C200 Programming Assignment № 7

Dr. M.M. Dalkilic
Computer Science
School of Informatics, Computing, and Engineering

Indiana University, Bloomington, IN, USA

November 5, 2023

The HW is due on **Saturday, November, 11 at 11:00 PM EST**. Please commit, push and submit your work to the Autograder before the deadline.

1. Make sure that you are **following the instructions** in the PDF, especially the format of output returned by the functions. For example, if a function is expected to return a numerical value, then make sure that a numerical value is returned (not a list or a dictionary). Similarly, if a list is expected to be returned then return a list (not a tuple, set or dictionary).

2. Test **debug** the code well (syntax, logical and implementation errors), before submitting to the Autograder. These errors can be easily fixed by running the code in VSC and watching for unexpected behavior such as, program failing with syntax error or not returning correct output.

3. Make sure that the **code does not have infinite loop (that never exits) or an endless recursion (that never completes)** before submitting to the Autograder. You can easily check for this by running in VSC and watching for program output, if it terminates timely or not.

4. Given that you already tried points 1-3, if you see that Autograder does not do anything (after you press 'submit') and waited for a while (30 seconds to 50 seconds), try refreshing the page or using a different browser.

5. Once you are done testing your code, comment out the tests i.e. the code under the __name__ == "__main__" section.

Problem 1: SciKit Library

In the previous homework we looked at linear regression coding the solution for a univariate problem. We found that the “best" values for our model were:

$$\begin{aligned} y(x) &= \hat{m}x + \hat{b} && (1)\\ &= 0.125x + 67.5 && (2)\\ R^2 &= 0.298 && (3) \end{aligned}$$

Scikit-learn is among the most popular free machine learning library for Python. Currently it is almost always required if someone is working in any data-intensive fields like computational science, computational business. In this problem we will use their package to (1) redo our homework problem (2) do a larger problem. Please visit: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html` to read about the linear regression tools available. Make sure you pay attention to the types it uses. We will return a structure that not only includes the slope, intercept, and R^2 (as we did in the previous homework), but also the linear object itself to use other properties). The following code:

```
1     data6 = get_data("D:\\C200\\Fall 2023\\Assignments\\Assignment 6\\↩
          ", "payrollwins.txt")
2     print(f"Model built from parameters applied to 10: {(lambda x↩
          :0.1250*x + 67.498)(10)}")
3     plt.plot([x for x,_ in data6],[y for _,y in data6],'ro')
4     m_hat, b_hat, R_sq, model = my_scikit_LR(data6)
5     print(f"m_hat: {m_hat}, b_hat: {b_hat}, R^2: {R_sq}")
6     plt.plot([x for x,_ in data6],model(np.array([[x] for x,_ in data6↩
          ])),'b')
7     print(f"Scikit Model applied to 10: {model(np.array([[10]]))↩
          [0][0]}")
8     plt.xlabel("$M cost")
9     plt.ylabel("Wins")
10    plt.title(f"Wins as function of $M cost R^2 = {round(R_sq,3)}")
11    plt.show()
```

has output:

```
1   Model built from parameters applied to 10: 68.748
2 m_hat: 0.12501409085785145, b_hat: 67.4984922782099, R^2: ↩
      0.29851298993160025
3 Scikit Model applied to 10: 68.74863318678841
```

Observe that we can build the model too–this is nothing more than a λ expression. The plot is shown below.

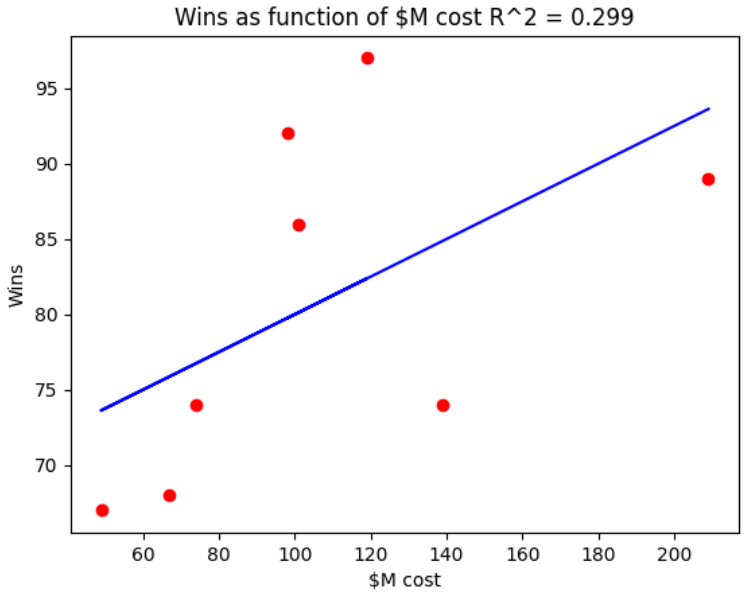

Figure 1: Using scikit-learn allows for much less code.

Deliverables

- The function takes data and returns slope, intercept, R^2, and the model. Use only this module **from sklearn.linear_model import LinearRegression**

Problem 2: Recursion to Generator

You worked with these last homework. For this homework, you'll write the tail recursion, while, and generator. This is a reminder that in the starter code, we are providing you with the regular recursion code, and the functions you must implement must utilize tail recursion, a while loop, or a generator as specified. In your starter code, function names will end in _t, _w, or _g if the function needs to be implemented using tail recursion, a while loop, or a generator, respectively. Review the lecture slides to learn more about generators.

$$
\begin{aligned}
p(0) &= 10000 && (4)\\
p(n) &= p(n-1) + 0.02p(n-1) && (5)
\end{aligned}
$$

$$
\begin{aligned}
c(1) &= 9 && (6)\\
c(n) &= 9c(n-1) + 10^{n-1} - c(n-1) && (7)
\end{aligned}
$$

$$
\begin{aligned}
d(0) &= 1 && (8)\\
d(n) &= 3d(n-1) + 1 && (9)
\end{aligned}
$$

(10)

Deliverables

- For reach function you'll write the tail recursive form, while, and generator.
- We have added the signature to help you–in particular, c(n) requres two accumulators.

Problem 3: Recursion

While equation 13 seems to be very complex, you have all the tools you need: functionality of sigma can be obtianed by using the range function, the c_2( ), function is provided in the starter code and it provide the functionality of choosing k values out of n. You are explicitly allowed (and encouraged) to use both sum( ) and list comprehension.

$$B_0 = 1 \tag{11}$$

$$B_n = \frac{-\sum_{k=0}^{n-1} \binom{n+1}{k} B_k}{n+1} \tag{12}$$

$$B(1) = \frac{-\sum_{k=0}^{0} \binom{2}{k} B_0}{1+1} \tag{13}$$

$$= -\frac{\binom{2}{0} 1}{2} \tag{14}$$

$$= -.5 \tag{15}$$

$$B(2) = -\frac{[\sum_{k=0}^{1} \binom{3}{k} B_k]}{3} \tag{16}$$

$$= -\frac{[\binom{3}{0} B_0 + \binom{3}{1} B_1]}{3} \tag{17}$$

$$= -\frac{[1(1) + 3(-.5)]}{3} \tag{18}$$

$$= -\frac{[1 - 1.5]}{3} = .5/3 = .1\bar{6}$$

$$B(3) = -\frac{[\sum_{k=0}^{2} \binom{4}{k} B_k]}{4} \tag{19}$$

$$= -\frac{[\binom{4}{0} B_0 + \binom{4}{1} B1 + \binom{4}{2} B2]}{4} \tag{20}$$

$$= -\frac{[1(1) + 4(-.5) + 6(1.\bar{6})]}{4} \tag{21}$$

$$= -\frac{[1 - 2 + 1]}{4} = 0 \tag{22}$$

Here are some outputs:

output

```
B(0) = 1
B(1) = -0.5
B(2) = 0.16666666666666666
B(3) = -0.0
B(4) = -0.033333333333333305
B(5) = -7.401486830834377e-17
```

Deliverables

- Complete the function B() using recursion.
- You can (and are encouraged to) use sum().
- List comprehension is useful too!

Problem 4: Approximating the Derivative

Calculus is used in virtually every field and is fundamental to understanding AI. In this problem we approximate the derivate which is instantaneous change. We'll focus on univariate functions initially:

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} \tag{23}$$

This is really a λ function, since we're given a function f and a value ϵ and returning a function f' that has x as an argument. We can approximate $f'(x)$ shown here:

$$f'(x) \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \tag{24}$$

We will write a function called derivative that takes a function and epsilon and returns a lambda function. for example,

$$f(x) = x^2 - 3x \tag{25}$$
$$f'(x) = 2x - 3 \tag{26}$$
$$f'(2) = 4 - 3 = 1 \tag{27}$$

```
1  def derivative(f,epsilon):
2      pass
3
4  def f(x):
5      return x**2 - 3*x
6
7  data = 2
8  epsilon = 10e-8
9  print((derivative(f,epsilon)(data)))
10 f_prime = derivative((lambda x:x**2-3*x),epsilon)
11 print(f_prime(data))
```

has outputs:

```
1 0.9999999961429751
2 0.9999999961429751
```

Deliverables

- Complete the derivate function

Problem 5: SciKit-learn on bigger data

Now that you've used scikit-learn linear regression on small data, we'll use on larger data. Let's discuss R^2 some more. R^2 is a statistical measure of how well the regression line approximates the actual data. The rule of thumb is that $R^2 \leq 30\%$ indicates that the model is very poor. On the other hand $R^2 \geq 77\%$ means the model is good. The interval describes an unclear model. In the next homework we'll use a more recent extension of linear regression that takes into accout the distance x has from the model (the residual to remind you). The intuition is that the smaller residuals would inversely be weighted for x, since the farther away a value is, the less contribution it should have to building the model. When complete the plot you'll see is shown in Fig. 2. Note that R^2 is much better than the last model. You'll reusue your scikit function exactly as it is, but send it different data–to this end, observe that the code to get the data is different. The following code (which only differs with file):

```
1     path,name = "", "income_data.csv"
2     data4 = get_data_2(path,name)
3     plt.plot([x for x,_ in data4],[y for _,y in data4],'ro')
4     m_hat, b_hat, R_sq, model = my_scikit_LR(data4)
5     print(f"m_hat: {m_hat}, b_hat: {b_hat}, R^2: {R_sq}")
6     plt.plot([x for x,_ in data4],model(np.array([[x] for x,_ in data4
          ])),'b')
7     plt.xlabel("$M cost")
8     plt.ylabel("Wins")
9     plt.title(f"Wins as function of $M cost R^2 = {round(R_sq,3)}")
10    plt.show()
```

generates the plot and outputs:

```
1 m_hat: 0.7138255122719709, b_hat: 0.20427039623160193, R^2:
      0.7493217544889172
```

Hint: You can also revisit the lab7 on file reading to understand how to read the files. Again, if you want to use csv_reader then you won't be penalized.

Figure 2: Plots of happiness as a function of income and the model (blue). There are 498 data points.

Deliverables

- Complete the get_data_2 by changing the data.
- The data is in a file named income_data.csv and it contains scores for happiness (column name: happiness) as a function of income (column name: income).
- Keep in mind that after you have tested your code, you must comment out the 'import matplotlib' and the plotting code given under the __main__ before submitting to the Autograder. Autograder can not draw graphical plots on the web browser so it will return an error if you do not comment out the plotting code/import matplotlib.

Problem 6: Candlesticks

Given a sequence of values $[v_0, v_1, \ldots, v_n]$ where $0, 1, \ldots, n$ represents time, a candlestick is the most popular visualization that is used to both summarize and inform financial analysts. The colors, red and green, indicate whether the valuation has negatively or positively changed, respectively. The line gives the range of valuations. The rectangle's height indicates the starting and ending value for that time period.

For a given sequence, a box and lines are drawn as shown in Fig. 3 (right)

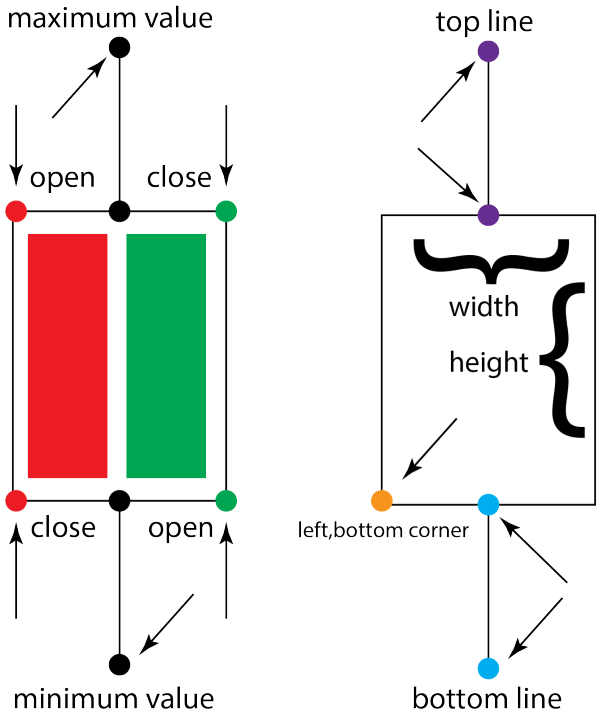

Figure 3: A candlestick image (right). The box is created by the opening and closing values. The top line is from the maximal value to the top middle of the box. The bottom line is from the minimal value to the bottom middle of the box. The color is determined by the open, close values. If open is greater than close, then the color is red; otherwise it's green. We use matplotlib patches. The candlestick image is determined the rectangle and two lines. The rectangle is determined by the lower-left point (x, y), width, and height. The two lines are determined by two points computed from max, min, open, and close.

For any sequence we can quickly get the four values we need:

```
1 >>> import random as rn
2 >>> prices = [rn.randint(2,50) for _ in range(200)]
3 >>> open,close,max_p,min_p = prices[0],prices[len(prices)-1],max(↩
    prices),min(prices)
4 >>> open,close,max_p,min_p
5 (37, 2, 50, 2)
```

Using matplotlib's patches `https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.`

`Rectangle.html` we can translate the four values into a candlestick, Fig. 3. (left). To plot a rectangle, we'll need the lower-lefthand corner (x,y), width, and height.

Note: Note that the input is a collection of values contained inside a list so each sublist represents a candelstick. You will iterate over the contents of the list one-by-one and process each sublist insde the make() function to plot the candlesticks. The plotting is taken care of by the matplotlib patches but you still need to finish the make() function to return the values that are needed by the matplotlib patches. For quickly understanding how it works, try following the starter code under "__main__".

Hint: As each sublist is passed inside the make() function, you will first find the correct values of open, close, max_p and min_p (from the sublist), and then use those to calculate the values required to plot the candlestick. How to find, open, close, max_p and min_p is shown in the code listing above. The open and close will be used as the y values (i.e. the values on the y axis) and you have to find them from the sublist.

Let's look at the code:

```
1  data = [[20,15,32,10],[10,14,15,9],
2          [22,23,27,9],[15,16,16,15],
3          [26,12,30,2],[5,30,40,4]]
4
5  # open,close,max_p,min_p = 20, 15, 32, 10
6
7  # INPUTS ith candle, starting value of x, default width, and the four critical values: open, close, max\_p, min\_p.
8  # RETURN three tuples: (point, width, height, color), topline, bottomline
9  # First tuple contains: point is the coordinates of the lower-left point x, y, width and height are numeric values and color will be a string of color ex. 'red' or 'green'
10 # second tuple is: topline ((xt0,yt0),(xt1,yt1)) coordinates of line from max to top middle of box
11 # thord tuple is: bottomline ((xb0,yb0),(xb1,yb1)) coordinates of line from min to bottom middle of box
12
13 # When you see the code for testing under __main__, you will see that the first three values of the first tuple i.e.,
14 # point, width and height are passed as the first arguement of matplotlib.patches.Rectangle() function and the
15 # last value i.e. color is passed as the second arguement. Feel free to play around with the test code to
16 # get a feeling of how it is working. You will understand it much better with a bit of experimentation.
```

```
17
18 def make(i, start, width_default, d):
19     pass
20
21 fig = plt.figure()
22 ax = fig.add_subplot(111)
23 start = 0
24 default_width = 10
25 for i in range(len(data)):
26
27     candle_box, top_line, bottom_line = make(i,start,default_width, ↩
          data[i])
28     print(candle_box)
29     ax.add_patch(matplotlib.patches.Rectangle(*candle_box[0:3],color =↩
           candle_box[3]))
30     plt.plot([x for x,_ in top_line],[y for _,y in top_line],'black')
31     plt.plot([x for x,_ in bottom_line],[y for _,y in bottom_line],'↩
          black')
32     start += default_width
33
34 plt.xlabel("time (hour)")
35 plt.ylabel("Stock X price")
36 plt.title("Candlestick for Stock X mm/dd/yyyy")
37 plt.xlim([0, 60])
38 plt.ylim([0, 35])
39 plt.show()
```

will produce

```
1 ((0, 15), 10, 5, 'red')
2 ((10, 10), 10, 4, 'green')
3 ((20, 22), 10, 1, 'green')
4 ((30, 15), 10, 1, 'green')
5 ((40, 12), 10, 14, 'red')
6 ((50, 5), 10, 25, 'green')
```

and the plot.

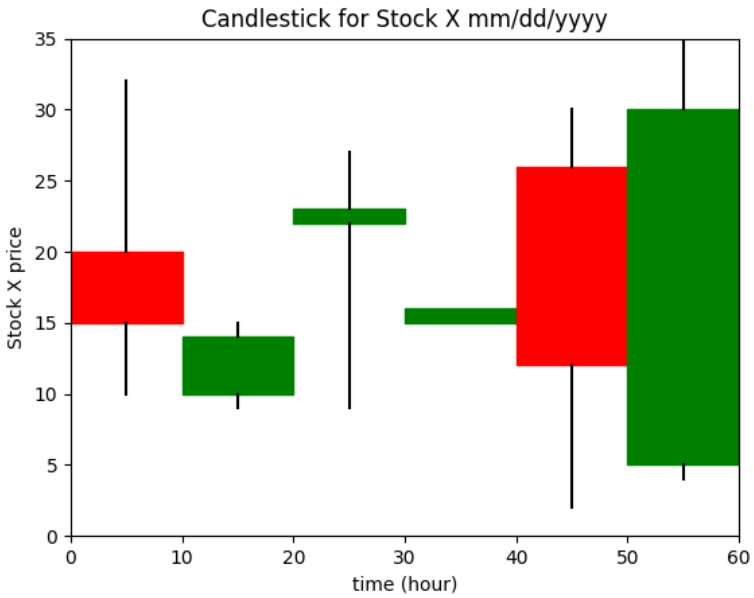

Figure 4: Six candlesticks.

Deliverables

- Complete the function make() that returns three things: the data for the rectangle, the top line, and the bottom line.
- **Hint:** If you see the output produced by printing candel_box, notice how the value of start changes for each candle_box-this is what makes each candlebox shift in the plot so that each candelbox start a specific distance after the previous one so that all of them looks seperated for easy visualization.
- Keep in mind that after you have tested your code, you must comment out the 'import matplotlib' and the plotting code given under the __main__ before submitting to the Autograder. Autograder can not draw graphical plots on the web browser so it will return an error if you do not comment out the plotting code/import matplotlib.

Programming partners

hamac@iu.edu, schwajaw@iu.edu
adiyer@iu.edu, mz24@iu.edu
ayoajayi@iu.edu, chrimanu@iu.edu
dja1@iu.edu, isaramir@iu.edu
dkkosim@iu.edu, dsummit@iu.edu
clearle@iu.edu, tzuyyen@iu.edu
nihanas@iu.edu, tpandey@iu.edu
blacount@iu.edu, tangtom@iu.edu
jhhudgin@iu.edu, jwember@iu.edu
coopjose@iu.edu, lvansyck@iu.edu
alscarr@iu.edu, pravulap@iu.edu
tcconnol@iu.edu, emgward@iu.edu
gmhowell@iu.edu, wwtang@iu.edu
joehawl@iu.edu, gszopin@iu.edu
arnadutt@iu.edu, utwade@iu.edu
ovadeley@iu.edu, zshamo@iu.edu
zacbutle@iu.edu, aveluru@iu.edu
johnguen@iu.edu, cnyarko@iu.edu
spgreenf@iu.edu, mzagotta@iu.edu
aroraarn@iu.edu, cjwaller@iu.edu
flynncj@iu.edu, jdw14@iu.edu
delkumar@iu.edu, rwan@iu.edu
jtbland@iu.edu, voram@iu.edu
spdamani@iu.edu, sahimann@iu.edu
krisgupt@iu.edu, ajtse@iu.edu
nmcastan@iu.edu, evataylo@iu.edu
cpkerns@iu.edu, leolin@iu.edu
maudomin@iu.edu, rosenbbj@iu.edu
brownset@iu.edu, muyusuf@iu.edu
howardbw@iu.edu, samrile@iu.edu
sydecook@iu.edu, dwo@iu.edu
vkommar@iu.edu, abiparri@iu.edu
davgourl@iu.edu, patedev@iu.edu
daminteh@iu.edu, kvpriede@iu.edu
khannni@iu.edu, kamaharj@iu.edu
masharre@iu.edu, cstancom@iu.edu
laharden@iu.edu, skp2@iu.edu
adhuria@iu.edu, vyeruba@iu.edu
arklonow@iu.edu, mveltri@iu.edu

wilcusic@iu.edu, ism1@iu.edu
grafe@iu.edu, aselki@iu.edu
jhar@iu.edu, jarlmint@iu.edu
kapgupta@iu.edu, megapaul@iu.edu
wanjiang@iu.edu, btasa@iu.edu
stkimani@iu.edu, antando@iu.edu
gcopus@iu.edu, wtatoole@iu.edu
keswar@iu.edu, jnzheng@iu.edu
ruqchen@iu.edu, asultano@iu.edu
jabbarke@iu.edu, nichojop@iu.edu
apbabu@iu.edu, giomayo@iu.edu
fdonfrio@iu.edu, nmr1@iu.edu
ohostet@iu.edu, gmpierce@iu.edu
hawkjod@iu.edu, nniranj@iu.edu
mrfehr@iu.edu, aditpate@iu.edu
oakinsey@iu.edu, deturne@iu.edu
mdiazrey@iu.edu, clmcevil@iu.edu
abellah@iu.edu, smremmer@iu.edu
jwdrew@iu.edu, aranjit@iu.edu
marganey@iu.edu, jomeaghe@iu.edu
hk120@iu.edu, liwitte@iu.edu
kjj6@iu.edu, lmeldgin@iu.edu
ceub@iu.edu, rafir@iu.edu
escolber@iu.edu, jsadiq@iu.edu
hermbrar@iu.edu, gavilleg@iu.edu
ckdiallo@iu.edu, fshamrin@iu.edu
skunduru@iu.edu, rtrammel@iu.edu
jkielcz@iu.edu, cmarcuka@iu.edu
austdeck@iu.edu, nsatti@iu.edu
ejhaas@iu.edu, asaokho@iu.edu
garcied@iu.edu, patekek@iu.edu
bencho@iu.edu, majtorm@iu.edu
josespos@iu.edu, sahishah@iu.edu
wgurley@iu.edu, benprohm@iu.edu
aberkun@iu.edu, vrradia@iu.edu
kekchoe@iu.edu, ysanghi@iu.edu
colrkram@iu.edu, ijvelmur@iu.edu
kraus@iu.edu, vmungara@iu.edu
efritch@iu.edu, ragmahaj@iu.edu
tfreson@iu.edu, ksadiq@iu.edu
eakanle@iu.edu, lqadan@iu.edu

migriswo@iu.edu, nrizvi@iu.edu
jakchap@iu.edu, mmarotti@iu.edu
jdemirci@iu.edu, jaslnu@iu.edu
nfarhat@iu.edu, sezinnkr@iu.edu
fkanmogn@iu.edu, anajmal@iu.edu
quecox@iu.edu, lmadiraj@iu.edu
swconley@iu.edu, epautsch@iu.edu
saecohen@iu.edu, justyou@iu.edu
nharkins@iu.edu, etprince@iu.edu
makinap@iu.edu, mehtriya@iu.edu
twfine@iu.edu, gs29@iu.edu
micahand@iu.edu, thnewm@iu.edu
saganna@iu.edu, scotbray@iu.edu
ahavlin@iu.edu, ntorpoco@iu.edu
zguising@iu.edu, erschaef@iu.edu
cuizek@iu.edu, jlzhao@iu.edu
mrcoons@iu.edu, awsaunde@iu.edu
cgkabedi@iu.edu, myeralli@iu.edu
bellcol@iu.edu, vpolu@iu.edu
kdembla@iu.edu, wtrucker@iu.edu
mkames@iu.edu, jwu6@iu.edu
mwclawso@iu.edu, ansakrah@iu.edu
ryanbren@iu.edu, apavlako@iu.edu
coopelki@iu.edu, pricemo@iu.edu
sakalwa@iu.edu, qshamsid@iu.edu
caegrah@iu.edu, ammulc@iu.edu
hh35@iu.edu, thomps16@iu.edu
skatiyar@iu.edu, hasiddiq@iu.edu
ethickma@iu.edu, bdyiga@iu.edu
milhavil@iu.edu, wardjohn@iu.edu
aketcha@iu.edu, tarturnm@iu.edu
aakindel@iu.edu, asidda@iu.edu
sfuneno@iu.edu, pw18@iu.edu
jacklapp@iu.edu, arirowe@iu.edu
aaragga@iu.edu, emisimps@iu.edu
agrevel@iu.edu, drsnid@iu.edu
diebarro@iu.edu, reddyrr@iu.edu
anrkram@iu.edu, pp31@iu.edu
phjhess@iu.edu, fmahamat@iu.edu
kaihara@iu.edu, joshroc@iu.edu
leegain@iu.edu, nlippman@iu.edu

adwadash@iu.edu, patel89@iu.edu
tchapell@iu.edu, jcn1@iu.edu
sg40@iu.edu, orrostew@iu.edu
achordi@iu.edu, jwetherb@iu.edu
loggreen@iu.edu, reedkier@iu.edu
jwcase@iu.edu, blswing@iu.edu
gandhira@iu.edu, woodsky@iu.edu
nolakim@iu.edu, gavsteve@iu.edu
anlego@iu.edu, krbpatel@iu.edu
alelefeb@iu.edu, cialugo@iu.edu
bkante@iu.edu, cmvanhov@iu.edu
ajeeju@iu.edu, sasayini@iu.edu
rl29@iu.edu, mszczas@iu.edu
ddrotts@iu.edu, sahaan@iu.edu
aaamoako@iu.edu, asteini@iu.edu
earuland@iu.edu, impofujr@iu.edu
nbernot@iu.edu, gsilingh@iu.edu
laburkle@iu.edu, crmoll@iu.edu
jacobben@iu.edu, jneblett@iu.edu
nfelici@iu.edu, lukastef@iu.edu
bcdutka@iu.edu, rnschroe@iu.edu
mbrockey@iu.edu, bcmarret@iu.edu
egoldsto@iu.edu, dyashwar@iu.edu
althart@iu.edu, savebhat@iu.edu
mbeigie@iu.edu, mmunaf@iu.edu
jdgonzal@iu.edu, madymcsh@iu.edu
spgerst@iu.edu, perezand@iu.edu
cannan@iu.edu, jactrayl@iu.edu
ag69@iu.edu, ir1@iu.edu
fu7@iu.edu, wtubbs@iu.edu
avulas@iu.edu, snyderjk@iu.edu
fkeele@iu.edu, audtravi@iu.edu
deombeas@iu.edu, dernguye@iu.edu
evacoll@iu.edu, ruska@iu.edu
lpfritsc@iu.edu, amanocha@iu.edu
mdonato@iu.edu, ntuhl@iu.edu
dce@iu.edu, iperine@iu.edu
mohiambu@iu.edu, coenthom@iu.edu
huhasan@iu.edu, aptheria@iu.edu
apchavis@iu.edu, linjaso@iu.edu
howamatt@iu.edu, lpelaez@iu.edu

alchatz@iu.edu, rvinzant@iu.edu
brhint@iu.edu, jpochyly@iu.edu
simadams@iu.edu, rorymurp@iu.edu
maxklei@iu.edu, tolatinw@iu.edu
maladwa@iu.edu, jtsuter@iu.edu
sgaladim@iu.edu, masmatth@iu.edu
ajgrego@iu.edu, maklsmit@iu.edu
lflenoy@iu.edu, aamathew@iu.edu
bencalex@iu.edu, ltmckinn@iu.edu
edfran@iu.edu, rpoludas@iu.edu
ek37@iu.edu, surapapp@iu.edu
ethbrock@iu.edu, jwmullis@iu.edu
greenpat@iu.edu, rt11@iu.edu
matgarey@iu.edu, pateishi@iu.edu
lcoveney@iu.edu, ryarram@iu.edu
nokebark@iu.edu, aidnschi@iu.edu
liansia@iu.edu, ao9@iu.edu
daxbills@iu.edu, emluplet@iu.edu
cfampo@iu.edu, adsize@iu.edu
tychid@iu.edu, anemlunc@iu.edu
mkleinke@iu.edu, aledminc@iu.edu
mdoxsee@iu.edu, dukthang@iu.edu
allencla@iu.edu, mnimmala@iu.edu
schinitz@iu.edu, gepearcy@iu.edu
dblackme@iu.edu, wlyzun@iu.edu
ameydesh@iu.edu, clscheum@iu.edu
amkhatri@iu.edu, avraya@iu.edu
agawrys@iu.edu, annaum@iu.edu
rcaswel@iu.edu, wilsori@iu.edu
ridbhan@iu.edu, cltran@iu.edu
kaneai@iu.edu, owysmit@iu.edu
lawmat@iu.edu, keasandl@iu.edu
leokurtz@iu.edu, bmpool@iu.edu
bjdahl@iu.edu, ap79@iu.edu
jdc6@iu.edu, mjroelle@iu.edu
oeichenb@iu.edu, patelsak@iu.edu
seangarc@iu.edu, brayrump@iu.edu
jc168@iu.edu, apathma@iu.edu