R Data Wrangling

Spencer Lourens

January 19 2017

Outline of Lecture

- Data Wrangling
- The dplyr package
  - Selecting, filtering observations
  - Calculating new variables
  - Summarizing the data
- The tidyr package
  - Converting wide to long
  - Converting long to wide

Data Wrangling! (Using dplyr, tidyr)

- Data wrangling: The practical task of transforming the layout, substance, and display of your data - Garrett Grolemund, RStudio

Data Wrangling! (Using dplyr, tidyr)

- Data wrangling: The practical task of transforming the layout, substance, and display of your data - Garrett Grolemund, RStudio
- Data wrangling can be broken into three basic domains: data manipulation, data tidying (for analysis/visualization), and data visualization

Data Wrangling! (Using dplyr, tidyr)

- Data wrangling: The practical task of transforming the layout, substance, and display of your data - Garrett Grolemund, RStudio
- Data wrangling can be broken into three basic domains: data manipulation, data tidying (for analysis/visualization), and data visualization
    - Data manipulation - Change the variables, their values, or the sample units used for analysis (aggregation?)

Data Wrangling! (Using dplyr, tidyr)

- Data wrangling: The practical task of transforming the layout, substance, and display of your data - Garrett Grolemund, RStudio
- Data wrangling can be broken into three basic domains: data manipulation, data tidying (for analysis/visualization), and data visualization
  - Data manipulation - Change the variables, their values, or the sample units used for analysis (aggregation?)
  - Data tidying - Change the LAYOUT of the data - wide, long, etc., most of the time for software use

Data Wrangling! (Using dplyr, tidyr)

- Data wrangling: The practical task of transforming the layout, substance, and display of your data - Garrett Grolemund, RStudio
- Data wrangling can be broken into three basic domains: data manipulation, data tidying (for analysis/visualization), and data visualization
  - Data manipulation - Change the variables, their values, or the sample units used for analysis (aggregation?)
  - Data tidying - Change the LAYOUT of the data - wide, long, etc., most of the time for software use
  - Data visualization - Make use of our visual way of thinking by representing data in a visual format

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)
- A tbl is a dataframe

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)
- A tbl is a dataframe
  - that works seemlessly with dplyr functions

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)
- A tbl is a dataframe
  - that works seemlessly with dplyr functions
  - that optimizes display when printed without printing too much and taking up unnecessary cache memory (slows down your R session)

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)
- A tbl is a dataframe
  - that works seemlessly with dplyr functions
  - that optimizes display when printed without printing too much and taking up unnecessary cache memory (slows down your R session)
- The pipe (%>%) operator

Preliminaries

- The dplyr package creates another structure, called a tbl (table, or as Hadley says, tibble)
- A tbl is a dataframe
  - that works seemlessly with dplyr functions
  - that optimizes display when printed without printing too much and taking up unnecessary cache memory (slows down your R session)
- The pipe (%>%) operator
  - allows more concise, perhaps narrative-looking production of code

Piping example

- Consider the two approaches below

```
x <- c(1,3,4,1,4,10,19,2.5,1)
x %>% mean(na.rm = T)
```

```
## [1] 5.055556
```

```
mean(x)
```

```
## [1] 5.055556
```

- Same result!

Piping example (Cont'd)

- Piping really shines when you have many nested calls as in toy example below

Piping example (Cont'd)

- Piping really shines when you have many nested calls as in toy example below
- The following are equivalent:
  - reportResult(groupResult(calculateResult(df)))
  - df %>% calculateResult() %>% groupResult() %>% reportResult()

Piping example (Cont'd)

- Piping really shines when you have many nested calls as in toy example below
- The following are equivalent:
  - reportResult(groupResult(calculateResult(df))
  - df %>% calculateResult() %>% groupResult() %>% reportResult()
- Notice that the second line with piping is more natural to read from left to right, similar to how we read English

Piping example (Cont'd)

- Piping really shines when you have many nested calls as in toy example below
- The following are equivalent:
  - reportResult(groupResult(calculateResult(df))
  - df %>% calculateResult() %>% groupResult() %>% reportResult()
- Notice that the second line with piping is more natural to read from left to right, similar to how we read English
- Very helpful in more complicated settings

Example datasets

- It may be helpful to go back over these notes in your own R session - we'll test a lot of dplyr functionality on different datasets

Example datasets

- It may be helpful to go back over these notes in your own R session - we'll test a lot of dplyr functionality on different datasets
- Functions from the dplyr package to demonstrate: select, filter, mutate, group_by, summarise . . .

Example datasets

- It may be helpful to go back over these notes in your own R session - we'll test a lot of dplyr functionality on different datasets
- Functions from the dplyr package to demonstrate: select, filter, mutate, group_by, summarise ...
- We will illustrate use of these functions on the flights data set from package nycflights13, previewed on the following slide, and the storms data set from package EDAWR (and other EDAWR datasets), on the subsequent slide.

Dataset flights

- The actual dimension of flights is 336776 by 19

year	month	day	dep_time	sched_dep_time	dep_delay	arr_tim
2013	1	1	517	515	2	83
2013	1	1	533	529	4	85
2013	1	1	542	540	2	92
2013	1	1	544	545	-1	100
2013	1	1	554	600	-6	81
2013	1	1	554	558	-4	74
2013	1	1	555	600	-5	91
2013	1	1	557	600	-3	70
2013	1	1	557	600	-3	83
2013	1	1	558	600	-2	75

Dataset storms

- The actual dimension of storms is 6 by 4

storm	wind	pressure	date
Alberto	110	1007	2000-08-03
Alex	45	1009	1998-07-27
Allison	65	1005	1995-06-03
Ana	40	1013	1997-06-30
Arlene	50	1010	1999-06-11
Arthur	45	1010	1996-06-17

The dplyr::select function

- select: Select/rename variables by name

The dplyr::select function

- select: Select/rename variables by name
- select(.data, . . . , .dots)

The dplyr::select function

- select: Select/rename variables by name
- select(.data, . . . , .dots)
  - .data: a tbl

The dplyr::select function

- select: Select/rename variables by name
- select(.data, . . . , .dots)
  - .data: a tbl
  - . . . : comma separated list of unquoted expressions

The dplyr::select function

- select: Select/rename variables by name
- select(.data, ..., .dots)
  - .data: a tbl
  - ...: comma separated list of unquoted expressions
  - .dots: Use select_() to do standard evaluation (using strings)

The dplyr::select function

```
flights %>% select(year, month)
```

The dplyr::select function

```
flights %>% select(year, month)
```

year	month
2013	1
2013	1
2013	1

The dplyr::select function

- We often have a fixed vector of string variables we would like to keep in the analysis

The dplyr::select function

- We often have a fixed vector of string variables we would like to keep in the analysis
- Use select_() with .dots argument!

The dplyr::select function

- We often have a fixed vector of string variables we would like to keep in the analysis
- Use select_() with .dots argument!

```
flights %>% select_(.dots = c("year", "month", "dep_time",
                              "dep_delay", "arr_time",
                              "arr_delay", "flight"))
```

year	month	dep_time	dep_delay	arr_time	arr_delay	flight
2013	1	517	2	830	11	1545
2013	1	533	4	850	20	1714
2013	1	542	2	923	33	1141
2013	1	544	-1	1004	-18	725
2013	1	554	-6	812	-25	461

The dplyr::select function

- Sometimes we might only want to remove a variable or variables

```
flights %>% select(-year)
flights %>% select_(.dots = c("-year", "-origin", "-day"))
flights %>% select(-sched_dep_time:-arr_time)
```

Special functions for dplyr::select

- Several variables in your dataset may have a similar naming convention, i.e. variablet1, variablet2, etc. that we want to extract

Special functions for dplyr::select

- Several variables in your dataset may have a similar naming convention, i.e. variablet1, variablet2, etc. that we want to extract
- Use ends_with, starts_with, contains, matches, num_range, one_of. . . .

Special functions for dplyr::select

- Several variables in your dataset may have a similar naming convention, i.e. variablet1, variablet2, etc. that we want to extract
- Use ends_with, starts_with, contains, matches, num_range, one_of. . . .

```
## Select variables ending with time and the year/month
flights %>% select(year, month, ends_with("time"))
```

year	month	dep_time	sched_dep_time	arr_time
2013	1	517	515	830
2013	1	533	529	850
2013	1	542	540	923
2013	1	544	545	1004
2013	1	554	600	812

Special functions for dplyr::select

```
## Select variables starting with arr and the year/month
flights %>% select(year, month, starts_with("arr"))
```

year	month	arr_time	arr_delay
2013	1	830	11
2013	1	850	20
2013	1	923	33
2013	1	1004	-18
2013	1	812	-25
2013	1	740	12
2013	1	913	19
2013	1	709	-14

The dplyr::filter function

- Sometimes we need to include only observations which satisfy some criterion for analysis

The dplyr::filter function

- Sometimes we need to include only observations which satisfy some criterion for analysis
  - Include only cases

The dplyr::filter function

- Sometimes we need to include only observations which satisfy some criterion for analysis
  - Include only cases
  - Include only those above certain exposure level

The dplyr::filter function

- Sometimes we need to include only observations which satisfy some criterion for analysis
  - Include only cases
  - Include only those above certain exposure level
  - Include only those with positive survival times

The dplyr::filter function

- Sometimes we need to include only observations which satisfy some criterion for analysis
  - Include only cases
  - Include only those above certain exposure level
  - Include only those with positive survival times
- The filter function comes to the rescue!

The dplyr::filter function

- filter: Return rows with matching conditions

The dplyr::filter function

- filter: Return rows with matching conditions
- filter(.data, ...)

The dplyr::filter function

- filter: Return rows with matching conditions
- filter(.data, ...)
  - .data: a tbl

The dplyr::filter function

- filter: Return rows with matching conditions
- filter(.data, ...)
  - .data: a tbl
  - ...: logical predicates. Multiple conditions are conmbined with &

The dplyr::filter function

- filter: Return rows with matching conditions
- filter(.data, ...)
  - .data: a tbl
  - ...: logical predicates. Multiple conditions are conmbined with &

```
storms %>% filter(pressure == 1010)
```

storm	wind	pressure	date
Arlene	50	1010	1999-06-11
Arthur	45	1010	1996-06-17

The dplyr::filter function

```
storms %>% filter(storm == 'Alberto' | pressure > 1009)
```

storm	wind	pressure	date
Alberto	110	1007	2000-08-03
Ana	40	1013	1997-06-30
Arlene	50	1010	1999-06-11
Arthur	45	1010	1996-06-17

The dplyr::filter function

- Can use all comparisons, logical expressions in filter(), filter_(). The filter_() function takes string arguments rather than explicit expressions

The dplyr::filter function

- Can use all comparisons, logical expressions in filter(), filter_(). The filter_() function takes string arguments rather than explicit expressions
- Logical expressions: &, |, !, any, all, ...

The dplyr::filter function

- Can use all comparisons, logical expressions in filter(), filter_(). The filter_() function takes string arguments rather than explicit expressions
- Logical expressions: &, |, !, any, all, . . .
- Comparisons: <, >, ==, <=, >=, !=, %in%, is.na, !is.na, . . .

The dplyr::filter function

- Can use all comparisons, logical expressions in filter(), filter_(). The filter_() function takes string arguments rather than explicit expressions
- Logical expressions: &, |, !, any, all, ...
- Comparisons: <, >, ==, <=, >=, !=, %in%, is.na, !is.na, ...

```
flights %>% filter(!is.na(air_time))
```

year	month	day	dep_time	sched_dep_time
2013	1	1	517	515
2013	1	1	533	529
2013	1	1	542	540
2013	1	1	544	545

The dplyr::mutate function

- mutate: Add new variables

The dplyr::mutate function

- mutate: Add new variables
- mutate(.data, ...)
  - .data: a tbl
  - ... name-value pairs of expressions, i.e. mean = mean(var)

The dplyr::mutate function

- mutate: Add new variables
- mutate(.data, ...)
  - .data: a tbl
  - ...: name-value pairs of expressions, i.e. mean = mean(var)

storm	wind	pressure	date
Alberto	110	1007	2000-08-03
Alex	45	1009	1998-07-27
Allison	65	1005	1995-06-03
Ana	40	1013	1997-06-30
Arlene	50	1010	1999-06-11
Arthur	45	1010	1996-06-17

The dplyr::summarise function

- summarise: Summarise multiple values to a single value.

The dplyr::summarise function

- summarise: Summarise multiple values to a single value.
- summarise(.data, ...)

The dplyr::summarise function

- summarise: Summarise multiple values to a single value.
- summarise(.data, ...)
  - .data: a tbl

The dplyr::summarise function

- summarise: Summarise multiple values to a single value.
- summarise(.data, . . .)
  - .data: a tbl
  - . . . : name-value pairs of summary functions. For example, mean(), sd(), etc.

The dplyr::summarise function

- summarise: Summarise multiple values to a single value.
- summarise(.data, ...)
  - .data: a tbl
  - ...: name-value pairs of summary functions. For example, mean(), sd(), etc.

```
storms %>% summarise(mean_wind = mean(wind),
                     mean_pressure = mean(pressure))
```

mean_wind	mean_pressure
59.16667	1009

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector
  - last(): last value in a vector

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector
  - last(): last value in a vector
  - nth(): nth value in a vector

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector
  - last(): last value in a vector
  - nth(): nth value in a vector
  - n(): number of values in a vector

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector
  - last(): last value in a vector
  - nth(): nth value in a vector
  - n(): number of values in a vector
  - n_distinct(): number of distinct values in a vector

The dplyr::summarise function

- The dplyr package provides other helpful functions for use with the summarise function
  - first(): first value in a vector
  - last(): last value in a vector
  - nth(): nth value in a vector
  - n(): number of values in a vector
  - n_distinct(): number of distinct values in a vector
- All the usual R functions that take a vector can be used, too!

The dplyr::group_by function

- Summarizing a data overall is helpful, but often times it is most helpful to provide summaries separately for different categories or values in our data

The dplyr::group_by function

- Summarizing a data overall is helpful, but often times it is most helpful to provide summaries separately for different categories or values in our data
- group_by(.data, ..., add = FALSE)

The dplyr::group_by function

- Summarizing a data overall is helpful, but often times it is most helpful to provide summaries separately for different categories or values in our data
- group_by(.data, ..., add = FALSE)
    - .data: a tbl

The dplyr::group_by function

- Summarizing a data overall is helpful, but often times it is most helpful to provide summaries separately for different categories or values in our data
- group_by(.data, ..., add = FALSE)
  - .data: a tbl
  - ...: variables to group by

The dplyr::group_by function

- Summarizing a data overall is helpful, but often times it is most helpful to provide summaries separately for different categories or values in our data
- group_by(.data, ..., add = FALSE)
  - .data: a tbl
  - ...: variables to group by
  - add: If add = FALSE, groups are overridden for tbl .data. Otherwise, groups are added to .data.

The dplyr::group_by function

```
flights %>% group_by(dest) %>% filter(!is.na(air_time))
%>% summarise(totalTime = sum(air_time), totalFlights = n()
```

The dplyr::group_by function

```
flights %>% group_by(dest) %>% filter(!is.na(air_time))
%>% summarise(totalTime = sum(air_time), totalFlights = n(
```

dest	totalTime	totalFlights
ABQ	63289	254
ACK	11106	264
ALB	13287	418
ANC	3305	8
ATL	1901410	16837
AUS	512887	2411
AVL	23461	261
BDL	10492	412
BGR	19374	358
BHM	33027	269
BNA	695901	6084
BOS	585152	15022
BQN	173056	888

The dplyr::group_by function

- Using summarize with group_by makes it really easy to:

The dplyr::group_by function

- Using summarize with group_by makes it really easy to:
  - retain baseline values via the first() function

The dplyr::group_by function

- Using summarize with group_by makes it really easy to:
  - retain baseline values via the first() function
  - calculate the number of observations per individual

The dplyr::group_by function

- Using summarize with group_by makes it really easy to:
    - retain baseline values via the first() function
    - calculate the number of observations per individual
    - calculate the number of distinct observations in an individual for assessing change

The dplyr::group_by function

- Using summarize with group_by makes it really easy to:
  - retain baseline values via the first() function
  - calculate the number of observations per individual
  - calculate the number of distinct observations in an individual for assessing change
  - Recreate other procedures in R that give statistical summaries by groups (describeBy from the psych package comes to mind...)
- You can always ungroup a tbl by running:

```
## grped_df is grouped
flights %>% ungroup()
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay a
##    <int> <int> <int>    <int>          <int>     <dbl>
## 1   2013     1     1      517            515         2
## 2   2013     1     1      533            529         4
## 3   2013     1     1      542            540         2
```

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines "tidy" data

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines “tidy” data
  - Provides rules of thumbs or guidances for creating “tidy” data

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines “tidy” data
  - Provides rules of thumbs or guidances for creating “tidy” data
  - Read a version of his paper on tidy data by typing ??tidyr in the R console and navigating to the tidyr vignette

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines "tidy" data
  - Provides rules of thumbs or guidances for creating "tidy" data
  - Read a version of his paper on tidy data by typing ??tidyr in the R console and navigating to the tidyr vignette
- Tidy data is defined in terms of:

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines "tidy" data
  - Provides rules of thumbs or guidances for creating "tidy" data
  - Read a version of his paper on tidy data by typing ??tidyr in the R console and navigating to the tidyr vignette
- Tidy data is defined in terms of:
  - variables - hold values and represent constructs, such as height, weight, count, gender. . . .

Tidying your data with tidyr

- The tidyr package is a package created by Hadley Wickham
  - Defines "tidy" data
  - Provides rules of thumbs or guidances for creating "tidy" data
  - Read a version of his paper on tidy data by typing ??tidyr in the R console and navigating to the tidyr vignette
- Tidy data is defined in terms of:
  - variables - hold values and represent constructs, such as height, weight, count, gender. . . .
  - observations - one entity representing a unit in our dataset, composed of realizations of one or more variables

Tidying your data with tidyr

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does “type” of observation refer to?

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does "type" of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does "type" of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:
  - usernames and passwords

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does "type" of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:
  - usernames and passwords
  - purchase records (dates, amounts, products purchased)

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does “type” of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:
  - usernames and passwords
  - purchase records (dates, amounts, products purchased)
  - browse records (searches, interests of users)

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does "type" of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:
  - usernames and passwords
  - purchase records (dates, amounts, products purchased)
  - browse records (searches, interests of users)
  - demographic information (further information for targeted ads)

Tidying your data with tidyr

- Essentially, tidy data satisfies the following requirements
  1. each variable forms a column
  2. each observation forms a row
  3. each type of observational unit forms a table
- What does “type” of observation refer to?
- Consider an eCommerce customer database. The following tables might exist:
  - usernames and passwords
  - purchase records (dates, amounts, products purchased)
  - browse records (searches, interests of users)
  - demographic information (further information for targeted ads)
- Tidy data dictates these data be held in separate tables

Tidying your data with tidyr

Tidying your data with tidyr

- Why do the extra work of keeping these data separate when I want to analyze them at the same time?

Tidying your data with tidyr

- Why do the extra work of keeping these data separate when I want to analyze them at the same time?
  - Data changes over time

Tidying your data with tidyr

- Why do the extra work of keeping these data separate when I want to analyze them at the same time?
  - Data changes over time
  - When you merge/join, your analysis dataset becomes much larger which takes more memory

Tidying your data with tidyr

- Why do the extra work of keeping these data separate when I want to analyze them at the same time?
  - Data changes over time
  - When you merge/join, your analysis dataset becomes much larger which takes more memory
  - Some information is repeated across many lines, creating more opportunity for error

The tidyr::gather function

The tidyr::gather function

- gather: Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed

The tidyr::gather function

- gather: Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed
- gather(data, key, value, . . .)

The tidyr::gather function

- gather: Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed
- gather(data, key, value, . . .)
  - data: a data frame (or tbl)

The tidyr::gather function

- gather: Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed
- gather(data, key, value, . . .)
  - data: a data frame (or tbl)
  - key, value: Names of key and value columns to create in output

The tidyr::gather function

- gather: Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed
- gather(data, key, value, ...)
  - data: a data frame (or tbl)
  - key, value: Names of key and value columns to create in output
- Consider dataset cases from the EDAWR package:

country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

The tidyr::gather function

The tidyr::gather function

- Use the gather function to create a tidy data set

The tidyr::gather function

- Use the gather function to create a tidy data set
  - Variables are country, year, and count

The tidyr::gather function

- Use the gather function to create a tidy data set
  - Variables are country, year, and count
  - Each row represents an observation for each country/year pair

The tidyr::gather function

- Use the gather function to create a tidy data set
  - Variables are country, year, and count
  - Each row represents an observation for each country/year pair

```
cases %>% gather(year, count, 2:4)
```

country	year	count
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation
- spread: Spread a key-value pair across multiple columns

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation
- spread: Spread a key-value pair across multiple columns
- spread(data, key, value, ...)

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation
- spread: Spread a key-value pair across multiple columns
- spread(data, key, value, . . .)
  - data: a data frame (or tbl)

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation
- spread: Spread a key-value pair across multiple columns
- spread(data, key, value, . . .)
  - data: a data frame (or tbl)
  - key: the bare (unquoted) name of the column whose values will be used as column headings

The tidyr::spread function

- Perhaps we have a reason to go the opposite direction
  - Have one row per observation, but would like to change our unit of observation
- spread: Spread a key-value pair across multiple columns
- spread(data, key, value, ...)
  - data: a data frame (or tbl)
  - key: the bare (unquoted) name of the column whose values will be used as column headings
  - value: the bare (unquoted) name of the column whose values will populate the cells

The tidyr::spread function

- Use the spread() function to spread out the year information into separate columns (wide format)

The tidyr::spread function

- Use the spread() function to spread out the year information into separate columns (wide format)
- Consider the caseslong dataset we just created below:

The tidyr::spread function

- Use the spread() function to spread out the year information into separate columns (wide format)
- Consider the caseslong dataset we just created below:

country	year	count
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000

The tidyr::spread function

```
caseslong %>% spread(year, count)
```

	country	2011	2012	2013
1	DE	5800	6000	6200
2	FR	7000	6900	7000
3	US	15000	14000	13000
NA	NA	NA	NA	NA
NA.1	NA	NA	NA	NA

Summary of R Data Wrangling

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())
  - Mutate dataset by calculating new variables (mutate())

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())
  - Mutate dataset by calculating new variables (mutate())
  - Summarizing data set with group_by()

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())
  - Mutate dataset by calculating new variables (mutate())
  - Summarizing data set with group_by()
- Data tidying using package tidyr

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())
  - Mutate dataset by calculating new variables (mutate())
  - Summarizing data set with group_by()
- Data tidying using package tidyr
  - Convert to long format (gather())

Summary of R Data Wrangling

- Discussion of Data wrangling and it's components
  - Visualization to come in subsequent lecture using package ggplot2
- Data manipulation using package dplyr
  - Select variables to trim data set (select())
  - Filter observations to trim data set (filter())
  - Mutate dataset by calculating new variables (mutate())
  - Summarizing data set with group_by()
- Data tidying using package tidyr
  - Convert to long format (gather())
  - Convert to wide format (spread())