Appendix for the VADMBC R Workshop

So much to cover, so little time. See also http://dataservices.gmu.edu/software/r/

Data Frames

Fine-tuning import

Here are some arguments that are useful for modifying data when you are reading it in.

```
mydata <- read.csv("titanic.csv",
    as.is = "name",
    stringsAsFactors= FALSE ,
    na.strings = "99"
)
```

Making factors

If you have a numeric variable, you can easily make a factor. For example, if pclass were just 1, 2, 3:

```
mydata$pclass.f <- factor(  mydata$pclass,
    levels = c(1,2,3),
    labels = c("1st Class",  "2nd Class", "3rd Class"),
    ordered = TRUE
)
```

Changing labels

If you were to want to change the factor labels:

```
labels(mydata$gender) <- c("Males", "Females")
```

Formula Notation

```
object  <- goal( formula ,  data = mydata )
```

~	predicted from	+	include
:	interaction	*	factorial

In the below examples, X, Y, and Z are the names of variables in *mydata* .

Statistical Equation	R Formula
$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_I$	Y ~ X
$Y_i = \beta_0 + \beta_1 X_i + \beta_2 Z_i + \varepsilon_I$	Y ~ X + Z
$Y_i = \beta_0 + \beta_1 X_i + \beta_2 Z_i + \beta_3 X_i Z_i + \varepsilon_I$	Y ~ X + Z + X:Z
or	Y ~ X * Z

Creating Data for Tutorials

Many tutorials create data in order to show how functions work. Here are some ways this is done.

Use Vectors

It is very common to use vectors as variables without making them into a dataframe.

```
A  <- c(1,2,3,4,5)
B  <-  c(7:20, 200)
t.test(A,B)
```

Nesting

Everything in R can be "nested" and it can be very confusing to have several functions within other functions. This is identical to using Vectors.

```
t.test(1:10, c(7:20, 200))
```

Make a Dataframe

It is easy to turn vectors into data.frames.

```
group  <-  1:2
value  <-  rnorm(20)
data  <-  data.frame(group, value)
t.test(value ~ group, data=data)
```

Notice that t.test will accept **two vectors** of data or a **formula**. Formulas are better if one variable is the grouping variable and the other has values. Some will wrongly use vectors in this situation. Here are two ways to get the data for group 1, both are ugly.

```
data[data["group"]==1,2]
data$value[data$group==1]
```

Use Included Dataset

R comes with many datasets that are often used in examples, including iris and sleep. Many packages also include datasets that are used for examples. List installed datasets with: **??datasets**

```
t.test(extra ~ group, data = sleep)
```

© 2015 by Debby Kermer, Mason Library Data Services This work is licensed under the Attribution-NonCommercial-ShareAlike International License. http://creativecommons.org/licenses/by-nc-sa/4.0/

Creating Functions

Functions are objects, just like everything else. Here are two simple functions you can make quickly.

```
square <- function ( z )  z*z
square(5)

raise.to <- function ( q , n = 3 ) { q^n }
raise.to(5, n=2)  # 4^2
raise.to(5)       # 4^3, because 3 is default
raise.to(5, 4)    #  4^4; 2nd arg is n if not named
```

Bare Words that are NOT Objects

TRUE or **T**	**NaN**	(**N**ot **a** **N**umber)	**Inf**	(**Inf**inity)
FALSE or **F**	**NA**	(**N**ot **A**vailable)	**NULL**	(Empty)

NA – Not Available

R uses NA to represent **missing values**. However, many functions result in NA when ***any*** of the values are missing. The argument "na.rm" (rm = remove) is typically available and can be set to TRUE.

```
> ages <- mydata$age
> mean(ages)
[1] 43.76827
> ages[ages==99] <- NA
> mean(ages)
[1] NA
> mean(ages, na.rm = TRUE)
[1] 29.88114
```

NULL - Empty

To delete objects, use the rm() function. To remove variables in data.frames, set the value(s) to NULL.

```
> mydata$age <- ages
> rm(ages)
> mydata$embarked <- NULL
```

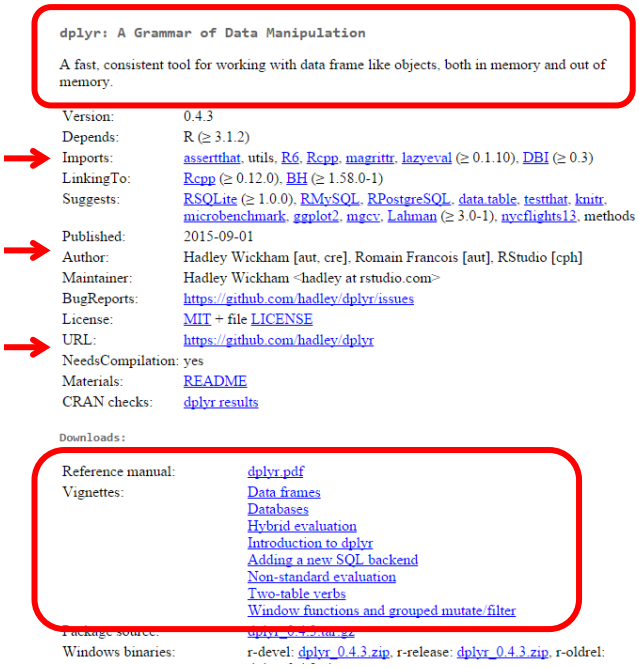
Piping & Chaining

Here is a situation where using the pipe operator is especially useful: creating a pivot table. The dplyr and tidyr packages work great with *chaining*.

```
library(dplyr)
library(tidyr)
mydata %>% group_by(pclass, gender) %>%
    summarize( pct = mean(survived) ) %>%
    spread( gender, pct )
```

R/CRAN Package Help Page

Documentation includes the **Reference Manual** and **Vignettes**. Go to the **URL** listed as it may also have more information. Note the **Published** date and **Author**/Maintainer to help identify good packages. The packages in **Imports** will also be installed.



dplyr: A Grammar of Data Manipulation

A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

Version:	0.4.3
Depends:	R (≥ 3.1.2)
Imports:	assertthat, utils, R6, Rcpp, magrittr, lazyeval (≥ 0.1.10), DBI (≥ 0.3)
LinkingTo:	Rcpp (≥ 0.12.0), BH (≥ 1.58.0-1)
Suggests:	RSQLite (≥ 1.0.0), RMySQL, RPostgreSQL, data.table, testthat, knitr, microbenchmark, ggplot2, mgcv, Lahman (≥ 3.0-1), nycflights13, methods
Published:	2015-09-01
Author:	Hadley Wickham [aut, cre], Romain Francois [aut], RStudio [cph]
Maintainer:	Hadley Wickham <hadley at rstudio.com>
BugReports:	https://github.com/hadley/dplyr/issues
License:	MIT + file LICENSE
URL:	https://github.com/hadley/dplyr
NeedsCompilation:	yes
Materials:	README
CRAN checks:	dplyr results

Downloads:

Reference manual:	dplyr.pdf
Vignettes:	Data frames Databases Hybrid evaluation Introduction to dplyr Adding a new SQL backend Non-standard evaluation Two-table verbs Window functions and grouped mutate/filter
Windows binaries:	r-devel: dplyr_0.4.3.zip, r-release: dplyr_0.4.3.zip, r-oldrel:

RStudio Keyboard Shortcuts

Action	Windows	MacOS
Insert assignment operator: **<-**	Alt+-	Option+-
Move Lines Up/Down	Alt+Up/Down	Option+Up/Down
Run current line/selection (retain cursor position)	Alt+Enter	Option+Enter
Run the current line/selection (move to next line)	Ctrl+Enter	Cmd+Enter
Attempt Code Completion	Tab or Ctrl+Space	Tab or Cmd+Space
Jump to Matching Brace/Paren	Ctrl+P	Cmd+P
Comment/uncomment current line/selection	Ctrl+Shift+C	Cmd+Shift+C
Insert pipe operator: **%>%**	Ctrl+Shift+M	Cmd+Shift+M