Teaching statistics to business students… the hard way

- https://jmaurit.github.io/anv_statistikk
- https://jmaurit.github.io/stats

Johannes Mauritzen, NTNU Business School. jmaurit.github.io

Agenda

- Quick background
- Aspirations for the course
- Materials and texts
- Course setup
- Cases/Examples
  - Visualisation
  - Simulation approach
- Student feedback
-
- Discussion, feedback

My background for teaching applied stats

- BA in math, MA Economics, PhD Management Science (2012) (?)
- Started at NTNU HHS in January, “Business Analytics(?)”
- Empirical research in energy markets (mostly)
- Interest in applying Bayesian models
- First time teaching a pure statistics course

About the course

- 3rd year in bachelor
- Previously taught by an adjunct faculty
- Currently an elective
  - will be obligatory for new 5-year master program
- Students have had an intro to statistics, quantitative/qualitative methodology

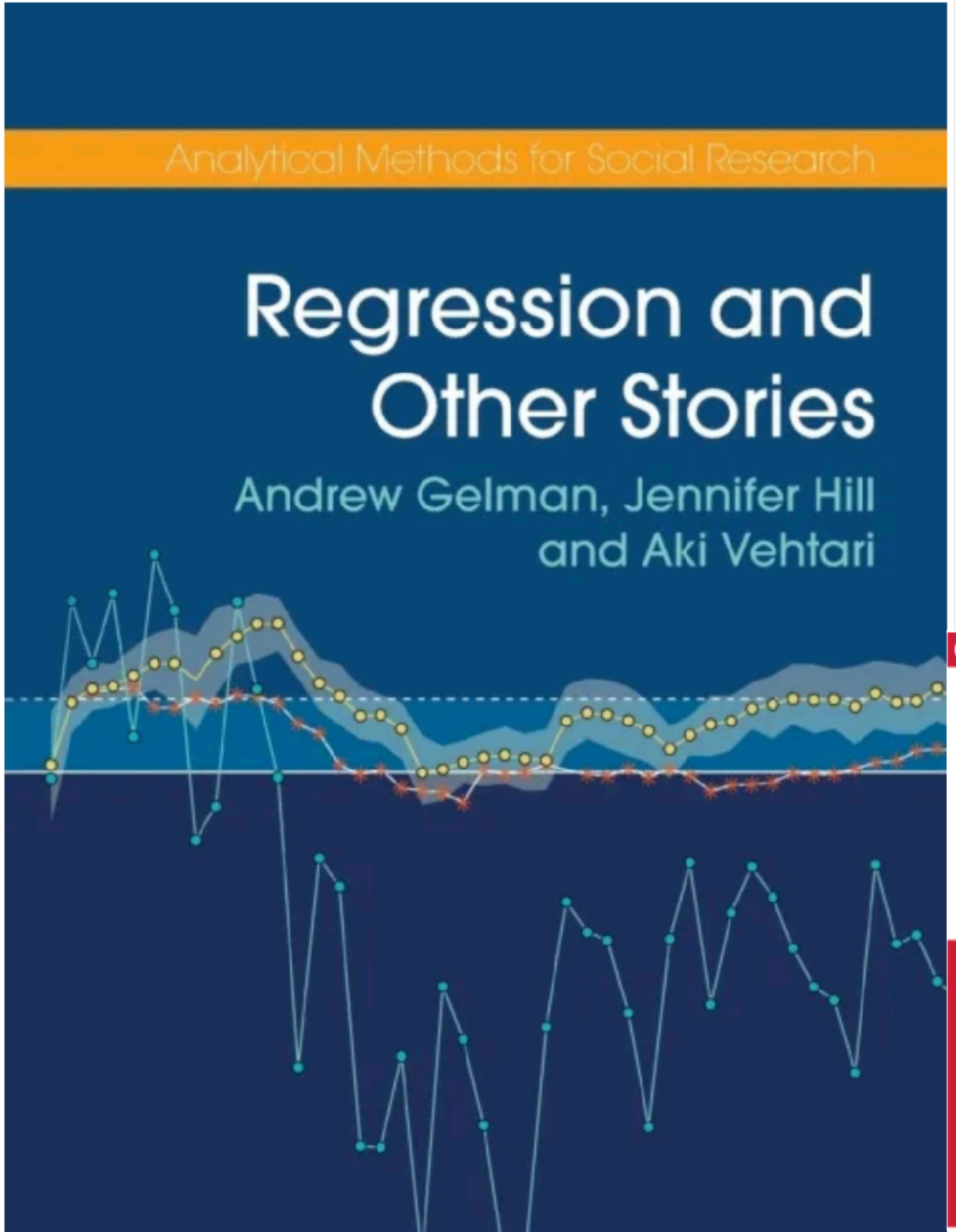
Aspirations for applied statistics

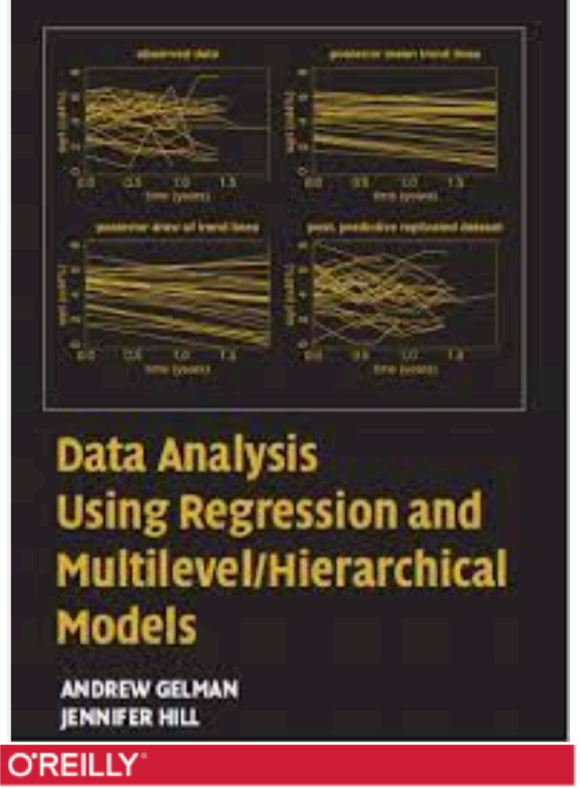
- Data literate, statistically literate students
- Using grown-up technology
- Hands-on, learning-by-doing
- Statistics and data analysis as tools for thinking
- Encourage engagement, exploration and independent thinking

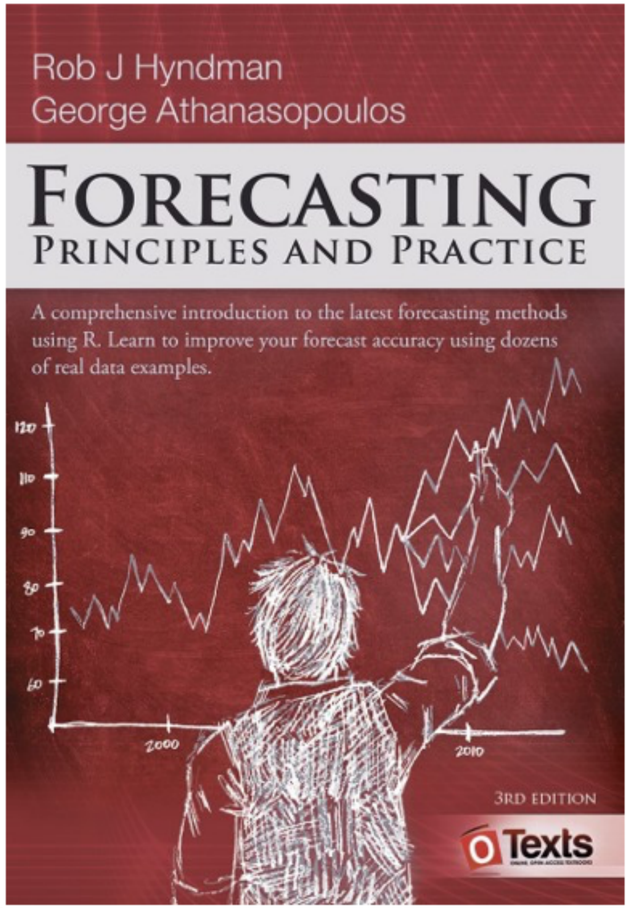
Who are business students

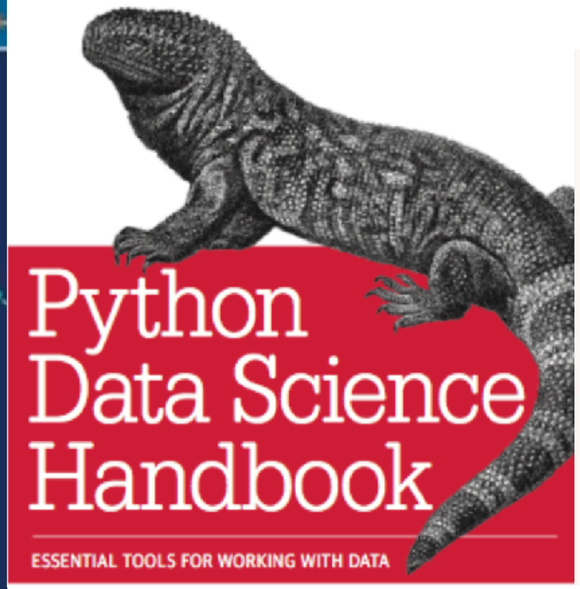
- Varied interest/background in quantitative methods/fields
- Varied experience with programming
- Strategic students
- Many balls in the air - jobs, activities (parties on Sunday night!)
- Lingering covid effect/Attendance?

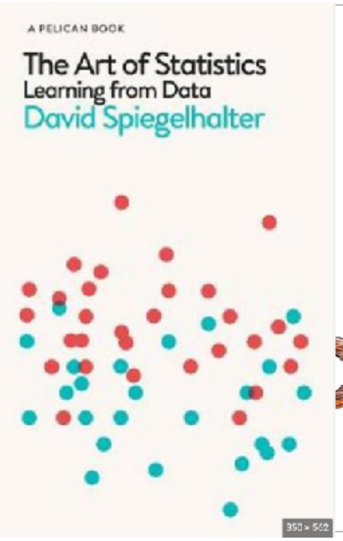
Course Set-Up

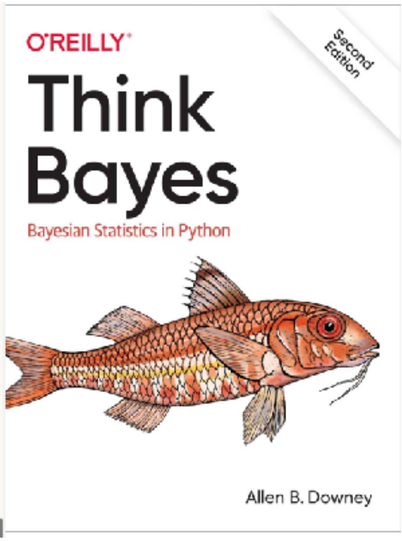

Analytical Methods for Social Research
Regression and Other Stories
Andrew Gelman, Jennifer Hill and Aki Vehtari


Data Analysis Using Regression and Multilevel/Hierarchical Models
ANDREW GELMAN
JENNIFER HILL
O'REILLY


Rob J Hyndman
George Athanasopoulos
FORECASTING
PRINCIPLES AND PRACTICE
A comprehensive introduction to the latest forecasting methods using R. Learn to improve your forecast accuracy using dozens of real data examples.
3RD EDITION
OTexts


Python Data Science Handbook
ESSENTIAL TOOLS FOR WORKING WITH DATA

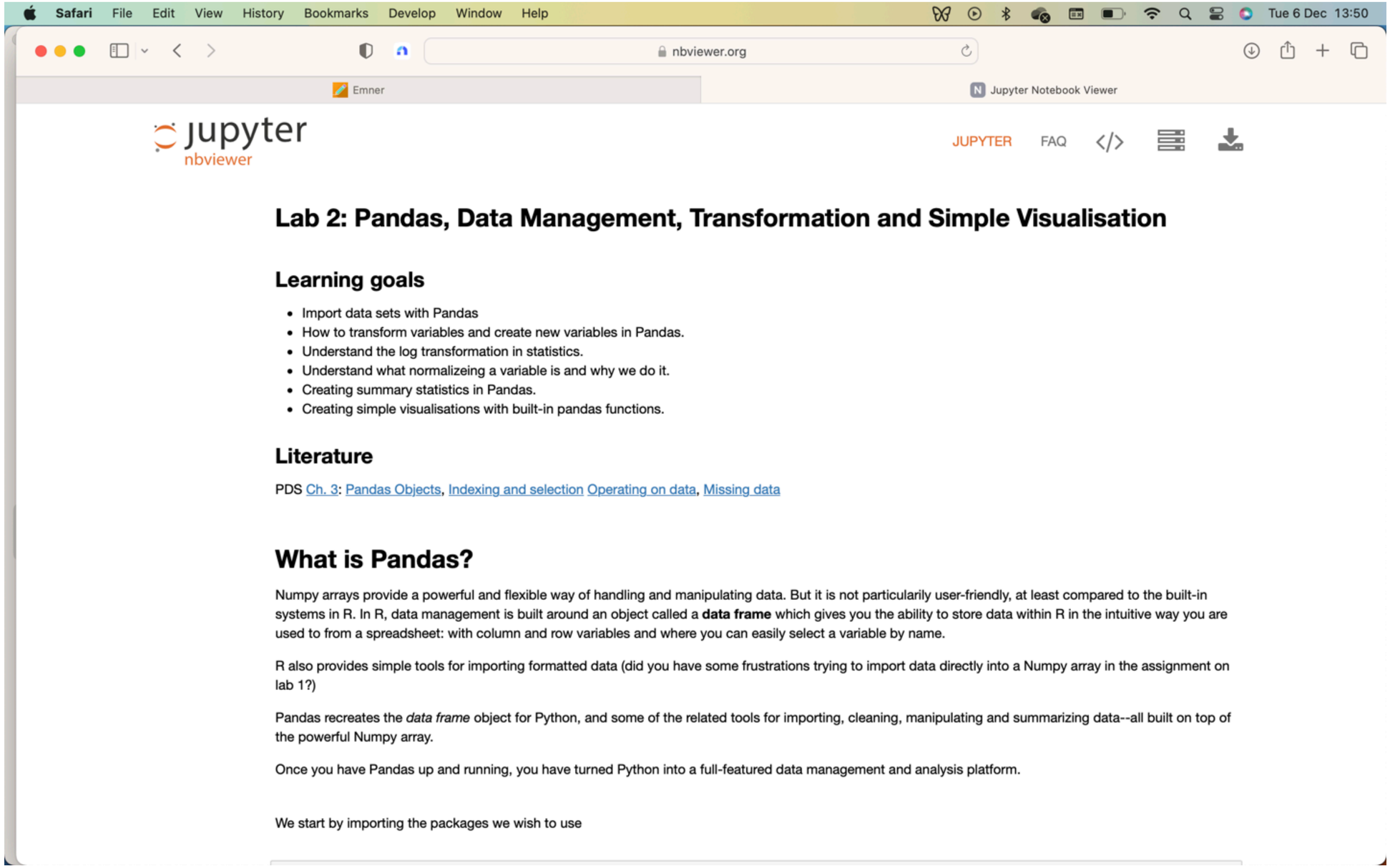

A PELICAN BOOK
The Art of Statistics
Learning from Data
David Spiegelhalter

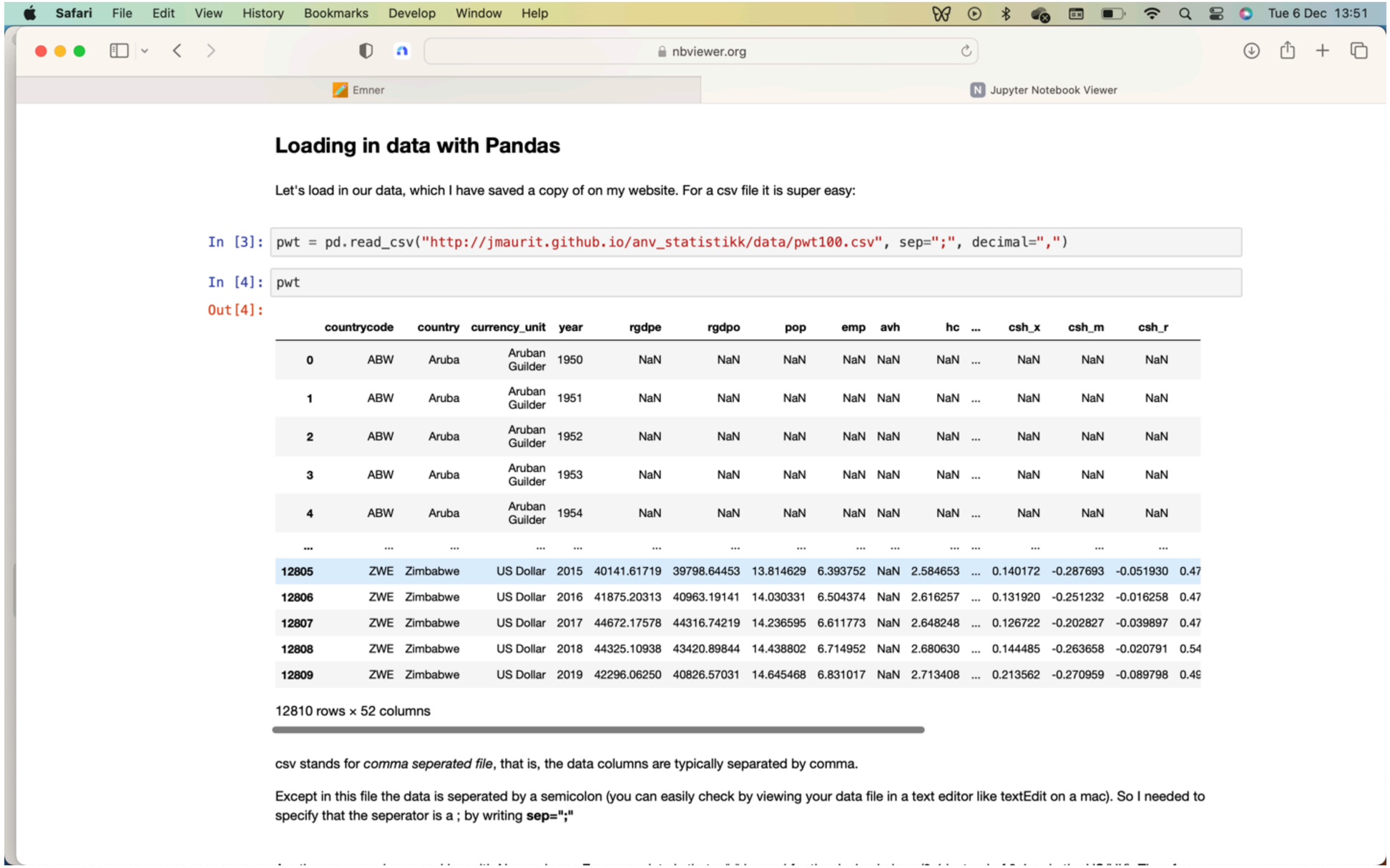

O'REILLY
Second Edition
Think Bayes
Bayesian Statistics in Python
Allen B. Downey

Uke 34	Introduksjon, Python og Numpy	Prelab Lab 1	Sketch, lab 1	PDS Ch 1, 2	22.08, 1115-1300
Uke 35	Datahåndtering med Pandas	Lab 2	Sketch, lab 2	PDS Ch 3	29.08, 1115-1300
Uke 36	Håndtering av større data: Split-Apply-Combine og Merge	Lab 3	Sketch, lab 3	PDS Ch 3	05.09, 1115-1300
Uke 37	Visualisering og transformasjon	Lab 4	Sketch, lab 4	ROS Ch 2, PDS Ch 4	12.09, 1115-1300
Uke 38	Sansynlighet og simulasjon	Lab 5	Sketch, lab 5	ROS Ch 3, Ch 4	19.09, 1115-1300
Uke 39	Inferens og simulasjon	Lab 6	Sketch, lab 6	ROS ch 5	26.09, 1115-1300
Uke 40	Enkelregresjon	Lab 7	Sketch, lab 7	ROS 6-8	03.10, 1115-1300
Uke 41	Multippelregresjon	Lab 8	Sketch, Lab 8	ROS 10, ROS 12.1-12.4	10.10, 1115-1300
Uke 42	Modell evaluering og diagnostikk	Lab 9	Sketch, Lab 9	ROS 11	17.10, 1115-1300
Uke 43	GLM og logistisk regresjon	Lab 10	Sketch, lab 10	ROS 13-14	24.10, 1115-1300
Uke 44	Bayesiansk teori og estimering	Lab 11	Sketch, lab 11	ROS 9	31.10, 1115-1300
Uke 45	Identifikasjon og kausal modellering	Lab 12	Sketch, lab 12	ROS 19-21	07.11.2022, 1115-1300
Uke 46	Tidsrekke statistikk og prognose	Lab 13	Sketch, lab 13	FPP3 Ch. 9	14.11, 1115-1300
Uke 47	Tidsrekke statistikk og prognose	Lab 14	Sketch, lab 14	FPP3 Ch. 9	21.11, 1115-1300
Uke 49	Frist: Valgfritt prosjektoppgave-innlevering for tilbakemelding	innlevering via blackboard			5.12, kl. 12

Term project

- Students choose theme and data themselves
- Turn in a “lab notebook”: Showing and discussing all steps of analysis
- Required elements: Intro, descriptive analysis/visualisation, regression model with visualisation, diagnostics and discussion of causality/underlying mechanisms
- Choose between: Time series analysis, Logistic model/GLM, Decision analysis
- Get an opportunity to turn in a draft for feedback towards end of semester


Safari File Edit View History Bookmarks Develop Window Help
Tue 6 Dec 13:50
nbviewer.org
Emner
Jupyter Notebook Viewer
jupyter
nbviewer
JUPYTER
FAQ
Lab 2: Pandas, Data Management, Transformation and Simple Visualisation
Learning goals
Import data sets with Pandas
How to transform variables and create new variables in Pandas.
Understand the log transformation in statistics.
Understand what normalizeing a variable is and why we do it.
Creating summary statistics in Pandas.
Creating simple visualisations with built-in pandas functions.
Literature
PDS Ch. 3: Pandas Objects, Indexing and selection Operating on data, Missing data
What is Pandas?
Numpy arrays provide a powerful and flexible way of handling and manipulating data. But it is not particularily user-friendly, at least compared to the built-in systems in R. In R, data management is built around an object called a data frame which gives you the ability to store data within R in the intuitive way you are used to from a spreadsheet: with column and row variables and where you can easily select a variable by name.
R also provides simple tools for importing formatted data (did you have some frustrations trying to import data directly into a Numpy array in the assignment on lab 1?)
Pandas recreates the data frame object for Python, and some of the related tools for importing, cleaning, manipulating and summarizing data--all built on top of the powerful Numpy array.
Once you have Pandas up and running, you have turned Python into a full-featured data management and analysis platform.
We start by importing the packages we wish to use



Loading in data with Pandas

Let's load in our data, which I have saved a copy of on my website. For a csv file it is super easy:

In [3]:
```
pwt = pd.read_csv("http://jmaurit.github.io/anv_statistikk/data/pwt100.csv", sep=";", decimal=",")
```

In [4]:
```
pwt
```

Out[4]:

	countrycode	country	currency_unit	year	rgdpe	rgdpo	pop	emp	avh	hc	...	csh_x	csh_m	csh_r	
0	ABW	Aruba	Aruban Guilder	1950	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	ABW	Aruba	Aruban Guilder	1951	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
2	ABW	Aruba	Aruban Guilder	1952	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	ABW	Aruba	Aruban Guilder	1953	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	ABW	Aruba	Aruban Guilder	1954	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
12805	ZWE	Zimbabwe	US Dollar	2015	40141.61719	39798.64453	13.814629	6.393752	NaN	2.584653	...	0.140172	-0.287693	-0.051930	0.47
12806	ZWE	Zimbabwe	US Dollar	2016	41875.20313	40963.19141	14.030331	6.504374	NaN	2.616257	...	0.131920	-0.251232	-0.016258	0.47
12807	ZWE	Zimbabwe	US Dollar	2017	44672.17578	44316.74219	14.236595	6.611773	NaN	2.648248	...	0.126722	-0.202827	-0.039897	0.47
12808	ZWE	Zimbabwe	US Dollar	2018	44325.10938	43420.89844	14.438802	6.714952	NaN	2.680630	...	0.144485	-0.263658	-0.020791	0.54
12809	ZWE	Zimbabwe	US Dollar	2019	42296.06250	40826.57031	14.645468	6.831017	NaN	2.713408	...	0.213562	-0.270959	-0.089798	0.49

12810 rows × 52 columns

csv stands for *comma seperated file*, that is, the data columns are typically separated by comma.

Except in this file the data is seperated by a semicolon (you can easily check by viewing your data file in a text editor like textEdit on a mac). So I needed to specify that the seperator is a ; by writing **sep=";"**

Safari File Edit View History Bookmarks Develop Window Help Tue 6 Dec 13:52

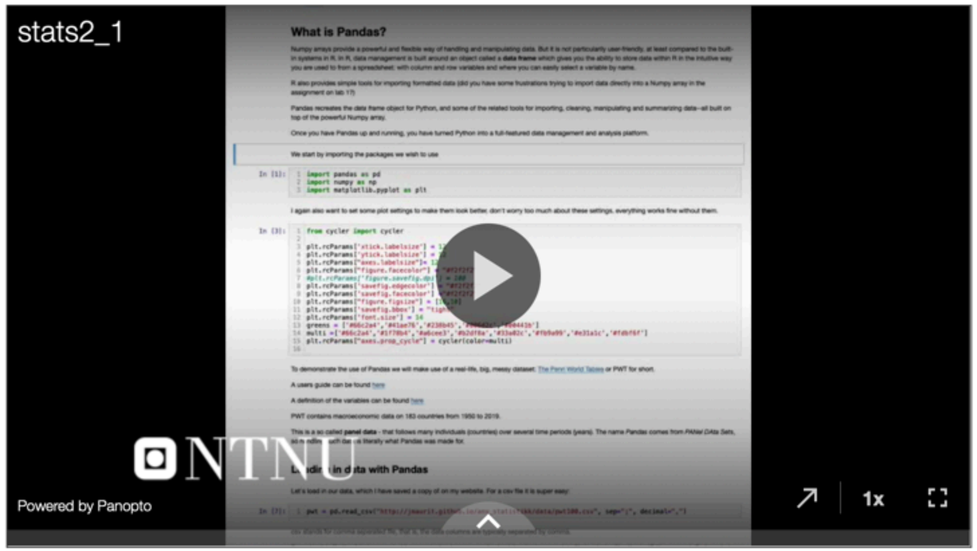
nbviewer.org

Emner | Jupyter Notebook Viewer

jupyter nbviewer

JUPYTER FAQ

Working with a Pandas object



Let's take a look at our data. We can see a snippet of our data by simply writing

In [16]:
```
pwt
```

Out[16]:

	countrycode	country	currency_unit	year	rgdpe	rgdpo	pop	emp	avh	hc	ccon	cda	cgdpe
0	ABW	Aruba	Aruban Guilder	1950	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	ABW	Aruba	Aruban	1951	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Assignment

1. In the lab we looked at real GDP expenditure. Now do an analysis with output-side real GDP.

Roughly, expenditure-side GDP measures the consumption of an economy, while the output-side GDP measures the productive capacity of an economy. In most situations these two sides should be roughly similar, but they can and do diverge.

- Create a new variable called "rgdpo_per_pers" which divides output-side real gdp ("rdgpo") by population ("pop").
- Extract a dataframe with Norwegian, Swedish and Danish data and create a chart of the rgdpo_per_pers comparing the three countries.
- Also do a log transformation and plot.
- How does this measure of GDP differ from the one above? Can you explain why?

2. Creating fake data.

You might think that statisticians would frown upon faking data (and indeed, in certain contexts they do), but it can be a useful tool in understanding models - also in this course. In this exercise you are going to create your own fake data set.

- We have a relationship we wish to model which we think has the following form:

$$y_i = \beta_1 * x_{1i} + \beta_2 * x_{2i} + e_i$$

Perhaps y_i is the test score on a standardized exam which is a linear function of how much a student, i, studies, x_{1i}, and how high their IQ is, x_{2i}, plus some random component e_i.

- Generate NP arrays for x_{1i} and x_{2i}. Generate data for N=1000 students, assuming that both x_{1i} and x_{2i} come from uniform distribution (see lab 1) between 0 and 1.
- Generate the series e_i which is from a standard normal distribution - mean 0, standard deviation 1.
- Generate y_i series by letting $\beta_1 = 0.5$ and $\beta_2 = -0.2$ and computing based on the formula above.
- Create a Pandas data frame with the 4 NP arrays you have created (See here).
- Use the Pandas function: pandas.DataFrame.corr to generate correlation coefficients between your variables. How do these estimates compare to the β_1 and β_2 values you chose

3. Open-ended assignment:

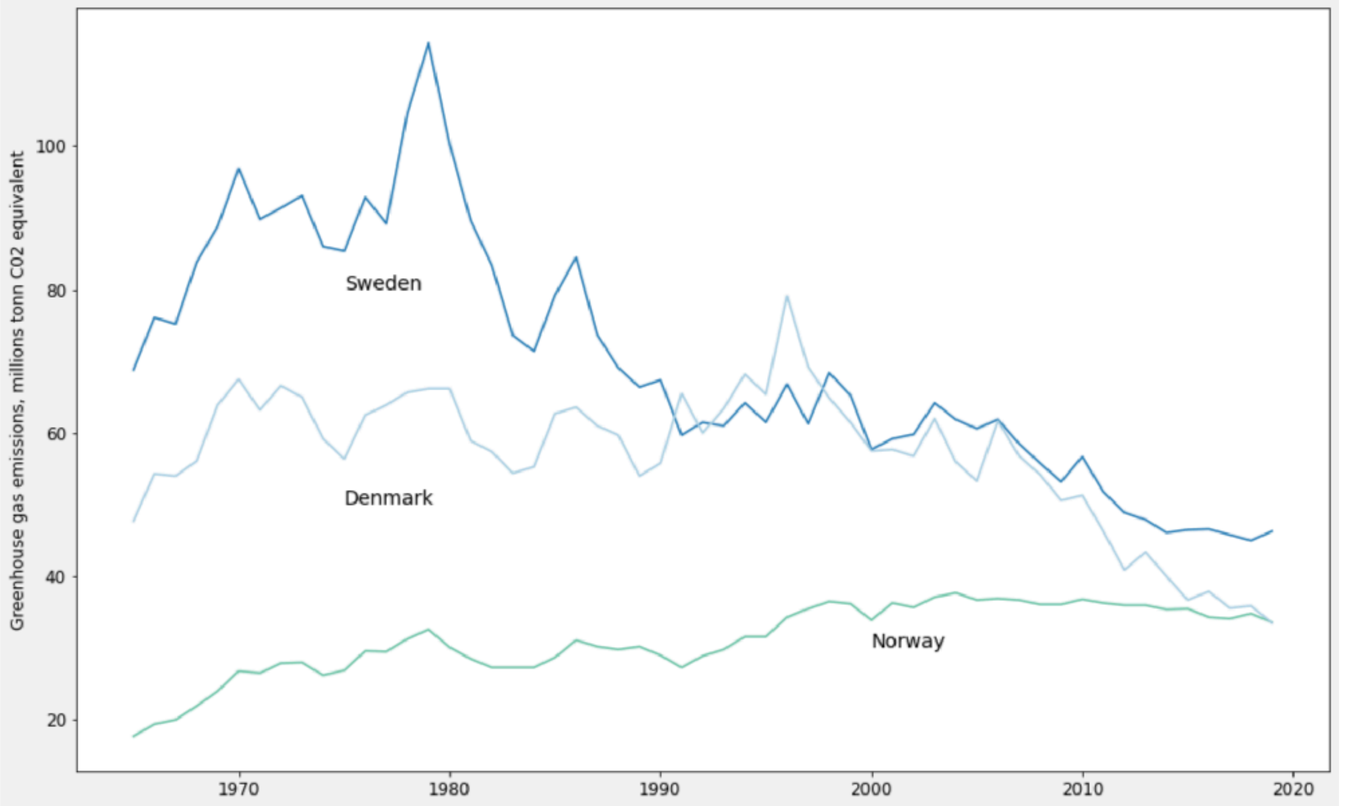
Find an interesting data set or series, and load it in as as a Pandas data frame or series. Do some initial transformations or cleaning if necessary. Plot the series or multiple series. Try to explain some of the patterns you see.

You can and should see this as the first step of starting your course project.

Examples/Cases

Visualisation

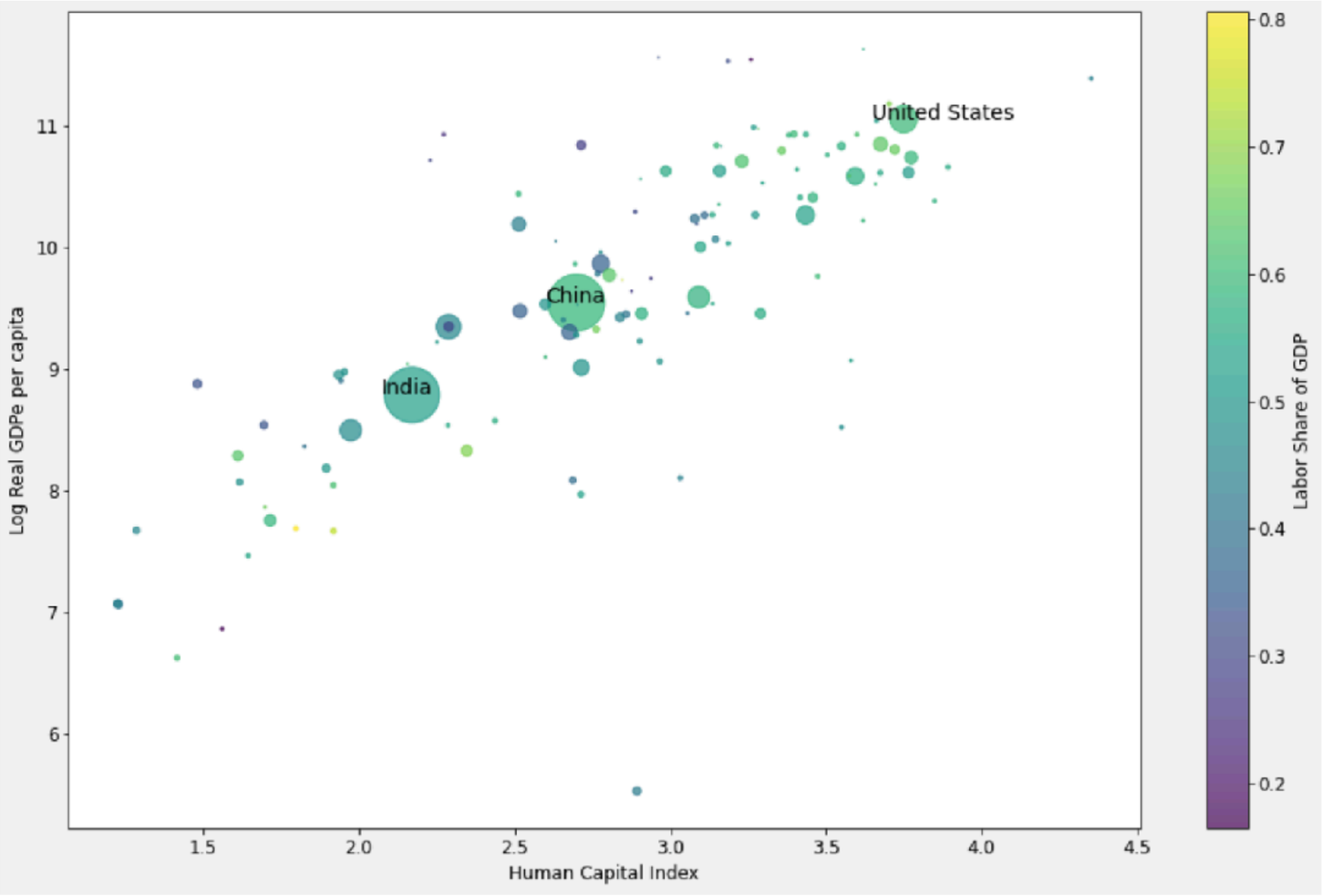
```
fig, ax = plt.subplots()
ax.plot(utslipp["year"], utslipp["Norway"])
ax.annotate("Norway", xy=(2000, 30))
ax.plot(utslipp["year"], utslipp["Sweden"])
ax.annotate("Sweden", xy=(1975, 80))
ax.plot(utslipp["year"], utslipp["Denmark"])
ax.annotate("Denmark", xy=(1975, 50))
#ax.plot(utslipp["year"], utslipp["European Union"])
#ax.annotate("Air and sea traffic, fishing", xy=(2010, 6000))
ax.set_ylabel("Greenhouse gas emissions, millions tonn C02 equivalent")
plt.show()
```



This is a fairly basic time-series chart, but it actually gives us a lot of information. We will talk more about visualisation later, but an important principle is that an important point of visualisation is to make comparisons. Here we get a very stark comparison between emissions in Norway, on the one hand, and Denmark and Sweden on the other in this 50 year period.

- Why do you think they have such diverging emissions trends?
- Why did Norway have such low emissions compared to Denmark and Sweden in the 1960s?
- What has Denmark done since 2000 that has led to such a large fall in emissions?
- Are there ways that this visualisation is *lying* or perhaps misrepresenting reality? Are there transformations we could do to make the visualisation more honest?

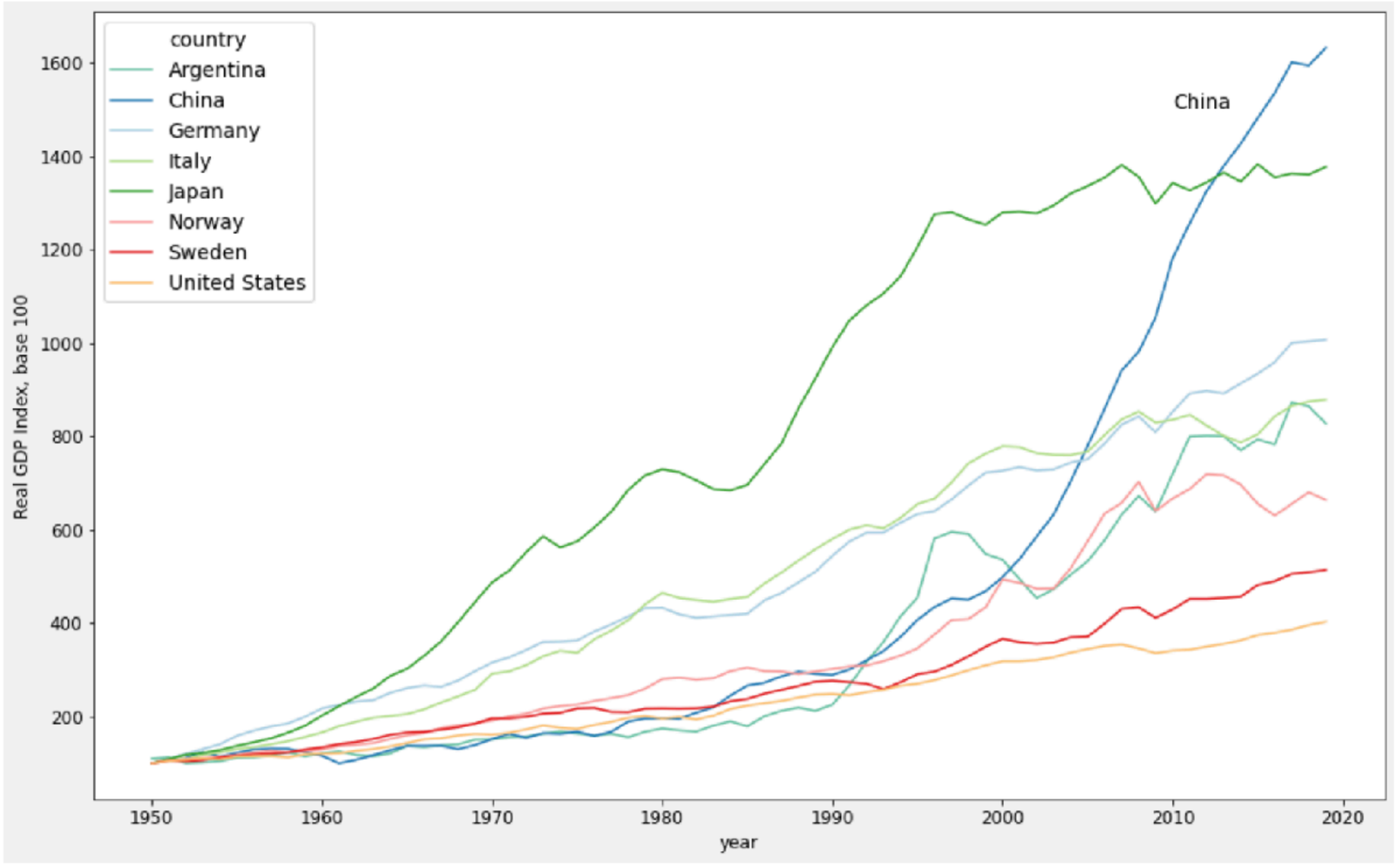
In [56]:
```
fig, ax = plt.subplots()
s=ax.scatter(x=pwt2019.hc, y=np.log(pwt2019.rgdpe_per_pers),
             c=pwt2019.labsh, s=pwt2019["pop"], alpha=.7)
for index, country in BIG.iterrows():
    s.axes.text(country.hc-.1, np.log(country.rgdpe_per_pers), country.country)
ax.set_xlabel("Human Capital Index")
ax.set_ylabel("Log Real GDPe per capita")
cb = plt.colorbar(s, ax=ax)
cb.set_label("Labor Share of GDP")

fig.savefig("rgdpxhc.png", dpi=200)
```



Why did I use pwt2019["pop"] instead pwt2019.pop above? Because .pop is already used as an attribute of a list

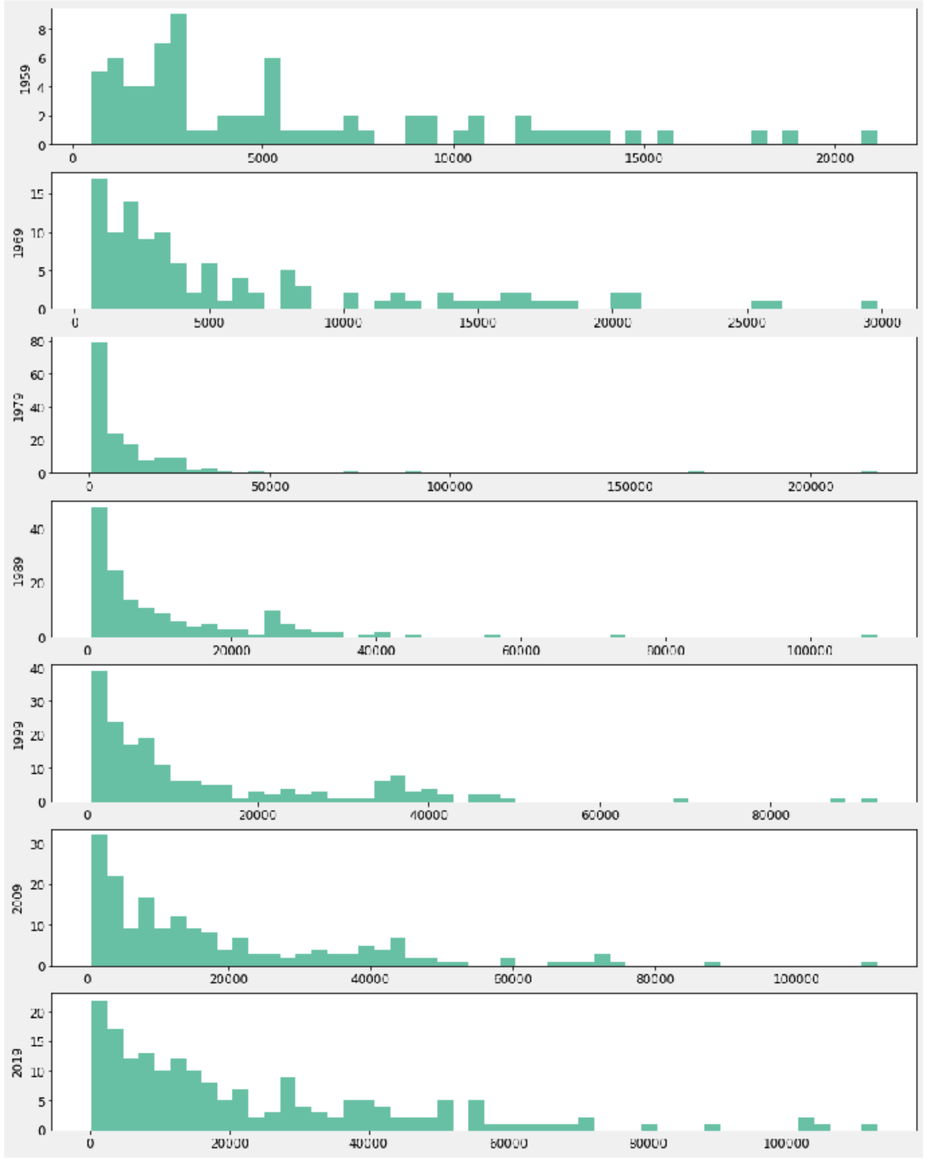
In [87]:
```
fig, ax = plt.subplots()
pwtLim.pivot(index="year", columns="country", values="gdp_index").plot(ax=ax)
ax.set_ylabel("Real GDP Index, base 100")
ax.text(2010,1500, "China")
```

Out[87]:
```
Text(2010, 1500, 'China')
```



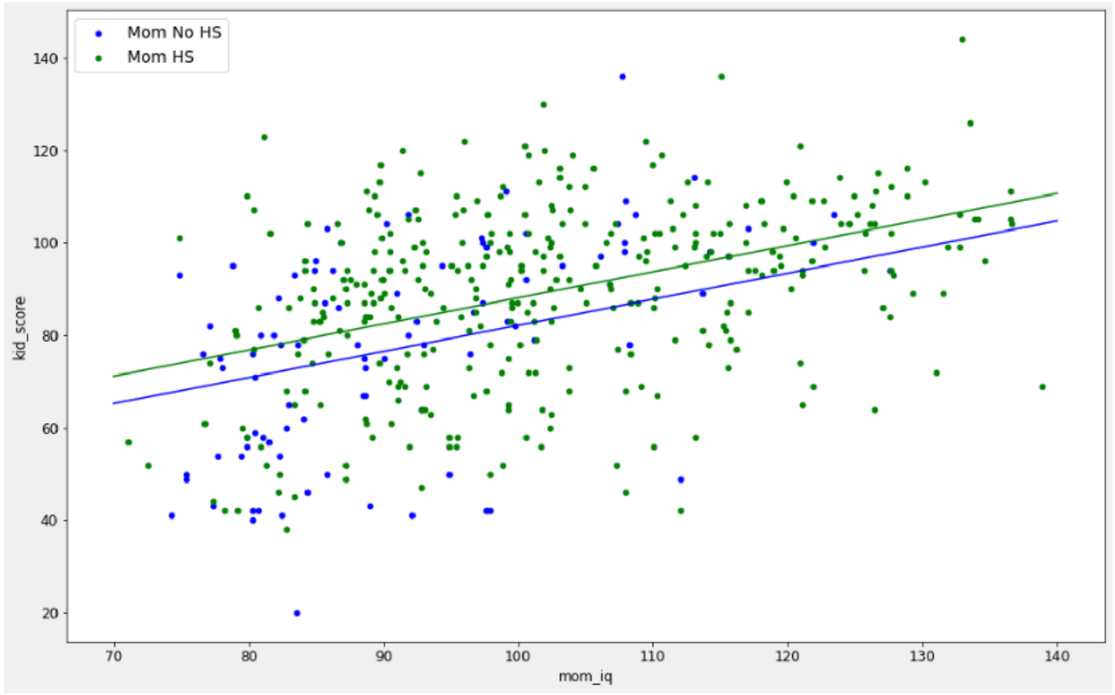
You have to be a little careful when interpreting index data. The yellow line represents USA, and looking at the 2020 data, you might be tempted to think that the US is a relatively poor country. This is not the case! Rather, it mainly represents that in 1950, the US was already a quite rich country, and its relative growth has been lower. Real GDP has increased by about a factor of 4, where in China (which was a very poor country in 1950), real per-capita GDP has increased by a factor of 160! But absolute real GDP per-capita is still higher in the US.

In [81]:
```
labels = ["1959","1969", "1979", "1989", "1999", "2009", "2019"]

fig, ax = plt.subplots(7)
i=0
for decade in byDecade:
    ax[i].hist(decade[1].rgdpe_per_pers, bins=50)
    ax[i].set_ylabel(labels[i])
    i+=1
fig.set_size_inches(15, 20)
```



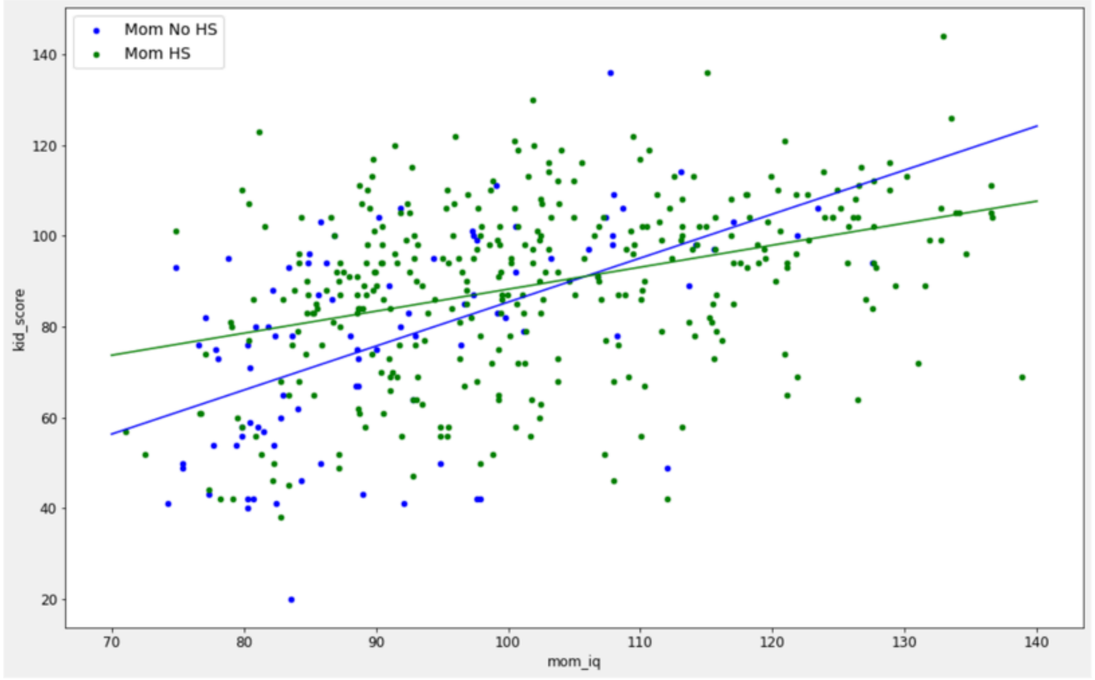
In [12]:
```
a_hat = mod3.params[0]
b1_hat = mod3.params[1]
b2_hat = mod3.params[2]

x_line = np.linspace(70,140,200)
y_line_noHS = a_hat + b1_hat*x_line
y_line_HS = a_hat + b1_hat*x_line + b2_hat

fig, ax = plt.subplots()
iq.loc[iq.mom_hs==0].plot.scatter("mom_iq", "kid_score", label="Mom No HS", color="blue", ax=ax)
iq.loc[iq.mom_hs==1].plot.scatter("mom_iq", "kid_score", label="Mom HS", color="green", ax=ax)
ax.plot(x_line,y_line_noHS, color="blue")
ax.plot(x_line,y_line_HS, color="green")
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x7f7cf1e0e7c0>]
```




```
b3_hat = mod4.params[3]

x_line = np.linspace(70,140,200)
y_line_noHS = b0_hat + b1_hat*x_line

y_line_HS = (b0_hat + b2_hat) + (b1_hat + b3_hat)*x_line

fig, ax = plt.subplots()
iq.loc[iq.mom_hs==0].plot.scatter("mom_iq", "kid_score", label="Mom No HS", color="blue", ax=ax)
iq.loc[iq.mom_hs==1].plot.scatter("mom_iq", "kid_score", label="Mom HS", color="green", ax=ax)
ax.plot(x_line,y_line_noHS, color="blue")
ax.plot(x_line,y_line_HS, color="green")
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x7fda2100a460>]
```

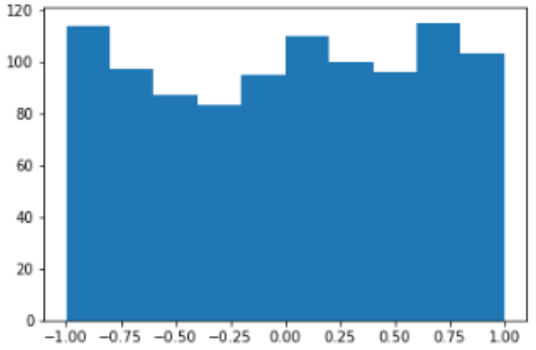


Simulation

In [24]:
```
num = 1000

error = np.random.uniform(low=-1, high=1, size=num)
```

In [25]:
```
plt.hist(error)
```

Out[25]:
```
(array([114.,  97.,  87.,  83.,  95., 110., 100.,  96., 115., 103.]),
 array([-9.99524339e-01, -7.99690569e-01, -5.99856798e-01, -4.00023028e-01,
        -2.00189258e-01, -3.55487145e-04,  1.99478283e-01,  3.99312054e-01,
         5.99145824e-01,  7.98979594e-01,  9.98813365e-01]),
 <BarContainer object of 10 artists>)
```



$$\mu = \frac{a+b}{2} = \frac{1--1}{2} = 0$$

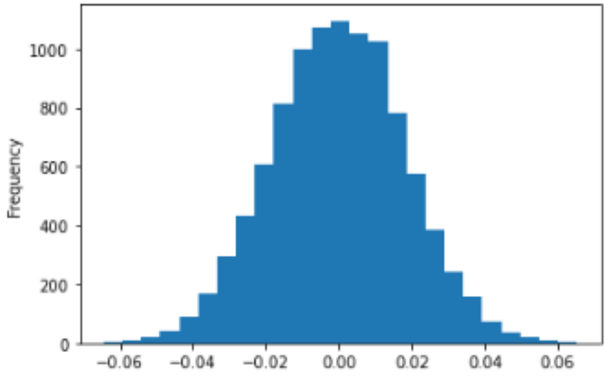
$$\sigma^2 = \frac{(b-a)^2}{12} = \frac{(1--1)^2}{12} = \frac{1}{3}$$

$$\sigma = \sqrt{(1/3)} \approx 0.577$$

In [29]:
```
n = 1000 #number of draws from a uniform distribution
m = 10000 #number of mean values of the 100 draws
meanValues = [] # initialize an empty array

for i in range(0,m): #create a for loop
    uniData = np.random.uniform(low=-1, high=1, size=n) #generate 100 random numbers from a uniform distribution
    meanValues.append(uniData.mean()) #take the mean of the generated data, then store than mean in MeanValues
```

In [30]:
```
pd.Series(meanValues).plot.hist(bins=25)
```

Out[30]:
```
<AxesSubplot:ylabel='Frequency'>
```

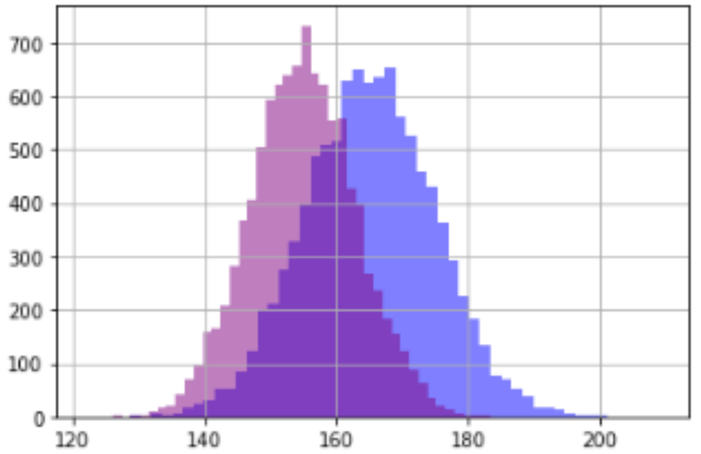


Let us say we now have a distribution of Female heights and a distribution of Male heights:

$$\sigma_D = \sqrt{\sigma_{XY} + \sigma_{XX}}$$

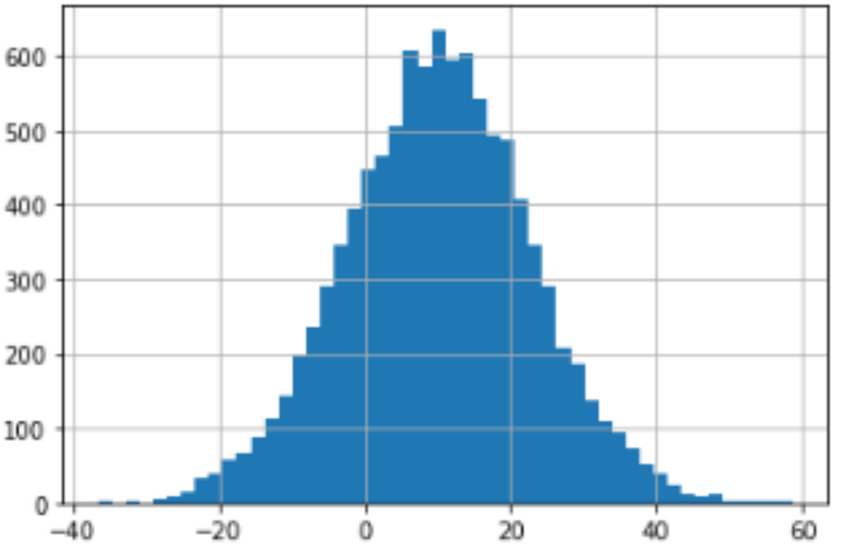
In [110]:
```
XY = pd.Series(np.random.normal(165,10,10000))
XX = pd.Series(np.random.normal(155,8, 10000))

fig, ax = plt.subplots()
XY.hist(ax=ax, color="Blue", alpha=.5, bins=50)
XX.hist(ax=ax, color="Purple", alpha=.5, bins=50)
```

Out[110]: <AxesSubplot:>



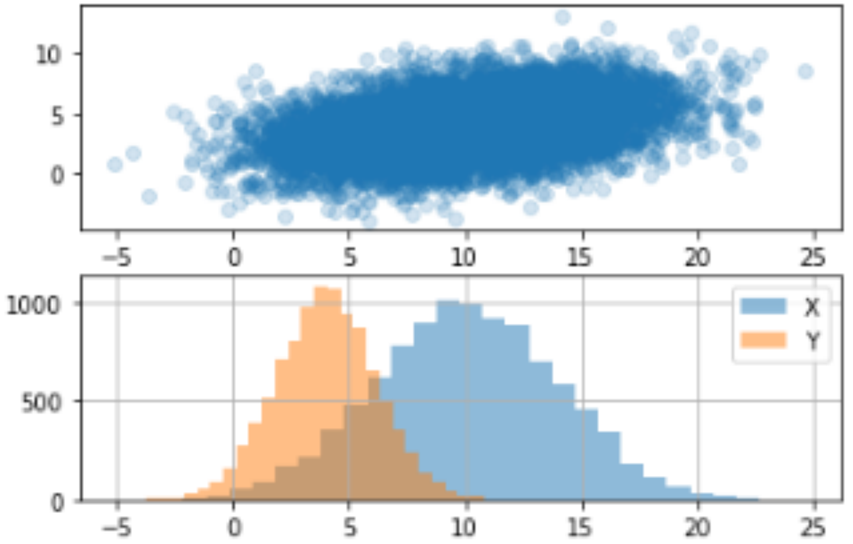
In [111]:
```
D = XY-XX
D.hist(bins=50)
print("Mean:", D.mean(), "Std. Dev:", D.std())
```

```
Mean: 10.090006359997435 Std. Dev: 12.634029955688597
```



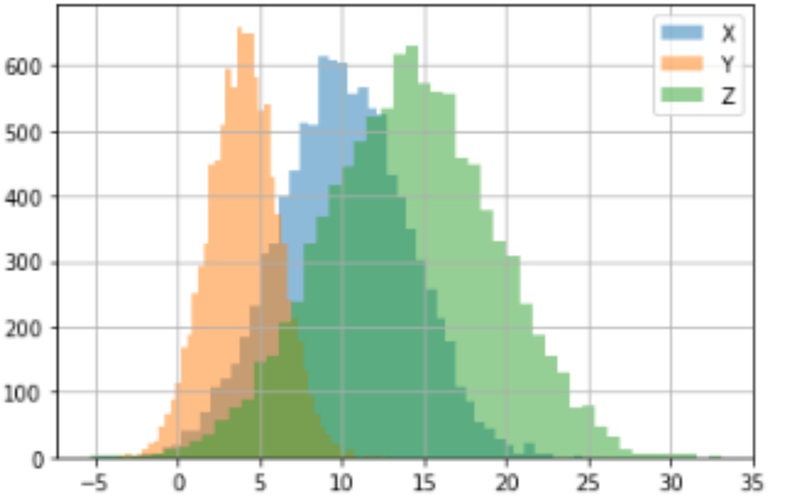
In [50]:
```
fix, ax = plt.subplots(2)
ax[0].scatter(X,Y, alpha=.2)
X.hist(bins=30, ax=ax[1], alpha=.5, label="X")
Y.hist(bins=30, ax=ax[1], alpha=.5, label="Y")
plt.legend()
```

Out[50]: <matplotlib.legend.Legend at 0x7f9d391b5550>



In [53]:
```
Z=X+Y

fix, ax = plt.subplots(1)
X.hist(bins=50, ax=ax, alpha=.5, label="X")
Y.hist(bins=50, ax=ax, alpha=.5, label="Y")
Z.hist(bins=50, ax=ax, alpha=.5, label="Z")


plt.legend()
```

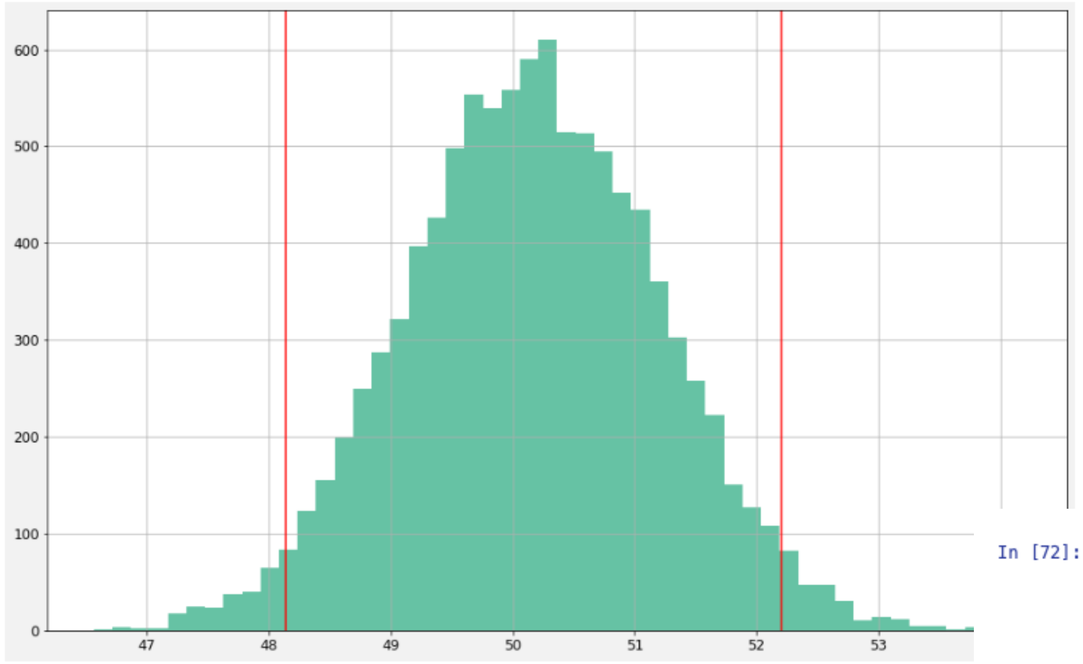
Out[53]: <matplotlib.legend.Legend at 0x7f9d485dd040>



Confidence intervals

In [30]:
```
fig, ax =plt.subplots()
sampleMeans.hist(bins=50, ax=ax)
ax.axvline(CI_neg, color="red")
ax.axvline(CI_pos, color="red")
```

Out[30]: <matplotlib.lines.Line2D at 0x7f77b0611a60>



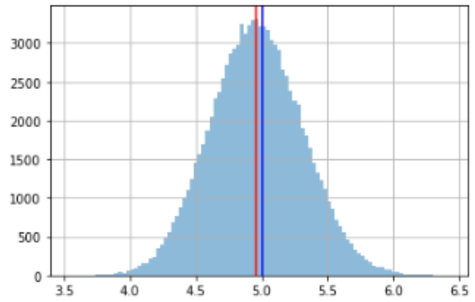
Bias

$$s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

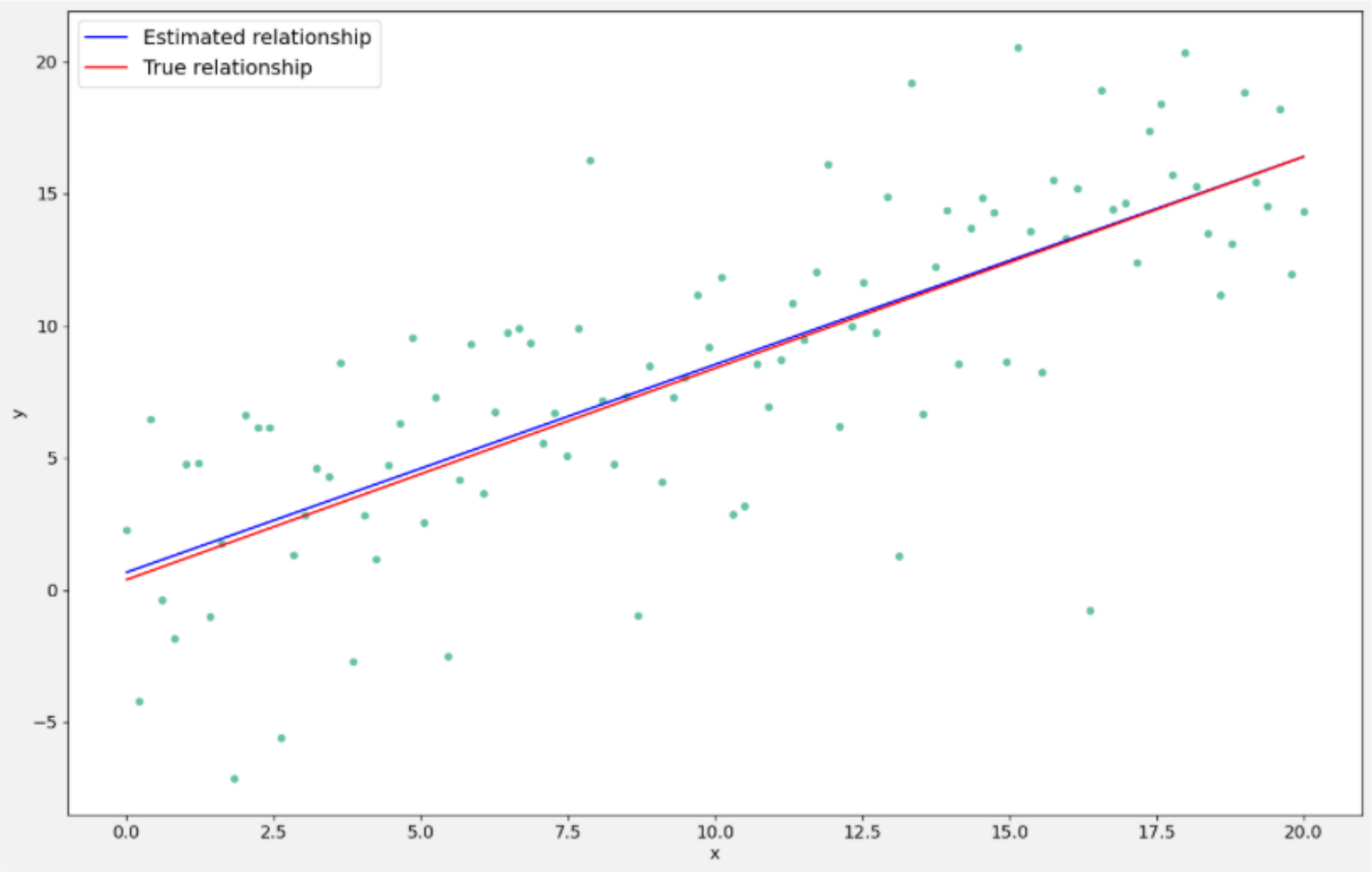
why do we multiply by $\frac{1}{n-1}$ rather than the more intuitive $\frac{1}{n}$

In [72]:
```
true_mean = 10
true_sd = 5
n=100

#we'll generate a sample 100,000 times and take the sample standard deviation **without** the n-1 adjustment
nsim=100000

sstds = pd.Series([np.random.normal(true_mean, true_sd, n).std() for i in range(nsim)])

simMeanSD = sstds.mean()

fig, ax = plt.subplots()
sstds.hist(bins=100, alpha=.5)
ax.axvline(x=simMeanSD, color="red")
ax.axvline(x=true_sd, color="blue")
```

Out[72]: <matplotlib.lines.Line2D at 0x7f8c48df00a0>



```
n = 100
x = np.linspace(0,20,n) # create n equally spaced numbers between 0 and 20
sigma = 4
a= .4
b= .8
epsilon = spt.norm.rvs(0,sigma,n,random_state=1234)
y = a + b*x + epsilon
```

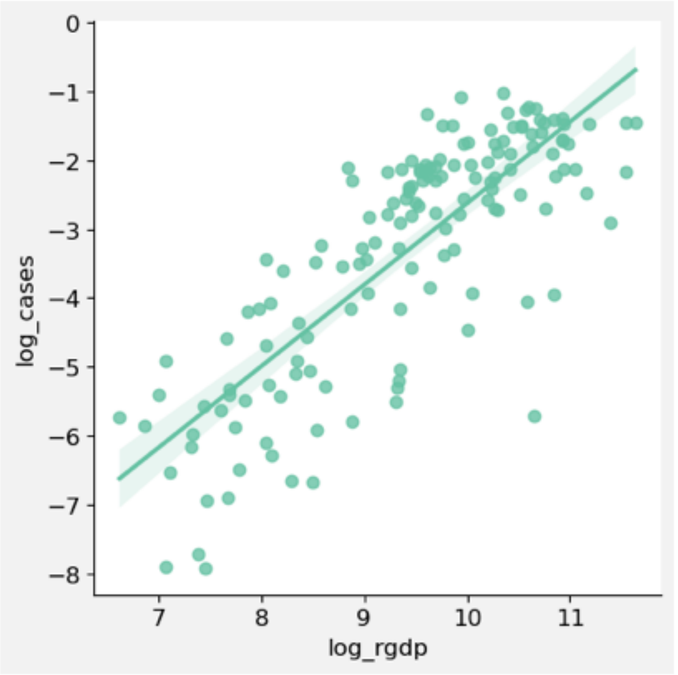


<—Fake data

Real data

```
sns.lmplot(x="log_rgdp", y="log_cases", data=pwt)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa23057fdc0>
```

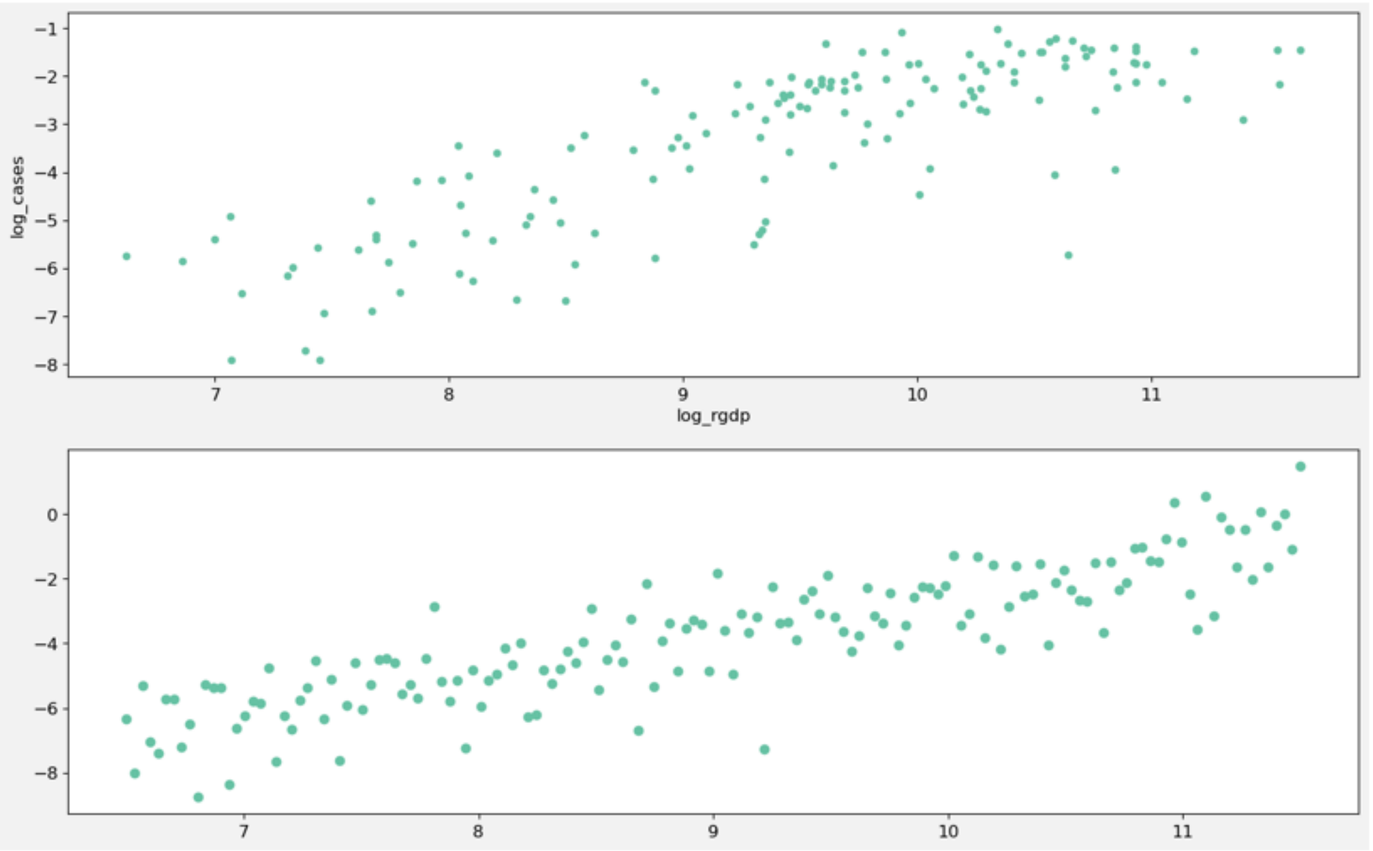


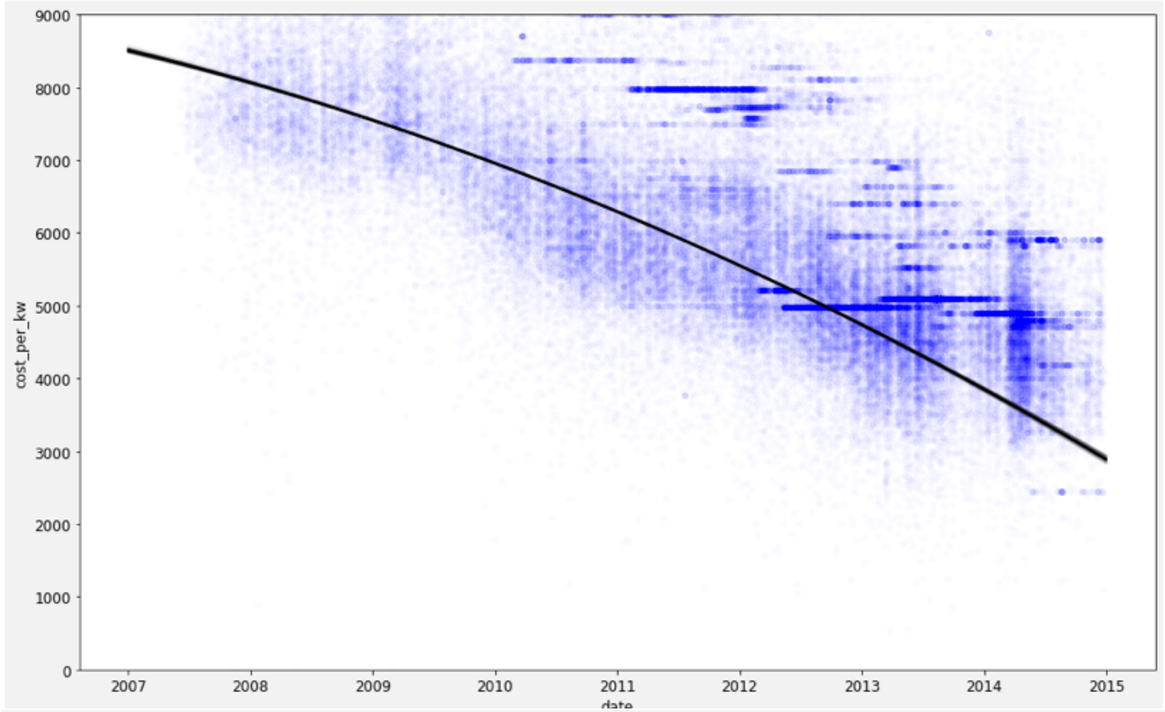
Fake data after estimation (predictive check)

```
a= -14.5
b=1.18
n=150
sigma = 1.02
x=np.linspace(6.5,11.5,n)
epsilon = spt.norm.rvs(0,sigma,n,random_state=1234)

y=a+b*x + epsilon
```

In [37]:
```
fig, ax = plt.subplots(2)
pwt.plot.scatter("log_rgdp", "log_cases", ax=ax[0])
ax[1].scatter(x,y)
```

Out[37]: <matplotlib.collections.PathCollection at 0x7fa222fc3e80>

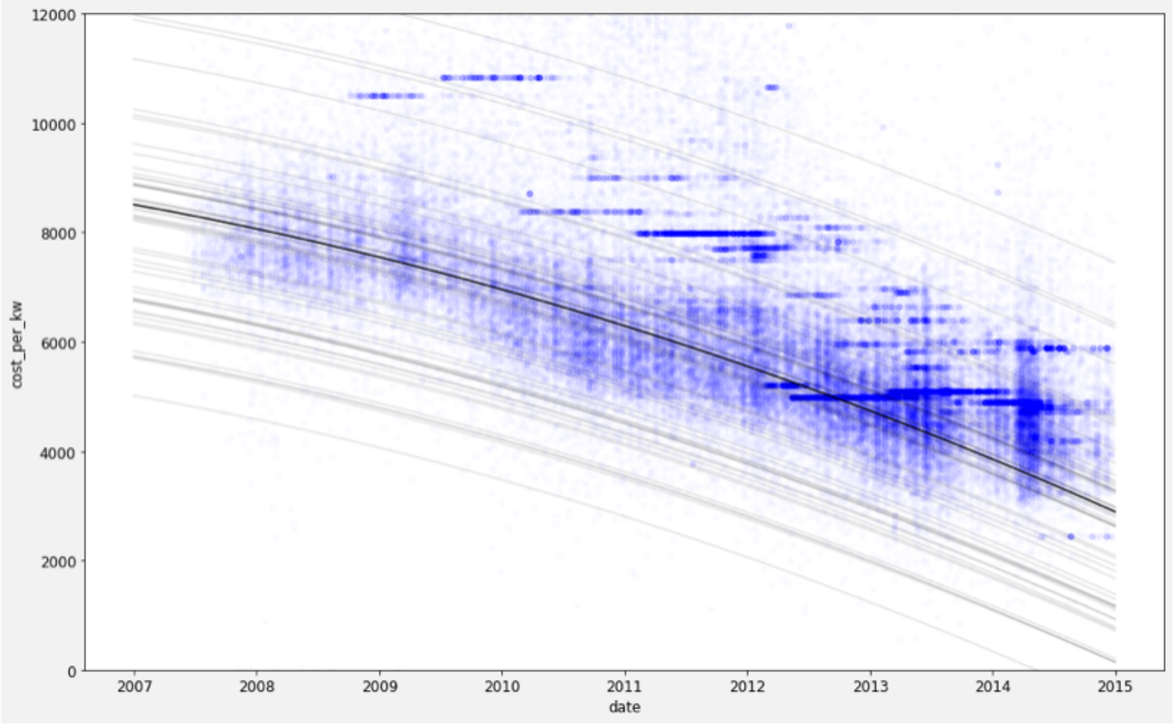




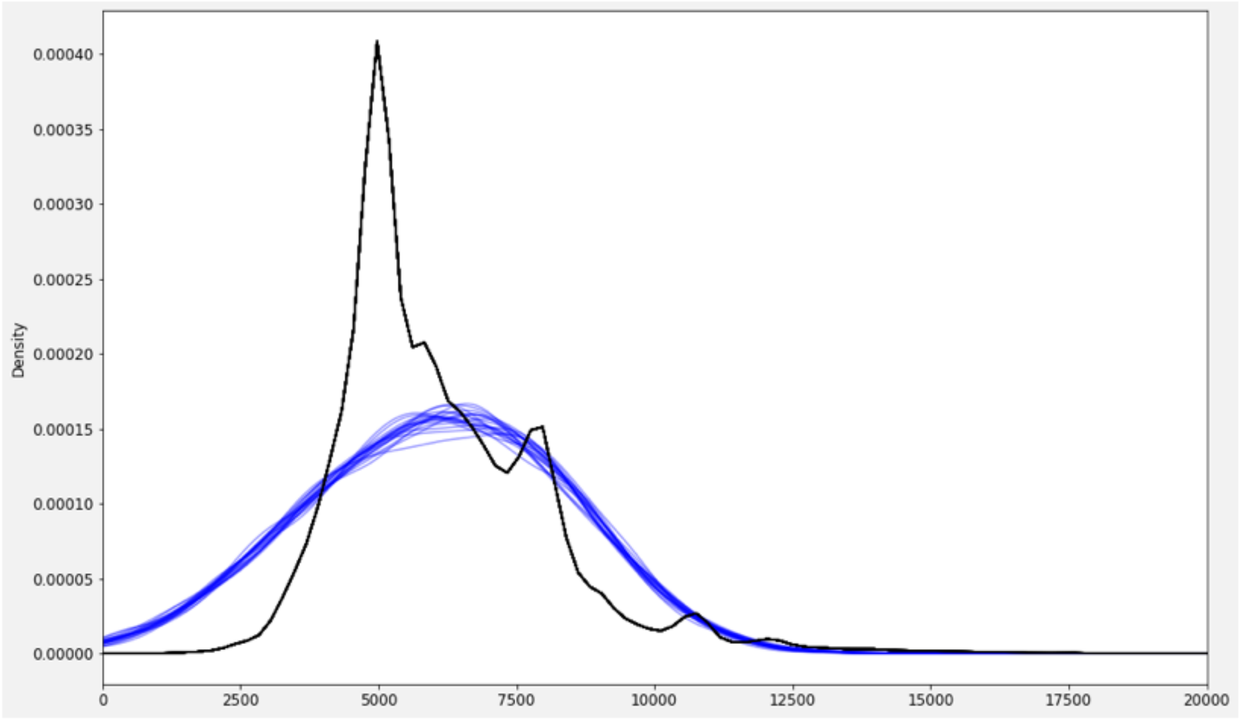
Inferential uncertainty

```
In [16]: def regSim(regMod):
             #extract values from regression model
             nmk = regMod.df_resid #n-k
             sigma_hat = np.sqrt(regMod.mse_resid)
             bs_vcov = regMod.cov_params(scale=1)
             bs = regMod.params

             #create simulated values
             sigma_sim = sigma_hat*np.sqrt((nmk/np.random.chisquare(nmk,1)))
             V_sim = np.array(bs_vcov) * sigma_sim**2
             bs_sim = np.random.multivariate_normal(bs, V_sim, 1)

             return([bs_sim.flatten(), sigma_sim])
```

Predictive uncertainty



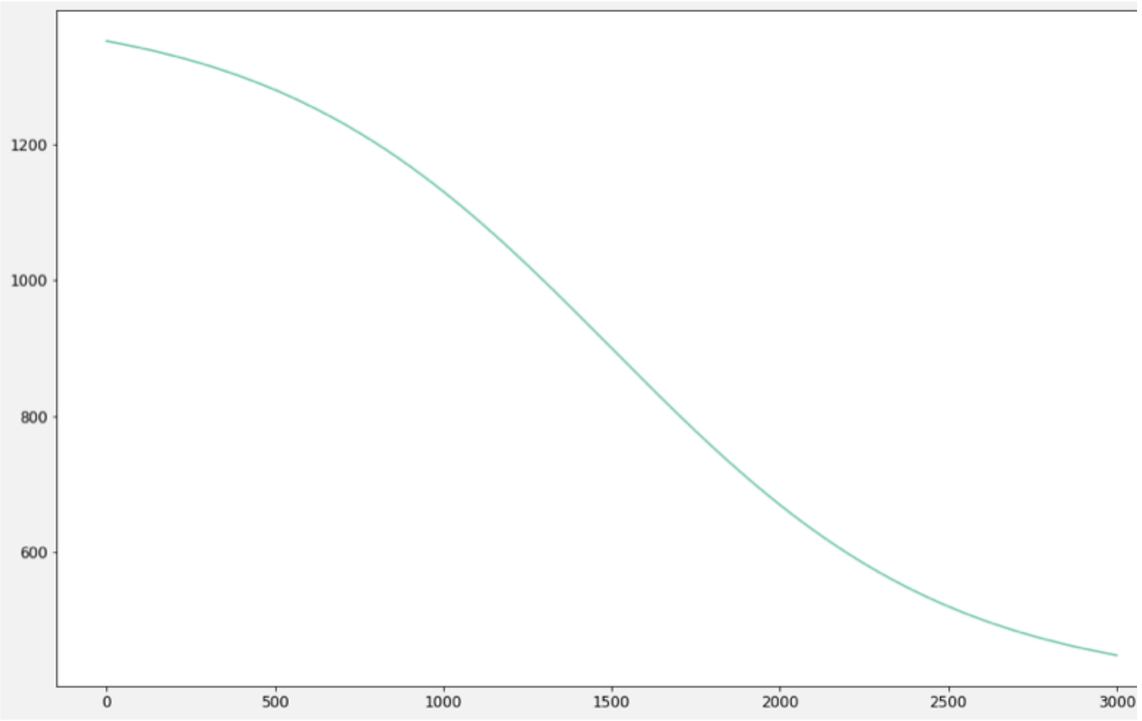
Predictive check



In [15]:
```
T = 3000
ts = np.linspace(0,T, T)
L = 1000
B = 400
k = .002
t0 = ts.mean()
costFunc =L - L/(1 + np.exp(-k*(ts-t0))) + B
```

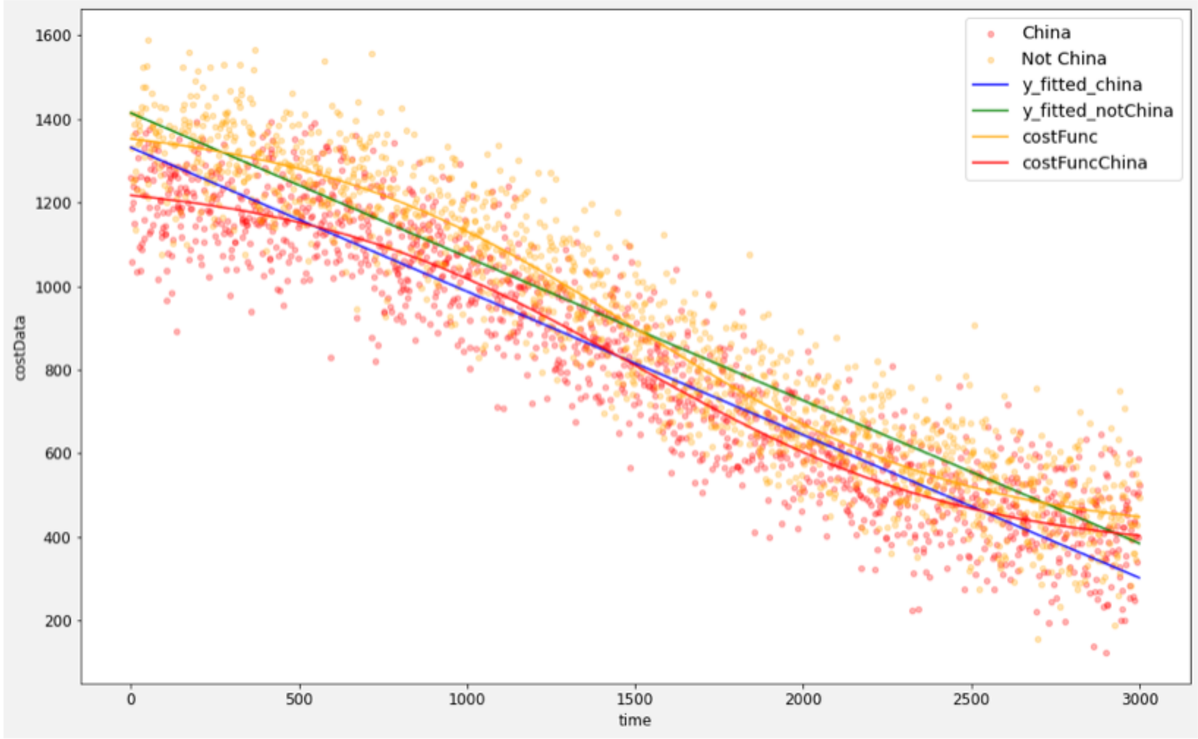
In [16]:
```
fig, ax = plt.subplots()
ax.plot(ts, costFunc)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7fdf41a5a8e0>]
```

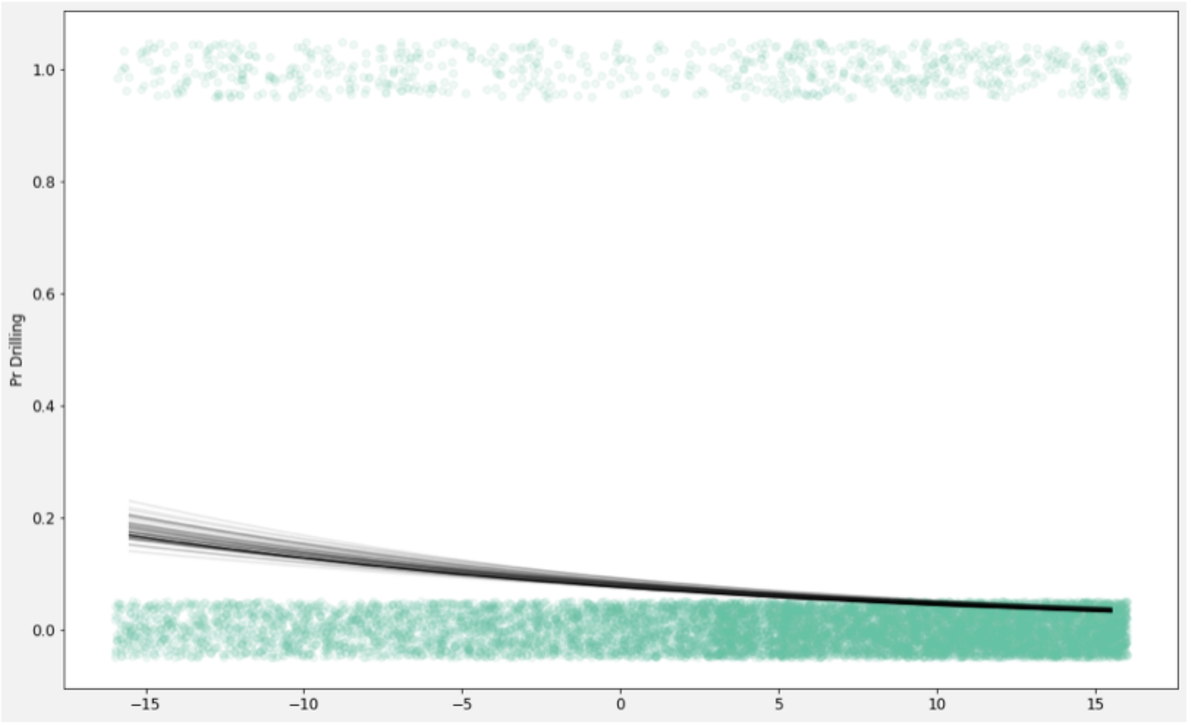



```
fig, ax = plt.subplots()
fakeDF.loc[fakeDF.china==1, :].plot.scatter("time", "costData", ax=ax, color="red", label="China", alpha=.3)
fakeDF.loc[fakeDF.china==0, :].plot.scatter("time", "costData", ax=ax, color="Orange", label="Not China", alpha=.3)
fakeDF.plot("time", "y_fitted_china", color="blue", ax=ax)
fakeDF.plot("time", "y_fitted_notChina", color="green", ax=ax)
fakeDF.plot("time", "costFunc", color="orange", ax=ax)
fakeDF.plot("time", "costFuncChina", color="red", ax=ax)

ax.legend()
```

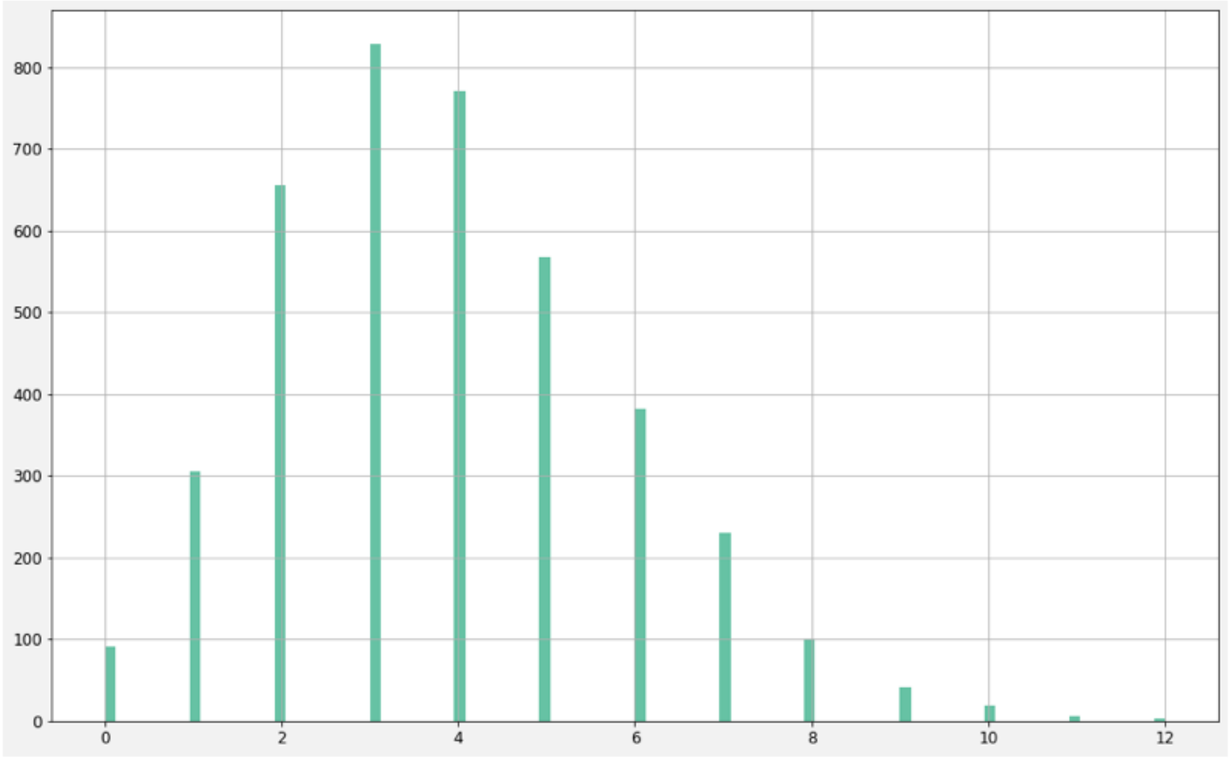
```
<matplotlib.legend.Legend at 0x7fdf24aa4c40>
```



Intro to logistic/glm




```
In [87]: drilling_sim.hist(bins=100)
Out[87]: <AxesSubplot:>
```



Bayesian analysis/ Decision Analysis

An application of Bayes' Theorem: When is a positive corona test really positive?

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

Or specifically, in our case:

$$P(Corona = 1|Test = 1) = \frac{P(Corona = 1) * P(Test = 1|Corona = 1)}{P(Test = 1)}$$

We can then create a low and high scenario:

$$P(Corona = 1|Test = 1) = \frac{.01 * .72}{.017}$$
$$P(Corona = 1|Test = 1) = \frac{.1 * .72}{.08}$$

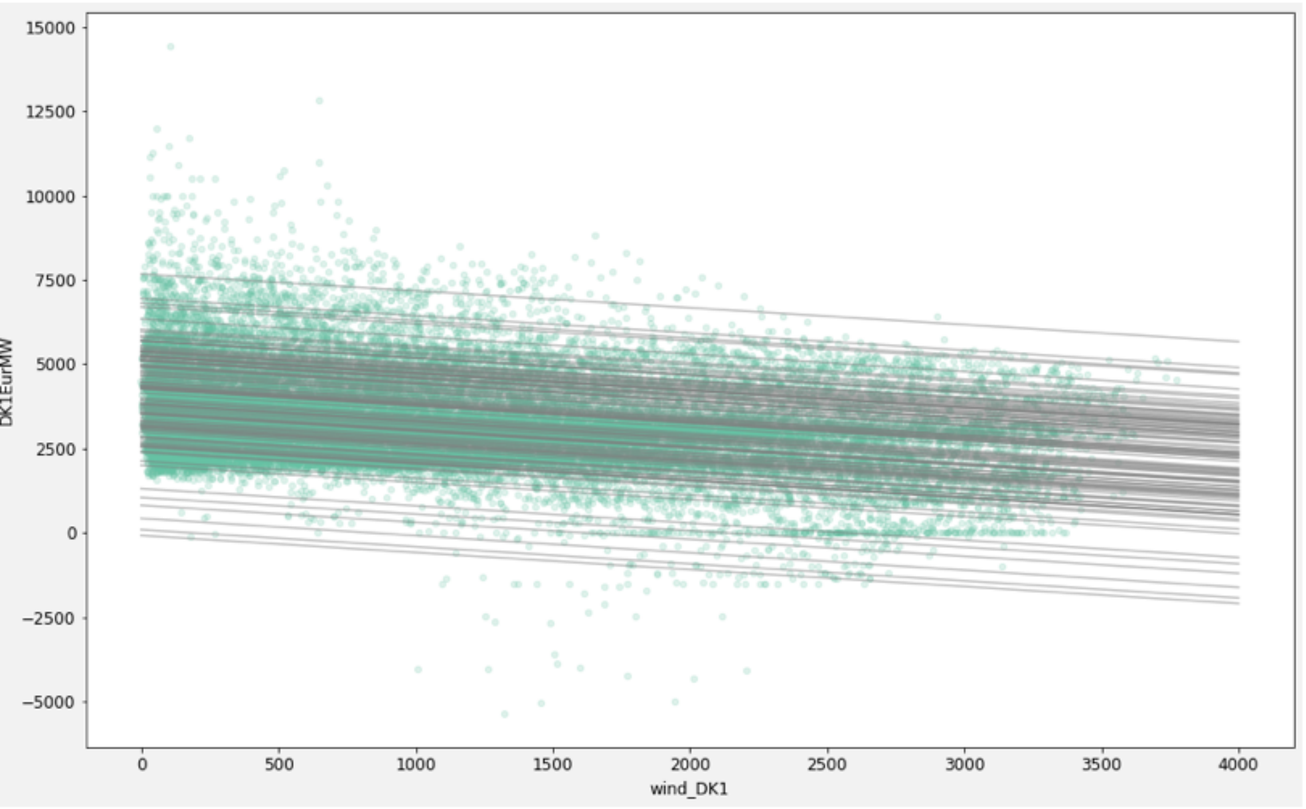
The Monty Hall problem

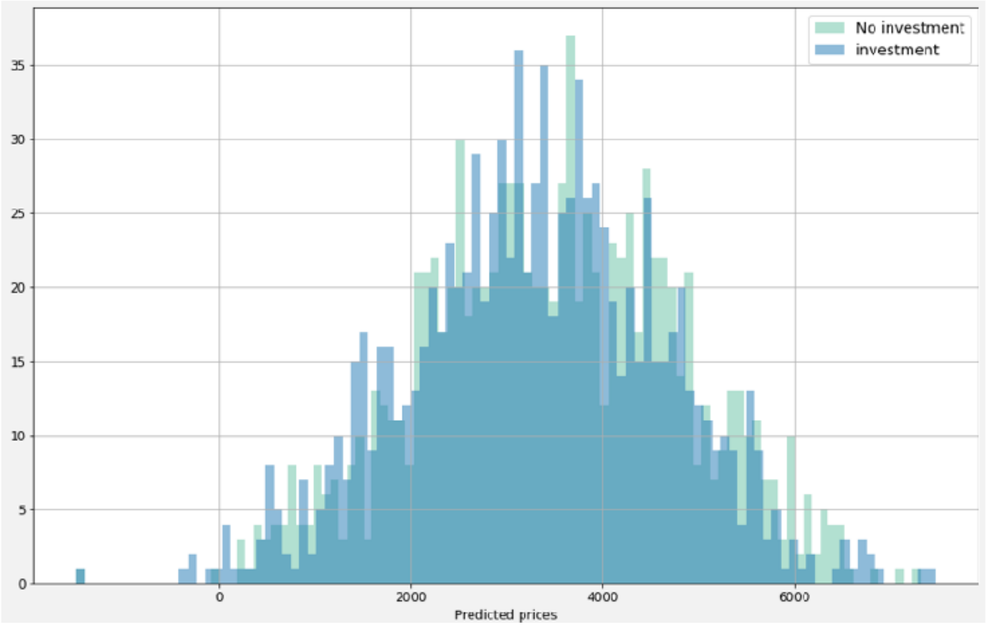
```
def update(table):
    """Compute the posterior probabilities."""
    table['unnorm'] = table['prior'] * table['likelihood']
    prob_data = table['unnorm'].sum()
    table['posterior'] = table['unnorm'] / prob_data
    return prob_data
```

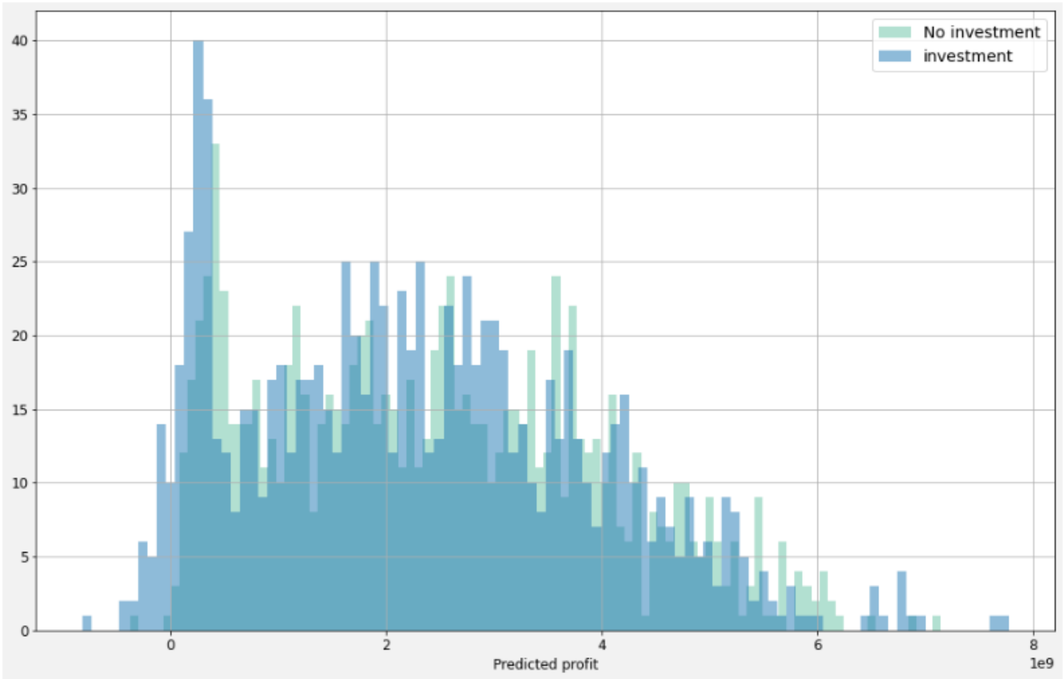
```
update(MHtable)
MHtable
```

	prior	likelihood	unnorm	posterior
Door 1	1/3	1/2	1/6	1/3
Door 2	1/3	1	1/3	2/3
Door 3	1/3	0	0	0

Propogation of uncertainty and decision analysis: Investing in a new wind power farm



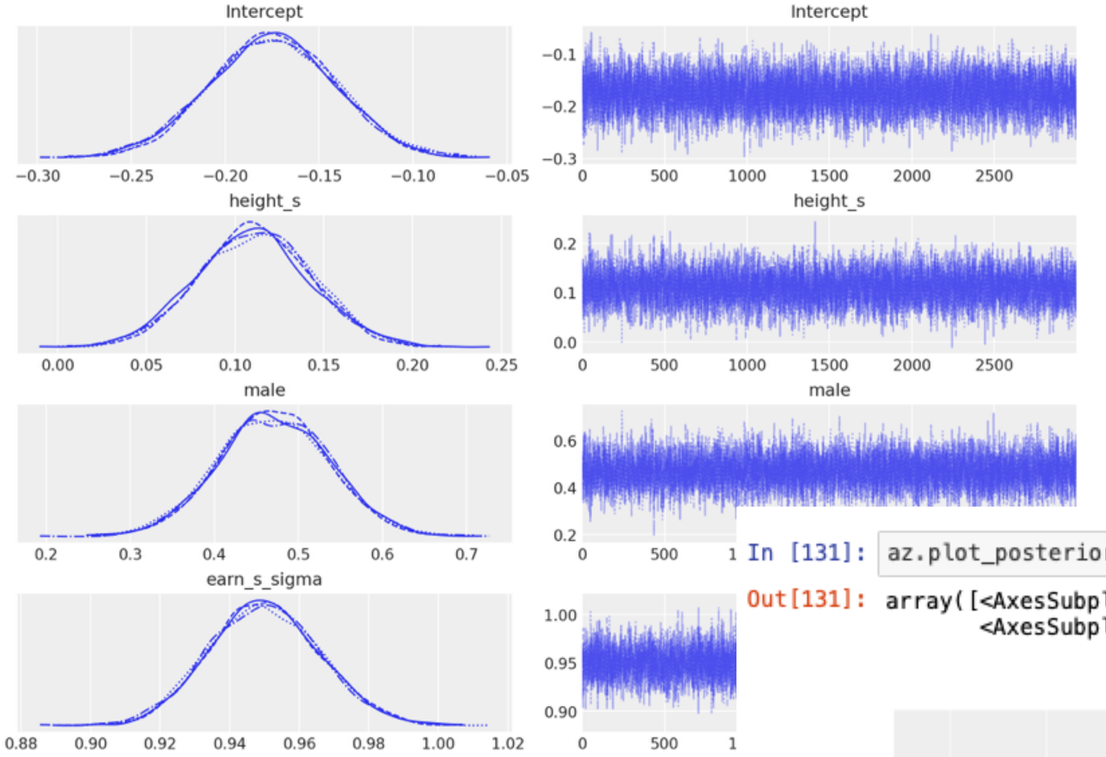




Bayesian analysis with MCMC (PYMC3)

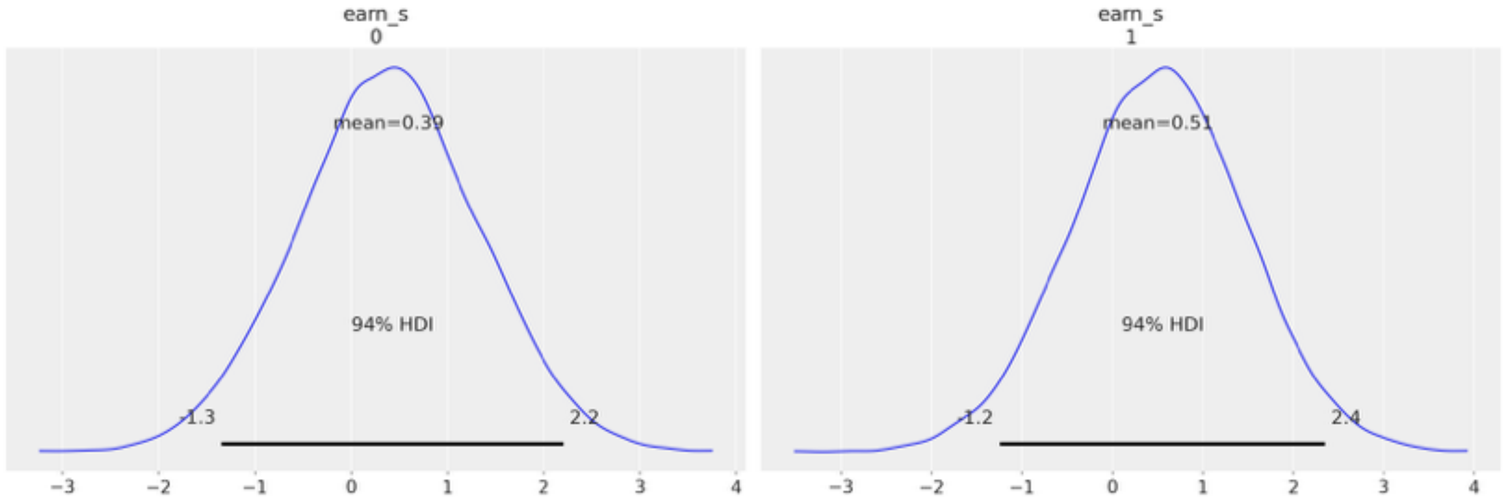
```
earnings_mod = bmb.Model("earn_s ~ height_s + male", earnings)
earnings_trace = earnings_mod.fit(draws=3000)
```

```
In [78]: az.plot_trace(earnings_trace, figsize=(10, 7));
```



Controlling for gender, the mean coefficient value can be estimated to be:

```
In [131]: az.plot_posterior(y_pred.posterior_predictive)
Out[131]: array([<AxesSubplot:title={'center':'earn_s\n0'}>,
                 <AxesSubplot:title={'center':'earn_s\n1'}>], dtype=object)
```



Student feedback

Positive:

- Liked the lab set-up with mini-videos
- Focus on “hands-on” practical applications
- Liked use of programming
- Well structured
- Forced to work throughout the semester (4 assignments)

Negative:

- Wanted more statistics earlier
- Found it hard to get help (wanted teaching assistants)
- Difficult if not working in a group
- Difficult to understand how the semester project would be evaluated
  - How do I get an A??

If I could do it all over again…

- Term project in groups (of 2)
- teaching assistants
- Be clear about learning period and test period
- More guidance on format, structure of term project
- More guidance on type of data: cross section vs. panel vs. time series