Data and Variables in R

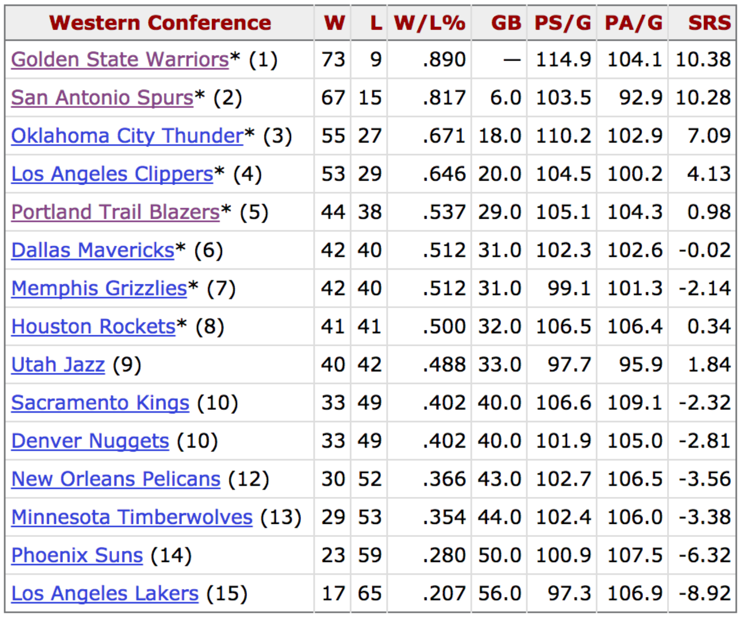
Gaston Sanchez

Learning Objectives:

- Basics of vectors
- Variables (as vectors and factors)
- Quantitative variables as numeric vectors
- Qualitative variables (as factors)
- Manipulating vectors

NBA Data

In this tutorial we'll consider some NBA data from the website *Basketball Reference*. More specifically, let's look at the Western Conference Standings (season 2015-2016) shown in the following screenshot:



Western Conference	W	L	W/L%	GB	PS/G	PA/G	SRS
Golden State Warriors* (1)	73	9	.890	—	114.9	104.1	10.38
San Antonio Spurs* (2)	67	15	.817	6.0	103.5	92.9	10.28
Oklahoma City Thunder* (3)	55	27	.671	18.0	110.2	102.9	7.09
Los Angeles Clippers* (4)	53	29	.646	20.0	104.5	100.2	4.13
Portland Trail Blazers* (5)	44	38	.537	29.0	105.1	104.3	0.98
Dallas Mavericks* (6)	42	40	.512	31.0	102.3	102.6	-0.02
Memphis Grizzlies* (7)	42	40	.512	31.0	99.1	101.3	-2.14
Houston Rockets* (8)	41	41	.500	32.0	106.5	106.4	0.34
Utah Jazz (9)	40	42	.488	33.0	97.7	95.9	1.84
Sacramento Kings (10)	33	49	.402	40.0	106.6	109.1	-2.32
Denver Nuggets (10)	33	49	.402	40.0	101.9	105.0	-2.81
New Orleans Pelicans (12)	30	52	.366	43.0	102.7	106.5	-3.56
Minnesota Timberwolves (13)	29	53	.354	44.0	102.4	106.0	-3.38
Phoenix Suns (14)	23	59	.280	50.0	100.9	107.5	-6.32
Los Angeles Lakers (15)	17	65	.207	56.0	97.3	106.9	-8.92

http://www.basketball-reference.com/leagues/NBA_2016.html#all_confs_standings_E

The above table contains 15 rows with 8 columns. The first column contains the names of the teams in the Western Conference, and the rest of the columns are:

- *W*: wins
- *L*: losses
- *W/L%*: win-loss percentange
- *GB*: games behind (the top team)

- *PS/G*: points per game
- *PA/G*: opponent points per game
- *SRS*: simple rating system

From the statistical standpoint, we say that the table has 8 variables measured (or observed) on 15 individuals. In this case the "individuals" are the basketball teams.

Basics of vectors

In order to use R as the computational tool in this course, one of the first things you need to learn is how to input data. Before describing how to read in tables in R (we'll cover that later), we must talk about **vectors**.

R vectors are the most basic structure to store data in R. Virtually all other data structures in R are based or derived from vectors. Using a vector is also the most basic way to manually input data.

You can create vectors in several ways. The most common option is with the function `c()` (combine). Simply pass a series of values separated by commas. Here is how to create a vector `wins` with the first five values from the column *W* of the conference standings table:

```
wins = c(73, 67, 55, 53, 44)
```

Likewise, we can create a vector `losses` like this:

```
losses = c(9, 15, 27, 29, 38)
```

Having the vectors `wins` and `losses`, we can use them to create another vector `win_loss_perc` for the column *W/L%* (win-loss percentange):

```
win_loss_perc = wins / (wins + losses)
win_loss_perc
```

```
## [1] 0.8902439 0.8170732 0.6707317 0.6463415 0.5365854
```

You can think of vectors as variables. The previous vectors `wins`, `losses`, and `win_loss_perc` are what it's known as **quantitative** variables. This means that each value in those variables (the numbers) reflect a quantity.

Not all variables are quantitative. For instance, the first column of the table does not contain numbers but names. The name of a basketball team is referred to as a **qualitative** variable.

In R you can create a vector of names using a character vector. Again, we use the `c()` function and we pass it names surrounded by either single or double quotes. Here's how to create a vector `teams` with the names of the first five teams in the standings table:

```
teams = c('GSW', 'SAS', 'OCT', 'LAC', 'PTB')
```

The vector `teams` is referred in R to as a **character vector** because it is formed by characters.

Manipulating Vectors: Subsetting

In addition to creating variables, you should also learn how to do some basic manipulation of vectors. The most common type of manipulation is called *subsetting* or *indexing* or *subscripting* which refers to extracting elements of a vector (or another R object). To do so, you use what is known as **bracket notation**. This implies using (square) brackets `[ ]` to get access to the elements of a vector. Inside the brackets you can specify one or more numeric values that correspond to the position(s) of the vector element(s):

```
# first element of 'wins'
wins[1]

# third element of 'losses'
losses[3]

# last element of teams
teams[5]
```

Some common functions that you can use on vectors are:

- `length()` gives the number of values
- `sort()` sorts the values in increasing or decreasing ways
- `rev()` reverses the values

```
length(teams)
teams[length(teams)]
sort(wins, decreasing = TRUE)
rev(wins)
```

Subsetting with Logical Indices

In addition to using numbers inside the brackets, you can also do *logical subsetting*. This type of subsetting involves using a **logical** vector inside the brackets. A logical vector is a particular type of vector that takes the special values `TRUE` and `FALSE`, as well as `NA` (Not Available).

This type of subsetting is very powerful because it allows you to extract elements based on some logical condition. Here are some examples of logical subsetting:

```
# wins of Golden State Warriors
wins[teams == 'GSW']

# teams with wins > 40
teams[wins > 40]

# name of teams with losses between 10 and 29
teams[losses >= 10 & losses <= 29]
```

Factors and Qualitative Variables

As mentioned before, vectors are the most essential type of data structure in R. Related to vectors, there is another important data structure in R called **factor**. Factors are data structures exclusively designed to handle qualitative or categorical data.

The term *factor* as used in R for handling categorical variables, comes from the terminology used in *Analysis of Variance*, commonly referred to as ANOVA. In this statistical method, a categorical variable is commonly referred to as *factor* and its categories are known as *levels*.

To create a factor you use the homonym function `factor()`, which takes a vector as input. The vector can be either numeric, character or logical.

```
# numeric vector
num_vector <- c(1, 2, 3, 1, 2, 3, 2)

# creating a factor from num_vector
first_factor <- factor(num_vector)

first_factor
```

```
## [1] 1 2 3 1 2 3 2
## Levels: 1 2 3
```

You can take the `teams` vector and convert it as a factor:

```
teams = factor(teams)
teams
```

```
## [1] GSW SAS OCT LAC PTB
## Levels: GSW LAC OCT PTB SAS
```

Sequences

It is very common to generate sequences of numbers. For that R provides:

- the colon operator `":"`
- sequence function `seq()`

```
# colon operator
1:5
1:10
-3:7
10:1
```

```
# sequence function
seq(from = 1, to = 10)
seq(from = 1, to = 10, by = 1)
seq(from = 1, to = 10, by = 2)
seq(from = -5, to = 5, by = 1)
```

Repeated Vectors

There is a function `rep()`. It takes a vector as the main input, and then it optionally takes various arguments: `times`, `length.out`, and `each`.

```
rep(1, times = 5)         # repeat 1 five times
```

```
## [1] 1 1 1 1 1
```

```
rep(c(1, 2), times = 3)  # repeat 1 2 three times
```

```
## [1] 1 2 1 2 1 2
```

```
rep(c(1, 2), each = 2)
```

```
## [1] 1 1 2 2
```

```
rep(c(1, 2), length.out = 5)
```

```
## [1] 1 2 1 2 1
```

Here are some more complex examples:

```
rep(c(3, 2, 1), times = 3, each = 2)
```

From vectors to data frames

Now that we've seen how to create some vectors and do some basic manipulation, we can describe how to combine them in a table in R. The standard tabular structure in R is a **data frame**. To manually create a data frame you use the function `data.frame()` and you pass it one or more vectors. Here's how to create a small data frame `dat` with the vectors `teams`, `wins`, `losses`, and `win_loss_perc`:

```
dat = data.frame(
  Teams = teams,
  Wins = wins,
  Losses = losses,
  WLperc = win_loss_perc
)

dat
```

```
##   Teams Wins Losses    WLperc
## 1   GSW   73      9 0.8902439
## 2   SAS   67     15 0.8170732
## 3   OCT   55     27 0.6707317
## 4   LAC   53     29 0.6463415
## 5   PTB   44     38 0.5365854
```

Manipulating data frames is more complex than manipulating vectors. However, manipulating the column of a data frame is essentially the same as manipulating a vector.

There are a couple of ways to "select" a column of a data frame. One option consists of using the dollar `$` operator. This involves typing the name of the data frame, followed by the `$`, followed by the name of the column. For instance, to extract the values in column `Teams` simply type:

```
dat$Teams
```

```
## [1] GSW SAS OCT LAC PTB
## Levels: GSW LAC OCT PTB SAS
```

Moreover, you can use bracket notation on the extracted column like with any type of vector:

```
dat$Wins[1]
```

```
## [1] 73
```

```
dat$Wins[5]
```

```
## [1] 44
```

Likewise, you can do logical subsetting:

```
# wins of Golden State Warriors
dat$Wins[dat$Teams == 'GSW']

# teams with wins > 40
dat$Teams[dat$Wins > 40]

# name of teams with losses between 10 and 29
dat$Teams[dat$Losses >= 10 & dat$Losses <= 29]
```

Your Turn

Refer to the table of Western Conference Standings shown at the beginning of this document. Your mission consists of creating a data frame `standings`. In order to create such data frame, you will have to first create the following eight vectors:

- `teams`
- `wins`
- `losses`
- `win_loss_perc`

- `games_behind`
- `points_scored`
- `points_against`
- `rating`

You can create the vector `games_behind` by taking the won games of Golden State Warriors and subtracting the wins of the rest of the teams, that is:

```
wins[1] - wins
```

Once you have the previous listed vectors, use the function `data.frame()` to build `standings`.

Select the *Points Scored* from the table `standings` and sort it both in increasing as well as decreasing order.