Between-element interactivity

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

What is between-element?

- Previously: Interact with **input** (e.g., dropdown); to trigger change in a figure
- Now: Interact with **figure**; to trigger change in a figure
- Two specific ways:
  - Hover to filter and regenerate (via callback)
  - Click to filter and regenerate (via callback)

The hoverData property

A familiar bar chart and text component:

```
# Create figure & dash.Dash()
html.Div(children=[
    dcc.Graph(id='bar_fig',
              figure=ecom_bar),
    html.Br(),
    html.H2("The Hover Data:"),
    html.P(id='text_output')
```

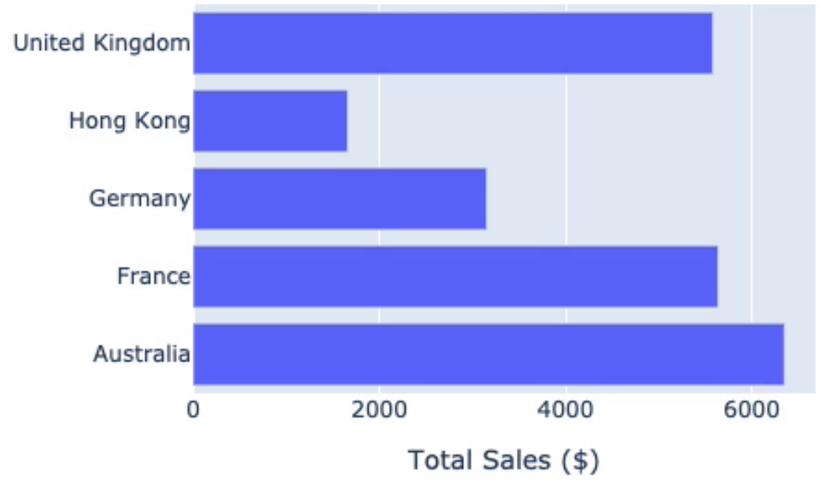
Set up a callback

```
@app.callback(
    Output('text_output', 'children'),
    Input('bar_fig', 'hoverData'))
def capture_hover_data(hoverData):
    return str(hoverData)
```

Our first hover

What happens:

- User hovers: `hoverData` property of the figure changes
- Callback takes and returns this to `.P()` tag
- We can see point-related information
  - Now: Use that information!

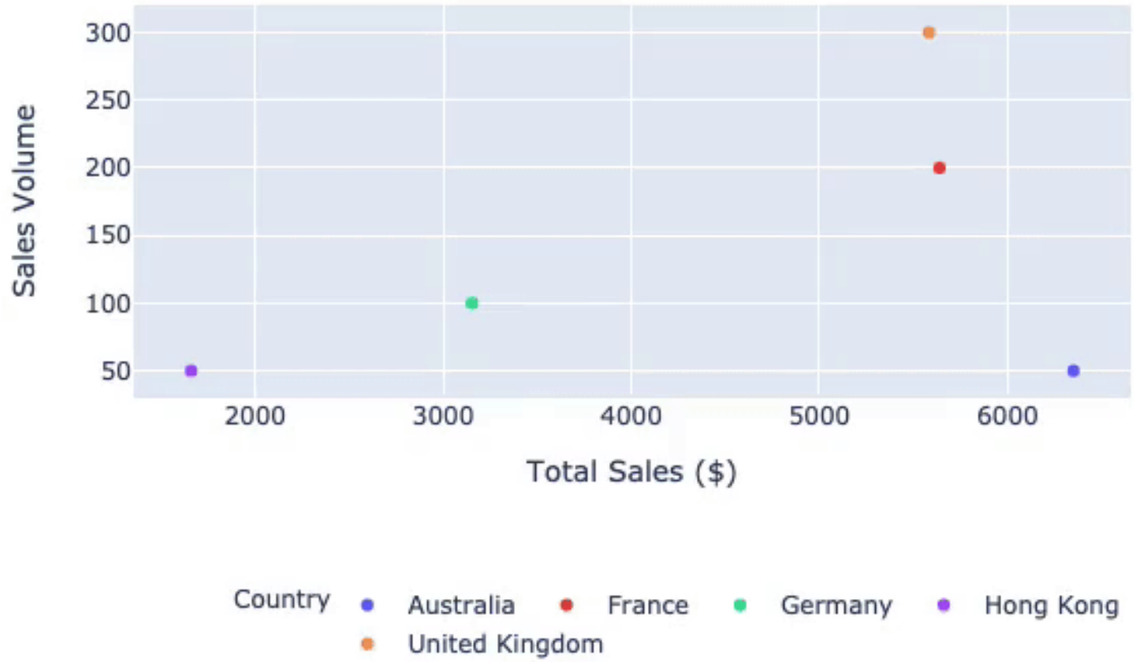


The Hover Data:

None

Beware missing info

- Aim: Use 'country' in the callback (filter and regenerate a graph)
  - Previous example - in `hoverData`
- See this scatter plot (graph type and relevant ID's updated)
  - There is no country!



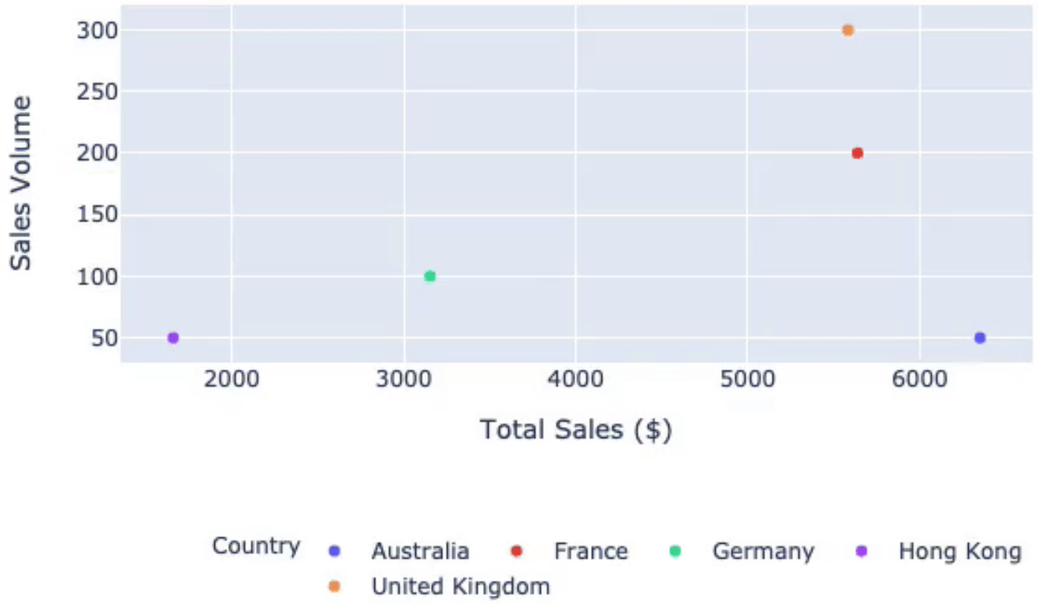
The Hover Data:

None

Adding custom data

```
ecom_scatter = px.scatter(ecom_data,
    x='Total Sales ($)',
    y='Sales Volume', color='Country',
    custom_data=['Country'])
```

- Now `customData` contains the country!



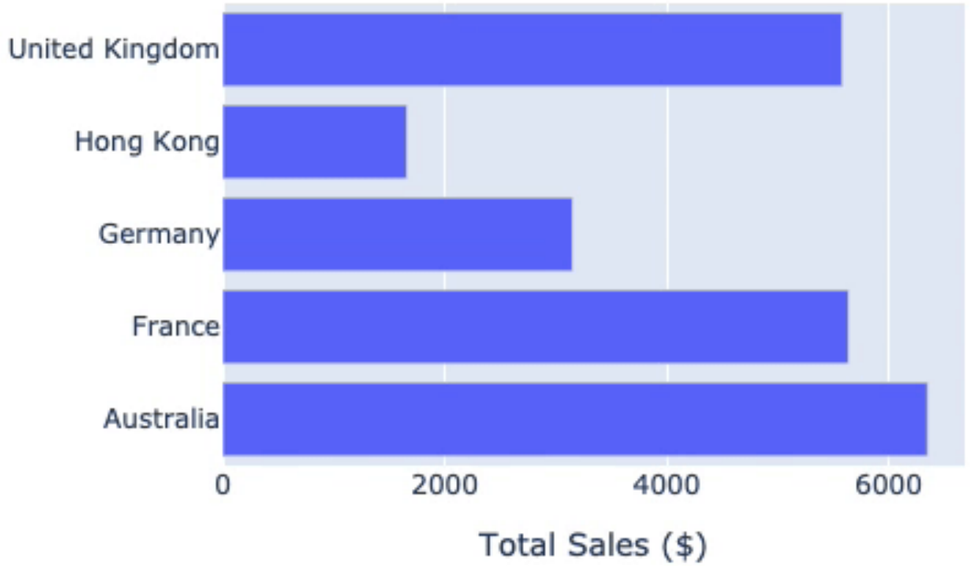
The Hover Data:

None

What about clicking?

- Can also trigger a callback when a point is clicked
  - Only one change required (`clickData` property)

```
@app.callback(
    Output('text_output', 'children'),
    Input('bar_fig', 'clickData'))
def capture_hover_data(clickData):
    return str(clickData)
```



The Click Data:

None

- Notice: Did not immediately appear

Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Chained callbacks

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

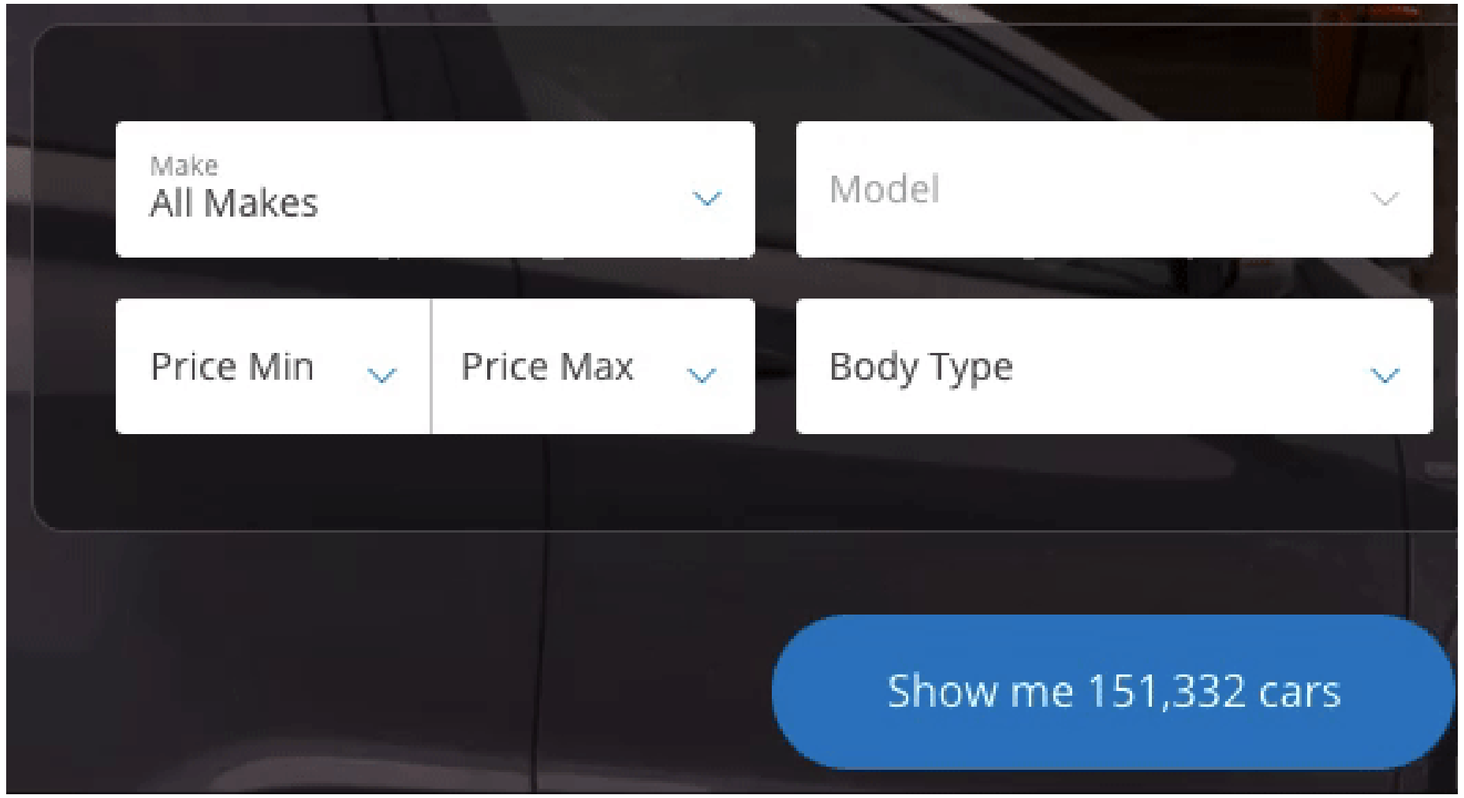
datacamp

Why chain callbacks?

- So far callbacks cause:
  - Regenerate plots
  - Change HTML / text
- What about callback triggering another callback?
- Use case: conditional dropdown

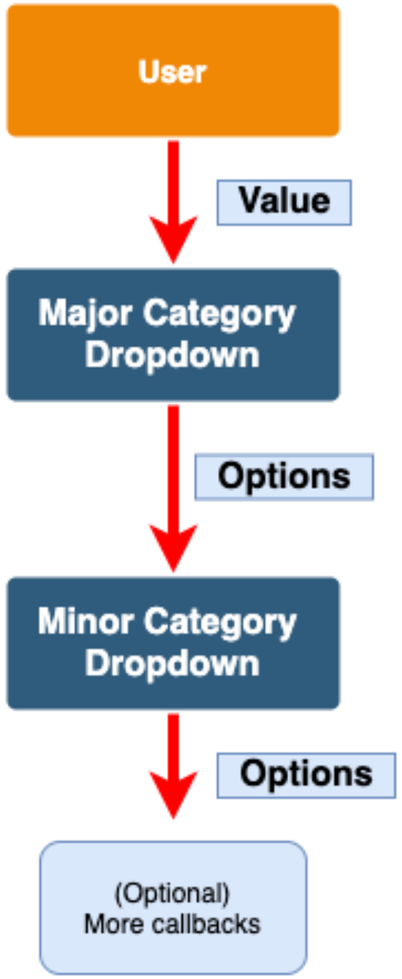
Let's do this in Dash!

A common example



Inputs and outputs

- The trick: be aware of callback pathways (inputs and outputs)
- Helpful tool: an input-output diagram
- The flow:
  - User changes `value` of first dropdown (**INPUT**)
  - A callback subsets and returns `options` of second dropdown (**OUTPUT**)
  - Another callback could be triggered (**INPUT**) by `options` change on second dropdown, and so on



Chained callbacks in Dash

The callbacks involved:

```
@app.callback(
  Output('minor_cat_dd', 'options'),
  Input('major_cat_dd', 'value'))
def update_dd(major_cat_dd):
    # Filter options (list of dicts)
    return minor_options
```

Set a default value

```
@app.callback(
  Output('minor_cat_dd', 'value'),
  Input('minor_cat_dd', 'options'))
def update_dd(minor_cat_options):
    # Pick a default value
    return chosen_value
```

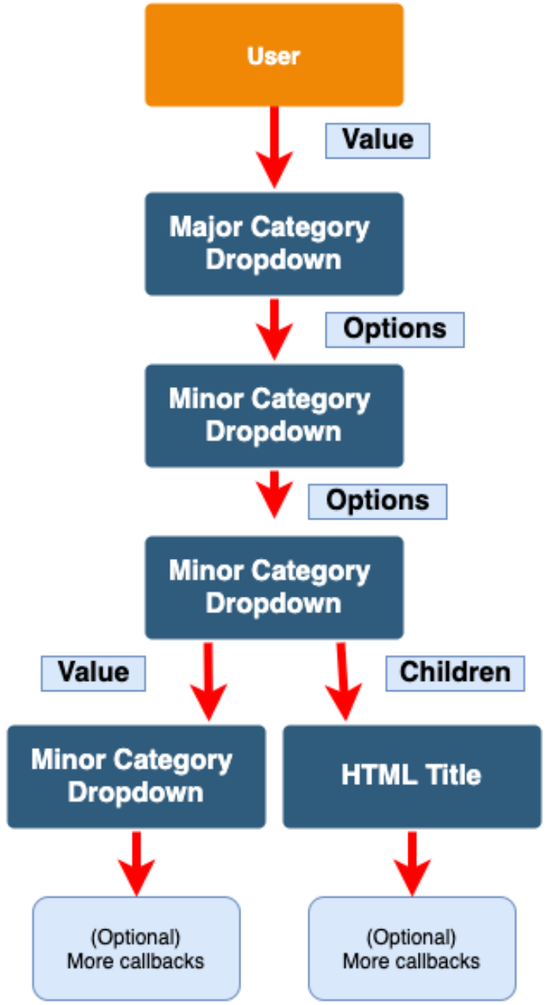
Multiple outputs

May wish to update multiple elements

- In our example: update a HTML title as well
- Add another output

```
@app.callback(
  Output('my_title', 'children'),
  Output('minor_cat_dd', 'value'),
  Input('minor_cat_dd', 'options')
)
def some_function(input):
  # function body
  return title_value, dropdown_value
```

Multiple outputs diagram



Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Dash Data Table introduction

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

What is a Dash Data Table?

- HTML has a native table tag (available in **Dash** `html.Table()`)
- Problem: HTML tables are static
- Introducing Dash Data Tables (component for the `app.layout()`)
  - Many visual & interactive customizations
    - e.g., Filter, hiding, export, pagination, hover, styling
  - Enhance user experience

The basic table

```
from dash_table import DataTable
d_columns = [
 {"name": 'Major Category',
  "id": "Major Category"},
 {"name": 'Total Sales ($)',
  "id": "Total Sales ($)"},
 {"name": 'Sales Volume',
  "id": "Sales Volume"}]
```

```
d_table = DataTable(
  columns=d_columns,
  data=major_cat_tb.to_dict('records'),
  cell_selectable = False)
```

Major Category	Total Sales ($)	Sales Volume
Clothes	4950.370000000002	176
Garden	6040.080000000002	189
Household	4986.580000000001	163
Kitchen	6407.9	172

Format the numbers

- Problem 1: Financial number formatting
  - Solution: `FormatTemplate`

```
from dash_table import FormatTemplate
money_format = FormatTemplate.money(2)
d_columns=[{"name": 'Total Sales ($)',
    "id": "Total Sales ($)",
    'type':'numeric',
    'format':money_format}
  # Other column definitions
]
```

Nicely formatted!

Major Category Stats

Major Category	Total Sales ($)	Sales Volume
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Add sorting

Adding sorting:

```
d_table = DataTable(
  columns=d_columns,
  data=major_cat_tb.to_dict('records'),
  cell_selectable=False,
  # Add sort ability
  sort_action='native')
```

With Sorting:

Major Category	Total Sales ($)	Sales Volume
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Add filtering

Adding Filtering:

```
d_table = DataTable(
  columns=d_columns,
  data=major_cat_tb.to_dict('records'),
  cell_selectable=False,
  # Add filter ability
  filter_action='native'
)
```

With Filtering:

Major Category	Total Sales ($)	Sales Volume
filter da...a...		
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Pagination

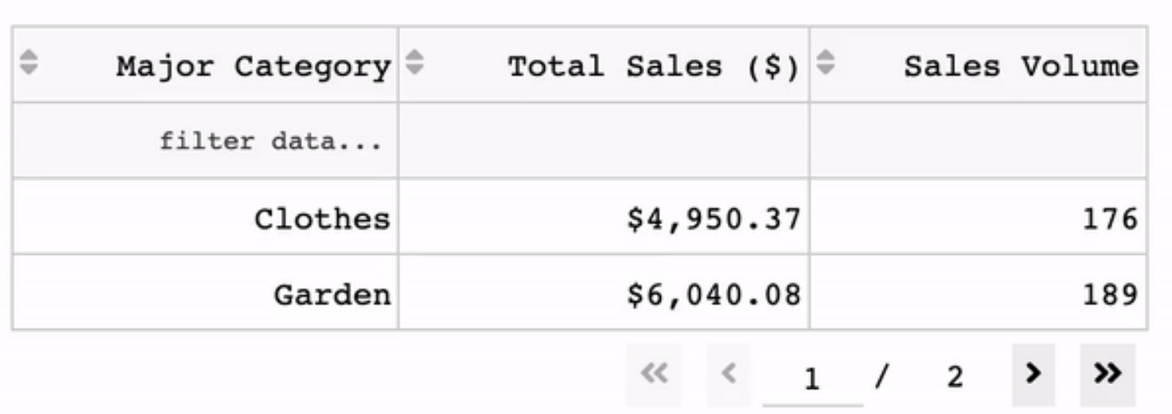
Problem: long tables

- Pagination: show (n) entries per 'page' with navigation buttons
  - `page_current` = page to start on
  - `page_size` = entries per page

```
d_table = DataTable(
    # Previous options
    page_current= 0,
    page_size= 2,
    page_action="native")
```

Pagination in action:

- Next, previous, first and last buttons
- Enter page number
- Filter and sort still works!



Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Dash Data Table interactivity

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

datacamp

Styling all Data Table cells

- Can't use the 'style'
- For all cells use `style_cell`

```
d_table = DataTable(
  # Other table properties
  style_cell=(
    {'textAlign':'left'}
))
```

Major Category	Total Sales ($)
Clothes	$4,950.37
Garden	$6,040.08
Household	$4,986.58
Kitchen	$6,407.90

Styling some Data Table cells

- For a specific cell use `style_cell_conditional`

```
d_table = DataTable(
  # Other table properties
  style_cell={
    {'textAlign':'left'}),
  style_cell_conditional=[
 {'if': {'column_id':'Sales Volume'},
 'textAlign':'center'}])
```

Major Category	Total Sales ($)	Sales Volume
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Styling Data Table headers

- Styling column headers is similar
  - All: `style_header`
  - Specific: `style_header_conditional`

```
d_table = DataTable(
    # Other table properties
style_header={
  'background-color':'black',
  'color':'white'},
style_header_conditional=[
  {'if': {'column_id':'Sales Volume'},
   'background-color':'blue'}])
```

Styled column headers;

Major Category Stats

Major Category	Total Sales ($)	Sales Volume
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Selecting cells

- Selecting cells first (then rows, columns)
- Set DataTable's `cell_selectable` argument to `True`
- A callback to print available data

```
@app.callback(
  Output('test_text', 'children'),
  Input('my_dt', 'selected_cells'))
def print_it(input):
  return str(input)
```

Selecting cells:

Major Category Stats

Major Category	Total Sales ($)	Sales Volume
Clothes	$4,950.37	176
Garden	$6,040.08	189
Household	$4,986.58	163
Kitchen	$6,407.90	172

Select output

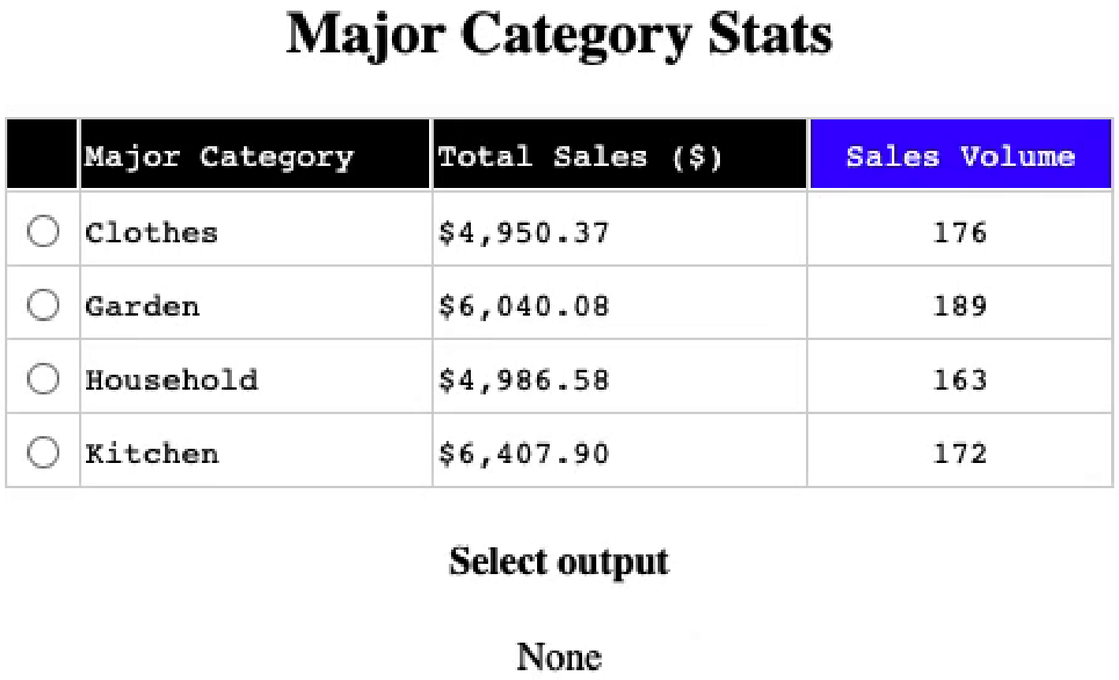
None

Selecting rows

- Set Data Table `row_selectable` to `single` or `multi`
- A callback to print available data

```
@app.callback(
  Output('test_text', 'children'),
  Input('my_dt', 'selected_rows'))
def print_it(input):
  return str(input)
```

The row index is returned;

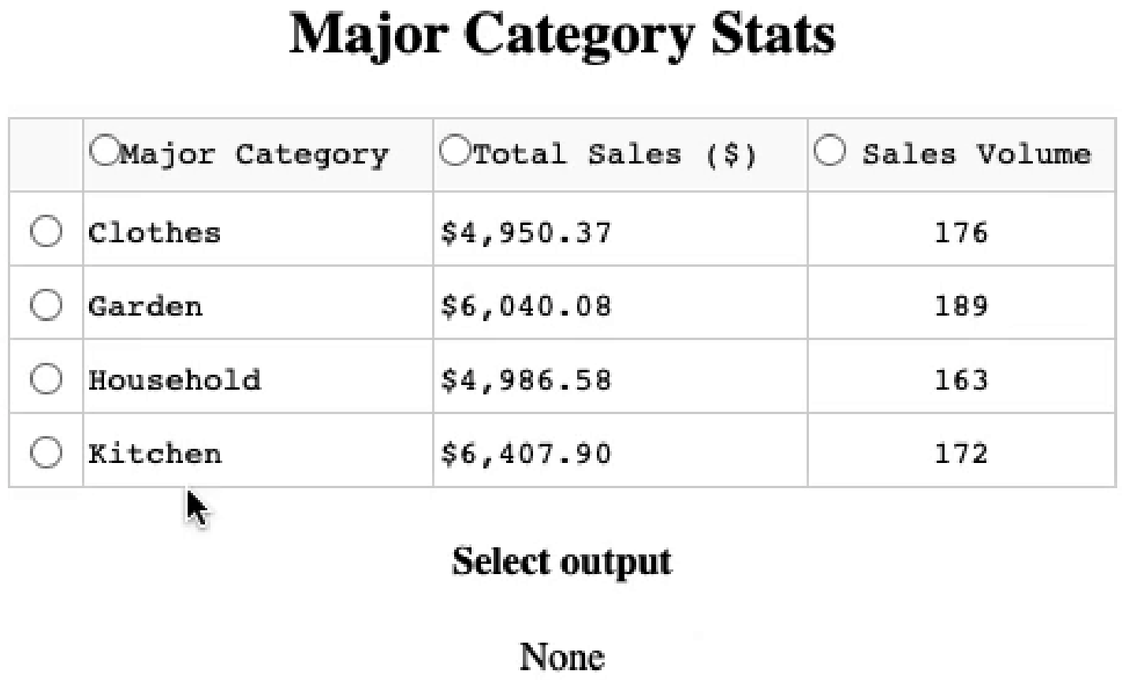


Selecting columns

- Set Data Table `column_selectable` to `single` or `multi`
  - Add `'selectable': True` to column definitions
  - e.g.,
    ```
    {"name": "Sales Volume", "id":
    "Sales Volume", "selectable":True}
    ```
- A callback to print available data

```
@app.callback(
  Output('test_text', 'children'),
  Input('my_dt', 'selected_columns'))
def print_it(input):
  return str(input)
```

Column ID is returned



Let's practice!

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Wrap-up video

BUILDING DASHBOARDS WITH DASH AND PLOTLY

Alex Scriven
Data Scientist

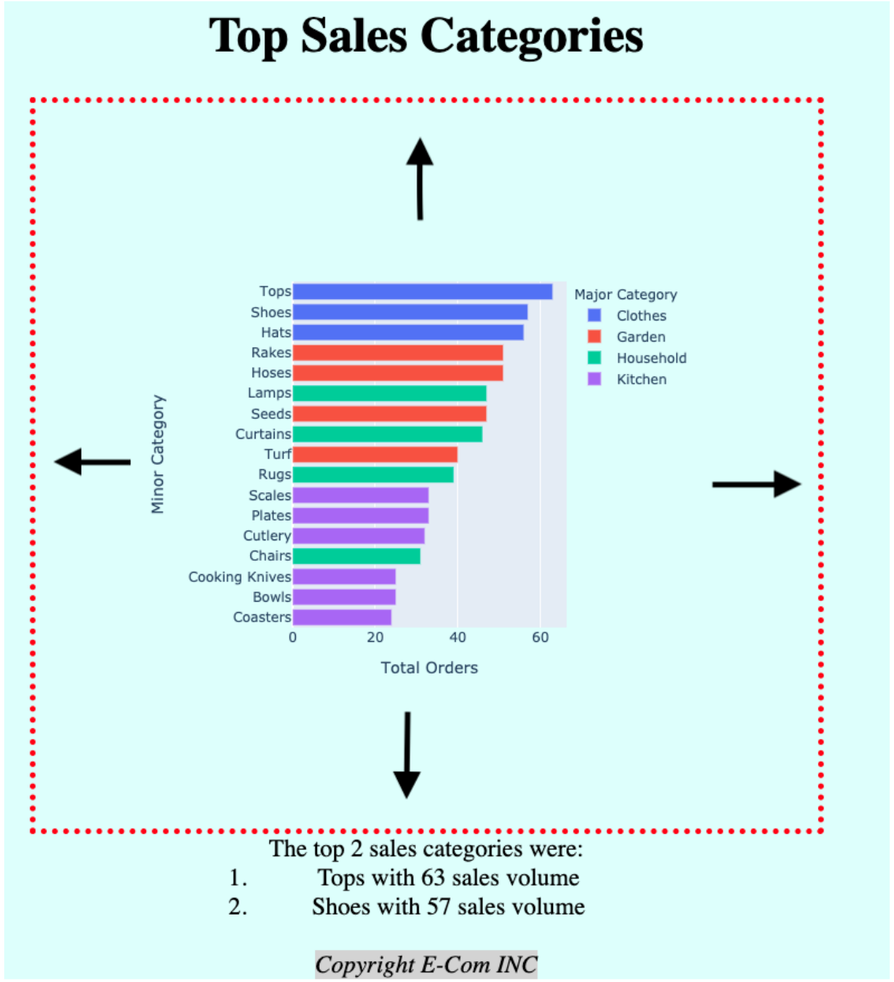
datacamp

Chapter 1

- Revised Plotly, discovered Dash
- Created first Dash app
- An overview of HTML

Chapter 2

- Deeper dive into HTML and CSS
- Place, size, and style app elements
  - The important `style` dictionary

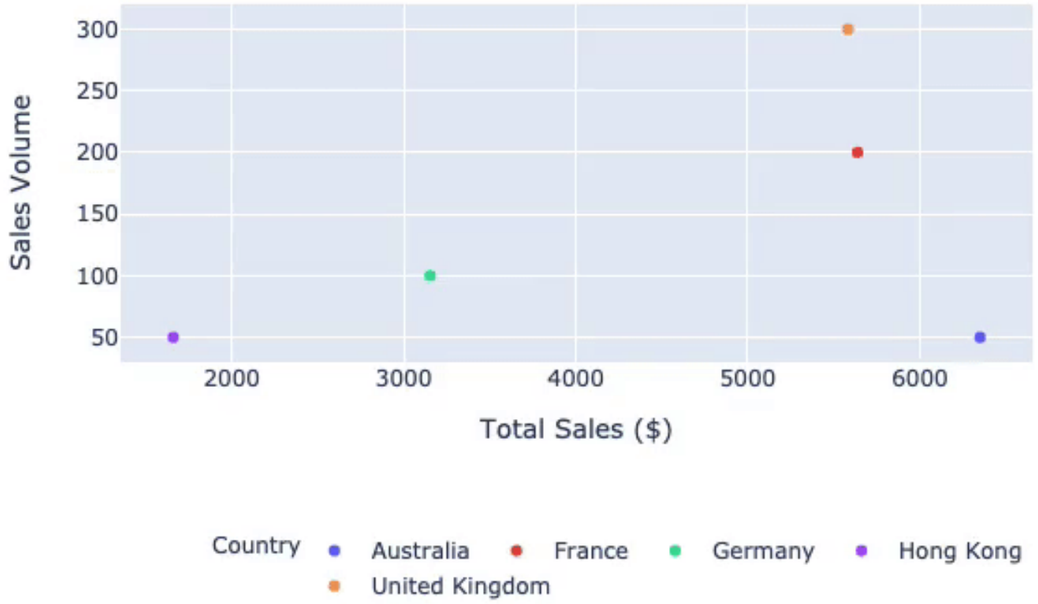


Chapter 3

- Enhanced interactivity with callbacks
- Advanced user experiences with interactive components
  - Dropdowns, date pickers and free-text entry

Chapter 4

- Next level interactivity from hover and click
- Building Data Tables
  - Making them interactive



The Hover Data:

None

Next steps?

- Build your own dashboards (practice!)
- Experiment with callback chains
- Continue learning DataTable

Thank you!

BUILDING DASHBOARDS WITH DASH AND PLOTLY