Visualization Reference Sheet

Altair (Python)

Create a data-aware chart using `alt.Chart(df)`.

Core Methods

- `alt.Chart(df)`: Creates the initial data-aware chart object.
- `alt.mark_point()` / `mark_line()` / `mark_area()` / `mark_bar()` / `mark_rect()` / `mark_boxplot()`: Graphical mark. `mark_rect()` is for 2D histogram/heatmap.
- `alt.encode(x='col1', color='col2', ...)`: Specify how data columns are encoded as visual channels.
- **Special Encoding Strings**: Use strings like `'count()'`, `'mean(col1)'`, `'sum(col1)'`, etc., for aggregate statistics.
- `alt.facet('col2', columns=1)`: Create multiple views (panels).
- `alt.properties(width=100, height=100)`: Set chart dimensions/properties.
- `alt.resolve_scale(y='independent')`: Control shared/independent scales.
- `alt.transform_density('col')`: Compute the **KDE** (Kernel Density Estimate).
- `alt.mark_errorbar(extent='ci')` / `mark_errorband()`: Show error bars/bands (`'ci'` or `'stdev'`).
- `alt.transform_regression('x', 'y', groupby=['col'])`: Add a **regression line** (linear or other via `method`).
- `alt.transform_loess('x', 'y', groupby=['col'])`: Add a **LOESS** curve.
- `alt.scale(scheme='colormap name')`: Apply a specific **colorscheme**.
- **Layering / Concatenation**: `chart1 + chart2` (Layer), `chart1 & chart2` (Vertical), `chart1 | chart2` (Horizontal).
- **Interactivity**: `alt.interactive()`, `alt.selection_interval()`, `alt.selection_point()`, `alt.condition(...)`.

Encoding Helper Methods

Passed to helpers like `alt.Color`, `alt.X`, etc.

- `.bin(maxbins=30)`: Group values into bins/buckets.
- `.scale(zero=False, range=(5, 40))`: Modify scale properties (e.g., range for size).
- `.stack(False)`: Control if marks (e.g. bars/areas) should stack.
- `.title('Col 1')`: Add a title to the axis.
- `.sort('-x')`: Sort/reorder based on another value (e.g., `'-x'` for descending 'x').
- `.axis(format=)`: Format labels (`'e'` for scientific, `'s'` for SI units, `' s'` to remove trailing zeros).
- `.axis(tickCount = )`: Set the number of ticks.

Altair (Python) Examples

Simple Chart:

```
alt.Chart(df).mark_point().encode(
    x='col1',
    y=alt.Y('col2').title('Col 2')
)
```

Density Plot:

```
alt.Chart(df).transform_density(
    'col1', groupby=['col2'], as_=['col1', 'density']
).mark_area().encode(
    x='col1', y=alt.Y('density:Q').stack(False),
    color='col2'
)
```

2D Histogram:

```
alt.Chart(diamonds).mark_rect().encode(
    alt.X('carat').bin(maxbins=40),
    alt.Y('price').bin(maxbins=40),
    alt.Color('count()'),
    alt.Size('count()')
)
```

Scatter Plot Matrix (`alt.repeat`):

```
alt.Chart(df).mark_point().encode(
    alt.X(alt.repeat('row')).type('quantitative'),
    alt.Y(alt.repeat('column')).type('quantitative')
).repeat(
    column=['col1', 'col2', 'col3'],
    row=['col1', 'col2', 'col3'],
)
```

Interactivity:

```
brush = alt.selection_interval()
alt.Chart(df).mark_circle().encode(
    x='xcol', y='ycol',
    color = alt.condition(brush, 'column name', alt.value('color
        code'))
).add_params(brush)
```

ggplot (R)

Create a data-aware chart using `ggplot(df)`.

Core Functions

- `ggplot(df)`: Creates the initial data-aware chart object.
- `aes(x=col1, color=col2, ...)`: Specify visual **aesthetics**.
- `geom_point()` / `geom_line()` / `geom_area()` / `geom_bar()` / `geom_histogram()` / `geom_boxplot()` / `geom_density()` / `geom_violin()`: Geometric marks.
- `geom_*(stat='summary', fun=mean)`: Compute aggregate statistics (e.g., `mean`).
- `facet_wrap( col1, ncol=1, scales='free')`: Create multiple views.
- `reorder(col1, -col2)`: Sort/reorder a column based on another value (e.g., `-col2` for descending).
- `ggtitle(title)`: Adds a title.
- `labs(x = 'Col 1', fill = 'Col 2')`: Modifies labels.
- `geom_() + geom_()`: Layer multiple geoms.
- `geom_bin2d()`: **2D histogram**. `geom_hex()` for hexagonal.
- `GGally::ggpairs(df)`: **Scatterplot matrix** (requires `select_if(is.numeric)`).
- `geom_smooth(se = FALSE, method = 'lm')`: Add a smooth curve. `se` controls the confidence interval.
- `cowplot::plot_grid(p1, p2, labels)`: Arrange multiple plots (requires `cowplot` package).

Scale and Theme Modifications

- `scale_size(range = c(5, 40))`: Change the size scale range.
- `theme(text = element_text(size = 24))`: Modify non-data components (e.g., text size).
- `scale_y_continuous(labels, breaks, limits)`: Modify Y-axis properties. Labels can use `scales::label_scientific()`, `scales::label_dollar()`.
- `scale_fill_continuous(labels, trans='reverse')`: Control color fill scale. `trans='reverse'` reverses the gradient.
- `scale_color_brewer(palette='Dark2')`: Apply RColorBrewer palettes.
- `scale_color_viridis_c(trans, direction=1)`: Apply Viridis scale (`direction=-1` reverses the gradient).

ggplot (R) Examples

Simple Chart:

```
library(tidyverse)
my_data |> ggplot(aes(x = col1, y = col2)) +
    geom_point()
```

Density Plot (Geom Shortcut):

```
ggplot(df, aes(x = col1, color = col2)) +
    geom_density()
```

2D Histogram:

```
ggplot(df, aes(x = xcol, y = ycol)) +
    geom_bin2d()
```

Hiding an Axis:

```
ggplot(df, aes(x = xcol, y = ycol)) +
    geom_point() +
    theme(axis.title.x = element_blank(),
          axis.text.x = element_blank(),
          axis.ticks.x = element_blank())
```

Scatter Plot Matrix:

```
GGally::ggpairs(df %>% select_if(is.numeric))
```