R Notebook example: Hello World

What is a library?

```
# Cntrl-Alt-I is your friend

library(parallel);          # install.packages("parallel", dependencies=TRUE);
parallel::detectCores();
```

```
## [1] 8
```

```
# Cntrl-Shift-C is also your friend
```

Can you find another library that counts cores? Is it a core or a thread? My laptop is a QUAD-core.

Does Python Work?

```
library(reticulate);        # install.packages("reticulate", dependencies=TRUE);
use_python("C:/Python/Python39")
```

Pay attention to detail. This will only work if all the building blocks are in place.

```
print("hello world");
```

```
## hello world
```

```
def pownum(base, pow):
  return base ** pow

print(pownum(9,5));
```

```
## 59049
```

Does C++ Work?

```
library(Rcpp);        # install.packages("Rcpp", dependencies=TRUE);
# https://stackoverflow.com/questions/64839024/
```

Bits and Such

```
Rcpp::cppFunction("long long RShift(long long a, int b) { return a >> b;}")
```

Shifty shifting

The “right shift” operator in R is based on S+ and has some limitations when it comes to signed (negative) integers.

```
y = 1732584194;
RShift(y, 16);
```

```
## [1] 26437
```

```
bitwShiftR(y, 16);
```

```
## [1] 26437
```

```
y = -1732584194;
RShift(y, 16);
```

```
## [1] -26438
```

```
bitwShiftR(y, 16);
```

```
## [1] 39098
```

Libraries ‘bit’ and ‘bit64’ may be of some benefit, but beforewarned when working with bits using R.

Convert decimal number to a binary string (and vice versa)

```
dec2bin = decbin = function(decnum)
    {
    bvect = rep(0, 1 + floor(log(decnum, 2))); # pre-populate with zeroes
    while (decnum >= 2)
        {
        power = floor(log(decnum, 2));
        bvect[1 + power] = 1;
        decnum = decnum - 2^power;
        }
    bvect[1] = decnum %% 2;
    paste(rev(bvect), collapse = ""); # convert to a string
    }
```

```
decbin(57);
```

```
## [1] "111001"
```

If you write a function, you should also have its inverse.

```
bin2dec = bindec = function(binstr)
  {
  n = strlen(binstr);
    res = 0; power = 0;
    for(i in n:1)
        {
        bit = as.integer(charAt(binstr,i));
        add = 0;
        if(bit == 1) { add = 2^power; }

        res = res + add;
        power = 1 + power;
        }
    res;
  }
```

```
## bin2dec('111001');  # you may want to comment this out when you Knit-HTML as it may throw an "intent
```

```
strlen = function(str)
  {
  # history :: # https://en.cppreference.com/w/c/string/byte/strlen
  # http://www.cplusplus.com/reference/cstring/
  # https://en.wikipedia.org/wiki/C99
  # https://www.programiz.com/c-programming/library-function/string.h/strlen
  # vectorized ... already
  nchar( as.character(str), type="chars");
  }

charAt = function(str,idx)
  {
  substr(str,idx,idx);
  }
```

```
bin2dec('111001');
```

```
## [1] 57
```

```
bindec('111001');
```

```
## [1] 57
```

```
bindec( decbin(57) );
```

```
## [1] 57
```

```
decbin( bindec('111001') );
```

```
## [1] "111001"
```

```
typeof( 57 );
```

```
## [1] "double"
```

```
typeof( decbin(57) );
```

```
## [1] "character"
```

You could left-side 'strPadLeft' with zeroes if you wanted it to be a certain bit length

```
strPadLeft = function(str, final.str.len, padding="0", method="stringi")
  {
  if( isTRUE(requireNamespace("stringi", quietly = TRUE)) && method=="stringi" )
    {
    stringi::stri_pad_left(str, final.str.len, pad = padding);
    } else {
            n = strlen(str);
            r = final.str.len - n;
            if(r < 0) { stop("strPadLeft is too short!"); }
            paste0(paste(rep(padding,r),collapse=""),str);

            }
  }
```

```
strPadLeft( decbin(57), 8);
```

```
## [1] "00111001"
```

```
strPadLeft( decbin(57), 8, method="base");
```

```
## [1] "00111001"
```

```
strPadLeft( decbin(57), 8, method="Adljblkjadlk");
```

```
## [1] "00111001"
```

Benchmarking speed

```
library(microbenchmark);
microbenchmark(strPadLeft( decbin(57), 8) , strPadLeft( decbin(57), 8, method="base"), strPadLeft( decb
```

```
## Unit: microseconds
##                                               expr   min     lq    mean median
##                             strPadLeft(decbin(57), 8) 105.7 108.50 121.169 111.05
##           strPadLeft(decbin(57), 8, method = "base")  95.5  97.95 106.089 101.85
##   strPadLeft(decbin(57), 8, method = "Adljblkjadlk")  95.6  97.30 107.286  99.20
##      uq   max neval cld
##  114.75 420.3   100   b
##  106.45 216.7   100  a
##  105.90 251.5   100  a
```

We are also benchmarking the ‘decbin’ function which likely can also be improved upon. It is a good idea to isolate what you are actually timing, but testing in context is not a bad idea.

Since the library ‘stringi’ is written in C++, it has some native efficiencies over the R interpreted ‘base’ solution. [https://cran.r-project.org/web/packages/stringi/index.html]

Matrices with External C++ file

This will source and compile the code. Maybe give it a minute.

```
sourceCpp("multiply.cpp");
```

```
## Registered S3 methods overwritten by 'RcppEigen':
##   method               from
##   predict.fastLm       RcppArmadillo
##   print.fastLm         RcppArmadillo
##   summary.fastLm       RcppArmadillo
##   print.summary.fastLm RcppArmadillo
```

```
A = matrix(rnorm(10000), 100, 100); # fully populated, 100 x 100, relatively small
B = matrix(rnorm(10000), 100, 100);
```

```
library(microbenchmark);
microbenchmark(eigenMatTrans(A),A%*%B, armaMatMult(A, B), eigenMatMult(A, B), eigenMapMatMult(A, B))
```

```
## Unit: microseconds
##                  expr    min      lq     mean  median      uq     max neval
##      eigenMatTrans(A)   79.8  178.70  225.743  191.40  210.90  4314.0   100
##               A %*% B 2779.5 2907.60 2993.355 2942.60 3017.20  3842.2   100
##     armaMatMult(A, B) 2819.4 2939.70 3091.217 2974.90 3076.15  6759.2   100
##    eigenMatMult(A, B)  562.1  643.95 1293.086  689.85 1247.70  9087.4   100
##  eigenMapMatMult(A, B)  468.9  594.65 2715.551  671.05 1653.50 26184.9   100
##  cld
##  a
##    c
##    c
##    b
##     c
```

It appears ‘eigen’ performs well for me; ‘arma’ is about equivalent to the built-in R multiplication.

Can we use sparse matrices and pass them into C++? Can we pass an ‘R’ sparse matrix into a C++ function call for speed purposes?

Imagery

Two powerful C/C++ tools now accessible within R.

```
library(magick); #install.packages("magick", dependencies=TRUE);
```

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, freetype, fftw, ghostscript, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fontconfig, x11
```

```
# https://cran.r-project.org/web/packages/magick/vignettes/intro.html#The_grid_package
# https://www.datanovia.com/en/blog/easy-image-processing-in-r-using-the-magick-package/
# https://www.imagemagick.org/discourse-server/viewtopic.php?t=18433
# install.packages("tesseract")
# https://github.com/ropensci/magick/issues/154
#
image_content <- function(x, ...){
  x <- image_data(x, ...)
  as.integer(x)
}

tiger <- image_read_svg('http://jeroen.github.io/images/tiger.svg', width = 350);
tiger_png <- image_convert(tiger, "png");

tiger_matrix = image_content(tiger_png);
dim(tiger_matrix);
```

```
## [1] 350 350   4
```

```
# 3D matrix
# tiger_matrix[,,1];  # x,y, z ... z is likely RGBa

tiger;
```

```r
library(tesseract); # install.packages("tesseract");

img.file = "iris-ocr-intro.png";
img = image_read( img.file );
img.txt = image_ocr(img);

cat(img.txt);
```

```
## + I. Inrropuction
```

```r
img.file = "iris-ocr.png";

img = image_read( img.file );
img.txt = image_ocr(img);

cat(img.txt);
```

```
## Geneticist to the Missouri Botanical Garden
## Professor of Botany in the Henry Shaw School of Botany of Washington University
## I. Iytropuction
##
## Asa biological phenomenon the species problem is worthy of
## serious study as an end in itself, and not as a mere corollary to
## work in some other field. It is, to be sure, a problem so funda-
## mentally important that it touches many such fields. Workers
## in any one of these are humanly prone to regard the evidence
## from that field as all important and its techniques as all suffi-
## cient (particularly if they are themselves unacquainted with
## other aspects of the problem). When, however, one takes up the
## problem, as a problem, and studies it from the diverse view-
## points of genetics, taxonomy, cytology, and biometry, he real-
## izes that he not only needs most of the existing techniques but
## that he must devise new ones as well.
```

Does Java Work?

Natural language processing requires java running under the hood.

```
library(openNLP);  # this requires rJava ... Java
library(NLP);

sentence.a = Maxent_Sent_Token_Annotator();
word.a     = Maxent_Word_Token_Annotator();
```

```
s = anna = "Happy families are all alike; every unhappy family is unhappy in its own way.";

sw.a       = annotate(s, list(sentence.a, word.a));

pos.a      = Maxent_POS_Tag_Annotator(probs=TRUE);
swpos.a    = annotate(s, list(pos.a), sw.a);
```

```
swpos.a.words = subset(swpos.a, type=="word");

(swpos.a.words);
```

```
##  id type start end features
##   2 word     1   5 POS=JJ, POS_prob=0.8770581
##   3 word     7  14 POS=NNS, POS_prob=0.9943596
##   4 word    16  18 POS=VBP, POS_prob=0.993953
##   5 word    20  22 POS=RB, POS_prob=0.4868905
##   6 word    24  28 POS=RB, POS_prob=0.8186156
##   7 word    29  29 POS=:, POS_prob=0.9326554
##   8 word    31  35 POS=DT, POS_prob=0.9445861
##   9 word    37  43 POS=JJ, POS_prob=0.9951879
##  10 word    45  50 POS=NN, POS_prob=0.9890899
##  11 word    52  53 POS=VBZ, POS_prob=0.9826753
##  12 word    55  61 POS=JJ, POS_prob=0.9860051
##  13 word    63  64 POS=IN, POS_prob=0.994442
##  14 word    66  68 POS=PRP$, POS_prob=0.9906345
##  15 word    70  72 POS=JJ, POS_prob=0.9929793
##  16 word    74  76 POS=NN, POS_prob=0.9987191
##  17 word    77  77 POS=., POS_prob=0.9947943
```

```
tags = sapply(swpos.a.words$features, `[[`, "POS");
sort(table(tags),decreasing = TRUE);
```

```
## tags
##    JJ    NN    RB     .     :    DT    IN   NNS  PRP$   VBP   VBZ
##     4     2     2     1     1     1     1     1     1     1     1
```