PP4RS | R Module

Slot 3

Dora Simon

06.09.2018

Outline of the R-Module

Slot 1: Intro & Data Types

Slot 2: Conditionals and Functions & Loops

Slot 3: Read in Data

Slot 4: Data Manipulation

Slot 5: Regressions

Slot 6: Graphs

Slot 7: knitR

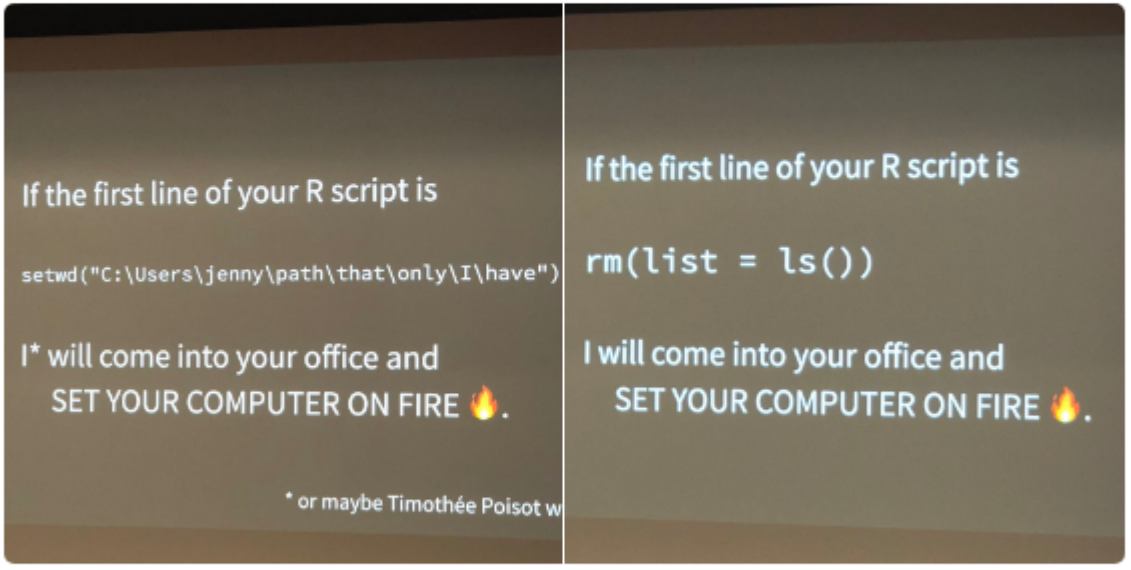
Now: Read in Data

Workflow in R

Hadley Wickham
@hadleywickham

Follow

The only two things that make @JennyBryan 😤😠🤯. Instead use projects + here::here() #rstats



4:50 pm - 10 Dec 2017

295 Retweets **970** Likes

63 295 970

Why no setwd()?[1]

The chance this `setwd()` will work for anyone besides the author is 0.

```
setwd("~/git/teaching/pp4rs/2018-uzh-course-material")
#read in, regress, plot and save data
```

Why no setwd()?[1]

The chance this `setwd()` will work for anyone besides the author is 0.

```
setwd("~/git/teaching/pp4rs/2018-uzh-course-material")
#read in, regress, plot and save data
```

What to do instead?

- Make sure the top-level folder advertises itself as such
  - Empty file named .here
  - Using Git or RProjects also work

Why no setwd()?[1]

The chance this `setwd()` will work for anyone besides the author is 0.

```
setwd("~/git/teaching/pp4rs/2018-uzh-course-material")
#read in, regress, plot and save data
```

What to do instead?

- Make sure the top-level folder advertises itself as such
  - Empty file named .here
  - Using Git or RProjects also work
- Use the `here()` function from the `here` package to build the path when you read or write a file.
  - Create paths relative to the top-level directory.

Why no setwd()?[1]

The chance this `setwd()` will work for anyone besides the author is 0.

```
setwd("~/git/teaching/pp4rs/2018-uzh-course-material")
#read in, regress, plot and save data
```

What to do instead?

- Make sure the top-level folder advertises itself as such
  - Empty file named .here
  - Using Git or RProjects also work
- Use the `here()` function from the `here` package to build the path when you read or write a file.
  - Create paths relative to the top-level directory.
- Launch the R process from the project's top-level directory.
  - If you launch R from the shell, cd to the correct folder first.

Why no setwd()?[1]

The chance this `setwd()` will work for anyone besides the author is 0.

```
setwd("~/git/teaching/pp4rs/2018-uzh-course-material")
#read in, regress, plot and save data
```

What to do instead?

- Make sure the top-level folder advertises itself as such
  - Empty file named .here
  - Using Git or RProjects also work
- Use the `here()` function from the `here` package to build the path when you read or write a file.
  - Create paths relative to the top-level directory.
- Launch the R process from the project's top-level directory.
  - If you launch R from the shell, cd to the correct folder first.

[1] All credit goes here

The here function

The `here` function creates paths relative to the top-level directory.

- In our case, the top level directory is the folder with your course materials - You should open R from that folder
- Or set your working directory to that folder in your shell

The here function

The `here` function creates paths relative to the top-level directory.

- In our case, the top level directory is the folder with your course materials - You should open R from that folder
- Or set your working directory to that folder in your shell

How you usually use a function that requires a path

```
somefct("~/git/teaching/pp4rs/2018-uzh-course-material/
        14-r-data-manipulation/some-data.csv")

# Or if you have specified your path above:

somefct("./14-r-data-manipulation/some-data.csv")
```

The here function

The `here` function creates paths relative to the top-level directory.

- In our case, the top level directory is the folder with your course materials - You should open R from that folder
- Or set your working directory to that folder in your shell

How you usually use a function that requires a path

```
somefct("~/git/teaching/pp4rs/2018-uzh-course-material/
        14-r-data-manipulation/some-data.csv")

# Or if you have specified your path above:

somefct("./14-r-data-manipulation/some-data.csv")
```

How the `here` function helps

```
library(here)

somefct(some-argument, here("14-r-data-manipulation"))
```

RProjects

> RStudio projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents.

Projects provide an alternative to the usage of the `here` function.

Look at this short video to understand what Projects are in R:

Video

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Any packages that have been loaded are still available. Any options that have been set to non-default values remain that way. Working directory is not affected.

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Any packages that have been loaded are still available. Any options that have been set to non-default values remain that way. Working directory is not affected.

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Any packages that have been loaded are still available. Any options that have been set to non-default values remain that way. Working directory is not affected.

- Setup: Do not save .RData when you quit R
  - This can be embedded into your preferences in R Studio or in Terminal

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Any packages that have been loaded are still available. Any options that have been set to non-default values remain that way. Working directory is not affected.

- Setup: Do not save .RData when you quit R
  - This can be embedded into your preferences in R Studio or in Terminal
- Daily work habit: Restart R very often and re-run your under-development script from the top.

What about objects that take a long time to create?

Why no rm(list = ls()) ?

> The problem is that `rm(list = ls())` does NOT create a fresh R process. All it does is delete user-created objects from the global workspace.

Any packages that have been loaded are still available. Any options that have been set to non-default values remain that way. Working directory is not affected.

- Setup: Do not save .RData when you quit R
  - This can be embedded into your preferences in R Studio or in Terminal
- Daily work habit: Restart R very often and re-run your under-development script from the top.

What about objects that take a long time to create?

- Save & reload with `saveRDS()`
- Use Snakemake!

Read in Data

How to read in a data file

A csv file is just a text file, where values are separated by a comma.

How to read in a data file

A csv file is just a text file, where values are separated by a comma.

The `readr` package is your friend

- `read_csv()`: comma delimited
- `read_csv2()`: semicolon delimited
- `read_tsv()`: tab delimited
- `read_delim()`: any delimiter

How to read in a data file

A csv file is just a text file, where values are separated by a comma.

The `readr` package is your friend

- `read_csv()`: comma delimited
- `read_csv2()`: semicolon delimited
- `read_tsv()`: tab delimited
- `read_delim()`: any delimiter

Excel and Stata's .dta

- `read_excel` from the `readxl` package
- `read_dta` from the `haven` package

Parsing

Sometimes, columns do not have the correct format.

In that case, you want to parse them correctly. The `readr` package helps there.

- `parse_character()`
- `parse_number()`
- `parse_date(), parse_datetime()`

Parsing

Sometimes, columns do not have the correct format.

In that case, you want to parse them correctly. The `readr` package helps there.

- `parse_character()`
- `parse_number()`
- `parse_date(), parse_datetime()`

R uses 'UTF-8' for character encoding

- Parsing characters might be problematic if your data is in a different character encoding format
- More about character encodings here

Saving data sets

- `saveRDS` and `readRDS` for one file

Saving data sets

- `saveRDS` and `readRDS` for one file
- `save` and `load` for several files[1]

Saving data sets

- `saveRDS` and `readRDS` for one file
- `save` and `load` for several files[1]
- `save.image` for the entire workspace

Saving data sets

- `saveRDS` and `readRDS` for one file
- `save` and `load` for several files[1]
- `save.image` for the entire workspace

Data files in R:

- `rda` is short for `RData` and comes from the `save` family
- `rds` stores a single R object

[1] Read more a bout the difference between `save` and `saveRDS` here and here

Exercises

1. Read in `simons.csv` from the course material folders using the `readr` package and print the table. (hint: Use the `here` function to have the right path.)
2. Define a new variable `age2` that is parsed as a number. Define a new variable `height2` that is parsed as a number as well and gives the height in meters (not centimeters). Then, print your data frame.
3. Write a conditional that checks whether a folder for your clean data exists. If not, it creates a new folder. (hint: use `if`, `file.exists` and `dir.create`)
4. Write your dataset into an .rds file in your new clean data folder.

Solutions

Read in simons.csv from your copy of the course material folder number 15 using the `readr` package and print the table. (hint: Use the `here` function to have the right path.) Do not do it from the real course material folders because it might mess up your git pull.

Solutions

Read in simons.csv from your copy of the course material folder number 15 using the `readr` package and print the table. (hint: Use the `here` function to have the right path.) Do not do it from the real course material folders because it might mess up your git pull.

Solutions

Define a new variable `age2` that is parsed as a number. Define a new variable `height2` that is parsed as a number as well and gives the height in meters (not centimeters). Then, print your data frame.

Solutions

Define a new variable `age2` that is parsed as a number. Define a new variable `height2` that is parsed as a number as well and gives the height in meters (not centimeters). Then, print your data frame.

Solutions

Write a conditional that checks whether a folder for your clean data exists. If not, it creates a new folder. (hint: use `if`, `file.exists` and `dir.create`)

Solutions

Write a conditional that checks whether a folder for your clean data exists. If not, it creates a new folder. (hint: use `if`, `file.exists` and `dir.create`)

Solutions

Write a conditional that checks whether a folder for your clean data exists. If not, it creates a new folder. (hint: use `if`, `file.exists` and `dir.create`)

Write your dataset into an .rds file in your new clean data folder.

Solutions

Write a conditional that checks whether a folder for your clean data exists. If not, it creates a new folder. (hint: use `if`, `file.exists` and `dir.create`)

Write your dataset into an .rds file in your new clean data folder.

Dates

Dates

Dates are represented as the number of days since 1970-01-01, with negative values for earlier dates.

Dates

Dates are represented as the number of days since 1970-01-01, with negative values for earlier dates.

When importing data, dates will usually be in character format.

Dates

Dates are represented as the number of days since 1970-01-01, with negative values for earlier dates.

When importing data, dates will usually be in character format.

If you don't like the way R displays the date, you can format it.[1<\sup>]

[1] You can find more about formatting dates here.

Dates

Sometimes, they will be in numeric format - e.g. when coming from Excel!

Dates

Sometimes, they will be in numeric format - e.g. when coming from Excel!

```
excel_dates <- c(0, 1)

windows_dates<-as.Date(excel_dates, origin="1899-12-30")
mac_dates<-as.Date(excel_dates, origin="1904-01-01")

windows_dates
```

```
## [1] "1899-12-30" "1899-12-31"
```

```
mac_dates
```

```
## [1] "1904-01-01" "1904-01-02"
```

Dates

Sometimes, they will be in numeric format - e.g. when coming from Excel!

```
excel_dates <- c(0, 1)

windows_dates<-as.Date(excel_dates, origin="1899-12-30")
mac_dates<-as.Date(excel_dates, origin="1904-01-01")

windows_dates
```

```
## [1] "1899-12-30" "1899-12-31"
```

```
mac_dates
```

```
## [1] "1904-01-01" "1904-01-02"
```

> Und was ist die Moral von der Geschicht'?
>
> Nutze Microsoft Excel nicht![1]

[1] Quote from this person.

Data Transformations

The dplyr package

- Developed by Hadley Wickham of RStudio

The dplyr package

- Developed by Hadley Wickham of RStudio
- An optimized and distilled version of plyr package (also by Hadley)

The dplyr package

- Developed by Hadley Wickham of RStudio
- An optimized and distilled version of plyr package (also by Hadley)
- Does not provide any "new" functionality per se, but greatly simplifies existing functionality in R

The dplyr package

- Developed by Hadley Wickham of RStudio
- An optimized and distilled version of plyr package (also by Hadley)
- Does not provide any "new" functionality per se, but greatly simplifies existing functionality in R
- Provides a "grammar" (in particular, verbs) for data manipulation

The dplyr package

- Developed by Hadley Wickham of RStudio
- An optimized and distilled version of plyr package (also by Hadley)
- Does not provide any "new" functionality per se, but greatly simplifies existing functionality in R
- Provides a "grammar" (in particular, verbs) for data manipulation

Is very fast, as many key operations are coded in C++

dplyr functions

- `select`: return a subset of the columns of a data frame

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables
- `group_by`: converts a table into a table where variables form groups on which the analysis can be done

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables
- `group_by`: converts a table into a table where variables form groups on which the analysis can be done
- `summarise` / `summarize`: generate summary statistics of different variables in the data frame, possibly within strata (subgroups)

dplyr functions

- `select`: return a subset of the columns of a data frame
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables
- `group_by`: converts a table into a table where variables form groups on which the analysis can be done
- `summarise` / `summarize`: generate summary statistics of different variables in the data frame, possibly within strata (subgroups)

There is also a handy `print` method that prevents you from printing a lot of data to the console.

How to use dplyr functions

- The first argument is a data frame
- The subsequent arguments describe what to do with it
- You can refer to columns in the data frame directly without using the $ operator (just use the names)
- The result is a new data frame

Aside

You can do the same operation with many types of functions.

```
my_data<-iris #this is some inbuilt data on flowers
head(my_data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Aside

Imagine you want to select the first two columns from a dataset.

```
library(dplyr)
head(select(my_data, Sepal.Length, Sepal.Width))
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
## 6          5.4         3.9
```

Aside

Imagine you want to select the first two columns from a dataset.

```
library(dplyr)
head(select(my_data, Sepal.Length, Sepal.Width))
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
## 6          5.4         3.9
```

```
head(my_data[,1:2])
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
## 6          5.4         3.9
```

Exercises

1. The `Ecdat` package contains datasets for economics. Load it and define `affairs<-as_data_frame(Fair)`. It contains a cross-section of 601 individuals in the United States, some of their characteristics and how many extramarital affairs they had in the past year. Have a look at the data using a command of your choice. You can find a data description is two slides ahead.
2. Show only entries for women.
3. Show only entries for very religious, childless women who had at least one affair and save the result in a new dataset. Then, delete the dataset.
4. Create a new dataset consisting only of the variables `sex`, `age`, `nbaffairs` and save it.
5. Rename the variable `nbaffairs` into `headcount` in your new dataset.
6. Keep all variables of your new dataset except `sex`.
7. Create a variable to capture the age at which a person got married
8. Order the data frame `affairs` so that you first have the women from old to young and at the end the men from old to young. (hint: use the `dplyr`package and the function `desc`)

Solutions

The `Ecdat` package contains datasets for economics. Load it and define `affairs<-as_data_frame(Fair)`. It contains a cross-section of 601 individuals in the United States, some of their characteristics and how many extramarital affairs they had in the past year. Have a look at the data using a command of your choice.

Solutions

As you can see, the dataframe contains 9 variables.

- **sex**: factor: male or female
- **age**: age in years
- **ym**: number of years married
- **child**: factor: yes or no?
- **religious**: How religious from 1 (anti) to 5 (very)?
- **education**: education in years
- **occupation**: occupation, from 1 to 7, according to hollingshead classification
- **rate**: self rating of marriage, from 1 (very unhappy) to 5 (very happy)
- **nbaffairs**: number of affairs in past year

Solutions

Show only entries for women.

Solutions

Show only entries for very religious, childless women who had at least one affair and save the result in a new dataset. Then, delete the dataset.

Solutions

Create a new dataset consisting only of the variables `sex`, `age`, `nbaffairs` and save it.

Rename the variable `nbaffairs` into `headcount` in your new dataset.

Keep all variables of your new dataset except `sex`.

Solutions

Create a variable to capture the age at which a person got married

Solutions

Order the data frame `affairs` so that you first have the women from old to young and at the end the men from old to young. (hint: use the `dplyr`package and the function `desc`)

Solutions

Let's check the top of the dataset.

Solutions

And let's see the bottom of the dataset.