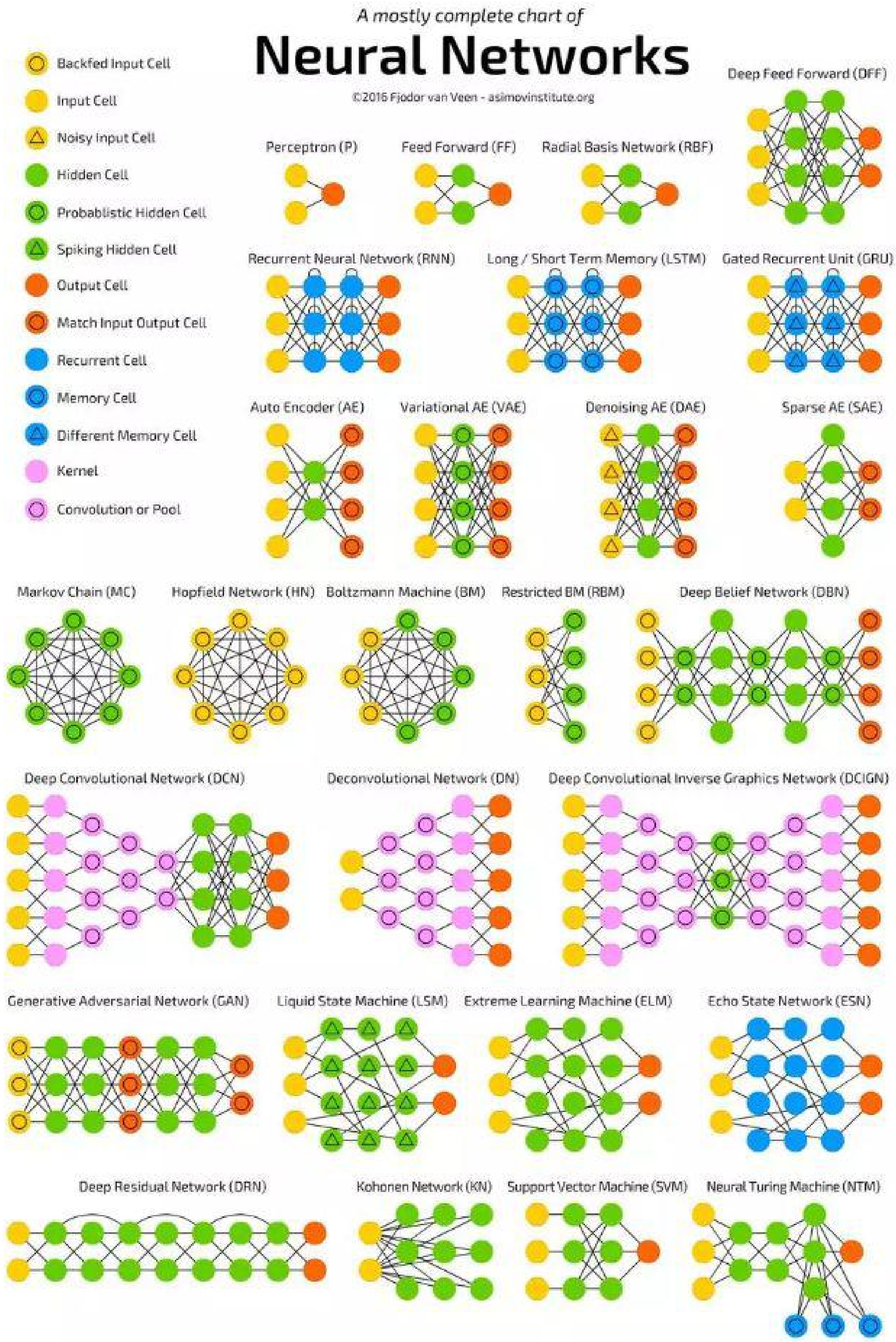
原文作者：Stefan Kojouharov

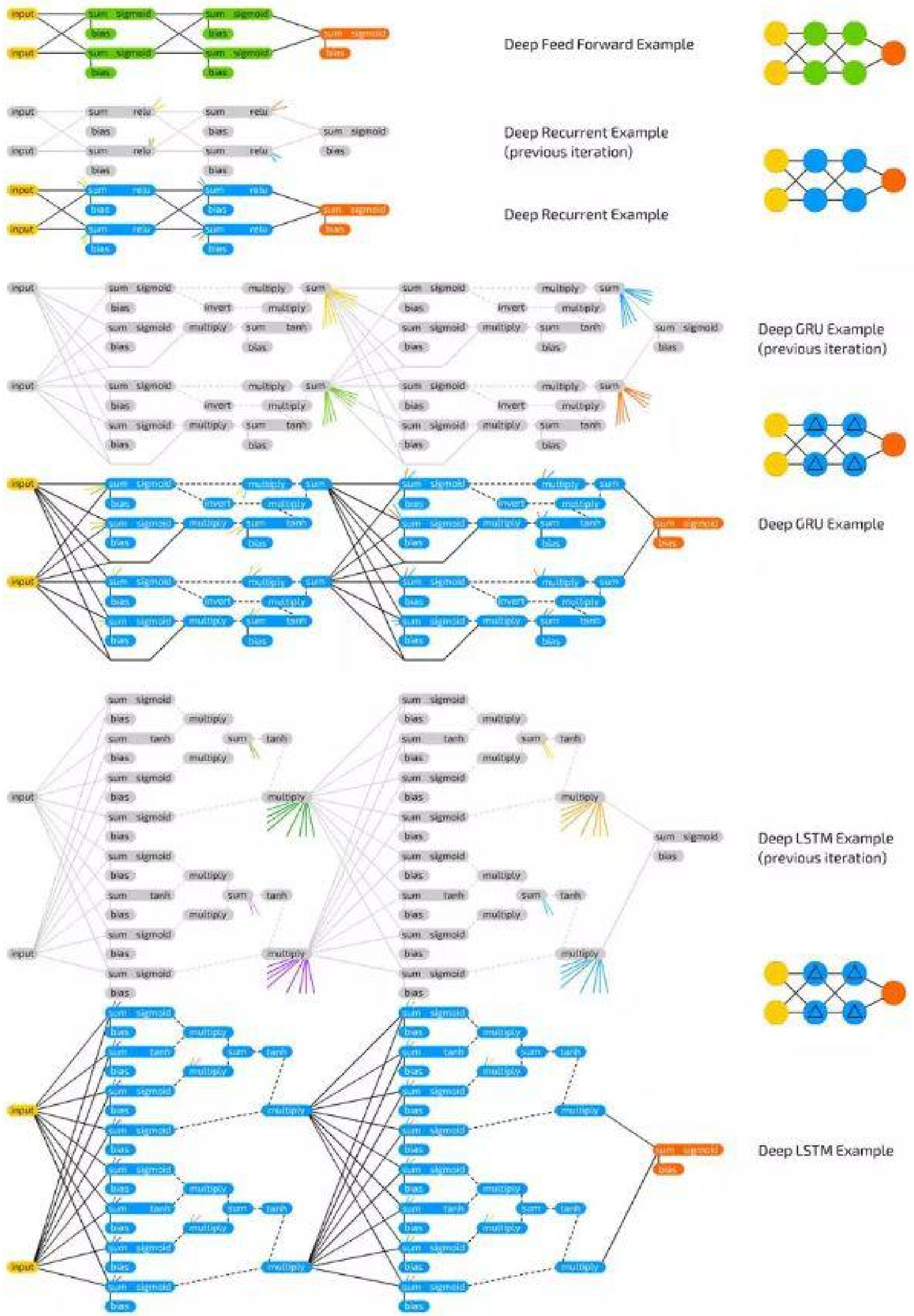
原文链接：

https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463



An informative chart to build

Neural Network Graphs



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) $(a+b)x = ax + bx$. **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 $\{u_1, u_2, \dots, u_m\}$ be a subset of vectors in X. This subset spans X if and only if for every vector $x \in X$ there exist scalars $x_1, x_2, \dots, x_m$ such that $x = x_1u_1 + x_2u_2 + \dots + x_mu_m$.

Inner Product: (x, y) 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: Ascalar 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 θ 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 $v_1, v_2, \dots, v_n$.

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

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

Vector Expansions:

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

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

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 $\mathbf{B} = [v_1\ v_2\ \dots\ v_n]$,
$\mathbf{R} = [r_1\ r_2\ \dots\ r_n]$, $\mathbf{R}^T = \mathbf{B}^{-1}$ In matrix form: $\mathbf{x}^v = \mathbf{B}^{-1}\mathbf{x}^s$

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 $\{u_1, u_2, \dots, u_m\}$ 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: $\mathbf{B}_t = [\mathbf{t}_1\ \mathbf{t}_2\ \dots\ \mathbf{t}_n]$, $\mathbf{B}_w = [\mathbf{w}_1\ \mathbf{w}_2\ \dots\ \mathbf{w}_n]$

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t]$$

Eigenvalues & Eigenvectors: $\mathbf{Az} = \lambda\mathbf{z}$, $|[\mathbf{A} - \lambda\mathbf{I}]| = 0$

Diagonalization: $\mathbf{B} = [\mathbf{z}_1\ \mathbf{z}_2\ \dots\ \mathbf{z}_n]$,
where $\{\mathbf{z}, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ are the eigenvectors of a square matrix A,

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

Perceptron Architecture:
$\mathbf{a} = hardlim(\mathbf{Wp} + \mathbf{b})$, $\mathbf{W} = [\ {}_1\mathbf{w}^T\ {}_2\mathbf{w}^T\ \dots\ {}_S\mathbf{w}^T]^T$,

$$a_i = hardlim(n_i) = hardlim(\ {}_i\mathbf{w}^T\mathbf{p} + b_i)$$

Decision Boundary: ${}_i\mathbf{w}^T\mathbf{p} + b_i = 0$
The decision boundary is always orthogonal to the weight vector.
Single-layer perceptrons can only classify linearly separable vectors.

Perceptron Learning Rule

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T, \ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e},$$

$$\text{where } \mathbf{e} = \mathbf{t} - \mathbf{a}$$

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: $\mathbf{a} = purelin(\mathbf{Wp})$

The Hebb Rule: *Supervised Form:* $w_{ij}^{new} = w_{ij}^{old} + t_{qi}p_{qj}$

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

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

Pseudoinverse Rule: $\mathbf{W} = \mathbf{TP}^+$
When the number, R, of rows of $\mathbf{P}$ is greater than the num ber 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 16): $\mathbf{W}^{new} = (1-\gamma)\mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{p}_q^T$

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

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

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

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

Hessian: $\nabla^2 F(\mathbf{x}) =$

$$\begin{bmatrix} \frac{\partial}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial}{\partial x_1 \partial x_2}F(\mathbf{x}) \dots & \frac{\partial}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial}{\partial x_2^2}F(\mathbf{x}) \dots & \frac{\partial}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial}{\partial x_n \partial x_2}F(\mathbf{x}) \dots & \frac{\partial}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

Directional Derivatives:
1st Dir.Der.: $\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|}$, **2nd 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 $F(x) < F(x + \Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.
Global Minimum: if $F(x) < F(x + \Delta x)$ for all $\Delta x \neq 0$
Weak Minimum: if it is not a strong minimum, and a scalar $\delta > 0$ exists, such that $F(x) \leq F(x + \Delta x)$ for all Δx such that $\delta > \|\Delta x\| > 0$.

Necessary Conditions for Optimality:
1st-Order Condition: $\nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} = 0$ (Stationary Points)
2nd-Order Condition: $\nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*} \geq 0$ (Positive Semi-definite Hessian Matrix).

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

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

General Minimization Algorithm:

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ *or* $\Delta\mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$

Steepest Descent Algorithm:

$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$ **where,** $\mathbf{g}_k = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$

Stable Learning Rate: $(\alpha_k = \alpha$, **constant**$)$ $\alpha < \frac{2}{\lambda_{max}}$

$\{\lambda_1\ \lambda_2, \ldots, \lambda_n\}$ Eigenvalues of Hessian matrix **A**

Learning Rate to Minimize Along the Line:

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \overset{is}{\Rightarrow} \alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k{}^T \mathbf{A} \mathbf{p}_k}$ (For quadratic fn.)

After Minimization Along the Line:

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \Rightarrow \mathbf{g}_{k+1}^T \mathbf{p}_k = 0$

ADALINE: $\mathbf{a} = purelin(\mathbf{Wp} + \mathbf{b})$

Mean Square Error: *(for ADALINE it is a quadratic fn.)*

$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t - \mathbf{x}^T\mathbf{z})^2]$

$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\,\mathbf{x}$,

$c = E[t^2]$, $\mathbf{h} = E[t\mathbf{z}]$ *and* $\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] \Rightarrow \mathbf{A} = 2\mathbf{R}, \mathbf{d} = -2\mathbf{h}$

Unique minimum, if it exists, is $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$,

where $\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}$ and $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$

LMS Algorithm: $\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\,\mathbf{e}(k)\,\mathbf{p}^T(k)$

$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\,\mathbf{e}(k)$

Convergence Point: $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$

Stable Learning Rate: $0 < \alpha < 1/\lambda_{max}$ where λ_{max} is the maximum eigenvalue of **R**

Adaptive Filter ADALINE:

$$a(k) = purelin(\mathbf{Wp}(k) + b) = \sum_{i=1}^{R} \mathbf{w}_{1,i}\, y(k - i + 1) + b$$

Backpropagation Algorithm:

Performance Index:

Mean Square error: $F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})]$

Approximate Performance Index: (single sample)

$\hat{F}(x) = \mathbf{e}^T(k)\mathbf{e}(k) = \left(\mathbf{t}(k) - \mathbf{a}(k)\right)^T(\mathbf{t}(k) - \mathbf{a}(k))$

Sensitivity: $\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left[\frac{\partial \hat{F}}{\partial \mathbf{n}_1^m} \quad \frac{\partial \hat{F}}{\partial \mathbf{n}_2^m} \quad \cdots \quad \frac{\partial \hat{F}}{\partial \mathbf{n}_{S^m}^m}\right]^T$

Forward Propagation: $\mathbf{a}^0 = \mathbf{p}$,

$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1})$ *for* $m = 0, 1, \ldots, M-1$

$$\mathbf{a} = \mathbf{a}^M$$

Backward Propagation: $\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$,

$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}$ *for* $m = M-1, \ldots, 2, 1$, where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \mathrm{diag}\left(\left[\dot{f}^m(n_1^m) \quad \dot{f}^m(n_2^m) \quad \ldots \quad \dot{f}^m(n_{S^m}^m)\right]\right)$$

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

Weight Update (Approximate Steepest Descent):

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

***Heuristic Variations of Backpropagation:**

Batching: 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.)

Backpropagation with Momentum (MOBP):

$$\Delta\mathbf{W}^m(k) = \gamma\Delta\mathbf{W}^m(k-1) - (1-\gamma)\alpha\,\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

$$\Delta\mathbf{b}^m(k) = \gamma\Delta\mathbf{b}^m(k-1) - (1-\gamma)\alpha\,\mathbf{s}^m$$

Variable Learning Rate Backpropagation (VLBP)

1. If the squared error (over the entire training set) increases by more than some set percentage ζ (typically one to five percent) after a weight update, then the weight update is discarded, the learning rate is multiplied by some factor $\rho < 1$, and the momentum coefficient γ (if it is used) is set to zero.
2. If the squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by some factor $\eta > 1$. If γ has been previously set to zero, it is reset to its original value.
3. If the squared error increases by less than ζ, then the weight update is accepted but the learning rate and the momentum coefficient are unchanged.

Association: $\mathbf{a} = hardlim(\mathbf{W}^0\mathbf{p}^0 + \mathbf{Wp} + b)$

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**.

Associative Learning Rules:

Unsupervised Hebb Rule:

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha\,\mathbf{a}(q)\mathbf{p}^T(q)$$

Hebb with Decay:

$$\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha\,\mathbf{a}(q)\mathbf{p}^T(q)$$

Instar: $\mathbf{a} = hardlim(\mathbf{Wp} + b)$, $\mathbf{a} = hardlim({}_1\mathbf{w}^T\mathbf{p} + b)$

The instar is activated for ${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\|\|\mathbf{p}\|cos\theta \geq -b$

where θ is the angle between $\mathbf{p}$ and ${}_1\mathbf{w}$.

Instar Rule:

${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\, a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$

${}_i\mathbf{w}(q) = (1-\alpha)\ {}_i\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q)$, $if\ (a_i(q) = 1)$

Kohonen Rule:

${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\,(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$ *for* $i \in X(q)$

Outstar Rule: $\mathbf{a} = satlins(\mathbf{Wp})$

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha\left(\mathbf{a}(q) - \mathbf{w}_j(q-1)\right)p_j(q)$$

Competitive Layer: $\mathbf{a} = compet(\mathbf{Wp}) = compet(\mathbf{n})$

Competitive Learning with the Kohonen Rule:

$${}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1) + \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}(q-1)\right)$$

$$= (1-\alpha)\,{}_{i^*}\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q)$$

${}_{i}\mathbf{w}(q) = {}_{i}\mathbf{w}(q-1)$, $i \neq i^*$ where i^* is the winning neuron.

Self-Organizing with the Kohonen Rule:

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha\left(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)\right)$$

$= (1-\alpha)\ {}_i\mathbf{w}(q-1) + \alpha\,\mathbf{p}(q)$, $i \in N_{i^*}(d)$

$$N_i(d) = \{j, d_{i,j} \leq d\}$$

LVQ Network: $(w_{k,i}^2 = 1) \Rightarrow$ subclass i is a part of class k

$n_i^1 = -\|{}_i\mathbf{w}^1 - \mathbf{p}\|$, $\mathbf{a}^1 = compet(\mathbf{n}^1)$, $\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1$

LVQ Network Learning with the Kohonen Rule:

$${}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) + \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right), \quad if\ a_{k^*}^2 = t_{k^*} = 1$$

$${}_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) - \alpha\left(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)\right), \quad if\ a_{k^*}^2 = 1 \neq t_{k^*} = 0$$

hardlim: $a = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases}$, ***hardlims***: $a = \begin{cases} -1 & n < 0 \\ +1 & n \geq 0 \end{cases}$, ***purelin***: $a = n$, ***Logsig***: $a = \frac{1}{1+e^{-n}}$, ***tansig***: $a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$, ***poslin***: $a = \begin{cases} 0 & n < 0 \\ n & n \geq 0 \end{cases}$,

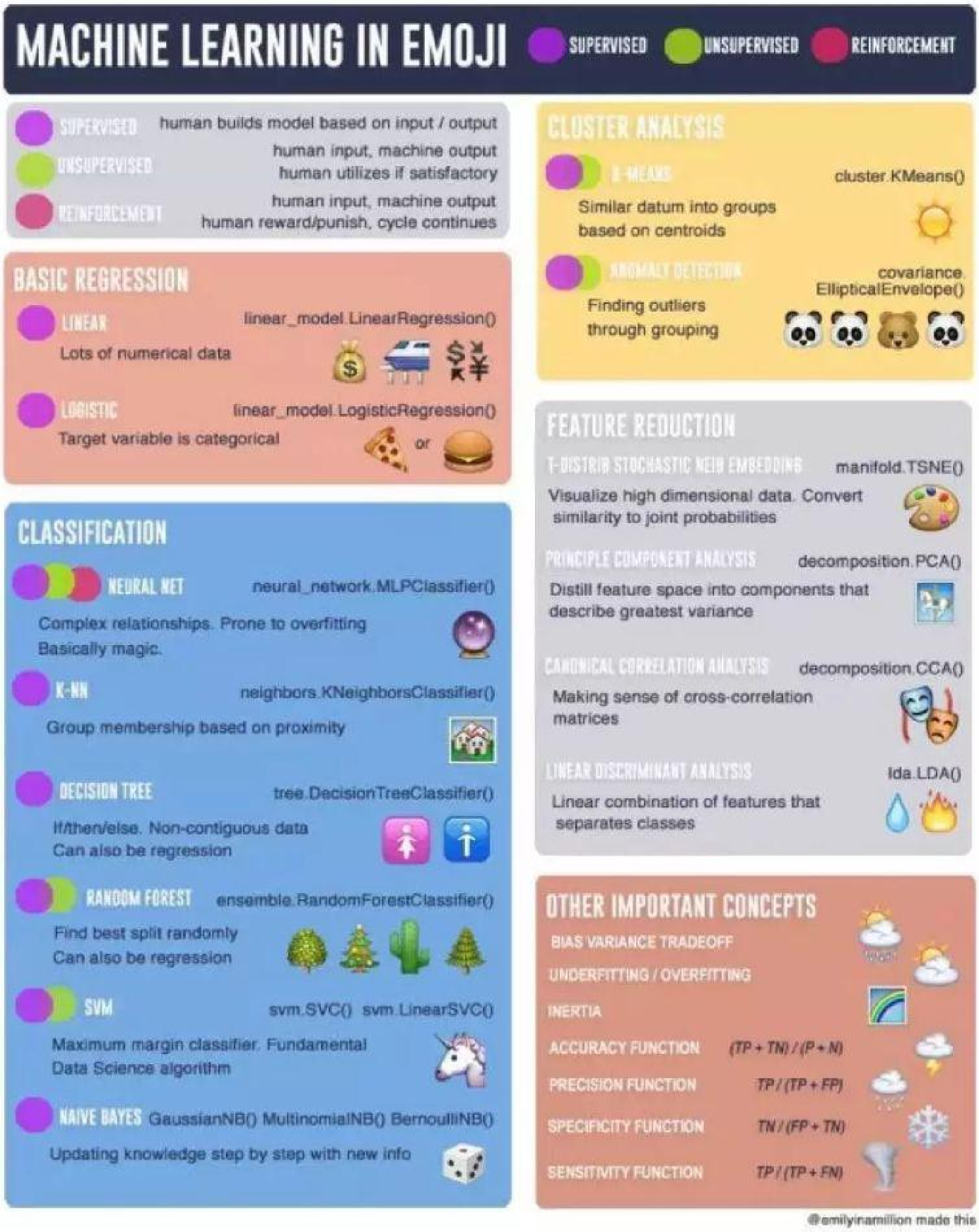
compet: $a = \begin{cases} 1 & \text{neuron with max } n \\ 0 & \text{all other neurons} \end{cases}$, ***satlin***: $a = \begin{cases} 0 & n < 0 \\ n & 0 \leq n \leq 1 \\ 1 & n > 1 \end{cases}$, ***satlins***: $a = \begin{cases} -1 & n < -1 \\ n & -1 \leq n \leq 1 \\ 1 & n > 1 \end{cases}$,

Delay: $a(t) = u(t-1)$, ***Integrator***: $a(t) = \int_0^t u(\tau)d\tau + a(0)$

****HINT:**

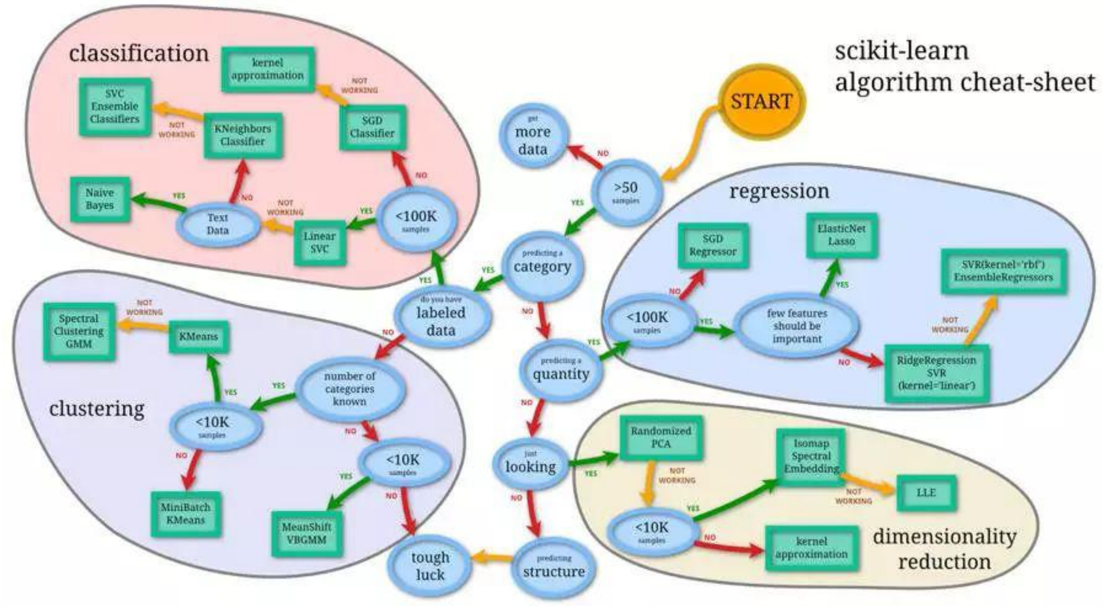
$diag([1\ 2\ 3]) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

机器学习概览



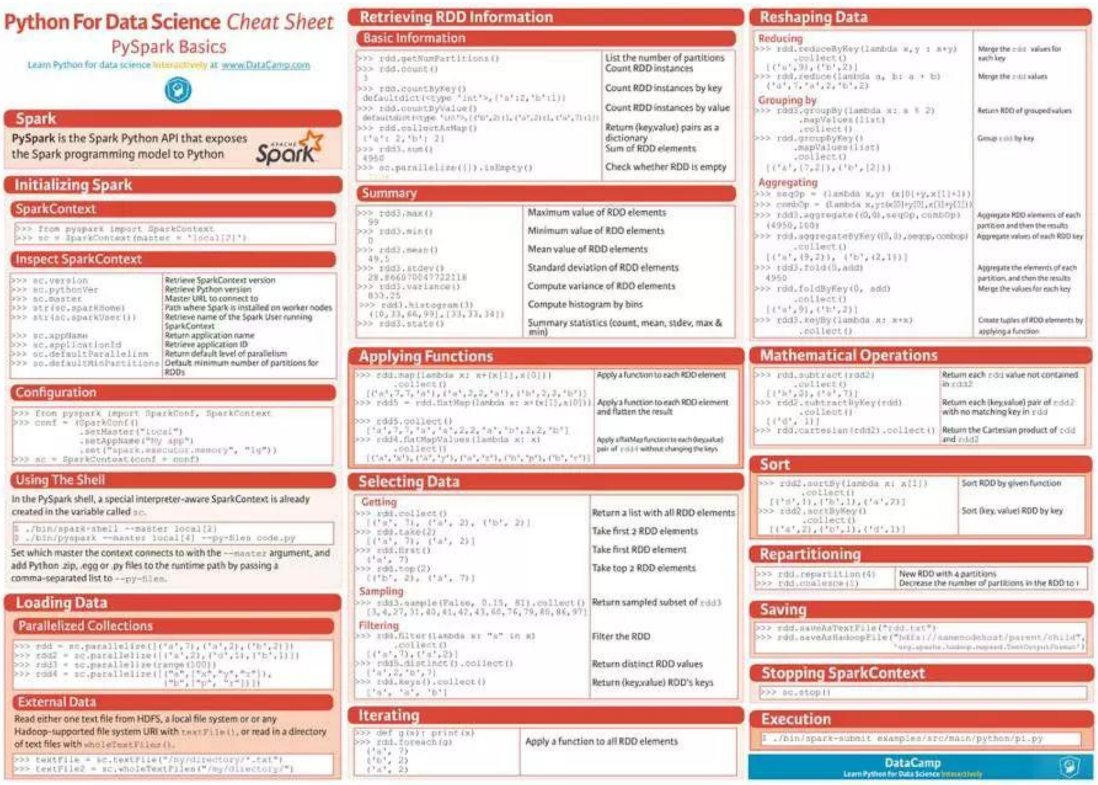
MACHINE LEARNING IN EMOJI
SUPERVISED
UNSUPERVISED
REINFORCEMENT
SUPERVISED human builds model based on input / output
UNSUPERVISED human input, machine output human utilizes if satisfactory
REINFORCEMENT human input, machine output human reward/punish, cycle continues
BASIC REGRESSION
LINEAR linear_model.LinearRegression()
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-DISTRIB STOCHASTIC NEIB 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 lda.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

机器学习：Scikit-learn 算法



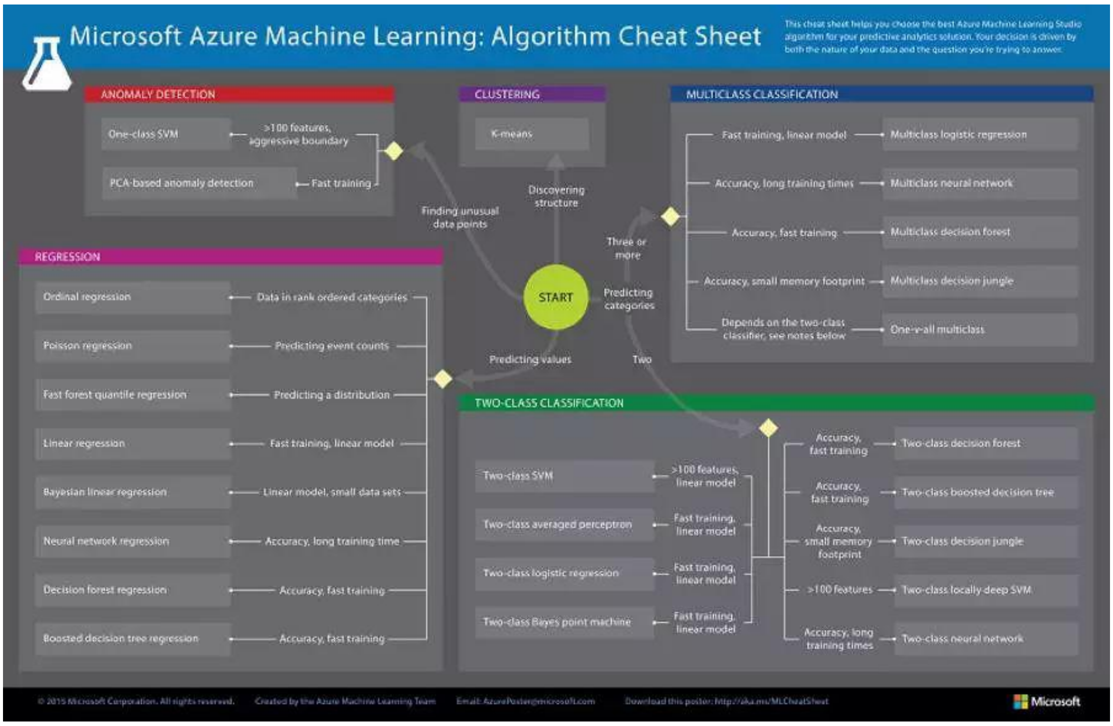
Scikit-learn

Scikit-learn 是 Python 实现的一个免费机器学习算法库，基本涵盖了机器学习涉及到的各个方面，包括数据预处理、特征提取、模型构建、模型训练、模型验证以及模型评价等等。Scikit-learn 还可以和 Python 的 NumPy 和 SciPy 库协同工作。

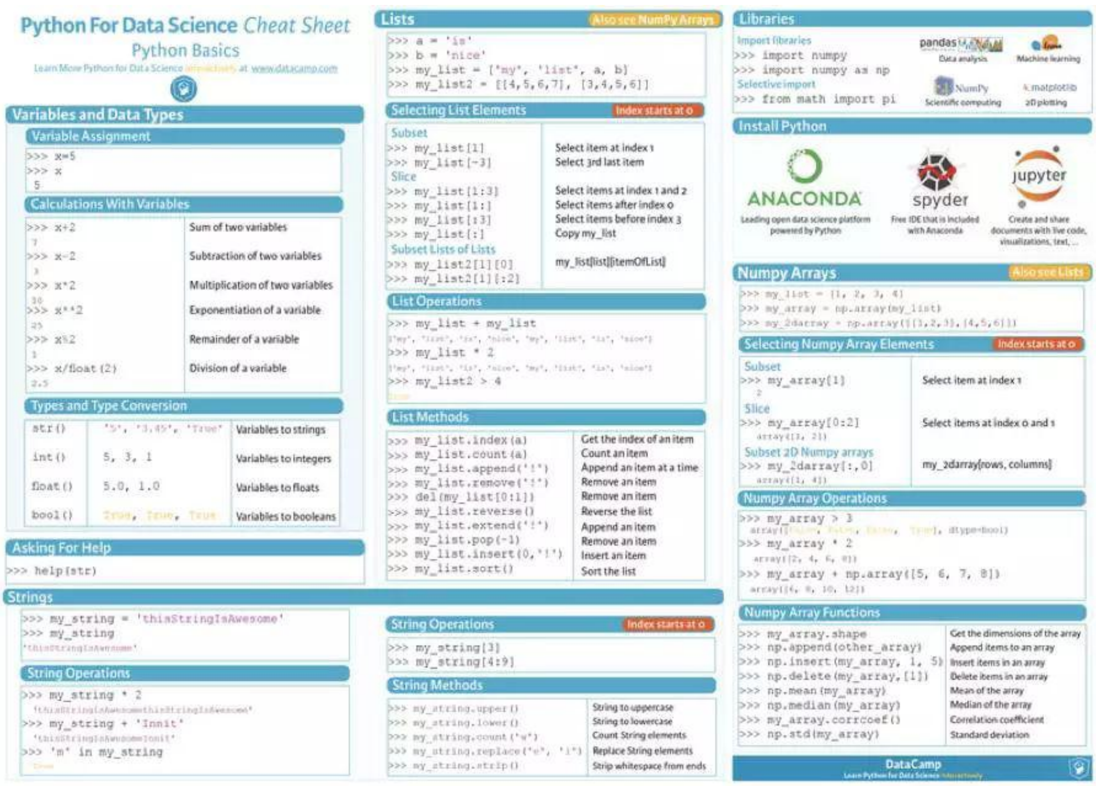


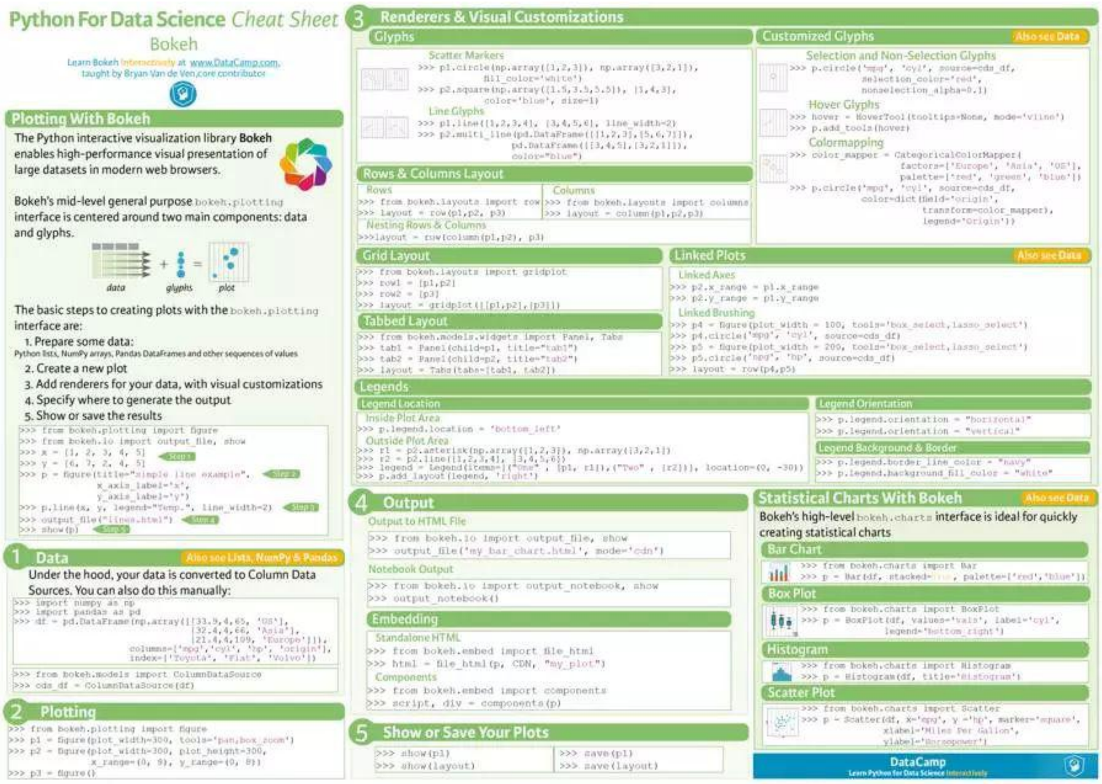
机器学习之算法

这一来自 Microsoft Azure 的 cheatsheet 能够帮助你为你的预测分析选择合适的机器学习算法，根据不同的数据性质为你推荐最合适的算法。



用 Python 实现的数据科学





Python For Data Science Cheat Sheet
Bokeh
Learn Bokeh Interactively at 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.

data + glyphs = plot

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",              Step 2
               x_axis_label='x',
               y_axis_label='y')
>>> p.line(x, y, legend="Temp.", line_width=2)           Step 3
>>> output_file("lines.html")                            Step 4
>>> show(p)                                              Step 5
```

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")
```

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(field='origin',
                        transform=color_mapper),
             legend='Origin'))
```

Rows & Columns Layout

Rows
```
>>> from bokeh.layouts import row
>>> layout = row(p1,p2, p3)
```
Columns
```
>>> from bokeh.layouts import columns
>>> 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])
```

Linked Plots (Also see Data)

Linked Axes
```
>>> p2.x_range = p1.x_range
>>> p2.y_range = p1.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)
```

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])
>>> legend = Legend(items=[("One" , [p1, r1]),("Two" , [r2])], location=(0, -30))
>>> p.add_layout(legend, 'right')
```

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"
```

4 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)
>>> show(layout)
>>> save(p1)
>>> save(layout)
```

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

Tensor Flow

TensorFlow™ 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个 CPU（或 GPU），服务器，移动设备等等。TensorFlow 最初由 Google 大脑小组（隶属于 Google 机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但这个系统的通用性使其也可广泛用于其他计算领域。

ALTOROS

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

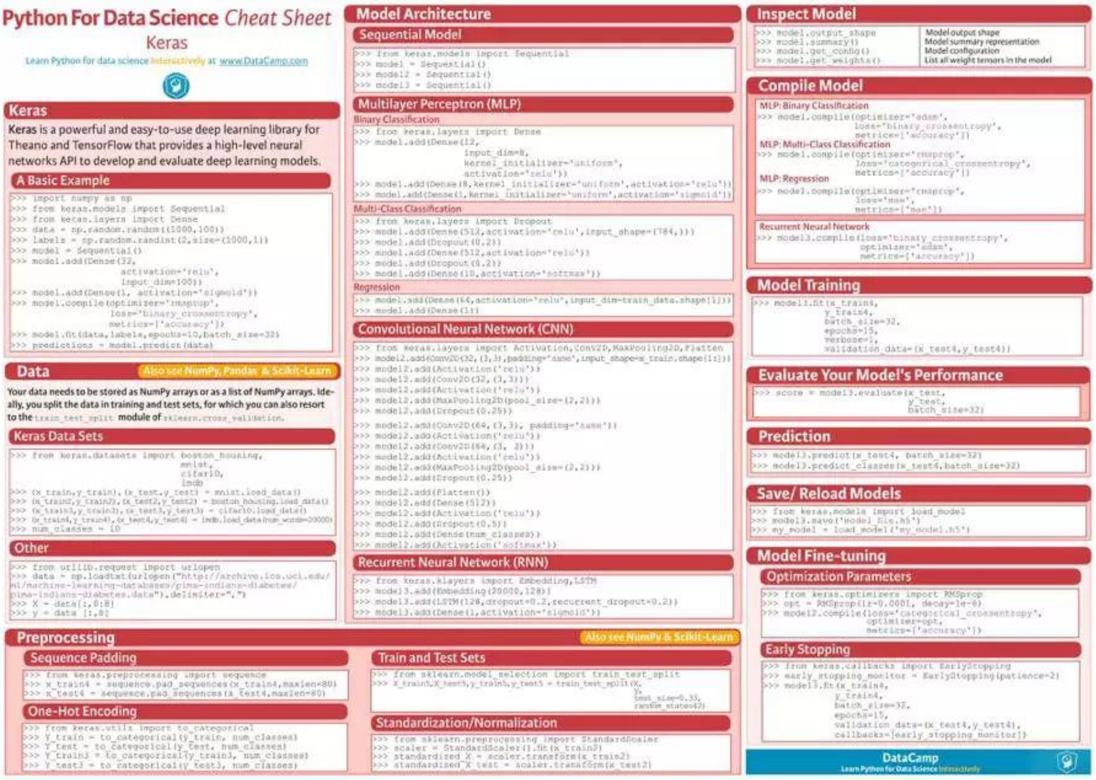
Main classes

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,

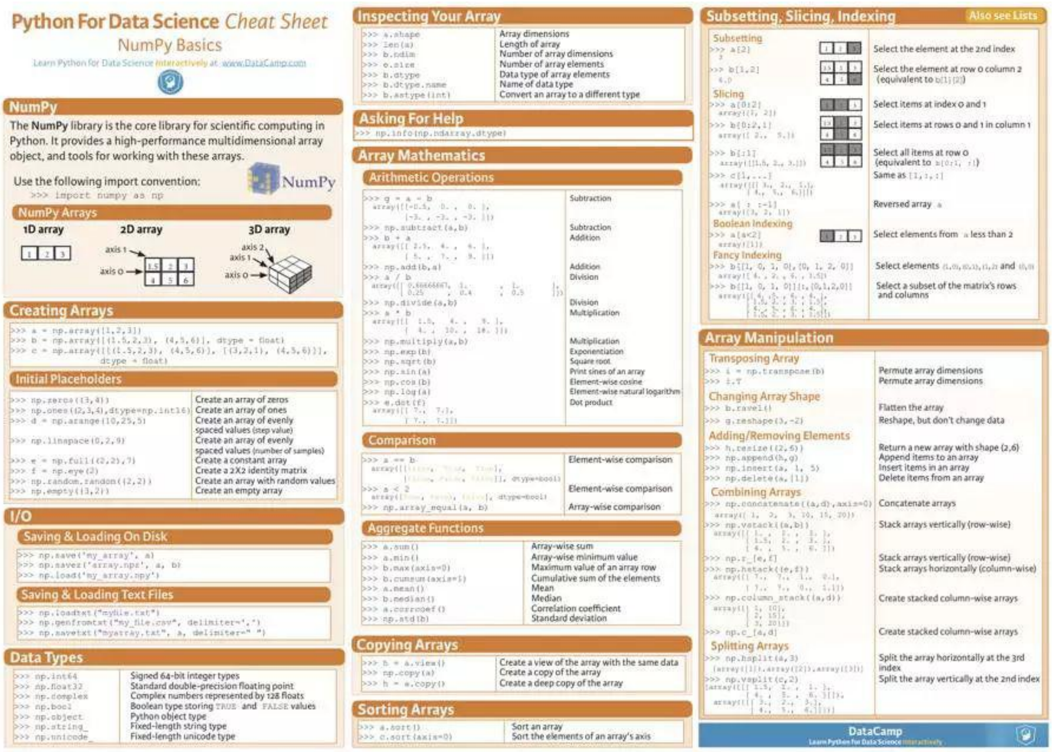
Keras

2017 年，Google 的 TensorFlow 团队决定在 TensorFlow 的核心库中支持 Keras。 Chollet 解释说，Keras 被认为是一个接口（interface）而不是一个端到端（end-to-end）的机器学习框架。 它提供了一个更高级，更直观的抽象集，使得配置神经网络变得更容易，无论后端科学计算库如何。



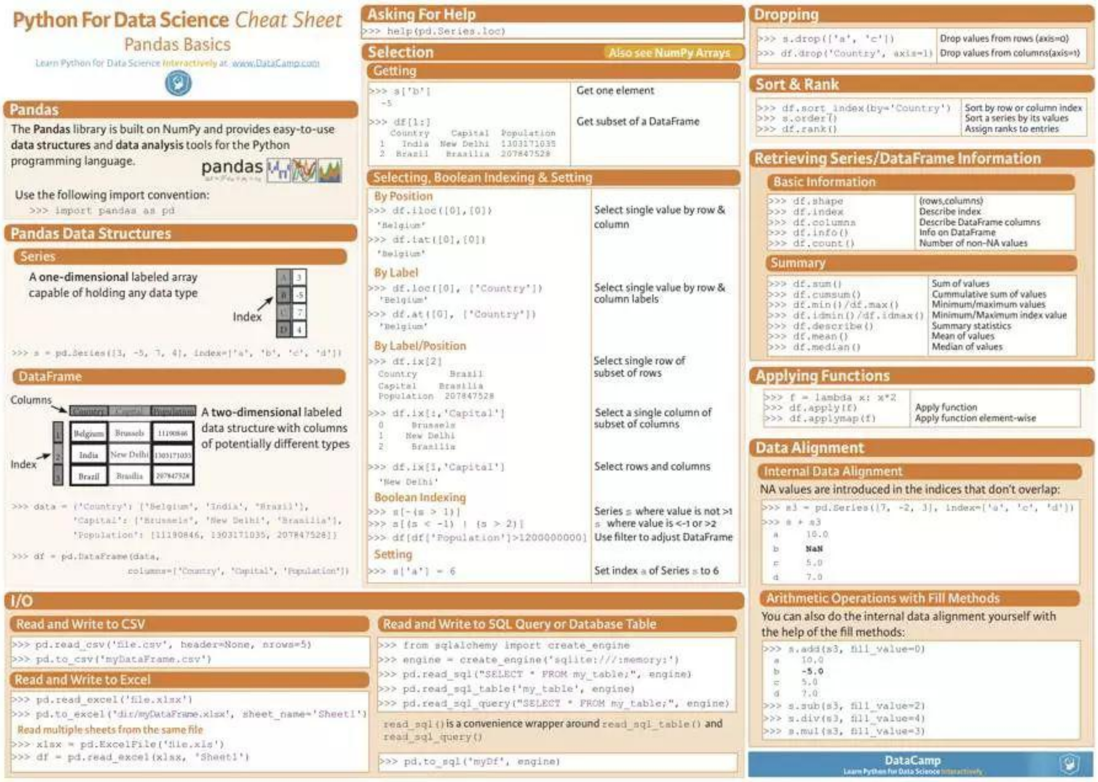
Numpy

NumPy 的目标是 Python 的 CPython 参考实现，这是一个非优化的字节码解释器（non-optimizingbytecodeinterpreter）。为这个版本的 Python 编写的数学算法通常运行速度要比其他版本的慢。NumPy 通过提供在数组上高效运行的多维数组和函数和运算符来解决缓慢问题，这需要重写一些代码，主要用到 NumPy 的内部循环。



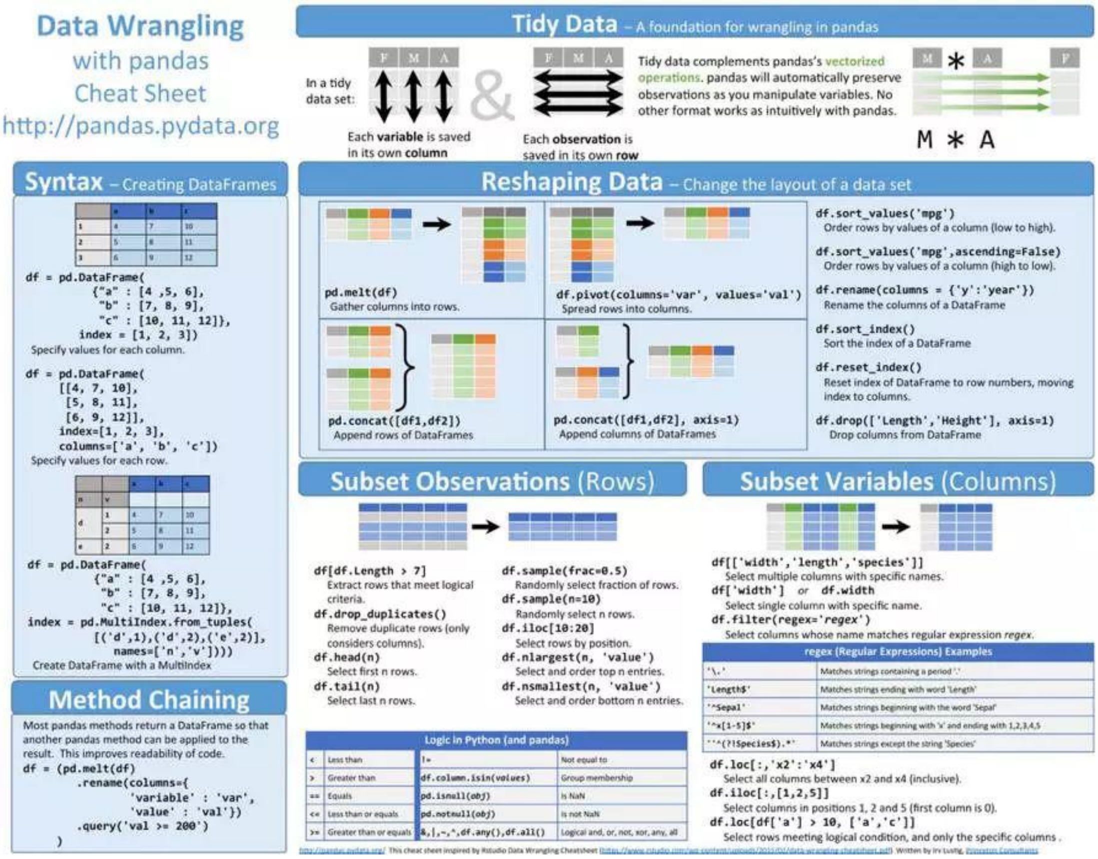
Pandas

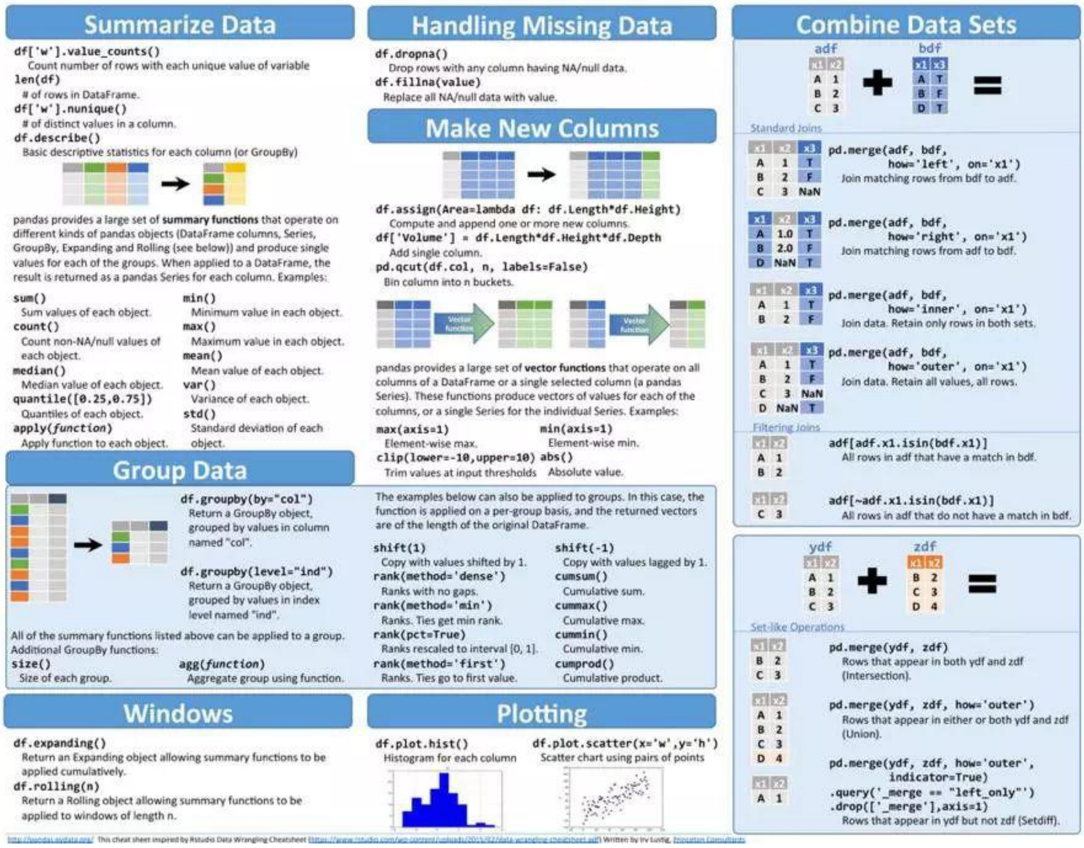
“Pandas”这 个 名 字 来 源 于 “面 板 数 据 （ panel data ）”一 词 ， 是 一 个 多 维 结 构 化 数 据 集 的 计 量 经 济 学 术 语 。



Data Wrangling

“Data wrangler”一 词 开 始 渗 透 到 流 行 文 化 之 中 。 在 2017 年 的 电 影 “金 刚 ： 骷 髅 岛 ”中 ， 演 员 马 克 ·埃 文 ·杰 克 逊 （ Marc Evan Jackson ） 饰 演 的 角 色 就 被 介 绍 为 “史 蒂 夫 ·伍 德 沃 德 （ SteveWoodward ） ， 我 们 的 data wrangler”。


Data Wrangling
with pandas
Cheat Sheet
http://pandas.pydata.org
Tidy Data – A foundation for wrangling in pandas
In a tidy data set:
Each variable is saved in its own column
Each observation is saved in its own row
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
Syntax – Creating DataFrames
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.
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')
)
Reshaping Data – Change the layout of a data set
pd.melt(df)
Gather columns into rows.
df.pivot(columns='var', values='val')
Spread rows into columns.
pd.concat([df1,df2])
Append rows of DataFrames
pd.concat([df1,df2], axis=1)
Append columns of DataFrames
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.
df.sample(frac=0.5)
Randomly select fraction of rows.
df.sample(n=10)
Randomly select n rows.
df.iloc[10:20]
Select rows by position.
df.nlargest(n, 'value')
Select and order top n entries.
df.nsmallest(n, 'value')
Select and order bottom n entries.
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.any(),df.all() Logical and, or, not, xor, any, all
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 .



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.

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

`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.
`shift(-1)` Copy with values lagged by 1.
`cumsum()` Cumulative sum.
`cummax()` Cumulative max.
`cummin()` Cumulative min.
`cumprod()` Cumulative product.

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

Standard Joins

`pd.merge(adf, bdf, how='left', on='x1')` Join matching rows from bdf to adf.
`pd.merge(adf, bdf, how='right', on='x1')` Join matching rows from adf to bdf.
`pd.merge(adf, bdf, how='inner', on='x1')` Join data. Retain only rows in both sets.
`pd.merge(adf, bdf, how='outer', on='x1')` Join data. Retain all values, all rows.

Filtering Joins

`adf[adf.x1.isin(bdf.x1)]` All rows in adf that have a match in bdf.
`adf[~adf.x1.isin(bdf.x1)]` All rows in adf that do not have a match in bdf.

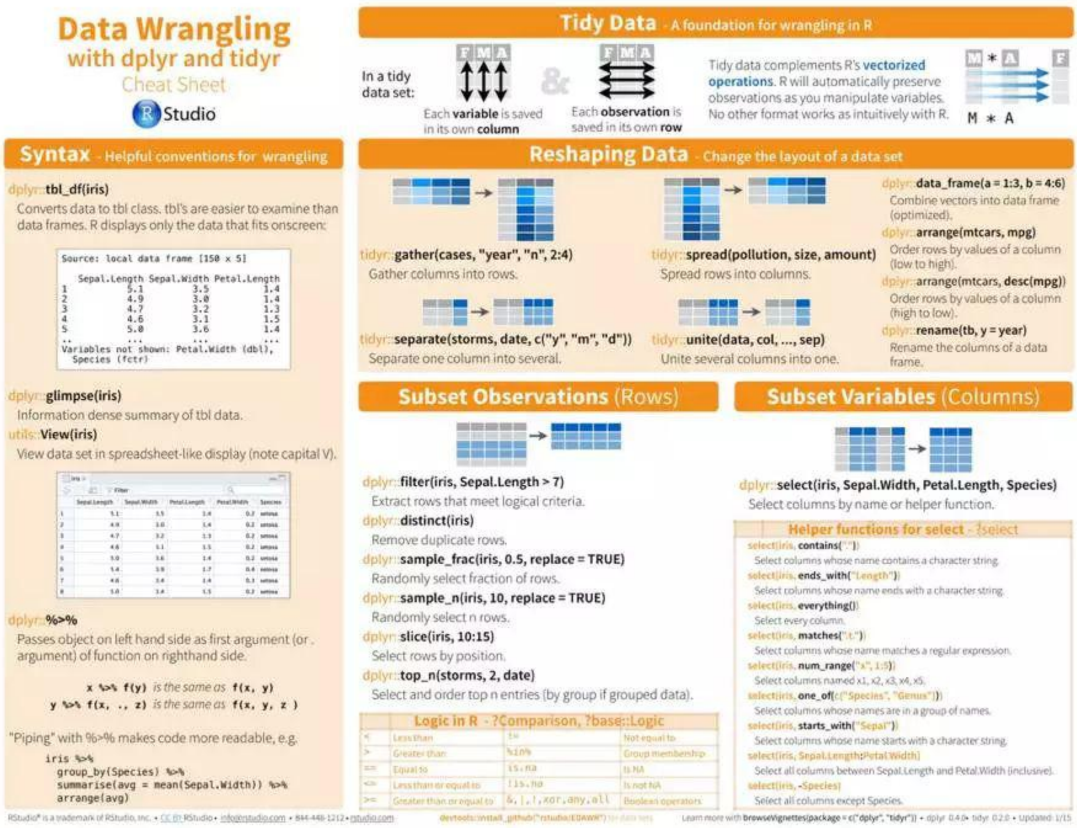
ydf, zdf

Set-like Operations

`pd.merge(ydf, zdf)` Rows that appear in both ydf and zdf (Intersection).
`pd.merge(ydf, zdf, how='outer')` Rows that appear in either or both ydf and zdf (Union).
`pd.merge(ydf, zdf, how='outer', indicator=True) .query('_merge == "left_only"') .drop(['_merge'],axis=1)` Rows that appear in ydf but not zdf (Setdiff).

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

Data Wrangling with dplyr and tidyr



Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

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
1          5.1         3.5          1.4
2          4.9         3.0          1.4
3          4.7         3.2          1.3
4          4.6         3.1          1.5
5          5.0         3.6          1.4
..         ...         ...          ...
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).

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)
```

Tidy Data - A foundation for wrangling in R

In a tidy data set: Each **variable** is saved in its own **column** & Each **observation** is saved in its own **row**

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

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)** Gather columns into rows.
tidyr::**spread(pollution, size, amount)** Spread rows into columns.
tidyr::**separate(storms, date, c("y", "m", "d"))** Separate one column into several.
tidyr::**unite(data, col, ..., sep)** Unite several columns into one.
dplyr::**data_frame(a = 1:3, b = 4:6)** Combine vectors into data frame (optimized).
dplyr::**arrange(mtcars, mpg)** Order rows by values of a column (low to high).
dplyr::**arrange(mtcars, desc(mpg))** Order rows by values of a column (high to low).
dplyr::**rename(tb, y = year)** 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	&,|,!,xor,any,all	Boolean operators

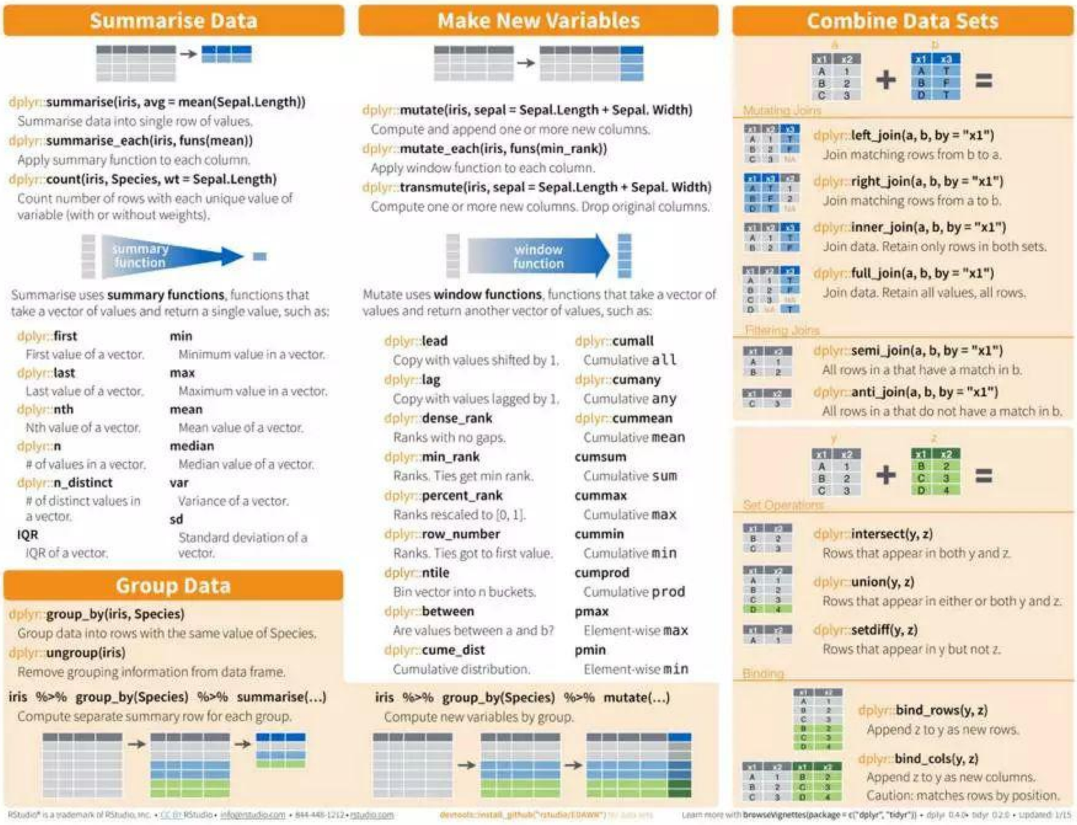
Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)** Select columns by name or helper function.

Helper functions for select - ?select

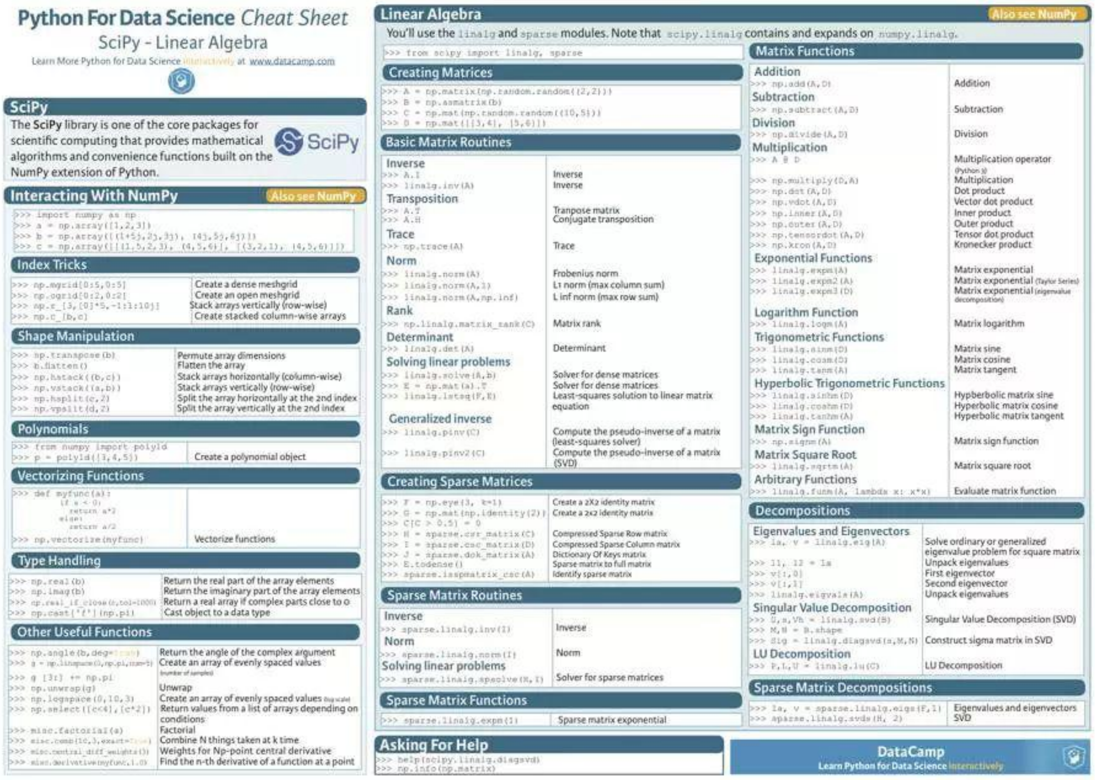
select(iris, **contains(".")**) Select columns whose name contains a character string.
select(iris, **ends_with("Length")**) Select columns whose name ends with a character string.
select(iris, **everything()**) Select every column.
select(iris, **matches(".t.")**) Select columns whose name matches a 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.

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", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15



SciPy

SciPy 构建在 NumPy 数组对象（array object）上，是 NumPy 堆栈（stack）的一部分。NumPy 堆栈包括 Matplotlib，pandas 和 SymPy 等工具，以及一套扩展的科学计算库。 这个 NumPy 堆栈与其他应用程序(如 MATLAB，GNU Octave 和 Scilab)具有相似的用户。 NumPy 堆栈有时也被称为 SciPy 堆栈。


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+5j,2j,3j), (4j,5j,6j)])
>>> 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.vpslit(d,2)`	Split the array vertically at the 2nd index

Polynomials
`>>> from numpy import poly1d`	
`>>> p = poly1d([3,4,5])`	Create a polynomial object

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)`	Return the angle of the complex argument
`>>> g = np.linspace(0,np.pi,num=5)`	Create an array of evenly spaced values (number of samples)
`>>> g [3:] += np.pi`	
`>>> np.unwrap(g)`	Unwrap
`>>> np.logspace(0,10,3)`	Create an array of evenly spaced values (log scale)
`>>> np.select([c<4],[c*2])`	Return values from a list of arrays depending on conditions
`>>> misc.factorial(a)`	Factorial
`>>> misc.comb(10,3,exact=True)`	Combine N things taken at k time
`>>> misc.central_diff_weights(3)`	Weights for Np-point central derivative
`>>> misc.derivative(myfunc,1.0)`	Find the n-th derivative of a function at a point

Linear Algebra
Also see NumPy

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

Creating Matrices
```
>>> A = np.matrix(np.random.random((2,2)))
>>> B = np.asmatrix(b)
>>> C = np.mat(np.random.random((10,5)))
>>> D = np.mat([[3,4], [5,6]])
```

Basic Matrix Routines
Inverse	
`>>> A.I`	Inverse
`>>> linalg.inv(A)`	Inverse
Transposition	
`>>> A.T`	Tranpose matrix
`>>> A.H`	Conjugate transposition
Trace	
`>>> np.trace(A)`	Trace
Norm	
`>>> linalg.norm(A)`	Frobenius norm
`>>> linalg.norm(A,1)`	L1 norm (max column sum)
`>>> linalg.norm(A,np.inf)`	L inf norm (max row sum)
Rank	
`>>> np.linalg.matrix_rank(C)`	Matrix rank
Determinant	
`>>> linalg.det(A)`	Determinant
Solving linear problems	
`>>> linalg.solve(A,b)`	Solver for dense matrices
`>>> E = np.mat(a).T`	Solver for dense matrices
`>>> linalg.lstsq(F,E)`	Least-squares solution to linear matrix equation
Generalized inverse	
`>>> linalg.pinv(C)`	Compute the pseudo-inverse of a matrix (least-squares solver)
`>>> linalg.pinv2(C)`	Compute the pseudo-inverse of a matrix (SVD)

Creating Sparse Matrices
`>>> F = np.eye(3, k=1)`	Create a 2X2 identity matrix
`>>> G = np.mat(np.identity(2))`	Create a 2x2 identity matrix
`>>> C[C > 0.5] = 0`	
`>>> H = sparse.csr_matrix(C)`	Compressed Sparse Row matrix
`>>> I = sparse.csc_matrix(D)`	Compressed Sparse Column matrix
`>>> J = sparse.dok_matrix(A)`	Dictionary Of Keys matrix
`>>> E.todense()`	Sparse matrix to full matrix
`>>> sparse.isspmatrix_csc(A)`	Identify sparse matrix

Sparse Matrix Routines
Inverse	
`>>> sparse.linalg.inv(I)`	Inverse
Norm	
`>>> sparse.linalg.norm(I)`	Norm
Solving linear problems	
`>>> sparse.linalg.spsolve(H,I)`	Solver for sparse matrices

Sparse Matrix Functions
`>>> sparse.linalg.expm(I)`	Sparse matrix exponential

Matrix Functions
Addition	
`>>> np.add(A,D)`	Addition
Subtraction	
`>>> np.subtract(A,D)`	Subtraction
Division	
`>>> np.divide(A,D)`	Division
Multiplication	
`>>> A @ D`	Multiplication operator (Python 3)
`>>> np.multiply(D,A)`	Multiplication
`>>> np.dot(A,D)`	Dot product
`>>> np.vdot(A,D)`	Vector dot product
`>>> np.inner(A,D)`	Inner product
`>>> np.outer(A,D)`	Outer product
`>>> np.tensordot(A,D)`	Tensor dot product
`>>> np.kron(A,D)`	Kronecker product
Exponential Functions	
`>>> linalg.expm(A)`	Matrix exponential
`>>> linalg.expm2(A)`	Matrix exponential (Taylor Series)
`>>> linalg.expm3(D)`	Matrix exponential (eigenvalue decomposition)
Logarithm Function	
`>>> linalg.logm(A)`	Matrix logarithm
Trigonometric Functions	
`>>> linalg.sinm(D)`	Matrix sine
`>>> linalg.cosm(D)`	Matrix cosine
`>>> linalg.tanm(A)`	Matrix tangent
Hyperbolic Trigonometric Functions	
`>>> linalg.sinhm(D)`	Hypberbolic matrix sine
`>>> linalg.coshm(D)`	Hyperbolic matrix cosine
`>>> linalg.tanhm(A)`	Hyperbolic matrix tangent
Matrix Sign Function	
`>>> np.signm(A)`	Matrix sign function
Matrix Square Root	
`>>> linalg.sqrtm(A)`	Matrix square root
Arbitrary Functions	
`>>> linalg.funm(A, lambda x: x*x)`	Evaluate matrix function

Decompositions
Eigenvalues and Eigenvectors	
`>>> la, v = linalg.eig(A)`	Solve ordinary or generalized eigenvalue problem for square matrix
`>>> l1, l2 = la`	Unpack eigenvalues
`>>> v[:,0]`	First eigenvector
`>>> v[:,1]`	Second eigenvector
`>>> linalg.eigvals(A)`	Unpack eigenvalues
Singular Value Decomposition	
`>>> U,s,Vh = linalg.svd(B)`	Singular Value Decomposition (SVD)
`>>> M,N = B.shape`	
`>>> Sig = linalg.diagsvd(s,M,N)`	Construct sigma matrix in SVD
LU Decomposition	
`>>> P,L,U = linalg.lu(C)`	LU Decomposition

Sparse Matrix Decompositions
`>>> la, v = sparse.linalg.eigs(F,1)`	Eigenvalues and eigenvectors
`>>> sparse.linalg.svds(H, 2)`	SVD

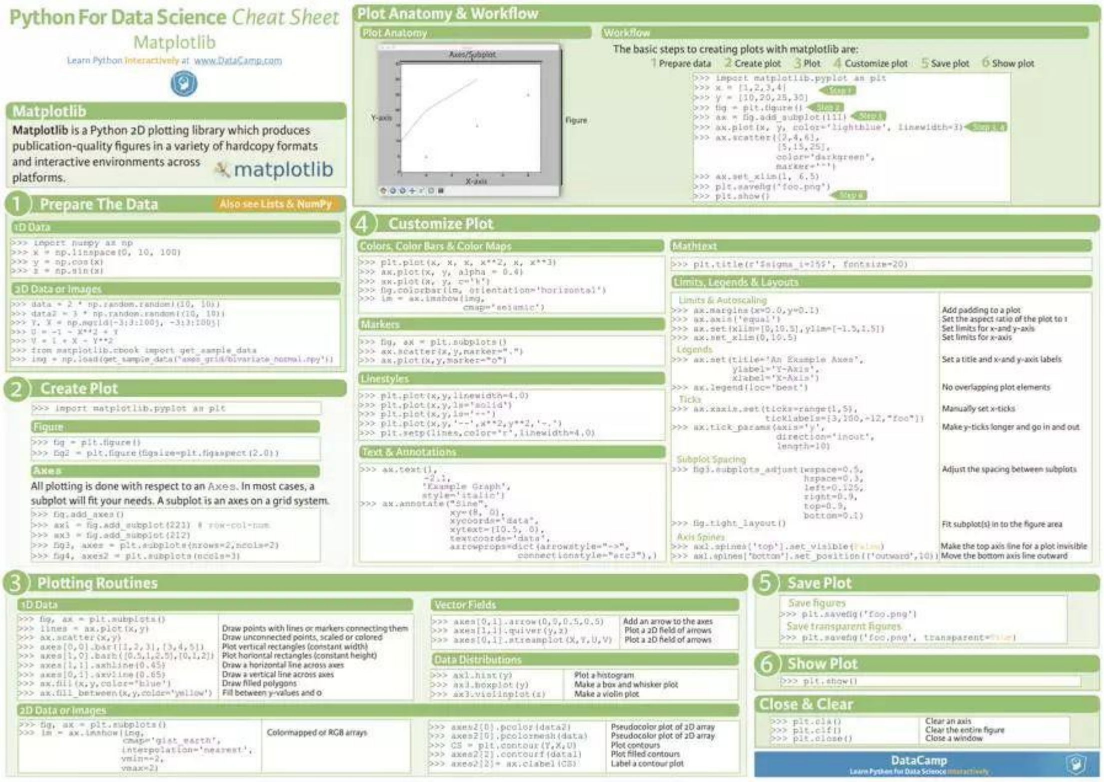
Asking For Help
```
>>> help(scipy.linalg.diagsvd)
>>> np.info(np.matrix)
```

DataCamp
Learn Python for Data Science Interactively

Matplotlib

matplotlib 是 Python 编程语言的一个绘图库及其数值数学扩展 NumPy。它为利用通用的图形用户界面工具包，如 Tkinter, wxPython, Qt 或 GTK+向应用程序嵌入式绘图提供了面向对象的应用程序接口（API）。还有一个基于状态机（如开放图形库 OpenGL）的程序 pylab 接口，设计成与 MATLAB 非常类似--尽管使用起来有些不堪。SciPy 就利用了 matplotlib。

pyplot 是 matplotlib 的一个模块，它提供了一个类似 MATLAB 的接口。matplotlib 被设计得用起来像 MATLAB，具有使用 Python 的能力。免费是其优点。



Python For Data Science *Cheat Sheet*

Matplotlib

Learn Python Interactively at 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.

matplotlib

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))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> 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_axes()
>>> 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)
```

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

3 Plotting Routines

1D Data

`>>> fig, ax = plt.subplots()`	
`>>> lines = ax.plot(x,y)`	Draw points with lines or markers connecting them
`>>> ax.scatter(x,y)`	Draw unconnected points, scaled or colored
`>>> axes[0,0].bar([1,2,3],[3,4,5])`	Plot vertical rectangles (constant width)
`>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])`	Plot horiontal rectangles (constant height)
`>>> axes[1,1].axhline(0.45)`	Draw a horizontal line across axes
`>>> axes[0,1].axvline(0.65)`	Draw a vertical line across axes
`>>> ax.fill(x,y,color='blue')`	Draw filled polygons
`>>> ax.fill_between(x,y,color='yellow')`	Fill between y-values and 0

Vector Fields

`>>> axes[0,1].arrow(0,0,0.5,0.5)`	Add an arrow to the axes
`>>> axes[1,1].quiver(y,z)`	Plot a 2D field of arrows
`>>> axes[0,1].streamplot(X,Y,U,V)`	Plot a 2D field of arrows

Data Distributions

`>>> ax1.hist(y)`	Plot a histogram
`>>> ax3.boxplot(y)`	Make a box and whisker plot
`>>> ax3.violinplot(z)`	Make a violin plot

2D Data or Images

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

Colormapped or RGB arrays

`>>> axes2[0].pcolor(data2)`	Pseudocolor plot of 2D array
`>>> axes2[0].pcolormesh(data)`	Pseudocolor plot of 2D array
`>>> CS = plt.contour(Y,X,U)`	Plot contours
`>>> axes2[2].contourf(data1)`	Plot filled contours
`>>> axes2[2]= ax.clabel(CS)`	Label a contour plot

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> 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,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
            -2.1,
            'Example Graph',
            style='italic')
>>> ax.annotate("Sine",
                xy=(8, 0),
                xycoords='data',
                xytext=(10.5, 0),
                textcoords='data',
                arrowprops=dict(arrowstyle="->",
                                connectionstyle="arc3"),)
```

Mathtext

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

Limits, Legends & Layouts

Limits & Autoscaling

`>>> ax.margins(x=0.0,y=0.1)`	Add padding to a plot
`>>> ax.axis('equal')`	Set the aspect ratio of the plot to 1
`>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])`	Set limits for x-and y-axis
`>>> ax.set_xlim(0,10.5)`	Set limits for x-axis

Legends

`>>> ax.set(title='An Example Axes', ylabel='Y-Axis', xlabel='X-Axis')`	Set a title and x-and y-axis labels
`>>> ax.legend(loc='best')`	No overlapping plot elements

Ticks

`>>> ax.xaxis.set(ticks=range(1,5), ticklabels=[3,100,-12,"foo"])`	Manually set x-ticks
`>>> ax.tick_params(axis='y', direction='inout', length=10)`	Make y-ticks longer and go in and out

Subplot Spacing

`>>> fig3.subplots_adjust(wspace=0.5, hspace=0.3, left=0.125, right=0.9, top=0.9, bottom=0.1)`	Adjust the spacing between subplots
`>>> fig.tight_layout()`	Fit subplot(s) in to the figure area

Axis Spines

`>>> ax1.spines['top'].set_visible(False)`	Make the top axis line for a plot invisible
`>>> ax1.spines['bottom'].set_position(('outward',10))`	Move the bottom axis line outward

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

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

6 Show Plot

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

Close & Clear

`>>> plt.cla()`	Clear an axis
`>>> plt.clf()`	Clear the entire figure
`>>> plt.close()`	Close a window

DataCamp

Learn Python for Data Science Interactively

数据可视化

Data Visualization
with ggplot2
Cheat Sheet

RStudio

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.**

data geom coordinate system plot

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

data geom coordinate system plot

Build a graph with **qplot()** or **ggplot()**

aesthetic mappings data geom

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(method ="lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

add layers, elements with +
layer = geom + default stat + layer specific mappings
additional elements

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()
Returns 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

Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path(**lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, linetype, size

d + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)**
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment(**aes(
xend = long + delta_long,
yend = lat + delta_lat)**)**
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect(**aes(xmin = long, ymin = lat,
xmax= long + delta_long,
ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

f + **geom_smooth(**model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text(**aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step(**direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map(**aes(map_id = state), map = map**)** +
expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

m + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

m + **geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size

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")**

data stat geom coordinate system plot

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. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function · layer specific mappings · variable created by transformation

i + **stat_density2d(aes(fill = ..level..), geom = "polygon", n = 100)**

geom for layer · parameters for stat

1D distributions

a + **stat_bin(**binwidth = 1, origin = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + **stat_bindot(**binwidth = 1, binaxis = "x"**)**
x, y, | ..count.., ..ncount..
a + **stat_density(**adjust = 1, kernel = "gaussian"**)**
x, y, | ..count.., ..density.., ..scaled..

2D distributions

f + **stat_bin2d(**bins = 30, drop = TRUE**)**
x, y, fill | ..count.., ..density..
f + **stat_binhex(**bins = 30**)**
x, y, fill | ..count.., ..density..
f + **stat_density2d(**contour = TRUE, n = 100**)**
x, y, color, size | ..level..

3 Variables

m + **stat_contour(**aes(z = z)**)**
x, y, z, order | ..level..
m+ **stat_spoke(**aes(radius= z, angle = z)**)**
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + **stat_summary_hex(**aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..
m + **stat_summary2d(**aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..

Comparisons

g + **stat_boxplot(**coef = 1.5**)**
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + **stat_ydensity(**adjust = 1, kernel = "gaussian", scale = "area"**)**
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

f + **stat_ecdf(**n = 40**)**
x, y | ..x.., ..y..
f + **stat_quantile(**quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq"**)**
x, y | ..quantile.., ..x.., ..y..
f + **stat_smooth(**method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95**)**
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

General Purpose

ggplot() + **stat_function(**aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5)**)**
x | ..y..
f + **stat_identity()**
ggplot() + **stat_qq(**aes(sample=1:100), distribution = qt, dparams = list(df=5)**)**
sample, x, y | ..x.., ..y..
f + **stat_sum()**
x, y, size | ..size..
f + **stat_summary(**fun.data = "mean_cl_boot"**)**
f + **stat_unique()**

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(**aes(fill = fl)**)**
n

scale_ · aesthetic to adjust · prepackaged scale to use · scale specific arguments

n + **scale_fill_manual(**
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), **breaks** =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R")**)**

range of values to include in mapping · title to use in legend/axis · labels to use in legend/axis · breaks to use in legend/axis

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()**)** - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = date_format("%m/%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

n <- b + **geom_bar(**aes(fill = fl)**)**

n + **scale_fill_brewer(**palette = "Blues"**)**
For palette choices: library(RcolorBrewer) display.brewer.all()

n + **scale_fill_grey(**start = 0.2, end = 0.8, na.value = "red"**)**

Continuous

o <- a + **geom_dotplot(**aes(fill = ..x..)**)**

o + **scale_fill_gradient(**low = "red", high = "yellow"**)**

o + **scale_fill_gradient2(**low = "red", hight = "blue", mid = "white", midpoint = 25**)**

o + **scale_fill_gradientn(**colours = terrain.colors(6)**)**
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

Shape scales

p <- f + **geom_point(**aes(shape = fl)**)**

p + **scale_shape(**solid = FALSE**)**

p + **scale_shape_manual(**values = c(3:7)**)**
Shape values shown in chart on right

Manual shape values

Size scales

q <- f + **geom_point(**aes(size = cyl)**)**

q + **scale_size_area(**max = 6**)**
Value mapped to area of circle (not radius)

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(**ytrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

z + **coord_map(**projection = "ortho", orientation=c(41, -74, 0)**)**
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

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

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

s + geom_bar(**position = "dodge"**)
Arrange elements side by side

s + geom_bar(**position = "fill"**)
Stack elements on top of one another, normalize height

s + geom_bar(**position = "stack"**)
Stack elements on top of one another

f + geom_point(**position = "jitter"**)
Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

s + geom_bar(**position = position_dodge(width = 1)**)

Themes

r + **theme_bw()**
White background with grid lines

r + **theme_classic()**
White background no gridlines

r + **theme_grey()**
Grey background (default theme)

r + **theme_minimal()**
Minimal theme

ggthemes - Package with additional ggplot2 themes

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()

t + **facet_grid(. ~ fl)**
facet into columns based on fl

t + **facet_grid(year ~ .)**
facet into rows based on year

t + **facet_grid(year ~ fl)**
facet into both rows and columns

t + **facet_wrap(~ fl)**
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(y ~ 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 **labeller** to adjust facet labels

t + facet_grid(. ~ fl, **labeller = label_both)**

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(. ~ fl, **labeller = label_bquote(**alpha ^ .(x)**))**

α^c	α^d	α^e	α^p	α^r

t + facet_grid(. ~ fl, **labeller = label_parsed)**

c	d	e	p	r

Labels

t + **ggtitle(**"New Plot Title"**)**
Add a main title above the plot

t + **xlab(**"New X label"**)**
Change the label on the X axis

t + **ylab(**"New Y label"**)**
Change the label on the Y axis

t + **labs(**title =" New title", x = "New x", y = "New y"**)**
All of the above

Use scale functions to update legend labels

Legends

t + **theme(**legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right"

t + **guides(**color = "none"**)**
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

t + **scale_fill_discrete(**name = "Title", labels = c("A", "B", "C")**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)

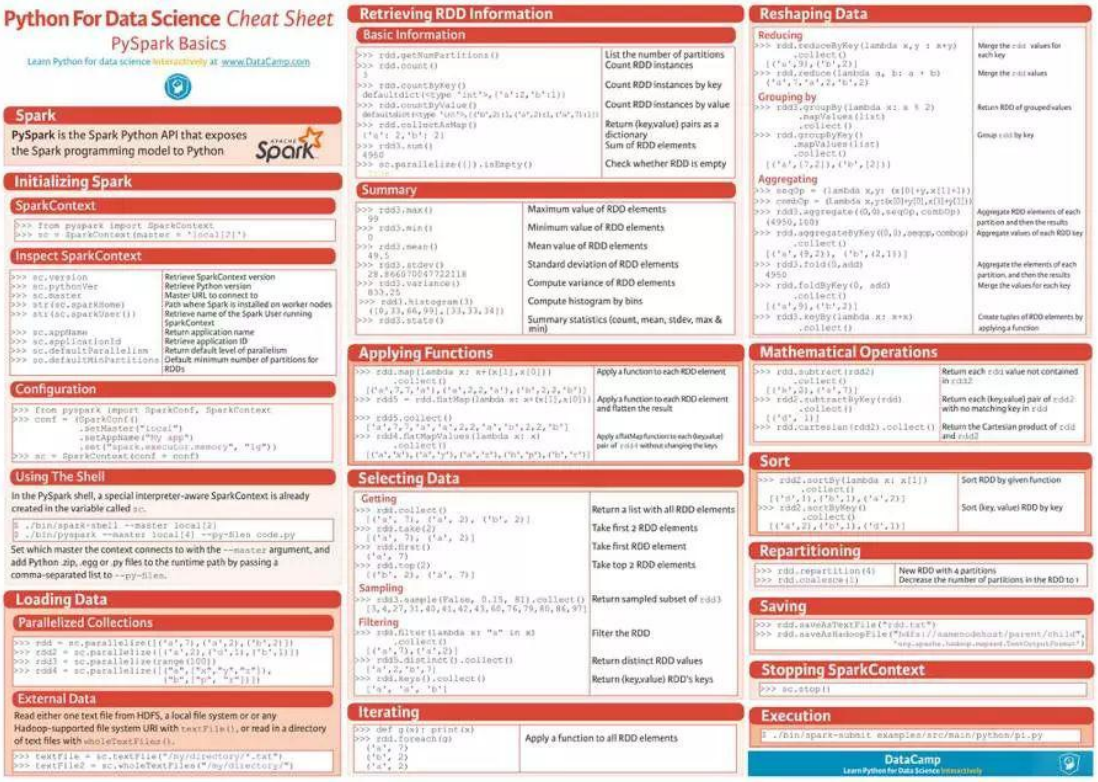
t + **coord_cartesian(**
xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points)

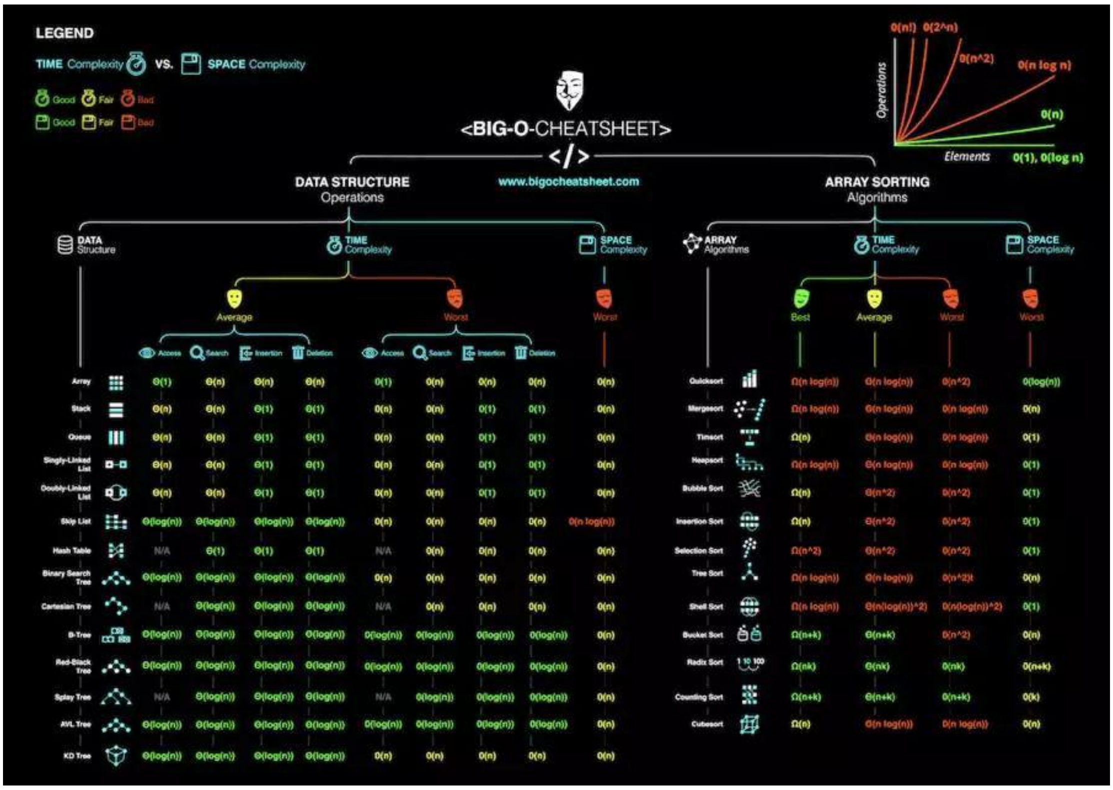
t + **xlim(**0, 100**)** + **ylim(**10, 20**)**

t + **scale_x_continuous(**limits = c(0, 100)**)** +
scale_y_continuous(limits = c(0, 100)**)**

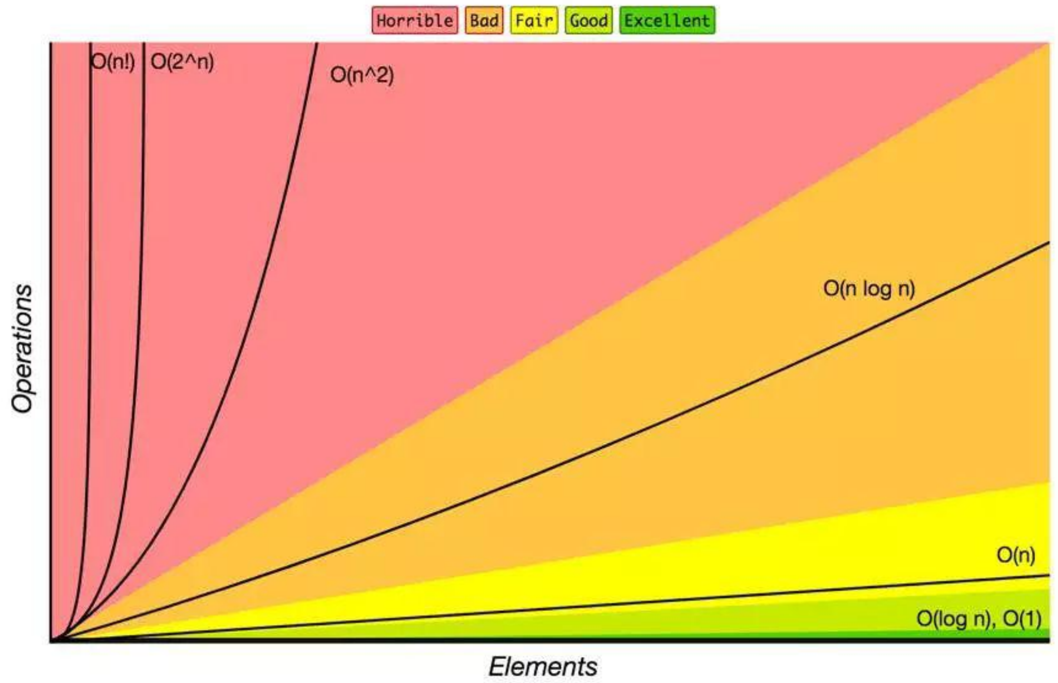
PySpark



Big-O



Big-O Complexity Chart



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

参考资料：

Big-O Algorithm Cheat Sheet: http://bigocheatsheet.com/
Bokeh Cheat Sheet:https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf
Data Science Cheat Sheet: https://www.datacamp.com/community/tutorials/python-data-science-cheat-sheet-basics
Data Wrangling Cheat Sheet:https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
Data Wrangling: https://en.wikipedia.org/wiki/Data_wrangling
Ggplot Cheat Sheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
Keras Cheat Sheet: https://www.datacamp.com/community/blog/keras-cheat-sheet#gs.DRKeNMs
Keras: https://en.wikipedia.org/wiki/Keras
Machine Learning Cheat Sheet: https://ai.icymi.email/new-machinelearning-cheat-sheet-by-emily-barry-abdsc/
Machine Learning Cheat Sheet: https://docs.microsoft.com/en-in/azure/machine-learning/machine-learning-algorithm-cheat-sheet
ML Cheat Sheet:: http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html
Matplotlib Cheat Sheet: https://www.datacamp.com/community/blog/py

thon-matplotlib-cheat-sheet#gs.uEKySpY
Matpotlib: https://en.wikipedia.org/wiki/Matplotlib
Neural Networks Cheat Sheet: http://www.asimovinstitute.org/neural-network-zoo/
Neural Networks Graph Cheat Sheet: http://www.asimovinstitute.org/blog/
Neural Networks: https://www.quora.com/Where-can-find-a-cheat-sheet-for-neural-network
Numpy Cheat Sheet: https://www.datacamp.com/community/blog/python-numpy-cheat-sheet#gs.AK5ZBgE
NumPy: https://en.wikipedia.org/wiki/NumPy
Pandas Cheat Sheet: https://www.datacamp.com/community/blog/python-pandas-cheat-sheet#gs.oundfxM
Pandas: https://en.wikipedia.org/wiki/Pandas_(software)
Pandas Cheat Sheet: https://www.datacamp.com/community/blog/pandas-cheat-sheet-python#gs.HPFoRIc
Pyspark Cheat Sheet: https://www.datacamp.com/community/blog/pyspark-cheat-sheet-python#gs.L=J1zxQ
Scikit Cheat Sheet: https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet
Scikit-learn: https://en.wikipedia.org/wiki/Scikit-learn
Scikit-learn Cheat Sheet: http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html
Scipy Cheat Sheet: https://www.datacamp.com/community/blog/python-scipy-cheat-sheet#gs.JDSg30I
SciPy: https://en.wikipedia.org/wiki/SciPy
TesorFlow Cheat Sheet: https://www.altoros.com/tensorflow-cheat-sheet.html
Tensor Flow: https://en.wikipedia.org/wiki/TensorFlow