edgetier

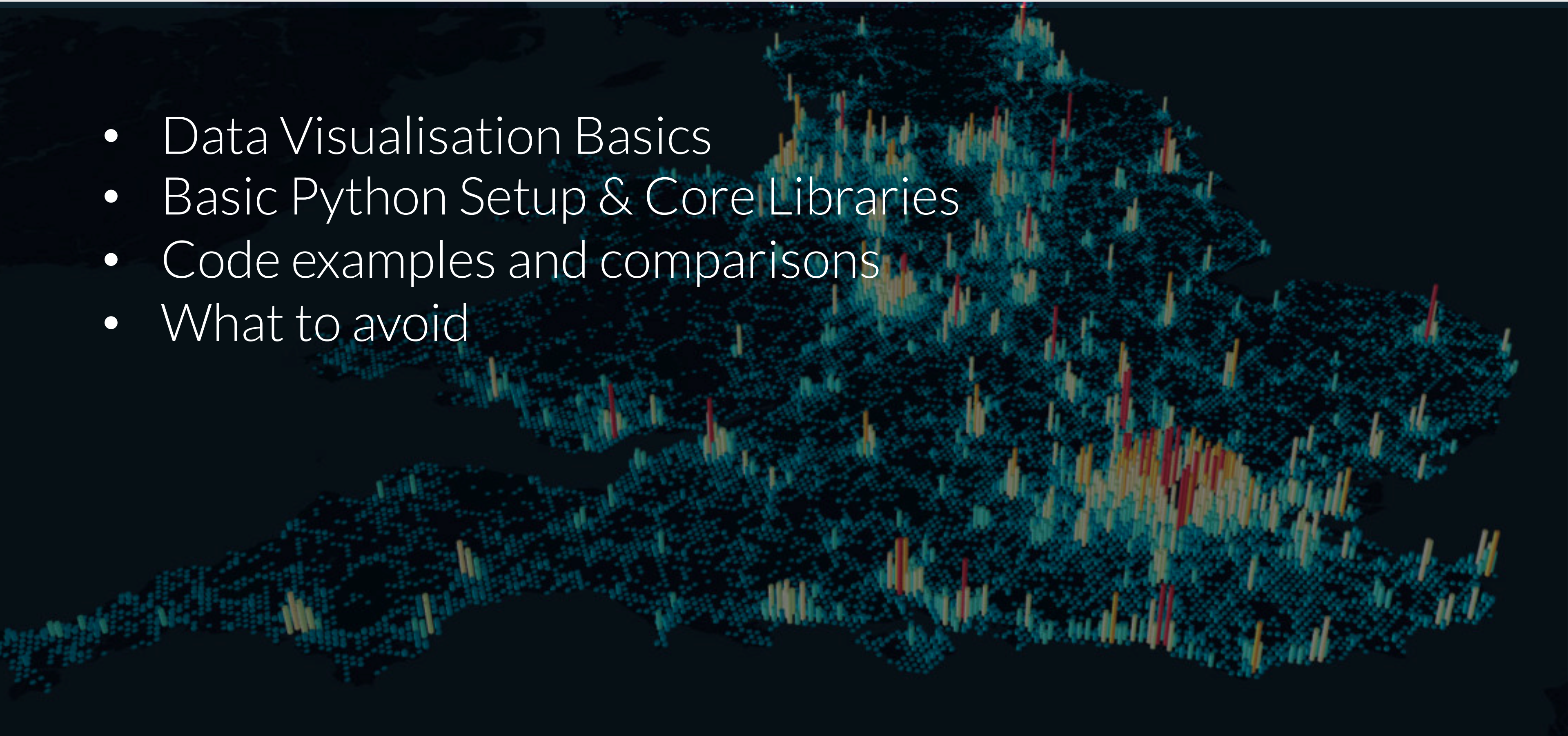
COMPLEX DECISIONS SIMPLIFIED

Data Visualisation in Python

Quick and easy routes to plotting magic

Shane Lynn Ph.D.
@shane_a_lynn

Outline



- Data Visualisation Basics
- Basic Python Setup & Core Libraries
- Code examples and comparisons
- What to avoid

EdgeTier

EdgeTier specialise in data and artificial intelligence products for customer contact centres.

Commercially focused SaaS to increase revenue and reduce costs

Focus on data science, machine learning, and automation

AI system works alongside customer service agents to increase efficiency by 100%

Data Visualisation

Data visualisation is a general term that describes any effort to help people understand the significance of **data** by placing it in a visual context.

Data Visualisation

Choice of Data Visualisation Tool is important

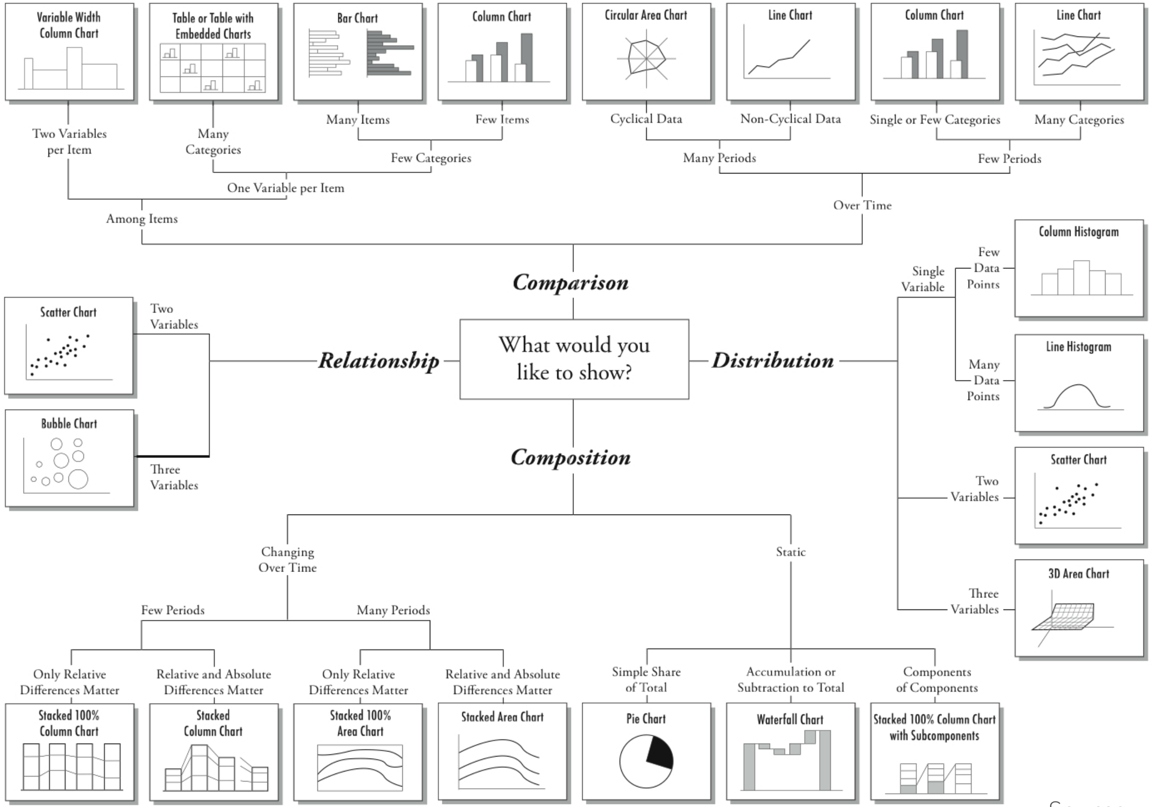
Iteration speed

Un-intrusive

Flexible

Aesthetically pleasing

Chart Choice



Source: www.extremepresentation.com

Chart Choice – Fearsome Foursome

BARPLOT

Represents the value of entities using bar of various length.

HISTOGRAM

An accurate graphical representation of the distribution of numeric data.

SCATTER PLOT

Show the relationship between 2 numeric variables.

LINE CHART

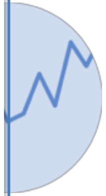
Shows the evolution of numeric variables.

Icons: www.data-to-viz.com

Chart Choice – Fearsome Foursome

BA

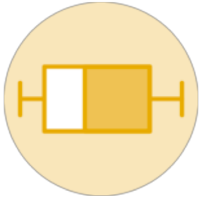
Repr
value
usi
vari



CHART

s the
tion of
variables.

Special Mentions



BOXPLOT

Summarize the distribution of numeric variables

SANKEY DIAGRAM

Showing flows with smooth links

CHOROPLETH MAP

Display an aggregated value for each region of a map

Icons: www.data-to-viz.com

Data Visualisation in Python

Python Visualisation

- Lots of choice of libraries
- Many tools, with varied APIs & outputs
- Best to conquer and become familiar with one / two

Interactive environment

jupyterhub

PC

SPYDER

Data Manipulation Library

pandas

$y_it = \beta' x_{it} + \mu_i + \epsilon_{it}$

xarray

PyTables

Visualisation Library

matplotlib

ggplot2

bokeh

plotly

seaborn

Matplotlib

Grand daddy of Python Plotting

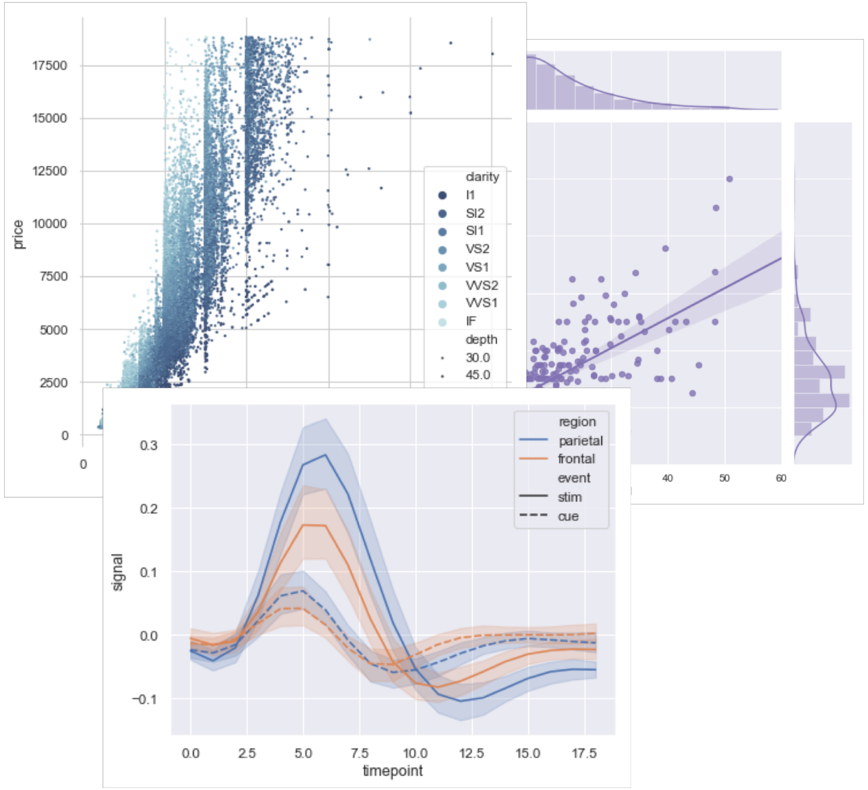
Low level plotting library with Matlab-like API

+ Very flexible, complete control

- Verbose plots, aesthetically lacking, sometimes difficult with Pandas

...need to know enough to debug...

Pandas / Seaborn / Altair



Higher level plotting

Pandas – Visualisation API built into DataFrame & Series objects, interface to Matplotlib.

Seaborn – extends and provides high-level API on Matplotlib with improved styling.

Altair – Built on "Vega-Lite" visualisation grammar. Allows some interactive plots in Jupyter Notebooks.

Basic Notebook Setup

Imports on Matplotlib

Top of notebook – inline vs notebook style.

Theme also can be chosen here

```python
# Pandas used for data analysis and managmeent
import pandas as pd
# Numpy library for arry and numerical functions
import numpy as np
# Matplotlib pyplot provides plotting API
import matplotlib as mpl
from matplotlib import pyplot as plt
# "Standard" to load seaborn as "sns"
import seaborn as sns
# Altair is a visualisation library based on Vega
import altair as alt
alt.renderers.enable('notebook')

# For output plots inline in notebook:
%matplotlib inline
# For interactive plot controls on MatplotLib output:
# %matplotlib notebook

# Set the default figure size for all inline plots
# (note: needs to be AFTER the %matplotlib magic)
plt.rcParams['figure.figsize'] = [8, 5]
```

Sample Data

EdgeTier relevant sample dataset on chat system performance.

Agents answering customer chats from different websites and languages – 5477 chats over 100 agents.

In [7]:
```
data.head()
```

Out[7]:

	user_id	chat_id	language	number_messages	handling_time	start_time	website	website_summary
0	User 1495	8347	English	7	187.0	2018-05-30 11:59:48.422292	Website B	Website B
1	User 1495	8348	English	7	258.0	2018-05-30 11:25:15.111164	Website B	Website B
2	User 1495	8349	English	8	529.0	2018-05-30 10:40:38.255353	Website B	Website B
3	User 1495	8350	English	14	840.0	2018-05-30 12:08:16.612382	Website B	Website B
4	User 1495	8351	English	11	1283.0	2018-05-30 11:15:28.393306	Website B	Website B

The Bar Plot

The Bar Plot - Matplotlib

Bar plot of chats per user

In [64]:
```
chats_per_user = data.groupby('user_id')['chat_id'].count().reset_index()
chats_per_user.columns = ['user_id', 'number_chats']
chats_per_user = chats_per_user.sort_values('number_chats', ascending=False)
chats_per_user.head()
```

Out[64]:

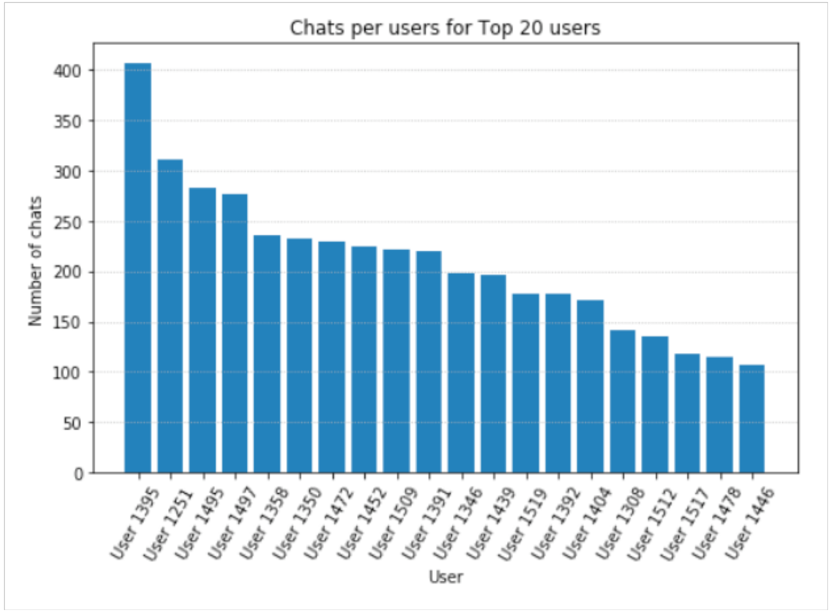
	user_id	number_chats
49	User 1395	406
1	User 1251	311
78	User 1495	283
79	User 1497	276
39	User 1358	236

Python visualisation libraries often require that the data for plotting is pre-formatted for visualisation.

For Pandas and Matplotlib, the visualisation library often only present the values, and does not do calculations.

The Bar Plot - Matplotlib

```python
top_n = 20
plt.bar(x=range(top_n),
        height=chats_per_user[0:top_n]['number_chats'])
plt.xticks(range(top_n), chats_per_user[0:top_n]['user_id'],
           rotation=60)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
ax = plt.gca()
ax.yaxis.grid(linestyle=':')
```



.bar() function does the work, manually position ‘x’ labels and positions.

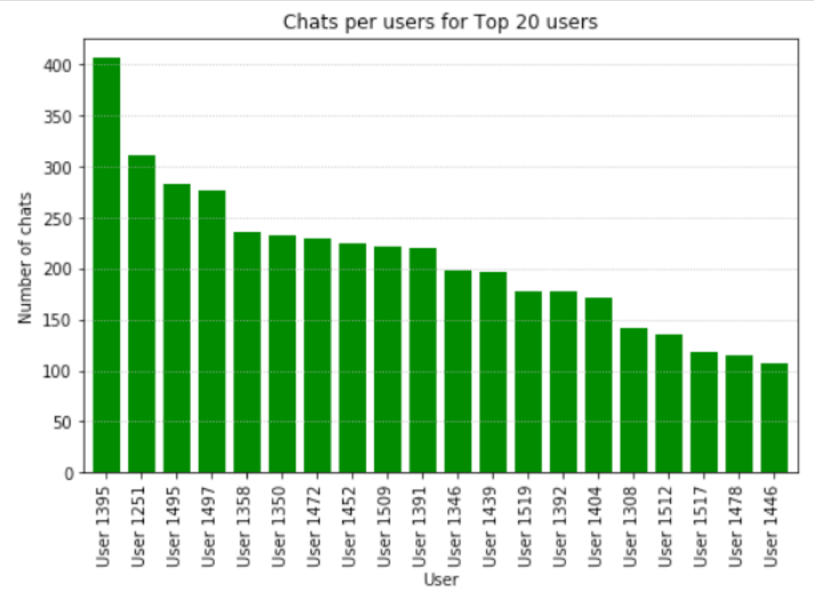
Most code here is formatting and display.

The Bar Plot - Pandas

pandas

$y_it = \beta' x_{it} + \mu_i + \epsilon_{it}$

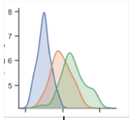
```python
chats_per_user[0:20].plot(
    x='user_id', y='number_chats',
    kind='bar', legend=False, color='green',
    width=0.8
)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
plt.gca().yaxis.grid(linestyle=':')
```



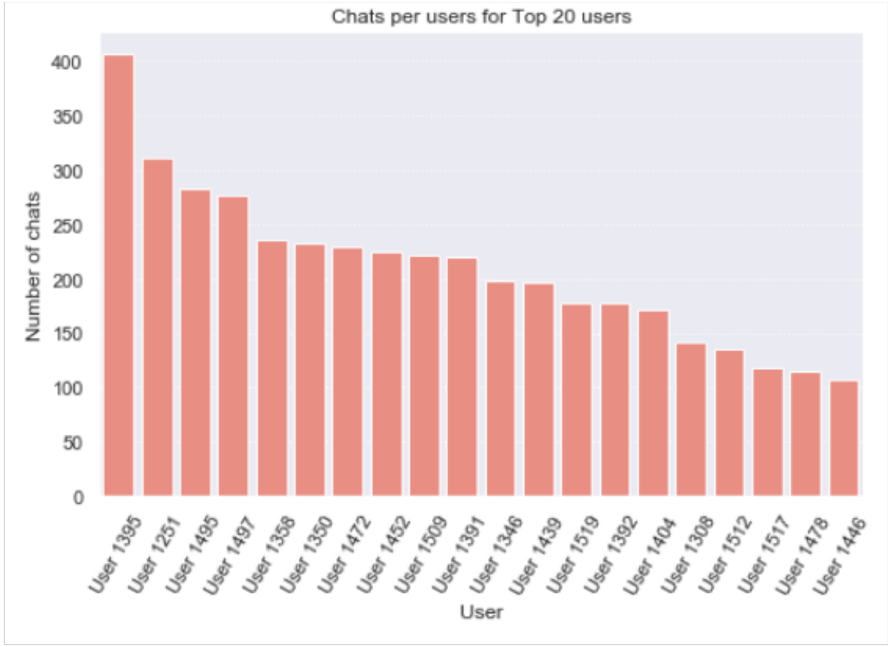
Plot output is Matplotlib – same manipulation.

Slightly simpler API / data access.

The Bar Plot - Seaborn




```
sns.barplot(x='user_id', y='number_chats',
            color='salmon', data=chats_per_user[0:20])
plt.xticks(rotation=60)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
plt.gca().yaxis.grid(linestyle=':')
```

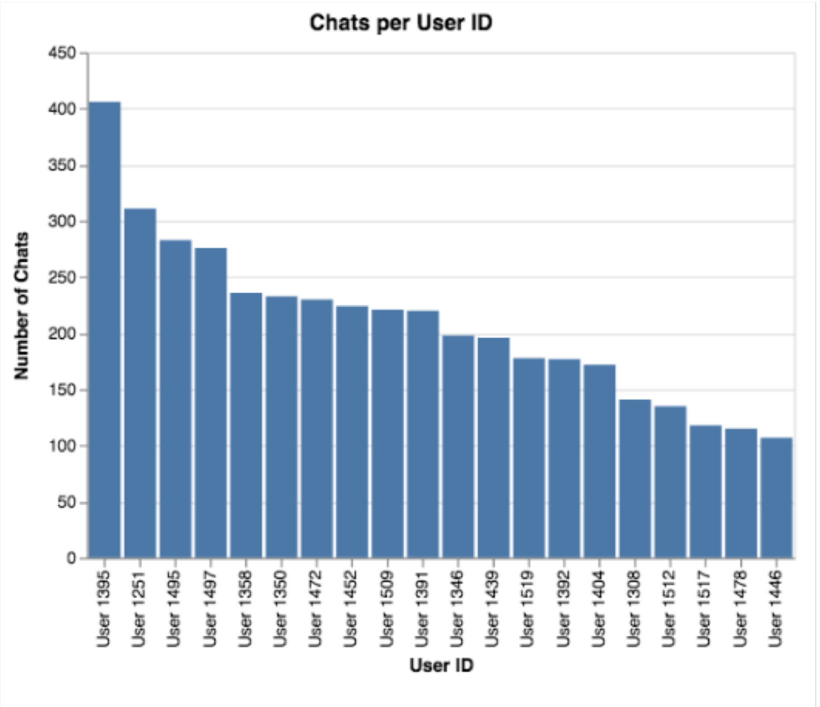


Simpler data access again.

Same Matplotlib formatting functions

The Bar Plot - Altair

```python
bars = alt.Chart(
    chats_per_user[0:20], title='Chats per User ID').mark_bar().encode(
    x=alt.X(
        'user_id',
        sort=alt.EncodingSortField(field='number_chats',
                                   op='sum',
                                   order='descending'),
        axis=alt.Axis(title='User ID')),
    y=alt.Y(
        'number_chats',
        axis=alt.Axis(title='Number of Chats')
    )
)
bars
```



Not Matplotlib-based – very different syntax and formatting.

Ordering was difficult here.

Only one command for everything. JSON format behind.

The Bar Plot - Altair

```python
bars = alt.Chart(
    chats_per_user[0:20], title='Chats per User ID').mark_bar().encode(
    x=alt.X(
        'user_id',
        sort=alt.EncodingSortField(field='number_chats',
                                   op='sum',
                                   order='descending'),
        axis=alt.Axis(title='User ID')),
    y=alt.Y(
        'number_chats',
        axis=alt.Axis(title='Number of Chats')
    )
)
bars
```

Not Matplotlib-based – very different syntax and formatting.

Ordering was difficult here.

Only one command for everything. JSON format behind.


```json
{
  "config": {"view": {"width": 400, "height": 300}},
  "data": {"name": "data-84c58b571b3ed04edf7929613936b11e"},
  "mark": "bar",
  "encoding": {
    "x": {
      "type": "nominal",
      "axis": {"title": "User ID"},
      "field": "user_id",
      "sort": {"op": "sum", "field": "number_chats", "order": "descending"}
    },
    "y": {
      "type": "quantitative",
      "axis": {"title": "Number of Chats"},
      "field": "number_chats"
    }
  },
  "selection": {
    "selector001": {
      "type": "interval",
      "bind": "scales",
      "encodings": ["x", "y"],
      "on": "[mousedown, window:mouseup] > window:mousemove!",
      "translate": "[mousedown, window:mouseup] > window:mousemove!",
      "zoom": "wheel!",
      "mark": {"fill": "#333", "fillOpacity": 0.125, "stroke": "white"},
      "resolve": "global"
    }
  },
  "title": "Chats per User ID",
  "$schema": "https://vega.github.io/schema/vega-lite/v2.6.0.json",
  "datasets": {
    "data-84c58b571b3ed04edf7929613936b11e": [
      {"user_id": "User 1395", "number_chats": 406},
      {"user_id": "User 1251", "number_chats": 311},
      {"user_id": "User 1495", "number_chats": 283},
      {"user_id": "User 1497", "number_chats": 276},
      {"user_id": "User 1358", "number_chats": 236},
```

The Bar Plot

matplotlib

```
top_n = 20
plt.bar(x=range(top_n),
        height=chats_per_user[0:top_n]['number_chats'])
plt.xticks(range(top_n), chats_per_user[0:top_n]['user_id'],
           rotation=60)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
ax = plt.gca()
ax.yaxis.grid(linestyle=':')
```

seaborn

```
sns.barplot(x='user_id', y='number_chats',
            color='salmon', data=chats_per_user[0:20])
plt.xticks(rotation=60)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
plt.gca().yaxis.grid(linestyle=':')
```

pandas

$y_it = \beta' x_{it} + \mu_i + \epsilon_{it}$

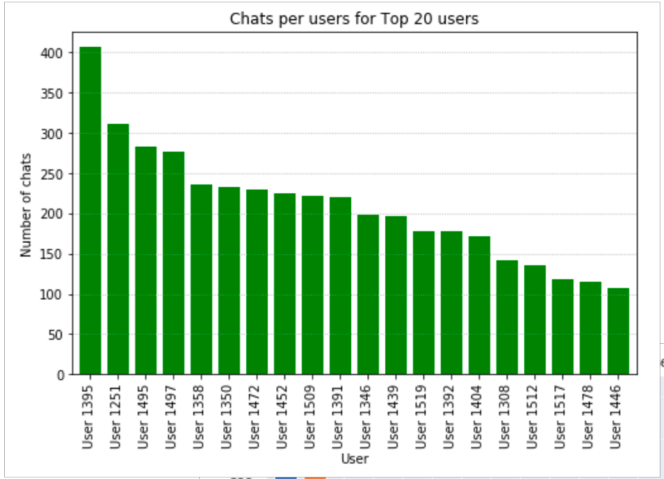
```
chats_per_user[0:20].plot(
    x='user_id', y='number_chats',
    kind='bar', legend=False, color='green',
    width=0.8
)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
plt.gca().yaxis.grid(linestyle=':')
```

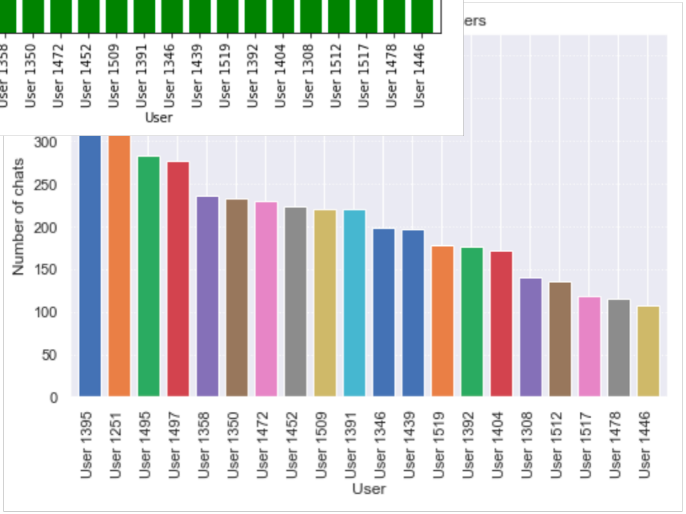
```
bars = alt.Chart(
    chats_per_user[0:20], title='Chats per User ID').mark_bar().encode(
    x=alt.X(
        'user_id',
        sort=alt.EncodingSortField(field='number_chats',
                                   op='sum',
                                   order='descending'),
        axis=alt.Axis(title='User ID')),
    y=alt.Y(
        'number_chats',
        axis=alt.Axis(title='Number of Chats')
    )
)
```

Prettier Pandas Plots

```
import seaborn
sns.set()
chats_per_user[0:20].plot(
    x='user_id', y='number_chats',
    kind='bar', legend=False,
    width=0.8
)
plt.ylabel("Number of chats")
plt.xlabel("User")
plt.title("Chats per users for Top 20 users")
plt.gca().yaxis.grid(linestyle=':')
```

Seaborn styles are applied to all matplotlib plots – Cheat your way to nicer looking Pandas Plots!





More Challenging Bar Plot

For the top 20 agents, what was the split of the top websites?

We want a 'stacked bar' for this visualisation.

```
# Only include the top 5 Websites
chats_per_user = data.groupby(
    ['user_id', 'website_summary']
)['chat_id'].count().reset_index()
chats_per_user.columns = ['user_id', 'website', 'number_chats']

# Only include the top 20 chatters again
chats_per_user = \
    chats_per_user[chats_per_user['user_id'].isin(
        data['user_id'].value_counts().index[0:20].tolist()
    )]

# Sort the data with agents with most chats first
# First get the total chats:
temp = chats_per_user.\
    groupby('user_id')['number_chats'].\
    sum().\
    reset_index().\
    sort_values('number_chats', ascending=False)
temp.columns=['user_id', 'total_chats']
# Now merge these sums back onto the original data
chats_per_user = pd.merge(
    chats_per_user, temp, on='user_id'
).sort_values('total_chats', ascending=False)
```

	user_id	website	number_chats	total_chats
35	User 1395	Other	28	406
36	User 1395	Website A	114	406
37	User 1395	Website B	96	406
38	User 1395	Website C	102	406
39	User 1395	Website D	66	406

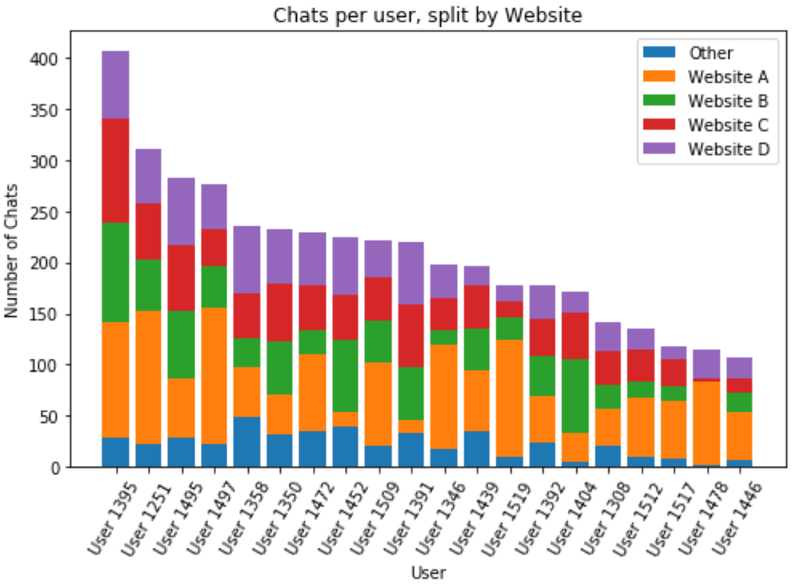
```
chats_per_user.shape
```

```
(100, 4)
```

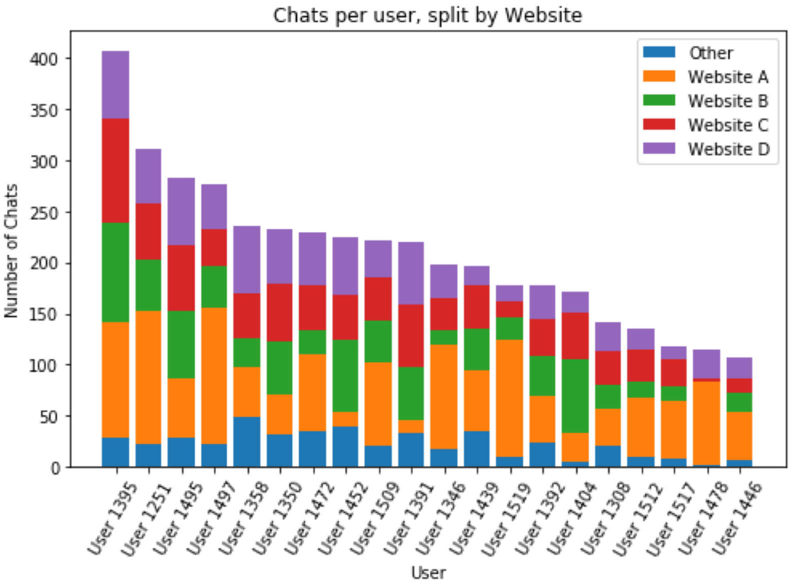
```
chats_per_user['website'].value_counts()
```

```
Website A     20
Website B     20
Website C     20
Other         20
Website D     20
Name: website, dtype: int64
```

Stacked Bar - Matplotlib

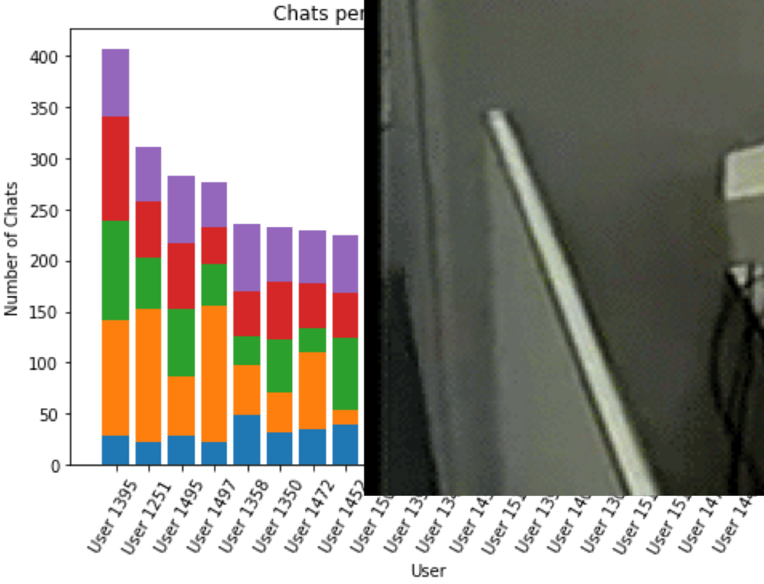


Stacked Bar - Matplotlib




```
x_position = list(range(20))
# For matplotlib, we manually track where bars start/finish
bars_bottom = np.zeros(20)

for website_string in chats_per_user['website'].unique():
    # Draw a set of bars for this website
    bar_height = chats_per_user[
        chats_per_user['website'] == website_string]['number_chats']
    plt.bar(
        x=x_position,
        height=bar_height,
        bottom=bars_bottom
    )
    # Remember where the bottom of these bars are
    bars_bottom = bars_bottom + bar_height.values

# Unique preserves order - so it's safe here.
plt.xticks(x_position, chats_per_user.user_id.unique(), rotation=60)
plt.xlabel('User')
plt.ylabel('Number of Chats')
plt.title('Chats per user, split by Website')
plt.legend(chats_per_user['website'].unique())
```

Stacked Bar - Matplotlib



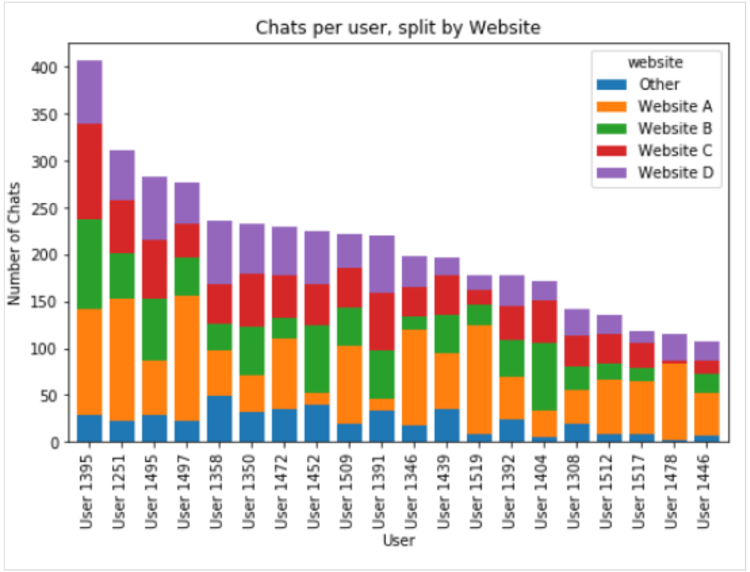
```
x_position = list(range(20))
rt/finish
ique():
g]['number_chats']
ue(), rotation=60)
plt.legend(chats_per_user['website'].unique())
```

Stacked Bar - Pandas

```python
# Stacked bar in Pandas requires a data transformation
plot_data = chats_per_user.pivot(index='user_id',
                                 columns='website',
                                 values='number_chats')
# Wide format - sort by total chats by user.
plot_data = plot_data.iloc[plot_data.sum(axis=1).argsort()[::-1], :]
plot_data.head()
```

	user_id	website	number_chats	total_chats
35	User 1395	Other	28	406
36	User 1395	Website A	114	406
37	User 1395	Website B	96	406
38	User 1395	Website C	102	406
39	User 1395	Website D	66	406

website	Other	Website A	Website B	Website C	Website D
user_id					
User 1395	28	114	96	102	66
User 1251	22	131	49	55	54
User 1495	28	59	66	63	67
User 1497	23	133	40	36	44
User 1358	49	49	28	43	67

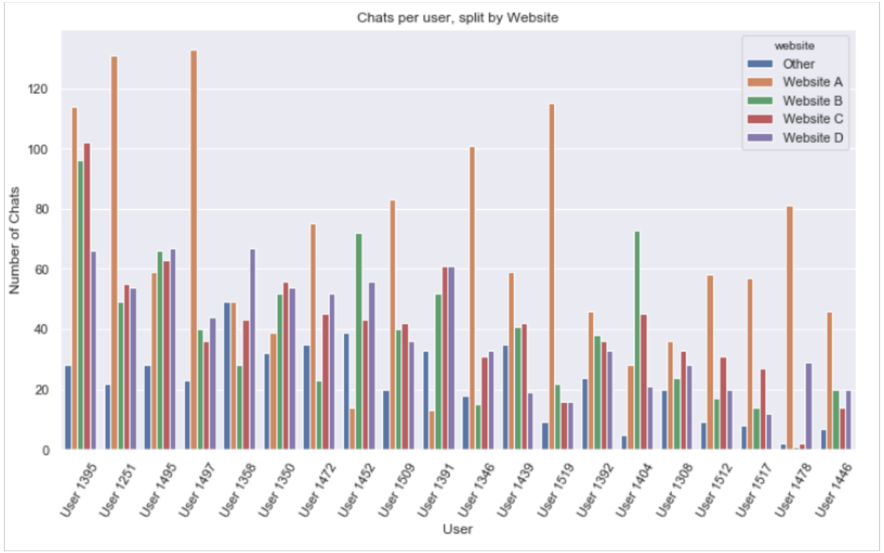



```python
plot_data.plot(kind='bar', stacked=True, width=0.8)
plt.xlabel('User')
plt.ylabel('Number of Chats')
plt.title('Chats per user, split by Website')
```

Plotting code is simple, but data manipulation required.

Stacked Bar - Seaborn

seaborn



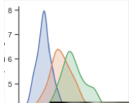

```
sns.set()
sns.barplot(x='user_id',
            y='number_chats',
            hue='website',
            data=chats_per_user)
plt.xlabel('User')
plt.ylabel('Number of Chats')
plt.title('Chats per user, split by Website')
plt.xticks(rotation=60)
```

Elegant API, simple code structure, but …

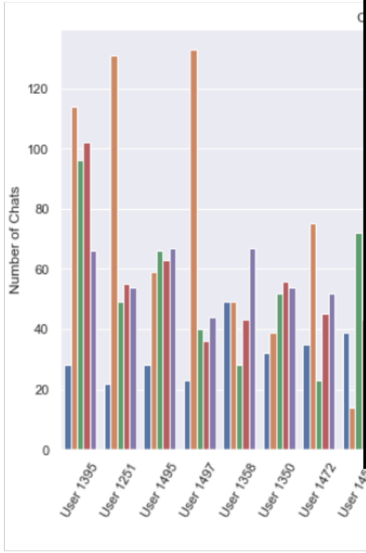
…embarrassingly…

no stacked-bar chart support!

Stacked Bar - Seaborn



se

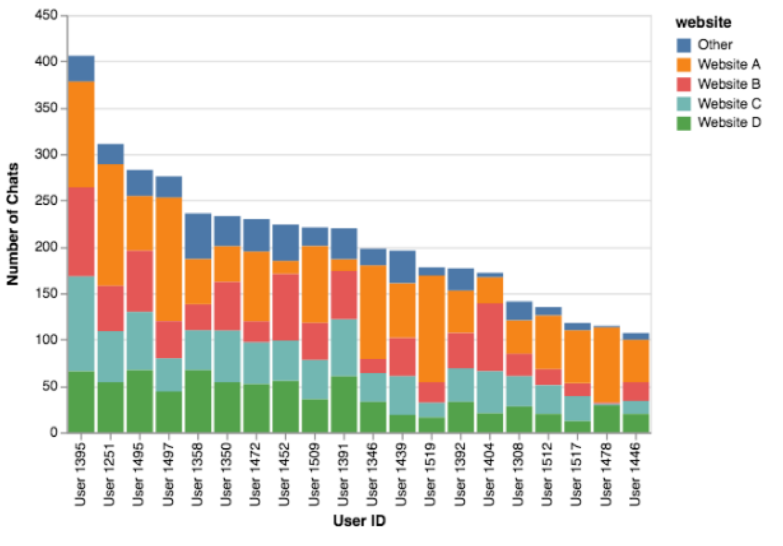


```
sns.set()
sns.barplot(x='user_id',
…
…y Website')
```

…ucture,

…t!

Stacked Bar - Altair

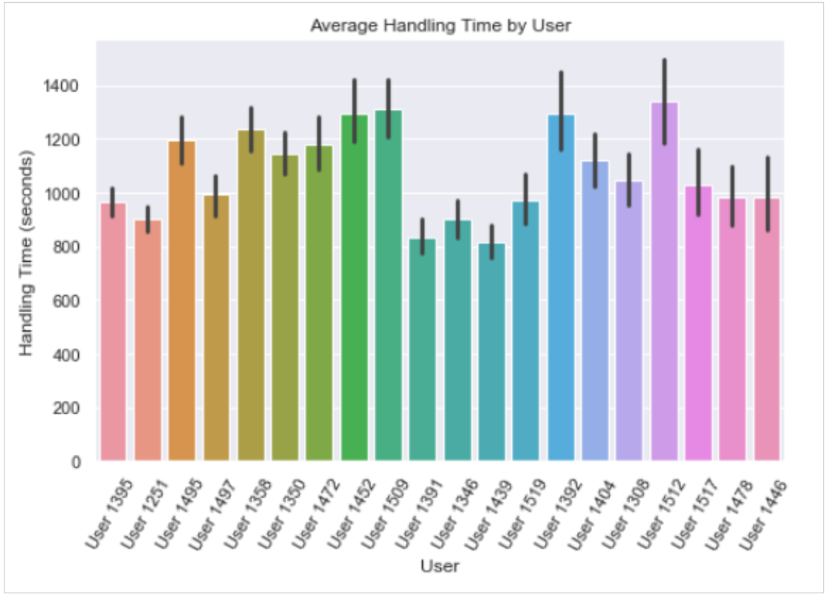



```
alt.Chart(
    chats_per_user      # choose dataset here.
).mark_bar().encode(
    x=alt.X(            # user_id on X Axis
        'user_id',
        sort=alt.EncodingSortField(field='number_chats',
                                   op='sum',
                                   order='descending'),
        axis=alt.Axis(title='User ID')
    ),
    y=alt.Y('sum(number_chats)', # Sum of chats on y-axis
            axis=alt.Axis(title='Number of Chats')),
    color='website'
)
```

Simple output, short code.

Some issues around data storage, JSON formats, and sorting is difficult.

Seaborn - Estimators

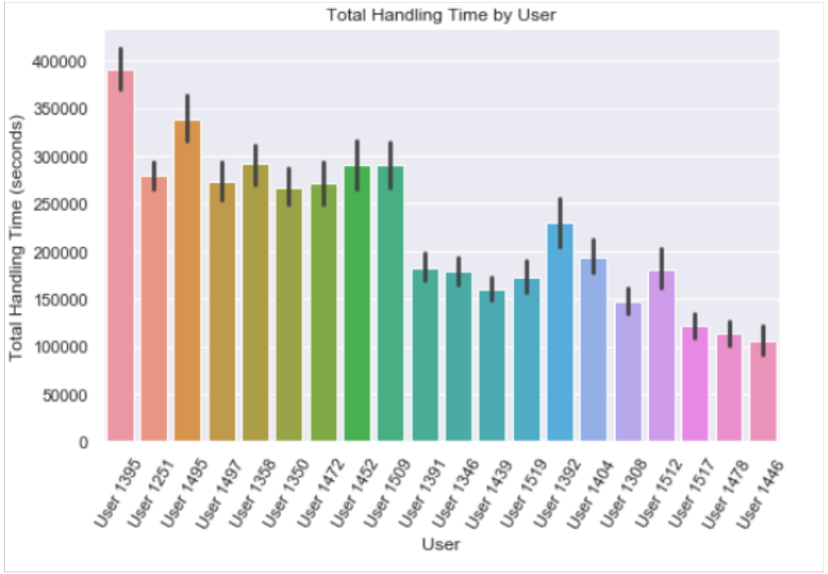



```
sns.barplot(
    x='user_id', y='handling_time', estimator=np.mean,
    data=data,
    order=data['user_id'].value_counts().index.tolist()[0:20]
)
plt.xlabel('User')
plt.ylabel('Handling Time (seconds)')
plt.title('Average Handling Time by User')
plt.xticks(rotation=60)
```

Calculations done as part of plotting – no previous data manipulations.

Separation of data and visualisation code.

Seaborn - Estimators




```
sns.barplot(
    x='user_id', y='handling_time', estimator=np.sum,
    data=data,
    order=data['user_id'].value_counts().index.tolist()[0:20]
)
plt.xlabel('User')
plt.ylabel('Total Handling Time (seconds)')
plt.title('Total Handling Time by User')
plt.xticks(rotation=60)
```

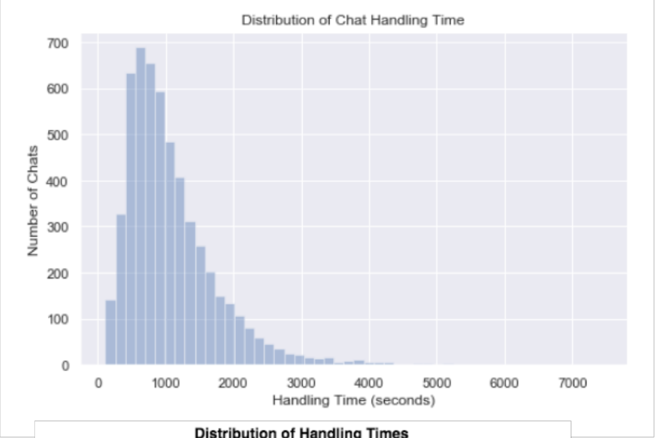
Very simple to change estimator function to calculate different statistics.

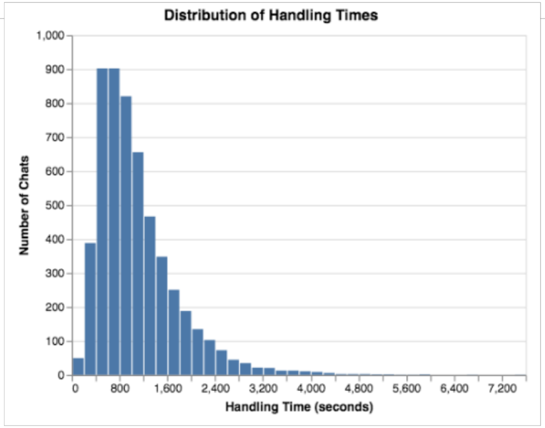
Similar functionality available in Altair

Histograms

Histograms

All libraries good at univariate distribution visualisations.






```python
data['handling_time'].hist(bins=50)
plt.xlabel('Handling Time (seconds)')
plt.ylabel('Number of Chats')
plt.title("Distribution of Chat Handling Time")
```

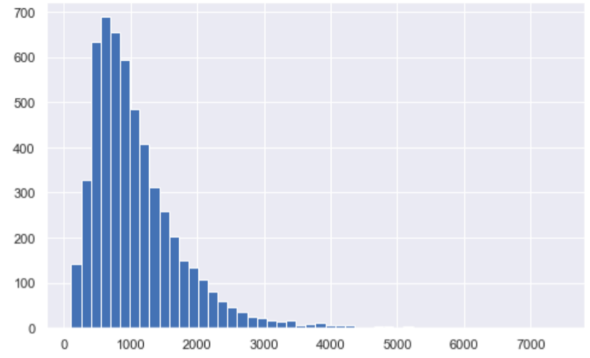
```python
sns.distplot(a=data['handling_time'],
             hist=True,
             kde=False,
             bins=50)

plt.xlabel('Handling Time (seconds)')
plt.ylabel('Number of Chats')
plt.title("Distribution of Chat Handling Time")
```

```python
alt.Chart(data,
          title='Distribution of Handling Times'
         ).mark_bar().encode(
    x=alt.X(
        'handling_time', bin={'maxbins': 50},
        axis=alt.Axis(title='Handling Time (seconds)')),
    y=alt.Y(
        'count()',
        axis=alt.Axis(title='Number of Chats')
    )
)
```

Histograms

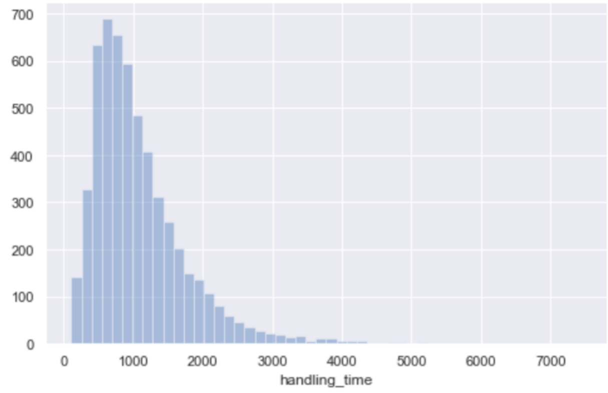
```python
data['handling_time'].hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x118526438>
```

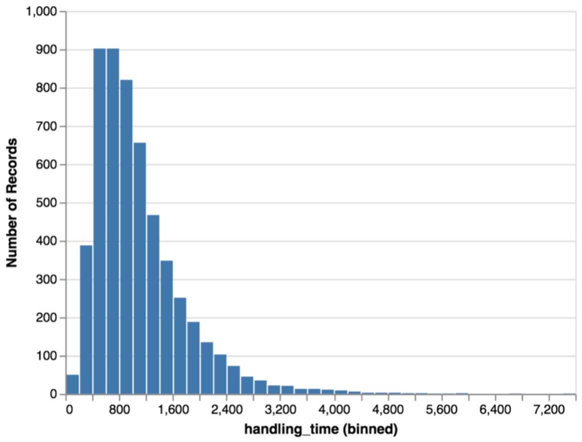



```python
sns.distplot(a=data['handling_time'],
             hist=True, kde=False, bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x118a3c400>
```



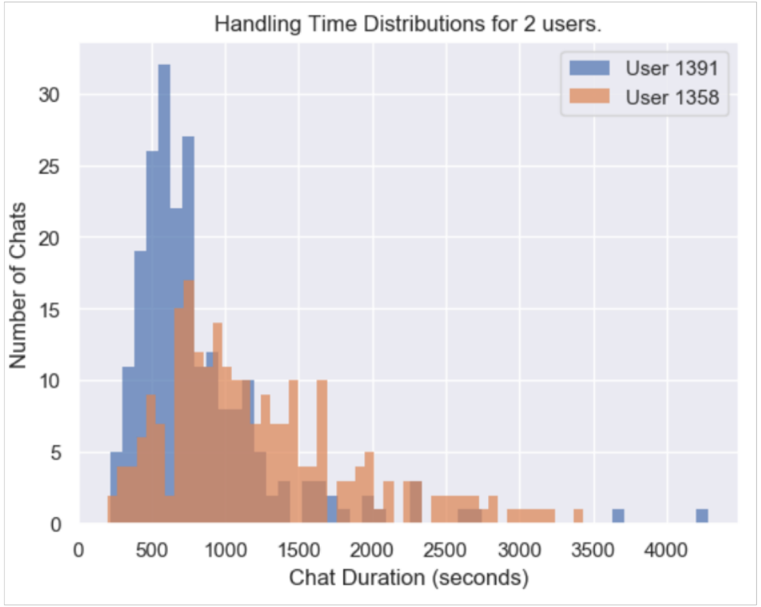

```python
alt.Chart(data).mark_bar().encode(
    x=alt.X('handling_time', bin={'maxbins': 50}),
    y='count()'
)
```



Histograms

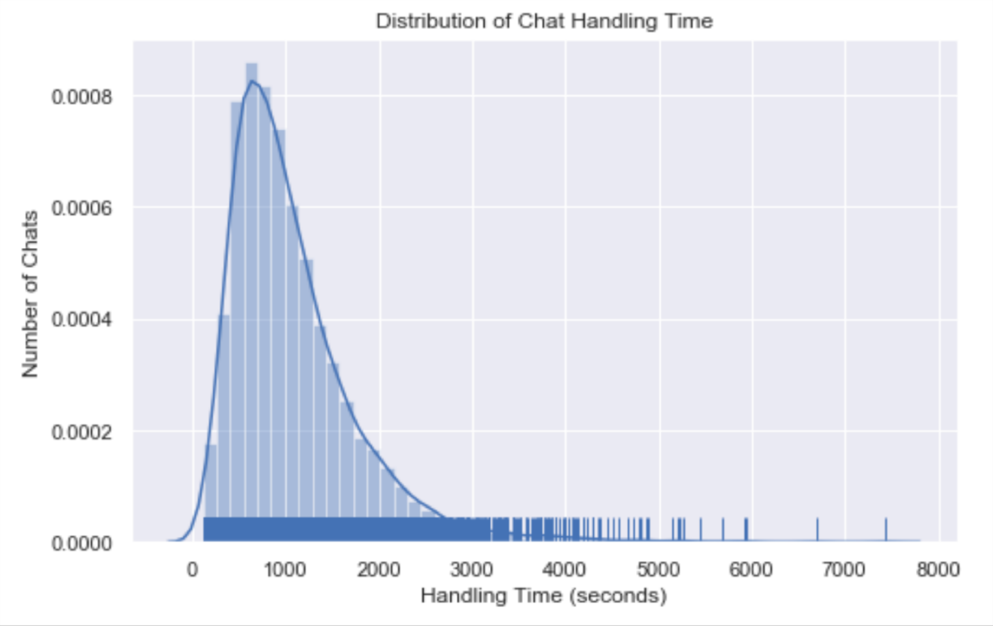
Layering / comparison achieved unfortunately by building up the histograms in place.

```
data.loc[data['user_id'] == 'User 1391', 'handling_time'].hist(
    bins=50, alpha=0.7, label='User 1391')
data.loc[data['user_id'] == 'User 1358', 'handling_time'].hist(
    bins=50, alpha=0.7, label='User 1358')
plt.legend()
plt.title("Handling Time Distributions for 2 users.")
plt.ylabel("Number of Chats")
plt.xlabel("Chat Duration (seconds)")
```



Histograms - Seaborn

```
sns.distplot(a=data['handling_time'],
             hist=True,
             kde=True,
             rug=True,
             bins=50)
plt.xlabel('Handling Time (seconds)')
plt.ylabel('Number of Chats')
plt.title("Distribution of Chat Handling Time")
```



Some really nice options for impressive and informative hints on Seaborn graphs.

Scatter Plots - Pandas

Scatter Plots - Pandas

pandas

Pandas: Good for quick single-coloured scatter visualisations. Messy with multiple categories.

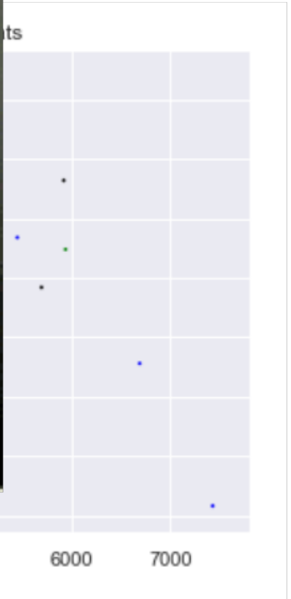
```
colours = {
    "Website A": 'red',
    "Website B": 'blue',
    "Website C": 'green',
    "Website D": 'black',
    "Other": 'orange'
}
data.plot(
    x='handling_time',
    y='number_messages',
    kind='scatter',
    c=data['website_summary'].apply(lambda x: colours[x])
)
plt.xlabel("Handling Time (seconds)")
plt.ylabel("Messages in chat")
plt.title("Handling time vs Messages in chats")
```

Scatter Plots - Pandas

pandas

Pandas: Good for quick single-coloured scatter visualisations.
Messy with multiple categories

```
colours = {
    "Website A": 'red
    "Website B": 'blu
    "Website C": 'gre
    "Website D": 'bla
    "Other": 'orange'
}
data.plot(
    x='handling_time'
    y='number_message
    kind='scatter',
    c=data['website_s
)
plt.xlabel("Handling
plt.ylabel("Messages in chat")
plt.title("Handling time vs Messages in chats")
```



Scatter Plots - Seaborn

seaborn

Seaborn / Altair: Better higher level representation, and better for multi-category scatters.

```python
sns.lmplot(
    x='handling_time',
    y='number_messages',
    hue='website_summary',
    data=data,
    fit_reg=False, # SNS fits a regression line by default
    scatter_kws={"s": 1}, # Size of point on figure
)
plt.title("Handling Time vs Number of Messages")
plt.xlabel("Handling Time (seconds)")
plt.ylabel("Number of Messages")
```

Scatter Plots - Altair

Seaborn / Altair: Better higher level representation, and better for multi-category scatters.

```
alt.Chart(
    data,
    title='Handling Time vs Number of Messages'
).mark_point().encode(
    x=alt.X('handling_time', title='Handling Time (seconds)'),
    y=alt.Y('number_messages', title='Number of Messages'),
    color='website_summary',
    tooltip='user_id'
).interactive()
```

Line Plots

Line Plots

Plot chats per language over time

Pandas: Needs data manipulation, simple thereafter.

```
plot_data = data\
    .groupby(['date', 'language'])['chat_id']\
    .count()\
    .reset_index()\
    .sort_values('date')
plot_data.columns = ['date', 'language', 'number_chats']
```

```
plot_data.head()
```

	date	language	number_chats
0	2018-05-30	English	24
1	2018-05-31	English	52
2	2018-06-01	English	37
3	2018-06-02	English	4
4	2018-06-03	English	9

Line Plots

pandas

Pandas: Needs data manipulation, simple thereafter.

```
plot_data = data\
    .groupby(['date', 'language'])['chat_id']\
    .count()\
    .reset_index()\
    .sort_values('date')
plot_data.columns = ['date', 'language', 'number_chats']
```

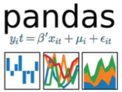
```
plot_data.head()
```

```
# Pandas again, for multiple lines requires data manipulation
pandas_plot = plot_data.pivot(
    index='date',
    columns='language',
    values='number_chats'
).fillna(0)
```

	date	language	number_chats
0	2018-05-30	English	24
1	2018-05-31	English	52
2	2018-06-01	English	37
3	2018-06-02	English	4
4	2018-06-03	English	9

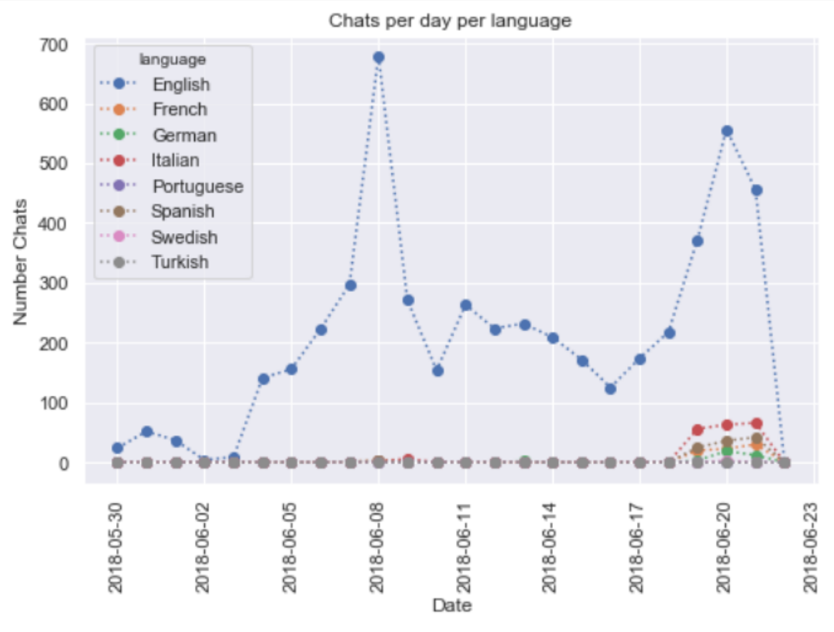
language	English	French	German	Italian	Portuguese	Spanish	Swedish	Turkish
date								
2018-05-30	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-05-31	52.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-01	37.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-02	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-03	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-04	141.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0
2018-06-05	156.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-06	223.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-06-07	297.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0

Line Plots



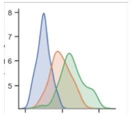
Pandas: Needs data manipulation, simple thereafter.

```python
# Pandas again, for multiple lines requires data manipulation
pandas_plot = plot_data.pivot(
    index='date',
    columns='language',
    values='number_chats'
).fillna(0)
# Actual plotting command is simple
pandas_plot.plot(kind='line', marker='o', linestyle=':')
# Formatting arguments:
plt.xlabel("Date")
plt.xticks(rotation=90)
plt.ylabel("Number Chats")
plt.title("Chats per day per language")
```

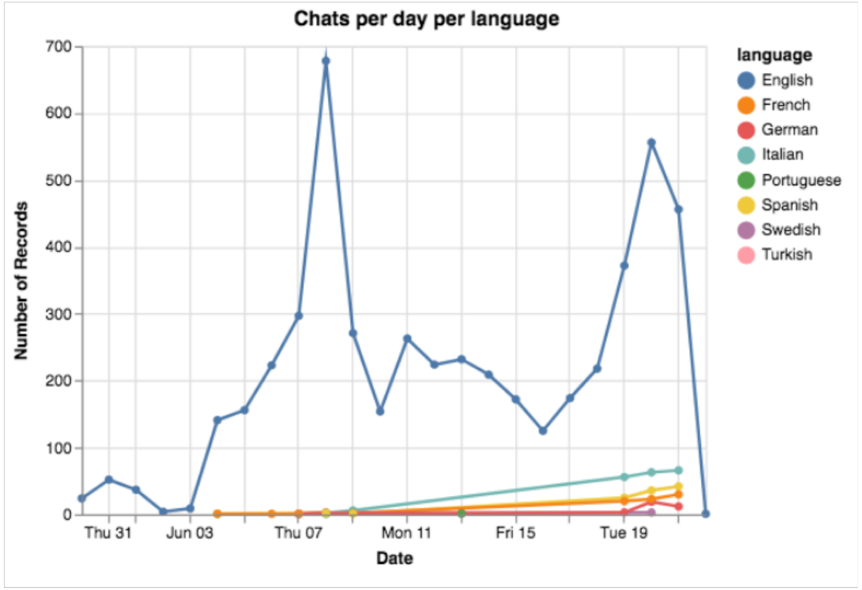


Line Plots

Seaborn/Altair:
Operate directly on raw data




```python
# Seaborn can act directly on the raw data!
sns.lineplot(
    x='date',
    y='chat_id',
    hue='language',
    data=data,
    estimator=len,
    marker='o'
)
# Formatting of plot output
plt.xlabel("Date")
plt.xticks(rotabtion=90)
plt.ylabel("Number Chats")
plt.title("Chats per day per language")
```



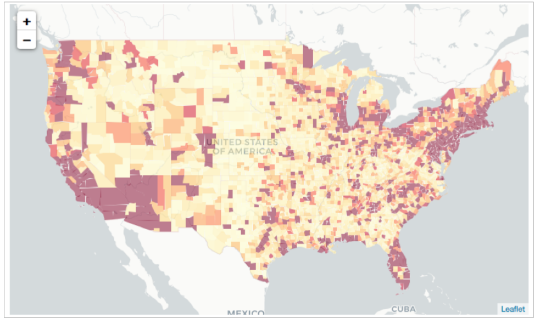

```python
# Altair works directly on the raw data
alt.Chart(
    data,
    title='Chats per day per language'
).mark_line(point=True).encode(
    x=alt.X('date', title='Date'),
    y='count(chat_id)',
    color='language'
)
```

More Options!

Geospatial Viz

Folium: Generate interactive maps using leaflet.js

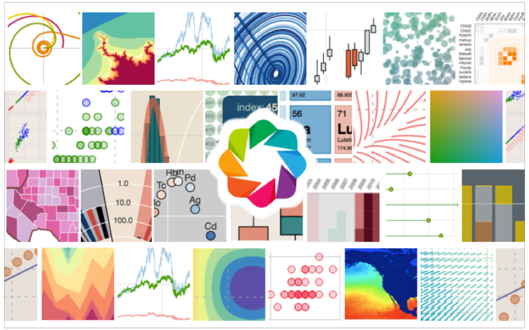
Matplotlib: Basemap plugin



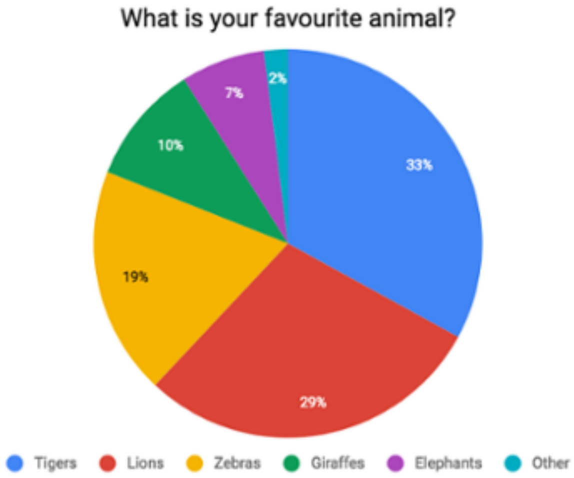
Interactive Plots

Bokeh: Makes visualisations for web browser interaction.

Plotly: Online visualisations – runs by default in cloud



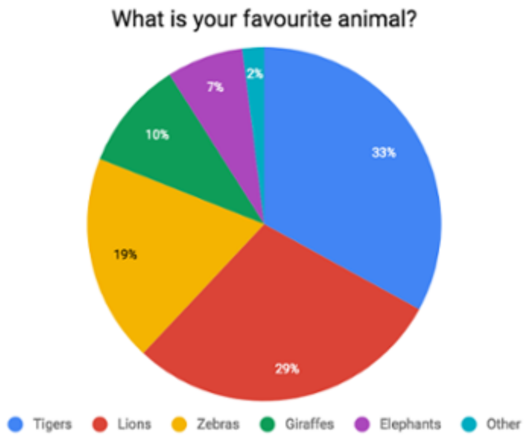
What to Avoid – Angles?

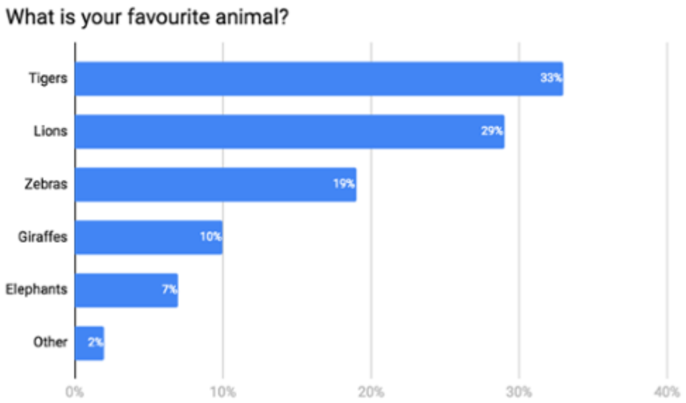


Pie Charts: Radial angle for comparison. Humans are very bad at accurate radial comparisons – we've evolved for speedy length / distance comparisons.

https://blog.funnel.io/why-we-dont-use-pie-charts-and-some-tips-on-better-data-visualizations

What to Avoid – Angles?

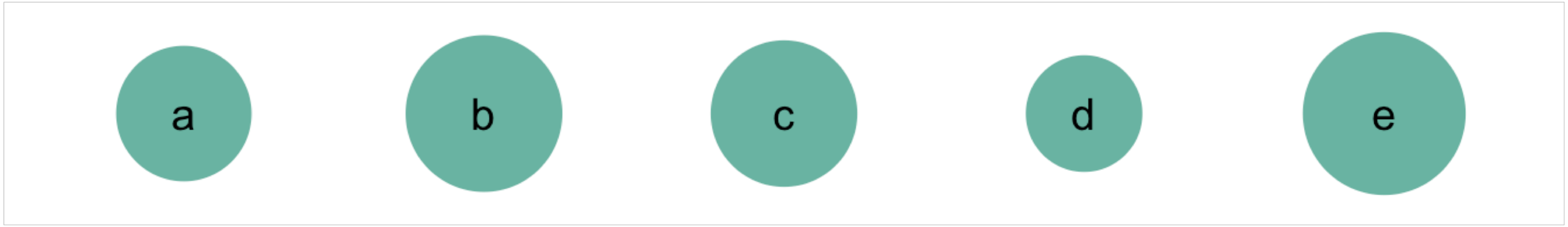




Pie Charts: Radial angle for comparison. Humans are very bad at accurate radial comparisons – we've evolved for speedy length / distance comparisons.

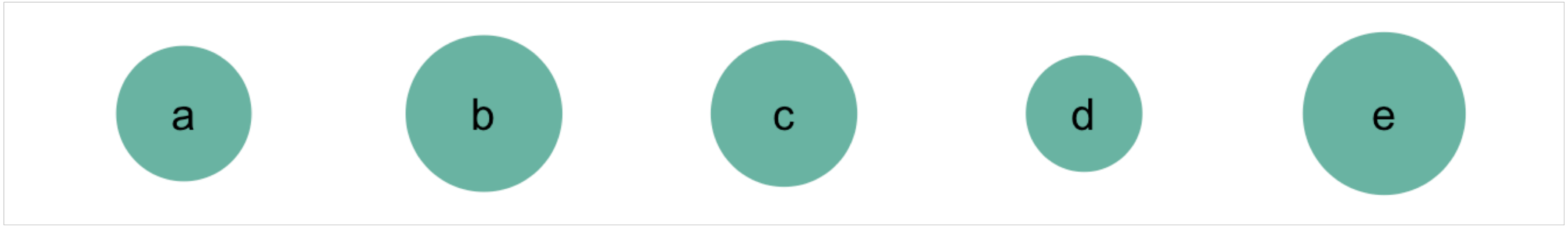
https://blog.funnel.io/why-we-dont-use-pie-charts-and-some-tips-on-better-data-visualizations

What to Avoid – Area?

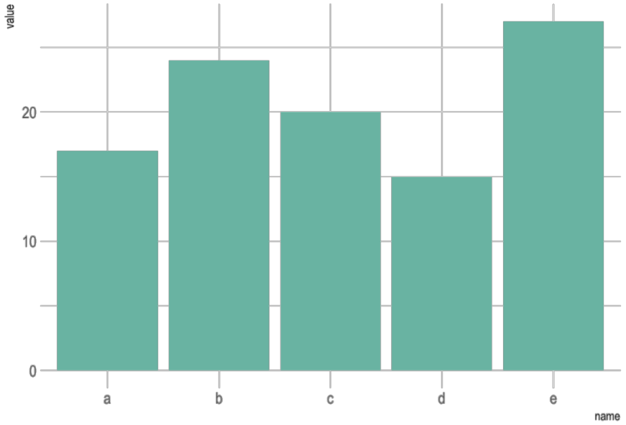


Area: We're bad at area – rank these bubbles by area, and compare them relative to each other.

What to Avoid – Area?

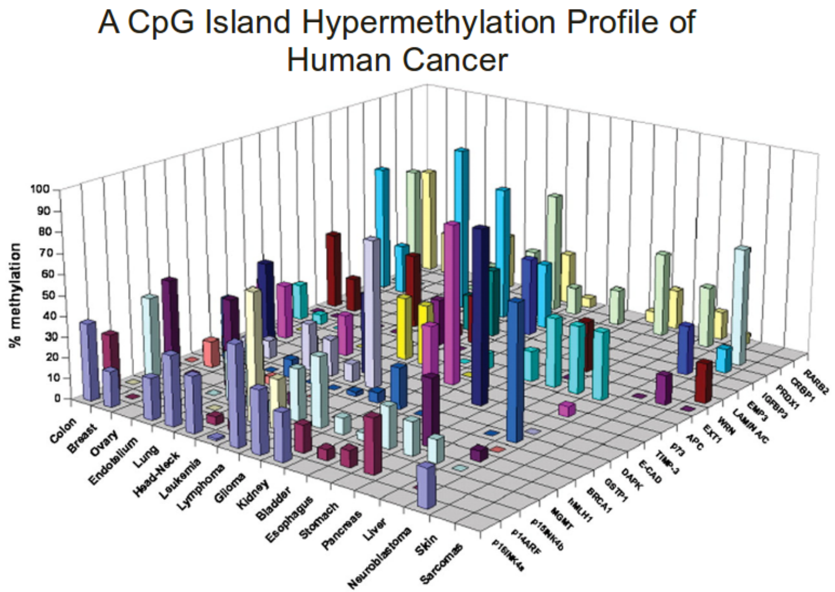


Area: We're bad at area – rank these bubbles by area, and compare them relative to each other.

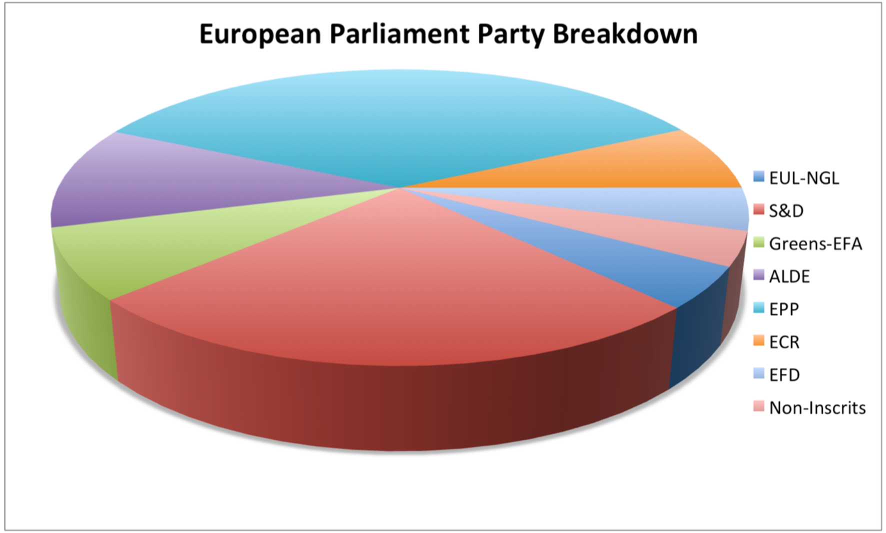


https://www.data-to-viz.com/caveat/area_hard.html

What to Avoid – 3d?



Hum. Mol. Genet. (2007) 16:R50-59



3d: In general, 3D is “fake fancy”. Impractical but gee-whizz – avoid!

Caveat: Interactive Scatters?

Conclusions

Wide variety of tools available in Python.

Get familiar with Pandas syntax for quick & simple exploration, and use with Seaborn themes.

Learn one more high-level library in detail – Seaborn or Altair for publication of output and more flexibility

"Simplicity is the ultimate sophistication"

Leonardo Da Vinci

COMPLEX DECISIONS SIMPLIFIED

Data Visualisation in Python

Quick and easy routes to plotting magic

Shane Lynn PhD
@shane_a_lynn | @TeamEdgeTier

More?

Resources

Tour of Python's Data Landscape
https://dsaber.com/2016/10/02/a-dramatic-tour-through-pythons-data-visualization-landscape-including-ggplot-and-altair/

Python Graph Gallery
https://python-graph-gallery.com/

From Data to Viz
https://www.data-to-viz.com/