lec04

August 9, 2021

[1]:
```
from datascience import *
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

0.1 Review of Tables

[2]:
```
nba = Table.read_table('nba_salaries.csv').relabeled(3, 'SALARY')
nba
```

[2]:
```
PLAYER            | POSITION | TEAM          | SALARY
Paul Millsap      | PF       | Atlanta Hawks | 18.6717
Al Horford        | C        | Atlanta Hawks | 12
Tiago Splitter    | C        | Atlanta Hawks | 9.75625
Jeff Teague       | PG       | Atlanta Hawks | 8
Kyle Korver       | SG       | Atlanta Hawks | 5.74648
Thabo Sefolosha   | SF       | Atlanta Hawks | 4
Mike Scott        | PF       | Atlanta Hawks | 3.33333
Kent Bazemore     | SF       | Atlanta Hawks | 2
Dennis Schroder   | PG       | Atlanta Hawks | 1.7634
Tim Hardaway Jr.  | SG       | Atlanta Hawks | 1.30452
… (407 rows omitted)
```

[3]:
```
point_guards = nba.where('POSITION', 'PG')
```

[4]:
```
point_guards
```

[4]:
```
PLAYER           | POSITION | TEAM              | SALARY
Jeff Teague      | PG       | Atlanta Hawks     | 8
Dennis Schroder  | PG       | Atlanta Hawks     | 1.7634
Avery Bradley    | PG       | Boston Celtics    | 7.73034
Isaiah Thomas    | PG       | Boston Celtics    | 6.91287
Marcus Smart     | PG       | Boston Celtics    | 3.43104
Terry Rozier     | PG       | Boston Celtics    | 1.82436
Jarrett Jack     | PG       | Brooklyn Nets     | 6.3
```

```
Shane Larkin    | PG       | Brooklyn Nets      | 1.5
Kemba Walker    | PG       | Charlotte Hornets  | 12
Brian Roberts   | PG       | Charlotte Hornets  | 2.85494
… (75 rows omitted)
```

```
[5]: point_guards.drop('POSITION')
```

```
[5]: PLAYER          | TEAM              | SALARY
     Jeff Teague     | Atlanta Hawks     | 8
     Dennis Schroder | Atlanta Hawks     | 1.7634
     Avery Bradley   | Boston Celtics    | 7.73034
     Isaiah Thomas   | Boston Celtics    | 6.91287
     Marcus Smart    | Boston Celtics    | 3.43104
     Terry Rozier    | Boston Celtics    | 1.82436
     Jarrett Jack    | Brooklyn Nets     | 6.3
     Shane Larkin    | Brooklyn Nets     | 1.5
     Kemba Walker    | Charlotte Hornets | 12
     Brian Roberts   | Charlotte Hornets | 2.85494
     … (75 rows omitted)
```

```
[6]: point_guards
```

```
[6]: PLAYER          | POSITION | TEAM              | SALARY
     Jeff Teague     | PG       | Atlanta Hawks     | 8
     Dennis Schroder | PG       | Atlanta Hawks     | 1.7634
     Avery Bradley   | PG       | Boston Celtics    | 7.73034
     Isaiah Thomas   | PG       | Boston Celtics    | 6.91287
     Marcus Smart    | PG       | Boston Celtics    | 3.43104
     Terry Rozier    | PG       | Boston Celtics    | 1.82436
     Jarrett Jack    | PG       | Brooklyn Nets     | 6.3
     Shane Larkin    | PG       | Brooklyn Nets     | 1.5
     Kemba Walker    | PG       | Charlotte Hornets | 12
     Brian Roberts   | PG       | Charlotte Hornets | 2.85494
     … (75 rows omitted)
```

```
[7]: point_guards = point_guards.drop('POSITION')
```

```
[8]: point_guards.sort('SALARY', descending=True).show()
```

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

```
[9]: point_guards.sort('SALARY', descending=True).show(15)
```

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

```
[10]: nba.drop('POSITION').where('POSITION', 'PG')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-774884f2a906> in <module>
----> 1 nba.drop('POSITION').where('POSITION', 'PG')

~\Anaconda3\lib\site-packages\datascience\tables.py in where(self,␣
 ↪column_or_label, value_or_predicate, other)
   1334         Color | Shape | Amount | Price
   1335         """
-> 1336         column = self._get_column(column_or_label)
   1337         if other is not None:
   1338             assert callable(value_or_predicate), "Predicate required fo␣
 ↪3-arg where"

~\Anaconda3\lib\site-packages\datascience\tables.py in _get_column(self,␣
 ↪column_or_label)
   1995             return self[c]
   1996         elif isinstance(c, str):
-> 1997             raise ValueError('label "{}" not in labels {}'.format(c,␣
 ↪self.labels))
   1998         else:
   1999             assert len(c) == self.num_rows, 'column length mismatch'

ValueError: label "POSITION" not in labels ('PLAYER', 'TEAM', 'SALARY')
```

0.2 Numbers

[11]:
```
30
```

[11]: 30

[12]:
```
10 * 3     # int
```

[12]: 30

[13]:
```
10 / 3     # float
```

[13]: 3.3333333333333335

[14]:
```
10 / 2
```

[14]: 5.0

[15]:
```
10 ** 3
```

[15]: 1000

```
[16]: 10 ** 0.5
```

```
[16]: 3.1622776601683795
```

```
[17]: 1234567 ** 890
```

```
[17]: 2805763342107369230793809808310885724030151698643898788380662816980113897853769
      50568255334199536184086628316383967493806384746509818159941351603092383031502029
      74291144848424728060376016331831773923424195522814460988823331133604989360606894
      17981060580169581781517081000887466986920196787365735472976637384790660527656183
      52910805767762881289789891741572746740387325169710028431811361317109083277010762
      73734292975781004525413311379723943955188797482524976299531691897425466887438380
      14842998797222488578482742255590278830326264415887042424349338336685586108014796
      84364729785468663205961101779259311413364376617682664547150513528433960828644079
      01086051146872614462141067032383046267474303202831705056235935981924821832533232
      27473361217459497593514576103139354263910235117552944853835259328240635891089203
      24750139530179424413258418687475579783185239070441745704404855756861863316028414
      95833403526865120096119757842398592643368783923874827910491569128541597155162147
      06166214466230037339823780492774337274675800289580696315819494418004733412855205
      15502791805834489185066847378078752053985111060193717777725852763547625805234282
      71797651863818349735223031368298528342426795598883719905465477737351428726113831
      02742753311234958929602179017609453450788021576889968090558526946002867067155433
      93702516460327849632506473208065508596712055212235864342590633941946808878581257
      66411703997815551985185422287557640908125147881637588772280314746262499823201732
      30513790766211051807071509018394299594496298553425032888639575233308427922962072
      69111001641083261325484555853347977913739454653334826726020865414766466166248983
      77221664655359844703001301953615267246579992477376641195451663679701189605396537
      00728246780841243422865927822029745430687779341494835511048331091708435435810632
      67728496886767121735060993255553142427642845484544668114879980414445112719539667
      03737331555036884407663761173081452313392045145181141824426697269269584602635047
      47823171357759585770665369994677944653246927717262087721616975390825217507408220
      51086101863433813346406041829504793392763586862619992578846319138008093570144384
      73465327468383494498332041837195204252735084614273632688109018164392075206024557
      92069406367821658839121277649227555623273564677301331404339603368354187279170542
      95408801709905655227723031806238321654205692742222699981120545054948558924623781
      98748696389173035416447143560816024686165121440278086216699179675232988477332809
      98188849126660410144516840751121177122218727312917235905337333060556516611924333
      66189353004006704250537929412543640118435225677556546314270563730678387188231379
      47288314275295831508461494270048023220877883579689922644386929930098098838754655
      18646172470495839410492599060455716367637440364630131471750106709132465895371670
      41294525607382269790351052221176569729592205359322927641922653133140831789693683
      06558158736663551973053787089646682098992913447054442763335787193239041709520743
      28643811404544256843025709425474387814973960542018630128201161272329024792011740
      74553021257987816137896112791803809664802616202980887874786621089443254300202943
      45656131323550437773857638312380609523634497978717616225417774536537840814567199
      40419766751192867804269268505636574832199689538986715250742630200523490339094085
      42517856869890326832022097356749028842091926101848216853817191000071199207687368
      1
```

```
129307598675285503345960161016391830728260326693428202177160119543736800344297777
148107962707698803197597857589564765422155862009299188468709186261360572214427147
9073816319065050855876118684499471715240326955787858118407527593640893925601764
6715559306260196259010170427037361073267573147026510628695621128132172515464076
41874294010311396461168169940795096777441342818435524904133471476582748067589980
02355377733854432296220672335233542132614124033750309325768866544993178081720359
10623439488923497729990034132955991616175315301110003136655552464696428279778365
47740616430752501111978305504766486479881842630760417238107369579868837788919341
54236450421889273981458644458716804316177728235846587077050574261327976336012814
78887542871708884776132993498113629791075212269131263135104211901928478309667400
64247714520136243336565857870726701812238890685310326479783755031490682632796458
98115378001513480782786895264922665416001681523341740875519925092229756580509021
51573849522445074588731109036554789680280345720046009120507928152632018078697710
12021836814757059914103167330068234832615649724401157481780218709983488032734621
06245563696600186788599292716541085969511934794530344521147798478505042805254903
03492730731848704497321639579545359051407002629069912400518470707512308477494989
00639024935127356145395776710374910509948685944182349912010335997456209536187821
99937853126821082043491150814038548739686936511448169920087870944136545418407502
67945428940655147102103776877924495354589674471203663800973702240269677784610346
01044182426181776359259987054659498712880202088987862369480024757525166143927615
70088475295292371921625774633565036173231327600796886277064445104257794418712970
35026396768851129501722454220316418777074912838238391162658683371634959415544181
38405815425196076721710163213925831215090477762315678893730971354056930676523216
56716189358680398162921102216575604937429088048679526182005150055393169607240885
51539133626302522557345413249384658492371698805093289004717340532810196692880918
81686283240594192082830256846191390408870214765471387805621811701183909476991816
8741678648022940613796829566893389100562532347981860917540264049
```

[18]: `10 / 3`

```
[18]: 3.3333333333333335
```

[19]: `75892745.2154892475892749857 12`

```
[19]: 75892745.21548925
```

```
[23]: int(10 / 4)

[23]: 2

[24]: float(3)

[24]: 3.0

[25]: 6 / 4

[25]: 1.5

[26]: 6 / 4000

[26]: 0.0015

[27]: 6 / 400000000000000000000000000000000000000000000000000000000

[27]: 1.5e-56

[28]: 400000000000000000000000000000000000000000000000000000000 * 1.5e-56

[28]: 6.0

[29]: 1.5e-56

[29]: 1.5e-56

[30]: x = 5

[31]: 2x

  File "<ipython-input-31-6d17da6697ca>", line 1
    2x
     ^
SyntaxError: invalid syntax

[32]: 2 * x

[32]: 10

[33]: round(3.7)

[33]: 4
```

[34]:
```
round(2.00000052345324, 2)
```

[34]:
```
2.0
```

[35]:
```
10 * 3.0
```

[35]:
```
30.0
```

0.3 Strings

[36]:
```
'Flavor'
```

[36]:
```
'Flavor'
```

[37]:
```
Flavor
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-c281a38ccebf> in <module>
----> 1 Flavor

NameError: name 'Flavor' is not defined
```

[38]:
```
"Flavor"
```

[38]:
```
'Flavor'
```

[39]:
```
"Don't always use single quotes"
```

[39]:
```
"Don't always use single quotes"
```

[40]:
```
'Don't always use single quotes'
```

```
  File "<ipython-input-40-8490ca019922>", line 1
    'Don't always use single quotes'
         ^
SyntaxError: invalid syntax
```

[41]:
```
'straw' + 'berry' # concatenation
```

[41]:
```
'strawberry'
```

[42]:
```
'Chirs' + 'Paul' # spaces aren't added for you
```

[42]:

```
'ChirsPaul'
```

[43]:

```
'Chris' + ' ' + 'Paul'
```

[43]:

```
'Chris Paul'
```

[44]:

```
'ha' * 100
```

[44]:

```
'hahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahah
ahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahahah
ahahahahahahahahahahahahahahahahahahahaha'
```

[45]:

```
'lo' * 5.5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-45-b21ea21e1530> in <module>
----> 1 'lo' * 5.5

TypeError: can't multiply sequence by non-int of type 'float'
```

[46]:

```
'ha' + 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-46-e09ba789268d> in <module>
----> 1 'ha' + 10

TypeError: can only concatenate str (not "int") to str
```

[47]:

```
int('3')
```

[47]:

```
3
```

[48]:

```
int('3.0')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-90ae876cd031> in <module>
----> 1 int('3.0')

ValueError: invalid literal for int() with base 10: '3.0'
```

[49]:

```
float('3.0')
```

```
[49]: 3.0
```

```
[50]: str(3)
```

```
[50]: '3'
```

```
[51]: str(4.5)
```

```
[51]: '4.5'
```

0.4 Types

```
[52]: type(10)
```

```
[52]: int
```

```
[53]: a = 10
```

```
[54]: type(a)
```

```
[54]: int
```

```
[55]: type(4.5)
```

```
[55]: float
```

```
[56]: type('abc')
```

```
[56]: str
```

```
[57]: type(nba)
```

```
[57]: datascience.tables.Table
```

```
[58]: type(True)
```

```
[58]: bool
```

```
[59]: type(abs)
```

```
[59]: builtin_function_or_method
```

0.5 Arrays

```
[60]: my_array = make_array(1, 2, 3, 4)
```

[61]:
```
my_array
```

[61]:
```
array([1, 2, 3, 4], dtype=int64)
```

[62]:
```
my_array * 2
```

[62]:
```
array([2, 4, 6, 8], dtype=int64)
```

[63]:
```
my_array ** 2
```

[63]:
```
array([ 1,  4,  9, 16], dtype=int64)
```

[64]:
```
my_array + 1
```

[64]:
```
array([2, 3, 4, 5], dtype=int64)
```

[65]:
```
my_array # array is unchanged, just like when we call show/select/drop on Table
```

[65]:
```
array([1, 2, 3, 4], dtype=int64)
```

[66]:
```
another = make_array(5, 6, 7, 8)
```

[67]:
```
my_array + another
```

[67]:
```
array([ 6,  8, 10, 12], dtype=int64)
```

[68]:
```
yet_another = make_array(5, 6, 7)
```

[69]:
```
my_array + yet_another
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-69-a4a5e45ad569> in <module>
----> 1 my_array + yet_another

ValueError: operands could not be broadcast together with shapes (4,) (3,)
```

[70]:
```
str_array = make_array('ha', 'he', 'ho')
```

[71]:
```
str_array * 4
```

```
---------------------------------------------------------------------------
UFuncTypeError                            Traceback (most recent call last)
<ipython-input-71-1f62457f1a5c> in <module>
----> 1 str_array * 4
```

```
UFuncTypeError: ufunc 'multiply' did not contain a loop with signature matching
 ↪types (dtype('<U3'), dtype('<U3')) -> dtype('<U3')
```

```
[72]: sum(my_array)
```

```
[72]: 10
```

```
[73]: np.average(my_array)
```

```
[73]: 2.5
```

```
[74]: len(my_array)
```

```
[74]: 4
```

0.6 Columns of Tables are Arrays

```
[75]: warriors = nba.where('TEAM', 'Golden State Warriors')
```

```
[76]: warriors
```

```
[76]: PLAYER             | POSITION | TEAM                  | SALARY
      Klay Thompson      | SG       | Golden State Warriors | 15.501
      Draymond Green     | PF       | Golden State Warriors | 14.2609
      Andrew Bogut       | C        | Golden State Warriors | 13.8
      Andre Iguodala     | SF       | Golden State Warriors | 11.7105
      Stephen Curry      | PG       | Golden State Warriors | 11.3708
      Jason Thompson     | PF       | Golden State Warriors | 7.00847
      Shaun Livingston   | PG       | Golden State Warriors | 5.54373
      Harrison Barnes    | SF       | Golden State Warriors | 3.8734
      Marreese Speights  | C        | Golden State Warriors | 3.815
      Leandro Barbosa    | SG       | Golden State Warriors | 2.5
      … (4 rows omitted)
```

```
[77]: warriors.select('SALARY')
```

```
[77]: SALARY
      15.501
      14.2609
      13.8
      11.7105
      11.3708
      7.00847
      5.54373
      3.8734
```

```
        3.815
        2.5
        … (4 rows omitted)
```

```
[78]: warriors.column('SALARY')
```

```
[78]: array([15.501   , 14.26087 , 13.8     , 11.710456, 11.370786,  7.008475,
              5.543725,  3.873398,  3.815   ,  2.5     ,  2.008748,  1.270964,
              1.13196 ,  0.289755])
```

```
[79]: np.average(warriors.column('SALARY'))
```

```
[79]: 6.72036692857143
```

```
[80]: raptors = nba.where('TEAM', 'Toronto Raptors')
```

```
[81]: np.average(warriors.column('SALARY')) - np.average(raptors.column('SALARY'))
```

```
[81]: 2.3278598697479005
```

```
[ ]:
```