

Lesson 1, part 1

Let's learn some R!

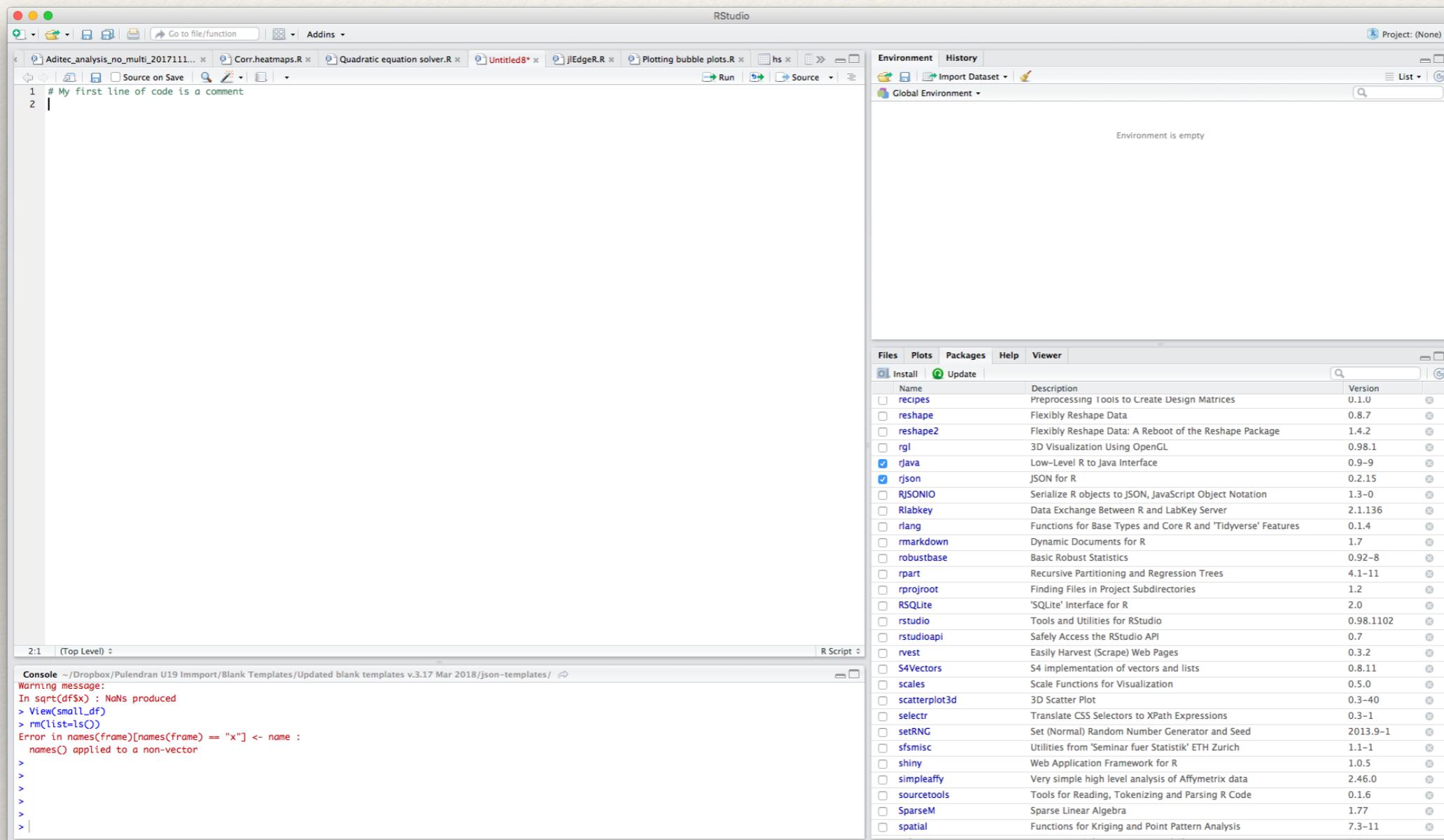
Variables
Functions
Indexing

The first line of code

This is the single most important line that you will ever type...

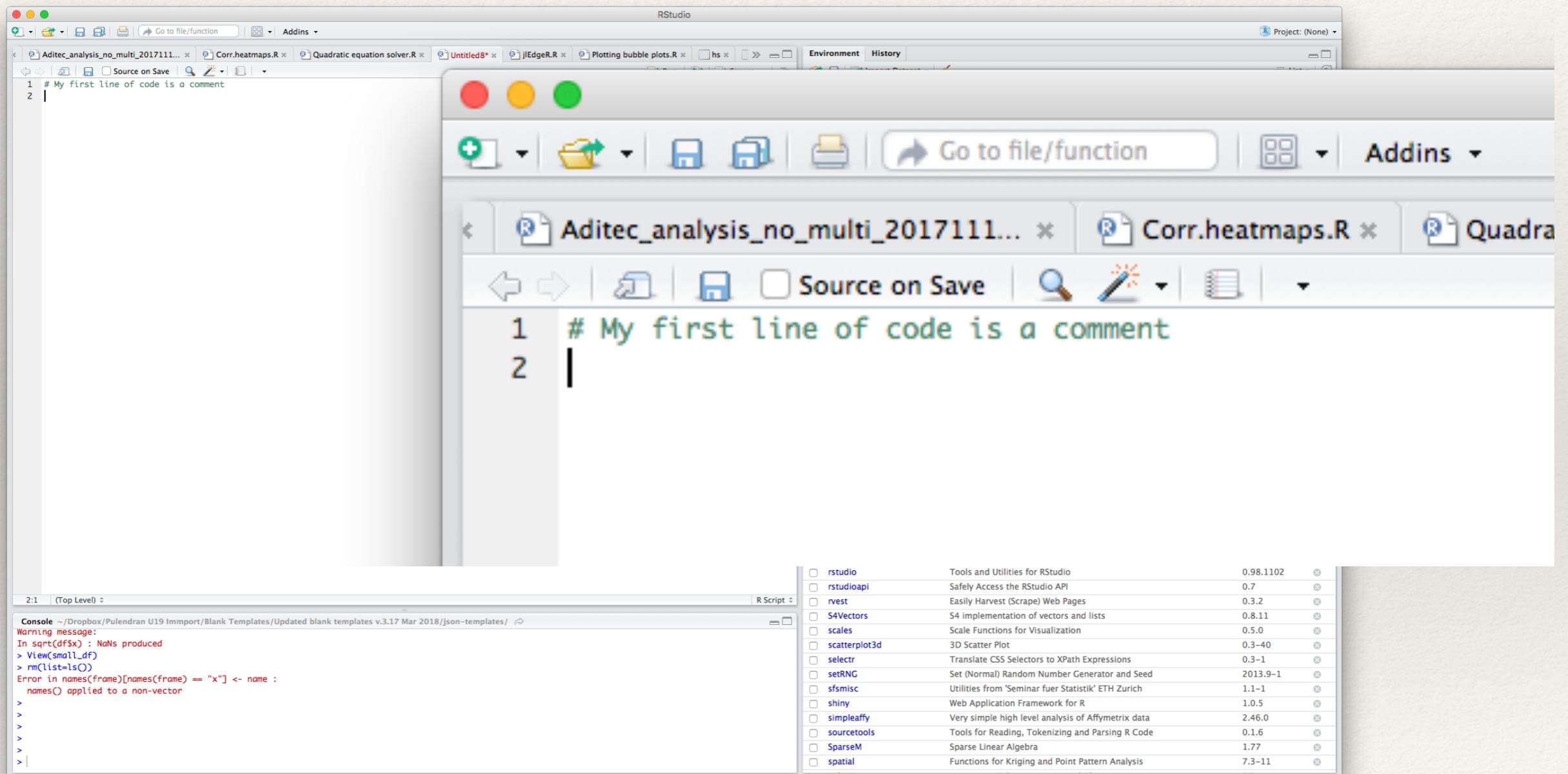
The first line of code

This is the single most important line that you will ever type...

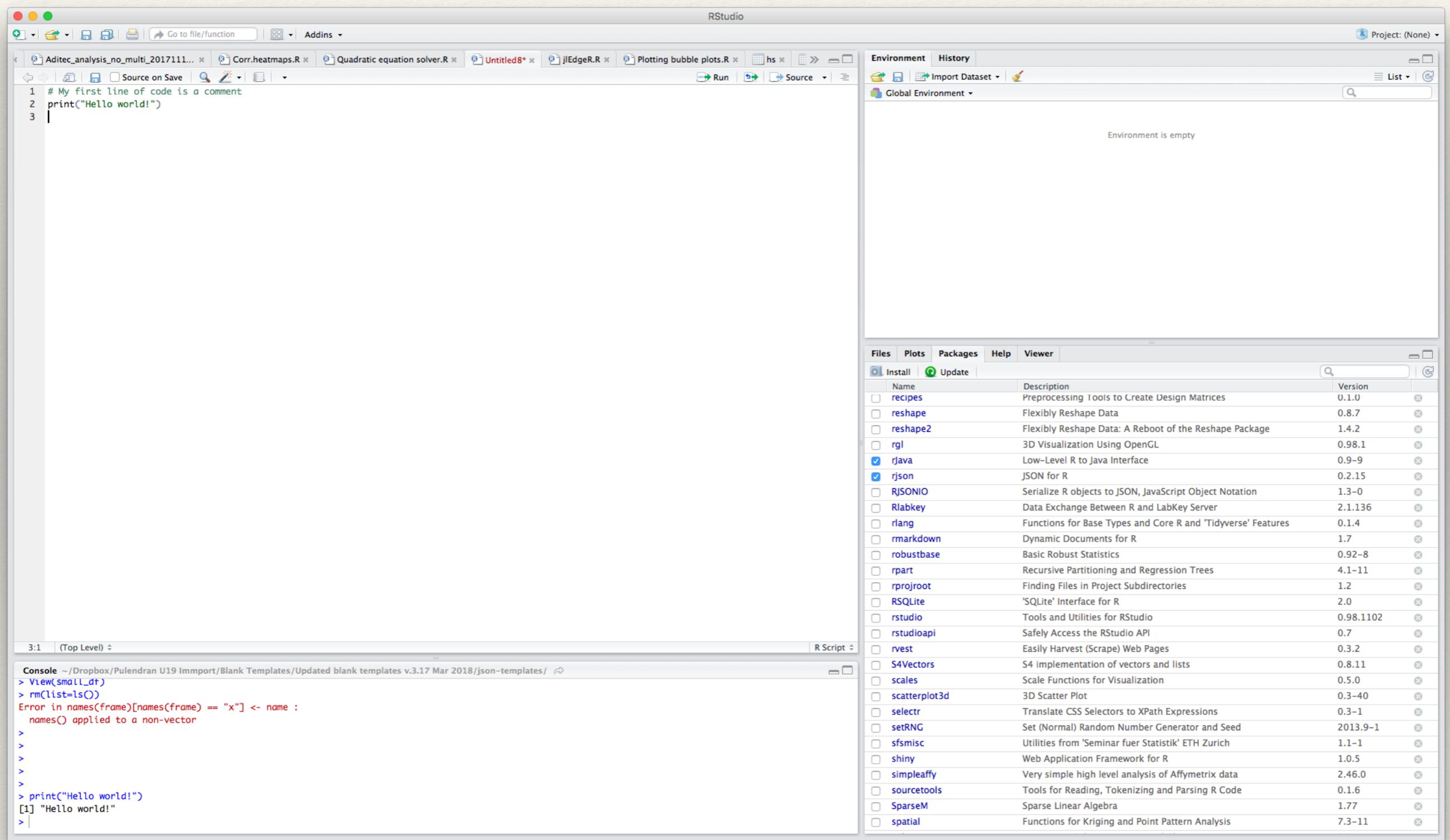


The first line of code

This is the single most important line that you will ever type...



My second line of code



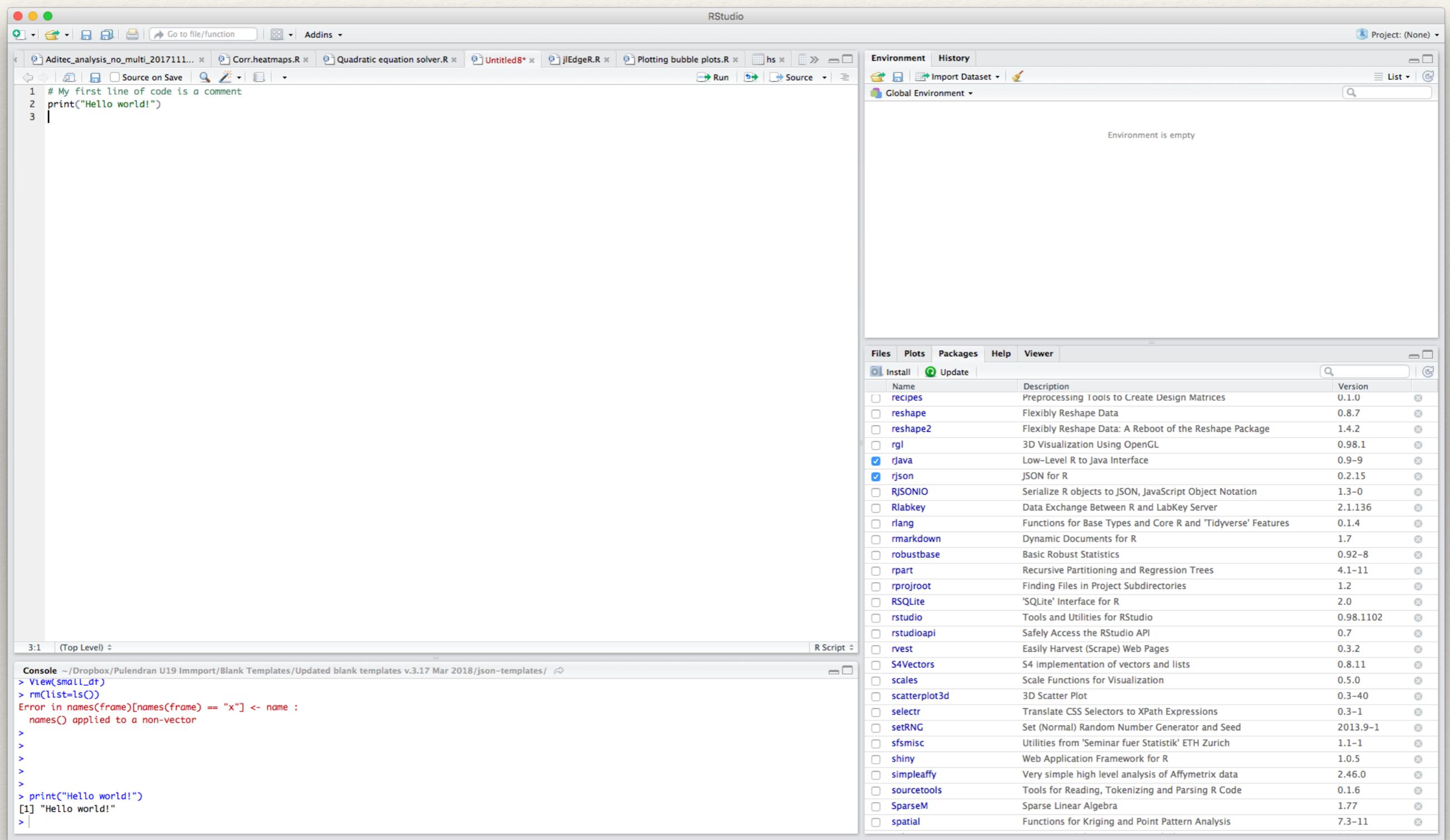
The screenshot shows the RStudio interface with the following components:

- Top Bar:** Includes standard OS X window controls (red, yellow, green buttons), a "Go to file/function" search bar, and an "Addins" dropdown.
- Left Sidebar:** Shows a tree view of files and projects, with "Aditec_analysis_no_multi_2017111..." selected. It also includes "Source on Save" and search/filter options.
- Script Editor:** Displays the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
```
- Console:** Shows the R session output:

```
3:1 | (Top Level) ▾
= R Script ▾
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/
> View(small_df)
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
```
- Package Browser:** A list of available packages under the "reshape2" category, including rgl, rJava, rjson, RJSONIO, Rlabkey, rlang, rmarkdown, robustbase, rpart, rprojroot, RSQLite, rstudio, rstudioapi, rvest, S4Vectors, scales, scatterplot3d, selectr, setRNG, sfsmisc, shiny, simpleaffy, sourcetools, SparseM, and spatial. Each entry shows the package name, description, version, and a "View Details" link.

My second line of code



My second line of code

The screenshot shows the RStudio interface. In the top-left pane, a script file named 'Untitled8.R' contains the following code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
```

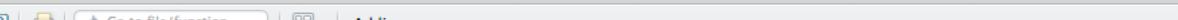
In the bottom-left pane, the R Console shows the following output:

```
3:1 | (Top Level) | R Script |
> View(small_df)
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
```

The right side of the interface includes the Environment pane (which says "Environment is empty"), the Packages pane (listing various R packages), and the Global Environment pane.

Name	Description	Version
recipes	Preprocessing Tools to Create Design Matrices	0.1.0
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11

My second line of code



The screenshot shows the RStudio interface. The top bar includes standard window controls (red, yellow, green), a title bar 'RStudio', and a menu bar with 'File', 'Edit', 'View', 'Tools', 'Help'. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a 'Go to file/function' search bar. A dropdown menu 'Addins' is open. The main area shows a code editor with the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
```

Below the code editor, a tab bar lists several open files: 'Aditec_analysis_no_multi_20171111...', 'Corr.heatmaps.R', 'Quadratic equation solver.R', 'Untitled8*', 'jEdgeR.R', 'Plotting bubble plots.R', and 'hs'. At the bottom right are buttons for 'Run', 'Source', and other navigation.

“Print” is a function.

The name of a function is always followed by parentheses ()

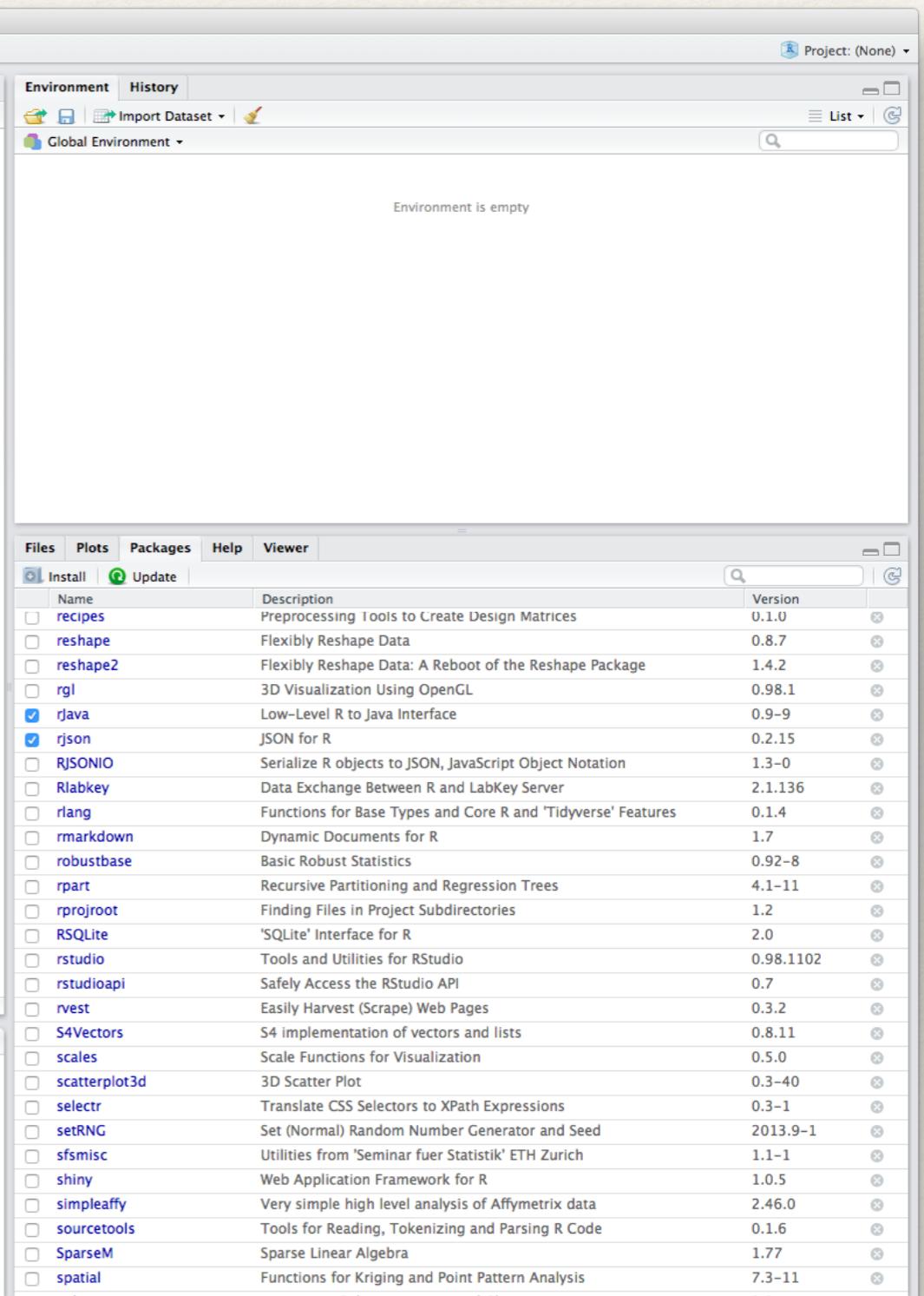
Parentheses contain ARGUMENTS to
the function

“Hello world!” is a STRING

Strings are a type of variables

When referring to a string, we always enclose it in quotes to indicate where it begins and where it ends. R does not make a distinction between single ‘ ’ and double “ ” quotes.

```
3:1 | (Top Level) | R Script  
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/  
> View(small_dt)  
> rm(list=ls())  
Error in names(frame)[names(frame) == "x"] <- name :  
  names() applied to a non-vector  
>  
>  
>  
>  
>  
> print("Hello world!")  
[1] "Hello world!"  
> |
```



My first variable

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays a script with the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
```
- Environment View:** Shows the global environment with one entry:

Values	My_first_variable	"Hello world!"
--------	-------------------	----------------
- Packages View:** Shows a list of installed packages in the RStudio interface.
- Console:** Displays the output of the R code execution, including error messages and the assignment of the variable.

Output from the Console:

```
6:1 | (Top Level) ▾
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> |
```

Mv first variable

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** The main window displays R code. The code includes comments and a print statement to output "Hello world!". It then creates a variable named "My_first_variable" and assigns it the value "Hello world!".
- Console:** The bottom-left window shows the R console output. It starts with some error messages related to frame names, followed by the execution of the R code from the editor, which prints "Hello world!" and creates the variable "My_first_variable".
- Packages:** The bottom-right window shows the "Packages" tab of the "Session" tab in the sidebar. It lists various R packages with their descriptions, versions, and install/update checkboxes. Several packages are checked for update, including rjava, rjson, RJSONIO, Rlabkey, rlang, rmarkdown, robustbase, rpart, rprojroot, RSQlite, rstudio, rstudioapi, rvest, S4Vectors, scales, scatterplot3d, selectr, setRNG, sfsmisc, shiny, simpleaffy, sourcetools, SparseM, and spatial.

My first variable

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays a script named "Untitled8.R" containing the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
```
- Environment View:** Shows the global environment with one entry:

Values	My_first_variable	"Hello world!"
--------	-------------------	----------------
- Packages View:** Shows a list of installed packages in the RStudio interface.
- Console:** Displays the output of the R code execution, including error messages and the assignment of the variable "My_first_variable".

```
6:1 | (Top Level) | R Script
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> |
```

My first variable

The screenshot shows the RStudio interface. In the top-left pane, there is a code editor with several tabs open. The active tab contains the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
```

In the top-right pane, the Environment tab is selected, showing the global environment. A table titled "Values" displays the variable "My_first_variable" with the value "Hello world!".

At the bottom of the interface, the R console window shows the output of the code execution:

```
6:1 | (Top Level) ▾
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> |
```

On the right side of the interface, there is a package manager window titled "Packages". It lists various R packages with their names, descriptions, and versions. The "rJava" package is checked for installation.

Name	Description	Version
recipes	Preprocessing Tools to Create Design Matrices	0.1.0
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
<input checked="" type="checkbox"/> rJava	Low-Level R to Java Interface	0.9-9
<input checked="" type="checkbox"/> rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11

My first variable

The screenshot shows the RStudio interface. In the top-left pane, there is a script editor with the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
```

In the top-right pane, the "Environment" window shows a table of variables:

Values	My_first_variable	"Hello world!"
--------	-------------------	----------------

At the bottom, the "Console" window shows the output of the code execution:

```
6:1 | (Top Level) | R Script
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/
> rm(list=ls())
Error in names(frame)[names(frame) == "x"] <- name :
  names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
>
```

The "Packages" window is also visible on the right side of the interface.

Referring to variables by name

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays an R script with the following content:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
```

- Environment View:** Shows the variable `My_first_variable` with the value `"Hello world!"`.
- Packages View:** Shows a list of installed packages, with `rJava` and `rjson` checked.
- Console View:** Displays the execution of the R script, showing the creation of the variable and its value.

Referring to variables by name

The screenshot shows the RStudio interface with three main panes:

- Code Editor:** Displays two R scripts side-by-side. Both scripts contain the same code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
```
- Console:** Shows the execution of the R code from the scripts. It includes a warning message and the output of the print statements.

```
9:1 (Top Level) ▾
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/ ↵
names() applied to a non-vector
>
>
>
>
>
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
>
```
- Package Manager:** Shows the available packages in the RStudio environment. The 'rJava' package is selected.

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Referring to variables by name

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays an R script with the following content:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
```

- Environment View:** Shows the variable `My_first_variable` with the value `"Hello world!"`.
- Packages View:** Shows a list of installed packages, with `rJava` and `rjson` checked.
- Console View:** Displays the execution of the R script, showing the creation of the variable and its value.

Referring to variables by name

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Shows R code in a script editor. Lines 1-8 are visible:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
```
- Environment:** Shows the global environment with one entry: `My_first_variable` with value `"Hello world!"`.
- Packages:** Shows a list of installed packages in RStudio's package manager.
- Console:** Shows the command history and output. It includes a warning about applying names() to a non-vector, followed by the execution of the script code, resulting in the output `[1] "Hello world!"`.

Text Content:

Here we have passed a variable as an argument to the “print” function.

The result is the printing of the contents of the variable.

Classes of variables

There are many different classes of variables: character arrays, numeric arrays, matrices, data frames, factors.

Class is often times also referred as data type.

There are the following data types in base R:

- Matrices
- Vectors
- Arrays
- Data frames
- Lists
- Factors

To find out the data type, we can use the **class()** function

Classes of variables

The screenshot shows the RStudio interface. In the top-left pane, there is an R script editor with the following code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
```

In the bottom-left pane, the R console shows the output of the code:

```
12:1 (Top Level) 
> 
> 
> 
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
>
```

In the top-right pane, the Environment tab of the Global Environment viewer shows a single variable:

Values	My_first_variable	"Hello world!"
--------	-------------------	----------------

In the bottom-right pane, the Packages tab of the Global Environment viewer lists available packages:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Classes of variables

The screenshot shows the RStudio interface. The top panel displays a script editor with the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
```

The bottom-left panel shows the R Console output:

```
12:1 (Top Level) 
> 
> 
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello worl!d"
> class(My_first_variable)
[1] "character"
```

The right side of the interface shows the 'Packages' tab of the 'Session' menu, listing various R packages with their descriptions and versions.

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Classes of variables

The screenshot shows the RStudio interface. In the top-left pane, there is an R script editor with the following code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
```

In the bottom-left pane, the R console shows the output of the code:

```
12:1 (Top Level) 
> 
> 
> 
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
>
```

In the top-right pane, the Environment tab of the Global Environment viewer shows a single variable:

Values	My_first_variable	"Hello world!"
--------	-------------------	----------------

In the bottom-right pane, the Packages tab of the Global Environment viewer lists available packages:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Let's create a more complex variable

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Shows R code for creating variables and printing "Hello world!".
- Environment:** Shows the global environment with two variables: `My_first_variable` and `My_second_variable`.
- Packages:** Shows a list of installed packages.
- Console:** Shows the output of the R code executed in the console.

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
```

```
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
> dim(My_first_variable)
NULL
> My_second_variable <- c("Hello", "world", "!")
>
```

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Let's create a more complex variable

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** Displays an R script with numbered comments and code. The code includes printing "Hello world!", defining a variable `My_first_variable`, printing its class, finding its dimensions, and defining a more complex variable `My_second_variable`.
- Console:** Shows the output of the R code run in the console.
- Addins:** A list of available packages for installation, including `rjson`, `RJSONIO`, `Rlabkey`, `rlang`, `rmarkdown`, `robustbase`, `rpart`, `rprojroot`, `RSQlite`, `rstudio`, `rstudioapi`, `rvest`, `S4Vectors`, `scales`, `scatterplot3d`, `selectr`, `setRNG`, `sfsmisc`, `shiny`, `simpleaffy`, `sourcetools`, `SparseM`, `spatial`, and `splines`. `rjson` is checked.

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable.
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world","!")
18
```

```
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
> dim(My_first_variable)
NULL
> My_second_variable <- c("Hello", "world","!")
>
```

Let's create a more complex variable

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Shows R code for creating variables and printing "Hello world!".
- Environment:** Shows the global environment with two variables: `My_first_variable` and `My_second_variable`.
- Packages:** Shows a list of installed packages.
- Console:** Shows the output of the R code executed in the console.

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
```

```
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
> dim(My_first_variable)
NULL
> My_second_variable <- c("Hello", "world", "!")
>
```

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Let's create a more complex variable

The screenshot shows the RStudio interface. On the left, the code editor displays a script with R code. The code includes comments, variable assignment, and function calls. On the right, the Environment pane shows two variables: `My_first_variable` and `My_second_variable`. The `My_first_variable` is a character string "Hello world!". The `My_second_variable` is a character vector `chr [1:3] "Hello" "world" "!"`. Below the environment pane, the Packages tab of the global toolbar is selected, showing a list of installed packages with their names, descriptions, and versions.

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
```

Here we have created a character VECTOR, consisting of three elements

The accessory function `c()` is used to concatenate its arguments into a single array

```
> print("Hello world!")
[1] "Hello world!"
> My_first_variable <- "Hello world!"
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
> dim(My_first_variable)
NULL
> My_second_variable <- c("Hello", "world", "!")
>
```

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Dimensions and length of a vector

Vectors do not have dimensions:
`dim(My_second_variable)` produces NULL output

However, vectors have length:
`length(My_second_variable)` produces output:

Dimensions and length of a vector

The screenshot shows the RStudio interface with several panes:

- Code pane:** Displays R code for creating variables and calculating dimensions.
- Environment pane:** Shows the values of created variables.
- Packages pane:** Lists installed packages.
- Console pane:** Displays the R session history.

Code pane content:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
19 # We can find out the dimensions of the new variable: However it wil not be 1 by 3, as vetors do not have dimensions
20 dim(My_second_variable)
21
22 # However, vectors have length, which can be found out by using the length() function
23 length(My_second_variable)
24
```

Environment pane:

Values	
My_first_variable	"Hello world!"
My_second_variable	chr [1:3] "Hello" "world" "!"

Packages pane:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Console pane:

```
24:1 (Top Level) ▾
Console ~/Dropbox/Pulendran U19 Import/Blank Templates/Updated blank templates v.3.17 Mar 2018/json-templates/ ▾
> print(My_first_variable)
[1] "Hello world!"
> class(My_first_variable)
[1] "character"
> dim(My_first_variable)
NULL
> My_second_variable <- c("Hello", "world", "!")
> dim(My_second_variable)
NULL
> length(My_second_variable)
[1] 3
>
```

Dimensions and length of a vector

Vectors do not have dimensions:
`dim(My_second_variable)` produces NULL output

However, vectors have length:
`length(My_second_variable)` produces output:

Referring to the elements of a vector

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays R code demonstrating how to create variables and access their elements. The code includes comments explaining each step.
- Environment View:** Shows the global environment with variables defined:

Variable	Value
My_first_variable	"Hello world!"
My_second_variable	chr [1:3] "Hello" "world" "!"
one	"Hello"
three	"!"
two	"world"

- Packages View:** Shows the installed packages and their details:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

- Console:** Displays the R session history, showing the execution of the code and its output.

Putting the elements back together

We can do the inverse trick:
put the three variables back together into a character vector, and even add something to it:

Putting the elements back together

The screenshot shows the RStudio interface with several tabs open in the top bar: Aditec_analysis_no_multi_2017111... (active), Corr.heatmaps.R, Quadratic equation solver.R, R lesson 1.R*, jlEdgeR.R, Plotting bubble plots.R, hs, and Run.

The left pane displays an R script with numbered comments and code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
19 # We can find out the dimensions of the new variable: However it wil not be 1 by 3, as vetors do not have dimensions
20 dim(My_second_variable)
21
22 # However, vectors have length, which can be found out by using the length() function
23 length(My_second_variable)
24
25 # Now let's refer to the individual elements of the character vector:
26 My_second_variable[1] # This will display the contents of the first element in the command line window
27 one <- My_second_variable[1] # This will assign the contents of the first element of the My_second_variable to the new variable, "one"
28
29 # And likewise we can do for the second and the third elements:
30 My_first_variable[2]
31 two <- My_second_variable[2]
32 My_second_variable[3]
33 three <- My_second_variable[3]
34
35 # We can put the three variables back together into a character variable, and can also add something to it:
36 My_third_variable <- c(one, two, three, "My name is Vanya", "I am", 38, "years", "old")
```

The right pane shows the Environment tab with the Global Environment table:

Values	
My_first_variable	"Hello world!"
My_second_variable	chr [1:3] "Hello" "world" "!"
My_third_variable	chr [1:8] "Hello" "world" "!" "My name is Vanya" "I am" "38" "years" "old"
one	"Hello"
three	"!"
two	"world"

The Packages tab shows a list of installed packages:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rjava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

The bottom pane shows the Console output:

```
>
>
>
>
>
>
> My_third_variable
[1] "Hello"      "world"      "!"           "My name is Vanya" "I am"       "38"          "years"      "old"
[8] "old"
>
```

Putting the elements back together

This new character vector consists of 8 elements. Importantly, even though the 6th element of this vector is a number 38, it is still stored as text, which can be confirmed if you type `class(My_third_variable[6])`

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_var)
9
10 # Let's find out what class it is
11 class(My_first_variable)
12
13 # We can also find out its dimension
14 dim(My_first_variable)
15
16 # Let's create another variable
17 My_second_variable <- "world"
18
19 # We can find out its dimension
20 dim(My_second_variable)
21
22 # However, vector length is different
23 length(My_second_variable)
24
25 # Now let's refer to the second variable
26 My_second_variable
27 one <- My_second_variable
28
29 # And likewise we can do for the third variable
30 My_first_variable
31 two <- My_first_variable
32 My_second_variable
33 three <- My_second_variable
34
35 # We can put them all together
36 My_third_variable <- c(one, two, three)
```

Environment

Values	Global Environment
My_first_variable	"Hello world!"
My_second_variable	chr [1:3] "Hello" "world" !"
My_third_variable	chr [1:8] "Hello" "world" !" "My name is Vanya" "I am" "38" "years" "old"
one	"Hello"
three	!"
two	"world"

Packages

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rjava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Can we output the contents of a character vector as a printed line?

The screenshot shows the RStudio interface with several tabs open in the top bar: Aditec_analysis_no_multi_2017111...*, Corr.heatmaps.R, Quadratic equation solver.R, R lesson 1.R*, JIEdgeR.R, Plotting bubble plots.R, hs, Run, Source, and Environment.

The left pane displays the following R code:

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
19 # We can find out the dimensions of the new variable: However it wil not be 1 by 3, as vetors do not have dimensions
20 dim(My_second_variable)
21
22 # However, vectors have length, which can be found out by using the length() function
23 length(My_second_variable)
24
25 # Now let's refer to the individual elements of the character vector:
26 My_second_variable[1] # This will display the contents of the first element in the command line window
27 one <- My_second_variable[1] # This will assign the contents of the first element of the My_second_variable to the new variable, "one"
28
29 # And likewise we can do for the second and the third elements:
30 My_first_variable[2]
31 two <- My_second_variable[2]
32 My_second_variable[3]
33 three <- My_second_variable[3]
34
35 # We can put the three variables back together into a character variable, and can also add something to it:
36 My_third_variable <- c(one, two, three, "My name is Vanya", "I am", 38, "years", "old")
37
38 # Can we print the contents of a character array as a readable text?
39 print(My_third_variable) # Ugh... ths is not what we want
```

The right pane shows the Environment tab with the following variable values:

Values	My_first_variable	"Hello world!"
My_second_variable	chr [1:3]	"Hello" "world" "!"
My_third_variable	chr [1:8]	"Hello" "world" "!" "My name is Vanya" "I am" "38" "years" "old"
one		"Hello"
three		"!"
two		"world"

The bottom pane shows the Console tab with the following output:

```
>
>
> My_third_variable
[1] "Hello"      "world"      "!"          "My name is Vanya" "I am"      "38"        "years"      "old"
[8] "old"
> class(My_third_variable[6])
[1] "character"
> print(My_third_variable)
[1] "Hello"      "world"      "!"          "My name is Vanya" "I am"      "38"        "years"      "old"
[8] "old"
>
```

The Packages tab lists the following installed packages:

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.9.8.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSQlite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Using cat() function, and the first case of function modifiers

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays an R script with numbered lines of code. The code demonstrates variable assignment, printing, and the use of the `cat()` function.
- Environment:** Shows the global environment with variables and their values.
- Packages:** Shows a list of installed packages with their descriptions and versions.
- Console:** Shows the output of the R code, including the results of `cat()` and the final printed text.

```
1 # My first line of code is a comment
2 print("Hello world!")
3
4 # Now we are going to create the first variable. Let's call it "My_first_variable"
5 My_first_variable <- "Hello world!"
6
7 # We can make functions to take variables as arguments
8 print(My_first_variable) # We do not enclose variable names in quotes
9
10 # Let's find out the class of the My_first_variable variable
11 class(My_first_variable)
12
13 # We can also find out the dimensions of the variable (in this case, the length of the vector)
14 dim(My_first_variable) # Which is "NULL", indicating that this variable has no dimensions
15
16 # Let's create a more complex variable:
17 My_second_variable <- c("Hello", "world", "!")
18
19 # We can find out the dimensions of the new variable: However it wil not be 1 by 3, as vetors do not have dimensions
20 dim(My_second_variable)
21
22 # However, vectors have length, which can be found out by using the length() function
23 length(My_second_variable)
24
25 # Now let's refer to the individual elements of the character vector:
26 My_second_variable[1] # This will display the contents of the first element in the command line window
27 one <- My_second_variable[1] # This will assign the contents of the first element of the My_second_variable to the new variable, "one"
28
29 # And likewise we can do for the second and the third elements:
30 My_first_variable[2]
31 two <- My_second_variable[2]
32 My_second_variable[3]
33 three <- My_second_variable[3]
34
35 # We can put the three variables back together into a character variable, and can also add something to it:
36 My_third_variable <- c(one, two, three, "My name is Vanya", "I am", 38, "years", "old")
37
38 # Can we print the contents of a character array as a readable text?
39 print(My_third_variable) # Ugh... ths is not what we want
40 # but what if we try it with cat()? (Anotehr concatenation function. The same function also exists in Bash)
41 cat(My_third_variable, "\n", sep=" ")
42
43
```

Console output:

```
>
>
>
>
>
>
>
> cat(My_third_variable, "\n", sep=" ")
Hello world ! My name is Vanya I am 38 years old
>
```

Name	Description	Version
reshape	Flexibly Reshape Data	0.8.7
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rgl	3D Visualization Using OpenGL	0.98.1
rJava	Low-Level R to Java Interface	0.9-9
rjson	JSON for R	0.2.15
RJSONIO	Serialize R objects to JSON, JavaScript Object Notation	1.3-0
Rlabkey	Data Exchange Between R and LabKey Server	2.1.136
rlang	Functions for Base Types and Core R and 'Tidyverse' Features	0.1.4
rmarkdown	Dynamic Documents for R	1.7
robustbase	Basic Robust Statistics	0.92-8
rpart	Recursive Partitioning and Regression Trees	4.1-11
rprojroot	Finding Files in Project Subdirectories	1.2
RSSQLite	'SQLite' Interface for R	2.0
rstudio	Tools and Utilities for RStudio	0.98.1102
rstudioapi	Safely Access the RStudio API	0.7
rvest	Easily Harvest (Scrape) Web Pages	0.3.2
S4Vectors	S4 implementation of vectors and lists	0.8.11
scales	Scale Functions for Visualization	0.5.0
scatterplot3d	3D Scatter Plot	0.3-40
selectr	Translate CSS Selectors to XPath Expressions	0.3-1
setRNG	Set (Normal) Random Number Generator and Seed	2013.9-1
sfsmisc	Utilities from 'Seminar fuer Statistik' ETH Zurich	1.1-1
shiny	Web Application Framework for R	1.0.5
simpleaffy	Very simple high level analysis of Affymetrix data	2.46.0
sourcetools	Tools for Reading, Tokenizing and Parsing R Code	0.1.6
SparseM	Sparse Linear Algebra	1.77
spatial	Functions for Kriging and Point Pattern Analysis	7.3-11
splines	Regression Spline Functions and Classes	3.2.4

Using cat() function, and the first case of function modifiers

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Shows a script with several lines of R code. Lines 38 and 39 are highlighted in blue, indicating they are being executed.
- Console:** Displays the output of the executed code:

```
> cat(My_third_variable, "\n", sep=" ")
Hello world ! My name is Vanya I am 38 years old
>
```
- Environment:** Shows the current environment variables and their values:

	Values
My_first_variable	"Hello world!"
My_second_variable	chr [1:3] "Hello" "world" "!"
My_third_variable	chr [1:8] "Hello" "world" "!" "My name is Vanya" "I am" "38" "years" "old"
one	"Hello"
- Session View:** Shows the session history with the same output as the console.
- Help:** A sidebar showing available packages and their descriptions.

Using cat() function, and the first case of function modifiers

Cat() function concatenates all the elements listed before the modifiers, and separated by commas.

In our case, we have the first element “My_third_variable” variable, which itself consists of 8 elements, and the newline character, \n, enclosed in quotes, to indicate that it is a string, and not a variable.

A backslash in front of “n” is a very special symbol called “an escape character”. It tells the program to treat whatever follows as a “special character”, and not as a regular letter.

For instance, “n” is just a letter n, but \n is a command to insert a new line (like pressing “enter” on a computer). Likewise, “t” is just a letter t, but “\t” becomes a command to insert a tab (like pressing a tab button on a computer).

Finally, the list of things to concatenate is followed by a function MODIFIER. In this case, it says sep=“ ”, which indicates that the elements in the resulting string should be separated from each other by a white space.

We could make this separator to be anything we want, for instance, an underscore, or a word “cooties”:

Using cat() function, and the first case of function modifiers

Cat() function concatenates all the elements listed before the modifiers, and separated by commas.

In our case, we have the first element “My_third_variable” variable, which itself consists of 8 elements, and the newline character, \n, enclosed in quotes, to indicate that it is a string, and not a variable.

```
> 
> cat(My_third_variable, "\n", sep=" ")
Hello world ! My name is Vanya I am 38 years old
> cat(My_third_variable, "\n", sep="COOTIES")
HelloCOOTIESworldCOOTIES!COOTIESMy name is VanyaCOOTIESI amCOOTIES38COOTIESyearsCOOTIESoldCOOTIES
> |
```

For instance, “n” is just a letter n, but “\n” is a command to insert a new line (like pressing “enter” on a computer). Likewise, “t” is just a letter t, but “\t” becomes a command to insert a tab (like pressing a tab button on a computer).

Finally, the list of things to concatenate is followed by a function MODIFIER. In this case, it says `sep=" "`, which indicates that the elements in the resulting string should be separated from each other by a white space.

We could make this separator to be anything we want, for instance, an underscore, or a word “cooties”:

Using paste() function to assign concatenated strings to variables

Paste() works in a way similar to cat(), but they serve different purposes. Cat() is used to output the string to the screen, while paste() works to output the resulting string to a variable.

For instance:

```
>  
> mitia <- paste("Dmitri", "Kazmin", sep=" ")  
> mitia  
[1] "Dmitri Kazmin"  
> |
```

Summary

- ❖ Today we have gone through a lot of introductory material
- ❖ We have learned about functions, arguments to functions, and function modifiers
- ❖ We have learned about variables and saw an example of one kind of a variable - a character vector
- ❖ We learned about addressing elements of a vector by their index
- ❖ We learned about functions print(), c(), dim(), length(), cat() and paste().