Tidy Data

Rules of Tidy data * Each variable has its own column * Each observation has its own row * Each value has its own cell.

These are interrelated, because if we satisfy two of them, then the third is automatically satisfied.

Let's look at some data sets using a read function

```
#Arjun Bhan
table1 <- read.csv("table1.csv", stringsAsFactors = FALSE)
table2 <- read.csv("table2.csv", stringsAsFactors = FALSE)
table3 <- read.csv("table3.csv", stringsAsFactors = FALSE)
table4cases <- read.csv("table4a.csv", stringsAsFactors = FALSE)
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## incomplete final line found by readTableHeader on 'table4a.csv'
```

```
table4deaths <- read.csv("table4b.csv", stringsAsFactors = FALSE)
```

```
table4cases
```

ï..county <chr>	November <int>	December <int>
Dutchess	1737	4798
Westchester	9896	18968
Suffolk	11676	34985

3 rows

```
table4deaths
```

ï..county <chr>	November <int>	December <int>
Dutchess	26	43
Westchester	44	150
Suffolk	33	246
	NA	*NA*

4 rows

```
table3
```

ï..county <chr>	year <int>	month <chr>	rate <chr>
Dutchess	2020	November	26/1737
Dutchess	2020	December	43/4798
Westchester	2020	November	44/9896
Westchester	2020	December	150/18968
Suffolk	2020	November	33/11676
Suffolk	2020	December	246/34985

6 rows

```
table2
```

ï..county <chr>	year <int>	month <chr>	type <chr>	count <int>
Dutchess	2020	November	cases	1737
Dutchess	2020	November	deaths	26
Dutchess	2020	December	cases	4798
Dutchess	2020	December	deaths	43
Westchester	2020	November	cases	9896
Westchester	2020	November	deaths	44
Westchester	2020	December	cases	18968
Westchester	2020	December	deaths	150
Suffolk	2020	November	cases	11676
Suffolk	2020	November	deaths	33

1-10 of 12 rows Previous **1** 2 Next

```
table1
```

ï..county <chr>	year <int>	month <chr>	cases <int>	deaths <int>
Dutchess	2020	November	1737	26
Dutchess	2020	December	4798	43
Westchester	2020	November	9896	44
Westchester	2020	December	18968	150

ï..county	year	month	cases	deaths
<chr>	<int>	<chr>	<int>	<int>
Suffolk	2020	November	11676	33
Suffolk	2020	December	34985	246

6 rows

Only table1 is tidy! Because of it's structure the tidyverse libraries all work as expected.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
table1rates <- table1 %>% mutate(rate = deaths/cases)
table1rates
```

ï..county	year	month	cases	deaths	rate
<chr>	<int>	<chr>	<int>	<int>	<dbl>
Dutchess	2020	November	1737	26	0.014968336
Dutchess	2020	December	4798	43	0.008962068
Westchester	2020	November	9896	44	0.004446241
Westchester	2020	December	18968	150	0.007908056
Suffolk	2020	November	11676	33	0.002826310
Suffolk	2020	December	34985	246	0.007031585

6 rows

Calculate the cases per month

```
table1 %>% count(month, wt = cases)  #wt gives "weight" to matching rows...here based on cases
```

	month	n
	<chr>	<int>
1	December	58751

	month <chr>	n <int>
2	November	23309

2 rows

Addressing untidy data

Consider table4cases and table4deaths. In order to make tidy data, we need to combine them into one data frame/tibble with appropriate headings and values.

```
table4cases
```

ï..county <chr>	November <int>	December <int>
Dutchess	1737	4798
Westchester	9896	18968
Suffolk	11676	34985

3 rows

We are going to “gather” the columns into a new variable named Month. The names of the columns will now be values of Month, and the current “values” are going to be renamed into the variable "cases.

```
table4cases.new <- table4cases %>% gather('November', 'December', key = "Month", value = "cases"
)
table4cases.new
```

ï..county <chr>	Month <chr>	cases <int>
Dutchess	November	1737
Westchester	November	9896
Suffolk	November	11676
Dutchess	December	4798
Westchester	December	18968
Suffolk	December	34985

6 rows

Same thing for table4deaths

```
table4deaths.new <- table4deaths %>% gather('November', 'December', key = "Month", value = "deat
hs")
table4deaths.new
```

ï..county <chr>	Month <chr>	deaths <int>
Dutchess	November	26
Westchester	November	44
Suffolk	November	33
	November	*NA*
Dutchess	December	43
Westchester	December	150
Suffolk	December	246
	December	*NA*

8 rows

Now, we want to “join” these two tables together so that everything is in one data frame

```
table4.tidy <- left_join(table4cases.new, table4deaths.new)
```

```
## Joining, by = c("ï..county", "Month")
```

```
table4.tidy
```

ï..county <chr>	Month <chr>	cases <int>	deaths <int>
Dutchess	November	1737	26
Westchester	November	9896	44
Suffolk	November	11676	33
Dutchess	December	4798	43
Westchester	December	18968	150
Suffolk	December	34985	246

6 rows

Tidying table 2

Now consider table 2

```
table2
```

ï..county <chr>	year <int>	month <chr>	type <chr>	count <int>
Dutchess	2020	November	cases	1737
Dutchess	2020	November	deaths	26
Dutchess	2020	December	cases	4798
Dutchess	2020	December	deaths	43
Westchester	2020	November	cases	9896
Westchester	2020	November	deaths	44
Westchester	2020	December	cases	18968
Westchester	2020	December	deaths	150
Suffolk	2020	November	cases	11676
Suffolk	2020	November	deaths	33

1-10 of 12 rows Previous **1** 2 Next

In this case, we want to “spread” the type column into 2 new variables for “cases” and "deaths. As for gather, the column that contains our new variable names will be the key, and the column that gives the values will be count

```
table2new <- table2 %>% spread(key = type, value = count)
table2new
```

ï..county <chr>	year <int>	month <chr>	cases <int>	deaths <int>
Dutchess	2020	December	4798	43
Dutchess	2020	November	1737	26
Suffolk	2020	December	34985	246
Suffolk	2020	November	11676	33
Westchester	2020	December	18968	150
Westchester	2020	November	9896	44

6 rows

Separators

Now let’s address table 3.

```
table3
```

ï..county <chr>	year <int>	month <chr>	rate <chr>
Dutchess	2020	November	26/1737
Dutchess	2020	December	43/4798
Westchester	2020	November	44/9896
Westchester	2020	December	150/18968
Suffolk	2020	November	33/11676
Suffolk	2020	December	246/34985

6 rows

We need to separate the “rate” variable so that the data is tidy.

```
table3 %>% separate(rate, into = c("deaths", "cases"), sep = "/")
```

ï..county <chr>	year <int>	month <chr>	deaths <chr>	cases <chr>
Dutchess	2020	November	26	1737
Dutchess	2020	December	43	4798
Westchester	2020	November	44	9896
Westchester	2020	December	150	18968
Suffolk	2020	November	33	11676
Suffolk	2020	December	246	34985

6 rows

Case Study

The who dataset provides World Health Organization Data for new tuberculosis cases broken down by year, country, age, gender, and diagnosis method.

```
who
```

country <chr>	is… <chr>	is… <chr>	y… <int>	new_sp_m… <int>	new_sp_m1… <int>	new_sp_m2… <int>	new_sp_m3… <int>	new_sp_
Afghanistan	AF	AFG	1980	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1981	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1982	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1983	*NA*	*NA*	*NA*	*NA*	

country <chr>	is… <chr>	is… <chr>	y… <int>	new_sp_m… <int>	new_sp_m1… <int>	new_sp_m2… <int>	new_sp_m3… <int>	new_sp_
Afghanistan	AF	AFG	1984	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1985	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1986	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1987	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1988	*NA*	*NA*	*NA*	*NA*	
Afghanistan	AF	AFG	1989	*NA*	*NA*	*NA*	*NA*	

1-10 of 7,240 rows | 1-9 of 60 columns Previous **1** 2 3 4 5 6 … 724 Next

So country, iso2, and iso3 are redundant. The other variables are specific counts of TB cases for specific types of groups, we can check the help file for details.

For tidy data, we'd really prefer that each column be a specific variable value for country, year, type of test, gender, age group, and total cases for that combination.

Step 1: Gather case data

We need to gather all the data columns together and the total cases from each column.

```
who1 <- who %>% gather(
  new_sp_m014:newrel_f65, key = "temp",
  value = "cases",
  na.rm = TRUE
)
who1
```

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	temp <chr>	cases <int>
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1998	new_sp_m014	30
Afghanistan	AF	AFG	1999	new_sp_m014	8
Afghanistan	AF	AFG	2000	new_sp_m014	52
Afghanistan	AF	AFG	2001	new_sp_m014	129
Afghanistan	AF	AFG	2002	new_sp_m014	90
Afghanistan	AF	AFG	2003	new_sp_m014	127
Afghanistan	AF	AFG	2004	new_sp_m014	139
Afghanistan	AF	AFG	2005	new_sp_m014	151

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	temp <chr>	cases <int>
Afghanistan	AF	AFG	2006	new_sp_m014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

Step 2: Fix strings in temp variable

This isn't really in the stream of our lecture, but there are a whole host of string commands that are useful.

In this case, all of the columns of who start with "new_" except those that refer to a relapse. This will be easy to fix after we've gathered the data as now we can refer specifically to the temp variable and just do a replacement using mutate.

```
who2 <- who1 %>% mutate(temp = stringr::str_replace(temp, "newrel", "new_rel"))
who2
```

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	temp <chr>	cases <int>
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1998	new_sp_m014	30
Afghanistan	AF	AFG	1999	new_sp_m014	8
Afghanistan	AF	AFG	2000	new_sp_m014	52
Afghanistan	AF	AFG	2001	new_sp_m014	129
Afghanistan	AF	AFG	2002	new_sp_m014	90
Afghanistan	AF	AFG	2003	new_sp_m014	127
Afghanistan	AF	AFG	2004	new_sp_m014	139
Afghanistan	AF	AFG	2005	new_sp_m014	151
Afghanistan	AF	AFG	2006	new_sp_m014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

Step 3: Separate the codes

We can now split the codes at each underscore and assign them to new variable names using separate

```
who3 <- who2 %>% separate(temp,
                          c("new", "type", "sexage"),
                          sep = "_")
who3
```

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	new <chr>	type <chr>	sexage <chr>	cases <int>
Afghanistan	AF	AFG	1997	new	sp	m014	0
Afghanistan	AF	AFG	1998	new	sp	m014	30
Afghanistan	AF	AFG	1999	new	sp	m014	8
Afghanistan	AF	AFG	2000	new	sp	m014	52
Afghanistan	AF	AFG	2001	new	sp	m014	129
Afghanistan	AF	AFG	2002	new	sp	m014	90
Afghanistan	AF	AFG	2003	new	sp	m014	127
Afghanistan	AF	AFG	2004	new	sp	m014	139
Afghanistan	AF	AFG	2005	new	sp	m014	151
Afghanistan	AF	AFG	2006	new	sp	m014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

and now we can separate the sex and age of the cases:

```
who4 <- who3 %>% separate(sexage, c("sex", "age"), sep = 1) #separate after 1st character
who4
```

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	new <chr>	type <chr>	sex <chr>	age <chr>	cases <int>
Afghanistan	AF	AFG	1997	new	sp	m	014	0
Afghanistan	AF	AFG	1998	new	sp	m	014	30
Afghanistan	AF	AFG	1999	new	sp	m	014	8
Afghanistan	AF	AFG	2000	new	sp	m	014	52
Afghanistan	AF	AFG	2001	new	sp	m	014	129
Afghanistan	AF	AFG	2002	new	sp	m	014	90
Afghanistan	AF	AFG	2003	new	sp	m	014	127
Afghanistan	AF	AFG	2004	new	sp	m	014	139
Afghanistan	AF	AFG	2005	new	sp	m	014	151
Afghanistan	AF	AFG	2006	new	sp	m	014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

Step 4: Remove redundant codes

Now let's drop out the iso2 and iso3 indicators, as well as the new column, since it's always the same.

```
who5 <- who4 %>% select(-new, -iso2, -iso3)
who5
```

country <chr>	year <int>	type <chr>	sex <chr>	age <chr>	cases <int>
Afghanistan	1997	sp	m	014	0
Afghanistan	1998	sp	m	014	30
Afghanistan	1999	sp	m	014	8
Afghanistan	2000	sp	m	014	52
Afghanistan	2001	sp	m	014	129
Afghanistan	2002	sp	m	014	90
Afghanistan	2003	sp	m	014	127
Afghanistan	2004	sp	m	014	139
Afghanistan	2005	sp	m	014	151
Afghanistan	2006	sp	m	014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

Our dataset is now tidy!

We could do this all in one step with clever piping:

```
who.new <- who %>% gather(code, cases, new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  mutate(
    code = stringr::str_replace(code, "newrel", "new_rel")
  ) %>%
  separate(code, c("new", "type", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
who.new
```

country <chr>	year <int>	type <chr>	sex <chr>	age <chr>	cases <int>
Afghanistan	1997	sp	m	014	0
Afghanistan	1998	sp	m	014	30
Afghanistan	1999	sp	m	014	8
Afghanistan	2000	sp	m	014	52
Afghanistan	2001	sp	m	014	129
Afghanistan	2002	sp	m	014	90

country <chr>	year <int>	type <chr>	sex <chr>	age <chr>	cases <int>
Afghanistan	2003	sp	m	014	127
Afghanistan	2004	sp	m	014	139
Afghanistan	2005	sp	m	014	151
Afghanistan	2006	sp	m	014	193

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
#-------------------------Week 1: Friday Activity--------------------------
table4.tidy %>% mutate(DeathRate= cases/deaths)
```

ï..county <chr>	Month <chr>	cases <int>	deaths <int>	DeathRate <dbl>
Dutchess	November	1737	26	66.80769
Westchester	November	9896	44	224.90909
Suffolk	November	11676	33	353.81818
Dutchess	December	4798	43	111.58140
Westchester	December	18968	150	126.45333
Suffolk	December	34985	246	142.21545

6 rows

```
table2new %>% mutate(DeathRate= cases/deaths)
```

ï..county <chr>	year <int>	month <chr>	cases <int>	deaths <int>	DeathRate <dbl>
Dutchess	2020	December	4798	43	111.58140
Dutchess	2020	November	1737	26	66.80769
Suffolk	2020	December	34985	246	142.21545
Suffolk	2020	November	11676	33	353.81818
Westchester	2020	December	18968	150	126.45333
Westchester	2020	November	9896	44	224.90909

6 rows

```
# table4.tidy was created through the combination of two separate databases table4cases and tabl
e4deaths. I had to use a join statement to combine the two databases. table2new was created from
table2. Through the use of the spread command, table 2 count column was separated into the cases
and deaths columns. I believe it was more difficult to work with table4.tidy as it was the resul
t of two separate columns merging. This can be very confusing to work with but, luckily they sha
red many of the same column values. I had used the mutate command on both of these databases to
 make a rate of death variable. While using the mutate command both these databases were easy to
alter. Making a new variable for the rate of death was equally difficult in both the databases.
```