getwd: Where Am I?

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2018

KU

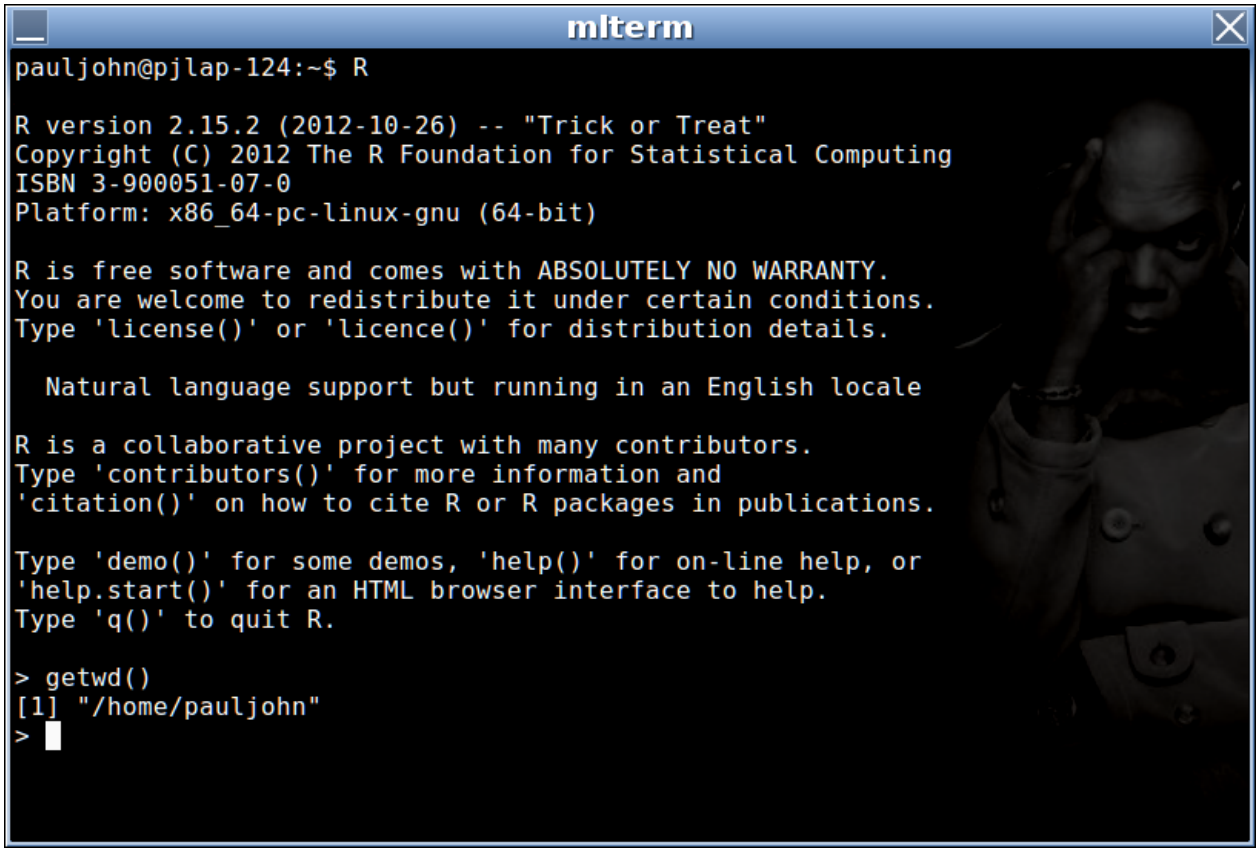
Outline

Introduction

- This part is about your operating system and how to understand the way R (R Core Team, 2017) interacts with it.
- Vital Terms:
  1. path
  2. current working directory
- Paths may be either relative or absolute
  - absolute path begins with highest OS level, such as "C:/" in Windows
  - relative begins with current working directory as starting point, move "down" or "up" from there.
- Key R functions that are emphasized
  - `getwd()`
  - `setwd()`

When R starts, Where are you?

- When you start R, "where are you"?
- This example: I started R in my Linux home directory, `/home/pauljohn`
- getwd() $\Longleftrightarrow$ get Working Directory (WD)!



Where Am I

- Sorry: this is one of the worst problems of modern GUI. Users are encouraged to give up control of their files.
- Launch R from the
  - Windows R icon, or
  - in the Mac Finder
- Ask R what it thinks is your current working directory:

```
getwd()
```

```
[1]
    "/home/pauljohn/GIT/pj-guides/Rcourse/summeR_workshop/summeR-1/summeR-
```

- **Working Directory**: default directory where files are found or written
- If R thinks you are working with files in one place, but you may wish it were working with files in another place, then necessary to reset with `setwd()`

```
setwd("...some...folder...you...specify...correctly"
```

- Some GUIs will have a pull down menu to change the working directory.

You Choose: Relative or Absolute Paths

- The output from getwd() is an absolute PATH.
- The FULL PATH would be (long, boring, typo-prone) like
  Mac: "/user/your-name/Documents/psych790/exercise-1/R"
  Windows:
  "C:/users/your-name-here/Documents/psych790/exercise-1/R"
- We can use RELATIVE PATHs to make things a little easier
  Relative path: If you don't specify an absolute, full path, R uses the relative directory address to look for files and folders.

KU

Why I prefer Relative Paths

- If a student's R program begins with a path-setter, like

```
setwd("C:/users/pauljohn/Documents/schoolwork/stats/exe
```

I cannot run that code "as is". However, if the student sends me the "exercise1" project folder, and uses relative paths inside the R code, then I can run it without editing.

- YES, I mean forward slashes, even in Windows.

Windows Slashes

- R (and programming in general) uses back slash is an "escape symbol",
  - "\n" a New Line in a text string
  - "\t" a TAB inserted
- To avoid confusion, PATHS in Windows can be written with forward slashes.
- Repeat, even in Windows,
  - **we do not type "backslashes".**
- If you run "getwd()", and R says this:
  "C:\user\your-name-here\Documents"

  Use all of your *mental super powers* to see that as
  "C:/user/your-name-here/Documents"

  Because if you want to specify a path in windows, you need to use the forward slashes.

My Proposal

- Keep a project folder
- Subdirectories “R”, “data”, “output”, “workingdata”
- Open an R script folder in the R folder.
  - Imagine a happy future in which R automatically knows your working directory!
  - Use a file manager to launch Emacs, Notepad++, or RStudio on your R file and you will be in that happy place!
- When you need to share your “project” to somebody else, Zip together the whole folder, including the directory name, and send it. They will be able to run without changing anything.

Relative file paths

Suppose your directories are like so

```
school
  pols706
    project1
      R
      data
      output
```

- Write an R file in the R directory.
- Start R with that working directory, then...
- Use relative paths like "../data" or "../output" in R programs.

Oh my gosh. Two dots and a slash?

`../` means go up one level

- Read data from `"../data/whatever.xlsx"`
- Write graphs to output, `"../output/whatever.pdf"`.

initProject() in kutils

- Open R and set the working directory in a project folder. Run setwd if you need to
- I choose a directory named "/tmp/proj" for demonstration

```
library(kutils)
initProject()
```

```
> initProject()
Creating: data
Creating: workingdata
Creating: output
Creating: tmp
Creating: lit
Creating: writeup
Creating: R
Initialized empty shared Git repository in /tmp/proj/.git/
[master (root-commit) 7ef920d] Initialized project in /tmp/proj
 2 files changed, 5 insertions(+)  create mode 100644 ChangeLog
 create mode 100644 README.md
Please consider creating a remote repository to which this repo
    should be linked
[1] "/tmp/proj"
```

initProject() in kutils ...

- After that, the project should be laid out with several folders.
- Close, R, and use the file manager to navigate into the project's R folder
- Open "template.R" with your editor

Recall PDF example from a previous lecture?

Put these lines into your R script

```
x <- rgamma(100, 1.5, 2.2)
y <- rnorm(100) + 0.2 * x
pdf("output/scatter1.pdf", height=6, width=6,
    paper="special")
plot(y ~ x, main="Scatterplot or Bust!")
dev.off() #turns off pdf output device
```

That creates a pdf and writes it in the output.

Where is the big payoff here?

- Easy re-production of graphs when the data changes slightly
- Ability to mass-produce graphs. I've produced thousands of graph files, which I could later snoop through to find something interesting.

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

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] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
[1] compiler_3.6.0 tools_3.6.0
```