R data Wragling: tidyr and dplyr

Wintersession 2024

Boriana Pratt, Office of Population Research

bpratt@princeton.edu

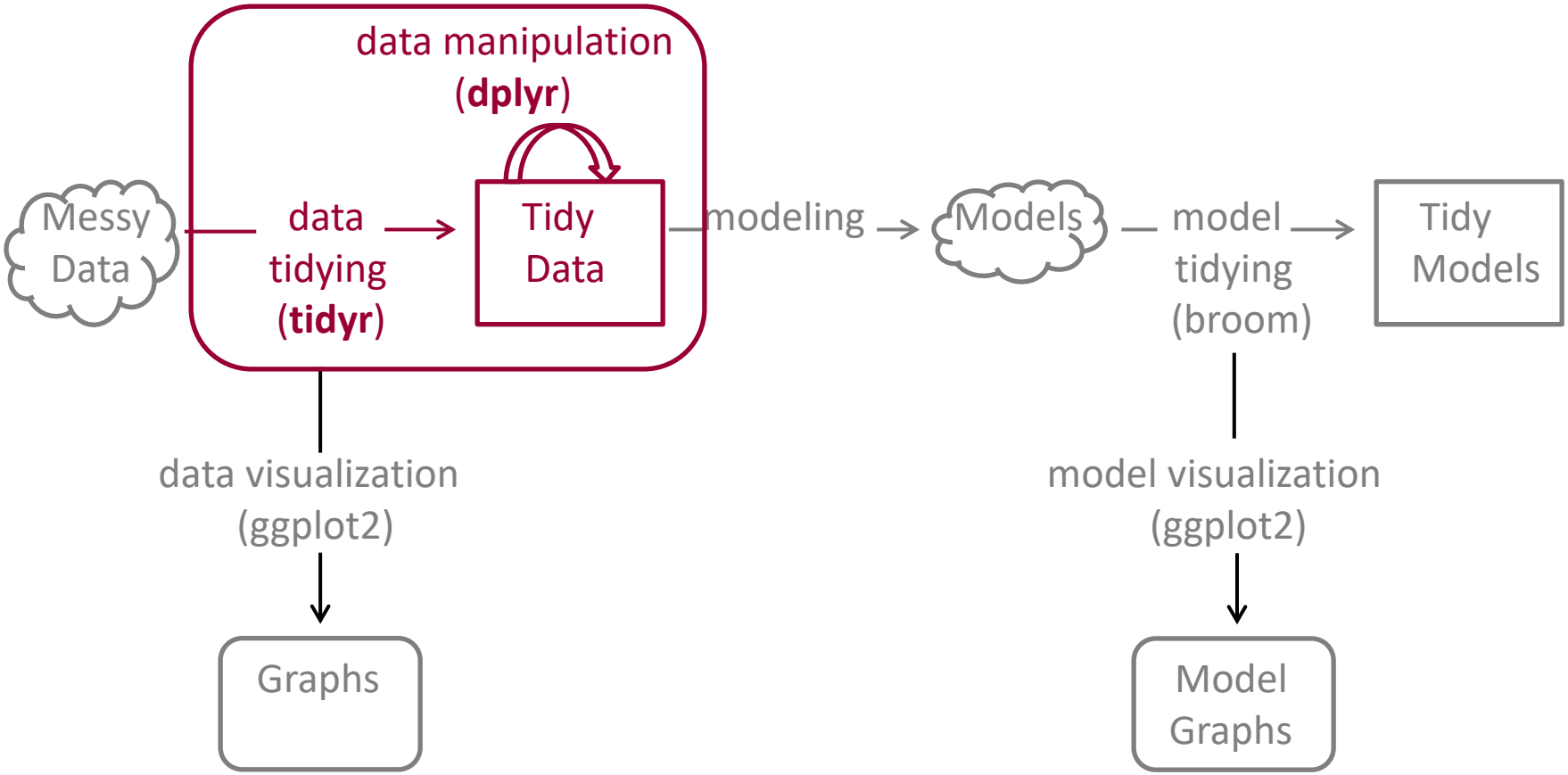
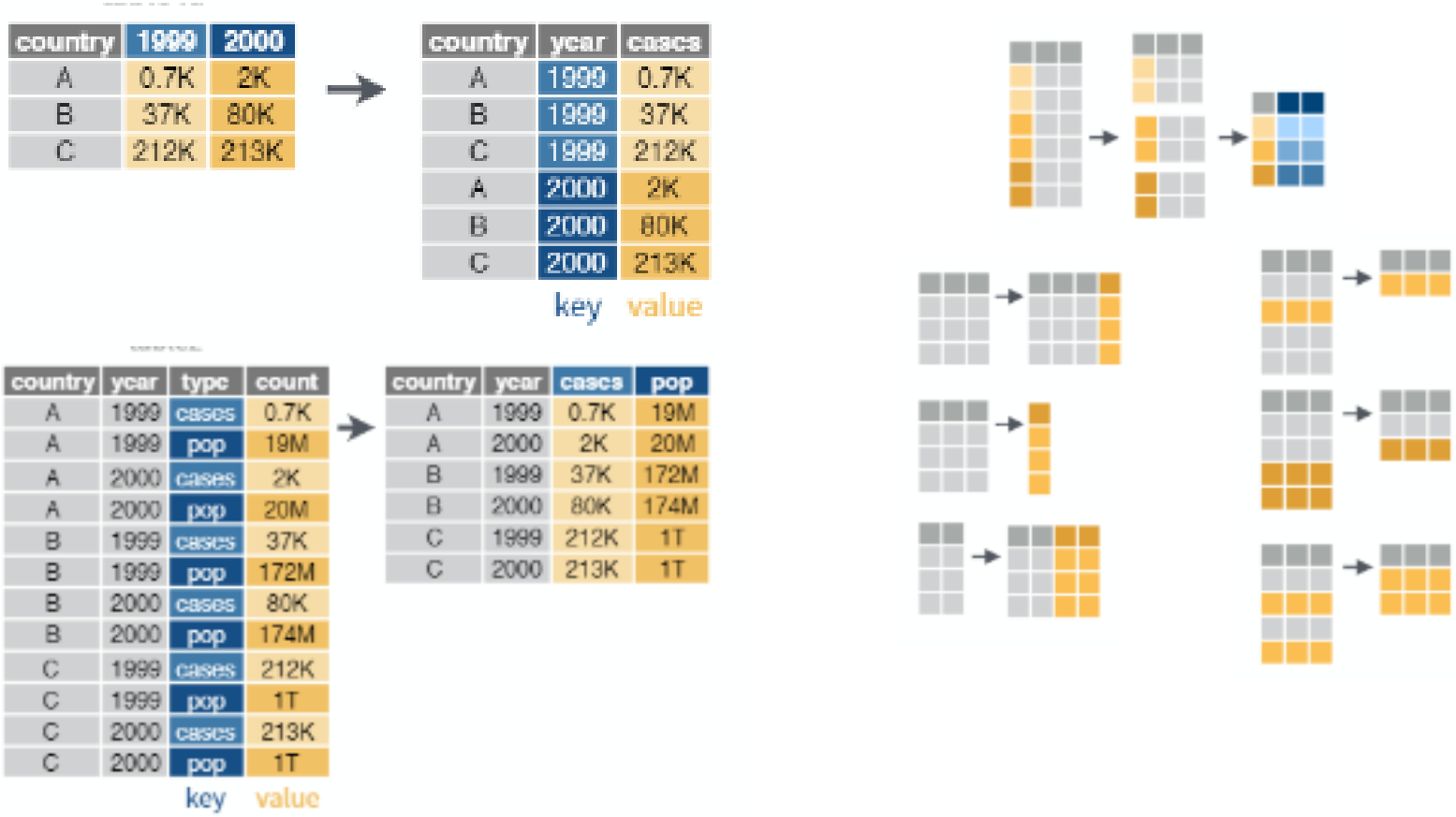


Diagram is based on David Robinson, 4/11/2015, "broom: an R Package to Convert Statistical Models into Tidy Data Frames"

Tidy Data

Data sets come in many formats … but many R tools work best with one format - a tidy dataset.

Tidy data is data stored in a **standardized** tidy format.



country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

key value

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

Source: RStudio tidyr Cheatsheet - https://github.com/rstudio/cheatsheets/blob/main/tidyr.pdf
and Rstudio Data Transformation Cheetsheet - https://github.com/rstudio/cheatsheets/blob/master/data-transformation.pdf

Tidy Format

tidyr

Some terms:
- a data set is a collection of values
- many data sets are organized as rectangular tables made up of rows and columns

Values are organized in two ways:

every value belongs to a variable and an observation

- a variable contains all values that measure the same underlying attribute (for example, life expectancy or total fertility rate) across units

- an observation contains all values measured on the same unit (for example, country-year)

Each **variable** is saved in its own column and column headers are variable names, not values.
Each **observation** is saved in its own row.
Each **cell** is a single value.
Each **type of observational unit** is stored in a single table. *

* this rule/feature is not part of “R for Data Science” by Hadley Wickham

Data Example 1

tidyr

Tidy?

year	le	le_male	le_female	le_w	le_wmale	le_wfemale	le_b	le_bmale	le_bfemale
1900	47.3	46.3	48.3	47.6	46.6	48.7	33.0	32.5	33.5
1901	49.1	47.6	50.6	49.4	48.0	51.0	33.7	32.2	35.3
1902	51.5	49.8	53.4	51.9	50.2	53.8	34.6	32.9	36.4
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
1994	75.7	72.4	79.0	76.5	73.3	79.6	69.5	64.9	73.9
1995	75.8	72.5	78.9	76.5	73.4	79.6	69.6	65.2	73.9
1996	76.1	73.1	79.1	76.8	73.9	79.7	70.2	66.1	74.2
1997	76.5	73.6	79.4	77.2	74.3	79.9	71.1	67.2	74.7
1998	76.7	73.8	79.5	77.3	74.5	80.0	71.3	67.6	74.8
1999	76.7	73.9	79.4	77.3	74.6	79.9	71.4	67.8	74.7

Data Example 1

tidyr

Tidy?

year	le	le_male	le_female	le_w	le_wmale	le_wfemale	le_b	le_bmale	le_bfemale
1900	47.3	46.3	48.3	47.6	46.6	48.7	33.0	32.5	33.5
1901	49.1	47.6	50.6	49.4	48.0	51.0	33.7	32.2	35.3
1902	51.5	49.8	53.4	51.9	50.2	53.8	34.6	32.9	36.4
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
1994	75.7	72.4	79.0	76.5	73.3	79.6	69.5	64.9	73.9
1995	75.8	72.5	78.9	76.5	73.4	79.6	69.6	65.2	73.9
1996	76.1	73.1	79.1	76.8	73.9	79.7	70.2	66.1	74.2
1997	76.5	73.6	79.4	77.2	74.3	79.9	71.1	67.2	74.7
1998	76.7	73.8	79.5	77.3	74.5	80.0	71.3	67.6	74.8
1999	76.7	73.9	79.4	77.3	74.6	79.9	71.4	67.8	74.7

Column headers contain **values**: male, female, b, w

Data Example 2

Tidy ?

country	measure	value
Algeria	imr	24
Algeria	tfr	2.9
Algeria	le	73
Egypt	imr	24
Egypt	tfr	2.9
Egypt	le	72
Libya	imr	14
Libya	tfr	2.6
Libya	le	75
Morocco	imr	30
Morocco	tfr	2.3
Morocco	le	72
South Sudan	imr	101
South Sudan	tfr	5.4
South Sudan	le	52
.	.	.
.	.	.
.	.	.

Data Example 2

tidyr

Tidy?

country	measure	value
Algeria	imr	24
Algeria	tfr	2.9
Algeria	le	73
Egypt	imr	24
Egypt	tfr	2.9
Egypt	le	72
Libya	imr	14
Libya	tfr	2.6
Libya	le	75
Morocco	imr	30
Morocco	tfr	2.3
Morocco	le	72
South Sudan	imr	101
South Sudan	tfr	5.4
South Sudan	le	52
.	.	.
.	.	.
.	.	.

Each variable is not saved in its own column.

Data Example 3

tidyr

Tidy?

country	pop2012	imr	tfr	le	leM	leF	region	area
Algeria	37.4	24	2.9	73	72	75	Northern Africa	Africa
Egypt	82.3	24	2.9	72	70	74	Northern Africa	Africa
Libya	6.5	14	2.6	75	72	77	Northern Africa	Africa
Morocco	32.6	30	2.3	72	70	74	Northern Africa	Africa
Sth Sudan	9.4	101	5.4	52	50	53	Northern Africa	Africa
Sudan	33.5	67	4.2	60	58	62	Northern Africa	Africa
Tunisia	10.8	20	2.1	75	73	77	Northern Africa	Africa
Benin	9.4	81	5.4	56	54	58	Western Africa	Africa
Gambia	1.8	70	4.9	58	57	59	Western Africa	Africa
Ghana	25.5	47	4.2	64	63	65	Western Africa	Africa
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
Serbia	7.1	7	1.3	74	71	77	Southern Europe	Europe
Slovenia	2.1	3	1.5	80	76	83	Southern Europe	Europe
Spain	46.2	3	1.4	82	79	85	Southern Europe	Europe
Australia	22.0	4	1.9	82	80	84	Oceania	Oceania
New Zeal.	4.4	5	2.1	81	79	83	Oceania	Oceania

Data Example 3

tidyr

Tidy?

country	pop2012	imr	tfr	le	leM	leF	region	area
Algeria	37.4	24	2.9	73	72	75	Northern Africa	Africa
Egypt	82.3	24	2.9	72	70	74	Northern Africa	Africa
Libya	6.5	14	2.6	75	72	77	Northern Africa	Africa
Morocco	32.6	30	2.3	72	70	74	Northern Africa	Africa
Sth Sudan	9.4	101	5.4	52	50	53	Northern Africa	Africa
Sudan	33.5	67	4.2	60	58	62	Northern Africa	Africa
Tunisia	10.8	20	2.1	75	73	77	Northern Africa	Africa
Benin	9.4	81	5.4	56	54	58	Western Africa	Africa
Gambia	1.8	70	4.9	58	57	59	Western Africa	Africa
Ghana	25.5	47	4.2	64	63	65	Western Africa	Africa
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
Serbia	7.1	7	1.3	74	71	77	Southern Europe	Europe
Slovenia	2.1	3	1.5	80	76	83	Southern Europe	Europe
Spain	46.2	3	1.4	82	79	85	Southern Europe	Europe
Australia	22.0	4	1.9	82	80	84	Oceania	Oceania
New Zeal.	4.4	5	2.1	81	79	83	Oceania	Oceania

Column name contains **value**: 2012

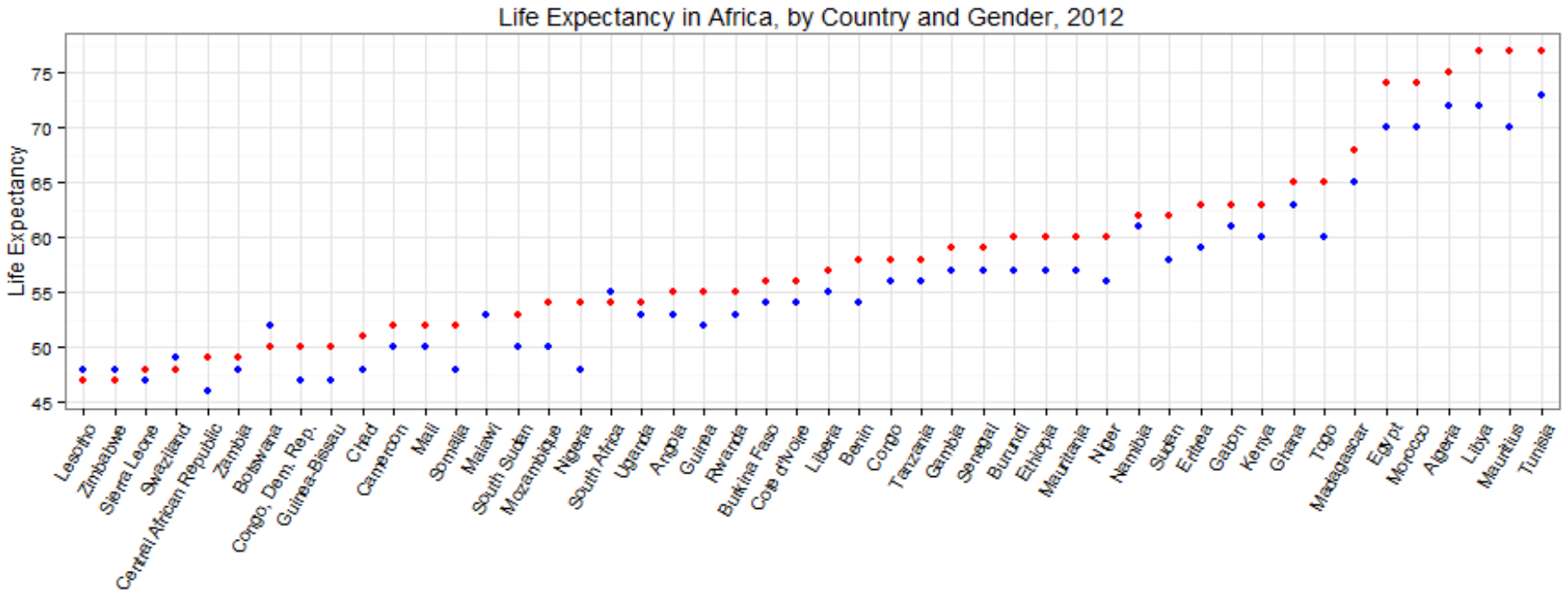
Column name contains **values**: M, F ... use a table with unit of observation country-year-sex

Data Example 3

tidyr

Using ggplot2 to display gender-specific life-expectancy for Africa (from data on previous slide)

```
p <- ggplot(data=filter(w,area=="Africa"),
     aes(x=reorder(factor(country),leF),y=leF))
p + geom_point(color="red") +
    geom_point(aes(y=leM), color="blue")
```



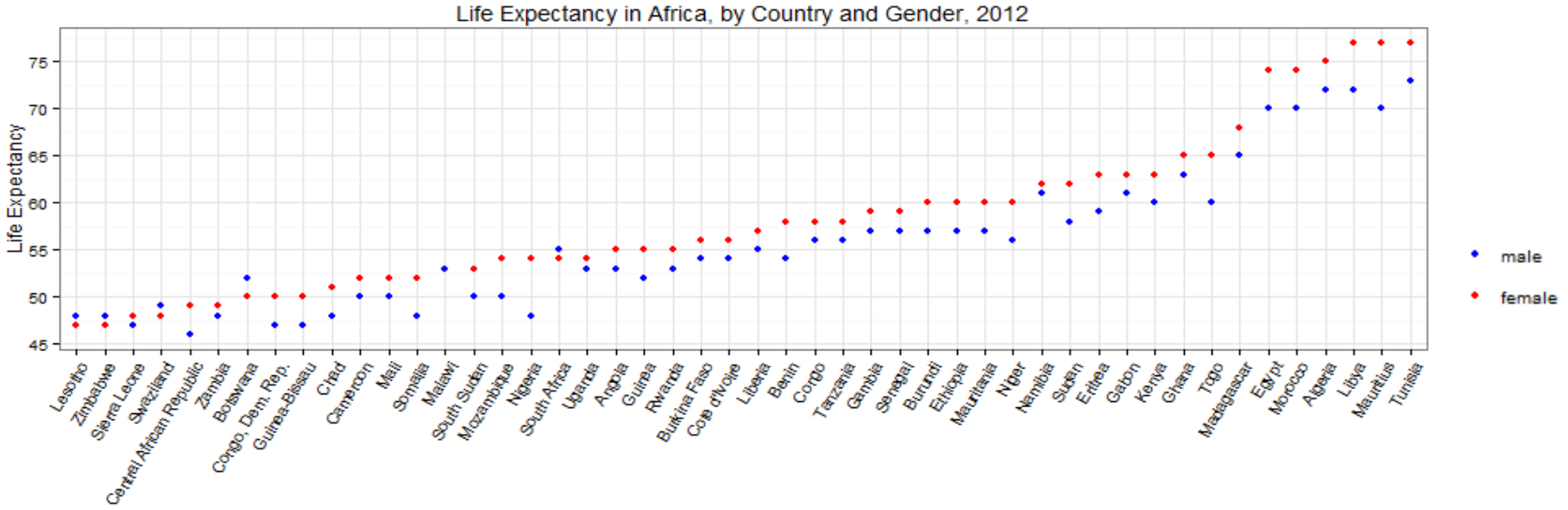
Data Example 3

tidyr

country	year	sex	le
Algeria	2012	male	72
Algeria	2012	female	75
.	.	.	.
.	.	.	.
Zambia	2012	male	48
Zambia	2012	female	49
Zimbabwe	2012	male	48
Zimbabwe	2012	female	47

Using ggplot2 to display gender-specific life-expectancy

```
p <- ggplot(data=filter(w,area=="Africa"),
     aes(x=reorder(factor(country),le), y=le,
     color=sex))
p + geom_point()
```



Data Example 4

tidyr

Tidy?

country	imr	tfr	le	region
Algeria	24	2.9	73	Northern Africa
Egypt	24	2.9	72	Northern Africa
Benin	81	5.4	56	Western Africa
Burkina Faso	65	6.0	55	Western Africa
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Albania	18	1.4	75	Southern Europe
Bosnia-Herz.	5	1.2	76	Southern Europe
Croatia	4	1.5	77	Southern Europe
Greece	4	1.5	80	Southern Europe
Italy	3	1.4	82	Southern Europe

Data Example 4

Tidy?

country	imr	tfr	le	region
Algeria	24	2.9	73	Northern Africa
Egypt	24	2.9	72	Northern Africa
Benin	81	5.4	56	Western Africa
Burkina Faso	65	6.0	55	Western Africa
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Albania	18	1.4	75	Southern Europe
Bosnia-Herz.	5	1.2	76	Southern Europe
Croatia	4	1.5	77	Southern Europe
Greece	4	1.5	80	Southern Europe
Italy	3	1.4	82	Southern Europe

Are **multiple** variables stored in **one column**?

Should there be a region column (Northern, Western, Southern, ...) and a separate continent column (Africa, Europe, ...)?

Data Example 5

tidyr

Tidy?

country	continent	lifeExp	pop	gdpPercap
Afghanistan	Asia	41.674	16317921	649.3414
Albania	Europe	71.581	3326498	2497.4379
Algeria	Africa	67.744	26298373	5023.2166
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Yemen, Rep.	Asia	55.599	13367997	1879.4967
Zambia	Africa	46.100	8381163	1210.8846
Zimbabwe	Africa	60.377	10704340	693.4208

file: world_data_1992.csv

country	continent	lifeExp	pop	gdpPercap
Afghanistan	Asia	42.129	25268405	726.7341
Albania	Europe	75.651	3508512	4604.2117
Algeria	Africa	70.994	31287142	5288.0404
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Yemen, Rep.	Asia	60.308	18701257	2234.8208
Zambia	Africa	39.193	10595811	1071.6139
Zimbabwe	Africa	39.989	11926563	672.0386

file: world_data_2002.csv

Data Example 5

tidyr

Tidy?

country	continent	lifeExp	pop	gdpPercap
Afghanistan	Asia	41.674	16317921	649.3414
Albania	Europe	71.581	3326498	2497.4379
Algeria	Africa	67.744	26298373	5023.2166
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Yemen, Rep.	Asia	55.599	13367997	1879.4967
Zambia	Africa	46.100	8381163	1210.8846
Zimbabwe	Africa	60.377	10704340	693.4208

file: world_data_1992.csv

country	continent	lifeExp	pop	gdpPercap
Afghanistan	Asia	42.129	25268405	726.7341
Albania	Europe	75.651	3508512	4604.2117
Algeria	Africa	70.994	31287142	5288.0404
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Yemen, Rep.	Asia	60.308	18701257	2234.8208
Zambia	Africa	39.193	10595811	1071.6139
Zimbabwe	Africa	39.989	11926563	672.0386

file: world_data_2002.csv

The same observational unit (country-year) is stored in multiple files ...
there is a hidden variable value stored in files names.

Tidy Data Summary

tidyr

Tidy data sets

- Each variable is stored in its own column; column headers do not contain values.

and

- Each observation is stored in its own row.

and

- Each cell is a single value.

and

- Each "type of observational unit" is stored in a single table.

Messy data sets

- Multiple variables are stored in one column.
- Column headers contain values.
- Variables are stored in both rows and columns
- Multiple types of observational units are stored in the same table
- Single observational unit is stored in multiple tables

Install and Load Packages, Read Data

```
install.packages("tidyr")
install.packages("dplyr")
install.packages("ggplot2")

library("tidyr")
library("dplyr")
library("ggplot2")

usle <- read.csv(file="uslifeexp.csv", head=TRUE, stringsAsFactors=FALSE)
usle
```

	year	le	le_male	le_female	le_w	le_wmale	le_wfemale	le_b	le_bmale	le_bfemale
1	1900	47.3	46.3	48.3	47.6	46.6	48.7	33.0	32.5	33.5
2	1901	49.1	47.6	50.6	49.4	48.0	51.0	33.7	32.2	35.3
3	1902	51.5	49.8	53.4	51.9	50.2	53.8	34.6	32.9	36.4
4	1903	50.5	49.1	52.0	50.9	49.5	52.5	33.1	31.7	34.6
.	.	.	.	.	.	.	.	.	.	.
97	1996	76.1	73.1	79.1	76.8	73.9	79.7	70.2	66.1	74.2
98	1997	76.5	73.6	79.4	77.2	74.3	79.9	71.1	67.2	74.7
99	1998	76.7	73.8	79.5	77.3	74.5	80.0	71.3	67.6	74.8
100	1999	76.7	73.9	79.4	77.3	74.6	79.9	71.4	67.8	74.7

Source: National Vital Statistics Report, Vol. 50, No. 6, 21mar2000

Reshape Data

```
pivot_longer(usle, cols= c(le_male, le_female), names_to="sex", values_to="lifeexp")
```

	year	le	le_w	le_wmale	le_wfemale	le_b	le_bmale	le_bfemale	sex	lifeexp
1	1900	47.3	47.6	46.6	48.7	33.0	32.5	33.5	le_male	46.3
2	1901	49.1	49.4	48.0	51.0	33.7	32.2	35.3	le_male	47.6
3	1902	51.5	51.9	50.2	53.8	34.6	32.9	36.4	le_male	49.8
4	1903	50.5	50.9	49.5	52.5	33.1	31.7	34.6	le_male	49.1
.	.	.	.	.	.	.	.	.	.	.
197	1996	76.1	76.8	73.9	79.7	70.2	66.1	74.2	le_female	79.1
198	1997	76.5	77.2	74.3	79.9	71.1	67.2	74.7	le_female	79.4
199	1998	76.7	77.3	74.5	80.0	71.3	67.6	74.8	le_female	79.5
200	1999	76.7	77.3	74.6	79.9	71.4	67.8	74.7	le_female	79.4

```
# "pipe operator" ... think of "then"
select(usle, year, le_male, le_female) %>%
pivot_longer(cols= c(le_male, le_female), names_to="sex", values_to="lifeexp") %>%
arrange(year)
```

	year	sex	lifeexp
1	1900	le_male	46.3
2	1900	le_female	48.3
3	1901	le_male	47.6
4	1901	le_female	50.6
.	.	.	.
197	1998	le_male	73.8
198	1998	le_female	79.5
199	1999	le_male	73.9
200	1999	le_female	79.4

Rename and Reshape Data

```
select(usle, year, le_male, le_female) %>%
rename(male = le_male, female = le_female) %>%
pivot_longer(cols=c(male, female),names_to="sex", values_to="lifeexp”) %>%
arrange(year)
```

	year	sex	lifeexp
1	1900	male	46.3
2	1900	female	48.3
3	1901	male	47.6
4	1901	female	50.6
.	.	.	.
.	.	.	.
.	.	.	.
197	1998	male	73.8
198	1998	female	79.5
199	1999	male	73.9
200	1999	female	79.4

Exercises

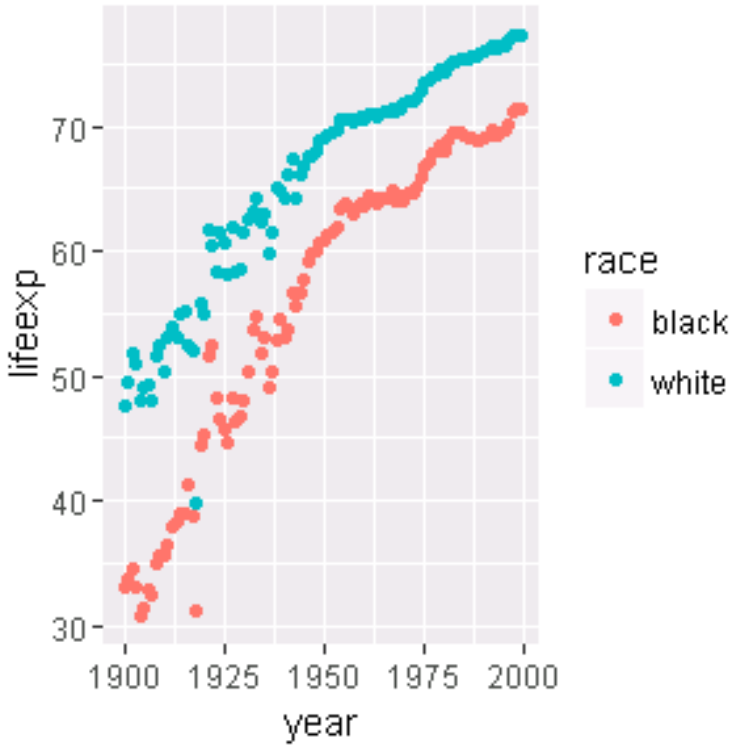
Exercise 1: Display tidy data that shows life expectancy by race (black, white)

```
select(usle, year, black = le_b, white = le_w) %>%
 rename(black = le_b, white = le_w) %>%
pivot_longer(cols = c(black, white), names_to="race", values_to="lifeexp") %>%
  arrange(year)
```

Exercises

Exercise 2: display a ggplot2 graph that shows life expectancy by race (black, white)

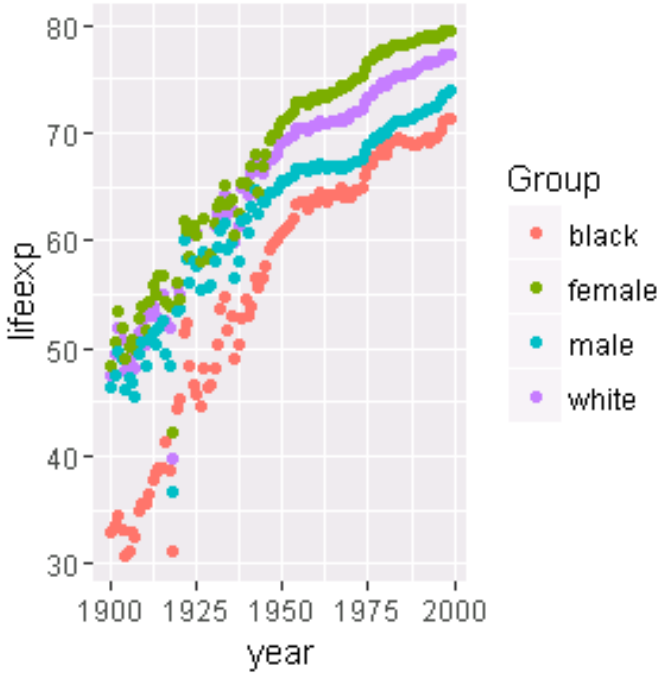
```
select(usle, year, le_b, le_w) %>%
rename(black = le_b, white = le_w) %>%
pivot_longer(cols = c(black, white), names_to="race", values_to="lifeexp") %>%
  arrange(year) %>%
  ggplot(aes(year, lifeexp, color= race)) + geom_point()
```



Reshape Data

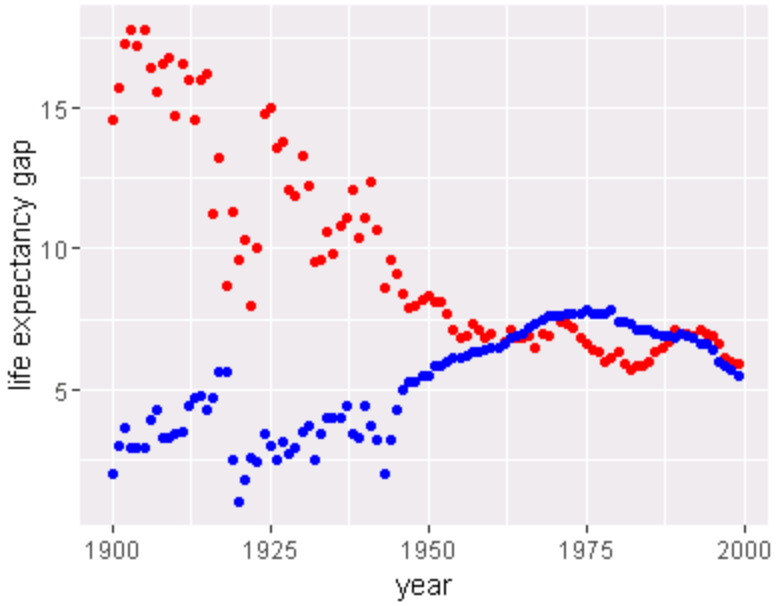
Exercise 3: display life expectancy for black, white, male, female on same graph

```
mfle <- select(usle, year, le_male, le_female) %>%
  rename(male = le_male, female = le_female) %>%
  pivot_longer(cols = c(male,female), names_to="sex", values_to="lifeexp") %>%
  arrange(year)

select(usle, year, le_b, le_w) %>% rename(black = le_b, white = le_w) %>%
  pivot_longer(cols = c(black, white), names_to="race", values_to="lifeexp") %>%
  arrange(year) %>%
  ggplot(aes(year, lifeexp, color= race)) +
  geom_point() +
  geom_point(data = mfle, aes(color = sex )) +
  scale_color_discrete(name="Group")
```



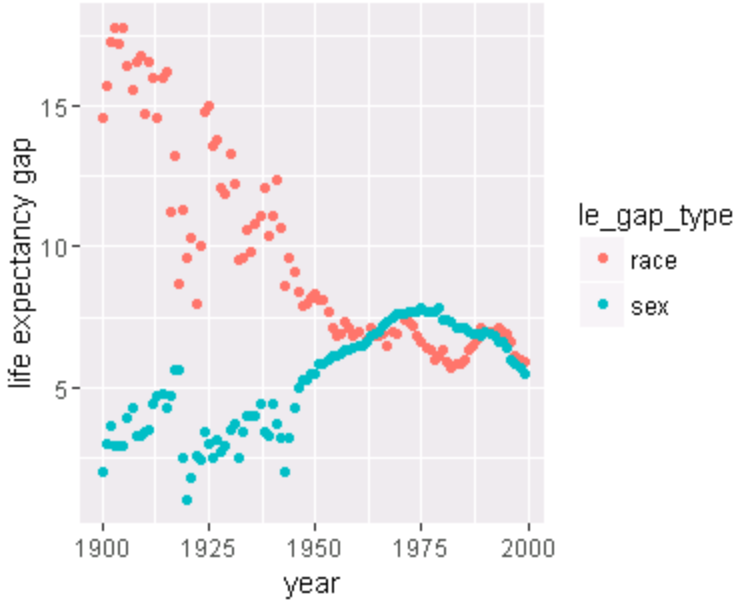
Add New Columns

```
# get life expectancy gap between male and female and also between black and white

mutate(usle, race_le_gap = le_w - le_b, sex_le_gap = le_female - le_male) %>%
  select(year, race_le_gap, sex_le_gap)

# are above data tidy?

mutate(usle, race_le_gap = le_w - le_b, sex_le_gap = le_female - le_male) %>%
  select(year, race_le_gap, sex_le_gap) %>%
  ggplot(aes(year, race_le_gap)) +
  geom_point(color="red") +
  geom_point(aes(y=sex_le_gap), color="blue")  +
  ylab("life expectancy gap")
```



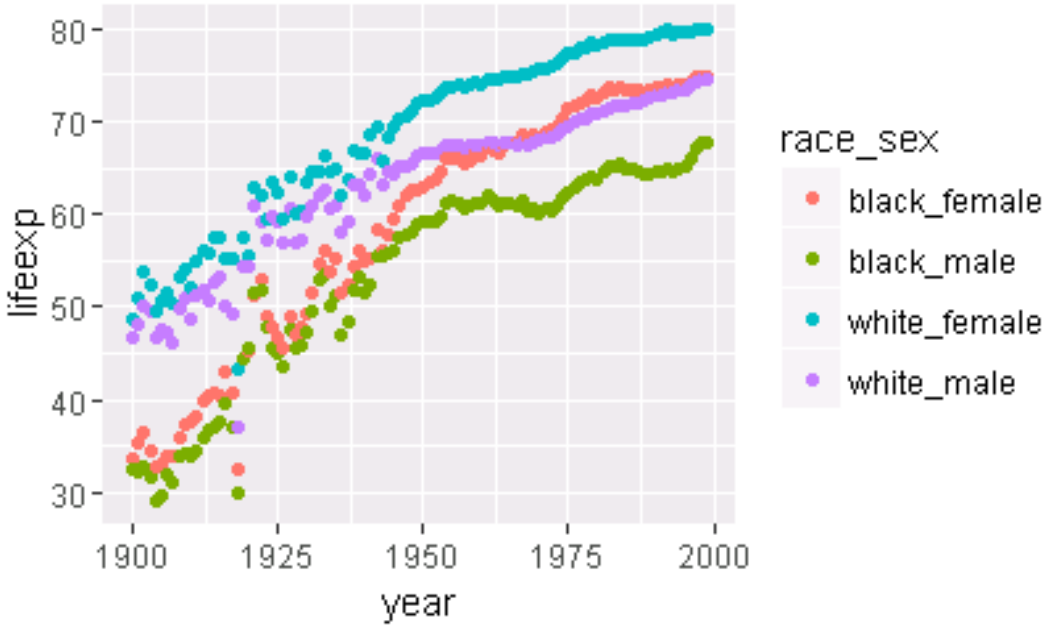
Reshape Data

```
# how to tidy these data?

mutate(usle, race = le_w - le_b, sex = le_female - le_male) %>%
  select(year, race, sex) %>%
 pivot_longer(cols = c(race,sex), names_to="le_gap_type", values_to="le_gap_years")

# use the tidy data to draw graph

mutate(usle, race = le_w - le_b, sex = le_female - le_male) %>%
  select(year, race, sex) %>%
 pivot_longer(cols = c(race,sex), names_to="le_gap_type", values_to="le_gap_years") %>%
  ggplot(aes(year, le_gap_years, color = le_gap_type)) +
  geom_point()+
  ylab("life expectancy gap")
```



Reshape Data

```
# Display tidy data where unit of observation is year-race-sex

select(usle, year, le_wmale, le_wfemale, le_bmale, le_bfemale) %>%
rename(white_male = le_wmale, white_female = le_wfemale, black_male = le_bmale,
       black_female = le_bfemale) %>%
pivot_longer(cols=c(white_male, white_female, black_male, black_female),
       names_to="race_sex", values_to="lifeexp") %>%
arrange(year, race_sex) %>%
ggplot(aes(year, lifeexp, color = race_sex)) +
geom_point()
```



Store Each Variable in its Own Column

```
# Reminder - tidy data: each variable is saved in ITS OWN column

select(usle, year, le_wmale, le_wfemale, le_bmale, le_bfemale) %>%
rename(white_male = le_wmale, white_female = le_wfemale, black_male = le_bmale,
          black_female = le_bfemale) %>%
pivot_longer(cols=c(white_male, white_female, black_male, black_female),
          names_to="racesex", values_to="lifeexp") %>%
arrange(year, racesex) %>%
separate(racesex, c("race", "sex"), sep = "_")
```

```
     year  race    sex lifeexp
 1   1900 black female    33.5
 2   1900 black   male    32.5
 3   1900 white female    48.7
 4   1900 white   male    46.6
 .     .    .       .       .
 .     .    .       .       .
397 1999 black female    74.7
398 1999 black   male    67.8
399 1999 white female    79.9
400 1999 white   male    74.6
```

Combine Multiple Columns

```
# tidyr  function that does the opposite of separate:  unite

tidy_le <- select(usle, year, le_wmale, le_wfemale, le_bmale, le_bfemale) %>%
rename(white_male = le_wmale, white_female = le_wfemale, black_male = le_bmale,
          black_female = le_bfemale) %>%
pivot_longer(cols=c(white_male, white_female, black_male, black_female),
             names_to="race_sex", values_to="lifeexp") %>%
arrange(year, race_sex)  %>%
separate(racesex, c("race", "sex"), sep = "_")

unite(tidy_le, "racesex", c(race, sex), sep = "_")
```

	year	race_sex	lifeexp
1	1900	black_female	33.5
2	1900	black_male	32.5
3	1900	white_female	48.7
4	1900	white_male	46.6
.	.	.	.
.	.	.	.
397	1999	black_female	74.7
398	1999	black_male	67.8
399	1999	white_female	79.9
400	1999	white_male	74.6

Reverse of pivot_longer(): pivot_wider()

```
# tidyr  function opposite of pivot_longer:  pivit_wider  (rows -> columns;  “long” to “wide”)

unite(tidy_le, "racesex", c(race, sex), sep = "_") %>%
  pivot_wider(names_from="racesex", values_from="lifeexp")
```

```
unite(tidy_le, "race_sex", c(race, sex), sep = "_") %>%
 pivot_wider(names_from="racesex", values_from="lifeexp")  %>%
 rename(le_bfemale = black_female, le_bmale = black_male, le_wfemale = white_female,
            le_wmale = white_male)
```

Older commands :

```
spread(key="racesex", value="lifeexp")  ---> pivot_wider() predecessor

gather(usle, key="sex", value="lifeexp", le_male, le_female) ---> pivot_longer() predecessor
```

Exercise

EXERCISE: list data in tidy format with group as one variable
(taking values: all, male, female, black, white, white_male, white_female, etc.)
and life_expectancy as the other variable.

	year	group	life_expectancy
1	1900	all	47.3
2	1900	black	33.0
3	1900	black_female	33.5
4	1900	black_male	32.5
5	1900	female	48.3
6	1900	male	46.3
7	1900	white	47.6
8	1900	white_female	48.7
9	1900	white_male	46.6
10	1901	all	49.1
11	1901	black	33.7
12	1901	black_female	35.3
13	1901	black_male	32.2
14	1901	female	50.6
15	1901	male	47.6
16	1901	white	49.4
17	1901	white_female	51.0
18	1901	white_male	48.0
.	..	.	.

Exercise Solution

```
# EXERCISE: list data in tidy format with group as one variable
# (taking values: all, male, female, black, white, white_male, white_female, etc.)
# and life_expectancy as the other variable.


rename(usle, all=le, male=le_male, female=le_female, black=le_b, white=le_w,
                white_male=le_wmale, white_female=le_wfemale, black_male=le_bmale,
                black_female=le_bfemale) %>%
pivot_longer(cols=c(2:10), names_to = "group", values_to = "life_expectancy") %>%
arrange(year, group)
```

Questions?


dplyr
dplyr.tidyverse.org

Basic **dplyr** Principles

consistent with tidyr philosophy

input: data frame
output: data frame, tibble

input data frame is never modified in place…
may want to save results in a new data frame

commands are optimized for
- clarity (clear, clean syntax)
- computational time (written in C++)

dplyr commands: Verbs

filter()	subset observations (rows)
arrange()	order observations (rows)
select()	subset variables (columns)
rename()	change name of variables (columns)
mutate()	add new variables (columns)
group_by()	partition observation into groups based on variable values
summarise()	collapse each group into a single row of values

Load gapminder data

```
# read-in gapminder data and check structure
library(gapminder)
str(gapminder)

# tibble: improved data.frame for which dplyr provides nice methods for high-level inspection
# these methods do something sensible for datasets with many observations and/or variables

gdf <- as.data.frame(gapminder)
str(gdf)

gtdf <- as_tibble(gdf)
str(gtdf)
# high-level inspection of tibble
glimpse(gapminder)
```

Rows: 1,704
Columns: 6
$ country <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan~
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Europe, Europe,~
$ year <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002, 2007, 1952, 1957, 196~
$ lifeExp <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.822, 41.674, 41.763, 42.129,~
$ pop <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12881816, 13867957, 16317921,~
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, 978.0114, 852.3959, 649.341~

```
View(gapminder) # note capital V
```

gapminder Data

	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	Asia	1957	30.332	9240934	820.8530
3	Afghanistan	Asia	1962	31.997	10267083	853.1007
4	Afghanistan	Asia	1967	34.020	11537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811
6	Afghanistan	Asia	1977	38.438	14880372	786.1134
7	Afghanistan	Asia	1982	39.854	12881816	978.0114
8	Afghanistan	Asia	1987	40.822	13867957	852.3959
9	Afghanistan	Asia	1992	41.674	16317921	649.3414
10	Afghanistan	Asia	1997	41.763	22227415	635.3414
11	Afghanistan	Asia	2002	42.129	25268405	726.7341
12	Afghanistan	Asia	2007	43.828	31889923	974.5803
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
1693	Zimbabwe	Africa	1952	48.451	3080907	406.8841
1694	Zimbabwe	Africa	1957	50.469	3646340	518.7643
1695	Zimbabwe	Africa	1962	52.358	4277736	527.2722
1696	Zimbabwe	Africa	1967	53.995	4995432	569.7951
1697	Zimbabwe	Africa	1972	55.635	5861135	799.3622
1698	Zimbabwe	Africa	1977	57.674	6642107	685.5877
1699	Zimbabwe	Africa	1982	60.363	7636524	788.8550
1700	Zimbabwe	Africa	1987	62.351	9216418	706.1573
1701	Zimbabwe	Africa	1992	60.377	10704340	693.4208
1702	Zimbabwe	Africa	1997	46.809	11404948	792.4500
1703	Zimbabwe	Africa	2002	39.989	11926563	672.0386
1704	Zimbabwe	Africa	2007	43.487	12311143	469.7093

Subset Observations


```
filter(gapminder, country == "United States")

filter(gapminder, lifeExp < 30)

filter(gapminder, pop < 1000000)

filter(gapminder, pop < 1000000, year == 2007)

filter(gapminder, pop < 1000000 & year == 2007)

filter(gapminder, country == "United States" | country == "Canada", year > 2000)

filter(gapminder, country %in% c("United States", "Canada"), year > 2000)

distinct(gapminder, country)
View(distinct(gapminder))

distinct(gapminder, country) %>% View()
```

Subset Columns


```
select(gapminder, country, continent)

country_continent <- select(gapminder, country, continent) %>% distinct()
country_continent

select(gapminder, -continent) # "-" means not ... gives TIDIER data set
tgap <- select(gapminder, -continent)

# how to combine tgap and country_continent to summarize values by continent???
# later will use a "join" function to combine

select(gapminder, year, country, continent, lifeExp) # select and re-order columns

select(gapminder, starts_with("co"))
select(gapminder, country:lifeExp) # range
```

Exercises

```
# EXERCISE:
# list all countries showing only life expectancy for 2007
```

```
# EXTRA CREDIT EXERCISE:
# list all countries showing only life expectancy for 2007
# with life expectancy variable named le (rather than lifeExp)
```

Exercises

```
# EXERCISE:
# list all countries showing only life expectancy for 2007

filter(gapminder, year == 2007) %>%
  select(country, year, lifeExp)
```

```
# EXTRA CREDIT EXERCISE:
# list all countries showing only life expectancy for 2007
# with life expectancy variable named le (rather than lifeExp)

filter(gapminder, year == 2007) %>%
  select(country, year, lifeExp) %>%
  rename(le = lifeExp)
```

Order Rows


```
arrange(gapminder, year)

rename(gapminder, le = lifeExp) %>% filter(year == 2007) %>%
  select(country, year, le) %>% arrange(le)

rename(gapminder, le = lifeExp) %>% filter(year == 2007) %>%
  select(country, year, le) %>% arrange(desc(le)) # order by descending le

# to list all rows, can use gapminder as a data frame
rename(as.data.frame(gapminder), le = lifeExp) %>%
  filter(year == 2007) %>%
  select(country, year, le) %>% arrange(desc(le))

# list 5 countries with highest life expectancy values in 2007,
# show country, year, and le
rename(gapminder, le = lifeExp) %>%
  filter(year == 2007) %>%
  select(country, year, le) %>%
  arrange(desc(le)) %>%
  slice(1:5) # filter rows a second time
```

Order Rows


```
# list 10 countries with lowest life expectancy values,
# with lowest value at the top;
# show country, year, and le
```

```
rename(gapminder, le=lifeExp) %>%
  filter(year == 2007) %>%
  select(country, year, le) %>%
  arrange(le) %>% # low to high
  slice(1:10) # filter rows a second time, by position
```

Construct New Columns

```
mutate(gapminder, popMil = round(pop / 1000000, 1), le = round(lifeExp, 0))
# A tibble: 1,704 x 8
   country     continent  year lifeExp      pop gdpPercap popMil    le
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>  <dbl> <dbl>
 1 Afghanistan Asia       1952    28.8  8425333       779.    8.4    29
 2 Afghanistan Asia       1957    30.3  9240934       821.    9.2    30
 3 Afghanistan Asia       1962    32.0 10267083       853.   10.3    32
 4 Afghanistan Asia       1967    34.0 11537966       836.   11.5    34
 5 Afghanistan Asia       1972    36.1 13079460       740.   13.1    36
       ...     ...         ...     ...   ...           ...    ...     ..
```

```
transmute(gapminder, country, y = year,
          popMil = round(pop / 1000000, 1), le = round(lifeExp, 0)) %>%
  arrange(y, country)
# A tibble: 1,704 x 4
   country         y popMil    le
   <fct>       <int>  <dbl> <dbl>
 1 Afghanistan  1952    8.4    29
 2 Albania      1952    1.3    55
 3 Algeria      1952    9.3    43
 4 Angola       1952    4.2    30
 5 Argentina    1952   17.9    62
   ...           ...     ...    ..
```

Vector Functions


```
# window functions take a vector of n values and return n values
# types of vector functions:
# - ranking and ordering functions
# - cumulative aggregates
# - access to previous and next values

# assign lowest rank to lowest life expectancy
filter(gapminder, year == 2007) %>%
mutate(le_rank = dense_rank(lifeExp)) %>%
select(country, continent, year, lifeExp, le_rank) %>%
arrange(le_rank)

# assign lowest rank to highest life expectancy
filter(gapminder, year == 2007) %>%
mutate(le_rank = dense_rank(-lifeExp)) %>%
select(country, continent, year, lifeExp, le_rank) %>%
arrange(le_rank)
```

Vector Functions – Cumulative Sum

```
filter(gapminder, year == 1952) %>%
  arrange(continent, country) %>%
  mutate(popMil = round(pop / 1000000, 1)) %>%
  mutate(cumpopMil = cumsum(popMil)) %>% View()
```

```
filter(gapminder, year == 2007) %>%
  arrange(continent, country) %>%
  mutate(popMil = round(pop / 1000000, 1)) %>%
  mutate(cumpopMil = cumsum(popMil)) %>% View()
```

Group Data:
Construct New Column Values By Group

```
filter(gapminder, year == 2007) %>%
  mutate(popMil = round(pop / 1000000, 1)) %>%
  arrange(continent, popMil) %>%
  group_by(continent) %>%
  mutate(cumpopMil = cumsum(popMil)) %>% View()
```

	country	continent	year	lifeExp	pop	gdpPercap	popMil	cumpopMil
	<fct>	<fct>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>
1	Sao Tome and Principe	Africa	2007	65.528	199579	1598.4351	0.2	0.2
2	Djibouti	Africa	2007	54.791	496374	2082.4816	0.5	0.7
3	Equatorial Guinea	Africa	2007	51.579	551201	12154.0897	0.6	1.3
4	Comoros	Africa	2007	65.152	710960	986.1479	0.7	2.0
5	Reunion	Africa	2007	76.442	798094	7670.1226	0.8	2.8
6	Swaziland	Africa	2007	39.613	1133066	4513.4806	1.1	3.9
7	Mauritius	Africa	2007	72.801	1250882	10956.9911	1.3	5.2
	...	...	...	...	...	...	..	..
52	Nigeria	Africa	2007	46.859	135031164	2013.9773	135.0	929.6
53	Trinidad and Tobago	Americas	2007	69.819	1056608	18008.5092	1.1	1.1
54	Jamaica	Americas	2007	72.567	2780132	7320.8803	2.8	3.9

```
select(gapminder, country, year, pop) %>%
  group_by(country) %>%
  mutate(pop_lag = lag(pop), pop_chg = pop - pop_lag,
       pop_pctchg = round(pop_chg/pop_lag * 100, 1)) %>% View()
```

Group Data

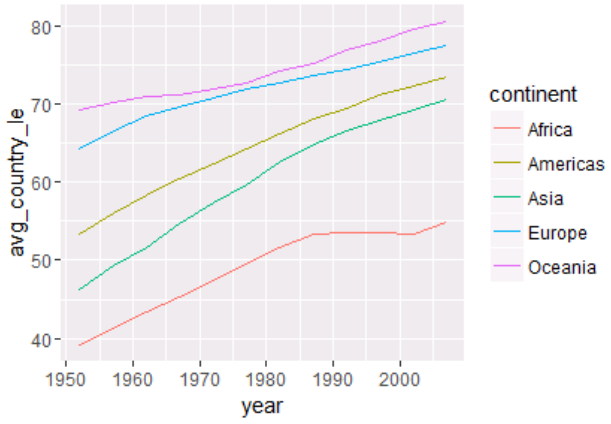
```
# list only rows that experienced a population decline during the previous 5-years
# show country, year, pop, pop_chg, pop_pctchg



select(gapminder, country, year, pop) %>%
  group_by(country) %>%
  mutate(pop_lag = lag(pop), pop_chg = pop - pop_lag,
         pop_pctchg = round(pop_chg/pop_lag * 100, 1)) %>%
  filter(pop_chg < 0) %>% View()
```

Summarize Data

```
# use summarise() with a summary function to change the unit of observation
# summary functions take a vector of values and return a single value
# often used with group_by()

filter(gapminder, year == 2007) %>%
  summarise(year = mean(year), ncountries = n(),
  avg_country_le = mean(lifeExp), sd_country_le = sd(lifeExp))


filter(gapminder, year == 2007) %>%
  group_by(continent) %>%
  summarise(avg_country_le = mean(lifeExp))


filter(gapminder, year == 2007) %>%
  group_by(continent) %>%
  summarise(year = mean(year), ncountries = n(),
  avg_country_le = mean(lifeExp), sd_country_le = sd(lifeExp))
```

Summarize Data

```
# Show data by continent and years 1952 AND 2007
# list number of countries, avg_country_le, and sd_country_le
filter(gapminder, year == 1952 | year == 2007) %>%
  group_by(continent, year) %>%
  summarise(ncountries = n(), avg_country_le = mean(lifeExp), sd_country_le =
sd(lifeExp))

# By continent and year, show ncountries, avg_country_le, sd_country_le for all 12 years
of data
group_by(gapminder, continent, year) %>%
  summarise(ncountries = n(), avg_country_le = mean(lifeExp), sd_country_le =
sd(lifeExp)) %>%
  View()

# Make a simple graph that shows avg_country_le over time,
# for each continent
group_by(gapminder, continent, year) %>%
  summarise(avg_country_le = mean(lifeExp)) %>%
  ggplot(aes(x = year, y = avg_country_le, color = continent)) +
  geom_line()
```



summarise() “Peels off” group_by()

```
# how many continents each country
# has belonged to over time
group_by(gapminder, country) %>%
  summarise(n_continents = n_distinct(continent))
```

country	n_continents
<fct>	<int>
1 Afghanistan	1
2 Albania	1
3 Algeria	1
4 Angola	1
5 Argentina	1
6 Australia	1
7 Austria	1
8 Bahrain	1
9 Bangladesh	1
10 Belgium	1

```
# each summarise "peels off" one level of group_by()
group_by(gapminder, country) %>%
  summarise(n_continents = n_distinct(continent)) %>%
  summarise(avg_n_continents = mean(n_continents))
```

avg_n_continents
<dbl>
1

summarise() “Peels off” group_by()

```
group_by(gapminder, continent, country) %>%
  summarise(avg_le_cc = mean(lifeExp))
```

```
group_by(gapminder, continent, country) %>%
  summarise(avg_le_cc = mean(lifeExp))  %>%
  summarise(avg_le_c =  mean(avg_le_cc))
```

```
group_by(gapminder, continent, country)  %>%
  summarise(avg_le_cc = mean(lifeExp))   %>%
  summarise(avg_le_c  = mean(avg_le_cc)) %>%
  summarise(avg_le    = mean(avg_le_c))
```

Summrize multiple columns

```
# Apply one or more functions to one or more columns.
# Grouping variables are always excluded from modification.

group_by(gapminder, continent, year) %>%
summarise_at(c("lifeExp", "pop"), list(min=min, med=median, max=max)) %>% View()
```

```
group_by(gapminder, continent, year) %>%
  summarise(across(c(lifeExp, pop), list(min=min, median=median, max=max))) %>%
  View()
```

More Summary Functions

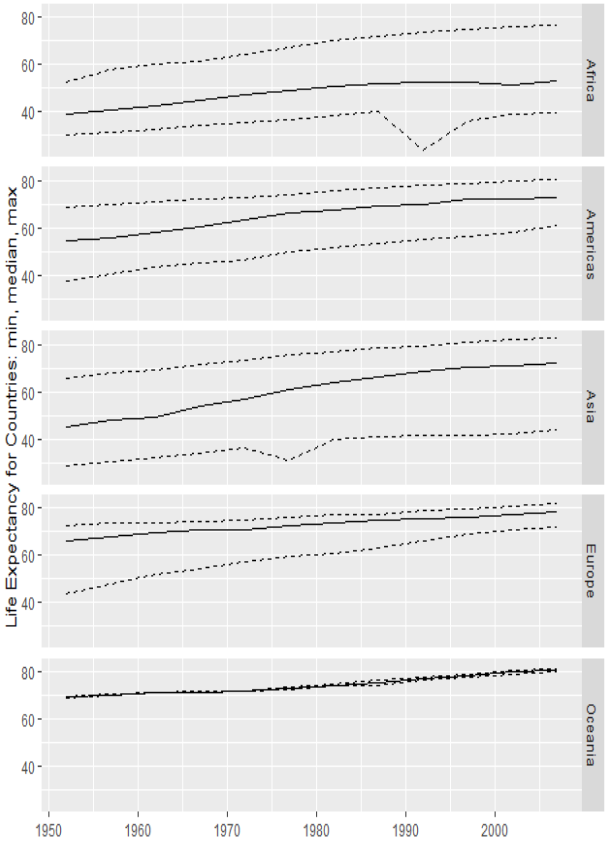
```
group_by(gapminder, country) %>%
  summarise(year = first(year), le = first(lifeExp))
```

```
group_by(gapminder, country) %>%
  summarise(year = last(year), le = last(lifeExp))
```

```
group_by(gapminder, country) %>%
  summarise(year = nth(year, 3), le = nth(lifeExp, 3))
```

Graphing Results of Multiple Summaries

```
# graph min, median, max country life expectancy, by continent

group_by(gapminder, continent, year) %>%
  summarise(across(c(lifeExp, pop),
list(min=min, median=median, max=max))) %>%


  ggplot(aes(x=year, y = lifeExp_median)) + geom_line() +
  geom_line(aes(y = lifeExp_min), linetype = "dashed") +
  geom_line(aes(y = lifeExp_max), linetype = "dashed") +
  facet_grid(continent ~ .) +
  labs(y="Life Expectancy for Countries: min, median, max",
   x="")
```



count() Function

```
# count() function wraps up the combination of group_by() and summarise():

# How many rows for each value of continent?
count(gapminder, continent)

# How many rows for each value of continent and year?
count(gapminder, continent, year) %>% View()


# How many rows for each continent, for years 2002 and 2007?
filter(gapminder, year == 2002 | year == 2007) %>%
  count(continent)
```

dplyr Commands to Combine Data Sets

left_join()

right_join()

inner_join()

full_join()

potentially add variables (columns) to data sets, making them wider

semi_join()

anti_join()

potentially remove observations (rows) from data sets, making them shorter

bind_rows()

add observations (rows) to data sets, making them longer

Combining Data Sets: left_join()

```
# saw above that every country has been associated with just one continent
# so ...
# continent could be in a table where unit of observation is country
# the other variables would be in a table where unit of observation is country-year:

country_continent <- select(gapminder, country, continent) %>% distinct()
country_continent
tgap <- select(gapminder, -continent)  # tidy version of gapminder data
tgap

# HOW TO COMBINE ("join" or "merge") tgap and country_continent?

# join matching rows from second data set to first
left_join(tgap, country_continent, by = "country") %>% View()
```

Combining Data Sets: left_join() and right_join()

```
country_continent_inc <- slice(country_continent, 6:142) # cut out rows 1-5
View(country_continent_inc)

tgap_inc <- slice(tgap, 49:144) # cut out rows 1-48 and rows 145-1704
View(tgap_inc)

# join matching rows from 2nd data set to first
left_join(tgap, country_continent_inc, by = "country") %>% View()

# join matching rows from first data set to 2nd
right_join(tgap, country_continent, by = "country") %>% View()

right_join(tgap, country_continent_inc, by = "country") %>% View()
# country_continent_inc is the driver ... result does not contain first 5 countries
```

Combining Data Sets: inner_join(), full_join, semi_join and anti_join()

```
# join and retain only rows in both data sets
inner_join(tgap_inc, country_continent_inc, by = "country") %>% View()

# join and retain all values, all rows
full_join(tgap_inc, country_continent_inc, by = "country") %>% View()

# retain all rows in first data set that have a match in second data set
# (but don't add columns)
semi_join(tgap_inc, country_continent_inc, by = "country") %>% View()

# retain all rows in first data set that do not have a match in second data set
# (but don't add columns)
anti_join(tgap_inc, country_continent_inc, by = "country") %>% View()
```

Appending Data Sets

TO APPEND ROWS use bind_rows() ... more efficient than rbind()

```
tgap1992 <- filter(tgap, year == 1992) %>% select(-year)
tgap1997 <- filter(tgap, year == 1997) %>% select(-year)
tgap2002 <- filter(tgap, year == 2002) %>% select(-year)
tgap2007 <- filter(tgap, year == 2007) %>% select(-year)

tgap1992
tgap2007

bind_rows(tgap1992, tgap1997, tgap2002, tgap2007) %>% View() # ... OOPS .. not quite right!

bind_rows(tgap1992, tgap1997, tgap2002, tgap2007, .id="id") %>% View()  # ... a bit better!

bind_rows("1992" = tgap1992, "1997" = tgap1997, "2002" = tgap2002, "2007" = tgap2007,
.id="year") %>% View()
```

Resources

Official documentation:

https://tidyr.tidyverse.org/
https://dplyr.tidyverse.org/

https://github.com/tidyverse/tidyr
https://github.com/tidyverse/dplyr

https://cran.r-project.org/web/packages/tidyr/tidyr.pdf
https://cran.r-project.org/web/packages/dplyr/dplyr.pdf

https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

Chapters from the book “R for Data Science” (2nd edition) by Hadley Wickham:

https://r4ds.hadley.nz/data-tidy.html - Data tidying (tidyr)
https://r4ds.hadley.nz/data-transform - Data transformation (dplyr)

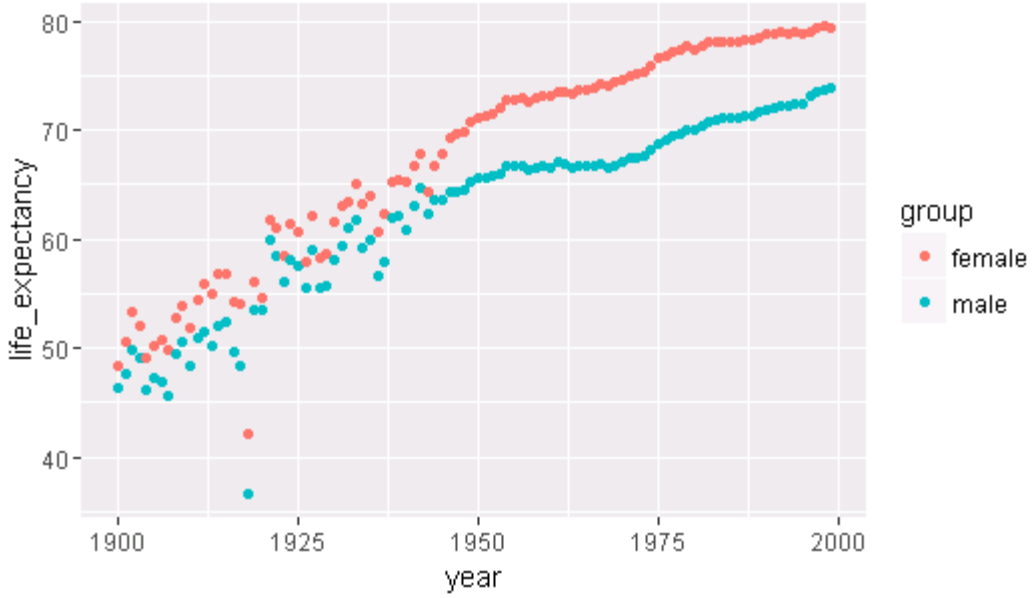
Filter Rows

```
# preview of dplyr
rename(usle, all=le, male=le_male, female=le_female, black=le_b, white=le_w,
                white_male=le_wmale, white_female=le_wfemale, black_male=le_bmale,
               black_female=le_bfemale) %>%
 pivot_longer(cols=c(2:10), names_to = "group", values_to = "life_expectancy") %>%
 arrange(year, group) %>%
 filter(group == "black" | group == "white")
```

	year	group	life_expectancy
1	1900	black	33.0
2	1900	white	47.6
3	1901	black	33.7
4	1901	white	49.4
5	1902	black	34.6
6	1902	white	51.9
7	1903	black	33.1
8	1903	white	50.9
9	1904	black	30.8
10	1904	white	48.0
.	.		.
.	.		.
.	.		.

Tidy Data, Filter Rows and Graph

```
rename(usle, all=le, male=le_male, female=le_female, black=le_b, white=le_w,
                     white_male=le_wmale, white_female=le_wfemale,
                     black_male=le_bmale,  black_female=le_bfemale) %>%
pivot_longer(cols=c(2:10), names_to = "group", values_to = "life_expectancy")) %>%
arrange(year, group) %>%
filter(group == "male" | group == "female") %>%
ggplot(aes(year, life_expectancy, color = group)) + geom_point()
```



Tidy Data, Filter Rows and Graph

```
rename(usle, all=le, male=le_male, female=le_female, black=le_b, white=le_w,
                     white_male=le_wmale, white_female=le_wfemale,
                     black_male=le_bmale, black_female=le_bfemale) %>%
  pivot_longer(cols=c(2:10), names_to = "group", values_to = "life_expectancy") %>%
  arrange(year, group) %>%
  filter(group %in% c("white_male", "white_female", "black_male", "black_female")) %>%
  ggplot(aes(year, life_expectancy, color = group)) + geom_point()
```