Rensselaer

why not change the world?®

What They Forgot to Teach You about R!

The stuff you need to know about R, besides data analysis
Wednesday, 28 Feb 2024

RPIrates: The RPI R Users Group
The Rensselaer IDEA
Rensselaer Polytechnic Institute

IDEA
Rensselaer Institute for Data Exploration and Applications

RPIrates
The RPI R Users Group

Tall

"What They Forgot to Teach You About R" (The Book)

What They Forgot to Teach You About R

What They Forgot to Teach You About R

The stuff you need to know about R, besides data analysis.

AUTHORS

Jennifer Bryan

Jim Hester

Shannon Pileggi

E. David Aja

1 What They Forgot to Teach You About R

Warning

This book is a work in progress.

1.1 About

This book focuses on content intrinsically related to the infrastructure surrounding data analysis in R, but does not delve into the data analysis itself.

1. *A holistic workflow* provides guidance on project-oriented workflows that address common sources of friction in data analysis.
2. *Personal R administration* empowers R users to confidently manage their R programming environment.
3. *All is Fail* showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

This book focuses on content intrinsically related to the **infrastructure** surrounding data analysis in R, but **does not** delve into the data analysis itself.

"What They Forgot to Teach You About R" (The Book)

What They Forgot to Teach You About R

What They Forgot to Teach You About R

The stuff you need to know about R, besides data analysis.

AUTHORS
Jennifer Bryan
Jim Hester
Shannon Pileggi
E. David Aja

1 What They Forgot to Teach You About R

Warning

This book is a work in progress.

1.1 About

This book focuses on content intrinsically related to the infrastructure surrounding data analysis in R, but does not delve into the data analysis itself.

1. *A holistic workflow* provides guidance on project-oriented workflows that address common sources of friction in data analysis.
2. *Personal R administration* empowers R users to confidently manage their R programming environment.
3. *All is Fail* showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

This book focuses on content intrinsically related to the **infrastructure** surrounding data analysis in R, but **does not** delve into the data analysis itself.

A holistic workflow provides guidance on project-oriented workflows that address common sources of friction in data analysis.

"What They Forgot to Teach You About R" (The Book)

What They Forgot to Teach You About R

What They Forgot to Teach You About R

The stuff you need to know about R, besides data analysis.

AUTHORS
Jennifer Bryan
Jim Hester
Shannon Pileggi
E. David Aja

1 What They Forgot to Teach You About R

Warning

This book is a work in progress.

1.1 About

This book focuses on content intrinsically related to the infrastructure surrounding data analysis in R, but does not delve into the data analysis itself.

1. *A holistic workflow* provides guidance on project-oriented workflows that address common sources of friction in data analysis.
2. *Personal R administration* empowers R users to confidently manage their R programming environment.
3. *All is Fail* showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

This book focuses on content intrinsically related to the **infrastructure** surrounding data analysis in R, but **does not** delve into the data analysis itself.

- **A holistic workflow** provides guidance on project-oriented workflows that address common sources of friction in data analysis.
  Personal R administration empowers R users to confidently manage their R programming environment.

"What They Forgot to Teach You About R" (The Book)

What They Forgot to Teach You About R

What They Forgot to Teach You About R

The stuff you need to know about R, besides data analysis.

AUTHORS
Jennifer Bryan
Jim Hester
Shannon Pileggi
E. David Aja

1 What They Forgot to Teach You About R

Warning

This book is a work in progress.

1.1 About

This book focuses on content intrinsically related to the infrastructure surrounding data analysis in R, but does not delve into the data analysis itself.

1. *A holistic workflow* provides guidance on project-oriented workflows that address common sources of friction in data analysis.
2. *Personal R administration* empowers R users to confidently manage their R programming environment.
3. *All is Fail* showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

This book focuses on content intrinsically related to the **infrastructure** surrounding data analysis in R, but **does not** delve into the data analysis itself.

- **A holistic workflow** provides guidance on project-oriented workflows that address common sources of friction in data analysis.
- **Personal R administration** empowers R users to confidently manage their R programming environment.
- **All is Fail** showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

"What They Forgot to Teach You About R" (The Book)

What They Forgot to Teach You About R

What They Forgot to Teach You About R

The stuff you need to know about R, besides data analysis.

AUTHORS
Jennifer Bryan
Jim Hester
Shannon Pileggi
E. David Aja

1 What They Forgot to Teach You About R

Warning

This book is a work in progress.

1.1 About

This book focuses on content intrinsically related to the infrastructure surrounding data analysis in R, but does not delve into the data analysis itself.

1. *A holistic workflow* provides guidance on project-oriented workflows that address common sources of friction in data analysis.
2. *Personal R administration* empowers R users to confidently manage their R programming environment.
3. *All is Fail* showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.

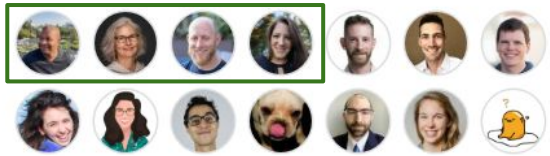
This book focuses on content intrinsically related to the **infrastructure** surrounding data analysis in R, but **does not** delve into the data analysis itself.

- **A holistic workflow** provides guidance on project-oriented workflows that address common sources of friction in data analysis.
- **Personal R administration** empowers R users to confidently manage their R programming environment.
- **All is Fail** showcases functions, options, and RStudio capabilities for debugging code, facilitating more efficient resolution of errant code.
- **Session Info** provides tips on viewing the state of your session

"What They Forgot to Teach You About R" (The Book)

https://rstats.wtf/

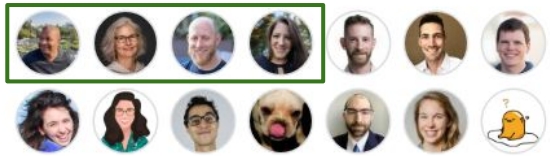
https://github.com/rstats-wtf/what-they-forgot



"What They Forgot to Teach You About R" (The Book)

https://rstats.wtf/

https://github.com/rstats-wtf/what-they-forgot



What has **seven years** of Data INCITE with R taught us?

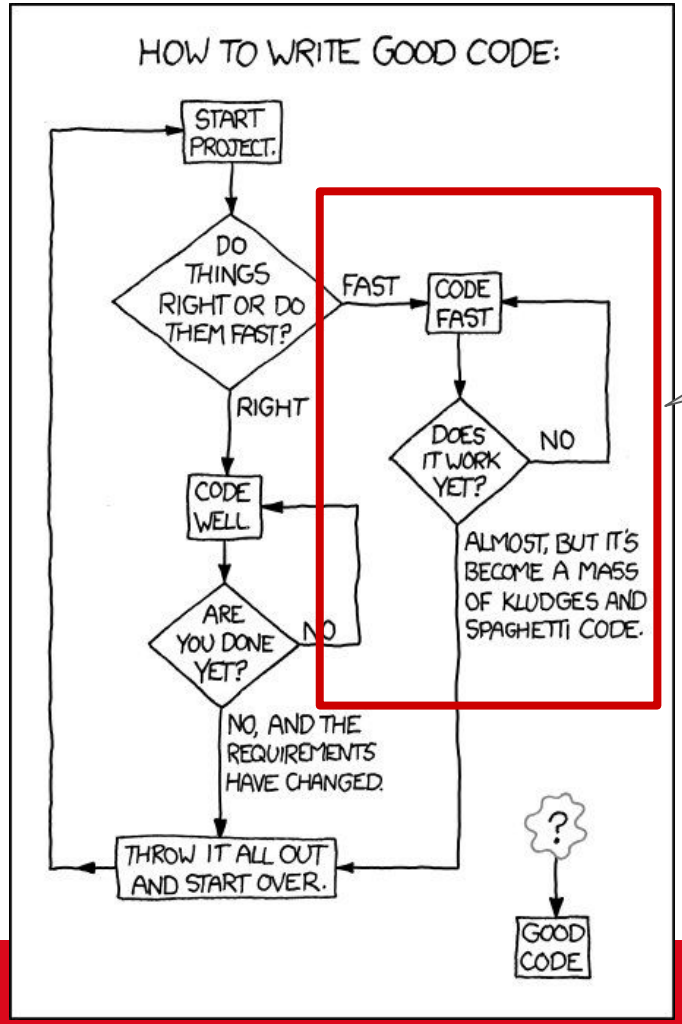
Rensselaer

IDEA
Rensselaer Institute for Data Exploration and Applications

RPIrates
The RPI R Users Group

What has **seven years** of Data INCITE with R taught us?

- Coding in **source files** is critical: .R *and/or* .Rmd *plus* github
- Interactive coding makes for **bad** environment and workspace habits
- Most new R users don't understand **file paths**
- Thoughtful **thing-naming** -- variables, dataframes, files -- makes life easier!
- Having a **project-oriented view** from the start makes life easier at the end!
- R sometimes behaves weirdly; **"clean living"** can avoid those occasions
- For long-time users, it's critical to **keep up-to-date** (R, RStudio, packages)
- Most users don't have a **strategy for debugging**
- Most users don't **communicate errors well** (ie "reproducible examples")



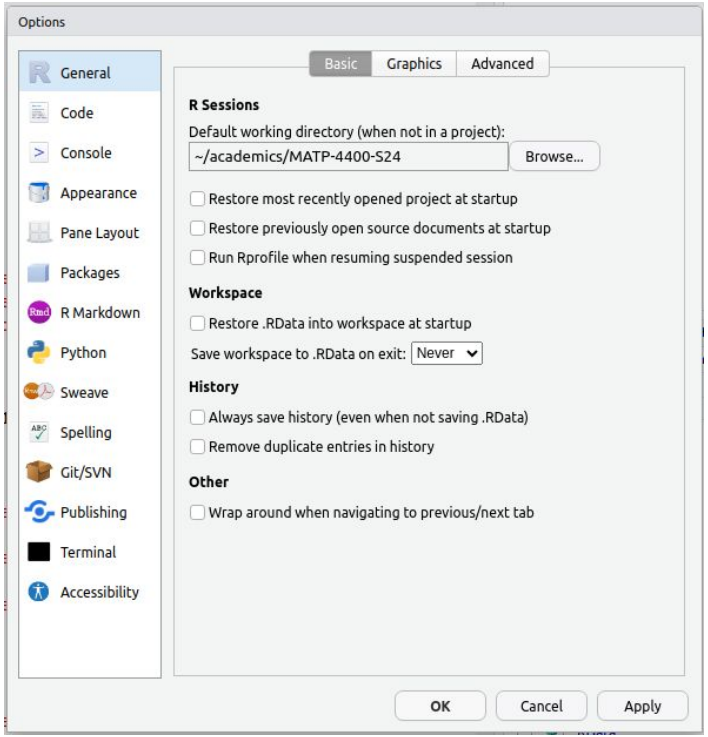
Most students are in this zone…

Save your source, not your workspace

- Do your coding in .R ("script") or .Rmd ("markdown" or "notebook") files
- RStudio spoils users by **aggressively saving state**
  - Makes it easy to walk away and come back to where you left off
  - Workspace (e.g. editing session, open files) and Environment (data and functions)
- Everything you do in the console goes into the environment
  - Esp. hacking variables or dataframes to make some code work
- This is the #1 reason why "My code works in RStudio but doesn't knit!"
  - Knitr only runs the code in your notebook and ignores your environment
- Saving your workspace (ie as mywork.RData) is rarely useful

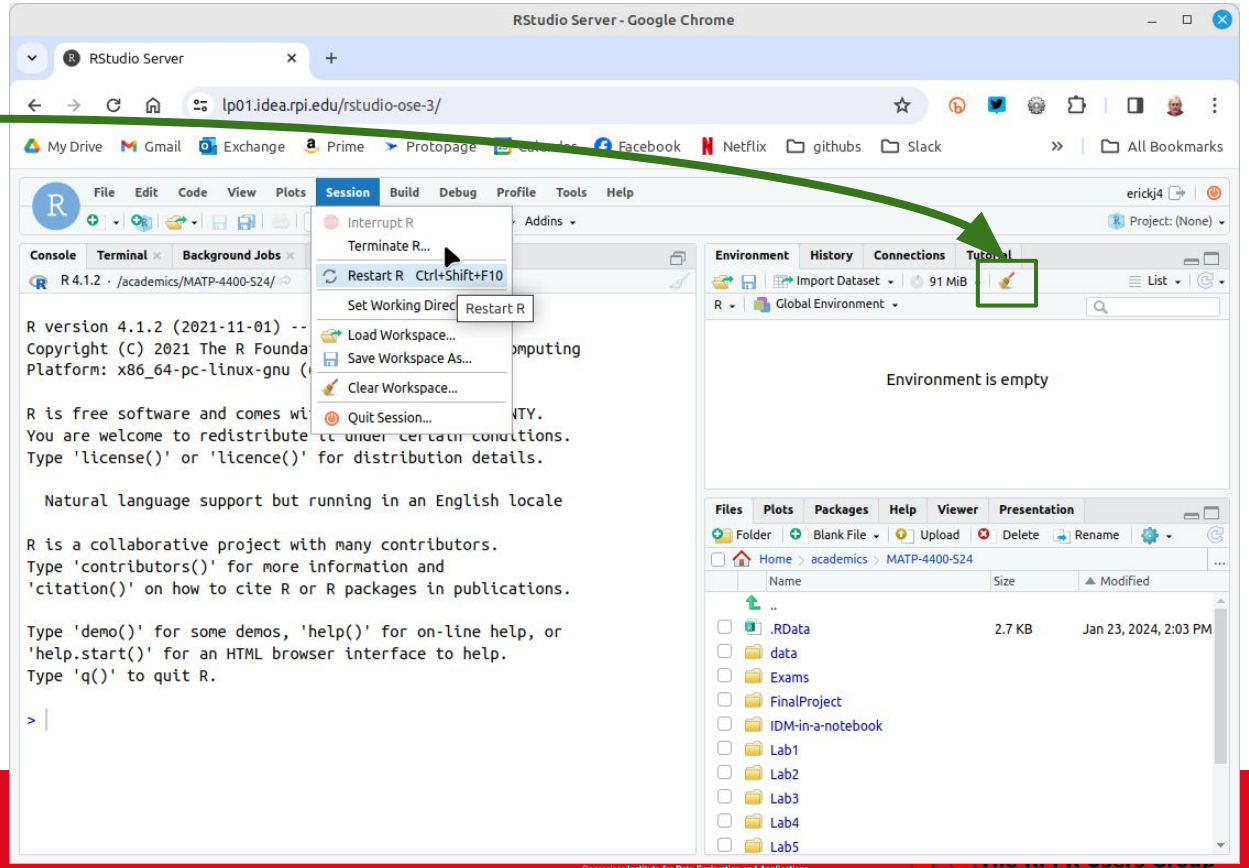
Always start R with a clean slate

- **Sweep** your Environment and restart your R session **early and often**
  - These are the first steps to debugging the "it won't knit" problem
  - Restarting your session only doesn't clear your environment
- The "pros" -- people who develop packages -- always start clean
- Think like a pro; start every work session with a completely blank slate!



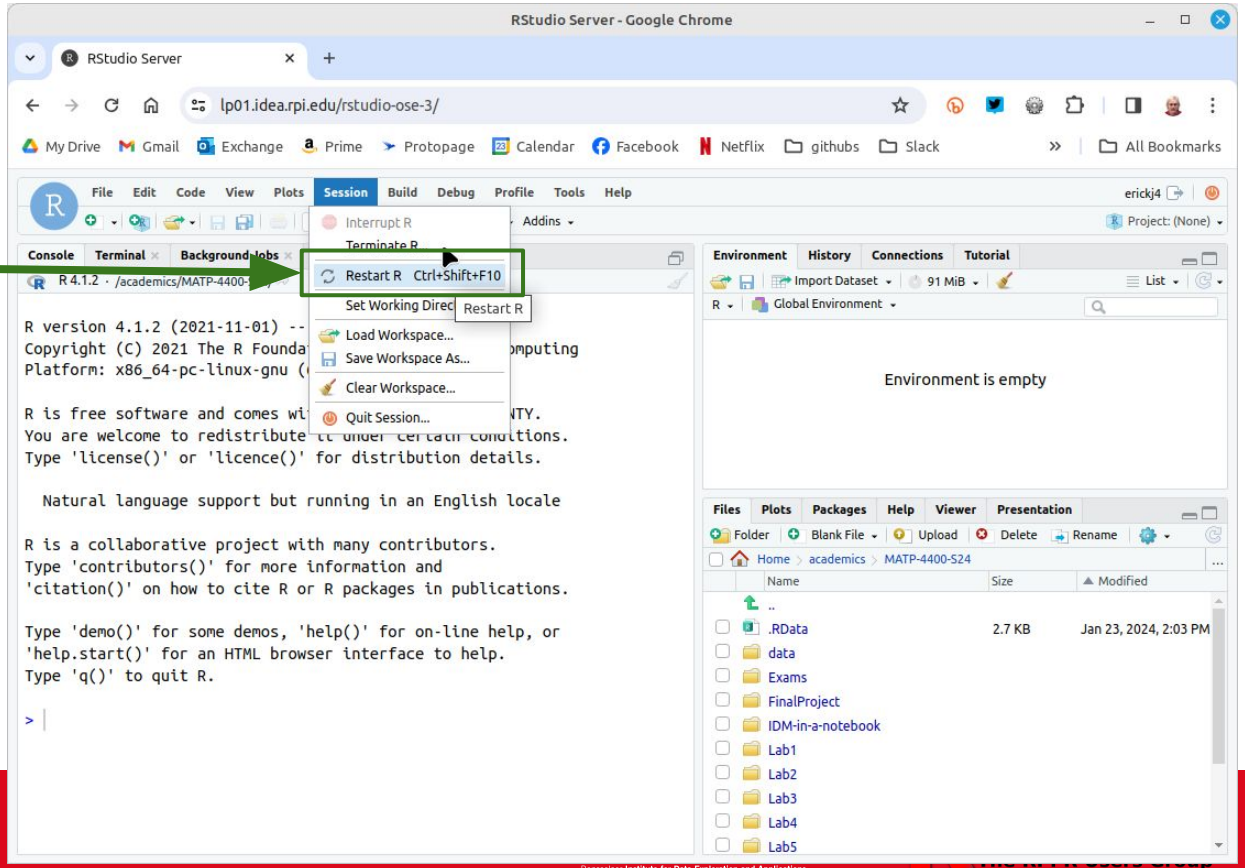
Restart R often during development

- Clear your environment and/or…



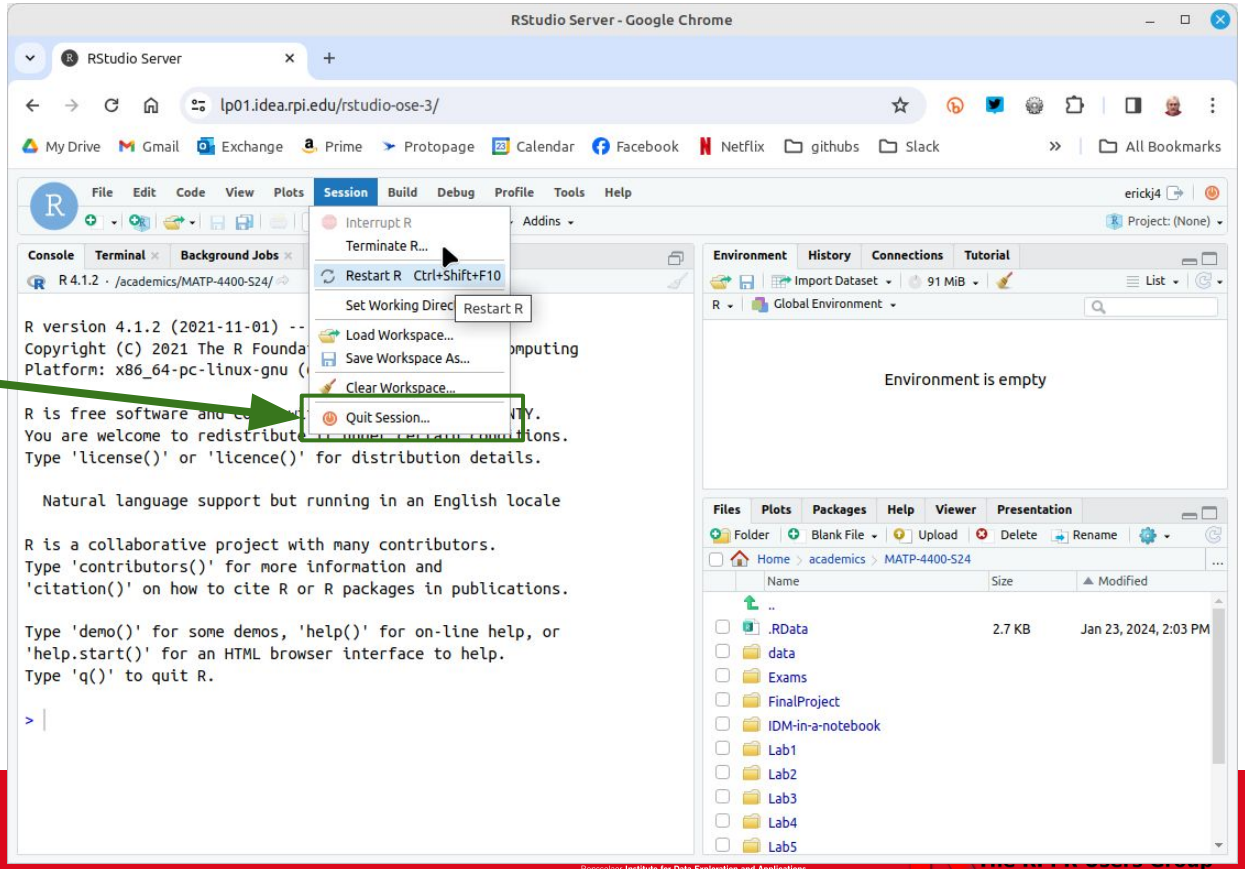
Restart R often during development

- Clear your environment and/or…
- Restart your R session or…



Restart R *often* during development

- Clear your environment and/or…
- Restart your R session or…
- Quit session



Objects that take a long time to create

- Parsing "raw" CSV files and running data preparation pipelines takes time
  - Loading a **pre-cooked binary** version of data is much faster!!
- Repeating data prep across teams is **wasteful** and sometimes **dangerous**
- **Data INCITE best practice:** Once the data prep is done, save out as .Rds
  - Create "helper" scripts or notebooks that read in and transform the data into dataframes
  - In those files, save the dataframes out to .Rds, e.g. saveRDS(myData, "myData.Rds")
  - In your analysis files, simply load the .Rds: myData <- readRDS("myData.Rds")
  - Share these .Rds files across your team (shared directory or github)
- Store these .Rds files in your project's github
  - Very large data may require github large file storage: https://git-lfs.com/

We need to talk about setwd("path/that/only/works/on/my/machine")

- Always assume your code might end up anywhere
  - Sharing scripts with teammates
  - Sharing github repositories with with colleagues
  - Cloning repos on different machines
- Use relative paths that don't depend on locality
  - Use the Linux "start from here" syntax, e.g. source("./utilities/myHelper.R")
- If you must use absolute paths, make sure they're valid in all cases!
  - Example: faces <- read_csv("/academics/MATP-4400-S24/data/faces.csv") (okay on Cluster)

Organize work into "projects"

A simple project

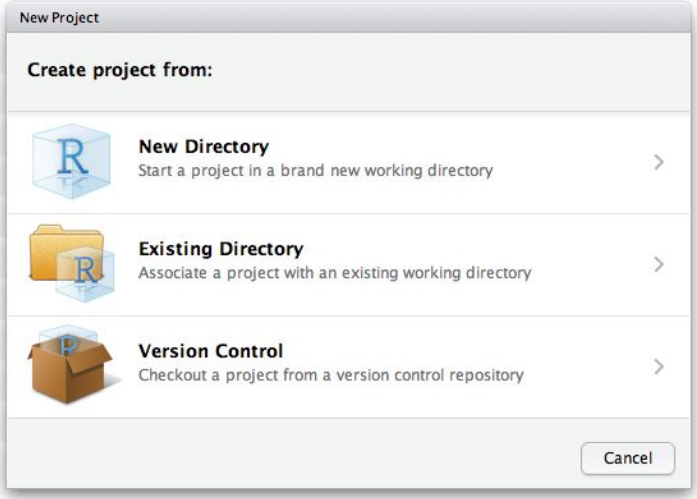
```
.
├── app.R
├── README.md
├── rest
│   ├── app.py
│   ├── Dockerfile
│   ├── README.md
│   └── requirements.txt
└── www
    ├── enter_button.js
    ├── README
    └── RensselaerLogo_black.png
```

A more complex project

```
.
├── app.R
├── data
│   ├── cong_districts
│   ├── general
│   ├── lowmod
│   ├── reservoirs
│   └── substations
├── dataset_generator.R
├── FIPS2STATENAME.csv
├── FIPS2STATENAME.Rds
├── README.md
├── rsconnect
│   └── shinyapps.io
├── STATENAME2ABB.Rds
└── www
    ├── 20190731-nrel-fpv-install-53282.jpg
    ├── IDEA_logo_500.png
    ├── substation.png
    └── usa_reservoirs.png
```

```
.
├── cong_districts
│   ├── alabama.Rds
│   ├── alaska.Rds
│   :
│   :
│   ├── wisconsin.Rds
│   └── wyoming.Rds
├── general
│   ├── district_elections.Rds
│   ├── districts.Rds
│   ├── electric_substations.csv
│   ├── national_lmi.Rds
│   ├── reservoirs.Rds
│   ├── STATENAME2ABB.csv
│   ├── STATENAME2ABB.Rds
│   └── Tract2District.Rds
├── lowmod
│   ├── alabama.Rds
│   ├── alaska.Rds
│   :
│   :
│   ├── wisconsin.Rds
│   └── wyoming.Rds
├── reservoirs
│   ├── alabama.Rds
│   ├── alaska.Rds
│   :
│   :
│   ├── wisconsin.Rds
│   └── wyoming.Rds
└── substations
    ├── alabama.Rds
    ├── alaska.Rds
    :
    :
    ├── wisconsin.Rds
    └── wyoming.Rds
```

RStudio Projects



- RStudio has built-in **project capabilities**
- The advantage is that it stores the state of your work in a .Rproj file, and makes it easy to resume:
  - A new R session (process) is started
  - The .Rprofile file in the project's main directory (if any) is sourced by R
  - The .RData file in the project's main directory is loaded
    - …if project options indicate that it should be loaded
  - The .Rhistory file in the project's main directory is loaded into the RStudio History pane
    - …and used for console Up/Down arrow command history
  - The current working directory is set to the **project directory**
  - Previously edited source documents are **restored** into editor tabs
  - Other RStudio settings (e.g. active tabs, splitter positions, etc.) are restored to where they were the last time the project was closed.

How to name files (and other stuff…)

See Jenny Bryan, **naming things**

three principles for (file) names

machine readable

human readable

plays well with default ordering

See Jenny Bryan, **naming things**

NO

myabstract.docx
Joe's Filenames Use Spaces and Punctuation.xlsx
figure 1.png
fig 2.png
JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt

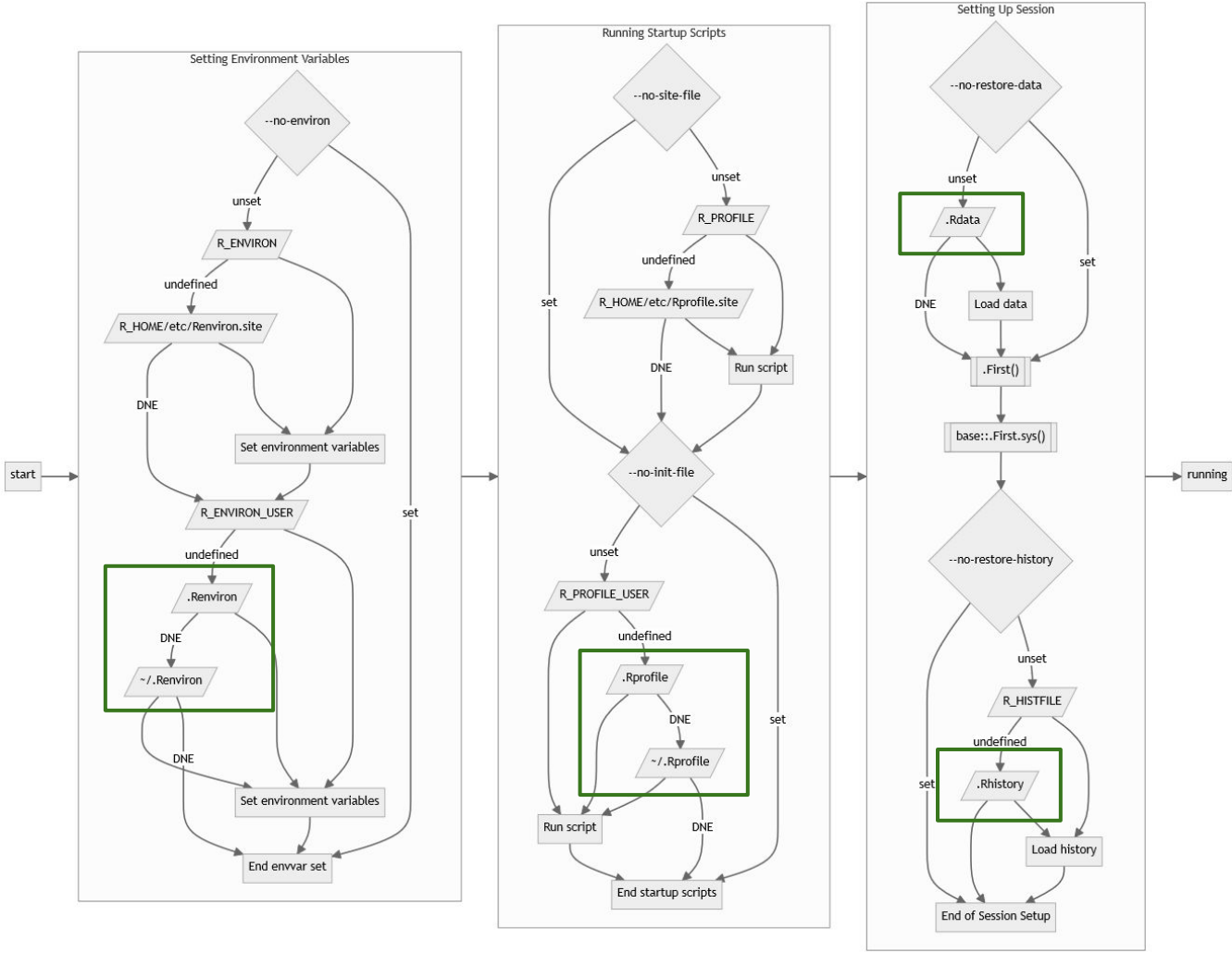
YES

2014-06-08_abstract-for-sla.docx
joes-filenames-are-getting-better.xlsx
fig01_scatterplot-talk-length-vs-interest.png
fig02_histogram-talk-attendance.png
1986-01-28_raw-data-from-challenger-o-rings.txt

See Jenny Bryan, **naming things**

R's startup procedures are complex…

See [this]



Summary of how to control R options and environment variables on startup

File	Who Controls	Level	Limitations
`.Rprofile`	User or Admin	User or Project	None, sourced as R code.
`.Renviron`	User or Admin	User or Project	Set environment variables only.
`Rprofile.site`	Admin	Version of R	None, sourced as R code.
`Renviron.site`	Admin	Version of R	Set environment variables only.
`rsession.conf`	Admin	Server	Only RStudio IDE settings, only single repository.
`repos.conf`	Admin	Server	Only for setting repositories.

.Renviron

- .Renviron is a **user-controllable file** that can be used to create environment variables
- Especially useful to avoid including credentials like API keys inside R scripts
- Written in a key-value format, so environment variables are created as follows:
  - Key1=value1
  - Key2=value2
  - ...
- Sys.getenv("Key1") will return "value1" in an R session.
- As with the .Rprofile file, .Renviron files can be at either the user or project level.
  - If there is a project-level .Renviron, the user-level file will not be sourced.
  - The usethis package includes a helper function for editing .Renviron files from an R session with usethis::edit_r_environ()

.Rprofile

- .Rprofile files are user-controllable files to set options and environment variables.
- .Rprofile files can be either at the **user** or **project** level.
  - **User-level** .Rprofile files live in the base of the user's home directory
  - **Project-level** .Rprofile files live in the base of the project directory
- **R will source only one .Rprofile file**
  - If you have both a project-specific .Rprofile file and a user .Rprofile file that you want to use, you must explicitly source the user-level .Rprofile at the top of your project-level .Rprofile with source("~/.Rprofile")
- .Rprofile files are sourced as **regular R code**; setting environment variables must be done inside a Sys.setenv(key = "value") call
- An easy way to edit .Rprofile files is to use the usethis::edit_r_profile() function from within an R session. You can specify whether you want to edit the user or project level .Rprofile

Reproducible Environments

The renv Package

- The renv package helps you create **reproducible environments** for your R projects.
- Use renv to make your R projects more **isolated**, **portable** and **reproducible**.
  - **Isolated:** Installing a new or updated package for one project won't break your other projects, and vice versa. That's because renv gives each project its own private library.
  - **Portable:** Easily transport your projects from one computer to another, even across different platforms. renv makes it easy to install the packages your project depends on.
  - **Reproducible:** renv records the exact package versions you depend on, and ensures those exact versions are the ones that get installed wherever you go.

Maintaining R

Common package installation issues

- "ERROR: failed to create lock directory"
  - R is unable to create the 00LOCK-<package> directory it needs for installation
  - Most of the time, this is due to a failed previous installation attempt (e.g. interrupted)
  - **SOLUTION:** In RStudio's Linux terminal, delete the existing directory
  - R tells you what to delete in the error message
- "package is not available for this version of R"
  - install.packages() cannot find a CRAN binary for the installed version of R
  - **SOLUTION:** Install the package from source (see other slides)
- Package install failed because installed dependencies are for older R
  - You installed the dependencies under a previous version of R and they're still lurking
  - Packages built under different versions of R can't co-exist
  - **SOLUTION:** In RStudio's Linux terminal, delete the old package installs:
    $ rm -Rf /home/**RCSID**/R/x86_64-pc-linux-gnu-library/**X.Y**

Common package installation issues

- "ERROR: failed to create lock directory"
  - R is unable to create the 00LOCK-<package> directory it needs for installation
  - Most of the time, this is due to a failed previous installation attempt (e.g. interrupted)
  - **SOLUTION:** In RStudio's Linux terminal, delete the
  - R tells you what to delete in the error message
- "package is not available for this version of R
  - install.packages() cannot find a CRAN binary for th ailed version of R
  - **SOLUTION:** Install the package from source (other slides)
- Package install failed because inst d dependencies
  - You installed the dependencies u r a previous version of R and th still lurking
  - Packages built under differe versions of R can't co-exist
  - **SOLUTION:** In RStudio's Linux terminal, delete the old package installs:
    $ rm -Rf /home/**RCSID**/R/x86_64-pc-linux-gnu-library/**X.Y**

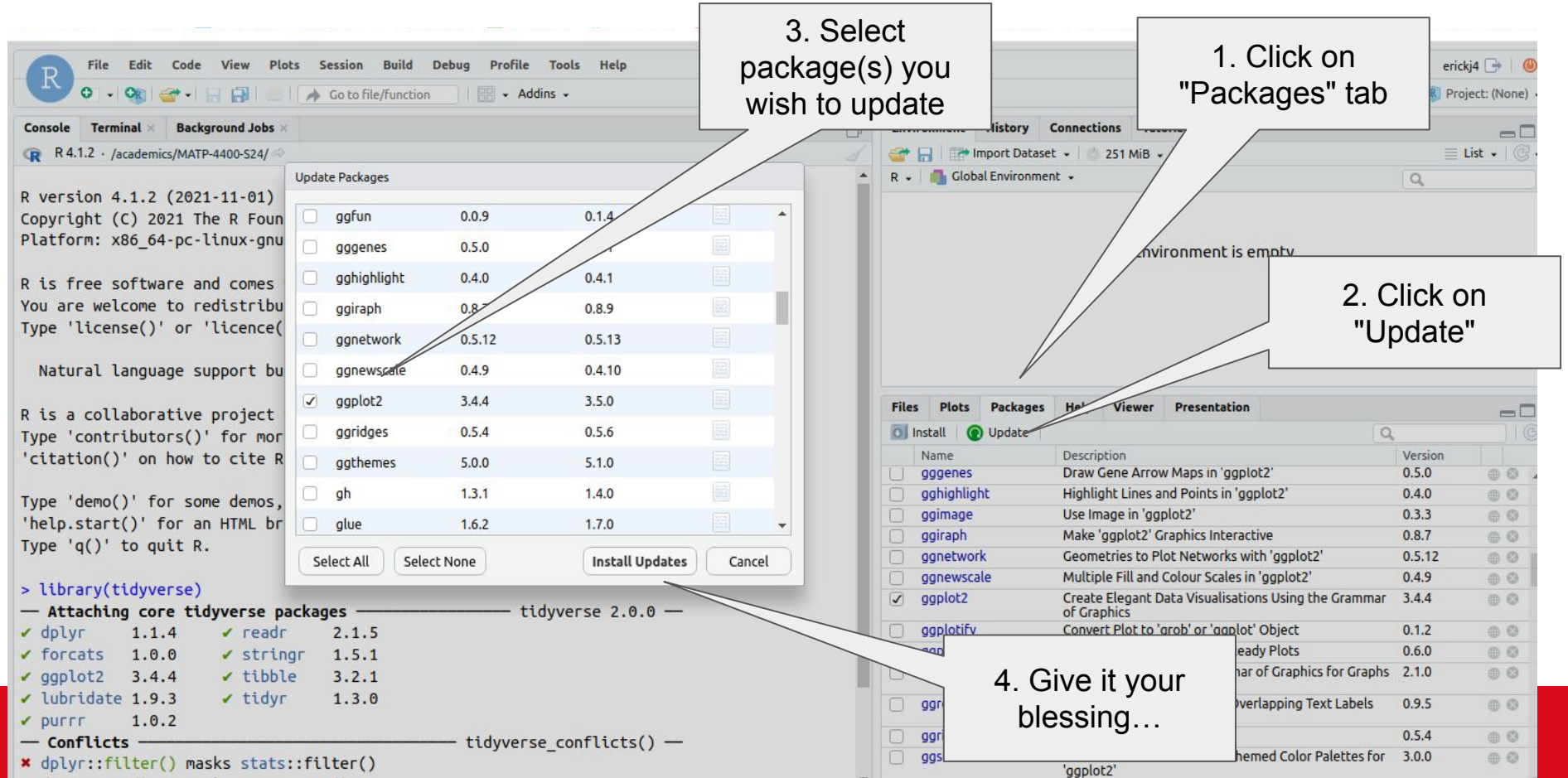
Your RCS username

Old R version number
e.g. 3.6 or 4.1 or…

How to install packages from source

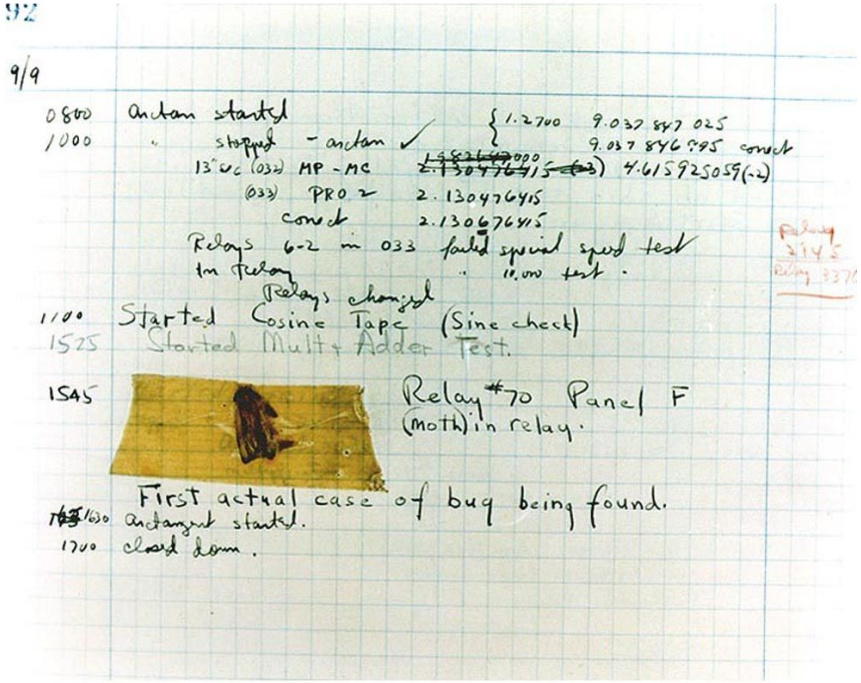
- The most common type of package you install is a **binary package**.
  - Packages released on CRAN are built as pre-compiled binaries for **specific versions of R**
- It is sometimes useful to install packages which **do not have a pre-built binary** available
  - e.g. development versions not yet released on CRAN
  - e.g. older versions of released packages
  - e.g. packages you've built locally
- There are a few main functions used to install source packages.
  - devtools::install_dev() to install the latest dev version of a CRAN package
  - devtools::install_github() or devtools::install_git() to install any package directly from GitHub
  - devtools::install_version() to install previously released CRAN versions of a package.
- You can also install the official CRAN version from source:
  - install.packages(path_to_file, repos = NULL, type="source")
  - …where "path_to_file" is listed on the package's CRAN page,
    e.g. for dplyr: https://cran.r-project.org/src/contrib/dplyr_1.1.4.tar.gz

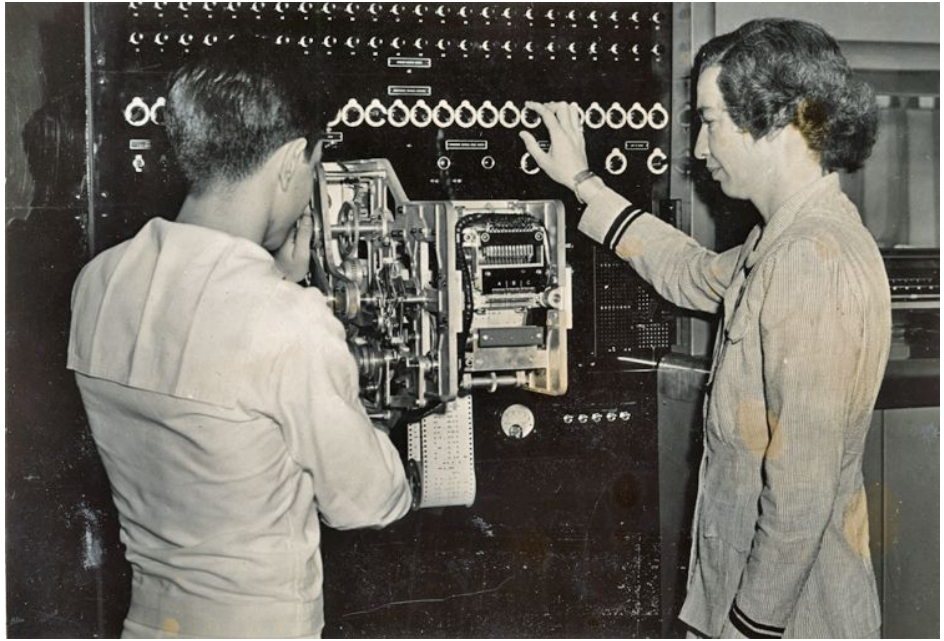
How to upgrade an installed package to the latest version



Debugging R code

Photo # NH 96566-KN (Color) First Computer "Bug", 1947





Debugging your code

- Sometimes bugs only appear after multiple levels of calls and are hard to diagnose.
- There are a few common strategies to use when debugging your code:
  - Use traceback() to determine where a given error is occurring.
  - Output diagnostic information in code with print(), cat() or message() statements.
  - Use str() to sanity-check the structure of objects
  - Use browser() to open an interactive debugger before the error
  - Use debug() to automatically open a debugger at the start of a function call.
  - Use trace() to start a debugger at a location inside a function.

More on browser()

```
16
17      # generate bins based on input$bins from ui.R
18      x    <- faithful[, 2]
19      browser()
20      bins <- seq(min(x), max(x), length.out = input$bins + 1)
21
```

- browser() is extremely useful for debugging R (esp. Shiny code)
- Insert a call to browser() in your code to stop execution at that point and open an interactive debugger.
  - Works just like the R console
  - In the browser console you can run R commands to examine at objects in the current environment, modify objects and continue executing.
- Some useful things to do:
  - Use ls() to determine what objects are available in the current environment
    - This allows you to see exactly what things you can examine
  - Use str(), print() etc. to examine the objects
  - Use n to evaluate the next statement. Use s to evaluate the next statement, but step into function calls.
  - Use where to print a stack trace
  - Use c to leave the debugger and continue execution
  - Use Q to exit the debugger and return to the R prompt.

Debugging in RStudio: Breakpoints

- In the RStudio editor you can set an editor breakpoint by **clicking to the left** of the line number in the source file
- A breakpoint is equivalent to a browser() call, but you avoid needing to change your code like browser().

```
18  best <- 0
19  for (x in 100:999) {
20    for (y in x:999) {
21      candidate <- x * y
22      if (candidate > best && palindrome(candidate)) {
23        best <- candidate
24      }
25    }
26  }
```

Console ~/r/pkg/rmarkdown/

Next | Continue | Stop

```
Browse[2]> Q
> rmarkdown::find_external_resources("~/rmd/alice.Rmd")
Called from: eval(expr, envir, enclos)
Browse[1]> n
debug at /Users/jmcphers/r/pkg/rmarkdown/R/html_resources.R#143: d
iscover_single_resource(res_file, FALSE, TRUE)
Browse[2]>
```

Debugging in R Markdown documents

- If your code "works" interactively but doesn't knit, try the following:
  - **Sweep** your environment
  - **Restart** your R session
  - **Run all** or **single-step** through your code chunks, playing close attention to your Environment
  - Most of the time, you fixed a problem interactively but didn't save your changes in your code
- The easiest way to debug most errors is to run the code chunk by itself
  - Be careful to **Run all chunks above** before running the problem chunk
  - For complex code, insert browser() and/or use the other methods
- **Don't debug by repeatedly knitting!!!**
  - You need to isolate the problem by stepping through the code and examining your internal structures as they are transformed

Creating a "reprex" (reproducible examples)

- R people can't help you if they can't **understand** and **replicate** your problem
- The universal currency for getting help is the **reprex**
- Generally, a reprex is a **sample of code** that reproduces the problem
- **Do not** ask for help without showing code that illustrates the problem!
- **Do not** paste only your error message!
- **Do not** simply say "Knitting failed; why?!"
- A package exists for creating beautiful R reprex's:
  See: https://reprex.tidyverse.org/

sessionInfo()

```
> sessionInfo()
R version 4.1.2 (2021-11-01)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS/LAPACK: /usr/lib64/libopenblasp-r0.3.3.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] reshape_0.8.9   ggrepel_0.9.5   ggbiplot_0.55   scales_1.3.0    plyr_1.8.8
[6] pheatmap_1.0.12 titanic_0.1.0   ggplot2_3.4.4

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.12        tidyr_1.3.0        prettyunits_1.1.1  ps_1.7.5
 [5] digest_0.6.34      utf8_1.2.3         mime_0.12          R6_2.5.1
 [9] evaluate_0.21      pillar_1.9.0       rlang_1.1.2        rstudioapi_0.14
[13] miniUI_0.1.1.1     callr_3.7.3        urlchecker_1.0.1   jquerylib_0.1.4
[17] rmarkdown_2.25     devtools_2.4.5     stringr_1.5.1      htmlwidgets_1.6.2
[21] igraph_1.4.3       munsell_0.5.0      shiny_1.8.0        compiler_4.1.2
[25] httpuv_1.6.11      xfun_0.39          pkgconfig_2.0.3    pkgbuild_1.3.1
[29] htmltools_0.5.5    tidyselect_1.2.0   tibble_3.2.1       fansi_1.0.4
[33] crayon_1.5.2       dplyr_1.1.4        withr_2.5.0        later_1.3.1
[37] jsonlite_1.8.8     xtable_1.8-4       gtable_0.3.3       lifecycle_1.0.4
[41] magrittr_2.0.3     cli_3.6.1          stringi_1.8.3      cachem_1.0.8
[45] fs_1.6.3           promises_1.2.1     remotes_2.4.2      bslib_0.4.2
[49] ellipsis_0.3.2     generics_0.1.3     vctrs_0.6.4        RColorBrewer_1.1-3
[53] tools_4.1.2        glue_1.6.2         purrr_1.0.2        rsconnect_1.2.0
[57] processx_3.8.1     pkgload_1.3.2      fastmap_1.1.1      yaml_2.3.7
[61] colorspace_2.1-0   sessioninfo_1.2.2  tidygraph_1.2.3    memoise_2.0.1
[65] knitr_1.45         profvis_0.3.7      sass_0.4.7         usethis_2.1.6
```

Running on the Cluster!

Attached packages

Installed non-"base" packages