Basic R

Kim Wong

09/14/2021

Copy Tutorial to working directory

##{r} ##system("cp -rp /ihome/crc/training/fall2019/R_Introduction $HOME") ##

Set working directory to $HOME/R_Introduction

##{r} ##myhome <- paste0(system("echo $HOME", intern = TRUE),"/R_Introduction") ##

##{r} ##myhome ## ##{r setup} ##setwd(myhome) ##knitr::opts_knit$set(root.dir=myhome) ##

```
getwd()
```

```
## [1] "/Users/kimwong/OneDrive - University of Pittsburgh/Documents/Kim F. Wong/CRC_Workshop/2021/Shaw
```

Output R version

```
R.version.string
```

```
## [1] "R version 4.0.3 (2020-10-10)"
```

Assignment, math, and printing

Syntax for assignment of variables

```
x <- 25
y <- 75
z <- x + y
```

Output value of variable

```
x; y; z
```

```
## [1] 25
```

```
## [1] 75
```

```
## [1] 100
```

Concatenate and output

```
cat(x,y,z)
```

```
## 25 75 100
```

Concatenate and output

```
cat(x,y,z,"\n")
```

```
## 25 75 100
```

Combine values into a vector or a list

```
s <- c(x,y,z)
print(s)
```

```
## [1]  25  75 100
```

Formatting

```
sprintf("%i %i %i", x,y,z)
```

```
## [1] "25 75 100"
```

Formatting

```
y <- 3.14823423
yy <- c(x,y,z)
sprintf("%1.2f", yy)
```

```
## [1] "25.00"  "3.15"   "100.00"
```

```
yy
```

```
## [1]  25.000000   3.148234 100.000000
```

Indexing in data structure

Syntax for a Vector

```
v <- c(2,4,6,8,10,12)
v
```

```
## [1]  2  4  6  8 10 12
```

Accessing specific elements of the vector

```
v[c(1,2,3,4,5,6)]
```

```
## [1]  2  4  6  8 10 12
```

```
v[c(1,2,3)]
```

```
## [1] 2 4 6
```

```
v[c(4,5,6)]
```

```
## [1]  8 10 12
```

```
v[3:5]
```

```
## [1]  6  8 10
```

Syntax for a Matrix

```
m <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, ncol=3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
matrix(c(1:5), nrow=2, ncol=5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    2    4
## [2,]    2    4    1    3    5
```

```
matrix(c(1:5), nrow=2, ncol=5,T)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    1    2    3    4    5
```

```
m[3,2]
```

```
## [1] 6
```

Syntax for data frame

```
students <- data.frame(
  name = c("Jack", "Jill", "Emma", "Billy", "Sarah"),
  hw1 = c(87, 90, 100, 75, 88),
  hw2 = c(95, 65, 95, 85, 100),
  hw3 = c(99, 95, 89, 93, 87),
  quiz1 = c(45, 55, 65, 70, 75),
  quiz2 = c(95, 85, 75, 65, 55),
  final = c(100, 95, 90, 85, 80)
  )
```

Print out data frame

```
print(students)
```

```
##    name hw1 hw2 hw3 quiz1 quiz2 final
## 1  Jack  87  95  99    45    95   100
## 2  Jill  90  65  95    55    85    95
## 3  Emma 100  95  89    65    75    90
## 4 Billy  75  85  93    70    65    85
## 5 Sarah  88 100  87    75    55    80
```

Accessing specific element of a data frame

```
students[1,6]
```

```
## [1] 95
```

Accessing a row of a data frame

```
students[2,]
```

```
##   name hw1 hw2 hw3 quiz1 quiz2 final
## 2 Jill  90  65  95    55    85    95
```

Accessing a range of rows and columns

```
students[2:3,c(1,5:7)]
```

```
##   name quiz1 quiz2 final
## 2 Jill    55    85    95
## 3 Emma    65    75    90
```

Accessing columns by name

```
students[ ,c("name", "hw2", "final")]
```

```
##    name hw2 final
## 1  Jack  95   100
## 2  Jill  65    95
## 3  Emma  95    90
## 4 Billy  85    85
## 5 Sarah 100    80
```

```
students[ ,c(1, 3, 7)]
```

```
##    name hw2 final
## 1  Jack  95   100
## 2  Jill  65    95
## 3  Emma  95    90
## 4 Billy  85    85
## 5 Sarah 100    80
```

Simple statistics on a data frame

Output students grades on the final

```
students[ , "final"]
```

```
## [1] 100  95  90  85  80
```

mean, min, max and standard deviation of grades on the final

```
mean(students[ , "final"])
```

```
## [1] 90
```

```
min(students[ , "final"])
```

```
## [1] 80
```

```
max(students[ , "final"])
```

```
## [1] 100
```

```
sd(students[ , "final"])
```

```
## [1] 7.905694
```

Summary statistics of grades on the final

```
summary(students[ , "final"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      80      85      90      90      95     100
```

Summary statistics of specific data columns

```
summary(students[ , c("hw1", "hw2", "hw3")])
```

```
##       hw1           hw2           hw3
##  Min.   : 75   Min.   : 65   Min.   :87.0
##  1st Qu.: 87   1st Qu.: 85   1st Qu.:89.0
##  Median : 88   Median : 95   Median :93.0
##  Mean   : 88   Mean   : 88   Mean   :92.6
##  3rd Qu.: 90   3rd Qu.: 95   3rd Qu.:95.0
##  Max.   :100   Max.   :100   Max.   :99.0
```

```
summary(students[c(2:4), c("name","hw1", "hw2", "hw3")])
```

```
##      name                hw1              hw2             hw3
##  Length:3           Min.   : 75.00   Min.   :65.00   Min.   :89.00
##  Class :character   1st Qu.: 82.50   1st Qu.:75.00   1st Qu.:91.00
##  Mode  :character   Median : 90.00   Median :85.00   Median :93.00
##                     Mean   : 88.33   Mean   :81.67   Mean   :92.33
##                     3rd Qu.: 95.00   3rd Qu.:90.00   3rd Qu.:94.00
##                     Max.   :100.00   Max.   :95.00   Max.   :95.00
```

Sumary statistics of all data

```
summary(students)
```

```
##      name                hw1           hw2           hw3            quiz1
##  Length:5           Min.   : 75   Min.   : 65   Min.   :87.0   Min.   :45
##  Class :character   1st Qu.: 87   1st Qu.: 85   1st Qu.:89.0   1st Qu.:55
##  Mode  :character   Median : 88   Median : 95   Median :93.0   Median :65
##                     Mean   : 88   Mean   : 88   Mean   :92.6   Mean   :62
##                     3rd Qu.: 90   3rd Qu.: 95   3rd Qu.:95.0   3rd Qu.:70
##                     Max.   :100   Max.   :100   Max.   :99.0   Max.   :75
##      quiz2         final
##  Min.   :55   Min.   : 80
##  1st Qu.:65   1st Qu.: 85
##  Median :75   Median : 90
##  Mean   :75   Mean   : 90
##  3rd Qu.:85   3rd Qu.: 95
##  Max.   :95   Max.   :100
```

```
dim(students)
```

```
## [1] 5 7
```

Querying info about variables

Let’s assign some variables

```
x <- 2.5
n <- 8L
nn <- 8.0
lett <- LETTERS[1:8]
```

Display the structure of above variables

```
str(x)
```

```
##  num 2.5
```

```
str(n)
```

```
##  int 8
```

```
str(nn)
```

```
##  num 8
```

```
str(lett)
```

```
##  chr [1:8] "A" "B" "C" "D" "E" "F" "G" "H"
```

Query if a variable had been declared

```
exists("x")
```

```
## [1] TRUE
```

```
exists("x_does_not_exist")
```

```
## [1] FALSE
```

```
is.integer(n)
```

```
## [1] TRUE
```

```
nn <- 318
is.integer(nn)
```

```
## [1] FALSE
```

```
nnn <- as.integer(nn)
nnn
```

```
## [1] 318
```

```
is.integer(nnn)
```

```
## [1] TRUE
```

```
x_does_not_exist <- FALSE
exists("x_does_not_exist")
```

```
## [1] TRUE
```

```
str(x_does_not_exist)
```

```
##  logi FALSE
```

Let’s create a data frame an query it’s dimensions

```
n <- 1:8
data <- data.frame(n,lett)
str(data)
```

```
## 'data.frame':    8 obs. of  2 variables:
##  $ n   : int  1 2 3 4 5 6 7 8
##  $ lett: chr  "A" "B" "C" "D" ...
```

```
length(n)
```

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

```
length(data)
```

```
## [1] 2
```

```
nrow(data)
```

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

```
ncol(data)
```

```
## [1] 2
```

```
dim(data)
```

```
## [1] 8 2
```

Common Random number operations

```
runif(1)
```

```
## [1] 0.08188471
```

```
myran <- runif(1000)
```

```
myran
```

```
##    [1] 0.0939396033 0.6887612997 0.7904354446 0.2662894099 0.1126635685
##    [6] 0.3085979342 0.0300356685 0.0006814257 0.4056720480 0.2506353252
##   [11] 0.8970022791 0.1798195404 0.6933642675 0.1195384250 0.2927912676
##   [16] 0.5419974448 0.4692307485 0.3339562900 0.2326093279 0.2303950628
##   [21] 0.4964529660 0.1134061986 0.1827083989 0.2572623021 0.1744378433
##   [26] 0.3409916749 0.1304934907 0.0274510013 0.4667390659 0.1779942526
##   [31] 0.4043178277 0.0408116707 0.0016196873 0.7815013924 0.5734106053
##   [36] 0.2062076384 0.2309789397 0.5193564927 0.1899986281 0.3572534269
##   [41] 0.0606574276 0.0641767583 0.4954473274 0.8376694084 0.5008270910
##   [46] 0.6192617912 0.3505610407 0.7479679678 0.4185677951 0.0091531824
##   [51] 0.8367935861 0.2920011475 0.4105252130 0.8876228142 0.5376530292
##   [56] 0.1685461879 0.9878308149 0.2456477440 0.5398389190 0.2040065920
##   [61] 0.4175832474 0.5288562537 0.3762932494 0.3367321782 0.1562202275
##   [66] 0.4412314079 0.2818208507 0.2880764278 0.1712844770 0.1465218819
##   [71] 0.8211114639 0.7417274327 0.0092351492 0.9267060182 0.1099766067
##   [76] 0.4876398344 0.1985384708 0.3240711545 0.1507999729 0.3522913523
##   [81] 0.7905699692 0.8462194218 0.0932428874 0.6798696942 0.9295359545
##   [86] 0.2148178751 0.8584536815 0.4193238171 0.0616085283 0.2524858380
##   [91] 0.4523631874 0.5131772179 0.0297025898 0.4093762510 0.1249930225
##   [96] 0.1310563171 0.0512110589 0.8502684366 0.5162248840 0.6783264189
## [101] 0.7240392799 0.0944581542 0.3901287054 0.3977377978 0.4417677519
## [106] 0.7002278427 0.0293933756 0.9571446232 0.1608984696 0.0735077304
## [111] 0.0571450123 0.3689966463 0.1292009486 0.2363046273 0.2693697650
## [116] 0.2505443916 0.8421783436 0.4507608698 0.5639818355 0.1621095308
## [121] 0.9052218704 0.8825114036 0.8302359516 0.5522066758 0.3946724944
## [126] 0.6784351827 0.7367667428 0.8650575480 0.4281631305 0.7282596861
## [131] 0.7122018959 0.9674724157 0.8751147150 0.9249198544 0.0418673970
## [136] 0.1272940941 0.4921882793 0.0579946169 0.3179154580 0.8042020418
## [141] 0.4192736016 0.2661906378 0.2737528356 0.5083238715 0.6046003730
## [146] 0.5243869459 0.6560031821 0.7097558356 0.1132064324 0.6249045543
## [151] 0.9851560185 0.3121269371 0.5855960164 0.1599011959 0.6578157211
## [156] 0.2098437326 0.5303128380 0.9300366472 0.2547825086 0.5670482100
## [161] 0.5947123815 0.1960889997 0.9162598550 0.7664131275 0.9563715416
## [166] 0.8750615853 0.9642907549 0.8729719527 0.7188137160 0.7068746914
## [171] 0.9731797853 0.4318308642 0.3485978332 0.1960138741 0.0492198879
## [176] 0.6760303348 0.9713527975 0.7881977051 0.6152094600 0.9466453709
## [181] 0.4315876917 0.8498751293 0.1151465226 0.6022644818 0.5062186718
## [186] 0.1314295917 0.9242976585 0.8740144845 0.1704753663 0.3131191949
## [191] 0.7130466807 0.9130255294 0.7106572837 0.4154760516 0.4704510088
## [196] 0.7273630132 0.9236768584 0.1859468855 0.1146271664 0.3622033754
## [201] 0.7595087087 0.6636588403 0.2917902239 0.2729968759 0.1051604669
## [206] 0.6746892717 0.0278281812 0.3380890500 0.0565778683 0.4390625246
## [211] 0.3291829366 0.3942525256 0.3242024181 0.3105127029 0.6339950401
## [216] 0.4938497185 0.5893600730 0.9430446194 0.0848548510 0.0728157782
## [221] 0.8679928069 0.5505903433 0.0510169014 0.9526126802 0.6104521398
## [226] 0.8608757046 0.8994144662 0.4064578416 0.9942150817 0.4088431683
## [231] 0.6379771899 0.3909900279 0.9269206540 0.2116173462 0.7308186733
## [236] 0.2657646565 0.0847509678 0.8058869988 0.5885290194 0.3977771394
## [241] 0.8021638317 0.4955203475 0.4593897988 0.7491057683 0.5637884103
## [246] 0.6151058453 0.7805080111 0.6985910160 0.7608432153 0.1276376592
## [251] 0.8513139852 0.5927405602 0.1143288815 0.9263595880 0.4324243821
## [256] 0.1457906137 0.5227505455 0.6065141517 0.3811914003 0.3677038078
## [261] 0.5449949396 0.3156406072 0.8041084951 0.6282298360 0.6227660549
## [266] 0.9523605243 0.7305070062 0.6618593137 0.4995846606 0.5393364641
## [271] 0.1128377323 0.0276234741 0.4530189638 0.9239236575 0.8358192674
## [276] 0.8352026294 0.9871501122 0.0081328140 0.0914094602 0.6699576867
## [281] 0.8078024460 0.3972794504 0.8328867801 0.4116974769 0.4486621257
## [286] 0.9211081362 0.1826206266 0.5902642461 0.2252716720 0.0081792863
## [291] 0.8636689333 0.9120381554 0.8199928477 0.5768997597 0.0875723641
## [296] 0.6069233930 0.2531228254 0.4256793950 0.1910996833 0.9547763790
## [301] 0.1858492685 0.1510747925 0.8275457735 0.8837430482 0.1650905630
## [306] 0.7782458146 0.2467099947 0.4785394599 0.0236855850 0.5913262533
## [311] 0.5339027031 0.7122984976 0.9533932568 0.7530592340 0.3605533324
## [316] 0.8152636627 0.7971641649 0.4908908308 0.4032504354 0.1999610239
## [321] 0.4499692204 0.2792372666 0.2542374239 0.6984789756 0.3620593853
## [326] 0.1317281253 0.0498507931 0.2363721908 0.2143257782 0.4574350514
## [331] 0.8056357517 0.5766861276 0.5658481575 0.8716982778 0.7262669371
## [336] 0.6694423519 0.1126396144 0.0402357623 0.1618374421 0.9734283483
## [341] 0.8073040275 0.5482659214 0.3797487309 0.8396696993 0.4415042463
## [346] 0.4425894860 0.3885735667 0.1894174987 0.8597254693 0.0322208509
## [351] 0.6008309950 0.1888212860 0.6010056839 0.7122806706 0.5799399884
## [356] 0.8959771560 0.5585605493 0.9371140748 0.5539802066 0.2345514384
## [361] 0.3851819485 0.9950272485 0.3163384574 0.7129361255 0.8537188782
## [366] 0.9814413085 0.1787470833 0.3881157073 0.4111930928 0.8697254027
## [371] 0.8282887468 0.8869546235 0.1373199010 0.7772532939 0.8690004230
## [376] 0.7029355878 0.2276328485 0.7618685309 0.8633816745 0.8254281993
## [381] 0.0166466148 0.2104359698 0.1528882813 0.3715699648 0.3015022103
## [386] 0.6994201725 0.1104182296 0.1432739245 0.4088383971 0.4347232280
## [391] 0.9495341405 0.6355366709 0.8565539229 0.8173326517 0.9521128915
## [396] 0.5068815500 0.1151376436 0.6218556287 0.2697772314 0.3254893741
## [401] 0.6481565482 0.3246125302 0.8512990072 0.5687778946 0.1540791255
## [406] 0.0689707678 0.1246748308 0.7343785118 0.1941006200 0.3032337672
## [411] 0.7385323781 0.9585880733 0.5291017613 0.6873103620 0.8427289657
## [416] 0.8804281191 0.3193529537 0.8918924937 0.5249941617 0.6847485462
## [421] 0.2970973363 0.5549520738 0.2036135236 0.5378630976 0.6298954138
## [426] 0.4291095173 0.2876311520 0.4527470809 0.9501362843 0.2020953693
## [431] 0.3632850999 0.9095534740 0.0656634243 0.9188869018 0.6254414176
## [436] 0.5318466898 0.1684995084 0.3593497486 0.4853309842 0.7839837386
## [441] 0.7667842458 0.0384463787 0.0038180815 0.9166225290 0.7556032734
## [446] 0.3146485849 0.7821884819 0.3751784321 0.2610662151 0.8350441188
## [451] 0.9936587322 0.0701960982 0.1489391387 0.8102566788 0.5775985769
## [456] 0.4636351359 0.5932476833 0.8616224395 0.8574518720 0.4584663184
## [461] 0.6289093622 0.0918791543 0.9953272203 0.7788172944 0.6212603711
## [466] 0.3987439752 0.0996788174 0.2852032892 0.3887600810 0.5095417679
## [471] 0.5700287805 0.3029384774 0.7210637359 0.3100208254 0.5013730377
## [476] 0.9303807705 0.4079083996 0.8213013711 0.7715674636 0.0622309993
## [481] 0.8780056231 0.2205900035 0.8463650013 0.8262163443 0.7094917379
## [486] 0.3277616489 0.9288132109 0.7788783824 0.8229646734 0.8151806032
## [491] 0.5575996242 0.7024966942 0.0802100380 0.4776709764 0.9663260118
## [496] 0.7215877911 0.1710534217 0.6688558075 0.6051018988 0.9218857463
## [501] 0.9471766609 0.6373148169 0.2725593636 0.4619051386 0.7413485565
## [506] 0.6132865979 0.9518363907 0.7943121411 0.3777683680 0.2342043347
## [511] 0.1050136539 0.2523824682 0.6682255010 0.3622859297 0.3210269383
## [516] 0.5758519564 0.2579091880 0.0948912634 0.2543606441 0.5601754826
## [521] 0.9359906560 0.8812102699 0.3447185578 0.0725118739 0.8923205347
## [526] 0.9703294877 0.0073125937 0.8620138124 0.8039370703 0.0866769871
## [531] 0.9845227792 0.8271301484 0.3713492923 0.2471085752 0.2356728569
## [536] 0.1995827269 0.8540458495 0.0416687869 0.1227259224 0.7127642236
## [541] 0.6207654092 0.2400910691 0.5330834906 0.0784988347 0.5954180541
## [546] 0.0876474935 0.8305340891 0.5628913715 0.5771221991 0.5753787961
## [551] 0.5288124366 0.8544167161 0.6937922742 0.2070398729 0.5237135219
## [556] 0.7874709426 0.8576842661 0.7204935411 0.9173604005 0.7972109169
## [561] 0.5219674555 0.9468243909 0.7203149055 0.9104569354 0.1360897131
## [566] 0.9804209508 0.7750016144 0.9526105782 0.6687164695 0.6394378375
## [571] 0.6318906299 0.5082113589 0.4811396545 0.1456183656 0.7898935752
## [576] 0.4541158397 0.5767792817 0.7592312004 0.4839500505 0.0464876799
## [581] 0.0989201050 0.6691947135 0.9507304626 0.5701354244 0.5086325521
## [586] 0.6527932172 0.4845950850 0.5592328236 0.1183170604 0.7690249372
## [591] 0.4406874748 0.2381988717 0.1386247054 0.5724454345 0.3405113858
## [596] 0.9052607862 0.0010074659 0.5633633686 0.4071637790 0.4867745566
## [601] 0.0267022287 0.7103525188 0.1281707517 0.5091522706 0.7208954773
## [606] 0.4767394923 0.9555777679 0.8099316531 0.8388434967 0.2824980482
## [611] 0.2462748804 0.7284906188 0.8681443571 0.7819446940 0.1637790133
## [616] 0.2643605703 0.1424979635 0.3334381306 0.2684773735 0.3295614794
## [621] 0.0507752737 0.2618974615 0.7107852476 0.9345208285 0.1977829090
## [626] 0.5465119942 0.2005477839 0.2439325654 0.1051890135 0.8474595156
## [631] 0.9354540617 0.2969604977 0.8827196264 0.4558543274 0.7609972779
## [636] 0.4783861963 0.3022558966 0.2405947833 0.0691370422 0.5348300121
## [641] 0.4244383399 0.3813616568 0.9697531299 0.5308033386 0.0306312072
## [646] 0.0968982975 0.5918248666 0.2889306669 0.5996025270 0.1838641921
## [651] 0.9027863098 0.7250002257 0.3332943982 0.1200625622 0.5138174493
## [656] 0.5700890715 0.9447017552 0.5747275960 0.2409965899 0.7277164145
## [661] 0.3168959967 0.8131377141 0.2117898827 0.0173676386 0.2105980648
## [666] 0.5389695188 0.6541364931 0.0617953052 0.2231620373 0.0094109613
## [671] 0.7276920751 0.4456724098 0.1029080329 0.4636249801 0.7099005610
## [676] 0.9574138555 0.2845544503 0.6702586967 0.2843963867 0.8825761098
## [681] 0.5342342274 0.4358388542 0.4506942621 0.8924161557 0.3004285377
## [686] 0.9496332372 0.3513135260 0.5317726731 0.4204419658 0.3372011804
## [691] 0.7808357095 0.2651869142 0.4106293241 0.3016873223 0.1292139248
## [696] 0.6709047733 0.2013889293 0.9451150643 0.7323305758 0.1413885858
## [701] 0.0425382429 0.7478681684 0.7280332872 0.4512268123 0.1460120003
## [706] 0.8014621546 0.3273198719 0.0419591896 0.1866993387 0.5787103046
## [711] 0.8081717214 0.9188636816 0.5120708439 0.9819374643 0.1037902699
## [716] 0.2168108402 0.5696962432 0.2819458833 0.8083244048 0.1609012224
## [721] 0.4935523893 0.7467824866 0.2245035686 0.0177179568 0.8888427687
## [726] 0.2150254129 0.7949995771 0.8405738412 0.8009228914 0.6932760938
## [731] 0.5621088212 0.6382215696 0.6640861132 0.7344289471 0.1894223343
## [736] 0.4168089919 0.7127358303 0.2204006556 0.7449998471 0.5663393226
## [741] 0.1384022331 0.8704200075 0.8949561871 0.8699559642 0.6324288750
## [746] 0.5904259300 0.0476153463 0.8299491056 0.2200349723 0.0262380932
## [751] 0.2733142399 0.0845364449 0.8864313662 0.4754110675 0.8075422375
## [756] 0.6263914574 0.5964190590 0.0479976204 0.2207510164 0.9550261553
## [761] 0.3169635478 0.1391071745 0.7436988212 0.3570265642 0.9477141569
## [766] 0.4411807265 0.4800106166 0.4890855816 0.1830398480 0.6985354298
## [771] 0.2950825710 0.9960725212 0.1480637295 0.3653361606 0.5299609592
## [776] 0.9619261706 0.8735123281 0.4898209882 0.4474457016 0.1949630214
## [781] 0.8189226305 0.6836423492 0.7002603502 0.8642986435 0.8505905527
## [786] 0.2970663293 0.1455728747 0.1823106927 0.9068188588 0.2864364220
## [791] 0.7249289108 0.4012571659 0.2382060583 0.4986142577 0.6449698545
## [796] 0.6397635748 0.0806049528 0.1420891106 0.5214008214 0.2340214897
## [801] 0.3999316329 0.8778799535 0.3415041375 0.9881520246 0.1444446174
## [806] 0.4965304495 0.7953965024 0.8561103391 0.7108748732 0.1759888935
## [811] 0.3209098887 0.6154427039 0.2780166811 0.9234309194 0.3245870657
## [816] 0.8905440904 0.3436728125 0.9947610325 0.8203775093 0.6777429590
## [821] 0.8153378821 0.2508745885 0.0592991055 0.3192721675 0.1273666404
## [826] 0.5052648415 0.1759678253 0.1487313835 0.3787497531 0.7582544705
## [831] 0.7695261647 0.8954064879 0.8183993525 0.3840011938 0.0037005737
## [836] 0.8572290749 0.2560783490 0.9635445122 0.1865659615 0.5321896907
## [841] 0.5979809482 0.1979588179 0.3829795618 0.9691257561 0.3242124289
## [846] 0.4961962805 0.1043679679 0.5796028317 0.9434452627 0.2014378353
## [851] 0.4701293753 0.8081145138 0.1323210327 0.1136213858 0.2091358732
## [856] 0.5740617590 0.5567068732 0.6807636416 0.4034429933 0.6522797206
## [861] 0.7006213868 0.4202909654 0.1270391317 0.6300017522 0.7518110103
## [866] 0.9723612075 0.0242626686 0.8699749417 0.9122037634 0.7659457123
## [871] 0.5400692134 0.2123904601 0.0342588550 0.7134878722 0.9313189129
## [876] 0.6708769610 0.2218053278 0.6238653923 0.0112383272 0.3989391518
## [881] 0.8101062854 0.6372878382 0.6249535910 0.2986421192 0.6641016915
## [886] 0.8191501105 0.5379519034 0.9325824871 0.4563769721 0.0156537804
## [891] 0.0834021312 0.8024539661 0.0065339028 0.1326010230 0.9981162043
## [896] 0.4135859297 0.3144617646 0.9833264691 0.8768944216 0.5512043308
## [901] 0.5032519421 0.8584151638 0.9922506995 0.8028534043 0.8320940097
## [906] 0.4153308964 0.3946747270 0.8531512925 0.0865751565 0.7435412675
##  [911] 0.2501830922 0.6909037232 0.0744245732 0.5445142814 0.3515719897
##  [916] 0.6937548551 0.7321377546 0.9223821079 0.0099603967 0.1058860379
##  [921] 0.7268217276 0.3825368646 0.0812132224 0.6386850646 0.0583218448
##  [926] 0.5916138815 0.8250324798 0.2556826256 0.6732580625 0.1132064972
##  [931] 0.6276124327 0.7552696625 0.3787150288 0.0697615957 0.5423344267
##  [936] 0.6859754603 0.8451726120 0.2453146342 0.1186391974 0.7496245892
##  [941] 0.3723123106 0.9838470907 0.3253749751 0.3986697556 0.7490624078
##  [946] 0.4229237877 0.9090606852 0.7326178350 0.3545721627 0.9840935848
##  [951] 0.5483897144 0.3796562166 0.1608531892 0.4695158175 0.9475602284
##  [956] 0.6047957290 0.8992182810 0.6414698041 0.1678473547 0.4113675768
##  [961] 0.8835114161 0.2546775511 0.6274990018 0.1906007542 0.9685770199
##  [966] 0.7607104187 0.6756128550 0.8094675282 0.0860794226 0.3002017606
##  [971] 0.5771271118 0.2664428991 0.3548859176 0.9827866594 0.6752260800
##  [976] 0.1300939864 0.1547371736 0.1383804288 0.7880421123 0.0194642181
##  [981] 0.7959278622 0.0234965875 0.3503060876 0.4501094946 0.4247067757
##  [986] 0.0866985542 0.8785540764 0.7082982005 0.6042240483 0.0510163805
##  [991] 0.7553643982 0.9211400917 0.9267345385 0.8411212550 0.1693175244
##  [996] 0.0499794714 0.7035267102 0.1251217581 0.1466732726 0.1227837936
```

```
runif(10, min=25, max=50)
```

```
##  [1] 41.15693 33.56195 29.29964 26.02539 41.05238 41.18279 34.07536 28.20301
##  [9] 31.61582 32.85715
```

```
sample(myran, 25, replace=TRUE)
```

```
##  [1] 0.68076364 0.89941447 0.38119140 0.77881729 0.25054439 0.70022784
##  [7] 0.67325806 0.67832642 0.72158779 0.98144131 0.20400659 0.39099003
## [13] 0.04997947 0.53983892 0.06566342 0.35725343 0.99365873 0.40884317
## [19] 0.85768427 0.97236121 0.10588604 0.79056997 0.78050801 0.55670687
## [25] 0.40125717
```

```
sample(1:11, 10, replace=FALSE)
```

```
##  [1]  8  1 11  3  4  9  5  2  6 10
```

```
rnorm(10)
```

```
##  [1]  1.37715563 -1.53183978 -0.78516780 -0.88798202  0.05247604  3.30244195
##  [7] -1.15191971 -1.55959370  0.24940903  0.47888318
```

```
rnorm(10, mean=5, sd=15)
```

```
##  [1] -18.2615720  -9.8018568  13.4668366  32.8726636 -19.1074299  17.9065940
##  [7]   0.1697977   5.1282920   8.1599756   1.2706731
```

```
set.seed(88899)
```

```
runif(10)
```

```
##  [1] 0.48098750 0.65454142 0.67515894 0.23419069 0.96723262 0.81835158
##  [7] 0.87185885 0.29308983 0.03261824 0.97078604
```

```
runif(10)
```

```
##  [1] 0.65245457 0.04737159 0.44235093 0.25562254 0.59711015 0.60203716
##  [7] 0.17897926 0.04146090 0.45153764 0.36984252
```

```
set.seed(88899)
```

```
runif(20)
```

```
##  [1] 0.48098750 0.65454142 0.67515894 0.23419069 0.96723262 0.81835158
##  [7] 0.87185885 0.29308983 0.03261824 0.97078604 0.65245457 0.04737159
## [13] 0.44235093 0.25562254 0.59711015 0.60203716 0.17897926 0.04146090
## [19] 0.45153764 0.36984252
```

Loops

```
for (i in 1:15){
  if (!i %% 2){
    next
  }
  print(paste(i, "is odd"))
}
```

```
## [1] "1 is odd"
## [1] "3 is odd"
## [1] "5 is odd"
## [1] "7 is odd"
## [1] "9 is odd"
## [1] "11 is odd"
## [1] "13 is odd"
## [1] "15 is odd"
```

```
imax <- 20
i <- 1
while (i <= imax){
  if( !i %% 2){
    print(paste(i, "is even"))
  }
  i = i + 1
}
```

```
## [1] "2 is even"
## [1] "4 is even"
## [1] "6 is even"
## [1] "8 is even"
## [1] "10 is even"
## [1] "12 is even"
## [1] "14 is even"
## [1] "16 is even"
## [1] "18 is even"
## [1] "20 is even"
```

```
imax <- 20
i <- 1
while (i <= imax){
  if( i %% 2 == 0){
    print(paste(i, "is even"))
  } else {
    print(paste(i, "is odd"))
  }
  i <- i + 1
}
```

```
## [1] "1 is odd"
## [1] "2 is even"
## [1] "3 is odd"
## [1] "4 is even"
## [1] "5 is odd"
## [1] "6 is even"
## [1] "7 is odd"
## [1] "8 is even"
## [1] "9 is odd"
## [1] "10 is even"
## [1] "11 is odd"
## [1] "12 is even"
## [1] "13 is odd"
## [1] "14 is even"
## [1] "15 is odd"
## [1] "16 is even"
## [1] "17 is odd"
## [1] "18 is even"
## [1] "19 is odd"
## [1] "20 is even"
```

Input: Reading ASCII data

```
getwd()
```

```
## [1] "/Users/kimwong/OneDrive - University of Pittsburgh/Documents/Kim F. Wong/CRC_Workshop/2021/Shaw
```

```
list.files()
```

```
##  [1] "Basic_R.html"                   "Basic_R.knit.md"
##  [3] "Basic_R.nb.html"                "Basic_R.pdf"
##  [5] "Basic_R.Rmd"                    "Basic_R.utf8.md"
##  [7] "CRC-On-Demand-1.png"            "CRC-On-Demand-2.png"
##  [9] "CRC-On-Demand-3.png"            "example.out"
## [11] "example.R"                      "Slide24.png"
## [13] "Slide25.png"                    "Slide26.png"
## [15] "Slide27.png"                    "Slide28.png"
## [17] "Slide46.png"                    "Slide47.png"
## [19] "students-ascii-nocompress.rds" "students-ascii.rds"
## [21] "students-bin.rds"               "students-hacked.csv"
## [23] "students-hacked.Rdmpd"          "students-hacked.tab"
## [25] "students.csv"                   "students.tab"
```

Read comma-separated data

```
grades <- read.csv("students.csv")
```

```
grades
```

```
##    name hw1 hw2 hw3 quiz1 quiz2 final
## 1  Jack  87  95  99    45    95   100
## 2  Jill  90  65  95    55    85    95
## 3  Emma 100  95  89    65    75    90
## 4 Billy  75  85  93    70    65    85
## 5 Sarah  88 100  87    75    55    80
```

Read tab-delimited data

```
grades_tab <- read.table("students.tab", header=TRUE, sep="\t")
```

```
grades_tab
```

```
##    name hw1 hw2 hw3 quiz1 quiz2 final
## 1  Jack  87  95  99    45    95   100
## 2  Jill  90  65  95    55    85    95
## 3  Emma 100  95  89    65    75    90
## 4 Billy  75  85  93    70    65    85
## 5 Sarah  88 100  87    75    55    80
```

Output: writing ASCII data

Let's modify some data first

```
grades[2, ]
```

```
##   name hw1 hw2 hw3 quiz1 quiz2 final
## 2 Jill  90  65  95    55    85    95
```

```
grades[2,2:7] <- c(100,100,100,100,100,100)
```

```
grades[2, ]
```

```
##   name hw1 hw2 hw3 quiz1 quiz2 final
## 2 Jill 100 100 100   100   100   100
```

```
grades
```

```
##    name hw1 hw2 hw3 quiz1 quiz2 final
## 1  Jack  87  95  99    45    95   100
## 2  Jill 100 100 100   100   100   100
## 3  Emma 100  95  89    65    75    90
## 4 Billy  75  85  93    70    65    85
## 5 Sarah  88 100  87    75    55    80
```

Write data in csv format

```
write.csv(grades, "students-hacked.csv")
```

Write data in tab-separated format

```
write.table(grades, "students-hacked.tab", sep="\t")
```

List files in working directory

```
list.files()
```

```
##  [1] "Basic_R.html"                "Basic_R.knit.md"
##  [3] "Basic_R.nb.html"             "Basic_R.pdf"
##  [5] "Basic_R.Rmd"                 "Basic_R.utf8.md"
##  [7] "CRC-On-Demand-1.png"         "CRC-On-Demand-2.png"
##  [9] "CRC-On-Demand-3.png"         "example.out"
## [11] "example.R"                   "Slide24.png"
## [13] "Slide25.png"                    "Slide26.png"
## [15] "Slide27.png"                    "Slide28.png"
## [17] "Slide46.png"                    "Slide47.png"
## [19] "students-ascii-nocompress.rds" "students-ascii.rds"
## [21] "students-bin.rds"               "students-hacked.csv"
## [23] "students-hacked.Rdmpd"          "students-hacked.tab"
## [25] "students.csv"                   "students.tab"
```

Output: beyond ASCII – dump

```
dump("grades", "students-hacked.Rdmpd")
```

```
rm("grades")
```

##{r} ##grades ##

```
source("students-hacked.Rdmpd")
```

```
grades
```

```
##    name hw1 hw2 hw3 quiz1 quiz2 final
## 1  Jack  87  95  99    45    95   100
## 2  Jill 100 100 100   100   100   100
## 3  Emma 100  95  89    65    75    90
## 4 Billy  75  85  93    70    65    85
## 5 Sarah  88 100  87    75    55    80
```

```
dump(c("grades", "students"), "students-hacked.Rdmpd")
```

```
##system("cat students-hacked.Rdmpd")
file.show("students-hacked.Rdmpd")
```

Output: beyond ASCII – saveRDS

```
saveRDS(grades, "students-bin.rds")
```

```
##system("cat students-bin.rds")
file.show("students-bin.rds")
```

```
saveRDS(grades, "students-ascii.rds", ascii=TRUE)
```

```
##system("cat students-ascii.rds")
file.show("students-ascii.rds")
```

```
saveRDS(grades, "students-ascii-nocompress.rds", ascii=TRUE, compress=FALSE)
```

```
##system("cat students-ascii-nocompress.rds")
file.show("students-ascii-nocompress.rds")
```

Running R scripts and outputting to file

This outputs to the console

```
source("example.R", print.eval=TRUE)
```

```
##  [1] 6.998114 5.015698 5.895312 6.382036 7.016532 6.062369 5.931604 5.107082
##  [9] 5.856181 5.256121
## [1] 10  2  8  3 10
##  [1] "South Carolina" "Alaska"         "Arizona"        "Virginia"
##  [5] "Missouri"       "Colorado"       "South Dakota"   "Massachusetts"
##  [9] "New York"       "New Mexico"     "Washington"     "Oklahoma"
## [13] "Texas"          "Montana"        "Delaware"       "Illinois"
## [17] "Ohio"           "Nevada"         "California"     "Maryland"
## [21] "Connecticut"    "Michigan"       "Louisiana"      "Rhode Island"
## [25] "Georgia"        "Indiana"        "Wisconsin"      "Utah"
## [29] "New Jersey"     "Florida"        "Vermont"        "West Virginia"
## [33] "North Carolina" "Mississippi"    "Wyoming"        "Alabama"
## [37] "North Dakota"   "Tennessee"      "Idaho"          "Nebraska"
## [41] "Kansas"         "New Hampshire"  "Maine"          "Hawaii"
## [45] "Minnesota"      "Pennsylvania"   "Kentucky"       "Arkansas"
## [49] "Oregon"         "Iowa"
```

This outputs to a specified file

##knit does not like sink`{r}` `##sink("example.out") ##`

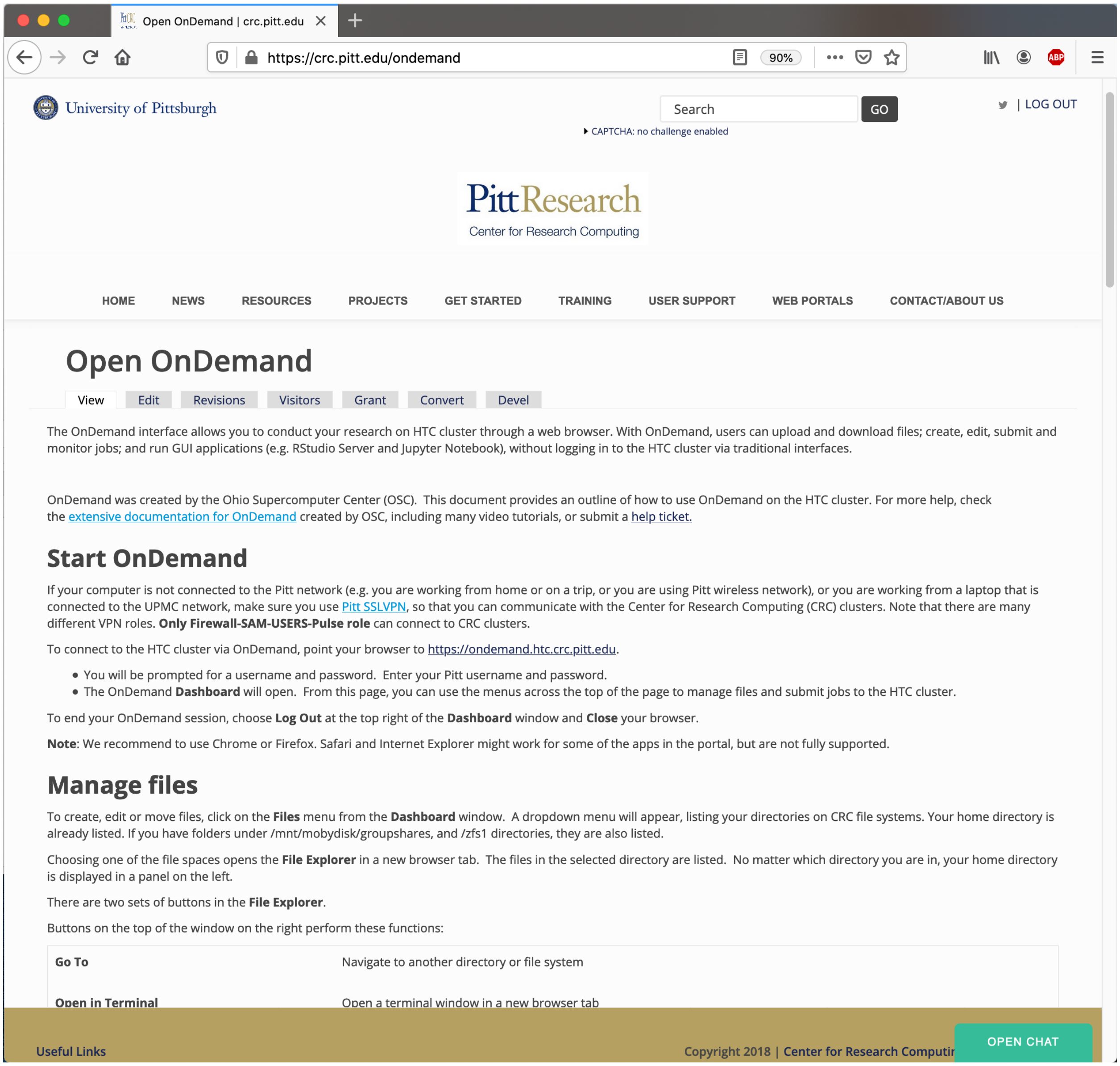
```
source("example.R", print.eval=TRUE)
```

```
##  [1] 6.950530 6.553134 5.300737 6.407774 7.161772 6.821719 6.137789 7.019207
##  [9] 7.439193 6.946376
## [1]  6  5 10  6  9
##  [1] "Pennsylvania"   "Tennessee"      "Alaska"         "Florida"
##  [5] "Ohio"           "Colorado"       "Arizona"        "West Virginia"
##  [9] "Iowa"           "Texas"          "New York"       "Kansas"
## [13] "New Mexico"     "Mississippi"    "Nebraska"       "Wisconsin"
## [17] "Oklahoma"       "Illinois"       "Oregon"         "Michigan"
## [21] "Arkansas"       "Rhode Island"   "Minnesota"      "New Hampshire"
## [25] "South Dakota"   "Idaho"          "California"     "South Carolina"
## [29] "Massachusetts"  "New Jersey"     "Vermont"        "Wyoming"
## [33] "Utah"           "Virginia"       "Delaware"       "Missouri"
## [37] "Maine"          "North Dakota"   "Indiana"        "Alabama"
## [41] "Hawaii"         "Georgia"        "Montana"        "Kentucky"
## [45] "Connecticut"    "Louisiana"      "North Carolina" "Washington"
## [49] "Nevada"         "Maryland"
```

```
system("ls | grep example.out")
```

```
##system("cat example.out")
file.show("example.out")
```

Using RStudio Server on the CRC cluster



Open OnDemand

The OnDemand interface allows you to conduct your research on HTC cluster through a web browser. With OnDemand, users can upload and download files; create, edit, submit and monitor jobs; and run GUI applications (e.g. RStudio Server and Jupyter Notebook), without logging in to the HTC cluster via traditional interfaces.

OnDemand was created by the Ohio Supercomputer Center (OSC). This document provides an outline of how to use OnDemand on the HTC cluster. For more help, check the extensive documentation for OnDemand created by OSC, including many video tutorials, or submit a help ticket.

Start OnDemand

If your computer is not connected to the Pitt network (e.g. you are working from home or on a trip, or you are using Pitt wireless network), or you are working from a laptop that is connected to the UPMC network, make sure you use Pitt SSLVPN, so that you can communicate with the Center for Research Computing (CRC) clusters. Note that there are many different VPN roles. **Only Firewall-SAM-USERS-Pulse role** can connect to CRC clusters.

To connect to the HTC cluster via OnDemand, point your browser to https://ondemand.htc.crc.pitt.edu.

- You will be prompted for a username and password. Enter your Pitt username and password.
- The OnDemand **Dashboard** will open. From this page, you can use the menus across the top of the page to manage files and submit jobs to the HTC cluster.

To end your OnDemand session, choose **Log Out** at the top right of the **Dashboard** window and **Close** your browser.

Note: We recommend to use Chrome or Firefox. Safari and Internet Explorer might work for some of the apps in the portal, but are not fully supported.

Manage files

To create, edit or move files, click on the **Files** menu from the **Dashboard** window. A dropdown menu will appear, listing your directories on CRC file systems. Your home directory is already listed. If you have folders under /mnt/mobydisk/groupshares, and /zfs1 directories, they are also listed.

Choosing one of the file spaces opens the **File Explorer** in a new browser tab. The files in the selected directory are listed. No matter which directory you are in, your home directory is displayed in a panel on the left.

There are two sets of buttons in the **File Explorer**.

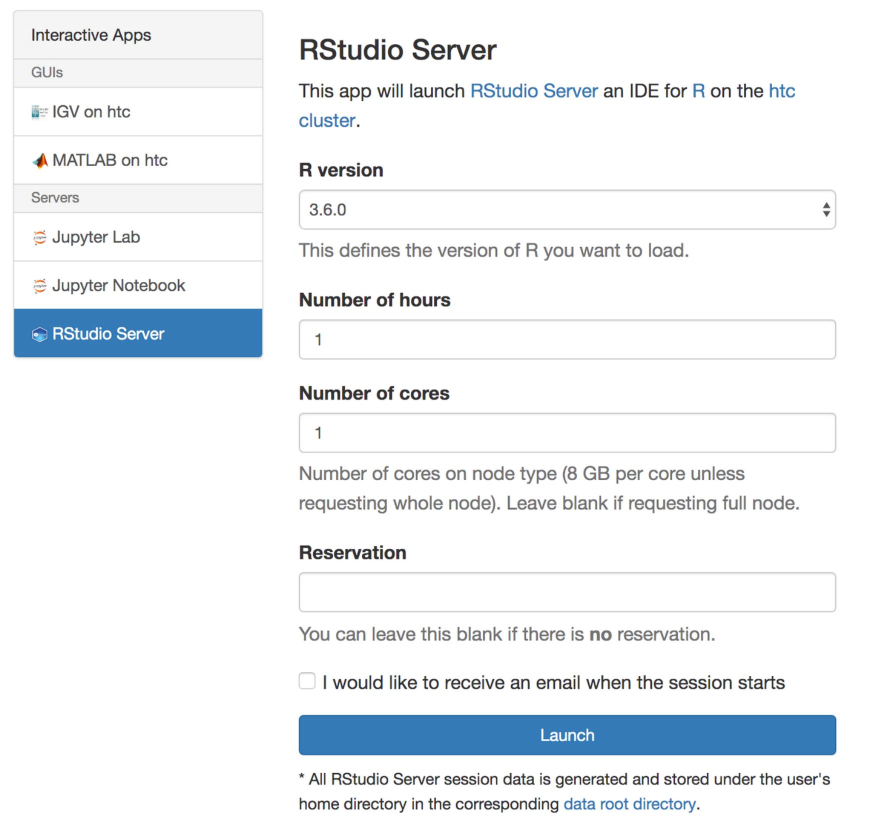
Buttons on the top of the window on the right perform these functions:

Go To	Navigate to another directory or file system
Open in Terminal	Open a terminal window in a new browser tab

RStudio

RStudio Server runs the RStudio interactive development environment inside of a browser. The OnDemand implementation allows to set up and launch the RStudio Server on a cluster compute node for dedicated resources, which allows to run more compute intensive R programs on the RStudio environment. To start RStudio Server job.

- Select **Interactive Apps** > **RStudio Server** from the top menu in the **Dashboard** window.
- In the screen that opens, specify the R version, time limit.



RStudio Server

This app will launch RStudio Server an IDE for R on the htc cluster.

R version

3.6.0

This defines the version of R you want to load.

Number of hours

1

Number of cores

1

Number of cores on node type (8 GB per core unless requesting whole node). Leave blank if requesting full node.

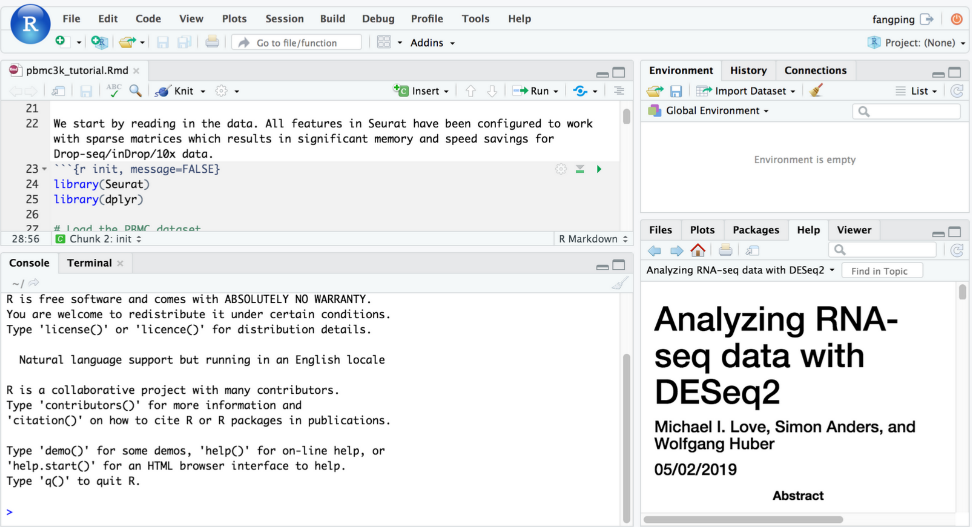
Reservation

You can leave this blank if there is **no** reservation.

□ I would like to receive an email when the session starts

* All RStudio Server session data is generated and stored under the user's home directory in the corresponding data root directory.

- Choose the appropriate number of cores, keeping in mind that R can internally thread parallelize vector based data processing, for which more than one CPU can be utilized.
- Click the blue **Launch** button to start your RStudio session. You may have to wait in the queue for resources to be available.
- When your session starts, click the blue **Connect to RStudio Server** button. A new window opens with the RStudio interface.



~~~
We start by reading in the data. All features in Seurat have been configured to work
with sparse matrices which results in significant memory and speed savings for
Drop-seq/inDrop/10x data.
```{r init, message=FALSE}
library(Seurat)
library(dplyr)

# Load the PBMC dataset
~~~

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Analyzing RNA-seq data with DESeq2

Michael I. Love, Simon Anders, and Wolfgang Huber

05/02/2019

Abstract

The R version 3.5.1 loads gcc/8.2.0 r/3.5.1 module on HTC cluster, and the R version 3.6.0 loads gcc/8.2.0 r/3.6.0 module on HTC cluster. Within each R module, various R packages and bioconductor packages have been installed. Within the R console, load the library to check whether it is already installed. If you need specific R packages, submit a help ticket.

You can also install your own R packages. R searches the user's path for libraries followed by the root installation. R will stop searching when it finds the first instance of the library within the path hierarchy.Use ".libPaths()" to check the searching path.

To install your own packages, Clusters >_HTC Shell Access

Load the R module, for example, module load gcc/8.2.0 r/3.6.0

Run R, R

Within R environment, install packages: install.packages("pkg_name")

For bioconductor packages, use: BiocManager::install("Bioconductor_pkg_name")

Errors

If you exceed the time limit you requested when setting up your RStudio session, you will see this error:

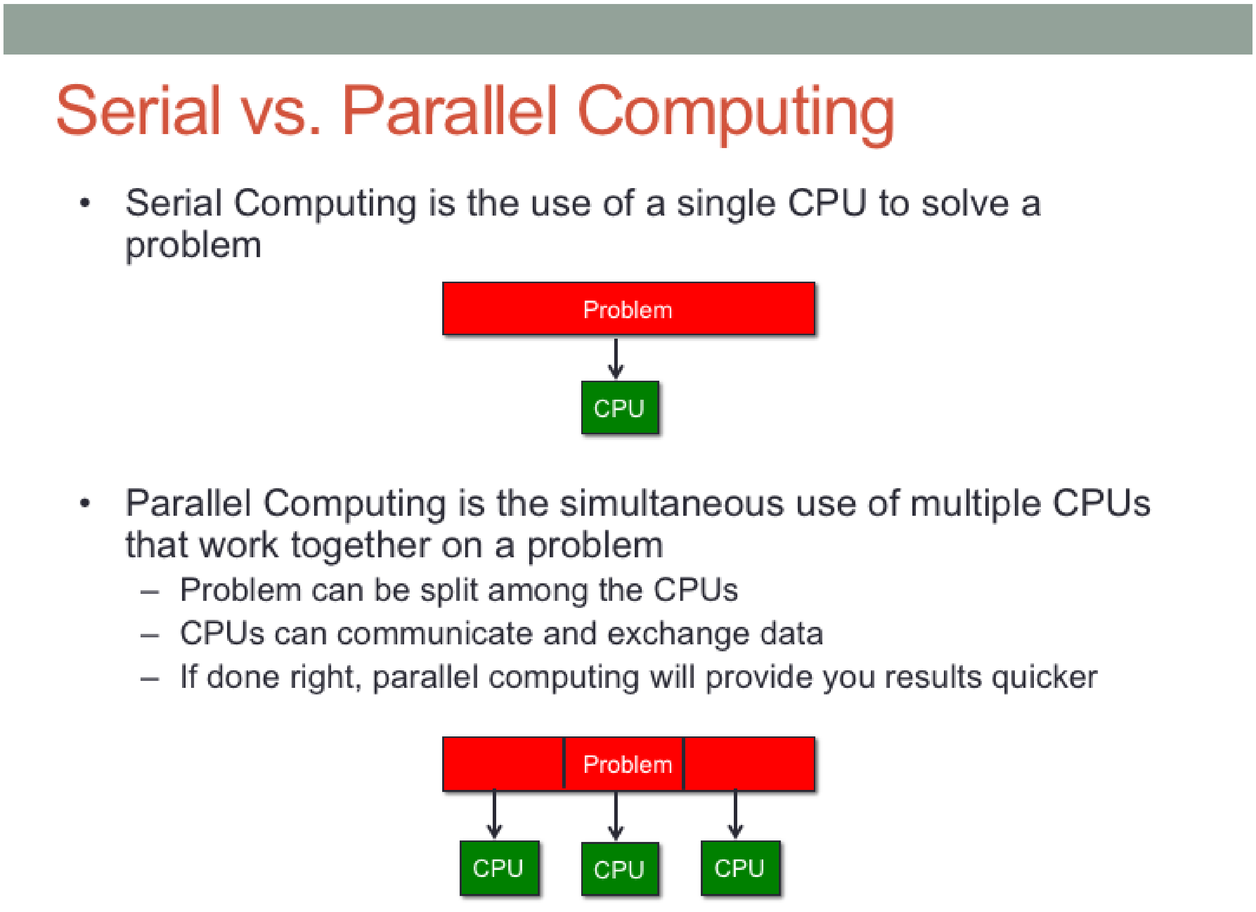
Error: Status code 503 returned

To continue using RStudio, go to Interactive Apps > RStudio from the top menu in the **Dashboard** window and start a new session.

Stopping your RStudio session

To end your RStudio session, either select **File > Quit Session** or click the red icon in the upper right of your RStudio window. **NOTE** that this only closes your RStudio session; **it does not close your interactive HTC session**. You are still consuming CPU hours on the HTC cluster.

To end your interactive HTC session, return to the Dashboard window and click the red Delete button.



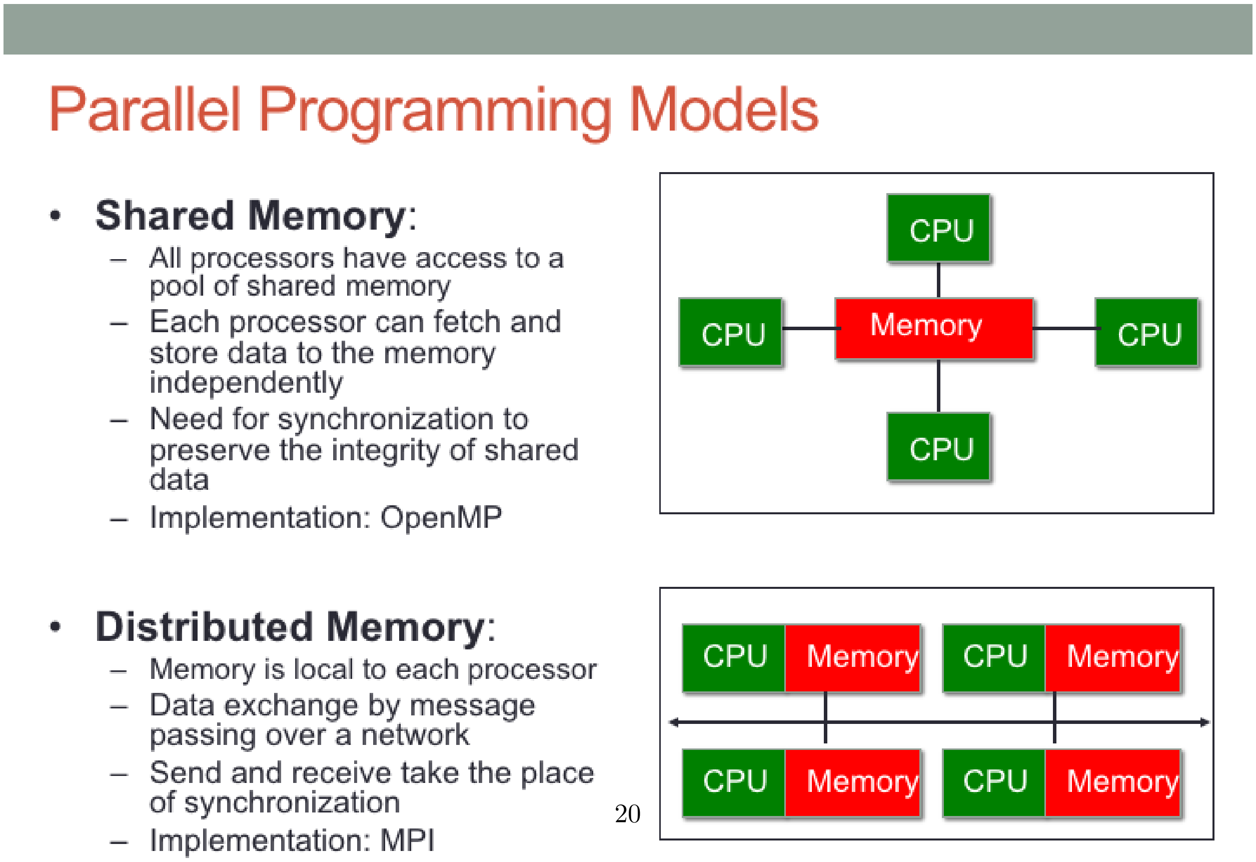
Serial vs. Parallel Computing

- Serial Computing is the use of a single CPU to solve a problem

Problem
CPU

- Parallel Computing is the simultaneous use of multiple CPUs that work together on a problem
  - Problem can be split among the CPUs
  - CPUs can communicate and exchange data
  - If done right, parallel computing will provide you results quicker

Problem
CPU
CPU
CPU



Parallel Programming Models

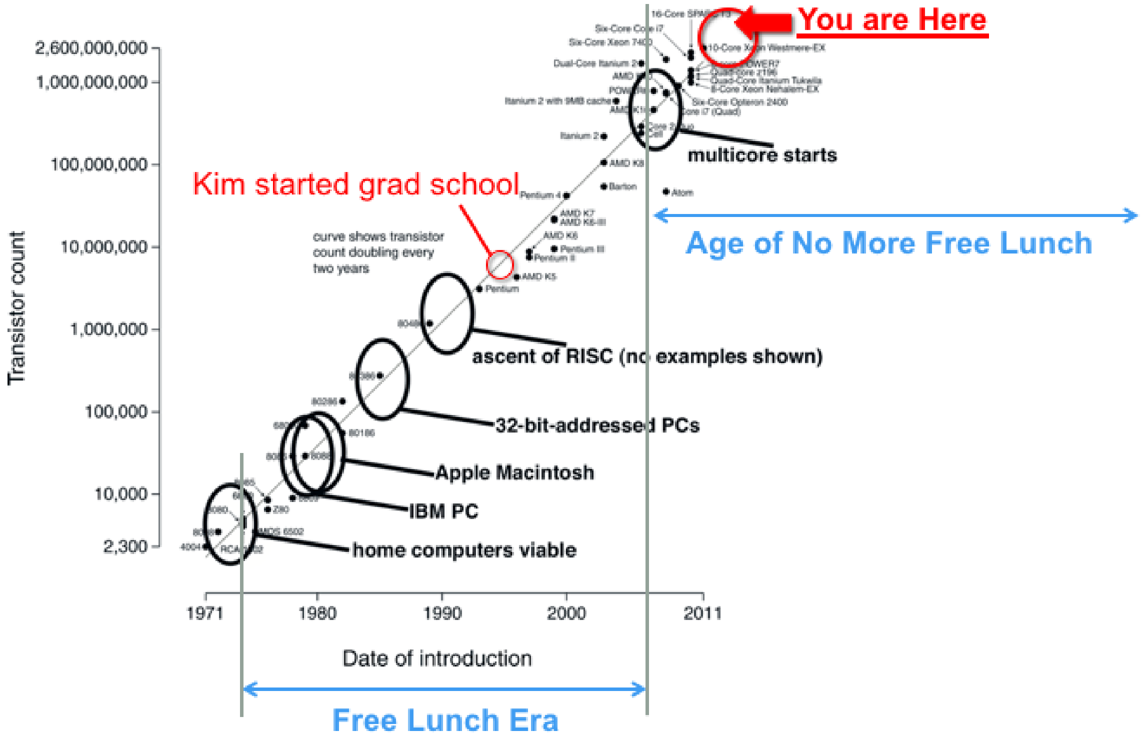
- **Shared Memory**:
  - All processors have access to a pool of shared memory
  - Each processor can fetch and store data to the memory independently
  - Need for synchronization to preserve the integrity of shared data
  - Implementation: OpenMP

CPU
CPU
Memory
CPU
CPU

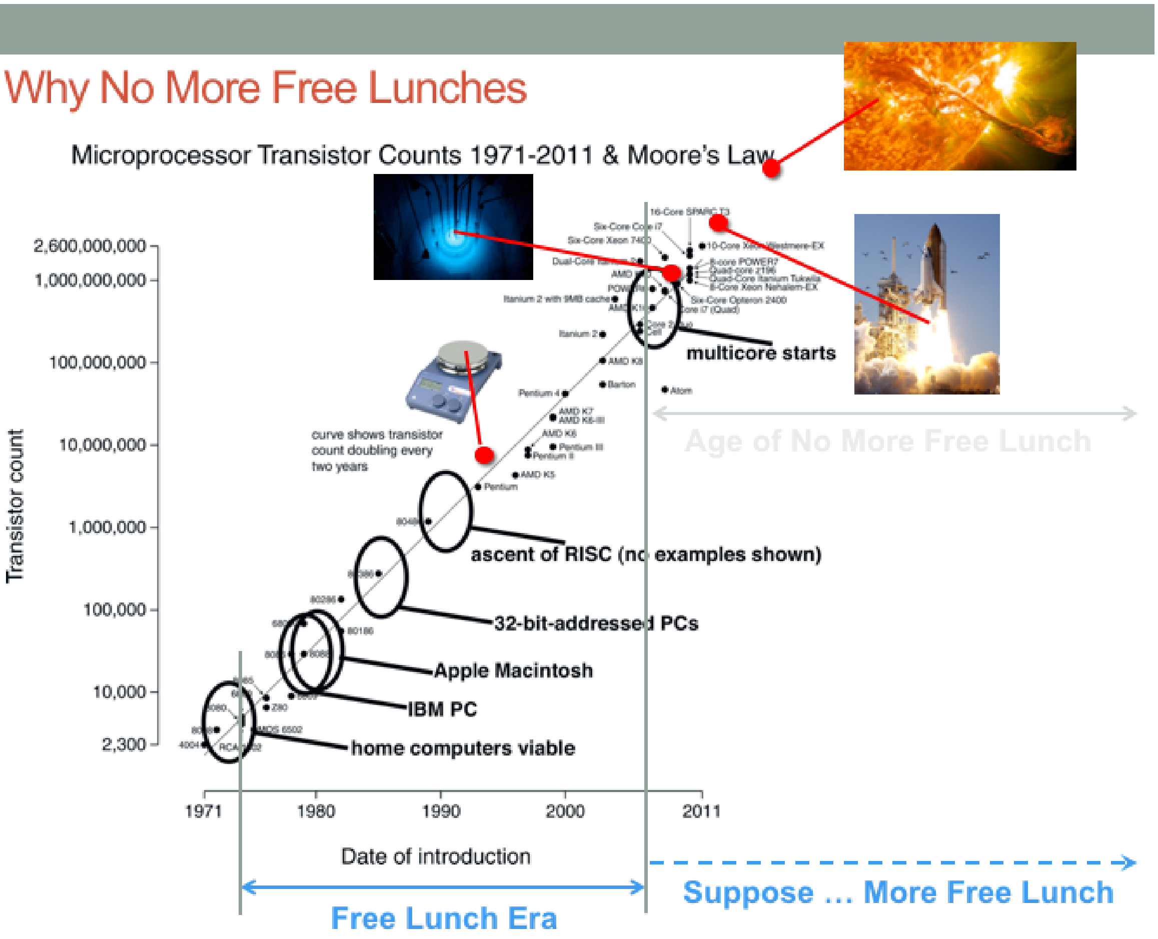
- **Distributed Memory**:
  - Memory is local to each processor
  - Data exchange by message passing over a network
  - Send and receive take the place of synchronization
  - Implementation: MPI

Long Live Moore's Law & End of Free Lunch

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Why No More Free Lunches



Microprocessor Transistor Counts 1971-2011 & Moore's Law

multicore starts

Age of No More Free Lunch

ascent of RISC (no examples shown)

32-bit-addressed PCs

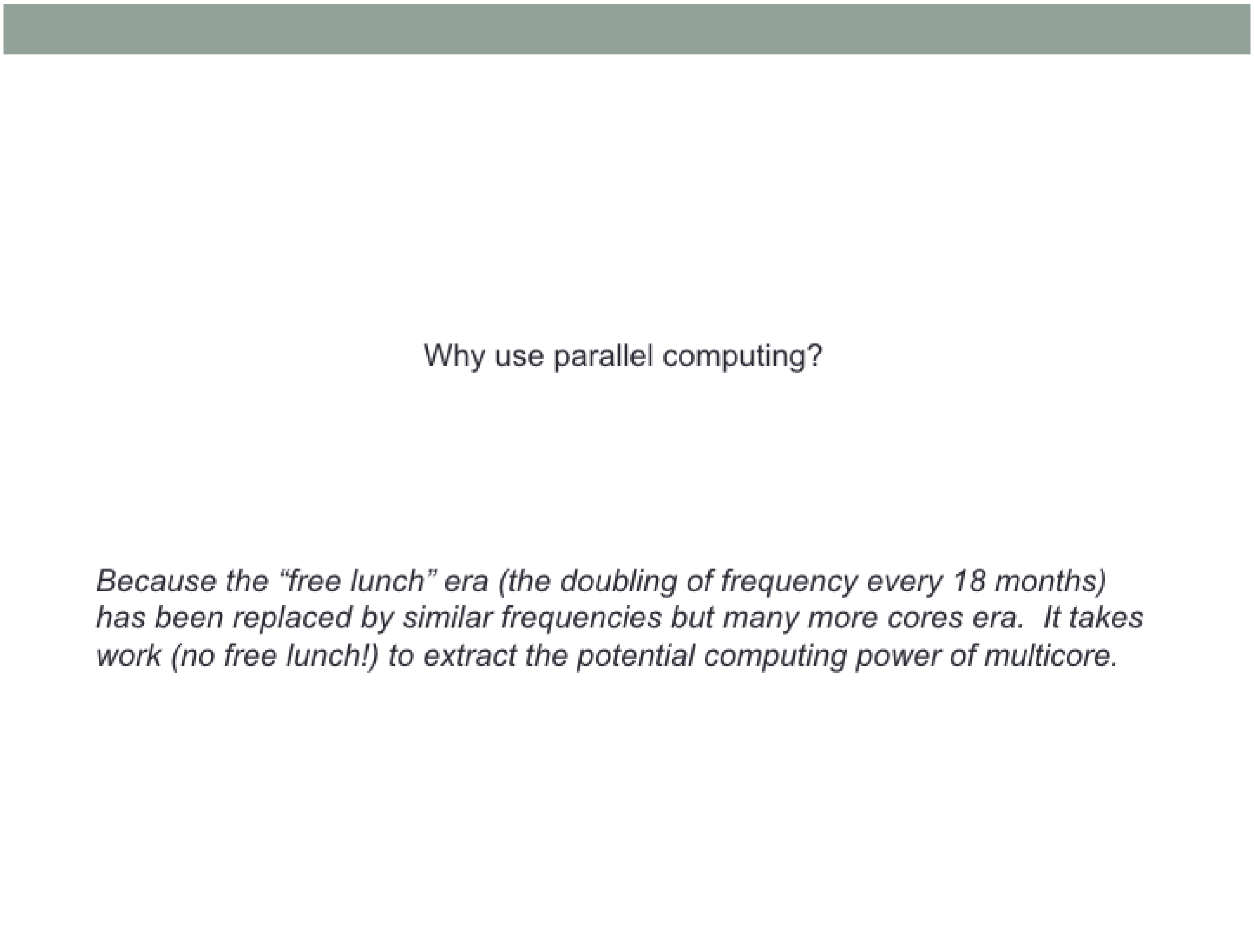
Apple Macintosh

IBM PC

home computers viable

Free Lunch Era

Suppose … More Free Lunch



Why use parallel computing?

Because the "free lunch" era (the doubling of frequency every 18 months) has been replaced by similar frequencies but many more cores era. It takes work (no free lunch!) to extract the potential computing power of multicore.

For the latest landscape overview for performing parallel computing within R, see CRAN Task View: High-Performance and Parallel Computing with R

Here, we will only discuss the following R packages

- **foreach**. Provides a new loop construct that can execute repeated tasks in parallel on multiple cores or multiple nodes of a cluster
- **parallel**. Direct support for "coarse-grained" parallel execution. Coarse-grained in the sense that large chunks of computation tasks can be farmed out to the cores simultaneously.
- **doMC**. Provides a parallel backend for the %dopar% function using the multicore functionality of the parallel package.

Example: foreach

```
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
system.time( foreach(i=1:10000) %do% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.535   0.357   1.902
```

```
registerDoParallel()
getDoParWorkers()
```

```
## [1] 12
```

```
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.625   1.744   1.348
```

```
registerDoSEQ(); getDoParWorkers()
```

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

```
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.458   0.308   1.767
```

```
registerDoParallel(cores=1); getDoParWorkers()
```

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

```
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.210   0.608   1.819
```

```
registerDoParallel(cores=16); getDoParWorkers()
```

```
## [1] 16
```

```
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.747   2.754   1.236
```

```
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.797   2.888   1.259
```

```
registerDoParallel(cores=4)
system.time( foreach(i=1:10000) %dopar% sum(tanh(1:i)) )
```

```
##    user  system elapsed
##   1.361   0.663   1.153
```

Example: foreach randomForest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
x <- matrix(runif(500), 100)
y <- gl(2, 50)
```

```
rf <- foreach(ntree=rep(250, 4), .combine=combine) %do% randomForest(x, y, ntree=ntree)
rf
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = ntree)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
```

```
rf <- foreach(ntree=rep(250, 4), .combine=combine, .packages='randomForest') %dopar% randomForest(x, y,
rf
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = ntree)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
```

Example: Multicore processing

```
library(doMC)
registerDoMC(cores=4)
```

```
library(rbenchmark)
```

```
max.eig <- function(N, sigma) {
  d <- matrix(rnorm(N**2, sd = sigma), nrow = N)
  E <- eigen(d)$values
  abs(E)[[1]]
}
```

```
benchmark(foreach(n = 1:50) %do% max.eig(n, 1),
          foreach(n = 1:50) %dopar% max.eig(n, 1)
)
```

```
##                                       test replications elapsed relative
## 1    foreach(n = 1:50) %do% max.eig(n, 1)          100   3.712    1.000
## 2 foreach(n = 1:50) %dopar% max.eig(n, 1)          100   3.870    1.043
##   user.self sys.self user.child sys.child
## 1     3.533    0.172      0.000     0.000
## 2     0.838    1.605      3.778     2.773
```

Example: R on a cluster

```
library(doSNOW)
```

```
## Loading required package: snow
```

```
##
## Attaching package: 'snow'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, clusterSplit, makeCluster, parApply,
##     parCapply, parLapply, parRapply, parSapply, splitIndices,
##     stopCluster
```

```
cluster = makeCluster(4, type = "SOCK")
registerDoSNOW(cluster)
```

```
benchmark(foreach(n = 1:50) %do% max.eig(n, 1),
          foreach(n = 1:50) %dopar% max.eig(n, 1)
)
```

```
##                                       test replications elapsed relative
## 1    foreach(n = 1:50) %do% max.eig(n, 1)          100   3.716    2.054
## 2 foreach(n = 1:50) %dopar% max.eig(n, 1)          100   1.809    1.000
##   user.self sys.self user.child sys.child
## 1     3.548    0.166          0         0
## 2     1.477    0.133          0         0
```

```
stopCluster(cluster)
```

```
cluster = makeCluster(20, type = "SOCK")
registerDoSNOW(cluster)
```

```
benchmark(foreach(n = 1:50) %do% max.eig(n, 1),
          foreach(n = 1:50) %dopar% max.eig(n, 1)
)
```

```
##                                       test replications elapsed relative
## 1    foreach(n = 1:50) %do% max.eig(n, 1)          100   3.650     1.96
## 2 foreach(n = 1:50) %dopar% max.eig(n, 1)          100   1.862     1.00
##   user.self sys.self user.child sys.child
## 1     3.507    0.140          0         0
## 2     1.685    0.168          0         0
```

```
stopCluster(cluster)
```