Problem Solving and Abstraction (CMPU 101)

Tom Ellman

Lecture 8
English contains **over 200,000 words**, with 171,476 active words and 47,156 inactive words.

https://thelanguagedoctors.org
What questions can we answer with this table?

• Let’s work with word length.

• Find the mean word-length.

• Make a bar-chart of word-length frequencies.
Load the English Table

english-ssid =
  "1FXXGnXEPKZS8gWiABJ-dgF3qLo1gC-Rid0XvpSt1NFQ"
eng-spreadsheet = load-spreadsheet(english-ssid)

english = load-table: word
  source: eng-spreadsheet.sheet-by-name("english", true)
sanitize word using string-sanitizer
end
Add a **length** Column
Add a **length** Column

```plaintext
fun word-length(r :: Row) -> Number:
    string-length(r["word"])
end

english-word-length =
    build-column(english,"length",word-length)
```
<table>
<thead>
<tr>
<th>word</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;a&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;aaa&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;aaas&quot;</td>
<td>4</td>
</tr>
<tr>
<td>&quot;aarhus&quot;</td>
<td>6</td>
</tr>
<tr>
<td>&quot;aaron&quot;</td>
<td>5</td>
</tr>
<tr>
<td>&quot;aau&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;aba&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;ababa&quot;</td>
<td>5</td>
</tr>
<tr>
<td>&quot;aback&quot;</td>
<td>5</td>
</tr>
<tr>
<td>&quot;abacus&quot;</td>
<td>6</td>
</tr>
</tbody>
</table>
Computing Mean Length
Computing Mean Length

mean-word-length = mean(english-word-length,"length")
Find the frequencies of word-lengths.
Find the frequencies of word-lengths.

```
english-length-counts = count(english-word-length, "length")
```

<table>
<thead>
<tr>
<th>value</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>279</td>
</tr>
</tbody>
</table>

There is 1 word of length 22
There are 7 words of length 17.
Etc.
Order the table by **value**, increasing.
Order the table by **value**, increasing.

```plaintext
group: english-length-counts-inc
   = order-by(english-length-counts,"value",true)
```

<table>
<thead>
<tr>
<th>value</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>133</td>
</tr>
<tr>
<td>3</td>
<td>791</td>
</tr>
<tr>
<td>4</td>
<td>2193</td>
</tr>
<tr>
<td>5</td>
<td>3169</td>
</tr>
<tr>
<td>6</td>
<td>3877</td>
</tr>
<tr>
<td>7</td>
<td>4072</td>
</tr>
<tr>
<td>8</td>
<td>3621</td>
</tr>
</tbody>
</table>
bar-chart(english-length-counts-inc,"value","count")
Dealing with Multiple Tables

• What if the data of interest lies in two or more different tables?
• Consider the example of RI municipalities.
• Find population density by municipality.
  – Population is on one sheet.
  – Land area is on another sheet.
Open the spreadsheet.

```
include shared-gdrive("dcic-2021",
   "1wyQZj_L0qqV9Ekgr9au6RX2iqt2Ga8Ep")
include gdrive-sheets
ssid = "1jHvn5CPE6RkTTQRlXQbY5n5p4aiOH7fZsnwK2s6s6tc"
spreadsheet = load-spreadsheet(ssid)
```
Load the municipality data.

ri-municipalities = load-table:
    name :: String, city :: Boolean,
    population-2000 :: Number,
    population-2010 :: Number
source: spreadsheet.sheet-by-name("municipalities", true)
end
Load the land area data.

ri-areas = load-table: name :: String, area :: Number
    source: spreadsheet.sheet-by-name("municipalities-area", true)
end
Write a function to compute area for a municipality.
Write a function to compute area for a municipality.

```plaintext
fun mun-name-to-area(areas :: Table, name :: String) -> Number:
    fun row-matches-namep(r :: Row) -> Boolean:
        r["name"] == name
    end
    filter-with(areas, row-matches-namep).row-n(0)["area"]
end
```

A nested predicate (row-matches-namep) checks whether a row contains the data of a named municipality. We use filter-width to make a table including only the row(s) of the named municipality. Finally, we get the first row and return the datum in the area column.
Anonymous Function (Lambda Expression)

Instead of:

```plaintext
fun row-matches-namep (r :: Row) -> Boolean: 
  r["name"] == name
end
```

we can write:

```plaintext
lam(r :: Row): r["name"] == name end
```

Since `row-matches-namep` is used only in `mun-name-to-area`, it does not need a name for future reference. An anonymous function is more concise, but perhaps less clear without a name.
Area for a Municipality Using an Anonymous Function
Area for a Municipality Using an Anonymous Function

fun mun-name-to-area(areas :: Table, name :: String) -> Number:
    filter-with(areas, lam(r :: Row): r["name"] == name end).row-n(0)["area"]
end

Here we use an anonymous function (predicate) as a parameter to the filter-width function.
Add an area column to the municipalities table.
Add an area column to the municipalities table.

```plaintext
fun add-area-column(municipalities :: Table, areas :: Table) -> Table:
  build-column(municipalities, "area",
    lam(r :: Row): mun-name-to-area(areas, r["name"])
  )
end
```

Here we use an anonymous function as a parameter to the `build-column` function.
Add a density column to the municipalities table.
Add a density column to the municipalities table.

```plaintext
fun density(r :: Row) -> Number:
    r["population-2010"] / r["area"]
end

fun add-density-column(municipalities :: Table) -> Table:
    build-column(municipalities, "density", density)
end
```
Putting it all Together
Putting it all Together

fun muns-by-density(municipalities :: Table, areas :: Table) -> Table:
  with-area = add-area-column(municipalities, areas)
  with-density = add-density-column(with-area)
  order-by(with-density, "density", false)
End

ri-muns-by-density = muns-by-density (ri-municipalities, ri-areas)

densest-municipality = mun-by-density.row-n(0)

Starting with the municipalities table, we first add the area
column and then add the density column. Finally we order
the table by density, decreasing.