Project Organization

Abhijit Dasgupta

November 13, 2019

Objectives today

- Project Organization
  - How to maintain long-term sanity
- Project Reporting
  - Rich documents using RMarkdown

Why organize?

Common Objectives

- Maximize
  - Time to think about a project
  - Reliability/Reproducibility
- Minimize
  - Data errors
  - Programmer/Analyst errors
  - Programming Time
  - Re-orientation time when revisiting

Our inclination

- Once we get a data set
    - Dig in!!
    - Start "playing" with tables and figures
    - Try models on-the-fly
    - Cut-and-paste into reports and presentations

DON'T DO THIS!!

Abhijit's story

Eight years ago

- 25 year study of rheumatoid arthritis
- 5600 individuals
- Several cool survival analysis models
- Needed data cleaning, validation and munging, and some custom computations
- Lots of visualizations

Eight years ago

- Resulted in a muddle of 710 files (starting from 4 data files)
- Unwanted cyclic dependencies for intermediate data creation
- Lots of ad hoc decisions and function creation with scripts
- Almost impossible to re-factor and clean up
- Had to return to this project for 3 research papers and revision cycles!!!

Who's the next consumer of your work

- Yourself in
    - 3 months
    - 1 year
    - 5 years
- Can't send your former self e-mail asking what the f**k you did.

Biggest reason for good practices is

YOUR OWN SANITY

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

RStudio Projects

When you create a Project, the following obvious things happen:

1. RStudio puts you into the right directory/folder
2. Creates a .Rproj file containing project options
    - You can double-click on the .Rproj file to open the project in RStudio
3. Displays the project name in the project toolbar (right top of the window)

RStudio Projects

The following not-so-obvious things happen:

1. A new R session (process) is started
2. The .Rprofile file in the project's main directory (if any) is sourced by R
3. The .RData file in the project's main directory is loaded (this can be controlled by an option).
4. 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).
5. The current working directory is set to the project directory.
6. Previously edited source documents are restored into editor tabs, and
7. Other RStudio settings (e.g. active tabs, splitter positions, etc.) are restored to where they were the last time the project was closed.

RStudio Projects

I use Projects so that:

1. I'm always in the right directory for the project
2. I don't contaminate one project's analysis with another (different sandboxes)
3. I can access different projects quickly
4. I can version control them (Git) easily (topic for beyond this class)
5. I can customize options per project

RStudio Projects

Project organization

Project structure

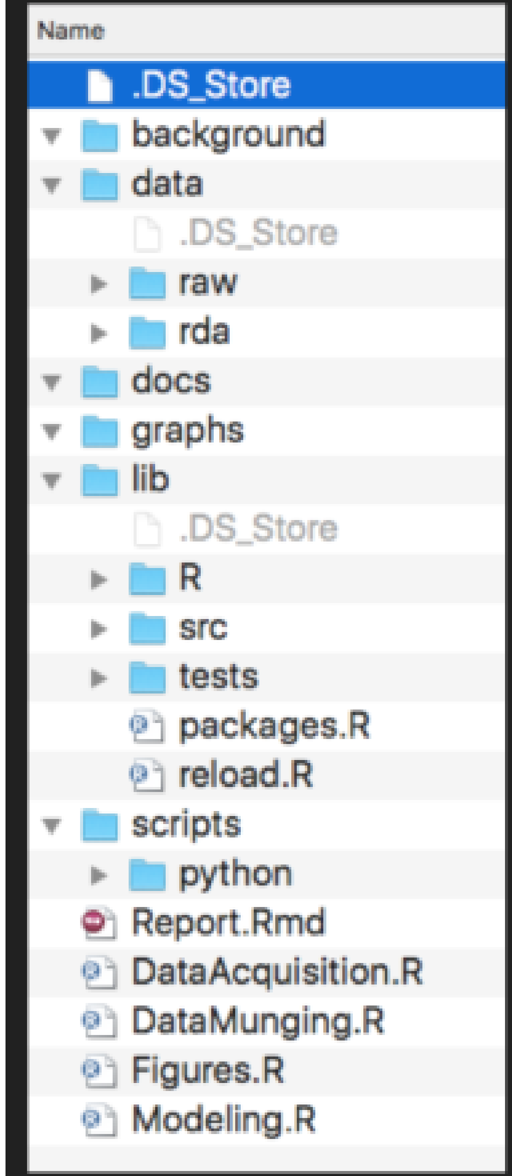
I always work with RStudio Projects to encapsulate my projects.

However, each project needs to maintain a file structure to know where to find things

Use a template to organize each project

- Before you even get data
- Set up a particular folder structure where
    - You know what goes where
    - You already have canned scripts/packages set up
- Make sure it's the same structure **every time**
- Next time you visit, you don't need to go into desperate search mode

MY STRUCTURE



Name	
.DS_Store	
background	Background materials
data	
.DS_Store	
raw	Raw data (storage, not to be touched)
rda	Intermediate and final R data sets
docs	Generated documents (docx, html, pdf)
graphs	Graphs (pdf, png, tiff)
lib	
.DS_Store	
R	Custom R functions (all of them, without exception)
src	Custom C/C++ functions
tests	Unit tests
packages.R	List of R packages for the project
reload.R	Automated loading of functions and packages in a separate environment
scripts	
python	Scripts for file management and conversion
Report.Rmd	
DataAcquisition.R	
DataMunging.R	
Figures.R	
Modeling.R	

File naming

- Use descriptive file names
- Be explicit
  - File1.R, File4.R won't help you
  - DataMunging.R, RegressionModels.R will
- Well-chosen names saves a lot of time and heartache

Documentation

- Create at least a README file to describe what the project is about.
- I've started creating a "lab notebook" for data analyses
    - Usually named `Notebook.Rmd`
    - Either a straight R Markdown file or a R Notebook
    - Keep notes on
        - What products (data sets, tables, figures) I've created
        - What new scripts I've written
        - What new functions I've written
        - Notes from discussions with colleagues on decisions regarding data, analyses, final products

Documentation

- Document your code as much as you can
    - Copious comments to state what you're doing and why
- If you write functions
    - Use Roxygen to document the inputs, outputs, what the function does and an example

Function sanity

The computer follows direction really well

- Use scripts/functions to derive quantities you need for other functions
- Don't hard-code numbers

```
runif(n = nrow(dat), min = min(dat$age), max = max(dat$age))
```

rather than

```
runif(n = 135, min = 18, max = 80)
```

- This reduces potential errors in data transcription
  - These are really hard to catch

Create functions rather than copy-paste code

- If you're doing the same thing more than twice, write a function (*DRY principle*)
- Put the function in its own file, stored in a particular place
    - I store them in `lib/R`.
    - Don't hide them in general script files where other stuff is happening
    - Name the file so you know what's in it
    - One function or a few related functions per file
- Write the basic documentation **NOW**!

Loading your functions

```r
funcfiles <- dir('lib/R', pattern = '.R')
for(f in funcfiles){
  source(f)
}
```

Package sanity

Suppose you need to load a bunch of packages and aren't sure whether they are installed on your system or not. You can certainly look in `installed.packages`, but if you have 1000s of packages, this can be slow.

You can use `require`:

```
x <- require(ggiraph)
x
```

```
[1] TRUE
```

A more elegant solution is using the `pacman` package

```
if (!require("pacman")) install.packages("pacman") # make sure pacman is installed
pacman::p_load(ggiraph, stargazer, kableExtra)
```

This will install the package if it's not installed, and then load it up.

Manipulate data with care

- Keep a pristine copy of the data
- Use scripts to manipulate data for reproducibility
    - Can catch analyst mistakes and fix
- Systematically verify and clean
    - Create your own Standard Operating Plan
- Document what you find
    - Lab notebook (example)

Manipulate data with care

- The laws of unintended consequences are vicious and unforgiving, and appear all too frequenty at the data munging stage
- For example, data types can change (factor to integer)
- Test your data at each stage to make sure you still have what you think you have

Track data provenance through the pipeline

- Typically:

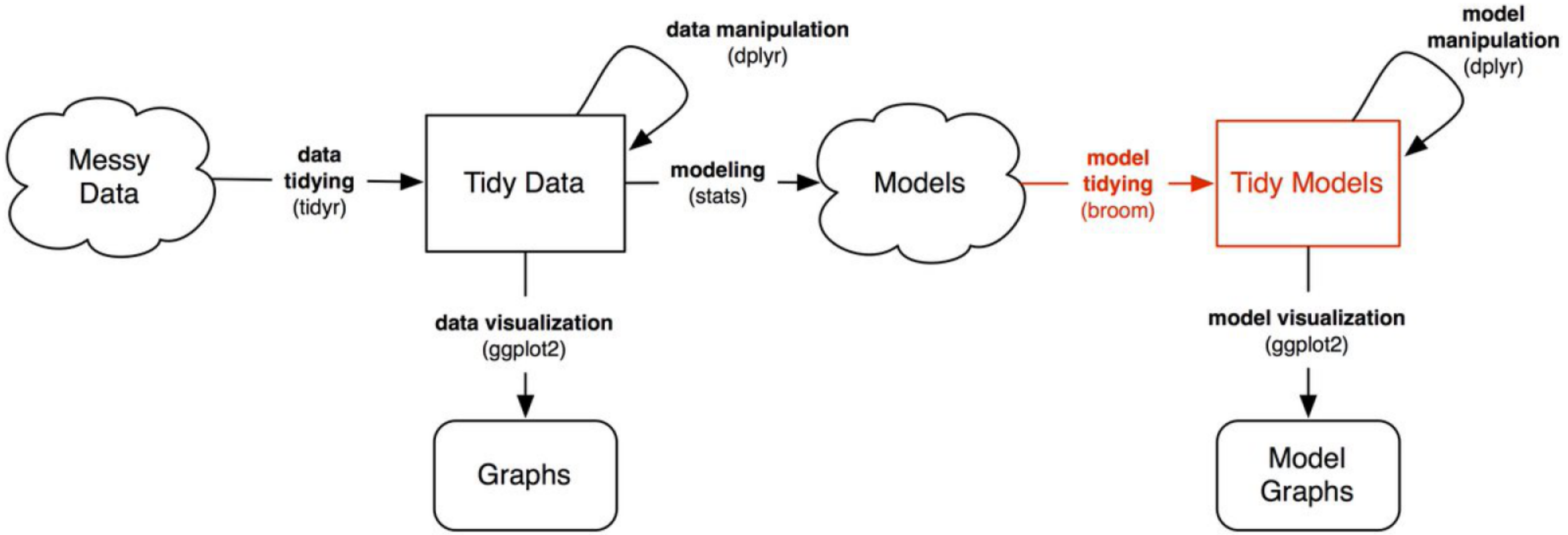
> Raw data >> Intermediate data >> Final data >> data for sub-analyses >> data for final tables and figures

- Catalog and track where you create data, and where you ingest it
- Make sure there are no loops!!

Share preliminary analysis for a sniff

- Share initial explorations with colleagues so they pass a "sniff" test
    - Are data types what you expect
    - Are data ranges what you expect
    - Are distributions what you expect
    - Are relationships what you expect
- This stuff is important and requires deliberate brain power
- May require feedback loop and more thinking about the problem

A general pipeline



David Robinson, 2016

Know where final tables and figures come from

- I create separate files for creating figures and tables for a paper
  - They're called `FinalTables.R` and `FinalFigures.R`. Duh!
- This provides final check that right data are used, and can be updated easily during revision cycle
- It's a long road to this point, so make sure things are good.

RMarkdown

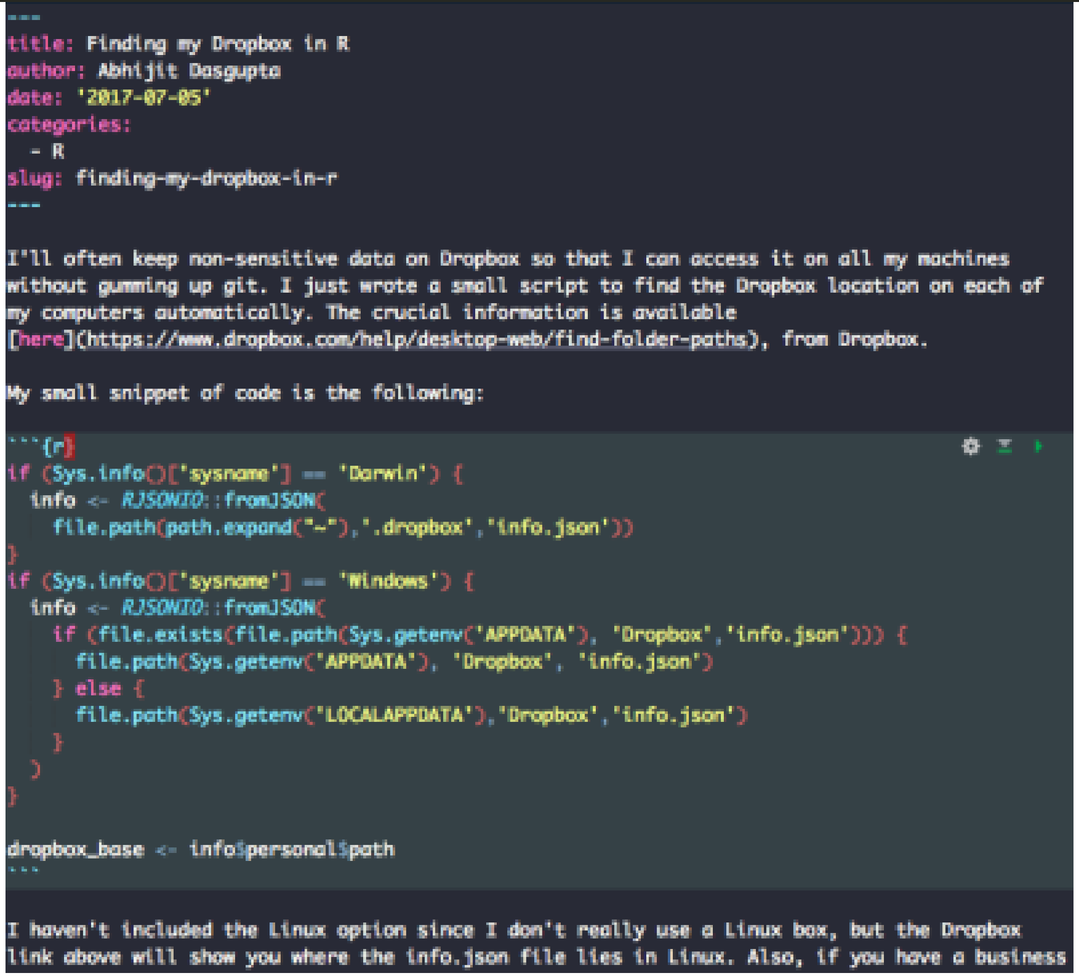
RMarkdown

Many of you are already using RMarkdown in your R Notebooks.

RMarkdown documents are text with code chunks. Great for reporting, not so great for development

Ideally when you develop, you want an annotated R script (text as comments), and then transform it to a RMarkdown document for a nicely formatted document

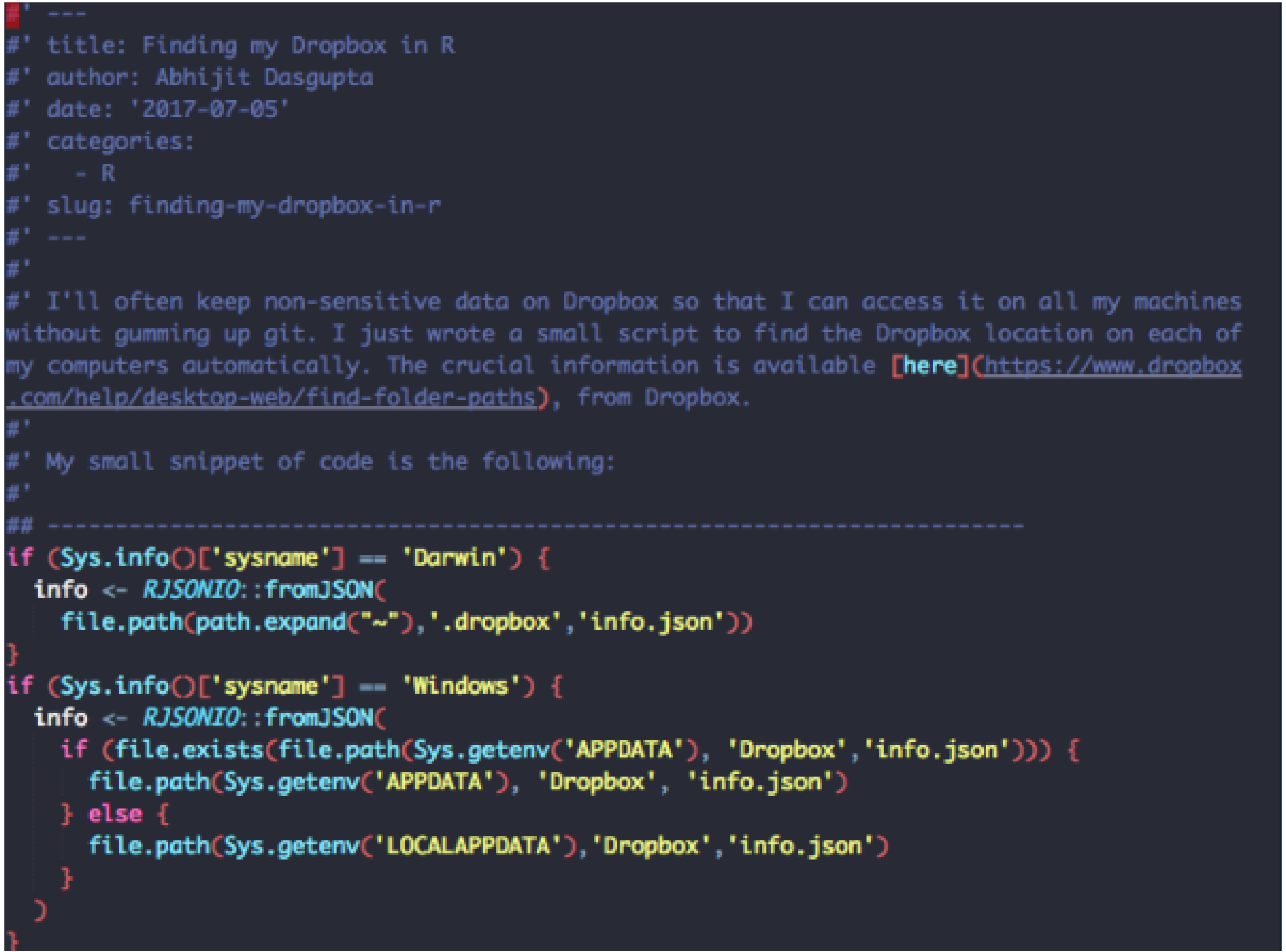
Take any RMarkdown document, and pass it through the function `knitr::purl`, and bring it back with `knitr::spin`



~~~
---
title: Finding my Dropbox in R
author: Abhijit Dasgupta
date: '2017-07-05'
categories:
  - R
slug: finding-my-dropbox-in-r
---

I'll often keep non-sensitive data on Dropbox so that I can access it on all my machines
without gumming up git. I just wrote a small script to find the Dropbox location on each of
my computers automatically. The crucial information is available
[here](https://www.dropbox.com/help/desktop-web/find-folder-paths), from Dropbox.

My small snippet of code is the following:

```{r}
if (Sys.info()['sysname'] == 'Darwin') {
  info <- RJSONIO::fromJSON(
    file.path(path.expand("~"),'.dropbox','info.json'))
}
if (Sys.info()['sysname'] == 'Windows') {
  info <- RJSONIO::fromJSON(
    if (file.exists(file.path(Sys.getenv('APPDATA'), 'Dropbox','info.json'))) {
      file.path(Sys.getenv('APPDATA'), 'Dropbox', 'info.json')
    } else {
      file.path(Sys.getenv('LOCALAPPDATA'),'Dropbox','info.json')
    }
  )
}

dropbox_base <- info$personal$path
```

I haven't included the Linux option since I don't really use a Linux box, but the Dropbox
link above will show you where the info.json file lies in Linux. Also, if you have a business
~~~

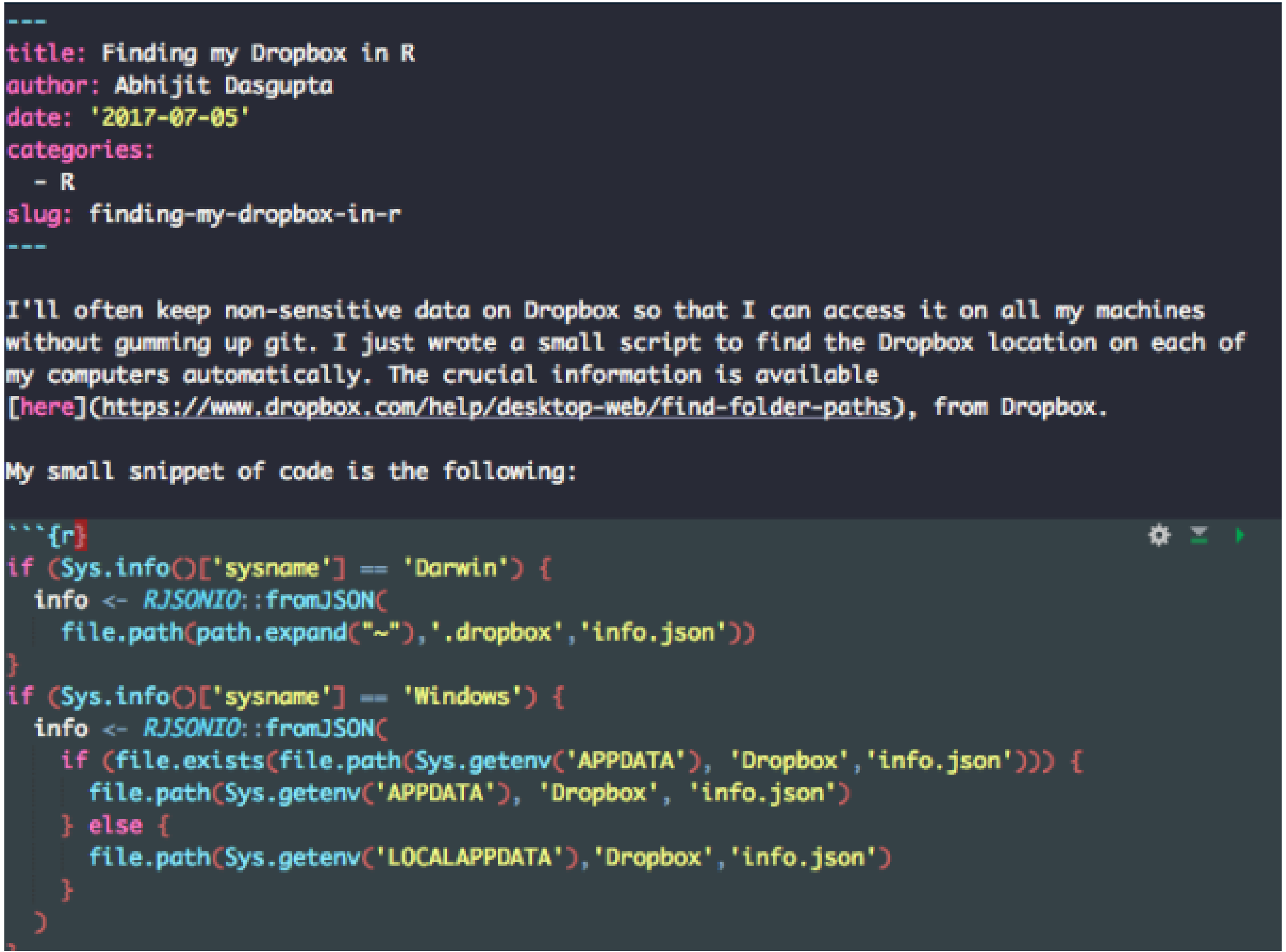
https://webbedfeet.netlify.com/post/interchanging-rmarkdown-and-spinnable-r/

```
knitr::purl('finding-my-dropbox.Rmd', documentation=2)
```



```
#' ---
#' title: Finding my Dropbox in R
#' author: Abhijit Dasgupta
#' date: '2017-07-05'
#' categories:
#'   - R
#' slug: finding-my-dropbox-in-r
#' ---
#'
#' I'll often keep non-sensitive data on Dropbox so that I can access it on all my machines
without gumming up git. I just wrote a small script to find the Dropbox location on each of
my computers automatically. The crucial information is available [here](https://www.dropbox
.com/help/desktop-web/find-folder-paths), from Dropbox.
#'
#' My small snippet of code is the following:
#'
## ----------------------------------------------------------------------
if (Sys.info()['sysname'] == 'Darwin') {
  info <- RJSONIO::fromJSON(
    file.path(path.expand("~"),'.dropbox','info.json'))
}
if (Sys.info()['sysname'] == 'Windows') {
  info <- RJSONIO::fromJSON(
    if (file.exists(file.path(Sys.getenv('APPDATA'), 'Dropbox','info.json'))) {
      file.path(Sys.getenv('APPDATA'), 'Dropbox', 'info.json')
    } else {
      file.path(Sys.getenv('LOCALAPPDATA'),'Dropbox','info.json')
    }
  )
}
```

```
knitr::spin('finding-my-dropbox.R', knit = F, format='Rmd')
```



~~~
---
title: Finding my Dropbox in R
author: Abhijit Dasgupta
date: '2017-07-05'
categories:
  - R
slug: finding-my-dropbox-in-r
---

I'll often keep non-sensitive data on Dropbox so that I can access it on all my machines
without gumming up git. I just wrote a small script to find the Dropbox location on each of
my computers automatically. The crucial information is available
[here](https://www.dropbox.com/help/desktop-web/find-folder-paths), from Dropbox.

My small snippet of code is the following:

```{r}
if (Sys.info()['sysname'] == 'Darwin') {
  info <- RJSONIO::fromJSON(
    file.path(path.expand("~"),'.dropbox','info.json'))
}
if (Sys.info()['sysname'] == 'Windows') {
  info <- RJSONIO::fromJSON(
    if (file.exists(file.path(Sys.getenv('APPDATA'), 'Dropbox','info.json'))) {
      file.path(Sys.getenv('APPDATA'), 'Dropbox', 'info.json')
    } else {
      file.path(Sys.getenv('LOCALAPPDATA'),'Dropbox','info.json')
    }
  )
~~~

Rich RMarkdown Documents

What can you create from RMarkdown?

- Documents
    - HTML
    - Microsoft Word
    - PDF (requires LaTeX)
- Presentations
    - HTML (`ioslides`, `revealjs`, `xaringan`)
    - PDF (`beamer`)
    - PowerPoint

What can you create from RMarkdown?

- Interactive documents
  - The `htmlwidgets` meta-package
- Dashboards
  - The `flexdashboard` package
- Books
  - The `bookdown` package
- Websites & Blogs
  - RMarkdown
  - `blogdown` package

What can you create with RMarkdown?

- Resumes/CVs
    - The `vitae` package
- Research papers
    - include citations
    - include appropriate formatting
    - probably need LaTeX

See the RMarkdown gallery

What can you create with RMarkdown?

The basic differences are in the front-matter at the top of your RMarkdown document

HTML document

```
---
title: "Lectures"
date: "Fall 2018"
output: html_document
---
```

Word document

```
---
title: "Lectures"
date: "Fall 2018"
output: word_document
---
```

Presentations

ioslides

```
---
title: "Lecture 2: \nData Frame, Matrix, List"
author: "Abhijit Dasgupta"
date: "September 19, 2018"
output: ioslides_presentation
---
```

revealjs

```
---
title: "Lecture 2: \nData Frame, Matrix, List"
author: "Abhijit Dasgupta"
date: "September 19, 2018"
output:
  revealjs::revealjs_presentation:
    theme: default
    highlight: default
    transition: fade
    slide_level: 1
```

Slides delimited by markdown sections

```
# Slide 1

This is my first slide

# Slide 2

This is my second slide
```

Presentations

Powerpoint

```
---
title: "Lecture 2: \nData Frame, Matrix, List"
author: "Abhijit Dasgupta"
date: "September 19, 2018"
output: powerpoint_presentation
---
```

Slides delimited by markdown sections

```
# Slide 1

This is my first slide

# Slide 2

This is my second slide
```

Presentations

xaringan

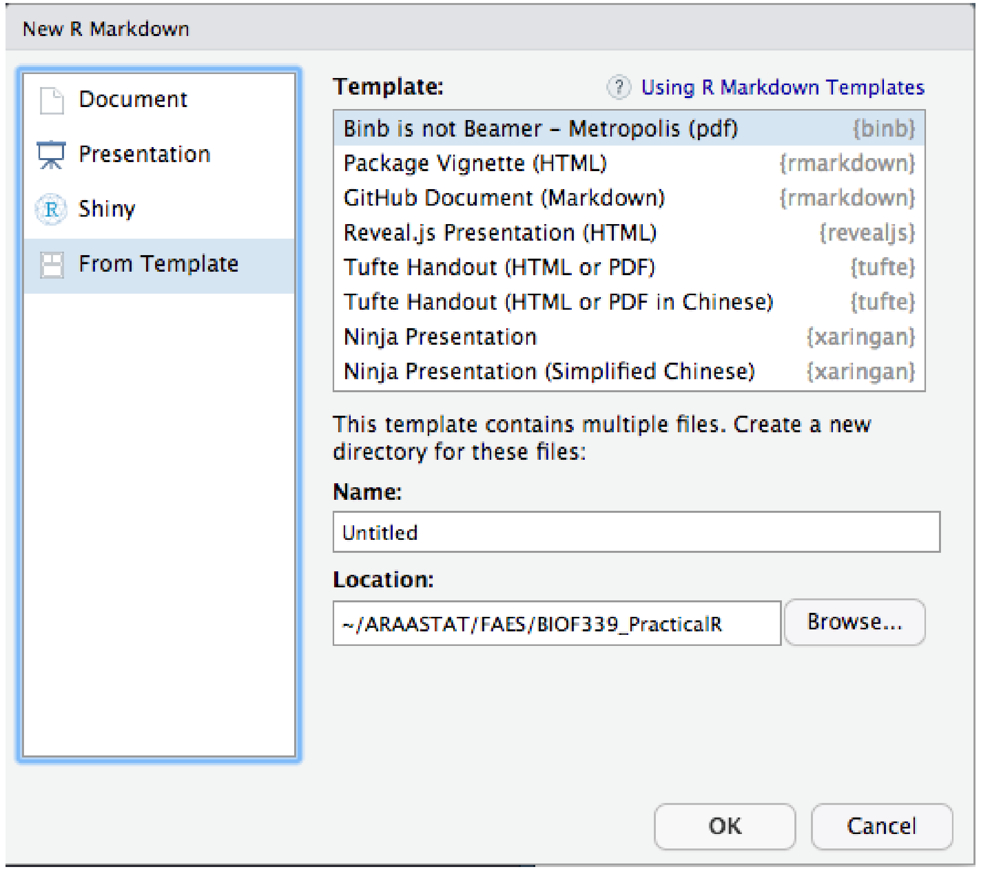
```
---
title: "Lecture 2: \nData Frame, Matrix, List"
author: "Abhijit Dasgupta"
date: "September 19, 2018"
output:
  xaringan::moon_reader:
    css: [default, './robot.css', './robot-fonts.css'
    #css: [default, metropolis, metropolis-fonts]
    nature:
      ratio: '16:9'
      highlightLanguage: R
      countIncrementalSlides: false
      highlightStyle: zenburn
      highlightLines: true
---
```

Slides delimited by ---

```
---

# Slide 1

This is my first slide

---

# Slide 2

This is my second slide
```

RMarkdown Templates

- Several packages provide RMarkdown templates
- You can include citations
  - EndNote, MEDLINE, RIS, BibTeX formats for references
  - See https://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html



Template:

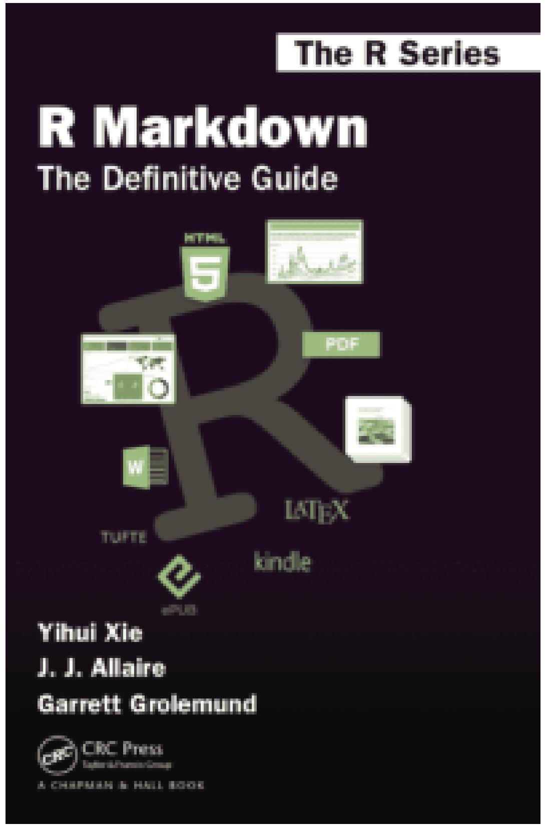
Binb is not Beamer – Metropolis (pdf)	{binb}
Package Vignette (HTML)	{rmarkdown}
GitHub Document (Markdown)	{rmarkdown}
Reveal.js Presentation (HTML)	{revealjs}
Tufte Handout (HTML or PDF)	{tufte}
Tufte Handout (HTML or PDF in Chinese)	{tufte}
Ninja Presentation	{xaringan}
Ninja Presentation (Simplified Chinese)	{xaringan}

This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/ARAASTAT/FAES/BIOF339_PracticalR

Resources



The R Series

R Markdown

The Definitive Guide

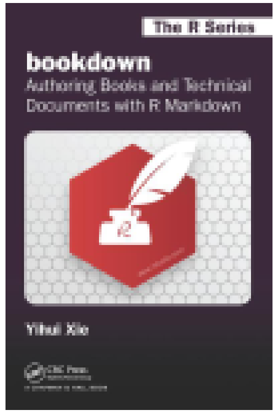
Yihui Xie

J. J. Allaire

Garrett Grolemund

CRC Press

A CHAPMAN & HALL BOOK



The R Series

bookdown

Authoring Books and Technical Documents with R Markdown

Yihui Xie

CRC Press

The R Series

blogdown

Creating Websites with R Markdown

Yihui Xie

Amber Thomas

Alison Presmanes Hill

CRC Press