```
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

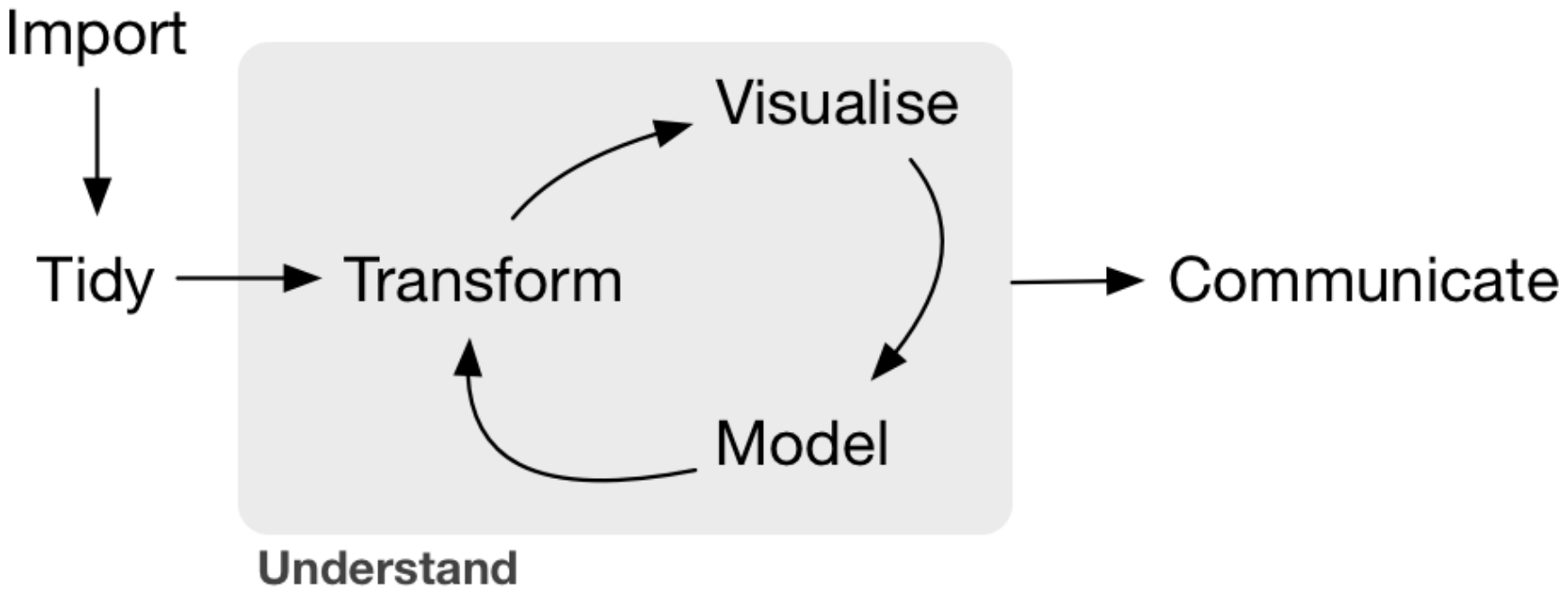
Quick introductory presentation

Hands-on exercises

adamwilson.us/SpatialDataScience

Please interrupt!

Data Science?



--Grolemund & Wickham, R for Data Science, O'Reilly 2016

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.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'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

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

Vector Functions

`sort(x)`

Return x sorted.

`rev(x)`

Return x reversed.

`table(x)`

See counts of values.

`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^{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

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`sig.fig(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	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]]`	`l[1]`	`l$x`	`l['y']`
Second element of l.	New list with only the first element.	Element named x.	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

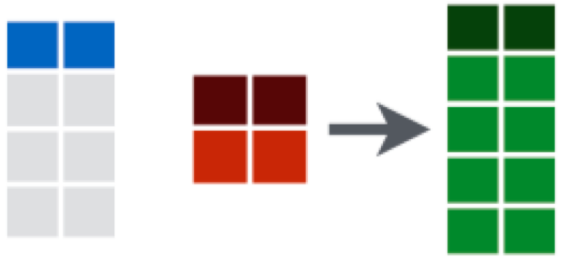
`nrow(df)`
Number of rows.

`ncol(df)`
Number of columns.

`dim(df)`
Number of columns and rows.

`cbind` - Bind columns.

`rbind` - Bind rows.



Strings

Also see the **stringr** library.

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

Factors

`factor(x)`

Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)`

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.

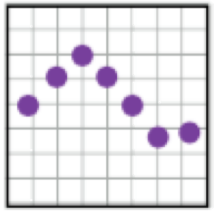
`aov`
Analysis of variance.

Distributions

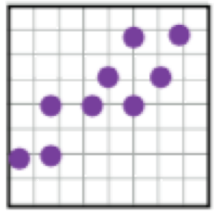
	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	`rnorm`	`dnorm`	`pnorm`	`qnorm`
Poisson	`rpois`	`dpois`	`ppois`	`qpois`
Binomial	`rbinom`	`dbinom`	`pbinom`	`qbinom`
Uniform	`runif`	`dunif`	`punif`	`qunif`

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.

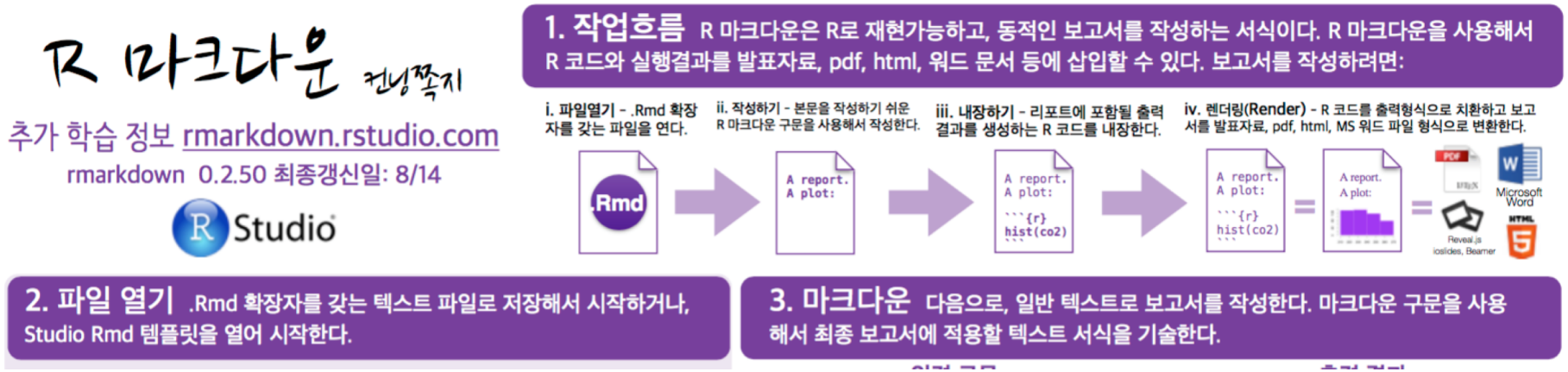
Use projects in RStudio to set the working directory to the folder you are working in.

Reading and Writing Data

Input	Ouput	Description
`df <- read.table('file.txt')`	`write.table(df, 'file.txt')`	Read and write a delimited text file.
`df <- read.csv('file.csv')`	`write.csv(df, 'file.csv')`	Read and write a comma separated value file. This is a special case of read.table/ write.table.
`load('file.RData')`	`save(df, file = 'file.Rdata')`	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…



RMarkdown Cheatsheet in Korean

Homework Notes

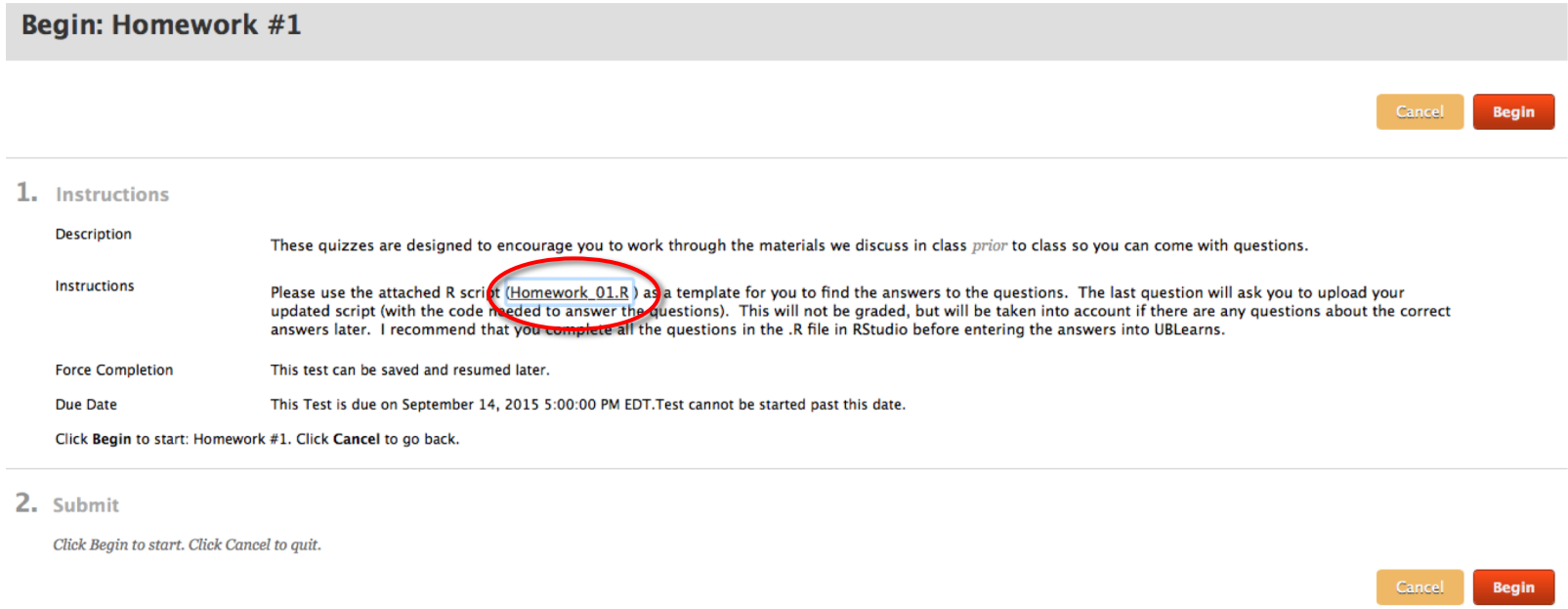
Homework

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

’Example’ Homework available

Find it in UBLearns

Homework submitted in UBlearns



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 UBLearns.
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 — 1 points — Save Answer

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

Question 2 — 1 points — Save Answer

Create a vector with the following values:

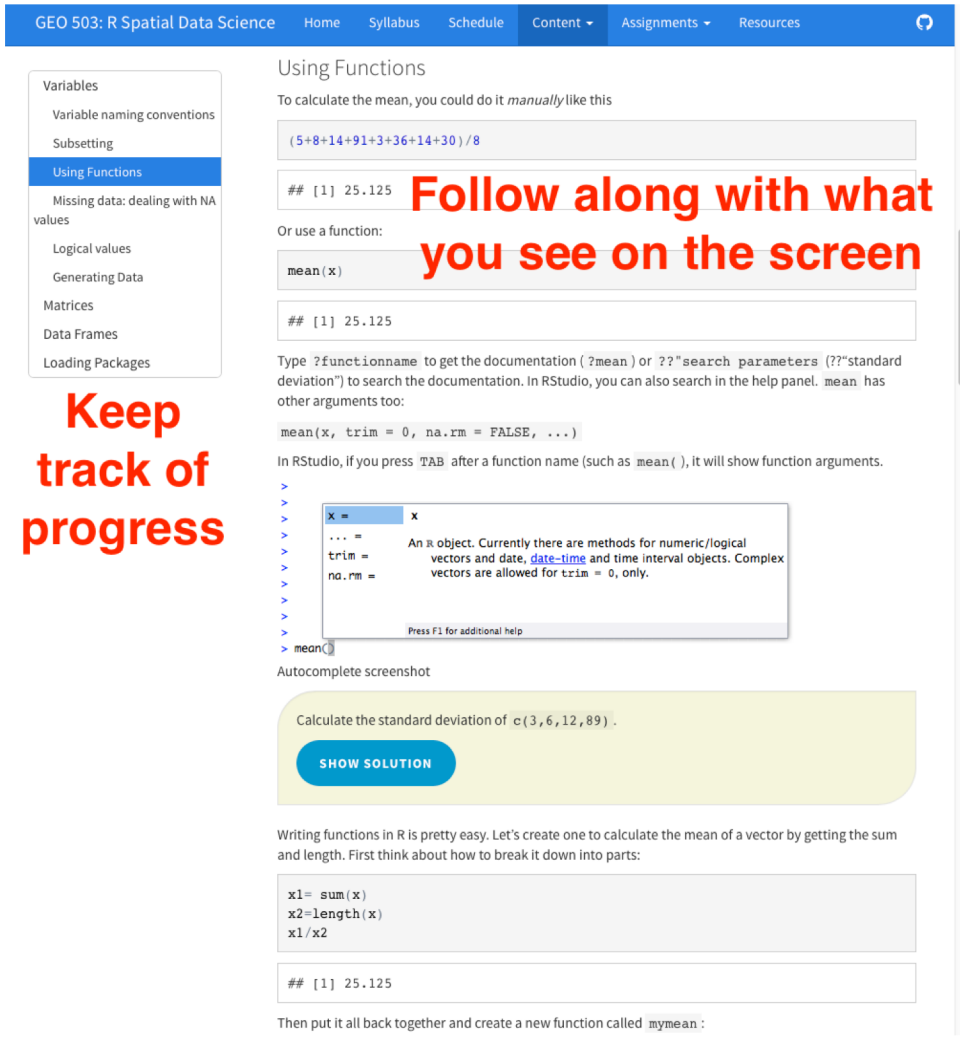
23, 45, 12, 89, 1, 13, 28, 18

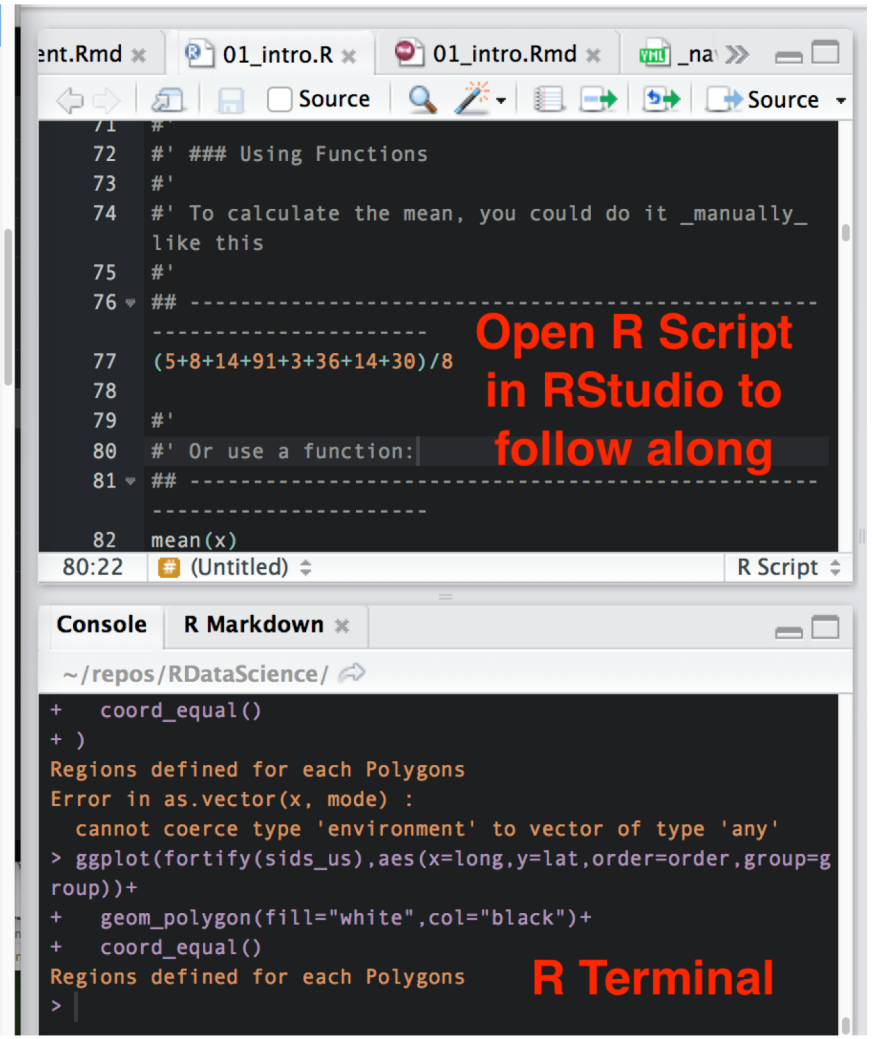
Then multiply each element of the vector by 15. What is the standard deviation of the new vector?

R Introduction

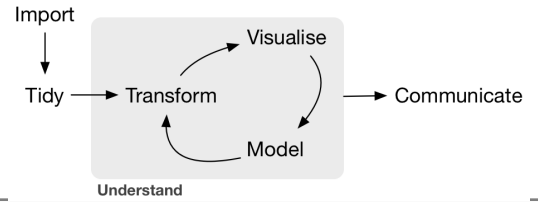
Please interrupt!

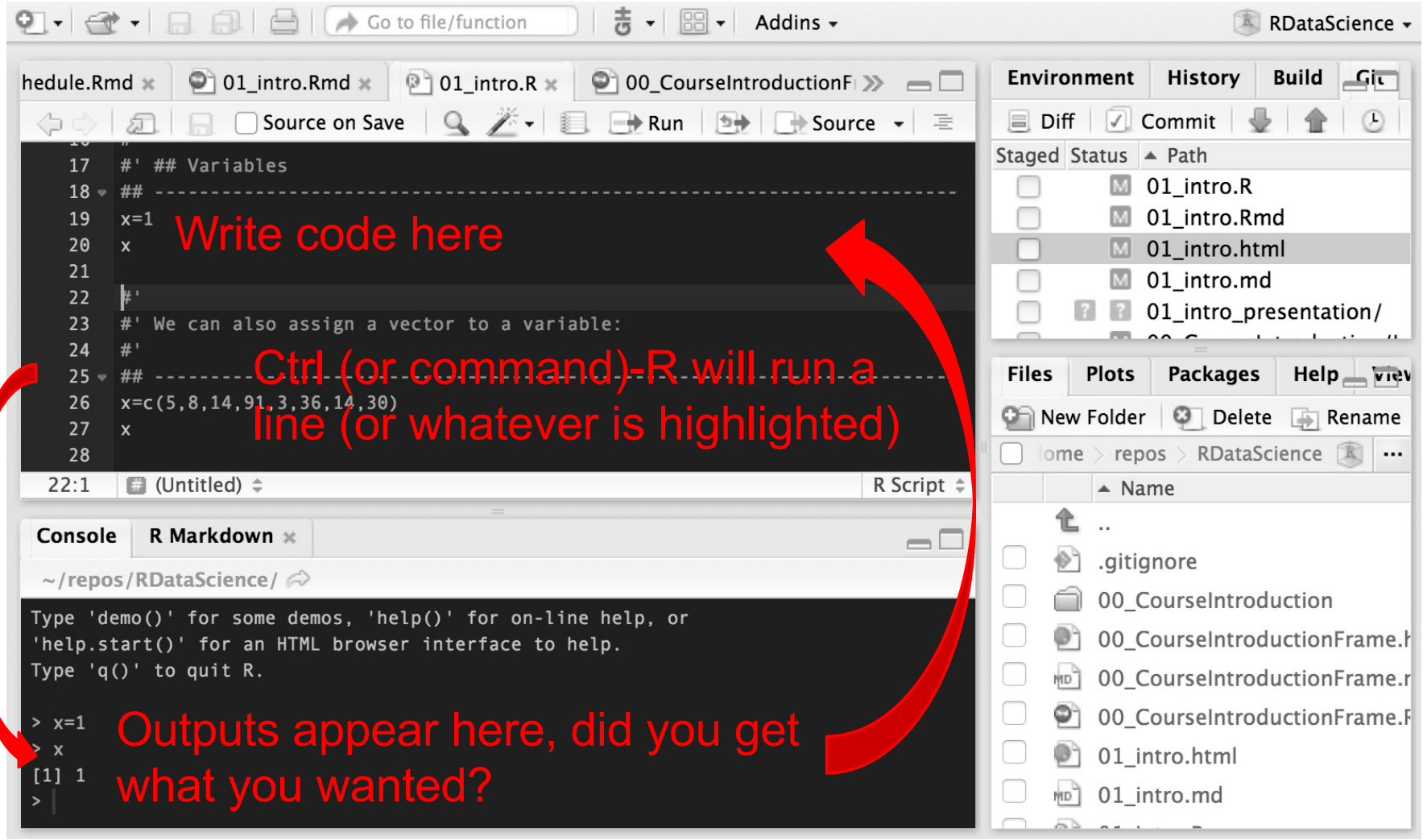
Set up your screen



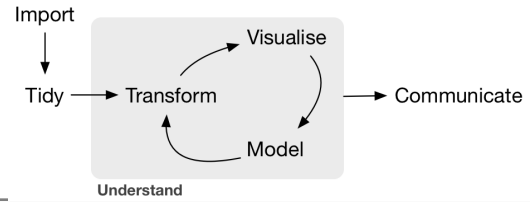


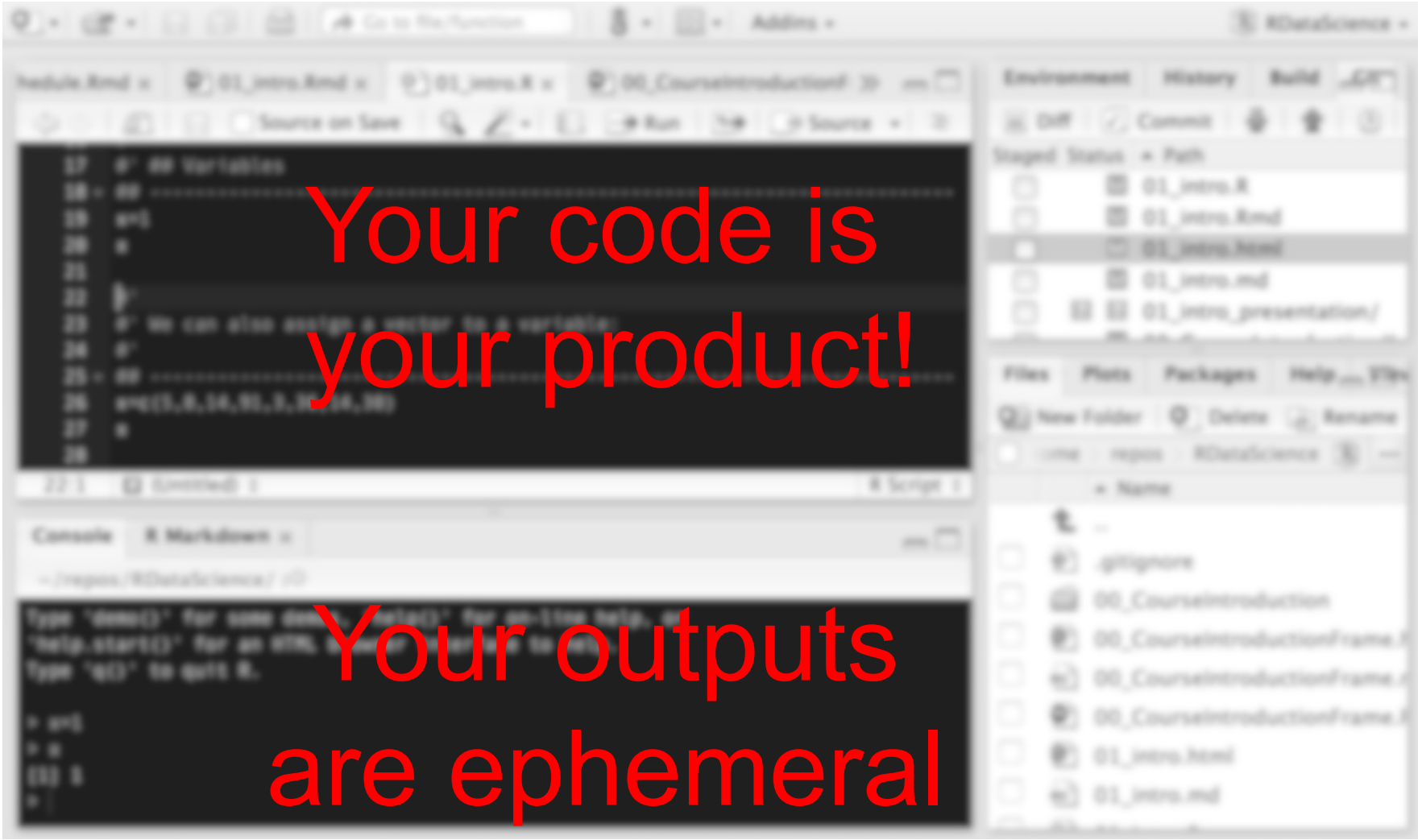
Take time to learn efficient flows…





Take time to learn efficient flows…





Write Code

Navigate tabs

Open in new window

Save

Find and replace

Compile as notebook

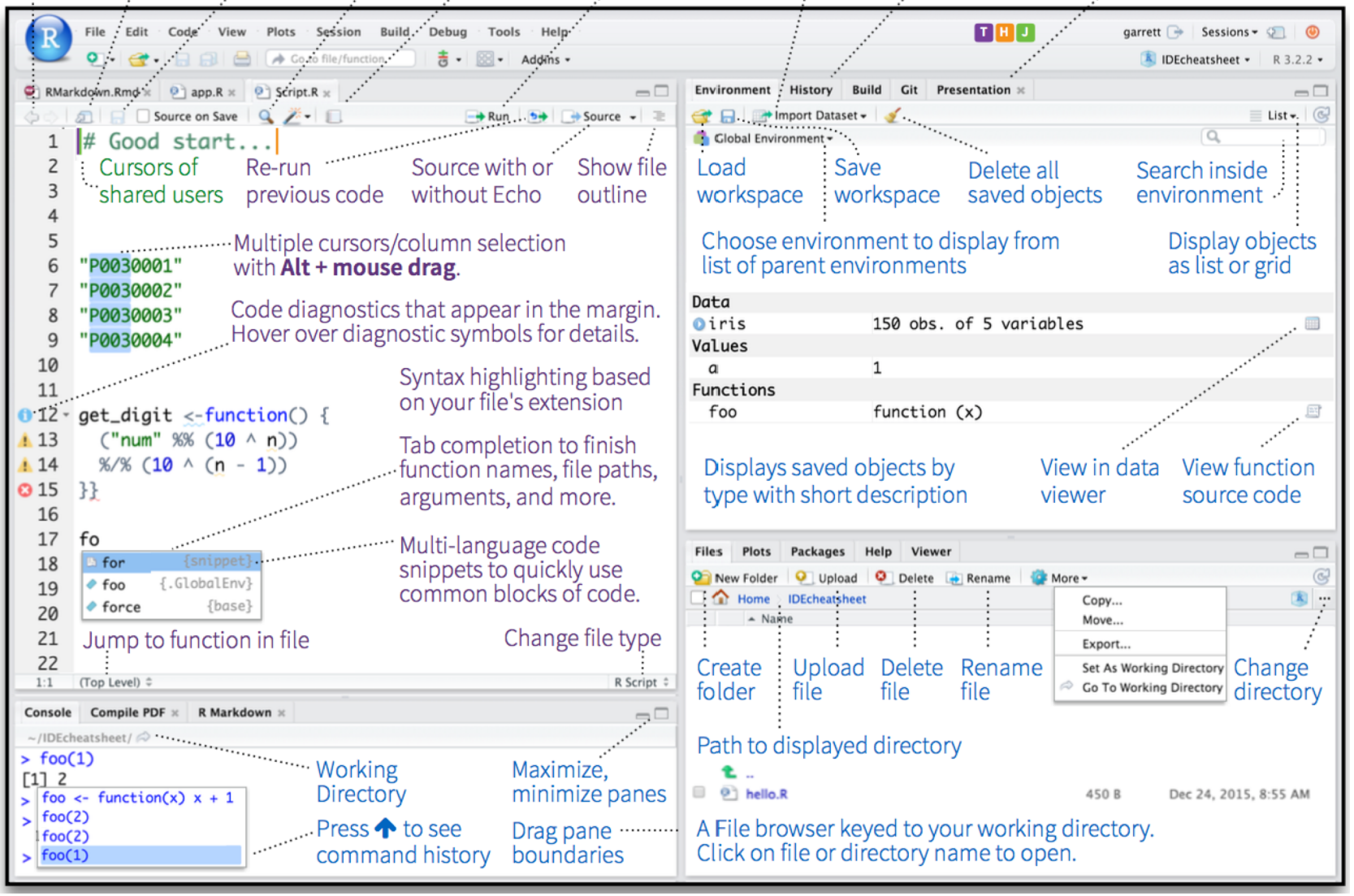
Run selected code

R Support

Import data file with wizard

History of past commands to run/add to source

Display .RPres slideshows **File > New File > R Presentation**



RStudio Keyboard shortcuts for (nearly) everything

1 LAYOUT	Windows/Linux	Mac
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