Naming & Organization

Reproducible Computing

@ JSM 2019

Colin Rundel

July 27, 2019

Face it

- There are going to be files
- LOTS of files
- The files will change over time
- The files will have relationships to each other
- It'll will get complicated

Mighty weapon

- File organization and naming is a mighty weapon against chaos
- Make a file's name and location VERY INFORMATIVE about what it is, why it exists, how it relates to other things
- The more things are self-explanatory, the better
- READMEs are great, but don't document something if you could just make that thing self-documenting by definition

What works, what doesn't?

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
```

Three principles for (file) names

1. Human readable
2. Machine parsable
3. Plays well with OS ordering

Machine Parsable

Machine readable

- Search (Regular expression and globbing) friendly: Avoid spaces, punctuation, accented characters, case sensitivity
- Easy to compute on: Deliberate use of delimiters

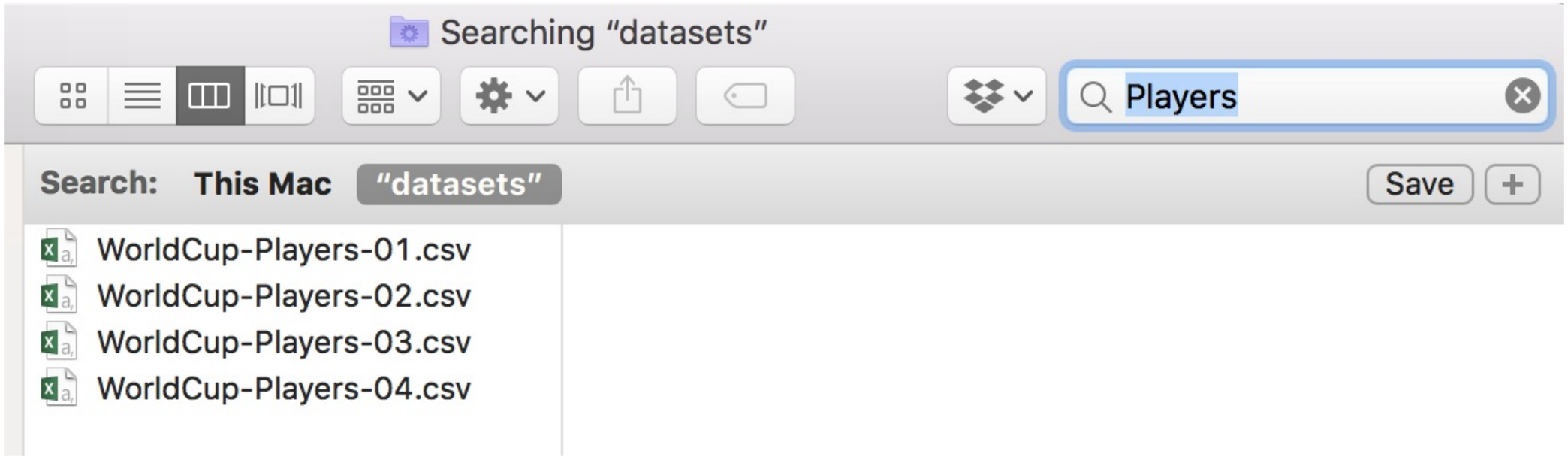
Globbing

Excerpt of complete file listing:

Example of globbing to narrow file listing:

```
datasets rundel$ ls *Players*
WorldCup-Players-01.csv    WorldCup-Players-03.csv
WorldCup-Players-02.csv    WorldCup-Players-04.csv
```

Same using Mac OS Finder search facilities



Same using regex in R

```
> library(fs)
> dir_ls(glob = "*Players*")
WorldCup-Players-01.csv WorldCup-Players-02.csv WorldCup-Players-03.csv
WorldCup-Players-04.csv
```

Punctuation

Deliberate use of "−" and "_" allows recovery of meta-data from the filenames:

- Use one to delimit units of meta-data you might want later
- Use the other to delimit words
- Stay consistent

For example:

```
2019-09-01_Experiment-1_Rep-A.csv
2019-09-01_Experiment-1_Rep-B.csv
2019-09-07_Experiment-1_Rep-C.csv
2019-09-07_Experiment-2_Rep-A.csv
```

Recap: Machine readable

- Easy to search for files later
- Easy to narrow file lists based on names
- Easy to extract info from file names, e.g. by splitting
- New to regular expressions and globbing? be kind to yourself and avoid
  - Spaces in file names
  - Punctuation
  - Accented characters (Unicode in general)
  - Different files named `foo` and `Foo`

Human readable

Human readable

- Name contains info on content
- Connects to concept of a slug from semantic URLs

Example

Which set of file(name)s do you want at 3 a.m. before a deadline?

01_marshal-data.md	01.md
01_marshal-data.r	01.r
02_pre-dea-filtering.md	02.md
02_pre-dea-filtering.r	02.r
03_dea-with-limma-voom.md	03.md
03_dea-with-limma-voom.r	03.r
04_explore-dea-results.md	04.md
04_explore-dea-results.r	04.r
90_limma-model-term-name-fiasco.md	90.md
90_limma-model-term-name-fiasco.r	90.r
Makefile	Makefile
figure	figure
helper01_load-counts.r	helper01.r
helper02_load-exp-des.r	helper02.r
helper03_load-focus-statinf.r	helper03.r
helper04_extract-and-tidy.r	helper04.r
tmp.txt	tmp.txt

Recap: Human readable

Easy to figure out what it is, based on its name

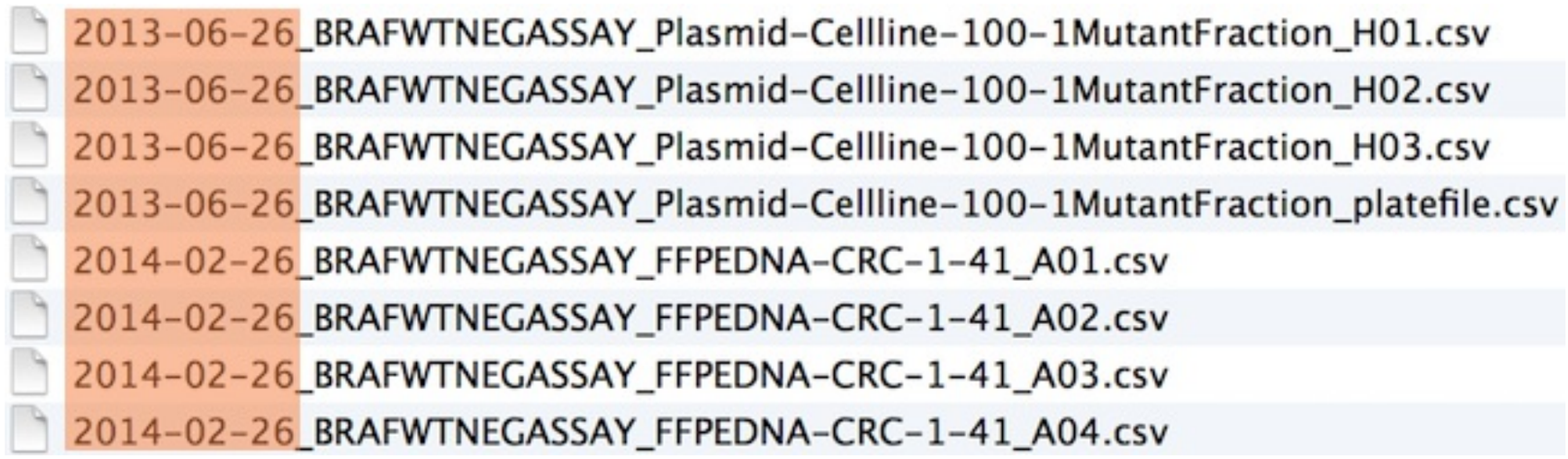
Plays well with OS ordering

Plays well with default ordering

- Put something numeric first
- Use the ISO 8601 standard for dates
- Left pad other numbers with zeros

Examples

Chronological order:



Logical order: Put something numeric first

```
01_marshal-data.r
02_pre-dea-filtering.r
03_dea-with-limma-voom.r
04_explore-dea-results.r
90_limma-model-term-name-fiasco.r
helper01_load-counts.r
helper02_load-exp-des.r
helper03_load-focus-statinf.r
helper04_extract-and-tidy.r
```

Dates

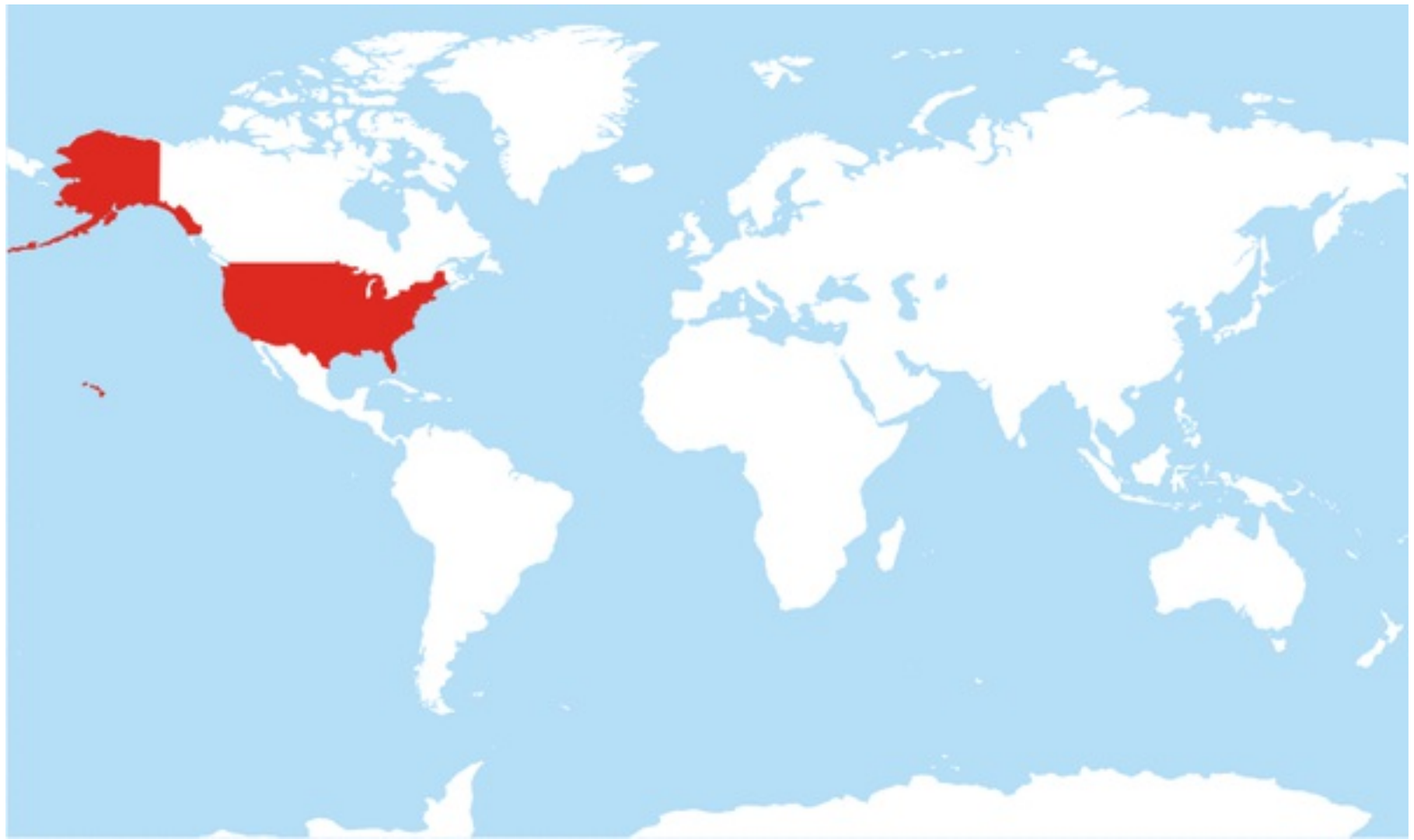
Use the ISO 8601 standard for dates: YYYY-MM-DD

2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H01.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H02.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_H03.csv
2013-06-26_BRAFWTNEGASSAY_Plasmid-Cellline-100-1MutantFraction_platefile.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A01.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A02.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A03.csv
2014-02-26_BRAFWTNEGASSAY_FFPEDNA-CRC-1-41_A04.csv

ISO8601

Source: XKCD #1179

Comprehensive map of all countries that use the MM-DD-YYYY format



Source: https://twitter.com/donohoe/status/597876118688026624.

Left pad other numbers with zeros

```
> ls
01_data-cleaning.R
02_fit-model.R
10_final-figs-for-publication.R
```

If you don't left pad, you get this:

```
> ls
10_final-figs-for-publication.R
1_data-cleaning.R
2_fit-model.R
```

Organizing your workflow

There is no one formula

that will work for all projects, but use an organization that will allow

- you to come back to the project a year later and resume work fairly quickly
- your collaborators to figure out what you did and decided and what files they need to look at
- works with your tools not against them

Tip 1 - Use Projects / Project Folders

Specifically within RStudio, but also more generally as an organizing principal.

- Use one (master) folder per project
    - Everything related to that project needs to live within that folder. (e.g. data, scripts, etc.)
    - IDEs, git, R sessions are all designed around this principal (don't fight it)
    - Organize related files within your folder (e.g. `data/`, `scripts/`, `figures/`)

Aside - Raw data is sacrosanct

Raw data is foundational to reproducibility and it is critical to have a auditable log of any changes at all times.

Create a folder for it, put it there and never touch it.

BAD:

```
project/
  - data/
```

vs.

GOOD

```
project/
  - data-raw/
  - data-clean/
```

```
project/
  - data/
    - raw/
    - clean/
```

Relative vs absolute paths

If the first line of your R script is

```
setwd("C:\Users\jenny\path\that\only\I\have")
```

I* will come into your office and SET YOUR COMPUTER ON FIRE 🔥.

* or maybe Timothée Poisot will

Source: Jenny Bryan's Zen and the Art of Workflow Maintenance

Working directories

Keeping track of working directories can be painful.

Take a project that looks something like the following:

```
project/
  - project.Rproj
  - data/
    - raw/
      - data.csv
    - clean/
  - script/
    - 01_clean.R
```

When I run `01_clean.R` what is its working directory?

Using `here`

`here` is a package that tries to simplify the process by identifying the root of your project, `project/` in this case and then providing relative paths from that root directory to everything else in your project.

```
here::here()
## [1] "/home/rundel/Desktop/project"

here::here("data/raw", "data.csv")
## [1] "/home/rundel/Desktop/project/data/raw/data.csv"
```

Tips

- Use a `from_joe` directory: Suppose your collaborator and data producer is Joe. Just leave communucation and files from Joe in this directory and copy or symlink as needed and record this in the project README.
- Give yourself less rope
- Avoid monoliths (modularize your code / scripts into logical steps)
- Keep the life cycle of data in mind