

R* and *RStudio

Math functions

Data sources

Setting up *R* and *R Studio*

**This only applies to your own computers!
Lab computers have both installed.**

Windows and Mac

R home page: <http://cran.r-project.org/>

Download and install R.

RStudio home page: <https://www.rstudio.com/products/rstudio/download/>

Download and install R Studio

Start R Studio

(Vista Users: Right-click the icon and choose Compatibility, run as administrator)

R and R Studio

On the Lab computers

- you can install packages (necessary!) - **every time**
Tools|Install Packages| *[type in name]*
- then you load the library using: **library(*name*)**
- **you can load data** from a USB or web
- **you can save text** from commands, etc. to a USB

On your computer

- you can install packages - **needed only once**
Tools|Install Packages| *[type in name]*
- then you load the library using: **library(*name*)**
- **you can load data** from local drives or web
- **you can save text** from commands, etc. to local drive

Start RStudio

The bottom left should show the console

try entering: (notice the **BOLD BLACK** text)

2 + 2 (or: **2+2**) and hit enter

[1] 4 <--- by convention, output will usually be black, not bolded in slides

Commands: the Console

The bottom left should show the Console

First commands:

try addition:

```
2 + 2
```

```
[1] 4
```

subtract: `2 - 2 # notice I can add a comment on this line`

multiply: `2 * 2; I can also add a semicolon at the end`

divide: `2 / 2`

raise to a power `23: 2^3 # OR: 2**3`

raise to a power `0.32: 0.3^2`

raise to a power `91/2 or 90.5: 9^1/2 ??? 9^(1/2) or 9^0.5`

raise to a power `34591/7 : 3459^(1/7) # [1] 3.203055`

Commands

```
# (notice the #) Comment: get an average of 5 numbers
```

```
# Parentheses are very important!
```

```
(1 + 3 + 2 + 12 + 8)/5
```

```
[1] 5.2
```

```
# watch parentheses!
```

```
(1 + 3 + 2 + 12 + 8/5
```

```
+
```

```
+ means you forgot something (usually a right parenthesis)
```

```
)
```

```
[1] 19.6
```

```
# nested parentheses
```

```
( (2 - 1)^2 + (1 - 3)^2 ) ^(1/2)
```

```
((2-1)^2+(1-3)^2)^(1/2)
```

```
( (2-1)^2 + (1-3)^2 ) ^(1/2)
```

Commands

```
# Errors need some interpretation
```

```
2^^2
```

```
Error: unexpected '^' in "2^^"
```

$(2^2)^2$

```
# 2 squared, then squared
```

```
# so it is 4 squared or 2 x 2 x 2 x 2 =  $2^{2+2} = 2^4 = 2^{2^2}$ 
```

```
2^2^2
```

```
[1] 16
```

When copying and pasting code...

CAUTION: Microsoft's "smart quotes" are dumb!!!!

```
install.packages("sqldf", dependencies=TRUE)
```

```
Error: unexpected input in "install.packages(""
```

Commands

- Command history: use the up arrow to get to a previous command
(may be several lines)
- then, use left/right first, then up/down within that command

```
#this is a multi-line set of commands
# assign a variable to a value: use a left arrow (<-)
# if copying and pasting, you may need a semi-colon at end of line
x <- 3 # or x = 3;
y <- 5 # or y = 5;
x * y;
[1] 15

x <- 20 # reassign x
x * y
[1] 15
```

You can also see the commands in the top right pane, on the History page
To re-enter lines, highlight them and click on "To Console"

Functions

Functions: name of function then parentheses

- example: Square root

```
sqrt(9)
```

```
[1] 3
```

Logs of numbers - default uses the natural log , base = e

```
log(90)
```

```
[1] 4.49981
```

#get the inverse, using exp (e to some power) #exact answer:

```
exp(4.49981)      exp(log(90))
```

```
[1] 90.00003
```

#what is e, though?

```
exp(1)
```

```
[1] 2.718282
```

hmm - where does e come from?

$$e = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \dots$$

Functions

Another function: log using base 10

Base 10 Logs - notice there are two numbers separated by a comma

```
log(100,10)
```

```
[1] 2
```

```
log(90,10)
```

```
[1] 1.954243
```

#get the inverse of a base 10 log

```
10^1.954243
```

```
[1] 90.00001
```

#exact answer:

```
10^(log(90,10))
```

Scientific notation: base number >1 and < 10, and power of 10

```
1.2413234234 / 2353463456756476578678
```

```
[1] 5.274454e-22
```

= 0.000000000000000000005274454

1234567890123456789012

111111111222

Commands

Use variable names easy to recognize and remember

```
ControlN <- 55  
TreatmentN <- 27
```

```
# or cmn and tmn?
```

But everything in R is case-sensitive!

cmn is not the same as **CMN** or **Cmn**

```
# some variable names are reserved for constants (pi)  
pi  
[1] 3.141593  
pi <- 5  
pi  
[1] 5 OOPS!!!!!!!!!!
```

You can also see the history in the top right pane

To re-enter lines, highlight them and click on "To Console"

Functions

Assign multiple numbers to list (a vector: one row of numbers)

- use the c function (combine)

NOTE: you will get nothing back unless you ask for it

(unless there is an error!)

```
x <- c(74, 122, 235, 111, 292)
```

```
# what did I do? Check:
```

```
x
```

```
[1] 74 122 235 111 292
```

```
# use the mean function
```

```
mean(x)
```

```
[1] 166.8
```

```
# which is equivalent to using two other functions
```

```
# sum adds the elements up; length counts the elements
```

```
sum(x) / length(x)
```

```
[1] 166.8
```

```
# vectors have an index for each value in it : note brackets [ ]
```

```
x[3]
```

```
[1] 235
```

Functions

Functions can act on vectors

x is now a vector, will add elements

x

```
[1] 74 122 235 111 292
```

x + x

```
[1] 148 244 470 222 584
```

2*x

```
[1] 148 244 470 222 584
```

sqrt(x)

```
[1] 8.602325 11.045361 15.329710 10.535654 17.088007
```

x-mean(x)

```
[1] -92.8 -44.8 68.2 -55.8 125.2
```

sum(x-mean(x)) # should be zero - and IS (rounding); a nested function

```
[1] -5.684342e-14
```

round(sum(x-mean(x)),10) # a nested function, rounding to 10 digits

```
[1] 0
```

Functions

Median: 50th percentile

Get the median of our vector

x

```
[1] 74 122 235 111 292
```

median(x)

```
[1] 122
```

get a statistical summary of our vector

summary(x)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
74.0	111.0	122.0	166.8	235.0	292.0

Built-in functions need parentheses:

mean(), median(), sum(), min(), max()

NA, NULL, NaN, Inf answers

NA is used for missing data

```
x[7] # only 5 items
```

```
[1] NA
```

NULL is used for an undefined action or undefined value

```
babies$ID # oops, case-sensitive ; babies$id will work
```

```
NULL
```

NaN means not a number, an undefined number

```
0/0
```

```
[1] NaN
```

Inf means infinity, usually division by zero

```
1/0
```

```
[1] Inf
```

Help!

You can get help with a question mark in front of something

```
?median
```

Also, the bottom right pane shows a number of things

One is a Help tab

Look up median (top right of pane)

- provides explanation of function and gives examples

```
x
```

```
[1] 74 122 235 111 292
```

```
median(x)
```

```
[1] 122
```

```
# get a statistical summary of our vector
```

```
summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
74.0	111.0	122.0	166.8	235.0	292.0

Data

an easy way to load data: type into console:

```
babies <- read.csv("http://math.mercyhurst.edu/~sousley/STAT_139/data/babies.csv", header=T)
```

top left pane shows datasets

babies

...

```
431 6101      5      1 1382      280  1 146      1  0 23  2 61 145      0 25  2 99 999      1 98      0  0      0
432 6112      5      1 1406      283  1 112      0  0 21  3 62 102      0 23  2 72 165      1  5      1  1      1
433 6114      5      1 1404      269  1 115      2  6 30  2 62 115      5 29  1 67 130      1  1      9 98      98
434 6120      5      1 1676      278  1 132      2  0 20  1 64 150      9 24  1 99 999      1  4      1  1      5
```

```
[ reached getOption("max.print") -- omitted 802 rows ]
```

get the first few records

head(babies)

```
      id plurality outcome date gestation sex  wt parity race age ed ht wtl drace dage ded dht dwt marital inc smoke time number
1  15          5      1 1411      284  1 120      1  8 27  5 62 100      8 31  5 65 110      1  1      0  0      0
2  20          5      1 1499      282  1 113      2  0 33  5 64 135      0 38  5 70 148      1  4      0  0      0
3  58          5      1 1576      279  1 128      1  0 28  2 64 115      5 32  1 99 999      1  2      1  1      1
4  61          5      1 1504      999  1 123      2  0 36  5 69 190      3 43  4 68 197      1  8      3  5      5
5  72          5      1 1425      282  1 108      1  0 23  5 67 125      0 24  5 99 999      1  1      1  1      5
6 100          5      1 1673      286  1 136      4  0 25  2 62  93      3 28  2 64 130      1  4      2  2      2
```

get the last few records

tail (babies)

Data

data.frame = table (rows and columns)

```
str(babies) # get the structure
```

```
'data.frame':    1236 obs. of  23 variables:
 $ id      : int  15 20 58 61 72 100 102 129 142 148 ...
 $ plurality: int   5 5 5 5 5 5 5 5 5 5 ...
 $ outcome : int   1 1 1 1 1 1 1 1 1 1 ...
 $ date    : int 1411 1499 1576 1504 1425 1673 1449 1562 1408 1568 ...
 $ gestdays: int  284 282 279 999 282 286 244 245 289 299 ...
 $ sex     : int   1 1 1 1 1 1 1 1 1 1 ...
 $ bwt     : int  120 113 128 123 108 136 138 132 120 143 ...
 $ parity  : int   1 2 1 2 1 4 4 2 3 3 ...
 $ mrace   : int   8 0 0 0 0 0 7 7 0 0 ...
 $ mage    : int   27 33 28 36 23 25 33 23 25 30 ...
 $ med     : int   5 5 2 5 5 2 2 1 4 5 ...
 $ mht     : int   62 64 64 69 67 62 62 65 62 66 ...
 $ mwt     : int  100 135 115 190 125 93 178 140 125 136 ...
 $ frace   : int   8 0 5 3 0 3 7 7 3 0 ...
 $ fage    : int   31 38 32 43 24 28 37 23 26 34 ...
 $ fed     : int   5 5 1 4 5 2 4 4 1 5 ...
 $ fht     : int   65 70 99 68 99 64 99 71 70 99 ...
 $ fwt     : int  110 148 999 197 999 130 999 192 180 999 ...
 $ marital : int   1 1 1 1 1 1 1 1 0 1 ...
 $ inc     : int   1 4 2 8 1 4 98 2 2 2 ...
 $ msmove  : int   0 0 1 3 1 2 0 0 0 1 ...
 $ time    : int   0 0 1 5 1 2 0 0 0 1 ...
 $ number  : int   0 0 1 5 5 2 0 0 0 4 ...
```

Dataframes

Data arranged in rows and columns

`data(babies)` # notice top right pane

Next, load Cavendish data (estimates of gravitational constant from 1798):

```
Cavendish <- read.csv("http://math.mercyhurst.edu/~sousley/STAT_139/data/Cavendish.csv", header=T)
```

```
head(Cavendish)
```

	density	density2	density3
1	5.50	5.50	NA
2	5.61	5.61	NA
3	4.88	5.88	NA
4	5.07	5.07	NA
5	5.26	5.26	NA
6	5.55	5.55	NA

```
str(Cavendish)
```

```
'data.frame': 29 obs. of 3 variables:
```

```
$ density : num  5.5 5.61 4.88 5.07 5.26 5.55 5.36 5.29 5.58 5.65 ...  
$ density2: num  5.5 5.61 5.88 5.07 5.26 5.55 5.36 5.29 5.58 5.65 ...  
$ density3: num  NA NA NA NA NA NA 5.36 5.29 5.58 5.65 ...
```

Dataframes

We can refer to a column in a dataframe using \$:

```
Cavendish$Density    # case-sensitive, oops
```

```
NULL
```

```
Cavendish$density # case-sensitive
```

```
[1] 5.50 5.61 4.88 5.07 5.26 5.55 5.36 5.29 5.58 5.65 5.57 5.53 5.62 5.29 5.44 5.34
```

```
[17] 5.79 5.10 5.27 5.39 5.42 5.47 5.63 5.34 5.46 5.30 5.75 5.68 5.85
```

Get a general-purpose statistical summary of the column in a dataframe using \$:

```
summary(Cavendish$density)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
4.880	5.300	5.460	5.448	5.610	5.850

```
summary(Cavendish) # ALL columns
```

density		density2		density3	
Min.	:4.880	Min.	:5.070	Min.	:5.100
1st Qu.	:5.300	1st Qu.	:5.340	1st Qu.	:5.340
Median	:5.460	Median	:5.470	Median	:5.460
Mean	:5.448	Mean	:5.482	Mean	:5.483
3rd Qu.	:5.610	3rd Qu.	:5.620	3rd Qu.	:5.625
Max.	:5.850	Max.	:5.880	Max.	:5.850
		NA's	:6		

R sites:
BEST general sites with examples:

Quick-R:
<http://www.statmethods.net/index.html>

Tutorials:
<http://thomasleeper.com/Rcourse/Tutorials/>

<http://www.wessa.net/stat.wasp?outtype=>

Wessa.net offers these software applications free of charge:

Descriptive Statistics Software	This collection of Free Statistics Calculators offers a wide range of descriptive and explorative types of statistical measures and analysis: Central Tendency, Average, Mean, Median, Variability, Interquartile Range, Concentration, Lorenz Curve, Gini Coefficient, Skewness, Kurtosis, Quartiles, Percentiles, Notched Boxplot, Histogram, Correlation, Partial Correlation, Rank Correlation (Spearman and Kendall), Simple Regression, Kernel Density Estimation, Harrell-Davis Quantiles, Bivariate KDE, Correlation Matrix, Stem-and-leaf plot, Explorative Data Analysis
Regression Software	This is a collection of Regression-based types of analysis for single and multiple equations: Multiple Regression Equations, Bias-Reduced Logistic Regression (Firth method), Recursive Partitioning (Classification & Regression Trees), Variance-based Structural Equation Modeling (Partial Least Squares - Path Modeling).
Statistical Distributions	Features Random Number Generators, PPCC Plots (incl. Tukey lambda), and Statistical Distribution Fitting Modules (Maximum Likelihood) for a series of important distributions: Beta, Inverted Beta, Cauchy 1, Cauchy (2 parameters), Chi, Chi Square (1 parameter), Chi Square (2 parameters), Erlang, Exponential, Fisher F, Gamma, Inverted Gamma, Gumbel, Laplace, Logistic, Lognormal, Normal, Pareto, Power, Rayleigh, r-Distribution, Rectangular (Uniform), Student t, Triangular, and Weibull.
Statistical Hypothesis Testing Software	Offers statistical testing of a variety of hypotheses: Population Mean, Mean (critical value, p-value, type II error, sample size), Skewness/Kurtosis, Quasi Random-Walk
Time Series Analysis (R modules)	Performs Univariate Box-Jenkins ARIMA modeling, forecasting, and residual model checking. In future this module will replace the old versions based on C code.
Statistics Education at Simon Fraser University	This section contains various types of Statistics Software for Statistics Education that have been created by Larry Weldon at Simon Fraser University, Vancouver, Canada.

Academic Computations Archive
Search Computations R Project
FAQ
About Wessa.net
Powered by Linux

Server status page
Any R Server
'Herman Ole Andreas Wold'
'Gwilym Jenkins'
'Sir Ronald Aylmer Fisher'
'AstonUniversity'
'Gertrude Mary Cox'
Installed Packages

History list

Google Custom Search

NEW!
Corel VideoStudio X4 Pro

Interactive examples with data!

Homework 1 due before the next class

Do problems from VCH1-2014 (page 18):

1.1 1.2 1.3 1.4 1.6 1.7 1.8 1.11 1.12

In general, data files can be found here:

http://math.mercyhurst.edu/~sousley/STAT_139/data/

```
exec.pay <- read.csv("http://math.mercyhurst.edu/~sousley/STAT_139/data/exec.pay.csv", header=T)
```

If not found:

```
install.packages("UsingR") # download package and many other linked packages
```

```
library(UsingR) #
```

```
exec.oay # will show the data
```

Orange data is part of R

Columns:

Tree: tree number (five trees measured at various ages)

age: number of days since December 31, 1968

circumference: trunk circumference in mm

Homework Procedures

IMPORTANT:

You will need to put your answers into a **Word document** with your name at the top and "Homework 1".

Include all R code and output with each answer.

Answer the question using a sentence and any appropriate units!

Name your file "*[LastName]*-Homework1" and include your full name in it. Example: Smith-Homework1.docx

Attach the Word file to an email to me with the subject:

"DATA 500 Homework 1"

sousley@mercyhurst.edu

