

Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86\_64-apple-darwin13.4.0 (64-bit)

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

Natural language support but running in an English locale

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

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

Spatial environmental data analysis with R  
GEO 503

# Agenda

---

Quick introductory presentation

Hands-on exercises

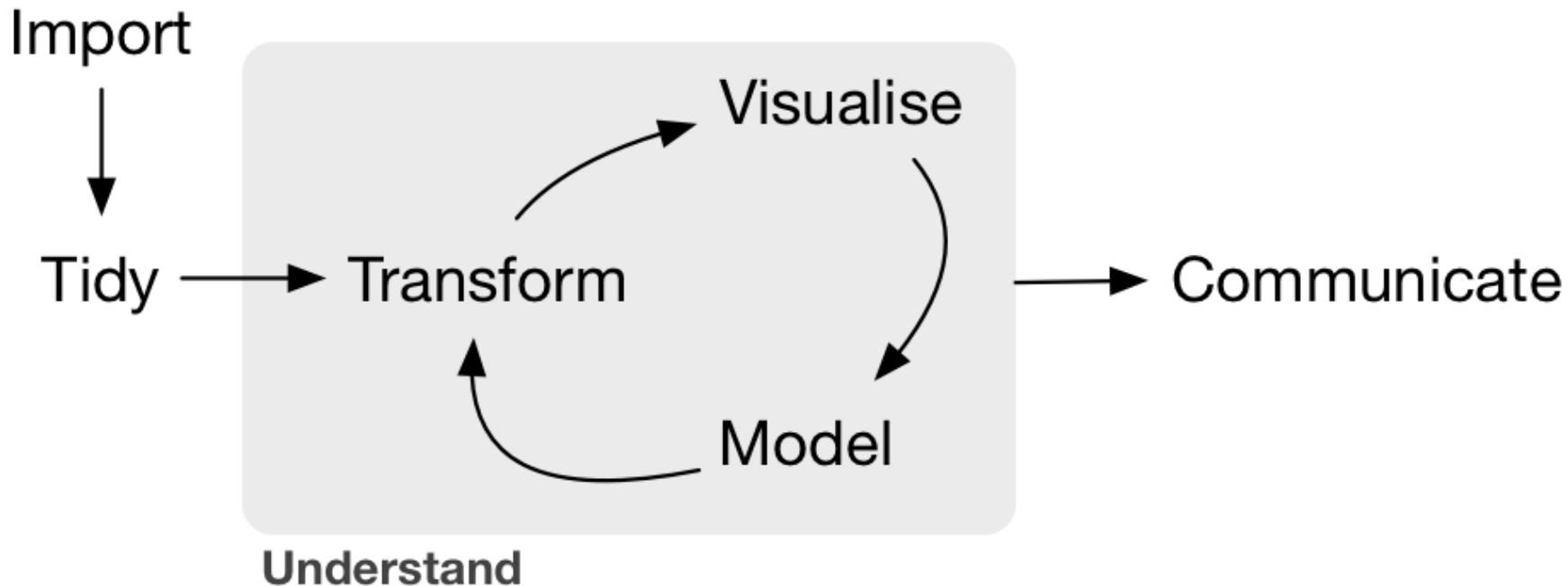
[adamwilson.us/SpatialDataScience](http://adamwilson.us/SpatialDataScience)

Please interrupt!



# Data Science?

---



On programming

“Programming ought to be regarded as an integral part of effective and responsible data analysis”

- Venables and Ripley. 1999. **S Programming**

You can figure it out!

# A table ‘named’ iris.

---

Row	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa

```
mean(iris$Sepal.Width)  
filter(iris, Sepal.Length<4.9)  
iris$Sepal.Length + iris$Petal.Length
```

# It won't take long before you can 'read' this...

---

```
space.only <- gam(present~s(X_CEN, Y_CEN),
                    data=finch@data, family="binomial")
preds.space.only <- as.numeric(predict(space.only,
                                         type="response"))
resid.space.only <- residuals(space.only)
finch$space=as.numeric(predict(space.only,type="terms"))

ggplot(finch@data,
       aes(x=x, y=y, z=space, map_id = id)) +
  geom_map(aes(fill = space), map = pfinch) +
  geom_point(aes(col=as.logical(present))) +
  expand_limits(x = pfinch$long,
                y = pfinch$lat) +
  scale_fill_gradientn(colours=
    c("darkblue", "blue", "grey", "yellow", "orange", "red")) +
  scale_color_manual(values=
    c("transparent", "black"), name="Present") + coord_equal()
```

# Data Types

---

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

<b>as.logical</b>	TRUE, FALSE, TRUE	Boolean values (TRUE or FALSE).
<b>as.numeric</b>	1, 0, 1	Integers or floating point numbers.
<b>as.character</b>	'1', '0', '1'	Character strings. Generally preferred to factors.
<b>as.factor</b>	'1', '0', '1', levels: '1', '0'	Character strings with preset levels. Needed for some statistical models.

# Creating and destroying objects

---

## Variable Assignment

```
> a <- 'apple'  
> a  
[1] 'apple'
```

## The Environment

`ls()` List all variables in the environment.

`rm(x)` Remove x from the environment.

`rm(list = ls())` Remove all variables from the environment.

**You can use the environment panel in RStudio to browse variables in your environment.**

# Vectors

---

Creating Vectors		
<code>c(2, 4, 6)</code>	2 4 6	Join elements into a vector
<code>2:6</code>	2 3 4 5 6	An integer sequence
<code>seq(2, 3, by=0.5)</code>	2.0 2.5 3.0	A complex sequence
<code>rep(1:2, times=3)</code>	1 2 1 2 1 2	Repeat a vector
<code>rep(1:2, each=3)</code>	1 1 1 2 2 2	Repeat elements of a vector

## Vector Functions

**sort(x)**

Return x sorted.

**table(x)**

See counts of values.

**rev(x)**

Return x reversed.

**unique(x)**

See unique values.

More with dplyr later

## Selecting Vector Elements

### By Position

**x[4]** The fourth element.

**x[-4]** All but the fourth.

**x[2:4]** Elements two to four.

**x[-(2:4)]** All elements except  
two to four.

**x[c(1, 5)]** Elements one and  
five.

### By Value

**x[x == 10]** Elements which  
are equal to 10.

**x[x < 0]** All elements less  
than zero.

**x[x %in%  
c(1, 2, 5)]** Elements in the set  
1, 2, 5.

### Named Vectors

**x['apple']** Element with  
name 'apple'.

# Working with Matrixes (matrices)

---

## Matrixes

```
m <- matrix(x, nrow = 3, ncol = 3)
```

Create a matrix from x.



`m[2, ]` - Select a row



`m[, 1]` - Select a column



`m[2, 3]` - Select an element

`t(m)`

Transpose

`m %*% n`

Matrix Multiplication

`solve(m, n)`

Find x in:  $m * x = n$

# 3 Basic indexing operators

---

`x[i]`   `x[i, j]`   `x[[i]]`   `x[[i, j]]`      `x$a`   `x$"a"`

Vectors and matrices: use [ rarely used

- [[drops any names or dimnames attribute & partial matching is used

Multi-dimensional structures with a single index: `x[[i]]` or `x[i]`

- return the  $i^{\text{th}}$  sequential element of `x`

Lists:

- [[ select single element
- [ returns a list of selected elements.

[[

- allows only a single element to be selected using integer or character indices,

[

- allows indexing by vectors.

\$: recursive objects (lists and data.frames). Only literal character string or a symbol as the index. The index is *not computable*. If you need to evaluate an expression to find the index, use `x[[expr]]`.

# Maths Functions

<b>log(x)</b>	Natural log.	<b>sum(x)</b>	Sum.
<b>exp(x)</b>	Exponential.	<b>mean(x)</b>	Mean.
<b>max(x)</b>	Largest element.	<b>median(x)</b>	Median.
<b>min(x)</b>	Smallest element.	<b>quantile(x)</b>	Percentage quantiles.
<b>round(x, n)</b>	Round to n decimal places.	<b>rank(x)</b>	Rank of elements.
<b>sig.fig(x, n)</b>	Round to n significant figures.	<b>var(x)</b>	The variance.
<b>cor(x, y)</b>	Correlation.	<b>sd(x)</b>	The standard deviation.

# Lists

```
l <- list(x = 1:5, y = c('a', 'b'))
```

A list is collection of elements which can be of different types.

---

**l[2]**

Second element  
of l.

**l[1]**

New list with  
only the first  
element.

**l\$x**

Element named  
x.

**l['y']**

New list with  
only element  
named y.

Also see the  
**dplyr** library.

# Data Frames

```
df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))
```

A special case of a list where all elements are the same length.

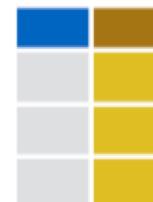
x	y
1	a
2	b
3	c

## List subsetting

df\$x



df[[2]]



*Understanding a data frame*

**View(df)**

See the full data frame.

**head(df)**

See the first 6 rows.

## Matrix subsetting

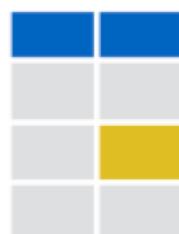
`df[ , 2]`



`df[2, ]`



`df[2, 2]`



## Data Frames

`nrow(df)`

Number of rows.

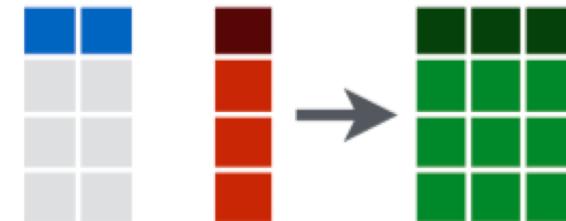
`ncol(df)`

Number of columns.

`dim(df)`

Number of columns and rows.

`cbind` - Bind columns.



`rbind` - Bind rows.



## Strings

Also see the **stringr** library.

<code>paste(x, y, sep = ' ')</code>	Join multiple vectors together.
<code>paste(x, collapse = ' ')</code>	Join elements of a vector together.
<code>grep(pattern, x)</code>	Find regular expression matches in x.
<code>gsub(pattern, replace, x)</code>	Replace matches in x with a string.
<code>toupper(x)</code>	Convert to uppercase.
<code>tolower(x)</code>	Convert to lowercase.
<code>nchar(x)</code>	Number of characters in a string.

## Factors

<code>factor(x)</code>	<code>cut(x, breaks = 4)</code>
Turn a vector into a factor. Can set the levels of the factor and the order.	Turn a numeric vector into a factor but ‘cutting’ into sections.

# Statistics

**lm(x ~ y, data=df)**

Linear model.

**glm(x ~ y, data=df)**

Generalised linear model.

**summary**

Get more detailed information  
out a model.

**t.test(x, y)**

Preform a t-test for  
difference between  
means.

**pairwise.t.test**

Preform a t-test for  
paired data.

**prop.test**

Test for a  
difference  
between  
proportions.

**aov**

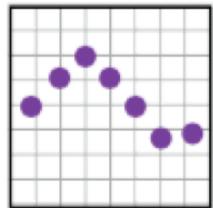
Analysis of  
variance.

# Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	<code>rnorm</code>	<code>dnorm</code>	<code>pnorm</code>	<code>qnorm</code>
Poisson	<code>rpois</code>	<code>dpois</code>	<code>ppois</code>	<code>qpois</code>
Binomial	<code>rbinom</code>	<code>dbinom</code>	<code>pbinom</code>	<code>qbinom</code>
Uniform	<code>runif</code>	<code>dunif</code>	<code>punif</code>	<code>qunif</code>

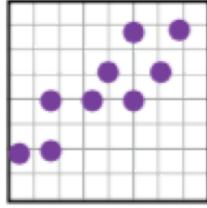
# Plotting

Also see the **ggplot2** library.



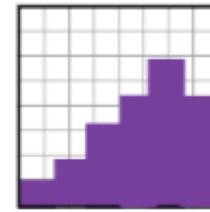
**plot(x)**

Values of x in  
order.



**plot(x, y)**

Values of x  
against y.



**hist(x)**

Histogram of  
x.

# Dates

See the **lubridate** library.

# Programming

## For Loop

```
for (variable in sequence){  
    Do something  
}
```

### Example

```
for (i in 1:4){  
    j <- i + 10  
    print(j)  
}
```

## While Loop

```
while (condition){  
    Do something  
}
```

### Example

```
while (i < 5){  
    print(i)  
    i <- i + 1  
}
```

# If statements and functions

## If Statements

```
if (condition){  
    Do something  
} else {  
    Do something different  
}
```

### Example

```
if (i > 3){  
    print('Yes')  
} else {  
    print('No')  
}
```

## Functions

```
function_name <- function(var){  
    Do something  
    return(new_variable)  
}
```

### Example

```
square <- function(x){  
  
    squared <- x*x  
  
    return(squared)  
}
```

## Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

# Libraries

---

## Using Libraries

**install.packages('dplyr')**

Download and install a package from CRAN.

**library(dplyr)**

Load the package into the session, making all its functions available to use.

**dplyr::select**

Use a particular function from a package.

**data(iris)**

Load a built-in dataset into the environment.

# Working Directory

---

## Working Directory

### **getwd()**

Find the current working directory (where inputs are found and outputs are sent).

### **setwd('C://file/path')**

Change the current working directory.

**Use projects in RStudio to set the working directory to the folder you are working in.**

## Reading and Writing Data

Input	Ouput	Description
<code>df &lt;- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df &lt;- read.csv('file.csv')</code>	<code>write.csv(df, 'file.csv')</code>	Read and write a comma separated value file. This is a special case of read.table/write.table.
<code>load('file.RData')</code>	<code>save(df, file = 'file.Rdata')</code>	Read and write an R data file, a file type special for R.

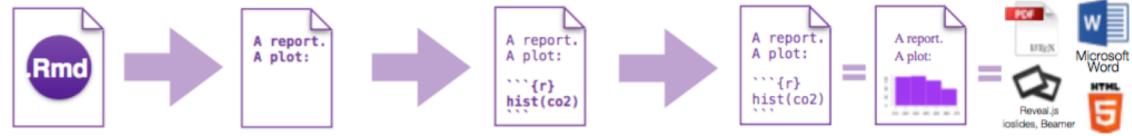
# Cheatsheets

Check out <https://www.rstudio.com/resources/cheatsheets/> for more cheat sheets summarizing R and related projects...



**1. 작업흐름** R 마크다운은 R로 재현가능하고, 동적인 보고서를 작성하는 서식이다. R 마크다운을 사용해서 R 코드와 실행결과를 발표자료, pdf, html, 워드 문서 등에 삽입할 수 있다. 보고서를 작성하려면:

- i. 파일열기 - .Rmd 확장자를 갖는 파일을 열어 시작하거나, R 마크다운 구문을 사용해서 작성한다.
- ii. 작성하기 - 본문을 작성하기 쉬운 R 마크다운 구문을 사용해서 작성한다.
- iii. 내장하기 - 리포트에 포함될 출력 결과를 생성하는 R 코드를 내장한다.
- iv. 렌더링(Render) - R 코드를 출력형식으로 치환하고 보고서를 발표자료, pdf, html, MS 워드 파일 형식으로 변환한다.



**2. 파일 열기** .Rmd 확장자를 갖는 텍스트 파일로 저장해서 시작하거나, Studio Rmd 템플릿을 열어 시작한다.

**3. 마크다운** 다음으로, 일반 텍스트로 보고서를 작성한다. 마크다운 구문을 사용해서 최종 보고서에 적용할 텍스트 서식을 기술한다.

RMarkdown Cheatsheet in Korean

# Homework Notes

# Homework

---

Homework #1 Due Next Monday before class  
(8:30AM)

'Example' Homework available  
Find it in UBLearn

# Homework submitted in UBlearns

---

## Begin: Homework #1

[Cancel](#) [Begin](#)

### 1. Instructions

Description	These quizzes are designed to encourage you to work through the materials we discuss in class <i>prior</i> to class so you can come with questions.
Instructions	Please use the attached R script ( <a href="#">Homework_01.R</a> ) as a template for you to find the answers to the questions. The last question will ask you to upload your updated script (with the code needed to answer the questions). This will not be graded, but will be taken into account if there are any questions about the correct answers later. I recommend that you complete all the questions in the .R file in RStudio before entering the answers into UBlearns.
Force Completion	This test can be saved and resumed later.
Due Date	This Test is due on September 14, 2015 5:00:00 PM EDT. Test cannot be started past this date.
Click Begin to start: Homework #1. Click Cancel to go back.	

### 2. Submit

*Click Begin to start. Click Cancel to quit.*

[Cancel](#) [Begin](#)

Working collaboratively is encouraged but you are responsible for developing your own code to answer the questions:

**Acceptable:** “which functions did you use to answer #4?”

**Unacceptable:** “please email me your code for #4.”

# Homework format

## Take Test: Homework #1

### Test Information

Description These quizzes are designed to encourage you to work through the materials we discuss in class *prior* to class so you can come with questions.

Instructions Please use the attached R script ([Homework\\_01.R](#)) as a template for you to find the answers to the questions. The last question will ask you to upload your updated script (with the code needed to answer the questions). This will not be graded, but will be taken into account if there are any questions about the correct answers later. I recommend that you complete all the questions in the .R file in RStudio before entering the answers into UBLearn.

Multiple Attempts Not allowed. This test can only be taken once.

Force Completion This test can be saved and resumed later.

### Question Completion Status:

[Save All Answers](#)

[Save and Submit](#)

#### Question 1

Load the `iris` dataset by running `data(iris)`. How many observations (rows) are there for the versicolor species?

1 points [Save Answer](#)

#### Question 2

Create a vector with the following values:

23, 45, 12, 89, 1, 13, 28, 18

Then multiply each element of the vector by 15. What is the standard deviation of the new vector?

1 points [Save Answer](#)

# R Introduction



Please interrupt!

# Set up your screen

GEO 503: R Spatial Data Science Home Syllabus Schedule Content Assignments Resources

- Variables
- Variable naming conventions
- Subsetting
- Using Functions**
- Missing data: dealing with NA values
- Logical values
- Generating Data
- Matrices
- Data Frames
- Loading Packages

## Using Functions

To calculate the mean, you could do it *manually* like this

```
(5+8+14+91+3+36+14+30) / 8
```

## [1] 25.125

Or use a function:

```
mean(x)
```

## [1] 25.125

Type `?functionname` to get the documentation (`?mean`) or `??"search parameters (??standard deviation")` to search the documentation. In RStudio, you can also search in the help panel. `mean` has other arguments too:

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

In RStudio, if you press `TAB` after a function name (such as `mean()`), it will show function arguments.

>  
>  
> **x =** x  
> ... = An R object. Currently there are methods for numeric/logical  
> trim = vectors and date, [date-time](#) and time interval objects. Complex  
> na.rm = vectors are allowed for `trim = 0`, only.  
>  
>  
>  
>  
Press F1 for additional help  
> mean

Autocomplete screenshot

Calculate the standard deviation of `c(3, 6, 12, 89)`.

**SHOW SOLUTION**

Writing functions in R is pretty easy. Let's create one to calculate the mean of a vector by getting the sum and length. First think about how to break it down into parts:

```
x1= sum(x)  
x2=length(x)  
x1/x2
```

## [1] 25.125

Then put it all back together and create a new function called `mymean`:

The screenshot shows the RStudio interface. The top panel displays an R script named '01\_intro.R' with code demonstrating various R functions. A large red text overlay on the right side reads 'Open R Script in RStudio to follow along'. The bottom panel shows the 'Console' tab with R command history and error messages. Another red text overlay at the bottom right reads 'R Terminal'.

```
71 #'
72 #' #### Using Functions
73 #
74 #' To calculate the mean, you could do it manually
75 # like this
76 ## -----
77 (5+8+14+91+3+36+14+30)/8
78 #
79 #
80 #' Or use a function:
81 ## -----
82 mean(x)
80:22 # (Untitled) ▾ R Script ▾
```

Console R Markdown ▾

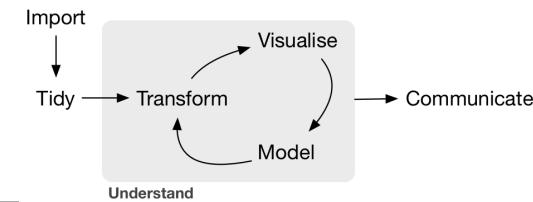
```
~/repos/RDataScience/ ▾
+ coord_equal()
+
Regions defined for each Polygons
Error in as.vector(x, mode) :
  cannot coerce type 'environment' to vector of type 'any'
> ggplot(fortify(sids_us),aes(x=long,y=lat,order=order,group=group))+
```

+ geom\_polygon(fill="white",col="black")+
+ coord\_equal()

```
Regions defined for each Polygons
> |
```

R Terminal

# Take time to learn efficient flows...



The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'View', 'Code', 'Tools', 'Help', and 'Addins'. The title bar shows multiple open files: 'chedule.Rmd', '01\_intro.Rmd', '01\_intro.R', and '00\_CourseIntroductionF'. The main workspace contains an R script with the following code:

```
17 #' ## Variables
18 ##
19 x=1
20 x
21 #
22 #' We can also assign a vector to a variable:
23 #
24 ##
25 x=c(5,8,14,91,3,36,14,30)
26 x
27 x
28
```

A red circle highlights the code area, and a red arrow points from the text 'Write code here' to the start of the code. Another red arrow points from the text 'Ctrl (or command)-R will run a line (or whatever is highlighted)' to the highlighted code area.

The bottom left shows the 'Console' tab with the R environment. The output window displays:

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

> x=1
> x
[1] 1
>
```

A red circle highlights the console area, and a red arrow points from the text 'Outputs appear here, did you get what you wanted?' to the console output.

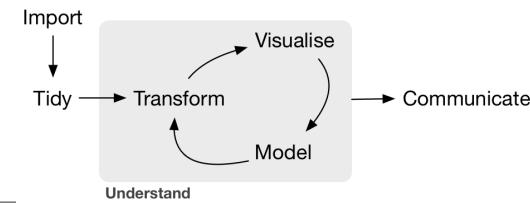
The right side of the interface features the 'Environment' tab of the file browser, showing a list of files and folders:

Path	01_intro.R	01_intro.Rmd	01_intro.html	01_intro.md	01_intro_presentation/	00_CourseIntroductionF/
Staged						
Status						

Below the file browser are tabs for 'Files', 'Plots', 'Packages', and 'Help'. The 'Files' tab is selected, showing a list of files:

- ..
- .gitignore
- 00\_CourseIntroduction
- 00\_CourseIntroductionFrame.R
- 00\_CourseIntroductionFrame.Rproj
- 00\_CourseIntroductionFrame.Rmd
- 01\_intro.html
- 01\_intro.md
- 01\_intro.R
- 01\_intro.Rmd
- 01\_intro.html
- 01\_intro.md

Take time to learn efficient flows...



A screenshot of the RStudio interface. The top-left pane shows a script editor with R code. The middle-left pane shows the R console with output. The right side shows a file browser with a list of files related to a '01\_intro' project. Overlaid on the bottom-right area is large red text:

Your code is  
your product!

Your outputs  
are ephemeral

The RStudio interface includes tabs for Environment, History, Build, and GitHub. The file browser shows files like 01\_intro.R, 01\_intro.Rmd, 01\_intro.html, 01\_intro.md, and 01\_intro\_presentation.R. The bottom of the interface shows navigation buttons for New Folder, Delete, Rename, and Help, along with a search bar and a 'RStudio' logo.

### Write Code

File Edit Code View Plots Session Build Debug Tools Help

Source on Save Run Source

```

1 # Good start...
2 Cursors of      Re-run      Source with or      Show file
3 'shared users' previous code without Echo outline
4
5 "P0030001"      Multiple cursors/column selection
6 "P0030002"      with Alt + mouse drag.
7 "P0030003"      Code diagnostics that appear in the margin.
8 "P0030004"      Hover over diagnostic symbols for details.
9
10
11
12 get_digit <-function() {
13   ("num" %% (10 ^ n))
14   %% (10 ^ (n - 1))
15 }
16
17 fo
18   for [snippet]
19     foo { .GlobalEnv }
20     force { base }
21 Jump to function in file
22
  
```

1:1 (Top Level) R Script

Console Compile PDF R Markdown

```

> foo(1)
[1] 2
> foo <- function(x) x + 1
> foo(2)
[1] 3
> foo(2)
[1] 3
> foo(1)
  
```

Working Directory Maximize, minimize panes

Press ↑ to see command history Drag pane boundaries

### R Support

Import data file with wizard History of past commands to run/add to source Display .RPres slideshows

**File > New File > R Presentation**

Environment History Build Git Presentation

Import Dataset Global Environment Load workspace Save workspace Delete all saved objects Search inside environment

Choose environment to display from list of parent environments

Data Values Data 150 obs. of 5 variables

Functions 1

View in data viewer View function source code

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Home IDEcheatsheet Name

Create folder Upload file Delete file Rename file

Copy... Move... Export... Set As Working Directory Go To Working Directory

Path to displayed directory

.. hello.R 450 B Dec 24, 2015, 8:55 AM

A File browser keyed to your working directory. Click on file or directory name to open.

# RStudio Keyboard shortcuts for (nearly) everything

---

<b>1 LAYOUT</b>	<b>Windows/Linux</b>	<b>Mac</b>
Move focus to Source Editor	Ctrl+1	Ctrl+1
Move focus to Console	Ctrl+2	Ctrl+2
Move focus to Help	Ctrl+3	Ctrl+3
Show History	Ctrl+4	Ctrl+4
Show Files	Ctrl+5	Ctrl+5
Show Plots	Ctrl+6	Ctrl+6
Show Packages	Ctrl+7	Ctrl+7
Show Environment	Ctrl+8	Ctrl+8
Show Git/SVN	Ctrl+9	Ctrl+9
Show Build	Ctrl+0	Ctrl+0

Focus on use of ctrl (command) -R for sending code