

Hashes

Sofia Robb

1

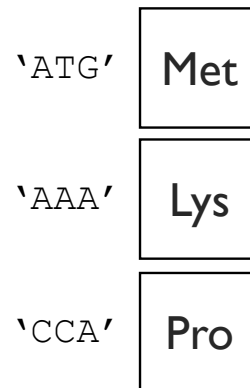
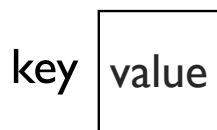
Hashes

- Perl hashes are denoted with a '%' symbol like this
%data
- Each key and each value contains a scalar value for example this could be
 - a number
 - a letter
 - a word
 - a sentence
 - a scalar variable like \$scalar_variable
 - a gene ID
 - a sequence

2

What is a hash?

- A hash is an associative array made up of key/value pairs.
- Like a dictionary
- And unlike an array a hash is unordered.



3

A key is like a descriptive array index.

An array

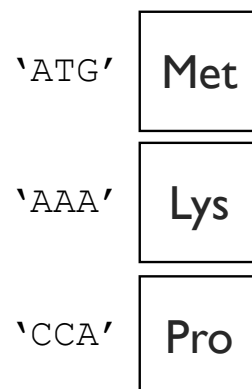


The array index [0] is similar to the key 'ATG'.

The key 'ATG' is used to access the value 'Met', just as [0] is used to access 'red'

But the key/value pairs are not stored in order

A hash



4

Creating a hash

The hash %genetic_code is built with key/value pairs

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);
```

key	value
'ATG'	Met

5

Accessing a hash value using a key

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);  
  
my $aa = $genetic_code{'ATG'};  
print "ATG translates to $aa\n";  
ATG translates to Met
```

Each value of the hash is a scalar therefore we use the '\$' when we refer to an individual value.

Hash keys are surrounded by squiggly brackets {}

6

keys() returns an unordered list of the keys of a hash

```
@array_of_keys = keys (%hash);

my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro',
);

my @codons = keys (%genetic_code);
print join('--', @codons), "\n";
CCA--AAA--ATG
```

7

Iterating through a hash by looping through an list of hash keys.

```
my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro',
);

foreach my $codon (keys %genetic_code) {
    my $aa = $genetic_code{$codon};
    print "$codon translates to $aa\n";
}
CCA translates to Pro
AAA translates to Lys
ATG translates to Met
```

Remember: the key is used to access
the value
\$value = \$hash{\$key}

8

Sorting and iterating through the keys of a hash

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);
```

Remember: hash keys are unordered so we use `sort` to be sure that the order is always the same.

```
foreach my $codon (sort keys %genetic_code) {  
  my $aa = $genetic_code{$codon};  
  print "$codon translates to $aa\n";  
}  
AAA translates to Lys  
ATG translates to Met  
CCA translates to Pro
```

9

Iterating through a hash and sorting by the values

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);
```

Remember: the key is used to access the value
`$value = $hash{$key}`

```
foreach my $codon (sort {$genetic_code{$a} cmp $genetic_code{$b}}  
  keys %genetic_code) {  
  my $aa = $genetic_code{$codon};  
  print "$codon translates to $aa\n";  
}  
AAA translates to Lys  
ATG translates to Met  
CCA translates to Pro
```

we can create a custom
sort function using `{ $a cmp $b }`

10

values() returns an unordered list of values

```
@array_of_values = values(%hash);
```

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);
```

You can use `sort values` to be sure that the order of the values is always the same.

```
my @amino_acids = values(%genetic_code);  
print join('--', @amino_acids), "\n";  
Pro--Lys--Met
```

11

Adding additional key/value pairs

```
my %genetic_code = (  
  'ATG' => 'Met',  
  'AAA' => 'Lys',  
  'CCA' => 'Pro',  
);
```

```
$genetic_code{'TGT'} = 'Cys';
```

```
foreach my $codon (keys %genetic_code){  
  print "$codon -- $genetic_code{$codon}\n";  
}  
CCA -- Pro  
AAA -- Lys  
ATG -- Met  
TGT -- Cys
```

12

Deleting key/value pairs

```
my %genetic_code = (  
    'ATG' => 'Met',  
    'AAA' => 'Lys',  
    'CCA' => 'Pro',  
);  
  
delete $genetic_code{'AAA'};  
  
foreach my $codon (keys %genetic_code){  
    print "$codon -- $genetic_code{$codon}\n";  
}  
  
CCA -- Pro  
ATG -- Met
```

13

Use exists() to test if a key exists.

```
my %genetic_code = (  
    'ATG' => 'Met',  
    'AAA' => 'Lys',  
    'CCA' => 'Pro',  
);
```

key exists?	return value
yes	1
no	'' empty string is false

```
my $codon = 'ATG';  
if (exists $genetic_code{$codon}){  
    print "$codon -- $genetic_code{$codon}\n";  
}else{  
    print "key: $codon does not exist\n";  
}  
  
ATG -- Met  
##when $codon='TTT', code prints "key: TTT does not exist"
```

14

Using hashes for keeping count

```
my $seq = "ATGGGCGTATGCAATT";
my @nucs = split "", $seq;
print "@nucs\n";
#A T G G G C G T A T G C A A T T

my %nt_count;
foreach my $nt (@nucs){
    $nt_count{$nt}++;
}

foreach my $nt (keys %nt_count){
    my $count = $nt_count{$nt};
    print "$nt\t$count\n";
}

A      4
T      5
C      2
G      5
```

A lot happens here: \$hash{key}++;

If a key/value does not exist and perl sees it in a script, it creates the key/value pair and sets the value to undef.

If we add 1 to undef with ++, the resulting value will be 1.

This is equivalent to perl code
\$hash{\$key} = undef;
\$value = \$hash{\$key};
\$hash{\$key} = \$value + 1;
\$hash{\$key} is now 1

If a key exists and its value is a number the value will be incremented by 1.

15

Creating a hash from variable input like data from a file

```
my $file = shift;
open (INFILE, '<', $file)
    or die "can't open file $file $!\n";
my %hash;
while (my $line = <INFILE>){
    chomp $line;
    my ($key, $value) = split /\t/, $line;
    $hash{$key} = $value;
}
foreach my $key (sort keys %hash){
    my $value = $hash{$key};
    print "key:$key value:$value\n";
}
```

16