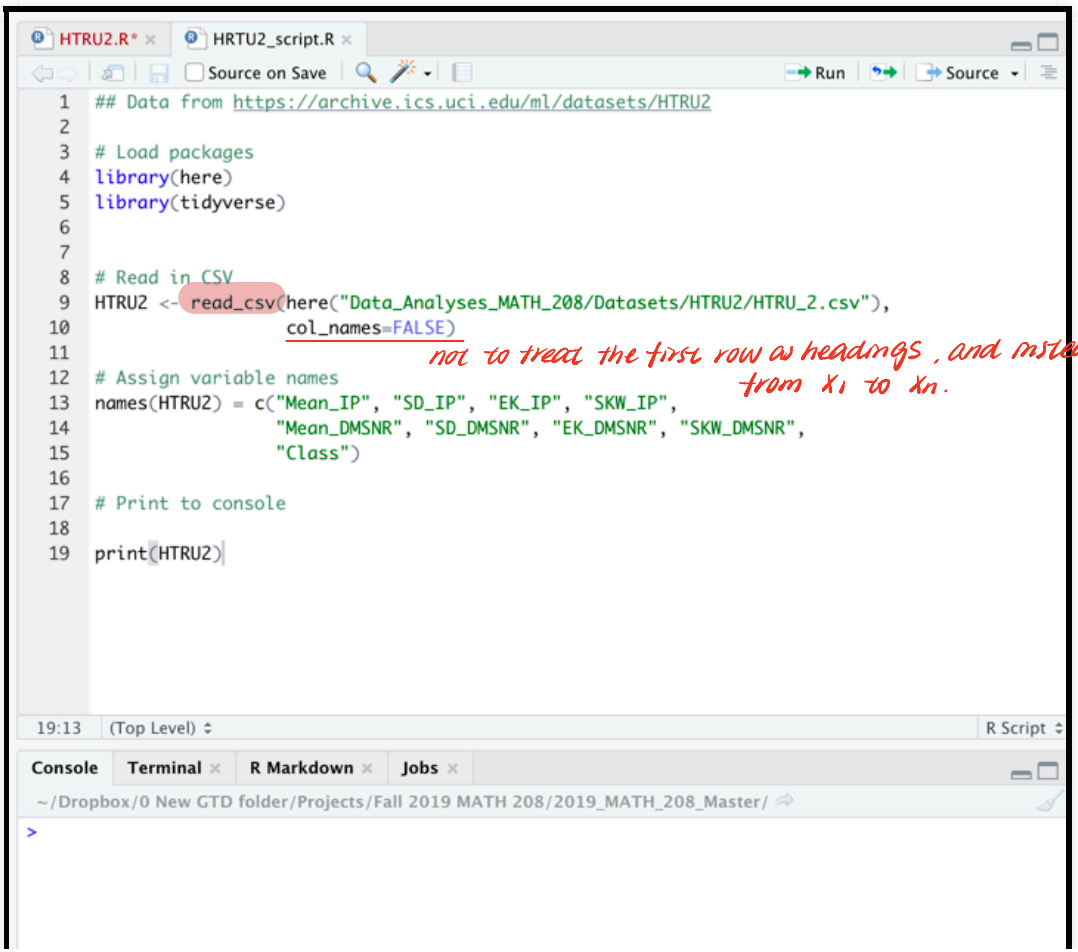
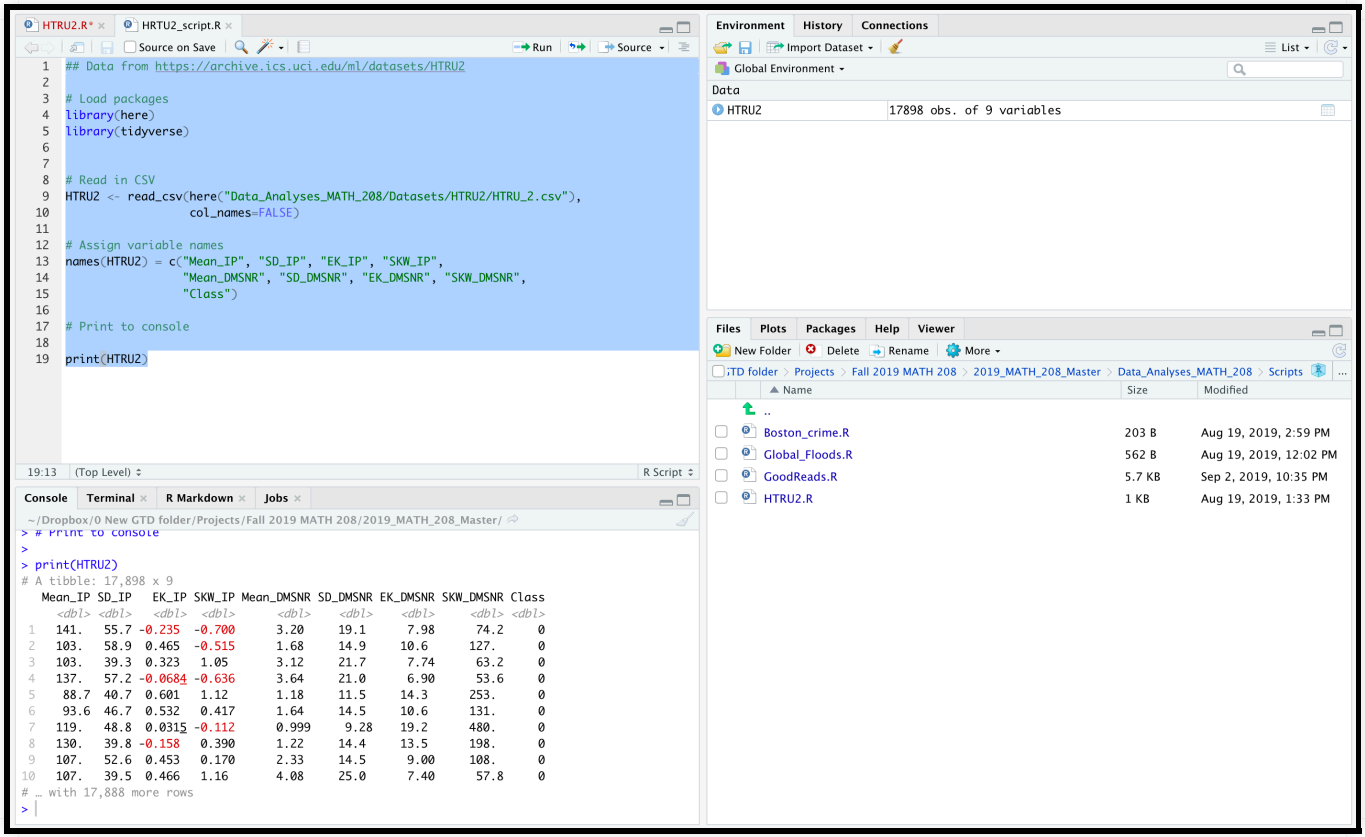
Getting data into R

Data structures

First tip: work in scripts



Results from running code



HTRU2.R* HRTU2_script.R

Source on Save | Run | Source

```
1  ## Data from https://archive.ics.uci.edu/ml/datasets/HTRU2
2
3  # Load packages
4  library(here)
5  library(tidyverse)
6
7
8  # Read in CSV
9  HTRU2 <- read_csv(here("Data_Analyses_MATH_208/Datasets/HTRU2/HTRU_2.csv"),
10                   col_names=FALSE)
11
12 # Assign variable names
13 names(HTRU2) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP",
14                  "Mean_DMSNR", "SD_DMSNR", "EK_DMSNR", "SKW_DMSNR",
15                  "Class")
16
17 # Print to console
18
19 print(HTRU2)
```

19:13 (Top Level) R Script

Console | Terminal | R Markdown | Jobs

~/Dropbox/0 New GTD folder/Projects/Fall 2019 MATH 208/2019_MATH_208_Master/

```
> # Print to console
>
> print(HTRU2)
# A tibble: 17,898 x 9
   Mean_IP SD_IP   EK_IP SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSNR Class
     <dbl> <dbl>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>     <dbl> <dbl>
 1   141.   55.7 -0.235  -0.700      3.20     19.1      7.98      74.2     0
 2   103.   58.9  0.465  -0.515      1.68     14.9     10.6      127.      0
 3   103.   39.3  0.323   1.05       3.12     21.7      7.74      63.2     0
 4   137.   57.2 -0.0684 -0.636      3.64     21.0      6.90      53.6     0
 5    88.7  40.7  0.601   1.12       1.18     11.5     14.3      253.      0
 6    93.6  46.7  0.532   0.417      1.64     14.5     10.6      131.      0
 7   119.   48.8  0.0315 -0.112      0.999     9.28    19.2      480.      0
 8   130.   39.8 -0.158   0.390      1.22     14.4     13.5      198.      0
 9   107.   52.6  0.453   0.170      2.33     14.5      9.00     108.      0
10   107.   39.5  0.466   1.16       4.08     25.0      7.40      57.8     0
# … with 17,888 more rows
>
```

Environment | History | Connections

Import Dataset | List

Global Environment

Data

HTRU2	17898 obs. of 9 variables

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

;TD folder > Projects > Fall 2019 MATH 208 > 2019_MATH_208_Master > Data_Analyses_MATH_208 > Scripts

Name	Size	Modified
..		
Boston_crime.R	203 B	Aug 19, 2019, 2:59 PM
Global_Floods.R	562 B	Aug 19, 2019, 12:02 PM
GoodReads.R	5.7 KB	Sep 2, 2019, 10:35 PM
HTRU2.R	1 KB	Aug 19, 2019, 1:33 PM

What is HTRU2?

```
# Read in the CSV
HTRU2 <-
        read_csv(here("Data_Analyses_MATH_208/Datasets/HTRU2/HTRU_2.csv"

    col_names = FALSE)
# Name the variables
names(HTRU2) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP", "Mean_DMSNR",
        "SD_DMSNR",
    "EK_DMSNR", "SKW_DMSNR", "Class")
```

```
class(HTRU2)
```

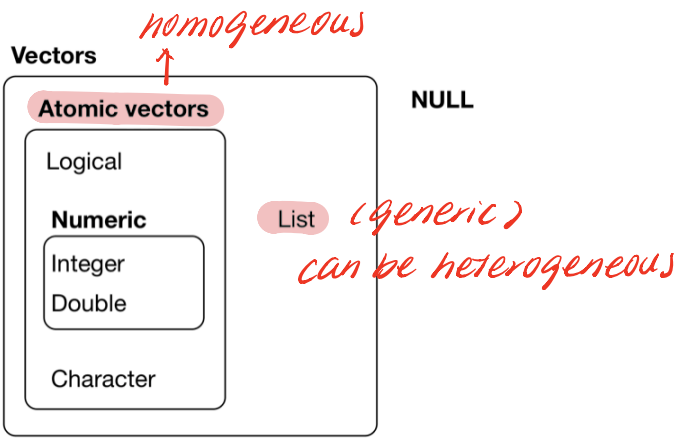
```
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

tibble vs. data.frame

main difference: printing

Tibbles are designed so that you don't accidentally overwhelm your console when you print large data frames. Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen.

The basics



Every vector has two key properties

1. type `typeof()`
2. length `length()`

The vector

```
Lengths_A <- c(52, 51, 60, 64, 69, 74, 78, 84, 86, 96, 104, 112, 118,
         125, 132,
    135)
mode(Lengths_A)
```

```
[1] "numeric"
```

```
Lengths_A[1]
```

```
[1] 52
```

```
Lengths_A[1] <- 53
Lengths_A[1]
```

```
[1] 53
```

The vector

```
basic = c(1, 2, 3)
basic[5] = 5
basic
```

```
[1]  1  2  3 NA  5
```

The vector

```
author_list = c("J.K. Rowling", "Stephen King", "Michael   Lewis",
        "Toni Morrison",
    "David McCullough")
mode(author_list)
```

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

```
boolean_vec = c(TRUE, FALSE, TRUE)
mode(boolean_vec)
```

```
[1] "logical"
```

Operators

```
Lengths_A
```

```
 [1]  53  51  60  64  69  74  78  84  86  96 104 112 118 125 132 135
```

```
Lengths_A + rep(1, 16)
```

```
 [1]  54  52  61  65  70  75  79  85  87  97 105 113 119 126 133 136
```

```
1:9
```

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

Recycling vectors

replicate vector

```
Lengths_A/rep(2.54, 16)
```

element , times

```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.7
[11] 40.945 44.094 46.457 49.213 51.969 53.150
```

"scalar"

R doesn't have scalars: instead, a single number is a vector of length 1.

```
Lengths_A/2.54
```

```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.7
[11] 40.945 44.094 46.457 49.213 51.969 53.150
```

```
c(1, 2, 3, 4) + c(3, 5)
```

```
[1] 4 7 6 9
```

R will expand the shorter vector to the same length as the longer. This is silent except when the length of the longer is not an integer multiple of the shorter.

(warning message)

Functions and Methods

Examples of functions

Function name	Argument	Action
c	Vector elements	Creates vector
rep	times/each/length.out	Replicates vector
seq.int	from/to/by/length.out/along.with	Creates sequence of integers
is.vector	Vector/mode	Returns TRUE if
atomic vector		

```
u1 = c(1,2,3)
u2 = c("one")

c(u1,u2) #  "1" "2" "3" "one"
list(u1,u2) #  [[1]]  1 2 3
               [[2]]  one
```

More examples of functions with differing arguments

```
sum(c(3, 5, NA))
```

```
[1] NA
```

```
sum(c(3, 5, NA), na.rm = T)
```

```
[1] 8
```

```
sum(c(3, 5, 7), c(1, 1, 1))
```

```
[1] 18
```

Methods

```
args(mean)
```

```
function (x, ...)
NULL
```

```
args(mean.default)
```

```
function (x, trim = 0, na.rm = FALSE, ...)
NULL
```

```
methods("mean")
```

```
[1] mean.Date     mean.default  mean.difftime mean.POSIXct  mean.POSIXlt
[6] mean.quosure*
see '?methods' for accessing help and source code
```

Moving beyond atomic vectors

Generic vectors

```
X <- matrix(1:9, nrow = 3, ncol = 3)
X
```

column first

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

```
class(X)
```

```
[1] "matrix"
```

```
mode(X)
```

```
[1] "numeric"
```

Generic vectors

```
attributes(X)
```

```
$dim
[1] 3 3
```

```
dim(X)
```

```
[1] 3 3
```

Subscripting matrices

```
X
```

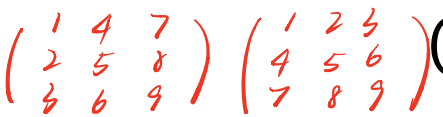
```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
X[2, 1]
```

```
[1] 2
```

```
X[4]
```

```
[1] 4
```



Operators for matrices

```
X %*% t(X)
```

```
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126
```

```
X * t(X)
```

```
     [,1] [,2] [,3]
[1,]    1    8   21
[2,]    8   25   48
[3,]   21   48   81
```

Lists

List examples

```
U1 = c(203, 204)
U2 = c(323, 324, 447)
U3 = c(208, 427, 423, 523, 545)

mymcgill_stats = list(U1, U2, U3, "Statistics Major")

mymcgill_stats
```

```
[[1]]
[1] 203 204

[[2]]
[1] 323 324 447

[[3]]
[1] 208 427 423 523 545

[[4]]
[1] "Statistics Major"
```

Naming elements and subscripting

```
mymcgill_stats = list(U1 = U1, U2 = U2, U3 = U3, Major = "Statistics
        Major")

mymcgill_stats
```

```
$U1
[1] 203 204

$U2
[1] 323 324 447

$U3
[1] 208 427 423 523 545

$Major
[1] "Statistics Major"
```

```
mymcgill_stats[["U2"]]
```

```
[1] 323 324 447
```

Comparing [[]], []. $

```
mymcgill_stats = list(U1 = U1, U2 = U2, U3 = U3)

mymcgill_stats$U2
```

same except don't need to use quotes.

```
[1] 323 324 447
```

```
mymcgill_stats[["U2"]]
```

extract a single component from a list. It removes a level of hierachy from the list.

```
[1] 323 324 447
```

```
mymcgill_stats["U2"]
```

return a new, smaller list

```
$U2
[1] 323 324 447
```

Comparing [[]], []. $ (cont.)

```
mymcgill_stats[c(2, 3)]
```

```
$U2
[1] 323 324 447

$U3
[1] 208 427 423 523 545
```

```
mymcgill_stats[c("U1", "U3")]
```

```
$U1
[1] 203 204

$U3
[1] 208 427 423 523 545
```

Comparing [[]], []. $ (cont.)

```
mymcgill_stats[[c(1, 2)]]  # Recursive indexing 1st of outer, 2nd of
        inner
```

```
[1] 204
```

```
mymcgill_stats[[1]][2]  # Access vector, access 2nd element
```

```
[1] 204
```

```
mymcgill_stats[[1]][[2]]  # Access vector, access 2nd element
```

atomic vector

```
[1] 204
```

```
mymcgill_stats[1]$U1[2]  # Access list, access U1, access 2nd element
```

```
[1] 204
```

How do we think about datasets usually?

data.frame

```
htru2_df <-
        read.csv(here("Data_Analyses_MATH_208/Datasets/HTRU2/HTRU_2.csv"


    header = FALSE)
class(htru2_df)
```

```
[1] "data.frame"
```

```
head(htru2_df)
```

show first 6 rows

tail() show last 6 rows.

	V1	V2	V3	V4	V5	V6	V7	V8	V9
1	140.562	55.684	-0.234571	-0.69965	3.1998	19.110	7.9755	74.242	0
2	102.508	58.882	0.465318	-0.51509	1.6773	14.860	10.5765	127.394	0
3	103.016	39.342	0.323328	1.05116	3.1212	21.745	7.7358	63.172	0
4	136.750	57.178	-0.068415	-0.63624	3.6430	20.959	6.8965	53.594	0
5	88.727	40.672	0.600866	1.12349	1.1789	11.469	14.2696	252.567	0
6	93.570	46.698	0.531905	0.41672	1.6363	14.545	10.6217	131.394	0

str() structure of data frame

- name, type and preview of data in each column.

tibble

```
library(tidyverse)
htru2_tbl =
        read_csv(here("Data_Analyses_MATH_208/Datasets/HTRU2/HTRU_2.csv"

    col_names = FALSE)
class(htru2_tbl)
```

```
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

tibble vs. data.frame

```
htru2_tbl
```

```
# A tibble: 17,898 x 9
      X1    X2       X3      X4    X5    X6    X7    X8    X9
   <dbl> <dbl>    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 141.   55.7 -0.235   -0.700 3.20  19.1   7.98  74.2      0
 2 103.   58.9  0.465   -0.515 1.68  14.9  10.6  127.       0
 3 103.   39.3  0.323    1.05  3.12  21.7   7.74  63.2      0
 4 137.   57.2 -0.0684 -0.636 3.64  21.0   6.90  53.6      0
 5  88.7  40.7  0.601    1.12  1.18  11.5  14.3  253.       0
 6  93.6  46.7  0.532    0.417 1.64  14.5  10.6  131.       0
 7 119.   48.8  0.0315 -0.112 0.999  9.28 19.2  480.       0
 8 130.   39.8 -0.158    0.390 1.22  14.4  13.5  198.       0
 9 107.   52.6  0.453    0.170 2.33  14.5   9.00 108.       0
10 107.   39.5  0.466    1.16  4.08  25.0   7.40  57.8      0
# … with 17,888 more rows
```

tibble vs. data.frame

main difference: printing

Tibbles are designed so that you don't accidentally overwhelm your console when you print large data frames. Tibbles have a refined print method that shows only the first 10 rows, and all the columns that fit on screen. Each column reports its type.

tibble vs. data.frame (cont.)

```
head(as.data.frame(htru2_tbl))
```

```
       X1     X2         X3        X4     X5     X6       X7       X8 X9
1 140.562 55.684 -0.234571 -0.69965 3.1998 19.110  7.9755  74.242  0
2 102.508 58.882  0.465318 -0.51509 1.6773 14.860 10.5765 127.394  0
3 103.016 39.342  0.323328  1.05116 3.1212 21.745  7.7358  63.172  0
4 136.750 57.178 -0.068415 -0.63624 3.6430 20.959  6.8965  53.594  0
5  88.727 40.672  0.600866  1.12349 1.1789 11.469 14.2696 252.567  0
6  93.570 46.698  0.531905  0.41672 1.6363 14.545 10.6217 131.394  0
```

tibble vs. data.frame, Round 2

```
mymcgill_stats_tbl = tibble(Courses = list(U1 = U1, U2 = U2, U3 = U3),
        Year = c("U1",
    "U2", "U3"), Major = rep("Statistics Major", 3))
mymcgill_stats_tbl
```

```
# A tibble: 3 x 3
  Courses   Year  Major
  <list>    <chr> <chr>
1 <dbl [2]> U1    Statistics Major
2 <dbl [3]> U2    Statistics Major
3 <dbl [5]> U3    Statistics Major
```

Back to HTRU2

```
names(attributes(htru2_tbl))
```

```
[1] "names"     "class"     "row.names" "spec"
```

```
attributes(htru2_tbl)$names
```

```
[1] "X1" "X2" "X3" "X4" "X5" "X6" "X7" "X8" "X9"
```

```
names(htru2_tbl) = c("Mean_IP", "SD_IP", "EK_IP", "SKW_IP",
        "Mean_DMSNR", "SD_DMSNR",
    "EK_DMSNR", "SKW_DMSNR", "Class")
htru2_tbl
```

```
# A tibble: 17,898 x 9
   Mean_IP SD_IP   EK_IP SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSNR
     <dbl> <dbl>   <dbl>  <dbl>      <dbl>    <dbl>    <dbl>      <dbl>
 1   141.   55.7 -0.235  -0.700       3.20     19.1      7.98       74.2
 2   103.   58.9  0.465  -0.515       1.68     14.9     10.6       127.
 3   103.   39.3  0.323   1.05        3.12     21.7      7.74       63.2
 4   137.   57.2 -0.0684 -0.636       3.64     21.0      6.90       53.6
 5    88.7  40.7  0.601   1.12        1.18     11.5     14.3       253.
 6    93.6  46.7  0.532   0.417       1.64     14.5     10.6       131.
```

```
 7   119.    48.8  0.0315 -0.112       0.999     9.28     19.2       480.
 8   130.    39.8 -0.158    0.390       1.22     14.4      13.5       198.
 9   107.    52.6  0.453    0.170       2.33     14.5       9.00      108.
10   107.    39.5  0.466    1.16        4.08     25.0       7.40       57.8
# … with 17,888 more rows, and 1 more variable: Class <dbl>
```

Subsetting

Subsetting

① With numeric vectors

```
basic[c(1,2)]      # 1 2
basic[c(1,2,1,2)]  # 1 2 1 2
basic[c(-1,-2)]    # 3
```

Negative values drop the element at the specified positions.

It's an error to mix positive and negative values.

② With logical vectors.

Subsetting vectors

```
Lengths_A
```

```
 [1]  53  51  60  64  69  74  78  84  86  96 104 112 118 125 132 135
```

```
## Subsetting by index
Lengths_A[c(1, 5, 6)]
```

```
[1] 53 69 74
```

Subsetting by logical vector

```
Lengths_A > 100
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[12]  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
Lengths_A[Lengths_A > 100]
```

```
[1] 104 112 118 125 132 135
```

Subsetting lists by index

```
mymcgill_stats
```

```
$U1
[1] 203 204

$U2
[1] 323 324 447

$U3
[1] 208 427 423 523 545
```

```
mymcgill_stats[c(1, 3)]
```

```
$U1
[1] 203 204

$U3
[1] 208 427 423 523 545
```

Subsetting lists by logical

```
lapply(mymcgill_stats, length)  ## Finds length of each element of list
```

```
$U1
[1] 2

$U2
[1] 3

$U3
[1] 5
```

```
lapply(mymcgill_stats, length) < 4
```

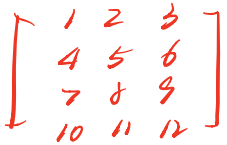
```
   U1    U2    U3
 TRUE  TRUE FALSE
```

Subsetting lists by logical

```
mymcgill_stats[lapply(mymcgill_stats, length) < 4]
```

```
$U1
[1] 203 204

$U2
[1] 323 324 447
```



Subsetting rectangular objects

```
A_mat = matrix(c(1:12), ncol = 3, nrow = 4, byrow = T)
A_mat[3, 2]
```

by default : by column.

```
[1] 8
```

```
A_mat[c(1:3), ]
```

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

Subsetting rectangular objects

```
A_mat[, c(1, 3)]
```

```
     [,1] [,2]
[1,]    1    3
[2,]    4    6
[3,]    7    9
[4,]   10   12
```

```
A_mat[c(1:3), c(1, 3)]
```

```
     [,1] [,2]
[1,]    1    3
[2,]    4    6
[3,]    7    9
```

Subsetting rectangular objects via logical indices

```
A_mat[c(TRUE, TRUE, TRUE, FALSE), c(TRUE, TRUE, FALSE)]
```

```
     [,1] [,2]
[1,]    1    2
[2,]    4    5
[3,]    7    8
```

```
A_mat%%2 == 0
```

```
      [,1]  [,2]  [,3]
[1,] FALSE  TRUE FALSE
[2,]  TRUE FALSE  TRUE
[3,] FALSE  TRUE FALSE
[4,]  TRUE FALSE  TRUE
```

Subsetting rectangular objects via logical indices

```
A_mat[A_mat%%2 == 0]
```

treat matrix in column order, turn it into a vector, give it back a vector

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

Moving onto data.frame's

```
names(htru2_df) = names(htru2_tbl)
htru2_df[1:3, ]
```

```
  Mean_IP  SD_IP    EK_IP   SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSN
1  140.56 55.684 -0.23457 -0.69965     3.1998   19.110   7.9755    74.24
2  102.51 58.882  0.46532 -0.51509     1.6773   14.860  10.5765   127.39
3  103.02 39.342  0.32333  1.05116     3.1212   21.745   7.7358    63.17
  Class
1     0
2     0
3     0
```

```
htru2_df[1:5, 1:2]
```

```
  Mean_IP  SD_IP
1 140.562 55.684
2 102.508 58.882
3 103.016 39.342
4 136.750 57.178
5  88.727 40.672
```

Moving onto data.frame's

```
htru2_df_MIP14 = htru2_df[htru2_df$Mean_IP > 140, 1:3]
dim(htru2_df_MIP14)
```

```
[1] 1276    3
```

```
head(htru2_df[htru2_df$Mean_IP > 140, c("Mean_IP", "SD_IP", "Class")])
```

```
   Mean_IP  SD_IP Class
1   140.56 55.684     0
11  142.08 45.288     0
34  142.05 53.873     0
37  147.84 53.623     0
59  141.97 50.471     0
67  143.09 49.922     0
```

Moving onto data.frame's using subset function

```
htru2_df_MIP14_sub = subset(htru2_df, Mean_IP > 140)
head(htru2_df[htru2_df$Mean_IP > 140, c("Mean_IP", "SD_IP", "Class")])
```

```
   Mean_IP  SD_IP Class
1   140.56 55.684     0
11  142.08 45.288     0
34  142.05 53.873     0
37  147.84 53.623     0
59  141.97 50.471     0
67  143.09 49.922     0
```

Subsetting data with tibbles

```
htru2_tbl[1:3, c(1, 5, 9)]
```

```
# A tibble: 3 x 3
  Mean_IP Mean_DMSNR Class
    <dbl>      <dbl> <dbl>
1    141.       3.20     0
2    103.       1.68     0
3    103.       3.12     0
```

```
htru2_tbl[1:3, c("Mean_IP", "SD_IP", "Class")]
```

```
# A tibble: 3 x 3
  Mean_IP SD_IP Class
    <dbl> <dbl> <dbl>
1    141.  55.7     0
2    103.  58.9     0
3    103.  39.3     0
```

Subsetting data with tibbles

```
head(select(htru2_tbl, Mean_IP, SD_IP))
```

column

```
# A tibble: 6 x 2
  Mean_IP SD_IP
    <dbl> <dbl>
1   141.   55.7
2   103.   58.9
3   103.   39.3
4   137.   57.2
5    88.7  40.7
6    93.6  46.7
```

```
slice(htru2_tbl, 1:3)
```

row

```
# A tibble: 3 x 9
  Mean_IP SD_IP  EK_IP SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSNR Cla
    <dbl> <dbl>  <dbl>  <dbl>      <dbl>    <dbl>    <dbl>     <dbl> <db
1    141.  55.7 -0.235 -0.700       3.20     19.1     7.98      74.2
2    103.  58.9  0.465 -0.515       1.68     14.9    10.6      127.
3    103.  39.3  0.323  1.05        3.12     21.7     7.74      63.2
```

Subsetting data with tibbles

```
slice(select(htru2_tbl, Mean_IP, SD_IP), 1:3)
```

```
# A tibble: 3 x 2
  Mean_IP SD_IP
    <dbl> <dbl>
1    141.  55.7
2    103.  58.9
3    103.  39.3
```

```
htru2_tbl %>% select(Mean_IP, SD_IP) %>% slice(1:3)
```

```
# A tibble: 3 x 2
  Mean_IP SD_IP
    <dbl> <dbl>
1    141.  55.7
2    103.  58.9
3    103.  39.3
```

Order matters only if considering speed.

Subsetting data with tibbles

```
htru2_tbl %>% select(Mean_IP, SD_IP) %>% filter(Mean_IP > 140) %>%
        slice(1:5)
```

```
# A tibble: 5 x 2
  Mean_IP SD_IP
    <dbl> <dbl>
1    141.  55.7
2    142.  45.3
3    142.  53.9
4    148.  53.6
5    142.  50.5
```

Digression: packages in R

Digression: packages in R

```
search()
```

```
 [1] ".GlobalEnv"          "package:kableExtra" "package:knitr"
 [4] "package:forcats"     "package:stringr"    "package:dplyr"
 [7] "package:purrr"       "package:readr"      "package:tidyr"
[10] "package:tibble"      "package:ggplot2"    "package:tidyverse"
[13] "package:here"        "package:stats"      "package:graphics"
[16] "package:grDevices"   "package:utils"      "package:datasets"
[19] "package:methods"     "Autoloads"          "package:base"
```

```
install.packages("tidyverse")
install.packages("kableExtra")
```

Back to the lengths data

```
Lengths_B <- c(51, 53.5, 56, 66, 68, 72.5, 79, 80, 81, 91, 96.8, 101,
        110)
Lengths_C <- c(52.5, 49.5, 60, 65, 67, 74, 78, 79, 83, 90)
```

When the lengths were measured

```
Dates_A <- c("2009-05-02", "2009-05-11", "2009-07-07", "2009-09-09",
        "2009-11-16",
    "2010-02-16", "2010-05-03", "2010-11-09", "2011-05-04", "2012-05-
        03", "2013-05-18",
    "2014-07-30", "2015-09-30", "2016-06-08", "2017-08-21", "2018-09-
        19")

Dates_B <- c("2012-08-02", "2012-08-15", "2012-10-31", "2012-12-06",
        "2013-02-18",
    "2013-05-13", "2013-09-05", "2014-02-17", "2014-07-30", "2015-09-
        30", "2016-08-04",
    "2017-08-21", "2018-09-19")

Dates_C <- c("2016-02-04", "2016-02-18", "2016-04-11", "2016-06-08",
        "2016-08-04",
    "2016-11-17", "2017-03-02", "2017-08-21", "2018-02-20", "2019-02-
```

When the lengths were measured

```
Dates_A <- c("2009-05-02","2009-05-11","2009-07-07","2009-09-09",
             "2009-11-16","2010-02-16","2010-05-03","2010-11-09",
             "2011-05-04","2012-05-03","2013-05-18","2014-07-30",
             "2015-09-30","2016-06-08","2017-08-21","2018-09-19")

mode(Dates_A)
```

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

```
class(Dates_A)
```

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

When the lengths were measured

```
Dates_A <- as.Date(Dates_A)
mode(Dates_A)
```

```
[1] "numeric"
```

```
head(as.numeric(Dates_A))
```

```
[1] 14366 14375 14432 14496 14564 14656
```

number of days since 1970-01-01

```
class(Dates_A)
```

```
[1] "Date"
```

When the lengths were measured

```
Dates_B <- as.Date(c("2012-08-02","2012-08-15","2012-10-31","2012-12-
        06",
                   "2013-02-18","2013-05-13","2013-09-05","2014-02-
        17",
                   "2014-07-30","2015-09-30","2016-08-04","2017-08-
        21",
                   "2018-09-19"))

Dates_C <- as.Date(c("2016-02-04","2016-02-18","2016-04-11","2016-06-
        08",
             "2016-08-04","2016-11-17",
             "2017-03-02","2017-08-21","2018-02-20","2019-02-13"))
```

How to create a dataframe or tibble?

How to create a dataframe or tibble?

don't hardcode: dependent on the data.

```
kids_heights_df <- data.frame(Kid = c(rep("A", length(Lengths_A)),
        rep("B",
    length(Lengths_B)), rep("C", length(Lengths_C))), Heights =
        c(Lengths_A,
    Lengths_B, Lengths_C), Dates = as.Date(c(Dates_A, Dates_B,
        Dates_C)))
head(kids_heights_df)
```

```
  Kid Heights      Dates
1   A      53 2009-05-02
2   A      51 2009-05-11
3   A      60 2009-07-07
4   A      64 2009-09-09
5   A      69 2009-11-16
6   A      74 2010-02-16
```

How to create a dataframe or tibble?

```
kids_heights_tbl <- tibble(Kid = c(rep("A", length(Lengths_A)),
        rep("B", length(Lengths_B)),
    rep("C", length(Lengths_C))), Heights = c(Lengths_A, Lengths_B,
        Lengths_C),
    Dates = as.Date(c(Dates_A, Dates_B, Dates_C)))
head(kids_heights_tbl)
```

```
# A tibble: 6 x 3
  Kid   Heights Dates
  <chr>   <dbl> <date>
1 A          53 2009-05-02
2 A          51 2009-05-11
3 A          60 2009-07-07
4 A          64 2009-09-09
5 A          69 2009-11-16
6 A          74 2010-02-16
```

Grouping by a variable

```
kids_heights_tbl_grp <- group_by(kids_heights_tbl,Kid)
str(kids_heights_tbl_grp)
```

Every operation is done individually and independently on each group

```
Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame': 39 obs. of  3 va
 $ Kid    : chr  "A" "A" "A" "A" ...
 $ Heights: num  53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date, format: "2009-05-02" "2009-05-11" ...
 - attr(*, "groups")=Classes 'tbl_df', 'tbl' and 'data.frame':  3 obs. c
  ..$ Kid  : chr  "A" "B" "C"
  ..$ .rows:List of 3
  .. ..$ : int  1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int  17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int  30 31 32 33 34 35 36 37 38 39
  ..- attr(*, ".drop")= logi TRUE
```

Adding a column using mutate

```
kids_heights_tbl_grp <- mutate(kids_heights_tbl_grp,Birth = min(Dates),
          Age = Dates - min(Dates))

head(kids_heights_tbl_grp)
```

```
# A tibble: 6 x 5
# Groups:   Kid [1]
  Kid   Heights Dates      Birth      Age
  <chr>   <dbl> <date>     <date>     <drtn>
1 A          53 2009-05-02 2009-05-02   0 days
2 A          51 2009-05-11 2009-05-02   9 days
3 A          60 2009-07-07 2009-05-02  66 days
4 A          64 2009-09-09 2009-05-02 130 days
5 A          69 2009-11-16 2009-05-02 198 days
6 A          74 2010-02-16 2009-05-02 290 days
```

Using %>% and magittr

Using %>% and magittr

Using %>% and magittr

```
kids_heights_tbl_grp <- kids_heights_tbl %>% group_by(Kid) %>%
  mutate(Birth = min(Dates),
         Age = Dates - min(Dates))
str(kids_heights_tbl_grp)
```

```
Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame': 39 obs. of  5 va
 $ Kid    : chr  "A" "A" "A" "A" ...
 $ Heights: num  53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date, format: "2009-05-02" "2009-05-11" ...
 $ Birth  : Date, format: "2009-05-02" "2009-05-02" ...
 $ Age    : 'difftime' num  0 9 66 130 ...
  ..- attr(*, "units")= chr "days"
 - attr(*, "groups")=Classes 'tbl_df', 'tbl' and 'data.frame':  3 obs. c
  ..$ Kid  : chr  "A" "B" "C"
  ..$ .rows:List of 3
  .. ..$ : int  1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int  17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int  30 31 32 33 34 35 36 37 38 39
  ..- attr(*, ".drop")= logi TRUE
```