~~~
---
title: "Pipe Operator"
author: "M. Affouf"
date: "January 9, 2017"
output: html_document
---

##The Pipe Operator
Functions in dplyr have been written in order to take advantage of what is
commonly
referred to as the "pipe" operator. The pipe operator, %>%, originates in
the magrittr
package and is by no means restricted to usage within dplyr. The pipe
operator allows us
to chain functions together such that the output from one function becomes
the input to the
first argument (by default) of the next. This has led to it being called
the "then" operator in
some quarters (do this, then this, then this, and so on). It is
particularly useful if we have
many steps to perform on a single type of object such as a data frame. The
advantage of
this approach is that it avoids intermediary objects (that is, those that
we create simply to
break up nested function calls).
Note: Piping to Other Arguments
When you use the pipe operator, the output from a function does not have
to be
used as the input to the first argument of the next function. It can in
fact become the
input to any argument within the following function. However, the code is
generally
a lot more readable if we feed the output into the first argument of the
following
function.
The dplyr package has been written with the pipe operator very much in
mind. In a typical
analysis workflow we might arrange, filter, select, mutate, group_by, and
summarize several times over. Each of these functions takes a data frame
as its first
input and returns another data frame as the output. This is ideal for
piping together
function calls. Consider the example in Listing 12.4 using mtcars. In the
first instance
we use the traditional approach to data processing. To avoid nesting, we
end up creating
three intermediate datasets on the way to obtaining our summary. We then
perform the
same operations using the pipe operator. In the second case, no
intermediate datasets are
required.
LISTING 12.4 Workflow Examples With and Without the Pipe Operator



```{r}
~~~

~~~~
# A standard workflow, mean mpg by cyl for manual cars
# The traditional way:
library(dplyr)
carsByCyl <- arrange(mtcars, cyl)
groupByCyl <- group_by(carsByCyl, cyl)
manualCars <- filter(groupByCyl, am == 1)
summarize(manualCars, Mean.MPG=mean(mpg))
```





```{r}
 # Using pipes
 mtcars %>%
 arrange(cyl) %>%
 group_by(cyl) %>%
 filter(am == 1) %>%
 summarize(Mean.MPG=mean(mpg))
```
~~~~