Data Cleaning

Data Wrangling in R

Dealing with Missing Data

Missing data types

One of the most important aspects of data cleaning is missing values.

Types of “missing” data:

- `NA` - general missing data
- `NaN` - stands for “**N**ot **a N**umber”, happens when you do 0/0.
- `Inf` and `-Inf` - Infinity, happens when you take a positive number (or negative number) by 0.

Finding Missing data

Each missing data type has a function that returns `TRUE` if the data is missing:

- `NA` - `is.na`
- `NaN` - `is.nan`
- `Inf` and `-Inf` - `is.infinite`
- `is.finite` returns `FALSE` for all missing data and `TRUE` for non-missing

Missing Data with Logicals

One important aspect (esp with subsetting) is that logical operations return `NA` for `NA` values. Think about it, the data could be `> 2` or not we don't know, so `R` says there is no `TRUE` or `FALSE`, so that is missing:

```
x = c(0, NA, 2, 3, 4)
x > 2
```

```
[1] FALSE    NA FALSE  TRUE  TRUE
```

Missing Data with Logicals

What to do? What if we want if `x > 2` and `x` isn't `NA`?
Don't do `x != NA`, do `x > 2` and `x` is NOT `NA`:

```
x != NA
```

```
[1] NA NA NA NA NA
```

```
x > 2 & !is.na(x)
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE
```

Missing Data with Logicals

What about seeing if a value is equal to multiple values? You can do `(x == 1 | x == 2) & !is.na(x)`, but that is not efficient.

```
(x == 0 | x == 2) # has NA
```

```
[1]  TRUE    NA  TRUE FALSE FALSE
```

```
(x == 0 | x == 2) & !is.na(x) # No NA
```

```
[1]  TRUE FALSE  TRUE FALSE FALSE
```

what to do?

Missing Data with Logicals: `%in%`

Introduce the `%in%` operator:

```
x %in% c(0, 2) # NEVER has NA and returns logical
```

```
[1]  TRUE FALSE  TRUE FALSE FALSE
```

reads “return `TRUE` if `x` is in 0 or 2”. (Like `inlist` in Stata).

Missing Data with Logicals: `%in%`

NEVER has NA, even if you put it there (BUT DON'T DO THIS):

```
x %in% c(0, 2, NA) # NEVER has NA and returns logical
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

```
x %in% c(0, 2) | is.na(x)
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE
```

Filtering and tibbles

Filter removes missing values, have to keep them if you want them:

```
df = tibble(x = x)
df %>% filter(x > 2)
```

```
# A tibble: 2 x 1
      x
  <dbl>
1     3
2     4
```

```
filter(df, between(x, -1, 3) | is.na(x))
```

```
# A tibble: 4 x 1
      x
  <dbl>
1     0
2    NA
3     2
4     3
```

Missing Data with Operations

Similarly with logicals, operations/arithmetic with `NA` will result in `NA`s:

```
x + 2
```

```
[1]  2 NA  4  5  6
```

```
x * 2
```

```
[1]  0 NA  4  6  8
```

Recoding to missing

Sometimes people code missing data in weird or inconsistent ways.

```
ages = c(23,21,44,32,57,65,-999,54)
range(ages)
```

```
[1] -999   65
```

Recoding to missing

How do we change the `-999` to be treated as missing?

```
ages[ages == -999] = NA
range(ages)
```

```
[1] NA NA
```

```
range(ages,na.rm=TRUE)
```

```
[1] 21 65
```

Recoding from missing

What if you were the person that coded the `-999`

```
is.na(ages)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

```
ages[is.na(ages)] = -999
ages
```

```
[1]   23   21   44   32   57   65 -999   54
```