THE UNIVERSITY OF TEXAS AT AUSTIN
McCOMBS SCHOOL OF BUSINESS

Introduction to R

Lecture 2

STA 371G

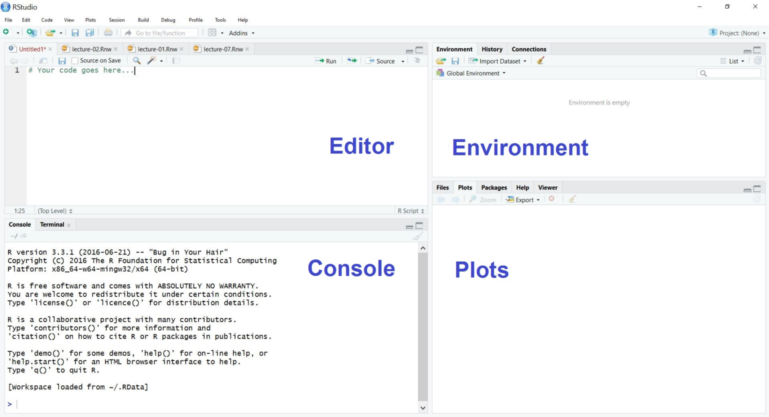
Again, what is R? What is RStudio?

R is the language, which we access through RStudio (interface).

Again, what is R? What is RStudio?

R is the language, which we access through RStudio (interface).

Here is what it looks like...



```
1  # Your code goes here...
```

```
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

[Workspace loaded from ~/.RData]

> 
```

RStudio Layout

- Console: This is where calculations/code are passed to R and results are observed.

RStudio Layout

- Console: This is where calculations/code are passed to R and results are observed.
- Editor: It is not practical to write long calculations/code in console. We write them in the editor, and "Run" to pass to the console.

RStudio Layout

- Console: This is where calculations/code are passed to R and results are observed.
- Editor: It is not practical to write long calculations/code in console. We write them in the editor, and "Run" to pass to the console.
- Environment: All data sets/variables we define can be found here.

RStudio Layout

- Console: This is where calculations/code are passed to R and results are observed.
- Editor: It is not practical to write long calculations/code in console. We write them in the editor, and "Run" to pass to the console.
- Environment: All data sets/variables we define can be found here.
- Plots: When we plot things, they will first appear here.

Let’s get started...

Assume you want to calculate your course grade.

Assignment	Weight	Grade
Class participation	5%	91
Reading assignments	5%	95
Homework	15%	86
Project	15%	83
Midterm 1	20%	88
Midterm 2	20%	76
Final exam	20%	84

Using the console

First try this in console.

```
>   0.05*91+0.05*95+0.15*86+0.15*83+0.2*88+0.2*76+0.2*84

[1] 84.25
```

Using the console

First try this in console.

```
>   0.05*91+0.05*95+0.15*86+0.15*83+0.2*88+0.2*76+0.2*84

[1] 84.25
```

It makes sense to save the result to a variable to be able to use later.

```
>   my371 <- 0.05*91+0.05*95+0.15*86+0.15*83+0.2*88+0.2*76+
```

Using the editor

It much convenient to do the calculations/coding in the editor and then "run" them.

Using the editor

It much convenient to do the calculations/coding in the editor and then "run" them.
Working with vectors is also common, which are simply data containers.

```
>   # This is the same calculation, using vectors.
>   weights <- c(0.05, 0.05, 0.15, 0.15, 0.2, 0.2, 0.2)
>   grades <- c(91, 95, 86, 83, 88, 76, 84)
>   weighted_grades <- weights*grades
>   my371 <- sum(weighted_grades)
```

The multiplication is element-wise.

Using the editor

It much convenient to do the calculations/coding in the editor and then "run" them.
Working with vectors is also common, which are simply data containers.

```
>   # This is the same calculation, using vectors.
>   weights <- c(0.05, 0.05, 0.15, 0.15, 0.2, 0.2, 0.2)
>   grades <- c(91, 95, 86, 83, 88, 76, 84)
>   weighted_grades <- weights*grades
>   my371 <- sum(weighted_grades)
```

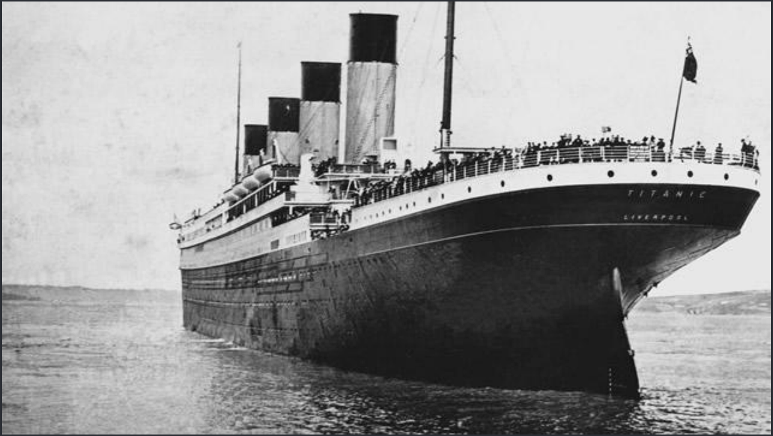
The multiplication is element-wise.
“sum” is a predefined function in R, which sums all the elements in a vector.

Working with tabular data

Many data sets we will work with are in tabular format, saved in .csv files.

Working with tabular data

Many data sets we will work with are in tabular format, saved in .csv files.
Let’s analyze the passenger data from the Titanic disaster.



Working with tabular data

In order to see the table, use "View(titanic)".

Working with tabular data

In order to see the table, use "View(titanic)".

"$" sign is used to refer to a particular column in the data, such as "titanic$Name".

Working with tabular data

In order to see the table, use "View(titanic)".

"$" sign is used to refer to a particular column in the data, such as "titanic$Name".

To access to an element in a particular position, e.g., row 1, column 4, use "titanic[1,4]".

Exploring Categorical Variables

The dataset has both quantitative and categorical data.

Exploring Categorical Variables

The dataset has both quantitative and categorical data.
Let’s explore the categorical variables through some frequency tables.

Exploring Categorical Variables

The dataset has both quantitative and categorical data.
Let’s explore the categorical variables through some frequency tables.
Below is the number of passengers in each class.

```
>   table(titanic$PClass)


1st 2nd 3rd
323 279 711
```

Exploring Categorical Variables

What is more interesting is how many people survived in each passenger class.

Exploring Categorical Variables

What is more interesting is how many people survived in each passenger class.

```
>   class_survival <- table(titanic$Survived, titanic$PClass)
>   class_survival


      1st 2nd 3rd
  No  130 160 573
  Yes 193 119 138
```

Exploring Categorical Variables

To get a better sense of the data, let’s calculate the survival percentage for each passenger class.

```
>   prop.table(class_survival,2)



           1st       2nd       3rd
  No  0.4024768 0.5734767 0.8059072
  Yes 0.5975232 0.4265233 0.1940928
```

Exploring Categorical Variables

To get a better sense of the data, let’s calculate the survival percentage for each passenger class.

```
>   prop.table(class_survival,2)


          1st       2nd       3rd
  No  0.4024768 0.5734767 0.8059072
  Yes 0.5975232 0.4265233 0.1940928
```

It looks like one’s chance of survival highly depended on his/her passenger class…

Slicing the data

One very common operation is slicing the data, i.e., selecting the portion that satisfy certain conditions.

Slicing the data

One very common operation is slicing the data, i.e., selecting the portion that satisfy certain conditions.

For example, we can select the rows that belong to female passenger data.

```
>    female_psg <- titanic[titanic$Sex=='female',]
```

Slicing the data

One very common operation is slicing the data, i.e., selecting the portion that satisfy certain conditions.

For example, we can select the rows that belong to female passenger data.

```
>    female_psg <- titanic[titanic$Sex=='female',]
```

This means: in the titanic dataset, select rows where the "Sex" is "female", select all columns, and save the resulting table to "female_psg" variable.

Slicing the data

We can create more complex conditions.

```
>   female_psg_1st <- titanic[(titanic$Sex=='female') &
+                                 (titanic$PClass=='1st'),]
```

Cleaning the data

If you want to analyze the "Age" data, you will realize rows with "NA", meaning Not Available.

Cleaning the data

If you want to analyze the "Age" data, you will realize rows with "NA", meaning Not Available.

Let’s select rows where we have age data available.

```
>   titanic_age <- titanic[!is.na(titanic$Age),]
```

Cleaning the data

If you want to analyze the "Age" data, you will realize rows with "NA", meaning Not Available.

Let’s select rows where we have age data available.

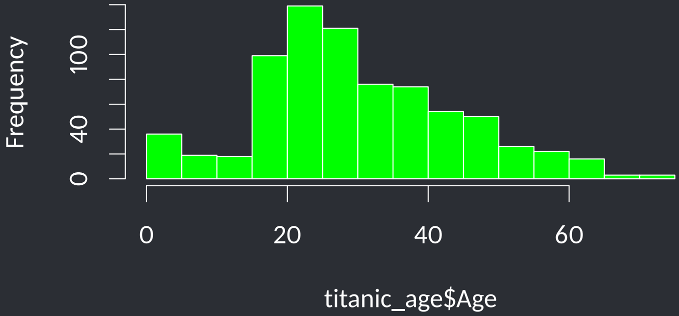
```
>   titanic_age <- titanic[!is.na(titanic$Age),]
```

This selects rows where the Age value is not "NA".

Exploring quantitative data

Let’s look into age distribution of the passengers.

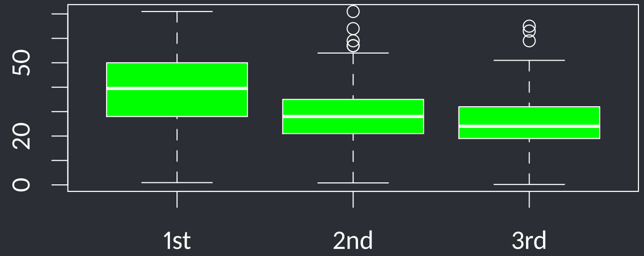
```
> hist(titanic_age$Age, col='green', main='')
```



Exploring quantitative data

Another way to look into it, by using a boxplot and compare between passenger classes.

```
> boxplot(Age ~ PClass, data=titanic, col='green', main='')
```


1st
2nd
3rd