Homework 4: Functions and Histograms

Please complete this notebook by filling in the cells provided. When you're done:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells.
2. Select `Download as HTML (.html)` from the `File` menu
3. Inspect your file to make sure it looks like it should.
4. Upload your file to Moodle.

Run the cell below to prepare the notebook and the tests. **Passing the automatic tests does not guarantee full credit on any question.** The tests are provided to help catch some common errors, but it is *your* responsibility to answer the questions correctly.

In [25]:
```
# Run this cell to set up the notebook, but please don't change it.
import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)
```

1. Predicting Temperatures

In this exercise, we will try to predict the weather in California using the prediction method discussed in section 8.1 of the textbook. Much of the code is provided for you; you will be asked to understand and run the code and interpret the results.

The US National Oceanic and Atmospheric Administration (NOAA) operates thousands of climate observation stations (mostly in the US) that collect information about local climate. Among other things, each station records the highest and lowest observed temperature each day. These data, called "Quality Controlled Local Climatological Data," are publicly available here.

`temperatures.csv` contains an excerpt of that dataset. Each row represents a temperature reading in Fahrenheit from one station on one day. (The temperature is actually the highest temperature observed at that station on that day.) All the readings are from 2015 and from California stations.

In [26]:
```
temperatures = Table.read_table("~/DS_113_S23/HW/HW_4/temperatures.csv")
temperatures
```

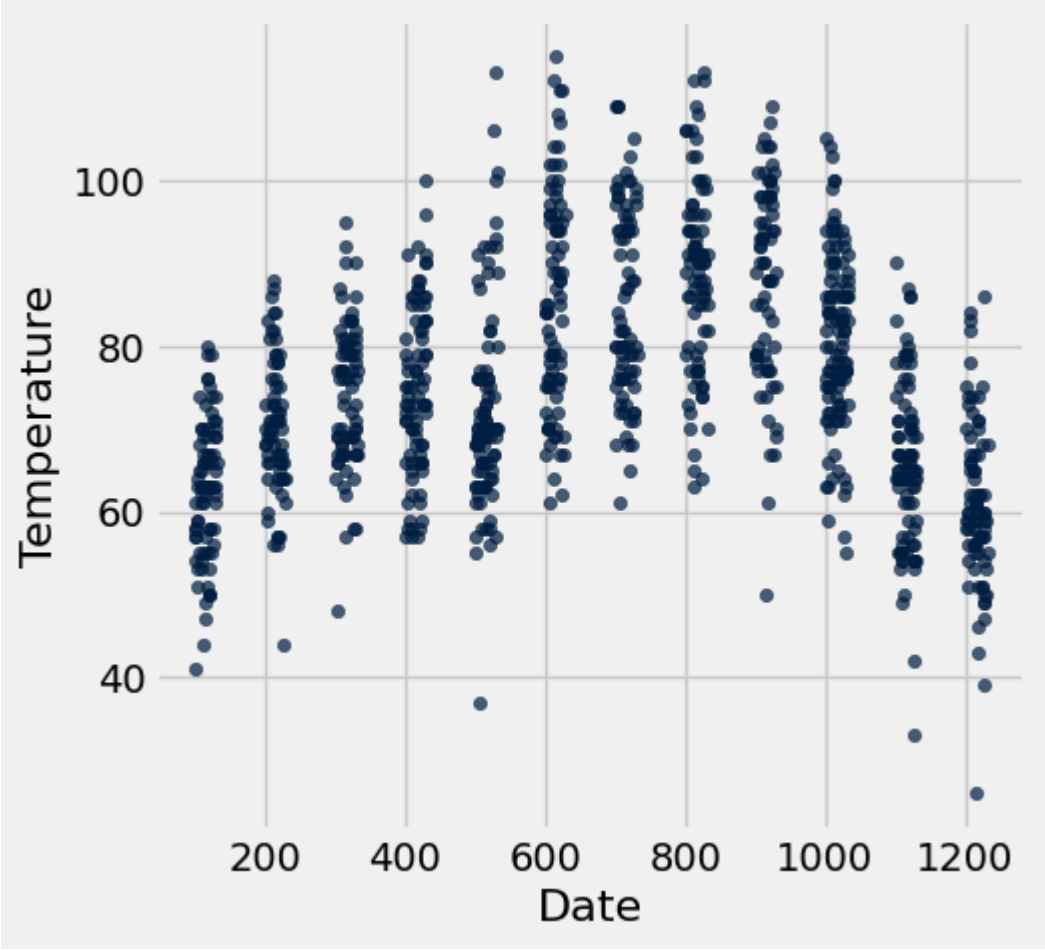
Out[26]:

Temperature	Date	Latitude	Longitude	Station name
67	1013	40.9781	-124.109	Arcata/Eureka
63	811	38.3208	-123.075	Bodega
94	706	39.1019	-121.568	Marysville
59	1211	36.9358	-121.789	Watsonville
111	620	32.8342	-115.579	Imperial
88	821	33.9	-117.25	Riverside
68	606	33.938	-118.389	Los Angeles
66	205	37.2847	-120.513	Merced
89	902	39.49	-121.618	Oroville
105	728	34.8536	-116.786	Daggett

... (990 rows omitted)

Here is a scatter plot:

In [27]:
```
temperatures.scatter("Date", "Temperature")
```



Each entry in the column "Date" is a number in MMDD format, meaning that the last two digits denote the day of the month, and the first 1 or 2 digits denote the month.

Question 1. Why do the data form vertical bands with gaps?

Since the last two digits represent the day of the month, the maximum number that can be the last two digits is 31. Therefore, the column "Date" does not have numbers that have the last two digits exceeding 31, which makes the gap formed by vertical bands.

Let us solve that problem. We will convert each date to the number of days since the start of the year.

In [28]:
```
def get_month(date):
    """The month in the year for a given date.

    >>> get_month(315)
    3
    """
    return int(date / 100)

def get_day_in_month(date):
    """The day in the month for a given date.

    >>> get_day_in_month(315)
    15
    """
    return date % 100

DAYS_IN_MONTHS = Table().with_columns(
    "Month", np.arange(1, 12+1),
    "Days in Month", make_array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,

# A table with one row for each month.  For each month, we have
# the number of the month (e.g. 3 for March), the number of
# days in that month in 2015 (e.g. 31 for March), and the
# number of days in the year before the first day of that month
# (e.g. 0 for January or 59 for March).
DAYS_SINCE_YEAR_START = DAYS_IN_MONTHS.with_column(
    "Days since start of year", np.cumsum(DAYS_IN_MONTHS.column("Days in Mor

def days_since_year_start(month):
    """The number of days in the year before this month starts.

    month should be the number of a month, like 3 for March.

    >>> days_since_year_start(3)
    59
    """
    return DAYS_SINCE_YEAR_START.where("Month", are.equal_to(month))\
                                .column("Days since start of year")\
                                .item(0)

# First, extract the month and day for each reading.
with_month_and_day = temperatures.with_columns(
    "Month", temperatures.apply(get_month, "Date"),
    "Day in month", temperatures.apply(get_day_in_month, "Date"))
# Compute the days-since-year-start for each month and day.
fixed_dates = with_month_and_day.apply(days_since_year_start, "Month") + wit
# Add those to the table.
with_dates_fixed = with_month_and_day.with_column("Days since start of year'
with_dates_fixed
```

Out[28]:

Temperature	Date	Latitude	Longitude	Station name	Days since start of year
67	1013	40.9781	-124.109	Arcata/Eureka	286
63	811	38.3208	-123.075	Bodega	223
94	706	39.1019	-121.568	Marysville	187
59	1211	36.9358	-121.789	Watsonville	345
111	620	32.8342	-115.579	Imperial	171
88	821	33.9	-117.25	Riverside	233
68	606	33.938	-118.389	Los Angeles	157
66	205	37.2847	-120.513	Merced	36
89	902	39.49	-121.618	Oroville	245
105	728	34.8536	-116.786	Daggett	209

... (990 rows omitted)

Question 2. In the cell above, what is the value of this expression?

```
np.cumsum(DAYS_IN_MONTHS.column("Days in Month")) -
DAYS_IN_MONTHS.column("Days in Month")
```

Describe its type and what its value (or the values in it, if it's an array or table) means.

Hint: You can write `np.cumsum?` to get documentation for the function `cumsum`.

In [29]:
```
# You can write code in this cell to help you answer this
# question, if you want to.

np.cumsum(DAYS_IN_MONTHS.column("Days in Month")) - DAYS_IN_MONTHS.column("D
```

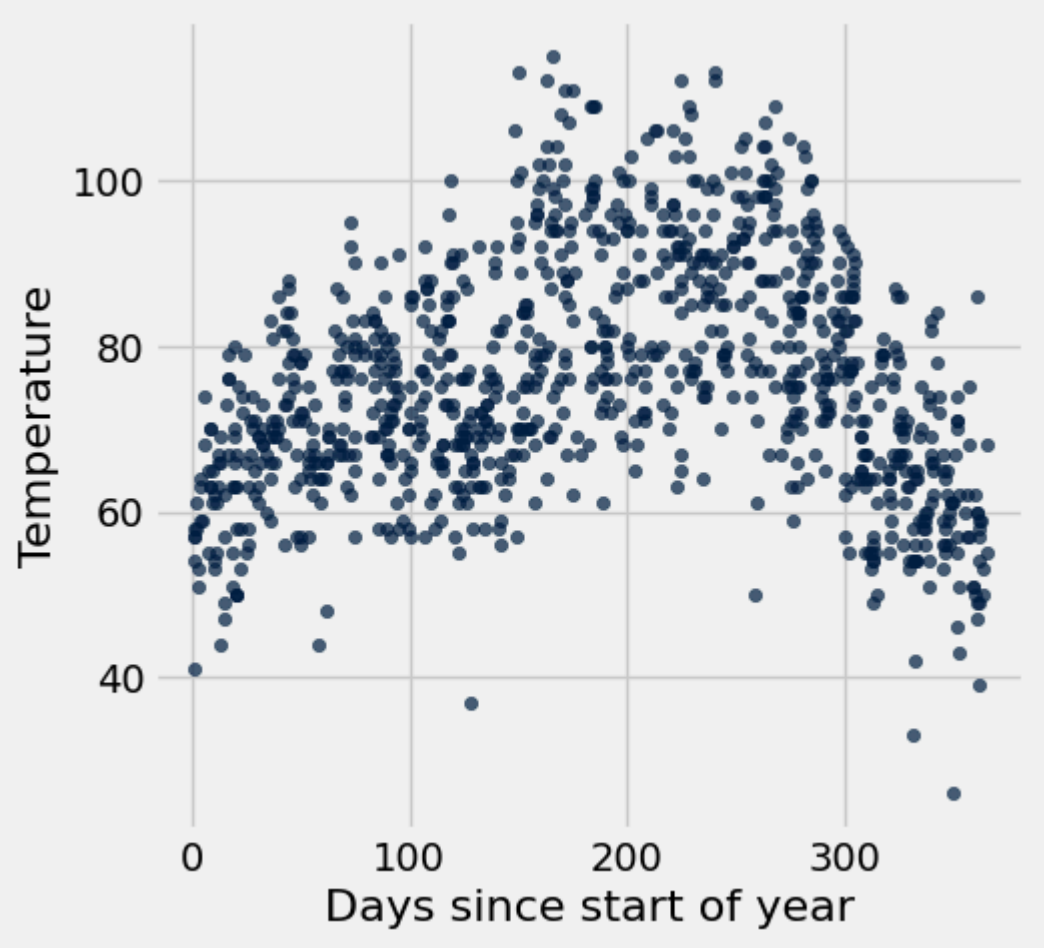
Out[29]:
```
array([  0,  31,  59,  90, 120, 151, 181, 212, 243, 273, 304, 334])
```

Its type is array and it returns an array of numbers related to the days in a year. The first element in the array is 0, which is before the year begins, and the rest of the array adds up the days of the month.

*np.cumsum returns the cumulative sum of the elements along a given axis.

Now we can make a better scatter plot.

In [30]:
```
with_dates_fixed.scatter("Days since start of year", "Temperature")
```



Let's do some prediction. For any reading on any day, we will predict its value using all the readings from the week before and after that day. A reasonable prediction is that the reading will be the average of all those readings. We will package our code in a function.

In [31]:
```
PREDICTION_RADIUS = 7

def predict_temperature(day):
    """A prediction of the temperature (in Fahrenheit) on a given day at som
    """
    nearby_readings = with_dates_fixed.where("Days since start of year", are
    return np.average(nearby_readings.column("Temperature"))
```

Question 3. Suppose you're planning a trip to Yosemite for Thanksgiving break this year, and you'd like to predict the temperature on November 26 (the Saturday after Thanksgiving). Use `predict_temperature` to compute a prediction for a temperature reading on that day.

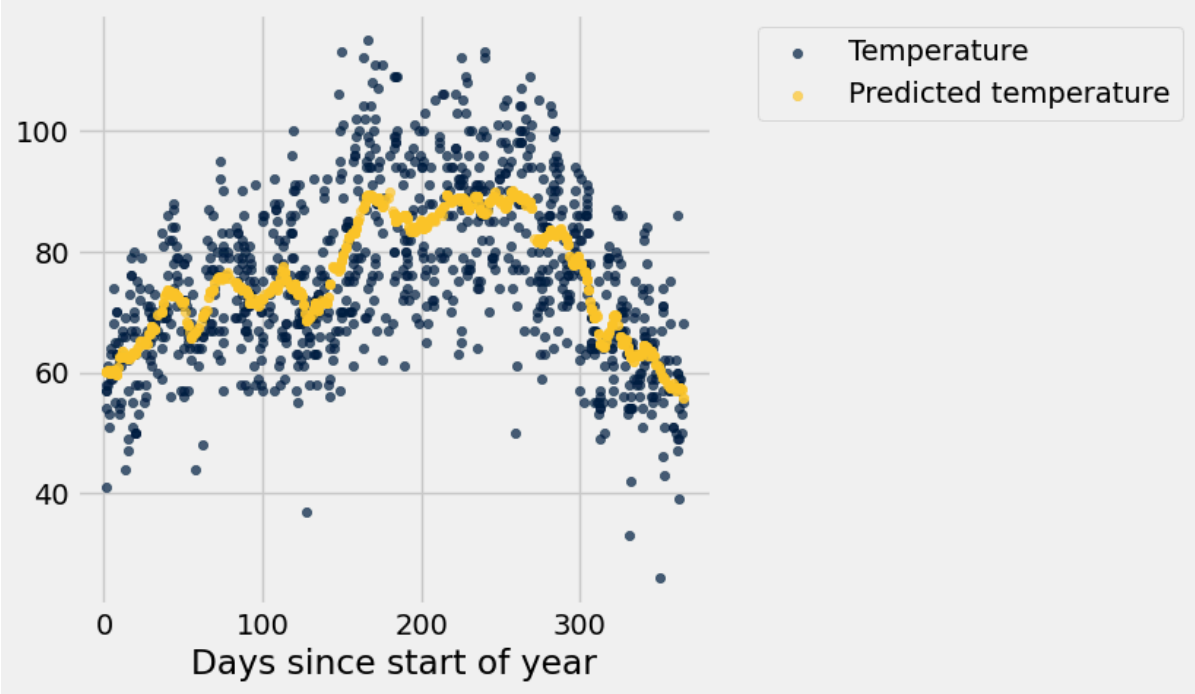
Hint: In addition to `predict_temperature`, another function we wrote earlier will be helpful.

In [32]:
```
## days since the year start (thanksgiving day)
thanksgiving_day = days_since_year_start(get_month(1126)) + get_day_in_month
thanksgiving_day
## predicted temperature on thanksgiving day
thanksgiving_prediction = predict_temperature(thanksgiving_day)
thanksgiving_prediction
```

Out[32]:
```
64.319148936170208
```

Below we have computed a predicted temperature for each reading in the table and plotted both. (It may take a **minute or two** to run the cell.)

In [33]:
```
with_predictions = with_dates_fixed.with_column(
    "Predicted temperature",
    with_dates_fixed.apply(predict_temperature, "Days since start of year"))
with_predictions.select("Days since start of year", "Temperature", "Predicte
                .scatter("Days since start of year")
```



Question 4. How many times was the first line of the body of the function `predict_temperature` (the one that starts with `nearby_readings = ...`) executed when you ran the cell above?

365 times (1 year has 365 days)

Question 5. The scatter plot is called a *graph of averages*. In the example in the textbook, the graph of averages roughly followed a straight line. Is that true for this one? Using your knowledge about the weather, explain why or why not.

For this graph, the graph of averages does not follow a staight line. It is rather a curved line. This is because the graph is about the predicted temperature of a year. There are seasonal changes in a year, which makes the first few days and the last days to have the coldest temperature because those days are winter. And the highest points are the days in summer, which have the highest temperatures in a year.

Question 6. According to the Wikipedia article on California's climate, "[t]he climate of California varies widely, from hot desert to subarctic." Suppose we limited our data to weather stations in a smaller area whose climate varied less from place to place (for example, the state of Vermont, or the San Francisco Bay Area). If we made the same graph for that dataset, in what ways would you expect it to look different?

The graph will also be a curved graph, gradually increasing and at some point gradually decreasing. However, the rate of change, which is the slope, would be less than the

previous graph.

2. Reading Documents

Often, we have to work with data that aren't in CSV format, but instead come in some less nice form. In this exercise, we'll look at the text of some Reuters news reports from 1987. Our dataset doesn't include *all* the news reports from that year, but it includes 1,000 of them. Reuters doesn't say how the articles were selected.

We've put the text of all the articles in a file called `reuters.txt`. The cell below loads that file into a single big string and prints a few thousand characters, which is just enough to see one full article and the start of the next. (Don't try to print the whole thing, because it's very long.)

You will need to download the file 'reuters.txt' and then upload it to the folder where you have this notebook saved.

In [34]:
```
# Just run this cell to load the dataset as one big string of text.
with open('reuters.txt', 'r') as file:
    big_reuters_string = file.read()

print("{:.5000}\n[...]".format(big_reuters_string))
```

```
<reuters cgisplit="PUBLISHED-TESTSET" lewissplit="NOT-USED" newid="12603" o
ldid="21460" topics="YES">
<date> 2-APR-1987 11:11:23.62</date>
<topics><d>livestock</d><d>carcass</d><d>trade</d></topics>
<places><d>usa</d><d>japan</d></places>
<people><d>lyng</d></people>
<orgs></orgs>
<exchanges></exchanges>
<companies></companies>
<unknown>
C G L
f1408reute
u f BC-/LYNG-SETS-TOUGH-U.S.   04-02 0141</unknown>
<text>
<title>LYNG SETS TOUGH U.S. STANCE WITH JAPAN ON BEEF</title>
<dateline>    WASHINGTON, April 2 - </dateline>U.S. Agriculture Secretary R
ichard
Lyng warned Japan that the failure to remove a longstanding
import quota on Japanese beef might spark a protectionist
response in the United States.
    "Given the protectionist mood in the Congress and the
country, if I were a leader in Japan I would certainly be very
concerned...and the failure to remove it (the beef quota) might
be very serious," Lyng told a group of U.S cattlemen.
    Lyng said he and Trade Representative Clayton Yeutter,
during a visit to Japan later this month, will demand "total
elimination" of the beef import quota by April 1988.
    The current dispute with Japan over semiconductor may
strengthen the U.S. stance in farm trade negotiations, Lyng
said, because Japan does not want a trade war with the U.S.
    Lyng dismissed recent statements in Tokyo that Japan might
retaliate against U.S. products as a result of the
semiconductor dispute.
    "They (Japan) aren't going to pick a fight with us," Lyng
said, adding that with its huge bilateral trade surplus Japan
has more to lose in a trade war than the United States.
    Lyng told the U.S. cattlemen that the quota on Japanese
beef imports does not allow consumers there an adequate choice
in food purchases.
    He said in addition to beef, the U.S. will press for
eliminiation of import barriers on Japan's citrus and rice as
well.
    Lyng noted that Japan is the largest buyer of U.S. farm
products, principally grains and soybeans.
 Reuter
</text>
</reuters>***ARTICLE***<reuters cgisplit="TRAINING-SET" lewissplit="TEST" n
ewid="19096" oldid="6932" topics="NO">
<date>19-JUN-1987 08:19:25.45</date>
<topics></topics>
<places><d>usa</d></places>
<people></people>
<orgs></orgs>
<exchanges></exchanges>
<companies></companies>
<unknown>
F
f0606reute
u f BC-DE-LAURENTIIS-COMPANI   06-19 0050</unknown>
<text>
<title>DE LAURENTIIS COMPANIES SEE LOSS ON FILM</title>
```

<dateline>   LOS ANGELES, June 19 - </dateline>De Laurentiis Film Partners LP <DFP>
and De Laurentiis Entertainment Group Inc <DEG> said they will take charges of five mln and 6,500,000 dlrs repsectively on their film "Million Dollar Mystery" due to disappointing box office results.
 Reuter
</text>
</reuters>***ARTICLE***<reuters cgisplit="TRAINING-SET" lewissplit="TRAIN" newid="3541" oldid="8454" topics="YES">
<date>11-MAR-1987 08:16:54.73</date>
<topics><d>bop</d></topics>
<places><d>portugal</d></places>
<people></people>
<orgs></orgs>
<exchanges></exchanges>
<companies></companies>
<unknown>
RM
f0775reute
u f BC-PORTUGUESE-TRADE-DEFI   03-11 0115</unknown>
<text>
<title>PORTUGUESE TRADE DEFICIT NARROWS IN 1986</title>
<dateline>   LISBON, March 11 - </dateline>Portugal's trade deficit narrowed in
1986 to 336.5 billion escudos from 354.8 billion in 1985, according to provisional National Statistics Institute figures.
    Imports totalled 1,412.6 billion escudos and exports 1,076.1 billion compared with 1,326.5 billion and 971.7 billion in 1985.
    Expressed in terms of dollars, imports rose 21.2 pct and exports 26.1 pct and the trade deficit increased by 7.8 pct.
    In its first year as a member of the European Community, Portugal recorded a deficit of 98.1 billion escudos in its trade with the other Community states compared with a deficit of 2.4 billion escudos in 1985.
    Imports from the EC in 1986 totalled 830.2 billion escudos, while exports to the Community were 732.1 billion, compared with 609.5 billion and 607.1 billion the previous year.
    Portugal's deficit with Spain was 83.2 billion escudos against 57.7 billion in 1985, with Italy it was 70.4 billion against 30.3 billion, and with West Germany 40.5 billion against 19.1 billion.
 REUTER
</text>
</reuters>***ARTICLE***<reuters cgisplit="TRAINING-SET" lewissplit="TRAIN" newid="10003" oldid="15524" topics="NO">
<date>26-MAR-1987 12:15:35.31</date>
<topics></topics>
<places><d>usa</d><d>uk</d><d>switzerland</d><d>belgium</d><d>luxembourg</d></places>
<people></people>
<orgs></orgs>
<exchanges></exchanges>
<companies></companies>
<unknown>
F
f1735reute
r f BC-BIOGEN-<BGNF>-GETS-PA   03-26 0103</unknown>
<text>
<title>BIOGEN <BGNF> GETS PATENT FROM EUROPEAN OFFICE</title>

```
<dateline>    CAMBRIDGE, Mass., March 26 - </dateline>Biogen Inc said the E
uropean
Patent Office granted it a patent covering certain proteins
used to produce a hepatitis B vaccine through genetic
engineering techniques.
    Robert Gottlieb, Biogen spokesman, said the company has
licensed the vacci
[...]
```

There's a bunch of weird text for each article. Each article is separated from its neighbors by the string `"***ARTICLE***"`.

Question 1. Use the String method `split` to make an array of the text of all the articles. That is, each entry of this array should be the text of one article. Put that array in a new table called `reuters` as a column with the name "Raw text".

Hint: When you split the articles correctly, you should see that each article starts with `"<reuters..."` and ends with `"...</reuters>"`. There should be 1,000 articles.

Hint 2: As an example, `"steamcleaner".split("ea")` is the same as `make_array('st', 'mcl', 'ner')`. So you want to split the big string that contains all the data, splitting with the text `"***ARTICLE***"`.

Note: The text will not actually show up in the table. It will look empty, but it is not.

In [35]:
```
reuters = Table().with_column(
        "Raw text", (big_reuters_string.split("***ARTICLE***"))
    )
reuters
```

Out[35]:

Raw text

... (990 rows omitted)

Each article has a line containing its title that looks something like this:

```
    <title>LYNG SETS TOUGH U.S. STANCE WITH JAPAN ON BEEF</title>
```

You could find that yourself for a few of the articles, but it would be very tedious to do it for all 1,000 articles. So we'll write code to do it instead.

Question 2. Below, we've written a function called `get_text_in_markers` that will help you find the title text for an article. Use it to write a function called `get_title`, which is also documented below.

In [36]:
```
# This function is provided for you to use.  Read at least
# its documentation (the stuff at the beginning in red).
# You can also type in get_text_in_markers? somewhere and
# run it to see the documentation in a slightly nicer form.
```

```
# We haven't used any tools you haven't seen yet, so it
# wouldn't hurt to read the code itself, too.
def get_text_in_markers(text, marker):
    """Finds the part of a piece of text that's between specified markers.

    Parameters
    ----------
    text : str
        The text in which you want to find something.
    marker : str
        The name of the marker that delimits the part of the
        text you want to grab.  In the text itself, this string
        will be surrounded by "<>" or "</>", but don't include
        those angle brackets in this argument.

    Returns
    -------
    str
        The parts of the text that are inside the markers.

    Examples
    --------
    >>> get_text_in_markers("stuff <interesting>yay exciting</interesting> m
    'yay exciting'
    """
    start_marker = "<{}>".format(marker)
    end_marker = "</{}>".format(marker)
    split_before = np.array(text.split(start_marker))
    marker_text_and_after = split_before.item(1)
    split_on_end_marker = np.array(marker_text_and_after.split(end_marker))
    return split_on_end_marker.item(0)

# Fill in this function.
def get_title(article_text):
    """Takes the text of an article and returns its title."""
    ## title has the marker <title>
    title = get_text_in_markers(article_text,"title")
    return title

# When you're done, this should produce 'LYNG SETS TOUGH U.S. STANCE WITH JA
get_title(reuters.column("Raw text").item(0))
```

Out[36]: 'LYNG SETS TOUGH U.S. STANCE WITH JAPAN ON BEEF'

Question 3. Now use your function to find the title of every article in `reuters`. Create a new table called `with_titles` that's a copy of `reuters` with an extra column named "Title" that contains these titles.

Note: This might take a few seconds to run.

In [37]:
```
with_titles = reuters.with_column("Title", (reuters.apply(get_title, "Raw te
with_titles
```

Out[37]:

Raw text	Title
	DE LAURENTIIS COMPANIES SEE LOSS ON FILM
	PORTUGUESE TRADE DEFICIT NARROWS IN 1986
	BIOGEN GETS PATENT FROM EUROPEAN OFFICE
	MOODY'S MAY DOWNGRADE RESORTS INTERNATIONAL
	RECORD N.Z. FUTURES VOLUMES TRADED IN FEBRUARY
	SWISS SIGHT DEPOSITS RISE 743.7 MLN FRANCS
	IRAN SAYS IT INTENDS NO THREAT TO GULF SHIPPING
	BUSINESSLAND COMPLETES OFFERING

... (990 rows omitted)

Now we'll go through a similar process to get the date of each article. In each article, the date is on its own line, separated from the rest of the article by `<date>` and `</date>` markers. You can check one of the articles for an example.

Question 4. Write a function called `get_date`. It should take as its argument the whole text of an article and return the date. The date should be just the day of the year (so January 1 is day 1, and February 1 is day 32, since January has 31 days). Note that all the articles are from the year 1987, so the year is irrelevant.

We've written a function called `date_string_to_day` that will help you do this.

In [38]:
```
# This function is provided for you to use.  Read at least
# its documentation (the stuff at the beginning in red).
def date_string_to_day(date_string):
    """Converts a string that looks like a date into the day of the year.

    Parameters
    ----------
    date_string : str
        Text that contains a date in any reasonable format.
        For example, "September 13, 1994" or "9/13/94" or
        "13-SEP-1994 15:02:20.00" all work.

    Returns
    -------
    int
        The day of the year that the date represents.

    Examples
    --------
    >>> date_string_to_day("January 3, 2016")
    3

    >>> date_string_to_day("February 4, 2000")
    35
    """
    from dateutil import parser
    import re
    # Some of the Reuters dates have extraneous text at the end.
    # This removes that text.
    date_part = re.sub(" [A-Z]*$", "", date_string)
```

```
    try:
        date = parser.parse(date_part)
    except:
        print("Failed on", date_string)
    day_in_year = date.timetuple().tm_yday
    return day_in_year

# Fill in this function.
def get_date(article_text):
    """Takes the text of an article and returns its date."""
    ## date has the marker <date>
    date_text = get_text_in_markers(article_text,"date")
    day_in_year = date_string_to_day(date_text)
    return day_in_year

# When you're done, this should produce 92.
get_date(reuters.column("Raw text").item(0))
```

Out[38]: 92

Question 5. Use your function to find the date of every article in `with_titles`. Create a new table called `with_dates` that's a copy of `with_titles` with an extra column named "Date" that contains the dates.

In [39]:
```
with_dates = with_titles.with_column("Date", (reuters.apply(get_date, "Raw t
with_dates
```

```
/opt/tljh/user/lib/python3.9/site-packages/dateutil/parser/_parser.py:1207:
UnknownTimezoneWarning: tzname F identified but not understood.  Pass `tzin
fos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "
/opt/tljh/user/lib/python3.9/site-packages/dateutil/parser/_parser.py:1207:
UnknownTimezoneWarning: tzname RM identified but not understood.  Pass `tzi
nfos` argument in order to correctly return a timezone-aware datetime.  In
a future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "
/opt/tljh/user/lib/python3.9/site-packages/dateutil/parser/_parser.py:1207:
UnknownTimezoneWarning: tzname G identified but not understood.  Pass `tzin
fos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "
/opt/tljh/user/lib/python3.9/site-packages/dateutil/parser/_parser.py:1207:
UnknownTimezoneWarning: tzname V identified but not understood.  Pass `tzin
fos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "
/opt/tljh/user/lib/python3.9/site-packages/dateutil/parser/_parser.py:1207:
UnknownTimezoneWarning: tzname C identified but not understood.  Pass `tzin
fos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  warnings.warn("tzname {tzname} identified but not understood.  "
```

Out[39]:

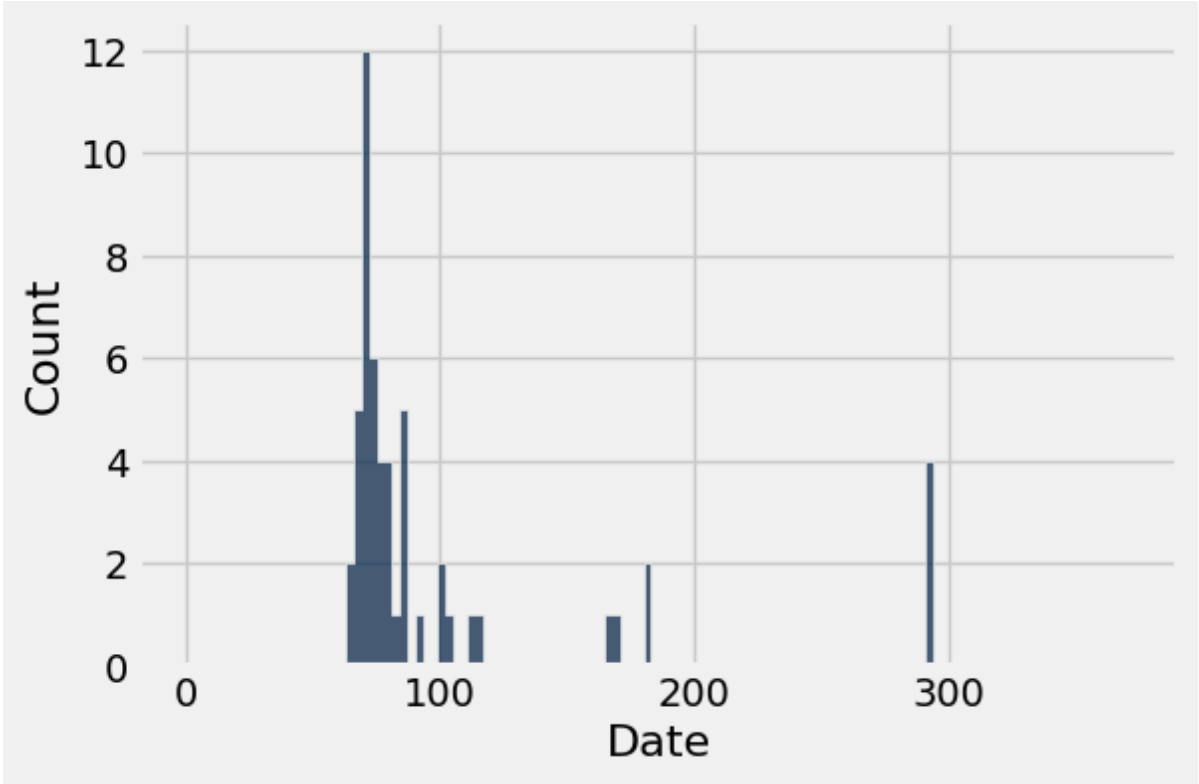
Raw text	Title	Date
DE LAURENTIIS COMPANIES SEE LOSS ON FILM		170
PORTUGUESE TRADE DEFICIT NARROWS IN 1986		70
BIOGEN GETS PATENT FROM EUROPEAN OFFICE		85
MOODY'S MAY DOWNGRADE RESORTS INTERNATIONAL		68
RECORD N.Z. FUTURES VOLUMES TRADED IN FEBRUARY		61
SWISS SIGHT DEPOSITS RISE 743.7 MLN FRANCS		82
IRAN SAYS IT INTENDS NO THREAT TO GULF SHIPPING		81
BUSINESSLAND COMPLETES OFFERING		70

... (990 rows omitted)

Question 6. There was a series of earthquakes in Ecuador on March 6, 1987. Most Reuters news stories about Ecuador from that period were related to the earthquake or its political and economic consequences. Find out when Reuters reported on the earthquake by making a histogram of all the dates of the articles whose *titles* include the word `"ECUADOR"`. Use bins of width 3.

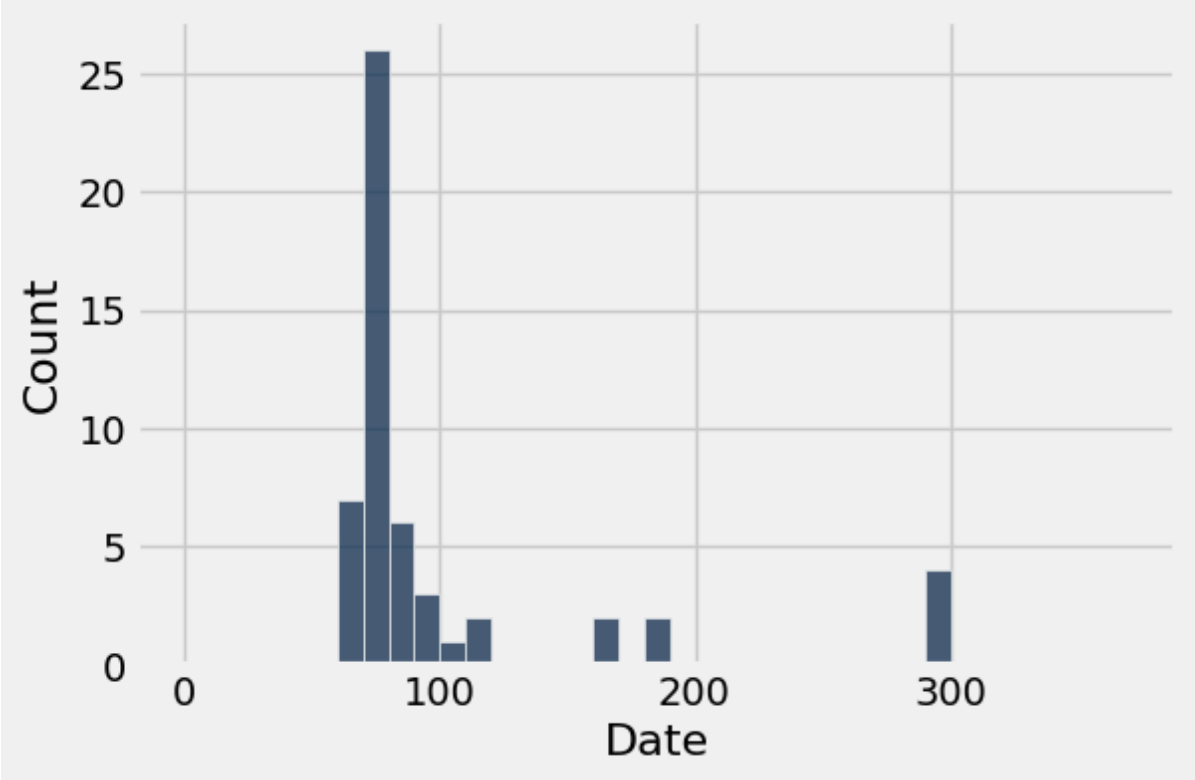
Hint: The function `are.containing` creates a predicate that matches strings that contain a given string. You can find its documentation by running `are.containing?`.

In [40]:
```
# Use these bins:
bins = np.arange(0, 375, 3)

## find titles including the word "ECUADOR"
ecuador = with_dates.where("Title",are.containing("ECUADOR"))

## make a histogram of all the dates with the title including "ECUADOR"
ecuador.hist("Date", bins= np.arange(0, 375, 3), normed = False)
```



Question 7. Make another histogram of the same data, but this time using different bins. The first bin should start at day 0, and each bin should have a width of 10 days. Then, **using only your own inspection of the histogram (and no other Python code)**, estimate the proportion of Ecuador articles that were reported between days 50 and 100 of the year (including day 50 but not day 100). (The proportion should be out of the total number of articles whose titles include "ECUADOR".) Give that number the name `proportion_50_to_100`.

In [41]:
```
# Make a histogram as described above.  (Be sure to use the
# bins described in the question.)
ecuador.hist("Date", bins= np.arange(0, 375, 10), normed = False)

# By inspecting your histogram, estimate the proportion of
# Ecuador articles that were reported between days 50 and 100
# of the year.  (It's hard to get exactly the right answer
# from a histogram like this, so it's okay if your answer is
# off by a little bit.)
proportion_50_to_100 = 37
```



Question 8. Your histogram should show several long gaps in coverage about Ecuador during the year. By exploring the dataset, try to explain this. Use the code cell below for your explorations.

In [42]:
```
ecuador["Title"]
ecuador.where("Date",are.above(150))
```

Out[42]:

	Raw text	Title	Date
		BALTEK BUYS ECUADORIAN PLANT	292
		ECUADOR POSTS 71.3 MLN DLR 8-MTH TRADE DEFICIT	293
		ECUADOR TO PRODUCE ABOVE OPEC QUOTA - MINISTER	180
		VENEZUELA BACKS INCREASE IN ECUADOR OPEC QUOTA	292
		ECUADOR HAS TRADE SURPLUS IN FIRST FOUR MONTHS	169
		ECUADOR'S CEPE NAMES NEW HEAD	293
		IMF APPROVES 48 MLN DLR LOAN TO ASSIST ECUADOR	180

Since there was a series of earthquakes in Ecuador on March 6, the articles from day 60 to day 120 are related Ecuador's earthquakes. The outliers, which are the articles written around day 160-170, day 180-190, and day 290-300 are not related to the earthquakes. Because those were off-topics about the earthquake, they have some gaps from the data about the earthquake.

3. Causes of Death in California

This exercise is designed to give you practice using the Table method `group`.

We'll be looking at a dataset from the California Department of Public Health (available here) that records the cause of death (as recorded on a death certificate) for everyone who died in California from 1999 to 2013. The data are in

Raw text Title Date

the file `causes_of_death.csv.zip`. Each row records the number of deaths by one cause in one year in one ZIP code.

To make the file smaller, we've compressed it; run the next cell to unzip and load it. **You will need to download the .zip file and upload it to the same folder as this notebook.**

In [43]:
```
!unzip -o causes_of_death.csv.zip
causes = Table.read_table('causes_of_death.csv')
causes
```

```
Archive:  causes_of_death.csv.zip
  inflating: causes_of_death.csv
```

Year	ZIP Code	Cause of Death	Count	Location
1999	90002	SUI	1	(33.94969, -118.246213)
1999	90005	HOM	1	(34.058508, -118.301197)
1999	90006	ALZ	1	(34.049323, -118.291687)
1999	90007	ALZ	1	(34.029442, -118.287095)
1999	90009	DIA	1	(33.9452, -118.3832)
1999	90009	LIV	1	(33.9452, -118.3832)
1999	90009	OTH	1	(33.9452, -118.3832)
1999	90010	STK	1	(34.060633, -118.302664)
1999	90010	CLD	1	(34.060633, -118.302664)
1999	90010	DIA	1	(34.060633, -118.302664)

... (320142 rows omitted)

The causes of death in the data are abbreviated. If you want to know what the abbreviations mean, we've provided a table called `abbreviations.csv`.

In [44]:
```
abbreviations = Table.read_table('abbreviations.csv')
abbreviations.show()
```

Raw text Title Date

Cause of Death	Cause of Death (Full Description)
AID	Acquired Immune Deficiency Syndrome (AIDS)
ALZ	Alzheimer's Disease
CAN	Malignant Neoplasms (Cancers)
CLD	Chronic Lower Respiratory Disease (CLRD)
CPD	Chronic Obstructive Pulmonary Disease (COPD)
DIA	Diabetes Mellitus
HIV	Human Immunodeficiency Virus Disease (HIVD)
HOM	Homicide
HTD	Diseases of the Heart
HYP	Essential Hypertension and Hypertensive Renal Disease
INJ	Unintentional Injuries
LIV	Chronic Liver Disease and Cirrhosis
NEP	Kidney Disease (Nephritis)
OTH	All Other Causes
PNF	Pneumonia and Influenza
STK	Cerebrovascular Disease (Stroke)
SUI	Intentional Self Harm (Suicide)

Question 1. Find the top 5 causes of death in California over the entire period covered by the data. To do that, create a table with one row for each of the top 5 causes of death, a column called "Cause of Death", and a column called "Count" that records the total number of deaths due to that cause. Sort it in descending order by count, and call it `top_5_causes`.

In [45]:
```
## group the data by causes of death and
 select only causes and the sum of counts
cause_of_death = causes.group("Cause of D
                    eath",np.sum).select(
    "Cause of Death","Count sum").sort("C
                ount sum", descending=True)
                      cause_of_death.show()

## make a new table with the top 5 causes
                                    of death
      top5_causes = Table().with_columns(
    "Cause of Death",(cause_of_death["Cau
                      se of Death"][:5]),
    "Count sum", (cause_of_death["Count s
                                    um"][:5]))
                                  top5_causes
```

Raw text Title Date

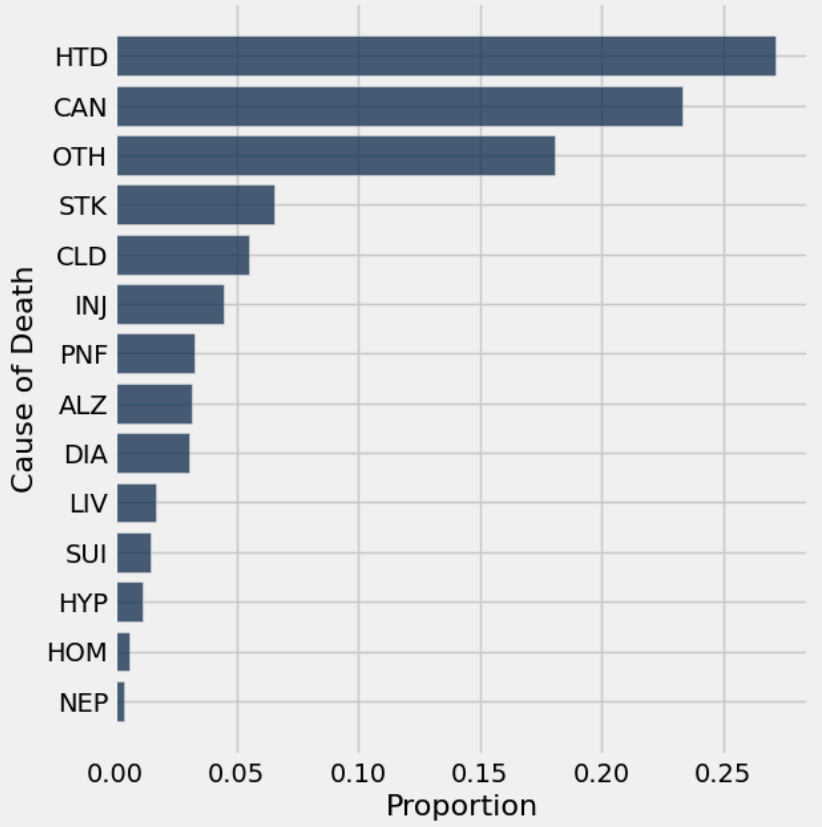
Cause of Death	Count sum
HTD	957108
CAN	822906
OTH	637764
STK	231897
CLD	194961
INJ	157313
PNF	115926
ALZ	111178
DIA	106960
LIV	60526
SUI	52572
HYP	41251
HOM	21336
NEP	14338

Cause of Death	Count sum
HTD	957108
CAN	822906
OTH	637764
STK	231897
CLD	194961

Question 2. Create a bar chart that displays the *proportion of all deaths* by each cause.

In [46]:
```
cause_of_death = cause_of_death.with_colu
mn("Proportion", (cause_of_death["Count s
  um"]/sum(cause_of_death["Count sum"])))
cause_of_death.barh("Cause of Death","Pro
                                portion")
```

Raw text Title Date



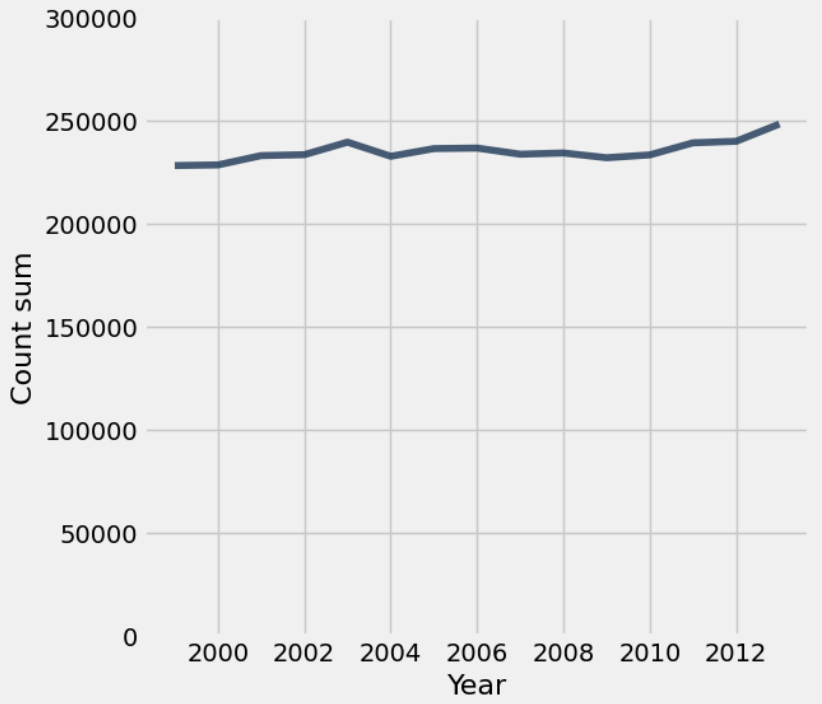
Question 3. Create a plot of the total number of deaths per year in California.

Hint: Use the Table method `plot`. The first argument is the name or index of the column to put on the horizontal axis.

In [47]:
```
## group the data by years and select yea
                    rs and the sum of counts
year_cause = causes.group("Year",np.sum).
                  select("Year","Count sum")
# This line will make the vertical axis s
                  tart at 0.  You can remove
# it if you want to see the default plot,
                  which is more zoomed-in.
                   year_cause.plot("Year")
                       plt.ylim(0, 300000)
```

```
(0.0, 300000.0)
```

Raw text Title Date



Question 4. You should see that deaths have increased a little over time, though not uniformly. How would you explain that? Describe a dataset you'd like to see to test whether your explanation is valid.

The graph above shows the sum of death counts plotted against year. The count sum have increase, but they are between 220000 and 250000 for every year, so the change is little. Also, the count sum for deaths have not uniformly increased, because we can see some parts in the line that decreases (eg. line decreases between year 2003 and 2004).

To see whether the explanation is valid, we can sort the table "year_cause". If the count sum increased uniformly, we can see the years in order.

In [48]:
```
year_cause.sort("Count sum")
```

	Raw text	Title	Date

Year	Count sum
1999	227965
2000	228281
2009	231764
2004	232464
2001	232790
2010	233143
2002	233246
2007	233467
2008	234072
2005	236196

... (5 rows omitted)

As we can see from the sorted table, year 2009 and 2010 have lower count sums than the earlier years. Therefore, the deaths have increased a little over time, though not uniformly.