R: Intro to Cleaning Data - Nick Carchedi
Note as of July 2019
Note Taker: Paris Zhang

Chapters:

1. Exploring raw data
2. Tidying data
3. Preparing data for analysis
4. Putting it all together

Ch 1 - Exploring raw data

- Understanding the structure of data
  e.g., `str()`, `summary()`, **`dplyr`** package -> `glimpse()`
- Looking at data
  e.g., `head()`, `tail()`, `print()`
- Visualizing data
  e.g., `hist()`, `plot()`

Ch 2 - Intro to tidy data

(In 2014, Hadley Wickham, writer of **`tidyr`**, published "Tidy Data" on Journal of Statistical Software.)

- Principles of tidy data
  - Observations as rows
  - Variables as columns
  - One type of observational unit per table
- Key **`tidyr`** functions
  - `gather()` - Gather columns into key-value pairs
  - `spread()` - Spread key-value pairs into columns
  - `separate()` - Separate one column into multiple
  - `unite()` - Unite multiple columns into one
- `Gather` columns into key-value pairs
  `gather(wide_df, my_key, my_val, -col)`

```
# Look at wide_df
> wide_df
  col A B C
1   X 1 2 3
2   Y 4 5 6

# Gather the columns of wide_df
> gather(wide_df, my_key, my_val, -col)
  col my_key my_val
1   X      A      1
2   Y      A      4
3   X      B      2
4   Y      B      5
5   X      C      3
6   Y      C      6
```

gather(data, key, value, …)

data: a data frame

key: bare name of new key column

value: bare name of new value column

…: bare names of columns to gather (or not)

- `Spread` key-value pairs into clumns
  `spread(long_df, my_key, my_val)`

```
# Look at long_df
> long_df
  col my_key my_val
1   X      A      1
2   Y      A      4
3   X      B      2
4   Y      B      5
5   X      C      3
6   Y      C      6

# Spread the key-value pairs of long_df
> spread(long_df, my_key, my_val)
  col A B C
1   X 1 2 3
2   Y 4 5 6
```

spread(data, key, value)

data: a data frame

key: bare name of column containing keys

value: bare name of column containing values

- `Separate` columns

`separate(treatments, year_mo, c("year", "month"))`

```
# View the treatments data
> treatments
  patient treatment year_mo response
1       X         A 2010-10        1
2       Y         A 2010-10        4
3       X         B 2012-08        2
4       Y         B 2012-08        5
5       X         C 2014-12        3
6       Y         C 2014-12        6

# Separate year_mo into two columns
> separate(treatments, year_mo, c("year", "month"))
  patient treatment year month response
1       X         A 2010    10        1
2       Y         A 2010    10        4
3       X         B 2012    08        2
4       Y         B 2012    08        5
5       X         C 2014    12        3
6       Y         C 2014    12        6
```

separate(data, col, into)

data: a data frame sep = "-"

col: bare name of column to separate

into: character vector of new column names

- `Unite` columns

`unite(treatments, year_mo, year, month)`

```
# View treatments data
> treatments
  patient treatment year month response
1       X         A 2010    10        1
2       Y         A 2010    10        4
3       X         B 2012    08        2
4       Y         B 2012    08        5
5       X         C 2014    12        3
6       Y         C 2014    12        6

# Unite year and month to form year_mo column
> unite(treatments, year_mo, year, month)
  patient treatment year_mo response
1       X         A 2010_10        1
2       Y         A 2010_10        4
3       X         B 2012_08        2
4       Y         B 2012_08        5
5       X         C 2014_12        3
6       Y         C 2014_12        6
```

unite(data, col, …)

data: a data frame sep = "-"

col: bare name of new column

…: bare names of columns to unite

- Common symptoms of messy data
  - Column headers are values, not variable names

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

- Variables are stored in both rows and columns

name	measurement	value
Jake	n_dogs	1
Jake	n_cats	0
Jake	n_birds	1
Alice	n_dogs	1
Alice	n_cats	2
Alice	n_birds	0

name	n_dogs	n_cats	n_birds
Jake	1	0	1
Alice	1	2	0

- Multiple variables are stored in one column

name	sex_age	eye_color	height
Jake	M.34	Other	6'1"
Alice	F.55	Blue	5'9"
Tim	M.76	Brown	5'7"
Denise	F.19	Other	5'1"

name	sex	age	eye_color	height
Jake	M	34	Other	6'1"
Alice	F	55	Blue	5'9"
Tim	M	76	Brown	5'7"
Denise	F	19	Other	5'1"

- A single observational unit is stored in multiple tables
- Multiple types of observational units are stored in the same table

name	age	height	pet_name	pet_type	pet_height
Jake	34	6'1"	Larry	Dog	25"
Jake	34	6'1"	Chirp	Bird	3"
Alice	55	5'9"	Wally	Dog	30"
Alice	55	5'9"	Sugar	Cat	10"
Alice	55	5'9"	Spice	Cat	12"

Alice's name, age, and height are 3x duplicated

Ch 3 - Preparing data for analysis

- Type conversions (coerce)
  - Types of variables in R: character, numeric, integer, factor, logical
  - Overview of **`lubridate`**
    - Written by Garrett Grolemund & Hadley Wickham

- Coerce strings to dates

`> ymd("2015-08-25")`	`[1] "2015-08-25 UTC"`
`> ymd("2015 Auguest 25")`	
`> mdy("August 25, 2015")`	
`> hms("13:33:09")`	`[1] "13H 33M 9S"`
`> ymd_hms("2015/05/25 13.33.09")`	`[1] "2015-08-25 13:33:09 UTC"`

- String manipulation
  - Overview of **`stringr`**
    - Consistent interface for working with strings
    - Written by Hadley Wickham
    - Covers all common operations
  - Key functions in **`stringr`** for cleaning data
    - `str_trim()` - Trim leading and trailing white space
    - `str_pad()` - Pad with additional characters
    - `str_detect()` - Detect a pattern
    - `str_replace()` - Find and replace a pattern

```
# Trim leading and trailing white space
> str_trim("  this is a test     ")
[1] "this is a test"    white space removed

# Pad string with zeros
> str_pad("24493", width = 7, side = "left", pad = "0")
[1] "0024493"  7 digits

# Create character vector of names
> friends <- c("Sarah", "Tom", "Alice")

# Search for string in vector
> str_detect(friends, "Alice")
[1] FALSE FALSE  TRUE

# Replace string in vector
> str_replace(friends, "Alice", "David")
[1] "Sarah" "Tom"   "David"
```

  - More functions
    - `toupper()`
    - `tolower()`
- Missing and special values
  - Missing values
    `NA` - not available ("." in SPSS/SAS; #N/A in Excel)
  - Special values
    `Inf` - infinite value
    `NaN` - not a number
  - Finding missing values

        - `is.na(df)`
        - `any(is.na(df))` # Are there any NAs?
        - `sum(is.na(df))` # Count number of NAs
        - `summary()`
    - Dealing with missing values
        - `complete.cases(df)` # Find rows w/ no missing values
        - `df[complete.cases(df), ]` # Subset to keep only complete cases
        - `na.omit(df)` # Remove rows w/ NAs
- Outliers and obvious errors
    - Finding outliers and errors
        - `summary()`
        - `hist(df$var, breaks = 20)`
        - `boxplot(df$var)`

Ch 4 - Sum up and practice

1. Understanding the structure of data
    - `class(), dim(), str(), summary(), names()`
    - `glimpse()` - a better version of `str()` from **dplyr**
2. Looking at data
    - `head(), tail()`
3. Visualizing data
    - `hist(), plot()`
4. Tidying data
    - Common problems:
        - Column names are values
        - Values are variable names
5. Preparing data for analysis
    - Dates with **lubridate**
    - Type conversions
6. Missing, extreme, and unexpected values
    - Finding missing values:
        - `is.na()`
        - `sum(is.na())`
        - `which(is.na())`
    - Identifying errors
7. Example
   Run the call to mutate_at as-is to conveniently apply as.numeric() to all columns from CloudCover through WindDirDegrees (reading left to right in the data)

```
weather6 <- mutate_at(weather5, vars(CloudCover:WindDirDegrees),
funs(as.numeric))
```