Lecture 2 (part 2): Data Wrangling

Advanced Business Analytics (CIS442D/85)

Simon Business School

1/11/2017

Lecture Outline

Goal: explore packages that facilitate data handling and visualization

1. NumPy
2. Pandas
3. Visualization
   - Matplotlib
   - pandas
   - Additional resources
4. Summary

NumPy

numpy

Numerical Python (NumPy)

- Fundamental package for scientific computing
- Provides
  - Efficient implementation of arrays
    - ndarray
    - Sequential collection of objects
    - Homogeneous
    - Pre-allocated
  - Compatible with other programming languages
  - Tools for reading and writing array data to disk
  - Implementation of common algorithms in Linear algebra, Probability, Signal processing, etc.

ndarray

- Creating arrays: np.array([1,3]), np.zeros(3), np.zeros((3,2)), np.ones(3), np.zeros_like(a), np.eye(3), np.empty(3), np.arange(8), np.arange(1,10,2), a.copy()
- Attributes: shape (dimensions), dtype (object types)
- Data types: integers, floating point numbers, boolean, strings, datetime, timedelta (time difference)
- Precision loss: 0.1+0.2+0.3+0.3+0.1==1 versus np.isclose(0.1+0.2+0.3+0.3+0.1,1)
- Casting: a.astype('int')

Vectorized Operations

- Operations on arrays
  - Faster implementation
  - Readability
- Array operators: a+b, a-b, a*b
- Operations with scalars: a+2, a*2, a**2, 2/a, -2/a
- Special values: np.nan, np.inf, -np.inf, np.isnan(a), np.isinf(a), np.isfinite(a)

Vectorized Operations - cont.

- Math: a.max(), a.min(), a.std(), a.sum(), a.prod(), a.argmax(), a.argmin(), a.cumsum(), a.cumprod(), np.abs(a), np.sqrt(a), np.sin(a), np.sign(a)
- Sort (inplace): a.sort()
- Boolean arrays: $\sim b1$, $b1|b2$, $b1\&b2$, b1==b2, b.all(), b.any(), a<3, a==3, a!=3, np.array_equal(b1,b2), np.where(a%3==0, 0, 1)
- Set operations: np.unique(a), np.intersect1d(a,b), np.union1d(a,b), np.in1d(a,b), np.setdiff1d(a,b)

Exercise 1

See numpy.ipynb

Indexing

- Start from 0
- Basic indexing (using ranges)
  - Numbers or slice objects 1:3:10
  - Examples: a[0,:], a[-1,:], a[:,1], b = a[1:3,0:3:2]
  - Returns a view
- Integer array indexing (using lists of indexes)
  - Extracting values: a[[1],:], a[[-1],:],a[:,[2,3]], a[:,[3,2]], a[[0,1],[3,2]], a[np.ix_([0,1],[3,2])],
  - Setting values: a[:,[3,2]] = 999
  - Returns a copy

Indexing - cont.

- Boolean indexing (using boolean array)
  - Same size booleans array
  - Create a 1-dimensional array copy using indexes: x=a[a%3==0], x=a[(a%2==0)&(a>=4)]
  - Set values: a[a%3==0]=999
- Operations by axis: a.sum(), a.sum(axis=0), a.sum(axis=1), a.std(axis=0), sorting, etc.

Exercise 2: magic square

See numpy.ipynb

Math Packages

- Linear Algebra
  - Transpose matrix: a.T (more generally, use a.transpose() to vary axis order)
  - Dot product: np.dot(a,b)
  - Algorithms for computing: norms, determinants, rank, eigenvalues, inverse, least-squares solution decomposition, solve linear equations, ...
- Probability
  - Random numbers: np.random.rand(3,2), randn(3,2), randint(3,2,-5,5), choice([1,10,20,7])
  - Permutations: permutation([1,2,3,4]), shuffle([1,2,3,4])
  - Other distributions: beta, binomial, gamma, poisson, ...
  - np.random.seed(8)
- SciPy Library - additional packages for scientific computing (optimization, signal processing, statistics, etc.)

Manipulating arrays

- Reshape:
  - a.reshape((3,8)), a.reshape((3,8),order='f'), b.flatten(), b.flatten(order='f'), b.ravel()
  - flatten - returns a copy; reshape, ravel - try to return a view
- Concatenate: np.concatenate((a,b)), np.vstack((a,b)) np.concatenate((a,b),axis=1), np.hstack((a,b))
- Split: np.split(a,2,axis=0), np.split(a,2,axis=1), np.split(a,[2,7],axis=0)
- Broadcasting: operations on arrays with different dimensions
  - a+1 : conceptually, stretches 1 to an array with similar size
  - a+b : broadcasting works only if the dimensions of a match the dimensions of b, component-wise starting from the right:

a.shape	(3,4)	(3,4)	(3,4)	(3,4)
b.shape	(4,)	(3,)	(3,1)	(3,2)
a+b Works?	Yes	No	Yes	No

Pandas

pandas

pandas

- Limitations of numpy
  - homogeneous data
  - querying
- "... an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language" (official website)

pandas

- Limitations of numpy
  - homogeneous data
  - querying
- "... an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language" (official website)
- Key features
  - Implementation of Data Frames
  - Querying capabilities: slicing, aggregating, reshaping, merge, etc.
  - Data alignment
  - Time series functionality
  - Handling of missing data
- Data structures: Series, DataFrame, and Panel

Series

- One dimensional labeled array (homogeneous)
- The label is called index
- Initialization: from list, array, dictionary, series, or scalar
- Attributes: s1.index, s1.dtype, s1.values, s1.shape
- Selection: s1[:3], s1['a']
- Series act as dictionaries: 'a' in s1
- Series act as ndarrays: np.sum(s1), 1/s1
- Default index is 0,1,...,length-1
- Insert: s1['y']=0
- Null values: assigned by np.nan or None and represented as NaN
- Removing items: del s1['y'] (or slice a subset)

Label alignment

- Operations between series is aligned by the label/index
  - Example: s1+s2
  - The new index is the union
  - nan when one of the labels is unavailable
  - s1.add(s2, fill_value=0), (s1+s2).dropna()

DataFrame

- Represent tables
  - collection of same size columns / Series
  - columns are homogeneous
- Initialization: dict of arrays, lists or tuples; dict of series; dict of dicts; list of dicts or series; list of lists; from another dataframe
- Column selection: `df['a']`
- Column addition: `df['new']=0`, `df['new']=df['a']+df['b']`, `df['new']=s`, `df['new']=[7,8,9]`, `df.insert(3,'new',[10,11,12])`
- Delete column: `del df['new']`

Exploring Series and DataFrames

- Basic information: df.shape, df.index, df.dtypes, df.values, df.columns (DataFrame only)
- df.info()
- Print first and last rows: df.head(5), df.tail(5)
- Descriptive statistics:
  - df.describe()
  - examples: df.sum(), df.mean(), df.std(), df.quantile()
  - excludes null values
  - also: count, min, max, abs, prod, cumsum, cumprod,...
  - Index of max/min values: df.idxmin(), df.idxmax()
  - Count frequencies: s.value_counts()

Indexing

- Similar in spirit to numpy (but different objects)
- Selection by label (.loc)
  - Series: s.loc['Alice'], s.loc[['Alice','Davice']], s.loc['Alice':'James'] (inclusive!), s['Alice']
  - DataFrame: df.loc['Alice'], df.loc[['Alice','David']], df.loc['Alice':'James'], df.loc['Alice':'James',['age','country']]
- Selection by position (.iloc)
  - Series: s.iloc[2], s.iloc[:2], s.iloc[[0,3]]
  - DataFrame: df.iloc[0], df.iloc[0,1], df.iloc[1:3,1:3], df.iloc[[0,3],[1,2]]
- Selection by boolean array (.loc):
  s.loc[s<30], df.loc[df.age>30]

Indexing - cont.

- Operator[ ]: Intuitive to use but works slower than .loc and .iloc
- ix: A primarily label-location based indexer, with integer position fallback
  - Series: s.ix['Alice'], s.ix['Alice':'James'], s.ix[2], s.ix[:2], s.ix[s>30]
  - DataFrame:
    Label based: df.ix['Alice'], df.ix[['Alice','Bob']], df.ix[:,'age'], df.ix[:,['age','grade']]
    Position based: df.ix[1:3], df.ix[:,1:3]
    Using boolean array: df.ix[df['grade']>80]
  - When an axis is integer based, only label based indexing is supported
- **use ix unless the index is integer, in which case use .loc or .iloc**

Exercise 1

See pandas.ipynb

Combining Series and DataFrames - Merge

- Similar to SQL join clause - matches all records from two DataFrames that share key
- Merge/join types: outer, inner, left, right
- Column based merging: `pd.merge(df1, df2, on="key", how="inner")`
- Index based merging `pd.merge(df1, df2, left_index=True, right_index=True, how="outer")`
- Specify suffixes, column names, multiple columns or indices, mix index and column, sort, source indicator

Combining Series and DataFrames - Concatenate

- Intuitively, creates large DataFrame by placing smaller DataFrames next to each other (vertically or horizontally)
- Vertically: `pd.concat([df1,df2],axis=0)`
- Horizontally: `pd.concat([df1,df2],axis=1)`
- union or intersection columns/indices (with 'join'), add index that specifies the source (with 'keys')
- Can be used with Series

- Comparison with merge:
  - merge - operates on columns, cartesian product
  - concat - placing/aligning dataframes next or after another

Exercise 2

See pandas.ipynb

Split-Apply-Combine using GroupBy

- Common data analysis task
- split()
  - Split the DataFrame df to smaller DataFrames by column "A" values - `grouped=df.groupby("A")`
  - Split by customized values: `df.groupby([8,9,8,9,8,9])`
  - `df.groupby("A")` is a shortcut for `df.groupby(df['A'])`
  - Main features: iterate over all groups, split by multiple keys, retrieve specific group (`grouped.get_group('y')`), split based on columns or rows
- filter(): filter out groups from dataframes
- agg(): applies aggregating function on each group and concatenates them into a single dataframe. Example: `df.groupby("A").agg(np.sum)`
- apply(): converts groups into dataframes and concatenates them to a single dataframe: `df.groupby('course').apply(lambda d:d.describe())`

Exercise 3

See pandas.ipynb

Pivot tables

- Aggregation tool: creates a summary table by specifying which columns become the indexes and columns of the new table
- Example: `pd.pivot_table(data=adult, index='education', columns='sex', values='over50k', aggfunc=np.mean)`
- Other features: multiple columns on each axis, multiple functions
- Stacking: pivoting columns to indexes
- Unstacking: pivoting indexes to columns

Miscellaneous

- Alignment - cont.
  - DataFrames and Series are aligned
- Missing values
  - np.nan and None denote missing values
  - Series: isnull(), isnotnull(),
  - Replacing missing values: df.fillna(0)
  - Dropping rows with missing values: df.dropna()

Visualization

Visualization

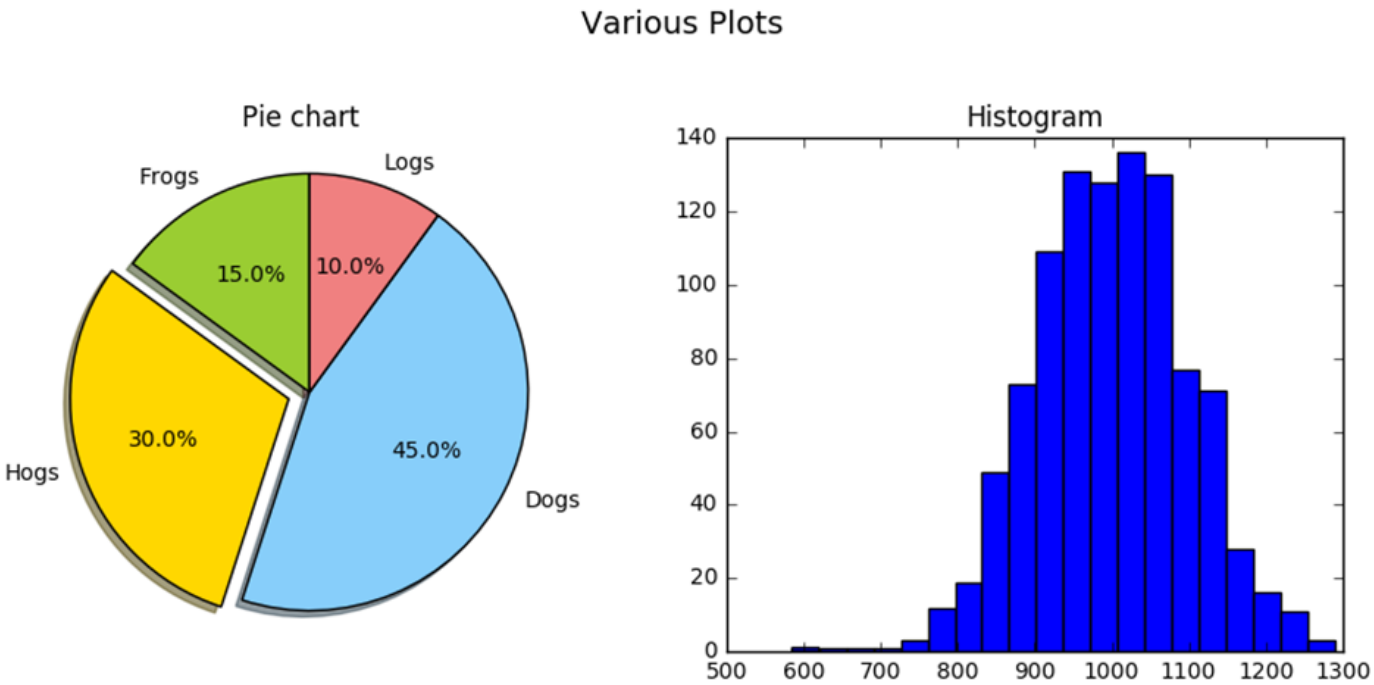
Visualization

- Graphical representation of data
- Communicating information
- Exploring data
  - Identifying trends in time-series
  - Comparison across categories
  - Proportions
  - Correlations
  - Understanding geographic/locational attributes
- In this lecture: basic data visualization in Python
  - Matplotlib
  - Pandas
  - Other packages

Matplotlib

- Cross-platform Python library for creating high-quality plots
- Variety of plots (lines, bars, pie charts, customized, ...)
- Well documented with a large user base (stackoverflow, sourceforge)
- Simple Matlab-like interface using pyplot

Basic Plot Elements

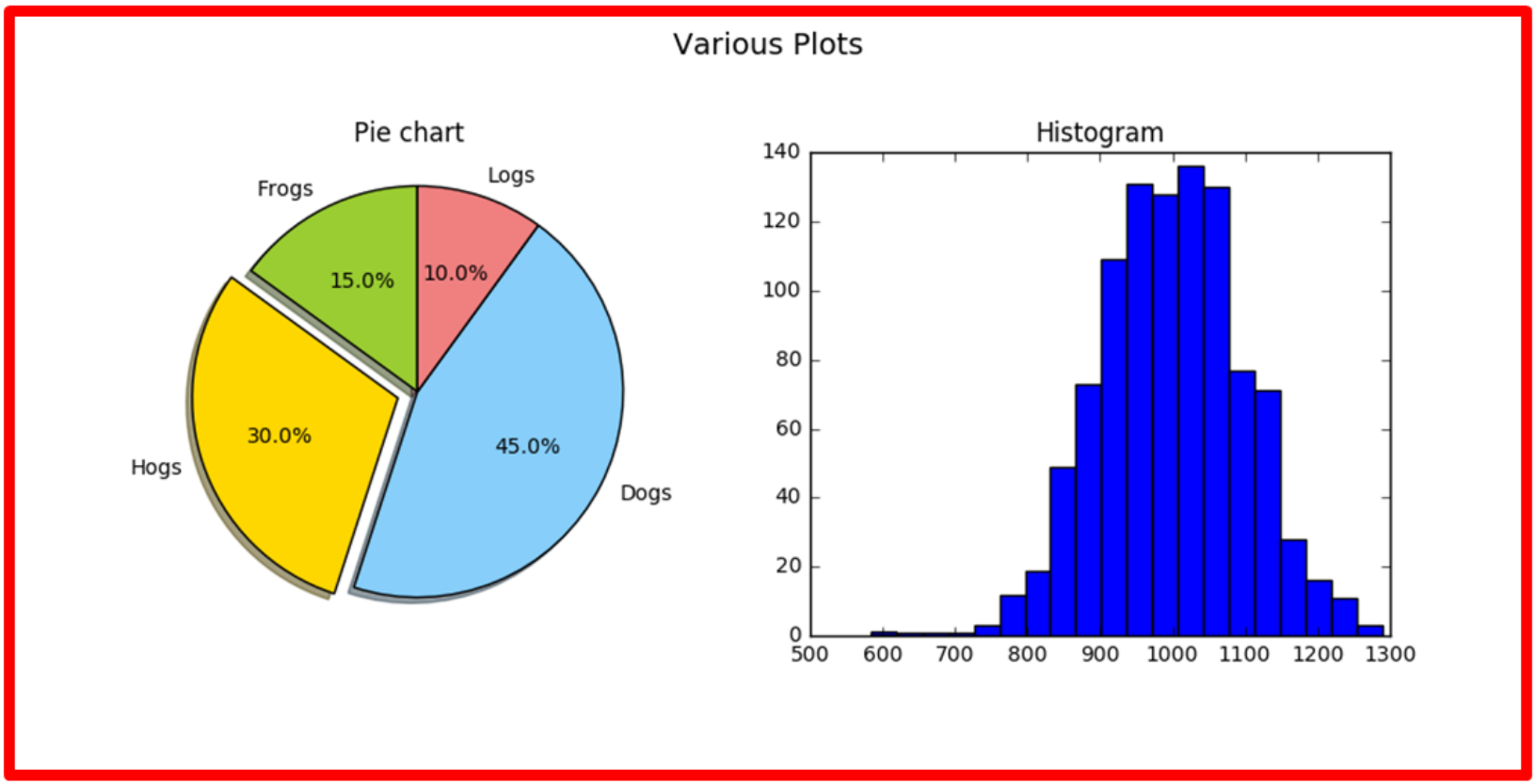


Source: matplotlib documentation

Basic Plot Elements - Figure

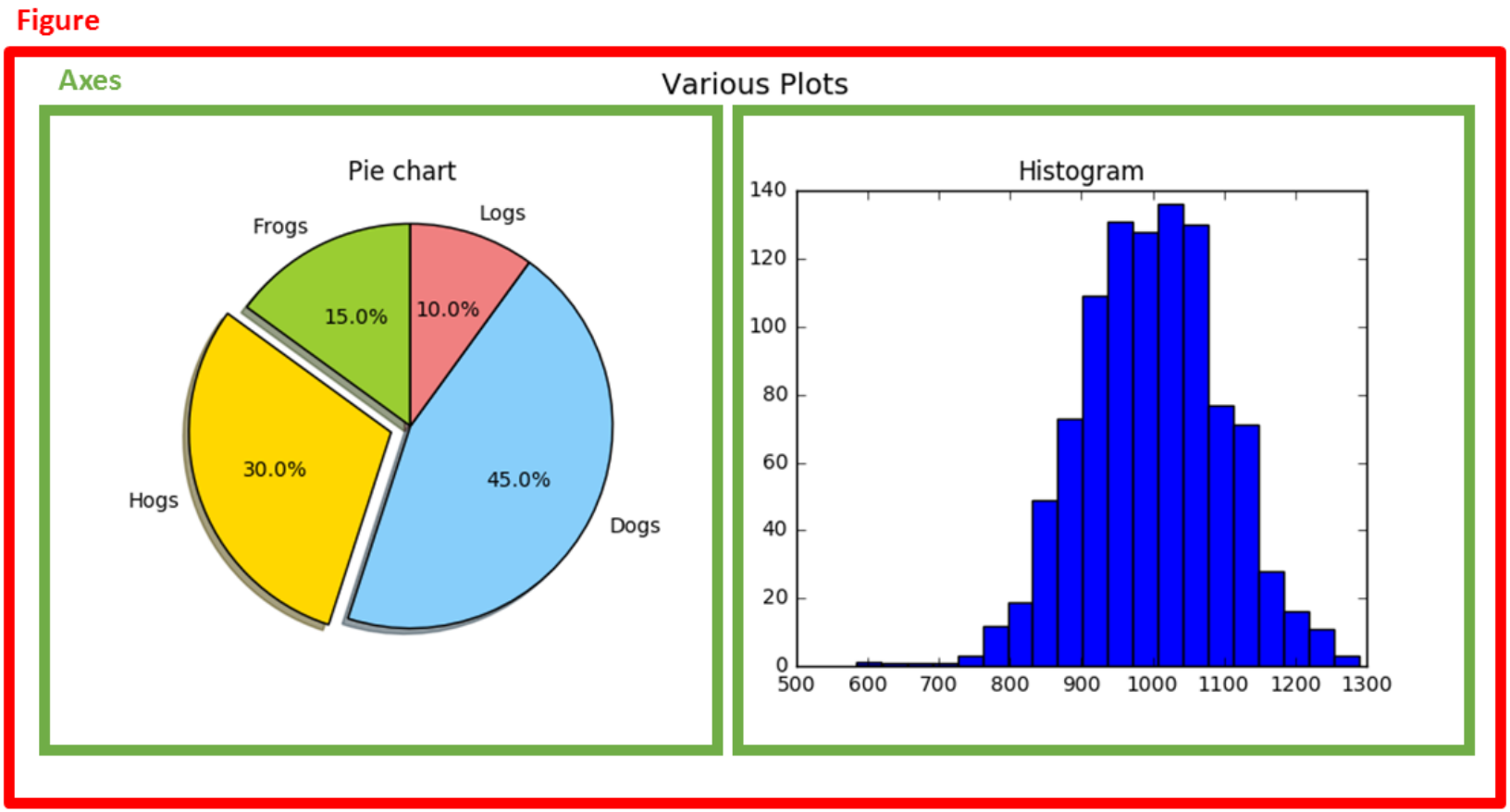
- Top level container for all plot elements
- Contains multiple Axes

Figure



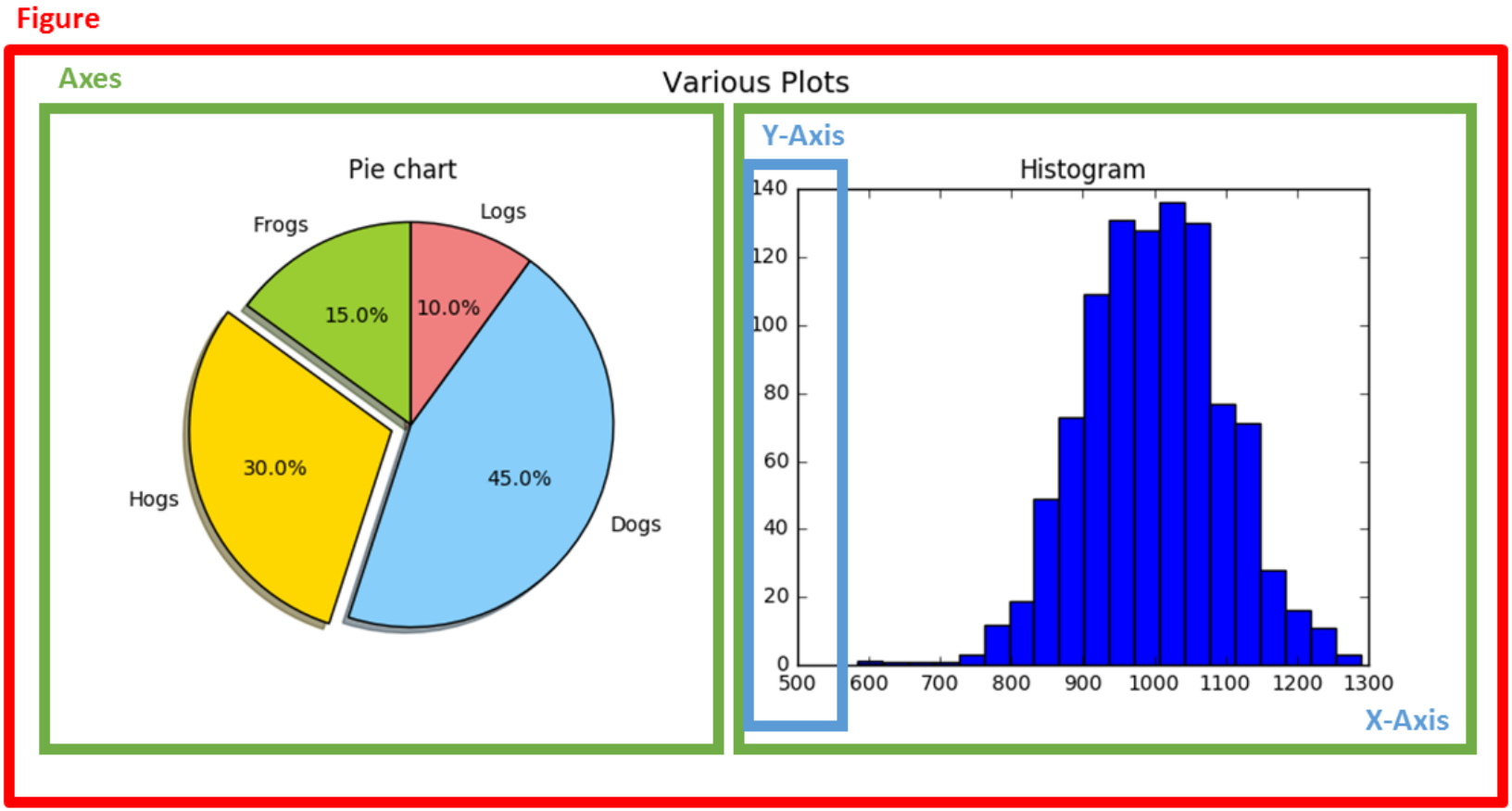
Basic Plot Elements - Axes

- Contain all figure elements associated with a "Sub-plot"



Basic Plot Elements - X/Y Axis

- Control limits, labels and location of ticks on each axis



Matplotlib

- Import library: `import matplotlib.pyplot as plt`
- Backends: 'osx', 'qt4', 'qt5', 'gtk3', 'notebook', 'wx', 'qt', 'nbagg','gtk', 'tk', 'inline'
  - inline - shows plots (inside jupyter)
  - nbagg - interactive (inside jupyter)
  - tk - shows plots in a separate windows (useful dialog window)
  - Setting jupyter backend: `%matplotlib inline`
  - Restart kernel to change the backend
- Create plot with a single axis: `fig, ax = plt.subplots()`
- In jupyter, plots are automatically displayed after each cell. Use plt.show() and plt.close() outside jupyter.
- Plot: `ax.plot([1,2,3,4],[5,1,4,2])`
- Change property: `ax.set_title('title')`

Matplotlib - cont.

- Pyplot
  - Matlab-like interface to matplotlib
  - Allows running one-line plot commands
  - plt.figure() - creates a figure and assigns a numeric identifier to it
  - plt.figure(i) - change the active figure to figure i
  - plt.gcf() - returns current figure
  - plt.gca() - returns current axis of the current figure
  - plt.get_fignums() - returns the list of active figures identifiers
  - fig.get_axes() - returns all axes of a given figure
  - In jupyter, all figures are displayed and deleted from memory after the execution of each cell
  - In Python, one must show (plt.show()) and close (plt.close()) figures to display and close them

Pandas

- Simple interface to matplotlib
- Uses index and column names to generate labels automatically
- Plot series *s* on a given axis: `s.plot(ax=axis)`
- One-liner: `s.plot()`
- Plot dataframe: `df.plot(kind='bar')`, `df.plot.bar()`
- barplots, histogram, boxplots, area plots, scatter plots, pie charts, ...
- Online documentation

Other Libraries

- Geospacial: geoplotlib
- Interactive: gleam
- Networks: NetworkX
- Bokeh
- Seaborn
- ggplot
- pygal
- Online (commercial): Plotly

Summary

- Numpy
  - Arrays (creating, vectorized operations, manipulating)
  - Mathematical libraries (algebra, probability)
- pandas
  - Series and DataFrames
  - Querying, splitting, merging, concatenating, pivoting, cleaning
- Visualization: matplotlib, pandas

Further Reading

- Python for Data Analysis by William Wesley McKinney (numpy, pandas and visualization)
- Numerical computations: NumPy documentation, SciPy documentation, Symbolic computation
- pandas: Online documentation
- Visualization: Matplotlib tutorial, Matplotlib gallery