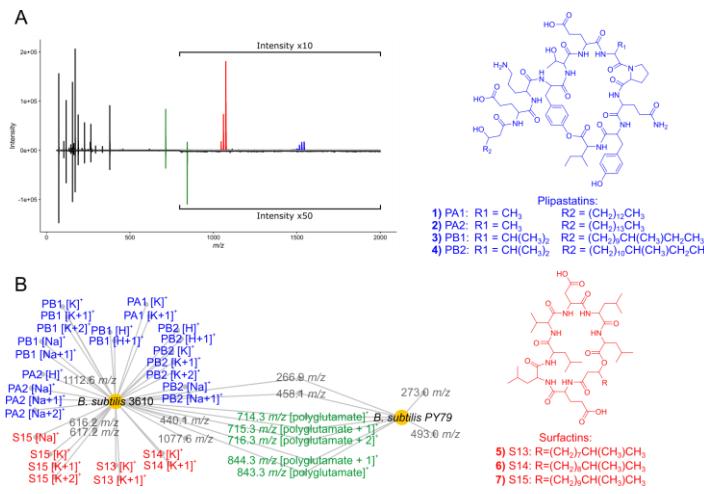
@ChasingMicrobes

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}

^aDepartment of Medicinal Chemistry and Pharmacognosy, College of Pharmacy, University of Illinois at Chicago, Chicago, IL; and ^bFaculty 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)





Simple Interface to Complicated Data

IDBac

IDBac Introduction Starting with Raw Data Work with Previous Experiments Protein Data Analysis Small Molecule Data Analysis

Analysis Settings

Choose how Peaks are Retained for Analyses
(Effects all protein analysis)

Select Samples

Choose Clustering Settings

Optional Settings

Adjust the Dendrogram

Insert Samples from Another Experiment

PCA, PCoA, t-SNE

Save Dendrogram

The following samples were removed because they contained no peaks:

172-11

172-1

172-10

Suggestions for Reporting Protein Analysis:
This dendrogram was created by analyzing 3 samples, and retaining peaks with a signal to noise ratio above 4 and occurring in greater than 70 % of replicate spectra. Peaks occurring below 3000 m/z or above 15000 m/z were removed from the analyses. For clustering spectra, distance and algorithms were used.

A screenshot of the IDBac Shiny app interface. The top navigation bar includes links for IDBac, Introduction, Starting with Raw Data, Work with Previous Experiments, Protein Data Analysis, and Small Molecule Data Analysis. The main content area has two tabs: 'Mirror Plots' (selected) and 'Dendrogram'. A message states: 'The following samples were removed because they contained no peaks:' followed by sample IDs 172-11, 172-1, and 172-10. To the left, there are two sections: 'Analysis Settings' containing 'Select Samples' and 'Choose Clustering Settings'; and 'Optional Settings' containing 'Adjust the Dendrogram' (underlined and circled), 'Insert Samples from Another Experiment', 'PCA, PCoA, t-SNE', and 'Save Dendrogram'. At the bottom, a section titled 'Suggestions for Reporting Protein Analysis' provides details about the dendrogram's creation, mentioning 3 samples, a signal-to-noise ratio of 4, and a retention threshold of 70% of replicate spectra. It also notes peak removal below 3000 m/z and above 15000 m/z, and the use of distance and clustering algorithms. The bottom of the screen shows a Windows taskbar with various icons and a search bar.



@ChasingMicrobes



Interactively Working with Disparate Data Sets

IDBac

IDBac Introduction Starting with Raw Data Work with Previous Experiments Protein Data Analysis Small Molecule Data Analysis

Show/Hide MAN Settings

Select a Sample to Subtract

Adjust Protein Dendrogram

[Download Current Network Data](#)

Hint 1: Use mouse to select parts of the tree and display the MAN of corresponding samples.

Hint 2: Use mouse to click & drag parts (nodes) of the MAN if it appears congested.

Note 1: For publication-quality networks click "Download Current Network Data". This will download a .csv file of the currently-displayed network that can be easily imported into Gephi or Cytoscape.

Suggestions for Reporting MAN Analysis:
This MAN was created by analyzing 3 samples, retaining peaks with a signal to noise ratio above 4, and occurring in greater than 70 % of replicate spectra. Peaks occurring below 200 m/z or above 2000 m/z were removed from the analysis.

Protein Dendrogram

Small Molecule MAN Small Molecule PCA Small Molecule Mirror Plot

Nodes in the MAN are colored by modularity score, not the same colors as the protein dendrogram.



@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]
# ... with 10 rows and 4 variables:
# $ peakMatrix: chr [1:10] "{\"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~,"..."]
# $ spectrum: blob [1:10]
# $ Intensi~: int [1:10] 375.28,374.61,82.38,80.59,308.50,323.60,79.39,78.46,389.08,387.35
# $ maxMass : num [1:10] 2698,2698,20924,20924,2698,2698,20924,20924,2698,2698
# $ minMass : num [1:10] 60,60,1919,1919,60,60,1919,1919,60,60
# ... 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
#       <chr>
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~
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~
3 "{\"mass\":[1919.93905,1925.91941,1946.20174,1959.50804,1973.48944,1987.52339,2000.97804,2015.11476,2028.66757,2042.26833,2056.7717,~
4 "{\"mass\":[1919.93905,1925.71302,1945.99421,1959.29976,1973.28037,1987.31354,2000.97804,2015.11476,2028.45543,2042.05545,2056.7717,~
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~
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~
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~
8 "{\"mass\":[1919.93905,1953.47268,1977.04532,1987.94312,2002.45255,2018.07534,2025.91077,2039.92729,2076.26532,2085.07985,2099.74247~
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~
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.~
# ... with more rows
```

spectrumIntensity	maxMass	minMass
<blob>	<int>	<int>
<raw 375.28 kB>	2698	60
<raw 374.61 kB>	2698	60
<raw 82.38 kB>	20924	1919
<raw 80.59 kB>	20924	1919
<raw 308.50 kB>	2698	60
<raw 323.60 kB>	2698	60
<raw 79.39 kB>	20924	1919
<raw 78.46 kB>	20924	1919
<raw 389.08 kB>	2698	60
<raw 387.35 kB>	2698	60



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
```

Storing compressed vectors in a SQL BLOB

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

```
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?



Recycle Bin



SpaceSniffer

Docker
Desktop

Python_3-7



RStudio



IDBac

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

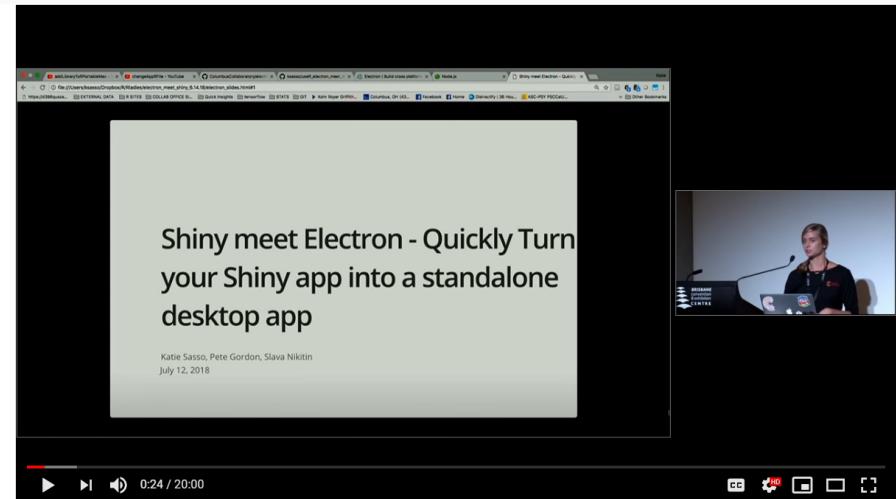
12:10 PM
3/25/2019

Creating/Distributing Local Shiny Applications



Only Windows
Can be Complicated
Not Self-Sufficient

github.com/ficonsulting/RInno



 [@KatieSasso](https://twitter.com/KatieSasso)

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

<https://www.youtube.com/watch?v=ARrbvviGvjc>

electricShine R Package

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>



@ChasingMicrobes

Thanks for Listening

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

Please empathize with your users.

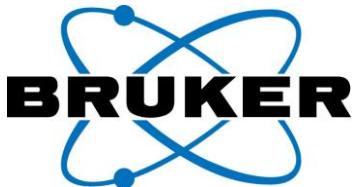
Please continue being a welcoming community.



Special Thanks



Dr. Laura Sanchez



MEDICINAL
CHEMISTRY
AND
PHARMACOGNOSY
COLLEGE
OF PHARMACY

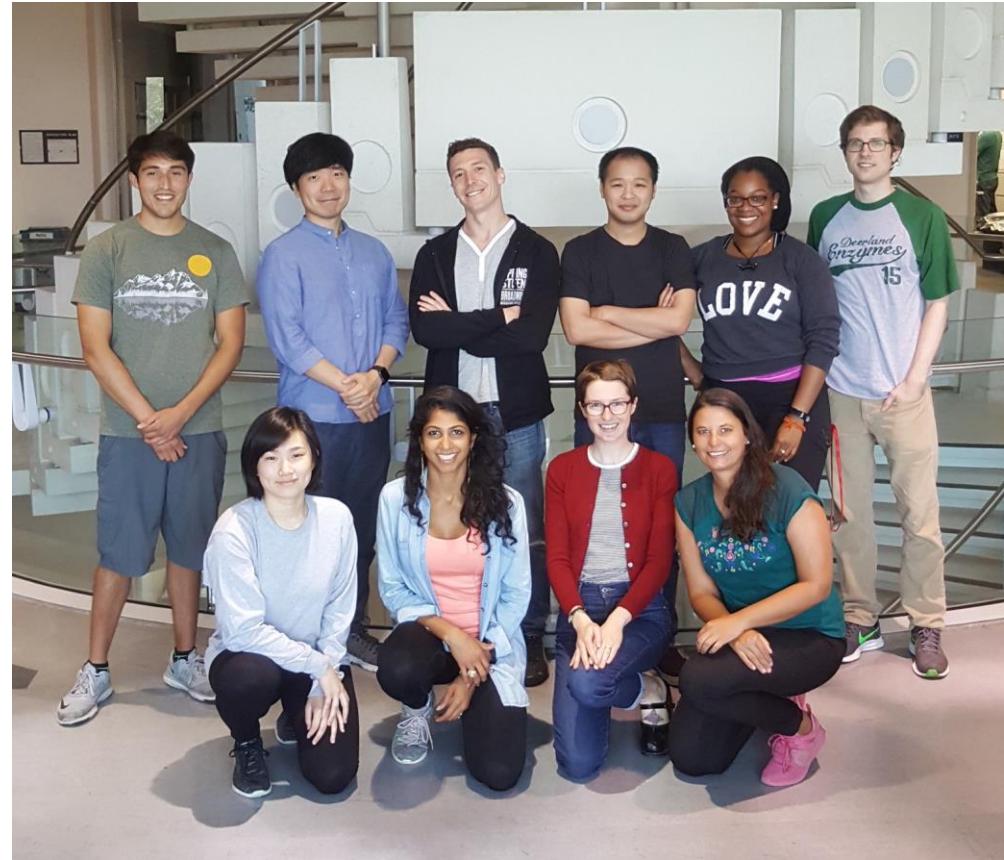


Murphy Lab

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



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



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

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.