TIDY DATA

CREATING TIDY DATA

Tidy data is data that is well structured - that makes it very easy to import - start analysis immediately, with little doctoring if data beforehand

INSTALL AND LOAD PACKAGES

```
pacman::p_load(datasets, tidyverse, tsibble, lubridate, XML)
```

- **datasets** - for demonstration purposes
- **pacman** - for loading/unloading packages
- **tidyverse** - for so many reasons
- **tsibble**- for time series
- **lubridate**- for working with dates
- **XML** - for reading and parsing XML data

ESSENTIALS OF TIDY DATA

1) column == variable/field/attribute
2) row == case/observation (no headers, no spaces, no images, no comments
3) cell == single value encoded with text or numbers
    a) no colours, no shapes, no font sizes - to indicate something that is actually data — bc that dn get preserved when exporting a csv file
4) file == one level of observation/abstraction
    a) like relational databases - put info about customer accounts, about items that selling, particular transactions - each in different tables
    b) do same thing in tidy data - if you were dealing with theses dift levels of abstraction

UNTIDY DATA

- **spreadsheets** (bc so flexible) - biggest offender
    - (things like merged cells, formulas that refer back to things)
- **scraped data** - from website
- **pdf**

⇒ dn meet the structured, rectangular norms of tidy data

TURN UNTIDY DATA INTO TIDY DATA

1) **time-series data**
2) **XML/JSON data** (hierarchically structured and labelled)
3) **compound values** (cells with 2 values)

#1) TIME SERIES DATA

howto- turn untidy time-series -> into tidy data

*About **`sunpsots data`**:*

```
?sunspots              # get info on “sunspots” dataset
```

- bulletin dataset - looks at monthly sunspots bw 1749-1983
- gives monthly mean relative - is time series

```
sunspots               # see full dataset in time series format
```

- results:

-year down side (rows) & months across top (columns)
-convenient way to show data - BUT NOT tidy data - bc variables in 2 directions
-not how data represented in description either - ie. does not show the dates , only the values (head shows values , not header names)

```
> # About the "sunspots" dataset
> ?sunspots       # Get info on "sunspots" dataset
> sunspots        # See full dataset in time series format
      Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1749  58.0  62.6  70.0  55.7  85.0  83.5  94.8  66.3  75.9  75.5 158.6  85.2
1750  73.3  75.9  89.2  88.3  90.0 100.0  85.4 103.0  91.2  65.7  63.3  75.4
1751  70.0  43.5  45.3  56.4  60.7  50.7  66.3  59.8  23.5  23.2  28.5  44.0
1752  35.0  50.0  71.0  59.3  59.7  39.6  78.4  29.3  27.1  46.6  37.6  40.0
1753  44.0  32.0  45.7  38.0  36.0  31.7  22.2  39.0  28.0  25.0  20.0   6.7
```

```
head(subspots)         # show first few observations
```

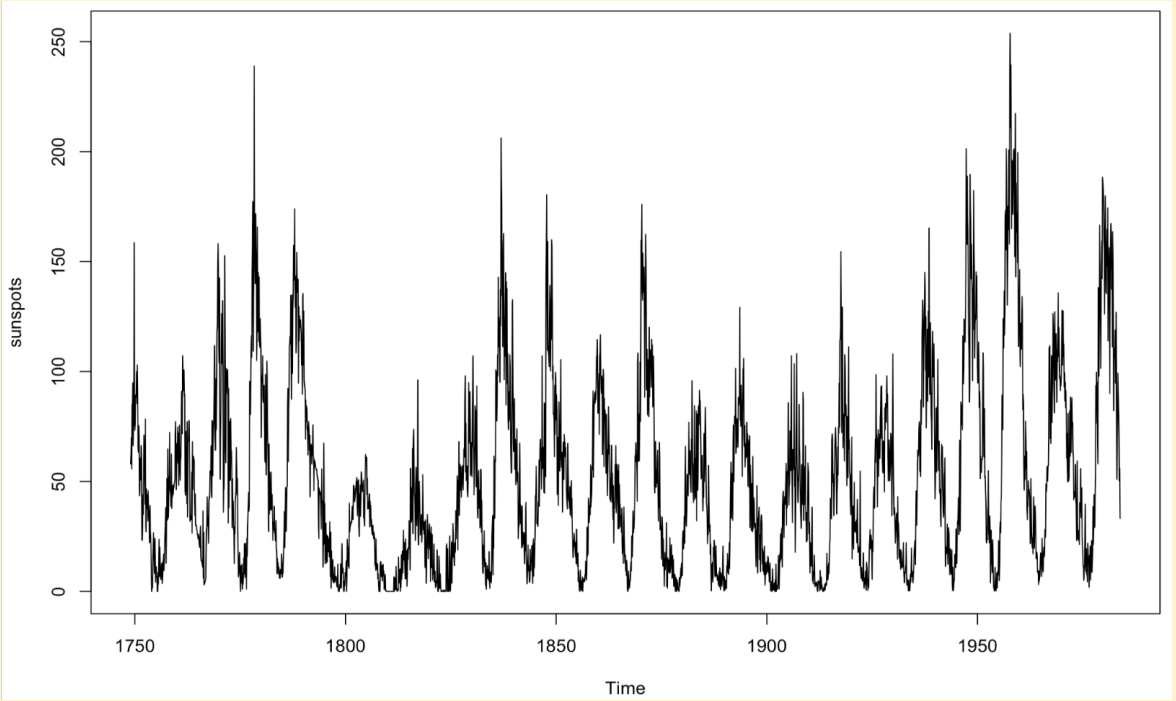
- results

```
> head(sunspots)  # Show first few observations
[1] 58.0 62.6 70.0 55.7 85.0 83.5
```

```
plot(sunspots)        # plot data using base graphics
```

- results:

line plot - shows each of monthly observations > can see strong patterns in sunspots



Tidy the data:

- take data and feed it into new object, called `tidy_ts`
- → rearrange data (via mutate) - create new column for year, new column just month
- → select and rename variables

```
tidy_ts <- sunspots %>%
    as.tibble() %>%          # convert to time series tibble
  mutate(                    # create new variable
       year = year(index),   # create a year column
       month = month(index)  # create a month column
  ) %>%
    select(                  # select, reorder, rename vars
       date = index,         # rename “index” to “date”
       year,                 # use “year” as second variable
       month,                # use “month” as third variable”
       spots = value         # rename “value as “spots”
   ) %>%
    print()                  # show data
```

- results:

environ - data: tidy_ts - 2820 obs of 4 variables
console: date <mth> year <dbl> month <dbl> spots <dbl>

```
# A tsibble: 2,820 x 4 [1M]
      date  year month spots
     <mth> <dbl> <dbl> <dbl>
 1 1749 Jan  1749     1  58
 2 1749 Feb  1749     2  62.6
 3 1749 Mar  1749     3  70
 4 1749 Apr  1749     4  55.7
 5 1749 May  1749     5  85
 6 1749 Jun  1749     6  83.5
 7 1749 Jul  1749     7  94.8
 8 1749 Aug  1749     8  66.3
 9 1749 Sep  1749     9  75.9
10 1749 Oct  1749    10  75.5
# ℹ 2,810 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Data	
tidy_ts	2820 obs. of 4 variables

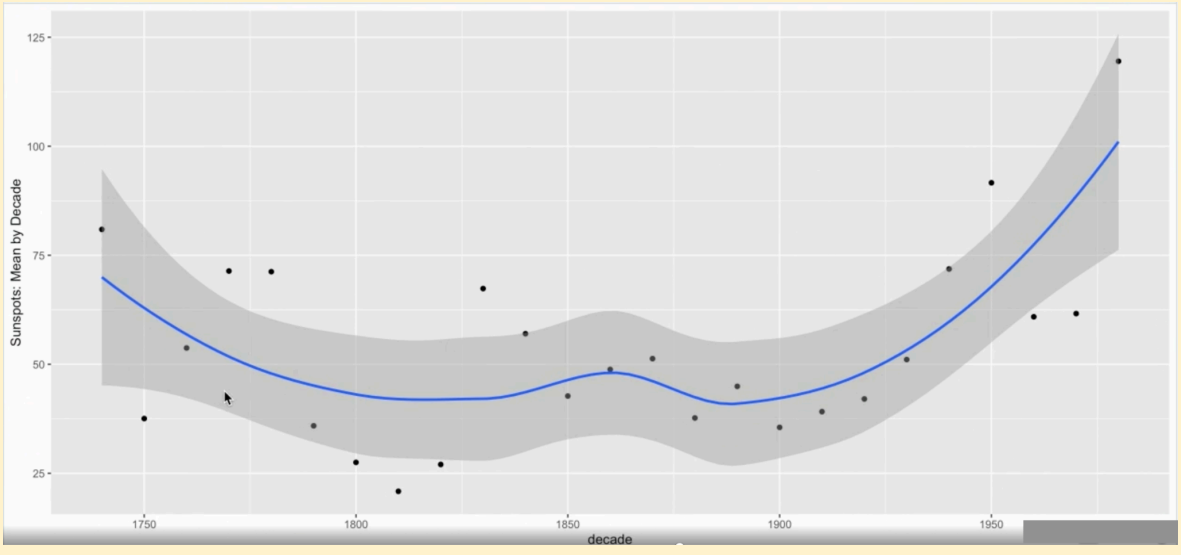
Graph this tidy data by decade:

- index by decade > find mean of each decade > plot

```
tidy_ts %>%
    index_by(                                   # by decade
       decade = ~ floor_date(tidy_ts$date, years(10))
    ) %>%
    summarise(mean_s = mean(spots)) %>%         # mean for decade
   ggplot(aes(decade, mean_s)) +                # plot means
  geom_point() +                                # scatterplots
   geom_smooth() +                              # smoother
    ylab("Sunspots: Mean by Decade")            # label
```

- results graph:

-shows 1 dot for each 10 year decade, along with general trend over those decades in grey,
-easy with ggplot - bc data set up in tibble format



#2) XML/JSON DATA

howto - turn untidy XML/JSON —> into tidy data
- take structural hierarchical XML form into a tidy rectangular dataset

Parse data:

```
tidy_xml <- xmlParse("data/XMLdata.xml")   # parse data
```

- results:

environment - data:values - tidy_xml external pointer of class 'XMLInternalDocument'

Values	
tidy_xml	External pointer of class 'XMLInternalDocument'

Convert to tidy format:

```
tidy_xml <- XML:::xmlAttrsToDataFrame(getNodeSet(tidy_xml,
path='//Row')) %>%                          # read data
  as.tibble()  %>%                          # save as tibble
 mutate_all(as.character) %>%               # save as characters
  'colnames<-' (.[1,]) %>%                  # import var names
  slice(-1) %>%                             # remove first line
  mutate(birthdate = mdy(birthdate))  # format birthdate
   - results: data - tidy_xml  1000 obs of 13 variables
tidy_xml                                    # show data
```

- results:

console -
-is artificial data - program was used to get mock data
-columns - id / first_name / last_name / gender / birthdate / street_address / phone / city / state

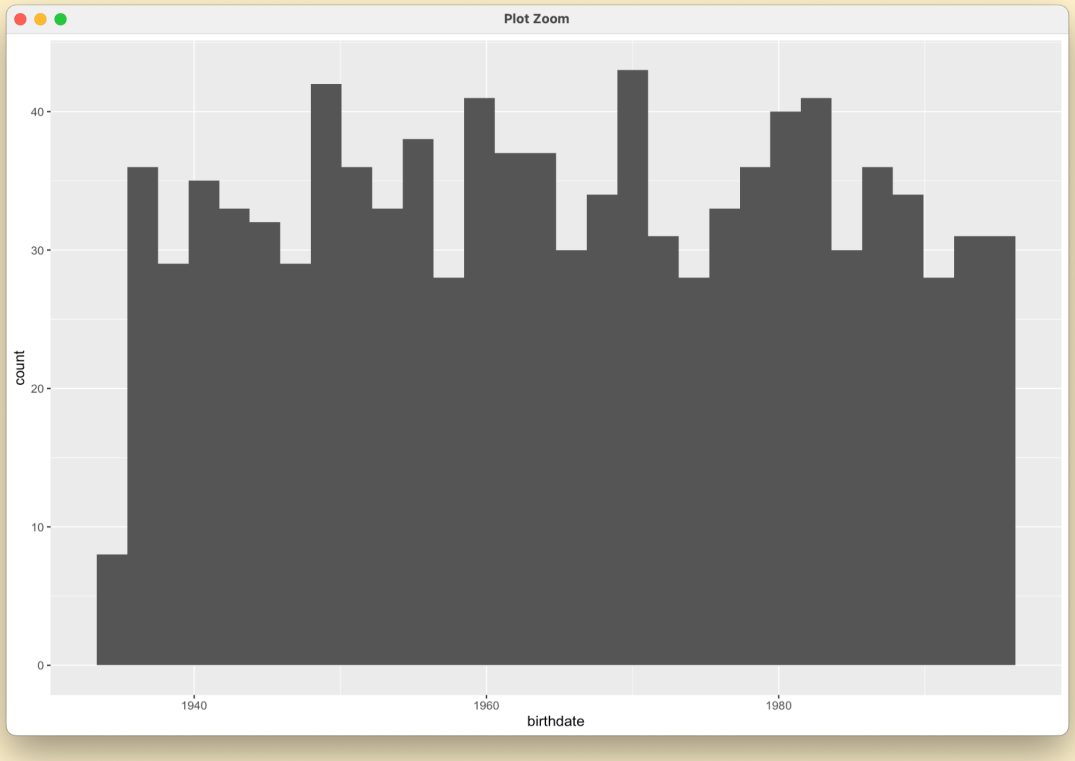
```
> # Show data
> tidy_xml
# A tibble: 1,000 × 13
   id    first_name last_name gender birthdate  street_address      phone city
   <chr> <chr>      <chr>     <chr>  <date>     <chr>               <chr> <chr>
 1 1     Patricia   Armstrong Female 1990-03-30 38636 Twin Pines H… 3-(5… San …
 2 2     Tina       Day       Female 1942-01-16 0 Melrose Court     9-(8… Aust…
 3 3     Karen      Reid      Female 1939-06-23 3224 Cherokee Junc… 7-(3… Colo…
 4 4     Carl       Simmons   Male   1953-05-26 2 Meadow Vale Way   8-(4… Omaha
 5 5     Gloria     Fuller    Female 1936-12-03 166 Eggendart Plaza 3-(1… Dall…
 6 6     Amanda     Watson    Female 1953-03-18 3 Coolidge Place    0-(8… Nort…
 7 7     Katherine  Nichols   Female 1938-07-16 93 Texas Parkway    0-(5… Tucs…
 8 8     Joyce      Perez     Female 1971-03-17 32084 Main Lane     5-(5… Vero…
 9 9     Benjamin   Reid      Male   1988-08-24 40690 Blaine Court  0-(5… Kans…
10 10    Brenda     Peters    Female 1956-05-03 1507 Merchant Cent… 5-(4… Okla…
# ℹ 990 more rows
# ℹ 5 more variables: state <chr>, state_ab <chr>, postal_code <chr>,
#   company <chr>, job_title <chr>
# ℹ Use `print(n = ...)` to see more rows
```

Graph birthdates:

```
tidy_xml %>%
    ggplot(aes(birthdate)) +
    geom_histogram()
```

- results:

basically uniform - birthdates spread ut uniformly



#3) COMPOUND VARIABLES

howto - turn untidy compound variables → into tidy data
- when have more than 1 piece of data in a single cell

saving 4 (very simple) names to "names":

```
names <- c("Angela Ng",
           "Samuel Leeds",
           "Joseph Hernandez",
           "Michelle Howell")
```

- results:

environment - values: names - chr [1:4} "Angela Ng" ..

Values	
names	chr [1:4] "Angela Ng" "Samuel Leeds" "Joseph Hernande…

tidy to names:

```
tidy_names <- names %>%
    enframe() %>%          # convert vector to tibble (bc character is saved in vector)
  separate(value,          # separated "value" into columns
       c("First", "Last")  # names of new columns
  ) %>%
    print()                # show data
```

- results:

console: a tibble of 4 x 3
<int> / <chr> first name / <chr> last name

```
# A tibble: 4 × 3
   name First    Last
  <int> <chr>    <chr>
1     1 Angela   Ng
2     2 Samuel   Leeds
3     3 Joseph   Hernandez
4     4 Michelle Howell
```

environment - data: tidy_names 4 obs of 3 variables

tidy_names	4 obs. of 3 variables