Project 1: World Progress

In this project, you'll explore data from [Gapminder.org (http://gapminder.org)](http://gapminder.org), a website dedicated to providing a fact-based view of the world and how it has changed. That site includes several data visualizations and presentations, but also publishes the raw data that we will use in this project to recreate and extend some of their most famous visualizations.

The Gapminder website collects data from many sources and compiles them into tables that describe many countries around the world. All of the data they aggregate are published in the [Systema Globalis (https://github.com/open-numbers/ddf--gapminder--systema_globalis/blob/master/README.md)](https://github.com/open-numbers/ddf--gapminder--systema_globalis/blob/master/README.md). Their goal is "to compile all public statistics; Social, Economic and Environmental; into a comparable total dataset." All data sets in this project are copied directly from the Systema Globalis without any changes.

This project is dedicated to [Hans Rosling (https://en.wikipedia.org/wiki/Hans_Rosling)](https://en.wikipedia.org/wiki/Hans_Rosling) (1948-2017), who championed the use of data to understand and prioritize global development challenges.

Logistics

Deadline. This project is due at 11:59pm on Friday 2/28. Projects will be accepted up to 2 days (48 hours) late; a project submitted less than 24 hours after the deadline will receive 2/3 credit, a project submitted between 24 and 48 hours after the deadline will receive 1/3 credit, and a project submitted 48 hours or more after the deadline will receive no credit. It's **much** better to be early than late, so start working now.

Checkpoint. For full credit, you must also complete the first 8 questions, pass all public autograder tests, and submit to okpy by 11:59pm on Friday 2/21. After you've submitted the checkpoint, you may still change your answers before the project deadline - only your final submission will be graded for correctness. You will have some lab time to work on these questions, but we recommend that you start the project before lab and leave time to finish the checkpoint afterward.

Partners. You may work with one other partner; your partner must be from your assigned lab section. Only one of you is required to submit the project. On okpy.org (http://okpy.org), the person who submits should also designate their partner so that both of you receive credit.

Rules. Don't share your code with anybody but your partner. You are welcome to discuss questions with other students, but don't share the answers. The experience of solving the problems in this project will prepare you for exams (and life). If someone asks you for the answer, resist! Instead, you can demonstrate how you would solve a similar problem.

Support. You are not alone! Come to office hours, post on Piazza, and talk to your classmates. If you want to ask about the details of your solution to a problem, make a private Piazza post and the staff will respond. If you're ever feeling overwhelmed or don't know how to make progress, email your TA or tutor for help. You can find contact information for the staff on the course website (http://data8.org/sp20/staff.html).

Tests. The tests that are given are **not comprehensive** and passing the tests for a question **does not** mean that you answered the question correctly. Tests usually only check that your table has the correct column labels. However, more tests will be applied to verify the correctness of your submission in order to assign your final score, so be careful and check your work! You might want to create your own checks along the way to see if your answers make sense. Additionally, before you submit, make sure that none of your cells take a very long time to run (several minutes).

Free Response Questions: Make sure that you put the answers to the written questions in the indicated cell we provide. Check to make sure that you have a Gradescope (http://gradescope.com) account, which is where the scores to the free response questions will be posted. If you do not, make sure to reach out to your assigned (u)GSI.

Advice. Develop your answers incrementally. To perform a complicated table manipulation, break it up into steps, perform each step on a different line, give a new name to each result, and check that each intermediate result is what you expect. You can add any additional names or functions you want to the provided cells. Make sure that you are using distinct and meaningful variable names throughout the notebook. Along that line, **DO NOT** reuse the variable names that we use when we grade your answers. For example, in Question 1 of the Global Poverty section, we ask you to assign an answer to `latest`. Do not reassign the variable name `latest` to anything else in your notebook, otherwise there is the chance that our tests grade against what `latest` was reassigned to.

You **never** have to use just one line in this project or any others. Use intermediate variables and multiple lines as much as you would like!

To get started, load `datascience`, `numpy`, `plots`, and `ok`.

In [1]:
```
from datascience import *
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

from client.api.notebook import Notebook
ok = Notebook('project1.ok')
```

```
=====================================================================
Assignment: Project 1: World Progress
OK, version v1.14.20
=====================================================================
```

Before continuing the assignment, select "Save and Checkpoint" in the File menu and then execute the submit cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org and flag the correct version. There will be another submit cell at the end of the assignment when you finish!

In [2]:
```
_ = ok.submit()
```

```
Saving notebook...

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-2-cc46ca874451> in <module>
----> 1 _ = ok.submit()

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
submit(self)
     69         messages = {}
     70         self.assignment.set_args(submit=True)
---> 71         if self.save(messages):
     72             return self.run('backup', messages)
     73         else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save(self, messages, delay, attempts)
     78
     79     def save(self, messages, delay=0.5, attempts=3):
---> 80         saved = self.save_notebook()
     81         if not saved:
     82             return None

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save_notebook(self)
    115         # Wait for first .ipynb to save
    116         if ipynbs:
--> 117             if wait_for_save(ipynbs[0]):
    118                 print("Saved '{}'.".format(ipynbs[0]))
    119             else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
wait_for_save(filename, timeout)
    160     Returns True if a save was detected, and False otherwise.
    161     """
--> 162     modification_time = os.path.getmtime(filename)
    163     start_time = time.time()
    164     while time.time() < start_time + timeout:

~/work/teaching/datasp20/lib/python3.6/genericpath.py in getmtime(filename)
     53 def getmtime(filename):
     54     """Return the last modification time of a file, reported by os.stat
()."""
---> 55     return os.stat(filename).st_mtime
     56
     57

FileNotFoundError: [Errno 2] No such file or directory: 'project1.ipynb'
```

1. Global Population Growth

The global population of humans reached 1 billion around 1800, 3 billion around 1960, and 7 billion around 2011. The potential impact of exponential population growth has concerned scientists, economists, and politicians alike.

The UN Population Division estimates that the world population will likely continue to grow throughout the 21st century, but at a slower rate, perhaps reaching 11 billion by 2100. However, the UN does not rule out scenarios of more extreme growth.

(http://www.pewresearch.org/fact-tank/2015/06/08/scientists-more-worried-than-public-about-worlds-growing-population/ft_15-06-04_popcount/)

In this section, we will examine some of the factors that influence population growth and how they are changing around the world.

The first table we will consider is the total population of each country over time. Run the cell below.

In [3]:
```
population = Table.read_table('population.csv')
population.show(3)
```

geo	time	population_total
abw	1800	19286
abw	1801	19286
abw	1802	19286

... (87792 rows omitted)

Note: The population csv file can also be found here (https://github.com/open-numbers/ddf--gapminder--systema_globalis/raw/master/ddf--datapoints--population_total--by--geo--time.csv). The data for this project was downloaded in February 2017.

Bangladesh

In the `population` table, the `geo` column contains three-letter codes established by the International Organization for Standardization (https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO) in the Alpha-3 (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3#Current_codes) standard. We will begin by taking a close look at Bangladesh. Inspect the standard to find the 3-letter code for Bangladesh.

Question 1. Create a table called `b_pop` that has two columns labeled `time` and `population_total`. The first column should contain the years from 1970 through 2015 (including both 1970 and 2015) and the second should contain the population of Bangladesh in each of those years.

```
BEGIN QUESTION
name: q1_1
```

In [4]:
```
b_pop = population.where('geo', 'bgd').where('time', are.between(1970, 2016)).d
rop('geo') # SOLUTION
b_pop
```

Out[4]:

time	population_total
1970	65048701
1971	66417450
1972	67578486
1973	68658472
1974	69837960
1975	71247153
1976	72930206
1977	74848466
1978	76948378
1979	79141947

... (36 rows omitted)

In [5]:
```
# TEST
# Check your column labels and spelling
b_pop.labels == ('time', 'population_total')
```

Out[5]: True

In [6]:
```
# TEST
# Times should range from 1970 through 2015
all(b_pop.sort("time").column("time") == np.arange(1970, 2016))
```

Out[6]: True

In [7]:
```
# HIDDEN TEST
b_pop.sort("time", descending=True).column("population_total").item(0)
```

Out[7]: 160995642

In [8]:
```
# HIDDEN TEST
b_pop.sort("time").column("population_total").item(0)
```

Out[8]: 65048701

Run the following cell to create a table called `b_five` that has the population of Bangladesh every five years. At a glance, it appears that the population of Bangladesh has been growing quickly indeed!

In [9]:
```
b_pop.set_format('population_total', NumberFormatter)

fives = np.arange(1970, 2016, 5) # 1970, 1975, 1980, ...
b_five = b_pop.sort('time').where('time', are.contained_in(fives))
b_five
```

Out[9]:

time	population_total
1970	65,048,701
1975	71,247,153
1980	81,364,176
1985	93,015,182
1990	105,983,136
1995	118,427,768
2000	131,280,739
2005	142,929,979
2010	151,616,777
2015	160,995,642

Question 2. Assign `initial` to an array that contains the population for every five year interval from 1970 to 2010. Then, assign `changed` to an array that contains the population for every five year interval from 1975 to 2015. You should use the `b_five` table to create both arrays, first filtering the table to only contain the relevant years.

We have provided the code below that uses `initial` and `changed` in order to add a column to `b_five` called `annual_growth`. Don't worry about the calculation of the growth rates; run the test below to test your solution.

If you are interested in how we came up with the formula for growth rates, consult the [growth rates (https://www.inferentialthinking.com/chapters/03/2/1/growth)](https://www.inferentialthinking.com/chapters/03/2/1/growth) section of the textbook.

```
BEGIN QUESTION
name: q1_2
```

In [10]:
```
initial = b_five.where("time", are.below(2015)).column('population_total') # SO
LUTION
changed = b_five.where("time", are.above(1970)).column('population_total') # SO
LUTION

b_1970_through_2010 = b_five.where('time', are.below_or_equal_to(2010))
b_five_growth = b_1970_through_2010.with_column('annual_growth', (changed/initi
al)**0.2-1)
b_five_growth.set_format('annual_growth', PercentFormatter)
```

Out[10]:

time	population_total	annual_growth
1970	65,048,701	1.84%
1975	71,247,153	2.69%
1980	81,364,176	2.71%
1985	93,015,182	2.64%
1990	105,983,136	2.25%
1995	118,427,768	2.08%
2000	131,280,739	1.71%
2005	142,929,979	1.19%
2010	151,616,777	1.21%

In [11]:
```
# TEST
first = round(b_five_growth.sort(0).column(2).item(0), 8)
0.005 <= first <= 0.5
```

Out[11]: True

In [12]:
```
# TEST
# Compute the annual exponential growth rate
max(b_five_growth.column(2)) < 0.03
```

Out[12]: True

In [13]:
```
# HIDDEN TEST
# Compute the annual exponential growth rate
np.allclose(np.round(sum(b_five_growth.column(2)), 5), 0.18323)
```

Out[13]: True

In [14]:
```
# HIDDEN TEST
# Table does not have expected values
round(b_five_growth.column(2).item(0), 8) == 0.01837042
```

Out[14]: True

In [15]:
```
# HIDDEN TEST
round(b_five_growth.column(2).item(3), 8) == 0.02644713
```

Out[15]: True

In [16]:
```
# HIDDEN TEST
round(b_five_growth.column(2).item(8), 8) == 0.01207657
```

Out[16]: True

While the population has grown every five years since 1970, the annual growth rate decreased dramatically from 1985 to 2005. Let's look at some other information in order to develop a possible explanation. Run the next cell to load three additional tables of measurements about countries over time.

In [17]:
```
life_expectancy = Table.read_table('life_expectancy.csv')
child_mortality = Table.read_table('child_mortality.csv').relabel(2, 'child_mor
tality_under_5_per_1000_born')
fertility = Table.read_table('fertility.csv')
```

The `life_expectancy` table contains a statistic that is often used to measure how long people live, called *life expectancy at birth*. This number, for a country in a given year, [does not measure how long babies born in that year are expected to live (http://blogs.worldbank.org/opendata/what-does-life-expectancy-birth-really-mean)](http://blogs.worldbank.org/opendata/what-does-life-expectancy-birth-really-mean). Instead, it measures how long someone would live, on average, if the *mortality conditions* in that year persisted throughout their lifetime. These "mortality conditions" describe what fraction of people at each age survived the year. So, it is a way of measuring the proportion of people that are staying alive, aggregated over different age groups in the population.

Run the following cells below to see `life_expectancy` , `child_mortality` , and `fertility` . Refer back to these tables as they will be helpful for answering further questions!

In [18]:
```
life_expectancy
```

Out[18]:

geo	time	life_expectancy_years
afg	1800	28.21
afg	1801	28.2
afg	1802	28.19
afg	1803	28.18
afg	1804	28.17
afg	1805	28.16
afg	1806	28.15
afg	1807	28.14
afg	1808	28.13
afg	1809	28.12

... (43847 rows omitted)

In [19]:
```
child_mortality
```

Out[19]:

geo	time	child_mortality_under_5_per_1000_born
afg	1800	468.6
afg	1801	468.6
afg	1802	468.6
afg	1803	468.6
afg	1804	468.6
afg	1805	468.6
afg	1806	470
afg	1807	470
afg	1808	470
afg	1809	470

... (40746 rows omitted)

In [20]:
```
fertility
```

Out[20]:

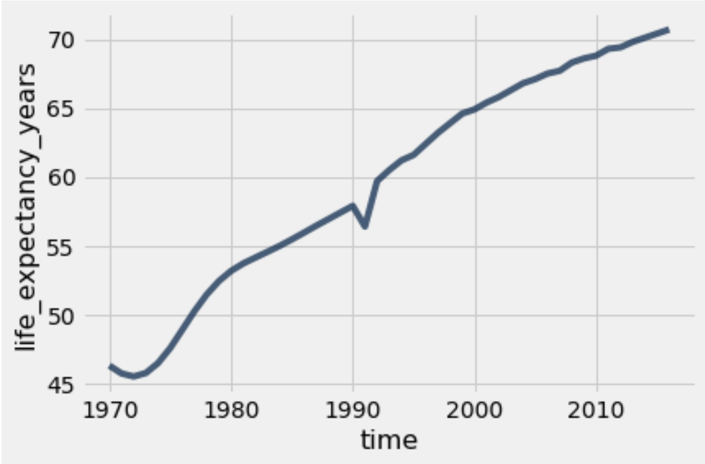
geo	time	children_per_woman_total_fertility
afg	1800	7
afg	1801	7
afg	1802	7
afg	1803	7
afg	1804	7
afg	1805	7
afg	1806	7
afg	1807	7
afg	1808	7
afg	1809	7

... (43402 rows omitted)

Question 3. Perhaps population is growing more slowly because people aren't living as long. Use the `life_expectancy` table to draw a line graph with the years 1970 and later on the horizontal axis that shows how the *life expectancy at birth* has changed in Bangladesh.

```
BEGIN QUESTION
name: q1_3
manual: true
```

In [21]:
```
#Fill in code here
life_expectancy.where("geo", are.equal_to("bgd")).where("time",are.above(196
9)).plot("time", "life_expectancy_years") # SOLUTION
```



Question 4. Assuming everything else stays the same, do the trends in life expectancy in the graph above directly explain why the population growth rate decreased from 1985 to 2010 in Bangladesh? Why or why not?

Hint: What happened in Bangladesh in 1991, and does that event explain the overall change in population growth rate?

```
BEGIN QUESTION
name: q1_4
manual: true
```

SOLUTION: This graph indicates that people are living longer, which would increase population growth if everything else stayed the same. The tragic cyclone in 1991 certainly affected population size, but life expectancy contnued to increase shortly afterward, so it does not explain the 25-year trend in population growth rate decline.

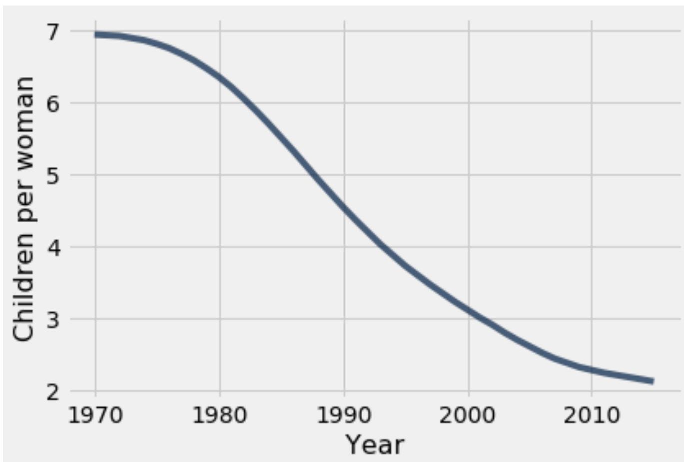
The `fertility` table contains a statistic that is often used to measure how many babies are being born, the *total fertility rate*. This number describes the [number of children a woman would have in her lifetime (https://www.measureevaluation.org/prh/rh_indicators/specific/fertility/total-fertility-rate)](https://www.measureevaluation.org/prh/rh_indicators/specific/fertility/total-fertility-rate), on average, if the current rates of birth by age of the mother persisted throughout her child bearing years, assuming she survived through age 49.

Question 5. Write a function `fertility_over_time` that takes the Alpha-3 code of a `country` and a `start` year. It returns a two-column table with labels `Year` and `Children per woman` that can be used to generate a line chart of the country's fertility rate each year, starting at the `start` year. The plot should include the `start` year and all later years that appear in the `fertility` table.

Then, in the next cell, call your `fertility_over_time` function on the Alpha-3 code for Bangladesh and the year 1970 in order to plot how Bangladesh's fertility rate has changed since 1970. Note that the function `fertility_over_time` should not return the plot itself. **The expression that draws the line plot is provided for you; please don't change it.**

```
BEGIN QUESTION
name: q1_5
```

In [22]:
```
def fertility_over_time(country, start):
    """Create a two-column table that describes a country's total fertility rat
e each year."""
    country_fertility = fertility.where('geo', are.equal_to(country)) # SOLUTIO
N
    country_fertility_after_start = country_fertility.where('time', are.above_o
r_equal_to(start)) # SOLUTION
    cleaned_table = country_fertility_after_start.select('time', 'children_per_
woman_total_fertility').relabel('time', 'Year').relabel('children_per_woman_tot
al_fertility', 'Children per woman') # SOLUTION
    return cleaned_table # SOLUTION

bangladesh_code = 'bgd' # SOLUTION
fertility_over_time(bangladesh_code, 1970).plot(0, 1) # You should *not* change
this line.
```



In [23]:
```
# TEST
# Check your column labels and spelling
fertility_over_time('usa', 2010).labels == ('Year', 'Children per woman')
```

Out[23]: True

In [24]:
```
# TEST
# Check that you use the start year to determine the data range.
all(fertility_over_time('usa', 2010).column('Year') == np.arange(2010, 2016))
```

Out[24]: True

In [25]:
```
# TEST
# Check that you use the start year to determine the data range.
all(fertility_over_time('usa', 2005).column('Year') == np.arange(2005, 2016))
```

Out[25]: True

In [26]:
```
# HIDDEN TEST
print(fertility_over_time('bgd', 2009))
```

```
Year | Children per woman
2009 | 2.32
2010 | 2.28
2011 | 2.24
2012 | 2.21
2013 | 2.18
2014 | 2.15
2015 | 2.12
```

In [27]:
```
# HIDDEN TEST
print(fertility_over_time('usa', 2010))
```

```
Year | Children per woman
2010 | 1.93
2011 | 1.9
2012 | 1.9
2013 | 1.98
2014 | 1.97
2015 | 1.97
```

Question 6. Assuming everything else is constant, do the trends in fertility in the graph above help directly explain why the population growth rate decreased from 1985 to 2010 in Bangladesh? Why or why not?

```
BEGIN QUESTION
name: q1_6
manual: true
```

SOLUTION: Yes, a declining fertility rate shows that fewer babies are being born each year, which directly explains decreasing population growth.

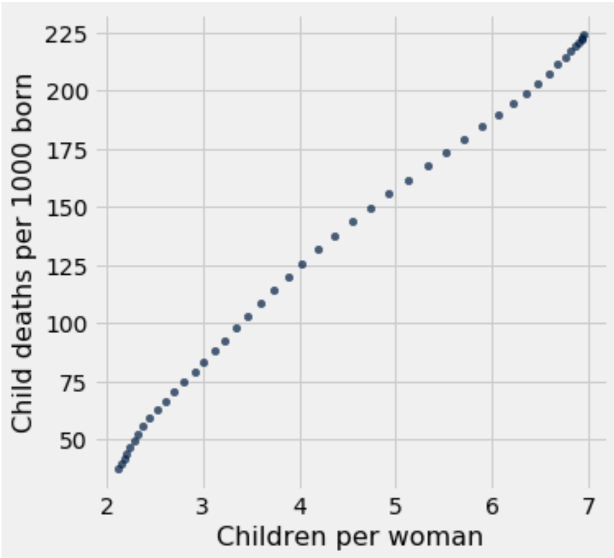
It has been observed that lower fertility rates are often associated with lower child mortality rates. The link has been attributed to family planning: if parents can expect that their children will all survive into adulthood, then they will choose to have fewer children. We can see if this association is evident in Bangladesh by plotting the relationship between total fertility rate and [child mortality rate per 1000 children (https://en.wikipedia.org/wiki/Child_mortality)](https://en.wikipedia.org/wiki/Child_mortality).

Question 7. Using both the `fertility` and `child_mortality` tables, draw a scatter diagram that has Bangladesh's total fertility on the horizontal axis and its child mortality on the vertical axis with one point for each year, starting with 1970.

The expression that draws the scatter diagram is provided for you; please don't change it. Instead, create a table called `post_1969_fertility_and_child_mortality` with the appropriate column labels and data in order to generate the chart correctly. Use the label `Children per woman` to describe total fertility and the label `Child deaths per 1000 born` to describe child mortality.

```
BEGIN QUESTION
name: q1_7
manual: false
```

In [28]:
```
bgd_fertility = fertility.where('geo', are.equal_to('bgd')).drop('geo') # SOLUTION
bgd_child_mortality = child_mortality.where('geo', are.equal_to('bgd')).drop('geo') # SOLUTION
fertility_and_child_mortality = bgd_fertility.join('time', bgd_child_mortality) # SOLUTION
post_1969_fertility_and_child_mortality = fertility_and_child_mortality.where('time', are.above(1969)).relabeled('children_per_woman_total_fertility', 'Children per woman').relabel('child_mortality_under_5_per_1000_born', 'Child deaths per 1000 born') # SOLUTION

post_1969_fertility_and_child_mortality.scatter('Children per woman', 'Child deaths per 1000 born') # You should *not* change this line.
```



In [29]:
```
# TEST
# Make sure you are using the date range 1970-2015
post_1969_fertility_and_child_mortality.num_rows
```

Out[29]: 46

In [30]:
```
# TEST
# Check your column labels and spelling
all([label in post_1969_fertility_and_child_mortality.labels for label in ['Children per woman', 'Child deaths per 1000 born']])
```

Out[30]: True

In [31]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Child deaths per 1000 born')
np.allclose(facm.column(1).item(0), 224.1)
```

Out[31]: True

In [32]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Ch
ild deaths per 1000 born')
np.allclose(facm.column(1).item(-1), 37.6)
```
Out[32]: True

In [33]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Ch
ild deaths per 1000 born')
np.allclose(facm.column(0).item(-1), 2.12)
```
Out[33]: True

In [34]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Ch
ild deaths per 1000 born')
np.allclose(facm.column(0).item(0), 6.95)
```
Out[34]: True

In [35]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Ch
ild deaths per 1000 born')
np.allclose(facm.column(1).mean(), 131.41521739130437)
```
Out[35]: True

In [36]:
```
# HIDDEN TEST
facm = post_1969_fertility_and_child_mortality.select('Children per woman', 'Ch
ild deaths per 1000 born')
np.allclose(facm.column(0).mean(), 4.3958695652173922)
```
Out[36]: True

Question 8. In one or two sentences, describe the association (if any) that is illustrated by this scatter diagram. Does the diagram show that reduced child mortality causes parents to choose to have fewer children?

```
BEGIN QUESTION
name: q1_8
manual: true
```

SOLUTION: We can observe a very strong linear association between fertility rate and child mortality rate. It is strong because the points fall so near to a line drawn through the diagram. However, this association does not tell us whether one of these changes caused a change in the other.

Checkpoint (due Friday 2/21)

Congratulations, you have reached the checkpoint! Run the submit cell below to generate the checkpoint submission.

In [37]:
```
_ = ok.submit()
```

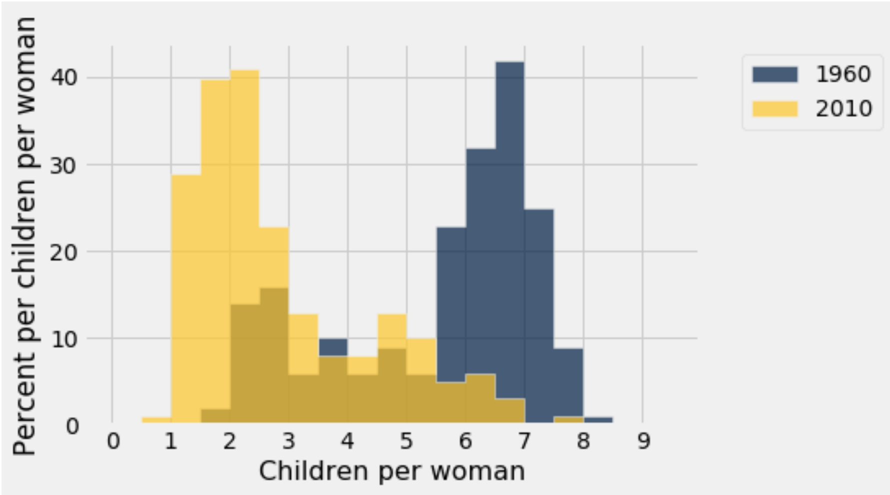
```
Saving notebook...

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-37-cc46ca874451> in <module>
----> 1 _ = ok.submit()

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
submit(self)
     69         messages = {}
     70         self.assignment.set_args(submit=True)
---> 71         if self.save(messages):
     72             return self.run('backup', messages)
     73         else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save(self, messages, delay, attempts)
     78
     79     def save(self, messages, delay=0.5, attempts=3):
---> 80         saved = self.save_notebook()
     81         if not saved:
     82             return None

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save_notebook(self)
    115         # Wait for first .ipynb to save
    116         if ipynbs:
--> 117             if wait_for_save(ipynbs[0]):
    118                 print("Saved '{}'.".format(ipynbs[0]))
    119             else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
wait_for_save(filename, timeout)
    160     Returns True if a save was detected, and False otherwise.
    161     """
--> 162     modification_time = os.path.getmtime(filename)
    163     start_time = time.time()
    164     while time.time() < start_time + timeout:

~/work/teaching/datasp20/lib/python3.6/genericpath.py in getmtime(filename)
     53 def getmtime(filename):
     54     """Return the last modification time of a file, reported by os.stat
()."""
---> 55     return os.stat(filename).st_mtime
     56
     57

FileNotFoundError: [Errno 2] No such file or directory: 'project1.ipynb'
```

The World

The change observed in Bangladesh since 1970 can also be observed in many other developing countries: health services improve, life expectancy increases, and child mortality decreases. At the same time, the fertility rate often plummets, and so the population growth rate decreases despite increasing longevity.

Run the cell below to generate two overlaid histograms, one for 1960 and one for 2010, that show the distributions of total fertility rates for these two years among all 201 countries in the `fertility` table.

In [38]:
```
Table().with_columns(
    '1960', fertility.where('time', 1960).column(2),
    '2010', fertility.where('time', 2010).column(2)
).hist(bins=np.arange(0, 10, 0.5), unit='child per woman')
_ = plots.xlabel('Children per woman')
_ = plots.ylabel('Percent per children per woman')
_ = plots.xticks(np.arange(10))
```



Question 9. Assign `fertility_statements` to an array of the numbers of each statement below that can be correctly inferred from these histograms.

1. About the same number of countries had a fertility rate between 3.5 and 4.5 in both 1960 and 2010.
2. In 2010, about 40% of countries had a fertility rate between 1.5 and 2.
3. In 1960, less than 20% of countries had a fertility rate below 3.
4. More countries had a fertility rate above 3 in 1960 than in 2010.
5. At least half of countries had a fertility rate between 5 and 8 in 1960.
6. At least half of countries had a fertility rate below 3 in 2010.

```
BEGIN QUESTION
name: q1_9
```

In [39]:
```
fertility_statements = make_array(1, 3, 4, 5, 6) # SOLUTION
```

In [40]:
```
# TEST
# Please use a list of integers from 1 to 6
all(x in range(1, 7) for x in set(fertility_statements))
```

Out[40]: `True`

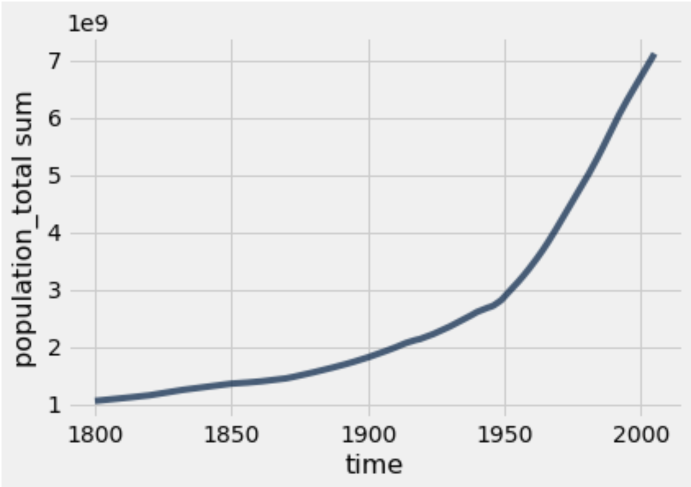
In [41]:
```
# HIDDEN TEST
set(fertility_statements)
```

Out[41]: `{1, 3, 4, 5, 6}`

Question 10. Draw a line plot of the world population from 1800 through 2005. The world population is the sum of all the country's populations.

```
BEGIN QUESTION
name: q1_10
manual: true
```

In [42]:
```
#Fill in code here
population.where('time', are.between(1800, 2006)).drop('geo').group('time', sum).plot('time', "population_total sum") # SOLUTION
```



Question 11. Create a function `stats_for_year` that takes a `year` and returns a table of statistics. The table it returns should have four columns: `geo` , `population_total` , `children_per_woman_total_fertility` , and `child_mortality_under_5_per_1000_born` . Each row should contain one Alpha-3 country code and three statistics: population, fertility rate, and child mortality for that `year` from the `population` , `fertility` and `child_mortality` tables. Only include rows for which all three statistics are available for the country and year.

In addition, restrict the result to country codes that appears in `big_50` , an array of the 50 most populous countries in 2010. This restriction will speed up computations later in the project.

After you write `stats_for_year` , try calling `stats_for_year` on any year between 1960 and 2010. Try to understand the output of stats_for_year.

Hint: The tests for this question are quite comprehensive, so if you pass the tests, your function is probably correct. However, without calling your function yourself and looking at the output, it will be very difficult to understand any problems you have, so try your best to write the function correctly and check that it works before you rely on the `ok` tests to confirm your work.

```
BEGIN QUESTION
name: q1_11
manual: false
```

In [43]:
```
# We first create a population table that only includes the
# 50 countries with the largest 2010 populations. We focus on
# these 50 countries only so that plotting later will run faster.
big_50 = population.where('time', are.equal_to(2010)).sort("population_total",
descending=True).take(np.arange(50)).column('geo')
population_of_big_50 = population.where('time', are.above(1959)).where('geo', a
re.contained_in(big_50))

def stats_for_year(year):
    """Return a table of the stats for each country that year."""
    p = population_of_big_50.where('time', are.equal_to(year)).drop('time')
    f = fertility.where('time', are.equal_to(year)).drop('time')
    c = child_mortality.where('time', are.equal_to(year)).drop('time')
    return p.join('geo', f.join('geo', c)) # SOLUTION

stats_for_year(2008) # SOLUTION
```

Out[43]:

geo	population_total	children_per_woman_total_fertility	child_mortality_under_5_per_1000_born
afg	26528741	6.2	110.4
arg	40381860	2.24	15.4
bgd	148252473	2.38	55.9
bra	194769696	1.9	18.6
can	33363256	1.68	5.8
chn	1326690636	1.53	18.5
cod	61809278	6.45	124.5
col	44901660	2.43	19.7
deu	80665906	1.37	4.4
dza	34811059	2.73	29.5

... (40 rows omitted)

In [44]:
```
# TEST
# Incorrect labels for columns
t = stats_for_year(1990)
t.labels == ('geo', 'population_total', 'children_per_woman_total_fertility', '
child_mortality_under_5_per_1000_born')
```

Out[44]: True

In [45]:
```
# TEST
# Incorrect number of rows
t = stats_for_year(1990)
t.num_rows
```

Out[45]: 50

In [46]:
```
# TEST
print(stats_for_year(1960).sort('geo').take(np.arange(5, 50, 5)))
```

```
geo  | population_total | children_per_woman_total_fertility | child_mortality_
under_5_per_1000_born
chn  | 644450173        | 3.99                               | 309
egy  | 27072397         | 6.63                               | 312.8
gha  | 6652285          | 6.75                               | 210.9
ita  | 49714962         | 2.37                               | 52
mex  | 38174114         | 6.78                               | 142.9
npl  | 10056945         | 5.99                               | 327.1
prk  | 11424179         | 4.58                               | 127.3
tur  | 27553280         | 6.3                                | 249
uzb  | 8789492          | 6.71                               | 175.7
```

In [47]:
```
# TEST
print(stats_for_year(2010).sort('geo').take(np.arange(3, 50, 5)))
```

```
geo  | population_total | children_per_woman_total_fertility | child_mortality_
under_5_per_1000_born
bra  | 198614208        | 1.84                               | 16.7
deu  | 80435307         | 1.39                               | 4.2
fra  | 62961136         | 1.98                               | 4.3
irn  | 74253373         | 1.9                                | 19.2
kor  | 49090041         | 1.27                               | 4.1
mys  | 28119500         | 2                                  | 8.3
phl  | 93038902         | 3.15                               | 31.9
sdn  | 36114885         | 4.64                               | 80.2
ukr  | 45647497         | 1.44                               | 11.8
yem  | 23591972         | 4.5                                | 58.8
```

Question 12. Create a table called `pop_by_decade` with two columns called `decade` and `population`. It has a row for each `year` since 1960 that starts a decade. The `population` column contains the total population of all countries included in the result of `stats_for_year(year)` for the first `year` of the decade. For example, 1960 is the first year of the 1960's decade. You should see that these countries contain most of the world's population.

Hint: One approach is to define a function `pop_for_year` that computes this total population, then `apply` it to the `decade` column. The `stats_for_year` function from the previous question may be useful here.

This first test is just a sanity check for your helper function if you choose to use it. You will not lose points for not implementing the function `pop_for_year`.

Note: The cell where you will generate the `pop_by_decade` table is below the cell where you can choose to define the helper function `pop_for_year`. You should define your `pop_by_decade` table in the cell that starts with the table `decades` being defined.

```
BEGIN QUESTION
name: q1_12_0
manual: false
points: 0
```

In [48]:
```
def pop_for_year(year):
    return sum(stats_for_year(year).column('population_total')) # SOLUTION
```

In [49]:
```
# TEST
pop_for_year(1972) == 3345978384
```
Out[49]: True

In [50]:
```
# TEST
pop_for_year(1989) == 4567880153
```
Out[50]: True

In [51]:
```
# TEST
pop_for_year(2002) == 5501335945
```
Out[51]: True

Now that you've defined your helper function (if you've chosen to do so), define the `pop_by_decade` table.

```
BEGIN QUESTION
name: q1_12
manual: false
```

In [52]:
```
decades = Table().with_column('decade', np.arange(1960, 2011, 10))

pop_by_decade = decades.with_column('population', decades.apply(pop_for_year, '
decade')) # SOLUTION
pop_by_decade.set_format(1, NumberFormatter)
```
Out[52]:

decade	population
1960	2,624,944,597
1970	3,211,487,418
1980	3,880,722,003
1990	4,648,434,558
2000	5,367,553,063
2010	6,040,810,517

In [53]:
```
# TEST
# Check your column labels and spelling
pop_by_decade.labels == ('decade', 'population')
```
Out[53]: True

In [54]:
```
# TEST
# The first year of the 1960's is 1960.
pop_by_decade.column(0).item(0) == 1960
```
Out[54]: True

In [55]:
```
# HIDDEN TEST
pop_by_decade.num_rows == 6
```
Out[55]: True

In [56]:
```
# HIDDEN TEST
pop_by_decade.column(0).item(0) == 1960
```
Out[56]: True

In [57]:
```
# HIDDEN TEST
pop_by_decade.column(0).item(5) == 2010
```
Out[57]: True

In [58]:
```
# HIDDEN TEST
pop_by_decade.column(1).item(1) == 3211487418
```
Out[58]: True

In [59]:
```
# HIDDEN TEST
pop_by_decade.column(1).item(5) == 6040810517
```
Out[59]: True

The `countries` table describes various characteristics of countries. The `country` column contains the same codes as the `geo` column in each of the other data tables (`population` , `fertility` , and `child_mortality`). The `world_6region` column classifies each country into a region of the world. Run the cell below to inspect the data.

In [60]:
```
countries = Table.read_table('countries.csv').where('country', are.contained_in
(population.group('geo').column('geo')))
countries.select('country', 'name', 'world_6region')
```
Out[60]:

country	name	world_6region
afg	Afghanistan	south_asia
akr_a_dhe	Akrotiri and Dhekelia	europe_central_asia
alb	Albania	europe_central_asia
dza	Algeria	middle_east_north_africa
asm	American Samoa	east_asia_pacific
and	Andorra	europe_central_asia
ago	Angola	sub_saharan_africa
aia	Anguilla	america
atg	Antigua and Barbuda	america
arg	Argentina	america

... (245 rows omitted)

Question 13. Create a table called `region_counts` that has two columns, `region` and `count`. It should contain two columns: a region column and a count column that contains the number of countries in each region that appear in the result of `stats_for_year(1960)`. For example, one row would have `south_asia` as its `world_6region` value and an integer as its `count` value: the number of large South Asian countries for which we have population, fertility, and child mortality numbers from 1960.

```
BEGIN QUESTION
name: q1_13
```

In [61]:
```
region_counts = countries.join('country', stats_for_year(1960), 'geo').group('w
orld_6region').relabeled('world_6region', 'region') # SOLUTION
region_counts
```

Out[61]:

region	count
america	8
east_asia_pacific	10
europe_central_asia	10
middle_east_north_africa	7
south_asia	5
sub_saharan_africa	10

In [62]:
```
# TEST
# Check your column labels and spelling
region_counts.labels == ('region', 'count')
```

Out[62]: True

In [63]:
```
# TEST
# Counts must sum to 50
sum(region_counts.column('count')) == 50
```

Out[63]: True

In [64]:
```
# HIDDEN TEST
list(region_counts.sort("count").column("count"))
```

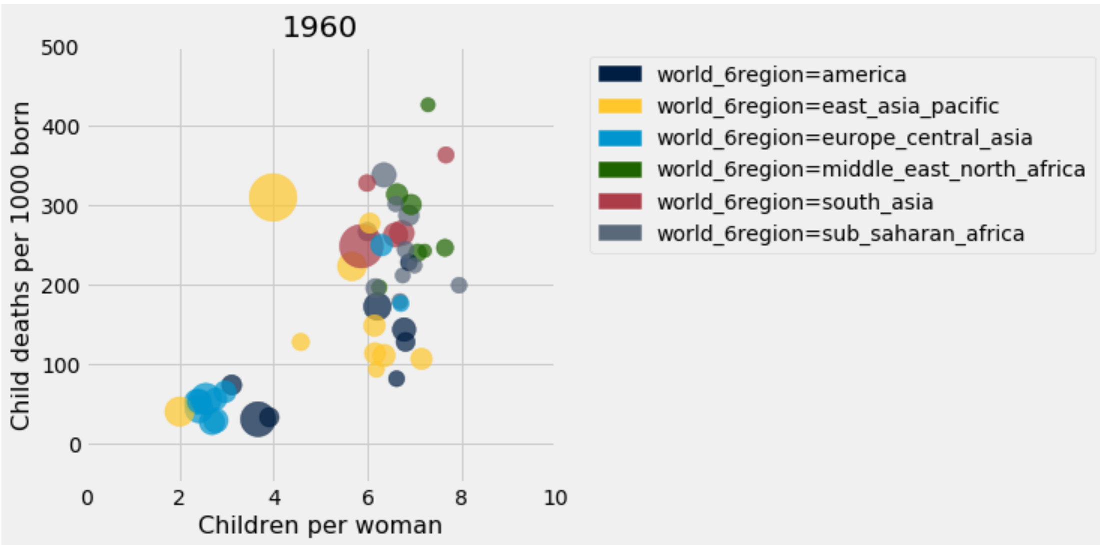
Out[64]: [5, 7, 8, 10, 10, 10]

In [65]:
```
# HIDDEN TEST
print(list(region_counts.sort("count").column("region")))
```

```
['south_asia', 'middle_east_north_africa', 'america', 'east_asia_pacific', 'eur
ope_central_asia', 'sub_saharan_africa']
```

The following scatter diagram compares total fertility rate and child mortality rate for each country in 1960. The area of each dot represents the population of the country, and the color represents its region of the world. Run the cell. Do you think you can identify any of the dots?

In [66]:
```
from functools import lru_cache as cache

# This cache annotation makes sure that if the same year
# is passed as an argument twice, the work of computing
# the result is only carried out once.
@cache(None)
def stats_relabeled(year):
    """Relabeled and cached version of stats_for_year."""
    return stats_for_year(year).relabel(2, 'Children per woman').relabel(3, 'Ch
ild deaths per 1000 born')

def fertility_vs_child_mortality(year):
    """Draw a color scatter diagram comparing child mortality and fertility."""
    with_region = stats_relabeled(year).join('geo', countries.select('country',
'world_6region'), 'country')
    with_region.scatter(2, 3, sizes=1, group=4, s=500)
    plots.xlim(0,10)
    plots.ylim(-50, 500)
    plots.title(year)

fertility_vs_child_mortality(1960)
```



Question 14. Assign `scatter_statements` to an array of the numbers of each statement below that can be inferred from this scatter diagram for 1960.

1. As a whole, the `europe_central_asia` region had the lowest child mortality rate.
2. The lowest child mortality rate of any country was from an `east_asia_pacific` country.
3. Most countries had a fertility rate above 5.
4. There was an association between child mortality and fertility.
5. The two largest countries by population also had the two highest child mortality rate.

```
BEGIN QUESTION
name: q1_14
```

In [67]:
```
scatter_statements = make_array(1, 3, 4) # SOLUTION
```

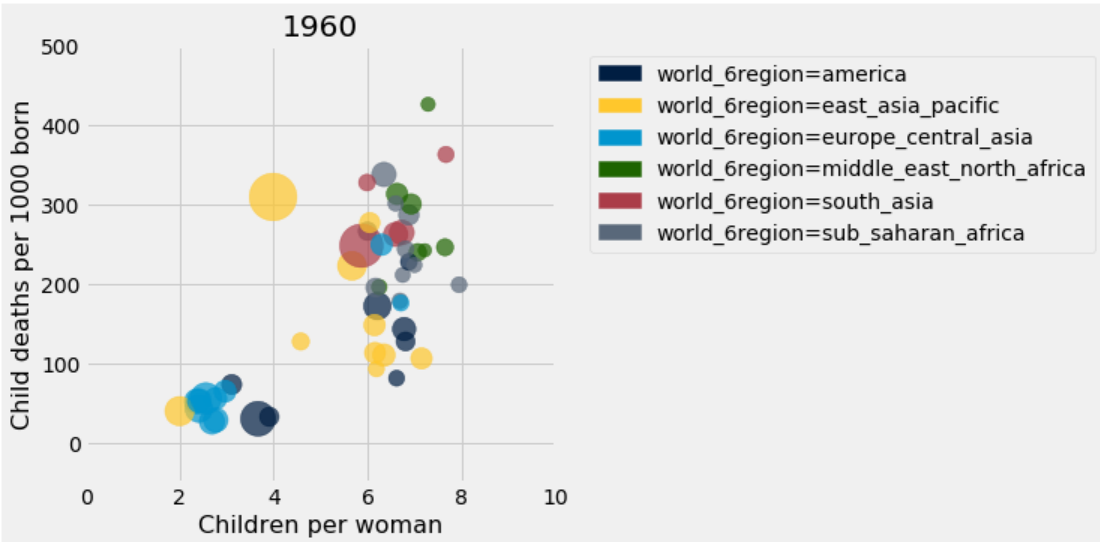
In [68]:
```
# TEST
# Please use a list of integers from 1 to 5
all(x in range(1, 6) for x in set(scatter_statements))
```
Out[68]: True

In [69]:
```
# HIDDEN TEST
set(scatter_statements) == {1, 3, 4}
```
Out[69]: True

The result of the cell below is interactive. Drag the slider to the right to see how countries have changed over time. You'll find that the great divide between so-called "Western" and "developing" countries that existed in the 1960's has nearly disappeared. This shift in fertility rates is the reason that the global population is expected to grow more slowly in the 21st century than it did in the 19th and 20th centuries.

Note: Don't worry if a red warning pops up when running the cell below. You'll still be able to run the cell!

In [70]:
```
import ipywidgets as widgets

# This part takes a few minutes to run because it
# computes 55 tables in advance: one for each year.
Table().with_column('Year', np.arange(1960, 2016)).apply(stats_relabeled, 'Year
')

_ = widgets.interact(fertility_vs_child_mortality,
                     year=widgets.IntSlider(min=1960, max=2015, value=1960))
```



Now is a great time to take a break and watch the same data presented by [Hans Rosling in a 2010 TEDx talk (https://www.gapminder.org/videos/reducing-child-mortality-a-moral-and-environmental-imperative)](https://www.gapminder.org/videos/reducing-child-mortality-a-moral-and-environmental-imperative) with smoother animation and witty commentary.

2. Global Poverty

In 1800, 85% of the world's 1 billion people lived in *extreme poverty*, defined by the United Nations as "a condition characterized by severe deprivation of basic human needs, including food, safe drinking water, sanitation facilities, health, shelter, education and information." A common measure of extreme poverty is a person living on less than $1.25 per day.

In 2018, the proportion of people living in extreme poverty was estimated to be 8%. Although the world rate of extreme poverty has declined consistently for hundreds of years, the number of people living in extreme poverty is still over 600 million. The United Nations recently adopted an [ambitious goal (http://www.un.org/sustainabledevelopment/poverty/)](http://www.un.org/sustainabledevelopment/poverty/): "By 2030, eradicate extreme poverty for all people everywhere." In this section, we will examine extreme poverty trends around the world.

First, load the population and poverty rate by country and year and the country descriptions. While the `population` table has values for every recent year for many countries, the `poverty` table only includes certain years for each country in which a measurement of the rate of extreme poverty was available.

In [71]:
```
population = Table.read_table('population.csv')
countries = Table.read_table('countries.csv').where('country', are.contained_in
(population.group('geo').column('geo')))
poverty = Table.read_table('poverty.csv')
poverty.show(3)
```

geo	time	extreme_poverty_percent_people_below_125_a_day
alb	1996	0.2
alb	2002	0.73
alb	2004	0.53

... (1096 rows omitted)

Question 1. Assign `latest_poverty` to a three-column table with one row for each country that appears in the `poverty` table. The first column should contain the 3-letter code for the country. The second column should contain the most recent year for which an extreme poverty rate is available for the country. The third column should contain the poverty rate in that year. **Do not change the last line, so that the labels of your table are set correctly.**

Hint: think about how `group` works: it does a sequential search of the table (from top to bottom) and collects values in the array in the order in which they appear, and then applies a function to that array. The `first` function may be helpful, but you are not required to use it.

```
BEGIN QUESTION
name: q2_1
```

In [72]:
```
def first(values):
    return values.item(0)

latest_poverty = poverty.sort('time', descending=True).group('geo', first) # SO
LUTION
latest_poverty = latest_poverty.relabeled(0, 'geo').relabeled(1, 'time').relabe
led(2, 'poverty_percent') # You should *not* change this line.
latest_poverty
```

Out[72]:

geo	time	poverty_percent
ago	2009	43.37
alb	2012	0.46
arg	2011	1.41
arm	2012	1.75
aus	2003	1.36
aut	2004	0.34
aze	2008	0.31
bdi	2006	81.32
bel	2000	0.5
ben	2012	51.61

... (135 rows omitted)

In [73]:
```
# TEST
# Please don't edit the last line.
latest_poverty.labels == ('geo', 'time', 'poverty_percent')
```

Out[73]: True

In [74]:
```
# TEST
# The result should have one row per country.
latest_poverty.num_rows
```

Out[74]: 145

In [75]:
```
# HIDDEN TEST
print(latest_poverty.sort('geo').take(np.arange(7, 107, 10)))
```

```
geo  | time | poverty_percent
bdi  | 2006 | 81.32
bra  | 2012 | 3.75
cod  | 2006 | 87.72
dom  | 2012 | 2.25
fsm  | 2000 | 31.15
guy  | 1998 | 8.7
isr  | 2010 | 0.39
lbr  | 2007 | 83.76
mex  | 2012 | 1.03
nga  | 2010 | 62.03
```

Question 2. Using both `latest_poverty` and `population` , create a four-column table called `recent_poverty_total` with one row for each country in `latest_poverty` . The four columns should have the following labels and contents:

1. `geo` contains the 3-letter country code,
2. `poverty_percent` contains the most recent poverty percent,
3. `population_total` contains the population of the country in 2010,
4. `poverty_total` contains the number of people in poverty **rounded to the nearest integer**, based on the 2010 population and most recent poverty rate.

```
BEGIN QUESTION
name: q2_2
```

In [76]:

```
""" # BEGIN PROMPT
poverty_and_pop = ...
recent_poverty_total = ...
recent_poverty_total
"""; # END PROMPT
# BEGIN SOLUTION NO PROMPT
pop_2010 = population.where("time", are.equal_to(2010)).drop('time')
poverty_and_pop = latest_poverty.drop('time').join('geo', pop_2010)
poverty_total_arr = np.round(poverty_and_pop.column('poverty_percent')/100*pove
rty_and_pop.column('population_total'))
recent_poverty_total = poverty_and_pop.with_column('poverty_total', poverty_tot
al_arr)
recent_poverty_total
# END SOLUTION
```

Out[76]:

geo	poverty_percent	population_total	poverty_total
ago	43.37	21219954	9.20309e+06
alb	0.46	2901883	13349
arg	1.41	41222875	581243
arm	1.75	2963496	51861
aus	1.36	22162863	301415
aut	0.34	8391986	28533
aze	0.31	9099893	28210
bdi	81.32	9461117	7.69378e+06
bel	0.5	10929978	54650
ben	51.61	9509798	4.90801e+06

... (135 rows omitted)

In [77]:

```
# TEST
# Check your column labels and spelling
recent_poverty_total.labels == ('geo', 'poverty_percent', 'population_total', '
poverty_total')
```

Out[77]: True

In [78]:
```
# TEST
# Careful, the population of Australia in 2010 was 22,162,863
recent_poverty_total.where('geo', 'aus').column(2).item(0)
```

Out[78]: 22162863

In [79]:
```
# TEST
# The number of people estimated to be living in extreme poverty
# in Australia should be 301,415. That's 22,162,863 * 0.0136
# rounded to the nearest integer.
float(recent_poverty_total.where('geo', 'aus').column(3).item(0))
```

Out[79]: 301415.0

In [80]:
```
# HIDDEN TEST
sum(recent_poverty_total.column(3))
```

Out[80]: 990911651.0

Question 3. Assign the name `poverty_percent` to the known percentage of the world's 2010 population that were living in extreme poverty. Assume that the `poverty_total` numbers in the `recent_poverty_total` table describe **all** people in 2010 living in extreme poverty. You should find a number that is above the 2018 global estimate of 8%, since many country-specific poverty rates are older than 2018.

Hint: The sum of the `population_total` column in the `recent_poverty_total` table is not the world population, because only a subset of the world's countries are included in the `recent_poverty_total` table (only some countries have known poverty rates). Use the `population` table to compute the world's 2010 total population..

```
BEGIN QUESTION
name: q2_3
```

In [81]:
```
""" # BEGIN PROMPT
poverty_percent = ...
poverty_percent
"""; # END PROMPT
# BEGIN SOLUTION NO PROMPT
total_2010_pop = sum(population.where('time', are.equal_to(2010)).column("popul
ation_total"))
poverty_2010_pop = sum(recent_poverty_total.column('poverty_total'))
poverty_percent = 100* poverty_2010_pop/total_2010_pop
poverty_percent
# END SOLUTION
```

Out[81]: 14.299370218520854

In [82]:
```
# TEST
10 <= poverty_percent <= 20
```

Out[82]: True

In [83]:
```
# HIDDEN TEST
np.allclose(np.round(poverty_percent, 2), 14.3)
```

Out[83]: True

The `countries` table includes not only the name and region of countries, but also their positions on the globe.

In [84]:
```
countries.select('country', 'name', 'world_4region', 'latitude', 'longitude')
```

Out[84]:

country	name	world_4region	latitude	longitude
afg	Afghanistan	asia	33	66
akr_a_dhe	Akrotiri and Dhekelia	europe	nan	nan
alb	Albania	europe	41	20
dza	Algeria	africa	28	3
asm	American Samoa	asia	-11.056	-171.082
and	Andorra	europe	42.5078	1.52109
ago	Angola	africa	-12.5	18.5
aia	Anguilla	americas	18.2167	-63.05
atg	Antigua and Barbuda	americas	17.05	-61.8
arg	Argentina	americas	-34	-64

... (245 rows omitted)

Question 4. Using both `countries` and `recent_poverty_total` , create a five-column table called `poverty_map` with one row for every country in `recent_poverty_total` . The five columns should have the following labels and contents:

1. `latitude` contains the country's latitude,
2. `longitude` contains the country's longitude,
3. `name` contains the country's name,
4. `region` contains the country's region from the `world_4region` column of `countries` ,
5. `poverty_total` contains the country's poverty total.

```
BEGIN QUESTION
name: q2_4
```

In [85]:
```
""" # BEGIN PROMPT
poverty_map = ...
poverty_map
"""; # END PROMPT
# BEGIN SOLUTION NO PROMPT
small_countries = countries.select('country', 'latitude', 'longitude', 'name',
'world_4region').relabeled('world_4region', 'region')
small_recent_poverty = recent_poverty_total.select('geo', 'poverty_total')
poverty_map = small_countries.join('country', small_recent_poverty, 'geo' ).dro
p('country')
poverty_map
# END SOLUTION
```

Out[85]:

latitude	longitude	name	region	poverty_total
-12.5	18.5	Angola	africa	9.20309e+06
41	20	Albania	europe	13349
-34	-64	Argentina	americas	581243
40.25	45	Armenia	europe	51861
-25	135	Australia	asia	301415
47.3333	13.3333	Austria	europe	28533
40.5	47.5	Azerbaijan	europe	28210
-3.5	30	Burundi	africa	7.69378e+06
50.75	4.5	Belgium	europe	54650
9.5	2.25	Benin	africa	4.90801e+06

... (135 rows omitted)

In [86]:
```
# TEST
# Check your column labels and spelling
poverty_map.labels == ('latitude', 'longitude', 'name', 'region', 'poverty_tota
l')
```

Out[86]: True

In [87]:
```
# TEST
# Something is wrong with your region column.
list(np.sort(np.unique(poverty_map.column('region'))))
```

Out[87]: ['africa', 'americas', 'asia', 'europe']

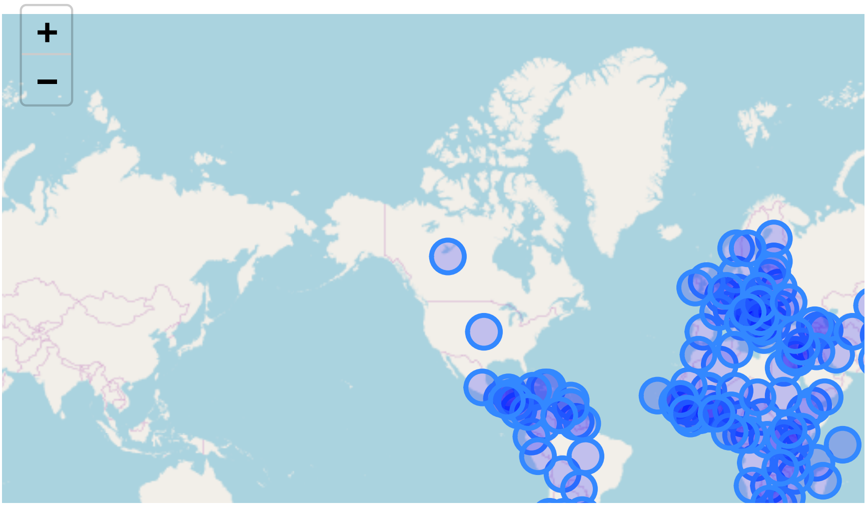
In [88]:
```
# HIDDEN TEST
print(poverty_map.sort('name').take(np.arange(10)).drop(4))
```

```
latitude | longitude | name       | region
41       | 20        | Albania    | europe
28       | 3         | Algeria    | africa
-12.5    | 18.5      | Angola     | africa
-34      | -64       | Argentina  | americas
40.25    | 45        | Armenia    | europe
-25      | 135       | Australia  | asia
47.3333  | 13.3333   | Austria    | europe
40.5     | 47.5      | Azerbaijan | europe
24       | 90        | Bangladesh | asia
53       | 28        | Belarus    | europe
```

In [89]:
```
# HIDDEN TEST
sum(poverty_map.column(4)) == sum(recent_poverty_total.column(3))
```
Out[89]:
```
True
```

Run the cell below to draw a map of the world in which the areas of circles represent the number of people living in extreme poverty. Double-click on the map to zoom in.

In [90]:
```
# It may take a few seconds to generate this map.
colors = {'africa': 'blue', 'europe': 'black', 'asia': 'red', 'americas': 'gree
n'}
scaled = poverty_map.with_columns(
    'poverty_total', 1e-4 * poverty_map.column('poverty_total'),
    'region', poverty_map.apply(colors.get, 'region')
)
Circle.map_table(scaled)
```
Out[90]:



Although people live in extreme poverty throughout the world (with more than 5 million in the United States), the largest numbers are in Asia and Africa.

Question 5. Assign `largest` to a two-column table with the `name` (not the 3-letter code) and `poverty_total` of the 10 countries with the largest number of people living in extreme poverty.

```
BEGIN QUESTION
name: q2_5
```

In [91]:
```
largest = poverty_map.sort('poverty_total', descending=True).take(np.arange(1
0)).select('name', 'poverty_total') #SOLUTION
largest.set_format('poverty_total', NumberFormatter)
```

Out[91]:

name	poverty_total
India	290,881,638.00
Nigeria	98,891,167.00
China	83,944,643.00
Bangladesh	65,574,256.00
Congo, Dem. Rep.	57,841,438.00
Indonesia	39,141,326.00
Ethiopia	32,213,991.00
Pakistan	21,663,595.00
Tanzania	19,847,979.00
Madagascar	18,480,426.00

In [92]:
```
# TEST
# Check your column labels and spelling
largest.labels == ('name', 'poverty_total')
```

Out[92]: True

In [93]:
```
# TEST
# India is the country with the largest number of people living
# in extreme poverty.
largest.column(0).item(0)
```

Out[93]: 'India'

In [94]:
```
# TEST
# The table should contain exactly 10 rows.
largest.num_rows
```

Out[94]: 10

In [95]:
```
# HIDDEN TEST
print(largest.take(np.arange(3)).column(1))
```

```
[2.90881638e+08 9.88911670e+07 8.39446430e+07]
```

Question 6. Write a function called `poverty_timeline` that takes **the name of a country** as its argument. It should draw a line plot of the number of people living in poverty in that country with time on the horizontal axis. The line plot should have a point for each row in the `poverty` table for that country. To compute the population living in poverty from a poverty percentage, multiply by the population of the country **in that year**.

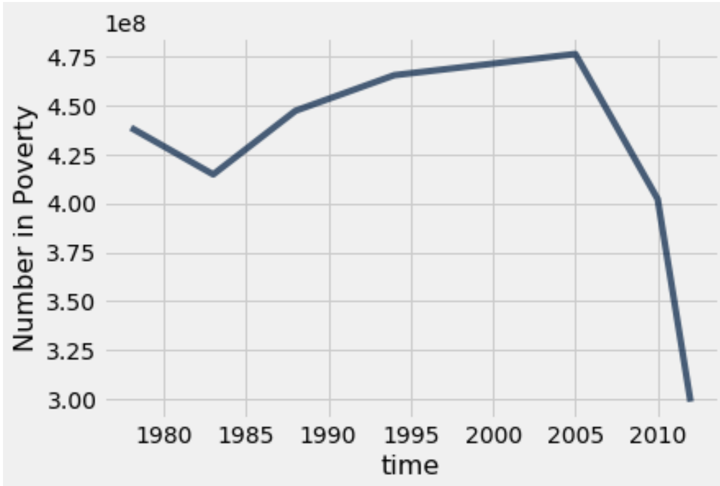
Hint: This question is long. Feel free to create cells and experiment.

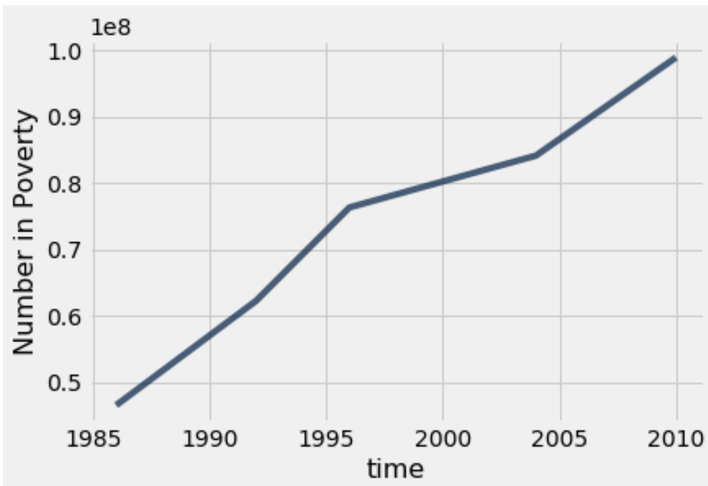
In [96]:
```
""" # BEGIN PROMPT
def poverty_timeline(country):
    '''Draw a timeline of people living in extreme poverty in a country.'''
    geo = ...
    # This solution will take multiple lines of code. Use as many as you need
    ...
"""; # END PROMPT
# BEGIN SOLUTION NO PROMPT
def poverty_timeline(country):
    geo = countries.where('name', are.equal_to(country)).column('country').item
(0)
    country_poverty = poverty.where('geo', are.equal_to(geo)).drop('geo')
    country_population = population.where('geo', are.equal_to(geo)).drop('geo')
    country_pov_and_pop = country_poverty.join('time', country_population)
    num_poverty = country_pov_and_pop.column('population_total') * country_pov_
and_pop.column('extreme_poverty_percent_people_below_125_a_day')/100
    final_tbl = country_pov_and_pop.with_column('Number in Poverty', num_povert
y)
    final_tbl.plot('time', 'Number in Poverty')
# END SOLUTION
```

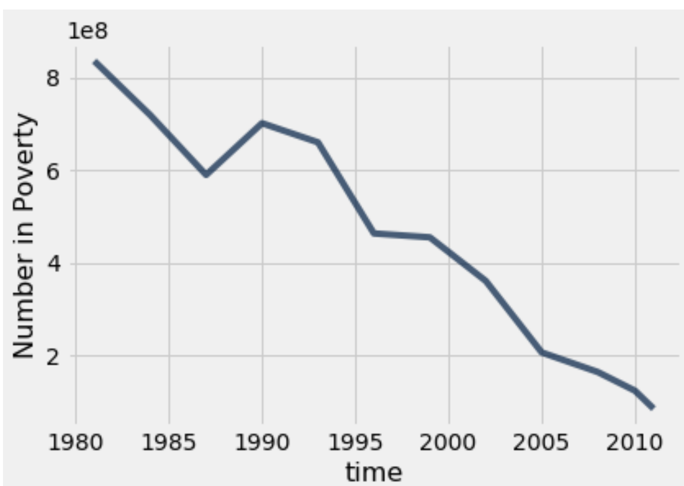
Finally, draw the timelines below to see how the world is changing. You can check your work by comparing your graphs to the ones on [gapminder.org (https://www.gapminder.org/tools/#$state$entities$show$country$/$in@=ind;;;;&marker$axis_y$which=number_of_people_in_poverty&scaleType=linear&spaceRef:null;;;&chart-type=linechart)](https://www.gapminder.org/tools/#$state$entities$show$country$/$in@=ind;;;;&marker$axis_y$which=number_of_people_in_poverty&scaleType=linear&spaceRef:null;;;&chart-type=linechart).

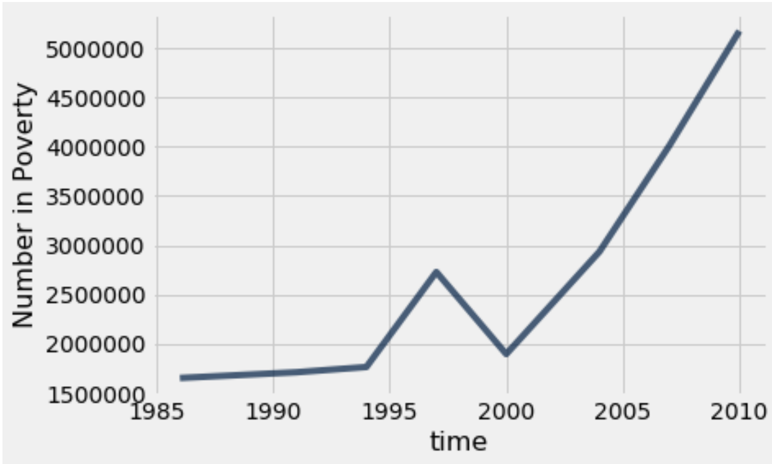
```
BEGIN QUESTION
name: q2_6
manual: true
```

In [97]:
```
poverty_timeline('India')
poverty_timeline('Nigeria')
poverty_timeline('China')
poverty_timeline('United States')
```


1e8
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
Number in Poverty
1980
1985
1990
1995
2000
2005
2010
time


1e8
1.0
0.9
0.8
0.7
0.6
0.5
Number in Poverty
1985
1990
1995
2000
2005
2010
time


1e8
8
6
4
2
Number in Poverty
1980
1985
1990
1995
2000
2005
2010
time

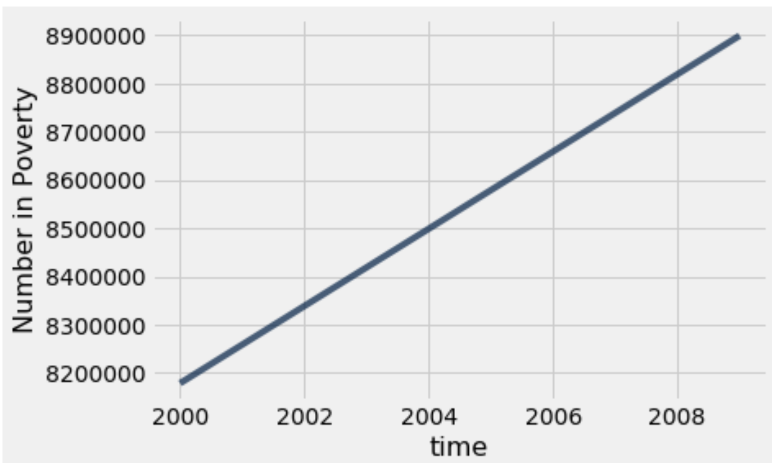


Although the number of people living in extreme poverty has been increasing in Nigeria and the United States, the massive decreases in China and India have shaped the overall trend that extreme poverty is decreasing worldwide, both in percentage and in absolute number.

To learn more, watch [Hans Rosling in a 2015 film (https://www.gapminder.org/videos/dont-panic-end-poverty/)](https://www.gapminder.org/videos/dont-panic-end-poverty/) about the UN goal of eradicating extreme poverty from the world.

Below, we've also added an interactive dropdown menu for you to visualize `poverty_timeline` graphs for other countries. Note that each dropdown menu selection may take a few seconds to run.

In [98]:
```
# Just run this cell

all_countries = poverty_map.column('name')
_ = widgets.interact(poverty_timeline, country=list(all_countries))
```



You're finished! Congratulations on mastering data visualization and table manipulation. Time to submit.

3. Submission

Once you're finished, select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to [okpy.org (https://okpy.org/)](https://okpy.org/) and flag the correct version. To do so, go to the website, click on this assignment, and find the version you would like to have graded. There should be an option to flag that submission for grading!

In [99]:
```
_ = ok.submit()
```

```
Saving notebook...

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-99-cc46ca874451> in <module>
----> 1 _ = ok.submit()

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
submit(self)
     69         messages = {}
     70         self.assignment.set_args(submit=True)
---> 71         if self.save(messages):
     72             return self.run('backup', messages)
     73         else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save(self, messages, delay, attempts)
     78
     79     def save(self, messages, delay=0.5, attempts=3):
---> 80         saved = self.save_notebook()
     81         if not saved:
     82             return None

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
save_notebook(self)
    115         # Wait for first .ipynb to save
    116         if ipynbs:
--> 117             if wait_for_save(ipynbs[0]):
    118                 print("Saved '{}'.".format(ipynbs[0]))
    119             else:

~/work/teaching/datasp20/lib/python3.6/site-packages/client/api/notebook.py in
wait_for_save(filename, timeout)
    160     Returns True if a save was detected, and False otherwise.
    161     """
--> 162     modification_time = os.path.getmtime(filename)
    163     start_time = time.time()
    164     while time.time() < start_time + timeout:

~/work/teaching/datasp20/lib/python3.6/genericpath.py in getmtime(filename)
     53 def getmtime(filename):
     54     """Return the last modification time of a file, reported by os.stat
().""" 
---> 55     return os.stat(filename).st_mtime
     56
     57

FileNotFoundError: [Errno 2] No such file or directory: 'project1.ipynb'
```

In [100]:
```
# For your convenience, you can run this cell to run all the tests at once!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q') and len
(q) <= 10]
print("Finished running all tests.")
```

```
Running all tests...
Finished running all tests.
```