Getting Data into R

Data structures

Why we're starting here

> "Bad programmers worry about the code. Good programmers worry about data structures and their relationships."
>
> --- Linus Torvalds

HTRU2

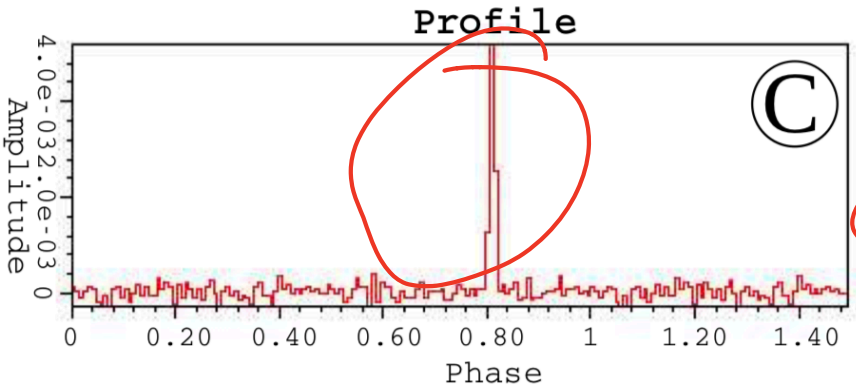
Monthly Notices of the Royal Astronomical Society / Volume 409, Issue 2

The High Time Resolution Universe Pulsar Survey – I. System configuration and initial discoveries

M. J. Keith ✉, A. Jameson, W. van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. D. Bates, N. D. R. Bhat, M. Burgay, S. Burke-Spolaor, N. D'Amico, L. Levin, Peter L. McMahon, S. Milia, B. W. Stappers

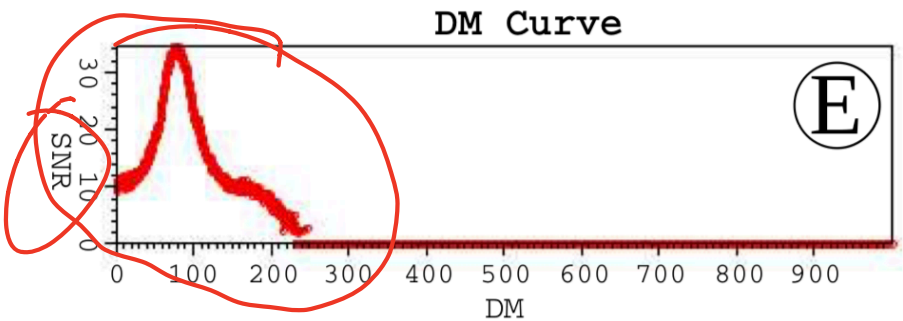
First published: 14 September 2010
https://doi.org/10.1111/j.1365-2966.2010.17325.x
Cited by: 1

Pulsar emission profiles from
https://arxiv.org/pdf/1603.05166.pdf



Symmetric

Skew

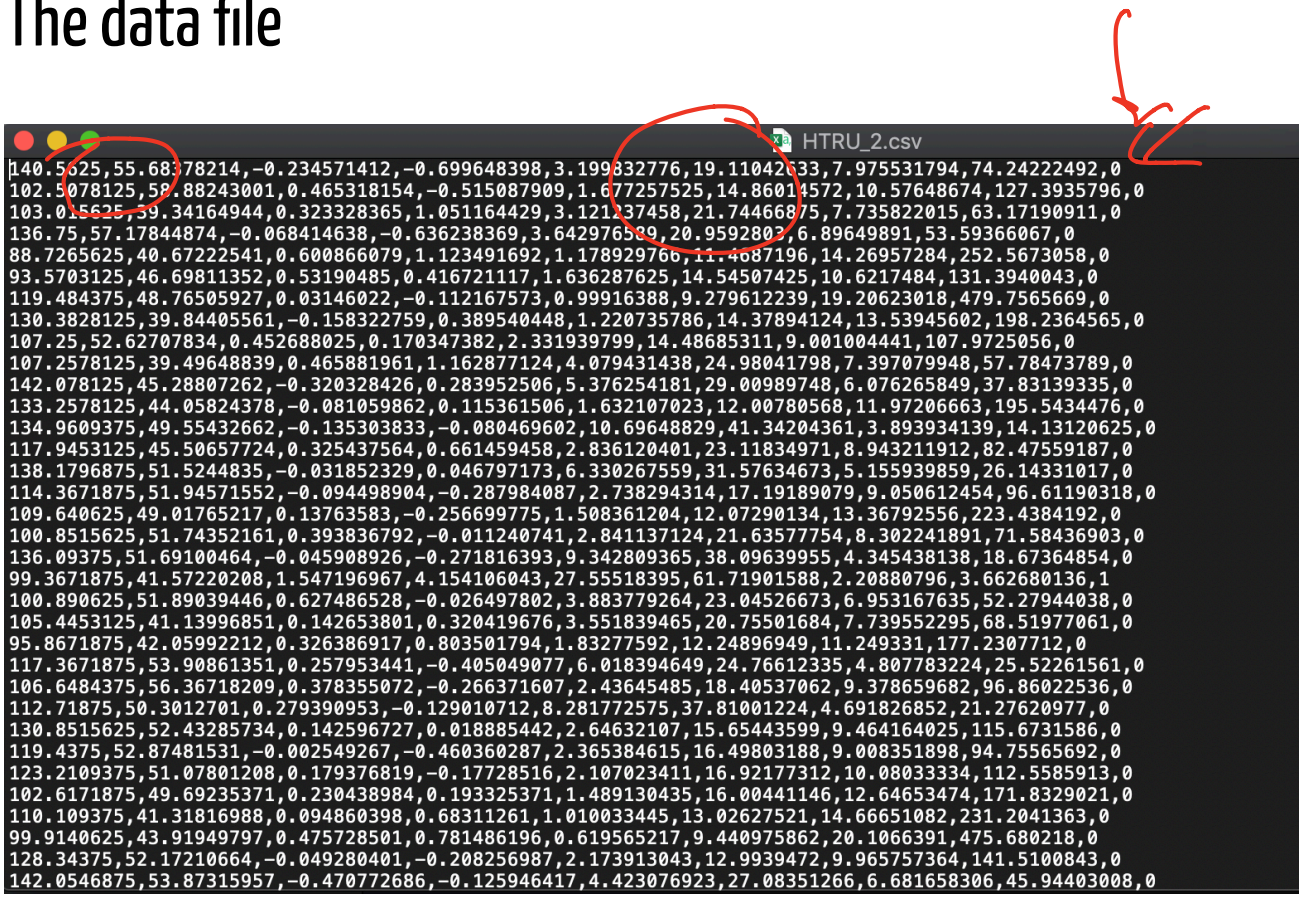


The data

Measurements
1. Mean of the integrated profile
2. Standard deviation of the integrated profile
3. Excess kurtosis of the integrated profile
4. Skewness of the integrated profile
5. Mean of the DM-SNR curve
6. Standard deviation of the DM-SNR curve
7. Excess kurtosis of the DM-SNR curve
8. Skewness of the DM-SNR curve
9. True or false pulsar (human-verified)

Goal: Discriminate between True & False pulsars using only 8 characteristics

The data file

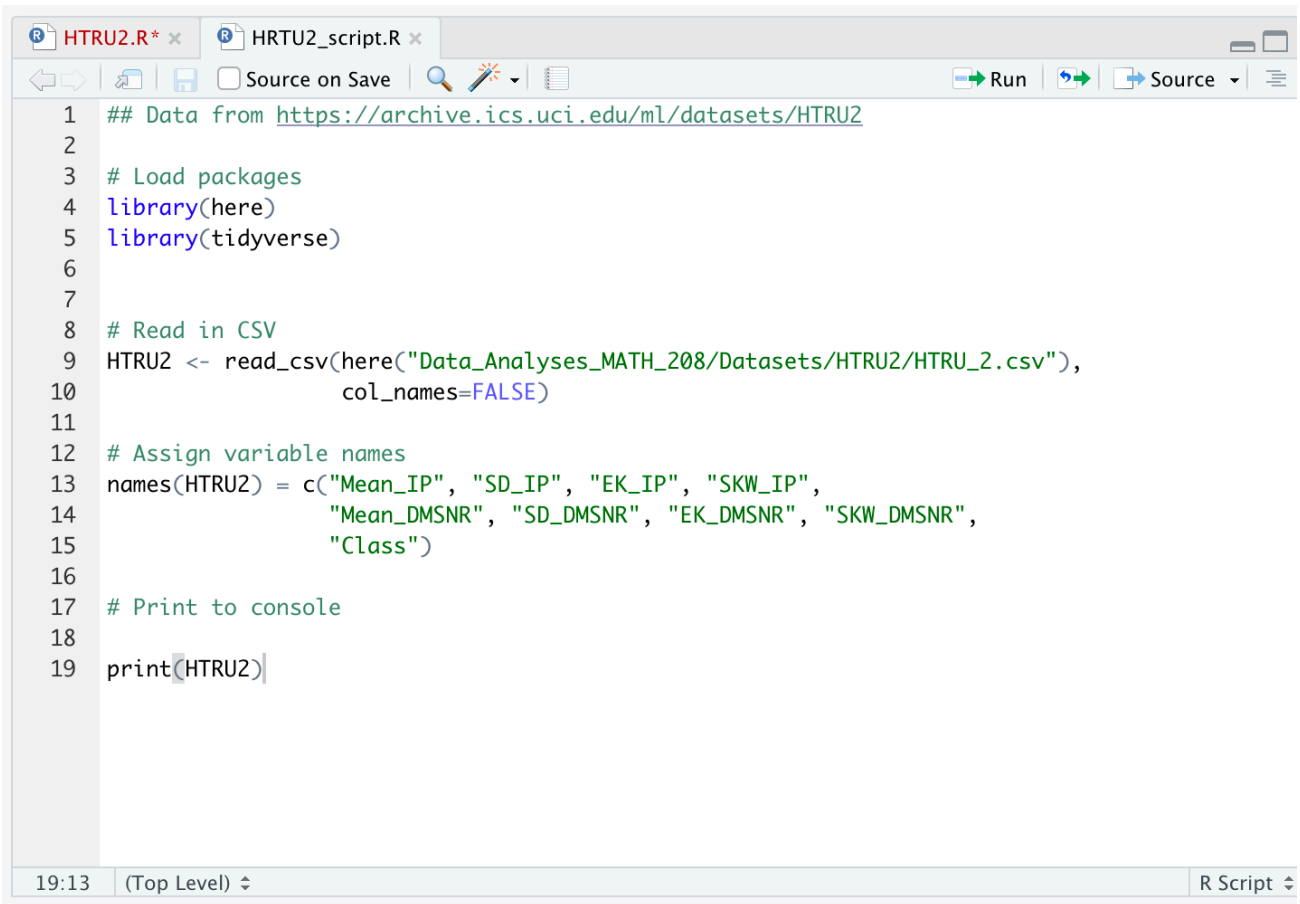


Downloaded from: https://archive.ics.uci.edu/ml/machine-learning-databases/00372/HTRU2.zip

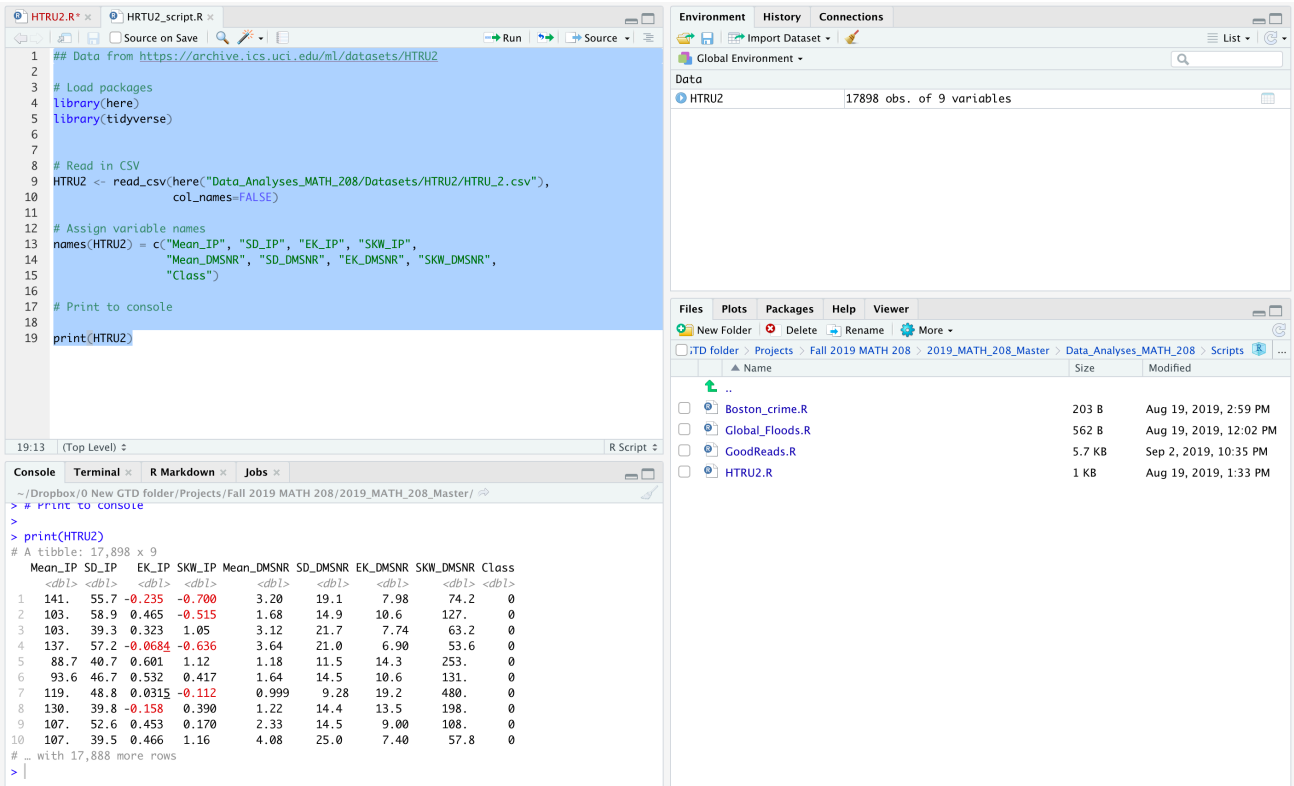
Where do we go from here?

- Get data into R and RStudio
- Create summaries of the data for true and false pulsars

First tip: work in scripts



Results from running code



What is HTRU2?

```
# Read in the CSV
HTRU2 <-
  read_csv(here("Data_Analyses_MATH_208_2020/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"
```

The basics

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
```

Indexing starts at one!

1

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"
```

Store information in objects

Different kinds of information require different modes

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

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

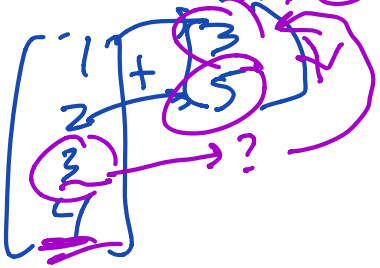
```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.795
[11] 40.945 44.094 46.457 49.213 51.969 53.150
```

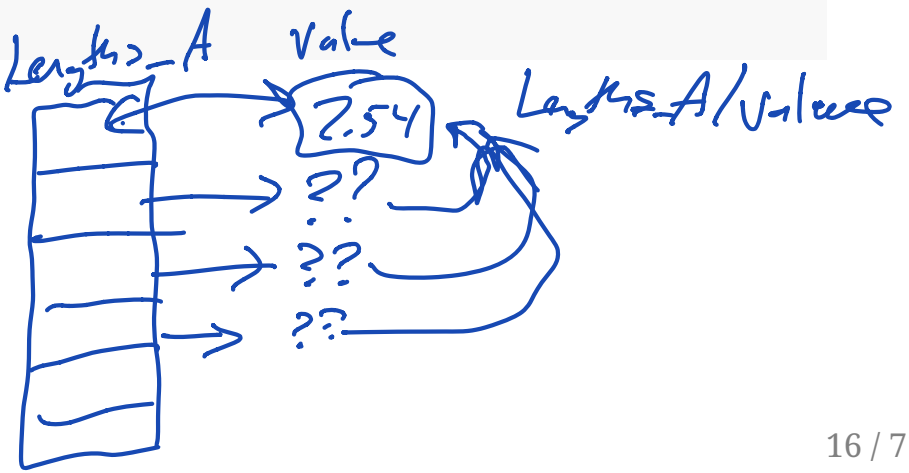
```
Lengths_A / 2.54
```

```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.795
[11] 40.945 44.094 46.457 49.213 51.969 53.150
```

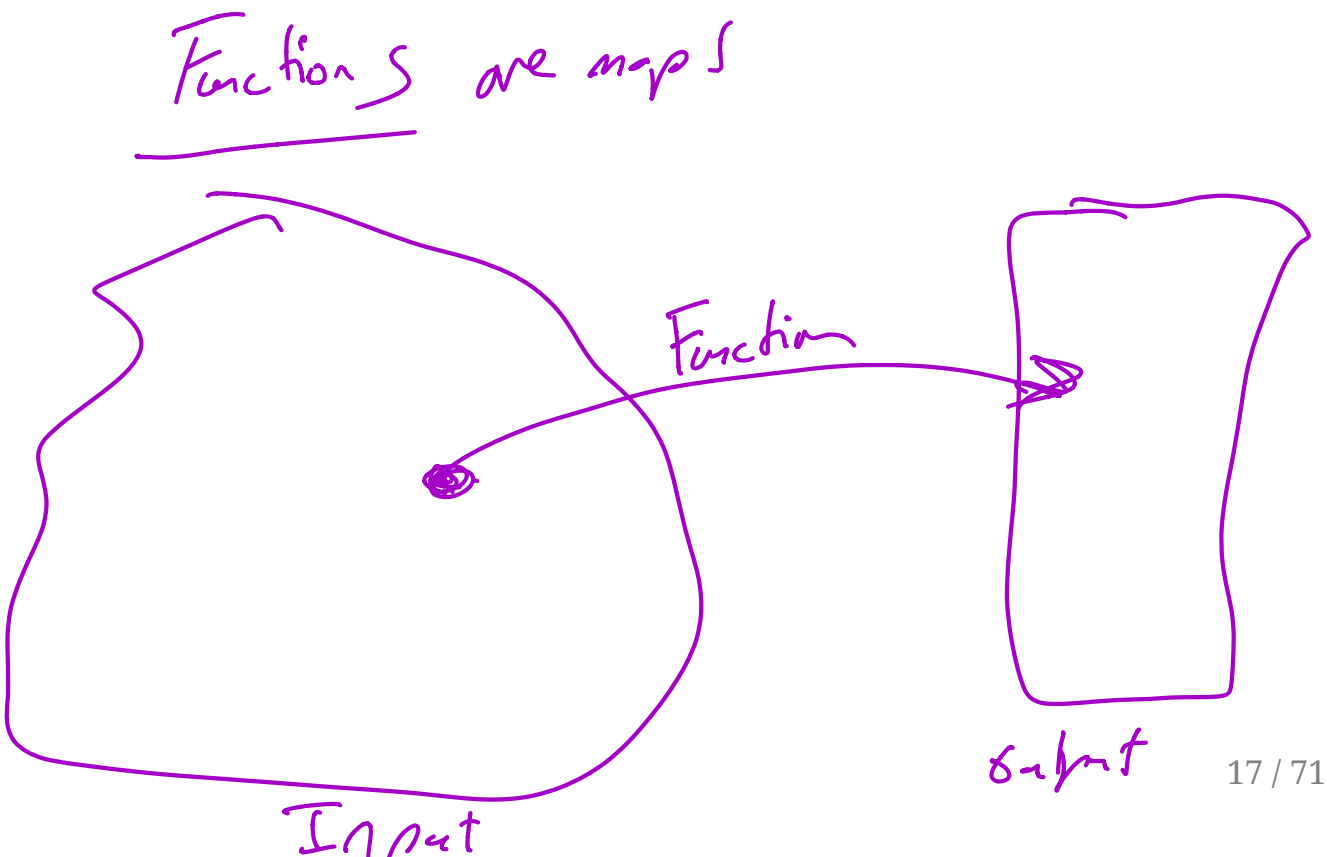
```
c(1,2,3,4)+c(3,5)
```

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





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

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
```

In R, function output (actions) depend on the kinds & numbers of arguments

Methods (S3 Methods, to be exact)

```
args(mean)
```

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

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

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

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

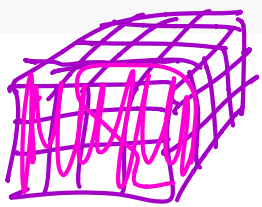
```
[1] mean.Date     mean.default    mean.difftime   mean.POSIXct
[5] mean.POSIXlt  mean.quosure*   mean.vctrs_vctr*
see '?methods' for accessing help and source code
```

Moving beyond atomic vectors

Generic vectors

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

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



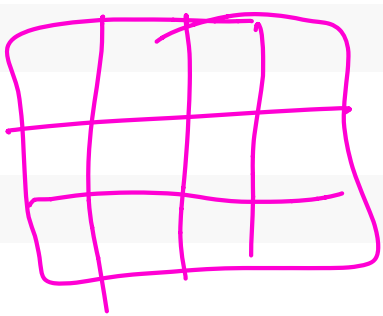
3D array

```
class(X)
```

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

```
mode(X)
```

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



Generic vectors

Generic vectors have attributes!

```
attributes(X)
```

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

```
dim(X)
```

```
[1] 3 3
```

Subscripting matrices

```
X
```

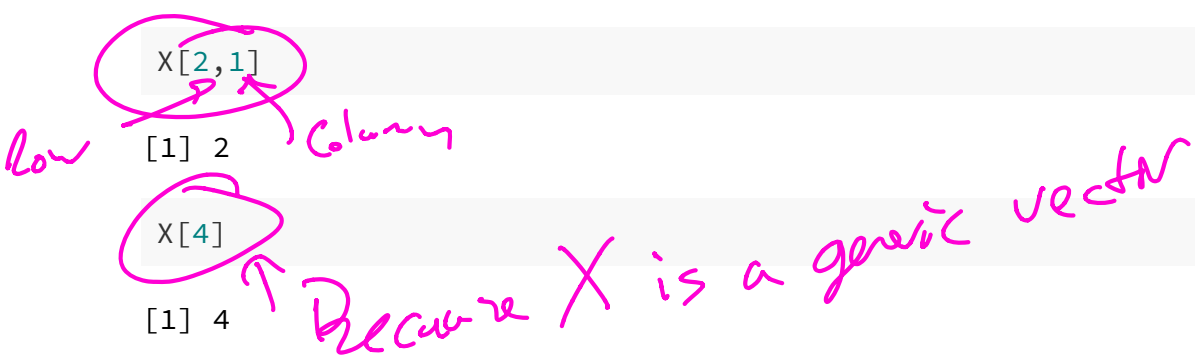
```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

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

```
[1] 2
```

```
X[4]
```

```
[1] 4
```



Operators for matrices

matrix multiplication

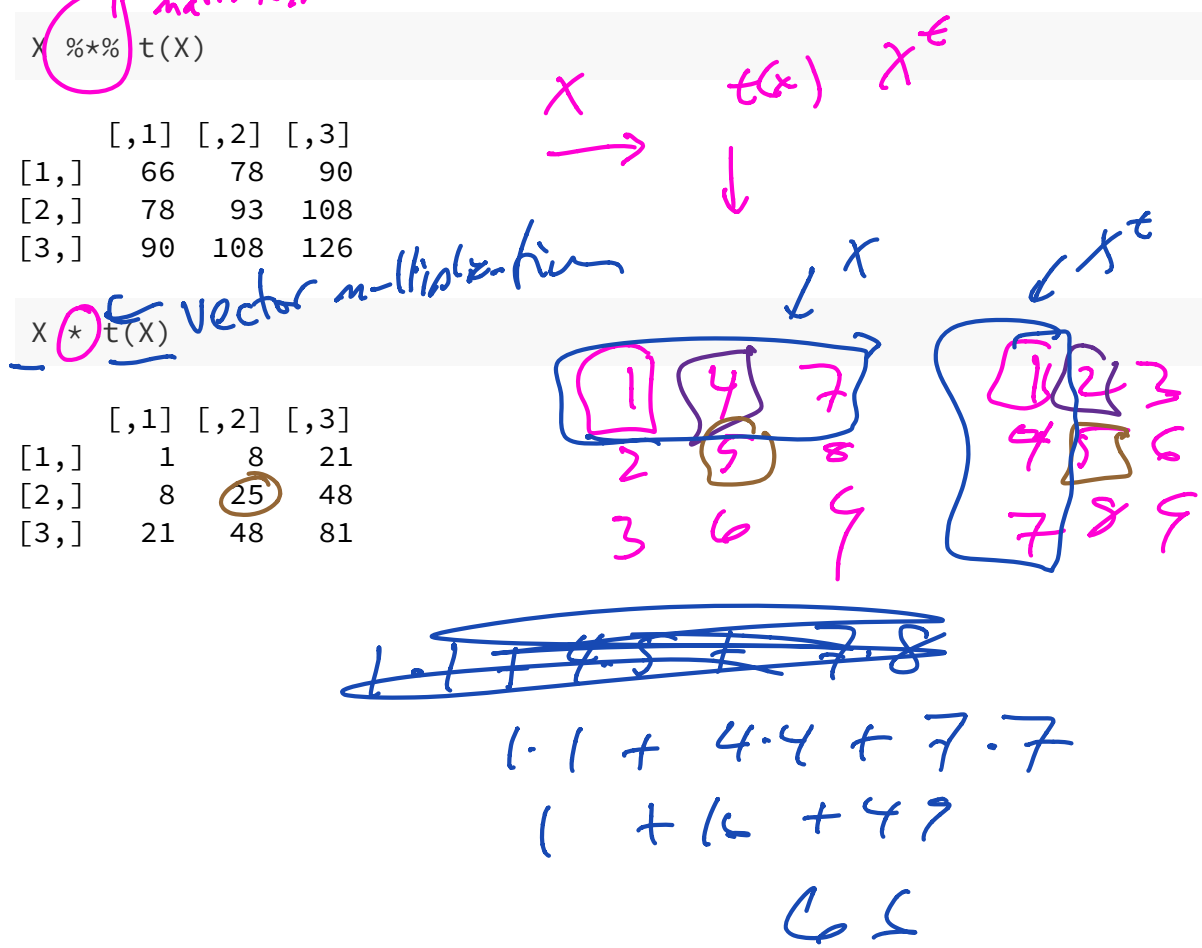
```
X %*% t(X)
```

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

vector multiplication

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

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



First Quiz will be posted tomorrow.

First Assignment will be given out next week.

Need list of dates

Lists

The point of a list is to allow different nodes and types of data inside a single object

Problem: how do we { put stuff in
                     { take stuff out

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

vectors that are the elements of my list

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

mymcgill_stats
```

element of list

names to list elements

```
$U1
[1] 203 204

$U2
[1] 323 324 447

$U3
[1] 208 427 423 523 545

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

```
mymcgill_stats[['U2']]
```

go to element named "U2"
return that element

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

$ allows me to avoid using [[]]'s

Comparing [[]], [], $

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

```
mymcgill_stats$U2
```

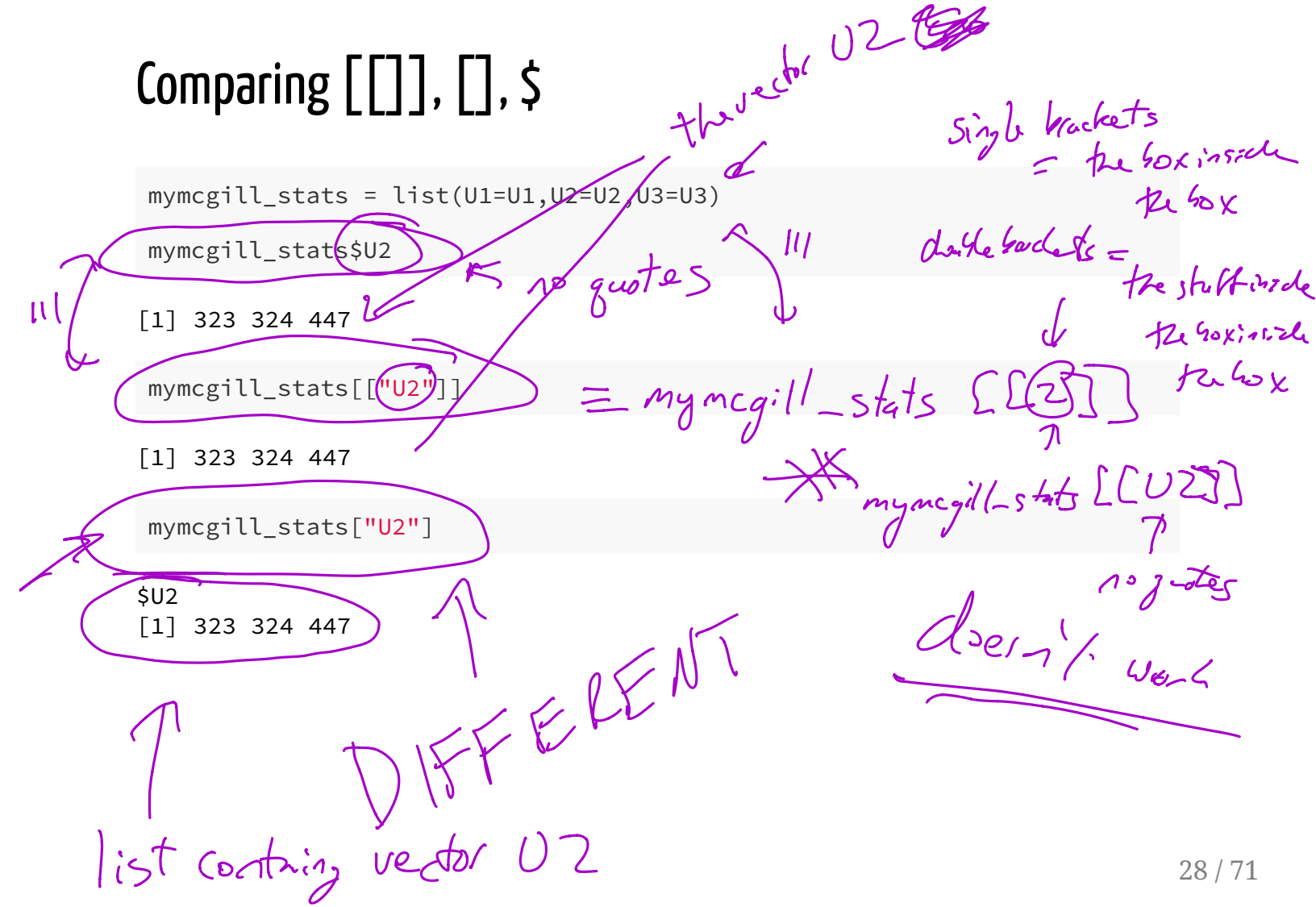
```
[1] 323 324 447
```

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

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

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

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



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

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

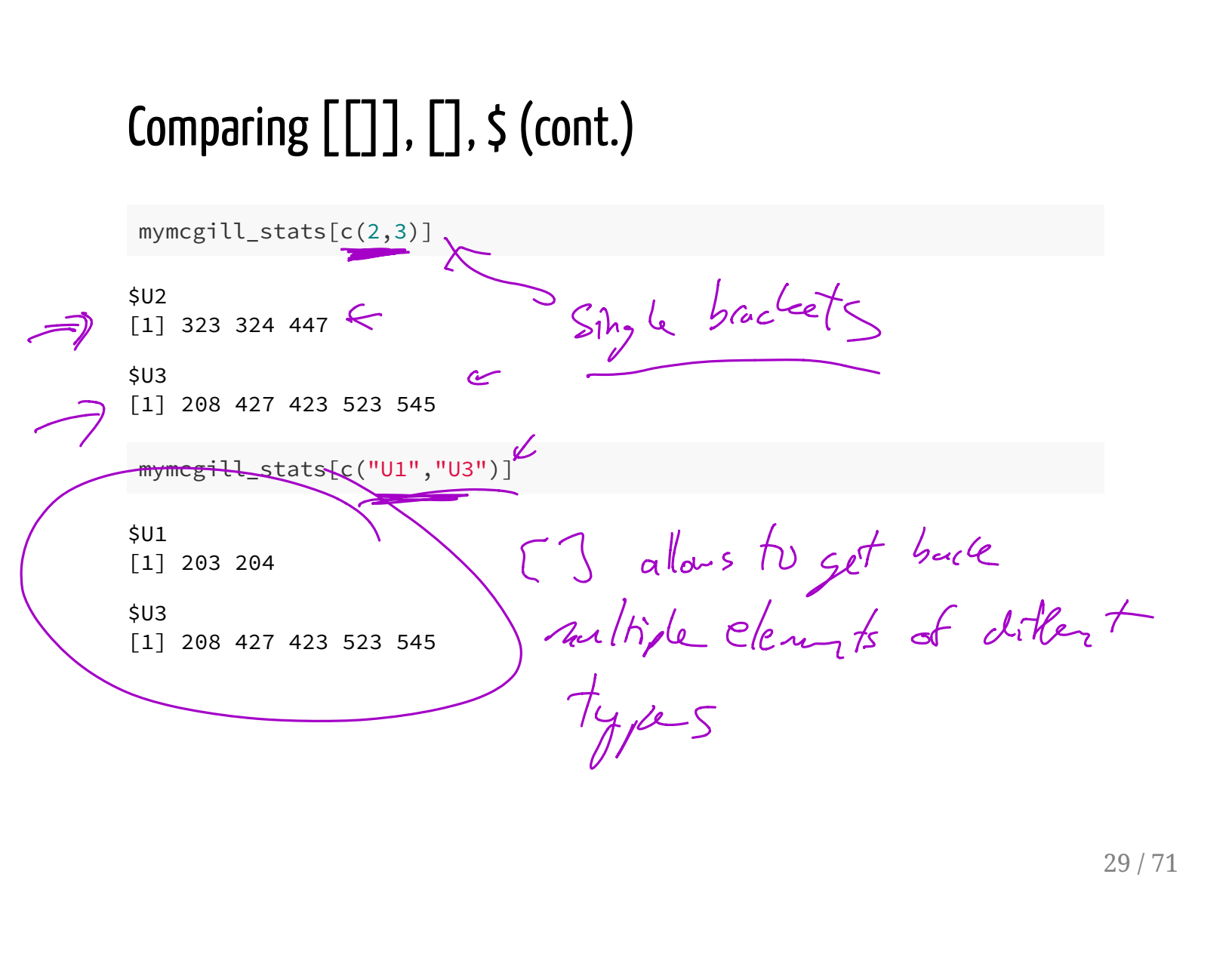
```
$U2
[1] 323 324 447

$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
```

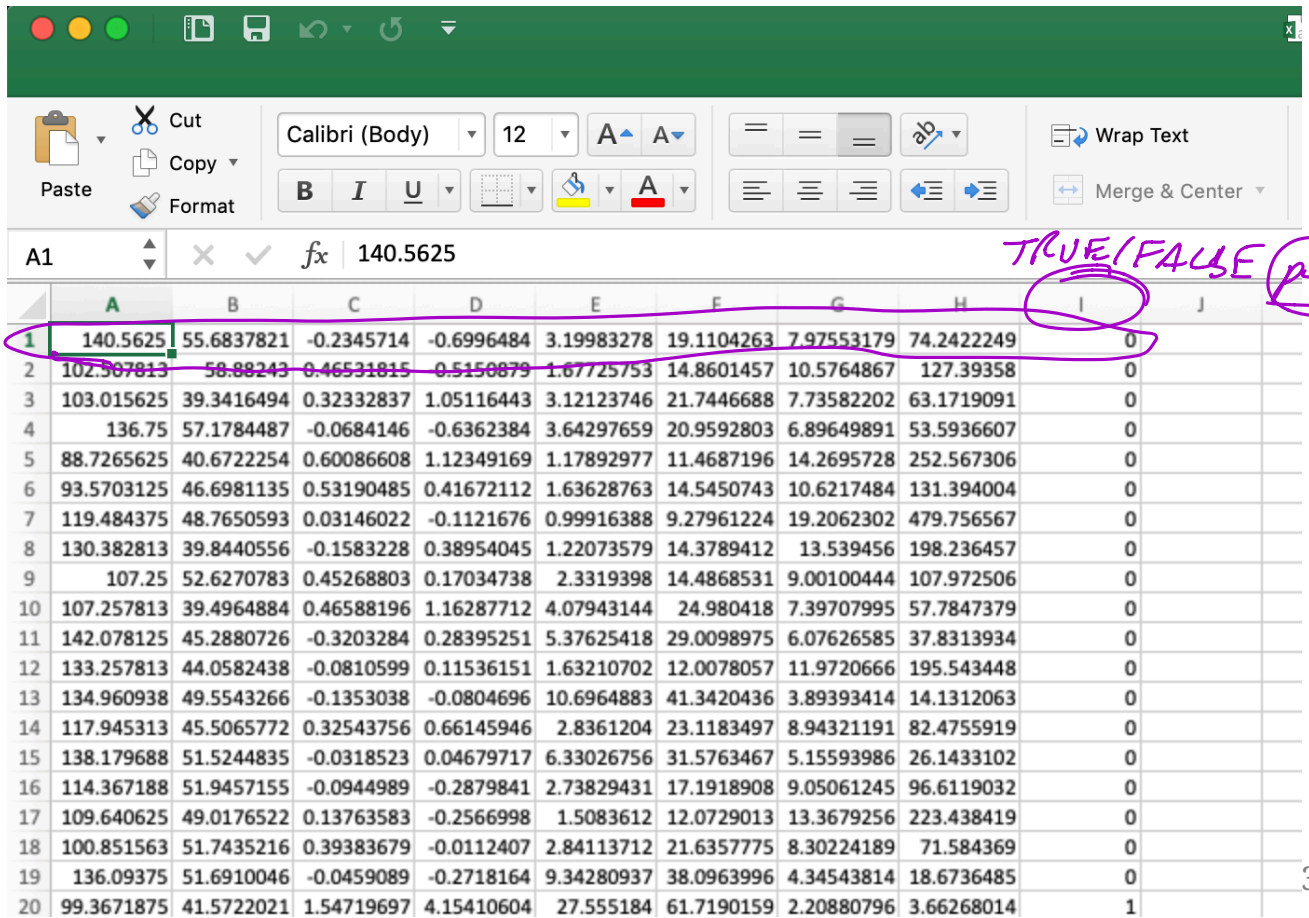
```
[1] 204
```

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

```
[1] 204
```

How do we think about datasets usually?

Like this



	A	B	C	D	E	F	G	H	I	J
1	140.5625	55.6837821	-0.2345714	-0.6996484	3.19983278	19.1104263	7.97553179	74.2422249	0	
2	102.507813	58.88243	0.46531815	-0.5150879	1.67725753	14.8601457	10.5764867	127.39358	0	
3	103.015625	39.3416494	0.32332837	1.05116443	3.12123746	21.7446688	7.73582202	63.1719091	0	
4	136.75	57.1784487	-0.0684146	-0.6362384	3.64297659	20.9592803	6.89649891	53.5936607	0	
5	88.7265625	40.6722254	0.60086608	1.12349169	1.17892977	11.4687196	14.2695728	252.567306	0	
6	93.5703125	46.6981135	0.53190485	0.41672112	1.63628763	14.5450743	10.6217484	131.394004	0	
7	119.484375	48.7650593	0.03146022	-0.1121676	0.99916388	9.27961224	19.2062302	479.756567	0	
8	130.382813	39.8440556	-0.1583228	0.38954045	1.22073579	14.3789412	13.539456	198.236457	0	
9	107.25	52.6270783	0.45268803	0.17034738	2.3319398	14.4868531	9.00100444	107.972506	0	
10	107.257813	39.4964884	0.46588196	1.16287712	4.07943144	24.980418	7.39707995	57.7847379	0	
11	142.078125	45.2880726	-0.3203284	0.28395251	5.37625418	29.0098975	6.07626585	37.8313934	0	
12	133.257813	44.0582438	-0.0810599	0.11536151	1.63210702	12.0078057	11.9720666	195.543448	0	
13	134.960938	49.5543266	-0.1353038	-0.0804696	10.6964883	41.3420436	3.89393414	14.1312063	0	
14	117.945313	45.5065772	0.32543756	0.66145946	2.8361204	23.1183497	8.94321191	82.4755919	0	
15	138.179688	51.5244835	-0.0318523	0.04679717	6.33026756	31.5763467	5.15593986	26.1433102	0	
16	114.367188	51.9457155	-0.0944989	-0.2879841	2.73829431	17.1918908	9.05061245	96.6119032	0	
17	109.640625	49.0176522	0.13763583	-0.2566998	1.5083612	12.0729013	13.3679256	223.438419	0	
18	100.851563	51.7435216	0.39383679	-0.0112407	2.84113712	21.6357775	8.30224189	71.584369	0	
19	136.09375	51.6910046	-0.0459089	-0.2718164	9.34280937	38.0963996	4.34543814	18.6736485	0	
20	99.3671875	41.5722021	1.54719697	4.15410604	27.555184	61.7190159	2.20880796	3.66268014	1	

data.frame

is a list

```
htru2_df <- read.csv(
  here("Data_Analyses_MATH_208_2020/Datasets/HTRU2/HTRU_2.csv"),
  header=FALSE)
class(htru2_df)
```

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

shows first 6 rows of a data.frame

```
head(htru2_df)
```

```
      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
```

Each column is an element of a list

tibble

is a data.frame which means it is also a list

```
library(tidyverse)
htru2_tbl = read_csv(
  here("Data_Analyses_MATH_208_2020/Datasets/HTRU2/HTRU_2.csv"),
  col_names =FALSE)
class(htru2_tbl)
```

underscore

```
[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 (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
  <named 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 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
```

Subsetting

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  TRUE
[13]  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
```

```
$U1
[1] 2

$U2
[1] 3

$U3
[1] 5
```

```
                              ## element of list
```

```
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]
```

```
[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]
```

```
[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_DMSNR Class
1  140.56 55.684 -0.23457 -0.69965     3.1998   19.110   7.9755    74.242     0
2  102.51 58.882  0.46532 -0.51509     1.6773   14.860  10.5765   127.394     0
3  103.02 39.342  0.32333  1.05116     3.1212   21.745   7.7358    63.172     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_MIP14_sub)
```

```
   Mean_IP  SD_IP    EK_IP   SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSNR
1   140.56 55.684 -0.23457 -0.69965     3.1998   19.110   7.9755    74.242
11  142.08 45.288 -0.32033  0.28395     5.3763   29.010   6.0763    37.831
34  142.05 53.873 -0.47077 -0.12595     4.4231   27.084   6.6817    45.944
37  147.84 53.623 -0.13108 -0.28885     2.6923   17.081   8.8492    92.202
59  141.97 50.471  0.24497 -0.34266     2.8236   16.238   8.2077    85.533
67  143.09 49.922 -0.15756 -0.15333     3.5635   21.288   7.3371    59.168
   Class
1      0
11     0
34     0
37     0
59     0
67     0
```

```
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
```

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))
```

```
# 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)
```

```
# A tibble: 3 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
```

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
```

Subsetting data with tibbles

```
slice(select(filter(htru2_tbl,Mean_IP>140), Mean_IP,SD_IP),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
```

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

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

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")

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
```

```
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?

```
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 = 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 = 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
```

How to create a dataframe or tibble?

```
str(kids_heights_tbl)
```

```
tibble [39 × 3] (S3: tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
```

How to create a dataframe or tibble?

```
glimpse(kids_heights_tbl)
```

```
Rows: 39
Columns: 3
$ Kid     <chr> "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "…
$ Heights <dbl> 53.0, 51.0, 60.0, 64.0, 69.0, 74.0, 78.0, 84.0, 86.0, 96.0, 1…
$ Dates   <date> 2009-05-02, 2009-05-11, 2009-07-07, 2009-09-09, 2009-11-16, …
```

Grouping by a variable

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

```
tibble [39 × 3] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
 - attr(*, "groups")= tibble [3 × 2] (S3: tbl_df/tbl/data.frame)
  ..$ Kid  : chr [1:3] "A" "B" "C"
  ..$ .rows: list<int> [1:3]
  .. ..$ : int [1:16] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int [1:13] 17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int [1:10] 30 31 32 33 34 35 36 37 38 39
  .. ..@ ptype: int(0)
  ..- 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 magrittr

Using %>% and magrittr

> ---The Treachery of Images, 1929, René Magritte

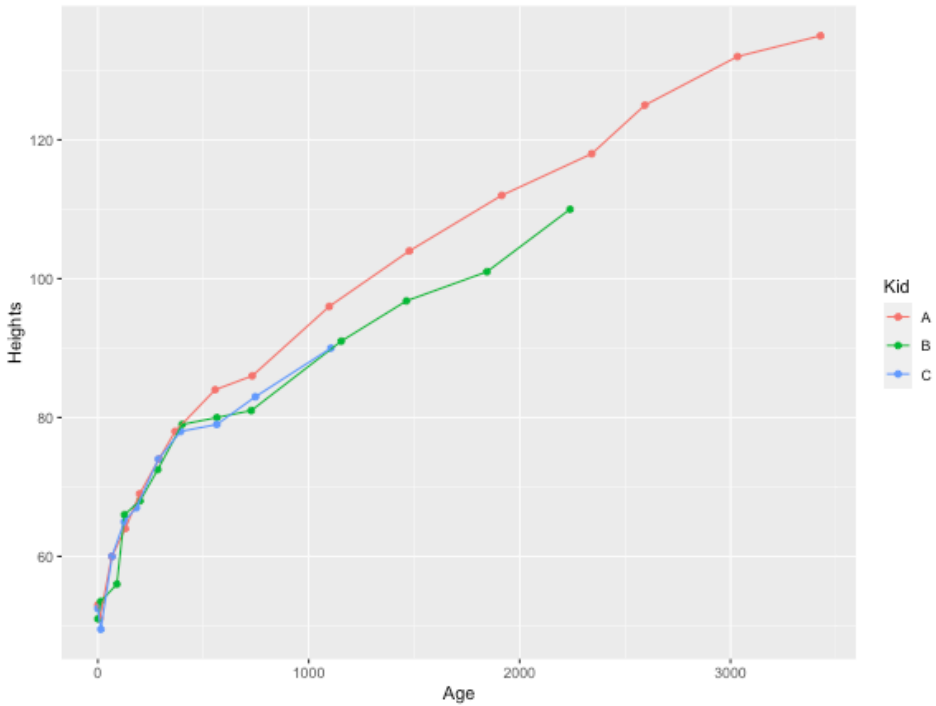
Image taken from https://en.wikipedia.org/wiki/The_Treachery_of_Images

Using %>% and magrittr

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

```
tibble [39 × 5] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
 $ Birth  : Date[1:39], format: "2009-05-02" "2009-05-02" ...
 $ Age    : 'difftime' num [1:39] 0 9 66 130 ...
  ..- attr(*, "units")= chr "days"
 - attr(*, "groups")= tibble [3 × 2] (S3: tbl_df/tbl/data.frame)
  ..$ Kid  : chr [1:3] "A" "B" "C"
  ..$ .rows: list<int> [1:3]
  .. ..$ : int [1:16] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int [1:13] 17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int [1:10] 30 31 32 33 34 35 36 37 38 39
  .. ..@ ptype: int(0)
  ..- attr(*, ".drop")= logi TRUE
```

What's next?



Getting Data into R

Data structures

Why we're starting here

> "Bad programmers worry about the code. Good programmers worry about data structures and their relationships."
>
> --- Linus Torvalds

HTRU2

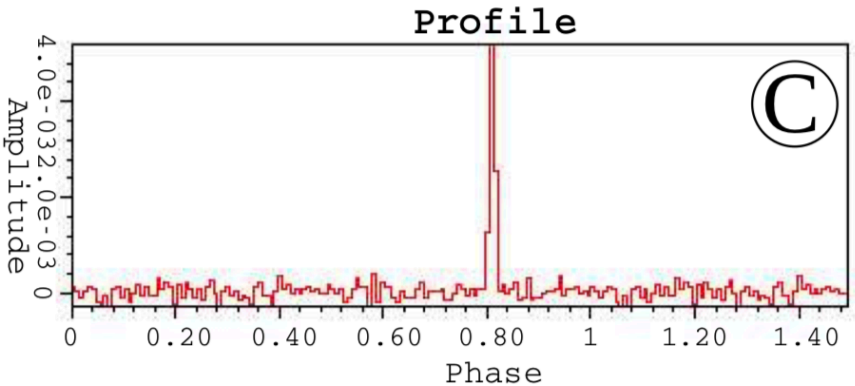
Monthly Notices of the Royal Astronomical Society / Volume 409, Issue 2

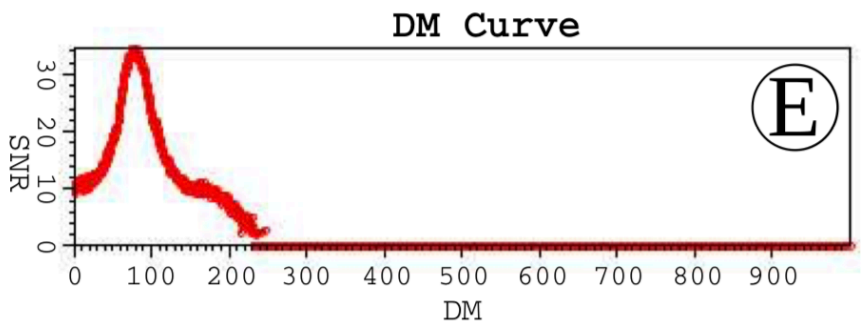
The High Time Resolution Universe Pulsar Survey – I. System configuration and initial discoveries

M. J. Keith ✉, A. Jameson, W. van Straten, M. Bailes, S. Johnston, M. Kramer, A. Possenti, S. D. Bates, N. D. R. Bhat, M. Burgay, S. Burke-Spolaor, N. D'Amico, L. Levin, Peter L. McMahon, S. Milia, B. W. Stappers

First published: 14 September 2010
https://doi.org/10.1111/j.1365-2966.2010.17325.x
Cited by: 1

Pulsar emission profiles from
https://arxiv.org/pdf/1603.05166.pdf


Profile
C
Amplitude
4.0e-03
2.0e-03
0
0
0.20
0.40
0.60
0.80
1
1.20
1.40
Phase


DM Curve
E
SNR
30
20
10
0
0
100
200
300
400
500
600
700
800
900
DM

The data

Measurements
1. Mean of the integrated profile
2. Standard deviation of the integrated profile
3. Excess kurtosis of the integrated profile
4. Skewness of the integrated profile
5. Mean of the DM-SNR curve
6. Standard deviation of the DM-SNR curve
7. Excess kurtosis of the DM-SNR curve
8. Skewness of the DM-SNR curve
9. True or false pulsar (human-verified)

The data file

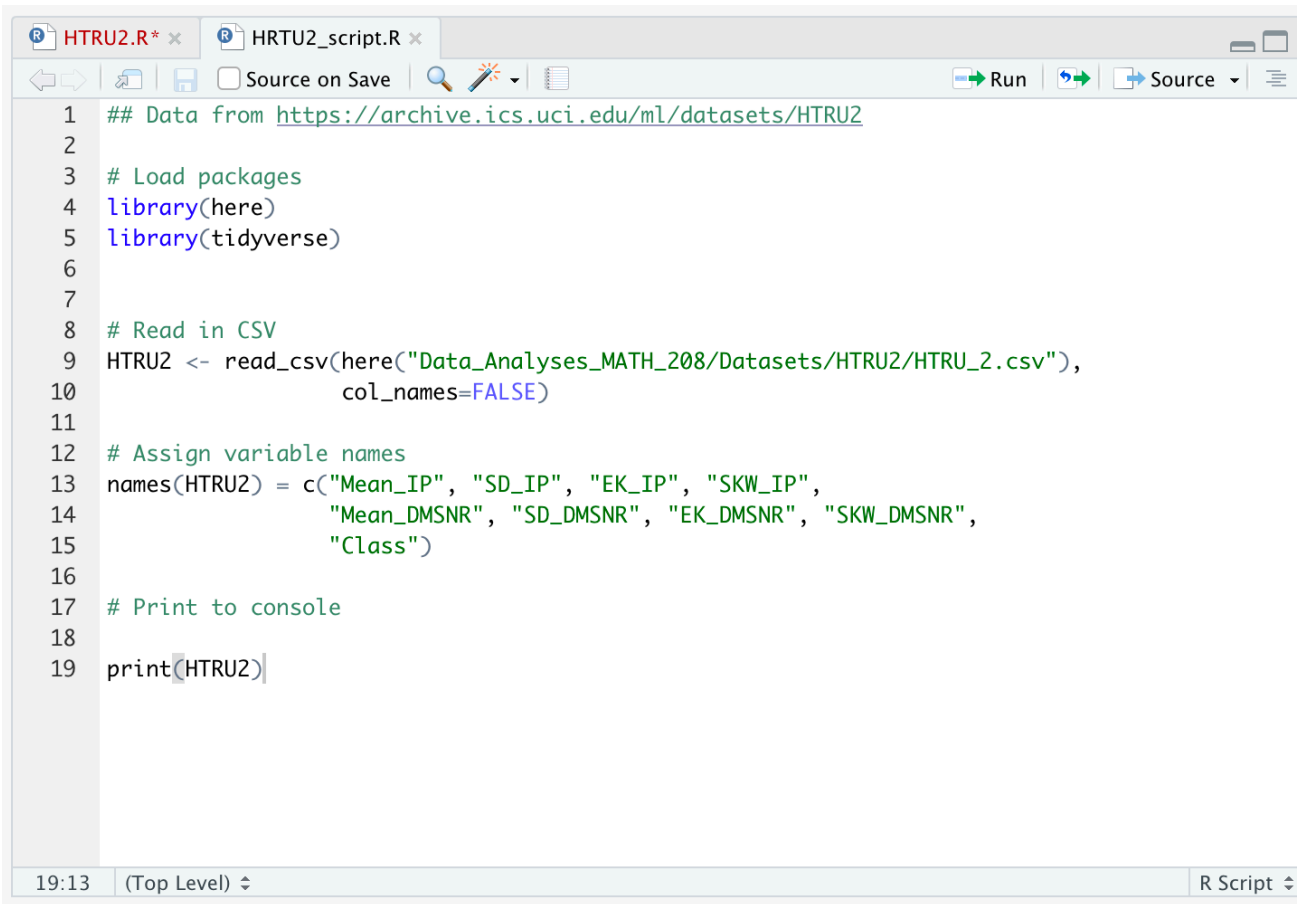
HTRU_2.csv

```
140.5625,55.68378214,-0.234571412,-0.699648398,3.199832776,19.11042633,7.975531794,74.24222492,0
102.5078125,58.88243001,0.465318154,-0.515087909,1.677257525,14.86014572,10.57648674,127.3935796,0
103.015625,39.34164944,0.323328365,1.051164429,3.121237458,21.74466875,7.735822015,63.17190911,0
136.75,57.17844874,-0.068414638,-0.636238369,3.642976589,20.9592803,6.89649891,53.59366067,0
88.7265625,40.67222541,0.600866079,1.123491692,1.178929766,11.4687196,14.26957284,252.5673058,0
93.5703125,46.69811352,0.53190485,0.416721117,1.636287625,14.54507425,10.6217484,131.3940043,0
119.484375,48.76505927,0.03146022,-0.112167573,0.99916388,9.279612239,19.20623018,479.7565669,0
130.3828125,39.84405561,-0.158322759,0.389540448,1.220735786,14.37894124,13.53945602,198.2364565,0
107.25,52.62707834,0.452688025,0.170347382,2.331939799,14.48685311,9.001004441,107.9725056,0
107.2578125,39.49648839,0.465881961,1.162877124,4.079431438,24.98041798,7.397079948,57.78473789,0
142.078125,45.28807262,-0.320328426,0.283952506,5.376254181,29.00989748,6.076265849,37.83139335,0
133.2578125,44.05824378,-0.081059862,0.115361506,1.632107023,12.00780568,11.97206663,195.5434476,0
134.9609375,49.55432662,-0.135303833,-0.080469602,10.69648829,41.34204361,3.893934139,14.13120625,0
117.9453125,45.50657724,0.325437564,0.661459458,2.836120401,23.11834971,8.943211912,82.47559187,0
138.1796875,51.5244835,-0.031852329,0.046797173,6.330267559,31.57634673,5.155939859,26.14331017,0
114.3671875,51.94571552,-0.094498904,-0.287984087,2.738294314,17.19189079,9.050612454,96.61190318,0
109.640625,49.01765217,0.13763583,-0.256699775,1.508361204,12.07290134,13.36792556,223.4384192,0
100.8515625,51.74352161,0.393836792,-0.011240741,2.841137124,21.63577754,8.302241891,71.58436903,0
136.09375,51.69100464,-0.045908926,-0.271816393,9.342809365,38.09639955,4.345438138,18.67364854,0
99.3671875,41.57220208,1.547196967,4.154106043,27.55518395,61.71901588,2.20880796,3.662680136,1
100.890625,51.89039446,0.627486528,-0.026497802,3.883779264,23.04526673,6.953167635,52.27944038,0
105.4453125,41.13996851,0.142653801,0.320419676,3.551839465,20.75501684,7.739552295,68.51977061,0
95.8671875,42.05992212,0.326386917,0.803501794,1.83277592,12.24896949,11.249331,177.2307712,0
117.3671875,53.90861351,0.257953441,-0.405049077,6.018394649,24.76612335,4.807783224,25.52261561,0
106.6484375,56.36718209,0.378355072,-0.266371607,2.43645485,18.40537062,9.378659682,96.86022536,0
112.71875,50.3012701,0.279390953,-0.129010712,8.281772575,37.81001224,4.691826852,21.27620977,0
130.8515625,52.43285734,0.142596727,0.018885442,2.64632107,15.65443599,9.464164025,115.6731586,0
119.4375,52.87481531,-0.002549267,-0.460360287,2.365384615,16.49803188,9.008351898,94.75565692,0
123.2109375,51.07801208,0.179376819,-0.17728516,2.107023411,16.92177312,10.08033334,112.5585913,0
102.6171875,49.69235371,0.230438984,0.193325371,1.489130435,16.00441146,12.64653474,171.8329021,0
110.109375,41.31816988,0.094860398,0.68311261,1.010033445,13.02627521,14.66651082,231.2041363,0
99.9140625,43.91949797,0.475728501,0.781486196,0.619565217,9.440975862,20.1066391,475.680218,0
128.34375,52.17210664,-0.049280401,-0.208256987,2.173913043,12.9939472,9.965757364,141.5100843,0
142.0546875,53.87315957,-0.470772686,-0.125946417,4.423076923,27.08351266,6.681658306,45.94403008,0
```

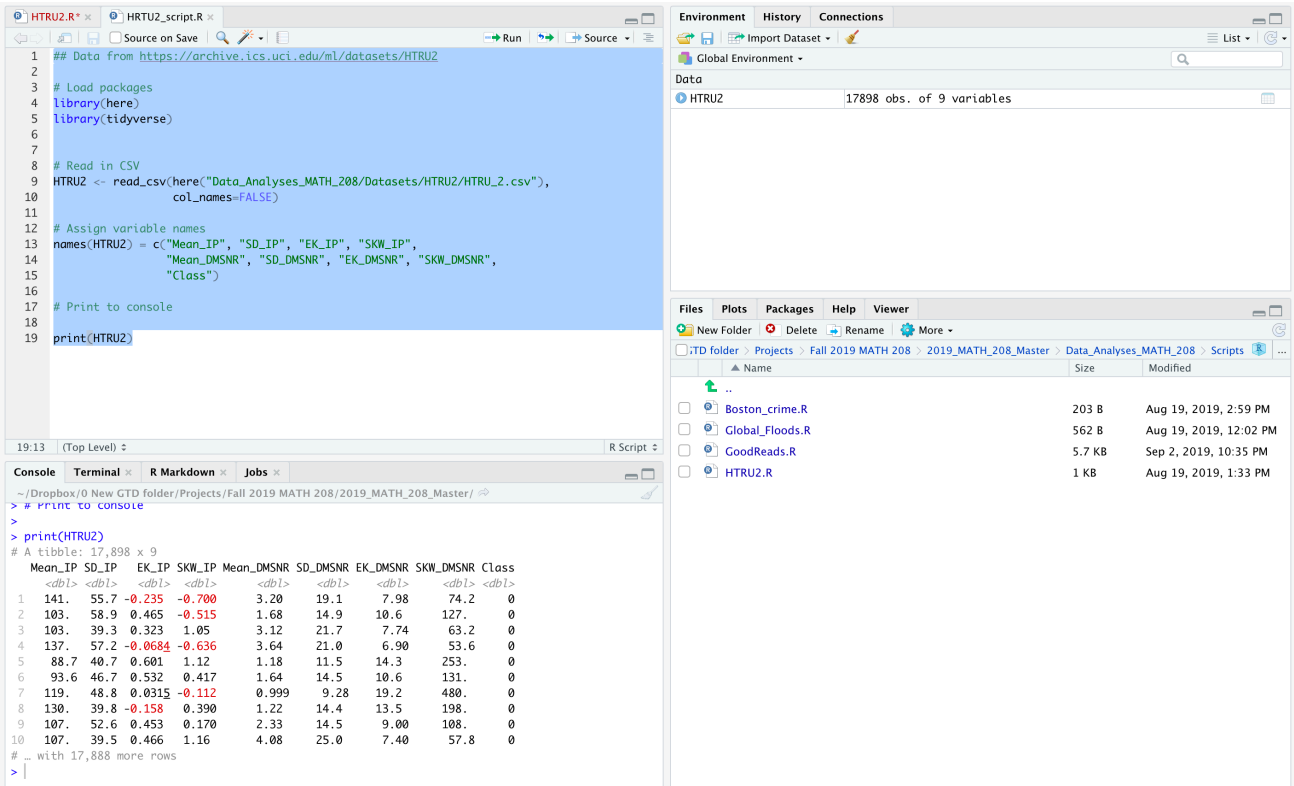
Downloaded from: https://archive.ics.uci.edu/ml/machine-learning-databases/00372/HTRU2.zip

Where do we go from here?

First tip: work in scripts



Results from running code



What is HTRU2?

```
# Read in the CSV
HTRU2 <-
  read_csv(here("Data_Analyses_MATH_208_2020/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"
```

The basics

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

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

```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.795
[11] 40.945 44.094 46.457 49.213 51.969 53.150
```

```
Lengths_A / 2.54
```

```
 [1] 20.866 20.079 23.622 25.197 27.165 29.134 30.709 33.071 33.858 37.795
[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
```

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

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 (S3 Methods, to be exact)

```
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
[5] mean.POSIXlt      mean.quosure*     mean.vctrs_vctr*
see '?methods' for accessing help and source code
```

Moving beyond atomic vectors

Generic vectors

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

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

```
class(X)
```

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

```
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
```

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

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

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

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

```
$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
```

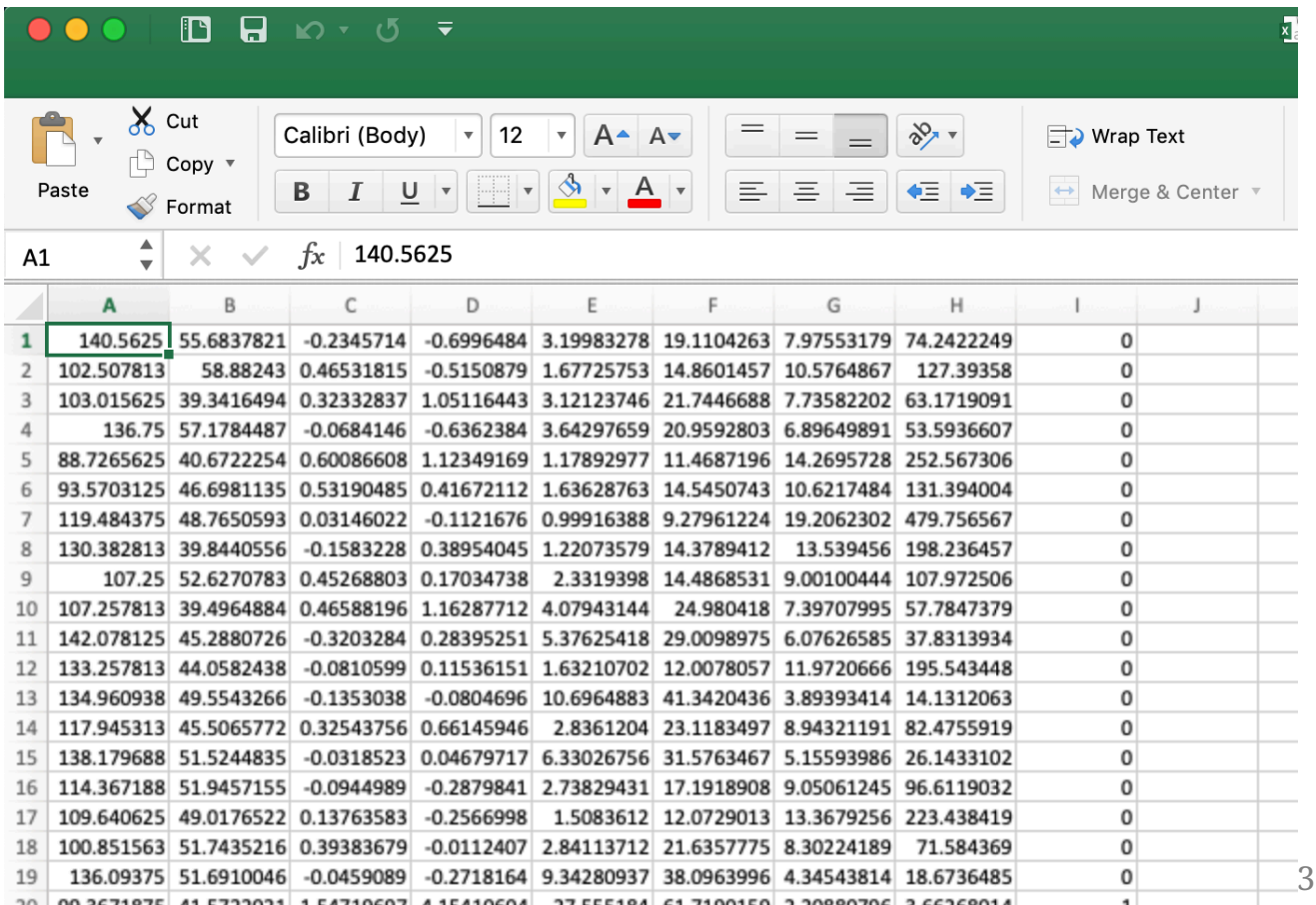
```
[1] 204
```

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

```
[1] 204
```

How do we think about datasets usually?

Like this



	A	B	C	D	E	F	G	H	I	J
1	140.5625	55.6837821	-0.2345714	-0.6996484	3.19983278	19.1104263	7.97553179	74.2422249	0	
2	102.507813	58.88243	0.46531815	-0.5150879	1.67725753	14.8601457	10.5764867	127.39358	0	
3	103.015625	39.3416494	0.32332837	1.05116443	3.12123746	21.7446688	7.73582202	63.1719091	0	
4	136.75	57.1784487	-0.0684146	-0.6362384	3.64297659	20.9592803	6.89649891	53.5936607	0	
5	88.7265625	40.6722254	0.60086608	1.12349169	1.17892977	11.4687196	14.2695728	252.567306	0	
6	93.5703125	46.6981135	0.53190485	0.41672112	1.63628763	14.5450743	10.6217484	131.394004	0	
7	119.484375	48.7650593	0.03146022	-0.1121676	0.99916388	9.27961224	19.2062302	479.756567	0	
8	130.382813	39.8440556	-0.1583228	0.38954045	1.22073579	14.3789412	13.539456	198.236457	0	
9	107.25	52.6270783	0.45268803	0.17034738	2.3319398	14.4868531	9.00100444	107.972506	0	
10	107.257813	39.4964884	0.46588196	1.16287712	4.07943144	24.980418	7.39707995	57.7847379	0	
11	142.078125	45.2880726	-0.3203284	0.28395251	5.37625418	29.0098975	6.07626585	37.8313934	0	
12	133.257813	44.0582438	-0.0810599	0.11536151	1.63210702	12.0078057	11.9720666	195.543448	0	
13	134.960938	49.5543266	-0.1353038	-0.0804696	10.6964883	41.3420436	3.89393414	14.1312063	0	
14	117.945313	45.5065772	0.32543756	0.66145946	2.8361204	23.1183497	8.94321191	82.4755919	0	
15	138.179688	51.5244835	-0.0318523	0.04679717	6.33026756	31.5763467	5.15593986	26.1433102	0	
16	114.367188	51.9457155	-0.0944989	-0.2879841	2.73829431	17.1918908	9.05061245	96.6119032	0	
17	109.640625	49.0176522	0.13763583	-0.2566998	1.5083612	12.0729013	13.3679256	223.438419	0	
18	100.851563	51.7435216	0.39383679	-0.0112407	2.84113712	21.6357775	8.30224189	71.584369	0	
19	136.09375	51.6910046	-0.0459089	-0.2718164	9.34280937	38.0963996	4.34543814	18.6736485	0	
20	99.3671875	41.5722021	1.54719697	4.15410604	27.555184	61.7190159	2.20880796	3.66268014	1	

data.frame

```
htru2_df <- read.csv(
  here("Data_Analyses_MATH_208_2020/Datasets/HTRU2/HTRU_2.csv"),
  header=FALSE)
class(htru2_df)
```

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

```
head(htru2_df)
```

```
       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
```

tibble

```
library(tidyverse)
htru2_tbl = read_csv(
  here("Data_Analyses_MATH_208_2020/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 (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
  <named 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 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
```

Subsetting

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  TRUE
[13]  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
```

```
$U1
[1] 2

$U2
[1] 3

$U3
[1] 5
```

```
                              ## element of list
```

```
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]
```

```
[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]
```

```
[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_DMSNR Class
1  140.56 55.684 -0.23457 -0.69965     3.1998   19.110   7.9755    74.242     0
2  102.51 58.882  0.46532 -0.51509     1.6773   14.860  10.5765   127.394     0
3  103.02 39.342  0.32333  1.05116     3.1212   21.745   7.7358    63.172     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_MIP14_sub)
```

```
   Mean_IP  SD_IP    EK_IP   SKW_IP Mean_DMSNR SD_DMSNR EK_DMSNR SKW_DMSNR
1   140.56 55.684 -0.23457 -0.69965     3.1998   19.110   7.9755    74.242
11  142.08 45.288 -0.32033  0.28395     5.3763   29.010   6.0763    37.831
34  142.05 53.873 -0.47077 -0.12595     4.4231   27.084   6.6817    45.944
37  147.84 53.623 -0.13108 -0.28885     2.6923   17.081   8.8492    92.202
59  141.97 50.471  0.24497 -0.34266     2.8236   16.238   8.2077    85.533
67  143.09 49.922 -0.15756 -0.15333     3.5635   21.288   7.3371    59.168
   Class
1      0
11     0
34     0
37     0
59     0
67     0
```

```
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
```

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))
```

```
# 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)
```

```
# A tibble: 3 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
```

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
```

Subsetting data with tibbles

```
slice(select(filter(htru2_tbl,Mean_IP>140), Mean_IP,SD_IP),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
```

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

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

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")

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
```

```
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?

```
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 = 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 = 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
```

How to create a dataframe or tibble?

```
str(kids_heights_tbl)
```

```
tibble [39 × 3] (S3: tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
```

How to create a dataframe or tibble?

```
glimpse(kids_heights_tbl)
```

```
Rows: 39
Columns: 3
$ Kid     <chr> "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "…
$ Heights <dbl> 53.0, 51.0, 60.0, 64.0, 69.0, 74.0, 78.0, 84.0, 86.0, 96.0, 1…
$ Dates   <date> 2009-05-02, 2009-05-11, 2009-07-07, 2009-09-09, 2009-11-16, …
```

Grouping by a variable

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

```
tibble [39 × 3] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
 - attr(*, "groups")= tibble [3 × 2] (S3: tbl_df/tbl/data.frame)
  ..$ Kid  : chr [1:3] "A" "B" "C"
  ..$ .rows: list<int> [1:3]
  .. ..$ : int [1:16] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int [1:13] 17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int [1:10] 30 31 32 33 34 35 36 37 38 39
  .. ..@ ptype: int(0)
  ..- 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 magrittr

Using %>% and magrittr

> ---The Treachery of Images, 1929, René Magritte

Image taken from https://en.wikipedia.org/wiki/The_Treachery_of_Images

Using %>% and magrittr

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

```
tibble [39 × 5] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ Kid    : chr [1:39] "A" "A" "A" "A" ...
 $ Heights: num [1:39] 53 51 60 64 69 74 78 84 86 96 ...
 $ Dates  : Date[1:39], format: "2009-05-02" "2009-05-11" ...
 $ Birth  : Date[1:39], format: "2009-05-02" "2009-05-02" ...
 $ Age    : 'difftime' num [1:39] 0 9 66 130 ...
  ..- attr(*, "units")= chr "days"
 - attr(*, "groups")= tibble [3 × 2] (S3: tbl_df/tbl/data.frame)
  ..$ Kid  : chr [1:3] "A" "B" "C"
  ..$ .rows: list<int> [1:3]
  .. ..$ : int [1:16] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ : int [1:13] 17 18 19 20 21 22 23 24 25 26 ...
  .. ..$ : int [1:10] 30 31 32 33 34 35 36 37 38 39
  .. ..@ ptype: int(0)
  ..- attr(*, ".drop")= logi TRUE
```

What's next?