# Excel files

*Andrew Ba Tran*

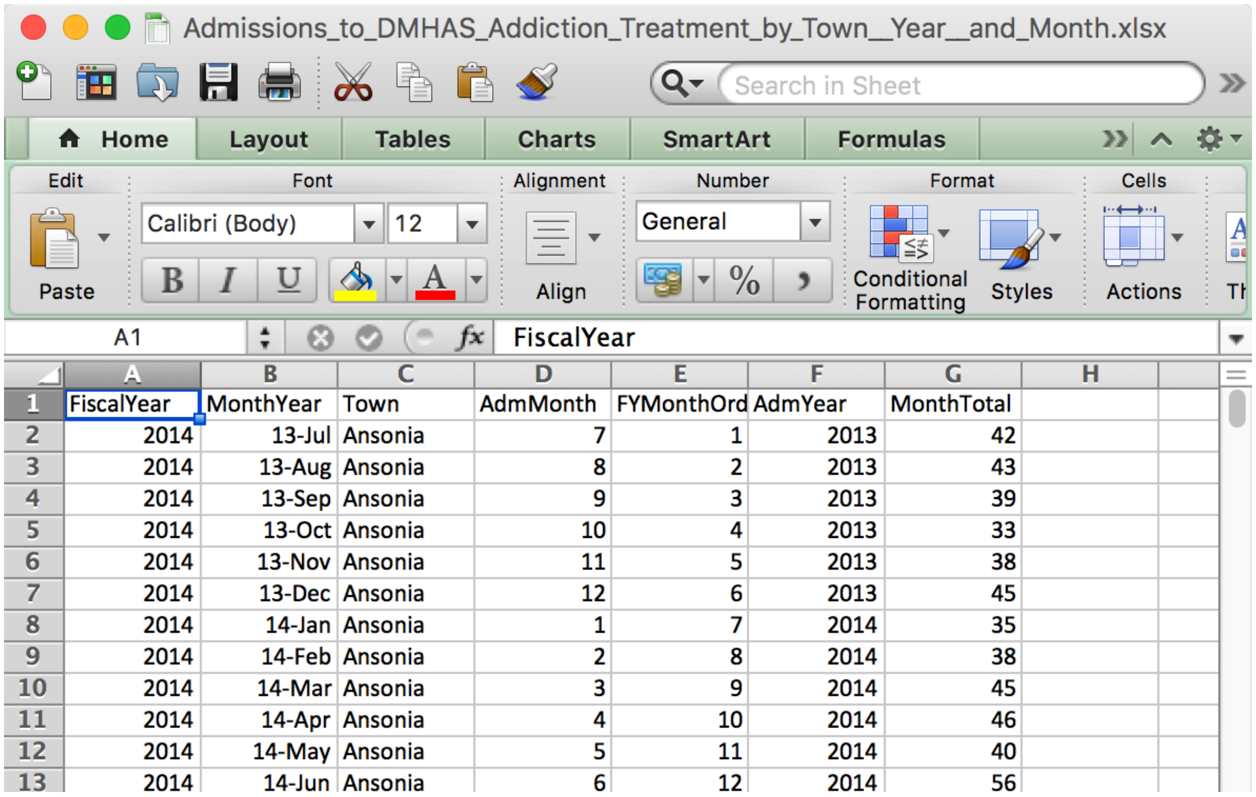
## Contents

This is from the second chapter of learn.r-journalism.com.

Excel spreadsheets are unique in that they can contain multiple spreadsheets as a workbook.

## What an Excel file looks like

Excel file names end with a **.xls** or **.xlsx**

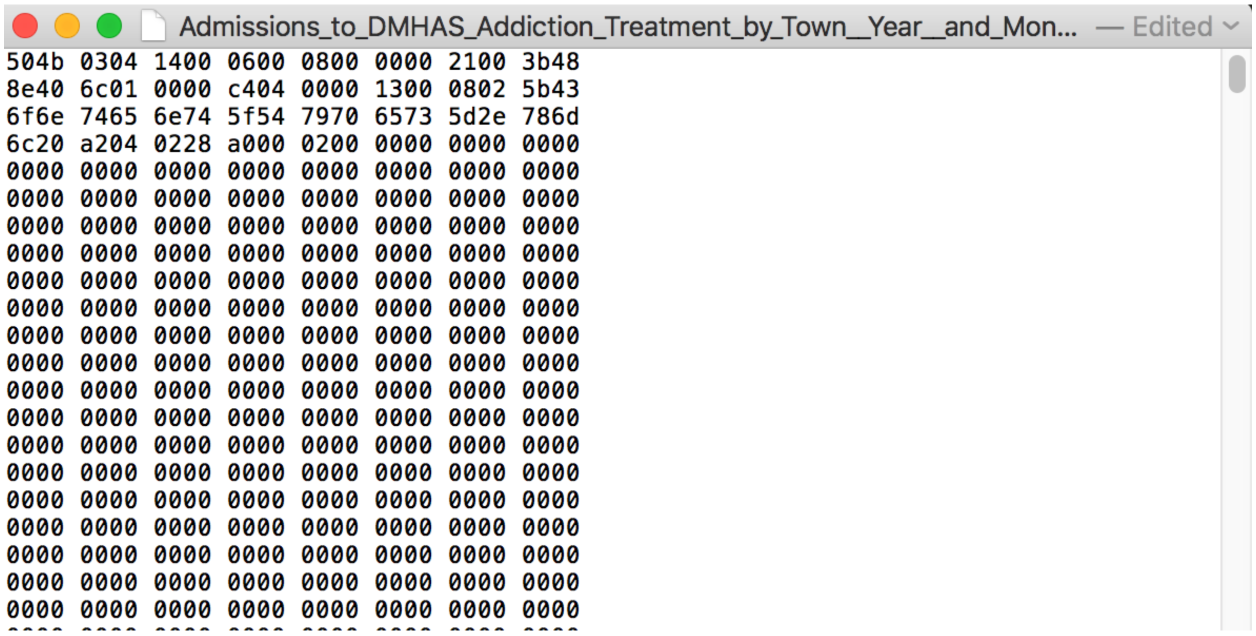


A1 FiscalYear

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | FiscalYear | MonthYear | Town | AdmMonth | FYMonthOrd | AdmYear | MonthTotal | |
| 2 | 2014 | 13-Jul | Ansonia | 7 | 1 | 2013 | 42 | |
| 3 | 2014 | 13-Aug | Ansonia | 8 | 2 | 2013 | 43 | |
| 4 | 2014 | 13-Sep | Ansonia | 9 | 3 | 2013 | 39 | |
| 5 | 2014 | 13-Oct | Ansonia | 10 | 4 | 2013 | 33 | |
| 6 | 2014 | 13-Nov | Ansonia | 11 | 5 | 2013 | 38 | |
| 7 | 2014 | 13-Dec | Ansonia | 12 | 6 | 2013 | 45 | |
| 8 | 2014 | 14-Jan | Ansonia | 1 | 7 | 2014 | 35 | |
| 9 | 2014 | 14-Feb | Ansonia | 2 | 8 | 2014 | 38 | |
| 10 | 2014 | 14-Mar | Ansonia | 3 | 9 | 2014 | 45 | |
| 11 | 2014 | 14-Apr | Ansonia | 4 | 10 | 2014 | 46 | |
| 12 | 2014 | 14-May | Ansonia | 5 | 11 | 2014 | 40 | |
| 13 | 2014 | 14-Jun | Ansonia | 6 | 12 | 2014 | 56 | |

## What an Excel file looks like on the inside

Weird, right? Definitely difficult to parse.


Admissions_to_DMHAS_Addiction_Treatment_by_Town__Year__and_Mon... — Edited

```
504b 0304 1400 0600 0800 0000 2100 3b48
8e40 6c01 0000 c404 0000 1300 0802 5b43
6f6e 7465 6e74 5f54 7970 6573 5d2e 786d
6c20 a204 0228 a000 0200 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

## Importing Excel files

- Importing Excel is complicated, **readxl package** is needed
- There are more other packages that handle Excel files and can build extra sheets, but we won't be needing them for this instance

## Importing Excel files

First, install the `readxl` package if you have not yet done so.

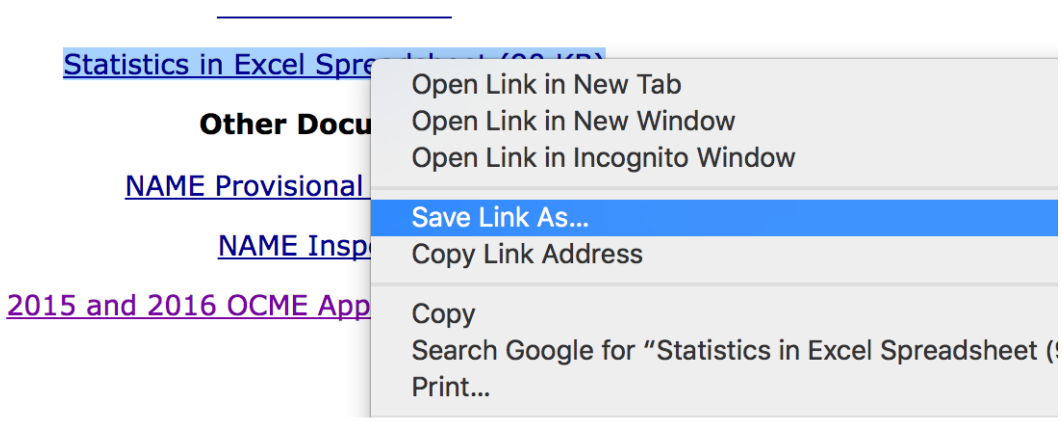
That will have `readxl` as part of the group of packages.

```
## If you don't have readxl installed, uncomment the line below and run it
#install.packages("readxl")
library(readxl)
```

Unlike a csv, you can't just copy and paste the URL for an Excel sheet.

You gotta download the file first.

Right click the link of the Excel data link and click **Save File As...**



## read_excel()

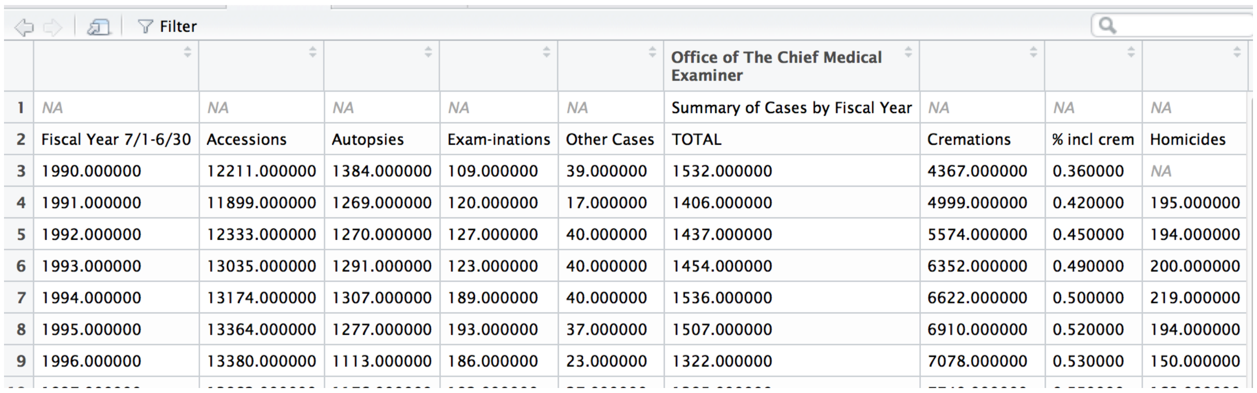
Excel spreadsheets have multiple sheets and it's best to explore what it looks like in Excel first because `read_excel()` requires specific sheets to be referred to when importing.

Give it a shot with the first sheet.

```
df_xl <- read_excel("data/StatisticsSummary.xls", sheet=1)
```

Check it

```
View(df_xl)
```



| | | | | | | Office of The Chief Medical Examiner | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | NA | NA | NA | NA | NA | Summary of Cases by Fiscal Year | NA | NA | NA |
| 2 | Fiscal Year 7/1-6/30 | Accessions | Autopsies | Exam-inations | Other Cases | TOTAL | Cremations | % incl crem | Homicides |
| 3 | 1990.000000 | 12211.000000 | 1384.000000 | 109.000000 | 39.000000 | 1532.000000 | 4367.000000 | 0.360000 | NA |
| 4 | 1991.000000 | 11899.000000 | 1269.000000 | 120.000000 | 17.000000 | 1406.000000 | 4999.000000 | 0.420000 | 195.000000 |
| 5 | 1992.000000 | 12333.000000 | 1270.000000 | 127.000000 | 40.000000 | 1437.000000 | 5574.000000 | 0.450000 | 194.000000 |
| 6 | 1993.000000 | 13035.000000 | 1291.000000 | 123.000000 | 40.000000 | 1454.000000 | 6352.000000 | 0.490000 | 200.000000 |
| 7 | 1994.000000 | 13174.000000 | 1307.000000 | 189.000000 | 40.000000 | 1536.000000 | 6622.000000 | 0.500000 | 219.000000 |
| 8 | 1995.000000 | 13364.000000 | 1277.000000 | 193.000000 | 37.000000 | 1507.000000 | 6910.000000 | 0.520000 | 194.000000 |
| 9 | 1996.000000 | 13380.000000 | 1113.000000 | 186.000000 | 23.000000 | 1322.000000 | 7078.000000 | 0.530000 | 150.000000 |

**This isn't right**.

The problem with Excel files is that people love to format it in ways that make it look nice in Excel but makes no sense in R.

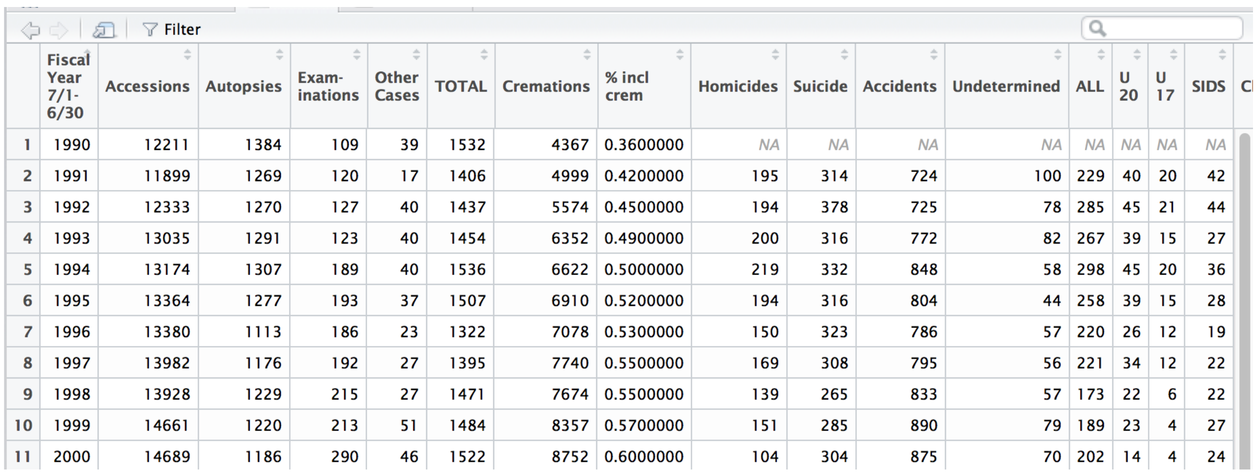
## read_excel() again

But this time we'll add `skip=2` so it skips the first rows when bringing in the data.

```
df_xl <- read_excel("data/StatisticsSummary.xls", sheet=1, skip=2)
```

Much better

```
View(df_xl)
```



| | Fiscal Year 7/1-6/30 | Accessions | Autopsies | Exam-inations | Other Cases | TOTAL | Cremations | % incl crem | Homicides | Suicide | Accidents | Undetermined | ALL | U 20 | U 17 | SIDS | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1990 | 12211 | 1384 | 109 | 39 | 1532 | 4367 | 0.3600000 | NA | NA | NA | NA | NA | NA | NA | NA | |
| 2 | 1991 | 11899 | 1269 | 120 | 17 | 1406 | 4999 | 0.4200000 | 195 | 314 | 724 | 100 | 229 | 40 | 20 | 42 | |
| 3 | 1992 | 12333 | 1270 | 127 | 40 | 1437 | 5574 | 0.4500000 | 194 | 378 | 725 | 78 | 285 | 45 | 21 | 44 | |
| 4 | 1993 | 13035 | 1291 | 123 | 40 | 1454 | 6352 | 0.4900000 | 200 | 316 | 772 | 82 | 267 | 39 | 15 | 27 | |
| 5 | 1994 | 13174 | 1307 | 189 | 40 | 1536 | 6622 | 0.5000000 | 219 | 332 | 848 | 58 | 298 | 45 | 20 | 36 | |
| 6 | 1995 | 13364 | 1277 | 193 | 37 | 1507 | 6910 | 0.5200000 | 194 | 316 | 804 | 44 | 258 | 39 | 15 | 28 | |
| 7 | 1996 | 13380 | 1113 | 186 | 23 | 1322 | 7078 | 0.5300000 | 150 | 323 | 786 | 57 | 220 | 26 | 12 | 19 | |
| 8 | 1997 | 13982 | 1176 | 192 | 27 | 1395 | 7740 | 0.5500000 | 169 | 308 | 795 | 56 | 221 | 34 | 12 | 22 | |
| 9 | 1998 | 13928 | 1229 | 215 | 27 | 1471 | 7674 | 0.5500000 | 139 | 265 | 833 | 57 | 173 | 22 | 6 | 22 | |
| 10 | 1999 | 14661 | 1220 | 213 | 51 | 1484 | 8357 | 0.5700000 | 151 | 285 | 890 | 79 | 189 | 23 | 4 | 27 | |
| 11 | 2000 | 14689 | 1186 | 290 | 46 | 1522 | 8752 | 0.6000000 | 104 | 304 | 875 | 70 | 202 | 14 | 4 | 24 | |

**Warning**: Notice that the column names are preserved with spaces and symbols.

```
# the colnames() function lists the column names of the dataframe
colnames(df_xl)
```

```
##  [1] "Fiscal Year      7/1-6/30" "Accessions"
##  [3] "Autopsies"                  "Exam-inations"
##  [5] "Other Cases"                "TOTAL"
##  [7] "Cremations"                 "% incl crem"
##  [9] "Homicides"                  "Suicide"
## [11] "Accidents"                  "Undetermined"
## [13] "ALL"                        "U 20"
## [15] "U 17"                       "SIDS"
## [17] "Clinicals"
```

So how would one refer to the data in the columns with spaces

If we did it like normal with the $ to pull the column we'd try

```
head(df_xl$Other Cases)
```

```
## Error: <text>:1:18: unexpected symbol
## 1: head(df_xl$Other Cases
##                      ^
```

See, spaces won't work. This is how to deal with columns with spaces– add the back tick next to the 1 button on your keyboard.

```
head(df_xl$`Other Cases`)
```

```
## [1] 39 17 40 40 40 37
```

It's some extra finger work that you might be okay with if it was in a limited basis.

However, in anticipation of the work we're going to be doing, we should go ahead and simplify the column names so there are no characters or spaces. Here's how

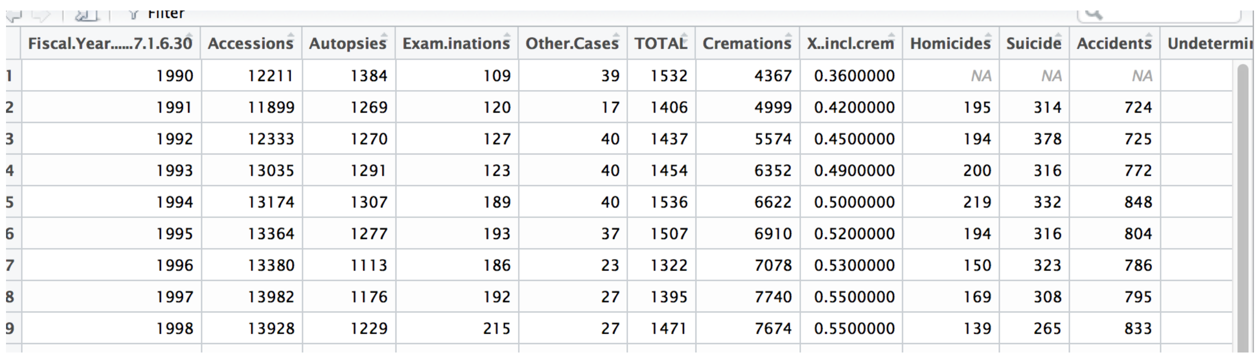
## Cleaning (part 1)

We'll use the `make.names()` function on the column names. This function makes syntactically valid names out of character vectors (as in in strips out the spaces and replaces them with periods)

```r
colnames(df_xl) <- make.names(colnames(df_xl))
```

Check it

```r
View(df_xl)
```



| | Fiscal.Year......7.1.6.30 | Accessions | Autopsies | Exam.inations | Other.Cases | TOTAL | Cremations | X..incl.crem | Homicides | Suicide | Accidents | Undetermi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1990 | 12211 | 1384 | 109 | 39 | 1532 | 4367 | 0.3600000 | NA | NA | NA | |
| 2 | 1991 | 11899 | 1269 | 120 | 17 | 1406 | 4999 | 0.4200000 | 195 | 314 | 724 | |
| 3 | 1992 | 12333 | 1270 | 127 | 40 | 1437 | 5574 | 0.4500000 | 194 | 378 | 725 | |
| 4 | 1993 | 13035 | 1291 | 123 | 40 | 1454 | 6352 | 0.4900000 | 200 | 316 | 772 | |
| 5 | 1994 | 13174 | 1307 | 189 | 40 | 1536 | 6622 | 0.5000000 | 219 | 332 | 848 | |
| 6 | 1995 | 13364 | 1277 | 193 | 37 | 1507 | 6910 | 0.5200000 | 194 | 316 | 804 | |
| 7 | 1996 | 13380 | 1113 | 186 | 23 | 1322 | 7078 | 0.5300000 | 150 | 323 | 786 | |
| 8 | 1997 | 13982 | 1176 | 192 | 27 | 1395 | 7740 | 0.5500000 | 169 | 308 | 795 | |
| 9 | 1998 | 13928 | 1229 | 215 | 27 | 1471 | 7674 | 0.5500000 | 139 | 265 | 833 | |

```r
colnames(df_xl)
```

```
##  [1] "Fiscal.Year......7.1.6.30" "Accessions"
##  [3] "Autopsies"                 "Exam.inations"
##  [5] "Other.Cases"               "TOTAL"
##  [7] "Cremations"                "X..incl.crem"
##  [9] "Homicides"                 "Suicide"
## [11] "Accidents"                 "Undetermined"
## [13] "ALL"                       "U.20"
## [15] "U.17"                      "SIDS"
## [17] "Clinicals"
```

Alright, that's a bit better.

Still, there's some oddness in the names but that's because enters were replaced with periods.

Check out the first column: `Fiscal.Year......7.1.6.30`

Let's change that so it's easier to type later on.

## Change the name of a single column

I'll show you how to do it in Base R and using the dplyr package

Copy `Fiscal.Year......7.1.6.30` and paste it into `colnames(dataframe_name)[colnames(dataframe_name) == 'ColumnNameToBeChanged'] <- 'NewColumnName'`

```r
# Don't run this, I just want to show you the process
colnames(df_xl)[colnames(df_xl) == 'Fiscal.Year......7.1.6.30'] <- 'Year'
```

Here's how to do it with dplyr: By using the `rename()` function.

```r
## If you don't have dplyr installed yet, uncomment the line below and run it
# install.packages("dplyr")

library(dplyr)
df_xl <- rename(df_xl, Year=Fiscal.Year......7.1.6.30)
```

It's slightly different– there are less parentheses and brackets and equal signs.

And you don't need to add quotation marks.

Check it

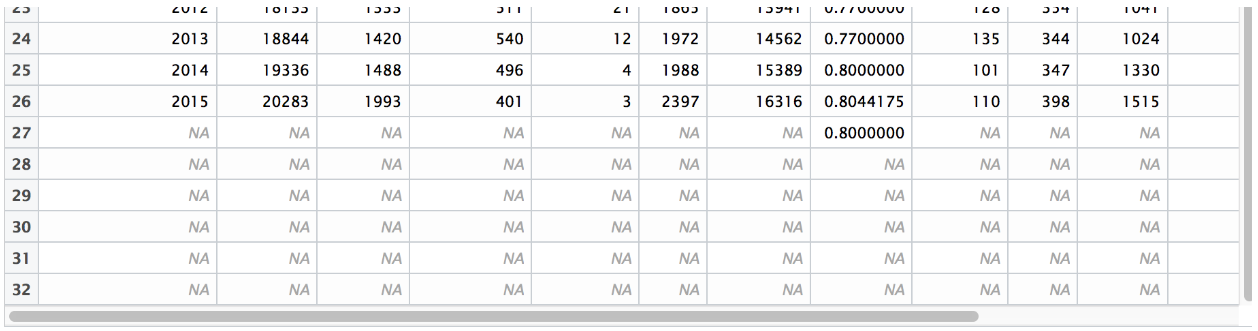
```r
colnames(df_xl)
```

```
##  [1] "Year"          "Accessions"    "Autopsies"     "Exam.inations"
##  [5] "Other.Cases"   "TOTAL"         "Cremations"    "X..incl.crem"
##  [9] "Homicides"     "Suicide"       "Accidents"     "Undetermined"
## [13] "ALL"           "U.20"          "U.17"          "SIDS"
## [17] "Clinicals"
```

Fix the other names if you want. I'm going to leave them as is for now.

## Is the df_xl sheet clean enough to work with?

Scroll down to the bottom of the data.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | 2013 | 18844 | 1420 | 540 | 12 | 1972 | 14562 | 0.7700000 | 135 | 344 | 1024 |
| 25 | 2014 | 19336 | 1488 | 496 | 4 | 1988 | 15389 | 0.8000000 | 101 | 347 | 1330 |
| 26 | 2015 | 20283 | 1993 | 401 | 3 | 2397 | 16316 | 0.8044175 | 110 | 398 | 1515 |
| 27 | NA | NA | NA | NA | NA | NA | NA | 0.8000000 | NA | NA | NA |
| 28 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 29 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 30 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 31 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 32 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |

Not clean yet. There are a bunch of `NA`s.

That might give us some issues later on so let's take care of it now.

## Eliminating NAs

Easiest way to get rid of NAs is to subset or filter out the `NA`s based on one column.

Let's use the `Year` column.

There are two ways: `subset()` or `filter()`

1. Base R

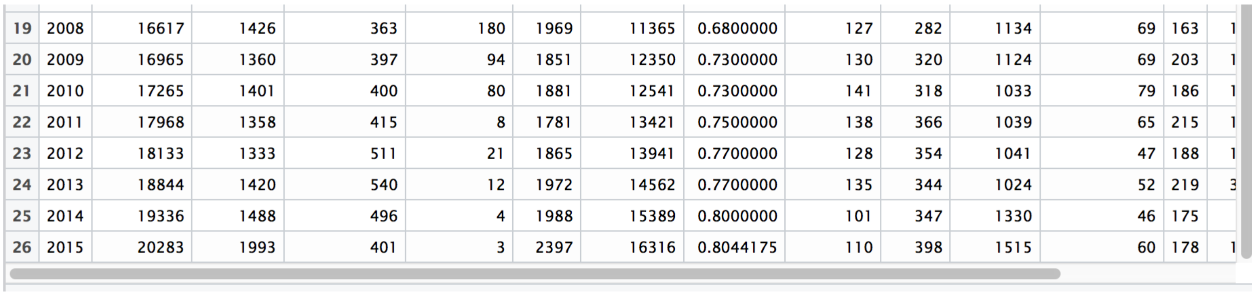
```r
df_xl <- subset(df_xl, !is.na(Year))
```

2. dplyr

```r
## If you don't have dplyr installed yet, uncomment the line below and run it
# install.packages("dplyr")
library(dplyr)
df_xl <- filter(df_xl, !is.na(Year))
```

What's the difference? dplyr is part of the tidyverse suite of packages that we'll be getting into later on in the course. Go ahead and use that.

### Check it



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 19 | 2008 | 16617 | 1426 | 363 | 180 | 1969 | 11365 | 0.6800000 | 127 | 282 | 1134 | 69 | 163 | 1 |
| 20 | 2009 | 16965 | 1360 | 397 | 94 | 1851 | 12350 | 0.7300000 | 130 | 320 | 1124 | 69 | 203 | 1 |
| 21 | 2010 | 17265 | 1401 | 400 | 80 | 1881 | 12541 | 0.7300000 | 141 | 318 | 1033 | 79 | 186 | 1 |
| 22 | 2011 | 17968 | 1358 | 415 | 8 | 1781 | 13421 | 0.7500000 | 138 | 366 | 1039 | 65 | 215 | 1 |
| 23 | 2012 | 18133 | 1333 | 511 | 21 | 1865 | 13941 | 0.7700000 | 128 | 354 | 1041 | 47 | 188 | 1 |
| 24 | 2013 | 18844 | 1420 | 540 | 12 | 1972 | 14562 | 0.7700000 | 135 | 344 | 1024 | 52 | 219 | 3 |
| 25 | 2014 | 19336 | 1488 | 496 | 4 | 1988 | 15389 | 0.8000000 | 101 | 347 | 1330 | 46 | 175 | |
| 26 | 2015 | 20283 | 1993 | 401 | 3 | 2397 | 16316 | 0.8044175 | 110 | 398 | 1515 | 60 | 178 | 1 |

No `NAs` at the bottom.

It took a few lines of code but the data has been cleaned up enough to analyze or visualize with.

## Exporting to Excel

It's preferable to save data frames as CSVs because it's more open and doesn't require a paid program for others to open.

But if you must, there are some decent walkthroughs:

- Using the xlsx package
- Reading and importing Excel files into R