lab05

March 22, 2022

Name: 0131

Section: Allan Gongora

1 Lab 5: World Progress

Welcome to Lab 5!

This final lab in Data 8.1x brings together many of the topics so far, including data table manipulation, visualization, and iteration. The content of the lab is based on a series of talks by Hans Rosling, a statistician who advised many world leaders about the changing state of the world's population.

(Optional) For a video introduction to the topic of Global population change, you can watch Hans Rosling's video, Don't Panic: The Facts About Population.

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests.

[1]:
```
pip install gofer-grader
```

```
Collecting gofer-grader
  Using cached gofer_grader-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (6.1)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-
packages (from gofer-grader) (2.11.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages
(from gofer-grader) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-
packages (from jinja2->gofer-grader) (2.0.1)
Installing collected packages: gofer-grader
Successfully installed gofer-grader-1.1.0
Note: you may need to restart the kernel to use updated packages.
```

[58]:
```
# Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *
```

```
# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

from gofer.ok import check
```

Recommended Reading:

- Visualization
- Functions and Tables
- Iteration

1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include:
   A) Sentence reponses to questions that ask for an explanation
   B) Numeric responses to multiple choice questions
   C) Programming code
2) Moreover, throughout this lab and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Otherwise, you will fail tests that you thought you were passing previously!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

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.

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.

[59]:
```
# The population.csv file can also be found online here:
# https://github.com/open-numbers/ddf--gapminder--systema_globalis/raw/master/
  ↪ddf--datapoints--population_total--by--geo--time.csv
# The version in this project was downloaded in February, 2017.
population = Table.read_table('population.csv')
population.show(3)
```

```
<IPython.core.display.HTML object>
```

1.1 1. Bangladesh

In the `population` table, the `geo` column contains three-letter codes established by the International Organization for Standardization (ISO) in the Alpha-3 standard. We will begin by taking a close look at Bangladesh. Inspect the standard to find the 3-letter code for Bangladesh.

Question 1.1 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.

[60]:
```
b_pop = population.where("time", are.between_or_equal_to(1970, 2015)).
  ↪where("geo", "bgd").drop("geo")
b_pop
```

[60]:
```
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)
```

[61]:
```
check('tests/q1_1.py')
```

[61]:
```
<gofer.ok.OKTestsResult at 0x7f2153c08410>
```

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!

[62]:
```
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
```

[62]:
```
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
```

Run the next cell to create a table called `b_five_growth` which shows the growth rate for each five-year period from 1970 through 2010.

[63]:
```
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', (b_five.
  ↪exclude(0).column(1)/b_1970_through_2010.column(1))**0.2-1)
b_five_growth.set_format('annual_growth', PercentFormatter)
```

[63]:
```
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%
```

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.

[64]:
```
life_expectancy = Table.read_table('life_expectancy.csv')
child_mortality = Table.read_table('child_mortality.csv').relabeled(2,␣
  ↪'child_mortality_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. 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.

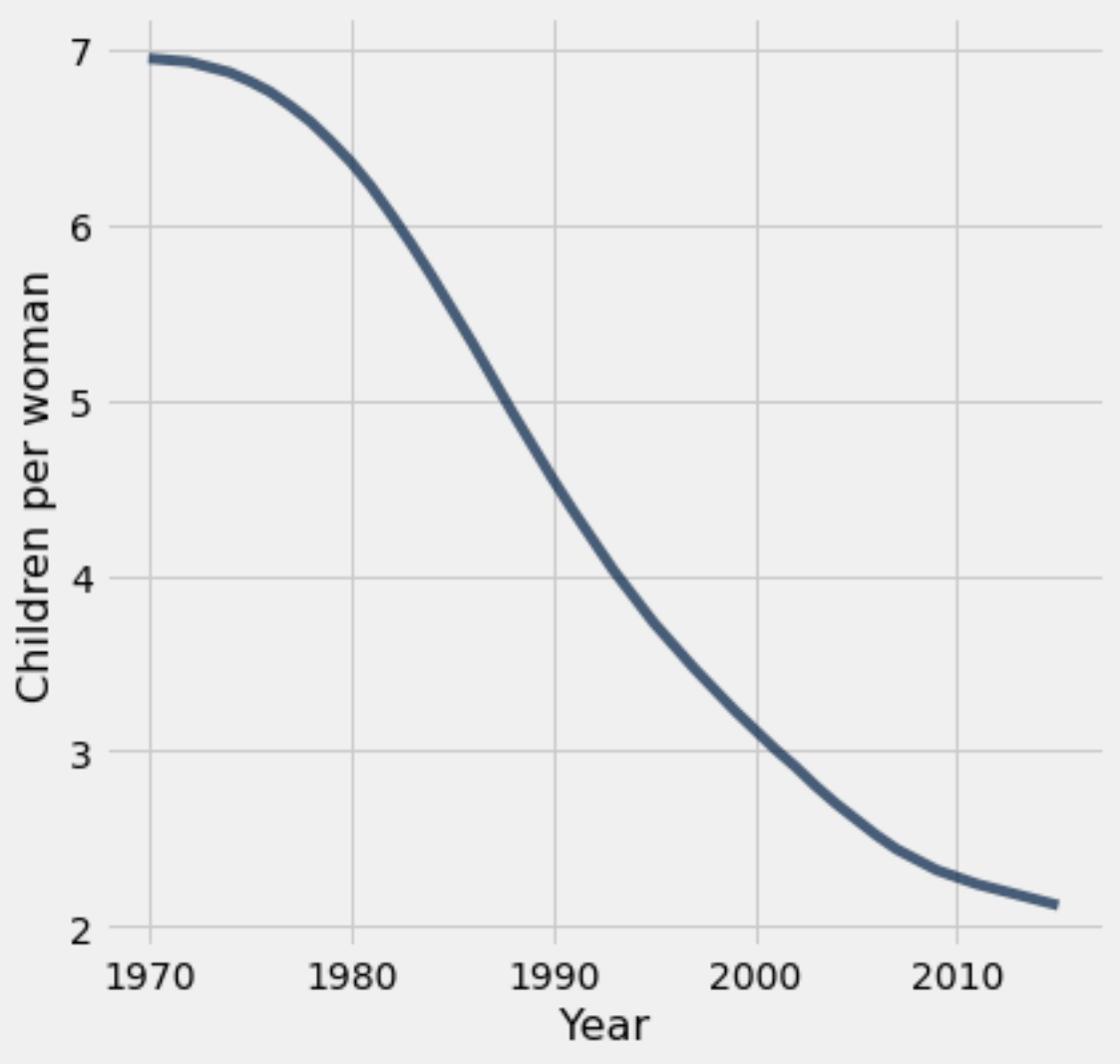
The `fertility` table contains a statistic that is often used to measure how many babies are being born, the *total fertility rate*. This number describes the number of children a woman would have in her lifetime, on average, if the current rates of birth by age of the mother persisted throughout her child bearing years, assuming she survived through age 49.

Question 1.2 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.**

[65]:
```
def fertility_over_time(country, start):
    """Create a two-column table that describes a country's total fertility␣
 ↪rate each year."""
    return fertility.where("geo", country).where(
        "time", are.above_or_equal_to(start)
    ).drop("geo").relabel(["children_per_woman_total_fertility", "time"],␣
 ↪["Children per woman", "Year"])
```

[66]:
```
bangladesh_code = "bgd"
fertility_over_time(bangladesh_code, 1970).plot(0, 1) # You should *not* change␣
 ↪this line.
```



[67]:
```
check('tests/q1_2.py')
```

[67]:
```
<gofer.ok.OKTestsResult at 0x7f215310a310>
```

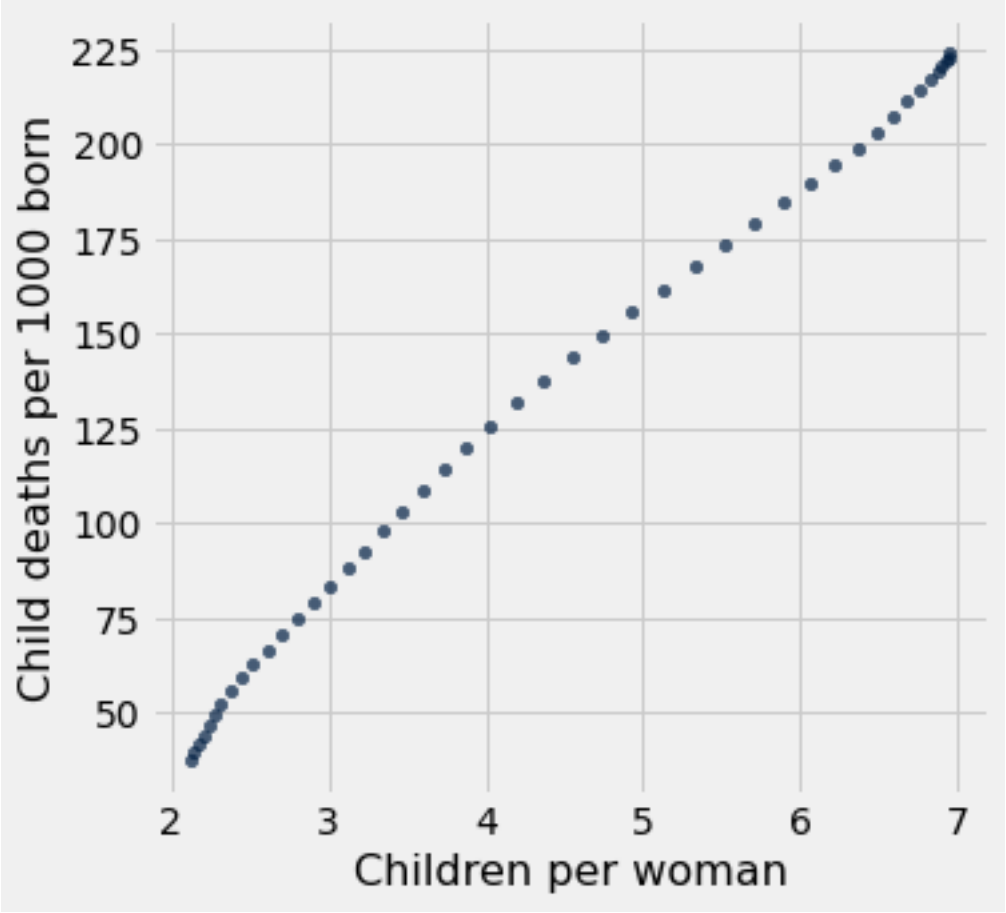
Question 1.3 Using both the `fertility` and `child_mortality` tables, draw a scatter diagram with one point for each year, starting with 1970, that has Bangladesh's total fertility on the horizontal axis and its child mortality on the vertical axis.

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.

[68]:
```
child_mortality
```

[68]:
```
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)
```

[69]:
```
bgd_fertility = fertility_over_time("bgd", 1970)
bgd_child_mortality = child_mortality.where("geo", "bgd").where(
    "time", are.above_or_equal_to(1970)).drop("geo").relabel(
    ["time", "child_mortality_under_5_per_1000_born"], ["Year", "Child deaths␣
 ↪per 1000 born"])
# fertility_and_child_mortality =
post_1969_fertility_and_child_mortality = bgd_fertility.join("Year",␣
 ↪bgd_child_mortality)
post_1969_fertility_and_child_mortality.scatter('Children per woman', 'Child␣
 ↪deaths per 1000 born') # You should *not* change this line.
```



[70]:
```
check('tests/q1_3.py')
```

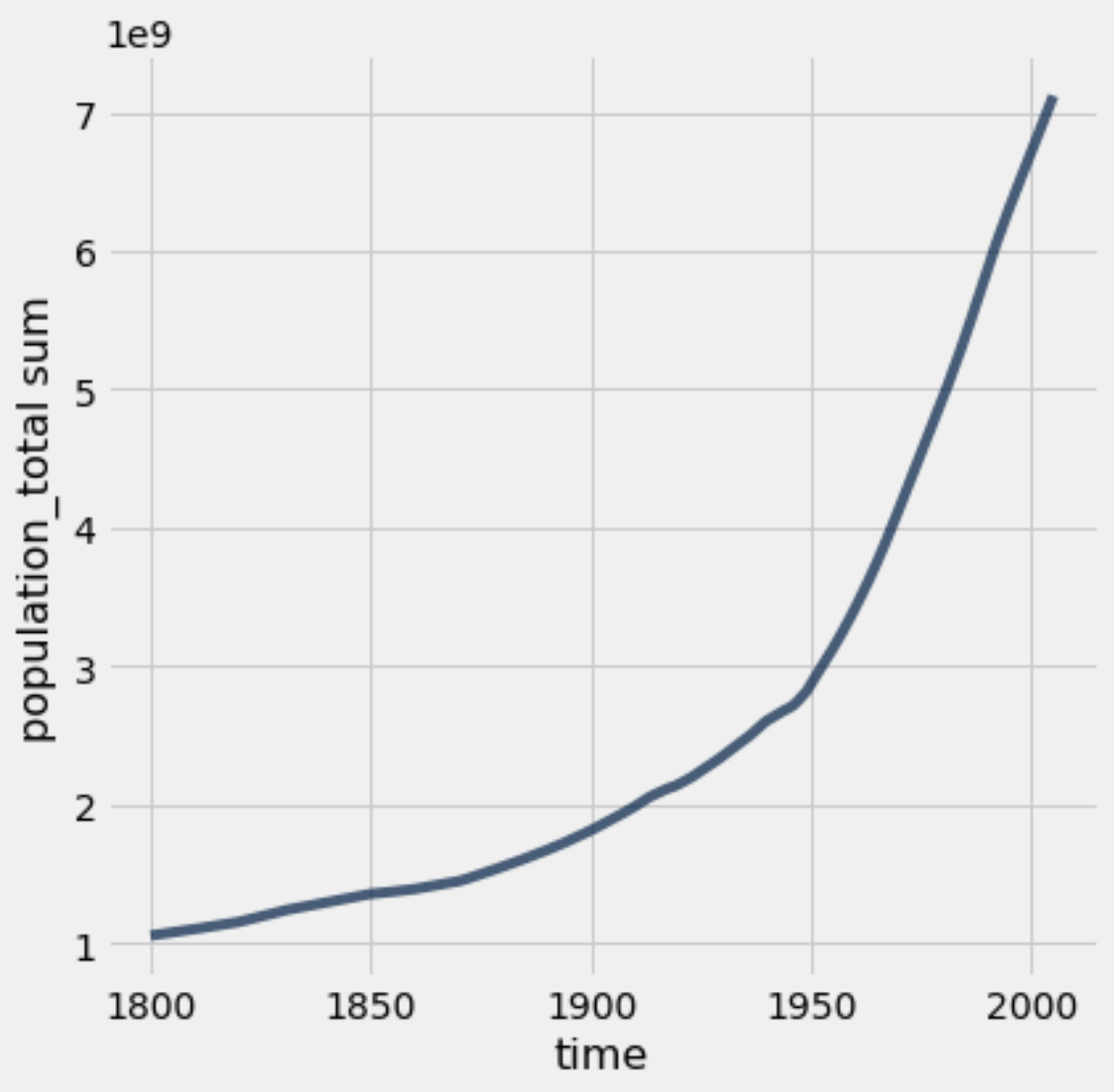
[70]:
```
<gofer.ok.OKTestsResult at 0x7f2157fc6250>
```

1.2 2. 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 next cell to see a line plot of the world population from 1800 through 2005. You might recognize some of the code used!

[71]:
```
population.where('time', are.between(1800, 2006)).drop('geo').group('time',␣
↪sum).plot(0)
```



Question 2.1 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.

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.

[72]:
```
# 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', 2010).sort(2, descending=True).take(np.
 ↪arange(50)).column('geo')
population_of_big_50 = population.where('time', are.above(1959)).where('geo',␣
 ↪are.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', year).drop('time')
    f = fertility.where('time', year).drop('time')
    c = child_mortality.where('time', year).drop('time')
    return p.join("geo", f.join("geo", c))
```

Try calling your function `stats_for_year` on any year between 1960 and 2010 in the cell below. Try to understand the output of `stats_for_year`.

[73]:
```
stats_for_year(1990)
```

[73]:
```
geo  | population_total | children_per_woman_total_fertility |
child_mortality_under_5_per_1000_born
afg  | 12067570         | 7.69                               | 181
arg  | 32729740         | 2.99                               | 27.6
bgd  | 105983136        | 4.55                               | 143.7
bra  | 150393143        | 2.81                               | 61.1
can  | 27662440         | 1.72                               | 8.3
chn  | 1154605773       | 2.43                               | 53.8
cod  | 34962676         | 7.13                               | 186.5
col  | 34271563         | 3.1                                | 35.1
deu  | 78958237         | 1.36                               | 8.5
dza  | 25912364         | 4.76                               | 46.9
… (40 rows omitted)
```

[74]:
```
check('tests/q2_1.py')
```

[74]:
```
<gofer.ok.OKTestsResult at 0x7f2151e000d0>
```

Question 2.2 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: It may be helpful to use the provided `pop_for_year` that computes this total population, then `apply` it to the `decade` column.

2 TERRIBLE var names. I suppose you could say terrible reading comprehension on my part but with a variable called population in a table called pop by decade my gut reaction is to sum the population in each decade but the assignment just want s the 1st year

[75]:
```
def pop_for_year(year):
    return sum(stats_for_year(year).column('population_total'))
```

[76]:
```
decades = Table().with_column('decade', np.arange(1960, 2011, 10))

pop_by_decade = decades.with_column("population", [pop_for_year(i) for i in␣
 ↪range(1960, 2011, 10)])
pop_by_decade.set_format(1, NumberFormatter)
```

[76]:
```
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
```

[77]:
```
check('tests/q2_2.py')
```

[77]:
```
<gofer.ok.OKTestsResult at 0x7f2151bbab10>
```

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.

[78]:
```
countries = Table.read_table('countries.csv').where('country', are.
 ↪contained_in(population.group('geo').column(0)))
countries.select('country', 'name', 'world_6region')
```

[78]:
```
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 2.3 Create a table called `region_counts` that has two columns, `region` and `count`. It should describe the count of how many countries in each region 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.

[79]:
```
# surely my solutionis over enginered
def attach_appropriate_region(tag: str) -> str:
    return countries.where("country", tag).column("world_6region")[0]
```

[80]:
```
temp_tbl = Table().with_columns(
    "geo_1960", stats_for_year(1960).column("geo"),
    "region", stats_for_year(1960).apply(attach_appropriate_region, "geo")
).group("region")
region_counts = Table().with_columns(
    "region", temp_tbl.column("region"),
    "count", temp_tbl.column("count")
)
region_counts
```

[80]:
```
region                  | count
america                 | 8
east_asia_pacific       | 10
europe_central_asia     | 10
middle_east_north_africa | 7
south_asia              | 5
sub_saharan_africa      | 10
```

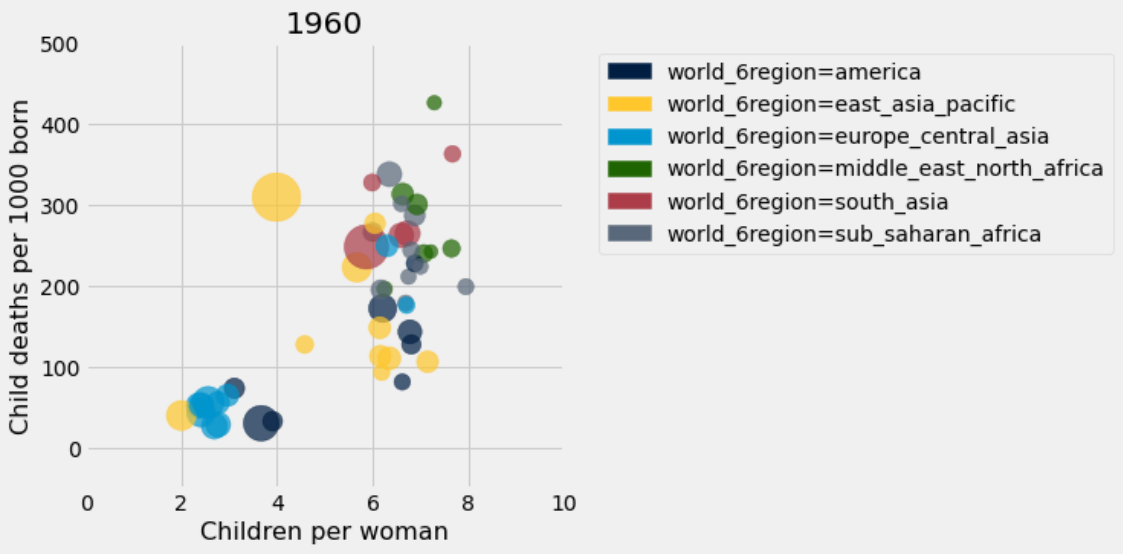
[81]:
```
check('tests/q2_3.py')
```

[81]:
```
<gofer.ok.OKTestsResult at 0x7f215338c210>
```

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?

[55]:
```
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 and then saved.
@cache(None)
def stats_relabeled(year):
    """Relabeled and cached version of stats_for_year."""
```

```
    return stats_for_year(year).relabeled(2, 'Children per woman').relabeled(3,␣
 ↪'Child 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)
```



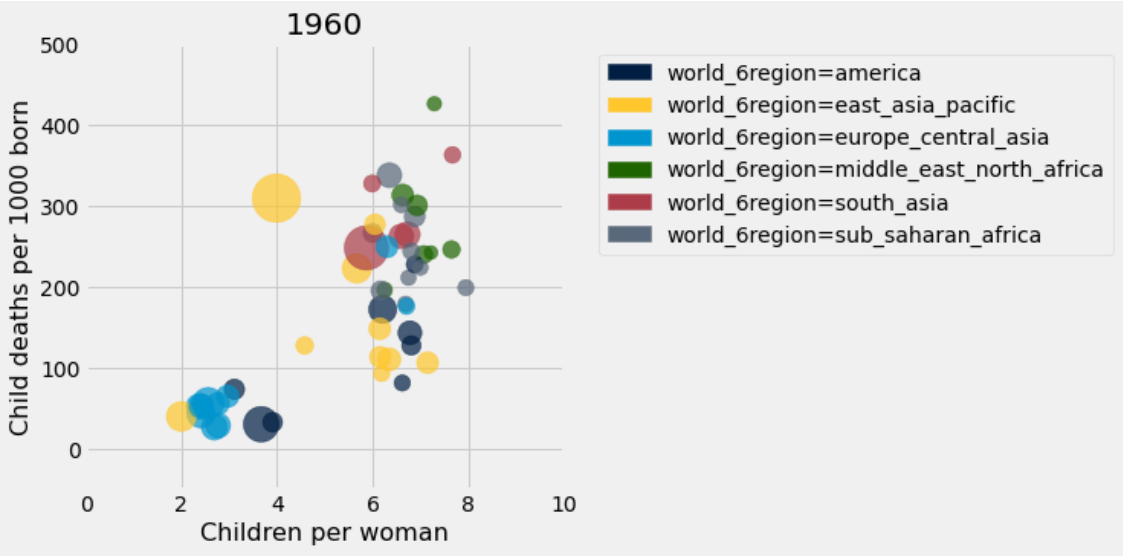
The result of the cell below is interactive. It may take several minutes to run because it computers 55 tables (one for each year). When it's done, a scatter plot and a slider should appear.

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.

[56]:
```
import ipywidgets as widgets

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

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

```
interactive(children=(IntSlider(value=1960, description='year', max=2015,␣
↪min=1960), Output()), _dom_classes=(…
```



2.1 3. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the auto-grader does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this lab. When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

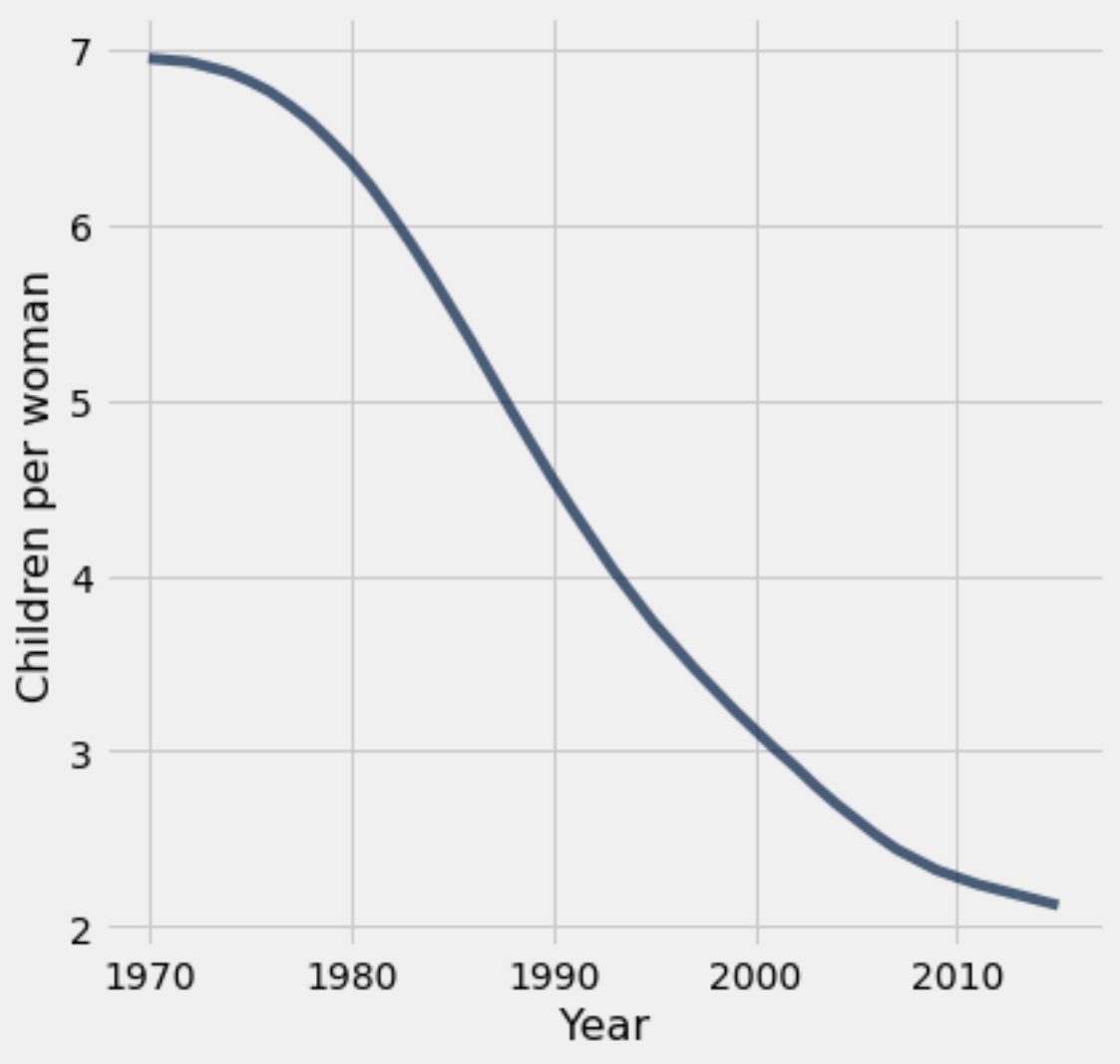
Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

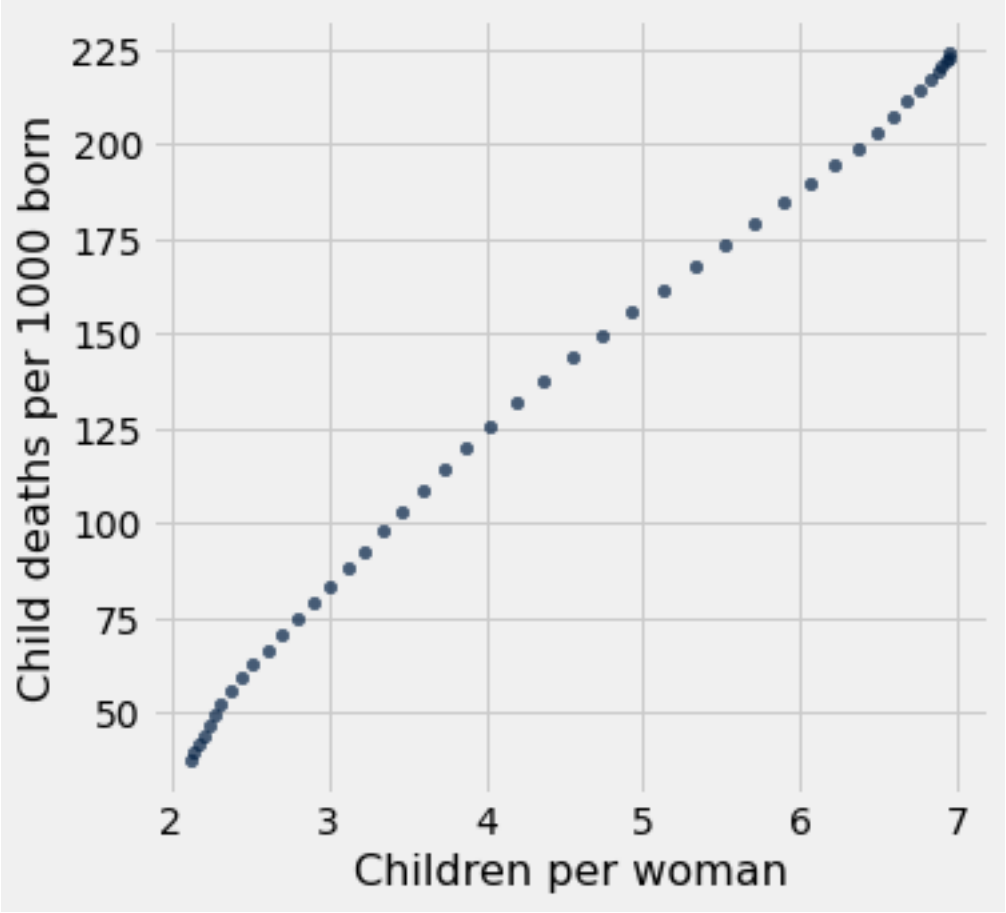
Now is a great time to watch the same data presented by Hans Rosling in a 2010 TEDx talk with smoother animation and witty commentary.

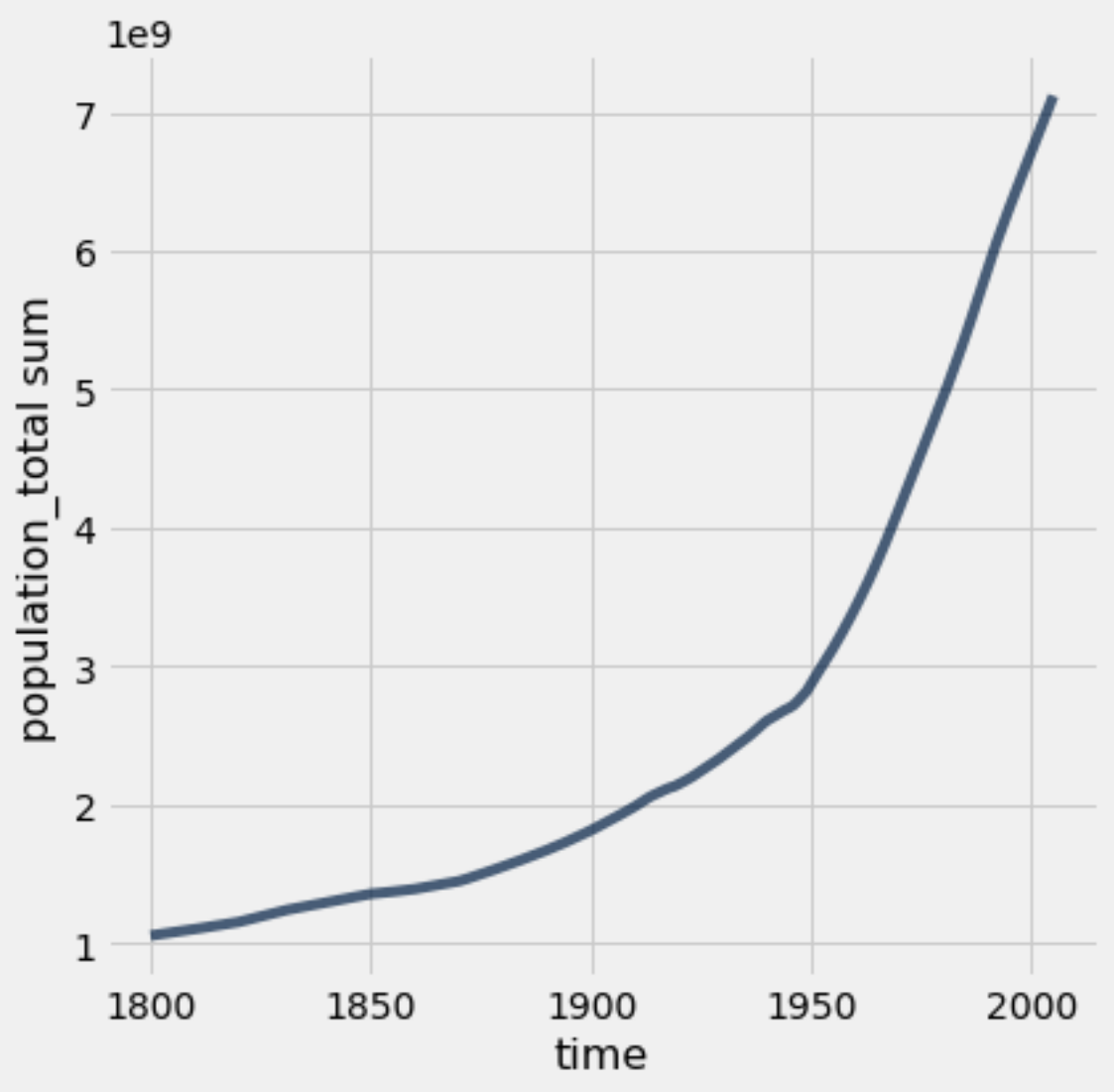
[82]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
```

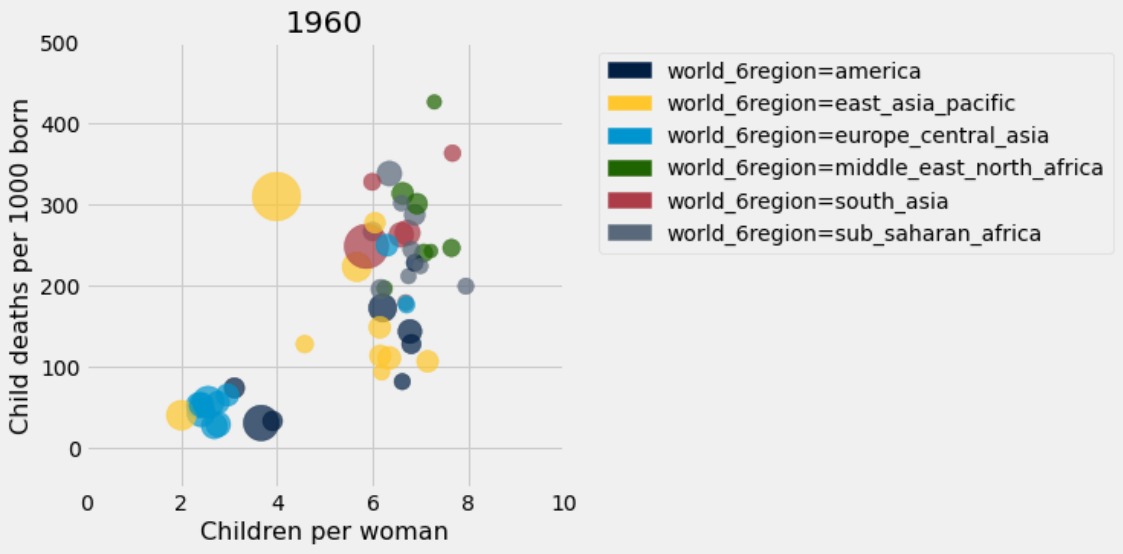
```
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('lab05.ipynb', sorted(glob.glob('tests/q*.py'))))
```

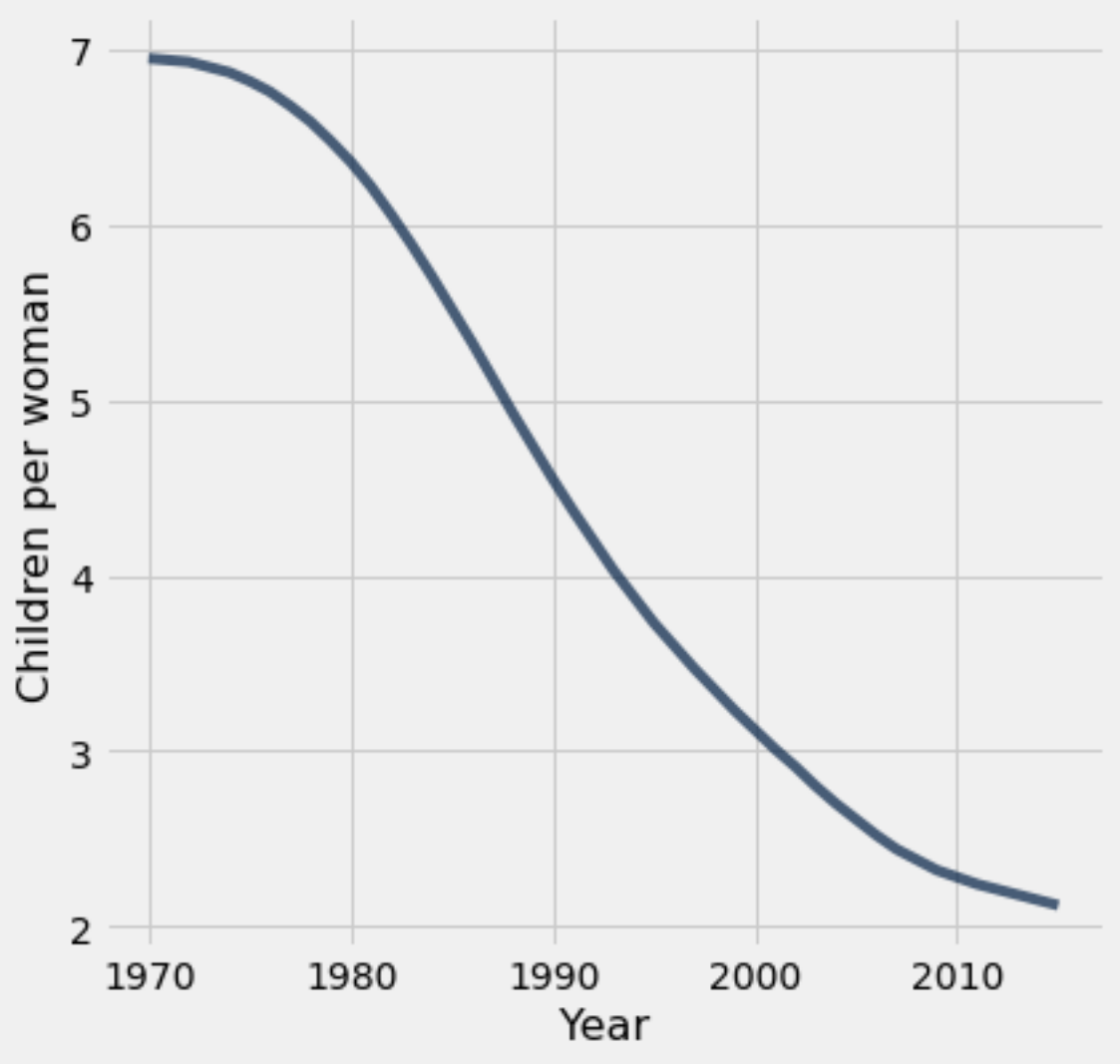
```
['tests/q1_1.py', 'tests/q1_2.py', 'tests/q1_3.py', 'tests/q2_1.py',
'tests/q2_2.py', 'tests/q2_3.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7f21550a2610>

Question 2:

<gofer.ok.OKTestsResult at 0x7f2153cca590>

Question 3:

<gofer.ok.OKTestsResult at 0x7f2153008490>

Question 4:

<gofer.ok.OKTestsResult at 0x7f2153008d10>

Question 5:

<gofer.ok.OKTestsResult at 0x7f2151aa8d90>

Question 6:

<gofer.ok.OKTestsResult at 0x7f2151e64450>

1.0
```

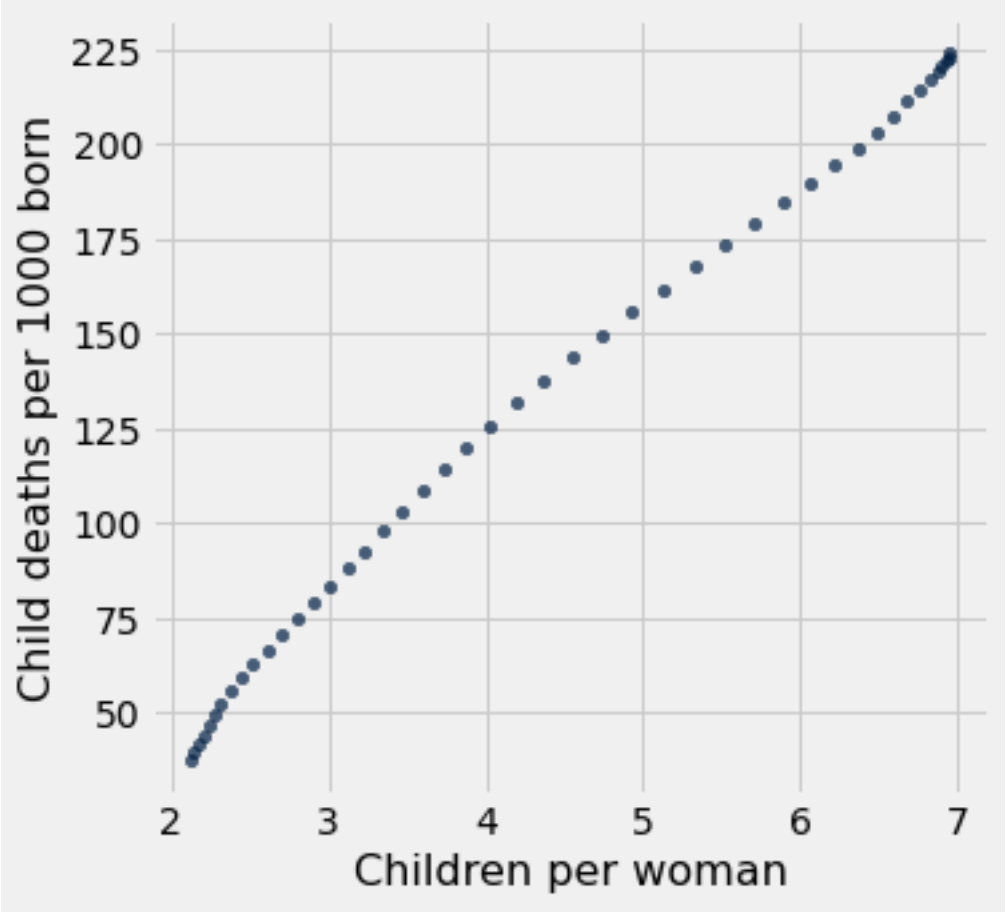

7
6
5
4
3
2
Children per woman
1970
1980
1990
2000
2010
Year

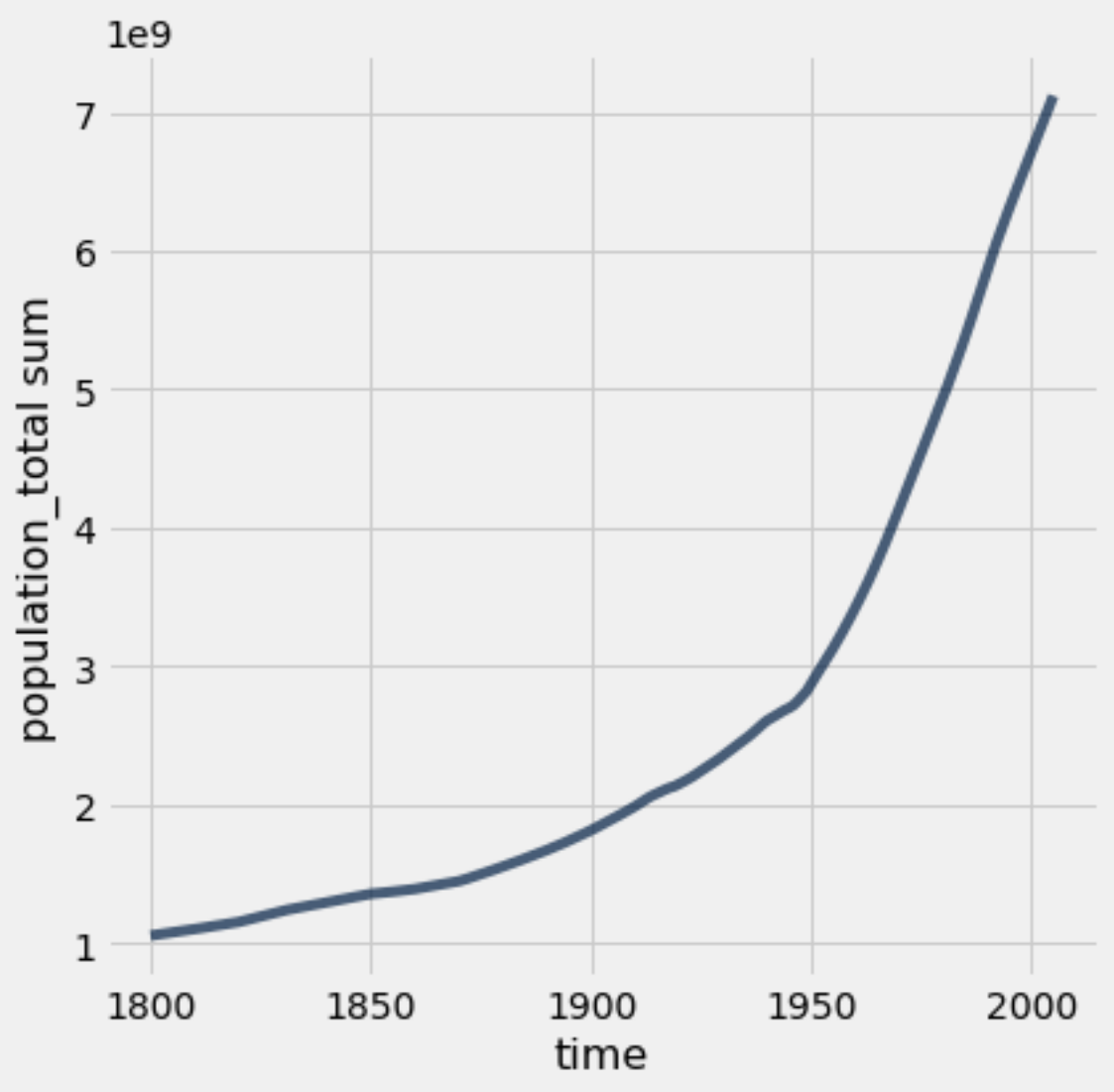

Child deaths per 1000 born
225
200
175
150
125
100
75
50
2
3
4
5
6
7
Children per woman

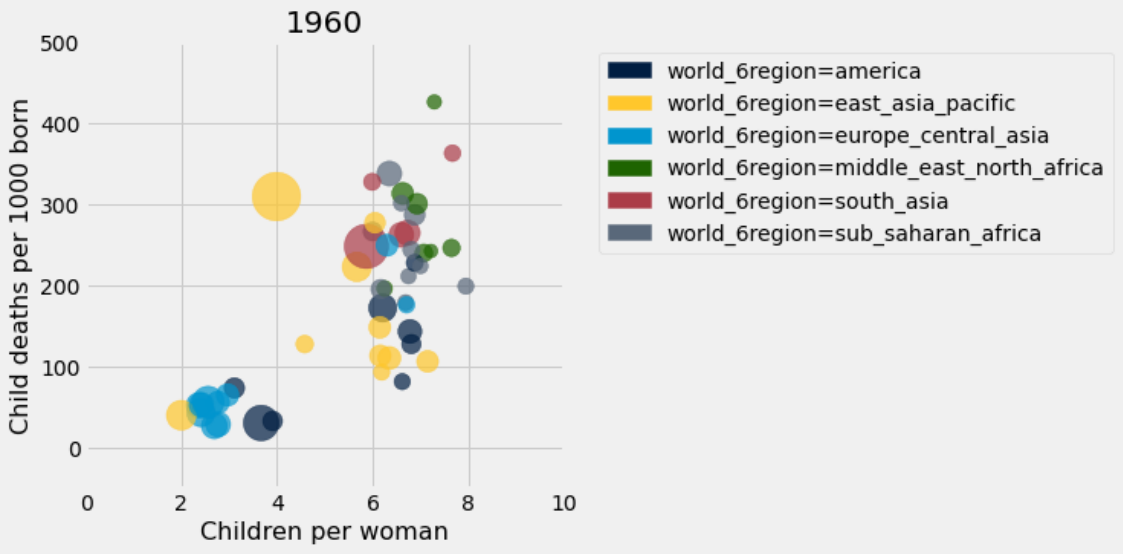

1e9
7
6
5
4
3
2
1
population_total sum
1800
1850
1900
1950
2000
time


1960
Child deaths per 1000 born
Children per woman
0
100
200
300
400
500
0
2
4
6
8
10
world_6region=america
world_6region=east_asia_pacific
world_6region=europe_central_asia
world_6region=middle_east_north_africa
world_6region=south_asia
world_6region=sub_saharan_africa


7
6
5
4
3
2
Children per woman
1970
1980
1990
2000
2010
Year


Child deaths per 1000 born
225
200
175
150
125
100
75
50
2
3
4
5
6
7
Children per woman


1e9
7
6
5
4
3
2
1
population_total sum
1800
1850
1900
1950
2000
time



Name: Allan Gongora

Section: 0131