tidy data

Long vs wide formats for data ? We hear this alot but not lots of discussion about it

Here is some data in a wide format. It is called wide because basically all the information about a given observation is on a single line. This format is usually quite good for humans to digest though if there are many columns then things get to be unweildy. But let's start with this idea.

```
##   subject sex control cond1 cond2
## 1       1   M     7.9  12.3  10.7
## 2       2   F     6.3  10.6  11.1
## 3       3   F     9.5  13.1  13.8
## 4       4   M    11.5  13.4  12.9
```

How would we make this into "long data" ? Make it look like the following. Okay well there is the question of why would you do this. It depends. Some graphics of visualization might be easier to do but it all depends. But the data is still usable it's just that now we have a variable that has possible values

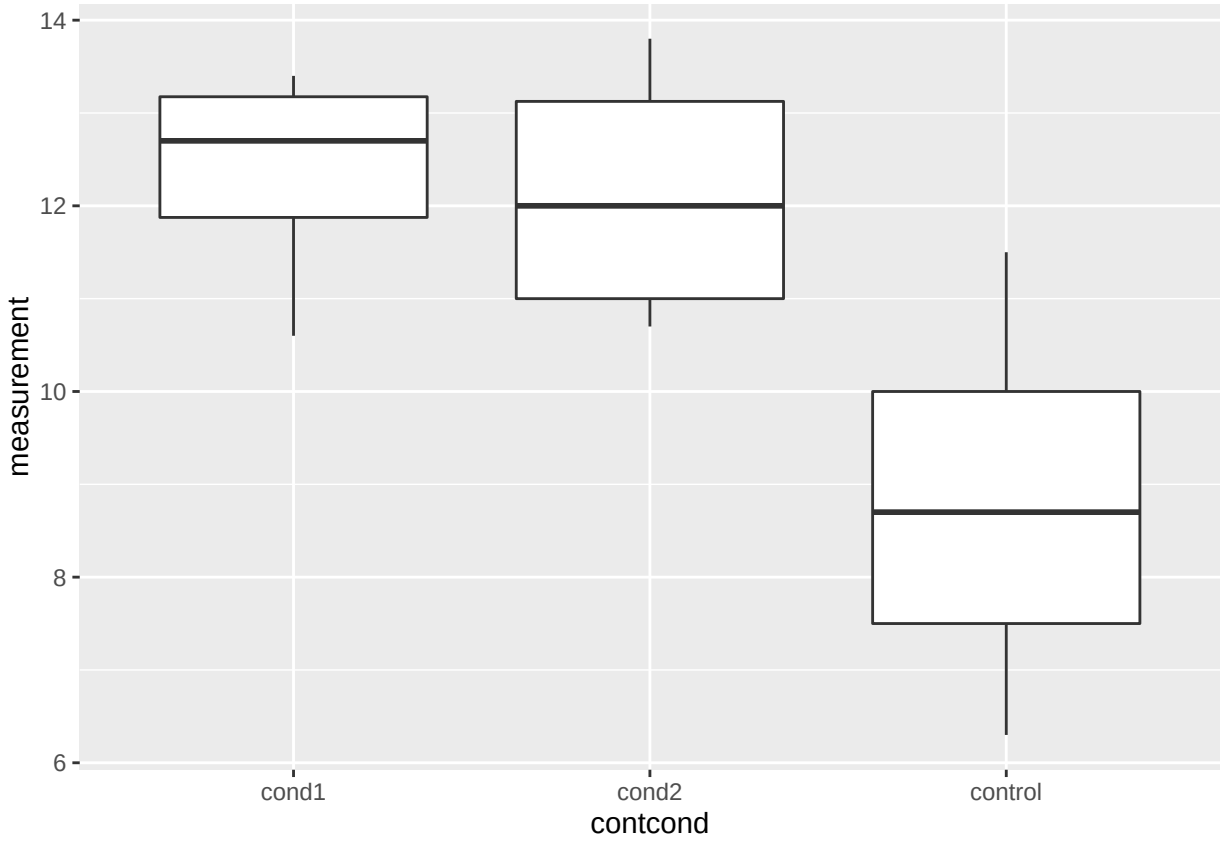
```
##    subject sex contcond measurement
## 1        1   M  control         7.9
## 2        2   F  control         6.3
## 3        3   F  control         9.5
## 4        4   M  control        11.5
## 5        1   M    cond1        12.3
## 6        2   F    cond1        10.6
## 7        3   F    cond1        13.1
## 8        4   M    cond1        13.4
## 9        1   M    cond2        10.7
## 10       2   F    cond2        11.1
## 11       3   F    cond2        13.8
## 12       4   M    cond2        12.9
```

```
data_wide %>% gather(contcond,measurement,control:cond2)
```

```
##    subject sex contcond measurement
## 1        1   M  control         7.9
## 2        2   F  control         6.3
## 3        3   F  control         9.5
## 4        4   M  control        11.5
## 5        1   M    cond1        12.3
## 6        2   F    cond1        10.6
## 7        3   F    cond1        13.1
## 8        4   M    cond1        13.4
## 9        1   M    cond2        10.7
## 10       2   F    cond2        11.1
## 11       3   F    cond2        13.8
## 12       4   M    cond2        12.9
```

Let's look at an application of this that makes some sense. What if we wanted a boxplot of measurements for each of the control, condition 1, and condition 2. If it's in the wide format it's not so easy. But in the long format it is in fact easy

```
data_wide %>% gather(contcond,measurement,control:cond2) %>%
  ggplot(aes(x=contcond,y=measurement)) + geom_boxplot()
```



Now if we started with data that was in long format how would we get to wide ? There is a function called spread that will help with that process. So we have this:

```
data_long
```

```
##    subject sex contcond measurement
## 1        1   M  control         7.9
## 2        2   F  control         6.3
## 3        3   F  control         9.5
## 4        4   M  control        11.5
## 5        1   M    cond1        12.3
## 6        2   F    cond1        10.6
## 7        3   F    cond1        13.1
## 8        4   M    cond1        13.4
## 9        1   M    cond2        10.7
## 10       2   F    cond2        11.1
## 11       3   F    cond2        13.8
## 12       4   M    cond2        12.9
```

```
data_long %>% spread(contcond,measurement)
```

```
##   subject sex cond1 cond2 control
```

```
## 1       1   M  12.3  10.7     7.9
## 2       2   F  10.6  11.1     6.3
## 3       3   F  13.1  13.8     9.5
## 4       4   M  13.4  12.9    11.5
```

What is tidy data ?

There are some formal definitions but inuitively it would be any data in any format that lends itself well to easy analysis and visualization. Ideally it would require minimal cleaning and work before being able to analyze it although rarely do we have the luxury of getting information how we want it ? Why ?

What is tidy data: A more formal idea

How many variables are there in this data ? Name them.

	Pregnant	Not Pregnant
Male	0	5
Female	1	4

Would it be easier to do if the information was presented differently ?

```
##   gender pregnant frequency
## 1   male      yes         0
## 2 female      yes         1
## 3   male       no         5
## 4 female       no         4
```

Let's agree on some naming conventions that should help us. I think we already have an intutive idea about what these mean but just in case:

Storage	Meaning
Table/File	Data Set/Frame
Rows	Observations
Columns	Variables

What are some causes of messiness ?

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Multuple types of experimental unit stored in the same table
- One type of experimental unit stored in multiple tables

Let's take a look at a couple of these. Real world data comes to us in all forms and we have to be able to figure out how to transform the information accordingly

Column headers are values, not variable names

Here is an example wherein each column is actally part of a larger interval describing income. this is basically a summary table. Conceptually we have one big column that could be called "income" with each column representing an interval of that income. In some ways this is a matrix - well it could be so if we have a mind to treat it like that then we could definitely work with the data. However, as is, the column names are not actual unique variables.

```
url <- "https://raw.githubusercontent.com/hadley/tidyr/master/vignettes/pew.csv"
pew <- tbl_df(read.csv(url, stringsAsFactors = FALSE, check.names = FALSE))

pew
```

```
## Source: local data frame [18 x 11]
##
##                   religion <$10k $10-20k $20-30k $30-40k $40-50k $50-75k
##                      (chr) (int)   (int)   (int)   (int)   (int)   (int)
## 1                 Agnostic    27      34      60      81      76     137
## 2                  Atheist    12      27      37      52      35      70
## 3                 Buddhist    27      21      30      34      33      58
## 4                 Catholic   418     617     732     670     638    1116
## 5       Don't know/refused    15      14      15      11      10      35
## 6         Evangelical Prot   575     869    1064     982     881    1486
## 7                    Hindu     1       9       7       9      11      34
## 8  Historically Black Prot   228     244     236     238     197     223
## 9        Jehovah's Witness    20      27      24      24      21      30
## 10                   Jewish    19      19      25      25      30      95
## 11           Mainline Prot   289     495     619     655     651    1107
## 12                  Mormon    29      40      48      51      56     112
## 13                  Muslim     6       7       9      10       9      23
## 14                Orthodox    13      17      23      32      32      47
## 15         Other Christian     9       7      11      13      13      14
## 16            Other Faiths    20      33      40      46      49      63
## 17   Other World Religions     5       2       3       4       2       7
## 18            Unaffiliated   217     299     374     365     341     528
## Variables not shown: $75-100k (int), $100-150k (int), >150k (int), Don't
##   know/refused (int)
```

What are the variables in this data set ? It's like our earlier example so one of them would be frequency. The other two are religion and income. We use the tidyr package to help us "gather" the non variable columns into a "key value pair".

```
pew %>% gather(income, frequency,-religion)
```

```
## Source: local data frame [180 x 3]
##
##                   religion income frequency
##                      (chr)  (chr)     (int)
## 1                 Agnostic  <$10k        27
## 2                  Atheist  <$10k        12
## 3                 Buddhist  <$10k        27
## 4                 Catholic  <$10k       418
## 5       Don't know/refused  <$10k        15
```

```
## 6         Evangelical Prot  <$10k       575
## 7                   Hindu  <$10k         1
## 8  Historically Black Prot  <$10k       228
## 9        Jehovah's Witness  <$10k        20
## 10                 Jewish  <$10k        19
## ..                    ...    ...       ...
```

How did that work ? We created our own label called "income" as a "key" and then told the gather function that there should be a "value" column called "frequency". We want to eliminate all other columns except "religion". This form is tidy because each column represents a variable and each row represents an observation, in this case a demographic unit corresponding to a combination of religion and income.

Multiple variables stored in one column

```
url <- "https://raw.githubusercontent.com/hadley/tidyr/master/vignettes/tb.csv"
tb <- tbl_df(read.csv(url, stringsAsFactors = FALSE))
```

After gathering columns, the key column is sometimes a combination of multiple underlying variable names. This happens in the tb (tuberculosis) dataset, shown below. This dataset comes from the World Health Organisation, and records the counts of confirmed tuberculosis cases by country, year, and demographic group. The demographic groups are broken down by sex (m, f) and age (0-14, 15-25, 25-34, 35-44, 45-54, 55-64, unknown). The first thing to do is to gather up the non variable columns:

```
tb2 <- tb %>%
  gather(demo, n, -iso2, -year, na.rm = TRUE)
tb2
```

```
## Source: local data frame [35,750 x 4]
##
##     iso2  year  demo     n
##    (chr) (int) (chr) (int)
## 1     AD  2005   m04     0
## 2     AD  2006   m04     0
## 3     AD  2008   m04     0
## 4     AE  2006   m04     0
## 5     AE  2007   m04     0
## 6     AE  2008   m04     0
## 7     AG  2007   m04     0
## 8     AL  2005   m04     0
## 9     AL  2006   m04     1
## 10    AL  2007   m04     0
## ..   ...   ...   ...   ...
```

Column headers in this format are often separated by a non-alphanumeric character (e.g. ., -, _, :), or have a fixed width format, like in this dataset. separate() makes it easy to split a compound variables into individual variables. You can either pass it a regular expression to split on (the default is to split on non-alphanumeric columns), or a vector of character positions. In this case we want to split after the first character:

```
tb3 <- tb2 %>%
  separate(demo, c("sex", "age"), 1)
tb3
```

```
## Source: local data frame [35,750 x 5]
##
##     iso2  year   sex   age     n
##    (chr) (int) (chr) (chr) (int)
## 1     AD  2005     m    04     0
## 2     AD  2006     m    04     0
## 3     AD  2008     m    04     0
## 4     AE  2006     m    04     0
## 5     AE  2007     m    04     0
## 6     AE  2008     m    04     0
## 7     AG  2007     m    04     0
## 8     AL  2005     m    04     0
## 9     AL  2006     m    04     1
## 10    AL  2007     m    04     0
## ..   ...   ...   ...   ...   ...
```

Alternative format to the above

As mentioned in class the original tb dataframe has columns which encode more than one variable. There are two pieces of information within the column names - gender and age range. The tabular format we have by default could be read by someone and with some chart annotation they could be walked through it. As I showed (well it's Hadley Wickham's example) the gather command could be used to make the data longer and easier to work with depending on intent.

One thing we could use that doesn't require any additional packages is to use a list wherein each element is a data frame corresponding to gender. This is entirely acceptable and could easily be with the sapply or lapply functions.

```
tblist <- list()

tblist$m <- tb %>% select(1:12)
tblist$f <- tb %>% select(c(1,2,13:22))

names(tblist)
```

```
## [1] "m" "f"
```

Now we still might want to gather the columns in each data frame into a single column matchin the age range.

```
tblonglist <- lapply(tblist, function(x) gather(x,demo,n,-iso2,-year,na.rm=T))
```

mtcars

```
mtcars$name <- rownames(mtcars)
rownames(mtcars) <- NULL

mtcars %>% separate(name,c("make","model"),extra="merge")
```

```
## Warning: Too few values at 1 locations: 6
```

```
##     mpg cyl  disp  hp drat    wt  qsec vs am gear carb     make
## 1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4    Mazda
## 2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4    Mazda
## 3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1   Datsun
## 4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1   Hornet
## 5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2   Hornet
## 6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1  Valiant
## 7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4   Duster
## 8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2     Merc
## 9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2     Merc
## 10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4     Merc
## 11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4     Merc
## 12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3     Merc
## 13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3     Merc
## 14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3     Merc
## 15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 Cadillac
## 16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4  Lincoln
## 17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 Chrysler
## 18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1     Fiat
## 19 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2    Honda
## 20 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1   Toyota
## 21 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1   Toyota
## 22 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2    Dodge
## 23 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2      AMC
## 24 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4   Camaro
## 25 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2  Pontiac
## 26 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1     Fiat
## 27 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2  Porsche
## 28 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2    Lotus
## 29 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4     Ford
## 30 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6  Ferrari
## 31 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 Maserati
## 32 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2    Volvo
##          model
## 1          RX4
## 2      RX4 Wag
## 3          710
## 4      4 Drive
## 5   Sportabout
## 6         <NA>
## 7          360
## 8         240D
## 9          230
## 10         280
## 11        280C
## 12       450SE
## 13       450SL
## 14      450SLC
## 15   Fleetwood
## 16 Continental
## 17    Imperial
## 18         128
## 19       Civic
## 20     Corolla
```

```
## 21      Corona
## 22  Challenger
## 23     Javelin
## 24         Z28
## 25    Firebird
## 26        X1-9
## 27       914-2
## 28      Europa
## 29   Pantera L
## 30        Dino
## 31        Bora
## 32        142E
```

```
mtcars %>% separate(name,c("make","model"),extra="merge") %>% group_by(make) %>% summarize(avg=mean(mpg)
```

```
## Warning: Too few values at 1 locations: 6
```

```
## Source: local data frame [22 x 2]
##
##       make   avg
##      (chr) (dbl)
## 1    Honda 30.40
## 2    Lotus 30.40
## 3     Fiat 29.85
## 4   Toyota 27.70
## 5  Porsche 26.00
## 6   Datsun 22.80
## 7    Volvo 21.40
## 8    Mazda 21.00
## 9   Hornet 20.05
## 10 Ferrari 19.70
## ..     ...   ...
```