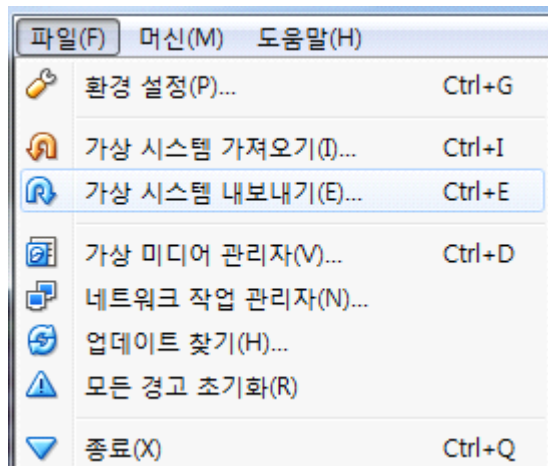
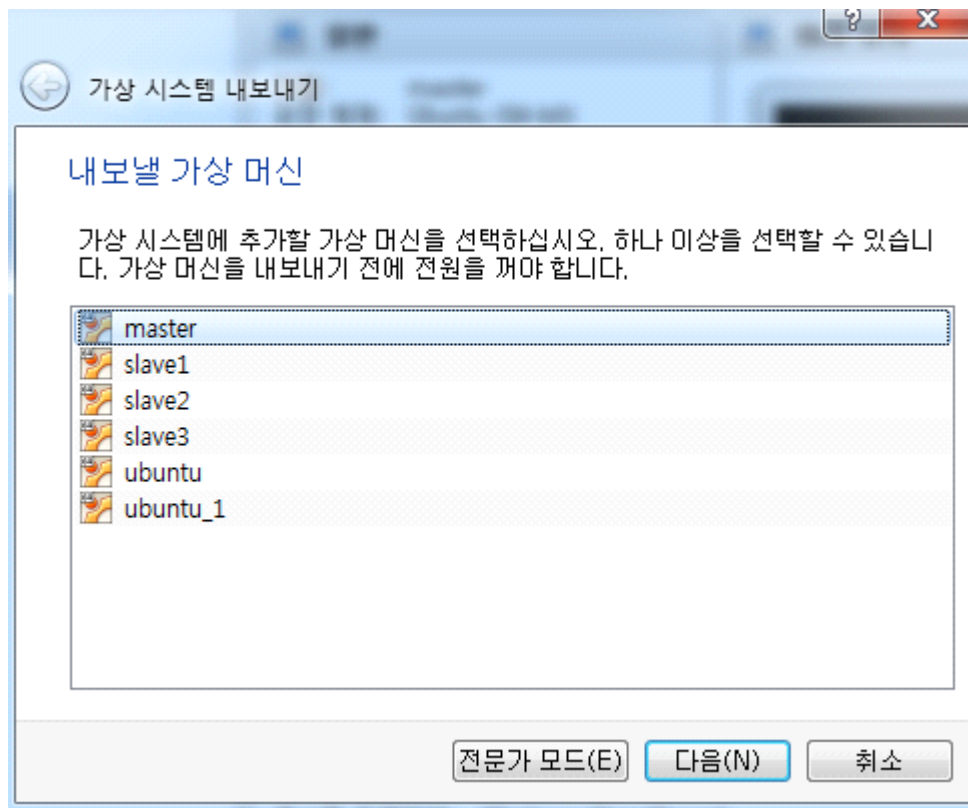


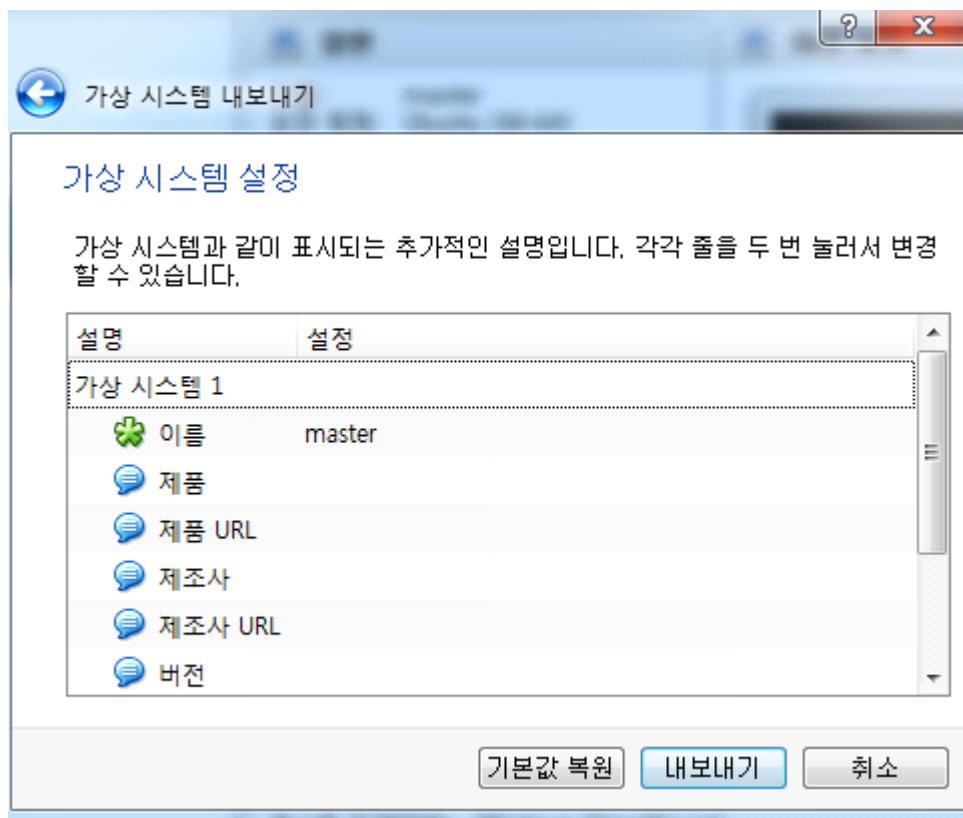
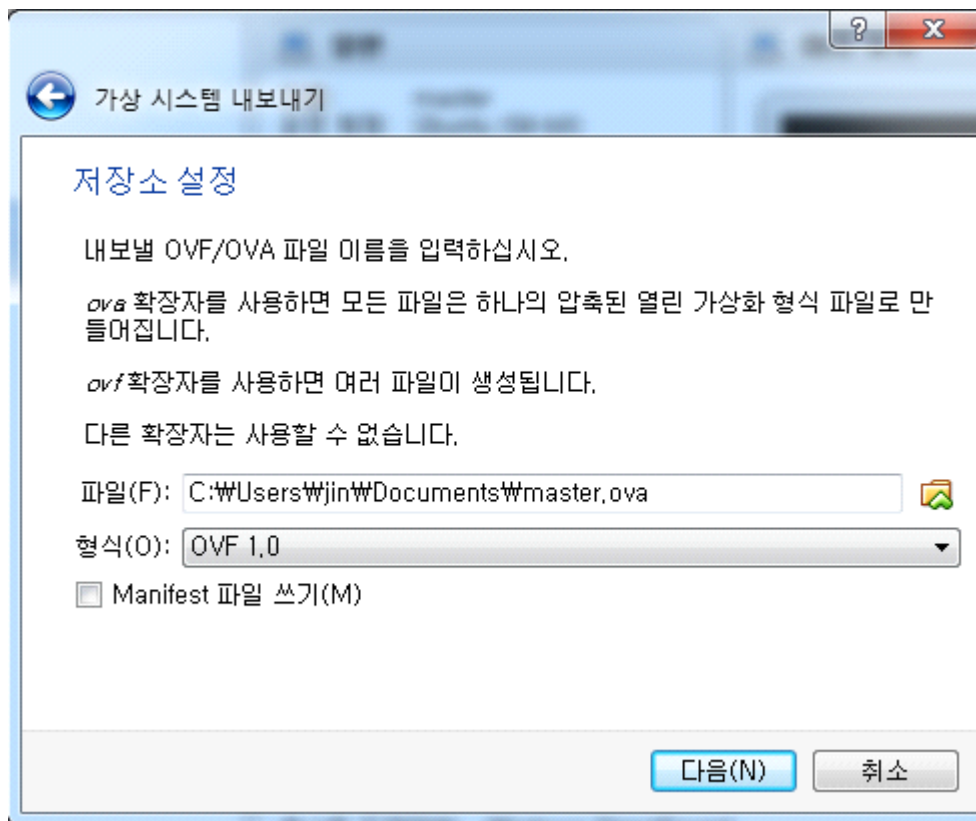
virtualBox 문제로 인한 vmware 변경

2018년 2월 27일 화요일 오후 3:34



내보내기 선택





단일 노드로 변경

2018년 2월 27일 화요일 오후 3:36

3. core-site

/usr/local/hadoop/etc/hadoop/core-site.xml:

```
jinuser@jin-VirtualBox:~$ sudo mkdir -p /app/hadoop/tmp
jinuser@jin-VirtualBox:~$ sudo chown jinuser:hadoop /app/hadoop/tmp/
```

```
jin@jin-VirtualBox:~$ sudo gedit
[sudo] password for jin:
```

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>|
    <value>/app/hadoop/tmp</value>
  </property>

  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:62350</value>
  </property>
</configuration>
```

sudo vim /usr/local/hadoop/etc/hadoop/core-site.xml

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:62350</value>
</property>
```

Master -> localhost 변경

4. mapred-site

```
<configuration>
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:62351</value>
</property>
</configuration>
```

sudo vim /usr/local/hadoop/etc/hadoop/mapred-site.xml

```
<property>
<name>mapred.job.tracker</name>
<value>localhost:62351</value>
</property>
```

master -> localhost 로 변경

5. hdfs-site

/usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
jinsuser@jin-VirtualBox:~/hadoop-2.8.1$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
[sudo] password for jinsuser:
jinsuser@jin-VirtualBox:~/hadoop-2.8.1$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
jinsuser@jin-VirtualBox:~/hadoop-2.8.1$ sudo chown -R jinsuser:hadoop /usr/local/hadoop_store
```

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

```
sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
sudo chown -R manager:hadoop /usr/local/hadoop_store
```

```
sudo vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
<property>
```

```
<name>dfs.http.address</name>
<value>localhost:50070</value>
</property>
<property>
  <name>dfs.secondary.http.address</name>
  <value>localhost:50090</value>
</property>
```

Value 변경 및 Datanode 추가

Master와 slave1을 localhost로 변경

Slaves 파일 내용 변경

2018년 2월 27일 화요일 오후 3:48

```
manager@jin-VirtualBox:~$ sudo vim /usr/local/hadoop/etc/hadoop/slaves
```

```
localhost
```

Slave들을 전부 지우고 localhost로 변경

하둡 실행

2018년 2월 27일 화요일 오후 3:50

```
hadoop namenode -format  
start-all.sh
```

```
manager@jin-VirtualBox:~$ jps  
2321 ResourceManager  
1829 NameNode  
2165 SecondaryNameNode  
2438 NodeManager  
2489 Jps  
1978 DataNode
```

R 설치

2017년 10월 5일 목요일 오후 7:12

```
sudo echo "deb http://cran.rstudio.com/bin/linux/ubuntu/artful/" | sudo tee -a /etc/apt/sources.list
sudo echo "deb http://cran.rstudio.com/bin/linux/ubuntu/zesty/" | sudo tee -a /etc/apt/sources.list
sudo echo "deb http://cran.rstudio.com/bin/linux/ubuntu/xenial/" | sudo tee -a /etc/apt/sources.list
sudo echo "deb http://cran.rstudio.com/bin/linux/ubuntu/trusty/" | sudo tee -a /etc/apt/sources.list
```

```
gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
gpg -a --export E084DAB9 | sudo apt-key add -
```








```
sudo apt-get update
sudo apt-get install r-base r-base-dev -y
```

```
sudo apt-get install gdebi-core
wget https://download1.rstudio.org/rstudio-0.99.896-amd64.deb
sudo gdebi -n rstudio-0.99.896-amd64.deb
```

[참고]The Comprehensive R Archive Network

2018년 2월 26일 월요일 오후 3:34

Index of /bin/linux/ubuntu

| | Name | Last modified | Size | Description |
|---|----------------------------------|-------------------------------|----------------------|-----------------------------|
|  | Parent Directory | | - | |
|  | README | 23-Dec-2017 18:39 | 6.2K | |
|  | README.html | 23-Dec-2017 18:39 | 726K | |
|  | artful/ | 26-Feb-2018 03:07 | - | |
|  | trusty/ | 26-Feb-2018 03:05 | - | |
|  | xenial/ | 26-Feb-2018 03:06 | - | |
|  | zesty/ | 26-Feb-2018 03:07 | - | |

UBUNTU PACKAGES FOR R

- [Installation](#)
- [Supported Packages](#)
- [Secure APT](#)
- [Administration and Maintanances of R Packages](#)
- [Reporting Problems](#)
- [Acknowledgements](#)

R packages for Ubuntu on i386 and amd64 are available for all stable Desktop releases of Ubuntu until their official end of life date. However, only the latest Long Term Support (LTS) release is fully supported. As of May 3, 2017 the supported releases are Artful Aardvark (17.10), Zesty Zapus (17.04), Xenial Xerus (16.04; LTS), and Trusty Tahr (14.04; LTS).

See <https://wiki.ubuntu.com/Releases> for details.

For additional binary packages for R (currently 3,600+), check out the CRAN2deb4ubuntu PPA:

<https://launchpad.net/~marutter/+archive/ubuntu/c2d4u>

Installation

To obtain the latest R packages, add an entry like

deb <https://<my.favorite.cran.mirror>/bin/linux/ubuntu artful/>

or

deb <https://<my.favorite.cran.mirror>/bin/linux/ubuntu zesty/>

or

deb <https://<my.favorite.cran.mirror>/bin/linux/ubuntu xenial/>

or

deb <https://<my.favorite.cran.mirror>/bin/linux/ubuntu trusty/>

in your `/etc/apt/sources.list` file, replacing `<my.favorite.cran.mirror>` by the actual URL of your favorite CRAN mirror. See <https://cran.r-project.org/mirrors.html> for the list of CRAN mirrors. To install the complete R system, use

```
sudo apt-get update
```

```
sudo apt-get install r-base
```

Users who need to compile R packages from source [e.g. package maintainers, or anyone installing packages with `install.packages()`] should also install the `r-base-dev` package:

```
sudo apt-get install r-base-dev
```

The R packages for Ubuntu otherwise behave like the Debian ones. One may find additional information in the Debian README file located at <https://cran.R-project.org/bin/linux/debian/>.

Installation and compilation of R or some of its packages may require Ubuntu packages from the “backports” repositories. Therefore, it is suggested to activate the backports repositories with an entry like

```
deb https://<my.favorite.ubuntu.mirror>/ trusty-backports main restricted universe
```

in your `/etc/apt/sources.list` file. See <https://launchpad.net/ubuntu/+archivemirrors> for the list of Ubuntu mirrors.

Supported Packages

A number of R packages are available from the Ubuntu repositories with names starting with `r-cran-`. The following ones are kept up-to-date on CRAN: all packages part of the `r-recommended` bundle, namely

- `r-cran-boot`
- `r-cran-class`
- `r-cran-cluster`
- `r-cran-codetools`
- `r-cran-foreign`
- `r-cran-kernsmooth`
- `r-cran-lattice`
- `r-cran-mass`
- `r-cran-matrix`
- `r-cran-mgcv`
- `r-cran-nlme`
- `r-cran-nnet`
- `r-cran-rpart`
- `r-cran-spatial`
- `r-cran-survival`

as well as

- `r-cran-rodbc`

The other `r-cran-*` packages are updated with Ubuntu releases only. Users who need to update one of these R packages (say `r-cran-foo`) should first make sure to obtain all the required build dependencies with

```
sudo apt-get build-dep r-cran-foo
```

Because they rely on the installed version of R, we also provide, on an experimental basis, versions of the following packages as up-to-date as the Ubuntu release allows:

- littler
- python-rpy
- python-rpy-doc
- python-rpy2

Please notice that the maintainers are not necessarily themselves users of these packages, so positive or negative feedback through the usual channels (see below) would be appreciated. Finally, as an added convenience to Ubuntu users who interact with R through Emacs, we also provide an up-to-date version of the package

- ess

Secure APT

The Ubuntu archives on CRAN are signed with the key of "Michael Rutter marutter@gmail.com" with key ID E084DAB9. To add the key to your system with one command use (thanks to Brett Presnell for the tip):

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9
```

An alternate method can be used by retrieving the key with

```
gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
```

and then feed it to apt-key with

```
gpg -a --export E084DAB9 | sudo apt-key add -
```

Some people have reported difficulties using this approach. The issue is usually related to a firewall blocking port 11371. If the first gpg command fails, you may want to try (thanks to Mischan Toosarani for the tip):

```
gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys E084DAB9
```

and then feed it to apt-key with

```
gpg -a --export E084DAB9 | sudo apt-key add -
```

Another alternative approach is to search for the key at <http://keyserver.ubuntu.com:11371/> and copy the key to a plain text file, say key.txt. Then, feed the key to apt-key with

```
sudo apt-key add key.txt
```

Administration and Maintanances of R Packages

The R packages part of the Ubuntu r-base and r-recommended packages are installed into the directory /usr/lib/R/library. These can be updated using apt-get with

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

The other r-cran-* packages shipped with Ubuntu are installed into the directory /usr/lib/R/site-library.

Installing R packages not provided with Ubuntu first requires tools to compile the packages from source. These tools are installed via the R development package with

```
sudo apt-get install r-base-dev
```

Then a site administrator can install R packages into the directory /usr/local/lib/R/site-library by

running R as root and using the

```
> install.packages()
```

function. A routine update can then be undertaken from R using

```
> update.packages(lib.loc = "/usr/local/lib/R/site-library")
```

The paths above are stored in the `R_LIBS_SITE` environment variable defined in the `/etc/R/Renviron` file.

Individual users can install R packages into their home directory. The simplest procedure is to create a file `~/.Renviron` containing, e.g.,

```
R_LIBS_USER=~/.lib/R/library"
```

The `install.packages()` and `update.packages()` functions will then work in directory `~/lib/R/library`. It is also possible to automatically create version-specific library trees; see `?libPaths` in R for more information.

Reporting Problems

The best place to report problems with these packages or ask R questions specific to Ubuntu is the R-SIG-Debian mailing list. See

<https://stat.ethz.ch/mailman/listinfo/r-sig-debian>

for more information.

Acknowledgements

The Debian R packages are maintained by Dirk Eddelbuettel. The Ubuntu packages are compiled for i386 and amd64 by Michael Rutter (marutter@gmail.com) using scripts developed by Vincent Goulet.

<<https://cran.rstudio.com/>>에서 삽입

환경설정

2017년 10월 6일 금요일 오전 8:53

```
sudo vim /etc/profile
```

```
export HADOOP_CMD=/usr/local/hadoop/bin/hadoop
export HADOOP_STREAMING=/usr/local/hadoop/share/hadoop/tools/lib/hadoop-
streaming-2.8.1.jar
export LD_LIBRARY_PATH=/usr/local/lib:/usr/lib/jvm/java-8-oracle/jre/lib/amd64/server
```

```
source /etc/profile
```

다운로드

2017년 10월 6일 금요일 오전 8:55

<https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads>

wget https://github.com/RevolutionAnalytics/plyrmr/releases/download/0.6.0/plyrmr_0.6.0.tar.gz

wget https://github.com/RevolutionAnalytics/rmr2/releases/download/3.3.1/rmr2_3.3.1.tar.gz

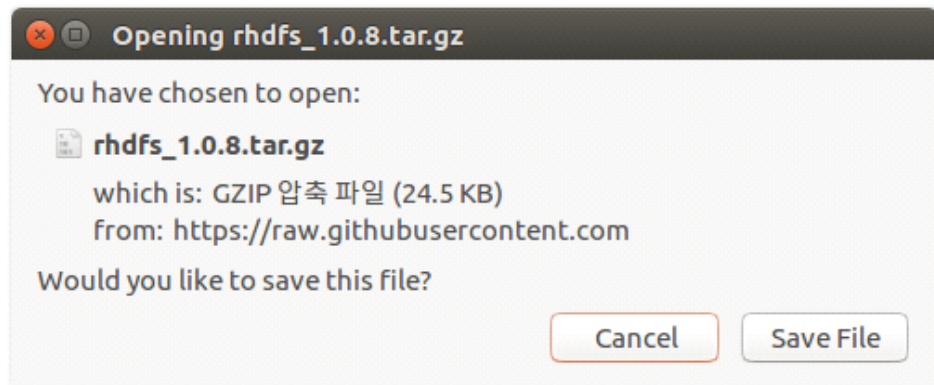
Downloads

Antonio Piccolboni edited this page on 14 Feb 2015 · 89 revisions

Download The Latest Official RHadoop Releases

- [ravro-1.0.3](#)
- [plyrmr-0.6.0](#)
- [rmr-3.3.1](#)
- [rmr-3.3.1 for Windows](#)
- [rhdfs-1.0.8](#)
- [rhdfs-1.0.8 for Windows](#)
- [rhbase-1.2.1](#)

Rhdfs-1.0.8은 직접 다운 받기



RHadoop 설치

2017년 10월 6일 금요일 오전 8:58

```
sudo R CMD javareconf
```

```
sudo R
```

```
install.packages(c("Rcpp","RJSONIO","digest","functional","reshape2","stringr","plyr","caTools"))
```

```
> install.packages(c("Rcpp","RJSONIO","digest","functional","reshape2","stringr",
,"plyr","caTools"))
'/usr/local/lib/R/site-library'의 위치에 패키지(들)을 설치합니다.
(왜냐하면 'lib'가 지정되지 않았기 때문입니다)
--- 현재 세션에서 사용할 CRAN 미러를 선택해 주세요 ---
Secure CRAN mirrors

 1: 0-Cloud [https]                2: Algeria [https]
 3: Australia (Canberra) [https]   4: Australia (Melbourne 1) [https]
 5: Australia (Melbourne 2) [https] 6: Australia (Perth) [https]
 7: Austria [https]                8: Belgium (Ghent) [https]
 9: Brazil (PR) [https]            10: Brazil (RJ) [https]
11: Brazil (SP 1) [https]          12: Brazil (SP 2) [https]
13: Bulgaria [https]              14: Chile 1 [https]
15: Chile 2 [https]               16: China (Guangzhou) [https]
17: China (Lanzhou) [https]        18: Colombia (Cali) [https]
19: Czech Republic [https]        20: Denmark [https]
21: Ecuador (Cuenca) [https]       22: Estonia [https]
23: France (Lyon 1) [https]        24: France (Lyon 2) [https]
25: France (Marseille) [https]    26: France (Montpellier) [https]
27: France (Paris 2) [https]       28: Germany (Göttingen) [https]
29: Germany (Münster) [https]      30: Greece [https]
31: Iceland [https]               32: Indonesia (Jakarta) [https]
33: Ireland [https]               34: Italy (Padua) [https]
35: Japan (Tokyo) [https]          36: Malaysia [https]
37: Mexico (Mexico City) [https]   38: Norway [https]
39: Philippines [https]            40: Serbia [https]
41: Spain (A Coruña) [https]       42: Spain (Madrid) [https]
43: Sweden [https]                44: Switzerland [https]
45: Turkey (Denizli) [https]       46: Turkey (Mersin) [https]
47: UK (Bristol) [https]           48: UK (Cambridge) [https]
49: UK (London 1) [https]          50: USA (CA 1) [https]
51: USA (IA) [https]               52: USA (KS) [https]
53: USA (MI 1) [https]             54: USA (OR) [https]
55: USA (TN) [https]               56: USA (TX 1) [https]
57: Vietnam [https]               58: (other mirrors)
```

선택 : 35



위의 둘 중 하나 나타남.

`install.packages("caTools")` 설치가 안되서 별도 설치

`install.packages("rmr2_3.3.1.tar.gz")`

`install.packages(c("dplyr", "R.methodsS3", "Hmisc", "memoise", "lazyeval", "rjson"))`

`install.packages("plyrmr_0.6.0.tar.gz")`

`install.packages("rJava")`

`Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")`

`Sys.setenv(HADOOP_STREAMING="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.1.jar")`

`library(rJava)`

`install.packages("/home/jin/다운로드/rhdfs_1.0.8.tar.gz")`

Wget으로 받아 했었으나 안되서 직접 다운로드 후 설치

테스트

2017년 10월 6일 금요일 오전 9:47

```
start-all.sh
```

```
R
```

```
Sys.setenv(HADOOP_HOME="/usr/local/hadoop")
Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")
Sys.setenv(HADOOP_LIB_NATIVE_DIR="/usr/local/hadoop/lib/native")
Sys.setenv(HADOOP_STREAMING="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar")
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-8-oracle/")
```

```
library(rJava)
library(rhdfs)
hdfs.init()
library(rmr2)
```

```
numarray <- to.dfs(1:10)
from.dfs(numarray)
result <- mapreduce(input = numarray,
  map = function(k,v) cbind(v,v^2)
)
out <- from.dfs(result)
out
```

110페이지 교재 분석

2017년 10월 23일 월요일 오전 10:46

기본

2017년 10월 23일 월요일 오전 10:46

```
library('rhdfs')
library('rmr2')

hdfs.init()

intsArr=to.dfs(1:10)

result<-mapreduce(
  input = intsArr,
  map=function(k, v){
    keyval(k, v)
  }
)
result

outputData<-from.dfs(result)
outputData
```

lapply

2017년 10월 23일 월요일 오전 10:47

데이터셋을 나누어서 각 조각들을 만들고, 각각의 조각에 특정 함수를 적용하고, 결과를 합쳐서 제공한다는 것이다.

```
> x<-c(1:5)
> lapply(x, function(y){y+1})
[[1]]
[1] 2

[[2]]
[1] 3

[[3]]
[1] 4

[[4]]
[1] 5

[[5]]
[1] 6
```

리스트 접근방법

리스트 변수[[인덱스] # 값을 접근함

리스트 변수[인덱스] # (키,값) 의 형태로 서브 리스트를 반환 한다.

```
> x<-c(1:5)
> result <- lapply(x, function(y){y+1})
> result
[[1]]
[1] 2

[[2]]
[1] 3

[[3]]
[1] 4

[[4]]
[1] 5

[[5]]
[1] 6

> class(result)
[1] "list"
> result[[1]]
[1] 2
> result[2][1]
```

```
[[1]]  
[1] 3
```

```
> result[3]  
[[1]]  
[1] 4
```

[참고]R의 lapply, sapply, vapply를 이해하자~

2017년 10월 23일 월요일 오전 10:27

R의 lapply, sapply, vapply를 이해하자~

| [Cloud&BigData/R](#) 2015.09.29 08:29

Posted by 미니~

R에는 lapply, sapply, vapply, apply, tapply 등 다양한 apply 함수들이 존재한다.

데이터 분석에서 "Split-Apply-Combine" 이라는 전략을 구현한 것이 바로 apply이다.

즉, 데이터셋을 나누어서 각 조각들을 만들고, 각각의 조각에 특정 함수를 적용하고, 결과를 합쳐서 제공한다는 것이다.

lapply

먼저 lapply부터 살펴보자.

lapply의 l은 리스트를 나타낸다. 즉, 입력으로 리스트를 받아서 결과로 리스트를 제공한다는 것이다.

lapply를 테스트하기 위해 먼저 리스트를 만들어 보자.

```
> x <- list(a = 1:5, b = rnorm(10))
> x
$a
[1] 1 2 3 4 5

$b
[1] 0.357568636 0.366356087 0.066221634 -0.441780815 2.362190273 -1.001365162 0.005135182
[8] 1.258957514 1.051370310 0.394752365
```

rnorm은 정규 분포를 따르는 수를 만드는 것으로, rnorm(10)은 기본값으로 평균 0, 표준편차 1인 정규분포를 갖는 10개의 수를 만든다.

리스트의 각 항목, 여기에서는 a와 b에 대해 각각의 평균을 구한다고 생각해 보자.

for 문과 같은 반복문을 통해서 각각의 평균을 구할 수도 있겠지만,

앞서 이야기한 "Split-Apply-Combine"을 활용한 lapply로 한번에 평균을 구할 수 있다.


```
> x_list <- lapply(x, mean)
> x_list
$a
[1] 3

$b
[1] 0.4419406

> class(x_list)
[1] "list"
```

`lapply(x, mean)`은 `x` 리스트의 각 요소에 대해 `mean()` 함수를 적용하라는 것이다.
 결과를 확인해보면, `a`와 `b` 각각의 평균을 보여주는 것을 알 수 있다.
`class()` 함수를 결과값을 확인하면, 역시 리스트로 전달되는 것을 알 수 있다.

경우에 따라서는 결과값을 리스트가 아닌 벡터 형태로 사용할 필요가 있다.
 그래서 `as.double(x_list)`로 결과를 벡터로 변경하면 다음과 같다.

이와 같이 리스트로 결과를 리턴하지 않고 벡터로 바로 리턴해 주는 것이 바로 `sapply`이다.

```
> as.double(x_list)
[1] 3.0000000 0.4419406
```

sapply

`sapply`에서 `s`는 `simplify`를 나타낸다. 즉, 결과를 벡터 형태로 단순화해서 리턴한다는 것이다.
 위에서 `as.double(x_list)`를 통해 길이가 2인 벡터를 가져오는 부분을 `sapply`로 한번에 처리하면 다음과 같다.

```
> sapply(x, mean)
      a      b
[1] 3.0000000 0.4419406
```

즉, `sapply`는 `lapply`와 동일하게 리스트를 입력으로 받아서 각각의 항목에 지정된 함수를 적용한다.

다만 유일한 차이점은 결과값을 리스트가 아닌 벡터 형태로 리턴한다는 점이다.

다음 예제를 한번 생각해보자.

`lapply`에 적용하는 함수를 익명함수로 각각의 원래 값에 1을 더하는 것으로 적용해보면 다음과 같다.

```
> lapply(x, function(y){y+1})
$a
[1] 2 3 4 5 6

$b
[1] 1.357568636 1.366356087 1.066221634 0.558219185 3.362190273 -0.001365162 1.005135182
[8] 2.258957514 2.051370310 1.394752365
```

결과를 보면 리스트 각각의 길이가 서로 다르다.

이 경우, 과연 `sapply`를 적용하면 어떻게 될까?

길이가 다르기 때문에 하나의 벡터로 합치는 것이 어렵다. 그래서 다음과 같이 `sapply`도 `lapply`와 동일하게 리스트를 리턴하게 된다.

```
> sapply(x, function(y){y+1})
$a
[1] 2 3 4 5 6

$b
[1] 1.357568636 1.366356087 1.066221634 0.558219185 3.362190273 -0.001365162 1.005135182
[8] 2.258957514 2.051370310 1.394752365
```

vapply

벡터로 결과를 가져오는 것을 기대했는데, 위처럼 리스트가 결과로 나타나면 문제가 되는 경우도 있다.

이런 경우, 차라리 결과가 나오지 않고 에러를 발생하는 것이 필요할 수도 있다.

이럴때 사용하는 것이 바로 `vapply`이다.

`vapply`는 함수 결과값의 벡터 형태를 미리 지정하고, 해당 형태가 아닐 경우, 에러를 발생시킨다.

```
> vapply(x, mean, double(1))
      a      b
3.0000000 0.4419406
> vapply(x, function(y){y+1}, double(5))
Error in vapply(x, function(y) { : 길이가 반드시 5이어야 하지만,
FUN(X[[2]])의 결과는 길이 10 입니다
Error during wrapup:
> vapply(x, function(y){y+1}, double(10))
Error in vapply(x, function(y) { : 길이가 반드시 10이어야 하지만,
FUN(X[[1]])의 결과는 길이 5 입니다
Error during wrapup:
```

결과값이 동일한 `mean` 함수를 적용할 때는 `sapply`와 똑같이 결과가 나왔지만, 결과값이 서로 다른 익명 함수에서는 에러가 발생한 것을 확인할 수 있다.

<<http://blog.acronym.co.kr/582>>에서 삽입

[참고]자료형(integer,character) 변환 및 데이터 구조 (dataFrame, list) 변환 :: GOOD to GREAT

2017년 10월 23일 월요일 오전 10:57

자료형(integer,character) 변환 및 데이터 구조(dataFrame, list) 변환

2015.10.04 01:41

자료형(integer,character) 변환 및 데이터 구조 (dataFrame, list) 변환

자료형

각각의 자료형들간의 변환은 아래의 함수들을 이용 한다.

as.character(x)

as.complex(x)

as.numeric(x) or as.double(x)

as.integer(x)

as.logical(x)

R에서는 데이터 분석을 위한 유용한 데이터 구조를 기본으로 제공해 준다.

데이터 구조(5개): **vector**, **list**, **data frame**, **matrix(array)**, **factor**

모드:

R에서 해당 객체를 메모리에 어떻게 저장할 것인가를 가리키는 것이 "모드"이다
숫자로 저장할 것인가?

문자열 ?

다른 객체로의 포인터로된 리스트들?

함수?

위의 다양한 것들로서 메모리에 저장 될 수 있다.

객체, 예, 모드

숫자, 3.21415, 수치형

숫자벡터, c(2, 3), 수치형

문자열, "More", 문자형

문자열 벡터, c("More","Larry","Curly"), 문자형

요인, factors(c("NY","CA","IL")),수치형

리스트, list("More","Larry","Curly"), 리스트

데이터 프레임, data.frame(x=1:3, y=c("NY","CA","IL")), 리스트

함수, print, 함수

알아내는 방법

mode()

Class: 추상적인 자료형

R에서는 모든 객체는 추상 자료형인 '클래스'도 가지고 있다.

거리, 시간, 무게 모두 어떤 숫자이므로 모드는 숫자(numeric)이다.

하지만 각각의 해석 방법이 다르므로 클래스는 상이할 수 있다.

결국 클래스가 R에서 직접적인 데이터 처리 방식을 결정하게 된다.

예를들면 print()라는 함수는 입력된 데이터의 class에 따라서 그 처리를 다르게 하는 것이다.

위에서 설명한 모드는 메모리에 저장되는 형태지, 논리적으로 프로그래밍 할 때 쓰는 단위는 아니다.

Factor (범주형 자료)

생성 방법

factor(

x, #팩터로 표현하고자 하는 값(주로 문자열 벡터로 지정)

levels, #값의 레벨

ordered #TRUE면 순서형, FALSE면 명목형 데이터를 뜻한다. 기본값은 FALSE다.

)

팩터에서 레벨의 개수를 반환함.

```
nlevels(  
x # 팩터 값  
)
```

중간에 값 넣기

```
factor(append(as.character(proximity),"TRUE"))
```

factor level rename

http://www.cookbook-r.com/Manipulating_data/Renaming_levels_of_a_factor/

Vectors

벡터는 동질적이다.

같은 자료형을 가진다.

같은 모드를 가지고 있어야 한다.

스칼라와 벡터의 관계는 통상 엄밀히 구분하지만 R에서는 딱히 구분하지 않는다.

하나의 값으로 구성된 벡터가 스칼라고

여러개 있으면 벡터라고 생각하면 된다.

List

이질적이다.

다양한 자료형을 포함 할수 있다.

생성방법

```
list <- list(x,y,z) #이질적인 데이터도 가능함.
```

```
list <- list(mid=0.5, right=0.841, far=0.9)
```

접근방법

리스트 변수[[인덱스] # 값을 접근함

리스트 변수[인덱스] # (키,값) 의 형태로 서브 리스트를 반환 한다.

제거방법

```
x[-2] # without 2nd element
```

```
x[-c(2,3)] # without 2nd and 3rd
```

```
x["arrival_time"] <- NULL
```

빈리스트를 생성하고 인덱스를 이름으로 지정하는 방법

```
vecNames <- list.files("./ESLABCollector/")  
filenames <- sapply(vecNames,list)
```

Foreach를 사용할 때 list 인덱스의 이름을 유지하는 방법

```
jsonDfList <- foreach (x = rawDataList, .final = function(x) setNames(x, names(filenames))) %  
do% {  
  jsonTodf(x)  
}
```

.final 을 이용한다.

결국 setNames란 아래의 것을 축약한 기능이다.

```
setNames( 1:3, c("foo", "bar", "baz") )
```

this is just a short form of

```
tmp <- 1:3
```

```
names(tmp) <- c("foo", "bar", "baz")
```

```
tmp
```

List 중간에 값을 삽입 하기

```
append(testList, list(x=42), 3)
```

```
$`1`
```

```
[1] 1
```

```
$`2`
```

```
[1] 2
```

```
$`3`
```

```
[1] 3
```

```
$x
```

```
[1] 42
```

```
$`4`
```

```
[1] 4
```

Matrix

벡터에 차원을 정해주면 바로 Matrix로 변경 된다.

벡터의 dim 속성이 처음엔 NULL 이지만 이것을 주게 되면 matrix가 된다.

```
dim(2,3) # 2 by 3 행렬
```

list도 dim 변경을해서 행렬화 할 수 있다.

하지만 list의 특성상 이질적인 행렬을 만들어 낼 수 있으므로 그러한 것들은 조심해야 한다.

Array (배열)

3차원 matrix 부터는 배열이라는 이름을 쓴다.

Factor(요인)

벡터처럼 생겼지만 특별한 속성이 있다.

R은 벡터에 있는 고유한 값(unique value)의 정보를 얻어 내는데, 이 고유 값들을 요인의 '수준(level)'이라고 일컫는다.

요인들은 간단하고 효율적인 형태로 데이터 프레임에 저장된다.

다른 프로그래밍 언어에서는 요인을 '열거형 값(enumerated value)들로 이루어진 벡터로 표현한다.

요인의 사용처는 아래와 같다.

범주형 변수>

요인 하나는 범주형 변수 하나를 나타낼 수 있다.

범주형 변수들은 분할표, 선형 회귀, 변량 분석, 로지스틱 회귀 분석 등 많은 분야에서 사용 된다.

집단 분류

이것은 데이터 항목에다가 집단에 따른 라벨을 붙이거나 태깅을 할 때 쓰는 방법이다.

데이터 프레임 (Data Frame)

가장 중요한 데이터 타입이다.

표 형태의 데이터를 표현하기 위함이다.

특징

행은 모두 같은 길이이다. 열은 하나의 엘리먼트를 나타낸다.

행렬과 다른점은 각각의 행들이 서로다른 데이터 타입들을 가질 수 있다는 것이다.

즉, 각각의 열들은 같은 타입으로 이뤄지지만, 모든 열이 모두 같을 필요는 없다.

데이터 프레임의 속성은 row.names()를 통해서 알 수 있다.

통상 read.table() 또는 read.csv()를 통해서 생성 한다.

행렬을 data frame으로 변경 할 수 있다. data.matrix() 호출을 통해서

▣ 데이터 프레임 생성하기

data.frame()을 사용

```
g <- c("A","B")
x <- 1:3
dat <- data.frame(g,x)
```

▣ 리스트를 프레임 데이터로 변경

frame.data(list)

▣ 프레임의 구조를 알려준다.

str(data)

```
> str(ac)
'data.frame': 356 obs. of 2 variables:
 $ Time : num 0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 ...
 $ Power: num 14.8 14.1 19.9 14.6 14.1 ...
```

▣ 프레임에서 열들만 따로 출력하는 방법이다.

data\$time

▣ 데이터 프레임의 Index를 알아내는 방법

```
> a <- data.frame(1:6,11:16)
```

```
> a
```

```
  X1.6 X11.16
1    1    11
2    2    12
3    3    13
4    4    14
5    5    15
6    6    16
```

```
> grep(15,a[,2])
```

```
[1] 5
```

▣ 해당 컬럼 이름이 몇번째 인지를 알아내는 방법

```
> grep("X11.16",colnames(a))
```

```
[1] 2
```


▣ 프레임에서 행과 열만 따로 출력하는 방법이다.

```
data$time[0]
```

▣ 위치로 데이터 프레임의 열 선택하기

스타일에 따라 **열** 또는 **데이터프레임** 두가지 자료형중 하나로 반환 된다.

데이터 프레임 스타일의 선택 방법

```
dfm[[n]] # 하나의 열을 반환 한다.
```

```
dfm[n] # n번째 열 단독으로만 구성된 '데이터 프레임'을 반환 한다.
```

```
dfm[c(n1,n2,...,nk)] # n1,n2,...,nk 위치에 있는 열들로 만든 "데이터 프레임"을 반환한다.
```

행렬 방식의 첨자 지정을 이용해서 선택 하는 방법

```
dfm[,n] # n번째 열을 반환 한다
```

```
dfm[,c(n1,n2,...,nk)] # n1,n2,...,nk 위치에 있는 열들로 만든 "데이터 프레임"을 반환 한다.
```

▣ 프레임 열의 이름을 변경하는 방법이다.

```
names(data) <- c("name1", "name2", "name3")
```

▣ 특정 열을 **찾아서** 이름을 변경하는 방법

```
> names(ac)[names(ac)=="Time"] <- c("newTime")
```

```
> names(ac)
```

```
[1] "newTime" "Power"
```

```
> names(ac)[names(ac)=="newTime"] <- c("Time")
```

```
> names(ac)
```

```
[1] "Time" "Power"
```

▣ 데이터 프레임에 열 추가하기

새로운 열의 이름을 설정하고 "NA"로 채운다.

```
data$newcol <- NA
```

새로운 열의 이름을 설정하고 벡터를 삽입한다.

```
data$newcol <- vec
```

▣ 데이터 프레임에 열을 제거하기

해당열에 NULL을 대입한다.

```
data$badcol <- NULL
subset(data, select = -badcol)
subset(data, select = c(-badcol, -othercol))
```

▣ 데이터 프레임 열 순서 변경하기

#숫자로 해당열을 선택해서 변경한다.

```
dat <- dat[c(1,3,2)]
```

#이름으로 선택해서 변경 한다.

```
data <- dat[c("col1","col3","col2")]
```

▣ 데이터 프레임의 부분 집합 취하기

`subset()` 함수를 이용해서 부분 집합을 취할 수 있다.

일련의 조건을 기입하고 그 조건에 맞는 행들을 추출 할 수 있다.

```
subset(climate, Source == "Berkeley", select = c(Year, Anomaly10y))
```

데이터 셋 climate에서

Source 열이 "Berkeley"인 것에 대해서

Year 와 Anomaly10y의 열에 대해서만 그 값을 추출

논리 연산자를 조합하면 일정 범위 내의 데이터를 추출 할 수도 있다.

1900년에서 2000년 사이의 source가 "Berkeley"인 행들만 선택 한다.

```
subset(climate, Source == "Berkeley" & Year >= 1900 & Year <= 2000, select = c(Year, Anomaly10y))
```

▣ 여러가지 방법을 이용한 데이터 프레임 합치기

데이터 프레임 합치기

`cbind` # 두개의 데이터를 열로 쌓는다.

`rbind` # 두개의 데이터를 행으로 쌓는다.

연속적인 리스트라면,

`do.call(rbind,mylist)`를 하면된다.

주의: 위방법들은 모두 행열이 같은 크기일때만 가능함

크기가 서로 다르면 반복해서 아랫부분을 채워준다고 하지만 이상하게 동작하지 않는다.

서로다른 데이터 프레임 합치기

merge # SQL에서 join에 해당하는 명령어이다

default로 동작은 그냥 단순히 합치는 것이다.

일치하지 않는것이 있다면 데이터의양이 NxM의 양으로 늘어 난다.

일치하는 부분만 합치고 싶으면,

merge(A,B, by="열 이름")

으로 설정한다.

merge의 다른 option들

```
## S3 method for class 'data.frame'
merge (x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x", ".y"),
      incomparables = NULL, ...)
```

x, y

data frames, or objects to be coerced to one.

by, by.x, by.y

해당 기준으로 합친다. 없는것는들은 제거한다.

즉, 가장 짧은 기준으로 합친다.

5, 10 이라면 5만큼 길이가 설정됨.

all

logical; all = L is shorthand for all.x = L and all.y = L, where L is either TRUE or FALSE.

all.x

logical; if TRUE, then extra rows will be added to the output, one for each row in x that has no matching row in y. These rows will have NAs in those columns that are usually filled with values from y. The default is FALSE, so that only rows with data from both x and y are included in the output.

all.y

logical; analogous to all.x.

sort

logical. Should the result be sorted on the by columns?

suffixes

a character vector of length 2 specifying the suffixes to be used for making unique the names of columns in the result which not used for merging (appearing in by etc).

incomparables

values which cannot be matched. See match. This is intended to be used for merging on one column, so these are incomparable values of that column.

서로다른 크기의 데이터 프레임 합치기

일치하지 않는 컬럼을 단순히 <NA>로 채우고 싶다면,

plyr package의 **rbind.fill**을 사용하면 된다.

향상된 데이터 프레임 합치기

reshape2 package

[본인 정리](#)

[Reshape and aggregate data with the R package reshape2](#)

plyr package

[본인정리](#)

데이터 프레임 행과 열 크기 알아내기

nrow()

ncol()

help(package = "reshape2")

원본 참고 사이트: <http://seananderson.ca/2013/10/19/reshape.html>

reshape 2 website: <http://had.co.nz/reshape/>

서로 다른 데이터 타입들간의 데이터 상호 변환

| 변환 | 방법 | 주석 |
|----------------|--|---------------------------|
| 벡터 -> 리스트 | as.list(vec) | |
| 벡터 -> 행렬 | cbind(), as.matrix() rbind(), matrix(vec,n,m) # n by m 행렬 생성 | |
| 벡터 -> 데이터 프레임 | #열 한 개짜리 데이터 프레임 as.data.frame(vec) # 행 한 개짜리 데이터 프레임 as.data.frame(rbind(vec)) | |
| 리스트 -> 벡터 | unlist(list) | as.vector 대신에 unlist가 좋다. |
| 리스트 -> 행렬 | 열 한 개짜리 행렬: as.matrix(list) 행 한 개짜리 행렬: as.matrix(rbind(list)) n by m 행렬: matrix(list,n,m) | |
| 리스트 -> 데이터 프레임 | 목록 원소들이 데이터의 열이면: as.data.frame(list) | |
| 행렬 -> 벡터 | as.vector(mat) | 행렬의 모든 원소들을 벡터로 반환한다. |
| 행렬 -> 리스트 | as.list(mat) | 행렬의 모든 원소들을 리스트로 변환한다. |
| 행렬 -> 데이터 프레임 | as.data.frame(mat) | |
| 데이터 프레임 -> 벡터 | dfm[1,] #행 하나 짜리 데이터 프레임 변환 dfm[,1] or dfm[[1]] #열 하나짜리 데이터 프 | |

| | | |
|----------------|----------------|--|
| | 레이미 변환 | |
| 데이터 프레임 -> 리스트 | as.list(dfm) | |
| 데이터 프레임 -> 행렬 | as.matrix(dfm) | |

<<http://goodtogreate.tistory.com/entry/%EB%8D%B0%EC%9D%B4%ED%84%B0%EA%B5%AC%EC%A1%B0>>에서 삽입

runif

2017년 10월 23일 월요일 오전 11:07

```
runif(n, min = 0, max = 1)
```

runif()

min~max까지의 임의의 수를 n개 만큼 출력

```
> runif(1)
[1] 0.3679235
> runif(2)
[1] 0.3249376 0.8924986
> runif(3)
[1] 0.90463168 0.25608766 0.05757015
```

```
> x<-runif(1)
> x
[1] 0.7962147
> max(x)
[1] 0.7962147
> min(x)
[1] 0.7962147
```

```
> x<-runif(4)
> x
[1] 0.2826901 0.2726814 0.1997790 0.2308234
> max(x)
[1] 0.2826901
> min(x)
[1] 0.199779
```

```
> x<-runif(4, min=1, max=10)
> x
[1] 3.445505 4.076404 7.894061 4.010095
> max(x)
[1] 7.894061
> min(x)
[1] 3.445505
```

[참고][R] 여러개의 데이터프레임을 한꺼번에 하나의 데이터프레임으로 묶기, data.table package : rbindlist(data)

2017년 10월 23일 월요일 오전 11:20

[R] 여러개의 데이터프레임을 한꺼번에 하나의 데이터프레임으로 묶기, data.table package : rbindlist(data)

[R 분석과 프로그래밍/R 프로그래밍](#) 2016.07.10 16:49

이번 포스팅에서는 여러개의 데이터 프레임을 한꺼번에 하나의 데이터프레임으로 묶는 몇가지 방법을 알아보고, 성능 측면을 비교해보겠습니다.

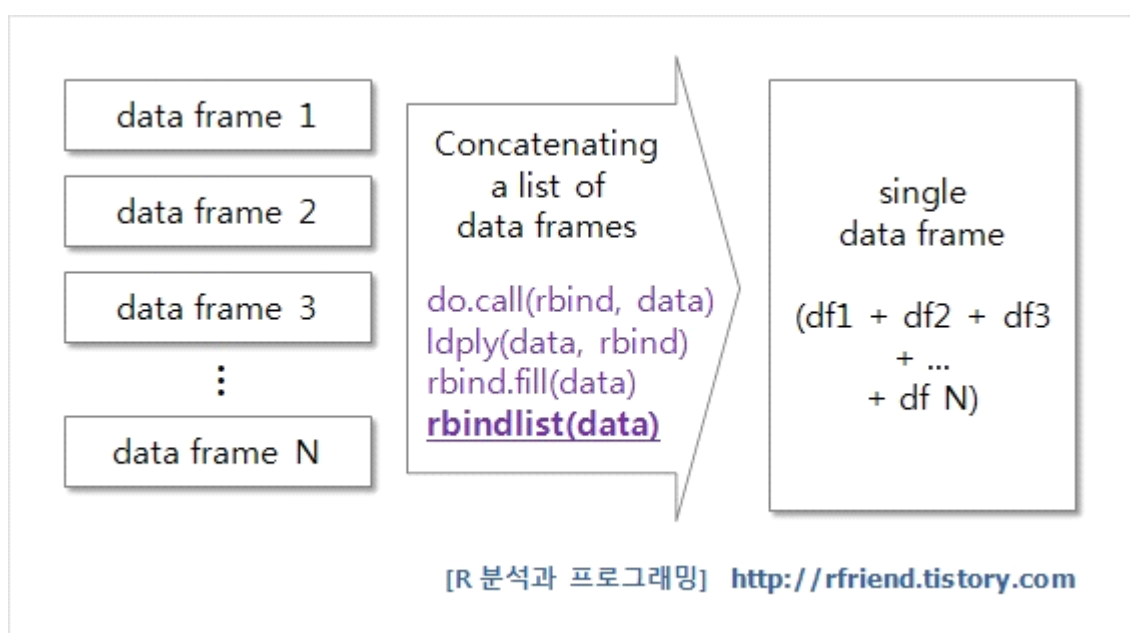
이번 포스팅은 [andrew 님이 r-bloggers.com 에 썼던 글](#)을 그대로 가져다가 번역을 한 내용입니다.

[Source] **Concatenating a list of data frames** , June 6, 2014, By andrew

* source : <http://www.r-bloggers.com/concatenating-a-list-of-data-frames/>

결론 먼저 말씀드리면, data.table package의 rbindlist(data) 함수가 속도 면에서 월등히 빠르네요.

[R로 여러개의 데이터프레임을 한꺼번에 하나의 데이터프레임으로 묶기]



0) 문제 (The problem)

아래처럼 3개의 칼럼으로 구성된 100,000 개의 자잘한 데이터 프레임을 한개의 커다란 데이터 프레임으로 합치는 것이 풀어야 할 문제, 미션입니다.

data = list() 로 해서 전체 데이터 프레임들을 data라는 리스트로 만들어서 아래 각 방법별 예제에 사용하였습니다.

```
> #####
> ## Concatenating a list of data frames
> ## do.call(rbind, data)
> ## ldply(data, rbind)
> ## rbind.fill(data)
> ## rbindlist(data) ** winner! **
> #####
>
> ## The Problem
>
> data = list()
>
> N = 100000
>
> for (n in 1:N) {
+   data[[n]] = data.frame(index = n,
+                           char = sample(letters, 1),
+                           z = runif(1))
+ }
>
> data[[1]]
  index char      z
1      1    j 0.2300154
```

1) The naive solution : do.call(rbind, data)

가장 쉽게 생각할 수 있는 방법으로 base package에 포함되어 있는 rbind() 함수를 do.call 함수로 계속 호출해서 여러개의 데이터 프레임을 위/아래로 합치는 방법입니다.

이거 한번 돌리니 정말 시간 오래 걸리네요. @@~ 낮잠 잠깐 자고 와도 될 정도로요.

```
> ## (1) The Naive Solution
> head(do.call(rbind, data))
  index char      z
1      1    j 0.2300154
2      2    f 0.63555284
3      3    d 0.65774397
4      4    y 0.46550511
5      5    b 0.02688307
6      6    u 0.19057217
```


2-1) plyr package : ldply(data, rbind)

두번째 방법은 plyr package의 ldply(data, rbind) 함수를 사용하는 방법입니다.

```
> ## (2) Alternative Solutions #1 and #2
> ## (2-1) plyr package : ldply(data, rbind)
> install.packages("plyr")
> library(plyr)
> head(ldply(data, rbind))
  index char      z
1     1   j 0.23001541
2     2   f 0.63555284
3     3   d 0.65774397
4     4   y 0.46550511
5     5   b 0.02688307
6     6   u 0.19057217
```

2-2) plyr package : rbind.fill(data)

세번째 방법은 plyr package의 rbind.fill(data) 함수를 사용하는 방법입니다. 결과는 앞의 두 방법과 동일함을 알 수 있습니다.

```
> ## (2-2) plyr package : rbind.fill(data)
> library(plyr)
> head(rbind.fill(data))
  index char      z
1     1   j 0.23001541
2     2   f 0.63555284
3     3   d 0.65774397
4     4   y 0.46550511
5     5   b 0.02688307
6     6   u 0.19057217
```

3) data.table package : rbindlist(data)

마지막 방법은 data.table package의 rbindlist(data) 함수를 사용하는 방법입니다.

```

> ## (3) Alternative solution
> ## data.table package : rbindlist(data)
> install.packages("data.table")
> library(data.table)
> head(rbindlist(data))
   index char      z
1:     1    j 0.23001541
2:     2    f 0.63555284
3:     3    d 0.65774397
4:     4    y 0.46550511
5:     5    b 0.02688307
6:     6    u 0.19057217

```

4) 벤치마킹 테스트 (bechmarking test)

```

> ## Benchmarking (performance comparison)
> install.packages("rbenchmark")
> library(rbenchmark)
> benchmark(do.call(rbind, data),
+           ldply(data, rbind),
+           rbind.fill(data),
+           rbindlist(data))

```

| | test | replications | elapsed | relative | user.self | sys.self | user.child | sys.child |
|---|----------------------|--------------|----------|----------|-----------|----------|------------|-----------|
| 1 | do.call(rbind, data) | 100 | 11387.82 | 668.692 | 11384.15 | 1.54 | NA | NA |
| 2 | ldply(data, rbind) | 100 | 4983.72 | 292.644 | 4982.90 | 0.52 | NA | NA |
| 3 | rbind.fill(data) | 100 | 1480.46 | 86.932 | 1480.23 | 0.17 | NA | NA |
| 4 | rbindlist(data) | 100 | 17.03 | 1.000 | 16.86 | 0.17 | NA | NA |

패키지/함수별 성능 비교를 해본 결과 data.table 패키지의 rbindlist(data) 함수가 월등히 빠르다는 것을 알 수 있습니다. 위의 벤치마킹 결과를 보면, 속도가 가장 빨랐던 rbindlist(data)를 1로 났을 때, 상대적인 속도(relative 칼럼)를 보면 rbind.fill(data)가 86.932로서 rbindlist(data)보다 86배 더 오래걸리고, ldply(data, rbind)가 292.644로서 rbindlist(data)보다 292배 더 오래걸린다는 뜻입니다. do.call(rbind, data)는 rbindlist(data) 보다 상대적으로 668.692 배 더 시간이 걸리는 것으로 나오네요.

rbindlist(data)가 월등히 속도가 빠른 이유는 두가지인데요,

- (1) rbind() 함수가 각 데이터 프레임의 칼럼 이름을 확인하고, 칼럼 이름이 다를 경우 재정렬해서 합치는데 반해, data.table 패키지의 rbindlist() 함수는 각 데이터 프레임의 칼럼 이름을 확인하지 않고 단지 위치(position)를 기준으로 그냥 합쳐버리기 때문이며,
- (따라서, rbindlist() 함수를 사용하려면 각 데이터 프레임의 칼럼 위치가 서로 동일해야 함)

(2) `rbind()` 함수는 R code로 작성된 반면에, `data.table` 패키지의 `rbindlist()` 는 C 언어로 코딩이 되어있기 때문입니다.

많은 도움이 되었기를 바랍니다.

이번 포스팅이 도움이 되었다면 아래의 '공감 ~♡'를 꾸욱 눌러주세요.

<<http://rfriend.tistory.com/225>>에서 삽입

최대, 최소값 구하기

2017년 10월 23일 월요일 오전 11:28

```
library('rhdfs')
library('rmr2')

hdfs.init()

intsArr=to.dfs(1:10)

result<-mapreduce(
  input = intsArr,
  map=function(k, v){
    lapply(seq_along(v), function(r){
      x<-runif(v[[r]])
      keyval(r, c(max(x), min(x)))
    })
  }
)
result
outputData <- from.dfs(result)
outputData
class(outputData)
outputData$val
lapply(outputData$val,"[", 1)
lapply(outputData$val,"[", 2)

tableOutput<-do.call('rbind', lapply(outputData$val,"[", 2))
tableOutput

barplot(tableOutput,beside = T, names=c('max', 'min'))
```

Wordcount

2017년 10월 7일 토요일 오후 3:39

```
install.packages("wordcloud")
install.packages("RColorBrewer")
Sys.setenv(HADOOP_HOME="/usr/local/hadoop")
Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")
Sys.setenv(HADOOP_LIB_NATIVE_DIR="/usr/local/hadoop/lib/native")
Sys.setenv(HADOOP_STREAMING="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.1.jar")
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-8-oracle/")
library(rJava)
library(rhdfs)
hdfs.init()
library(rmr2)
library(wordcloud)
library(RColorBrewer)

hdfs.del("/tmp/testoutput")
# Defining wordcount MapReduce function
wordcount = function(input,
                      output = NULL,
                      pattern = " "){

# Defining wordcount Map function
wc.map = function(., lines) {
  keyval(
    unlist(strsplit(
      x = lines,
      split = pattern)),
    1)}

# Defining wordcount Reduce function
wc.reduce = function(word, counts ) {
  keyval(word, sum(counts))}

# Defining MapReduce parameters by calling mapreduce function

mapreduce(input = input ,
          output = output,
          input.format = "text",
          map = wc.map,
          reduce = wc.reduce,
          combine = T)}

result = wordcount('/tmp/test.txt', '/tmp/testoutput')
fileData = from.dfs(result)

palette = brewer.pal(9,"Set1")
wordcloud(fileData$key, freq=fileData$val,
          scale=c(5,1), random.order = FALSE,
          rot.per= 0.1, min.freq=40, random.color = T, colors=palette)
```

[illegible]

min.freq는 찾고자하는 단어의 최소 개수이다.

2글자 이상

2017년 10월 23일 월요일 오후 2:31

```
install.packages("wordcloud")
install.packages("RColorBrewer")
Sys.setenv(HADOOP_HOME="/usr/local/hadoop")
Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")
Sys.setenv(HADOOP_LIB_NATIVE_DIR="/usr/local/hadoop/lib/native")
Sys.setenv(HADOOP_STREAMING="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.1.jar")
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-8-oracle/")
library(rJava)
library(rhdfs)
hdfs.init()
library(rmr2)
library(wordcloud)
library(RColorBrewer)

hdfs.del("/tmp/testoutput")
# Defining wordcount MapReduce function
wordcount = function(input,
                      output = NULL,
                      pattern = " "){

# Defining wordcount Map function
wc.map = function(., lines) {
  keyval(
    unlist(strsplit(
      x = lines,
      split = pattern)),
    1)}

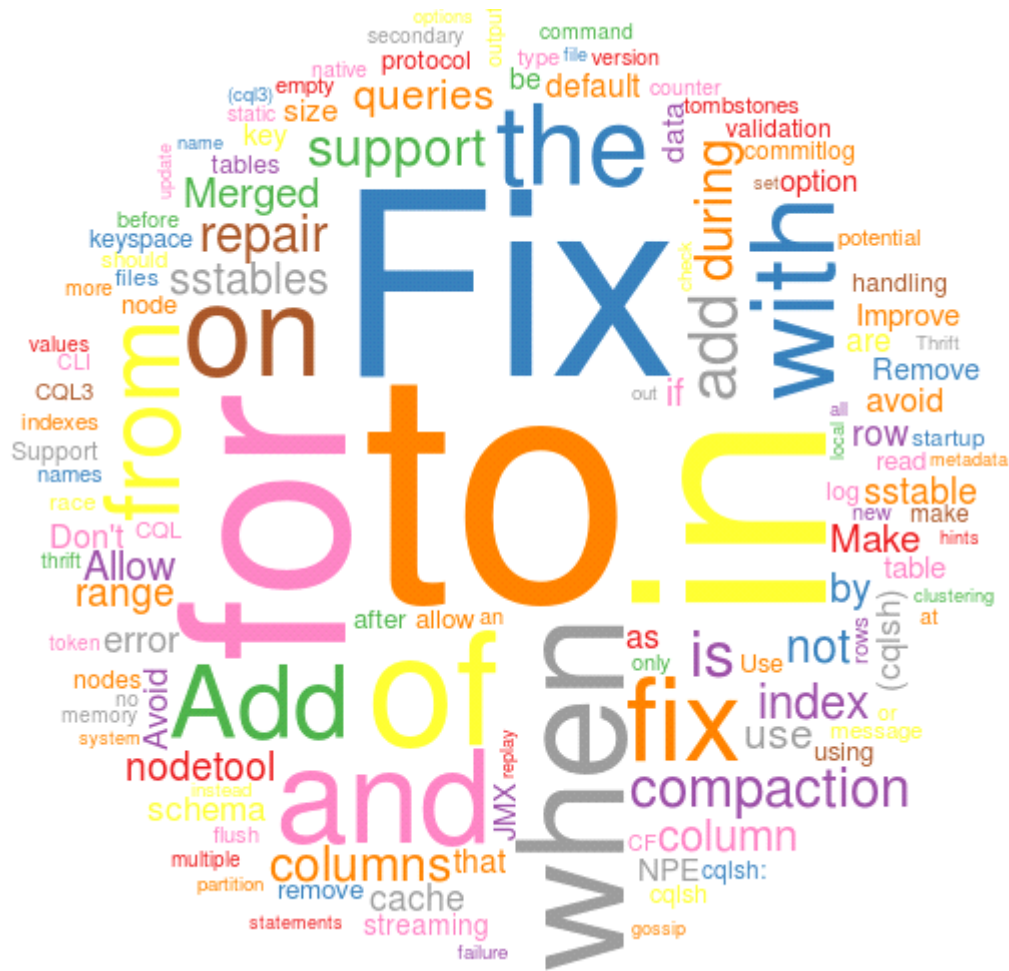
# Defining wordcount Reduce function
wc.reduce = function(word, counts ) {
  if(nchar(word)>=2)
    keyval(word, sum(counts))}

# Defining MapReduce parameters by calling mapreduce function

mapreduce(input = input ,
          output = output,
          input.format = "text",
          map = wc.map,
          reduce = wc.reduce,
          combine = T)}

result = wordcount('/tmp/test.txt', '/tmp/testoutput')
fileData = from.dfs(result)

palette = brewer.pal(9,"Set1")
wordcloud(fileData$key, freq=fileData$val,
          scale=c(10,0.2), random.order = F,
          rot.per= 0.1, min.freq=40, random.color = T, colors=palette)
```

상위 30개만 추출

2017년 10월 24일 화요일 오후 9:02

```
install.packages("wordcloud")
```

```
Sys.setenv(HADOOP_HOME="/usr/local/hadoop")
Sys.setenv(HADOOP_CMD="/usr/local/hadoop/bin/hadoop")
Sys.setenv(HADOOP_LIB_NATIVE_DIR="/usr/local/hadoop/lib/native")
Sys.setenv( HADOOP_STREAMING="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-
streaming-2.8.1.jar")
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-8-oracle/")
library(rJava)
library(rhdfs)
hdfs.init()
library(rmr2)
library(wordcloud)
library(RColorBrewer)
```

```
hdfs.del("/tmp/testoutput")
# Defining wordcount MapReduce function
wordcount = function(input,
                      output = NULL,
                      pattern = " "){
```

```
  # Defining wordcount Map function
  wc.map = function(., lines) {
    keyval(
      unlist(strsplit(
        x = lines,
        split = pattern)),
      1))
```

```
  # Defining wordcount Reduce function
  wc.reduce = function(word, counts ) {
    if(nchar(word)>=2)
      keyval(word, sum(counts))
  }
```

```
# Defining MapReduce parameters by calling mapreduce function
```

```
mapreduce(input = input ,
```

```
output = output,  
input.format = "text",  
map = wc.map,  
reduce = wc.reduce,  
combine = T})
```

```
result = wordcount('/wordcount/input/', '/tmp/testoutput')  
result  
fileData = from.dfs(result)
```

```
fileData.df <- as.data.frame(fileData, stringsAsFactors=F)  
colnames(fileData.df) <- c('word', 'count')  
head_test <- head(fileData.df[order(fileData.df$count, decreasing = T), ], 30)
```

```
head_test  
palette = brewer.pal(9,"Set1")  
wordcloud(head_test$word, freq=head_test$count,  
          scale=c(5,1), random.order = FALSE,  
          rot.per= 0.1, min.freq=100, random.color = T, colors=palette)
```

[참고]R강의(1) Data 다루기

2017년 10월 24일 화요일 오후 1:21

R강의(1) Data 다루기

문건웅

2014년 6월 25일

R, Rstudio 설치하기

R 설치하기

- R-project www.R-project.org
- RStudio www.rstudio.com

R에서 쓰는 표현식, 연산자

```
r=2
circle = pi*r^2
total=100; n=10
average <- total/n
5**2
## [1] 25
(1+2)*3
## [1] 9
```

R에서 모든 데이터는 벡터다

변수에 데이터를 넣는 방법

- 변수에 데이터 할당 ; =, <-
- combine 사용
- sequence 연산자 사용 (:)
- sequence 함수 사용 (seq)
- repeat함수 사용(rep)

```
x=1
y<-2
a=c(1,2,3)
a
## [1] 1 2 3
a[2]
## [1] 2
b=1:10
b[9]
## [1] 9
c=seq(5)
d=seq(1,3,0.25)
e=c(a,b)
f=rep(a,3)
f
## [1] 1 2 3 1 2 3 1 2 3
```

기본데이터형

- 숫자형(numeric) 12, 4, 0.45
- 논리형(logical) TRUE, FALSE, T, F, 1, 0
- 복소수형(complex) 3+2i
- 문자형(character) "St.Vincent's Hospital", "123", '3.14'

데이터 구조

- 벡터(vector)
- 행렬(matrix)
- 배열(array)
- 데이터프레임(dataframe)
- 리스트(list)
- 범주형자료(categorical variable)
- 시계열(Time series)

왜 벡터로 되어있을까?

- 언제든지 자료를 추가할 수 있다.

```
a=1:5
```

```
a=c(a,101,102)
```

```
b=c(a,103)
```

```
b
```

```
## [1] 1 2 3 4 5 101 102 103
```

- 자료의 연산이 아주 쉽다.

```
Height=c(168,173,160,145,180)
```

```
Weight=c(80,65,92,53,76)
```

```
BMI=Weight/(Height/100)^2
```

```
BMI
```

```
## [1] 28.34 21.72 35.94 25.21 23.46
```

- 연산에서 벡터는 재사용된다

```
a=1:10
```

```
b=c(1,-1)
```

```
a+b
```

```
## [1] 2 1 4 3 6 5 8 7 10 9
```

- $b^2 + 4ac$

```
b=10
```

```
a=c=2
```

```
b^2+c(1,-1)*4*a*c
```

```
## [1] 116 84
```

행렬이란 무엇인가 ?

- 자료를 2차원으로 배열한 것
- 수학의 행렬과 다른 점은 숫자 이외에도 가능하다는 것이다.

```
a=matrix(1:12,ncol=3)
```

```
a
```

```
##      [,1] [,2] [,3]
```

```
## [1,]  1  5  9
```

```
## [2,]  2  6 10
```

```
## [3,] 3 7 11
## [4,] 4 8 12
b=LETTERS[1:12]
b
## [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"
b=matrix(b,ncol=4)
b
##      [,1] [,2] [,3] [,4]
## [1,] "A"  "D"  "G"  "J"
## [2,] "B"  "E"  "H"  "K"
## [3,] "C"  "F"  "I"  "L"
b[3,2]
## [1] "F"
b[2,4]
## [1] "K"
b[2,]
## [1] "B" "E" "H" "K"
b[,3]
## [1] "G" "H" "I"
length(b)
## [1] 12
```

범주형자료

- 성별 : 남,여

```
sex=c("Male","Female","Female","Male","Male")
sex=factor(sex)
sex
## [1] Male Female Female Male Male
## Levels: Female Male
str(sex)
## Factor w/ 2 levels "Female","Male": 2 1 1 2 2
levels(sex)
## [1] "Female" "Male"
length(sex)
## [1] 5
```

- 흡연 : "none", "ex-smoker", "smoker"

```
smoking=c(1,1,2,3,1)
smoking=factor(smoking)
levels(smoking)=c("none","ex-smoker","smoker")
smoking
## [1] none none ex-smoker smoker none
## Levels: none ex-smoker smoker
```

배열 (array)

- 행렬과 비슷하나 다차원구조를 가질수 있다.
- 2차원 배열 = 행렬

데이터프레임

- 우리가 다루는 거의 모든 자료는 데이터프레임이다.

```
mydata=data.frame(height=Height,weight=Weight,sex=sex,smoking=smoking)
mydata
## height weight sex smoking
## 1 168 80 Male none
```

```
## 2 173 65 Female none
## 3 160 92 Female ex-smoker
## 4 145 53 Male smoker
## 5 180 76 Male none
```

- 데이터프레임의 자료를 일부 선택(subset)할 때는 행렬과 비슷하다.

```
mydata[3,]
## height weight sex smoking
## 3 160 92 Female ex-smoker
mydata[,1]
## [1] 168 173 160 145 180
```

- 데이터프레임의 열에 이름으로 접근할 때는 \$기호를 쓴다.

```
mydata$height
## [1] 168 173 160 145 180
```

- 새로운 열을 추가할 때

```
mydata$BMI=mydata$weight*10000/(mydata$height)^2
mydata
## height weight sex smoking BMI
## 1 168 80 Male none 28.34
## 2 173 65 Female none 21.72
## 3 160 92 Female ex-smoker 35.94
## 4 145 53 Male smoker 25.21
## 5 180 76 Male none 23.46
str(mydata)
## 'data.frame': 5 obs. of 5 variables:
## $ height : num 168 173 160 145 180
## $ weight : num 80 65 92 53 76
## $ sex : Factor w/ 2 levels "Female","Male": 2 1 1 2 2
## $ smoking: Factor w/ 3 levels "none","ex-smoker",...: 1 1 2 3 1
## $ BMI : num 28.3 21.7 35.9 25.2 23.5
summary(mydata)
## height weight sex smoking BMI
## Min. :145 Min. :53.0 Female:2 none :3 Min. :21.7
## 1st Qu.:160 1st Qu.:65.0 Male :3 ex-smoker:1 1st Qu.:23.5
## Median :168 Median :76.0 smoker :1 Median :25.2
## Mean :165 Mean :73.2 Mean :26.9
## 3rd Qu.:173 3rd Qu.:80.0 3rd Qu.:28.3
## Max. :180 Max. :92.0 Max. :35.9
plot(mydata)
```

진짜 데이터를 가지고 실습

```
data(mtcars)
head(mtcars,10)
## mpg cyl disp hp drat wt qsec vs am gear carb
## Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4
## Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4
## Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1
## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1
## Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2
## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1
## Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4
## Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
## Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4
str(mtcars)
## 'data.frame': 32 obs. of 11 variables:
```

```

## $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
## $ disp: num  160 160 108 258 360 ...
## $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num  16.5 17 18.6 19.4 17 ...
## $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
## $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
## $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
## $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
summary(mtcars)
##      mpg      cyl      disp      hp
## Min.   :10.4   Min.   :4.00   Min.   :71.1   Min.   :52.0
## 1st Qu.:15.4   1st Qu.:4.00   1st Qu.:120.8   1st Qu.: 96.5
## Median :19.2   Median :6.00   Median :196.3   Median :123.0
## Mean   :20.1   Mean   :6.19   Mean   :230.7   Mean   :146.7
## 3rd Qu.:22.8   3rd Qu.:8.00   3rd Qu.:326.0   3rd Qu.:180.0
## Max.   :33.9   Max.   :8.00   Max.   :472.0   Max.   :335.0
##      drat      wt      qsec      vs
## Min.   :2.76   Min.   :1.51   Min.   :14.5   Min.   :0.000
## 1st Qu.:3.08   1st Qu.:2.58   1st Qu.:16.9   1st Qu.:0.000
## Median :3.69   Median :3.33   Median :17.7   Median :0.000
## Mean   :3.60   Mean   :3.22   Mean   :17.8   Mean   :0.438
## 3rd Qu.:3.92   3rd Qu.:3.61   3rd Qu.:18.9   3rd Qu.:1.000
## Max.   :4.93   Max.   :5.42   Max.   :22.9   Max.   :1.000
##      am      gear      carb
## Min.   :0.000   Min.   :3.00   Min.   :1.00
## 1st Qu.:0.000   1st Qu.:3.00   1st Qu.:2.00
## Median :0.000   Median :4.00   Median :2.00
## Mean   :0.406   Mean   :3.69   Mean   :2.81
## 3rd Qu.:1.000   3rd Qu.:4.00   3rd Qu.:4.00
## Max.   :1.000   Max.   :5.00   Max.   :8.00
mtcars$mpg
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
stem(mtcars$mpg)
##
## The decimal point is at the |
##
## 10 | 44
## 12 | 3
## 14 | 3702258
## 16 | 438
## 18 | 17227
## 20 | 00445
## 22 | 88
## 24 | 4
## 26 | 03
## 28 |
## 30 | 44
## 32 | 49
hist(mtcars$mpg)
boxplot(mtcars$mpg)
fivenum(mtcars$mpg)
## [1] 10.40 15.35 19.20 22.80 33.90

```



```
quantile(mtcars$mpg)
## 0% 25% 50% 75% 100%
## 10.40 15.43 19.20 22.80 33.90
```

데이타의 정렬(order)

```
order(mtcars$mpg)
## [1] 15 16 24 7 17 31 14 23 22 29 12 13 11 6 5 10 25 30 1 2 4 32 21
## [24] 3 9 8 27 26 19 28 18 20
mtcars=mtcars[order(mtcars$mpg),]
head(mtcars)
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Cadillac Fleetwood 10.4  8 472 205 2.93 5.250 17.98 0 0  3  4
## Lincoln Continental 10.4  8 460 215 3.00 5.424 17.82 0 0  3  4
## Camaro Z28         13.3  8 350 245 3.73 3.840 15.41 0 0  3  4
## Duster 360         14.3  8 360 245 3.21 3.570 15.84 0 0  3  4
## Chrysler Imperial 14.7  8 440 230 3.23 5.345 17.42 0 0  3  4
## Maserati Bora      15.0  8 301 335 3.54 3.570 14.60 0 1  5  8
rownames(mtcars)
## [1] "Cadillac Fleetwood" "Lincoln Continental" "Camaro Z28"
## [4] "Duster 360"        "Chrysler Imperial"  "Maserati Bora"
## [7] "Merc 450SLC"       "AMC Javelin"        "Dodge Challenger"
## [10] "Ford Pantera L"    "Merc 450SE"         "Merc 450SL"
## [13] "Merc 280C"         "Valiant"            "Hornet Sportabout"
## [16] "Merc 280"          "Pontiac Firebird"   "Ferrari Dino"
## [19] "Mazda RX4"         "Mazda RX4 Wag"      "Hornet 4 Drive"
## [22] "Volvo 142E"        "Toyota Corona"      "Datsun 710"
## [25] "Merc 230"          "Merc 240D"          "Porsche 914-2"
## [28] "Fiat X1-9"         "Honda Civic"        "Lotus Europa"
## [31] "Fiat 128"          "Toyota Corolla"
order(rownames(mtcars))
## [1] 8 1 3 5 24 9 4 18 31 28 10 29 21 15 2 30 6 19 20 25 26 16 13
## [24] 11 12 7 17 27 32 23 14 22
mtcars=mtcars[order(rownames(mtcars)),]
mtcars=mtcars[order(mtcars$mpg,mtcars$wt),]
```

데이타의 일부 선택(subset)

```
# 4기통, 6기통, 8기통 중 4,6기통 만 선택
table(mtcars$cyl)
##
## 4 6 8
## 11 7 14
mtcars$cyl<7
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
## [23] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
mtcars1=mtcars[mtcars$cyl<7,]
table(mtcars1$cyl)
##
## 4 6
## 11 7
# subset(data.frame, subset(행), select(열))
mtcars1=subset(mtcars,cyl<7)
mtcars2=subset(mtcars1,select=c(mpg,cyl))
```

데이터의 요약 ; 테이블만들기, 카이제곱, 피셔검정

```
table(mtcars$cyl)
##
## 4 6 8
## 11 7 14
help(mtcars)
table(mtcars$cyl,mtcars$am)
##
## 0 1
## 4 3 8
## 6 4 3
## 8 12 2
mtcars$tm=factor(mtcars$am,labels=c("automatic","manual"))
# mtcars$tm=ifelse(mtcars$am=="automatic","automatic","manual")
str(mtcars)
## 'data.frame': 32 obs. of 12 variables:
## $ mpg : num 10.4 10.4 13.3 14.3 14.7 15 15.2 15.2 15.5 15.8 ...
## $ cyl : num 8 8 8 8 8 8 8 8 8 8 ...
## $ disp: num 472 460 350 360 440 ...
## $ hp : num 205 215 245 245 230 335 150 180 150 264 ...
## $ drat: num 2.93 3 3.73 3.21 3.23 3.54 3.15 3.07 2.76 4.22 ...
## $ wt : num 5.25 5.42 3.84 3.57 5.34 ...
## $ qsec: num 18 17.8 15.4 15.8 17.4 ...
## $ vs : num 0 0 0 0 0 0 0 0 0 0 ...
## $ am : num 0 0 0 0 0 1 0 0 0 1 ...
## $ gear: num 3 3 3 3 3 5 3 3 3 5 ...
## $ carb: num 4 4 4 4 4 8 2 3 2 4 ...
## $ tm : Factor w/ 2 levels "automatic","manual": 1 1 1 1 1 2 1 1 1 2 ...
result=table(mtcars$cyl,mtcars$tm)
result
##
## automatic manual
## 4 3 8
## 6 4 3
## 8 12 2
chisq.test(result)
## Warning: Chi-squared approximation may be incorrect
##
## Pearson's Chi-squared test
##
## data: result
## X-squared = 8.741, df = 2, p-value = 0.01265
plot(result)
barplot(result,legend=paste(rownames(result),"cyl"))
#xtabs(도수~가로+세로)
result1=xtabs(~cyl+tm,data=mtcars)
result1
## tm
## cyl automatic manual
## 4 3 8
## 6 4 3
## 8 12 2
addmargins(result1)
## tm
## cyl automatic manual Sum
```

```
## 4      3      8 11
## 6      4      3 7
## 8     12      2 14
## Sum    19    13 32
chisq.test(result1)
## Warning: Chi-squared approximation may be incorrect
##
## Pearson's Chi-squared test
##
## data: result1
## X-squared = 8.741, df = 2, p-value = 0.01265
#fisher.test(result1)
```

데이타의 요약 : 평균 구하기

```
plot(mtcars)
# 엔진수에 따른 연비 평균
tapply(mtcars$mpg,mtcars$cyl,mean)
## 4 6 8
## 26.66 19.74 15.10
aggregate(mpg~cyl,data=mtcars,mean)
## cyl mpg
## 1 4 26.66
## 2 6 19.74
## 3 8 15.10
aggregate(mpg~cyl+am,data=mtcars,mean)
## cyl am mpg
## 1 4 0 22.90
## 2 6 0 19.12
## 3 8 0 15.05
## 4 4 1 28.07
## 5 6 1 20.57
## 6 8 1 15.40
# 엔진수에 따른 엔진출력(마력) 평균
tapply(mtcars$hp,mtcars$cyl,mean)
## 4 6 8
## 82.64 122.29 209.21
plot(mpg~cyl,data=mtcars)
boxplot(mpg~cyl,data=mtcars)
out=lm(mpg~factor(cyl),data=mtcars)
anova(out)
## Analysis of Variance Table
##
## Response: mpg
##      Df Sum Sq Mean Sq F value Pr(>F)
## factor(cyl) 2 825 412 39.7 5e-09 ***
## Residuals 29 301 10
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

결측값의 처리

```
Height=c(168,173,160,145,NA,180)
mean(Height)
## [1] NA
lis.na(Height)
## [1] TRUE TRUE TRUE TRUE FALSE TRUE
```

```
mean(Height[!is.na(Height)])
## [1] 165.2
mean(Height,na.rm=TRUE)
## [1] 165.2
```

상관분석, 회귀분석

```
# 마력과 연비
cor.test(mtcars$mpg,mtcars$hp)
##
## Pearson's product-moment correlation
##
## data: mtcars$mpg and mtcars$hp
## t = -6.742, df = 30, p-value = 1.788e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.8853 -0.5861
## sample estimates:
## cor
## -0.7762
with(mtcars,cor.test(mpg,hp))
##
## Pearson's product-moment correlation
##
## data: mpg and hp
## t = -6.742, df = 30, p-value = 1.788e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.8853 -0.5861
## sample estimates:
## cor
## -0.7762
plot(mpg~hp,data=mtcars)
out1=lm(mpg~hp,data=mtcars)
summary(out1)
##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
##
## Residuals:
## Min 1Q Median 3Q Max
## -5.712 -2.112 -0.885 1.582 8.236
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.0989 1.6339 18.42 < 2e-16 ***
## hp -0.0682 0.0101 -6.74 1.8e-07 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.86 on 30 degrees of freedom
## Multiple R-squared: 0.602, Adjusted R-squared: 0.589
## F-statistic: 45.5 on 1 and 30 DF, p-value: 1.79e-07
abline(out1,col="red")
```

연속형 자료의 변형

자료를 다루다 보면 연속형 자료에서 새로운 범주형 자료를 만들어야 할 때가 있다. ggplot2패키

지에 있는 diamonds 자료 예를 들어보면

```
library(ggplot2)
##
## Attaching package: 'ggplot2'
##
## The following object is masked _by_ '.GlobalEnv':
##
## diamonds
data(diamonds)
str(diamonds)
## 'data.frame': 53940 obs. of 10 variables:
## $ carat : num 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
## $ cut : Ord.factor w/ 5 levels "Fair"<"Good"<...: 5 4 2 4 2 3 3 3 1 3 ...
## $ color : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<...: 2 2 2 6 7 7 6 5 2 5 ...
## $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<...: 2 3 5 4 2 6 7 3 4 5 ...
## $ depth : num 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
## $ table : num 55 61 65 58 58 57 57 55 61 61 ...
## $ price : int 326 326 327 334 335 336 336 337 337 338 ...
## $ x : num 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
## $ y : num 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
## $ z : num 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
summary(diamonds)
## carat cut color clarity
## Min. :0.200 Fair :1610 D: 6775 SI1 :13065
## 1st Qu.:0.400 Good :4906 E: 9797 VS2 :12258
## Median :0.700 Very Good:12082 F: 9542 SI2 :9194
## Mean :0.798 Premium :13791 G:11292 VS1 :8171
## 3rd Qu.:1.040 Ideal :21551 H: 8304 VVS2 :5066
## Max. :5.010 I: 5422 VVS1 :3655
## J: 2808 (Other): 2531
## depth table price x
## Min. :43.0 Min. :43.0 Min. : 326 Min. : 0.00
## 1st Qu.:61.0 1st Qu.:56.0 1st Qu.: 950 1st Qu.: 4.71
## Median :61.8 Median :57.0 Median : 2401 Median : 5.70
## Mean :61.8 Mean :57.5 Mean : 3933 Mean : 5.73
## 3rd Qu.:62.5 3rd Qu.:59.0 3rd Qu.: 5324 3rd Qu.: 6.54
## Max. :79.0 Max. :95.0 Max. :18823 Max. :10.74
##
## y z
## Min. : 0.00 Min. : 0.00
## 1st Qu.: 4.72 1st Qu.: 2.91
## Median : 5.71 Median : 3.53
## Mean : 5.73 Mean : 3.54
## 3rd Qu.: 6.54 3rd Qu.: 4.04
## Max. :58.90 Max. :31.80
##
```

다이아몬드 가격이 제일 싼 것은 326불 제일 비싼 것은 18823불이다.

diamonds 데이터에 PriceGroup이라는 새로운 변수를 만들고 1000불 미만은 1, 1000불-5000불은 2, 5000불 이상은 3으로 바꾸려면 다음과 같이 한다.

1. 첫번째 방법 :

```
diamonds$PriceGroup=1
diamonds$PriceGroup[diamonds$price>=1000]=2
diamonds$PriceGroup[diamonds$price>=5000]=3
```

```
table(diamonds$PriceGroup)
##
## 1 2 3
## 14499 24714 14727
```

2. ifelse 함수 사용

```
diamonds$PriceGroup=ifelse(diamonds$price<1000,1,ifelse(diamonds$price<5000,2,3))
table(diamonds$PriceGroup)
##
## 1 2 3
## 14499 24714 14727
```

3. cut, break 사용

```
diamonds$PriceGroup=cut(diamonds$price,breaks=c(0,999,4999,99999),labels=c(1,2,3))
table(diamonds$PriceGroup)
##
## 1 2 3
## 14499 24714 14727
```

특정 값이 아니라 price 순으로 k개의 구간으로 나누고 싶을때

예를 들어 전체 다이아몬드 가격을 1등 부터 53940등까지 순위를 매기고 이를 같은 숫자 만큼 k개의 군으로 나누고 싶다면 어떻게 할까? 다음과 같은 함수를 만들어 보았다. rank2group 함수는 y라는 벡터를 인자로 받아들여 순위별로 k개의 군으로 나누어진 새로운 벡터를 반환한다. 사용법은 다음과 같다.

```
rank2group <- function (y,k=4){
  count=length(y)
  z=rank(y,ties.method="min")
  return(floor((z-1)/(count/k))+1)
}
diamonds$PriceGroup=rank2group(diamonds$price,4)
table(diamonds$PriceGroup)
##
## 1 2 3 4
## 13490 13495 13470 13485
aggregate(price~PriceGroup,data=diamonds,range)
## PriceGroup price.1 price.2
## 1 1 326 950
## 2 2 951 2401
## 3 3 2402 5324
## 4 4 5325 18823
```

가격이 겹치는 데이터(즉, 순위가 같은 데이터)가 있어 네군별로 n수가 다르기는 하지만 우리가 원하는대로 작동한다. 세군, 다섯군으로 나누려면 다음과 같이 하면 된다.

```
diamonds$PriceGroup3=rank2group(diamonds$price,3)
table(diamonds$PriceGroup3)
##
## 1 2 3
## 17996 17964 17980
aggregate(price~PriceGroup3,data=diamonds,range)
## PriceGroup3 price.1 price.2
## 1 1 326 1240
## 2 2 1241 4287
## 3 3 4288 18823
```

```

diamonds$PriceGroup5=rank2group(diamonds$price,5)
table(diamonds$PriceGroup5)
##
##  1  2  3  4  5
## 10796 10784 10789 10783 10788
aggregate(price~PriceGroup5,data=diamonds,range)
##  PriceGroup5 price.1 price.2
## 1      1    326    837
## 2      2    838   1698
## 3      3   1699   3465
## 4      4   3466   6301
## 5      5   6302  18823

```

<https://rstudio-pubs-static.s3.amazonaws.com/21770_5c3e975ba6744ba18714f64fc7079676.html>에서 삽입