YData: Introduction to Data Science

Class 13: Data visualization with seaborn

Overview

Quick review of data visualization with matplotlib

Data visualization with seaborn

If there is time

- Text manipulation

Announcement: Homework 6

Homework 6 has been posted!

It is due on Gradescope on Sunday March 5th at 11pm

- **Be sure to mark each question on Gradescope along with the page that has the answers!**

How was homework 5?

Midterm exam

Thursday March 9th in person during regular class time

- Exam is on paper

As part of homework 6, you will post a practice problem to Ed

- Ideally do this soon
- I will take one of these problems and put it on the exam

A practice exam will be posted soon

Midterm exam “cheat sheet”

You are allowed an exam “cheat sheet”

One page, double sided, that contains **only code**

- No code comments allowed

Cheat sheet must be on a regular 8.5 x 11 piece of paper

- Your name on the upper left of both sides of the paper

Strongly recommend making a typed list of all functions discussed in class and on the homework

- This will be useful beyond the exam

You must turn in your cheat sheet with the exam

- Failure to do so will result in a 20 point deduction

Review of data visualization!

Review of data visualization!

*Statistical projections which **speak to the senses without fatiguing the mind**, possess the advantage of fixing the attention on a great number of important facts.*

—Alexander von Humboldt, 1811

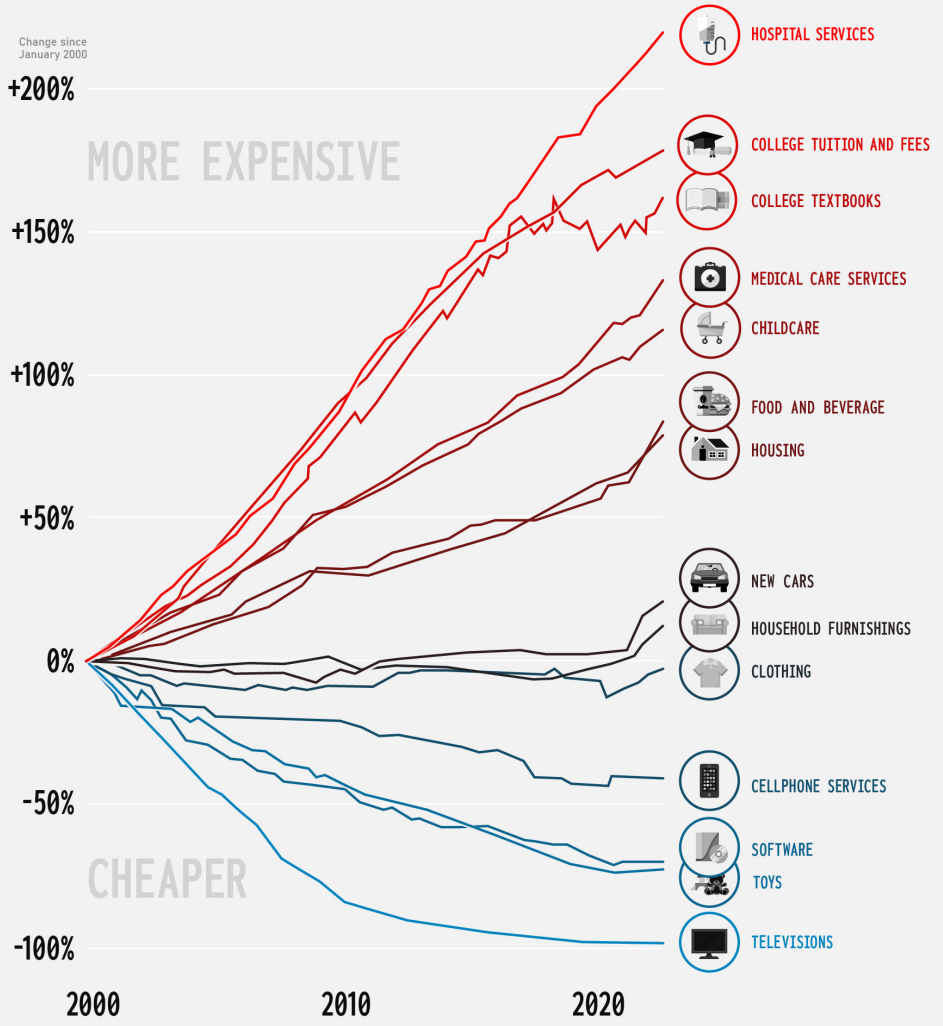
Let’s take a few minutes to explore the data visualizations you found and that you created as part of homework 5…

Line graphs

Price Changes

OF CONSUMER GOODS AND SERVICES

Broadly speaking, price levels have increased by 74% since 2000. That said, the actual numbers vary wildly depending on the type of good or service. Many consumer goods like toys and TVs have gotten cheaper, while critical categories like healthcare and education have skyrocketed.



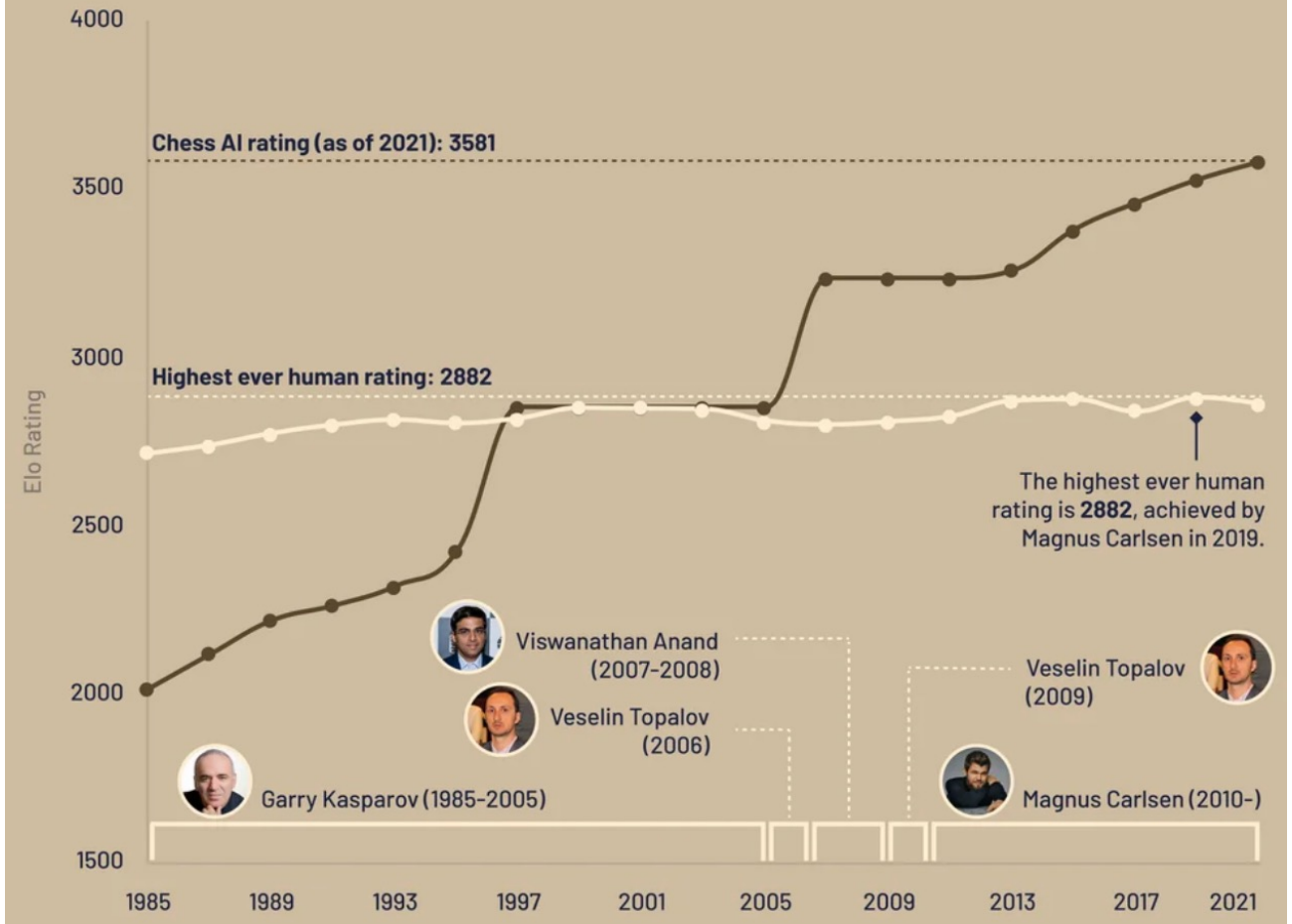
Source: Bureau of Labor Statistics Original design and concept by Mark J. Perry, Senior Fellow Emeritus, AEI

Chess

Created by genuine impact

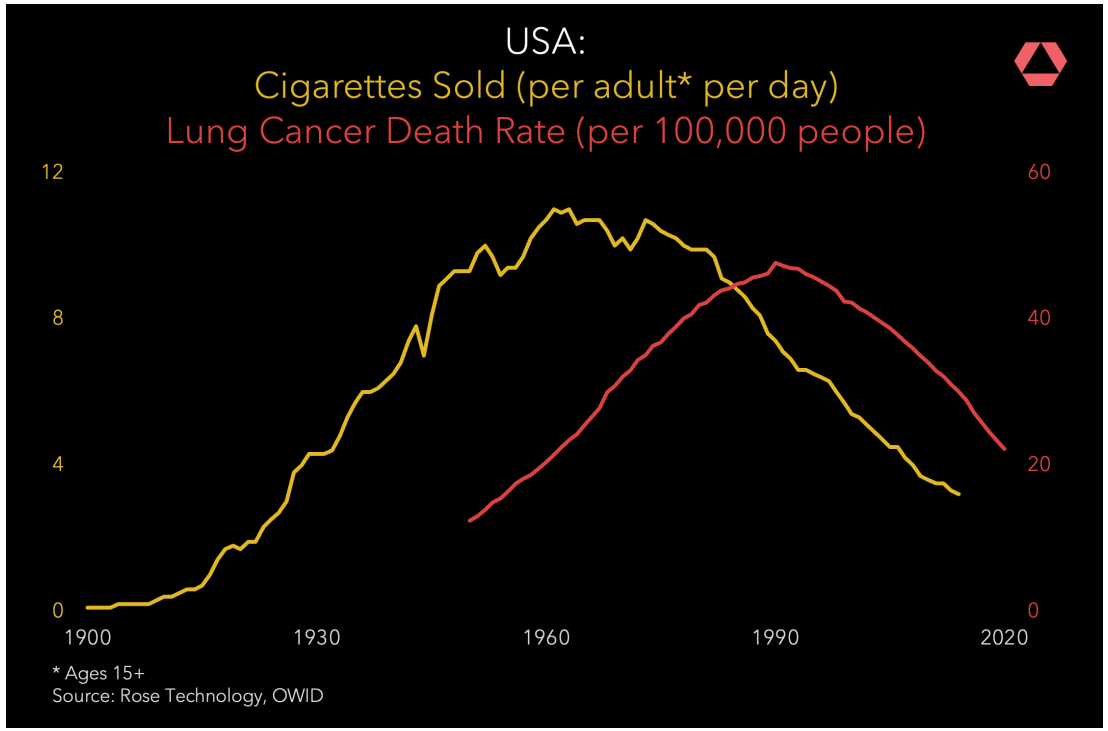
Artificial Intelligence vs Human

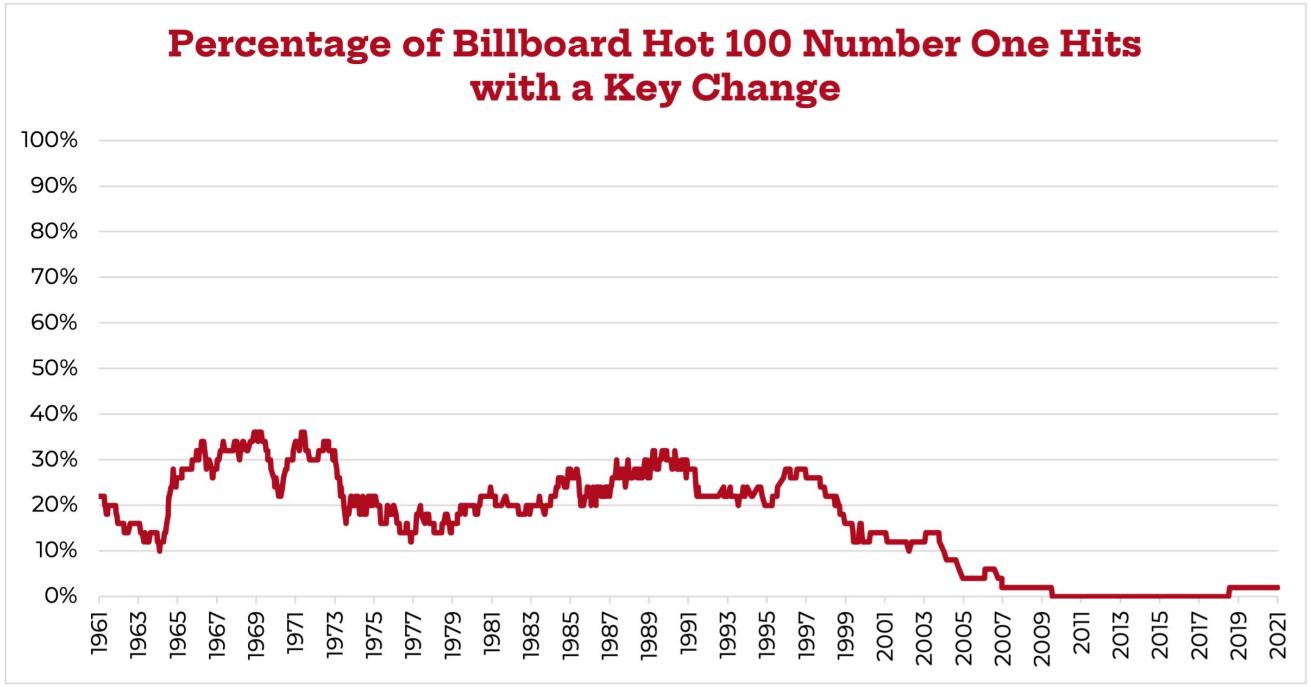
The Elo Rating system is used to measure chess ability, and is based on game results. A higher rating = greater ability.



Source: Our World in Data, FIDE

More charts: genuineimpact.substack.com


USA:
Cigarettes Sold (per adult* per day)
Lung Cancer Death Rate (per 100,000 people)
12
8
4
0
60
40
20
0
1900
1930
1960
1990
2020
* Ages 15+
Source: Rose Technology, OWID


Percentage of Billboard Hot 100 Number One Hits with a Key Change
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1961
1963
1965
1967
1969
1971
1973
1975
1977
1979
1981
1983
1985
1987
1989
1991
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013
2015
2017
2019
2021

Area comparison

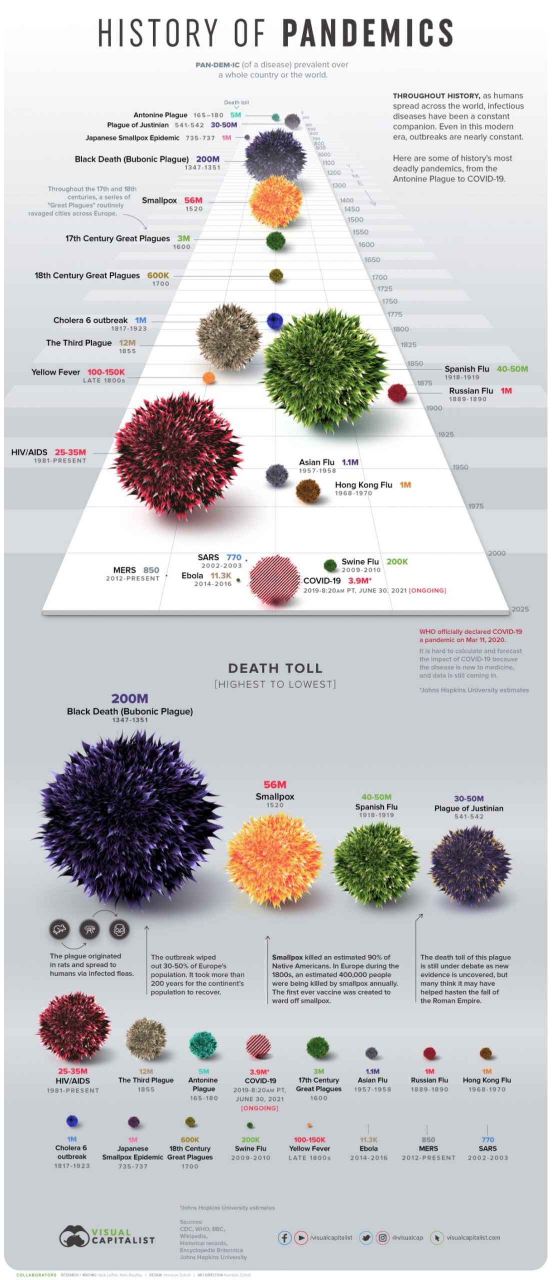
The Top 45 **Richest** celebrities in media/arts

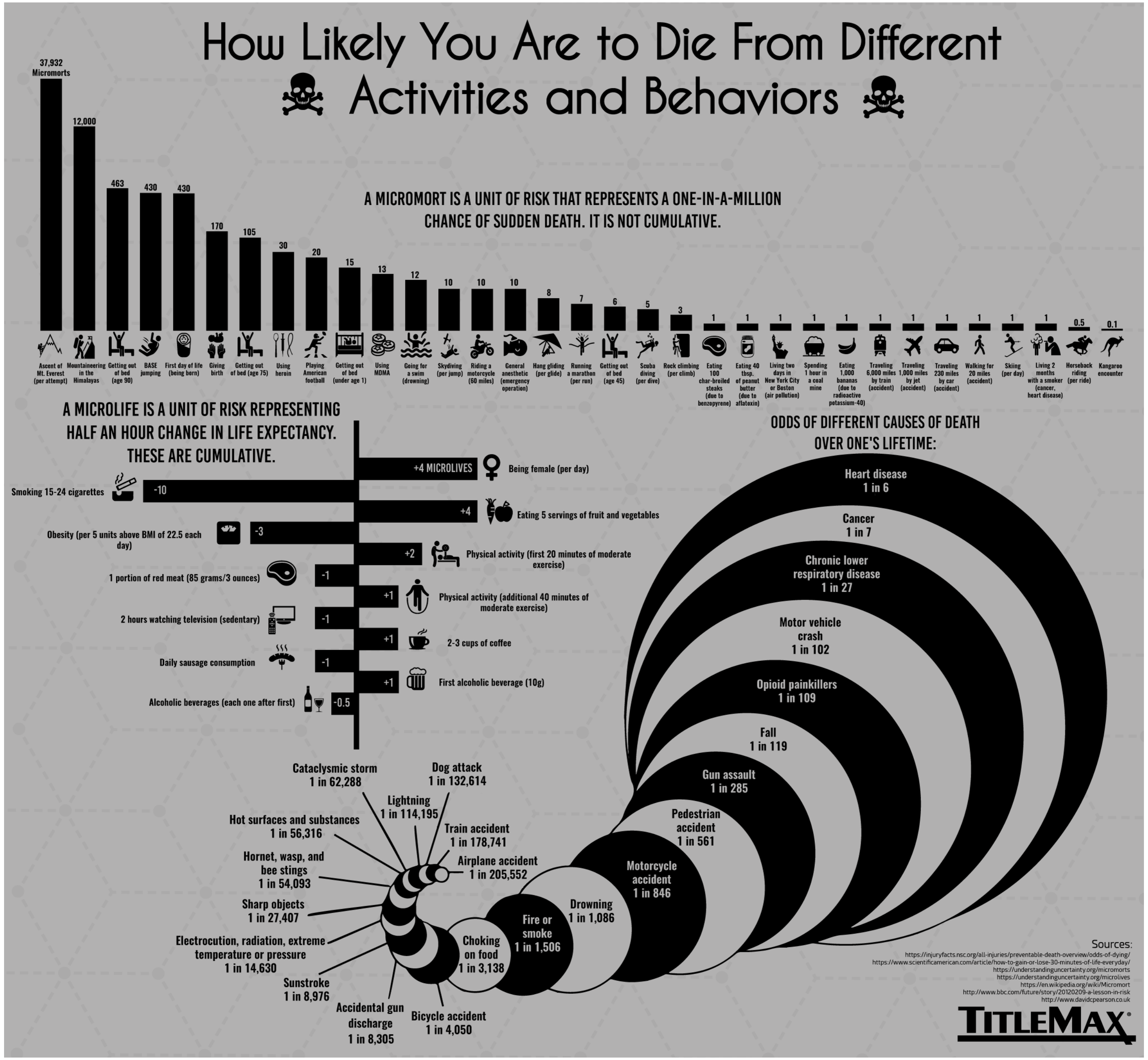
Not including businessmen/women, inflation adjusted for deceased celebrities

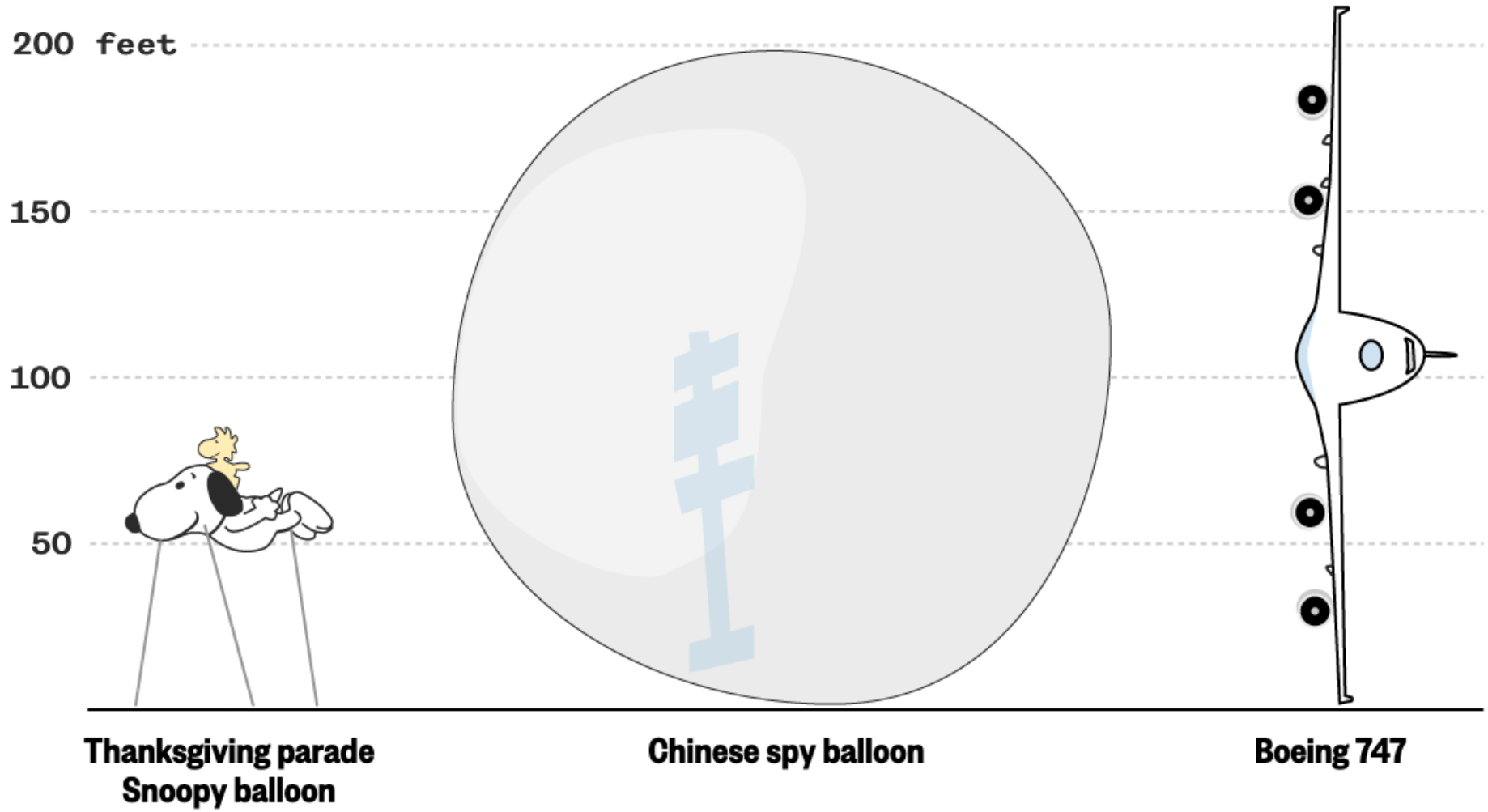
Source: Celebritynetworth.com

Created by genuine impact

More charts: genuineimpact.substack.com


HISTORY OF PANDEMICS
PAN·DEM·IC (of a disease) prevalent over a whole country or the world.
THROUGHOUT HISTORY, as humans spread across the world, infectious diseases have been a constant companion. Even in this modern era, outbreaks are nearly constant.
Here are some of history's most deadly pandemics, from the Antonine Plague to COVID-19.
Death toll
Antonine Plague 165–180 5M
Plague of Justinian 541-542 30-50M
Japanese Smallpox Epidemic 735-737 1M
Black Death (Bubonic Plague) 200M 1347-1351
Throughout the 17th and 18th centuries, a series of "Great Plagues" routinely ravaged cities across Europe.
Smallpox 56M 1520
17th Century Great Plagues 3M 1600
18th Century Great Plagues 600K 1700
Cholera 6 outbreak 1M 1817-1923
The Third Plague 12M 1855
Yellow Fever 100-150K LATE 1800s
Spanish Flu 40-50M 1918-1919
Russian Flu 1M 1889-1890
HIV/AIDS 25-35M 1981-PRESENT
Asian Flu 1.1M 1957-1958
Hong Kong Flu 1M 1968-1970
SARS 770 2002-2003
MERS 850 2012-PRESENT
Ebola 11.3K 2014-2016
Swine Flu 200K 2009-2010
COVID-19 3.9M* 2019-8:20AM PT, JUNE 30, 2021 [ONGOING]
TIME
0
200
400
500
600
700
800
900
1000
1100
1200
1300
1400
1450
1500
1550
1600
1650
1700
1725
1750
1775
1800
1825
1850
1875
1900
1925
1950
1975
2000
2025
WHO officially declared COVID-19 a pandemic on Mar 11, 2020.
It is hard to calculate and forecast the impact of COVID-19 because the disease is new to medicine, and data is still coming in.
*Johns Hopkins University estimates
DEATH TOLL
[HIGHEST TO LOWEST]
200M Black Death (Bubonic Plague) 1347-1351
56M Smallpox 1520
40-50M Spanish Flu 1918-1919
30-50M Plague of Justinian 541-542
The plague originated in rats and spread to humans via infected fleas.
The outbreak wiped out 30-50% of Europe's population. It took more than 200 years for the continent's population to recover.
Smallpox killed an estimated 90% of Native Americans. In Europe during the 1800s, an estimated 400,000 people were being killed by smallpox annually. The first ever vaccine was created to ward off smallpox.
The death toll of this plague is still under debate as new evidence is uncovered, but many think it may have helped hasten the fall of the Roman Empire.
25-35M HIV/AIDS 1981-PRESENT
12M The Third Plague 1855
5M Antonine Plague 165-180
3.9M* COVID-19 2019-8:20AM PT, JUNE 30, 2021 [ONGOING]
3M 17th Century Great Plagues 1600
1.1M Asian Flu 1957-1958
1M Russian Flu 1889-1890
1M Hong Kong Flu 1968-1970
1M Cholera 6 outbreak 1817-1923
1M Japanese Smallpox Epidemic 735-737
600K 18th Century Great Plagues 1700
200K Swine Flu 2009-2010
100-150K Yellow Fever LATE 1800s
11.3K Ebola 2014-2016
850 MERS 2012-PRESENT
770 SARS 2002-2003
*Johns Hopkins University estimates
Sources: CDC, WHO, BBC, Wikipedia, Historical records, Encyclopedia Britannica, Johns Hopkins University
VISUAL CAPITALIST
/visualcapitalist
@visualcap
visualcapitalist.com

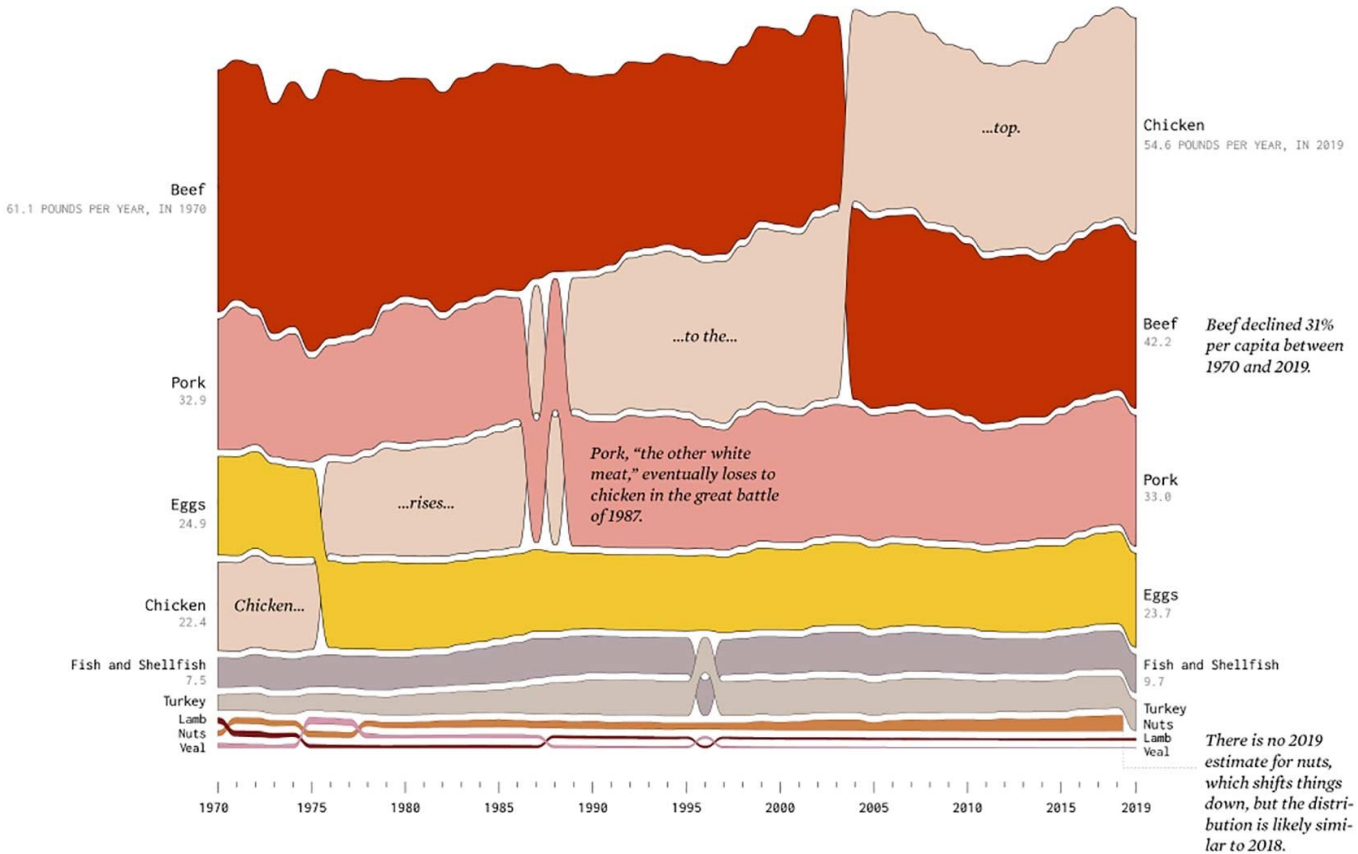

How Likely You Are to Die From Different Activities and Behaviors
A MICROMORT IS A UNIT OF RISK THAT REPRESENTS A ONE-IN-A-MILLION CHANCE OF SUDDEN DEATH. IT IS NOT CUMULATIVE.
37,932 Micromorts
Ascent of Mt. Everest (per attempt) 37,932
Mountaineering in the Himalayas 12,000
Getting out of bed (age 90) 463
BASE jumping 430
First day of life (being born) 430
Giving birth 170
Getting out of bed (age 75) 105
Using heroin 30
Playing American football 20
Getting out of bed (under age 1) 15
Using MDMA 13
Going for a swim (drowning) 12
Skydiving (per jump) 10
Riding a motorcycle (60 miles) 10
General anesthetic (emergency operation) 10
Hang gliding (per glide) 8
Running a marathon (per run) 7
Getting out of bed (age 45) 6
Scuba diving (per dive) 5
Rock climbing (per climb) 3
Eating 100 char-broiled steaks (due to benzopyrene) 1
Eating 40 tbsp. of peanut butter (due to aflatoxin) 1
Living two days in New York City or Boston (air pollution) 1
Spending 1 hour in a coal mine 1
Eating 1,000 bananas (due to radioactive potassium-40) 1
Traveling 6,000 miles by train (accident) 1
Traveling 1,000 miles by jet (accident) 1
Traveling 230 miles by car (accident) 1
Walking for 20 miles (accident) 1
Skiing (per day) 1
Living 2 months with a smoker (cancer, heart disease) 1
Horseback riding (per ride) 0.5
Kangaroo encounter 0.1
A MICROLIFE IS A UNIT OF RISK REPRESENTING HALF AN HOUR CHANGE IN LIFE EXPECTANCY. THESE ARE CUMULATIVE.
+4 MICROLIVES Being female (per day)
Smoking 15-24 cigarettes -10
+4 Eating 5 servings of fruit and vegetables
Obesity (per 5 units above BMI of 22.5 each day) -3
+2 Physical activity (first 20 minutes of moderate exercise)
1 portion of red meat (85 grams/3 ounces) -1
+1 Physical activity (additional 40 minutes of moderate exercise)
2 hours watching television (sedentary) -1
+1 2-3 cups of coffee
Daily sausage consumption -1
+1 First alcoholic beverage (10g)
Alcoholic beverages (each one after first) -0.5
ODDS OF DIFFERENT CAUSES OF DEATH OVER ONE'S LIFETIME:
Heart disease 1 in 6
Cancer 1 in 7
Chronic lower respiratory disease 1 in 27
Motor vehicle crash 1 in 102
Opioid painkillers 1 in 109
Fall 1 in 119
Gun assault 1 in 285
Pedestrian accident 1 in 561
Motorcycle accident 1 in 846
Drowning 1 in 1,086
Fire or smoke 1 in 1,506
Choking on food 1 in 3,138
Bicycle accident 1 in 4,050
Accidental gun discharge 1 in 8,305
Sunstroke 1 in 8,976
Electrocution, radiation, extreme temperature or pressure 1 in 14,630
Sharp objects 1 in 27,407
Hornet, wasp, and bee stings 1 in 54,093
Hot surfaces and substances 1 in 56,316
Cataclysmic storm 1 in 62,288
Lightning 1 in 114,195
Dog attack 1 in 132,614
Train accident 1 in 178,741
Airplane accident 1 in 205,552
Sources:
https://injuryfacts.nsc.org/all-injuries/preventable-death-overview/odds-of-dying/
https://www.scientificamerican.com/article/how-to-gain-or-lose-30-minutes-of-life-everyday/
https://understandinguncertainty.org/micromorts
https://understandinguncertainty.org/microlives
https://en.wikipedia.org/wiki/Micromort
http://www.bbc.com/future/story/20120209-a-lesson-in-risk
http://www.davidcpearson.co.uk
TITLEMAX®

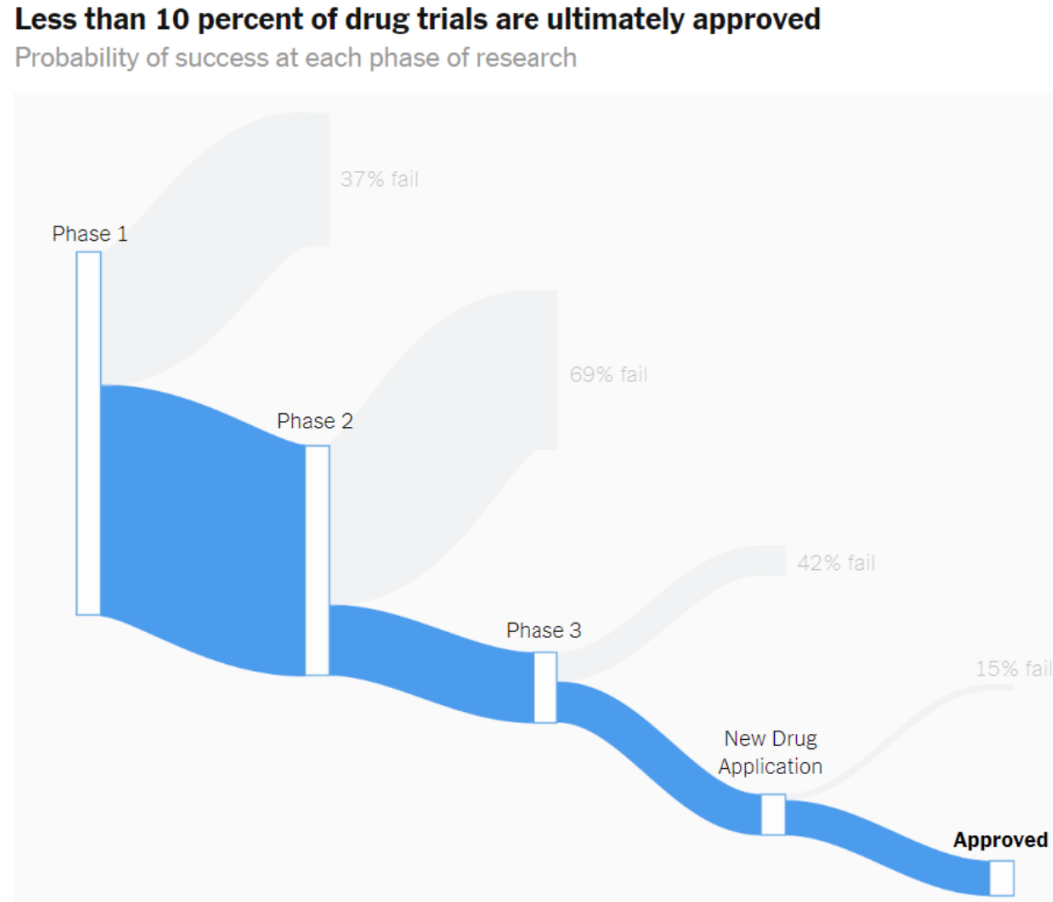


Source: NORAD, Boeing, NBC News

Graphic: JoElla Carman / NBC News

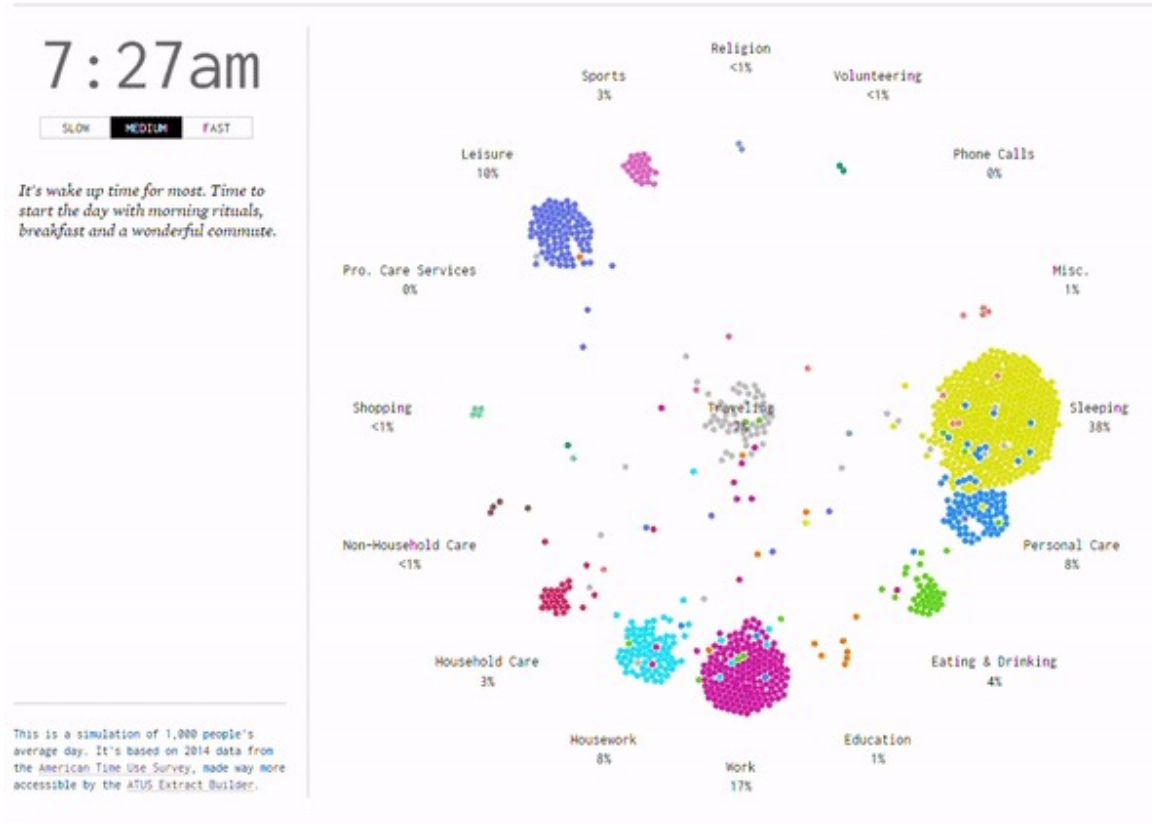
Stacked bargraphs/Sankey diagrams


Beef
61.1 POUNDS PER YEAR, IN 1970
Pork
32.9
Eggs
24.9
Chicken
22.4
Fish and Shellfish
7.5
Turkey
Lamb
Nuts
Veal
...rises...
Chicken...
...to the...
...top.
Pork, "the other white meat," eventually loses to chicken in the great battle of 1987.
Chicken
54.6 POUNDS PER YEAR, IN 2019
Beef
42.2
Pork
33.0
Eggs
23.7
Fish and Shellfish
9.7
Turkey
Nuts
Lamb
Veal
Beef declined 31% per capita between 1970 and 2019.
There is no 2019 estimate for nuts, which shifts things down, but the distribution is likely similar to 2018.
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2019

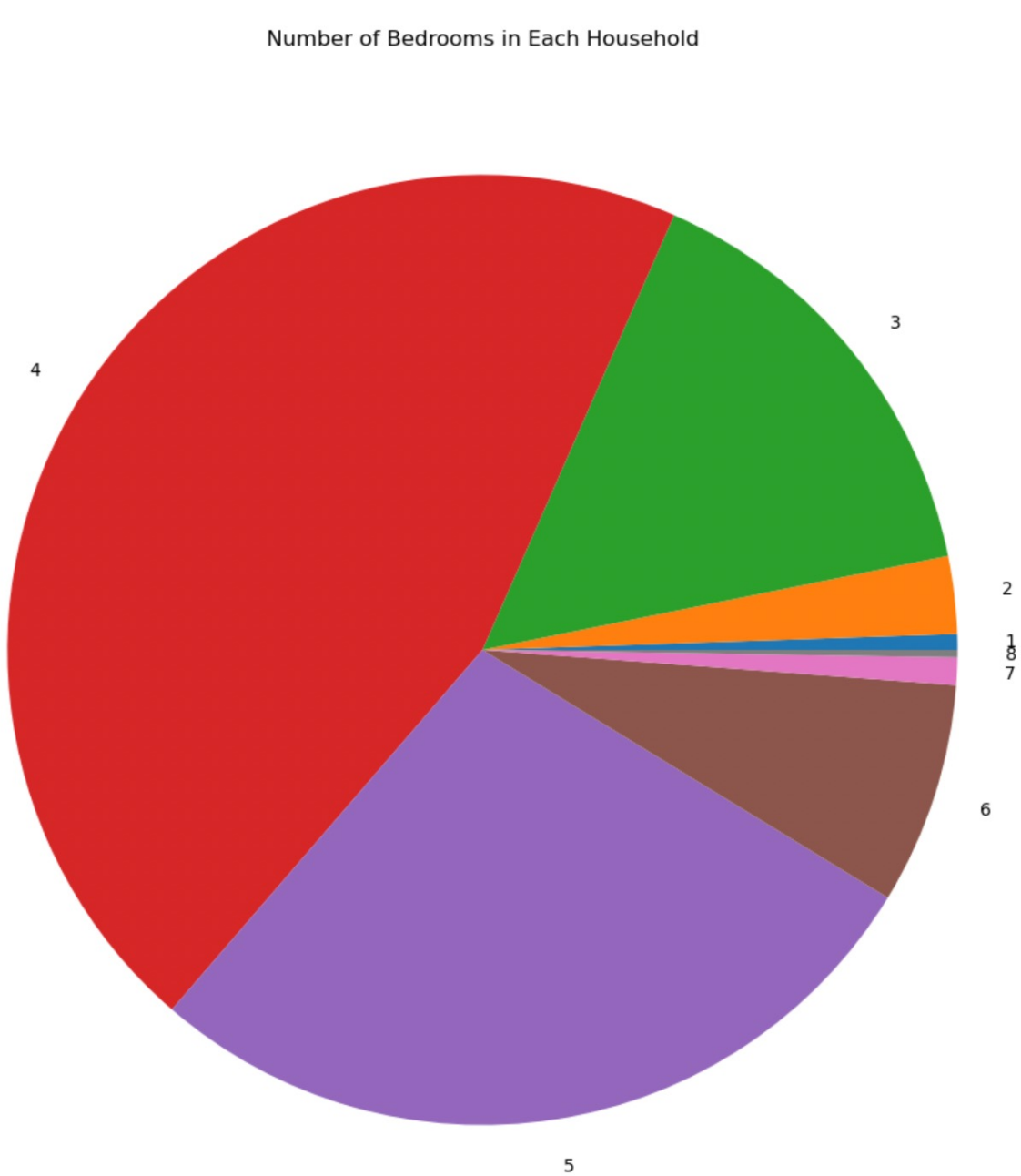

Less than 10 percent of drug trials are ultimately approved
Probability of success at each phase of research
37% fail
Phase 1
69% fail
Phase 2
42% fail
Phase 3
15% fail
New Drug Application
Approved

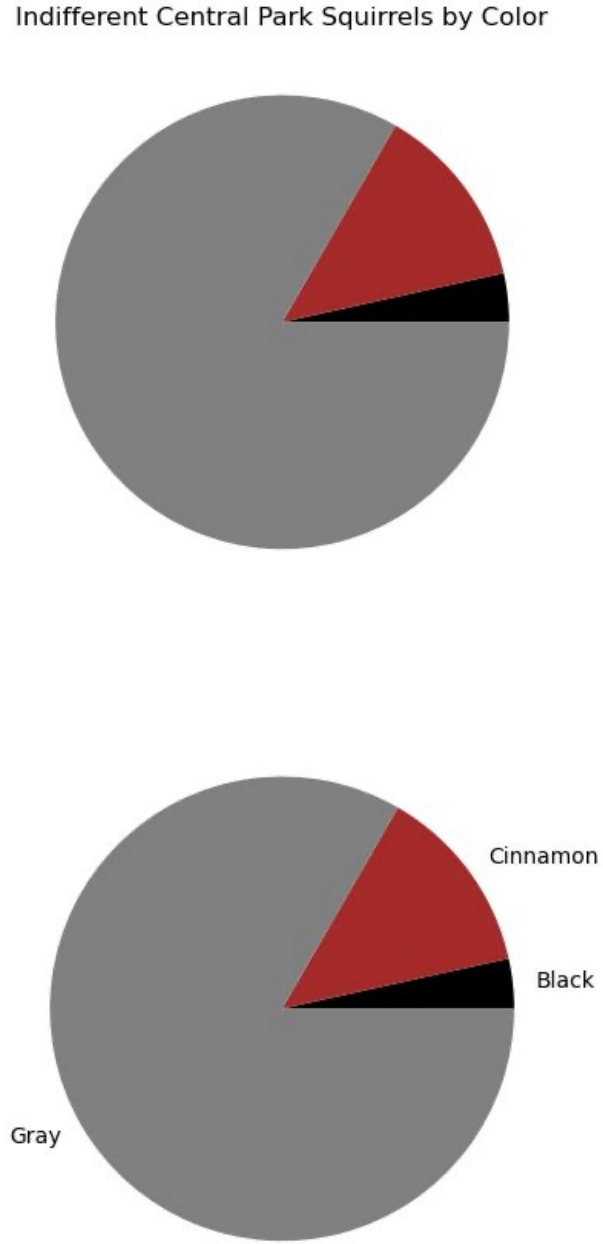
Ray tracing and animation

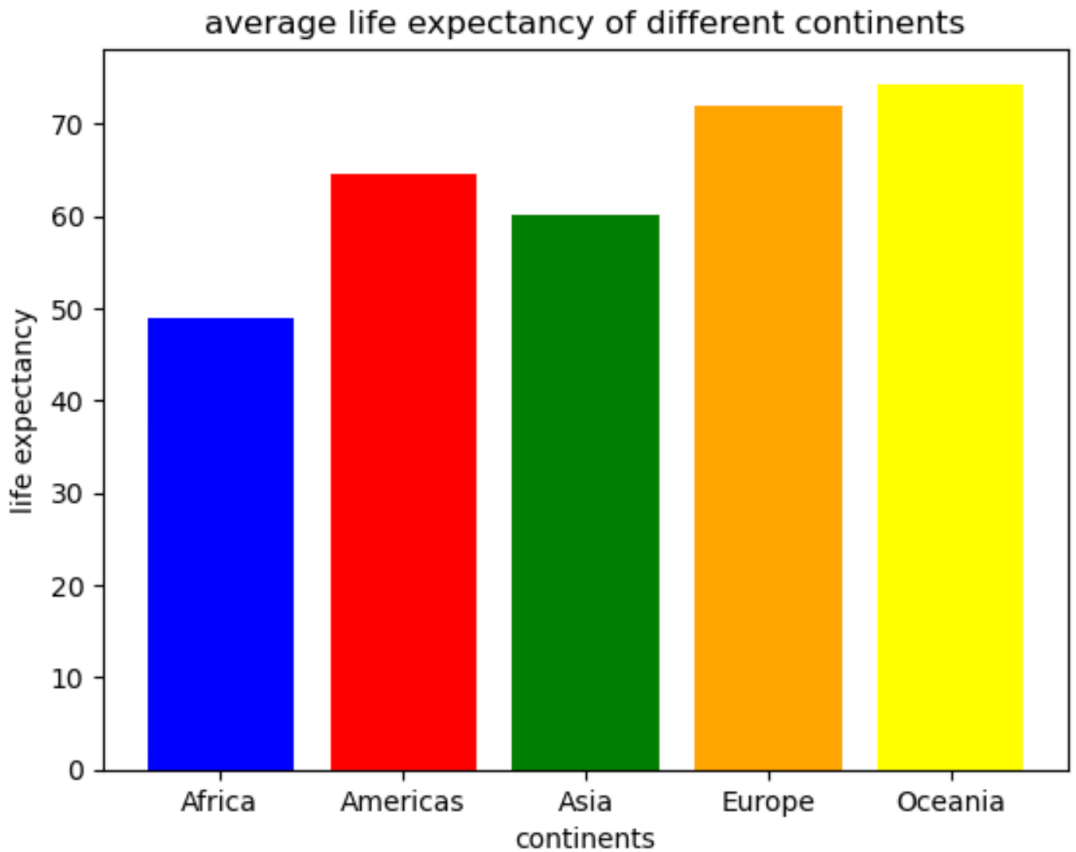

Population Density in Europe
1km cells / the big squares are 50km x 50km
Data: GHS_POP_EU
Software: Aerialod
Author: Alasdair Rae

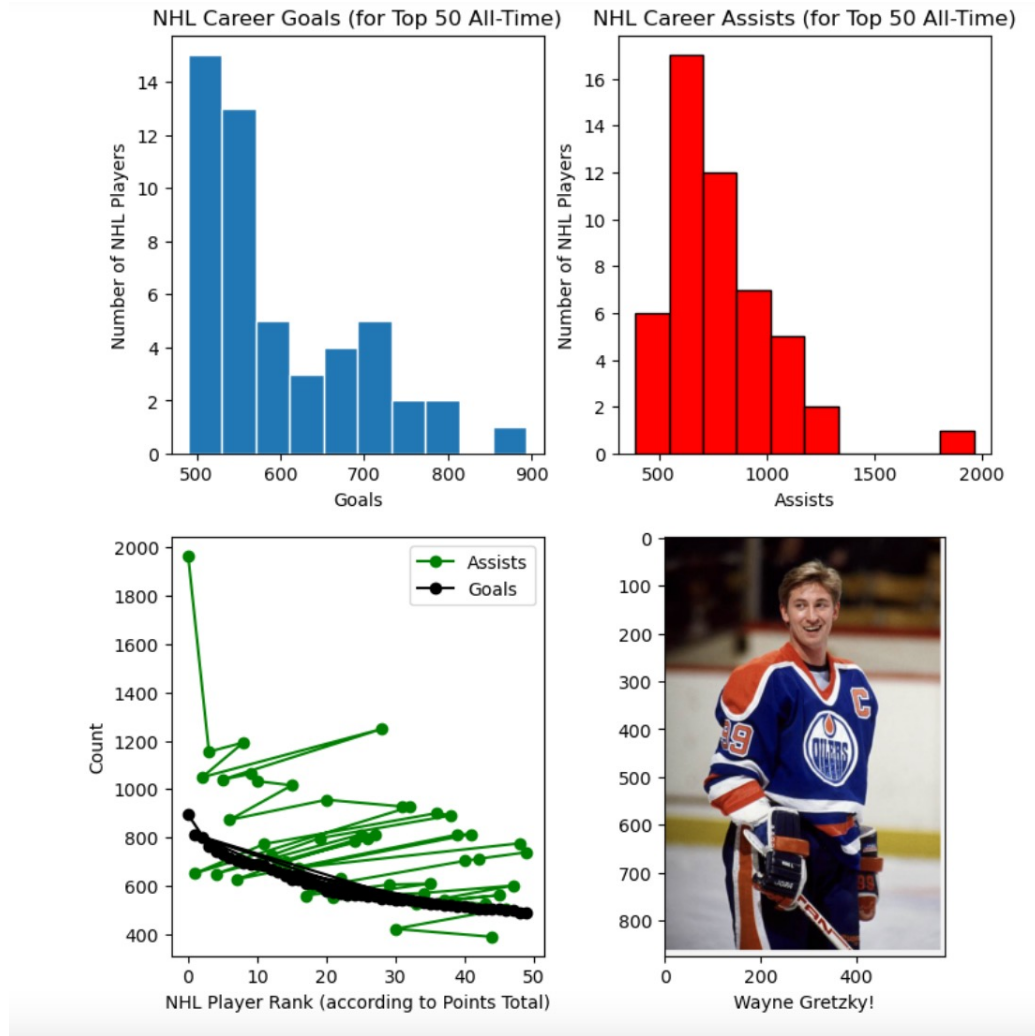

7:27am
SLOW
MEDIUM
FAST
It's wake up time for most. Time to start the day with morning rituals, breakfast and a wonderful commute.
This is a simulation of 1,000 people's average day. It's based on 2014 data from the American Time Use Survey, made way more accessible by the ATUS Extract Builder.
Religion
<1%
Sports
3%
Volunteering
<1%
Leisure
10%
Phone Calls
0%
Pro. Care Services
0%
Misc.
1%
Shopping
<1%
Sleeping
38%
Non-Household Care
<1%
Personal Care
8%
Household Care
3%
Eating & Drinking
4%
Housework
8%
Education
1%
Work
17%

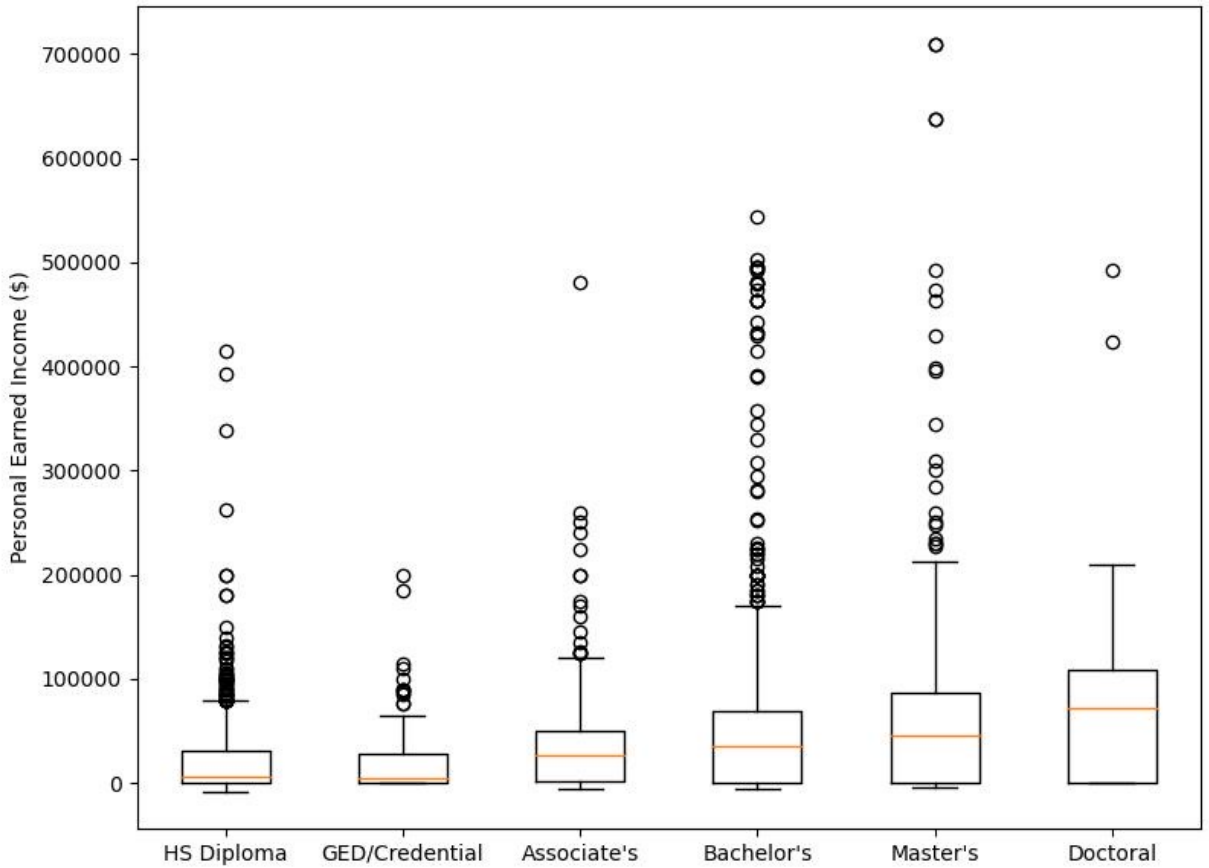
Your visualizations

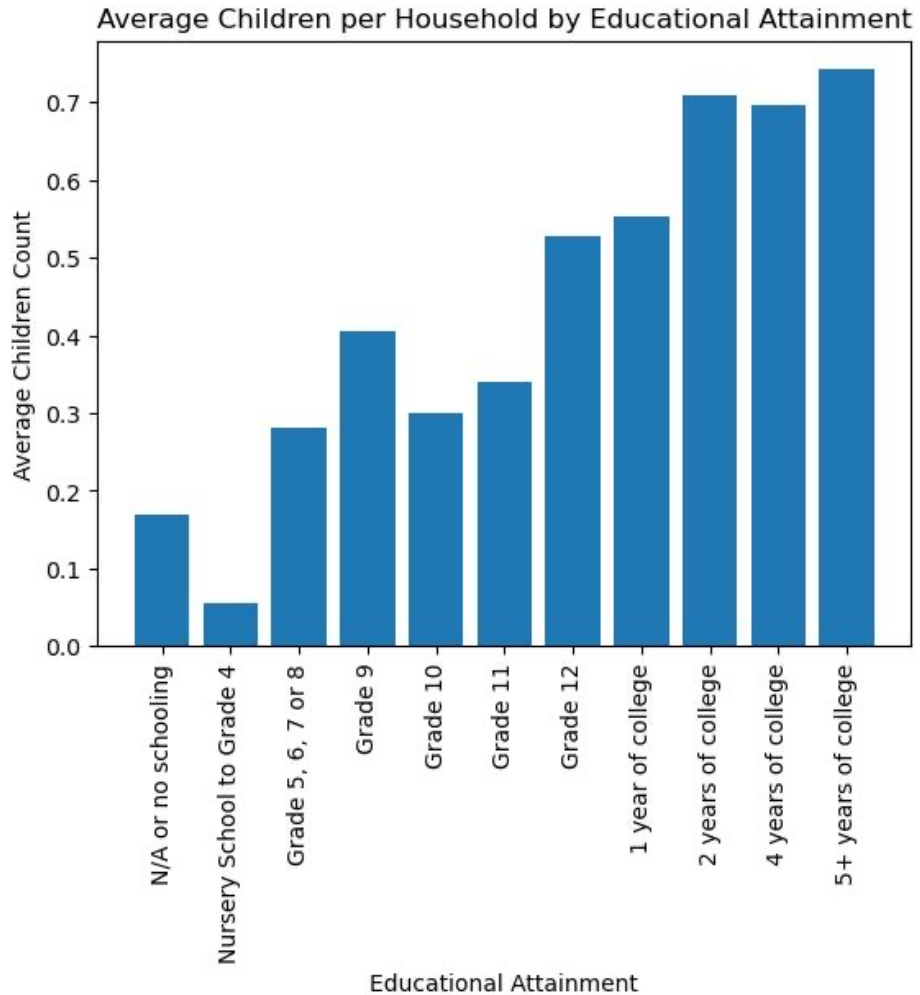

Number of Bedrooms in Each Household
3
4
2
1
8
7
6
5

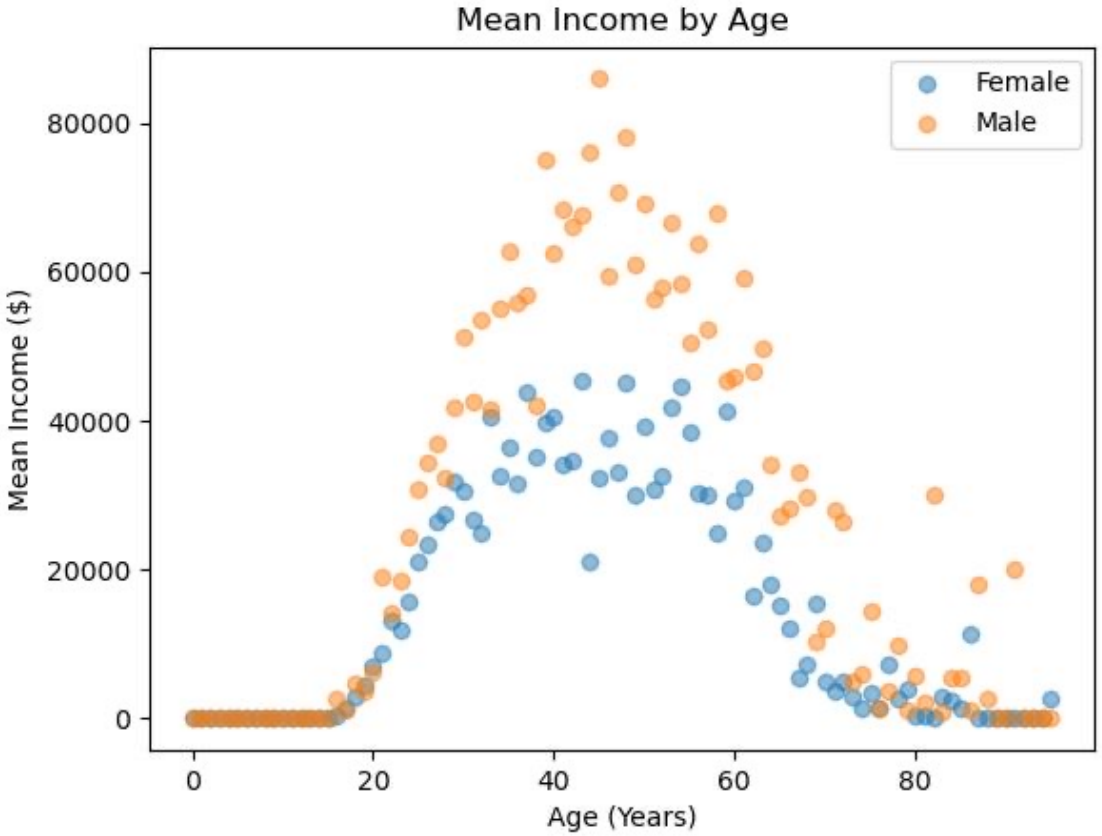

Indifferent Central Park Squirrels by Color
Cinnamon
Black
Gray

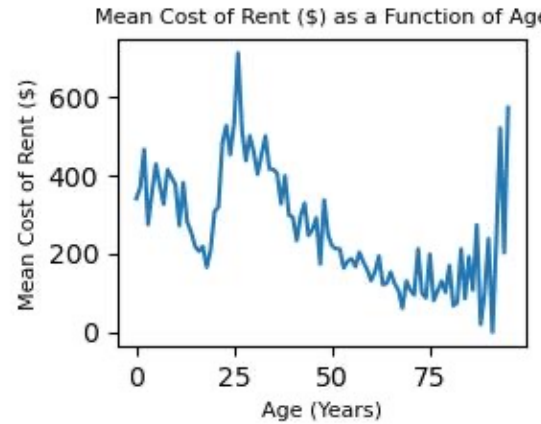

average life expectancy of different continents
life expectancy
0
10
20
30
40
50
60
70
Africa
Americas
Asia
Europe
Oceania
continents

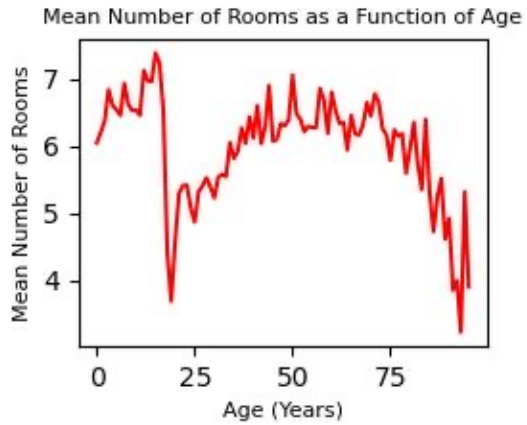

NHL Career Goals (for Top 50 All-Time)
Number of NHL Players
Goals
NHL Career Assists (for Top 50 All-Time)
Number of NHL Players
Assists
Assists
Goals
Count
NHL Player Rank (according to Points Total)
Wayne Gretzky!

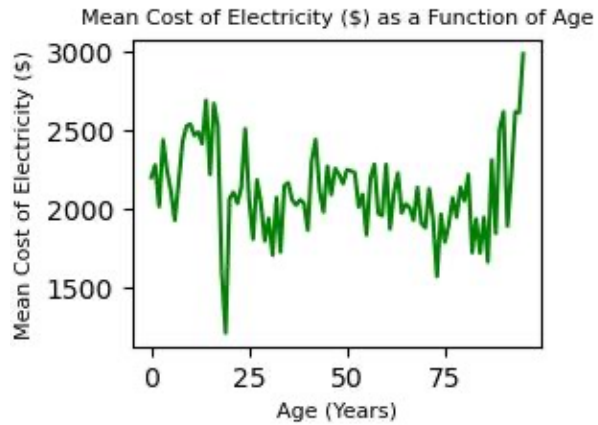

Personal Earned Income ($)
700000
600000
500000
400000
300000
200000
100000
0
HS Diploma
GED/Credential
Associate's
Bachelor's
Master's
Doctoral

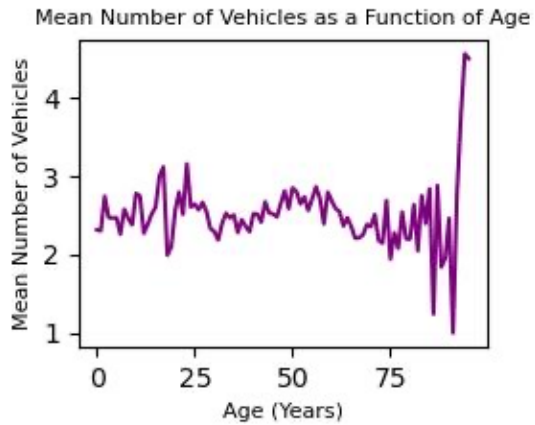

Average Children per Household by Educational Attainment
Average Children Count
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
N/A or no schooling
Nursery School to Grade 4
Grade 5, 6, 7 or 8
Grade 9
Grade 10
Grade 11
Grade 12
1 year of college
2 years of college
4 years of college
5+ years of college
Educational Attainment

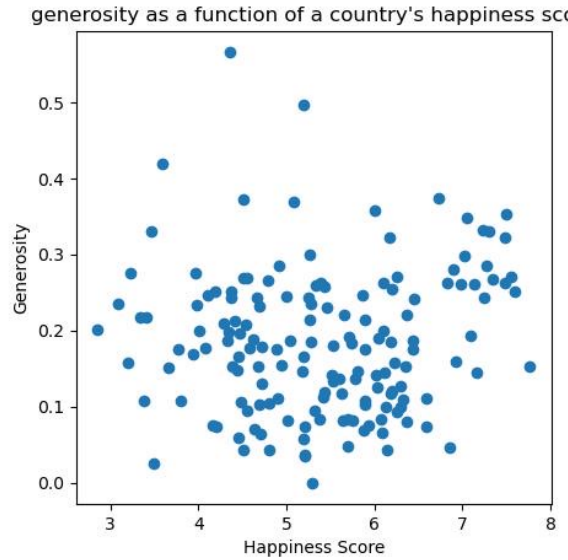

Mean Income by Age
Female
Male
Mean Income ($)
Age (Years)
0
20000
40000
60000
80000
0
20
40
60
80

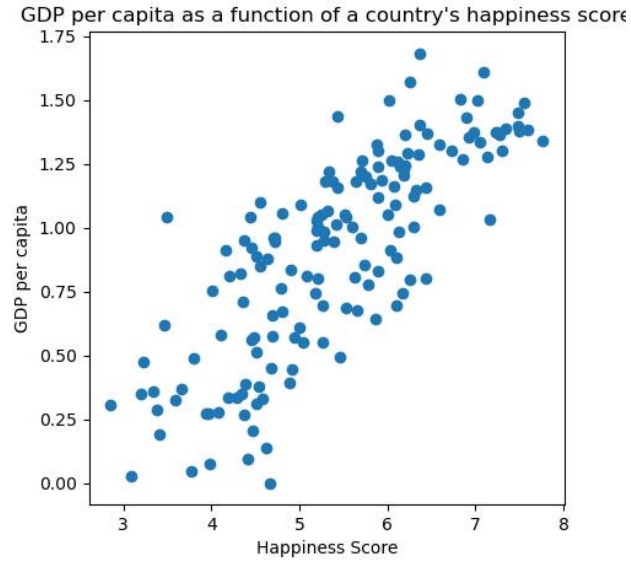

Mean Cost of Rent ($) as a Function of Age
Mean Cost of Rent ($)
Age (Years)
0
200
400
600
0
25
50
75

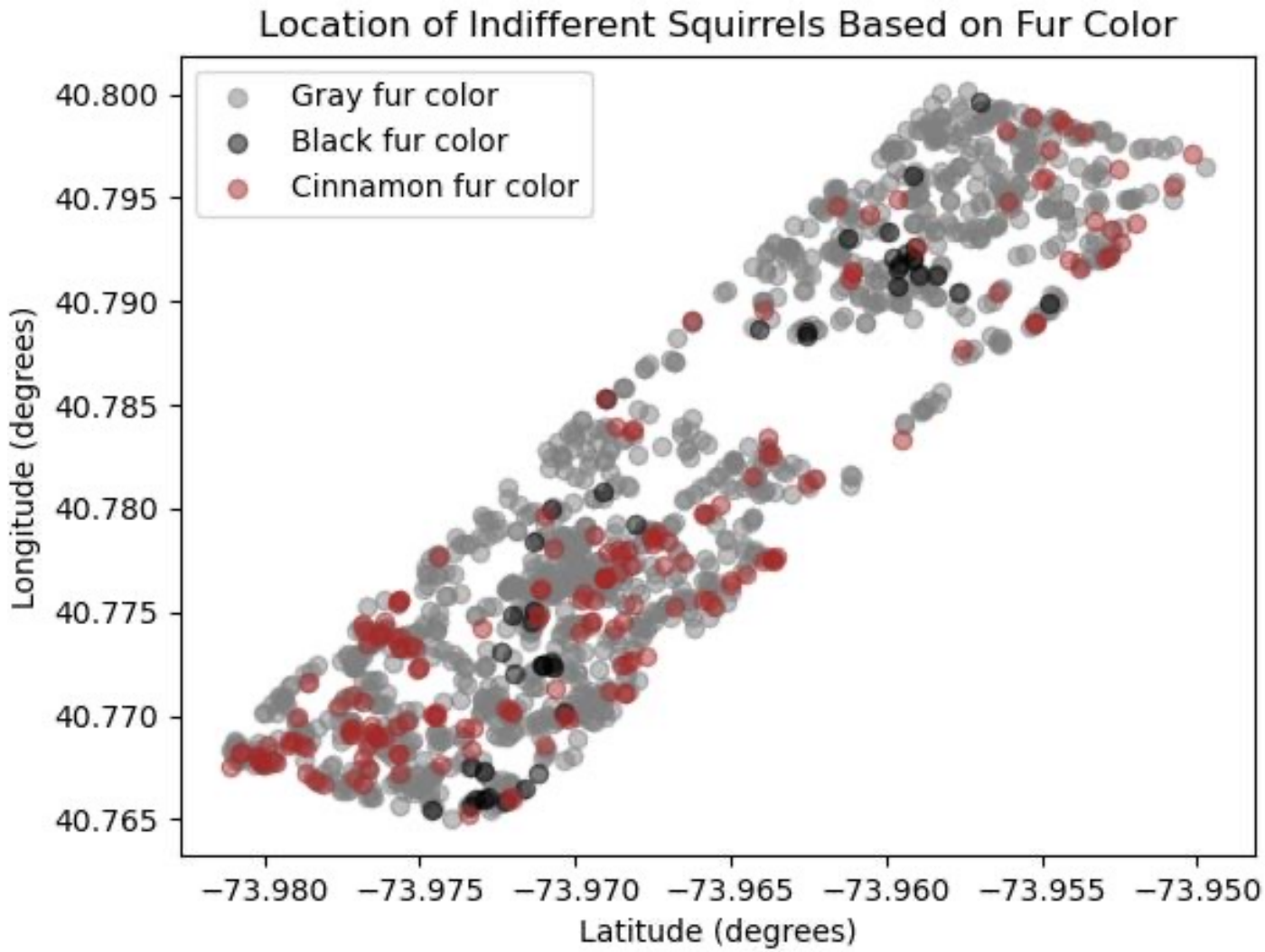

Mean Number of Rooms as a Function of Age
Mean Number of Rooms
Age (Years)
4
5
6
7
0
25
50
75


Mean Cost of Electricity ($) as a Function of Age
Mean Cost of Electricity ($)
Age (Years)
1500
2000
2500
3000
0
25
50
75


Mean Number of Vehicles as a Function of Age
Mean Number of Vehicles
Age (Years)
1
2
3
4
0
25
50
75


generosity as a function of a country's happiness score
Generosity
Happiness Score
0.0
0.1
0.2
0.3
0.4
0.5
3
4
5
6
7
8


GDP per capita as a function of a country's happiness score
GDP per capita
Happiness Score
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
3
4
5
6
7
8


Location of Indifferent Squirrels Based on Fur Color
Gray fur color
Black fur color
Cinnamon fur color
Longitude (degrees)
Latitude (degrees)
40.800
40.795
40.790
40.785
40.780
40.775
40.770
40.765
−73.980
−73.975
−73.970
−73.965
−73.960
−73.955
−73.950

Review: visualizing data with matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations.

- import matplotlib.pyplot as plt

Types of plots we have created

- plt.plot(x, y, '-o') # line plot/scatter plot
- plt.hist(data)
- plt.boxplot(data)
- plot.scatter(x, y, s = , color = , marker =)

Review: visualizing data with matplotlib

Make sure always label your axes:

- plt.ylabel("y label")
- plt.xlabel("x label")
- plt.title("my title")
- plt.plot(x, y, label = "blah")
- plt.legend()

We can create subplots:

- plt.subplot(1, 2, 1);
- plt.plot(x1, y1);

Let's do some warm-up exercises Jupyter!

Seaborn

"Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics."

- i.e., it will create better looking plots that are easier to make

There are ways to create visualizations in seaborn:

1. **axes-level** functions that plot on a single axis
2. **figure-level** functions that plot across multiple axes

We will focus on figure level plots

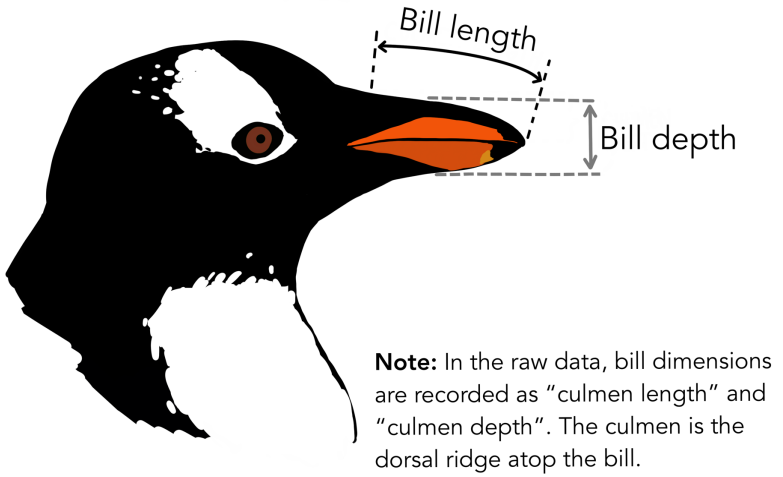
To make plots better looking we can set a theme

```
import seaborn as sns

sns.set_theme()
```

Inspiration: Palmer penguins

To explore seaborn, let's look at some data on penguins!



Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

Seaborn figure level plots

Figure level plots are grouped based on the types of variables being plotted

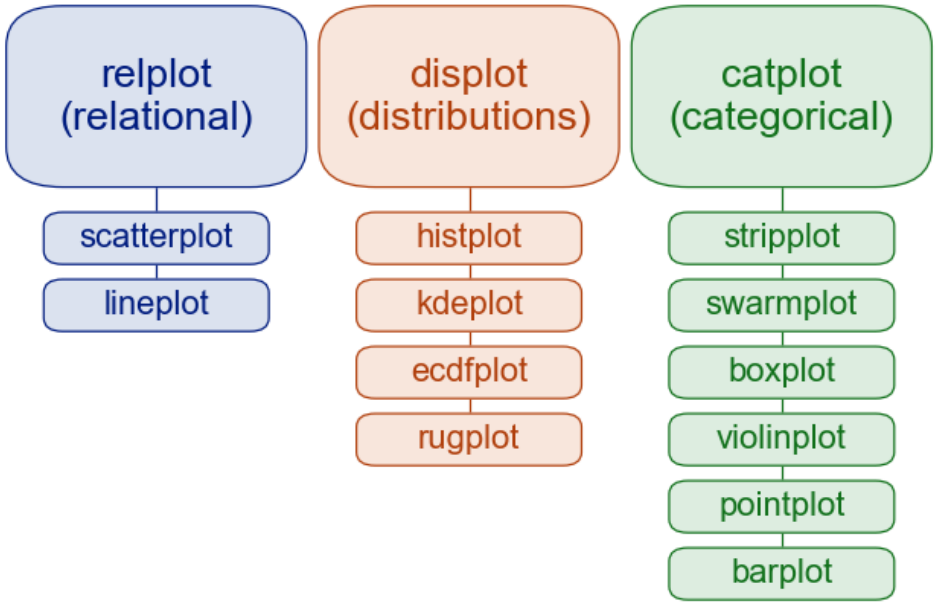
In particular, there are plots for:

1. Two quantitative variables
   - sns.relplot()

2. A single quantitative variable
   - sns.displot()

Quantitative variable compared across different categorical levels
   - sns.catplot()

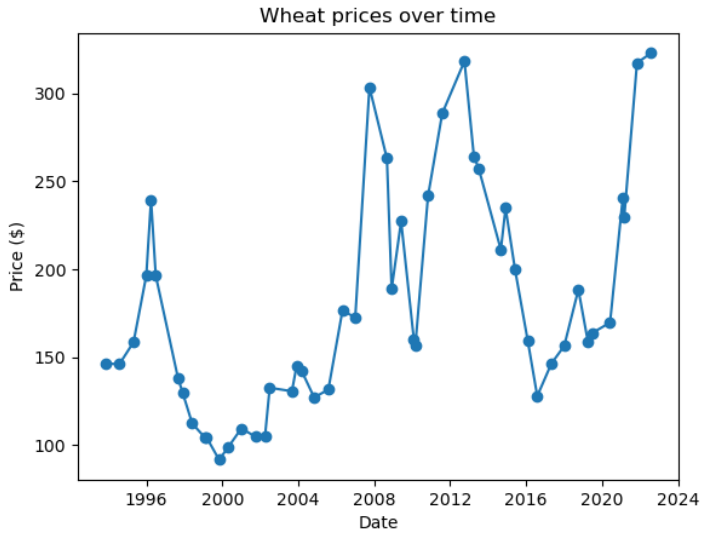
Figure level plots

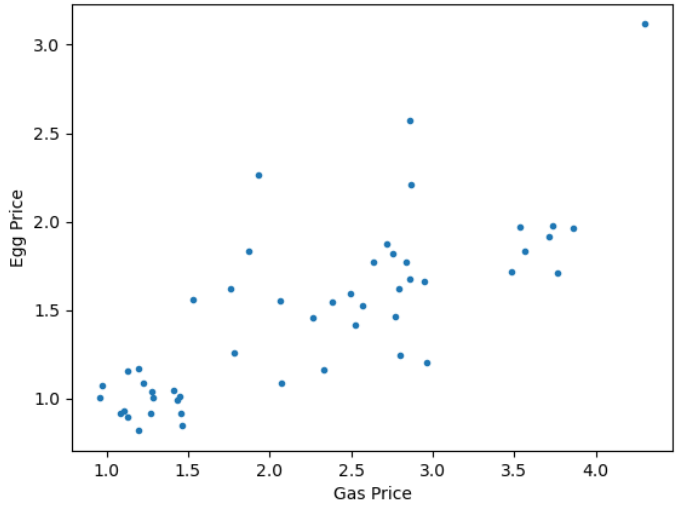


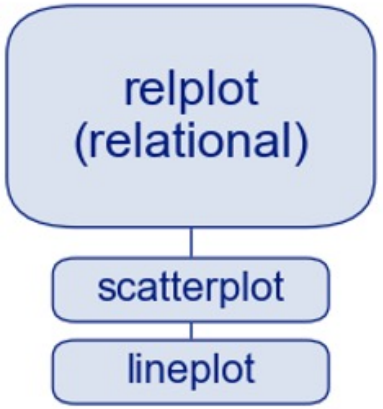
Plots for two quantitative variable

What types of plots have we seen for assessing the relationships between two quantitative variable?

- Line plots and scatter plots!





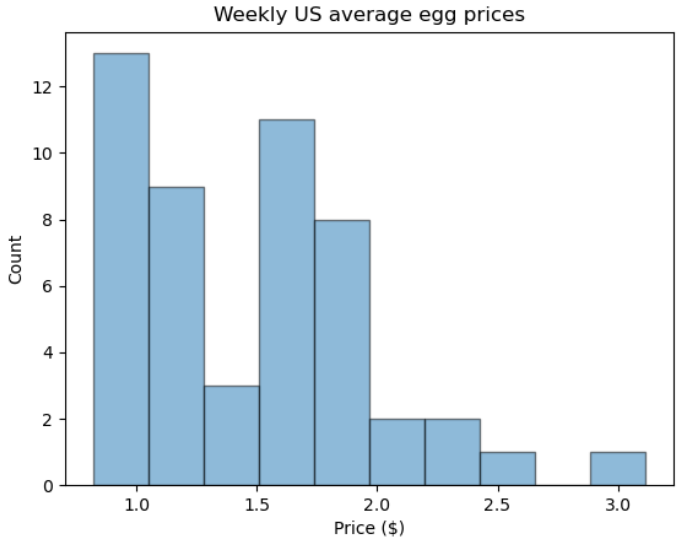


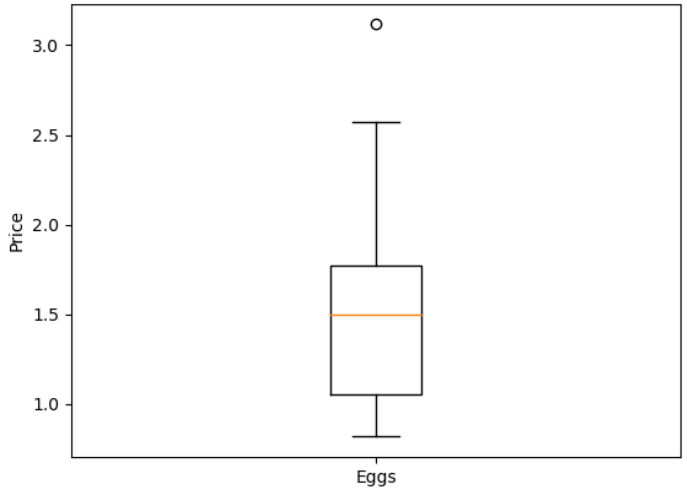
Let's explore this in Jupyter!

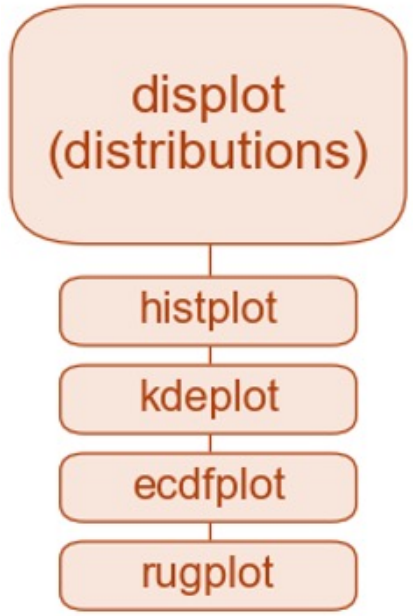
Plots for a single quantitative variable

What types of plots have we seen for plotting a single quantitative variable?

- Histograms and boxplots





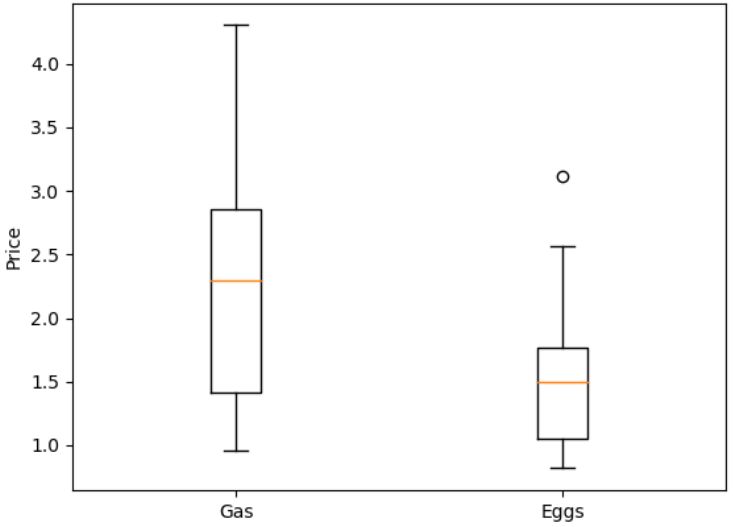


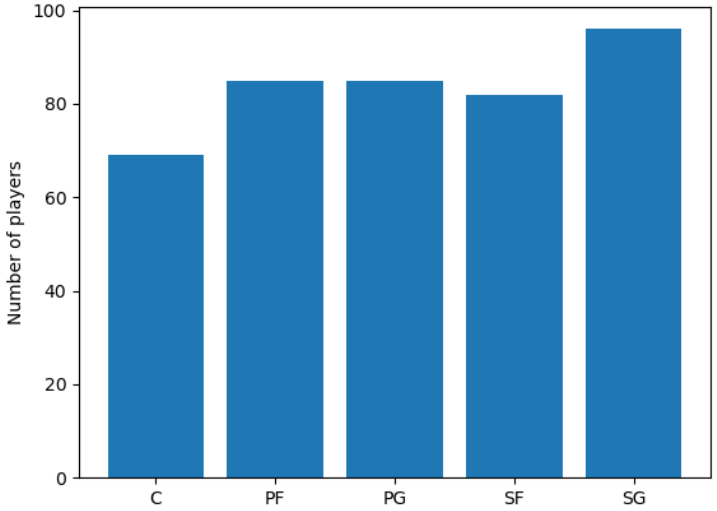
Let's explore this in Jupyter!

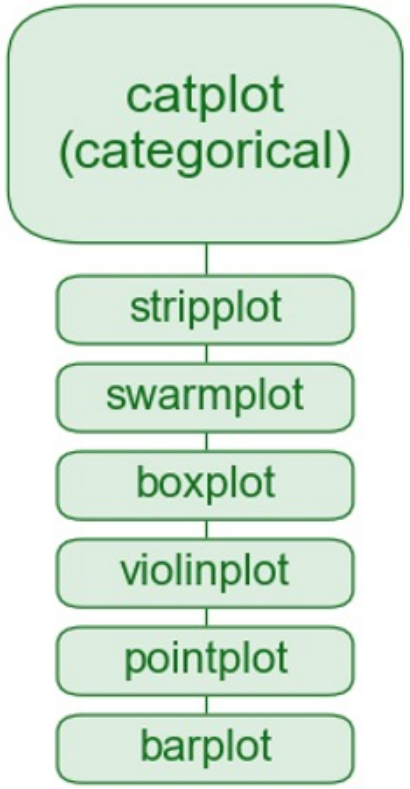
Plots for quantitative data comparing across different categorical levels

What types of plots have we seen comparing quantitative data at different levels of a categorical variable?

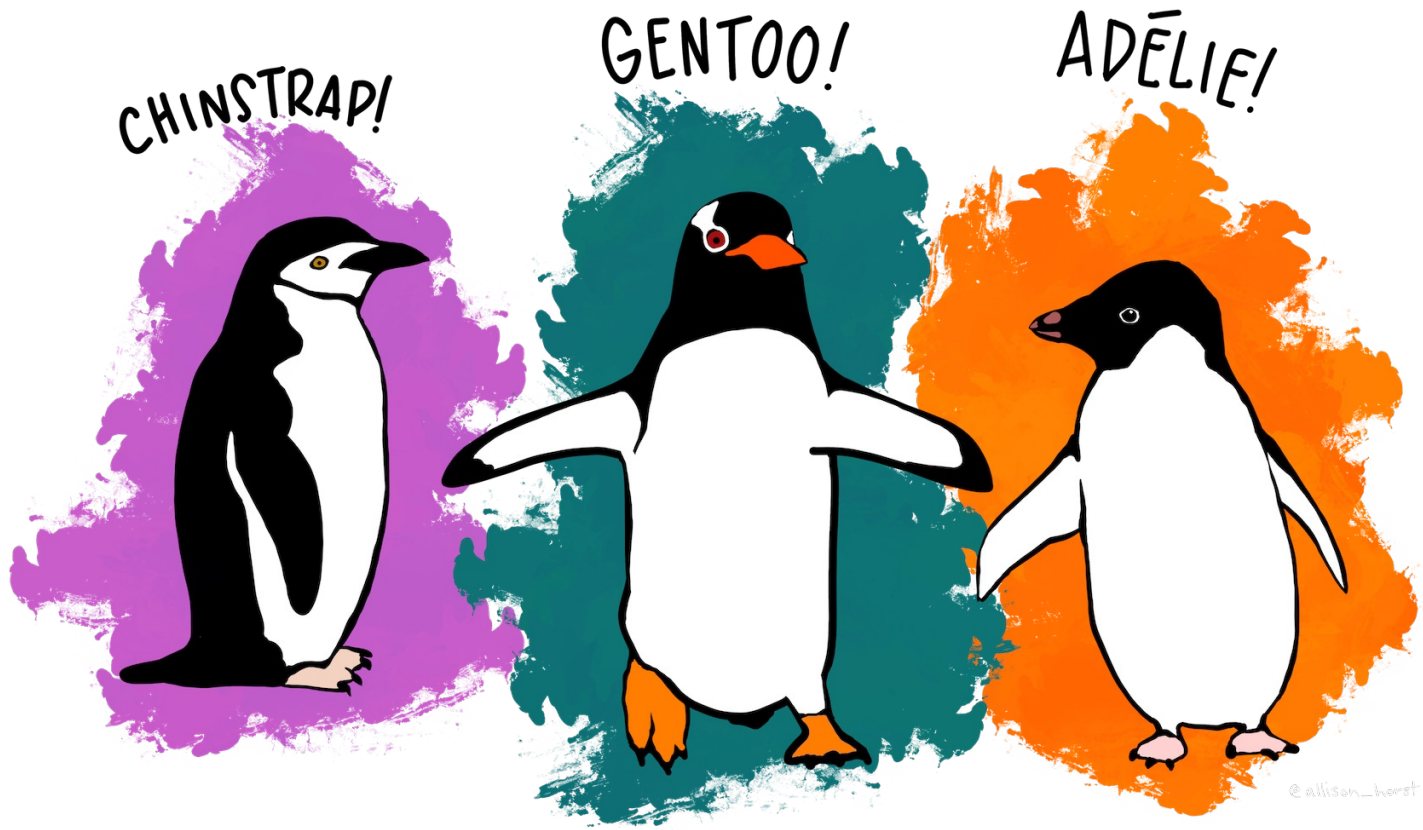
- Side-by-side boxplots, barplots (sort of)







Let's explore this in Jupyter!


CHINSTRAP!
GENTOO!
ADÉLIE!
@allison_horst


text
MaNiPuLaTiOn

Text manipulation

80% of a Data Scientists time is cleaning data

- Text manipulation is a big part of cleaning data

20% of a Data Scientists time is complaining about cleaning data

Python has many string methods that are useful for manipulating text and cleaning data!

Text manipulation: capitalization

Some of the simplest string methods involve changing capitalization.

Changing capitalization can be useful when joining DataFrames

- i.e., if they key values are the same, but the values have different capitalization.
  - For example, joining different countries, but in one DataFrame the country names are capitalized and in the other they are not

Text manipulation: capitalization

Python strings have a number of methods to change the capitalization of words including:

- .capitalize(): Converts the first character to upper case
- .lower(): Converts a string into lower case
- .upper(): Converts a string into upper case
- .title(): Converts the first character of each word to upper case
- .swapcase(): Swaps cases, lower case becomes upper case and vice versa

Let's explore this in Jupyter!

Text manipulation: string padding

Often we want to remove extra spaces (called "white space") from the front or end of a string.

Conversely, sometimes we want to add extra spaces to make a set of strings the same length

- This is known as "string padding"

Python strings have a number of methods that can pad/trim strings including:

- strip(): Returns a trimmed version of the string (i.e., with no leading or trailing white space).
  - Also, rstrip() and lstrip(): Returns a right/left trim version of the string
- center(num): Returns a centered string (with equal padding on both sides)
  - Also ljust(num) and rjust(num): Returns a right justified version of the string
- zfill(num): Fills the string with a specified number of 0 values at the beginning

Let's explore this in Jupyter!

Text manipulation: checking string properties

There are also many functions to check properties of strings including:

- isalnum(): Returns True if all characters in the string are alphanumeric
- isalpha(): Returns True if all characters in the string are in the alphabet
- isnumeric(): Returns True if all characters in the string are numeric

- isspace(): Returns True if all characters in the string are whitespaces

- islower(): Returns True if all characters in the string are lower case
- isupper(): Returns True if all characters in the string are upper case
- istitle()`: Returns True if the string follows the rules of a title

Let's explore this in Jupyter!

Text manipulation: splitting and joining strings

There are several methods that can help us join strings that are contained into a list into a single string, or conversely, parse a single string into a list of strings. These include:

- split(separator_string): Splits the string at the specified separator, and returns a list
- splitlines(): Splits the string at line breaks and returns a list
- join(a_list): Converts the elements of an iterable into a string

Let's explore this in Jupyter!

Text manipulation: finding and replacing substrings

Some methods for locating a substring within a larger string include:

- count(substring): Returns the number of times a specified value occurs in a string
- rfind(substring): Searches the string for a specified value and returns the last position of where it was found.
- startswith(substring): Returns true if the string starts with the specified value
- endswith(substring) : Returns true if the string ends with the specified value
- replace(original_str, replacement_str): Replace a substring with a different string.

Let's explore this in Jupyter!

Text manipulation: filling in strings with values

There are a number of ways to fill in strings parts of a string with particular values.

Perhaps the most useful is to use "f strings", which have the following syntax such as:

- value_to_fill = “my_value”
- f"my string {value_to_fill} will be filled in"

Let's explore this in Jupyter!