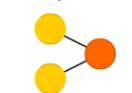


A mostly complete chart of  
**Neural Networks**

©2016 Fjodor van Veen - asimovinstitute.org

- 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
- Convolution or Pool

Perceptron (P)



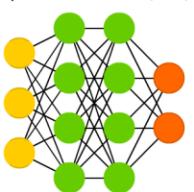
Feed Forward (FF)



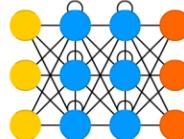
Radial Basis Network (RBF)



Deep Feed Forward (DFF)



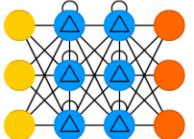
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



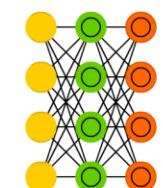
Gated Recurrent Unit (GRU)



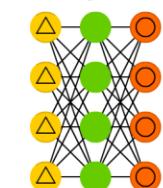
Auto Encoder (AE)



Variational AE (VAE)



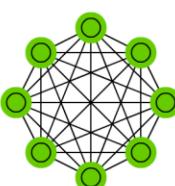
Denoising AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



Hopfield Network (HN)



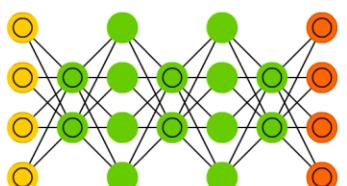
Boltzmann Machine (BM)



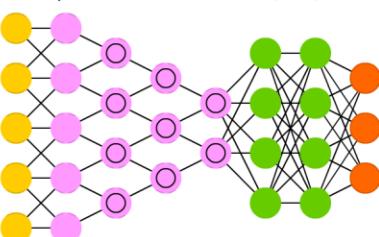
Restricted BM (RBM)



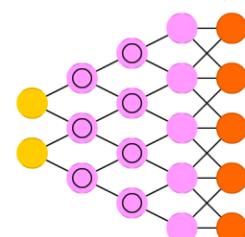
Deep Belief Network (DBN)



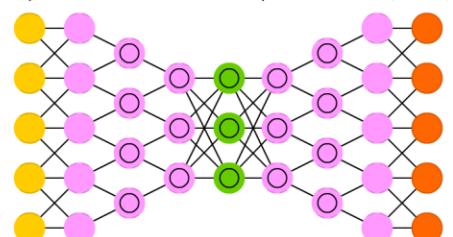
Deep Convolutional Network (DCN)



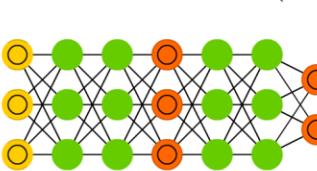
Deconvolutional Network (DN)



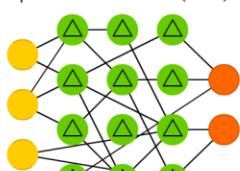
Deep Convolutional Inverse Graphics Network (DCIGN)



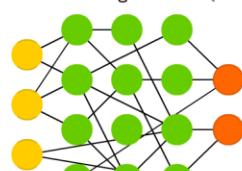
Generative Adversarial Network (GAN)



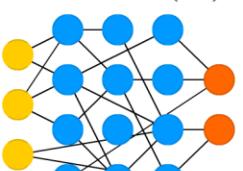
Liquid State Machine (LSM)



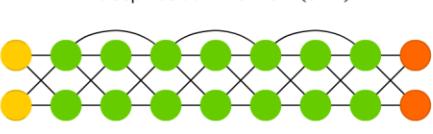
Extreme Learning Machine (ELM)



Echo State Network (ESN)



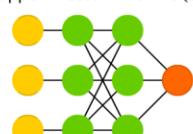
Deep Residual Network (DRN)



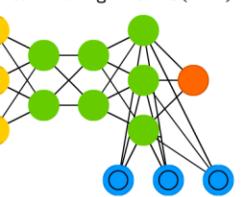
Kohonen Network (KN)



Support Vector Machine (SVM)

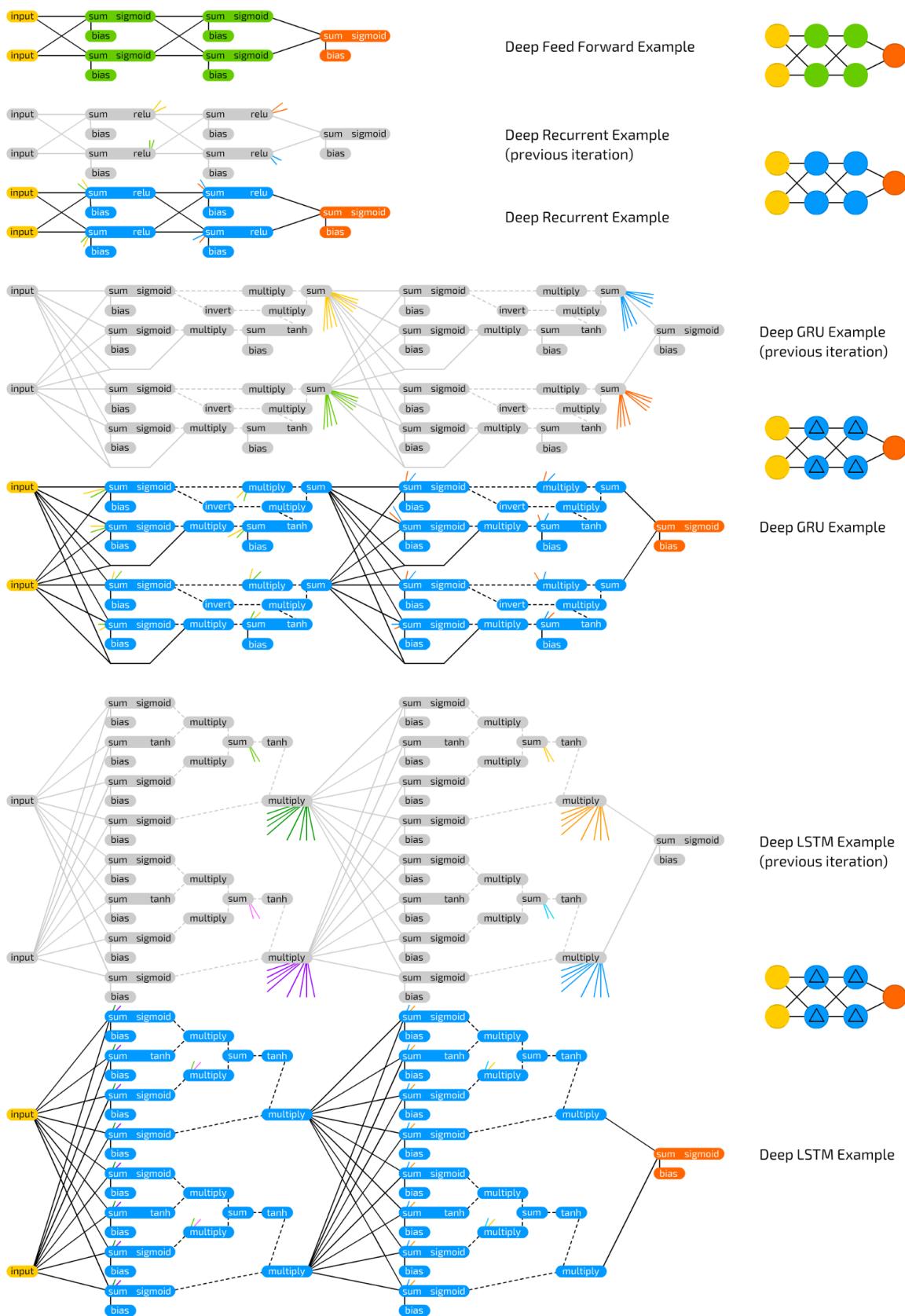


Neural Turing Machine (NTM)



An informative chart to build  
**Neural Network Graphs**

©2016 Fjodor van Veen - asimovinstitute.org



### Linear Vector Spaces:

- Definition: A linear vector space,  $X$ , is a set of elements (vectors) defined over a scalar field,  $F$ , that satisfies the following conditions:
- 1) if  $x \in X$  and  $y \in X$  then  $x+y \in X$ .
  - 2)  $x+y = y+x$ .
  - 3)  $(x+y)+z = x+(y+z)$
  - 4) There is a unique vector  $0 \in X$ , such that  $x+0=x$  for all  $x \in X$ .
  - 5) For each vector  $x \in X$  there is a unique vector in  $X$ , to be called  $(-x)$ , such that  $x+(-x)=0$ .
  - 6) multiplication, for all scalars  $a \in F$ , and all vectors  $x \in X$ ,
  - 7) For any  $x \in X$ ,  $1x=x$  (for scalar 1).
  - 8) For any two scalars  $a \in F$  and  $b \in F$  and any  $x \in X$ ,  $a(bx)=(ab)x$ .
  - 9)  $(a+b)x=a x+b x$ .
  - 10)  $a(x+y)=ax+ay$ .

**Linear Independence:** Consider  $n$  vectors  $\{x_1, x_2, \dots, x_n\}$ . If there exists  $n$  scalars  $a_1, a_2, \dots, a_n$ , at least one of which is nonzero, such that  $a_1x_1 + a_2x_2 + \dots + a_nx_n = 0$ , then the  $\{x_i\}$  are linearly dependent.

### Spanning a Space:

Let  $X$  be a linear vector space and let  $\{u_1, u_2, \dots, u_n\}$  be a subset of vectors in  $X$ . This subset spans  $X$  if and only if for every vector  $x \in X$  there exist scalars  $x_1, x_2, \dots, x_n$  such that  $x = x_1u_1 + x_2u_2 + \dots + x_nu_n$ .

**Inner Product:** for any scalar function of  $x$  and  $y$ .

1.  $(x,y) = (y,x)$
2.  $(x,ay_1+by_2) = a(x,y_1) + b(x,y_2)$
3.  $(x,x) \geq 0$ , where equality holds iff  $x$  is the zero vector.

**Norm:** A scalar function  $\|x\|$  is called a norm if it satisfies:

1.  $\|x\| \geq 0$
2.  $\|x\| = 0$  if and only if  $x = 0$ .
3.  $\|ax\| = |a|\|x\|$
4.  $\|x+y\| \leq \|x\| + \|y\|$

**Angle:** The angle  $\theta$  bet. 2 vectors  $x$  and  $y$  is defined by  $\cos \theta = \frac{(x,y)}{\|x\| \|y\|}$

**Orthogonality:** 2 vectors  $x, y \in X$  are said to be orthogonal if  $(x,y) = 0$ .

### Gram Schmidt Orthogonalization:

Assume that we have  $n$  independent vectors  $y_1, y_2, \dots, y_n$ . From these vectors we will obtain  $n$  orthogonal vectors  $v_1, v_2, \dots, v_n$ .

$$v_1 = y_1, \quad v_k = y_k - \sum_{i=1}^{k-1} \frac{(v_i, y_k)}{(v_i, v_i)} v_i,$$

where  $\frac{(v_i, y_k)}{(v_i, v_i)} v_i$  is the projection of  $y_k$  on  $v_i$

### Vector Expansions:

$$x = \sum_{i=1}^n x_i v_i = x_1 v_1 + x_2 v_2 + \dots + x_n v_n,$$

for orthogonal vectors,  $x_j = \frac{(v_j, x)}{(v_j, v_j)}$

### Reciprocal Basis Vectors:

$$(r_i, v_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}, \quad x_j = (r_j, x)$$

To compute the reciprocal basis vectors: set  $B = [v_1 \ v_2 \ \dots \ v_n]$ ,

$R = [r_1 \ r_2 \ \dots \ r_n]$ ,  $R^T = B^{-1}$  In matrix form:  $x^v = B^{-1} x^s$

### Transformations:

A transformation consists of three parts:

domain:  $X = \{x_i\}$ , range:  $Y = \{y_i\}$ , and a rule relating each  $x_i \in X$  to an element  $y_i \in Y$ .

**Linear Transformations:** transformation  $A$  is linear if:

1. for all  $x_1, x_2 \in X$ ,  $A(x_1+x_2) = A(x_1) + A(x_2)$
2. for all  $x \in X, a \in R$ ,  $A(ax) = aA(x)$

### Matrix Representations:

Let  $\{v_1, v_2, \dots, v_n\}$  be a basis for vector space  $X$ , and let  $\{u_1, u_2, \dots, u_n\}$  be a basis for vector space  $Y$ . Let  $A$  be a linear transformation with domain  $X$  and range  $Y$ :  $A(x) = y$

The coefficients of the matrix representation are obtained from

$$A(v_j) = \sum_{i=1}^m a_{ij} u_i$$

**Change of Basis:**  $B_t = [t_1 \ t_2 \ \dots \ t_n]$ ,  $B_w = [w_1 \ w_2 \ \dots \ w_n]$

$$A' = [B_w^{-1} A B_t]$$

**Eigenvalues & Eigenvectors:**  $Az = \lambda z$ ,  $|(A - \lambda I)| = 0$

**Diagonalization:**  $B = [z_1 \ z_2 \ \dots \ z_n]$ ,

where  $\{z_1, z_2, \dots, z_n\}$  are the eigenvectors of a square matrix  $A$ ,

$$[B^{-1} A B] = \text{diag}([\lambda_1 \ \lambda_2 \ \dots \ \lambda_n])$$

### Perceptron Architecture:

$$\mathbf{a} = \text{hardlim}(\mathbf{Wp} + \mathbf{b}), \quad \mathbf{W} = [\mathbf{w}_1^T \ \mathbf{w}_2^T \ \dots \ \mathbf{w}_s^T]^T, \quad a_i = \text{hardlim}(a_i) = \text{hardlim}(\mathbf{w}_i^T \mathbf{p} + b_i)$$

**Decision Boundary:**  $\mathbf{w}^T \mathbf{p} + b_i = 0$

The decision boundary is always orthogonal to the weight vector.

Single-layer perceptrons can only classify linearly separable vectors.

### Perceptron Learning Rule

$$\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \mathbf{e}\mathbf{p}^T, \quad \mathbf{b}^{\text{new}} = \mathbf{b}^{\text{old}} + \mathbf{e}, \quad \text{where } \mathbf{e} = \mathbf{t} - \mathbf{a}$$

**Hebb's Postulate:** "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

**Linear Associator:**  $\mathbf{a} = \text{purelin}(\mathbf{Wp})$

**The Hebb Rule:** Supervised Form:  $\mathbf{w}_{ij}^{\text{new}} = \mathbf{w}_{ij}^{\text{old}} + t_{qi}P_{qi}$

$$\mathbf{W} = \mathbf{t}_1 \mathbf{P}_1^T + \mathbf{t}_2 \mathbf{P}_2^T + \dots + \mathbf{t}_Q \mathbf{P}_Q^T$$

$$\mathbf{W} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_Q] \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \vdots \\ \mathbf{P}_Q^T \end{bmatrix} = \mathbf{T} \mathbf{P}^T$$

**Pseudoinverse Rule:**  $\mathbf{W} = \mathbf{T} \mathbf{P}^+$

When the number,  $R$ , of rows of  $\mathbf{P}$  is greater than the number of columns,  $Q$ , of  $\mathbf{P}$  and the columns of  $\mathbf{P}$  are independent, then the pseudoinverse can be computed by  $\mathbf{P}^+ = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T$

**Variations of Hebbian Learning:**

**Filtered Learning** (Ch.14):  $\mathbf{W}^{\text{new}} = (1 - \gamma) \mathbf{W}^{\text{old}} + \alpha \mathbf{t}_q \mathbf{p}_q^T$

**Delta Rule** (Ch.10):  $\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \alpha (\mathbf{t}_q - \mathbf{a}_q) \mathbf{p}_q^T$

**Unsupervised Hebb** (Ch.13):  $\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} + \alpha \mathbf{a}_q \mathbf{p}_q^T$

**Taylor:**  $F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^*) \nabla^2 F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x} - \mathbf{x}^*) + \dots$

**Grad**  $\nabla F(\mathbf{x}) = \left[ \frac{\partial}{\partial x_1} F(\mathbf{x}) \quad \frac{\partial}{\partial x_2} F(\mathbf{x}) \quad \dots \quad \frac{\partial}{\partial x_n} F(\mathbf{x}) \right]^T$

**Hessian**:  $\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial}{\partial x_1 x_2} F(\mathbf{x}) & \dots & \frac{\partial}{\partial x_1 x_n} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2 x_1} F(\mathbf{x}) & \frac{\partial}{\partial x_2^2} F(\mathbf{x}) & \dots & \frac{\partial}{\partial x_2 x_n} F(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_n x_1} F(\mathbf{x}) & \frac{\partial}{\partial x_n x_2} F(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n^2} F(\mathbf{x}) \end{bmatrix}$

**Directional Derivatives:**

**1<sup>st</sup> Dir.Der.:**  $\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$ , **2<sup>nd</sup> Dir.Der.:**  $\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x}) \mathbf{p}}{\|\mathbf{p}\|^2}$

**Minima:**

**Strong Minimum:** if a scalar  $\delta > 0$  exists, such that

$F(\mathbf{x}) < F(\mathbf{x} + \Delta \mathbf{x})$  for all  $\Delta \mathbf{x}$  such that  $\delta > \|\Delta \mathbf{x}\| > 0$ .

**Global Minimum:** if  $F(\mathbf{x}) < F(\mathbf{x} + \Delta \mathbf{x})$  for all  $\Delta \mathbf{x} \neq 0$

**Weak Minimum:** if it is not a strong minimum, and a scalar  $\delta > 0$  exists, such that  $F(\mathbf{x}) \leq F(\mathbf{x} + \Delta \mathbf{x})$  for all  $\Delta \mathbf{x}$  such that  $\delta > \|\Delta \mathbf{x}\| > 0$ .

**Necessary Conditions for Optimality:**

**1<sup>st</sup>-Order Condition:**  $\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = 0$  (Stationary Points)

**2<sup>nd</sup>-Order Condition:**  $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \geq 0$  (Positive Semi-definite Hessian Matrix).

**Quadratic fn.:**  $F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{d}^T \mathbf{x} + c$

$\nabla F(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{d}$ ,  $\nabla^2 F(\mathbf{x}) = \mathbf{A}$ ,  $\lambda_{\min} \leq \frac{\mathbf{p}^T \mathbf{A} \mathbf{p}}{\|\mathbf{p}\|^2} \leq \lambda_{\max}$

<p><b>General Minimization Algorithm:</b>  <math>\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k</math> or <math>\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k</math></p> <p><b>Steepest Descent Algorithm:</b>  <math>\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k</math> where, <math>\mathbf{g}_k = \nabla F(\mathbf{x}) _{\mathbf{x}=\mathbf{x}_k}</math></p> <p><b>Stable Learning Rate:</b> <math>(\alpha_k = \alpha, \text{constant}) \alpha &lt; \frac{2}{\lambda_{\max}}</math>  <math>\{\lambda_1, \lambda_2, \dots, \lambda_n\}</math> Eigenvalues of Hessian matrix A</p> <p><b>Learning Rate to Minimize Along the Line:</b>  <math>\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \xrightarrow{\text{is}} \alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}</math> (For quadratic fn.)</p> <p><b>After Minimization Along the Line:</b>  <math>\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \Rightarrow \mathbf{g}_{k+1}^T \mathbf{p}_k = 0</math></p> <p><b>ADALINE:</b> <math>\mathbf{a} = \text{purelin}(\mathbf{Wp} + \mathbf{b})</math></p> <p><b>Mean Square Error:</b> (for ADALINE it is a quadratic fn.)  <math>F(\mathbf{x}) = E[e^2] = E[(t - a)^2] = E[(t - \mathbf{x}^T \mathbf{z})^2]</math>  <math>F(\mathbf{x}) = c - 2\mathbf{x}^T \mathbf{h} + \mathbf{x}^T \mathbf{R} \mathbf{x}</math>,</p> <p><math>c = E[t^2]</math>, <math>\mathbf{h} = E[t\mathbf{z}]</math> and <math>\mathbf{R} = E[\mathbf{zz}^T] \Rightarrow \mathbf{A} = 2\mathbf{R}</math>, <math>d = -2c</math>  Unique minimum, if it exists, is <math>\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}</math>,  where <math>\mathbf{x} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}</math> and <math>\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}</math></p> <p><b>LMS Algorithm:</b> <math>\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k) \mathbf{p}^T(k)</math>  <math>\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k)</math></p> <p><b>Convergence Point:</b> <math>\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}</math></p> <p><b>Stable Learning Rate:</b> <math>0 &lt; \alpha &lt; 1/\lambda_{\max}</math> where <math>\lambda_{\max}</math> is the maximum eigenvalue of R</p> <p><b>Adaptive Filter ADALINE:</b>  <math>a(k) = \text{purelin}(\mathbf{Wp}(k) + b) = \sum_{i=1}^R w_{1,i} y(k-i+1) + b</math></p> <p><b>Backpropagation Algorithm:</b></p> <p><b>Performance Index:</b>  Mean Square error: <math>F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})]</math></p> <p><b>Approximate Performance Index:</b> (single sample)  <math>\hat{F}(\mathbf{x}) = \mathbf{e}^T(k) \mathbf{e}(k) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k))</math></p> <p><b>Sensitivity:</b> <math>\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left[ \frac{\partial \hat{F}}{\partial n_1^m} \quad \frac{\partial \hat{F}}{\partial n_2^m} \quad \dots \quad \frac{\partial \hat{F}}{\partial n_s^m} \right]^T</math></p> <p><b>Forward Propagation:</b> <math>\mathbf{a}^0 = \mathbf{p}</math>,  <math>\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1})</math> for <math>m = 0, 1, \dots, M-1</math>  <math>\mathbf{a} = \mathbf{a}^M</math></p> <p><b>Backward Propagation:</b> <math>\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})</math>,  <math>\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}</math> for <math>m = M-1, \dots, 2, 1</math>, where  <math>\dot{\mathbf{F}}^m(\mathbf{n}^m) = \text{diag}([\dot{f}^m(n_1^m) \quad \dot{f}^m(n_2^m) \quad \dots \quad \dot{f}^m(n_s^m)])</math>  <math>\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}</math></p> <p><b>Weight Update (Approximate Steepest Descent):</b>  <math>\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T</math>  <math>\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m</math></p>	<p><b>*Heuristic Variations of Backpropagation:</b></p> <p><b>Batching:</b> The parameters are updated only after the entire training set has been presented. The gradients calculated for each training example are averaged together to produce a more accurate estimate of the gradient. (If the training set is complete, i.e., covers all possible input/output pairs, then the gradient estimate will be exact.)</p> <p><b>Backpropagation with Momentum (MOBP):</b>  <math>\Delta \mathbf{W}^m(k) = \gamma \Delta \mathbf{W}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T</math>  <math>\Delta \mathbf{b}^m(k) = \gamma \Delta \mathbf{b}^m(k-1) - (1-\gamma)\alpha \mathbf{s}^m</math></p> <p><b>Variable Learning Rate Backpropagation (VLBP)</b></p> <ol style="list-style-type: none"> <li>If the squared error (over the entire training set) increases by more than some set percentage <math>\zeta</math> (typically one to five percent) after a weight update, then the weight update is discarded, the learning rate is multiplied by some factor <math>\rho &lt; 1</math>, and the momentum coefficient <math>\gamma</math> (if it is used) is set to zero.</li> <li>If the squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by some factor <math>\eta &gt; 1</math>. If <math>\gamma</math> has been previously set to zero, it is reset to its original value.</li> <li>If the squared error increases by less than <math>\zeta</math>, then the weight update is accepted but the learning rate and the momentum coefficient are unchanged.</li> </ol> <p><b>Association:</b> <math>\mathbf{a} = \text{hardlim}(\mathbf{Wp} + \mathbf{b})</math>  An association is a link between the inputs and outputs of a network so that when a stimulus A is presented to the network, it will output a response B.</p> <p><b>Associative Learning Rules:</b></p> <p><b>Unsupervised Hebb Rule:</b>  <math>\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q) \mathbf{p}^T(q)</math></p> <p><b>Hebb with Decay:</b>  <math>\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha \mathbf{a}(q) \mathbf{p}^T(q)</math></p> <p><b>Instar:</b> <math>\mathbf{a} = \text{hardlim}(\mathbf{Wp} + \mathbf{b})</math>, <math>\mathbf{a} = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b)</math>  The instar is activated for <math>\mathbf{w}^T \mathbf{p} = \ \mathbf{w}\  \ \mathbf{p}\  \cos\theta \geq -b</math> where <math>\theta</math> is the angle between <math>\mathbf{p}</math> and <math>\mathbf{w}</math>.</p> <p><b>Instar Rule:</b>  <math>i\mathbf{w}(q) = i\mathbf{w}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - i\mathbf{w}(q-1))</math>  <math>i\mathbf{w}(q) = (1-\alpha) i\mathbf{w}(q-1) + \alpha \mathbf{p}(q), \text{ if } (a_i(q) = 1)</math></p> <p><b>Kohonen Rule:</b>  <math>i\mathbf{w}(q) = i\mathbf{w}(q-1) + \alpha (\mathbf{p}(q) - i\mathbf{w}(q-1)) \text{ for } i \in X(q)</math></p> <p><b>Outstar Rule:</b> <math>\mathbf{a} = \text{satlim}(\mathbf{Wp})</math>  <math>w_j(q) = w_j(q-1) + \alpha (a(q) - w_j(q-1)) p_j(q)</math></p> <p><b>Competitive Layer:</b> <math>\mathbf{a} = \text{compet}(\mathbf{Wp}) = \text{compet}(\mathbf{n})</math></p> <p><b>Competitive Learning with the Kohonen Rule:</b>  <math>i^*\mathbf{w}(q) = i^*\mathbf{w}(q-1) + \alpha (\mathbf{p}(q) - i^*\mathbf{w}(q-1))</math>  <math>= (1-\alpha) i^*\mathbf{w}(q-1) + \alpha \mathbf{p}(q)</math>  <math>i^*\mathbf{w}(q) = i^*\mathbf{w}(q-1), \quad i \neq i^*</math> where <math>i^*</math> is the winning neuron.</p> <p><b>Self-Organizing with the Kohonen Rule:</b>  <math>i\mathbf{w}(q) = i\mathbf{w}(q-1) + \alpha (\mathbf{p}(q) - i\mathbf{w}(q-1))</math>  <math>= (1-\alpha) i\mathbf{w}(q-1) + \alpha \mathbf{p}(q), \quad i \in N_{i^*}(d)</math>  <math>N_i(d) = \{j, d_{i,j} \leq d\}</math></p> <p><b>LVO Network:</b> (<math>w_{k,i}^2 = 1</math>) <math>\Rightarrow</math> subclass <math>i</math> is a part of class <math>k</math>  <math>n_i^1 = -\ \mathbf{w}^1 - \mathbf{p}\ , \mathbf{a}^1 = \text{compet}(\mathbf{n}^1), \mathbf{a}^2 = \mathbf{W}^2 \mathbf{a}^1</math></p> <p><b>LVQ Network Learning with the Kohonen Rule:</b>  <math>i^*\mathbf{w}^1(q) = i^*\mathbf{w}^1(q-1) + \alpha (\mathbf{p}(q) - i^*\mathbf{w}^1(q-1)),</math>  <math>\text{if } a_{k^*}^2 = t_{k^*} = 1</math>  <math>i^*\mathbf{w}^1(q) = i^*\mathbf{w}^1(q-1) - \alpha (\mathbf{p}(q) - i^*\mathbf{w}^1(q-1)),</math>  <math>\text{if } a_{k^*}^2 = 1 \neq t_{k^*} = 0</math></p> <p><b>hardlim:</b> <math>a = \begin{cases} 0 &amp; n &lt; 0 \\ 1 &amp; n \geq 0 \end{cases}</math>, <b>hardlims:</b> <math>a = \begin{cases} -1 &amp; n &lt; 0 \\ +1 &amp; n \geq 0 \end{cases}</math>, <b>purelin:</b> <math>a = n</math>, <b>Logsig:</b> <math>a = \frac{1}{1+e^{-n}}</math>, <b>tansig:</b> <math>a = \frac{e^n - e^{-n}}{e^n + e^{-n}}</math>, <b>postlin:</b> <math>a = \begin{cases} 0 &amp; n &lt; 0 \\ n &amp; n \geq 0 \end{cases}</math></p> <p><b>compet:</b> <math>a = \begin{cases} 1 &amp; \text{neuron with max } n \\ 0 &amp; \text{all other neurons} \end{cases}</math>, <b>satlin:</b> <math>a = \begin{cases} n &amp; 0 &lt; n &lt; 0 \\ -1 &amp; -1 \leq n \leq 1 \\ 1 &amp; n &gt; 1 \end{cases}</math>, <b>satlim:</b> <math>a = \begin{cases} -1 &amp; n &lt; 0 \\ n &amp; -1 \leq n \leq 1 \\ 1 &amp; n &gt; 1 \end{cases}</math></p> <p><b>Delay:</b> <math>a(t) = u(t-1)</math>, <b>Integrator:</b> <math>a(t) = \int_0^t u(\tau) d\tau + a(0)</math></p> <p><b>**HINT:</b>  <math>\text{diag}([1 \ 2 \ 3]) = \begin{bmatrix} 1 &amp; 0 &amp; 0 \\ 0 &amp; 2 &amp; 0 \\ 0 &amp; 0 &amp; 3 \end{bmatrix}</math></p>
---	--

# MACHINE LEARNING IN EMOJI

SUPERVISED

UNSUPERVISED

REINFORCEMENT

- **SUPERVISED** human builds model based on input / output  
human input, machine output
- **UNSUPERVISED** human utilizes if satisfactory  
human input, machine output
- **REINFORCEMENT** human reward/punish, cycle continues  
human input, machine output

## BASIC REGRESSION

- **LINEAR** linear\_model.LinearRegression()  
Lots of numerical data   
- **LOGISTIC** linear\_model.LogisticRegression()  
Target variable is categorical  or 

## CLASSIFICATION

- **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 
- **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 

## CLUSTER ANALYSIS

- **K-MEANS** cluster.KMeans()  
Similar datum into groups based on centroids 
- **ANOMALY DETECTION** covariance.EllipticalEnvelope()  
Finding outliers through grouping    

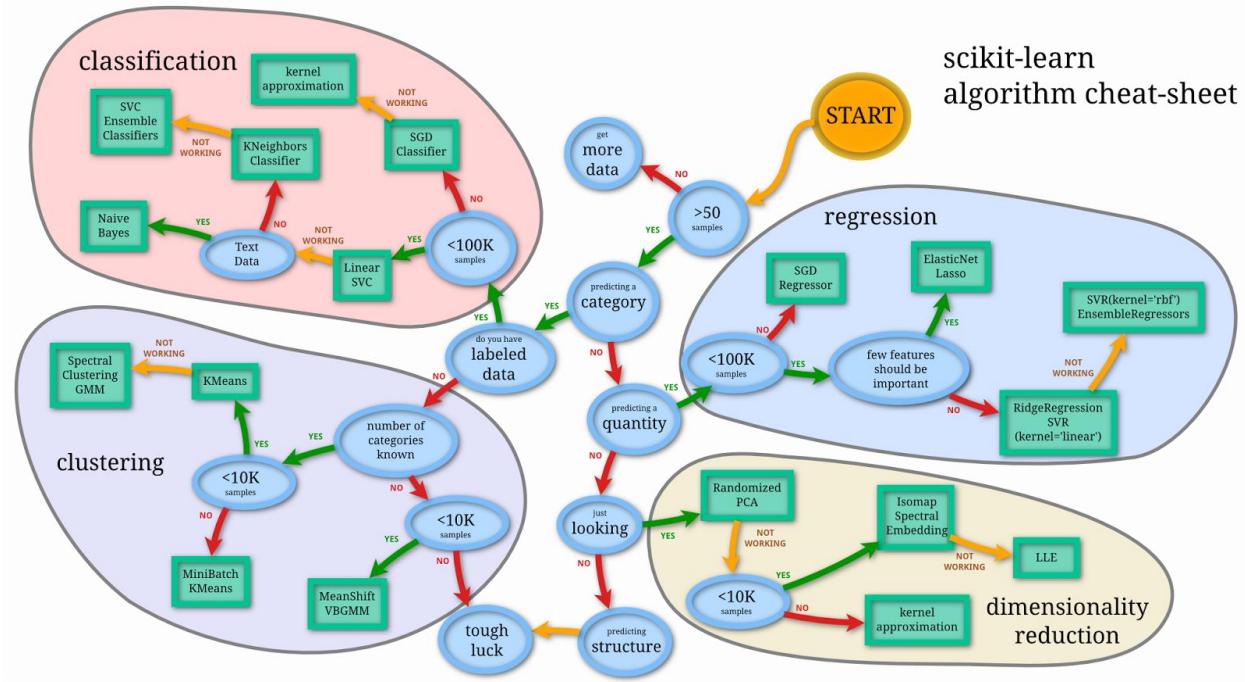
## FEATURE REDUCTION

- T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING** manifold.TSNE()  
Visualize 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** Ida.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**  $TP / (TP + FP)$
- Specificity Function**  $TN / (FP + TN)$
- Sensitivity Function**  $TP / (TP + FN)$

@emilyinamillion made this



## Scikit-Learn

**Scikit-learn** (formerly [scikits.learn](#)) is a free software machine learning library for the Python programming language. It features various [classification](#), [regression](#) and [clustering](#) algorithms including [support vector machines](#), [random forests](#), [gradient boosting](#), [k-means](#) and [DBSCAN](#), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](#) and [SciPy](#).

## Python For Data Science Cheat Sheet

### Scikit-Learn

Learn Python for data science interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

##### A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

#### Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','M','F','F','F','F'])
>>> X[X < 0.7] = 0
```

#### Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
...                                                    y,
...                                                    random_state=0)
```

#### Preprocessing The Data

##### Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

##### Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

##### Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

## Create Your Model

### Supervised Learning Estimators

**Linear Regression**  
>>> from sklearn.linear\_model import LinearRegression  
>>> lr = LinearRegression(normalize=True)  
**Support Vector Machines (SVM)**  
>>> from sklearn.svm import SVC  
>>> svc = SVC(kernel='linear')  
**Naive Bayes**  
>>> from sklearn.naive\_bayes import GaussianNB  
>>> gnb = GaussianNB()  
**KNN**  
>>> from sklearn import neighbors  
>>> knn = neighbors.KNeighborsClassifier(n\_neighbors=5)

### Unsupervised Learning Estimators

**Principal Component Analysis (PCA)**  
>>> from sklearn.decomposition import PCA  
>>> pca = PCA(n\_components=0.95)  
**K-Means**  
>>> from sklearn.cluster import KMeans  
>>> k\_means = KMeans(n\_clusters=3, random\_state=0)

### Model Fitting

#### Supervised Learning

```
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```

#### Unsupervised Learning

```
>>> k_means.fit(X_train)
>>> k_means = KMeans(n_clusters=3, random_state=0)
>>> pca.fit(X_train)
```

Fit the model to the data

Fit the model to the data

Fit to data, then transform it

### Prediction

#### Supervised Estimators

```
>>> y_pred = svc.predict(np.random.random((2,5)))
>>> y_pred = lr.predict(X_test)
>>> y_pred = knn.predict_proba(X_test)
```

#### Unsupervised Estimators

```
>>> y_pred = k_means.predict(X_test)
```

Predict labels

Predict labels

Estimate probability of a label

Predict labels in clustering algs

## Evaluate Your Model's Performance

### Classification Metrics

**Accuracy Score**  
>>> knn.score(X\_test, y\_test)
>>> from sklearn.metrics import accuracy\_score
>>> accuracy\_score(y\_test, y\_pred)

**Classification Report**  
>>> from sklearn.metrics import classification\_report
>>> print(classification\_report(y\_test, y\_pred))

**Confusion Matrix**  
>>> from sklearn.metrics import confusion\_matrix
>>> print(confusion\_matrix(y\_test, y\_pred))

Estimator score method  
Metric scoring functions

Precision, recall, f1-score and support

### Regression Metrics

**Mean Absolute Error**  
>>> from sklearn.metrics import mean\_absolute\_error
>>> mae = mean\_absolute\_error(y\_true, y\_pred)

**Mean Squared Error**  
>>> from sklearn.metrics import mean\_squared\_error
>>> mse = mean\_squared\_error(y\_true, y\_pred)

**R<sup>2</sup> Score**  
>>> from sklearn.metrics import r2\_score
>>> r2\_score(y\_true, y\_pred)

### Clustering Metrics

**Adjusted Rand Index**  
>>> from sklearn.metrics import adjusted\_rand\_score
>>> adjusted\_rand\_score(y\_true, y\_pred)

**Homogeneity**  
>>> from sklearn.metrics import homogeneity\_score
>>> homogeneity\_score(y\_true, y\_pred)

**V-measure**  
>>> from sklearn.metrics import v\_measure\_score
>>> metrics.v\_measure\_score(y\_true, y\_pred)

### Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

### Tune Your Model

#### Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
...            "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
...                      param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

#### Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
...            "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
...                               param_distributions=params,
...                               n_iter=8,
...                               n_jobs=-1,
...                               random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

DataCamp  
Learn Python for Data Science interactively



# Python for Data Science

## Python For Data Science Cheat Sheet

### Python Basics

Learn More Python for Data Science [Interactively](#) at [www.datacamp.com](#)



### Variables and Data Types

#### Variable Assignment

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

#### Calculations With Variables

>>> x+2 7 >>> x-2 3 >>> x*2 10 >>> x**2 25 >>> x%2 1 >>> x/float(2) 2.5	Sum of two variables Subtraction of two variables Multiplication of two variables Exponentiation of a variable Remainder of a variable Division of a variable
--	--

#### Types and Type Conversion

str()  int()  float()  bool()	*5, "3.45", 'True' 5, 3, 1 5.0, 1.0 True, True, True	Variables to strings Variables to integers Variables to floats Variables to booleans
---	---	---

### Asking For Help

```
>>> help(str)
```

### Strings

```
>>> my_string = 'thisStringIsAwesome'  
>>> my_string  
'thisStringIsAwesome'
```

#### String Operations

```
>>> my_string * 2  
'thisStringIsAwesome>thisStringIsAwesome'  
>>> my_string + "Innit"  
'thisStringIsAwesomeInnit'  
>>> 'm' in my_string  
True
```

### Lists

#### Also see NumPy Arrays

```
>>> a = 'is'  
>>> b = 'nice'  
>>> my_list = ['my', 'list', a, b]  
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

#### Selecting List Elements

Index starts at 0

##### Subset

```
>>> my_list[1]  
>>> my_list[-3]
```

Select item at index 1  
Select 3rd last item

##### Slice

```
>>> my_list[1:3]  
>>> my_list[1:]  
>>> my_list[:3]  
>>> my_list[::]
```

Select items at index 1 and 2  
Select items after index 0  
Select items before index 3  
Copy my\_list

```
>>> my_list[1][0]  
>>> my_list2[1][2]
```

my\_list[list][itemOfList]

#### List Operations

```
>>> my_list + my_list  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list * 2  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list2 > 4  
True
```

#### List Methods

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

### String Operations

Index starts at 0

```
>>> my_string[3]  
>>> my_string[4:9]
```

#### String Methods

>>> my_string.upper() >>> my_string.lower() >>> my_string.count("e") >>> my_string.replace("o", "i") >>> my_string.strip()	String to uppercase String to lowercase Count String elements Replace String elements Strip whitespace from ends
--	--

### Libraries

#### Import libraries

```
>>> import numpy  
>>> import numpy as np  
>>> Selective import  
>>> from math import pi
```

#### pandas

Data analysis

#### Machine learning

Machine learning

Scientific computing

2D plotting

### Install Python



Leading open data science platform powered by Python



Free IDE that is included with Anaconda



Create and share documents with live code, visualizations, text, ...

### Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]  
>>> my_array = np.array(my_list)  
>>> my_2darray = np.array(([1,2,3],[4,5,6]))
```

#### Selecting Numpy Array Elements

Index starts at 0

##### Subset

```
>>> my_array[1]  
2
```

Select item at index 1

##### Slice

```
>>> my_array[0:2]  
array([1, 2])
```

Select items at index 0 and 1

##### Subset 2D Numpy arrays

```
>>> my_2darray[:,0]  
array([1, 4])
```

my\_2darray[rows, columns]

### Numpy Array Operations

```
>>> 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([6, 8, 10, 12])
```

### Numpy Array Functions

>>> my_array.shape >>> np.append(other_array) >>> np.insert(my_array, 1, 5) >>> np.delete(my_array, [1]) >>> np.mean(my_array) >>> np.median(my_array) >>> my_array.corrcoef() >>> np.std(my_array)	Get the dimensions of the array Append items to an array Insert items in an array Delete items in an array Mean of the array Median of the array Correlation coefficient Standard deviation
--	--

DataCamp

Learn Python for Data Science [Interactively](#)



## Python For Data Science Cheat Sheet

### Bokeh

Learn Bokeh [Interactively](#) at [www.DataCamp.com](http://www.DataCamp.com), taught by Bryan Van de Ven, core contributor



#### Plotting With Bokeh

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.



The basic steps to creating plots with the `bokeh.plotting` interface are:

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
- ```
>>> from bokeh.plotting import figure
>>> from bokeh.io import output_file, show
>>> x = [1, 2, 3, 4, 5] Step 1
>>> y = [6, 7, 2, 4, 5]
>>> p = figure(title="simple line example",
    x_axis_label='x',
    y_axis_label='y')
>>> p.line(x, y, legend="Temp.", line_width=2) Step 2
>>> output_file("temp.html") Step 3
>>> show(p) Step 4
```

#### 1 Data

[Also see Lists, NumPy & Pandas](#)

Under the hood, your data is converted to Column Data Sources. You can also do this manually:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.array([[33.9, 4.65, 'US'],
    [32.4, 4.66, 'Asia'],
    [21.4, 4.109, 'Europe']]),
    columns=['mpg', 'cyl', 'hp', 'origin'],
    index=['Toyota', 'Fiat', 'Volvo'])
>>> from bokeh.models import ColumnDataSource
>>> cds_df = ColumnDataSource(df)
```

#### 2 Plotting

```
>>> from bokeh.plotting import figure
>>> p1 = figure(plot_width=300, tools='pan,box_zoom')
>>> p2 = figure(plot_width=300, plot_height=300,
    x_range=(0, 8), y_range=(0, 8))
>>> p3 = figure()
```

### 3 Renderers & Visual Customizations

#### Glyphs

##### Scatter Markers

```
>>> p1.circle(np.array([(1,2,3)], np.array([3,2,1])),
    fill_color='white')
>>> p2.square(np.array([(1,5,3,5,5,5)], [1,4,3],
    color='blue', size=1)
```

##### Line Glyphs

```
>>> p1.line([(1,2,3,4), [3,4,5,6], line_width=2)
>>> p2.multi_line(pd.DataFrame([(1,2,3),(5,6,7)]),
    pd.DataFrame([(3,4,5),(3,2,1)]), color="blue")
```

##### Rows & Columns Layout

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| Rows                                | Columns                              |
| >>> from bokeh.layouts import row   | >>> from bokeh.layouts import column |
| >>> layout = row(p1,p2, p3)         | >>> layout = column(p1,p2,p3)        |
| Nesting Rows & Columns              |                                      |
| >>> layout = row(column(p1,p2), p3) |                                      |

##### Grid Layout

```
>>> from bokeh.layouts import gridplot
>>> row1 = [p1,p2]
>>> row2 = [p3]
>>> layout = gridplot([[p1,p2],[p3]])
```

##### Tabbed Layout

```
>>> from bokeh.models.widgets import Panel, Tabs
>>> tab1 = Panel(child=p1, title="tab1")
>>> tab2 = Panel(child=p2, title="tab2")
>>> layout = Tabs(tabs=[tab1, tab2])
```

##### Legends

###### Legend Location

Inside Plot Area

>>> p.legend.location = 'bottom\_left'

###### Outside Plot Area

>>> r1 = p2.asterisk(np.array([(1,2,3)]), np.array([(3,2,1)]))

>>> r2 = p2.line([(1,2,3,4), [3,4,5,6], line\_width=2)

>>> legend = Legend(items=[("One", [r1]), ("Two", [r2])], location=(0, -30))

>>> p.add\_layout(legend, "right")

##### Output

###### Output to HTML File

```
>>> from bokeh.io import output_file, show
>>> output_file('my_bar_chart.html', mode='cdn')
```

###### Notebook Output

```
>>> from bokeh.io import output_notebook, show
>>> output_notebook()
```

##### Embedding

###### Standalone HTML

```
>>> from bokeh.embed import file_html
>>> html = file_html(p, CDN, "my_plot")
Components
>>> from bokeh.embed import components
>>> script, div = components(p)
```

#### 5 Show or Save Your Plots

|                  |                  |
|------------------|------------------|
| >>> show(p1)     | >>> save(p1)     |
| >>> show(layout) | >>> save(layout) |

#### Customized Glyphs

[Also see Data](#)

##### Selection and Non-Selection Glyphs

```
>>> p.circle('mpg', 'cyl', source=cds_df,
```

selection\_color='red', nonselection\_alpha=0.1)

##### Hover Glyphs

```
>>> hover = HoverTool(tooltips=None, mode='vline')
```

>>> p.add\_tools(hover)

##### Colormapping

```
>>> color_mapper = CategoricalColorMapper(factors=['Europe', 'Asia', 'US'],
```

palette=['red', 'green', 'blue'])

```
>>> p.circle('mpg', 'cyl', source=cds_df,
```

color=dict(fuel='origin', transform=color\_mapper), legend='origin')

[Also see Data](#)

#### Linked Plots

##### Linked Axes

>>> p2.x\_range = pl.x\_range

>>> p2.y\_range = pl.y\_range

##### Linked Brushing

>>> p4 = figure(plot\_width = 100, tools='box\_select,lasso\_select')

>>> p4.circle('mpg', 'cyl', source=cds\_df)

>>> p5 = Figure(plot\_width = 200, tools='box\_select,lasso\_select')

>>> p5.circle('mpg', 'hp', source=cds\_df)

>>> layout = row(p4,p5)

[Also see Data](#)

#### Legend Orientation

>>> p.legend.orientation = "horizontal"

>>> p.legend.orientation = "vertical"

#### Legend Background & Border

>>> p.legend.border\_line\_color = "navy"

>>> p.legend.background\_fill\_color = "white"

#### Statistical Charts With Bokeh

[Also see Data](#)

Bokeh's high-level `bokeh.charts` interface is ideal for quickly creating statistical charts

##### Bar Chart

>>> from bokeh.charts import Bar

>>> p = Bar(df, stacked=True, palette=['red','blue'])

##### Box Plot

>>> from bokeh.charts import BoxPlot

>>> p = BoxPlot(df, values='vals', label='cyl', legend='bottom\_right')

##### Histogram

>>> from bokeh.charts import Histogram

>>> p = Histogram(df, title='Histogram')

##### Scatter Plot

>>> from bokeh.charts import Scatter

>>> p = Scatter(df, x='mpg', y='hp', marker='square',

xlabel='Miles Per Gallon', ylabel='Horsepower')

DataCamp

Learn Python for Data Science Interactively

## TensorFlow

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.

## About

### TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

### Skflow

Scikit Flow provides a set of high level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code. Scikit Flow is a simplified interface for TensorFlow, to get people started on predictive analytics and data mining. Scikit Flow has been merged into TensorFlow since version 0.8 and now called TensorFlow Learn.

### Keras

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano

## 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/$device/tensorflow-0.8.0-$python_version-none-linux_x86_64.whl
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
help(object)
Get help for object (list of available
methods, attributes, signatures and so on)
dir(object)
Get list of object attributes
(fields, functions)
str(object)
Transform an object to string
object?
Shows documentations about the object
globals()
Return the dictionary containing the
current scope's global variables.
locals()
Update and return a dictionary containing
the current scope's local variables.
```

### id(object)

Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.

```
import __builtin__
dir(__builtin__)
```

Other built-in functions

## TensorFlow

### Main classes

```
tf.Graph()
tf.Operation()
tf.Tensor()
tf.Session()
```

### Some useful functions

```
tf.get_default_session()
tf.get_default_graph()
tf.reset_default_graph()
ops.reset_default_graph()
tf.device("/cpu:0")
tf.name_scope(value)
tf.convert_to_tensor(value)
```

### TensorFlow Optimizers

```
GradientDescentOptimizer
AdadeltaOptimizer
AdagradOptimizer
MomentumOptimizer
AdamOptimizer
FtrlOptimizer
RMSPropOptimizer
```

### Reduction

```
reduce_sum
reduce_prod
reduce_min
reduce_max
reduce_mean
reduce_all
reduce_any
accumulate_n
```

### Activation functions

```
tf.nn?
relu
relu6
elu
softplus
softsign
dropout
bias_add
sigmoid
tanh
sigmoid_cross_entropy_with_logits
softmax
log_softmax
softmax_cross_entropy_with_logits
sparse_softmax_cross_entropy_with_logits
weighted_cross_entropy_with_logits
etc.
```

## Skflow

### Main classes

```
TensorFlowClassifier
TensorFlowRegressor
TensorFlowDNNClassifier
TensorFlowDNNRegressor
TensorFlowLinearClassifier
TensorFlowLinearRegressor
TensorFlowRNNClassifier
TensorFlowRNNRegressor
```

## TensorFlowEstimator

### Each classifier and regressor have

**following fields**  
**n\_classes=0** (Regressor), **n\_classes** are  
**expected to be input** (Classifiers)  
batch\_size=32,  
steps=200, // except  
TensorFlowRNNClassifier - there is 50  
optimizer='Adagrad',  
learning\_rate=0.1,

## Keras

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than an end-to-end machine-learning framework. It presents a higher-level, more intuitive set of abstractions that make it easy to configure neural networks regardless of the backend scientific computing library.

**Python For Data Science Cheat Sheet**

**Keras**

Learn Python for data science interactively at [www.DataCamp.com](http://www.DataCamp.com)

**Keras**

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.

**A Basic Example**

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2, size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32, activation='relu',
    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['accuracy'])
>>> model.fit(data, labels, epochs=10, batch_size=32)
>>> predictions = model.predict(data)
```

**Data** *Also see NumPy, Pandas & Scikit-Learn*

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

**Keras Data Sets**

```
>>> from keras.datasets import boston_housing,
    mnist,
    cifar10,
    imdb
>>> (x_train,y_train), (x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2), (x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3), (x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4), (x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

**Other**

```
>>> from urllib import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"), delimiter=",")
>>> y = data[:,0:8]
>>> x = data[:,8:]
```

**Preprocessing**

**Sequence Padding**

```
>>> from keras.preprocessing import sequence
>>> train, test = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

**One-Hot Encoding**

```
>>> from keras.utils import to_categorical
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
>>> Y_test4 = to_categorical(y_test4, num_classes)
```

**Model Architecture**

**Sequential Model**

```
>>> from keras.models import Sequential
>>> model1 = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

**Multilayer Perceptron (MLP)**

```
>>> from keras.layers import Dense
>>> model.add(Dense(128, input_dim=8,
    kernel_initializer='uniform',
    activation='relu'))
>>> model.add(Dense(64, kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1, kernel_initializer='uniform',activation='sigmoid'))
```

**Multi-Class Classification**

```
>>> from keras.layers import Dropout
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

**Regression**

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

**Convolutional Neural Network (CNN)**

```
>>> from keras.layers import Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Flatten())
>>> model2.add(Dense(128))
>>> model2.add(Dropout(0.25))
>>> model2.add(Dense(64,(3, 3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

**Recurrent Neural Network (RNN)**

```
>>> from keras.layers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

**Train and Test Sets** *Also see NumPy & Scikit-Learn*

```
>>> from sklearn.model_selection import train_test_split
>>> x_train,x_test,y_train,y_test = train_test_split(x,
    y,
    test_size=0.33,
    random_state=42)
```

**Standardization/Normalization**

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train)
>>> standardized_X = scaler.transform(x_train)
>>> standardized_X_test = scaler.transform(x_test)
```

**Inspect Model**

|                         |                                      |
|-------------------------|--------------------------------------|
| >>> model.output_shape  | Model output shape                   |
| >>> model.summary()     | Model summary representation         |
| >>> model.get_config()  | Model configuration                  |
| >>> model.get_weights() | List all weight tensors in the model |

**Compile Model**

|                                                                                             |                                                                                                     |
|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <b>MLP: Binary Classification</b>                                                           | MLP: Multi-Class Classification                                                                     |
| >>> model.compile(optimizer='adam',<br>loss='binary_crossentropy',<br>metrics=['accuracy']) | >>> model.compile(optimizer='rmsprop',<br>loss='categorical_crossentropy',<br>metrics=['accuracy']) |
| <b>MLP: Regression</b>                                                                      |                                                                                                     |
| >>> model.compile(optimizer='rmsprop',<br>loss='mse',<br>metrics=['mae'])                   |                                                                                                     |

**Recurrent Neural Network**

```
>>> model3.compile(loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])
```

**Model Training**

```
>>> model3.fit(x_train4,
    y_train4,
    batch_size=32,
    epochs=15,
    verbose=1,
    validation_data=(x_test4,y_test4))
```

**Evaluate Your Model's Performance**

```
>>> score = model3.evaluate(x_test,
    y_test,
    batch_size=32)
```

**Prediction**

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

**Save / Reload Models**

```
>>> from keras.models import load_model
>>> model3.save('model.h5')
>>> my_model = load_model('my_model.h5')
```

**Model Fine-tuning**

**Optimization Parameters**

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model3.compile(loss='categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy'])
```

**Early Stopping**

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
    y_train4,
    batch_size=32,
    epochs=15,
    validation_data=(x_test4,y_test4),
    callbacks=[early_stopping_monitor])
```

**DataCamp**  
Learn Python for Data Science interactively

## Numpy

NumPy targets the [CPython](#) reference [implementation](#) of Python, which is a non-optimizing [bytecode](#) interpreter. Mathematical algorithms written for this version of Python often run much slower than [compiled](#) equivalents. NumPy address the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

## Python For Data Science Cheat Sheet

### NumPy Basics

Learn Python for Data Science interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### NumPy

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.

Use the following import convention:

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



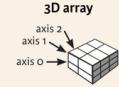
#### NumPy Arrays

##### 1D array

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
|---|---|---|

##### 2D array

|     |   |   |
|-----|---|---|
| 1.5 | 2 | 3 |
| 4   | 5 | 6 |



#### Creating Arrays

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

#### Initial Placeholders

```
>>> np.zeros((3,4))
>>> np.ones(2,(3,4),dtype=np.int16)
>>> d = np.arange(10,25,5)
>>> np.linspace(0,2,9)
>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty((3,2))
```

Create an array of zeros  
Create an array of ones  
Create an array of evenly spaced values (step value)  
Create an array of evenly spaced values (number of samples)  
Create a constant array  
Create a 2x2 identity matrix  
Create an array with random values  
Create an empty array

#### I/O

##### Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savetxt('array.npz', a, b)
>>> np.load('my_array.npy')
```

##### Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

#### Data Types

```
>>> np.int64
Signed 64-bit integer type
>>> np.float32
Standard double-precision floating point
>>> np.complex
Complex numbers represented by 128 floats
>>> np.bool
Boolean type storing TRUE and FALSE values
>>> np.object
Python object type
>>> np.string_
Fixed-length string type
>>> np.unicode_
Fixed-length unicode type
```

#### Inspecting Your Array

```
a.shape
Length of array
a.ndim
Number of array dimensions
a.size
Data type of array elements
a.dtype.name
Convert an array to a different type
```

#### Asking For Help

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

#### Array Mathematics

##### Arithmetic Operations

```
>>> g = a - b
array([[-0.5,  0.,  0.],
       [-3., -3., -3.]])
>>> np.subtract(a,b)
>>> b + a
array([(1.5, 4., 6.),
       (5., 7., 9.)])
>>> np.add(b,a)
>>> a / b
array([[ 0.66666667,  1.        ,  0.5       ],
       [ 0.25,  0.4,  0.5       ]])
>>> np.divide(a,b)
>>> a * b
array([ 1.5,  4.,  9.],
      [ 5.,  10., 18.])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.sqrt(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> np.log10(f)
array([ 7.,  7.],
```

##### Subtraction

##### Addition

##### Division

##### Multiplication

##### Exponentiation

##### Square root

##### Print sines of an array

##### Element-wise cosine

##### Element-wise natural logarithm

##### Dot product

#### Comparison

```
>>> a == b
array([[False, True, True],
       [False, False, False]], dtype=bool)
>>> a < 2
array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)
```

##### Element-wise comparison

##### Element-wise comparison

##### Array-wise comparison

#### Aggregate Functions

```
>>> a.sum()
Array-wise sum
>>> a.min()
Array-wise minimum value
>>> b.max(axis=0)
Maximum value of an array row
>>> b.cumsum(axis=1)
Cumulative sum of the elements
>>> a.mean()
Mean
>>> a.median()
Median
>>> a.correlcoef()
Correlation coefficient
>>> np.std(b)
Standard deviation
```

##### Array-wise sum

##### Array-wise minimum value

##### Maximum value of an array row

##### Cumulative sum of the elements

##### Mean

##### Median

##### Correlation coefficient

##### Standard deviation

#### Copying Arrays

```
>>> h = a.view()
Create a view of the array with the same data
>>> np.copy(a)
Create a copy of the array
>>> h = a.copy()
Create a deep copy of the array
```

#### Sorting Arrays

```
>>> a.sort()
Sort an array
>>> c.argsort(axis=0)
Sort the elements of an array's axis
```

#### Subsetting, Slicing, Indexing

Also see Lists

Select the element at the 2nd index

Select the element at row 0 column 2 (equivalent to b[1][2])

Select items at index 0 and 1

Select items at rows 0 and 1 in column 1

Select all items at row 0 (equivalent to b[0,:,:])

Same as [1,:,:]

Reversed array a

Select elements from a less than 2

Select elements (1,0),(0,1),(1,2) and (0,0)

Select a subset of the matrix's rows and columns

#### Array Manipulation

##### Transposing Array

```
>>> i = np.transpose(b)
```

i.T

##### Changing Array Shape

```
>>> b.ravel()
```

g.reshape(3,-2)

##### Adding/Removing Elements

```
>>> h.resize(2,(6))
```

Append items to an array

Insert items in an array

Delete items from an array

##### Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
```

array([ 1., 2., 3., 10., 15., 20.])

>>> np.vstack((a,b))

array([[ 1., 2., 3.],
 [ 4., 5., 6.],

[ 1., 2., 3.]]))

>>> np.r\_[e,f]

array([ 7., 7., 1., 0.,

[ 7., 7., 0., 1.]))

>>> np.column\_stack((a,d))

array([[ 1., 2., 3., 10., 15., 20.],

[ 4., 5., 6., 21., 22., 23.],

[ 1., 2., 3., 24., 25., 26.]]))

>>> np.c\_[a,d]

array([ 1., 2., 3., 10., 15., 20.],

[ 4., 5., 6., 21., 22., 23.],

[ 1., 2., 3., 24., 25., 26.]]))

>>> np.hsplit(a,3)

array([[[ 1., 2., 3.],

[ 4., 5., 6.],

[ 1., 2., 3.]]))

>>> np.vsplit(c,2)

array([[[ 1., 2., 3.],

[ 4., 5., 6.],

[ 1., 2., 3.]]],

[[[ 7., 7., 1., 0.,

[ 7., 7., 0., 1.]]])

>>> np.hsplit(c,3)

array([[[ 1., 2., 3.],

[ 4., 5., 6.],

[ 1., 2., 3.]]],

[[[ 7., 7., 0., 1.],

[ 7., 7., 1., 0.]]])

DataCamp

Learn Python for Data Science interactively



## Pandas

The name ‘Pandas’ is derived from the term “[panel data](#)”, an [econometrics](#) term for multidimensional structured data sets.

## Python For Data Science Cheat Sheet

### Pandas Basics

Learn Python for Data Science interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Pandas

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



Use the following import convention:

```
>>> import pandas as pd
```

#### Pandas Data Structures

##### Series

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



```
>>> s = pd.Series([3, -5, 7, 1], index=['a', 'b', 'c', 'd'])
```

##### DataFrame

Columns → A two-dimensional labeled data structure with columns of potentially different types

|   | Country | Capital   | Population |
|---|---------|-----------|------------|
| 1 | Belgium | Brussels  | 11190846   |
| 2 | India   | New Delhi | 1803171035 |
| 3 | Brazil  | Brasilia  | 207847528  |

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
   'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
   'Population': [11190846, 1803171035, 207847528]}
>>> df = pd.DataFrame(data,
   columns=['Country', 'Capital', 'Population'])
```

#### I/O

##### Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

##### Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Read multiple sheets from the same file
>>> xlxs = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlxs, 'Sheet1')
```

### Asking For Help

```
>>> help(pd.Series.loc)
```

Also see NumPy Arrays

#### Selection

##### Getting

|                                                                                                        |                           |
|--------------------------------------------------------------------------------------------------------|---------------------------|
| >>> s['b']<br>-5                                                                                       | Get one element           |
| >>> df[1]<br>Country Capital Population<br>1 India New Delhi 1303171035<br>2 Brazil Brasilia 207847528 | Get subset of a DataFrame |

#### Selecting, Boolean Indexing & Setting

##### By Position

|                                    |                                            |
|------------------------------------|--------------------------------------------|
| >>> df.iloc[[0], [0]]<br>'Belgium' | Select single value by row & column        |
| >>> df.iat[[0], [0]]<br>'Belgium'  | Select single value by row & column labels |

##### By Label

|                                           |                                             |
|-------------------------------------------|---------------------------------------------|
| >>> df.loc[[0], ['Country']]<br>'Belgium' | Select single row of subset of rows         |
| >>> df.at[[0], ['Country']]<br>'Belgium'  | Select a single column of subset of columns |

##### By Label/Position

|                                                                            |                                                                                                  |
|----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| >>> df.ix[2]<br>Country Brazil<br>Capital Brasilia<br>Population 207847528 | Select rows and columns                                                                          |
| >>> df.ix[:, ['Capital']]<br>0 Brussels<br>1 New Delhi<br>2 Brasilia       | Series s where value is not >= 0<br>s where value is <-1 or >2<br>Use filter to adjust DataFrame |

##### Boolean Indexing

|                                              |         |
|----------------------------------------------|---------|
| >>> s[~(s > 1)]<br>>>> s[(s < -1)   (s > 2)] | Setting |
|----------------------------------------------|---------|

|                                     |                              |
|-------------------------------------|------------------------------|
| >>> df[df['Population']>1200000000] | Set index a of Series s to 6 |
|-------------------------------------|------------------------------|

### Dropping

|                                |                                  |
|--------------------------------|----------------------------------|
| >>> s.drop(['a', 'c'])         | Drop values from rows (axis=0)   |
| >>> df.drop('Country', axis=1) | Drop values from columns(axis=1) |

### Sort & Rank

|                                 |                             |
|---------------------------------|-----------------------------|
| >>> df.sort_index(by='Country') | Sort by row or column index |
| >>> s.order()                   | Sort a series by its values |
| >>> df.rank()                   | Assign ranks to entries     |

### Retrieving Series/DataFrame Information

#### Basic Information

|                |                            |
|----------------|----------------------------|
| >>> df.shape   | (rows,columns)             |
| >>> df.index   | Describe index             |
| >>> df.columns | Describe DataFrame columns |
| >>> df.info()  | Info on DataFrame          |
| >>> df.count() | Number of non-NA values    |

#### Summary

|                             |                             |
|-----------------------------|-----------------------------|
| >>> df.sum()                | Sum of values               |
| >>> df.cumsum()             | Cumulative sum of values    |
| >>> df.amin() / dfamax()    | Minimum/maximum values      |
| >>> df.idmin() / df.idmax() | Minimum/Maximum index value |
| >>> df.describe()           | Summary statistics          |
| >>> df.mean()               | Mean of values              |
| >>> df.median()             | Median of values            |

#### Applying Functions

|                       |                             |
|-----------------------|-----------------------------|
| >>> f = lambda x: x*x | Apply function              |
| >>> df.apply(f)       | Apply function element-wise |

#### Data Alignment

##### Internal Data Alignment

NA values are introduced in the indices that don't overlap:

|                                                          |
|----------------------------------------------------------|
| >>> s3 = pd.Series([-2, 3, 1, 0], index=['a', 'c', 'd']) |
| >>> s + s3                                               |
| a 10.0                                                   |
| b NaN                                                    |
| c 5.0                                                    |
| d 7.0                                                    |

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

|                                                          |
|----------------------------------------------------------|
| >>> s3 = pd.Series([-2, 3, 1, 0], index=['a', 'c', 'd']) |
| >>> a 10.0                                               |
| b -5.0                                                   |
| c 5.0                                                    |
| d 7.0                                                    |

|                              |
|------------------------------|
| >>> s3.fill_value=2          |
| >>> s3.add(s3, fill_value=0) |
| a 10.0                       |
| b 2.0                        |
| c 2.0                        |
| d 2.0                        |

|                              |
|------------------------------|
| >>> s3.fill_value=4          |
| >>> s3.add(s3, fill_value=2) |
| a 10.0                       |
| b 4.0                        |
| c 4.0                        |
| d 4.0                        |

|                              |
|------------------------------|
| >>> s3.fill_value=3          |
| >>> s3.add(s3, fill_value=1) |
| a 10.0                       |
| b 3.0                        |
| c 3.0                        |
| d 3.0                        |

DataCamp

Learn Python for Data Science interactively



## Data Wrangling

The term “data wrangler” is starting to infiltrate pop culture. In the 2017 movie [Kong: Skull Island](#), one of the characters, played by actor [Marc Evan Jackson](#) is introduced as “Steve Woodward, our data wrangler”.

# Data Wrangling

## with pandas

### Cheat Sheet

<http://pandas.pydata.org>

#### Syntax – Creating DataFrames

|   | a | b | c  |
|---|---|---|----|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index=[1, 2, 3])
Specify values for each column.
```

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
Specify values for each row.
```

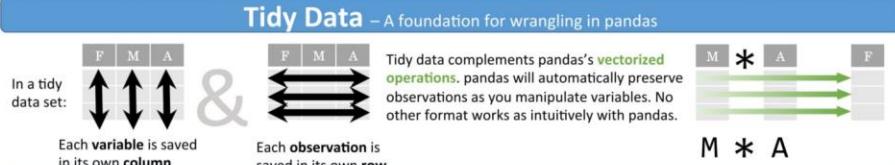
|   | a | b | c  |
|---|---|---|----|
| n | v |   |    |
| d | 1 | 4 | 7  |
| e | 2 | 5 | 10 |
| f | 2 | 6 | 9  |
| g | 2 | 7 | 12 |

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index=pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
        names=['n','v']))
Create DataFrame with a MultiIndex
```

#### Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

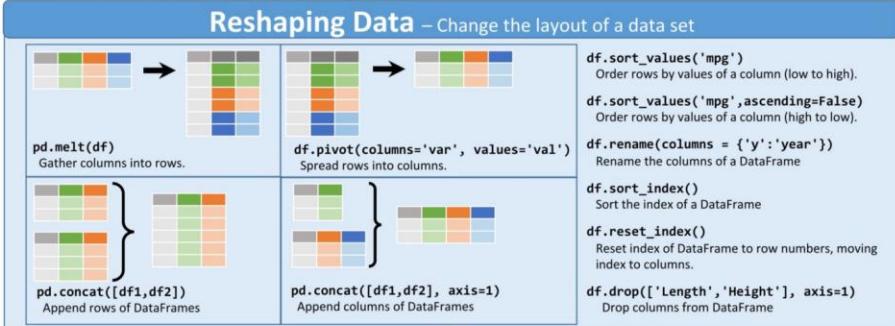
```
df = (pd.melt(df)
      .rename(columns={
          'variable' : 'var',
          'value' : 'val'})
      .query('val >= 200')
     )
```



Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



M \* A



df.sort\_values('mpg')  
Order rows by values of a column (low to high).

df.sort\_values('mpg', ascending=False)  
Order rows by values of a column (high to low).

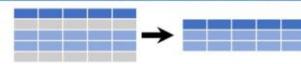
df.rename(columns = {'y':'year'})  
Rename the columns of a DataFrame

df.sort\_index()  
Sort the index of a DataFrame

df.reset\_index()  
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(['Length', 'Height'], axis=1)  
Drop columns from DataFrame

#### Subset Observations (Rows)



```
df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.
```

#### Subset Variables (Columns)



```
df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] or df.width
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression regex.
```

#### regex (Regular Expressions) Examples

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| '\.'                  | Matches strings containing a period.'                        |
| 'Length\$'            | Matches strings ending with word 'Length'                    |
| '^Sepal'              | Matches strings beginning with the word 'Sepal'              |
| '^x[1-5]\$'           | Matches strings beginning with 'x' and ending with 1,2,3,4,5 |
| '^(?!(Species\$)).*'} | Matches strings except the string 'Species'                  |

```
df.loc[:, 'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:, [1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns .
```

#### Logic in Python (and pandas)

|    |                        |                            |                                     |
|----|------------------------|----------------------------|-------------------------------------|
| <  | Less than              | !=                         | Not equal to                        |
| >  | Greater than           | df.column.isin(values)     | Group membership                    |
| == | Equals                 | pd.isnull(obj)             | Is NaN                              |
| <= | Less than or equals    | pd.notnull(obj)            | Is not NaN                          |
| >= | Greater than or equals | df[1] ~ df.any(), df.all() | Logical and, or, not, xor, any, all |

<http://pandas.pydata.org/> This cheat sheet inspired by RStudio Data Wrangling CheatSheet (<https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf>) Written by Irv Lustig, Princeton Consultants

## Summarize Data

```
df['w'].value_counts()
Count number of rows with each unique value of variable
len(df)
# of rows in DataFrame.
df['w'].nunique()
# of distinct values in a column.
df.describe()
Basic descriptive statistics for each column (or GroupBy)
```



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

```
sum()           Sum values of each object.
count()         Count non-NA/null values of each object.
median()        Median value of each object.
quantile([0.25,0.75]) Quantiles of each object.
apply(function) Apply function to each object.
```

```
min()           Minimum value in each object.
max()           Maximum value in each object.
mean()          Mean value of each object.
var()           Variance of each object.
std()           Standard deviation of each object.
```

## Group Data

```
df.groupby(by="col")
Return a GroupBy object, grouped by values in column named "col".
df.groupby(level="ind")
Return a GroupBy object, grouped by values in index level named "ind".
```

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

```
size()          Size of each group.
agg(function)  Aggregate group using function.
```

## Windows

```
df.expanding()
Return an Expanding object allowing summary functions to be applied cumulatively.
df.rolling(n)
Return a Rolling object allowing summary functions to be applied to windows of length n.
```

## Plotting

## Handling Missing Data

```
df.dropna()
Drop rows with any column having NA/null data.
df.fillna(value)
Replace all NA/null data with value.
```

## Make New Columns

df.assign(Area=lambda df: df.Length\*df.Height)  
 Compute and append one or more new columns.  
 df['Volume'] = df.Length\*df.Height\*df.Depth  
 Add single column.  
 pd.qcut(df.col, n, labels=False)  
 Bin column into n buckets.

pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

```
max(axis=1)      min(axis=1)
Element-wise max. Element-wise min.
clip(lower=-10,upper=10) abs()
Trim values at input thresholds Absolute value.
```

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

|                                                                                                                                                                                                                                                |                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>shift(1) Copy with values shifted by 1. rank(method='dense') Ranks with no gaps. rank(method='min') Ranks. Ties get min rank. rank(pct=True) Ranks rescaled to interval [0, 1]. rank(method='first') Ranks. Ties go to first value.</pre> | <pre>shift(-1) Copy with values lagged by 1. cumsum() Cumulative sum. cummax() Cumulative max. cummin() Cumulative min. cumprod() Cumulative product.</pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Plotting

df.plot.hist()
Histogram for each column  
 df.plot.scatter(x='w',y='h')
Scatter chart using pairs of points

## Combine Data Sets

| adf          | + | bdf     |
|--------------|---|---------|
| x1   x2   x3 |   | x1   x3 |
| A 1   T      |   | A   T   |
| B 2   F      |   | B   F   |
| C 3   NaN    |   | D   T   |

Standard Joins

|              |                      |
|--------------|----------------------|
| x1   x2   x3 | pd.merge(adf, bdf,   |
| A 1   T      | how='left', on='x1') |

|              |                       |
|--------------|-----------------------|
| x1   x2   x3 | pd.merge(adf, bdf,    |
| A 1.0   T    | how='right', on='x1') |

|              |                       |
|--------------|-----------------------|
| x1   x2   x3 | pd.merge(adf, bdf,    |
| A 1   T      | how='inner', on='x1') |

|              |                       |
|--------------|-----------------------|
| x1   x2   x3 | pd.merge(adf, bdf,    |
| A 1   T      | how='outer', on='x1') |

Filtering Joins

|           |                                           |
|-----------|-------------------------------------------|
| x1   x2   | adf[adf.x1.isin(bdf.x1)]                  |
| A 1   B 2 | All rows in adf that have a match in bdf. |

|         |                                                  |
|---------|--------------------------------------------------|
| x1   x2 | adf[~adf.x1.isin(bdf.x1)]                        |
| C 3     | All rows in adf that do not have a match in bdf. |

| ydf       | + | zdf             |
|-----------|---|-----------------|
| x1   x2   |   | x1   x2         |
| A 1   T   |   | B 2   C 3   D 4 |
| B 2   F   |   |                 |
| C 3   NaN |   |                 |

Set-like Operations

|         |                                                      |
|---------|------------------------------------------------------|
| x1   x2 | pd.merge(ydf, zdf)                                   |
| C 3     | Rows that appear in both ydf and zdf (Intersection). |

|           |                                                         |
|-----------|---------------------------------------------------------|
| x1   x2   | pd.merge(ydf, zdf, how='outer')                         |
| A 1   B 2 | Rows that appear in either or both ydf and zdf (Union). |

|           |                                                 |
|-----------|-------------------------------------------------|
| x1   x2   | pd.merge(ydf, zdf, how='outer', indicator=True) |
| A 1   B 2 | .query('_merge == "left_only")                  |
| C 3   NaN | .drop(['_merge'], axis=1)                       |

Rows that appear in ydf but not zdf (Setdiff).

# Data Wrangling with dplyr and tidyr

# Data Wrangling

## with dplyr and tidyr

Cheat Sheet



### Syntax - Helpful conventions for wrangling

`dplyr::tbl_df(iris)`

Converts data to `tbl` class. `tbl`'s are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4      0.2   setosa
2          4.9         3.0          1.4      0.2   setosa
3          4.7         3.2          1.3      0.2   setosa
4          4.6         3.1          1.5      0.2   setosa
5          5.0         3.6          1.4      0.2   setosa
..          ...         ...          ...      ...
Variables not shown: Petal.Width (dbl), Species (fctr)
```

`dplyr::glimpse(iris)`

Information dense summary of `tbl` data.

`utils::View(iris)`

View data set in spreadsheet-like display (note capital V).

| iris |              |             |              |             |         |
|------|--------------|-------------|--------------|-------------|---------|
|      | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 1    | 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 2    | 4.9          | 3.0         | 1.4          | 0.2         | setosa  |
| 3    | 4.7          | 3.2         | 1.3          | 0.2         | setosa  |
| 4    | 4.6          | 3.1         | 1.5          | 0.2         | setosa  |
| 5    | 5.0          | 3.6         | 1.4          | 0.2         | setosa  |
| 6    | 5.4          | 3.9         | 1.7          | 0.3         | setosa  |
| 7    | 4.6          | 3.4         | 1.4          | 0.3         | setosa  |
| 8    | 5.0          | 3.4         | 1.5          | 0.2         | setosa  |

`dplyr::%>%`

Passes object on left hand side as first argument (or . argument) of function on righthand side.

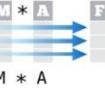
```
x %>% f(y) is the same as f(x, y)
y %>% f(x, .., z) is the same as f(x, y, z)
```

"Piping" with `%>%` makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

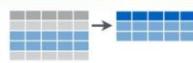
### Tidy Data - A foundation for wrangling in R



### Reshaping Data - Change the layout of a data set

|                                                             |                                                    |                                                  |
|-------------------------------------------------------------|----------------------------------------------------|--------------------------------------------------|
| <code>dplyr::gather(cases, "year", "n", 2:4)</code>         | <code>tidy::spread(pollution, size, amount)</code> | <code>dplyr::data_frame(a = 1:3, b = 4:6)</code> |
| Gather columns into rows.                                   | Spread rows into columns.                          | Combine vectors into data frame (optimized).     |
| <code>tidy::separate(storms, date, c("y", "m", "d"))</code> | <code>tidy::unite(data, col, ..., sep)</code>      | <code>dplyr::arrange(mtcars, mpg)</code>         |
| Separate one column into several.                           | Unite several columns into one.                    | Order rows by values of a column (low to high).  |
|                                                             |                                                    | <code>dplyr::arrange(mtcars, desc(mpg))</code>   |
|                                                             |                                                    | Order rows by values of a column (high to low).  |
|                                                             |                                                    | <code>dplyr::rename(tb, y = year)</code>         |
|                                                             |                                                    | Rename the columns of a data frame.              |

### Subset Observations (Rows)



`dplyr::filter(iris, Sepal.Length > 7)`

Extract rows that meet logical criteria.

`dplyr::distinct(iris)`

Remove duplicate rows.

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`

Randomly select fraction of rows.

`dplyr::sample_n(iris, 10, replace = TRUE)`

Randomly select n rows.

`dplyr::slice(iris, 10:15)`

Select rows by position.

`dplyr::top_n(storms, 2, date)`

Select and order top n entries (by group if grouped data).

### Logic in R - ?Comparison, ?base::Logic

|    |                          |        |                   |
|----|--------------------------|--------|-------------------|
| <  | Less than                | !=     | Not equal to      |
| >  | Greater than             | %in%   | Group membership  |
| == | Equal to                 | is.na  | Is NA             |
| <= | Less than or equal to    | !is.na | Is not NA         |
| >= | Greater than or equal to | isTRUE | Boolean operators |

`dplyr::select(iris, Sepal.Width, Petal.Length, Species)`

Select columns by name or helper function.

### Helper functions for select - ?select

`select(iris, contains("."))`

Select columns whose name contains a character string.

`select(iris, ends_with("Length"))`

Select columns whose name ends with a character string.

`select(iris, everything())`

Select every column.

`select(iris, matches("t"))`

Select columns whose name matches a regular expression.

`select(iris, num_range("x", 1:5))`

Select columns named x1, x2, x3, x4, x5.

`select(iris, one_of(c("Species", "Genus")))`

Select columns whose names are in a group of names.

`select(iris, starts_with("Sepal"))`

Select columns whose name starts with a character string.

`select(iris, Sepal.Length:Petal.Width)`

Select all columns between Sepal.Length and Petal.Width (inclusive).

`select(iris, -Species)`

Select all columns except Species.

Learn more with `browseVignettes(package = c("dplyr", "tidyverse"))` • dplyr 0.4.0+ tidyverse 0.2.0+ Updated: 1/15

### Summarise Data

```
dplyr::summarise(iris, avg = mean(Sepal.Length))
Summarise data into single row of values.
```

`dplyr::summarise_each(iris, funs(mean))`  
Apply summary function to each column.

```
dplyr::count(iris, Species, wt = Sepal.Length)
Count number of rows with each unique value of variable (with or without weights).
```

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

|                                |                                   |
|--------------------------------|-----------------------------------|
| <code>dplyr::first</code>      | First value of a vector.          |
| <code>dplyr::last</code>       | Last value of a vector.           |
| <code>dplyr::nth</code>        | Nth value of a vector.            |
| <code>dplyr::n</code>          | # of values in a vector.          |
| <code>dplyr::n_distinct</code> | # of distinct values in a vector. |
| <code>IQR</code>               | IQR of a vector.                  |
| <code>min</code>               | Minimum value in a vector.        |
| <code>max</code>               | Maximum value in a vector.        |
| <code>mean</code>              | Mean value of a vector.           |
| <code>median</code>            | Median value of a vector.         |
| <code>var</code>               | Variance of a vector.             |
| <code>sd</code>                | Standard deviation of a vector.   |

### Make New Variables

```
dplyr::mutate(iris, sepal = Sepal.Length + Sepal.Width)
Compute and append one or more new columns.
```

```
dplyr::mutate_each(iris, funs(min_rank))
Apply window function to each column.
```

```
dplyr::transmute(iris, sepal = Sepal.Length + Sepal.Width)
Compute one or more new columns. Drop original columns.
```

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

|                                  |                                 |
|----------------------------------|---------------------------------|
| <code>dplyr::lead</code>         | Copy with values shifted by 1.  |
| <code>dplyr::lag</code>          | Copy with values lagged by 1.   |
| <code>dplyr::dense_rank</code>   | Ranks with no gaps.             |
| <code>dplyr::min_rank</code>     | Ranks. Ties get min rank.       |
| <code>dplyr::percent_rank</code> | Ranks rescaled to [0, 1].       |
| <code>dplyr::row_number</code>   | Ranks. Ties got to first value. |
| <code>dplyr::ntile</code>        | Bin vector into n buckets.      |
| <code>dplyr::between</code>      | Are values between a and b?     |
| <code>dplyr::cume_dist</code>    | Cumulative distribution.        |
| <code>dplyr::cumall</code>       | Cumulative all                  |
| <code>dplyr::cumany</code>       | Cumulative any                  |
| <code>dplyr::cummean</code>      | Cumulative mean                 |
| <code>cumsum</code>              | Cumulative sum                  |
| <code>cummax</code>              | Cumulative max                  |
| <code>cummin</code>              | Cumulative min                  |
| <code>cumprod</code>             | Cumulative prod                 |
| <code>pmax</code>                | Element-wise max                |
| <code>pmin</code>                | Element-wise min                |

### Combine Data Sets

|                                                                                                                                                                                                                                                                                                           |                  |                |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|----------------|----|---|---|---|---|---|---|---|---|---|--|--|---|--|--|---|--|--|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|---|---|---|---|---|---|---|---|----|---|----|---|--|--|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|---|---|---|---|---|---|---|---|----|---|----|---|--|--|----|
| <code>a</code>                                                                                                                                                                                                                                                                                            | <code>+ b</code> | <code>=</code> |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| <table border="1"><tr><td>x1</td><td>x2</td><td>x3</td></tr><tr><td>A</td><td>1</td><td>A</td></tr><tr><td>B</td><td>2</td><td>B</td></tr><tr><td>C</td><td>3</td><td>F</td></tr><tr><td></td><td></td><td>T</td></tr><tr><td></td><td></td><td>D</td></tr><tr><td></td><td></td><td>NA</td></tr></table> | x1               | x2             | x3 | A | 1 | A | B | 2 | B | C | 3 | F |  |  | T |  |  | D |  |  | NA | <table border="1"><tr><td>x1</td><td>x2</td><td>x3</td></tr><tr><td>A</td><td>1</td><td>T</td></tr><tr><td>B</td><td>2</td><td>F</td></tr><tr><td>C</td><td>3</td><td>NA</td></tr><tr><td>D</td><td>NA</td><td>T</td></tr><tr><td></td><td></td><td>NA</td></tr></table> | x1 | x2 | x3 | A | 1 | T | B | 2 | F | C | 3 | NA | D | NA | T |  |  | NA | <table border="1"><tr><td>x1</td><td>x2</td><td>x3</td></tr><tr><td>A</td><td>1</td><td>T</td></tr><tr><td>B</td><td>2</td><td>F</td></tr><tr><td>C</td><td>3</td><td>NA</td></tr><tr><td>D</td><td>NA</td><td>T</td></tr><tr><td></td><td></td><td>NA</td></tr></table> | x1 | x2 | x3 | A | 1 | T | B | 2 | F | C | 3 | NA | D | NA | T |  |  | NA |
| x1                                                                                                                                                                                                                                                                                                        | x2               | x3             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| A                                                                                                                                                                                                                                                                                                         | 1                | A              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| B                                                                                                                                                                                                                                                                                                         | 2                | B              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| C                                                                                                                                                                                                                                                                                                         | 3                | F              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|                                                                                                                                                                                                                                                                                                           |                  | T              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|                                                                                                                                                                                                                                                                                                           |                  | D              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|                                                                                                                                                                                                                                                                                                           |                  | NA             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| x1                                                                                                                                                                                                                                                                                                        | x2               | x3             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| A                                                                                                                                                                                                                                                                                                         | 1                | T              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| B                                                                                                                                                                                                                                                                                                         | 2                | F              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| C                                                                                                                                                                                                                                                                                                         | 3                | NA             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| D                                                                                                                                                                                                                                                                                                         | NA               | T              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|                                                                                                                                                                                                                                                                                                           |                  | NA             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| x1                                                                                                                                                                                                                                                                                                        | x2               | x3             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| A                                                                                                                                                                                                                                                                                                         | 1                | T              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| B                                                                                                                                                                                                                                                                                                         | 2                | F              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| C                                                                                                                                                                                                                                                                                                         | 3                | NA             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
| D                                                                                                                                                                                                                                                                                                         | NA               | T              |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |
|                                                                                                                                                                                                                                                                                                           |                  | NA             |    |   |   |   |   |   |   |   |   |   |  |  |   |  |  |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |                                                                                                                                                                                                                                                                          |    |    |    |   |   |   |   |   |   |   |   |    |   |    |   |  |  |    |

**Mutating Joins**

|                                                 |                                           |
|-------------------------------------------------|-------------------------------------------|
| <code>dplyr::left_join(a, b, by = "x1")</code>  | Join matching rows from b to a.           |
| <code>dplyr::right_join(a, b, by = "x1")</code> | Join matching rows from a to b.           |
| <code>dplyr::inner_join(a, b, by = "x1")</code> | Join data. Retain only rows in both sets. |
| <code>dplyr::full_join(a, b, by = "x1")</code>  | Join data. Retain all values, all rows.   |

**Filtering Joins**

|                                                |                                              |
|------------------------------------------------|----------------------------------------------|
| <code>dplyr::semi_join(a, b, by = "x1")</code> | All rows in a that have a match in b.        |
| <code>dplyr::anti_join(a, b, by = "x1")</code> | All rows in a that do not have a match in b. |

**Set Operations**

|                                     |                                             |
|-------------------------------------|---------------------------------------------|
| <code>dplyr::intersect(y, z)</code> | Rows that appear in both y and z.           |
| <code>dplyr::union(y, z)</code>     | Rows that appear in either or both y and z. |
| <code>dplyr::setdiff(y, z)</code>   | Rows that appear in y but not z.            |

**Binding**

|                                     |                               |
|-------------------------------------|-------------------------------|
| <code>dplyr::bind_rows(y, z)</code> | Append z to y as new rows.    |
| <code>dplyr::bind_cols(y, z)</code> | Append z to y as new columns. |

Caution: matches rows by position.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com    devtools::install\_github("rstudio/EDAWR") for data sets    Learn more with browseVignettes(package=c("dplyr", "tidyverse")) • dplyr 0.4.0 • tidyverse 0.2.0 • Updated: 1/15

## Scipy

SciPy builds on the [NumPy](#) array object and is part of the NumPy stack which includes tools like [Matplotlib](#), [pandas](#) and [SymPy](#), and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as [MATLAB](#), [GNU Octave](#), and [Scilab](#). The NumPy stack is also sometimes referred to as the SciPy stack.[\[3\]](#)

## Python For Data Science Cheat Sheet

### SciPy – Linear Algebra

Learn More Python for Data Science [Interactively](#) at [www.datacamp.com](#)

#### SciPy

The SciPy library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.



#### Interacting With NumPy

[Also see NumPy](#)

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

#### Index Tricks

```
>>> np.mgrid[0:5,0:5] Create a dense meshgrid
>>> np.ogrid[0:2,0:2] Create an open meshgrid
>>> np.r_[3,[0]*5,-1:1:10j] Stack arrays vertically (row-wise)
>>> np.c_[b,c] Create stacked column-wise arrays
```

#### Shape Manipulation

```
>>> np.transpose(b) Permute array dimensions
>>> b.flatten() Flatten the array
>>> np.hstack((b,c)) Stack arrays horizontally (column-wise)
>>> np.vstack((a,b)) Stack arrays vertically (row-wise)
>>> np.hsplit(c,2) Split the array horizontally at the 2nd index
>>> np.vsplit(d,2) Split the array vertically at the 2nd index
```

#### Polynomials

```
>>> from numpy import polyd Create a polynomial object
>>> p = polyd([3,4,5])
```

#### Vectorizing Functions

```
>>> def myfunc(a):
...     if a < 0:
...         return a**2
...     else:
...         return a/2
>>> np.vectorize(myfunc)
```

Vectorize functions

#### Type Handling

```
>>> np.real(b) Return the real part of the array elements
>>> np.imag(b) Return the imaginary part of the array elements
>>> np.real_if_close(c,tol=1000) Return a real array if complex parts close to 0
>>> np.cast['f'](np.pi) Cast object to a data type
```

#### Other Useful Functions

```
>>> np.angle(b,deg=True) Create the angle of the complex argument
>>> g = np.linspace(0,np.pi,num=5) Create an array of evenly spaced values (number of samples)
>>> q [3:] *= np.pi Unwrap
>>> np.unwrap(g) Create an array of evenly spaced values (log scale)
>>> np.logspace(0,10,3) Return values from a list of arrays depending on conditions
>>> np.select([c>4], [c>2]) Combine N things taken at k time
>>> misc.factorial(a) Weights for N-point central derivative
>>> misc.comb(10,3,exact=True) Find the n-th derivative of a function at a point
>>> misc.central_diff_weights(3)
>>> misc.derivative(myfunc,1,0)
```

## Linear Algebra

You'll use the `linalg` and `sparse` modules. Note that `scipy.linalg` contains and expands on `numpy.linalg`.

[Also see NumPy](#)

### Matrix Functions

|                                            |                                    |
|--------------------------------------------|------------------------------------|
| <b>Addition</b><br>>>> np.add(A,D)         | Addition                           |
| <b>Subtraction</b><br>>>> np.subtract(A,D) | Subtraction                        |
| <b>Division</b><br>>>> np.divide(A,D)      | Division                           |
| <b>Multiplication</b><br>>>> A @ D         | Multiplication operator (python 3) |

|                                                    |                    |
|----------------------------------------------------|--------------------|
| <b>Dot product</b><br>>>> np.dot(A,D)              | Dot product        |
| <b>Vector dot product</b><br>>>> np.vdot(A,D)      | Vector dot product |
| <b>Inner product</b><br>>>> np.inner(A,D)          | Inner product      |
| <b>Outer product</b><br>>>> np.outer(A,D)          | Outer product      |
| <b>Tensor dot product</b><br>>>> np.tensordot(A,D) | Tensor dot product |
| <b>Kronecker product</b><br>>>> np.kron(K,D)       | Kronecker product  |

|                                                                   |                                    |
|-------------------------------------------------------------------|------------------------------------|
| <b>Matrix exponential</b><br>>>> linalg.expm(A)                   | Matrix exponential                 |
| <b>Matrix exponential (Taylor Series)</b><br>>>> linalg.expmt2(A) | Matrix exponential (decomposition) |
| <b>Matrix logarithm</b><br>>>> linalg.logm(A)                     | Matrix logarithm                   |

|                                             |                |
|---------------------------------------------|----------------|
| <b>Matrix sine</b><br>>>> linalg.sinm(D)    | Matrix sine    |
| <b>Matrix cosine</b><br>>>> linalg.cosm(D)  | Matrix cosine  |
| <b>Matrix tangent</b><br>>>> linalg.tanm(A) | Matrix tangent |

|                                                                 |                           |
|-----------------------------------------------------------------|---------------------------|
| <b>Hyperbolic Trigonometric Functions</b><br>>>> linalg.sinh(D) | Hyperbolic matrix sine    |
| <b>Hyperbolic matrix cosine</b><br>>>> linalg.coshm(D)          | Hyperbolic matrix cosine  |
| <b>Hyperbolic matrix tangent</b><br>>>> linalg.tanhm(A)         | Hyperbolic matrix tangent |

|                                                |                      |
|------------------------------------------------|----------------------|
| <b>Matrix sign function</b><br>>>> np.signm(A) | Matrix sign function |
|------------------------------------------------|----------------------|

|                                                  |                    |
|--------------------------------------------------|--------------------|
| <b>Matrix square root</b><br>>>> linalg.sqrtm(A) | Matrix square root |
|--------------------------------------------------|--------------------|

|                                                                 |                          |
|-----------------------------------------------------------------|--------------------------|
| <b>Arbitrary Functions</b><br>>>> linalg.funm(A, lambda x: x*x) | Evaluate matrix function |
|-----------------------------------------------------------------|--------------------------|

### Decompositions

|                                                                  |                                                                    |
|------------------------------------------------------------------|--------------------------------------------------------------------|
| <b>Eigenvalues and Eigenvectors</b><br>>>> la, v = linalg.eig(A) | Solve ordinary or generalized eigenvalue problem for square matrix |
| <b>Unpack eigenvalues</b><br>>>> l1, l2 = la                     | Unpack eigenvalues                                                 |
| <b>First eigenvector</b><br>>>> v[:,0]                           | First eigenvector                                                  |
| <b>Second eigenvector</b><br>>>> v[:,1]                          | Second eigenvector                                                 |
| <b>Unpack eigenvalues</b><br>>>> linalg.eigvals(A)               | Unpack eigenvalues                                                 |

|                                                                   |                                    |
|-------------------------------------------------------------------|------------------------------------|
| <b>Singular Value Decomposition</b><br>>>> U,s,Vh = linalg.svd(B) | Singular Value Decomposition (SVD) |
| <b>Construct sigma matrix in SVD</b><br>>>> M,N = B.shape         |                                    |
| <b>Sigma</b><br>>>> Sig = linalg.diagsvd(s,M,N)                   |                                    |

|                                                     |                  |
|-----------------------------------------------------|------------------|
| <b>LU Decomposition</b><br>>>> P,L,U = linalg.lu(C) | LU Decomposition |
|-----------------------------------------------------|------------------|

|                                                                            |                              |
|----------------------------------------------------------------------------|------------------------------|
| <b>Sparse Matrix Decompositions</b><br>>>> la, v = sparse.linalg.eigs(F,1) | Eigenvalues and eigenvectors |
| <b>SVD</b><br>>>> sparse.linalg.svds(H, 2)                                 | SVD                          |

DataCamp  
Learn Python for Data Science [Interactively](#)



## Matplotlib

**matplotlib** is a [plotting library](#) for the [Python](#) programming language and its numerical mathematics extension [NumPy](#). It provides an [object-oriented API](#) for embedding plots into applications using general-purpose [GUI toolkits](#) like [Tkinter](#), [wxPython](#), [Qt](#), or [GTK+](#). There is also a [procedural](#) “pylab” interface based on a [state machine](#) (like [OpenGL](#)), designed to closely resemble that of [MATLAB](#), though its use is discouraged.[\[2\]](#) [SciPy](#) makes use of [matplotlib](#).

[pyplot](#) is a [matplotlib](#) module which provides a [MATLAB-like](#) interface.[\[6\]](#) [matplotlib](#) is designed to be as usable as [MATLAB](#), with the ability to use [Python](#), with the advantage that it is free.

## Python For Data Science Cheat Sheet

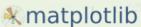
### Matplotlib

Learn Python Interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



### 1) Prepare The Data

Also see [Lists & NumPy](#)

#### 1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

#### 2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> X, Y = np.mgrid[-3:3:1.00j, -3:3:1.00j]
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data("axes_grid/bivariate_normal.npy"))
```

### 2) Create Plot

>>> import matplotlib.pyplot as plt

#### Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

#### Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_subplot()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

### 3) Plotting Routines

#### 1D Data

```
>>> fig, ax = plt.subplots()
>>> ax.plot(x,y)
>>> ax.scatter(x,y)
Draw points with lines or markers connecting them
Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].bar([1,2,3],[5,1,2])
Plot vertical rectangles (constant width)
Plot horizontal rectangles (constant height)
>>> axes[0,1].axhline(45)
Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65)
Draw filled polygons
>>> ax.fill(x,y,color='blue')
Fill between(x,y,color='yellow')
Fill between y-values and o
```

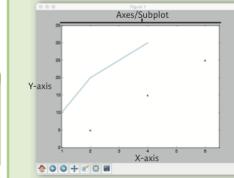
#### 2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
                  interpolation='nearest',
                  vmin=-2,
                  vmax=2)
```

Colormapped or RGB arrays

### Plot Anatomy & Workflow

#### Plot Anatomy



#### Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] <Step 1>
>>> y = [10,20,25,30] <Step 2>
>>> fig = plt.figure() <Step 3>
>>> ax = fig.add_subplot(111) <Step 3>
>>> ax.plot(x, y, color='lightblue', linewidth=3) <Step 4>
>>> ax.scatter([2,4,6], [5,15,25], color='darkgreen', marker='*')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png') <Step 5>
>>> plt.show() <Step 6>
```

### 4) Customize Plot

#### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x*x, x, x*x)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, alpha = 0.4)
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img, cmap='seismic')
```

#### Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker="*")
>>> ax.plot(x,y,marker="o")
```

#### Linestyles

```
>>> plt.plot(x,y, linewidth=4.0)
>>> plt.plot(x,y, linestyle='solid')
>>> plt.plot(x,y, '-.-', x*x, x*x, '--')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

#### Text & Annotations

```
>>> ax.text(2,1,
           'Example Graph',
           style='italic')
>>> ax.annotate(xy=(8, 0),
               xycoords='data',
               xytext=(0, 0),
               textcoords='data',
               arrowprops=dict(arrowstyle=">",
                               connectionstyle="arc3",))
```

#### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
>>> axes[0,1].streamplot(X,Y,U,V)
```

#### Data Distributions

```
>>> ax1.hist(y)
>>> ax3.boxplot(y)
>>> ax3.violinplot(z)
```

Pseudocolor plot of 2D array  
Pseudocolor plot of 2D array  
Plot contours  
Plot filled contours  
Label a contour plot

#### Mathtext

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

#### Limits, Legends & Layouts

Limits & Autoscaling  
Add padding to a plot  
Set the aspect ratio of the plot to 1

Set limits for x and y axes  
Set limits for x-axis

Set a title and x-and y-axis labels  
No overlapping plot elements

No overlapping plot elements  
Manually set x-ticks

Make y-ticks longer and go in and out  
Adjust the spacing between subplots

Subplot Spacing  
Fit subplot(s) in to the figure area

Axis Spines  
Make the top axis line for a plot invisible  
Move the bottom axis line outward

### 5) Save Plot

Save figures  
Save transparent figures  
Save transparent figures

```
>>> plt.savefig('foo.png')
>>> plt.savefig('foo.png', transparent=True)
```

Save figures  
Save transparent figures  
Save transparent figures

### 6) Show Plot

```
>>> plt.show()
```

### Close & Clear

Clear axis  
Clear the entire figure  
Close a window

Clear axis  
Clear the entire figure  
Close a window

## Data Visualization

DataCamp

Learn Python for Data Science Interactively



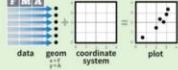
# Data Visualization with ggplot2

## Cheat Sheet

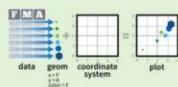


### Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data set**, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



#### Build a graph with `qplot()` or `ggplot()`

`qplot(x=cty, y=hwy, color=cyl, data=mpg, geom="point")`  
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

#### `ggplot(data = mpg, aes(x = cty, y = hwy))`

Begins a plot that you finish by adding layers to. No defaults, but provides more control than `qplot()`.

`data`  
`ggplot(mpg, aes(hwy, cty)) +  
geom_point(aes(color = cyl)) +  
geom_smooth(aes(method = "lm")) +  
coord_cartesian() +  
scale_color_gradient() +  
theme_bw()`

Add a new layer to a plot with a `geom.*()` or `stat_*` function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

#### `last_plot()`

Return the last plot.

`ggsave("plot.png", width = 5, height = 5)`

Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

**Geoms** - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

#### One Variable

|                                                                                                           |  |
|-----------------------------------------------------------------------------------------------------------|--|
| a + <code>geom_area(stat = "bin")</code>                                                                  |  |
| b + <code>geom_area(aes(= ..density..), stat = "bin")</code>                                              |  |
| a + <code>geom_density(kernel = "gaussian")</code>                                                        |  |
| x, y, alpha, color, fill, linetype, size<br>b + <code>geom_density(aes(y = ..count..))</code>             |  |
| a + <code>geom_dotplot()</code>                                                                           |  |
| x, y, alpha, color, fill                                                                                  |  |
| a + <code>geom_freqpoly()</code>                                                                          |  |
| x, y, alpha, color, linetype, size<br>b + <code>geom_freqpoly(aes(y = ..density..))</code>                |  |
| a + <code>geom_histogram(binwidth = 5)</code>                                                             |  |
| x, y, alpha, color, fill, linetype, size, weight<br>b + <code>geom_histogram(aes(y = ..density..))</code> |  |
| a + <code>geom_bar()</code>                                                                               |  |
| x, alpha, color, fill, linetype, size, weight                                                             |  |

#### Graphical Primitives

|                                                                                                                                                                      |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| c + <code>geom_bar(stat = "identity")</code>                                                                                                                         |  |
| c + <code>geom_polygon(aes(group = group))</code>                                                                                                                    |  |
| x, y, alpha, color, fill, linetype, size                                                                                                                             |  |
| d + <code>ggplot(economics, aes(date, unemploy))</code>                                                                                                              |  |
| d + <code>geom_path(linend="butt",<br/>linejoin="round", linemtire=1)</code>                                                                                         |  |
| x, y, alpha, color, linetype, size<br>d + <code>geom_ribbon(aes(ymin=unemploy - 900,<br/>ymax=unemploy + 900))</code>                                                |  |
| x, ymax, ymin, alpha, color, fill, linetype, size                                                                                                                    |  |
| e + <code>ggplot(seals, aes(x = long, y = lat))</code>                                                                                                               |  |
| e + <code>geom_segment(aes(<br/>xend = long + delta_long,<br/>yend = lat + delta_lat))</code>                                                                        |  |
| x, xend, y, yend, alpha, color, linetype, size<br>e + <code>geom_rect(aes(xmin = long, ymin = lat,<br/>xmax = long + delta_long,<br/>ymax = lat + delta_lat))</code> |  |
| xmax, xmin, ymax, ymin, alpha, color, fill,<br>linetype, size                                                                                                        |  |

#### Two Variables

|                                                                                       |  |
|---------------------------------------------------------------------------------------|--|
| f + <code>geom_blank()</code>                                                         |  |
| f + <code>geom_jitter()</code>                                                        |  |
| x, y, alpha, color, fill, shape, size                                                 |  |
| f + <code>geom_point()</code>                                                         |  |
| x, y, alpha, color, fill, shape, size                                                 |  |
| f + <code>geom_quantile()</code>                                                      |  |
| x, y, alpha, color, linetype, size, weight                                            |  |
| f + <code>geom_rug(sides = "bl")</code>                                               |  |
| alpha, color, linetype, size                                                          |  |
| f + <code>geom_smooth(model = lm)</code>                                              |  |
| x, y, alpha, color, fill, linetype, size, weight                                      |  |
| C + <code>geom_text(aes(label = cty))</code>                                          |  |
| x, y, label, alpha, angle, color, family, fontface,<br>hjust, lineheight, size, vjust |  |

#### Three Variables

|                                                                                                            |  |
|------------------------------------------------------------------------------------------------------------|--|
| i + <code>geom_bin2d(binwidth = c(5, 0.5))</code>                                                          |  |
| xmax, xmin, ymax, ymin, alpha, color, fill, linetype,<br>size, weight                                      |  |
| i + <code>geom_hex()</code>                                                                                |  |
| x, y, alpha, colour, fill, size                                                                            |  |
| j + <code>geom_area()</code>                                                                               |  |
| x, y, alpha, color, fill, linetype, size                                                                   |  |
| j + <code>geom_line()</code>                                                                               |  |
| x, y, alpha, color, linetype, size                                                                         |  |
| j + <code>geom_step(direction = "hv")</code>                                                               |  |
| x, y, alpha, color, linetype, size                                                                         |  |
| <b>Visualizing error</b>                                                                                   |  |
| df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)                                                   |  |
| k + <code>ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))</code>                               |  |
| k + <code>geom_crossbar(fatten = 2)</code>                                                                 |  |
| x, y, ymax, ymin, alpha, color, fill, linetype,<br>size                                                    |  |
| k + <code>geom_errorbar()</code>                                                                           |  |
| x, ymax, ymin, alpha, color, linetype, size,<br>width (also <code>geom_errorbarh()</code> )                |  |
| k + <code>geom_linerange()</code>                                                                          |  |
| x, ymin, ymax, alpha, color, linetype, size                                                                |  |
| k + <code>geom_pointrange()</code>                                                                         |  |
| x, y, ymin, ymax, alpha, color, fill, linetype,<br>shape, size                                             |  |
| <b>Maps</b>                                                                                                |  |
| data <- data.frame(murder = USArrests\$Murder,<br>state = USArrests\$State, region = USArrests\$Region)    |  |
| map <- map_data("state")                                                                                   |  |
| l + <code>ggplot(data, aes(murder))</code>                                                                 |  |
| l + <code>geom_map(aes(map_id = state), map = map) +<br/>expand_limits(x = map\$long, y = map\$lat)</code> |  |
| map_id, alpha, color, fill, linetype, size                                                                 |  |

#### Three Variables

|                                                                                           |  |
|-------------------------------------------------------------------------------------------|--|
| m + <code>geom_raster(aes(fill = z), hijost=0.5,<br/>vjust=0.5, interpolate=FALSE)</code> |  |
| x, y, alpha, fill, linetype, size                                                         |  |

Learn more at [docs.ggplot2.org](http://docs.ggplot2.org) • ggplot2 0.9.3.1 • Updated: 3/15

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • [info.rstudio.com](http://info.rstudio.com) • 844-448-1212 • [rstudio.com](mailto:rstudio.com)

**Stats - An alternative way to build a layer**

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. `a + geom_bar(stat = "bin")`

Each stat creates additional variables to map aesthetics to. These variables use a common `.name..` syntax. stat functions and geom functions both combine a stat with a geom to make a layer, i.e. `i + stat_bin(geom="bar")` does the same as `geom_bar(stat="bin")`

```

i + stat_bin(bins=1, origin = 10)
  x, y | .count..., .ncount..., .density...
  i + stat_bindot(bins=1, binaxis = "x")
  x, y | .count..., .ncount...
  i + stat_density(adjust = 1, kernel = "gaussian")
  x, y, color, size | .level...
  
```

**1D distributions**

```

i + stat_bin2d(bins = 30, drop = TRUE)
  x, y | .count..., .ncount..., .density...
  i + stat_bineq(bins = 30)
  x, y | .count..., .density...
  i + stat_density2d(contour = TRUE, n = 100)
  x, y, color, size | .level...
  
```

**2D distributions**

```

i + stat_contour(besid = z)
  x, y, z, order | .level...
  i + stat_speke(pesradius = z, angle = z)
  angle, radius, x, vend, y, vend | .xend..., .yend...
  i + stat_summary_hex(aes(z = z), bins = 30, fun = mean)
  x, y, z, fill | .value...
  i + stat_boxplot(coef = 1.5)
  Comparisons
  x, y | lower..., middle..., upper..., outliers...
  i + stat_density2d(adjust = 1, kernel = "gaussian", scale = "area")
  x, y | .density..., .scaled..., .count..., .n...
  i + stat_ecdf(n = 40)
  Functions
  x, y | .x...
  i + stat_quantile(quintiles = c(0.25, 0.5, 0.75), formula = y ~ log(x),
  method = "rq")
  x, y | .quintile..., .x...
  i + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80,
  fullrange = FALSE, level = 0.95)
  x, y | .se..., .x..., .ymin..., .ymax...
  i + ggplot() + stat_function(fun = f, n = 33,
  fun = dnorm, n = 101, args = list(sd = 0.5))
  General Purpose
  x | .y...
  
```

**Scales**

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

```

n ~ b + geom_bar(aesfill = fill)
  n + scale_fill_manual(
    values = c("royalblue", "blue", "navy"),
    limits = c("d", "e", "p", "r"),
    name = "fill",
    labels = c("D", "E", "P", "R"))
  range of values to include in mapping
  title to use in legend/aes
  labels to use in legend/aes
  breaks to use in legend/aes
  
  
```

**General Purpose scales**

Use with any aesthetic:

- alpha, color, fill, linetype, shape, size
- scale\_.continuous() - map cont values to visual values
- scale\_.discrete() - map discrete values to visual values
- scale\_.identity() - use data values as visual values
- scale\_.manual(values = c(), labels = c(), name = "fill", breaks = c("d", "e", "p", "r")) - map discrete values to manually chosen visual values

**X and Y location scales**

Use with x or y aesthetics (shown here)

```

scale_x_labels() - map labels to date format ("ymd/%d"),
  breaks = date_breaks("2 weeks"), treat x values as dates. See ?strptime for label formats.
  scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
  scale_x_log10() - Plot x on log10 scale
  scale_x_reverse() - Reverse direction of x axis
  scale_x_sqrt() - Plot x on square root scale
  
  
```

**Color and fill scales**

Discrete

Continuous

**Shape scales**

**Size scales**

**Coordinate Systems**

`r <- b + geom_bar()`

```

r + coord_cartesian(xlim = c(0, 5))
  xlim, ylim
  The default cartesian coordinate system
  r + coord_fixed(ratio = 1/2)
  ratio, xlim, ylim
  Cartesian coordinates with fixed aspect ratio between x and y units
  r + coord_flip()
  xlim, ylim
  Flipped Cartesian coordinates
  r + coord_polar(theta = "x", direction = 1)
  theta, start, direction
  Polar coordinates
  r + coord_trans(xform = "sqrt")
  xtrans, ytrans, limx, limy
  Transformed cartesian coordinates. Set extra and strains to the name of a function.
  
  
```

**Faceting**

Facets divide a plot into subplots based on the values of one or more discrete variables.

```

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
  
  
```

**Coordinate Systems**

`r <- b + geom_bar()`

```

r + coord_cartesian(xlim = c(0, 5))
  xlim, ylim
  The default cartesian coordinate system
  r + coord_fixed(ratio = 1/2)
  ratio, xlim, ylim
  Cartesian coordinates with fixed aspect ratio between x and y units
  r + coord_flip()
  xlim, ylim
  Flipped Cartesian coordinates
  r + coord_polar(theta = "x", direction = 1)
  theta, start, direction
  Polar coordinates
  r + coord_trans(xform = "sqrt")
  xtrans, ytrans, limx, limy
  Transformed cartesian coordinates. Set extra and strains to the name of a function.
  
  
```

**Faceting**

Facets divide a plot into subplots based on the values of one or more discrete variables.

```

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
  
  
```

**Scales**

Set scales to let axis limits vary across facets

```

t + facet_grid(~ x, scales = "free")
  x and y axis limits adjust to individual facets
  • "free_x" - x axis limits adjust
  • "free_y" - y axis limits adjust
  
  
```

**Set labeler** to adjust facet labels

```

t + facet_grid(~ fl, labeler = label_both)
  fl: c fl: d fl: e fl: p fl: r
  t + facet_grid(~ fl, labeler = label_bquote(alpha ^ (x)))
  alpha^c alpha^d alpha^e alpha^p alpha^r
  t + facet_grid(~ fl, labeler = label_parsed)
  c d e p r
  
  
```

**Position Adjustments**

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```

s <- ggplot(mpg, aes(fl, fill = drv))
  
  
```

**Labels**

**Legends**

**Zooming**

Learn more at [docs.ggplot2.org](http://docs.ggplot2.org) • ggplot2 0.9.3.1 • Updated: 3/15

# PySpark

## Python For Data Science Cheat Sheet

### PySpark Basics

Learn Python for data science interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Spark

PySpark is the Spark Python API that exposes the Spark programming model to Python



#### Initializing Spark

##### SparkContext

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

##### Inspect SparkContext

```
>>> sc.version          Retrieve SparkContext version
>>> sc.pythonVer        Retrieve Python version
>>> sc.master           Master URL to connect to
>>> sc._jsparkHome       Path where Java is installed on worker nodes
>>> sc._jsparkUser()    Retrieve name of the Spark User running
>>> sc.appName           Return application name
>>> sc._jlocationId     Returns the location ID
>>> sc.defaultParallelism  Return default level of parallelism
>>> sc.defaultMinPartitions  Default minimum number of partitions for
RDDs
```

##### Configuration

```
>>> from pyspark import SparkConf, SparkContext
>>> conf = (SparkConf()
...         .setAppName("My app")
...         .set("spark.executor.memory", "1g"))
>>> sc = SparkContext(conf = conf)
```

##### Using The Shell

In the PySpark shell, a special interpreter-aware SparkContext is already created in the variable called `sc`.

```
$ ./bin/spark-shell --master local[2]
$ ./bin/pyspark --master local[4] --pyfiles code.py
```

Set which master the context connects to with the `--master` argument, and add Python zip, egg or py files to the runtime path by passing a comma-separated list to `--pyfiles`.

#### Loading Data

##### Parallelized Collections

```
>>> rdd1 = sc.parallelize([('a',1),('a',2),('b',2)])
>>> rdd2 = sc.parallelize([('a',2),('d',1),('b',1)])
>>> rdd3 = sc.parallelize(range(100))
>>> rdd4 = sc.parallelize([('a', "x", "y", "z"),
...                      ('b', "p", "q", "r")])
```

##### External Data

Read either one text file from HDFS, a local file system or any Hadoop-supported file system URI with `textFile()`, or read in a directory of text files with `wholeTextFiles()`.

```
>>> textFile = sc.textFile("my/directory/*.txt")
>>> textFile2 = sc.wholeTextFiles("my/directory/*")
```

## Retrieving RDD Information

### Basic Information

```
>>> rdd.getNumPartitions()      List the number of partitions
>>> rdd.count()                Count RDD instances
3
>>> rdd.countByKey()           Count RDD instances by key
defaultdict(

```

### Summary

|                                             |                                                    |
|---------------------------------------------|----------------------------------------------------|
| <code>&gt;&gt;&gt; rdd1.max()</code>        | Maximum value of RDD elements                      |
| <code>&gt;&gt;&gt; rdd1.min()</code>        | Minimum value of RDD elements                      |
| <code>&gt;&gt;&gt; rdd1.mean()</code>       | Mean value of RDD elements                         |
| <code>&gt;&gt;&gt; rdd1.stdev()</code>      | Standard deviation of RDD elements                 |
| <code>&gt;&gt;&gt; rdd1.variance()</code>   | Compute variance of RDD elements                   |
| <code>&gt;&gt;&gt; rdd1.histogram(3)</code> | Compute histogram by bins                          |
| <code>&gt;&gt;&gt; rdd1.stats()</code>      | Summary statistics (count, mean, stdev, max & min) |

### Applying Functions

|                                                                       |                                                                                     |
|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <code>&gt;&gt;&gt; rdd.map(lambda x: x*(x[1],x[0]))</code>            | Apply a function to each RDD element                                                |
| <code>&gt;&gt;&gt; rdd1.collect()</code>                              | <code>[('a',1),('a',2),('b',2)]</code>                                              |
| <code>&gt;&gt;&gt; rdd2 = rdd.flatMap(lambda x: x*(x[1],x[0]))</code> | <code>[('a',1),('a',2),('b',2),('b',2),('b',2)]</code>                              |
| <code>&gt;&gt;&gt; rdd3.collect()</code>                              | <code>[('a',7),('b',2),('a',2),('a','b',2,2,'b')]</code>                            |
| <code>&gt;&gt;&gt; rdd4.flatMapValues(lambda x: x)</code>             | <code>[('a','a'),('a','b'),('a','c'),('b','p'),('b','c')]</code>                    |
| <code>&gt;&gt;&gt; rdd5.collect()</code>                              | Apply a flatMap function to each (key,value) pair of rdd4 without changing the keys |

### Selecting Data

|                                                                    |                                     |
|--------------------------------------------------------------------|-------------------------------------|
| <code>&gt;&gt;&gt; rdd.collect()</code>                            | Return a list with all RDD elements |
| <code>&gt;&gt;&gt; rdd1.collect(2)</code>                          | Take first 2 RDD elements           |
| <code>&gt;&gt;&gt; rdd2.first()</code>                             | Take first RDD element              |
| <code>&gt;&gt;&gt; rdd3.top(2)</code>                              | Take top 2 RDD elements             |
| <code>&gt;&gt;&gt; rdd3.sample(False, 0.15, 81).collect()</code>   | Return sampled subset of rdd3       |
| <code>&gt;&gt;&gt; rdd4.filter(lambda x: "a" in x)</code>          | Filter the RDD                      |
| <code>&gt;&gt;&gt; rdd5.distinct().collect()</code>                | Return distinct RDD values          |
| <code>&gt;&gt;&gt; rdd6.keys().collect()</code>                    | Return (key,value) RDD's keys       |
| <code>&gt;&gt;&gt; rdd7 = sc.parallelize([('a',1),('a',2)])</code> |                                     |

### Iterating

|                                                  |                                      |
|--------------------------------------------------|--------------------------------------|
| <code>&gt;&gt;&gt; def print(x): print(x)</code> |                                      |
| <code>&gt;&gt;&gt; rdd.foreach(g)</code>         | Apply a function to all RDD elements |
| <code>('a', 7)</code>                            |                                      |
| <code>('b', 2)</code>                            |                                      |
| <code>('a', 2)</code>                            |                                      |

## Reshaping Data

### Reducing

```
>>> rdd.reduceByKey(lambda x,y : x+y)
defaultdict(

```

Merge the rdd values for each key

Merge the rdd values

### Grouping by

```
>>> rdd3.groupByKey(lambda x: x % 2)
defaultdict(

```

Return RDD of grouped values

Group rdd by key

### Aggregating

```
>>> seqOp = (lambda x,y: (x[0]+y,x[1]+1))
>>> combOp = (lambda x,y:(x[0]*y[0],x[1]*y[1]))
>>> rdd3.aggregate((0,0),seqOp,combOp)
(4950, 100)
```

Aggregate RDD elements of each partition and then the results

Aggregate values of each RDD key

### Mathematical Operations

|                                                          |                                                                  |
|----------------------------------------------------------|------------------------------------------------------------------|
| <code>&gt;&gt;&gt; rdd.subtract(rdd2)</code>             | Return each rdd value not contained in rdd2                      |
| <code>&gt;&gt;&gt; rdd2.subtractByKey(rdd)</code>        | Return each (key,value) pair of rdd2 with no matching key in rdd |
| <code>&gt;&gt;&gt; rdd2.cartesian(rdd2).collect()</code> | Return the Cartesian product of rdd and rdd2                     |

### Sort

|                                                                 |                             |
|-----------------------------------------------------------------|-----------------------------|
| <code>&gt;&gt;&gt; rdd2.sortBy(lambda x: x[1]).collect()</code> | Sort RDD by given function  |
| <code>&gt;&gt;&gt; rdd2.sortByKey()</code>                      | Sort (key,value) RDD by key |

### Repartitioning

|                                              |                                                   |
|----------------------------------------------|---------------------------------------------------|
| <code>&gt;&gt;&gt; rdd.repartition(4)</code> | New RDD with 4 partitions                         |
| <code>&gt;&gt;&gt; rdd.coalesce(1)</code>    | Decrease the number of partitions in the RDD to 1 |

### Saving

|                                                                                                                                 |  |
|---------------------------------------------------------------------------------------------------------------------------------|--|
| <code>&gt;&gt;&gt; rdd.saveAsTextFile("rdd.txt")</code>                                                                         |  |
| <code>&gt;&gt;&gt; rdd.saveAsHadoopFile("hdfs://namenodehost/parent/child", "org.apache.hadoop.mapred.TextOutputFormat")</code> |  |

### Stopping SparkContext

```
>>> sc.stop()
```

### Execution

|                                                                   |  |
|-------------------------------------------------------------------|--|
| <code>\$ ./bin/spark-submit examples/src/main/python/pi.py</code> |  |
|-------------------------------------------------------------------|--|

DataCamp

Learn Python for Data Science interactively

