

## 5: EDA, Big Jobs, Automation

[bit.ly/2018rr](https://bit.ly/2018rr)

# Exploratory data analysis

- ▶ what were you trying to do?
- ▶ what were you thinking about?
- ▶ what did you observe?
- ▶ what did you conclude, and why?

# Avoid

- ▶ "How did I create this plot?"
- ▶ "Why did I decide to omit those six samples?"
- ▶ "Where (on the web) did I find these data?"
- ▶ "What was that interesting gene?"

# Basic principles

Step 1: slow down and document.

Step 2: have sympathy for your future self.

Step 3: have a system.

# Capturing EDA

- ▶ copy-and-paste from an R file
- ▶ grab code from the `.Rhistory` file
- ▶ Write an informal R Markdown file; make use of the “R Notebook” features.
- ▶ Write code for use with the KnitR function `spin()`
  - Comments like `#' This will become text`
  - Chunk options like so: `#+ chunk_label, echo=FALSE`

# A file to spin()

```
#' This is a simple example of an R file for use with spin().  
  
#' We'll start by setting the seed for the RNG.  
set.seed(53079239)  
  
#' We'll first simulate some data with  $x \sim N(\mu=10, \sigma=5)$  and  
#'  $y = 2x + e$ , where  $e \sim N(\mu=0, \sigma=2)$   
x <- rnorm(100, 10, 5)  
y <- 2*x + rnorm(100, 0, 2)  
  
#' Here's a scatterplot of the data.  
plot(x, y, pch=21, bg="slateblue", las=1)
```

# Activity

Try out `knitr::spin()`:

- ▶ Create an R script using
  - `#'` for text
  - `#+` for chunk options
- ▶ Compile the script to HTML using `knitr::spin()`
- ▶ Open the resulting HTML file.

# Big jobs

- ▶ You don't want `knitr` running for a year.
- ▶ You don't want to re-run things if you don't have to.



# Biggish jobs in knitr

- ▶ Manual caching
- ▶ Built-in `cache=TRUE`
- ▶ Split the work into a separate subdirectory.

# Manual caching

```
```{r a_code_chunk}
file <- "cache/myfile.RData"

if(file.exists(file)) {
  load(file)
} else{

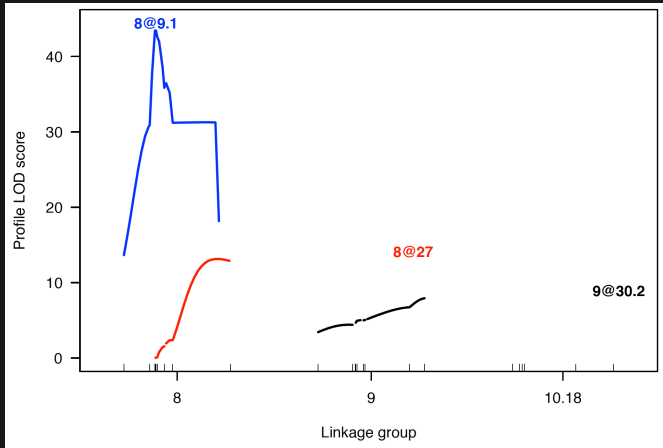
  ....

  save(object1, object2, object3, file=file)
}
```
```

# Chunk references

```
```{r not_shown, eval=FALSE}  
code_here <- 0  
```  
  
```{r a_code_chunk, echo=FALSE}  
file <- "cache/myfile.RData"  
  
if(file.exists(file)) {  
  load(file)  
} else{  
<<not_shown>>  
  save(code_here, file=file)  
}  
```
```

# A cache gone bad



# Knitr's cache system

```
```{r chunk_name, cache=TRUE}  
load("a_big_file.RData")  
med <- apply(object, 2, median, na.rm=TRUE)  
```
```

- ▶ Chunk is re-run if edited.
- ▶ Otherwise, objects from previous run are loaded.
- ▶ Don't cache things with side effects  
e.g., `options()`, `par()`

# Cache dependencies

## Manual dependencies

```
```{r chunkA, cache=TRUE}  
Sys.sleep(2)  
x <- 5  
```\n\n```{r chunkB, cache=TRUE, dependson="chunkA"}  
Sys.sleep(2)  
y <- x + 1  
```\n\n```{r chunkC, cache=TRUE, dependson="chunkB"}  
Sys.sleep(2)  
z <- y + 1  
```
```

# Cache dependencies

## Automatic dependencies

```
```{r setup, include=FALSE}  
opts_chunk$set(autodep = TRUE)  
dep_auto()  
```
```

# Parallel computing

If your computer has multiple processors, use `library(parallel)` to make use of them.

- ▶ `detectCores()`
- ▶ `RNGkind("L'Ecuyer-CMRG")` and `mclapply` (Unix/Mac)
- ▶ `makeCluster`, `clusterSetRNGStream`, `clusterApply`, and `stopCluster` (Windows)



# Systems for distributed computing

- ▶ At UW-Madison, we use HTCondor
- ▶ There are **odds of similar systems**
- ▶ "By hand"

# Simulations

- ▶ Computer simulations require RNG seeds (`.Random.seed` in R).
- ▶ Multiple parallel jobs need different seeds.
- ▶ Don't rely on the current seed, or on having it generated from the clock.
- ▶ Use something like `set.seed(91820205 + i)`
- ▶ An alternative is create a big batch of simulated data sets in advance.

# Save everything

- ▶ RNG seeds
- ▶ input
- ▶ output
- ▶ version numbers, with `sessionInfo()`
- ▶ raw results
- ▶ script to combine results
- ▶ combined results
- ▶ ReadMe describing the point

# Compartmentalize big jobs

- ▶ Separate directory for each batch of big computations.
- ▶ Collect the results in `‘.RData’` or `‘.rds’` files.
- ▶ KnitR-based documents for the analysis/use of those results.

# Potential problems

- ▶ Forgetting `save()` in your distributed jobs
- ▶ A bug in the `save()` command
- ▶ Keeping things synchronized
  - Have you re-run the big jobs when upstream data were revised?

# Big jobs summary

- ▶ Careful organization and modularization.
- ▶ Save everything.
- ▶ Document everything.
- ▶ Learn the basic skills for distributed computing.

# Activity

Try out the knitr cache system.

- ▶ Create an RMarkdown document with multiple dependent chunks.
- ▶ Maybe add `Sys.sleep(5)` to slow things down, as if the jobs were taking a while.
- ▶ Compile the document.
- ▶ Edit a chunk.
- ▶ Re-compile the document and see what was actually run and what was taken from cache.