Lecture 7

Brad McNeney

2018-03-01

Tidying data with the tidyverse

```
library(tidyverse) # loads dplyr, ggplot2, tidyr, etc
```

- Reading: Hadley Wickham's article on tidy data, available from GitHub (Notes folder) or canvas.
  - Today's notes closely follow the `tidyr` vignette.
- "Tidy" data is ready for analysis, with one row for each sampled unit and columns for the variables measured on the units.
  - Often classify variables as "explanatory" or "response"
- Tabular data and repeated measures data are often not in tidy form.
- Examples:
  - Tabular data on new tuberculosis cases from WHO
  - Repeated measures from Billboard Top 100

Tuberculosis (TB) cases

```
tb <- read.csv("tb.csv",stringsAsFactors=FALSE)
dim(tb)
```

```
## [1] 5769   23
```

```
tb[1:4,1:6]
```

```
##   iso2 year new_sp new_sp_m04 new_sp_m514 new_sp_m014
## 1   AD 1989     NA          NA           NA          NA
## 2   AD 1990     NA          NA           NA          NA
## 3   AD 1991     NA          NA           NA          NA
## 4   AD 1992     NA          NA           NA          NA
```

```
names(tb)[1:20]
```

```
##  [1] "iso2"          "year"          "new_sp"        "new_sp_m04"
##  [5] "new_sp_m514"   "new_sp_m014"   "new_sp_m1524" "new_sp_m2534"
##  [9] "new_sp_m3544" "new_sp_m4554" "new_sp_m5564" "new_sp_m65"
## [13] "new_sp_mu"     "new_sp_f04"    "new_sp_f514"   "new_sp_f014"
## [17] "new_sp_f1524" "new_sp_f2534" "new_sp_f3544" "new_sp_f4554"
```

Structure of TB table

- First column is 2-letter country code, second is year, third is number of new cases for that country/year.
- Then come TB counts for different gender/age categories.
  - `new_sp` is "new cases by positive pulmonary smear assay"
  - gender is `m` or `f`
  - two special age categories 0-4, 5-14,
  - age categories 0-14, 15-24, 25-34, 35-44, 45-54, 55-65, 65+, unknown (`u`)
- Gender/age columns are not variables, they are data on the observed units.
- Tidy data would have one row for each country, year, gender and age category, with a column of counts

Billboard Top 100 rankings of songs

```
bb <- read.csv("billboard.csv",stringsAsFactors = FALSE)
dim(bb)
```

```
## [1] 317  83
```

```
bb[1:3,1:6]
```

```
##   year artist.inverted                    track time genre date.entered
## 1 2000 Destiny's Child Independent Women Part I 3:38  Rock   2000-09-23
## 2 2000         Santana             Maria, Maria 4:18  Rock   2000-02-12
## 3 2000   Savage Garden       I Knew I Loved You 4:07  Rock   1999-10-23
```

```
names(bb)[c(1:10,ncol(bb))]
```

```
##  [1] "year"            "artist.inverted" "track"
##  [4] "time"            "genre"           "date.entered"
##  [7] "date.peaked"     "x1st.week"       "x2nd.week"
## [10] "x3rd.week"       "x76th.week"
```

Structure of the Billboard table

- Columns `year` through `date.peaked` describe the song, then `x1st.week` through `x76th.week` are the chart positions for the first through 76th weeks.
  - If a song is on the chart for less than 76 weeks, its position is `NA` for any missing weeks.
- Weeks are not variables, they are the time data for the time series.

Tidying the Billboard data

- Main step is to consolidate, or "gather" the rankings in the different weeks into a `rank` variable.
- Before gathering, will select/rename some of the variables.
- After gathering, will create some new variables and sort the data frame.

Select and rename

- Won't need `time` or `genre`.
  - `select()` from `dplyr` can use `-` to de-select
- Rename `artist.inverted`
  - `rename()` from `dplyr` takes arguments of the form `newname = oldname`

```
bb <-
  bb %>% select(-time,-genre) %>%
  rename(artist = artist.inverted)
```

Gather the weeks into a "long" version of the Billboard data

- Leave each song info variable as-is.
- The data, or "values", are the chart positions.
- The weeks are descriptors or "keys" for these values.
- We want to create key-value pairs for each observation.
  - There will be missing values, which we can remove.
- The `gather()` function from `tidyr` gathers specified columns into keys (e.g., week) and values (e.g., rank).

gather() for the Billboard data

```
# gather (data, key, value, ... ) where ... are the columns to collapse
bblong <- gather(bb,week,rank,x1st.week:x76th.week,na.rm=TRUE)
head(bblong,n=4)
```

```
##   year          artist                    track date.entered date.peaked
## 1 2000 Destiny's Child Independent Women Part I   2000-09-23  2000-11-18
## 2 2000         Santana             Maria, Maria   2000-02-12  2000-04-08
## 3 2000   Savage Garden       I Knew I Loved You   1999-10-23  2000-01-29
## 4 2000         Madonna                    Music   2000-08-12  2000-09-16
##        week rank
## 1 x1st.week   78
## 2 x1st.week   15
## 3 x1st.week   71
## 4 x1st.week   41
```

More cleaning suggested in the vignette

- Extract week numbers from `week` variable
- Coerce `date.entered` to a `Date` object
- Calculate the date of each ranking based on the date it entered the charts and the week.
- Sort ("arrange") on artist, track and week.

```
bblong %>% mutate(week = parse_number(week),
                  date = as.Date(date.entered) + 7*(week-1)) %>%
  select(-date.entered) %>% # don't need date.entered anymore
  arrange(artist,track,week) -> bb
head(bb,n=3)
```

```
##   year artist                                  track date.peaked week rank
## 1 2000  2 Pac Baby Don't Cry (Keep Ya Head Up II)  2000-03-11    1   87
## 2 2000  2 Pac Baby Don't Cry (Keep Ya Head Up II)  2000-03-11    2   82
## 3 2000  2 Pac Baby Don't Cry (Keep Ya Head Up II)  2000-03-11    3   72
##         date
## 1 2000-02-26
## 2 2000-03-04
## 3 2000-03-11
```

Tidying the TB data

- Recall structure of the data: country, year, count of new cases, counts of new cases by gender/age categories.

```
names(tb)[1:10]
```

```
##  [1] "iso2"          "year"          "new_sp"        "new_sp_m04"
##  [5] "new_sp_m514"   "new_sp_m014"   "new_sp_m1524"  "new_sp_m2534"
##  [9] "new_sp_m3544"  "new_sp_m4554"
```

- Main step is to "gather" TB prevalence in the different gender/age categories into a count variable.
  - Complicated by the coding of gender/age categories
- Before gathering, will remove unneeded variables and add country names to supplement 2-letter codes.

Remove variables

- Won't need overall count
- Special categories 0-4 and 5-14 overlap with 0-14 so remove
- Age unknown not useful for analysing trends, so remove

```
tb <- select(tb,-new_sp, -contains("04"), -contains("514"),
             -new_sp_mu, -new_sp_fu)
tb[1:3,1:10]
```

```
##   iso2 year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
## 1   AD 1989          NA            NA            NA            NA
## 2   AD 1990          NA            NA            NA            NA
## 3   AD 1991          NA            NA            NA            NA
##   new_sp_m4554 new_sp_m5564 new_sp_m65 new_sp_f014
## 1           NA           NA          NA          NA
## 2           NA           NA          NA          NA
## 3           NA           NA          NA          NA
```

Add country names to supplement country codes

- I found a translation of the ISO-2 country codes at [http://data.okfn.org/data/core/country-list] and saved as `countryCodes.csv` in the `Notes` folder.

```
cc <- read.csv("countryCodes.csv",stringsAsFactors = FALSE)
# cc has columns "Name" and "Code". "Code" matches "iso2" in tb
```

- Exercise: Find out which ISO-2 codes are in `tb` but not in `countryCodes.csv`, google the missing codes, and add the country names to `cc` manually.

```
unique(tb$iso2[!(tb$iso2 %in% cc$Code)])
```

```
## [1] "AN" "YU"
```

```
cc <- rbind(cc, data.frame(Name=c("Netherlands Antilles","Yugoslavia"),
                           Code=c("AN","YU")))
tb <- inner_join(cc,tb,by = c("Code" = "iso2"))
tb[1:2,1:6]
```

```
##          Name Code year new_sp_m014 new_sp_m1524 new_sp_m2534
## 1 Afghanistan   AF 1980           NA            NA           NA
## 2 Afghanistan   AF 1981           NA            NA           NA
```

Gather counts for demographic groups

- Create demographic variable `demog` and count variable `count` by gathering over all variables except `Name`, `Code` and `year`.

```
tblong <- gather(tb,demog,count,-Name,-Code,-year,na.rm=TRUE)
head(tblong)
```

```
##             Name Code year       demog count
## 13 Afghanistan   AF 1997 new_sp_m014     0
## 14 Afghanistan   AF 1998 new_sp_m014    30
## 15 Afghanistan   AF 1999 new_sp_m014     8
## 16 Afghanistan   AF 2000 new_sp_m014    52
## 17 Afghanistan   AF 2001 new_sp_m014   129
## 18 Afghanistan   AF 2002 new_sp_m014    90
```

Separate gender from age category.

- First remove `new_sp_`, then separate result on first column (`help(separate)`)

```
maxlen <- max(nchar(tblong$demog))
tblong %>% mutate(demog = substr(demog,8,maxlen)) %>%
  separate(demog, into=c("gender","agecat"),sep=1) -> tb
head(tb)
```

```
##          Name Code year gender agecat count
## 1 Afghanistan   AF 1997      m    014     0
## 2 Afghanistan   AF 1998      m    014    30
## 3 Afghanistan   AF 1999      m    014     8
## 4 Afghanistan   AF 2000      m    014    52
## 5 Afghanistan   AF 2001      m    014   129
## 6 Afghanistan   AF 2002      m    014    90
```

```
save(tb,file="tb.RData")
```