Introduction to R and R Scripting

R. R. White

Department of Animal and Poultry Sciences, Virginia Tech, Blacksburg, VA

rrwhite@vt.edu

At the end of this session, you should be able to:

Describe the structure and use of objects in R

Read data into R, visualize that data, and perform basic transformations

Describe how to use packages in R

What is R?

An integrated suite of software facilities for:

Data Handling

$$K_{11} = k_{11}^{(1)} = \frac{9}{16}$$

$$K_{12} = k_{12}^{(1)} = \frac{3\sqrt{3}}{16}$$

$$K_{33} = k_{33}^{(1)} + k_{11}^{(2)} = \frac{9}{16} + \frac{5}{2} = 3.0625$$

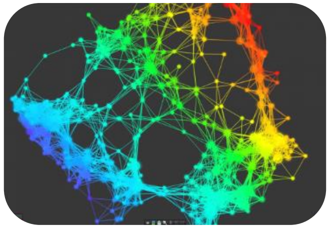
$$K_{34} = k_{34}^{(1)} + k_{12}^{(2)} = \frac{3\sqrt{3}}{16} + \left(-\frac{5}{2}\right) = -2.175$$

$$K_{43} = K_{34} = k_{43}^{(1)} + k_{21}^{(2)} = k_{34}^{(1)} + k_{12}^{(2)} = -2.175$$

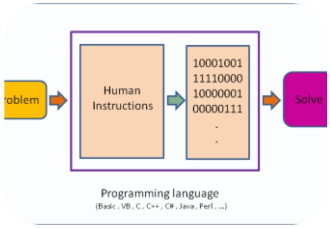
$$K_{44} = k_{44}^{(1)} + k_{22}^{(2)} = \frac{3}{16} + \frac{5}{2} = 2.6875$$

Calculation

Data Analytics



Graphical Displays



Programming Language

Some suggestions for learning a programming language

Know the difference between the language and the framework – learn the language

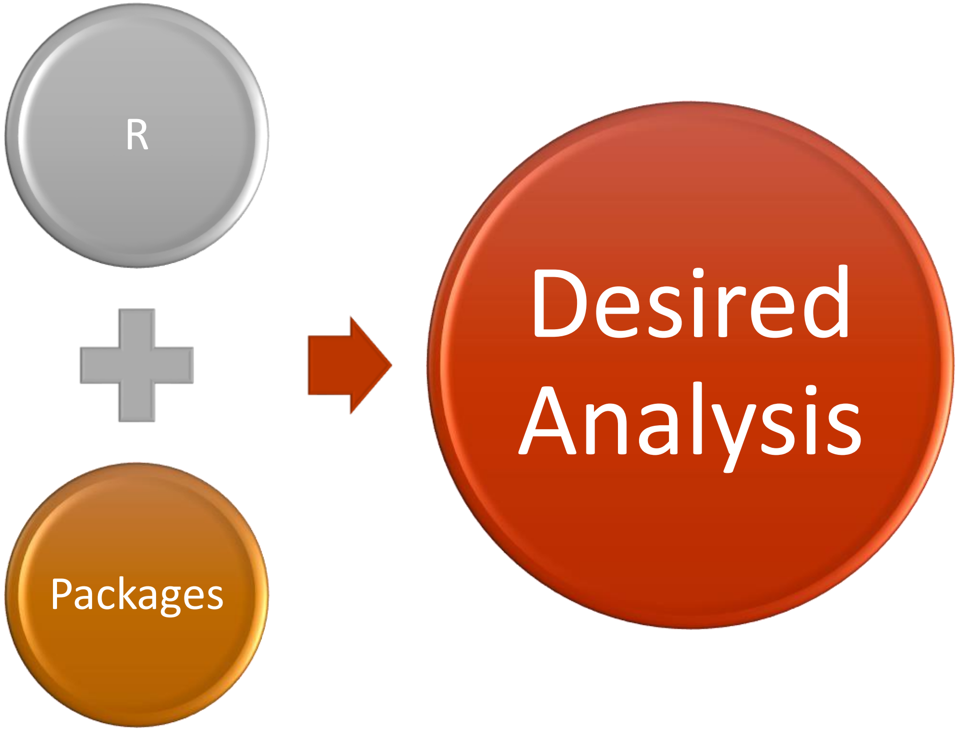
Use online resources

Websites like StackOverflow and CodeReview can be extremely helpful in learning how to solve coding challenges

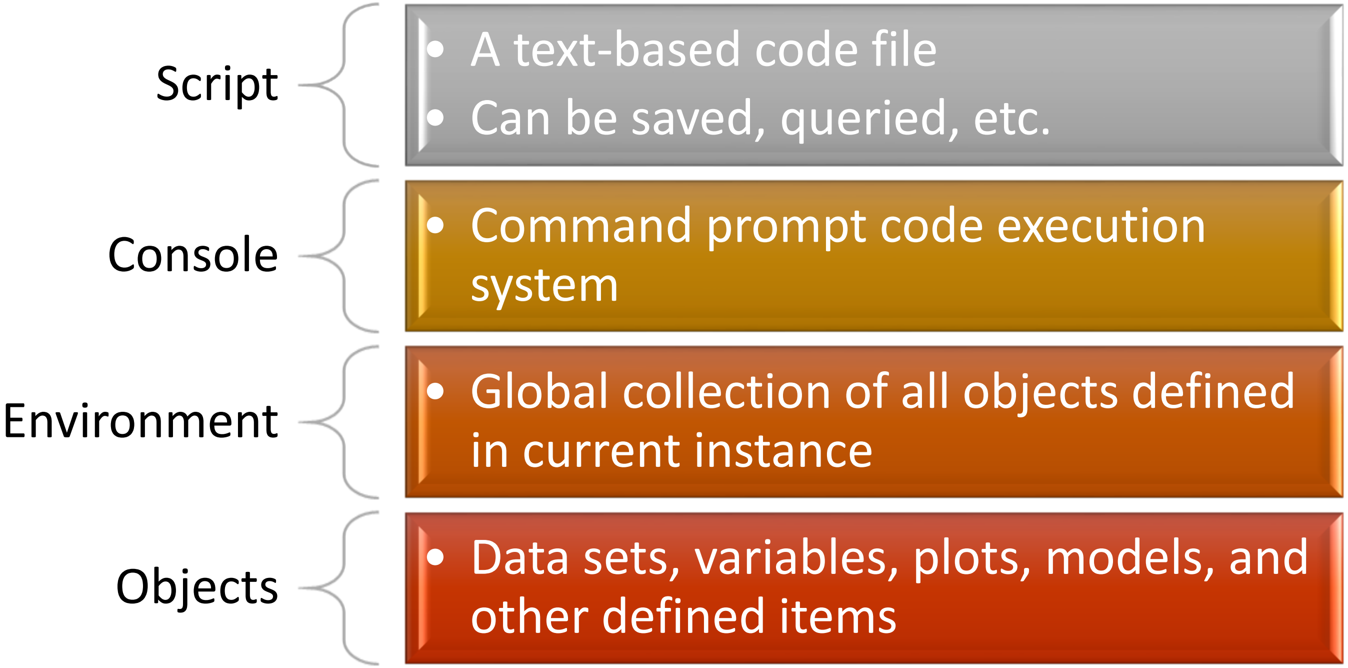
Read “Successful” Code

Use public repositories like GitHub to read successful code examples

Structure of R



Some Definitions



Script

- A text-based code file
- Can be saved, queried, etc.

Console

- Command prompt code execution system

Environment

- Global collection of all objects defined in current instance

Objects

- Data sets, variables, plots, models, and other defined items

Important Aspects of R commands

Expressions

- Operation is evaluated, printed, and the value is not retained in the environment

Assignments

- Operation is evaluated, value is passed to a variable retained in the environment, and result is not automatically printed

Important Aspects of R commands

Symbols for Code Entry

- R uses “>” to indicate it is ready to receive a new line of code
- “+” is used to show that the previous line was not complete

Commenting

- Comments can be placed almost anywhere.
- Place a “#” in the code to indicate the following information is to be a comment
- Comments run until the end of the line

Types of Data Structures

Vectors

- A single entity consisting of an ordered collection of items of the same type

Matrices

- Multi-dimensional generalizations of vectors of the same type

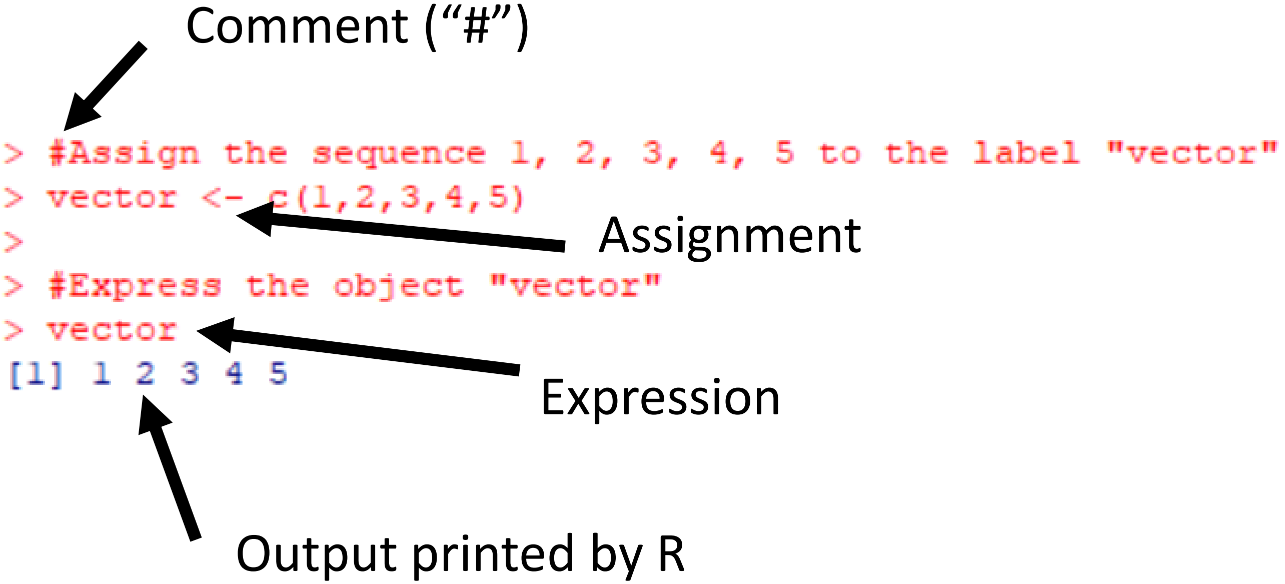
Lists

- General form of vector for which elements need not be the same type

Data Frames

- Generalized matrix structure in which columns need not be the same type

Assignment and Expression



Comment (“#”)

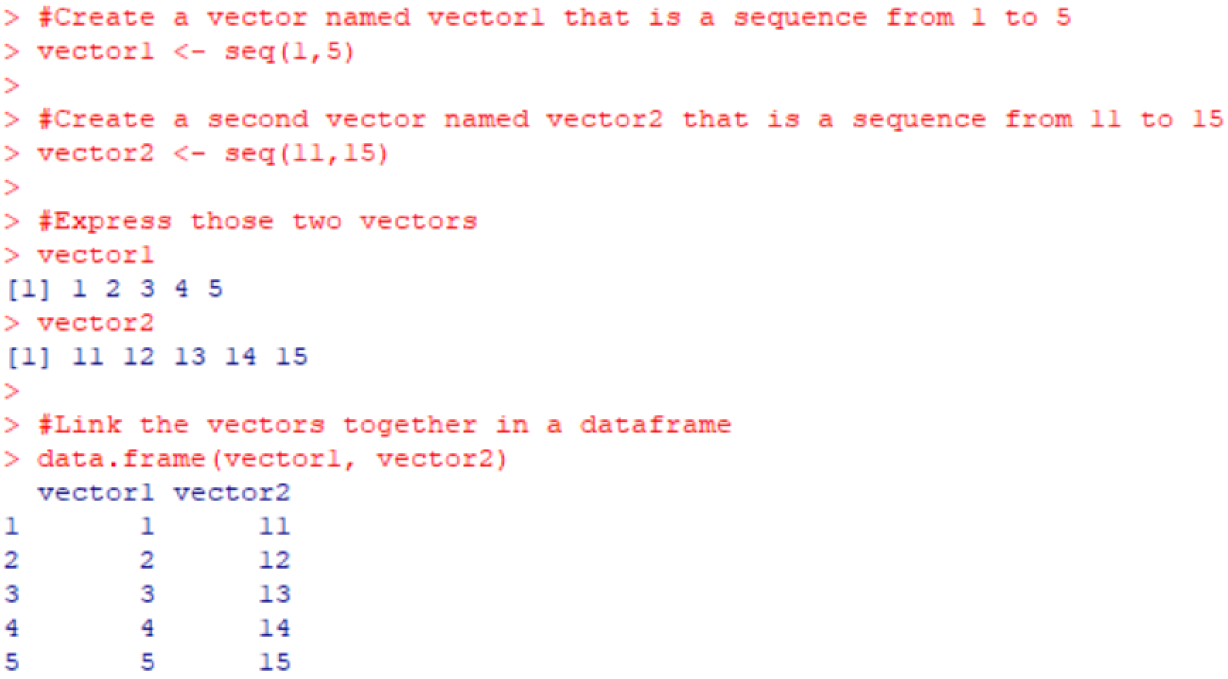
```
> #Assign the sequence 1, 2, 3, 4, 5 to the label "vector"
> vector <- c(1,2,3,4,5)
>
> #Express the object "vector"
> vector
[1] 1 2 3 4 5
```

Assignment

Expression

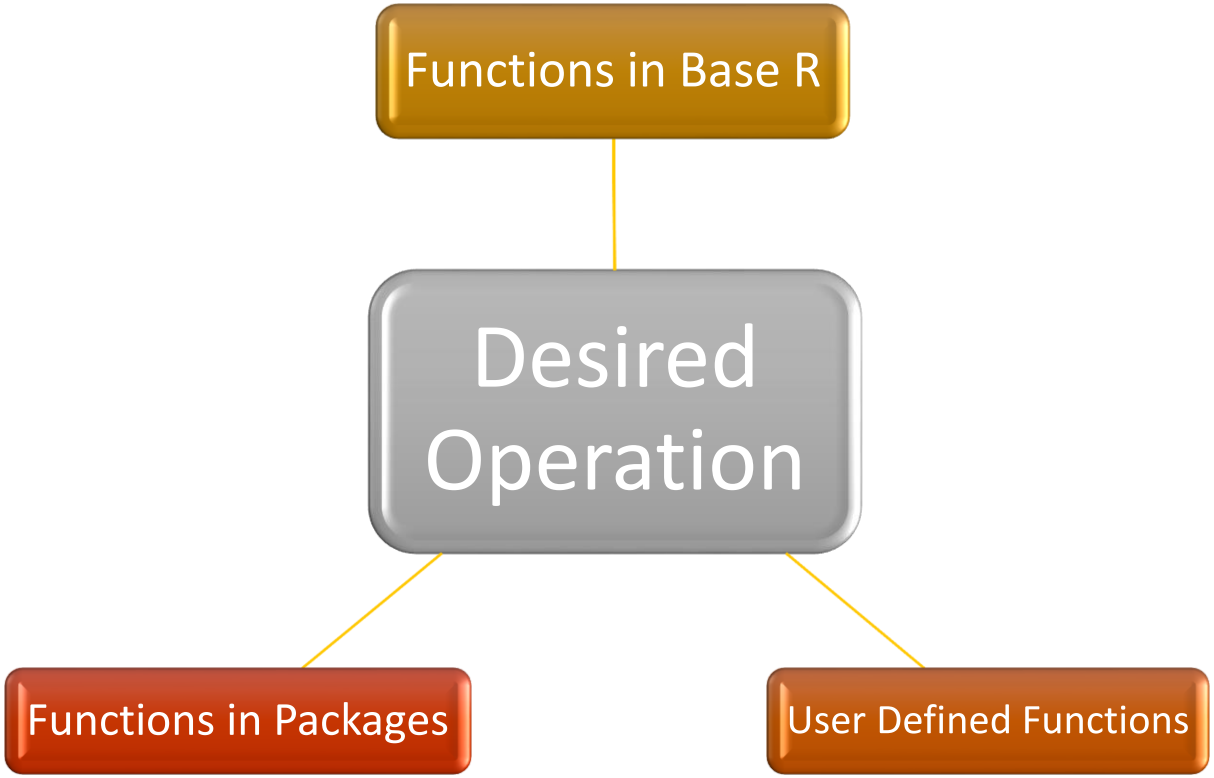
Output printed by R

Assignment and Expression



```
> #Create a vector named vector1 that is a sequence from 1 to 5
> vector1 <- seq(1,5)
>
> #Create a second vector named vector2 that is a sequence from 11 to 15
> vector2 <- seq(11,15)
>
> #Express those two vectors
> vector1
[1] 1 2 3 4 5
> vector2
[1] 11 12 13 14 15
>
> #Link the vectors together in a dataframe
> data.frame(vector1, vector2)
  vector1 vector2
1       1      11
2       2      12
3       3      13
4       4      14
5       5      15
```

Functions in R



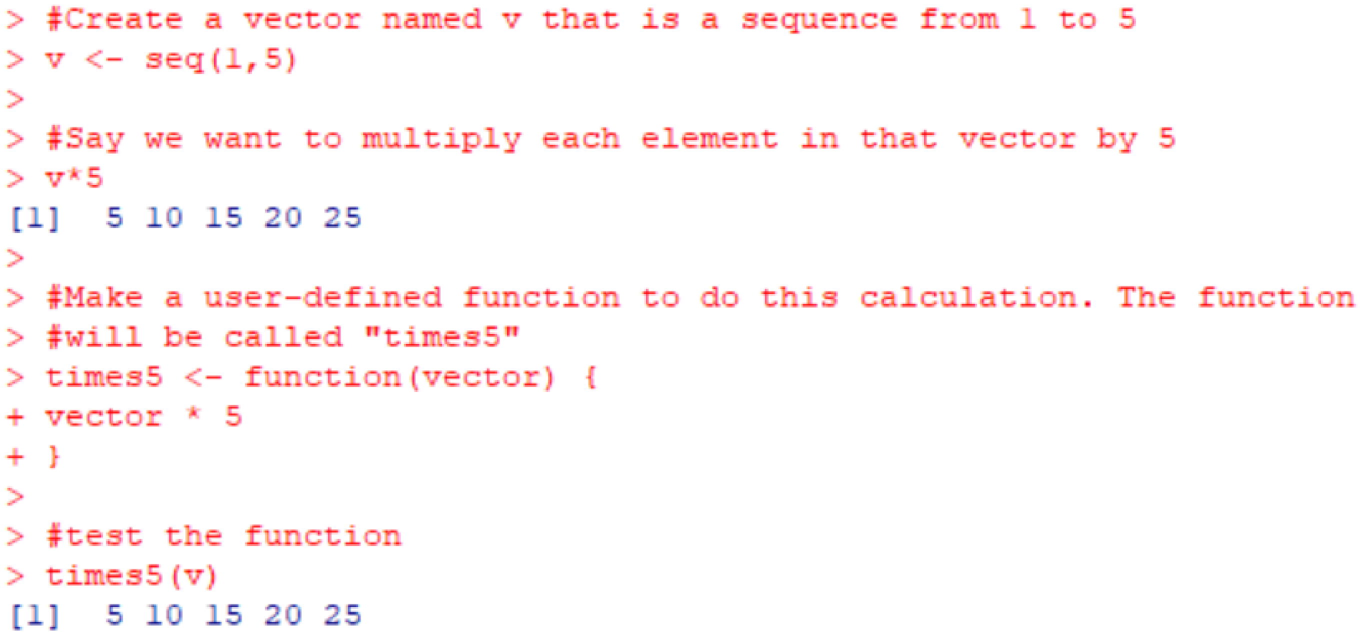
Functions in Base R

Desired Operation

Functions in Packages

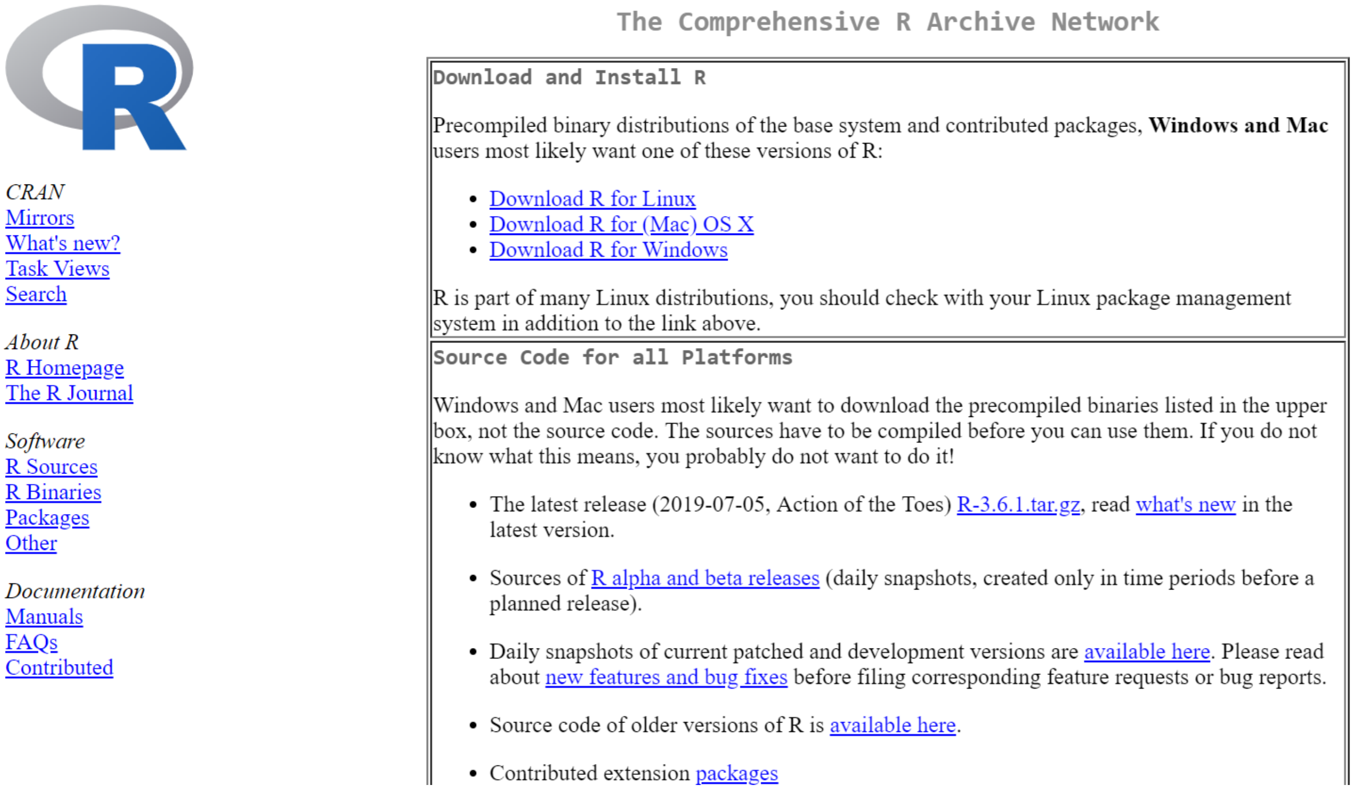
User Defined Functions

User-Defined Functions



```
> #Create a vector named v that is a sequence from 1 to 5
> v <- seq(1,5)
>
> #Say we want to multiply each element in that vector by 5
> v*5
[1]  5 10 15 20 25
>
> #Make a user-defined function to do this calculation. The function
> #will be called "times5"
> times5 <- function(vector) {
+ vector * 5
+ }
>
> #test the function
> times5(v)
[1]  5 10 15 20 25
```

Functions from Packages



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

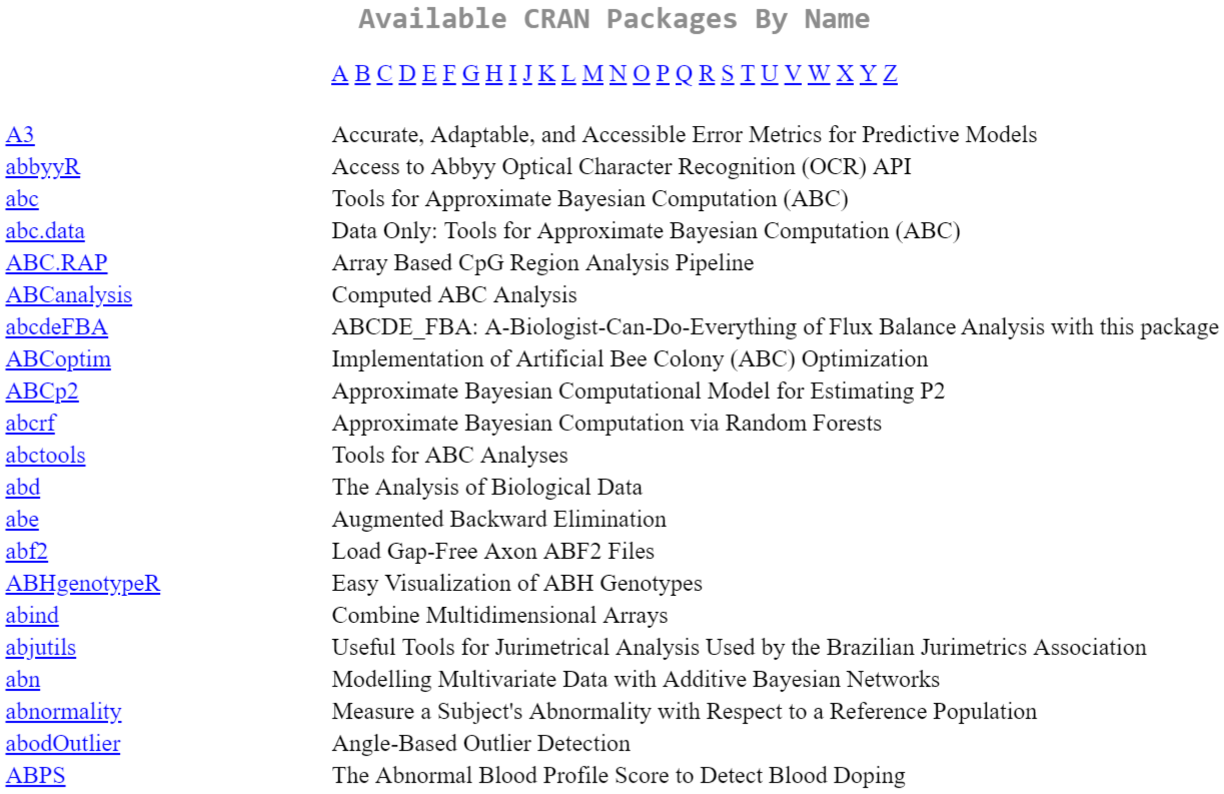
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) R-3.6.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Functions from Packages

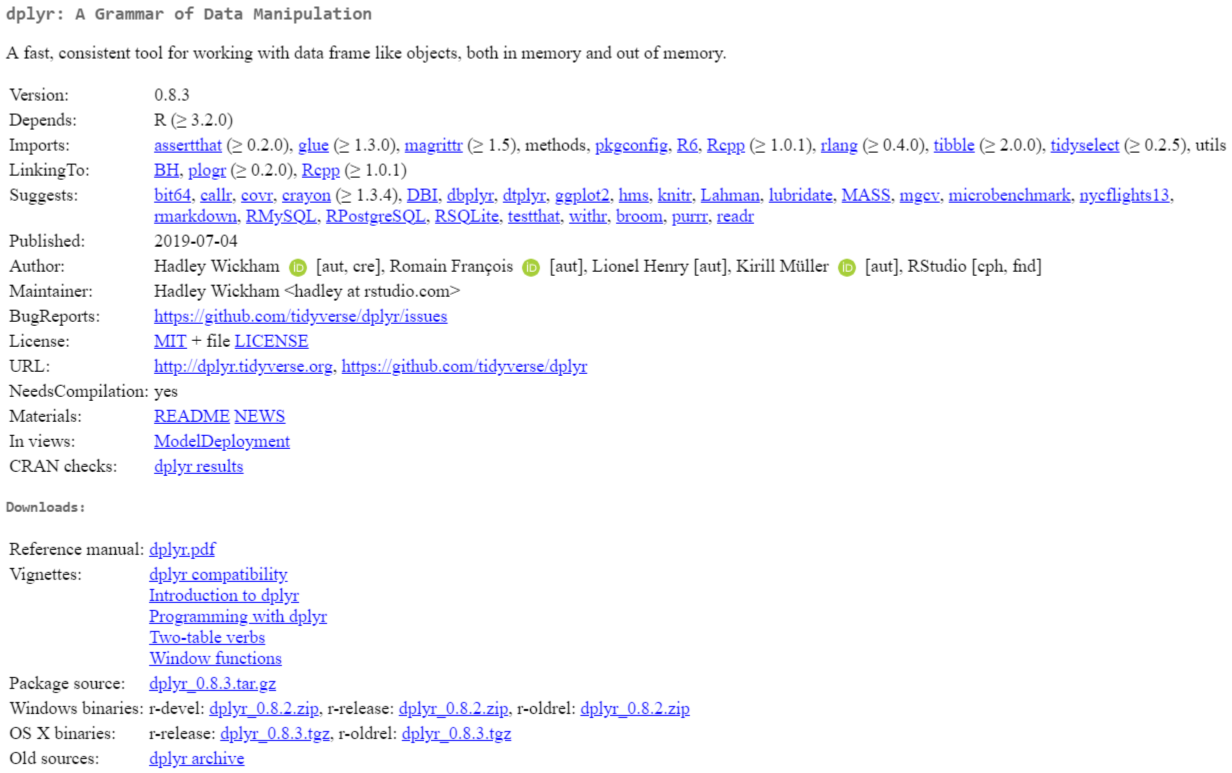


Available CRAN Packages By Name

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
ABC.RAP	Array Based CpG Region Analysis Pipeline
ABCanalysis	Computed ABC Analysis
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
abcrf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abe	Augmented Backward Elimination
abf2	Load Gap-Free Axon ABF2 Files
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjutils	Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association
abn	Modelling Multivariate Data with Additive Bayesian Networks
abnormality	Measure a Subject's Abnormality with Respect to a Reference Population
abodOutlier	Angle-Based Outlier Detection
ABPS	The Abnormal Blood Profile Score to Detect Blood Doping

Functions from Packages



dplyr: A Grammar of Data Manipulation

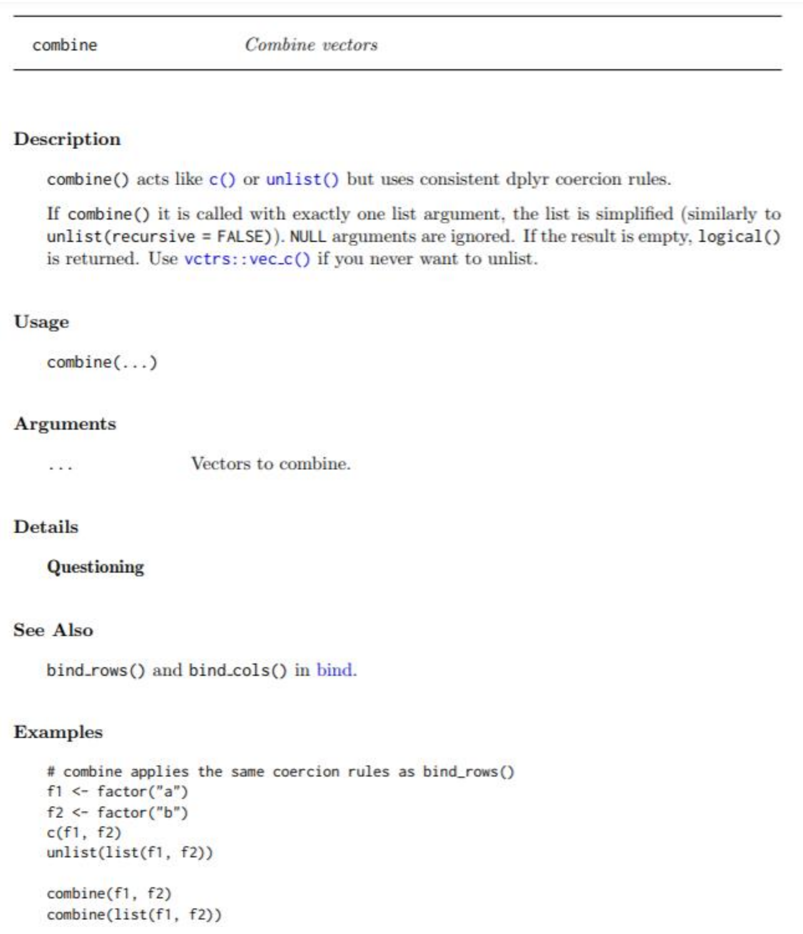
A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

Version:	0.8.3
Depends:	R (≥ 3.2.0)
Imports:	assertthat (≥ 0.2.0), glue (≥ 1.3.0), magrittr (≥ 1.5), methods, pkgconfig, R6, Rcpp (≥ 1.0.1), rlang (≥ 0.4.0), tibble (≥ 2.0.0), tidyselect (≥ 0.2.5), utils
LinkingTo:	BH, plogr (≥ 0.2.0), Rcpp (≥ 1.0.1)
Suggests:	bit64, callr, covr, crayon (≥ 1.3.4), DBI, dbplyr, dtplyr, ggplot2, hms, knitr, Lahman, lubridate, MASS, mgcv, microbenchmark, nycflights13, rmarkdown, RMySQL, RPostgreSQL, RSQLite, testthat, withr, broom, purrr, readr
Published:	2019-07-04
Author:	Hadley Wickham [aut, cre], Romain François [aut], Lionel Henry [aut], Kirill Müller [aut], RStudio [cph, fnd]
Maintainer:	Hadley Wickham <hadley at rstudio.com>
BugReports:	https://github.com/tidyverse/dplyr/issues
License:	MIT + file LICENSE
URL:	http://dplyr.tidyverse.org, https://github.com/tidyverse/dplyr
NeedsCompilation:	yes
Materials:	README NEWS
In views:	ModelDeployment
CRAN checks:	dplyr results

Downloads:

Reference manual:	dplyr.pdf
Vignettes:	dplyr compatibility Introduction to dplyr Programming with dplyr Two-table verbs Window functions
Package source:	dplyr_0.8.3.tar.gz
Windows binaries:	r-devel: dplyr_0.8.2.zip, r-release: dplyr_0.8.2.zip, r-oldrel: dplyr_0.8.2.zip
OS X binaries:	r-release: dplyr_0.8.3.tgz, r-oldrel: dplyr_0.8.3.tgz
Old sources:	dplyr archive

Functions from Packages



combine *Combine vectors*

Description

combine() acts like c() or unlist() but uses consistent dplyr coercion rules.

If combine() it is called with exactly one list argument, the list is simplified (similarly to unlist(recursive = FALSE)). NULL arguments are ignored. If the result is empty, logical() is returned. Use vctrs::vec_c() if you never want to unlist.

Usage

```
combine(...)
```

Arguments

...	Vectors to combine.

Details

Questioning

See Also

bind_rows() and bind_cols() in bind.

Examples

```
# combine applies the same coercion rules as bind_rows()
f1 <- factor("a")
f2 <- factor("b")
c(f1, f2)
unlist(list(f1, f2))

combine(f1, f2)
combine(list(f1, f2))
```

Functions from Packages

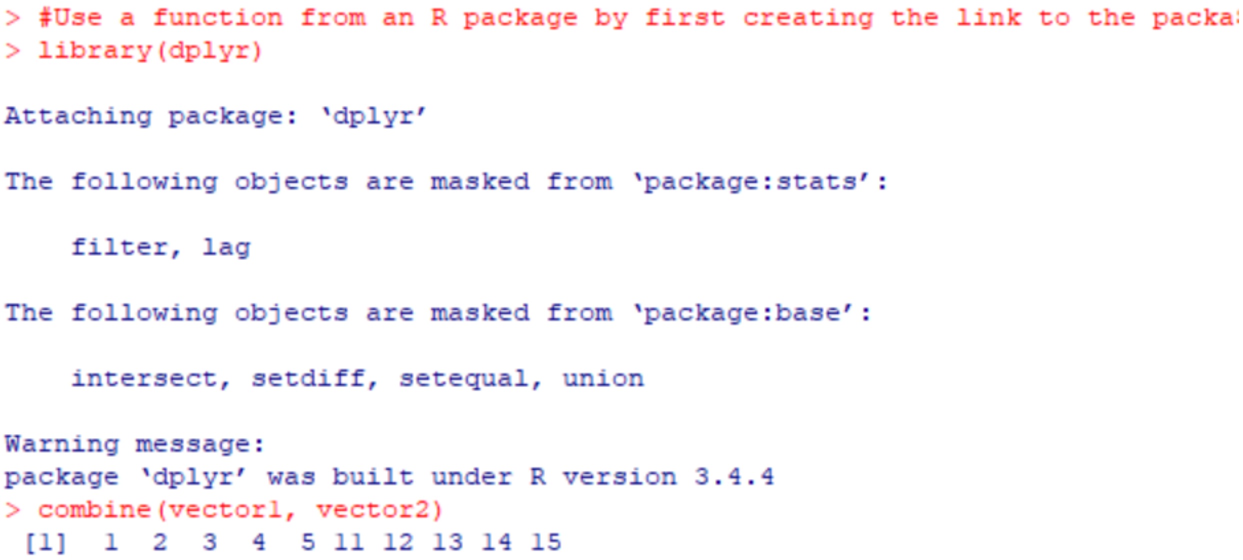
One-time package installation

- Can install from binaries, github, etc.
- Can install from program utilities
- Can install from code (install.packages(“package name”)

Opening packages (each instance of use)

- Open from code (library(package name))
    - Note – no quotes here

Functions from Packages



```
> #Use a function from an R package by first creating the link to the packa
> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 3.4.4
> combine(vector1, vector2)
 [1]  1  2  3  4  5 11 12 13 14 15
```

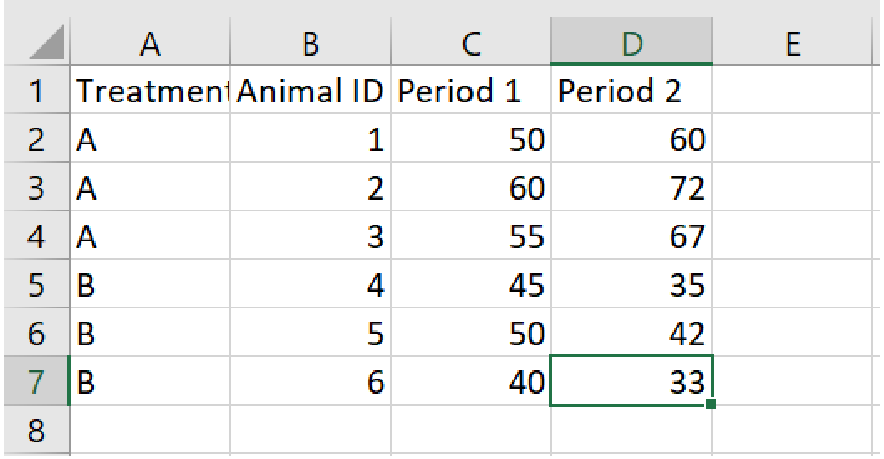
Example 1

Together, we will:

1. Read data from a csv file into R
2. Look at and summarize the data
3. Convert data between long and wide format
4. Visualize the data

Step 1:

Open Excel and Enter the Below Data

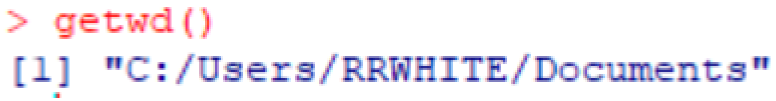


	A	B	C	D	E
1	Treatmen	Animal ID	Period 1	Period 2	
2	A	1	50	60	
3	A	2	60	72	
4	A	3	55	67	
5	B	4	45	35	
6	B	5	50	42	
7	B	6	40	33	
8					

Save the file in your Documents folder, as a CSV, using the name “ExampleData.csv”)

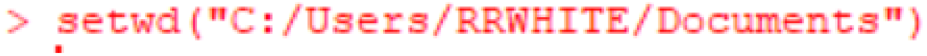
Step 2:

Open R and check your working directory:



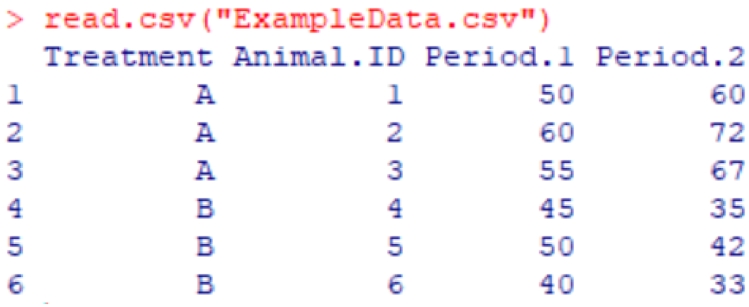
```
> getwd()
[1] "C:/Users/RRWHITE/Documents"
```

If need be, set your working directory to your documents folder:



```
> setwd("C:/Users/RRWHITE/Documents")
```

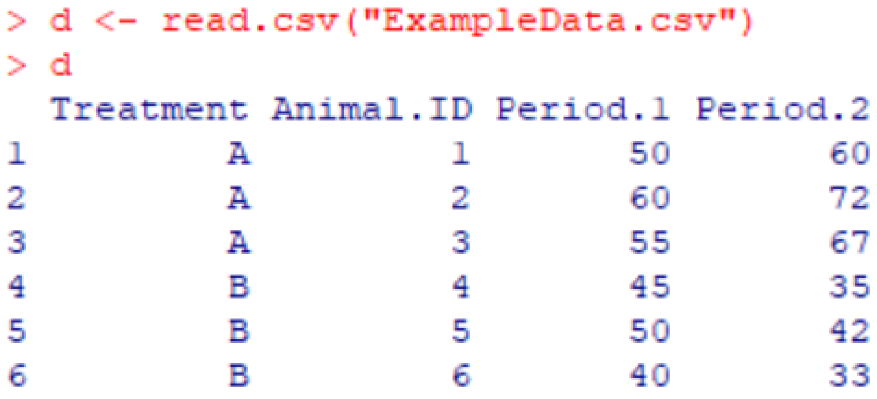
use the “read.csv” command to ExampleData



```
> read.csv("ExampleData.csv")
  Treatment Animal.ID Period.1 Period.2
1         A         1       50       60
2         A         2       60       72
3         A         3       55       67
4         B         4       45       35
5         B         5       50       42
6         B         6       40       33
```

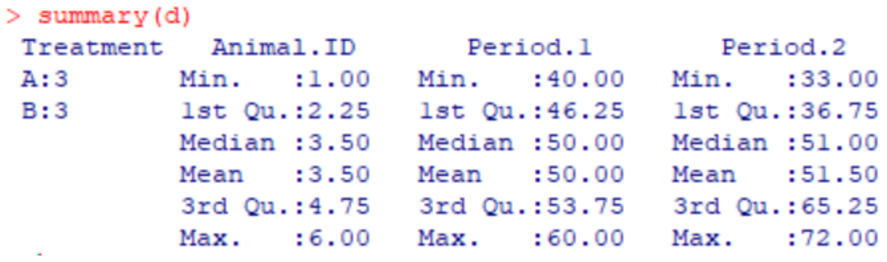
Step 3:

Assign the data the label “d”:



```
> d <- read.csv("ExampleData.csv")
> d
  Treatment Animal.ID Period.1 Period.2
1         A         1       50       60
2         A         2       60       72
3         A         3       55       67
4         B         4       45       35
5         B         5       50       42
6         B         6       40       33
```

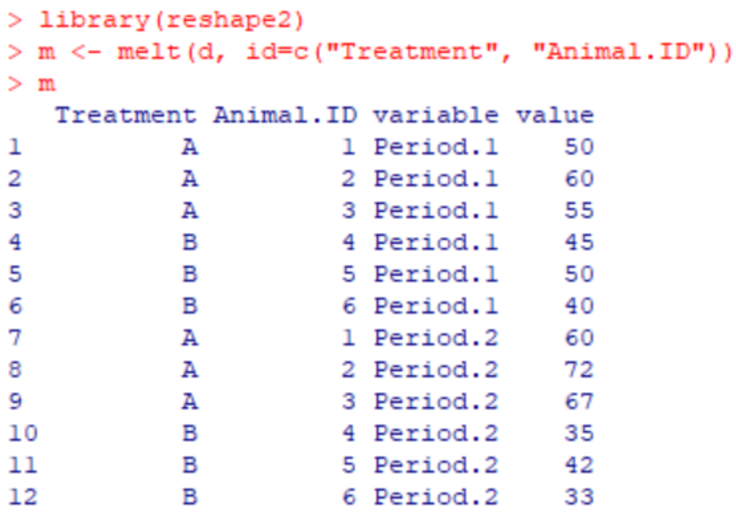
Summarize the data



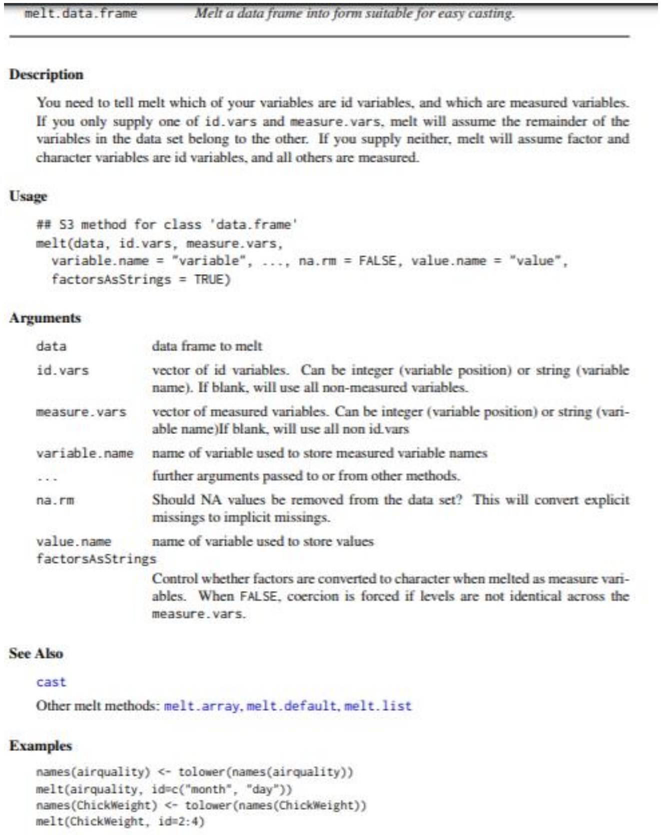
```
> summary(d)
 Treatment   Animal.ID        Period.1        Period.2
 A:3       Min.   :1.00   Min.   :40.00   Min.   :33.00
 B:3       1st Qu.:2.25   1st Qu.:46.25   1st Qu.:36.75
           Median :3.50   Median :50.00   Median :51.00
           Mean   :3.50   Mean   :50.00   Mean   :51.50
           3rd Qu.:4.75   3rd Qu.:53.75   3rd Qu.:65.25
           Max.   :6.00   Max.   :60.00   Max.   :72.00
```

Step 4:

Convert the data to long format



```
> library(reshape2)
> m <- melt(d, id=c("Treatment", "Animal.ID"))
> m
   Treatment Animal.ID variable value
1          A         1 Period.1    50
2          A         2 Period.1    60
3          A         3 Period.1    55
4          B         4 Period.1    45
5          B         5 Period.1    50
6          B         6 Period.1    40
7          A         1 Period.2    60
8          A         2 Period.2    72
9          A         3 Period.2    67
10         B         4 Period.2    35
11         B         5 Period.2    42
12         B         6 Period.2    33
```



`melt.data.frame` *Melt a data frame into form suitable for easy casting.*

Description

You need to tell melt which of your variables are id variables, and which are measured variables. If you only supply one of `id.vars` and `measure.vars`, melt will assume the remainder of the variables in the data set belong to the other. If you supply neither, melt will assume factor and character variables are id variables, and all others are measured.

Usage

```
## S3 method for class 'data.frame'
melt(data, id.vars, measure.vars,
  variable.name = "variable", ..., na.rm = FALSE, value.name = "value",
  factorsAsStrings = TRUE)
```

Arguments

`data` data frame to melt

`id.vars` vector of id variables. Can be integer (variable position) or string (variable name). If blank, will use all non-measured variables.

`measure.vars` vector of measured variables. Can be integer (variable position) or string (variable name)If blank, will use all non id.vars

`variable.name` name of variable used to store measured variable names

`...` further arguments passed to or from other methods.

`na.rm` Should NA values be removed from the data set? This will convert explicit missings to implicit missings.

`value.name` name of variable used to store values

`factorsAsStrings` Control whether factors are converted to character when melted as measure variables. When `FALSE`, coercion is forced if levels are not identical across the `measure.vars`.

See Also

`cast`

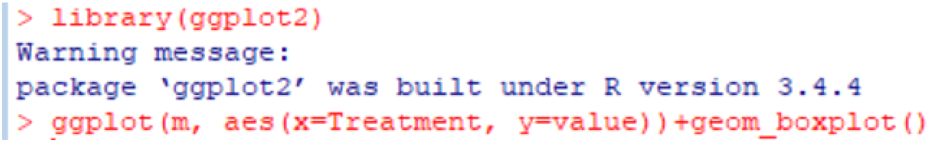
Other melt methods: `melt.array`, `melt.default`, `melt.list`

Examples

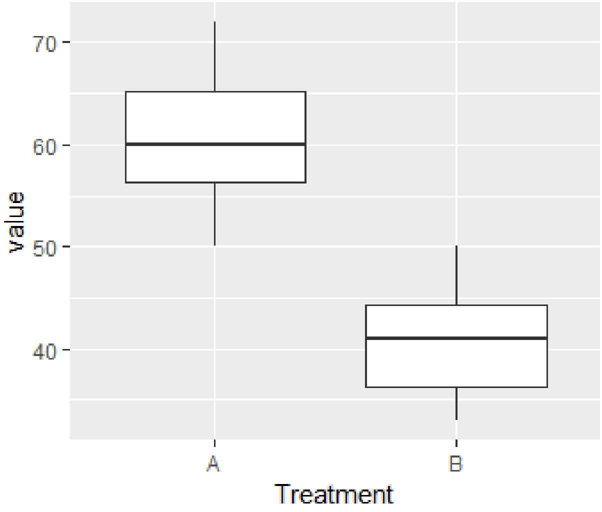
```
names(airquality) <- tolower(names(airquality))
melt(airquality, id=c("month", "day"))
names(ChickWeight) <- tolower(names(ChickWeight))
melt(ChickWeight, id=2:4)
```

Step 5:

Visualize the data



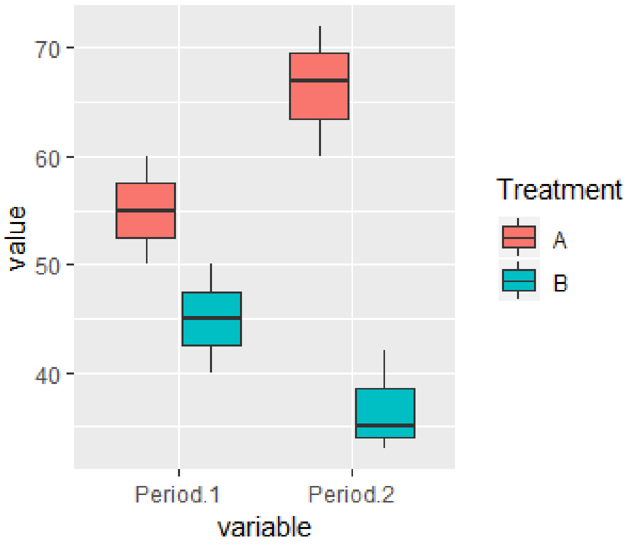
```
> library(ggplot2)
Warning message:
package ‘ggplot2’ was built under R version 3.4.4
> ggplot(m, aes(x=Treatment, y=value))+geom_boxplot()
```



Step 6:

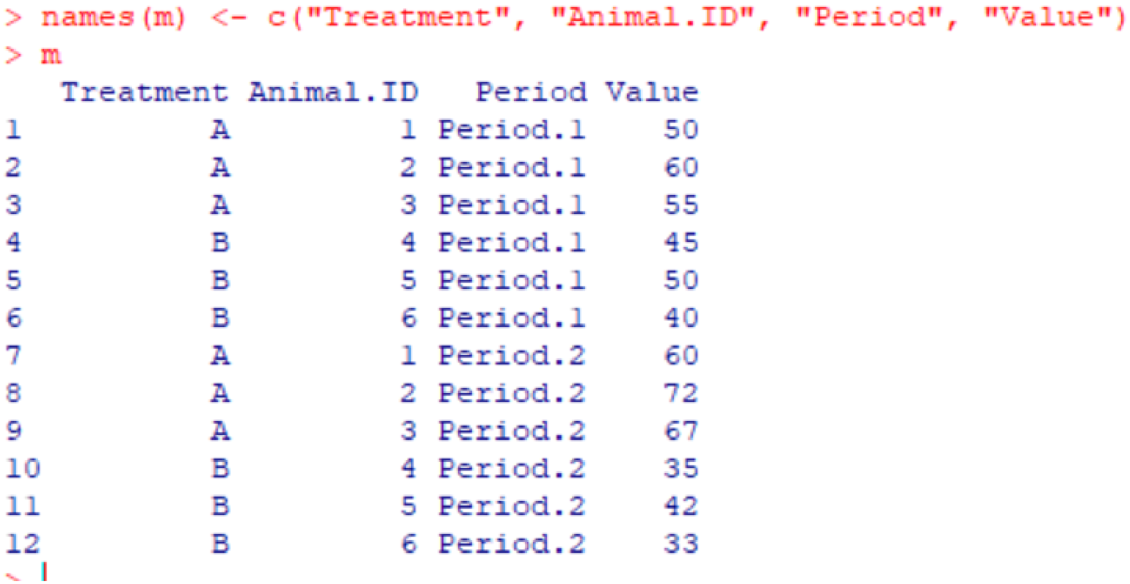
Visualize the data

```
> ggplot(m, aes(x=variable, y=value, fill=Treatment))+geom_boxplot()
```



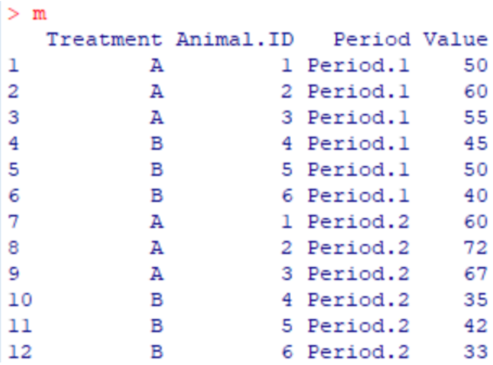
An Aside

Cleaning up Dataframes

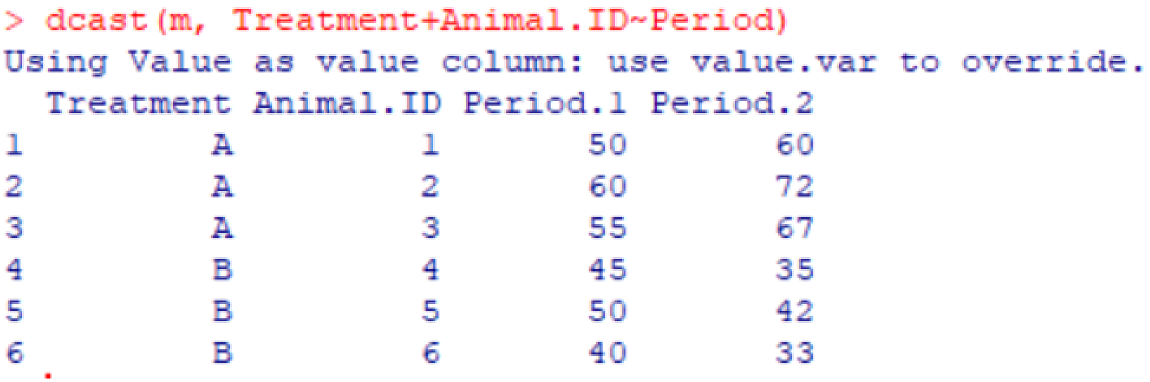


```
> names(m) <- c("Treatment", "Animal.ID", "Period", "Value")
> m
   Treatment Animal.ID   Period Value
1          A         1 Period.1    50
2          A         2 Period.1    60
3          A         3 Period.1    55
4          B         4 Period.1    45
5          B         5 Period.1    50
6          B         6 Period.1    40
7          A         1 Period.2    60
8          A         2 Period.2    72
9          A         3 Period.2    67
10         B         4 Period.2    35
11         B         5 Period.2    42
12         B         6 Period.2    33
>
```

Converting Data from Long to Wide



```
> m
   Treatment Animal.ID   Period Value
1          A         1 Period.1    50
2          A         2 Period.1    60
3          A         3 Period.1    55
4          B         4 Period.1    45
5          B         5 Period.1    50
6          B         6 Period.1    40
7          A         1 Period.2    60
8          A         2 Period.2    72
9          A         3 Period.2    67
10         B         4 Period.2    35
11         B         5 Period.2    42
12         B         6 Period.2    33
```



```
> dcast(m, Treatment+Animal.ID~Period)
Using Value as value column: use value.var to override.
  Treatment Animal.ID Period.1 Period.2
1         A         1       50       60
2         A         2       60       72
3         A         3       55       67
4         B         4       45       35
5         B         5       50       42
6         B         6       40       33
```

Example 2

Together, we will:

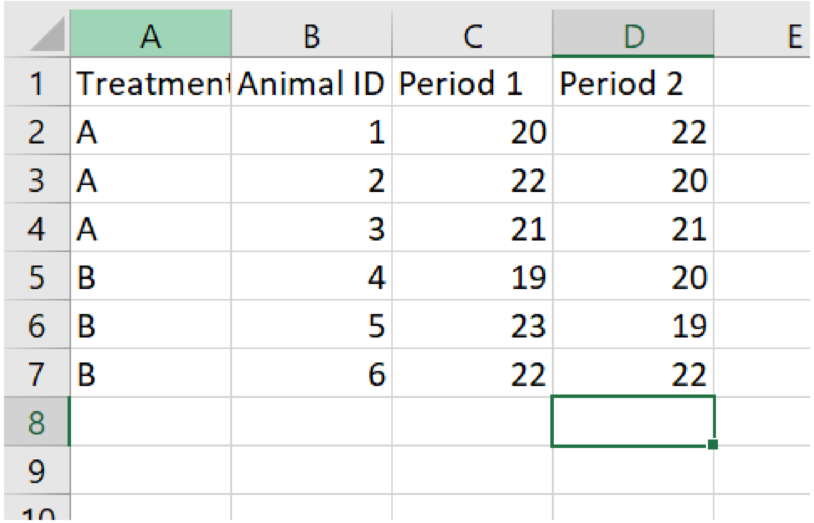
1. Read data from a csv file into R
2. Merge two dataframes
3. Perform calculations on data
4. Visualize the data

Workshop data example 2

Together, we will:

1. Read data from a csv file into R
2. Merge two dataframes
3. Perform calculations on data
4. Visualize the data

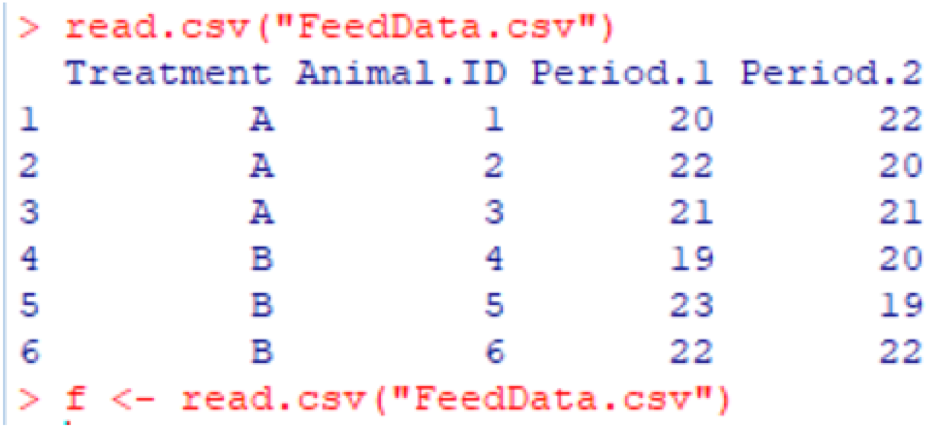
Make a new CSV file



	A	B	C	D	E
1	Treatmen	Animal ID	Period 1	Period 2	
2	A	1	20	22	
3	A	2	22	20	
4	A	3	21	21	
5	B	4	19	20	
6	B	5	23	19	
7	B	6	22	22	
8					
9					

Save the file as “FeedData.csv” in your working directory folder (My Documents)

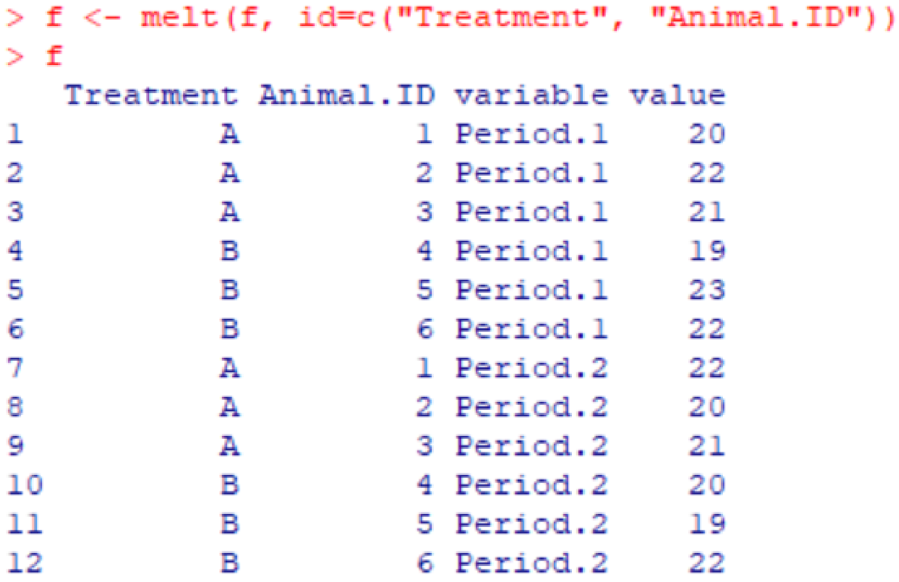
Read the CSV into R



```
> read.csv("FeedData.csv")
  Treatment Animal.ID Period.1 Period.2
1         A         1       20       22
2         A         2       22       20
3         A         3       21       21
4         B         4       19       20
5         B         5       23       19
6         B         6       22       22
> f <- read.csv("FeedData.csv")
```

Call the data-frame “f”

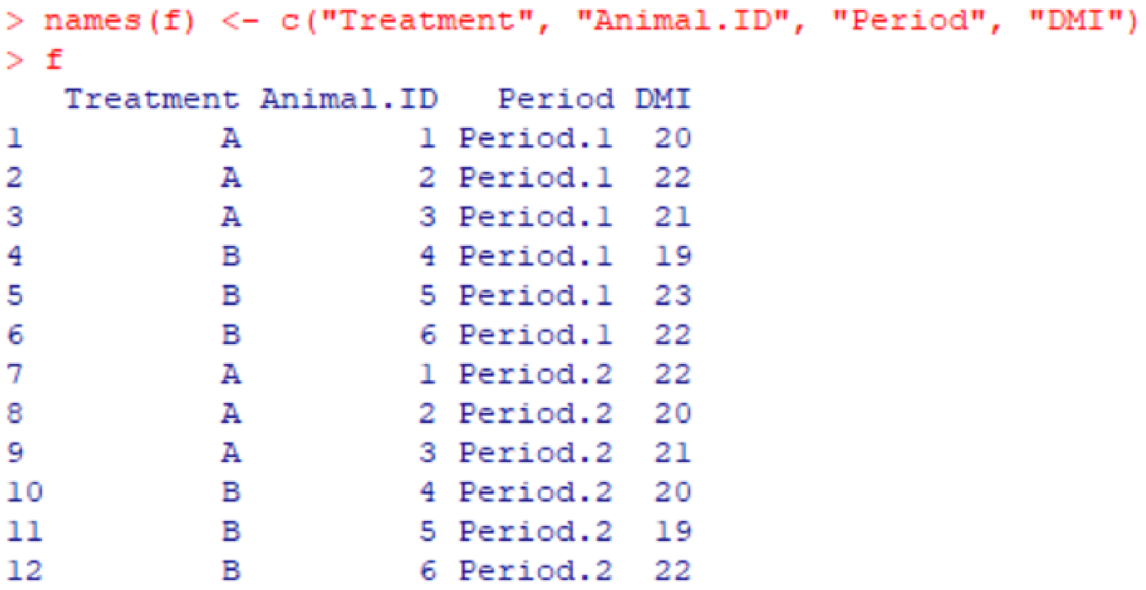
Convert from wide to long format



```
> f <- melt(f, id=c("Treatment", "Animal.ID"))
> f
   Treatment Animal.ID variable value
1          A         1 Period.1    20
2          A         2 Period.1    22
3          A         3 Period.1    21
4          B         4 Period.1    19
5          B         5 Period.1    23
6          B         6 Period.1    22
7          A         1 Period.2    22
8          A         2 Period.2    20
9          A         3 Period.2    21
10         B         4 Period.2    20
11         B         5 Period.2    19
12         B         6 Period.2    22
```

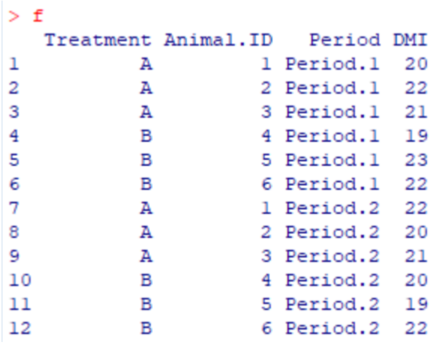
Call the data-frame “f”

Rename the columns of the data frame

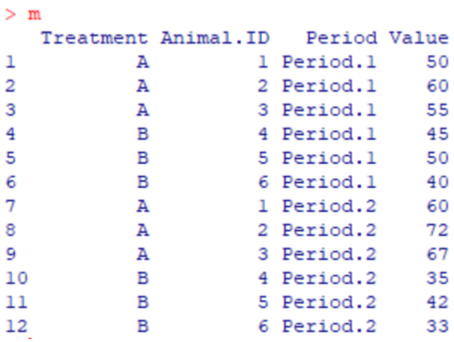


```
> names(f) <- c("Treatment", "Animal.ID", "Period", "DMI")
> f
   Treatment Animal.ID   Period DMI
1          A         1 Period.1  20
2          A         2 Period.1  22
3          A         3 Period.1  21
4          B         4 Period.1  19
5          B         5 Period.1  23
6          B         6 Period.1  22
7          A         1 Period.2  22
8          A         2 Period.2  20
9          A         3 Period.2  21
10         B         4 Period.2  20
11         B         5 Period.2  19
12         B         6 Period.2  22
```

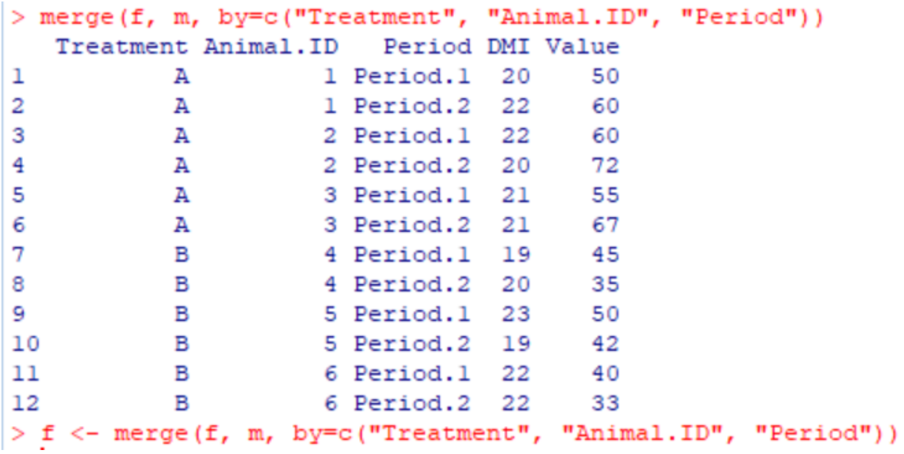
Merge the f and the m dataframes



```
> f
   Treatment Animal.ID   Period DMI
1          A         1 Period.1  20
2          A         2 Period.1  22
3          A         3 Period.1  21
4          B         4 Period.1  19
5          B         5 Period.1  23
6          B         6 Period.1  22
7          A         1 Period.2  22
8          A         2 Period.2  20
9          A         3 Period.2  21
10         B         4 Period.2  20
11         B         5 Period.2  19
12         B         6 Period.2  22
```

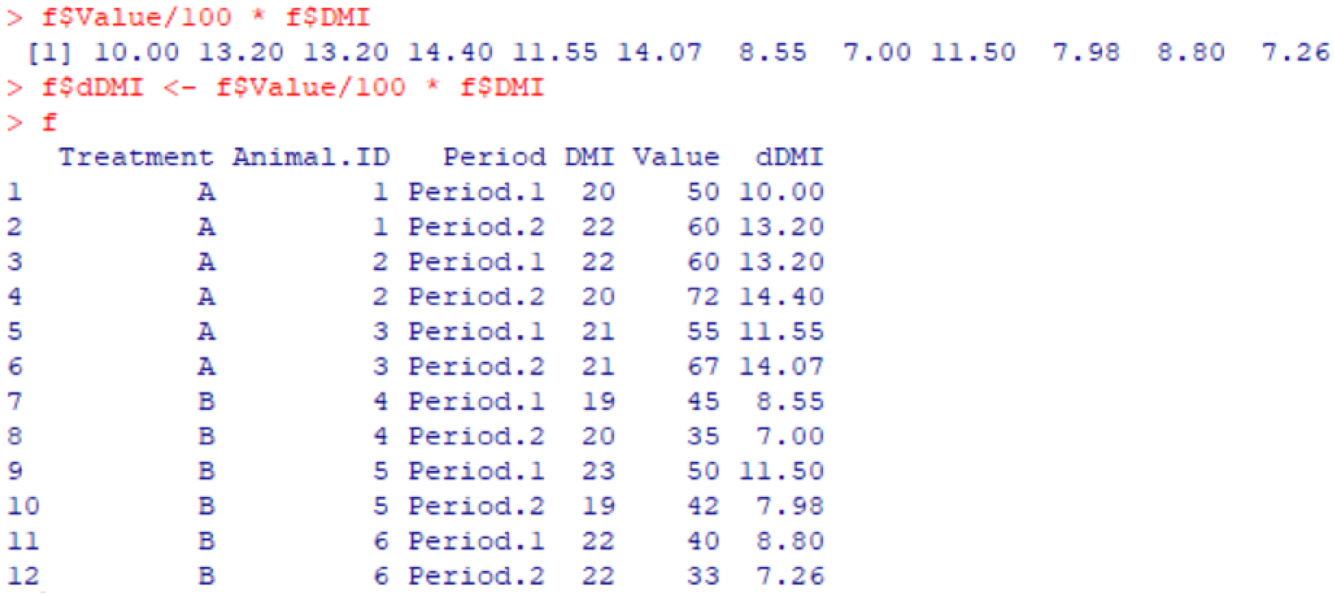


```
> m
   Treatment Animal.ID   Period Value
1          A         1 Period.1    50
2          A         2 Period.1    60
3          A         3 Period.1    55
4          B         4 Period.1    45
5          B         5 Period.1    50
6          B         6 Period.1    40
7          A         1 Period.2    60
8          A         2 Period.2    72
9          A         3 Period.2    67
10         B         4 Period.2    35
11         B         5 Period.2    42
12         B         6 Period.2    33
```



```
> merge(f, m, by=c("Treatment", "Animal.ID", "Period"))
   Treatment Animal.ID   Period DMI Value
1          A         1 Period.1  20    50
2          A         1 Period.2  22    60
3          A         2 Period.1  22    60
4          A         2 Period.2  20    72
5          A         3 Period.1  21    55
6          A         3 Period.2  21    67
7          B         4 Period.1  19    45
8          B         4 Period.2  20    35
9          B         5 Period.1  23    50
10         B         5 Period.2  19    42
11         B         6 Period.1  22    40
12         B         6 Period.2  22    33
> f <- merge(f, m, by=c("Treatment", "Animal.ID", "Period"))
```

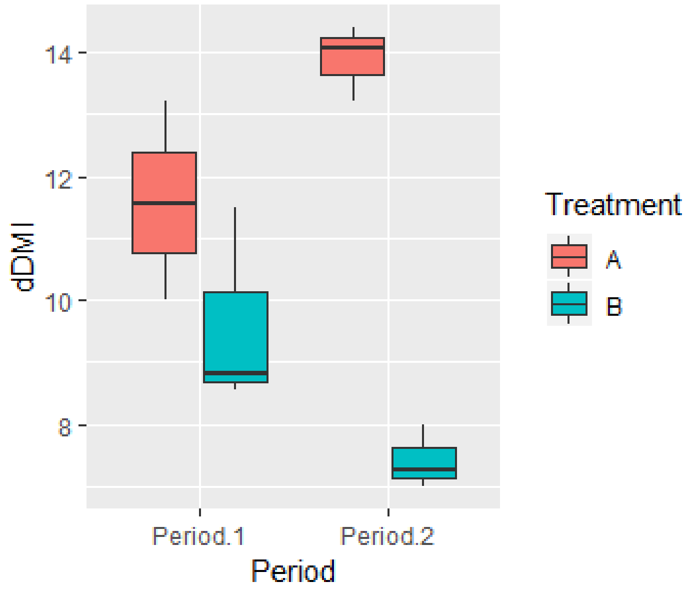
Perform some calculations



```
> f$Value/100 * f$DMI
 [1] 10.00 13.20 13.20 14.40 11.55 14.07  8.55  7.00 11.50  7.98  8.80  7.26
> f$dDMI <- f$Value/100 * f$DMI
> f
   Treatment Animal.ID   Period DMI Value  dDMI
1          A         1 Period.1  20    50 10.00
2          A         1 Period.2  22    60 13.20
3          A         2 Period.1  22    60 13.20
4          A         2 Period.2  20    72 14.40
5          A         3 Period.1  21    55 11.55
6          A         3 Period.2  21    67 14.07
7          B         4 Period.1  19    45  8.55
8          B         4 Period.2  20    35  7.00
9          B         5 Period.1  23    50 11.50
10         B         5 Period.2  19    42  7.98
11         B         6 Period.1  22    40  8.80
12         B         6 Period.2  22    33  7.26
```

Visualize the dDMI data

```
> ggplot(f, aes(x=Period, y=dDMI, fill=Treatment))+geom_boxplot()
```



An Aside

- X labels: +xlab(“label”)
- Y labels: +ylab(“label”)
- Preset themes (e.g., +theme_minimal())
- Other types
  - Geom_density()
  - Geom_point()
  - Geom_line()

Conditional Statements

Single Instance "if" statements

- Typically depends on single variable value (if *change* = "yes" then …)
- Can apply transformation across number of variables/vectors

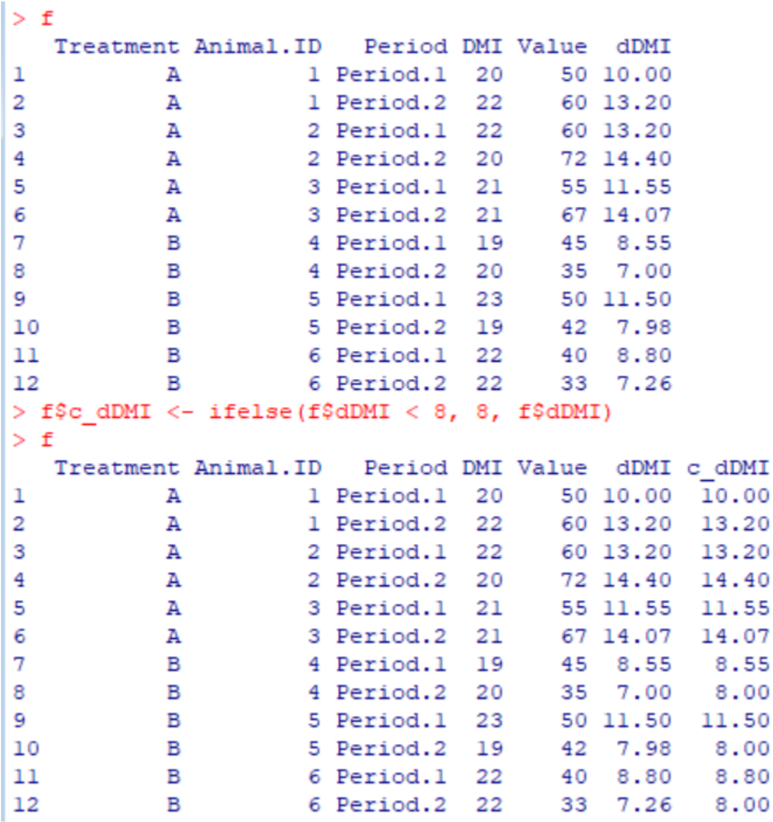
Vectorized "if" statements

- Executed for each element of a vector (if *element[i]* > 2 then …)
- Typically applies to corresponding element of the same vector or a different vector

Rules for conditionals

Conditional	Symbol
Is greater than	>
Is less than	<
Is equal to	==
Is within	%in%
Is not equal to	!=

Example Vectorized Conditional



```
> f
   Treatment Animal.ID   Period DMI Value  dDMI
1          A         1 Period.1  20    50 10.00
2          A         1 Period.2  22    60 13.20
3          A         2 Period.1  22    60 13.20
4          A         2 Period.2  20    72 14.40
5          A         3 Period.1  21    55 11.55
6          A         3 Period.2  21    67 14.07
7          B         4 Period.1  19    45  8.55
8          B         4 Period.2  20    35  7.00
9          B         5 Period.1  23    50 11.50
10         B         5 Period.2  19    42  7.98
11         B         6 Period.1  22    40  8.80
12         B         6 Period.2  22    33  7.26
> f$c_dDMI <- ifelse(f$dDMI < 8, 8, f$dDMI)
> f
   Treatment Animal.ID   Period DMI Value  dDMI c_dDMI
1          A         1 Period.1  20    50 10.00  10.00
2          A         1 Period.2  22    60 13.20  13.20
3          A         2 Period.1  22    60 13.20  13.20
4          A         2 Period.2  20    72 14.40  14.40
5          A         3 Period.1  21    55 11.55  11.55
6          A         3 Period.2  21    67 14.07  14.07
7          B         4 Period.1  19    45  8.55   8.55
8          B         4 Period.2  20    35  7.00   8.00
9          B         5 Period.1  23    50 11.50  11.50
10         B         5 Period.2  19    42  7.98   8.00
11         B         6 Period.1  22    40  8.80   8.80
12         B         6 Period.2  22    33  7.26   8.00
```

Additional things you want to learn?

Email: *rrwhite@vt.edu*

Office: *540-231-7384*

Cell: *509-701-9290*

VirginiaTech

Invent the Future®