Programming for Data Analytics

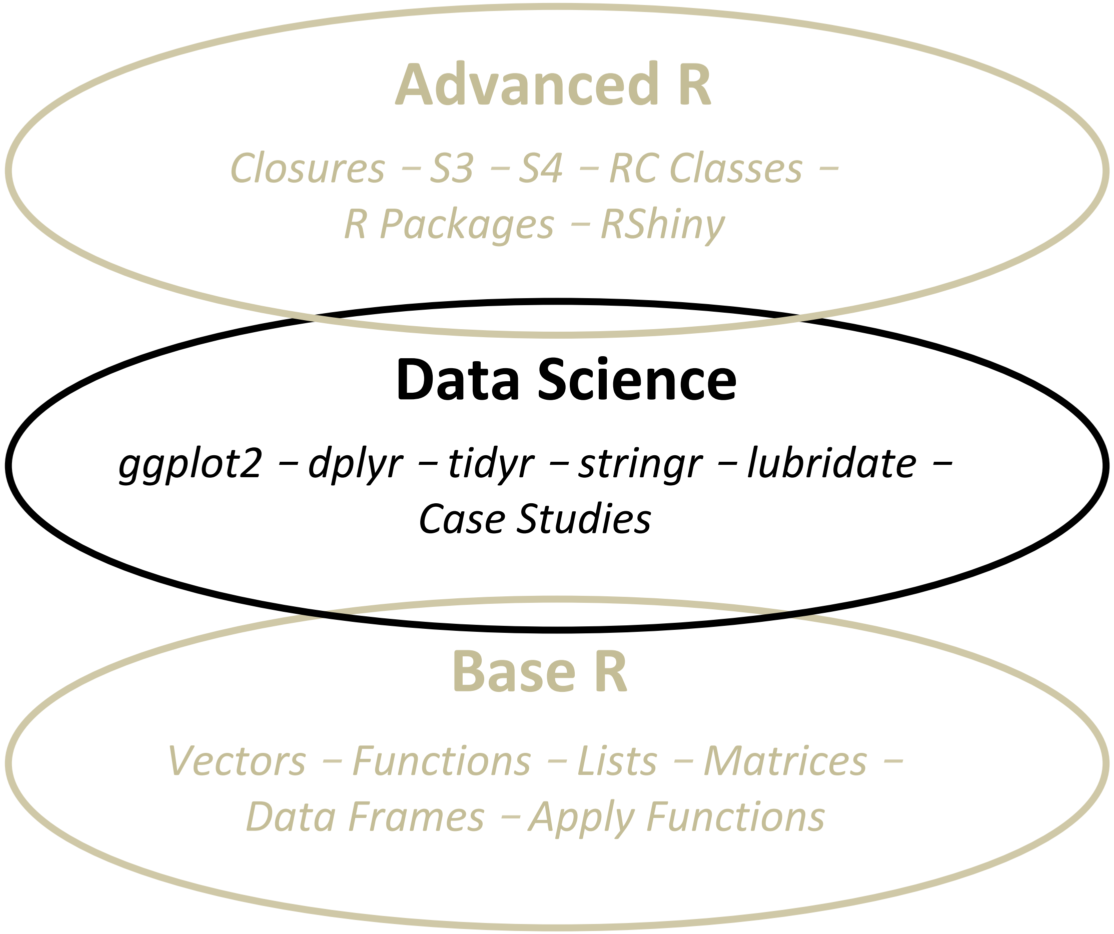
Lecture 7: Relational Data and tidyr

Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

https://github.com/JimDuggan/CT5102

Lecture Overview

- Relational data in dplyr
- Mutating joins
- Filtering joins
- tidyr overview
- gather()
- separate()
- Further topics



(1) Relational Data with dplyr

- Typically, data analysis involves many tables of data that must be combined to answer questions
- Collectively, multiple tables of data are called *relational data*
- Relations are always defined between a pair of tables

key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

Keys

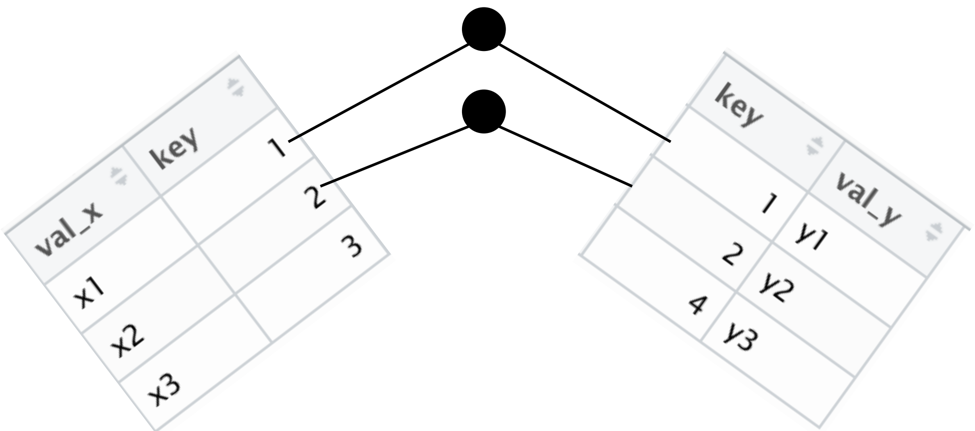
- The variables used to connect each pair of tables are called keys
- **A key is a variable (or set of variables) that uniquely identifies an observation**
- There are two types of keys:
  - A ***primary key*** uniquely identifies an observation in its own table
  - A ***foreign key*** uniquely identifies an observation in another table.

(2) Mutating Joins

- Allows you to combine variables from two tables
- First matches observations by their keys, and then copies across variables from one table to another
- Similar to mutate(), the join functions add variables to the right

Join Types

- Inner Join:
  - matches pairs of observations when their keys are equal
  - Unmatched rows are not included in the result



key	val_x	val_y
1	x1	y1
2	x2	y2

```
inner_join(x,y,by="key")
```

Outer Joins

- An outer join keeps observations that appear in at least one of the tables. There are three types of outer joins (x,y)
  - A *left join* keeps all observations in x
  - A *right join* keeps all observations in y
  - A *full join* keeps all observations in x and y

Left Join

`left_join(x,y,by="key")`

key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	*NA*

Right Join

right_join(x,y,by="key")

key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Full Join

full_join(x,y,by="key")

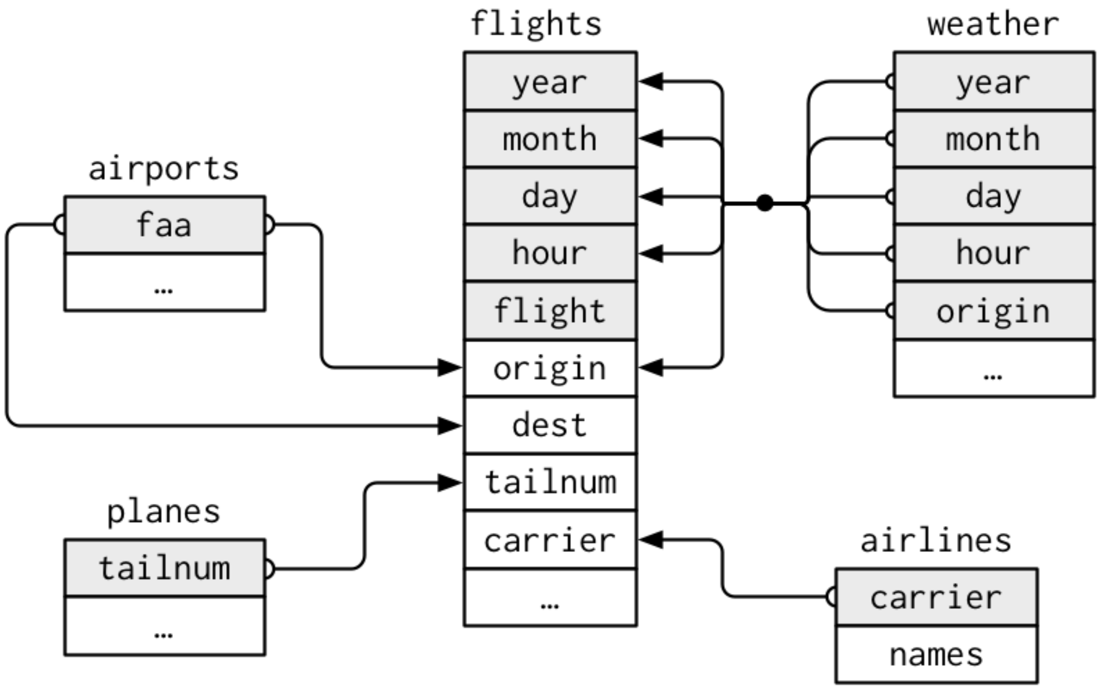
key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	*NA*
4	*NA*	y3

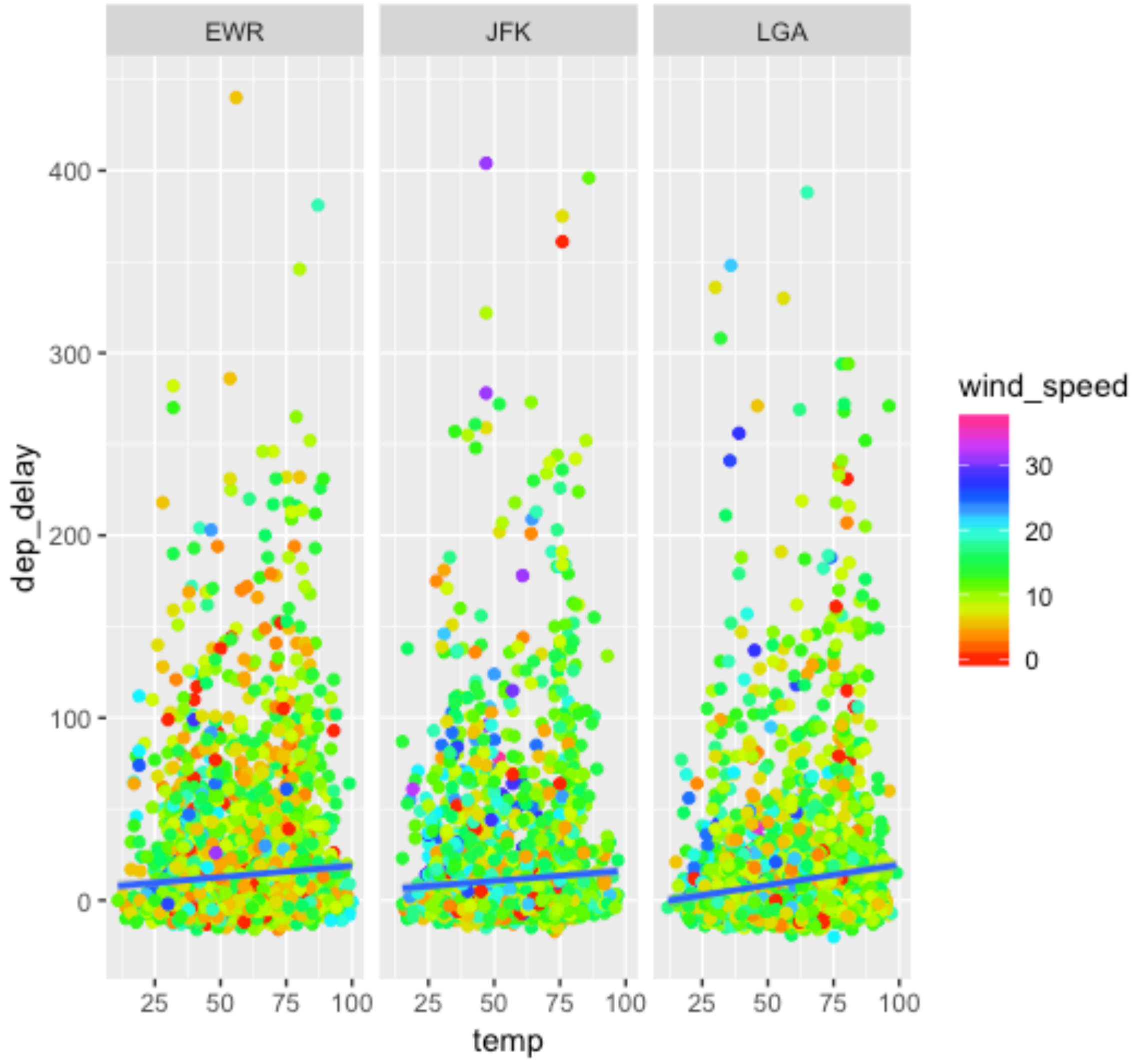
Airports Case Study

- flights connects to planes via a single variable, **tailnum**.
- flights connects to airlines through the **carrier** variable.
- flights connects to airports in two ways: via the **origin** and **dest** variables.
- flights connects to weather via **origin** (the location), and **year, month, day and hour** (the time).



Challenge 7.1

- Filter out incomplete flights from the dataset
- Join the flights data to the weather data
- Filter out missing temperature values
- Plot the relationship between temperatures and departure delays, facet by origin and colour by wind_speed
- Use a sample of 10000 for the plot, with seed 99.



(3) Filtering Joins

- Match observations in the same way as mutating joins, but affect the observations, not the variables
- Two types:
  - semi_join(x,y) keeps all observations in x that have a match in y
  - anti_join(x,y), drops all observations in x that have a match in y.

Semi Join

semi_join(x,y,by="key")

key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

key	val_x
1	x1
2	x2

keeps all observations in x that have a match in y

Anti Join

```
anti_join(x,y,by="key")
```

key	val_x
1	x1
2	x2
3	x3

key	val_y
1	y1
2	y2
4	y3

key	val_x
3	x3

drops all observations in x that have a match in y

Additional Example

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

```r
x <- data.frame(
  name = c("John", "Paul", "George", "Ringo", "Stuart", "Pete"),
  instrument = c("guitar","bass","guitar","drums","bass","drums"),
  stringsAsFactors = F
)
```

```r
y <- data.frame(
  name = c("John", "Paul", "George", "Ringo", "Brian"),
  band = c(T,T,T,T,F),
  stringsAsFactors = F
)
```

Type	Action
inner	Include only rows in **both** x and y

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

```
> inner_join(x,y)
Joining, by = "name"
    name instrument band
1   John     guitar TRUE
2   Paul       bass TRUE
3 George     guitar TRUE
4  Ringo      drums TRUE
```

Type	Action
left	Include all of x, and matching rows of y

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

```
> left_join(x,y)
Joining, by = "name"
    name instrument band
1   John     guitar TRUE
2   Paul       bass TRUE
3 George     guitar TRUE
4  Ringo      drums TRUE
5 Stuart       bass   NA
6   Pete      drums   NA
```

Type	Action
semi	Include rows of x that match y

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

```
>
> semi_join(x,y)
Joining, by = "name"
    name instrument
1   John     guitar
2   Paul       bass
3 George     guitar
4  Ringo      drums
```

Type	Action
anti	Include rows of x that **don't** match y

name	instrument
John	guitar
Paul	bass
George	guitar
Ringo	drums
Stuart	bass
Pete	drums

name	band
John	T
Paul	T
George	T
Ringo	T
Brian	F

```
> anti_join(x,y)
Joining, by = "name"
    name instrument
1   Pete      drums
2 Stuart       bass
```

(5) Tidy Data - Overview

- What is data tidying?
  - Structuring datasets to facilitate analysis
- The tidy data standard is designed to:
  - Facilitate initial exploration and analysis of data
  - Simplify the development of data analysis tools that work well together
- Principles closely related to relational algebra (Codd 1990)
- Related packages: tidyr, ggplot2, dplyr

Why tidy data? (Wickham et al. p150)

- Advantage to picking one consistent way of storing data. Easier to learn tools that work with tidy data because they have a underlying uniformity
- Specific advantage to placing variables in columns because it allows R's vectorised functions to shine.
- dplyr, ggplot2 designed to work with tidy data

Typical Structure: Rows and Columns (Wickham 2014)

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

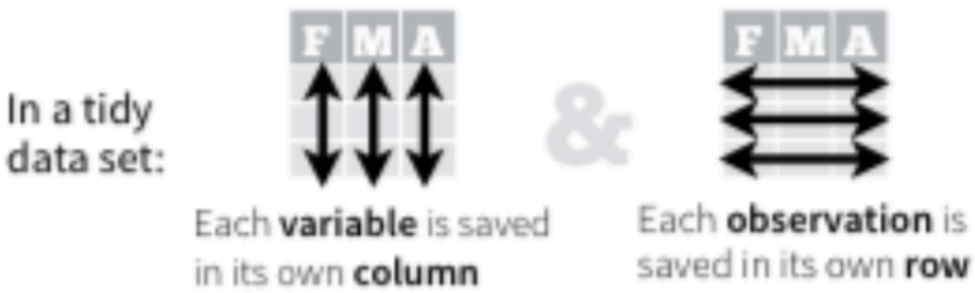
Table 2: The same data as in Table 1 but structured differently.

Numbers refer to the result of the treatments on a given person.

Rules for a Tidy Dataset

- Each variable must have its own column
- Each observation must have its own row
- Each value must have its own cell

- *Put every dataset in a tibble*
- *Put each variable in a column*



https://rpubs.com/bradleyboehmke/data_wrangling

Example in R

```r
untidy <- data.frame(
  name = c("John Smith", "Jane Doe", "Mary Johnson"),
  treatmenta = c(NA, 16, 3),
  treatmentb = c(2, 11, 1)
)
```

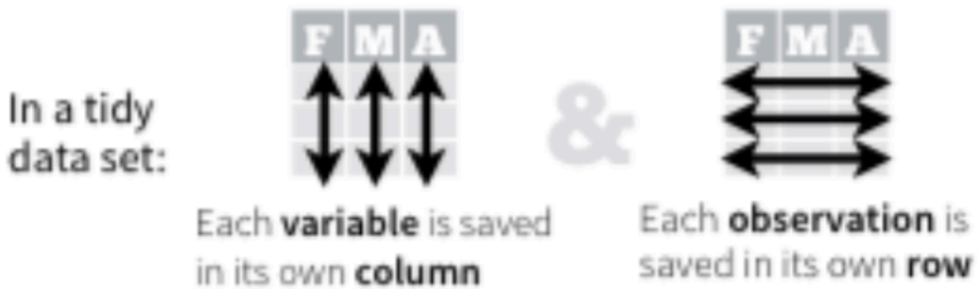
```
>
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```

In a tidy data set…

Variables

- Person (John, Jane, and Mary)
- Treatments (a or b)
- Result (6 values including NA)
- 6 observations

```
>
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```



https://rpubs.com/bradleyboehmke/data_wrangling

The goal…

```
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```

```
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

tidyr package – four fundamental functions of data tidying

- **gather()** takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer
- **separate()** splits a single column into multiple columns
- **spread()** takes two columns (key and value) and spreads into multiple columns, it makes long data wider
- **unite()** combines multiple columns into a single column

(5) gather() process

```
>
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
>
```

```
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

gather()

https://rpubs.com/bradleyboehmke/data_wrangling

```
Function:        gather(data, key, value, ..., na.rm = FALSE, convert = FALSE)
Same as:         data %>% gather(key, value, ..., na.rm = FALSE, convert = FALSE)

Arguments:
        data:            data frame
        key:             column name representing new variable
        value:           column name representing variable values
        ...:             names of columns to gather (or not gather)
        na.rm:           option to remove observations with missing values (represented by NAs)
        convert:         if TRUE will automatically convert values to logical, integer, numeric, complex or
                         factor as appropriate
```

```
> tidy <- gather(untidy,key=Treatment,value=Outcome,treatmenta:treatmentb)
>
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

```
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```

Challenge 7.2

- Convert the following to tidy data format

StudentID	CX1000	CX1001	CX1002	CX1003	CX1004	CX1005	CX1006	CX1007	CX1008	CX1009
1111111	56	51	78	85	63	45	55	59	52	76
1111112	56	64	68	80	70	39	46	60	55	74
1111113	52	61	63	81	71	49	54	61	54	76
1111114	50	42	72	81	63	44	62	59	56	68
1111115	67	53	77	84	65	52	63	62	52	71
1111116	45	57	62	32	61	56	62	51	55	79
1111117	67	58	54	77	75	44	58	62	57	77
1111118	69	50	66	78	72	39	60	58	57	84
1111119	70	56	62	80	71	52	60	63	54	70
1111120	51	52	46	82	74	42	66	63	55	73

(6) separate()

- Separate pulls apart one column into multiple columns
- It splits the information based on finding a non-alphanumeric character
- Separator can be defined (sep="/")
- A converter can find best type for the result, if needed.

```
> table3
# A tibble: 6 x 3
      country  year            rate
*       <chr> <int>           <chr>
1 Afghanistan  1999     745/19987071
2 Afghanistan  2000    2666/20595360
3      Brazil  1999  37737/172006362
4      Brazil  2000  80488/174504898
5       China  1999 212258/1272915272
6       China  2000 213766/1280428583
```

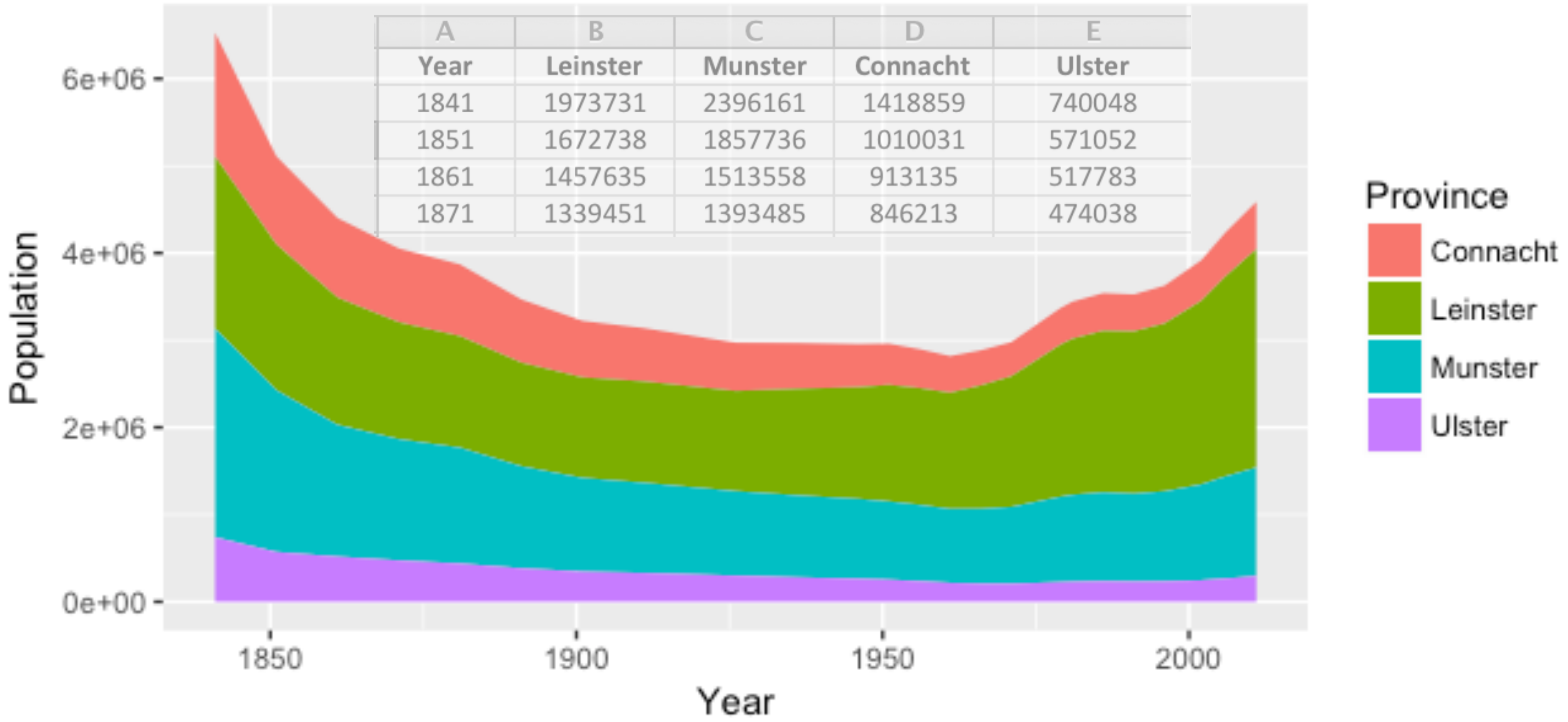
```
Function:        separate(data, col, into, sep = " ", remove = TRUE, convert = FALSE)
Same as:         data %>% separate(col, into, sep = " ", remove = TRUE, convert = FALSE)

Arguments:
        data:            data frame
        col:             column name representing current variable
        into:            names of variables representing new variables
        sep:             how to separate current variable (char, num, or symbol)
        remove:          if TRUE, remove input column from output data frame
        convert:         if TRUE will automatically convert values to logical, integer, numeric, complex or
                         factor as appropriate
```

```
> table3 %>%
+   separate(rate,into=c("cases","population"),
+                convert=TRUE)
# A tibble: 6 x 4
      country  year  cases population
*       <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```

Challenge 7.3

- Transform the census data to tidy format and create the following plot



A	B	C	D	E
Year	Leinster	Munster	Connacht	Ulster
1841	1973731	2396161	1418859	740048
1851	1672738	1857736	1010031	571052
1861	1457635	1513558	913135	517783
1871	1339451	1393485	846213	474038

RELATED TOPICS

Set Operations

- All operations work with a complete row, comparing the values of every variable
- *These expect the x and y inputs to have the same variables, and treat the observations like sets*
  - intersect(x,y) returns only observations in both x and y
  - union(x,y) returns unique observations in x and y
  - setdiff(x,y) returns observations in x, but not in y

intersect(df1,df2)

df1

x	y
1	1
2	1

df2

x	y
1	1
1	2

x	y
1	1

returns only observations in both df1 and df2

union(df1,df2)

df1

x	y
1	1
2	1

df2

x	y
1	1
1	2

x	y
1	2
2	1
1	1

returns unique observations in df1 and df2 (no duplicates)

setdiff(df1,df2)

df1

x	y
1	1
2	1

df2

x	y
1	1
1	2

x	y
2	1

returns observations in df1, but not in df2

spread()

https://rpubs.com/bradleyboehmke/data_wrangling

```
Function:        spread(data, key, value, fill = NA, convert = FALSE)
Same as:         data %>% spread(key, value, fill = NA, convert = FALSE)

Arguments:
        data:            data frame
        key:             column values to convert to multiple columns
        value:           single column values to convert to multiple columns' values
        fill:            If there isn't a value for every combination of the other variables and the key
                         column, this value will be substituted
        convert:         if TRUE will automatically convert values to logical, integer, numeric, complex or
                         factor as appropriate
```

```
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

```
> spread(tidy,Treatment,Outcome)
          name treatmenta treatmentb
1     Jane Doe         16         11
2   John Smith         NA          2
3 Mary Johnson          3          1
```

Spreading

- Spreading is the opposite of gathering
- Useful when observations are scattered across multiple rows

```
> table2
# A tibble: 12 x 4
      country  year       type     count
        <chr> <int>      <chr>     <int>
 1 Afghanistan  1999      cases       745
 2 Afghanistan  1999 population  19987071
 3 Afghanistan  2000      cases      2666
 4 Afghanistan  2000 population  20595360
 5      Brazil  1999      cases     37737
 6      Brazil  1999 population 172006362
 7      Brazil  2000      cases     80488
 8      Brazil  2000 population 174504898
```

To tidy up the data

- Two parameters needed
- The column that contains the variable names (key). Here it is type.
- The column that contains values from multiple variables (value). Here it's count.

```
> table2
# A tibble: 12 x 4
      country  year       type     count
        <chr> <int>      <chr>     <int>
 1 Afghanistan  1999      cases       745
 2 Afghanistan  1999 population  19987071
 3 Afghanistan  2000      cases      2666
 4 Afghanistan  2000 population  20595360
 5      Brazil  1999      cases     37737
 6      Brazil  1999 population 172006362
 7      Brazil  2000      cases     80488
 8      Brazil  2000 population 174504898
```

The spread operation…

```
> spread(table2,key=type,value=count)
# A tibble: 6 x 4
      country  year  cases population
*       <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```

unite()

- The inverse of separate()
- Combines multiple columns into a single column
- Can use this to revert the transformed table3 back to its original

```
> x
# A tibble: 6 x 4
      country  year  cases population
*       <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```