Introduction to R for Data Analysis in the Health Sciences: Lecture 3

Amy Willis, Biostatistics, UW

01 November, 2019

Getting your feedback

Next week I will ask you to fill out an anonymous poll to tell me how the course is going for you (pacing/topics/homework/lectures. . .)

In the meantime, I have created a form for you to give me anonymous feedback at any time

[only available to enrolled students]

I appreciate any constructive feedback!

Today

- Data cleaning
  - Finding errors
  - Correcting errors
  - Correcting ambiguities
  - Finding missing data
- Reading in Excel files
- R Projects

Data cleaning

Data cleaning as an underappreciated and under-taught skill that takes time and practice

There are many different ways to do data cleaning, and none of them is "best"

Guidelines are better than rules: be sensible, transparent and careful

Basic workflow: Check, double check, and triple check your data and analysis

Read in the data

This week's example is a Microsoft Excel spreadsheet, which `tidyverse` can't handle. I use `readxl` instead:

```
# install.packages("readxl")
library(readxl)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------- tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Read in the data

```
labs <- read_xlsx("datasets/TM_data.xlsx", sheet = 5)
```

Looking at the data

What do you see?

```
> labs
# A tibble: 3,553 x 15
   Identifier LabTime LabDate             LabTest LabResult `45lo, vegfr2 p…
   <chr>      <chr>   <dttm>              <chr>   <chr>     <chr>
 1 TM01       prestu… 2007-05-15 00:00:00 <NA>    <NA>      <NA>
 2 TM01       cycle … 2007-06-13 00:00:00 CERULO… 30        207
 3 TM01       cycle … 2007-06-26 00:00:00 CERULO… 22        <NA>
 4 TM01       cycle … 2007-07-11 00:00:00 CERULO… 16        46
 5 TM01       cycle … 2007-07-17 00:00:00 CERULO… 16        <NA>
 6 TM01       cycle … 2007-07-24 00:00:00 CERULO… 13        <NA>
 7 TM01       cycle … 2007-08-09 00:00:00 CERULO… 11        Not Done
 8 TM01       cycle … 2007-09-04 00:00:00 CERULO… 12        Not Done
 9 TM01       cycle … 2007-09-27 00:00:00 CERULO… 12        25
10 TM01       cycle … 2007-10-26 00:00:00 CERULO… 11        16
# … with 3,543 more rows, and 7 more variables: BSA <chr>, AvgDailyPillsTake
#   AvgDailyPillsPrescribedPerCycle <chr>, LOXL2 <chr>, C1M <chr>, `Pro-C3`
```

Figure 1:

Looking at the data

What do you see?

- `LabResult` and other variables look numeric, but `R` interpreted them as characters
- Missing data is coded in different ways

Looking at the data

```
labs %>% names
```

```
##  [1] "Identifier"
##  [2] "LabTime"
##  [3] "LabDate"
##  [4] "LabTest"
##  [5] "LabResult"
##  [6] "45lo, vegfr2 pos, CD133 pos EPC/ul"
##  [7] "45lo, CD133 pos EPC/ul"
##  [8] "45 pos, vegfr1+, 34+ HSC/ul"
##  [9] "BSA"
## [10] "AvgDailyPillsTakenPerCycle"
## [11] "AvgDailyPillsPrescribedPerCycle"
## [12] "LOXL2"
## [13] "C1M"
## [14] "Pro-C3"
## [15] "C6M"
```

Looking at the data

Let's start by making a copy of the data

```
cleaned_labs <- labs
```

Now we will edit `cleaned_labs`, and if we make a mistake, we can start again with `cleaned_labs <- labs`

Looking at the data

It's a good idea to avoid variable names with spaces, and excessively long variable names

```
cleaned_labs <- labs %>%
  rename(vegEPC = `45lo, vegfr2 pos, CD133 pos EPC/ul`,
         EPC = `45lo, CD133 pos EPC/ul`,
         HSC = `45 pos, vegfr1+, 34+ HSC/ul`,
         dose_taken = AvgDailyPillsTakenPerCycle,
         dose_prescribed = AvgDailyPillsPrescribedPerCycle)
```

Looking at the data

```
cleaned_labs
```

```
## # A tibble: 3,553 x 15
##    Identifier LabTime LabDate             LabTest LabResult vegEPC E
##    <chr>      <chr>   <dttm>              <chr>   <chr>     <chr>  <
##  1 TM01       prestu~ 2007-05-15 00:00:00 <NA>    <NA>      <NA>   <
##  2 TM01       cycle ~ 2007-06-13 00:00:00 CERULO~ 30        207    1
##  3 TM01       cycle ~ 2007-06-26 00:00:00 CERULO~ 22        <NA>   <
##  4 TM01       cycle ~ 2007-07-11 00:00:00 CERULO~ 16        46     3
##  5 TM01       cycle ~ 2007-07-17 00:00:00 CERULO~ 16        <NA>   <
##  6 TM01       cycle ~ 2007-07-24 00:00:00 CERULO~ 13        <NA>   <
##  7 TM01       cycle ~ 2007-08-09 00:00:00 CERULO~ 11        Not D~ N
##  8 TM01       cycle ~ 2007-09-04 00:00:00 CERULO~ 12        Not D~ N
##  9 TM01       cycle ~ 2007-09-27 00:00:00 CERULO~ 12        25     4
## 10 TM01       cycle ~ 2007-10-26 00:00:00 CERULO~ 11        16     2
## # ... with 3,543 more rows, and 8 more variables: HSC <chr>, BSA <ch
## #   dose_taken <chr>, dose_prescribed <chr>, LOXL2 <chr>, C1M <chr>,
## #   `Pro-C3` <chr>, C6M <chr>
```

Take care: the format of `rename` is "rename(new_name = old_name)"

Renaming: What's a good name?

Prioritise *guidelines* over *rules*

- Something memorable and not easily-confused with other objects
- Lowercase letters; separate words with underscores
  - `cancer_df`
  - `first_enrichment_test`
- Words better than numbers
  - `cancer_data_treatment_only` versus `cancer_data2`
- Avoid names of existing functions
  - e.g. `summary`
  - Note Some one-letter choices (`c`, `t`...) are already used by R as names of functions; use multiple letters

Save on typing/mousing with informative names!

Checking the data

And now we can work through the columns in turn, checking for craziness as we go

Checking the data

Looking for craziness in dates: check out the first few

```
cleaned_labs$LabDate %>% head
```

```
## [1] "2007-05-15 UTC" "2007-06-13 UTC" "2007-06-26 UTC" "2007-07-11 U
## [5] "2007-07-17 UTC" "2007-07-24 UTC"
```

Nothing out of the ordinary here. . .

Checking the data

Looking for craziness in dates: check out the first few.

```
cleaned_labs$LabDate %>% sort %>% head
```

```
## [1] "1900-01-03 UTC" "2007-05-15 UTC" "2007-06-07 UTC" "2007-06-08 U
## [5] "2007-06-12 UTC" "2007-06-13 UTC"
```

```
cleaned_labs$LabDate %>% sort(decreasing=T) %>% head
```

```
## [1] "2017-12-27 UTC" "2017-12-26 UTC" "2017-12-20 UTC" "2017-11-21 U
## [5] "2017-11-21 UTC" "2017-11-15 UTC"
```

We found our first error!

Checking the data

```
cleaned_labs %>%
  filter(LabDate == min(LabDate)) %>%
  print(width = Inf)
```

```
## # A tibble: 1 x 15
##   Identifier LabTime LabDate            LabTest LabResult vegEPC EP
##   <chr>      <chr>   <dttm>             <chr>   <chr>     <chr>  <c
## 1 TM54       <NA>    1900-01-03 00:00:00 <NA>    <NA>      <NA>   <N
##   HSC   BSA   dose_taken dose_prescribed LOXL2 C1M   `Pro-C3` C6M
##   <chr> <chr> <chr>      <chr>           <chr> <chr> <chr>    <chr>
## 1 <NA>  <NA>  <NA>       <NA>            <NA>  <NA>  <NA>     <NA>
```

This row appears to contain no information except an impossible date. Let's remove it!

Checking the data

Removing the offending data point:

```
cleaned_labs <- cleaned_labs %>%
  filter(LabDate != min(LabDate))
cleaned_labs$LabDate %>% sort %>% head
```

```
## [1] "2007-05-15 UTC" "2007-06-07 UTC" "2007-06-08 UTC" "2007-06-12 U
## [5] "2007-06-13 UTC" "2007-06-13 UTC"
```

```
cleaned_labs$LabDate %>% sort(decreasing=T) %>% head
```

```
## [1] "2017-12-27 UTC" "2017-12-26 UTC" "2017-12-20 UTC" "2017-11-21 U
## [5] "2017-11-21 UTC" "2017-11-15 UTC"
```

Checking the data

Now we can move onto the next variables. . .

Checking the data

```
labs$LabResult %>% unique %>% sort
```

```
##  [1] "<4"   "10"   "10.4" "10.8" "11"   "11.2" "11.6" "11.7" "12"
## [11] "12.3" "12.9" "13"   "13.1" "13.6" "13.7" "13.9" "14"   "14.1"
## [21] "14.5" "14.6" "14.7" "14.9" "15"   "15.1" "15.4" "15.9" "16"
## [31] "16.2" "16.4" "16.7" "16.9" "17"   "17.3" "17.8" "18"   "18.2"
## [41] "18.6" "181"  "19"   "19.1" "20"   "20.4" "20.7" "21"   "21.2"
## [51] "22"   "22.3" "22.5" "22.7" "23"   "23.3" "23.6" "23.9" "24"
## [61] "25.1" "25.8" "26"   "27"   "27.5" "27.8" "28"   "28.7" "29"
## [71] "29.6" "29.9" "3"    "30"   "31"   "32"   "33"   "34"   "35"
## [81] "37"   "38"   "39"   "4"    "40"   "41"   "42"   "43"   "47"
## [91] "6"    "7"    "8"    "8.2"  "8.9"  "9"    "9.2"  "9.3"  "9.5"
```

How will we deal with "<4"?

Checking the data

```
cleaned_labs <- cleaned_labs %>%
  mutate(LabResultImpute = ifelse(LabResult == "<4", 3, LabResult))
```

Caution – the most appropriate way to deal with this will depend on the context!

- Remember: `NA` = "I don't know"
  - Censored data, e.g., "Less than 4" is not the same as "I don't know"

Checking the data

```
cleaned_labs$LabResultImpute %>% unique %>% sort
```

```
##  [1] "10"   "10.4" "10.8" "11"   "11.2" "11.6" "11.7" "12"   "12.1"
## [11] "12.9" "13"   "13.1" "13.6" "13.7" "13.9" "14"   "14.1" "14.4"
## [21] "14.6" "14.7" "14.9" "15"   "15.1" "15.4" "15.9" "16"   "16.1"
## [31] "16.4" "16.7" "16.9" "17"   "17.3" "17.8" "18"   "18.2" "18.4"
## [41] "181"  "19"   "19.1" "20"   "20.4" "20.7" "21"   "21.2" "21.7"
## [51] "22.3" "22.5" "22.7" "23"   "23.3" "23.6" "23.9" "24"   "25"
## [61] "25.8" "26"   "27"   "27.5" "27.8" "28"   "28.7" "29"   "29.1"
## [71] "29.9" "3"    "30"   "31"   "32"   "33"   "34"   "35"   "36"
## [81] "38"   "39"   "4"    "40"   "41"   "42"   "43"   "47"   "5"
## [91] "7"    "8"    "8.2"  "8.9"  "9"    "9.2"  "9.3"  "9.5"
```

Looks good!

Checking the data

Now convert this column from character strings to numeric values:

```
cleaned_labs <- cleaned_labs %>%
  mutate(LabResultImputeNumeric = as.numeric(LabResultImpute))
```

Checking the data

```
cleaned_labs$LabResultImputeNumeric %>% unique %>% sort
```

```
##  [1]   3.0   4.0   5.0   6.0   7.0   8.0   8.2   8.9   9
## [12]   9.5  10.0  10.4  10.8  11.0  11.2  11.6  11.7  12
## [23]  12.9  13.0  13.1  13.6  13.7  13.9  14.0  14.1  14
## [34]  14.7  14.9  15.0  15.1  15.4  15.9  16.0  16.1  16
## [45]  16.9  17.0  17.3  17.8  18.0  18.2  18.4  18.6  19
## [56]  20.4  20.7  21.0  21.2  21.7  22.0  22.3  22.5  22
## [67]  23.6  23.9  24.0  25.0  25.1  25.8  26.0  27.0  27
## [78]  28.7  29.0  29.1  29.6  29.9  30.0  31.0  32.0  33
## [89]  36.0  37.0  38.0  39.0  40.0  41.0  42.0  43.0  47
```

Checking the data

```
cleaned_labs <- cleaned_labs %>%
  mutate(LabResultImputeNumeric = as.numeric(LabResultImpute))
```

Checking the data

Check that this looks correct by looking at the rows and columns that we changed:

```
cleaned_labs %>%
  select(LabResult, LabResultImpute, LabResultImputeNumeric
  filter(LabResult != LabResultImputeNumeric)
```

```
## # A tibble: 5 x 3
##   LabResult LabResultImpute LabResultImputeNumeric
##   <chr>     <chr>                           <dbl>
## 1 <4        3                                   3
## 2 <4        3                                   3
## 3 <4        3                                   3
## 4 <4        3                                   3
## 5 <4        3                                   3
```

Checking the data

We can now drop/overwrite the original column if we wish:

```
cleaned_labs <- cleaned_labs %>%
  select(-LabResult, -LabResultImpute)
```

And we can rename the column we created:

```
cleaned_labs <- cleaned_labs %>%
  rename(LabResult = LabResultImputeNumeric)
```

Take care: the format of `rename` is "rename(new_name = old_name)"

Recap

Putting it all together:

```
cleaned_labs <- labs %>%
  rename(vegEPC = `45lo, vegfr2 pos, CD133 pos EPC/ul`,
         EPC = `45lo, CD133 pos EPC/ul`,
         HSC = `45 pos, vegfr1+, 34+ HSC/ul`,
         dose_taken = AvgDailyPillsTakenPerCycle,
         dose_prescribed = AvgDailyPillsPrescribedPerCycle) %>%
  filter(LabDate != min(LabDate)) %>%
  mutate(LabResultImpute = ifelse(LabResult == "<4", 3, LabResult)) %>%
  mutate(LabResultImputeNumeric = as.numeric(LabResultImpute)) %>%
  select(-LabResult, -LabResultImpute) %>%
  rename(LabResult = LabResultImputeNumeric)
```

Tools for data cleaning

New functions:

- `rename`
- `ifelse`
- `mutate`
- `as.numeric`

See also `as.character`

Check the data

Let's move onto the next variable!

```
cleaned_labs$LabResult %>% sort %>% head
```

```
## [1] 3 3 3 3 3 3
```

```
cleaned_labs$LabResult %>% sort(decreasing=T) %>% head
```

```
## [1] 181  47  43  43  43  42
```

This was incorrectly entered – the actual value is 18.1

Check the data

```
cleaned_labs <- cleaned_labs %>%
  mutate(LabResult = ifelse(LabResult == 181, 18.1, LabResu
cleaned_labs$LabResult %>% sort %>% head
```

```
## [1] 3 3 3 3 3 3
```

```
cleaned_labs$LabResult %>% sort(decreasing=T) %>% head
```

```
## [1] 47 43 43 43 42 42
```

Check the data

```
cleaned_labs$vegEPC %>% unique %>% sort
```

```
##   [1] "0"                "0.00"
##   [3] "1"                "10"
##   [5] "10.38826126"      "10.55"
##   [7] "10.67257763"      "100"
##   [9] "101"              "102"
##  [11] "103"              "104"
##  [13] "105"              "106"
##  [15] "107"              "108"
##  [17] "109"              "11"
##  [19] "11.21728292"      "11.42992436"
##  [21] "11.47"            "11.78771237"
##  [23] "11.82"            "110"
##  [25] "111"              "112"
##  [27] "113"              "114"
##  [29] "115"              "116"
##  [31] "117"              "118"
```

Check the data

There are a number of rows that contain only NAs. How do we filter using multiple columns at once?

```
cleaned_labs %>%
  filter(!(is.na(vegEPC) & is.na(EPC)))
```

```
## # A tibble: 2,813 x 15
##    Identifier LabTime LabDate                LabTest vegEPC
##    <chr>      <chr>   <dttm>                 <chr>    <dbl>
##  1 TM01       cycle ~ 2007-06-13 00:00:00 CERULO~    207
##  2 TM01       cycle ~ 2007-07-11 00:00:00 CERULO~     46
##  3 TM01       cycle ~ 2007-08-09 00:00:00 CERULO~     NA
##  4 TM01       cycle ~ 2007-09-04 00:00:00 CERULO~     NA
##  5 TM01       cycle ~ 2007-09-27 00:00:00 CERULO~     25
##  6 TM01       cycle ~ 2007-10-26 00:00:00 CERULO~     16
##  7 TM01       cycle ~ 2007-11-27 00:00:00 CERULO~     42
##  8 TM01       cycle ~ 2007-12-12 00:00:00 CERULO~     42
##  9 TM01       cycle ~ 2008-01-17 00:00:00 CERULO~     NA
```

Where is the data missing?

```
which(is.na(cleaned_labs), arr.ind=TRUE) %>% head
```

```
##       row col
## [1,]  672   2
## [2,] 1282   2
## [3,]    1   4
## [4,]   16   4
## [5,]   18   4
## [6,]   19   4
```

Save your cleaned data

```
write_csv(cleaned_labs, path="postprocessed_lab_data.csv")
```

Recommendations

- Avoid guessing. . . ask folks who generated the data!
  - e.g., is `sex = 1` male or female?
- Check, double check, and triple check
- Always code in a script, not in the console
  - So you can keep track of what you have done
- Write `# comments` so you remember what you did

Future you will be grateful!

Wrap up on data cleaning

Data cleaning with `R` in 2019 is much easier than it was in 2009 because of the `tidyverse`. Don't forget: `library(tidyverse)` at the start of every script!

Workflow versus product

Your workflow:

- Your operating system
- Your code editor (e.g. RStudio)
- The name of your home directory (mine is `/Users/adwillis`)
- The code you ran yesterday
- The programs you have installed at this time

Workflow versus product

Your product:

- The raw data
- The code that needs to be run on the raw data to get results
  - including dependencies
- The report that you produce at the end of your analysis

Workflow versus product

"Any R script you write should be written assuming that it will be run from a fresh R process with working directory set to the project directory. It creates everything it needs, in its own workspace or folder, and it touches nothing it did not create..." – Jenny Bryan

Workflow versus product

Avoid hardwiring your workflow into your product:

- Every distinct intellectual unit you work on should have its own folder
  - every homework, data analysis, every method, every research project, every class
- You can type `install.packages` into your console (that's workflow!), but use `library` in a script (that's product!)

Workflow versus product

- Portable
  - other computers
  - robust to your own reorganisation
- Polite
  - Avoids overwriting or interfering with other projects

Prof. Jenny Bryan feels very strongly about this

> *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* 🔥 *.*

> *If the first line of your R script is*
>
> `rm(list = ls())`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE* 🔥 *.*

Figure 2:

This certainly violates our class norms ("inclusive teaching", "recognise different learning styles"), so we are not going to do it! But I do encourage you to go `File -> New Project` instead of `setwd()`.

RStudio Projects

RStudio Projects are an alternative to `setwd` that avoids hardwiring your workflow into your product

- Create a project with File -> New Project
- RStudio creates `myproj.Rproj`, which goes in your working directory for that project
- Double-click on your `Rproj` file to open a fresh RStudio instance in the correct working directory
  - Has the capacity to load your `RData`
- You can run multiple R sessions using multiple RStudio Projects at once

Summary

- Tools for data cleaning
  - Be sensible: inspect your data
  - New tools: `mutate`, `ifelse`
  - Old tools: `filter`
- Consider using RStudio Projects instead of `setwd`

Next week: making beautiful graphics and plots!

The plan

The School of Public Health is hosting Ijeoma Oluo, the author of our SPH Common Book *So you want to talk about race*.

At 3:30pm-5pm today in Hogness Auditorium there will be a Roundtable Discussion on *Race, Racism & Public Health*.

There is no In-Class Exercise this week – everyone gets 2 points "for free".

Instead, please consider attending this important roundtable discussion.

The plan

- 5 minute break
- Feel free to start Homework 3, but we will end at 3:20pm today so we can all attend the *Race, Racism & Public Health* discussion from 3:30pm-5pm today in Hogness Auditorium
- Place a *yellow sticky note* on your computer to indicate you are stuck, or a *blue sticky note* to indicate you have a non-urgent question
- Homework due next week by 1 p.m. Friday
- Office hours on the syllabus
- Give me feedback at any time via [only available to enrolled students]