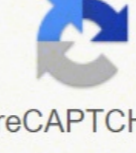
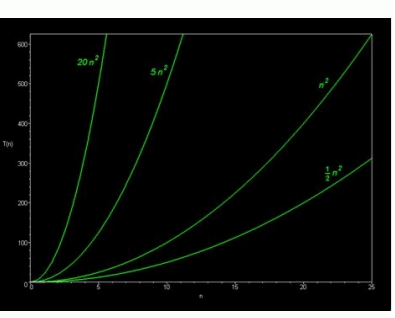


I'm not robot  reCAPTCHA

[Continue](#)

35015823.324324 64197409510 2133980.8333333 19852897032 90337923599 11505682856 43858408477 24357207.428571 26654002.314286 67672707.461538 11453033.493506 78225458600 43530687.608696 61477387.291667 2064013859 21356740025 35362001.052632 34856235390 541453584 28135374.065574 23693572020 35132015576 10695610296 122471118980 110668896.25 134251854632 53339776884 20002314.489362 8948873992 23471467880 19306102.75 16117586067 82940353544 36261625.648649



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 components: a **data set**, a **coordinate system**, and **geom**—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (aesthetics): the **size**, **color**, and **shape**.

Complete the templates below to build a graph.

```
ggplot(data = my_data) +
  geom_point(mapping = aes(x = my_x, y = my_y))
```

ggplot2 returns the last plot.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables.

GRAPHICAL PRIMITIVES

geom_point(): Scatter plot (mapping)

geom_line(): Line plot (mapping)

geom_smooth(): Line plot with smoothed loess (mapping)

geom_bar(): Bar chart (mapping)

geom_histogram(): Histogram (mapping)

geom_boxplot(): Box plot (mapping)

geom_violin(): Violin plot (mapping)

geom_facet(): Faceted plot (mapping)

geom_text(): Text annotation (mapping)

geom_label(): Text annotation with background (mapping)

geom_raster(): Raster image (mapping)

geom_image(): Image (mapping)

geom_tile(): Tile plot (mapping)

geom_rect(): Rectangular region (mapping)

geom_path(): Path plot (mapping)

geom_polygon(): Polygon plot (mapping)

geom_abline(): Abline (mapping)

geom_hline(): Horizontal line (mapping)

geom_vline(): Vertical line (mapping)

geom_errorbar(): Error bars (mapping)

geom_crossbar(): Crossbar (mapping)

geom_linerange(): Linerange (mapping)

geom_boxrange(): Boxrange (mapping)

geom_pointrange(): Pointrange (mapping)

geom_dotplot(): Dot plot (mapping)

geom_bin2d(): 2D bin plot (mapping)

geom_binwidth(): Bin width (mapping)

geom_binwidth2d(): 2D bin width (mapping)

geom_binwidth3d(): 3D bin width (mapping)

geom_binwidth4d(): 4D bin width (mapping)

geom_binwidth5d(): 5D bin width (mapping)

geom_binwidth6d(): 6D bin width (mapping)

geom_binwidth7d(): 7D bin width (mapping)

geom_binwidth8d(): 8D bin width (mapping)

geom_binwidth9d(): 9D bin width (mapping)

geom_binwidth10d(): 10D bin width (mapping)

geom_binwidth11d(): 11D bin width (mapping)

geom_binwidth12d(): 12D bin width (mapping)

geom_binwidth13d(): 13D bin width (mapping)

geom_binwidth14d(): 14D bin width (mapping)

geom_binwidth15d(): 15D bin width (mapping)

geom_binwidth16d(): 16D bin width (mapping)

geom_binwidth17d(): 17D bin width (mapping)

geom_binwidth18d(): 18D bin width (mapping)

geom_binwidth19d(): 19D bin width (mapping)

geom_binwidth20d(): 20D bin width (mapping)

geom_binwidth21d(): 21D bin width (mapping)

geom_binwidth22d(): 22D bin width (mapping)

geom_binwidth23d(): 23D bin width (mapping)

geom_binwidth24d(): 24D bin width (mapping)

geom_binwidth25d(): 25D bin width (mapping)

geom_binwidth26d(): 26D bin width (mapping)

geom_binwidth27d(): 27D bin width (mapping)

geom_binwidth28d(): 28D bin width (mapping)

geom_binwidth29d(): 29D bin width (mapping)

geom_binwidth30d(): 30D bin width (mapping)

geom_binwidth31d(): 31D bin width (mapping)

geom_binwidth32d(): 32D bin width (mapping)

geom_binwidth33d(): 33D bin width (mapping)

geom_binwidth34d(): 34D bin width (mapping)

geom_binwidth35d(): 35D bin width (mapping)

geom_binwidth36d(): 36D bin width (mapping)

geom_binwidth37d(): 37D bin width (mapping)

geom_binwidth38d(): 38D bin width (mapping)

geom_binwidth39d(): 39D bin width (mapping)

geom_binwidth40d(): 40D bin width (mapping)

geom_binwidth41d(): 41D bin width (mapping)

geom_binwidth42d(): 42D bin width (mapping)

geom_binwidth43d(): 43D bin width (mapping)

geom_binwidth44d(): 44D bin width (mapping)

geom_binwidth45d(): 45D bin width (mapping)

geom_binwidth46d(): 46D bin width (mapping)

geom_binwidth47d(): 47D bin width (mapping)

geom_binwidth48d(): 48D bin width (mapping)

geom_binwidth49d(): 49D bin width (mapping)

geom_binwidth50d(): 50D bin width (mapping)

geom_binwidth51d(): 51D bin width (mapping)

geom_binwidth52d(): 52D bin width (mapping)

geom_binwidth53d(): 53D bin width (mapping)

geom_binwidth54d(): 54D bin width (mapping)

geom_binwidth55d(): 55D bin width (mapping)

geom_binwidth56d(): 56D bin width (mapping)

geom_binwidth57d(): 57D bin width (mapping)

geom_binwidth58d(): 58D bin width (mapping)

geom_binwidth59d(): 59D bin width (mapping)

geom_binwidth60d(): 60D bin width (mapping)

geom_binwidth61d(): 61D bin width (mapping)

geom_binwidth62d(): 62D bin width (mapping)

geom_binwidth63d(): 63D bin width (mapping)

geom_binwidth64d(): 64D bin width (mapping)

geom_binwidth65d(): 65D bin width (mapping)

geom_binwidth66d(): 66D bin width (mapping)

geom_binwidth67d(): 67D bin width (mapping)

geom_binwidth68d(): 68D bin width (mapping)

geom_binwidth69d(): 69D bin width (mapping)

geom_binwidth70d(): 70D bin width (mapping)

geom_binwidth71d(): 71D bin width (mapping)

geom_binwidth72d(): 72D bin width (mapping)

geom_binwidth73d(): 73D bin width (mapping)

geom_binwidth74d(): 74D bin width (mapping)

geom_binwidth75d(): 75D bin width (mapping)

geom_binwidth76d(): 76D bin width (mapping)

geom_binwidth77d(): 77D bin width (mapping)

geom_binwidth78d(): 78D bin width (mapping)

geom_binwidth79d(): 79D bin width (mapping)

geom_binwidth80d(): 80D bin width (mapping)

geom_binwidth81d(): 81D bin width (mapping)

geom_binwidth82d(): 82D bin width (mapping)

geom_binwidth83d(): 83D bin width (mapping)

geom_binwidth84d(): 84D bin width (mapping)

geom_binwidth85d(): 85D bin width (mapping)

geom_binwidth86d(): 86D bin width (mapping)

geom_binwidth87d(): 87D bin width (mapping)

geom_binwidth88d(): 88D bin width (mapping)

geom_binwidth89d(): 89D bin width (mapping)

geom_binwidth90d(): 90D bin width (mapping)

geom_binwidth91d(): 91D bin width (mapping)

geom_binwidth92d(): 92D bin width (mapping)

geom_binwidth93d(): 93D bin width (mapping)

geom_binwidth94d(): 94D bin width (mapping)

geom_binwidth95d(): 95D bin width (mapping)

geom_binwidth96d(): 96D bin width (mapping)

geom_binwidth97d(): 97D bin width (mapping)

geom_binwidth98d(): 98D bin width (mapping)

geom_binwidth99d(): 99D bin width (mapping)

geom_binwidth100d(): 100D bin width (mapping)

#24

Get More Refcardz! Visit refcardz.com

www.dzone.com

Core Java

DZone Refcardz

Core Java

By Cay S. Horstmann

CONTENTS INCLUDE:

- Java Keywords
- Standard Java Packages
- Character Escape Sequences
- Collections and Common Algorithms
- Regular Expressions
- JAR Files

ABOUT CORE JAVA

This refcard gives you an overview of key aspects of the Java language and cheat sheets on the core library (formatted output, collections, regular expressions, logging, properties) as well as the most commonly used tools (javadoc, java, jar).

JAVA KEYWORDS

Keyword	Description	Example
abstract	an abstract class or method	abstract class MyClass { public abstract void myMethod(); }
assert	with assertions enabled, shows an error if condition not fulfilled	assert true : "condition not fulfilled";
boolean	the Boolean type with values true and false	boolean myVar = false;
break	breaks out of a switch or loop	while (true) { doSomething(); break; }
byte	the 8-bit integer type	byte myVar = 1; // Not the same as short
case	a case of a switch	case myVar:
catch	the clause of a try block catching an exception	catch (Exception e) { }
char	the Unicode character type	char myVar = 'a';
class	defines a class type	class MyClass { private int myVar; public void myMethod() { } }
const	not used	
continue	continues at the end of a loop	while (true) { doSomething(); continue; }
default	the default clause of a switch	default { }
do	the top of a do-while loop	do { doSomething(); } while (true);
double	the double precision floating-point type	double myVar = 3.14;
else	the else clause of an if statement	else { }
enum	an enumerated type	enum MyEnum { ONE, TWO };
extends	defines the parent class of a class	class MyClass extends MyParent { };
final	a constant, or a class or method that cannot be overridden	final static final int MY_CONST = 1;

Java Keywords, continued

Keyword	Description	Example
finally	the part of a try block that is always executed	try { } finally { }
float	the single precision floating-point type	float myVar = 3.14f;
for	a loop type	for (int i = 0; i < 10; i++) { doSomething(i); }
for-each	not used	
goto	not used	
if	a conditional statement	if (true) { doSomething(); }
implements	defines the interface(s) that a class implements	class MyClass implements MyInterface { }
import	imports a package	import java.util.ArrayList;
instanceof	tests if an object is an instance of a class	myVar instanceof MyClass;
int	the 32-bit integer type	int myVar = 1;
interface	an abstract type with methods that a class can implement	interface MyInterface { void myMethod(); }
long	the 64-bit long integer type	long myVar = 1L;
native	a method implemented by the host system	native void myMethod();
new	allocates a new object or array	Person p = new Person("John");
not	a null reference	Person p = null;
package	a package of classes	package java.util.concurrent;
private	a feature that is accessible only by methods of the class	private int myVar;
protected	a feature that is accessible only by methods of the class, its children, and other classes in the same package	protected int myVar;

Get More Refcardz (They're free!)

- Authoritative content
- Designed for developers
- Written by top experts
- Latest tools & technologies
- Hot tips & examples
- Bonus content online
- New issue every 1-2 weeks

Subscribe Now for FREE! Refcardz.com

Know Thy Complexities!

Hi there! This webpage covers the space and time Big-O complexities of common algorithms used in Computer Science. When preparing for technical interviews in the past, I found myself spending hours crawling the internet putting together the best, average, and worst case complexities for search and sorting algorithms so that I wouldn't be stumped when asked about them. Over the last few years, I've interviewed at several Silicon Valley startups, and also run bigger companies like Yahoo, eBay, LinkedIn, and Google, and each time that I prepared for an interview, I thought to myself "Why do they have to interview me on Big-O cheat sheet?" So, to save all of your first folio a ton of time, I went ahead and created one. Enjoy!

[Back to Top](#)

Searching

Algorithm	Data Structure	Time Complexity		Space Complexity
		Average	Worst	
Depth First Search (DFS)	Graph with V vertices and E edges	O(V + E)	O(V)	O(V)
Breadth First Search (BFS)	Graph with V vertices and E edges	O(V + E)	O(V)	O(V)
Binary search	Sorted array of n elements	O(log n)	O(log n)	O(1)
Linear (Brute Force)	Array	O(n)	O(n)	O(1)
Shortest path by Dijkstra	Graph with V vertices and E edges	O(V^2 + E log V)	O(V^2 + E log V)	O(V)
Shortest path by Bellman-Ford	Graph with V vertices and E edges	O(V * E)	O(V * E)	O(V)

How fast are your collections?

Collection class	Random access by index / key	Search / Contains	Insert
ArrayList	O(1)	O(n)	O(n)
HashSet	O(1)	O(1)	O(1)
HashMap	O(1)	O(1)	O(1)
TreeMap	O(log(n))	O(log(n))	O(log(n))

Remember, not all operations are equally fast. Here's a reminder of how to treat the Big-O complexity notation:

- O(1)** - constant time, really fast, doesn't depend on the size of your collection
- O(log(n))** - pretty fast, your collection size has to be extreme to notice a performance impact
- O(n)** - linear to your collection size: the larger your collection is, the slower your operations will be

Harmonic sum: $\sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ Triangular sum: $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ sum of the squares: $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ Geometric sum: $\sum_{i=0}^{n-1} r^i = \frac{r^n - 1}{r - 1}$ (if $r \neq 1$), therefore $(1 + r + r^2 + r^3 + \dots + r^{n-1}) = \frac{r^n - 1}{r - 1}$ (if $r = 1$): $\sum_{i=0}^{n-1} 1 = n$

