Introduction to “R”
Data wrangling

Arndt Leininger

@a_leininger
arndt.leininger@fu-berlin.de

28 September 2019

Installing packages

Installing packages

- In the following section we will use the packages `dplyr` and `tidyr`
- These packages were created by Hadley Wickham and are available on CRAN
- They are not shipped with R so you need to install them before loading and using them
- `install.packages()` takes a package name as input and installs the requisite package

```
install.packages("dplyr")
install.packages("tidyr")
```

or

```
install.packages(c("dplyr", "tidyr"))
```

- Package names must be passed on as a character vector

Loading packages

- Before you can use the functions provided by the new packages you need to load them

```r
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

Duplicate function names

Sometimes two different functions have the same name

```r
lag(c(1, 2, 3))
```

```
## [1] NA  1  2
```

```r
stats::lag(c(1, 2, 3))
```

```
## [1] 1 2 3
## attr(,"tsp")
## [1] 0 2 1
```

::

Using `[package]::[function]` you can use a function without first loading the package with `library`

```
stringr::str_replace(c("Hello", "World", "!"), "!", ".")
```

```
## [1] "Hello" "World" "."
```

Quick hands-on

1. Install the packages `dplyr` and `tidyr`.
2. Load `dplyr` and `tidyr`.
3. Create a vector containing the numbers `0, 1, 1, 2, 3, 1, 0, 0`
4. Use the function `lag()` on that vector.
5. use `dplyr::lag()` and `stats:lag()` on the vector. Which one is used if you use `lag()`?

Beware of `require()`

- Don't use `require()` which does not provide an error message if a package could not be loaded

```r
require("thispackagedoesnotexit")
```

```
## Loading required package: thispackagedoesnotexit
```

```
## Warning in library(package, lib.loc = lib.loc,
## character.only = TRUE, logical.return = TRUE, : there
## is no package called 'thispackagedoesnotexit'
```

```r
library("thispackagedoesnotexit")
```

```
## Error in library("thispackagedoesnotexit"): there is no packa
```

https://yihui.name/en/2014/07/library-vs-require/

Installing development versions

```r
install.packages("devtools")

devtools::install_github("bookdown")
```

Data wrangling

Data wrangling

- Remember that some say that quantitative social science research is 80% data collection, cleaning and wrangling and 20% actual analysis.
- Data don't always come in nicely formatted files (e.g. .dta). Sometimes you have a data source or sources in files that aren't set up for R to read and start analyzing right away.
- You often have to do a bit of work to get things into the format you want: the more of that work is recorded the better.
- `R` and the packages `dpylr` and `tidyr` amongst others help you reduce the time you spend wrangling and makes that time more fun

dplyr

- `dplyr` is a package which simplifies data wrangling a lot
- It provides a number of functions which together allow a wide range of data manipulation
  - `arrange()`: sorting data
  - `filter()`: subsetting data
  - `select()` (and `rename()`): subsetting data (by column)
  - `mutate()`: add or replace variables
  - `group_by()`: do a `mutate()` or `summarize()` group-wise
  - `summarize()`: aggregate data
- It provides 'piping' functionality

Wrangling

Sort domestic cars by price

```
df %>% filter(foreign == "Domestic") %>% arrange(price) %>%
    select(make, foreign, price)
```

```
##                  make  foreign price
## 1        Merc. Zephyr Domestic  3291
## 2      Chev. Chevette Domestic  3299
## 3         Chev. Monza Domestic  3667
## 4          AMC Spirit Domestic  3799
## 5        Merc. Bobcat Domestic  3829
## 6          Chev. Nova Domestic  3955
## 7          Dodge Colt Domestic  3984
## 8      Dodge Diplomat Domestic  4010
## 9        Plym. Volare Domestic  4060
## 10      Buick Skylark Domestic  4082
## 11        AMC Concord Domestic  4099
## 12      Pont. Sunbird Domestic  4172
## 13         Olds Omega Domestic  4181
## 14       Ford Mustang Domestic  4187
```

Piping

- %>% is a 'piping' operator, it passes on output from a function to another, allowing you to write code in the order of execution
- The package `dpylr` provides piping functionality for R through the package `magrittr`

```r
df %>% filter(price == max(price)) %>% select(make, price)
```

```
##           make price
## 1 Cad. Seville 15906
```

```r
# is easier to read than
select(filter(df, price == max(price)), make, price)
```

```
##           make price
## 1 Cad. Seville 15906
```

Keyboard shortcut Ctrl + ⇧ + M produces the piping symbol

Wrangling

The average price of domestic cars is also simple to obtain

```
df %>% filter(foreign == 'Domestic') %>%
  summarise(average = mean(price))
```

```
##    average
## 1 6072.423
```

Read more about it at http://blog.revolutionanalytics.com/2014/07/magrittr-simplifying-r-code-with-pipes.html

Creating a new variable

Create a variable which indicates whether a car's price is above average, distinguishing between domestic and foreign cars.

```
tmp <- df %>% group_by(foreign) %>%
  mutate(relativeprice = price - mean(price),
         expensive = relativeprice > 0)
tmp %>% select(price, relativeprice, expensive)
```

```
## Adding missing grouping variables: `foreign`

## # A tibble: 74 x 4
## # Groups:   foreign [2]
##    foreign  price relativeprice expensive
##    <fct>    <int>         <dbl> <lgl>
##  1 Domestic  4099        -1973. FALSE
##  2 Domestic  4749        -1323. FALSE
##  3 Domestic  3799        -2273. FALSE
##  4 Domestic  4816        -1256. FALSE
##  5 Domestic  7827         1755. TRUE
##  6 Domestic  5788         -284. FALSE
```

Aggregating data

- `summarize()` in conjunction with `group_by()` can be used to aggregate data
- function used within `summarize()` must return a single value

```
df %>% group_by(foreign) %>% summarize(price = mean(price))
```

```
## # A tibble: 2 x 2
##   foreign  price
##   <fct>    <dbl>
## 1 Domestic 6072.
## 2 Foreign  6385.
```

A tibble

- `as_tibble()` turns a data.frame into a slightly modified data.frame (a 'tibble') with improved printing capabilities

```
df <- as_tibble(df)
df
```

```
## # A tibble: 74 x 12
##    make price   mpg rep78 headroom trunk weight length
##  * <ch> <int> <int> <int>    <dbl> <int>  <int>  <int>
##  1 AMC~  4099    22     3      2.5    11   2930    186
##  2 AMC~  4749    17     3      3      11   3350    173
##  3 AMC~  3799    22    NA      3      12   2640    168
##  4 Bui~  4816    20     3      4.5    16   3250    196
##  5 Bui~  7827    15     4      4      20   4080    222
##  6 Bui~  5788    18     3      4      21   3670    218
##  7 Bui~  4453    26    NA      3      10   2230    170
##  8 Bui~  5189    20     3      2      16   3280    200
##  9 Bui~ 10372    16     3      3.5    17   3880    207
## 10 Bui~  4082    19     3      3.5    13   3400    200
## # ... with 64 more rows, and 4 more variables:
```

A tibble

- `as_tibble()` turns a data.frame into a slightly modified data.frame with improved printing capabilities

```
df <- as_tibble(df)
print(df, n = 5)
```

```
## # A tibble: 74 x 12
##   make  price   mpg rep78 headroom trunk weight length
## * <chr> <int> <int> <int>    <dbl> <int>  <int>  <int>
## 1 AMC ~  4099    22     3      2.5    11   2930    186
## 2 AMC ~  4749    17     3      3      11   3350    173
## 3 AMC ~  3799    22    NA      3      12   2640    168
## 4 Buic~  4816    20     3      4.5    16   3250    196
## 5 Buic~  7827    15     4      4      20   4080    222
## # ... with 69 more rows, and 4 more variables:
## #   turn <int>, displacement <int>, gear_ratio <dbl>,
## #   foreign <fct>
```

glimpse()

```r
glimpse(df)
```

```
## Observations: 74
## Variables: 12
## $ make         <chr> "AMC Concord", "AMC Pacer", "...
## $ price        <int> 4099, 4749, 3799, 4816, 7827,...
## $ mpg          <int> 22, 17, 22, 20, 15, 18, 26, 2...
## $ rep78        <int> 3, 3, NA, 3, 4, 3, NA, 3, 3, ...
## $ headroom     <dbl> 2.5, 3.0, 3.0, 4.5, 4.0, 4.0,...
## $ trunk        <int> 11, 11, 12, 16, 20, 21, 10, 1...
## $ weight       <int> 2930, 3350, 2640, 3250, 4080,...
## $ length       <int> 186, 173, 168, 196, 222, 218,...
## $ turn         <int> 40, 40, 35, 40, 43, 43, 34, 4...
## $ displacement <int> 121, 258, 121, 196, 350, 231,...
## $ gear_ratio   <dbl> 3.58, 2.53, 3.08, 2.93, 2.41,...
## $ foreign      <fct> Domestic, Domestic, Domestic,...
```

Tidyverse

```r
install.packages("tidyverse")
library(tidyverse)
```

- collection of useful R packages by Hadley Wickham (e.g. dplyr, tidyr, ggplot2, readr, stringr)
- https://www.tidyverse.org/

Hands-on I

Hands-on I

`hands-on/03_datawrangling/hands-on1.R`

Merging and reshaping data

Merging data

- `dplyr` provides great functions for merging data
- `left_join` keeps all observations in `d1` and adds those in `d2` that can be matched
- Takes two data.frames as input
- Identifier variable names can differ

```
d1 <- read.csv("data/d1.csv")
d2 <- read.csv("data/d2.csv")
m <- left_join(d1, d2, by = c(nation = "country", "year"))
```

- there are also: `inner_join`, `right_join`, `semi_join`, `anti_join`, `full_join`

Tidying data

- Tidy data refers to data in which each row is an observation and each column a variable
- This is also known as 'long' format as opposed to 'wide' format
- Real-world data do not always come in this form
- The package `tidyr` provides functions to 'clean up' data

```
install.packages("tidyr")
library(tidyr)
```

Reshaping data

- Let's use a very simple made-up `data.frame`

```
messy <- data.frame(name = c("Wilbur", "Petunia", "Gregory"),
    a = c(67, 80, 64), b = c(56, 90, 50))

messy
```

```
##      name  a  b
## 1  Wilbur 67 56
## 2 Petunia 80 90
## 3 Gregory 64 50
```

- Imagine this is data from a clinical trial. We have three variables, patient name, drug and heartrate, but only one is a variable yet
- Based on https://blog.rstudio.org/2014/07/22/introducing-tidyr/

Reshaping data

- We will use `tidyr`'s `gather()` to gather the a and b columns into key-value pairs of `drug` and `heartrate`

```
tidy <- gather(messy, key = drug, value = heartrate, a:b)

tidy
```

```
##      name drug heartrate
## 1  Wilbur    a        67
## 2 Petunia    a        80
## 3 Gregory    a        64
## 4  Wilbur    b        56
## 5 Petunia    b        90
## 6 Gregory    b        50
```

And back to wide format

```r
spread(tidy, drug, heartrate)
```

```
##      name  a  b
## 1 Gregory 64 50
## 2 Petunia 80 90
## 3  Wilbur 67 56
```

Hands-on II

Hands-on II

`hands-on/03_datawrangling/hands-on2.R`