Improving Your Text Life
We’ve all been there
We’ve all been there

Cancer Patient
Computer magic
Individualized Immunotherapy
We’ve all been there

- RNAseq
  - Variant protein prediction?
  - HLA typing?
  - Epitope Binding?
- MassSpec
  - Protein verification?
I Can Do This In Galaxy
Interactive Visualization
excel
Galaxy Text Tools

- Skip lines, select lines
- Add column, cut column, sort by column
- Group on column, join on column

- String enough text manipulation together ... you can
  almost simulate
  a relational database
Query Tabular

Why just simulate a relational Database?

Use the real thing

Query Tabular
uses
SQLite
Query Tabular

• Use existing SQLite database or create new
• Adds any number of tabular datasets as tables
• Tool default names for tables and columns or specify your own meaningful names

• Full power of SQL to generate tabular results with all the standard relational operations and common SQL functions

• Regex added as SQLite custom functions
Query Tabular

- Use existing SQLite database or create new
- Adds any number of tabular datasets as tables
- Tool default names for tables and columns or specify your own meaningful names
- Full power of SQL to generate tabular results with all the standard relational operations and common SQL functions
- Regex added as SQLite custom functions
Query Tabular

• Use existing SQLite database or create new
• Adds any number of tabular datasets as tables
• Tool default names for tables and columns or specify your own meaningful names

• Full power of SQL to generate tabular results with all the standard relational operations and common SQL functions

• Regex added as SQLite custom functions
Query Tabular

• Use existing SQLite database or create new
• Adds any number of tabular datasets as tables
• Tool default names for tables and columns or specify your own meaningful names

• Full power of SQL to generate tabular results with all the standard relational operations and common SQL functions

• Regex added as SQLite custom functions
Query Tabular

- Use existing SQLite database or create new
- Adds any number of tabular datasets as tables
- Tool default names for tables and columns or specify your own meaningful names

- Full power of SQL to generate tabular results with all the standard relational operations and common SQL functions

- Regex added as SQLite custom functions
Query Tabular

• Regex example: accession from fasta id:
  1. sp|Q8PSR0|Y3016_METMA Putative ABC transporter
  2. MCHU - Calmodulin - Human

SELECT re_sub('(?:(sp|tr)[|]+)([^|]+).*$', '\1', id) FROM search_db
WHERE re_search('^(sp|tr)[|]+', id)
Query Tabular

• Regex example: accession from fasta id:
  
  1. `sp|Q8PSR0|Y3016_METMA` Putative ABC transporter  
  2. `MCHU - Calmodulin - Human`

SELECT re_sub(‘(?:sp|tr)[| ]([^ | ]+).*’,’\1’,id) FROM search_db
WHERE re_search(‘^(sp|tr)[| ]’,id)
Query Tabular

• Regex example: accession from fasta id:
  1. `sp|Q8PSR0|Y3016_METMA` Putative ABC transporter
  2. MCHU - Calmodulin - Human

```
SELECT re_sub('(?:sp|tr)[[:|:]][^[:|:]]+.*$','\1',id)
FROM search_db
WHERE re_search('^((sp|tr)[[:|:]]',id)
```
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – regex replace value in column
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – regex replace value in column
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – regex replace value in column
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – \textit{regex replace value in column}
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – regex replace value in column
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
Query Tabular

• Input Reader Line Filters
  – skip leading lines
  – comment char
  – regex expression matching
  – select/reorder columns
  – regex replace value in column
  – prepend/append a line number column
  – normalize list columns

• Line Filters can be chained together
## Normalizing: Input Dataset

Input Dataset

<table>
<thead>
<tr>
<th>Person</th>
<th>Pets</th>
<th>Pet Name</th>
<th>Pet Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>1</td>
<td>Allie</td>
<td>cat</td>
</tr>
<tr>
<td>John</td>
<td>2</td>
<td>Spot, Frisky</td>
<td>dog, cat</td>
</tr>
<tr>
<td>Delphine</td>
<td>1</td>
<td>Indeed</td>
<td>cat</td>
</tr>
</tbody>
</table>
## Normalizing: People Table

### Input Dataset

<table>
<thead>
<tr>
<th>Person</th>
<th>Pets</th>
<th>Pet Name</th>
<th>Pet Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>1</td>
<td>Allie</td>
<td>cat</td>
</tr>
<tr>
<td>John</td>
<td>2</td>
<td>Spot, Frisky</td>
<td>dog, cat</td>
</tr>
<tr>
<td>Delphine</td>
<td>1</td>
<td>Indeed</td>
<td>cat</td>
</tr>
</tbody>
</table>

Line filters:
1. Append Line Number
2. Select columns: 5, 1, 2

### People

<table>
<thead>
<tr>
<th>id</th>
<th>Person</th>
<th>Pets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jim</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Delphine</td>
<td>1</td>
</tr>
</tbody>
</table>
## Normalizing: Pet Table

### Input Dataset

<table>
<thead>
<tr>
<th>Person</th>
<th>Pets</th>
<th>Pet Name</th>
<th>Pet Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>1</td>
<td>Allie</td>
<td>cat</td>
</tr>
<tr>
<td>John</td>
<td>2</td>
<td>Spot,Frisky</td>
<td>dog,cat</td>
</tr>
<tr>
<td>Delphine</td>
<td>1</td>
<td>Indeed</td>
<td>cat</td>
</tr>
</tbody>
</table>

Line filters:
1. Append Line Number
2. Normalize columns:3,4
3. Select columns:5,3,4

### Pet

<table>
<thead>
<tr>
<th>id</th>
<th>Pet Names</th>
<th>Pet Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Allie</td>
<td>cat</td>
</tr>
<tr>
<td>2</td>
<td>Spot</td>
<td>dog</td>
</tr>
<tr>
<td>2</td>
<td>Frisky</td>
<td>cat</td>
</tr>
<tr>
<td>3</td>
<td>Indeed</td>
<td>cat</td>
</tr>
</tbody>
</table>
Normalizing: Query

**People**

<table>
<thead>
<tr>
<th>id</th>
<th>Person</th>
<th>Pets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jim</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Delphine</td>
<td>1</td>
</tr>
</tbody>
</table>

**Pet**

<table>
<thead>
<tr>
<th>id</th>
<th>Pet Name</th>
<th>Pet Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Allie</td>
<td>cat</td>
</tr>
<tr>
<td>1</td>
<td>Spot</td>
<td>dog</td>
</tr>
<tr>
<td>2</td>
<td>Frisky</td>
<td>cat</td>
</tr>
<tr>
<td>3</td>
<td>Indeed</td>
<td>cat</td>
</tr>
</tbody>
</table>

Query:
```sql
SELECT Person, Pets, PetName
FROM People
JOIN Pet ON People.id = Pet.id
WHERE PetType = 'cat'
```

**Result**

<table>
<thead>
<tr>
<th>Person</th>
<th>Pets</th>
<th>Pet Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim</td>
<td>1</td>
<td>Allie</td>
</tr>
<tr>
<td>John</td>
<td>2</td>
<td>Frisky</td>
</tr>
<tr>
<td>Delphine</td>
<td>1</td>
<td>Indeed</td>
</tr>
</tbody>
</table>
Query Tabular in action

An Alignment-Free “Metapeptide” Strategy for Metaproteomic Characterization of Microbiome Samples Using Shotgun Metagenomic Sequencing

Damon H. May,† Emma Timmins-Schiffman,‡ Molly P. Mikan,§ H. Rodger Harvey,∥ Elhanan Borenstein,¶∥ Brook L. Nunn,¶ and William S. Noble†∥

†Department of Genome Sciences and ‡Department of Computer Science and Engineering, University of Washington, Seattle, Washington 98195-5065, United States
§Department of Ocean, Earth & Atmospheric Sciences, Old Dominion University, Norfolk, Virginia 23529, United States
¶Santa Fe Institute, Santa Fe, New Mexico 87501, United States
Query Tabular in action

Workflow Canvas | ABRF2017 UNIPEPT: Taxonomy Workflow

- PSM Report
  - output

- Query Tabular
  - Add tables to this Database
    - Database Table 1 > Tabular Dataset for Table
      - sqlitedb (sqlite)
      - output (tabular)

- Unipept
  - Tabular Input Containing Peptide column
    - output_json (d3_hierarchy, json)
    - output_tsv (tabular)
    - output_csv (csv)
    - output_unmatched (tabular)

- SQLite to tabular
  - SQLite Database
    - query_results (tabular)

- Query Tabular
  - Add tables to this Database
    - Database Table 1 > Tabular Dataset for Table
    - Database Table 2 > Tabular Dataset for Table
      - sqlitedb (sqlite)
      - output (tabular)

Peptide Spectral Match Report
Query Tabular in action

SELECT distinct sequence FROM psm WHERE score > 95
Query Tabular in action

Unipept assigns taxonomy to each peptide sequence
Query Tabular in action

Join PSM and Unipept tables to produce count summary for each genus
Join PSM and Unipept tables to produce count summary for each genus

<table>
<thead>
<tr>
<th>genus</th>
<th>PSMs</th>
<th>DISTINCT PEPTIDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidatus Pelagibacter</td>
<td>478</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>292</td>
<td>30</td>
</tr>
<tr>
<td>Planktomarina</td>
<td>161</td>
<td>20</td>
</tr>
<tr>
<td>Nitrosopumilus</td>
<td>122</td>
<td>27</td>
</tr>
<tr>
<td>Polaribacter</td>
<td>86</td>
<td>18</td>
</tr>
<tr>
<td>Candidatus Ruthia</td>
<td>75</td>
<td>10</td>
</tr>
<tr>
<td>Rhodobacter</td>
<td>75</td>
<td>11</td>
</tr>
<tr>
<td>Octadecabacter</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Pseudomonas</td>
<td>23</td>
<td>2</td>
</tr>
</tbody>
</table>
Query Tabular in action

http://z.umn.edu/metaproteomicsgateway
Warnings

Query Tabular is not recommend for those who suffer an averse reaction to the SQL.

If you experience a query lasting more than 4 hours, consult your relational DB expert, as this may indicate a need for better indices or a more efficient query.
Query Tabular

Don’t put up with lengthy, complicated text manipulation
Query Tabular

For the best relational experience in your galaxy

A Johnson Product
jj@umn.edu