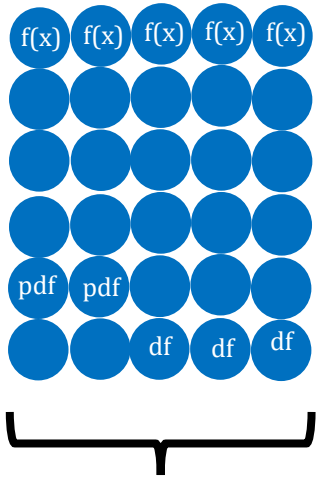
Workshop 2:
The tidyverse and beyond

Brendan Palmer,
Statistics & Data Analysis Unit,
Clinical Research Facility - Cork

PART A: We built this software on base R code

Recall:



Base R: Comes pre-loaded

Functions

```
> base::
```

```
max.col                {base}
mean                   {base}
mean.Date              {base}
mean.default           {base}
mean.difftime          {base}
mean.POSIXct           {base}
mean.POSIXlt           {base}
mem.limits             {base}
```

```
mean(x, ...)
Generic function for the (trimmed) arithmetic mean.
Press F1 for additional help
```

Data sets

```
> data()
```

```
faithful
freeny
infert
iris
iris3
islands
lh
longley
```

```
iris
This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.
Press F1 for additional help
```

Conflicts

```
> filter
```

```
filter     {dplyr}
filter_    {dplyr}
filter_all {dplyr}
filter_at  {dplyr}
filter_if  {dplyr}
Filter     {base}
Filters
```

```
filter(x, filter, method = c("convolution", "recursive"),
    sides = 2L, circular = FALSE, init = NULL)
Applies linear filtering to a univariate time series or to each series separately of a multivariate time series.
Press F1 for additional help
```

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)
Return x sorted.
rev(x)
Return x reversed.
table(x)
See counts of values.
unique(x)
See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/ write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Creating objects

For most of us, R is simply the creation of and manipulation of objects

```
new_object <- c(1, 2, 3)
```

- the objects are then fed into functions to create amazing new objects

```
amazing_new_object <- function(new_object)
```

Broadly speaking the following is true in R:

- information
- dataframe <- function(information)
- plot   <- function(dataframe)
- model   <- function(dataframe)

Data structures

Data structures encompass everything from our new object create in the last slide and beyond

Data can be stored in different forms

```
#double (for double precision floating point numbers)
typeof(1)

#character
typeof("string")

#logical
typeof(FALSE)

#missing values are represented by NA
example <- c(1, 2, NA, 4)
```

Types of data structures I

```
#vectors:

These come in two forms
- A: Atomic vectors contain exactly one type of data

all_numbers          <- c(1, 2, 0.5, -0.5, 3.4)

all_characters       <- c("One", "too", "3")

all_logical          <- c(TRUE, FALSE) #NOTE: Always type it out

- B: Lists allow combinations of different types of data

this_is_a_list       <- list(1, TRUE, "Three")

 typeof(this_is_a_list)
[1] "list"

this_is_also_a_list <- list(all_numbers, all_characters, all_logical)
```

Worksheet 2
Part A

Types of data structures II

```
#Matrices/Arrays:

- You can have a matrix of two or more dimensions
a_matrix <- matrix(1:9, 3, 3)

- Both vectors and matrices can only contain one type of data

- If you try to create a vector with more than one data type, then it will
  undergo coercion to the least common denominator

- The coercion rule goes:
                    logical -> integer -> numeric -> complex -> character

- You can perform coercions yourself on vectors
nums_as_characters <- as.character(all_numbers)

back_to_numbers    <- as.numeric(nums_as_characters)
```

Types of data structures III

#Dataframes:

- These are a special type of list
- Observations are in rows
- Variables are in columns
- Labels or other metadata may also be present

```
- a_data_frame <- data.frame(number = 1:10,
                             char   = sample(letters, 10),
                              this_really_a_col_name  = rep(c(TRUE, FALSE), 5))
```

- In the tidyverse dataframes are called “tibbles”
    - Tibbles are one of the unifying features of the tidyverse
    - The two main differences are with;
        - printing: tibbles have a defined print method
        - subsetting: use of the [function

- Some older functions don’t work with tibbles. To convert just type:

```
old_data_frame <- as.data.frame(tibble.data.frame)
```

Worksheet 2
Part B

Indexing

- Indexing can occur in one or two dimensions

- One dimension:

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

- Two dimensions

```
a_data_frame[1, 1]

a_data_frame$number[1]
```

- In the tidyverse we don't use `[` much as `dplyr::filter()` and `dplyr::select()` allow you to solve the same problems

- However, given so much of the R has been written using these, it's worth recognising and understanding them

Worksheet 2

Part C

Go to script1_baseR_indexing_and_functions

Types of data structures IV

#Factors:

- In R, factors are used to work with categorical variables

- historically they were easier to work with than characters, hence many baseR functions automatically convert characters to factors

- This does not happen in the tidyverse

- forcats packages deals with them

- One of the most important uses of factors is in statistical modeling; since categorical variables enter into statistical models differently than continuous variables, storing data as factors insures that the modeling functions will treat such data correctly

- For a brief factor example open and run script2_factors

PART B: The layered grammar of graphics

Template:

```
ggplot(data = <DATA>) +

      <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))    +

       linear model  +

       axes formatting  +

       legend formatting  +

       title   + etc. etc.
```

Worksheet 2

Part D

Go to script3_house_registrations.R