Ve572 Lecture 5

Manuel and Jing

UM-SJTU Joint Institute

May 31, 2018

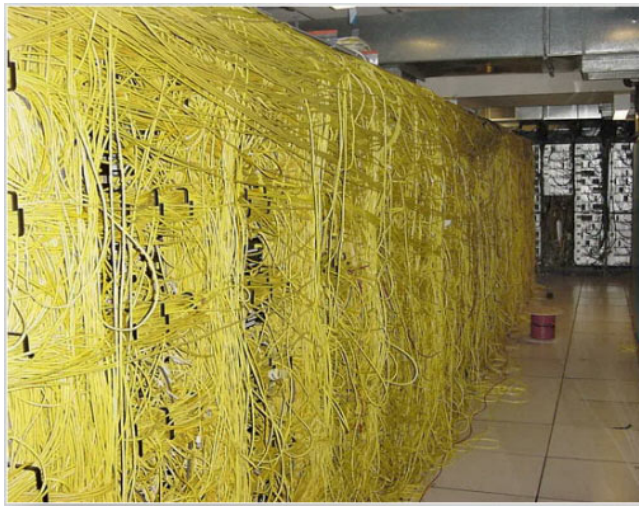
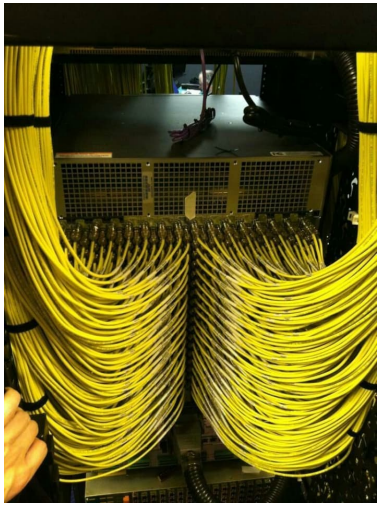
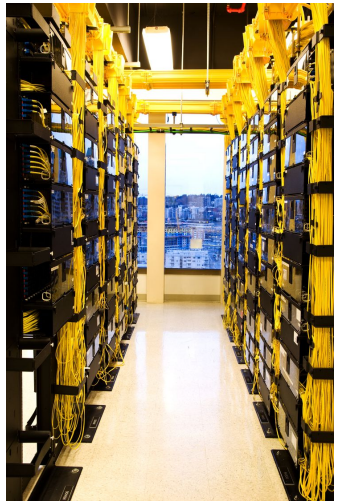
Q: There is a task that data scientists spend 50-80% of their time on, any idea?

data wrangling, aka data munging.

- It is an old but a key hurdle to insights, however, it is the least enjoyable task

according to articles in NY Times and Forbes

Q: What is data wrangling or data munging?

- So it refers to the process of cleaning or transforming the data set.

- We will start data wrangling in R using `library(tidyr)`.

```
> hp.df =
+   read.table("~/Desktop/hp.csv",
+              header = TRUE, sep = ",",
+              quote = "\"", na.strings = "?",
+              stringsAsFactors = FALSE)

> summary(hp.df)
```

```
      mpg              hp              year          name
 Min.   : 9.00   Min.   : 46.0   73     : 40   Length:397
 1st Qu.:17.50   1st Qu.: 75.0   78     : 36   Class :character
 Median :23.00   Median : 93.5   76     : 34   Mode  :character
 Mean   :23.52   Mean   :104.5   75     : 30
 3rd Qu.:29.00   3rd Qu.:126.0   82     : 30
 Max.   :46.60   Max.   :230.0   70     : 29
                 NA's   :5       (Other):198
```

```
> sapply(hp.df[,c(3,4)],function(x)length(unique(x)))
```

```
year name
  13  304
```

```
> sapply(hp.df, class)
```

```
       mpg          hp        year        name
 "numeric"   "integer"   "integer" "character"
```

```
> hp.df$year = factor(hp.df$year, order = TRUE)
```

```
> head(hp.df)
```

```
  mpg  hp year                       name
1  18 130   70 chevrolet chevelle malibu
2  15 165   70         buick skylark 320
3  18 150   70        plymouth satellite
4  16 150   70             amc rebel sst
5  17 140   70               ford torino
6  15 198   70          ford galaxie 500
```

The library(tidyr) works with a data type known as the tibble

```
> hp_tb = # tb is a df with additional features
+    tibble::as.tibble(hp.df)

> hp_tb
```

```
# A tibble: 397 x 4
     mpg     hp year  name
   <dbl> <int> <ord> <chr>
 1    18.   130 70    chevrolet chevelle malibu
 2    15.   165 70    buick skylark 320
 3    18.   150 70    plymouth satellite
 4    16.   150 70    amc rebel sst
 5    17.   140 70    ford torino
 6    15.   198 70    ford galaxie 500
 7    14.   220 70    chevrolet impala
 8    14.   215 70    plymouth fury iii
 9    14.   225 70    pontiac catalina
10    15.   190 70    amc ambassador dpl
# ... with 387 more rows
```

- For us, there are two major differences between data frame and tibbles

1. Printing
2. Subsetting

```
> class(hp.df[, 1])
[1] "numeric"
>
> class(hp.df[, 1, drop = FALSE])
[1] "data.frame"
```

- Tibbles clearly delineate `[` and `[[`.

```
> class(hp_tb[,1])
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

```
> class(hp_tb[[1]])
```

```
[1] "numeric"
```

```
# With tibbles, [ always returns another tibble
> hp_tb[, 1]
```

```
# A tibble: 397 x 1
     mpg
   <dbl>
 1    18.
 2    15.
 3    18.
 4    16.
 5    17.
 6    15.
 7    14.
 8    14.
 9    14.
10    15.
# ... with 387 more rows
```

```
> # For a single element, you should always use [[
> head(hp_tb[[1]], 6)
```

```
[1] 18 15 18 16 17 15
```

```
> class(as.data.frame(hp_tb)) # converting back
```

```
[1] "data.frame"
```

- Notice this dataset, like others you have seen, has been wrangled.
- Very often rectangular tables are preferred for most of analysis where
  1. Each variable forms a column
  2. Each observation forms a row
  3. Each type of observational unit forms a table
- Additionally, missing/possibly incorrect values are why we need to wrangle.
- Consider the following dataset from Billboard.

artist	track	time	entry date	wk1	wk2	wk3
Adele	Hello	4:55	2015-11-14	1	1	1
Justine Bieber	Sorry	3:20	2015-11-14	2	4	3
Justine Bieber	Love Yourself	3:53	2015-12-05	4	6	7

- Notice there are columns as well as rows that are not displayed here:

wk4—wk75

- In light of 1. and 2., one way to reshape the data is the following structure

Year	Artist	Track	Time	Date	Week	Rank
2016	Adele	Hello	4:55	2015-11-14	1	1
2016	Adele	Hello	4:55	2015-11-21	2	1
2016	Adele	Hello	4:55	2015-11-28	3	1
2016	Adele	Hello	4:55	2015-12-05	4	1
2016	Adele	Hello	4:55	2015-12-12	5	1
2016	Adele	Hello	4:55	2015-12-19	6	1
2016	Adele	Hello	4:55	2015-12-26	7	1
2016	Adele	Hello	4:55	2016-01-02	8	1
2016	Adele	Hello	4:55	2016-01-09	9	1
2016	Adele	Hello	4:55	2016-01-16	10	1
		⋮				
2016	Justine Bieber	Sorry	3:20	2015-11-14	1	2
2016	Justine Bieber	Sorry	3:20	2015-11-21	2	4
2016	Justine Bieber	Sorry	3:20	2015-11-28	3	3
2016	Justine Bieber	Sorry	3:20	2015-12-05	4	2
2016	Justine Bieber	Sorry	3:20	2015-12-12	5	2

- Now in the spirit of 3., one way is to split the dataset in the following way

id	artist	track	time	peak	total
1	Justin Bieber	Love Yourself	3:53	1	41
2	Justin Bieber	Sorry	3:20	1	42
3	Drake	One Dance	2:54	1	36
4	Rihanna	Work	3:39	1	26
5	Twenty One Pilots	Stressed Out	3:22	2	50
6	Desiigner	Panda	4:06	1	40
7	Adele	Hello	3:53	1	26
8	Chainsmokers	Don't Let Me Down	3:28	3	51
9	Justin Timberlake	Can't Stop The Feeling	3:56	1	52

date	rank1	rank2	rank3	rank4	rank5	rank6	rank7
2015-11-14	7	2	24	32	31	23	79
2015-12-05	7	24	32	2	31	23	101

- Note there are additional columns (rank8–rank100) in the second dataset.
- In this way, we have only one observational unit for each dataset track/time.

- Now consider the data collected and used by World Health Organization on

  Global Tuberculosis

- It comes along with tidyr package as a tb in its original form, i.e. raw data

```
> tidyr::who
```

```
# A tibble: 7,240 x 60
   country     iso2  iso3   year new_sp_m014 new_sp_m1524
   <chr>       <chr> <chr> <int>       <int>        <int>
 1 Afghanistan AF    AFG    1980          NA           NA
 2 Afghanistan AF    AFG    1981          NA           NA
 3 Afghanistan AF    AFG    1982          NA           NA
 4 Afghanistan AF    AFG    1983          NA           NA
 5 Afghanistan AF    AFG    1984          NA           NA
 6 Afghanistan AF    AFG    1985          NA           NA
 7 Afghanistan AF    AFG    1986          NA           NA
 8 Afghanistan AF    AFG    1987          NA           NA
 9 Afghanistan AF    AFG    1988          NA           NA
10 Afghanistan AF    AFG    1989          NA           NA
# ... with 7,230 more rows, and 54 more variables:
```

The output continues on the next page...

...output continues

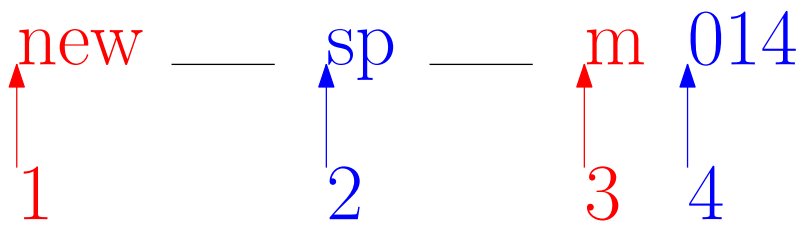
```
# ... with 7,230 more rows, and 54 more variables:
#   new_sp_m2534 <int>, new_sp_m3544 <int>, new_sp_m4554 <int>,
#   new_sp_m5564 <int>, new_sp_m65 <int>, new_sp_f014 <int>,
#   new_sp_f1524 <int>, new_sp_f2534 <int>, new_sp_f3544 <int>,
#   new_sp_f4554 <int>, new_sp_f5564 <int>, new_sp_f65 <int>,
#   new_sn_m014 <int>, new_sn_m1524 <int>, new_sn_m2534 <int>,
#   new_sn_m3544 <int>, new_sn_m4554 <int>, new_sn_m5564 <int>,
#   new_sn_m65 <int>, new_sn_f014 <int>, new_sn_f1524 <int>,
#   new_sn_f2534 <int>, new_sn_f3544 <int>, new_sn_f4554 <int>,
#   new_sn_f5564 <int>, new_sn_f65 <int>, new_ep_m014 <int>,
#   new_ep_m1524 <int>, new_ep_m2534 <int>, new_ep_m3544 <int>,
#   new_ep_m4554 <int>, new_ep_m5564 <int>, new_ep_m65 <int>,
#   new_ep_f014 <int>, new_ep_f1524 <int>, new_ep_f2534 <int>,
#   new_ep_f3544 <int>, new_ep_f4554 <int>, new_ep_f5564 <int>,
#   new_ep_f65 <int>, newrel_m014 <int>, newrel_m1524 <int>,
#   newrel_m2534 <int>, newrel_m3544 <int>, newrel_m4554 <int>,
#   newrel_m5564 <int>, newrel_m65 <int>, newrel_f014 <int>,
#   newrel_f1524 <int>, newrel_f2534 <int>, newrel_f3544 <int>,
#   newrel_f4554 <int>, newrel_f5564 <int>, newrel_f65 <int>
```

- `> summary(tidyr::who$new_sp_m014)`

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00    0.00    5.00   83.71   37.00 5001.00    4067
```

it is nature to suspect those mysterious columns store counts for TB cases.

- According to WHO, the following naming rule was used



1. Denote whether those are new or old TB cases.
2. Denote the type of those TB cases.
3. Denote the patient gender of those TB cases.
4. Denote the age group of those TB cases.

$$
\begin{aligned}
014 &= \ \ 0\text{–}14 \text{ years old} \\
1524 &= 15\text{–}24 \text{ years old} \\
2534 &= 25\text{–}34 \text{ years old} \\
3544 &= 35\text{–}44 \text{ years old} \\
4554 &= 45\text{–}54 \text{ years old} \\
5564 &= 55\text{–}64 \text{ years old} \\
65 &= \ \ \geq 65 \text{ years old}
\end{aligned}
$$

```
> col2row_tb =
+   gather(new_sp_m014:newrel_f65,
+          data = who,          # Dataset
+          key = "tmp",         # Name for column names
+          value = "counts",# Name for column values
+          na.rm = TRUE)     # remove NA values
> col2row_tb
```

```
# A tibble: 76,046 x 6
  country      iso2  iso3   year tmp           counts
* <chr>        <chr> <chr> <int> <chr>          <int>
1 Afghanistan AF    AFG    1997 new_sp_m014        0
2 Afghanistan AF    AFG    1998 new_sp_m014       30
3 Afghanistan AF    AFG    1999 new_sp_m014        8
# ... with 7.604e+04 more rows
```

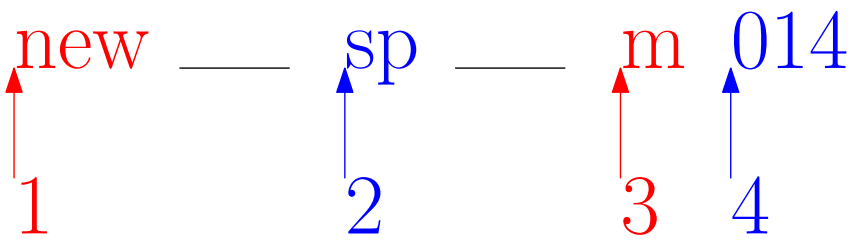
```
> sum(col2row_tb$tmp == "new_sp_m014") ==
+   sum(!is.na(who$new_sp_m014))
```

```
[1] TRUE
```

- We can study the tmp column by counting according to tmp.

```
> library(dplyr)
> # Define the grouping variable
> by_tmp = group_by(col2row_tb, tmp)
> # Specify count to be in the summary
> # Define column name freq to be used for count
> summarise(by_tmp, freq = n())
```

```
# A tibble: 56 x 2
   tmp            freq
   <chr>         <int>
1 new_ep_f014    1032
2 new_ep_f1524   1021
3 new_ep_f2534   1021
4 new_ep_f3544   1021
5 new_ep_f4554   1017
# ... with 51 more rows
```



- We need to split `tmp` into four variables:
  - newold
  - type
  - gender
  - age
- There is a small complication, which is difficult to see until an error pops-up.

```
> sep_tb =
+   separate(col2row_tb, # Dataset
+            col = tmp,  # Column
+            into = c("new", "type", "sexage"),
+            sep = "_")  # delimiter
```

```
# Warning message:
# Expected 3 pieces. Missing pieces filled with 'NA'
```

```
> all(stringr::str_sub(
+   col2row_tb$tmp, 1, 4) == "new_")
```

```
[1] FALSE
```

```
> subset(col2row_tb, !grepl("new_.", tmp))
```

```
# A tibble: 2,580 x 6
   country      iso2  iso3   year tmp          counts
   <chr>        <chr> <chr> <int> <chr>         <int>
1  Afghanistan  AF    AFG    2013 newrel_m014    1705
2  Albania      AL    ALB    2013 newrel_m014      14
3  Algeria      DZ    DZA    2013 newrel_m014      25
# ... with 2,577 more rows
```

```
> any(!grepl("new_.", col2row_tb$tmp)
+     & !grepl("newr.", col2row_tb$tmp))
```

```
[1] FALSE
```

```
> mut_tb =                    # Fix the inconsistency
+   mutate(col2row_tb,
+          tmp = stringr::str_replace(
+            tmp, "newrel", "new_rel"))
>
> all(stringr::str_sub(mut_tb$tmp, 1, 4) == "new_")
```

```
[1] TRUE
```

```
> (sep_tb =
+   separate(mut_tb, col = tmp, sep = "_",
+            into = c("new", "type", "sexage")))
```

```
# A tibble: 76,046 x 8
   country      iso2  iso3   year new   type  sexage
   <chr>        <chr> <chr> <int> <chr> <chr> <chr>
1 Afghanistan AF    AFG    1997 new   sp    m014
2 Afghanistan AF    AFG    1998 new   sp    m014
3 Afghanistan AF    AFG    1999 new   sp    m014
# ... with 7.604e+04 more rows, and 1 more
#   variable: counts <int>
```

- Notice all reported cases in the dataset is new, so we drop this constant

```
> by_new = group_by(sep_tb, new)
> summarise(by_new, freq = n())
```

```
# A tibble: 1 x 2
  new     freq
  <chr> <int>
1 new   76046
```

- Two reductant country code columns will also be dropped

```
> (sel_tb = select(sep_tb, -new, -iso2, -iso3))
```

```
# A tibble: 76,046 x 5
  country      year type  sexage counts
  <chr>       <int> <chr> <chr>   <int>
1 Afghanistan  1997 sp    m014        0
2 Afghanistan  1998 sp    m014       30
3 Afghanistan  1999 sp    m014        8
# ... with 7.604e+04 more rows
```

```
> (who_tidy_tb = # split after 1 ch at the far-left
+    separate(sel_tb, col = sexage, sep = 1,
+             into = c("gender", "age")))
```

```
# A tibble: 76,046 x 6
   country      year type  gender age   counts
   <chr>       <int> <chr> <chr>  <chr>  <int>
 1 Afghanistan  1997 sp    m      014        0
 2 Afghanistan  1998 sp    m      014       30
 3 Afghanistan  1999 sp    m      014        8
 4 Afghanistan  2000 sp    m      014       52
 5 Afghanistan  2001 sp    m      014      129
 6 Afghanistan  2002 sp    m      014       90
 7 Afghanistan  2003 sp    m      014      127
 8 Afghanistan  2004 sp    m      014      139
 9 Afghanistan  2005 sp    m      014      151
10 Afghanistan  2006 sp    m      014      193
# ... with 7.604e+04 more rows
```

- We have done the cleaning a piece at time, but `tidyr` has a compact syntax

```
> who_tidy_tb = who %>%
+   gather(
+     new_sp_m014:newrel_f65, key = "tmp",
+     value = "counts",na.rm = TRUE
+           ) %>%
+   mutate(
+     tmp = stringr::str_replace(
+       tmp, "newrel", "new_rel")
+     ) %>%
+   separate(
+     col = tmp, sep = "_",
+     into = c("new", "type", "sexage")
+     ) %>%
+   select(-new, -iso2, -iso3) %>%
+   separate(col = sexage,
+            into = c("gender", "age"),
+            sep = 1)
```

which will get everything done in one go.