Problem Solving and Abstraction (CMPU 101)

Tom Ellman

Lecture 4

Loading Tools for Working with Tables

```
use context essentials2021
#Load tools for working with tables.
include shared-gdrive(
  "dcic-2021",
  "1wyQZj_L0qqV9Ekgr9au6RX2iqt2Ga8Ep")
```

We've Seen Tables Before

lbd	ubd	rate
0	20,000	0
20,001	50,000	0.1
50,001	100,000	0.3

at-or-after	and-before	greeting
0	6	"Working Late?"
6	12	"Good Morning!"
12	18	"Good Afternoon!"
18	24	"Good Evening!"

Making a Table Manually

```
mt-rates = table: lbd, ubd, rate
  row: 0,     20,    0.0
  row: 20001, 50000, 0.1
  row: 50001, 10000, 0.3
end
```

```
››› mt-rates
```

lbd	ubd	rate
0	20	0
20001	50000	0.1
50001	10000	0.3

Making a Table Manually

```
greetings = table: at-or-after, and-before, greeting
  row:  0,  6, "Working Late?"
  row:  6, 12, "Good Morning!"
  row: 12, 18, "Good Afternoon!"
  row: 18, 24, "Good Evening!"
end
```

```
››› greetings
```

at-or-after	and-before	greeting
0	6	"Working Late?"
6	12	"Good Morning!"
12	18	"Good Afternoon!"
18	24	"Good Evening!"

Making a Table Manually

```
#table
covid-stats = table: name, population, deaths
  row: "United States", 331002651, 910104
  row: "United Kingdom", 67886011, 155754
  row: "France", 65273511, 130931
  row: "Germany", 83783942, 118527
  row: "Italy", 60461826, 146498
  row: "Spain", 46754778, 93225
  row: "Portugal", 10196709, 19905
end
```

- Each row has the same columns as the other rows, in the same order.
- A given column has uniform type, but different columns can have different types.
- The rows might be or might not be in some particular order.

```
››› covid-stats
```

name	population	deaths
"United States"	331002651	910104
"United Kingdom"	67886011	155754
"France"	65273511	130931
"Germany"	83783942	118527
"Italy"	60461826	146498
"Spain"	46754778	93225
"Portugal"	10196709	19905

Access to Rows

```
››› covid-stats.row-n(2)
```

"name"	"France"	"population"	65273511	"deaths"	130931

```
››› covid-stats.row-n(2)["population"]
65273511
›››
```

- Notice that row numbers start at zero.
- Notice that **`row-n`** is invoked using the **`.`** dot operator.
- The **`row-n`** function actually has two parameters.
- What are they?

```
high-bound = 100000
fun deaths-above(r :: Row) -> Boolean:
  doc: "Given a row, determine whether deaths are above high-bound."
  r["deaths"] > high-bound
end

low-bound = 50000
fun deaths-below(r :: Row) -> Boolean:
  doc: "Given a row, determine whether deaths are above high-bound."
  r["deaths"] < low-bound
end
```

```
rows-above = filter-with(covid-stats,deaths-above)

rows-below = filter-with(covid-stats,deaths-below)
```

`filter-with`: Given table t and row-predicate p, i.e., a function that takes a Row as input and returns a Boolean value. Return a new table that includes only the rows of t that satisfy predicate p.

```
››› filter-with(covid-stats,deaths-above)
```

name	population	deaths
"United States"	331002651	910104
"United Kingdom"	67886011	155754
"France"	65273511	130931
"Germany"	83783942	118527
"Italy"	60461826	146498

```
›››
```

```
››› filter-with(covid-stats,deaths-below)
```

name	population	deaths
"Portugal"	10196709	19905

```
›››
```

```
population-order-inc = order-by(covid-stats,"population",true)
population-order-dec = order-by(covid-stats,"population",false)
```

››› population-order-inc

name	population	deaths
"Portugal"	10196709	19905
"Spain"	46754778	93225
"Italy"	60461826	146498
"France"	65273511	130931
"United Kingdom"	67886011	155754
"Germany"	83783942	118527
"United States"	331002651	910104

››› population-order-dec

name	population	deaths
"United States"	331002651	910104
"Germany"	83783942	118527
"United Kingdom"	67886011	155754
"France"	65273511	130931
"Italy"	60461826	146498
"Spain"	46754778	93225
"Portugal"	10196709	19905

`order-by:` Given a table t, a column name n and a Boolean b, return a new table the same as t, but with the rows ordered by value in column n, increasing if Boolean b is true else decreasing.

```
death-order-inc = order-by(covid-stats,"deaths",true)
death-order-dec = order-by(covid-stats,"deaths",false)
```

››› death-order-inc

name	population	deaths
"Portugal"	10196709	19905
"Spain"	46754778	93225
"Germany"	83783942	118527
"France"	65273511	130931
"Italy"	60461826	146498
"United Kingdom"	67886011	155754
"United States"	331002651	910104

››› death-order-dec

name	population	deaths
"United States"	331002651	910104
"United Kingdom"	67886011	155754
"Italy"	60461826	146498
"France"	65273511	130931
"Germany"	83783942	118527
"Spain"	46754778	93225
"Portugal"	10196709	19905

`order-by:` Given a table t, a column name n and a Boolean b, return a new table the same as t, but with the rows ordered by value in column n, increasing if Boolean b is true else decreasing.

Getting Extremal Values in First or Last Row

```
#Row numbers start at zero.
tab-length = covid-stats.length()

biggest-state = population-order-dec.row-n(0)["name"]
biggest-state-alt = population-order-inc.row-n(tab-length - 1)["name"]

smallest-state = population-order-inc.row-n(0)["name"]
smallest-state-alt = population-order-dec.row-n(tab-length - 1)["name"]

happiest-state = death-order-inc.row-n(0)["name"]
saddest-state = death-order-dec.row-n(0)["name"]
```

Building a **death-rate** Column

```
fun deaths-per1K(r :: Row) -> Number:
  (r["deaths"] / r["population"]) * 1000
end

covid-stats-rel = build-column(covid-stats, "death rate", deaths-per1K)

death-rate-order-inc = order-by(covid-stats-rel,"death rate",true)
death-rate-order-dec = order-by(covid-stats-rel,"death rate",false)
local-happiest-state = death-rate-order-inc.row-n(0)["name"]
local-saddest-state = death-rate-order-dec.row-n(0)["name"]
```

build-column: Given a table t, a column name n and a function f, that takes a Row as input. Return a table with an additional column, named n, with values computed by applying f to each row.

Here we use the new column to find the nation states with the highest and lowest death rates. Each can be done in two different ways: placing the extremal value at the beginning or end of the table.

```
››› death-rate-order-inc
```

name	population	deaths	death rate
"Germany"	83783942	118527	1.41467442532126
"Portugal"	10196709	19905	1.95210042769681
"Spain"	46754778	93225	1.99391386266447
"France"	65273511	130931	2.00588260067701
"United Kingdom"	67886011	155754	2.29434603249850
"Italy"	60461826	146498	2.42298338789834
"United States"	331002651	910104	2.74953689117130

```
››› death-rate-order-dec
```

name	population	deaths	death rate
"United States"	331002651	910104	2.7495368911713036401
"Italy"	60461826	146498	2.4229833878983410127
"United Kingdom"	67886011	155754	2.2943460324985069457
"France"	65273511	130931	2.0058826006770188905
"Spain"	46754778	93225	1.9939138626644746340
"Portugal"	10196709	19905	1.9521004276968186500
"Germany"	83783942	118527	1.4146744253212626352

Transform some columns to show (fake) lower deaths & death rates.

```
fudge-factor = 0.5
fun fudge-deaths(n :: Number) -> Number:
  n * fudge-factor
end

fun fudge-table(t :: Table) -> Table:
  tf = transform-column(t, "deaths", fudge-deaths)
  transform-column(tf, "death rate", fudge-deaths)
end

covid-cover-up = fudge-table(covid-stats-rel)
```

transform-column: Given a table t, a column name n, and a function f with single input and output both of the same data type of the column. Return a new table same as t except that each value in column n, has been transformed by function f.

Here we first transform the deaths column. Then we transform the death rate column.

```
››› covid-stats-rel
```

name	population	deaths	death rate
"United States"	331002651	910104	2.7495368911
"United Kingdom"	67886011	155754	2.2943460324
"France"	65273511	130931	2.0058826006
"Germany"	83783942	118527	1.4146744253
"Italy"	60461826	146498	2.4229833878
"Spain"	46754778	93225	1.9939138626
"Portugal"	10196709	19905	1.9521004276

```
››› covid-cover-up
```

name	population	deaths	death rate
"United States"	331002651	455052	1.374768445
"United Kingdom"	67886011	77877	1.147173016
"France"	65273511	65465.5	1.002941300
"Germany"	83783942	59263.5	0.707337212
"Italy"	60461826	73249	1.211491693
"Spain"	46754778	46612.5	0.996956931
"Portugal"	10196709	9952.5	0.976050213