

```
AdamWilsonMac:~ adamw$ R
```

```
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"  
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.  
Type 'q()' to quit R.
```

Spatial environmental data analysis with R
GEO 503

Agenda

Quick introductory presentation

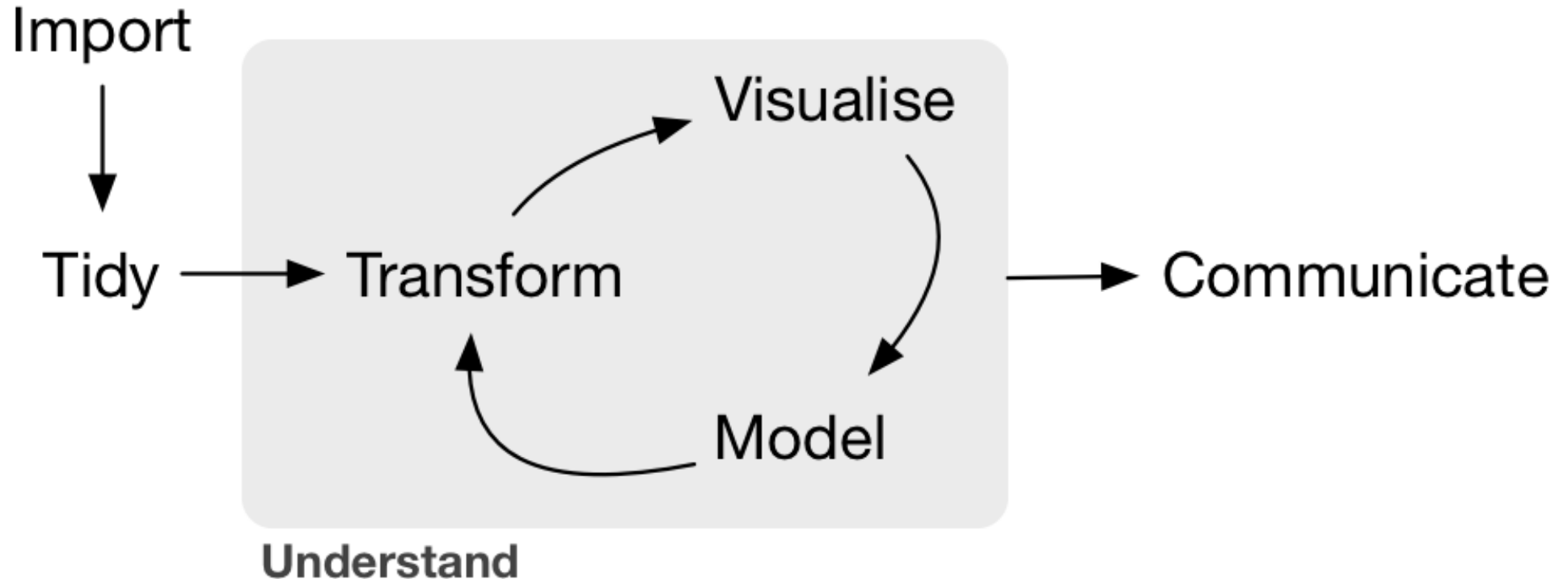
Hands-on exercises

adamwilson.us/RDataScience

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.

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

Creating and destroying objects

Variable Assignment

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

The Environment

<code>ls()</code>	List all variables in the environment.
<code>rm(x)</code>	Remove <code>x</code> from the environment.
<code>rm(list = ls())</code>	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: $m * x = n$

Maths Functions

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

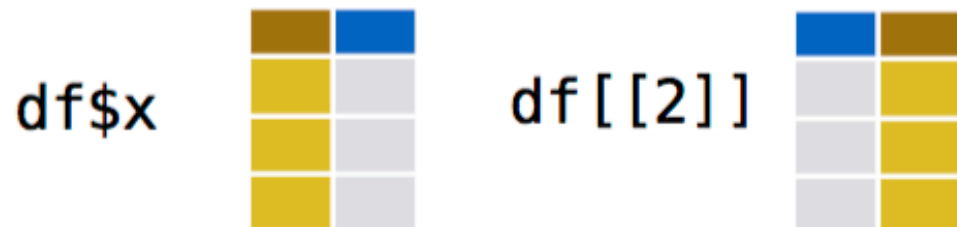
Data Frames

```
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



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)

Perform a t-test for difference between means.

pairwise.t.test

Perform a t-test for paired data.

prop.test

Test for a difference between proportions.

aov

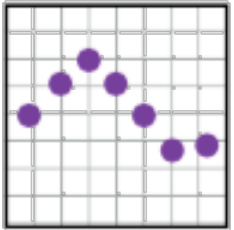
Analysis of variance.

Distributions

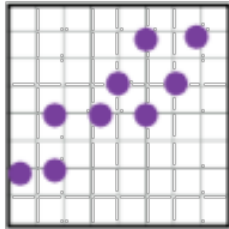
	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	<code>rnorm</code>	<code>dnorm</code>	<code>pnorm</code>	<code>qnorm</code>
Poisson	<code>rpois</code>	<code>dpois</code>	<code>ppois</code>	<code>qpois</code>
Binomial	<code>rbinom</code>	<code>dbinom</code>	<code>pbinom</code>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

Plotting

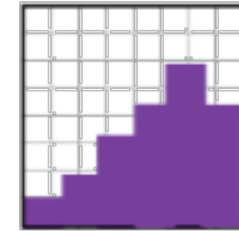
Also see the **ggplot2** library.



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

<code>a == b</code>	Are equal	<code>a > b</code>	Greater than	<code>a >= b</code>	Greater than or equal to	<code>is.na(a)</code>	Is missing
<code>a != b</code>	Not equal	<code>a < b</code>	Less than	<code>a <= b</code>	Less than or equal to	<code>is.null(a)</code>	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 <- read.table('file.txt')</code>	<code>write.table(df, 'file.txt')</code>	Read and write a delimited text file.
<code>df <- 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...

R 마크다운 권영쪽지

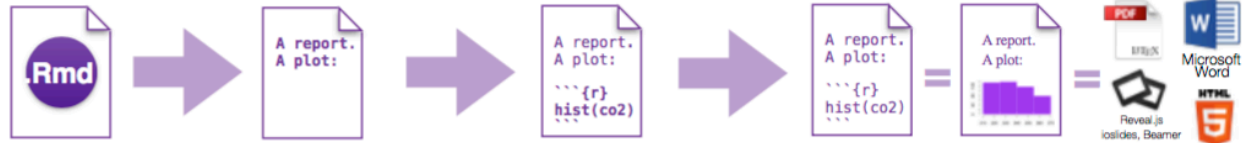
추가 학습 정보 rmarkdown.rstudio.com

rmarkdown 0.2.50 최종갱신일: 8/14



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 연다.
- 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

Homeworks

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

2 Attempts for First Homework

'Example' Homework available

Find it in UBLearns

Homework submitted in UBLearn

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 (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.
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
```

Follow along with what you see on the screen

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`:

Keep track of progress

ent.Rmd x 01_intro.R x 01_intro.Rmd x YML_na >>

Source Source

```
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)
```

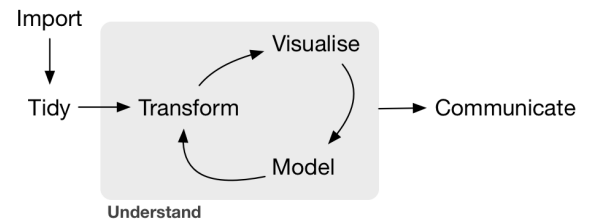
80:22 (Untitled) R Script

Console R Markdown x

```
~/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...



Write code here

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

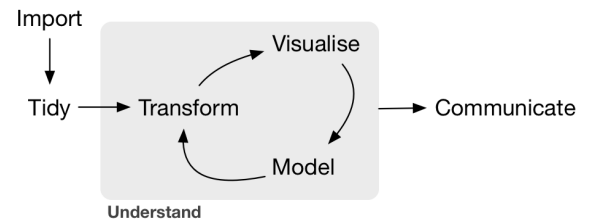
Ctrl (or command)-R will run a line (or whatever is highlighted)

Outputs appear here, did you get what you wanted?

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

The screenshot also shows the Environment pane with files like 01_intro.R, 01_intro.Rmd, 01_intro.html, and 01_intro.md. The Console shows the R prompt and the output of the code execution.

Take time to learn efficient flows...



Your code is your product!

Your outputs are ephemeral

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

The screenshot shows the RStudio IDE interface with several callouts pointing to specific features:

- Code Editor:**
 - Line 1: `# Good start...`
 - Line 2: `Cursors of shared users`
 - Line 3: `Re-run previous code`
 - Line 4: `Source with or without Echo`
 - Line 5: `Show file outline`
 - Line 6: `"P0030001"` (Multiple cursors/column selection with **Alt + mouse drag**)
 - Line 7: `"P0030002"`
 - Line 8: `"P0030003"` (Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.)
 - Line 9: `"P0030004"`
 - Line 12: `get_digit <- function() {` (Syntax highlighting based on your file's extension)
 - Line 13: `("num" %% (10 ^ n))` (Tab completion to finish function names, file paths, arguments, and more.)
 - Line 14: `%% (10 ^ (n - 1))`
 - Line 15: `}}`
 - Line 17: `fo` (Multi-language code snippets to quickly use common blocks of code.)
 - Line 18: `for {snippet}`
 - Line 19: `foo {.GlobalEnv}`
 - Line 20: `force {base}`
 - Line 21: `Jump to function in file`
 - Line 22: `Change file type`
- Environment Panel:**
 - Buttons: `Load workspace`, `Save workspace`, `Delete all saved objects`, `Search inside environment`
 - Text: `Choose environment to display from list of parent environments`, `Display objects as list or grid`
 - Table:

Data	
iris	150 obs. of 5 variables
Values	
a	1
Functions	
foo	function (x)
 - Text: `Displays saved objects by type with short description`, `View in data viewer`, `View function source code`
- Files Panel:**
 - Buttons: `Create folder`, `Upload file`, `Delete file`, `Rename file`
 - Text: `Path to displayed directory`, `Change directory`
 - Table:

Name	Size	Date
..		
hello.R	450 B	Dec 24, 2015, 8:55 AM
 - Text: `A File browser keyed to your working directory. Click on file or directory name to open.`
- Console:**
 - Text: `> foo(1)`
 - Text: `[1] 2`
 - Text: `> foo <- function(x) x + 1`
 - Text: `> foo(2)`
 - Text: `> foo(2)`
 - Text: `> foo(1)`
 - Text: `Working Directory`
 - Text: `Maximize, minimize panes`
 - Text: `Press ↑ to see command history`
 - Text: `Drag pane boundaries`

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