

Analytics of Business Intelligence

Chapter # 1 - Extract, Transform, and Load

Dr. Wajahat Gilani

Rutgers Business School

October 14, 2019

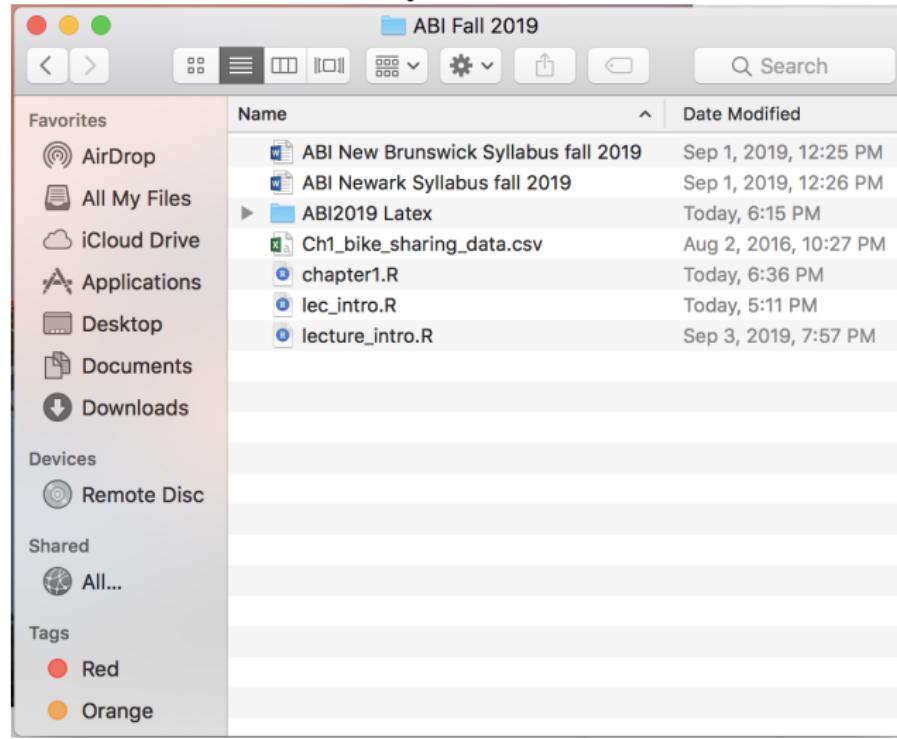
Extracting data from Sources

We will be using the text file `Ch1_bike_sharing_data.csv` and focus on:

- Importing CSV and other file formats
- Introducing the data container `data.table`
- Manipulating data in a `data.table` container

Importing CSV and other file formats

Download the file into your class folder.



Importing CSV and other file formats

Importing data using "Import Dataset" from the top right window

The screenshot shows the RStudio interface. On the left is the code editor with a script named 'chapter1.R'. The console below it contains R code and its output. On the right is the 'Environment' tab of the global pane, which has a context menu open over the 'Import Dataset' button. The menu items are: From Text (base)..., From Text (readr)..., From Excel..., From SPSS..., From SAS..., and From Stata... A blue arrow points from the text above to this menu. Below the global pane is the 'System Library' pane, which lists various R packages.

| Name | Description | Version |
|------------|--|----------|
| assertthat | Easy Pre and Post Assertions | 0.2.0 |
| base64enc | Tools for base64 encoding | 0.1-3 |
| BH | Boost C++ Header Files | 1.66.0-1 |
| bindr | Parametrized Active Bindings | 0.1.1 |
| bindrcpp | An 'Rcpp' Interface to Active Bindings | 0.2.2 |
| boot | Bootstrap Functions (Originally by Angelo Canty for S) | 1.3-20 |
| class | Functions for Classification | 7.3-14 |
| cli | Helpers for Developing Command Line Interfaces | 1.0.1 |
| cluster | "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. | 2.0.7-1 |
| codetools | Code Analysis Tools for R | 0.2-15 |
| colorspace | Color Space Manipulation | 1.3-2 |
| compare | Comparing Objects for Differences | 0.2-6 |
| compiler | The R Compiler Packaoe | 3.5.1 |

Importing CSV and other file formats

Go to the folder where you downloaded [Ch1_bike_sharing_data.csv](#) and click on "Open".

The screenshot shows the RStudio interface. The top menu bar includes 'File', 'Edit', 'View', 'Code', 'Tools', 'Help', 'Go to file/function', 'Addins', 'Environment', 'History', and 'Connections'. The 'Project' dropdown shows '(None)'. The left sidebar has sections for 'Favorites' (All My Files, iCloud Drive, Applications, Desktop, Documents, Downloads), 'Devices' (Remote Disc), and 'Shared' (All...). The main area shows a file tree under 'Management' for 'ABI Fall 2019'. A preview window on the right displays a grid of data with the label 'CSV'. The preview window title is 'Ch1_bike_sharing_data'. Below it, file details are shown: .CSV, 957 KB, Created 8/2/16, 10:27 PM, Modified 8/2/16, 10:27 PM, Last opened 8/2/16, 10:27 PM, and an 'Add Tags...' button. A scroll bar indicates there are more items. At the bottom, a list of packages is visible:

- cli
- cluster
- codetools
- colorspace
- compare
- compiler
- Helpers for Developing Command Line Interfaces
- "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.
- Code Analysis Tools for R
- Color Space Manipulation
- Comparing Objects for Differences
- The R Compiler Package

The bottom status bar shows navigation icons and the text '1.3 - 20'.

```
grep(pattern, all.names, value = TRUE)
}
else all.names
}

```

Importing CSV and other file formats

Uncheck the box that says: "Strings as factors", then click "Import"

The screenshot shows the RStudio interface with the 'chapter1.R' project open. The 'Import Dataset' dialog box is centered. On the left, there are settings for 'Name' (Ch1_bike_sharing_data), 'Encoding' (Automatic), 'Heading' (Yes), 'Row names' (Automatic), 'Separator' (Comma), 'Decimal' (Period), 'Quote' (Double quote (")), 'Comment' (None), and 'na.strings' (NA). A blue arrow points to the 'Strings as factors' checkbox, which is checked. On the right, the 'Input File' section shows the first few lines of the CSV data, and below it, the 'Data Frame' section shows the actual data structure with columns: datetime, season, holiday, workingday, weather. At the bottom right of the dialog, the 'Import' button is highlighted.

Uncheck the box

Import

| datetime | season | holiday | workingday | weather |
|----------------|--------|---------|------------|---------|
| 1/1/2011 0:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 1:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 2:00 | 1 | 0 | 1 | 0 |
| 1/1/2011 3:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 4:00 | 1 | 0 | 1 | 0 |
| 1/1/2011 5:00 | 1 | 0 | 0 | 2 |
| 1/1/2011 6:00 | 1 | 0 | 1 | 0 |
| 1/1/2011 7:00 | 1 | 0 | 2 | 1 |
| 1/1/2011 8:00 | 1 | 0 | 1 | 0 |
| 1/1/2011 9:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 10:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 11:00 | 1 | 0 | 0 | 1 |
| 1/1/2011 12:00 | 1 | 0 | 0 | 1 |

Importing CSV and other file formats

Table is loaded in the top right window and it's opened to view.

The screenshot shows the RStudio interface with several windows open:

- Environment** pane (top right): Shows the "Ch1_bike_sharing_data" dataset with 17379 observations and 12 variables. The variables include datetime, season, holiday, workingday, weather, temp, atemp, humidity, windspeed, and casual.
- Console** pane (bottom left): Displays the R code used to load the data and some initial exploratory commands.
- Code Editor** pane (middle left): Shows the same R code as the Console.
- Help** pane (bottom right): Shows the "System Library" with various R packages listed by name, description, and version.

```
Console | Terminal x
~/Documents/Investment in R/data/ ↵
}
<bytecode: 0x102b40538>
<environment: namespace:base>
> ls()
character(0)
> Ch1_bike_sharing_data <- read.csv("~/Documents/ABI/ABI Fall 2019/Ch1_bike_sharing_data.csv", stringsAsFactors=FALSE)
> View(Ch1_bike_sharing_data)
> |
```

| Variable | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| datetime | 1/1/2011 0:00 | 1/1/2011 1:00 | 1/1/2011 2:00 | 1/1/2011 3:00 | 1/1/2011 4:00 | 1/1/2011 5:00 | 1/1/2011 6:00 | 1/1/2011 7:00 | 1/1/2011 8:00 | 1/1/2011 9:00 | 1/1/2011 10:00 | 1/1/2011 11:00 | 1/1/2011 12:00 | 1/1/2011 13:00 | 1/1/2011 14:00 | 1/1/2011 15:00 | 1/1/2011 16:00 | 1/1/2011 17:00 |
| season | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| holiday | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| workingday | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| weather | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| temp | 9.84 | 9.02 | 9.02 | 9.84 | 9.84 | 9.84 | 9.02 | 8.20 | 9.84 | 13.12 | 15.58 | 14.76 | 17.22 | 18.86 | 18.86 | 18.04 | 17.22 | 18.04 |
| atemp | 14.395 | 13.635 | 13.635 | 14.395 | 14.395 | 12.880 | 13.635 | 12.880 | 14.395 | 17.425 | 19.695 | 16.665 | 21.210 | 22.725 | 22.725 | 21.970 | 21.210 | 21.070 |
| humidity | 81 | 80 | 80 | 75 | 75 | 75 | 80 | 86 | 75 | 76 | 76 | 81 | 77 | 72 | 72 | 77 | 82 | 82 |
| windspeed | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 6.0032 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 16.9979 | 19.0012 | 19.0012 | 19.9995 | 19.9995 | 19.9995 | 19.9995 | 19.0012 |
| casual | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

Importing CSV and other file formats

Notice the functions:

`read.csv()`

The screenshot shows the RStudio interface with the following components:

- Environment View:** Shows the global environment with the dataset `Ch1_bike_sharing_data` containing 17,379 observations and 12 variables.
- Data View:** Displays the first 18 rows of the `Ch1_bike_sharing_data` dataset.
- Console View:** Shows the R code used to load the dataset from a CSV file located at `~/Documents/ABI_Fall_2019/Ch1_bike_sharing_data.csv`.
- Code Editor:** Shows the R code for the `View` function applied to the dataset.

`View()`

Reading from Excel

The screenshot shows the RStudio interface with the following details:

- Title Bar:** chapter1.R, Ch1_bike_sharing_data.xsls, Go to file/function, Addins, Project: (None)
- Environment Tab:** Shows the first few rows of the 'datetime' column.
- Data Preview:** A large table showing the first 50 rows of the dataset. The columns include: datetime, season, holiday, workingday, weather, temp, atemp, humidity, windspeed, and casual.
- Import Options:**
 - Name: Ch1_bike
 - Max Rows: [empty]
 - First Row as Names:
 - Sheet: Default
 - Skip: 0
 - Open Data Viewer:
 - Range: A1:D10
 - NA: [empty]
- Code Preview:**

```
library(readxl)
Ch1_bike <- read_excel("~/Documents/ABI/ABI Fall 2019/Ch1_bike.xls")
View(Ch1_bike)
```
- Buttons:** Import, Cancel
- Version History:** A sidebar showing version history from 0.2.0 to 0.2.2.
- Help:** A link to "Reading Excel files using readxl".
- Console:** Shows the path ~/Documents/ABI/ABI Fall 2019/Ch1_bike.xls and some command history.

```
library(readxl)
read_excel()
```

3rd-Party Library Packages

The screenshot shows the RStudio interface with several panes:

- Environment** pane: Shows two datasets: Ch1_bike (17379 obs. of 12 variables) and Ch1_bike_sh... (17379 obs. of 12 variables).
- Packages** tab: Selected in the bottom navigation bar.
- System Library**: A table listing available packages:

| Name | Description | Version |
|------------|--|----------|
| assertthat | Easy Pre and Post Assertions | 0.2.0 |
| base64enc | Tools for base64 encoding | 0.1-3 |
| BH | Boost C++ Header Files | 1.66.0-1 |
| bindr | Parametrized Active Bindings | 0.1.1 |
| bindrccpp | An 'Rcpp' Interface to Active Bindings | 0.2.2 |
| boot | Bootstrap Functions (Originally by Angelo Canty for S) | 1.3-20 |
| class | Functions for Classification | 7.3-14 |
| cli | Helpers for Developing Command Line Interfaces | 1.0.1 |
| cluster | "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. | 2.0.7-1 |
| codetools | Code Analysis Tools for R | 0.2-15 |
| colorspace | Color Space Manipulation | 1.3-2 |
| compare | Comparing Objects for | 0.2-6 |

Arrows indicate the flow from the code in the Console to the resulting objects in the Environment pane, and from the Environment pane to the selected Packages tab.

Console pane (bottom-left):

```
~/Documents/Investment in R/data/ ↵
You currently have tibble v1.4.2.
Falling back to column name repair from tibble <= v1.4.2.
Message displays once per session.
> View(Ch1_bike)
> class(Ch1_bike)
[1] "tbl_df"     "tbl"        "data.frame"
> class(Ch1_bike_sharing_data)
[1] "data.frame"
>
```

Environment pane (top-right):

```
chapter1.R x Ch1_bike_sharing_data x Ch1_bike x
Go to file/function Addins Project: (None) ↵
Filter
datetime season holiday workingday weather temp atemp humidity windspeed casual
1 1/1/2011 0:00 1 0 0 1 9.84 14.395 81 0.0000
2 1/1/2011 1:00 1 0 0 1 9.02 13.635 80 0.0000
3 1/1/2011 2:00 1 0 0 1 9.02 13.635 80 0.0000
4 1/1/2011 3:00 1 0 0 1 9.84 14.395 75 0.0000
5 1/1/2011 4:00 1 0 0 1 9.84 14.395 75 0.0000
6 1/1/2011 5:00 1 0 0 2 9.84 12.880 75 6.0032
7 1/1/2011 6:00 1 0 0 1 9.02 13.635 80 0.0000
8 1/1/2011 7:00 1 0 0 1 8.20 12.880 86 0.0000
9 1/1/2011 8:00 1 0 0 1 9.84 14.395 75 0.0000
10 1/1/2011 9:00 1 0 0 1 13.12 17.425 76 0.0000
11 1/1/2011 10:00 1 0 0 1 15.58 19.695 76 16.9979
12 1/1/2011 11:00 1 0 0 1 14.76 16.665 81 19.0012
13 1/1/2011 12:00 1 0 0 1 17.22 21.210 77 19.0012
14 1/1/2011 13:00 1 0 0 2 18.86 22.725 72 19.9991
15 1/1/2011 14:00 1 0 0 2 18.86 22.725 72 19.0012
16 1/1/2011 15:00 1 0 0 2 18.04 21.970 77 19.9995
17 1/1/2011 16:00 1 0 0 2 17.22 21.210 77 19.9995
18 1/1/2011 17:00 1 0 0 2 18.04 21.970 82 19.0012
Showing 1 to 18 of 17,379 entries
```

Reading from Excel

```
1 class(Ch1_bike_sharing_data)
```

"data.frame"

```
1 class(Ch1_bike)
```

"tbl_df" "tbl" "data.frame"

These are two different types of data containers. A **data.frame** is native to R and is the simplest most basic type of table structure. A **tbl_df** comes from the **readxl** library, and it is more advanced and has more features. The industry standard has been to use **data.table** and that is the one we will use.

The pandas package ripped-off R, data.table than ripped-off pandas...and the circle of IT continues.

Create a new data.frame from an existing one

```
1 bike = Ch1_bike_sharing_data
```

The tables `bike` and `Ch1_bike_sharing_data` have the same exact data and structure but are completely different and independent from each other.

```
1 str(bike)
```

```
> str(bike)
'data.frame': 17379 obs. of 12 variables:
 $ datetime : chr "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" ...
 $ season   : int 1 1 1 1 1 1 1 1 1 ...
 $ holiday  : int 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int 0 0 0 0 0 0 0 0 0 ...
 $ weather   : int 1 1 1 1 1 2 1 1 1 ...
 $ temp      : num 9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num 14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : int 81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed : num 0 0 0 0 0 ...
 $ casual    : int 3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int 13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int 16 40 32 13 1 1 2 3 8 14 ...
```

Dictionary for Data

- **datetime:** hourly date + timestamp
- **season:** 1=spring, 2=summer, 3=fall, 4=winter
- **holiday:** Whether the day is considered a holiday
- **workingday:** Whether the day is neither a weekend nor holiday
- **weather:**
 - ▶ 1: clear, few clouds, partly cloudy
 - ▶ 2: mist+cloudy, mist+broken clouds, mist+few clouds, mist
 - ▶ 3: light snow, light rain+thunderstorm+scattered clouds, light rain+scattered clouds
 - ▶ 4: heavy rain + ice pellets + thunderstorm + mist, snow + fog
- **temp:** Temperature in celsius
- **atemp:** feels like temperature in celsius
- **humidity:** relative humidity
- **windspeed:** wind speed
- **casual:** number of non-registered user rentals initiated
- **registered:** number of registered user rentals initiated
- **count:** number of total rentals

Convert data.frame to data.table

The table `bike` is a **data.frame**, which is the native basic table structure in R, we want to convert it into a **data.table**, which is a more powerful table structure in R.

We have to first install the `data.table` package.

A screenshot of the RStudio interface. On the left, the code editor shows a script named `chapter1.R` with some R code. Below it, the `Ch1_bike_sharing_data` dataset is displayed as a data frame with 17,379 rows and 12 columns. On the right, the `Environment` pane shows two global variables: `Ch1_bike` and `Ch1_bike_sh...`. Below the environment is the `Data` pane, which lists the same two variables. At the bottom right is the `Packages` tab of the global options. Two arrows point from the text above to this area: one purple arrow points from "We have to first install the `data.table` package." to the `Packages` tab, and a blue arrow points from "which is a more powerful table structure in R." to the `data.table` entry in the list.

| Row | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual |
|-----|----------------|--------|---------|------------|---------|-------|--------|----------|-----------|--------|
| 1 | 1/1/2011 0:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0000 | |
| 2 | 1/1/2011 1:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0000 | |
| 3 | 1/1/2011 2:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0000 | |
| 4 | 1/1/2011 3:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0000 | |
| 5 | 1/1/2011 4:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0000 | |
| 6 | 1/1/2011 5:00 | 1 | 0 | 0 | 2 | 9.84 | 12.880 | 75 | 6.0032 | |
| 7 | 1/1/2011 6:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0000 | |
| 8 | 1/1/2011 7:00 | 1 | 0 | 0 | 1 | 8.20 | 12.880 | 86 | 0.0000 | |
| 9 | 1/1/2011 8:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0000 | |
| 10 | 1/1/2011 9:00 | 1 | 0 | 0 | 1 | 13.12 | 17.425 | 76 | 0.0000 | |
| 11 | 1/1/2011 10:00 | 1 | 0 | 0 | 1 | 15.58 | 19.695 | 76 | 16.9979 | |
| 12 | 1/1/2011 11:00 | 1 | 0 | 0 | 1 | 14.76 | 16.665 | 81 | 19.0012 | |
| 13 | 1/1/2011 12:00 | 1 | 0 | 0 | 1 | 17.22 | 21.210 | 77 | 19.0012 | |
| 14 | 1/1/2011 13:00 | 1 | 0 | 0 | 2 | 18.86 | 22.725 | 72 | 19.9995 | |
| 15 | 1/1/2011 14:00 | 1 | 0 | 0 | 2 | 18.86 | 22.725 | 72 | 19.0012 | |
| 16 | 1/1/2011 15:00 | 1 | 0 | 0 | 2 | 18.04 | 21.970 | 77 | 19.9995 | |
| 17 | 1/1/2011 16:00 | 1 | 0 | 0 | 2 | 17.22 | 21.210 | 77 | 19.9995 | |
| 18 | 1/1/2011 17:00 | 1 | 0 | 0 | 2 | 18.04 | 21.970 | 82 | 19.0012 | |

Showing 1 to 18 of 17,379 entries

Console Terminal ~ /Documents/Investment in R/data/ You currently have tibble v1.4.2.

Environment History Connections Project: (None) Import Dataset Global Environment List

Files Plots Packages Help Viewer Install Update Name Description Version

System Library

- assertthat Easy Pre and Post Assertions 0.2.0
- base64enc Tools for base64 encoding 0.1-3
- BH Boost C++ Header Files 1.66.0-1
- bindr Parametrized Active Bindings 0.1.1
- bindrcpp An 'Rcpp' Interface to Active Bindings 0.2.2
- boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-20
- class Functions for Classification 7.3-14

Install data.table

Type data.table into the Packages space.

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the R script `chapter1.R*` containing code to load and inspect the `Ch1_bike_sharing_data` dataset.
- Environment View:** Shows the global environment with objects `bike`, `Ch1_bike`, and `Ch1_bike_sha...`.
- Console View:** Shows the output of running the script, including the loading of the `data.table` package and its contents.
- Packages View:** Shows the installed packages list.
- Install Packages Dialog:** A modal dialog titled "Install Packages" is open, with "Repository (CRAN)" selected under "Install from". The "Packages" field contains `data.table`. Under "Install to Library", the path `/Library/Frameworks/R.framework/Versions/3.5/Resources/lib` is selected. The "Install dependencies" checkbox is checked. Buttons for "Install" and "Cancel" are at the bottom.

Load the data.table Package

Once you install a new package, you don't have to do it again, but you have to load it into memory when you use it in your code.

```
1 library(data.table)
```

Now we will convert the bike `data.frame` into a `data.table`.

```
1 setDT(bike)
```

If you run the `class` command:

```
1 class(bike)
```

This will be your output:

`"data.table" "data.frame"`

To convert it back into a `data.frame` (basic table), just use the command:

```
1 setDF(bike)
```

Filtering Data

Usual filtering capabilities:

`== > < >= <=`

`& | ! %in%`

`is.na() is.finite()`

`df[Row Section]`

`df[Row Section, Column Section]`

`df[RowSection, ColumnSection, Grouping Section]`

Filter for all the data in winter:

```
1 bike [season==4]
```

Filter for all the data in winter and summer:

```
1 bike [season==4 | season==1]
```

Notice the logic, even though we want bot winter AND summer, from a filtering point of view you can't have a row of data that has both the 4 AND 1 value, but you can filter for all the rows that have values 4 OR filter for rows that have values 1.

Filtering Data

Filter for all the data in winter and summer, using %in%:

```
1 bike[season %in% c(1,4)]  
2 f=c(1,4)  
3 bike[season %in% f]
```

Filter for the rows that are in winter AND the weather was clear (value 1)

```
1 bike[season==4 & weather==1]
```

Filter for the rows that are NOT in winter AND the weather was clear (value 1)

```
1 bike[season != 4 & weather==1]
```

Notice how when you run these lines, the results show up in the console.
BUT if you could also place the results in another table.

```
1 nowinter = bike[season != 4 & weather==1]
```

Selecting Columns

df[Row Section, Column Section]

Select the datetime and windspeed columns:

```
1 bike[, .(datetime, windspeed)]
```

The results come back as a data.table, but if you don't use .(), and have just one column, then it comes back as a vector.

```
1 bike[, windspeed]
```

```
> bike[,windspeed]
 [1] 0.0000 0.0000 0.0000 0.0000 0.0000 6.0032 0.0000 0.0000 0.0000
 [10] 0.0000 16.9979 19.0012 19.0012 19.9995 19.0012 19.9995 19.9995 19.0012
 [19] 16.9979 16.9979 16.9979 12.9980 15.0013 19.9995 19.9995 16.9979 19.0012
 [28] 12.9980 12.9980 19.9995 12.9980 15.0013 15.0013 15.0013 16.9979 19.9995
 [37] 8.9981 12.9980 11.0014 11.0014 12.9980 22.0028 30.0026 23.9994 22.0028
 [46] 19.9995 11.0014 23.9994 27.9993 26.0027 19.0012 26.0027 12.9980 19.0012
 [55] 26.0027 16.9979 22.0028 19.9995 19.0012 19.0012 16.9979 16.9979 15.0013
 [64] 7.0015 0.0000 7.0015 8.9981 8.9981 7.0015 7.0015 7.0015 8.9981
 [73] 6.0032 7.0015 7.0015 8.9981 11.0014 15.0013 22.0028 19.9995 11.0014
 [82] 12.9980 15.0013 16.9979 16.9979 15.0013 12.9980 7.0015 7.0015 0.0000
```

Selecting Columns - Applying Functions

We can apply functions to column data, for example, what is the average temperature:

```
1 bike[, mean(temp)]
```

20.37647

Now what if we want the average, standard deviation, minimum, and maximum temperatures:

```
1 bike[ , .(mean(temp), sd(temp), min(temp), max(temp))]
```

```
> bike[ , .(mean(temp), sd(temp), min(temp), max(temp))]
```

V1 V2 V3 V4

```
1: 20.37647 7.894801 0.82 41
```

We can store the results in another table and rename the columns:

```
1 bb=bike[ , .(mean(temp), sd(temp), min(temp), max(temp))]
```

```
2 names(bb)=c('Mean', 'SD', 'Min', 'Max')
```

Selecting Columns - Applying Functions

```
> bb
```

| | Mean | SD | Min | Max |
|----|----------|----------|------|-----|
| 1: | 20.37647 | 7.894801 | 0.82 | 41 |

What if we want to see the average, standard deviation, min and max, but now according to the seasons. we must now use the 3rd section of the `data.table`, to do aggregations.

`df[RowSection,ColumnSection,Grouping Section]`

The grouping section allows us to apply functions to subsections of the data. For example:

```
1 bike[, .(mean(temp), sd(temp), min(temp), max(temp)), by=season]
```

Selecting Columns - Applying Functions

```
> bike[, .(mean(temp), sd(temp), min(temp), max(temp)), by=season]
```

| season | V1 | V2 | V3 | V4 |
|--------|----------|----------|-------|-------|
| 1: | 12.26501 | 4.867764 | 0.82 | 29.52 |
| 2: | 22.33119 | 5.708559 | 6.56 | 38.54 |
| 3: | 28.96282 | 3.850012 | 15.58 | 41.00 |
| 4: | 17.34866 | 5.008863 | 5.74 | 31.16 |

3 Steps of Aggregation:

1. Create 4 subtables, 1 table for each season
2. Apply the functions to each subtable
3. Combine the results for each of the subtables into 1 table (that's why we will have 4 rows)

Now what if we wanted to find the mean, max, etc. for each season's own weather, i.e. for each particular weather type in each season.

```
1 bike[, .(mean(temp), sd(temp), min(temp), max(temp)), by=.(season, weather)]
```

Selecting Columns - Applying Functions

```
> bike[,.(mean(temp),sd(temp),min(temp),max(temp)),by=.(season,weather)]
```

| | season | weather | V1 | V2 | V3 | V4 |
|-----|--------|---------|-----------|----------|-------|-------|
| 1: | 1 | 1 | 12.274462 | 5.113609 | 0.82 | 29.52 |
| 2: | 1 | 2 | 12.272100 | 4.506817 | 0.82 | 29.52 |
| 3: | 1 | 3 | 12.211111 | 4.138957 | 3.28 | 22.96 |
| 4: | 1 | 4 | 7.653333 | 1.706966 | 5.74 | 9.02 |
| 5: | 2 | 3 | 19.552759 | 5.450124 | 6.56 | 33.62 |
| 6: | 2 | 2 | 21.561556 | 5.324698 | 9.02 | 34.44 |
| 7: | 2 | 1 | 23.033711 | 5.734585 | 7.38 | 38.54 |
| 8: | 3 | 1 | 29.405250 | 3.950161 | 15.58 | 41.00 |
| 9: | 3 | 3 | 26.825279 | 2.909918 | 19.68 | 37.72 |
| 10: | 3 | 2 | 28.037592 | 3.333418 | 18.86 | 39.36 |
| 11: | 4 | 2 | 17.936843 | 4.832713 | 6.56 | 31.16 |
| 12: | 4 | 3 | 18.424853 | 4.383840 | 9.02 | 27.06 |
| 13: | 4 | 1 | 16.912618 | 5.124104 | 5.74 | 30.34 |

Let's put the results in a new table bws, and put the season and weather in order:

Selecting Columns - Applying Functions

```
1 bws = df[,.(mean(oi),sd(oi),min(oi),max(oi)),by=.(  
 2   ticker,Date)]  
2 names(bws)=c('ticker','Date','Mean','SD','Min','Max')  
3 bws=bws[order(ticker,Date)]
```

```
> bws[order(season,weather)]
```

| | season | weather | Mean | SD | Min | Max |
|-----|--------|---------|-----------|----------|-------|-------|
| 1: | 1 | 1 | 12.274462 | 5.113609 | 0.82 | 29.52 |
| 2: | 1 | 2 | 12.272100 | 4.506817 | 0.82 | 29.52 |
| 3: | 1 | 3 | 12.211111 | 4.138957 | 3.28 | 22.96 |
| 4: | 1 | 4 | 7.653333 | 1.706966 | 5.74 | 9.02 |
| 5: | 2 | 1 | 23.033711 | 5.734585 | 7.38 | 38.54 |
| 6: | 2 | 2 | 21.561556 | 5.324698 | 9.02 | 34.44 |
| 7: | 2 | 3 | 19.552759 | 5.450124 | 6.56 | 33.62 |
| 8: | 3 | 1 | 29.405250 | 3.950161 | 15.58 | 41.00 |
| 9: | 3 | 2 | 28.037592 | 3.333418 | 18.86 | 39.36 |
| 10: | 3 | 3 | 26.825279 | 2.909918 | 19.68 | 37.72 |
| 11: | 4 | 1 | 16.912618 | 5.124104 | 5.74 | 30.34 |
| 12: | 4 | 2 | 17.936843 | 4.832713 | 6.56 | 31.16 |
| 13: | 4 | 3 | 18.424853 | 4.383840 | 9.02 | 27.06 |

Creating New Variables

What if we wanted to filter the original bike data, by only the days that had a temperature that was less than the average temperature for its particular season? The simplest way to do this is to simply create a new column, that has the average temperature for each particular season. Again, we will use aggregation and apply grouping by season, but this time we will create new columns within the table with those results:

```
1 bike[, seasonAvgTemp := mean(temp), by = season]
```

Lets view the top 10 rows:

```
1 head(bike, 10)
```

Lets view the last 10 rows:

```
1 tail(bike, 10)
```

Creating New Variables

Now we will simply filter and store the results in a new table named bike2:

```
1 bike2=bike [temp<seasonAvgTemp]
```

If we wanted to create multiple variables, from multiple columns, the syntax changes slightly:

```
1 bike [,c('seasonAvgTemp','seasonMinTemp'):=.(mean(temp),  
min(temp)),by=season]
```

Data.table - Shift Example (dfexp)

| ticker | oi |
|--------|-----------------|
| a | 2 |
| a | 4 |
| a | 6 |
| a | 8 ¹ |
| a | 10 ² |
| b | 5 ³ |
| b | 10 ⁴ |
| b | 15 |
| b | 20 |
| b | 25 |

The `shift()` function is a powerful tool that allows you to do analysis across different time-periods.

```
ticker=c(rep('a',5),rep('b',5))
oi = c(seq(2,10,2),seq(5,25,5))
dfexp=data.table(ticker,oi)

dfexp[, c('oi2', 'oi3', 'oi4'):=
      shift(oi, n = 1:3)]
```

Data.table - Shift Example (dfexp)

| ticker | oi | oi2 | oi3 | oi4 |
|---------------|-----------|------------|------------|------------|
| a | 2 | NA | NA | NA |
| a | 4 | 2 | NA | NA |
| a | 6 | 4 | 2 | NA |
| a | 8 | 6 | 4 | 2 |
| a | 10 | 8 | 6 | 4 |
| b | 5 | 10 | 8 | 6 |
| b | 10 | 5 | 10 | 8 |
| b | 15 | 10 | 5 | 10 |
| b | 20 | 15 | 10 | 5 |
| b | 25 | 20 | 15 | 10 |

Three new columns have now been created. Notice when there isn't enough previous rows, a NA is placed in the column.

What is wrong with the shift results?

```
1 dfexp[, c('oi2', 'oi3', 'oi4') := shift(oi, n = 1:3)]
```

Data.table - Shift Example (dfexp)

| ticker | oi | oi2 | oi3 | oi4 |
|---------------|-----------|------------|------------|------------|
| a | 2 | NA | NA | NA |
| a | 4 | 2 | NA | NA |
| a | 6 | 4 | 2 | NA |
| a | 8 | 6 | 4 | 2 |
| a | 10 | 8 | 6 | 4 |
| b | 5 | NA | NA | NA |
| b | 10 | 5 | NA | NA |
| b | 15 | 10 | 5 | NA |
| b | 20 | 15 | 10 | 5 |
| b | 25 | 20 | 15 | 10 |

Since the shift only happens in sub-groups, b doesn't look at any of a's previous rows.

```
1 dfexp[, c('oi2', 'oi3', 'oi4') := shift(oi, n = 1:3), by = ticker]
```