### Table of Content

<table>
<thead>
<tr>
<th>#</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>Neural Networks Basics</td>
</tr>
<tr>
<td>04</td>
<td>Neural Network Graphs</td>
</tr>
<tr>
<td>06</td>
<td>Machine Learning Basics</td>
</tr>
<tr>
<td>07</td>
<td>Scikit Learn with Python</td>
</tr>
<tr>
<td>08</td>
<td>Scikit Learn Algorithm</td>
</tr>
<tr>
<td>09</td>
<td>Choosing ML Algorithm</td>
</tr>
<tr>
<td>11</td>
<td>Tensor Flow</td>
</tr>
<tr>
<td>12</td>
<td>Python Basics</td>
</tr>
<tr>
<td>13</td>
<td>PySpark Basics</td>
</tr>
<tr>
<td>14</td>
<td>Numpy Basics</td>
</tr>
<tr>
<td>15</td>
<td>Bokeh</td>
</tr>
<tr>
<td>16</td>
<td>Karas</td>
</tr>
<tr>
<td>17</td>
<td>Pandas</td>
</tr>
<tr>
<td>18</td>
<td>Data Wrangling with Pandas</td>
</tr>
<tr>
<td>19</td>
<td>Data Wrangling with dplyr &amp; tidyr</td>
</tr>
<tr>
<td>20</td>
<td>SciPi</td>
</tr>
<tr>
<td>21</td>
<td>MatPlotLib</td>
</tr>
<tr>
<td>22</td>
<td>Data Visualization with ggplot</td>
</tr>
<tr>
<td>23</td>
<td>Big-O</td>
</tr>
</tbody>
</table>
Neural Networks Basic Cheat Sheet

Index
- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolutional or Pool

Original Copyright by AsimovInstitute.org See original here
Machine Learning Overview

MACHINE LEARNING IN EMOJI

BecomingHuman.AI

SUPERVISED

UNSUPERVISED

REINFORCEMENT

BASIC REGRESSION

LINEAR

linear_model.LinearRegression()
Lots of numerical data

LOGISTIC

linear_model.LogisticRegression()
Target variable is categorical

CLUSTER ANALYSIS

K-MEANS

cluster.KMeans()
Similar datum into groups based on centroids

ANOMALY DETECTION

covariance.EllipticalEnvelope()
Finding outliers through grouping

NEURAL NET

neural_network.MLPClassifier()
Complex relationships. Prone to overfitting. Basically magic.

K-NN

neighbors.KNeighborsClassifier()
Group membership based on proximity

DECISION TREE

tree.DecisionTreeClassifier()
If/then/else. Non-contiguous data. Can also be regression.

RANDOM FOREST

ensemble.RandomForestClassifier()
Find best split randomly. Can also be regression

CLASSIFICATION

SVM

svm.SVC()  svm.LinearSVC()
Maximum margin classifier. Fundamental Data Science algorithm

NAIVE BAYES

GaussianNB()  MultinomialNB()  BernoulliNB()
Updating knowledge step by step with new info

FEATURE REDUCTION

T-DISTRIBUTION STOCHASTIC NEIGHBOR EMBEDDING

manifold.TSNE()
Visual high dimensional data. Convert similarity to joint probabilities

PRINCIPLE COMPONENT ANALYSIS

decomposition.PCA()
Distill feature space into components that describe greatest variance

CANONICAL CORRELATION ANALYSIS

decomposition.CCA()
Making sense of cross-correlation matrices

LINEAR DISCRIMINANT ANALYSIS

de.LDA()
Linear combination of features that separates classes

OTHER IMPORTANT CONCEPTS

BIAS VARIANCE TRADEOFF

UNDERFITTING / OVERFITTING

INERTIA

ACCURACY FUNCTION

( TP + TN ) / ( P + N )

PRECISION FUNCTION

manifold.TSNE()

SPECIFICITY FUNCTION

TN / ( FP + TN )

SENSITIVITY FUNCTION

TP / ( TP + FN )

Originally Created by Emily Barry. See original here.
Skicit Learn

Skicit Learn is an open source Python library that implements a range of machine learning, processing, cross validation and visualization algorithm using a unified approach.

A Basic Example

```python
>>> from sklearn import neighbors, datasets
>>> X, y = iris.data[:, :2], iris.target
>>> knn = KNeighborsClassifier(n_neighbors=5)
```

Preprocessing The Data

Standardization

```python
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler() X_train = scaler.fit_transform(X_train)
```

Normalization

```python
>>> from sklearn.preprocessing import Normalizer
>>> normalizer = Normalizer() X_train = normalizer.fit_transform(X_train)
```

Binarization

```python
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0) X_train = binarizer.fit_transform(X_train)
```

Encoding Categorical Features

```python
>>> from sklearn.preprocessing import OneHotEncoder
>>> encoder = OneHotEncoder() X_train = encoder.fit_transform(X_train)
```

Imputing Missing Values

```python
>>> from sklearn.preprocessing import Imputer
>>> imputer = Imputer(missing_values='NaN', strategy='mean', axis=0) X_train = imputer.fit_transform(X_train)
```

Generating Polynomial Features

```python
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures() X_train = poly.fit_transform(X_train)
```

Encodings

```python
>>> from sklearn.preprocessing import LabelEncoder
>>> label_encoder = LabelEncoder() y = label_encoder.fit_transform(y)
```

Supervised Estimators

```python
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=3) knn.fit(X_train, y_train)
```

Unsupervised Estimators

```python
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=2) X_train = pca.fit_transform(X_train)
```

Model Fitting

```python
>>> from sklearn import linear_model
>>> lr = linear_model.LinearRegression() lr.fit(X_train, y_train)
```

Model Evaluation

```python
>>> from sklearn import metrics
>>> score = metrics.r2_score(y_test, lr.predict(X_test))
```

Features

```python
>>> X_train = np.array([[1., 2., 3., 4.], [5., 6., 7., 8.], [9., 10., 11., 12.]])
>>> y_train = np.array([1, 2, 3])
```

```
```
Skicit-learn Algorithm

BecomingHuman.AI

classification

- SVC Ensemble Classifiers
- KNeighbors Classifier
- Naive Bayes
- Linear SVC
- Text Data

regression

- SGD Regressor
- ElasticNet Lasso
- SVR(kernel='rbf') EnsembleRegressors

clustering

- Spectral Clustering GMM
- KMeans
- MiniBatch KMeans
- MeanShift VBGMM

Created by Skikit-Learn.org BSD Licence. See Original here.
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you’re trying to answer.
Part 3
Data Science with Python
In May 2017 Google announced the second-generation of the TPU, as well as the availability of the TPUs in Google Compute Engine.[12] The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs provide up to 11.5 petaflops.

TensorFlow is an open source software library created by Google for numerical computation and large scale machine learning. TensorFlow bundles together Machine Learning, Deep Learning models and frameworks and makes them useful by way of common metaphor. It is capable of running on top of Python and is built for fast experimentation via deep neural networks and modular design. It is expected to become the new standard for large scale machine learning and neural network experiments.

Keras
Keras is an open source neural networks library, written in Python and is built for fast experimentation via deep neural networks and modular design. It is capable of running on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, or PlaidML.

Skflow
Skflow is a high level interface based on tensorflow which can be used like sklearn. You can build your own model on your own level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code.

Installation
How to install new package in Python
pip install "package-name"
Example: pip install requests

How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.12-<python_version>-<cpu/gpu>.whl
sudo pip install
How to install Skflow
pip install sklearn
How to install Keras
pip install keras
update ~/.keras/keras.json – replace "theano" by "tensorflow"

Helpers
Python helper Important functions
type(object) Get object type
default(object) Get help for object (list of available methods, attributes, signatures and so on)
dir(object) Get list of object attributes (fields, functions)
attr(object) Transform an object to string object?
show(functions) Shows documentation about the object
locals() Return the dictionary containing the current scope’s local variables.
globals() Update and return a dictionary containing the current scope’s local variables.
delattr(object) Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.
import builtins
diri.builtin
Other built-in functions

Tensor Flow Cheat Sheet
BecomingHuman.AI

How to install Keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Information
How to install new package in Python
pip install "package-name"
Example: pip install requests

How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.12-<python_version>-<cpu/gpu>.whl
sudo pip install
How to install Skflow
pip install sklearn
How to install Keras
pip install keras
update ~/.keras/keras.json – replace "theano" by "tensorflow"

Keras
Keras is an open source neural networks library, written in Python and is built for fast experimentation via deep neural networks and modular design. It is capable of running on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, or PlaidML.

Skflow
Skflow is a high level interface based on tensorflow which can be used like sklearn. You can build your own model on your own level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code.

Installation
How to install new package in Python
pip install "package-name"
Example: pip install requests

How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.12-<python_version>-<cpu/gpu>.whl
sudo pip install
How to install Skflow
pip install sklearn
How to install Keras
pip install keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Helpers
Python helper Important functions
type(object) Get object type
default(object) Get help for object (list of available methods, attributes, signatures and so on)
dir(object) Get list of object attributes (fields, functions)
attr(object) Transform an object to string object?
show(functions) Shows documentation about the object
locals() Return the dictionary containing the current scope’s local variables.
globals() Update and return a dictionary containing the current scope’s local variables.
delattr(object) Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.
import builtins
diri.builtin
Other built-in functions

Main classes
tf.Graph
tf.Operation
tf.Tensor
tf.Session
Some useful functions
tf.get_default_graph()
tf.reset_default_graph()
tf.get_default_graph()
tf.device("cpu")
(n_name, scope()/value)
(tf.convert_to_tensor(value)

TensorFlow Optimizers
GradientDescentOptimizer
AdadeltaOptimizer
AdamOptimizer
MomentumOptimizer
RMSPropOptimizer

Activation functions
relu
elu
softplus
softmax
softmax_cross_entropy_with_logits
logsoftmax

Reduction functions
accumulate_n
reduce_any
reduce_all
reduce_max
reduce_min
reduce_prod
reduce_sum

Weight
Bias
Dropout

Integration
Keras
TensorFlow
Theano
Microsoft Cognitive Toolkit
PlaidML

How to install Keras
pip install keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Info
TensorFlow
TensorFlow™ is an open source software library created by Google for numerical computation and large scale machine learning. TensorFlow bundles together Machine Learning, Deep learning models and frameworks and makes them useful by way of common metaphor.

Keras
Keras is an open source neural networks library, written in Python and is built for fast experimentation via deep neural networks and modular design. It is capable of running on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, or PlaidML.

Skflow
Skflow is a high level interface based on tensorflow which can be used like sklearn. You can build your own model on your own level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code.

Installation
How to install new package in Python
pip install "package-name"
Example: pip install requests

How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.12-<python_version>-<cpu/gpu>.whl
sudo pip install
How to install Skflow
pip install sklearn
How to install Keras
pip install keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Helpers
Python helper Important functions
type(object) Get object type
default(object) Get help for object (list of available methods, attributes, signatures and so on)
dir(object) Get list of object attributes (fields, functions)
attr(object) Transform an object to string object?
show(functions) Shows documentation about the object
locals() Return the dictionary containing the current scope’s local variables.
globals() Update and return a dictionary containing the current scope’s local variables.
delattr(object) Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.
import builtins
diri.builtin
Other built-in functions

Main classes
tf.Graph
tf.Operation
tf.Tensor
tf.Session
Some useful functions
tf.get_default_graph()
tf.reset_default_graph()
tf.get_default_graph()
tf.device("cpu")
(n_name, scope()/value)
(tf.convert_to_tensor(value)

TensorFlow Optimizers
GradientDescentOptimizer
AdadeltaOptimizer
AdamOptimizer
MomentumOptimizer
RMSPropOptimizer

Activation functions
relu
elu
softplus
softmax
softmax_cross_entropy_with_logits
logsoftmax

Reduction functions
accumulate_n
reduce_any
reduce_all
reduce_max
reduce_min
reduce_prod
reduce_sum

Weight
Bias
Dropout

Integration
Keras
TensorFlow
Theano
Microsoft Cognitive Toolkit
PlaidML

How to install Keras
pip install keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Info
TensorFlow
TensorFlow™ is an open source software library created by Google for numerical computation and large scale machine learning. TensorFlow bundles together Machine Learning, Deep learning models and frameworks and makes them useful by way of common metaphor.

Keras
Keras is an open source neural networks library, written in Python and is built for fast experimentation via deep neural networks and modular design. It is capable of running on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, or PlaidML.

Skflow
Skflow is a high level interface based on tensorflow which can be used like sklearn. You can build your own model on your own level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code.

Installation
How to install new package in Python
pip install "package-name"
Example: pip install requests

How to install tensorflow?
device = cpu/gpu
python_version = cp27/cp34
sudo pip install https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.12-<python_version>-<cpu/gpu>.whl
sudo pip install
How to install Skflow
pip install sklearn
How to install Keras
pip install keras
update ~/.keras/keras.json – replace ‘theano’ by ‘tensorflow’

Helpers
Python helper Important functions
type(object) Get object type
default(object) Get help for object (list of available methods, attributes, signatures and so on)
dir(object) Get list of object attributes (fields, functions)
attr(object) Transform an object to string object?
show(functions) Shows documentation about the object
locals() Return the dictionary containing the current scope’s local variables.
globals() Update and return a dictionary containing the current scope’s local variables.
delattr(object) Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.
import builtins
diri.builtin
Other built-in functions

Main classes
tf.graph
tf.operation
tf.tensor
tf.session
Some useful functions
tf.get_default_graph()
tf.reset_default_graph()
tf.get_default_graph()
tf.device("cpu")
(n_name, scope()/value)
(tf.convert_to_tensor(value)

TensorFlow Optimizers
GradientDescentOptimizer
AdadeltaOptimizer
AdamOptimizer
MomentumOptimizer
RMSPropOptimizer

Activation functions
relu
elu
softplus
softmax
softmax_cross_entropy_with_logits
logsoftmax

Reduction functions
accumulate_n
reduce_any
reduce_all
reduce_max
reduce_min
reduce_prod
reduce_sum

Weight
Bias
Dropout

Integration
Keras
TensorFlow
Theano
Microsoft Cognitive Toolkit
PlaidML

Each classifier and regressor have following fields
n_classes=0 (Regressor), n_classes are expected to be input (Classifier)
batch_size=32, steps=200, // except
TensorFlowRNNClassifier - there is 50
optimizer=Adagrad, learning_rate=0.1,

Each class has a method fit
IRIX: x monitor:None (logdir=None)
Xi: matrix or tensor of shape [n_samples, n_features, …] Can be iterator that returns arrays of features. The training input samples for fitting the model.
V: vector or matrix [n_samples, …] or [n_samples, n_outputs]. Can be iterator that returns array of targets. The training target values (class labels in classification, real numbers in regression).
monitor: Monitor object to print training progress and invoke early stopping
logdir: the directory to save the log file that can be used for optional visualization.

predict(X, axis=None, batch_size=None)

Args:
X: array-like matrix, [n_samples, n_features, …] or iterator axis: Which axis to argmax for classification. By default axis 1 (next after batch) is used. Use 2 for sequence predictions.
batch_size: If text set is too big, use batch size to split it into mini batches. By default the batch_size member variable is used.

Returns:
y: array of shape [n_samples]. The predicted classes or values (class labels in classification, real numbers in regression).
**Phyton For Data Science**

**Cheat-Sheet Phyton Basic**

**BecomingHuman.AI**

---

### Variables and Data Types

**Variable Assignment**

```python
>>> x=5
>>> x
5
```

**Calculations With Variables**

```python
>>> x + 2
7
>>> x - 3
2
>>> x * 2
10
>>> x ** 2
25
>>> x / 2
1.25
```

---

### Lists

**Also see NumPy Arrays**

```python
>>> my_list = ['my', 'list', 'is', 'nice']
>>> my_list[1]  # Select item at index 1
'list'
>>> my_list[-3]  # Select 3rd last item
'list'
>>> my_list[1:3]  # Select items at index 1 and 2
['list', 'is']
>>> my_list[1:]  # Select items after index 0
['list', 'is', 'nice']
>>> my_list[:3]  # Select items before index 3
['my', 'list', 'is']
```

**List Operations**

```python
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
```

**List Methods**

```python
>>> my_list.index('is')  # Get the index of an item
1
>>> my_list.count('is')  # Count an item
2
>>> my_list.append('!')  # Append an item at a time
>>> my_list.remove('!')  # Remove an item
>>> del my_list[0:1]  # Delete items in an array
>>> my_list.reverse()  # Reverse the list
>>> my_list.extend('!')  # Append an item
>>> my_list.pop(-1)  # Remove an item
>>> my_list.insert(0,'!')  # Insert an item
>>> my_list.sort()  # Sort the list
```

---

### NumPy Arrays

**Also see Lists**

```python
>>> my_array = np.array([1, 2, 3, 4])
>>> my_array > 3
array([False, False, False, True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([7, 10, 13, 16])
```

**NumPy Array Operations**

```python
>>> my_array.shape  # Get the dimensions of the array
(4,)
>>> np.append(my_array, [5, 6])  # Append items to an array
array([1, 2, 3, 4, 5, 6])
>>> np.insert(my_array, 1, 5)  # Insert items in an array
array([1, 5, 2, 3, 4, 5, 6])
>>> np.delete(my_array, [0, 1])  # Delete items in an array
array([2, 3, 4, 5, 6])
>>> np.mean(my_array)  # Mean of the array
3.5
>>> np.median(my_array)  # Median of the array
3.5
>>> my_array.correlate()  # Correlation coefficient
array([0.25, 0.5, 0.75, 1])
>>> np.std(my_array)  # Standard deviation
1.5811388300841898
```

---

### Strings

**Also see NumPy Arrays**

```python
>>> my_string = 'This is awesome'
>>> my_string * 2
'This is awesomeThis is awesome'
>>> my_string + 'It is'
'This is awesomeIt is'
>>> 'a' in my_string
True
```

**String Operations**

```python
>>> my_string = 'This is awesome'
>>> my_string.upper()  # Convert to uppercase
'THIS IS AWESOME'
>>> my_string.lower()  # Convert to lowercase
'this is awesome'
>>> my_string.count('a')  # Count string elements
2
>>> my_string.replace('is', 'are')  # Replace string elements
'This are awesome'
>>> my_string.strip()  # Strip whitespaces
'This is awesome'
```

---

### Import Libraries

```python
>>> import numpy
>>> import numpy as np
```

**Selective Import**

```python
>>> from math import pi
```

---

### Libraries

**Import libraries**

```python
>>> import numpy
```

**Select import**

```python
>>> import numpy as np
```

---

### Install Python

**Anaconda**

Leading open data science platform powered by Python

**Free IDE that is included with Anaconda**

**Create and share documents with live code, visualizations, text...**
PySpark is the Spark Python API that exposes the Spark programming model to Python.

**PySpark**

---

### Initializing Spark

**SparkContext**

```python
>>> from pyspark import SparkContext
>>> sc = SparkContext(master = 'local[2]')
```

### Calculations With Variables

- `sc.version` : Retrieve SparkContext version
- `sc.pythonVersion` : Retrieve Python version
- `sc.master` : Master URL to connect to
- `sc.applicationId` : Path where Spark is installed on worker nodes
- `sc.applicationName` : Retrieve name of the Spark User running SparkContext
- `sc.hadoopConf` : Retrieve application name
- `sc.sparkUser` : Retrieve application ID
- `sc.defaultParallelism` : Return default level of parallelism
- `sc.defaultMinPartitions` : Default minimum number of partitions for RDDs

### Configuration

- `sparkconf = (SparkConf()
  >>> conf = SparkConf()
  >>> conf.setMaster("local[2]"
  >>> conf.setAppName("my app")
  >>> conf.sparkCamelCase="true"
  >>> conf.sparkExecutorMemory="1g"
)

### Selecting Data

- `rdd.collected()`
- `rdd.take(2)`
- `rdd.takeFirst(2)`
- `rdd.top(2)`
- `rdd.sample(False, 0.15, 81)()`

### Loading Data

- `conf = (SparkConf()
  >>> conf = (SparkConf()
  >>> conf.setMaster("local[2]"
  >>> conf.setAppName("my app")
  >>> conf.sparkCamelCase="true"
  >>> conf.sparkExecutorMemory="1g"
)

### Filtering

- `rdd.filter(lambda x: x in x)`
- `rdd.distinct().collect()`
- `rdd.keyValues().collect()`

### Mathematical Operations

- `rdd.subtract(triple(2))
  >>> rdd.subtract(triple(2))
  >>> rdd.subtract(triple(2), triple(1))
- `rdd.cartesian(rdd2).collect()`

### Reshaping Data

- `rdd.repartition(n)`
- `rdd.coalesce(n)`

### Executing

- `$ ./bin/spark-submit examples/src/main/python/pi.py`

---

Content Copyright by DataCamp.com. Design Copyright by BecomingHuman.AI. See Original here.
# NumPy Basics Cheat Sheet

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

## Creating Arrays

```python
>>> a = np.array([1, 2, 3])
>>> b = np.array([[1, 2, 3], [4, 5, 6]])
>>> c = np.array([[1.5, 2, 3], [4, 5, 6]], dtype=float)
```

## Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td>Signed 32-bit integer types</td>
</tr>
<tr>
<td><code>float</code></td>
<td>Signed 64-bit floating point numbers</td>
</tr>
<tr>
<td><code>complex</code></td>
<td>Complex numbers represented by 2 floats</td>
</tr>
<tr>
<td><code>bool</code></td>
<td>Boolean type storing TRUE and FALSE</td>
</tr>
</tbody>
</table>

## Arithmetic Operations

```python
>>> a = np.array([1, 2, 3])
>>> b = np.array([4, 5, 6])
>>> a + b
```

## Array Manipulation

```python
>>> a = np.array([1, 2, 3])
>>> b = np.array([[1, 2, 3], [4, 5, 6]])
```

## Saving & Loading On Disk

```python
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
```

## Splitting Arrays

```python
>>> a = np.array([1, 2, 3, 4, 5, 6])
>>> np.hsplit(a, 3)  # Split vertically at the 3rd index
```

## Sorting Arrays

```python
>>> a = np.array([3, 2, 1])
>>> a.sort()
```

## Asking For Help

```python
>>> np.info(np.ndarray.dtype)
```

## I/O

```python
>>> np.savetxt('myarray.txt', a, delimiter=' ')  # Save array to file
```

## Transposing Array

```python
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> a.T
```

## Sorting Arrays

```python
>>> a = np.array([3, 2, 1])
>>> a.sort()  # Sort an array
```

## Adding/Removing Elements

```python
>>> a = np.array([1, 2, 3])
>>> np.append(a, 4)
```

## Array Dimensions

```python
>>> a.shape
```

## Array Equivalences

```python
>>> a.all()  # Return a boolean array if True or False
>>> a.max()  # Maximum value of an array along a given axis
```

## Aggregate Functions

```python
>>> a.min()  # Minimum value of an array along a given axis
```

## Boolean Indexing

```python
>>> a[a > 2]  # Select elements where condition is True
```

## Comparisons

```python
>>> a > b
```

## Linear Algebra

```python
>>> np.dot(a, b)
```

## Sorting Arrays

```python
>>> np.sort(a)  # Sort an array
```

## Splitting Arrays

```python
>>> np.split(a, 2)  # Split an array into two sub-arrays
```

## Saving & Loading Text Files

```python
>>> np.savetxt('my_file.csv', a, delimiter=',')  # Save array to file
```

## Boolean Indexing

```python
>>> a == b
```

## Complex Numbers

```python
>>> np.complex(1, 2)  # Create a complex number
```

## Adding/Removing Elements

```python
>>> a = np.array([1, 2, 3])
>>> np.append(a, 4)
```

## Array Equivalences

```python
>>> a.all()  # Return a boolean array if True or False
>>> a.max()  # Maximum value of an array along a given axis
```

## Aggregate Functions

```python
>>> a.min()  # Minimum value of an array along a given axis
```

## Boolean Indexing

```python
>>> a[a > 2]  # Select elements where condition is True
```

## Comparisons

```python
>>> a > b
```

## Linear Algebra

```python
>>> np.dot(a, b)
```

## Sorting Arrays

```python
>>> np.sort(a)  # Sort an array
```

## Splitting Arrays

```python
>>> np.split(a, 2)  # Split an array into two sub-arrays
```

## Saving & Loading Text Files

```python
>>> np.savetxt('my_file.csv', a, delimiter=',')  # Save array to file
```
The Python interactive visualization library Bokeh enables high-performance visual presentation of large datasets in modern web browsers.

Bokeh’s mid-level general purpose bokeh plotting interface is centered around two main components: data and glyphs.

**Data**

- **Import Data**
  - Use `p = figure(x_range, y_range)`
  - Use `p = figure()`

- **Add Data**
  - Use `p.line(x, y)`
  - Use `p.scatter(x, y)`

- **Set Attributes**
  - Use `p.title.text = 'My Title'`
  - Use `p.outline_line_width = 2`

- **Add Tools**
  - Use `hover = HoverTool(tooltips=[('x', '@x'), ('y', '@y')])`
  - Use `p.add_tools(hover)`

**Glyphs**

- **Scatter Markers**
  - Use `p.scatter(x, y, size=10, color='blue')`
  - Use `p.circle(x, y, size=10)`

- **Line Glyphs**
  - Use `p.line(x, y)`
  - Use `p.line(x, y, line_width=2)`

- **Box Plot**
  - Use `p.boxplot('x', 'y', source=data)`
  - Use `p.boxwhisker('x', 'y', source=data)`

**Rows & Columns**

- **Row**
  - Use `p1 = figure()`
  - Use `p2 = figure()`

- **Column**
  - Use `p = gridplot([[p1, p2]])`
  - Use `p = column([p1, p2])`

**Legend**

- **Legend Location**
  - Use `p.legend.location = 'top_left'`
  - Use `p.legend.location = 'bottom_right'`

- **Legend Orientation**
  - Use `p.legend.orientation = 'vertical'`
  - Use `p.legend.orientation = 'horizontal'`

**Tabbed Layout**

- Use `p1 = Panel(child=p, title='Title')`
- Use `tabs = Tabs(tabs=[p1, p2, p3])`

**Output**

- **To File**
  - Use `p.save('my_plot.html')`

- **To Server**
  - Use `iframe = bokeh应用.ToStatic.html(p)`

**Show or Save Your Plots**

- **Show**
  - Use `show(p)`

- **Save**
  - Use `save(p)`

**Statistical Charts**

- **Bar Chart**
  - Use `p = figure(x_range, y_range, title='Bar Chart')`
  - Use `p.bar(x, y)`

- **Box Plot**
  - Use `p = figure(x_range, y_range, title='Box Plot')`
  - Use `p.boxplot('x', 'y', source=data)`

- **Histogram**
  - Use `p = figure(x_range, y_range, title='Histogram')`
  - Use `p.histogram('x', bins=10)`

- **Scatter Plot**
  - Use `p = figure(x_range, y_range, title='Scatter Plot')`
  - Use `p.scatter('x', 'y', source=data)`

**Data Types**

- **Python lists, NumPy arrays, Pandas DataFrames and other sequences of values**

**Interface are:**

- **The basic steps to creating plots with the bokeh.plotting interface:**
  1. Prepare some data: Python lists, NumPy arrays, Pandas DataFrames and other sequences of values
  2. Create a new plot
  3. Add renderers for your data, with visual customizations
  4. Specify where to generate the output
  5. Show or save the results

- **Under the hood, your data is converted to Column Data Sources.** You can also do this manually:
  - Use `import pandas as pd`
  - Use `df = pd.DataFrame()`

- **Plotting**
  - Use `p = figure(x_range, y_range)`
  - Use `p = figure()`

- **Show or Save Your Plots**
  - Use `show(p)`
  - Use `save(p)`
Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

Model Architecture

Sequential Model

>>> from keras.models import Sequential
>>> >>> model = Sequential()
>>> >>> model.add(Dense(12, input_dim=8, activation='relu'))
>>> >>> model.add(Dense(32, activation='relu'))
>>> >>> model.add(Flatten())
>>> >>> model.add(Dense(num_classes, activation='softmax'))

Binary Classification

>>> model = Sequential()
>>> >>> model.add(Dense(32, activation='relu'))
>>> >>> model.add(Dense(1, activation='sigmoid'))

Multilayer Perceptron (MLP)

Convolutional Neural Network (CNN)

Keras Data Sets

Data

Keras Data Sets

>>> from keras.datasets import boston_housing, mnist, cifar10

>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()

Data

A Basic Example

BecomingHuman.AI

DataCamp

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

_Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data
Pandas Basics
Cheat Sheet

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

# Pandas Data Structures

## Series
A one-dimensional labeled array capable of holding any data type.
```python
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

## DataFrame
A two-dimensional labeled data structure with columns potentially different types.
```python
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
          'Capital': ['Brussels', 'New Delhi', 'Brasília'],
          'Population': [11190846, 1303171035, 207847528]}
>>> df = pd.DataFrame(data,
                    columns=['Country', 'Capital', 'Population'])
```

---

# Selection

## Getting
- Get one element
  ```python
  >>> s['b']
  -5
  >>> df[1:]
  ```
- Get subset of a DataFrame
  ```python
  Select single value by row & column
  Select single value by row & column labels
  Select single row of subset of rows
  Select a single column of subset of columns
  Series s where value is not >1
  s where value is <-1 or >2
  Use filter to adjust DataFrame
  ```
- Set index a of Series s to 6
  ```python
  ```

## Dropping
- Drop values from rows (axis=0)
  ```python
  >>> s.drop(['a', 'c'])
  >>> df.drop('Country', axis=1)
  ```
- Drop values from columns (axis=1)
  ```python
  >>> df.sort_index()
  >>> df.sort_values(by='Country')
  >>> df.rank()
  ```

## Arithmetic Operations with Fill Methods
- You can also do the internal data alignment yourself with the help of the fill methods:
  ```python
  >>> s.add(s3, fill_value=0)
  >>> s.sub(s3, fill_value=2)
  >>> s.div(s3, fill_value=4)
  ```

## Applying Functions
- Apply function
- Apply function element-wise

## I/O
- Read and Write to CSV
  ```python
  >>> pd.read_csv('file.csv', header=None, nrows=5)
  >>> df.to_csv('myDataFrame.csv')
  ```
- Read and Write to Excel
  ```python
  >>> pd.read_excel('file.xlsx')
  >>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
  ```
- Read multiple sheets from the same file
  ```python
  >>> xlsx = pd.ExcelFile('file.xls')
  >>> df = pd.read_excel(xlsx, 'Sheet1')
  ```

---

# Retrieving Series/ DataFrame Information

## Columns
- Describe index
  ```python
  >>> df.index
  ```
- Describe DataFrame columns
  ```python
  >>> df.columns
  >>> df.info()
  >>> df.count()
  >>> df.shape
  ```
- Summarize
  ```python
  >>> df.describe()
  >>> df.mean()
  >>> df.median()
  >>> df.min()/df.max()
  >>> df.idxmin()/df.idxmax()
  >>> df.describe()
  >>> df.mean()
  >>> df.median()
  >>> df.min()/df.max()
  ```

## Sorting
- Sort labels along an axis
  ```python
  >>> df.sort_index()
  >>> df.sort_values(by='Country')
  >>> df.rank()
  ```
- Assign ranka to entries
  ```python
  >>> df['Rank'] = df['Population'].rank()
  ```

---

# Searching

## Boolean Indexing
- Series s where value is not <1
  ```python
  >>> s[(s < -1) | (s > 2)]
  ```
- Use filter to adjust DataFrame
  ```python
  >>> df[df['Population'] > 1200000000]
  ```

---

# Pandas Library

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

## Series
- A one-dimensional labeled array capable of holding any data type.

## DataFrame
- A two-dimensional labeled data structure with columns potentially different types.

---

# Cheat Sheet

## Use the following import convention:
```python
>>> import pandas as pd
```
Combining Data

Pandas Data Structures

Pivot

- `df3 = df2.pivot(index='Date', columns='Type', values='Value')`

Melt

- `df4 = pd.melt(df2, id_vars=['Date'], value_vars=["Type", "Value"], ignore_index=False)`

Pivot Table

- `df4 = pd.pivot_table(df2, values='Value', index='Date', columns='Type', aggfunc='sum')`

Reindexing

- `df = df.reindex(index=index)`

Multindexing

- `arrays = [np.array([1,2,3]), np.array([4,5,6])]`

Join

- `df1 = df1.join(df2, on='X1', how='inner')`

Duplicate Data

- `df = df.drop_duplicates(keep='first')`

Grouping Data

- `df.groupby('Type').mean()`

Missing Data

- `df.dropna()`

Advanced Indexing

- `df.loc[df['Value'] > 1]`

Visualization

- `import matplotlib.pyplot as plt`

Dates

- `pd.to_datetime(df2['Date'], utc=True)`

Concatenate

- `pd.concat([df1, df2], axis=1)`

Indexing With isin

- `df[df['Type'].isin(df2['Type'])]`

Query DataFrame

- `df.query('second > first')`

Setting/Resetting Index

- `df.set_index('Country')`

Drop NaN value

- `df.fillna(0)`

Drop duplicates

- `df.drop_duplicates()`

Filter on values

- `df.where(df > 0)`

Subset the data

- `df[(df.Country.isin(df2.Type))]

Find same elements

- `df[df.Country.duplicated()]`

Select cols with any vals > 1

- `df3.filter(items="a","b")`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]

Select specific elements

- `df3.loc[:,(df3>1).all()]

Select all elements (with values)

- `df3.loc[:,df3.notnull().all()]

Find same elements

- `df3.loc[:,(df3>1).any()]`

Filter on values

- `df3.filter(items="a","b")`

Select cols without NaN

- `df3.loc[:,df3.notnull().all()]`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]`

Select all elements (without values)

- `df3.loc[:,df3.notnull().all()]`

Select cols with any vals > 1

- `df3.filter(items='a','b')`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]`

Select all elements (with values)

- `df3.loc[:,df3.notnull().all()]`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]`

Select all elements (without values)

- `df3.loc[:,df3.notnull().all()]`

Select cols with any vals > 1

- `df3.filter(items='a','b')`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]`

Select all elements (with values)

- `df3.loc[:,df3.notnull().all()]`

Select cols with NaN

- `df3.loc[:,(df3<1).any()]`

Select all elements (without values)

- `df3.loc[:,df3.notnull().all()]`
Data Wrangling with dplyr and tidyr Cheat Sheet

Syntax

**dplyr::**
- **tbl_df()** Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen.
- **mutate()** Add new columns.
- **select()** Select columns.
- **filter()** Select rows.
- **arrange()** Sort rows.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.
- **summarise_all()** Functions for each column.
- **summarise_each()** Functions for each column.
- **summarise_n()** Count number of rows with each unique value of variable (with or without weights).
- **select()** Select columns.
- **mutate()** Add new columns.
- **arrange()** Sort columns.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.
- **summarise_all()** Functions for each column.
- **summarise_each()** Functions for each column.
- **summarise_n()** Count number of rows with each unique value of variable (with or without weights).

**tidyverse::**
- **glimpse()** Information dense summary of tbl data.
- **View()** Displays data set in spreadsheet-like display (note capital V).

**utils::**
- **View()** Displays data set in spreadsheet-like display.

Tidy Data

**In a tidy data set:**
- Each column refers to a variable
- Each row refers to an observation
- Each value refers to a single measurement

**Variables not shown: Petal.Width (df1), Species (fctr)**

**Tidy Data**

A foundation for wrangling in R

Reshaping Data

- **gather()** Gather columns into rows.
- **spread()** Spread rows into columns.
- **separate()** Separate columns.
- **discard()** Discard columns.

Subset Observations (Rows)

- **filter()** Select rows.
- **arrange()** Sort rows.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.

Subset Variables (Columns)

- **select()** Select columns.
- **mutate()** Add new columns.
- **arrange()** Sort columns.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.

Combine Data Sets

- **bind_rows()** Combine and append one or more new rows.
- **bind_cols()** Append one or more new columns.
- **bind_rows()** Combine and append one or more new rows.
- **bind_cols()** Append one or more new columns.

Merging Data

- **left_join()** Left join
- **right_join()** Right join
- **full_join()** Full join
- **inner_join()** Inner join

Summary Data

- **summarise()** Add new columns.
- **summarise_all()** Functions for each column.
- **summarise_each()** Functions for each column.
- **summarise_n()** Count number of rows with each unique value of variable (with or without weights).

Make New Variables

- **mutate()** Add new columns.
- **mutate_all()** Functions for each column.
- **mutate_each()** Functions for each column.

Data Wrangling with dplyr and tidyr Cheat Sheet

BecomingHuman.AI

Cheat Sheet

Syntax

**dplyr::**
- **tbl_df()** Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen.
- **mutate()** Add new columns.
- **select()** Select columns.
- **filter()** Select rows.
- **arrange()** Sort rows.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.
- **summarise_all()** Functions for each column.
- **summarise_each()** Functions for each column.
- **summarise_n()** Count number of rows with each unique value of variable (with or without weights).

**tidyverse::**
- **glimpse()** Information dense summary of tbl data.
- **View()** Displays data set in spreadsheet-like display (note capital V).

**utils::**
- **View()** Displays data set in spreadsheet-like display.

Tidy Data

**In a tidy data set:**
- Each column refers to a variable
- Each row refers to an observation
- Each value refers to a single measurement

**Variables not shown: Petal.Width (df1), Species (fctr)**

**Tidy Data**

A foundation for wrangling in R

Reshaping Data

- **gather()** Gather columns into rows.
- **spread()** Spread rows into columns.
- **separate()** Separate columns.
- **discard()** Discard columns.

Subset Observations (Rows)

- **filter()** Select rows.
- **arrange()** Sort rows.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.

Subset Variables (Columns)

- **select()** Select columns.
- **mutate()** Add new columns.
- **arrange()** Sort columns.
- **summarise()** Compute summary statistics.
- **group_by()** Group by one or more columns.

Combine Data Sets

- **bind_rows()** Combine and append one or more new rows.
- **bind_cols()** Append one or more new columns.
- **bind_rows()** Combine and append one or more new rows.
- **bind_cols()** Append one or more new columns.

Merging Data

- **left_join()** Left join
- **right_join()** Right join
- **full_join()** Full join
- **inner_join()** Inner join

Summary Data

- **summarise()** Add new columns.
- **summarise_all()** Functions for each column.
- **summarise_each()** Functions for each column.
- **summarise_n()** Count number of rows with each unique value of variable (with or without weights).

Make New Variables

- **mutate()** Add new columns.
- **mutate_all()** Functions for each column.
- **mutate_each()** Functions for each column.
### Linear Algebra

**Creating Matrices**
- ```np.array([[1, 2], [3, 4]])```  # Creates a 2x2 array
- ```np.zeros((3, 3))```  # Creates a 3x3 zero array

**Matrix Functions**
- ```np.linalg.det(A)```  # Calculates the determinant of a matrix
- ```np.linalg.inv(A)```  # Calculates the inverse of a matrix
- ```np.linalg.eig(A)```  # Calculates the eigenvalues and eigenvectors of a matrix

**Sparse Matrix Routines**
- ```spsolve(A, b)```  # Solves a sparse linear system` 

---

### Interacting With NumPy

#### Index Tricks
- ```np.greater(A, 0)```  # Returns a boolean array
- ```np.where(A > 0, A, 0)```  # Replaces elements greater than 0 with 0

#### Shape Manipulation
- ```np.flatten(A)```  # Flattens an array
- ```np.reshape(A, (3, 4))```  # Reshapes an array

#### Polynomials
- ```np.polyval(p, x)```  # Evaluates a polynomial at a point

#### Type Handling
- ```np.ascontiguousarray(A)```  # Converts an array to contiguous memory

#### Other Useful Functions
- ```np.frombuffer(b, dtype)```  # Creates an array from a buffer
- ```np.fromfile(f, dtype)```  # Reads data from a file

---

### Scipy Linear Algebra Cheat Sheet

**You’ll use the linalg and sparse modules.** Note that scipy.linalg contains and expands on numy.linalg

#### Creating Matrices
- ```np.random.randn(10, 1)```  # Creates a 10x1 array

#### Matrix Functions
- ```np.linalg.det(A)```  # Calculates the determinant of a matrix
- ```np.linalg.inv(A)```  # Calculates the inverse of a matrix

#### Sparse Matrix Routines
- ```spsolve(A, b)```  # Solves a sparse linear system` 

---

**Asking For Help**
- ```help(np.imag)```  # Gets help on a function or module

---

**Content Copyright by DataCamp.com. Design Copyright by BecomingHuman.AI. See Original Info.**
Matplotlib is a Python 2D plotting library that produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.
Data Visualisation
with ggplot2

Cheat Sheet

**Basics**

- **geom**
  - Use a geom to represent data points, use the geom’s aesthetic properties to represent variables. Each function returns a layer.
  
- **aes**
  - Adds a new layer to a plot with `aes(x, y, f, ...)`. Each provides a geom a set of aesthetic mappings, and a default stat and geom.
  
**Graphical Primitives**

- **geom_point**
  - Represents a single observation. Map a variable to color or size.

- **geom_label**
  - Plots labels at a specific location.

**Position Adjustments**

- **position**
  - Specifies how to arrange geoms that would otherwise occupy the same space.

**Faceting**

- **facet**
  - Splits a plot into sub-plots based on the values of one or more discrete variables.

**Themes**

- **theme**
  - Functions to control plot margins, base size, line width and text size and style.

**Scales**

- **scale**
  - Functions to control the transformation of data to be plotted.

**Coordinate Systems**

- **coord**
  - Functions to control the layout of a plot including aspect ratio, axis limits and ranges, coordinate systems.

**Examples**

- **ggplot()**
  - The very first function that creates a plot.

- **geom_histogram()**
  - Creates a histogram, which is a statistical plot that shows the distribution of a numeric variable.

- **geom_boxplot()**
  - Creates a boxplot, which is a graphical representation of the distribution of a variable.

- **geom_dotplot()**
  - Creates a dot plot, which is a visual representation of the distribution of a variable with each data point represented as a dot.