

4. Vectors and intro to the Tidyverse

Frank Edwards

R works with objects

```
a<-2  
a<-a+1
```

R works with objects

```
a
```

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

Objects can take many types

```
a<-2  
class(a)
```

```
## [1] "numeric"
```

Objects can take many types

```
b<-"howdy"  
class(b)
```

```
## [1] "character"
```

Objects can take many types

```
c<-TRUE  
class(c)
```

```
## [1] "logical"
```

Vectors are one-dimensional arrays of values of any class

```
vector1<-c(2,3,4,5,6)  
vector1
```

```
## [1] 2 3 4 5 6
```

Vectors are one-dimensional arrays of values of any class

```
vector2<-c("a", "fancy", "vector")  
vector2
```

```
## [1] "a"      "fancy"  "vector"
```


Vectors

```
vector3<-c(TRUE, FALSE, TRUE, FALSE, FALSE)  
vector3
```

```
## [1] TRUE FALSE TRUE FALSE FALSE
```

Vectorized operations

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
2 * vector1
```

```
## [1] 4 6 8 10 12
```

Vectorized operations

```
vector3
```

```
## [1] TRUE FALSE TRUE FALSE FALSE
```

```
vector3==FALSE
```

```
## [1] FALSE TRUE FALSE TRUE TRUE
```

Vector indexing

```
vector2
```

```
## [1] "a"      "fancy"  "vector"
```

Vector indexing

```
vector2
```

```
## [1] "a"      "fancy" "vector"
```

```
vector2[1]
```

```
## [1] "a"
```

Vector indexing

```
vector2
```

```
## [1] "a"      "fancy" "vector"
```

```
vector2[2]
```

```
## [1] "fancy"
```

Vector indexing

```
vector2
```

```
## [1] "a"      "fancy" "vector"
```

```
vector2[3]
```

```
## [1] "vector"
```

Operations and vector indexing

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
vector1[2] + 3
```

```
## [1] 6
```


Functions!

R has loads and loads of functions.

- Functions run a fixed set of operations on some argument(s)
- Functions return a value that can be assigned to an object
- Functions take the general form *function(arguments)*

Some common function in R for vectors

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
max(vector1)
```

```
## [1] 6
```

```
min(vector1)
```

```
## [1] 2
```

Some common function in R for vectors

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
length(vector1)
```

```
## [1] 5
```

Some common function in R for vectors

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
mean(vector1)
```

```
## [1] 4
```

Some common function in R for vectors

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
sum(vector1)
```

```
## [1] 20
```

Functions can work together

```
sum(vector1)
```

```
## [1] 20
```

```
length(vector1)
```

```
## [1] 5
```

```
# this is the same as mean()  
sum(vector1)/length(vector1)
```

```
## [1] 4
```

We can even write our own!

```
redundantMean<-function(x){  
  n<-length(x)  
  sum_x<-sum(x)  
  xbar<-sum_x/n  
  return(xbar)  
}  
redundantMean(vector1)
```

```
## [1] 4
```

Create a new vector (any name is fine) that contains the elements 4, 6, 22, 3, 5

- Access the 3rd element of your vector
- Use the **length** function to see how long your vector is
- Use the **mean** function to compute the mean of your vector
- Use **sum** to compute sum; confirm that this is equal to mean * length

Data frames

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1         3.5         1.4         0.2   setosa
## 2         4.9         3.0         1.4         0.2   setosa
## 3         4.7         3.2         1.3         0.2   setosa
## 4         4.6         3.1         1.5         0.2   setosa
## 5         5.0         3.6         1.4         0.2   setosa
## 6         5.4         3.9         1.7         0.4   setosa
```

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
head(iris)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
iris[1,1]
```

```
## [1] 5.1
```

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
iris[2,2]
```

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

Other means of indexing

```
iris[,1]
```

```
##      [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##     [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##    [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##   [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

Other means of indexing

```
iris[1,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1         5.1         3.5         1.4         0.2   setosa
```

Other means of indexing

```
iris$Petal.Length
```

```
## [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
## [19] 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2
## [37] 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0
## [55] 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0
## [73] 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5 4.5 4.7 4.4 4.1 4.0
## [91] 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3
## [109] 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0
## [127] 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9
## [145] 5.7 5.2 5.0 5.2 5.4 5.1
```

Other means of indexing

```
iris$Petal.Length[3]
```

```
## [1] 1.3
```


Other means of indexing

```
iris[1, "Petal.Length"]
```

```
## [1] 1.4
```

Some convenient data.frame functions

```
summary(iris)
```

```
##   Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##   Min.    :4.300   Min.      :2.000   Min.      :1.000   Min.      :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
##   Mean   :5.843   Mean    :3.057   Mean    :3.758   Mean    :1.199
##   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##   Max.    :7.900   Max.     :4.400   Max.     :6.900   Max.     :2.500
##           Species
##   setosa    :50
##   versicolor:50
##   virginica :50
##
##
##
```

Some convenient data.frame functions

```
nrow(iris)
```

```
## [1] 150
```

```
ncol(iris)
```

```
## [1] 5
```

```
dim(iris)
```

```
## [1] 150 5
```

Some convenient data.frame functions

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1         3.5         1.4         0.2   setosa
## 2         4.9         3.0         1.4         0.2   setosa
## 3         4.7         3.2         1.3         0.2   setosa
## 4         4.6         3.1         1.5         0.2   setosa
## 5         5.0         3.6         1.4         0.2   setosa
## 6         5.4         3.9         1.7         0.4   setosa
```

Some convenient data.frame functions

```
tail(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 145	6.7	3.3	5.7	2.5	virginica
## 146	6.7	3.0	5.2	2.3	virginica
## 147	6.3	2.5	5.0	1.9	virginica
## 148	6.5	3.0	5.2	2.0	virginica
## 149	6.2	3.4	5.4	2.3	virginica
## 150	5.9	3.0	5.1	1.8	virginica

Some convenient data.frame functions

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length"  "Petal.Width"  "Species"
```

How to do operations over a data.frame column

```
mean(iris$Petal.Length)
```

```
## [1] 3.758
```

```
sum(iris$Petal.Length)/nrow(iris)
```

```
## [1] 3.758
```

How to do operations over a data.frame column

```
iris$Petal.Length/2
```

```
## [1] 0.70 0.70 0.65 0.75 0.70 0.85 0.70 0.75 0.70 0.75 0.75 0.80 0.70 0.55 0.60
## [16] 0.75 0.65 0.70 0.85 0.75 0.85 0.75 0.50 0.85 0.95 0.80 0.80 0.75 0.70 0.80
## [31] 0.80 0.75 0.75 0.70 0.75 0.60 0.65 0.70 0.65 0.75 0.65 0.65 0.65 0.80 0.95
## [46] 0.70 0.80 0.70 0.75 0.70 2.35 2.25 2.45 2.00 2.30 2.25 2.35 1.65 2.30 1.95
## [61] 1.75 2.10 2.00 2.35 1.80 2.20 2.25 2.05 2.25 1.95 2.40 2.00 2.45 2.35 2.15
## [76] 2.20 2.40 2.50 2.25 1.75 1.90 1.85 1.95 2.55 2.25 2.25 2.35 2.20 2.05 2.00
## [91] 2.20 2.30 2.00 1.65 2.10 2.10 2.10 2.15 1.50 2.05 3.00 2.55 2.95 2.80 2.90
## [106] 3.30 2.25 3.15 2.90 3.05 2.55 2.65 2.75 2.50 2.55 2.65 2.75 3.35 3.45 2.50
## [121] 2.85 2.45 3.35 2.45 2.85 3.00 2.40 2.45 2.80 2.90 3.05 3.20 2.80 2.55 2.80
## [136] 3.05 2.80 2.75 2.40 2.70 2.80 2.55 2.55 2.95 2.85 2.60 2.50 2.60 2.70 2.55
```


Using `iris`

- Compute the sum of all petal lengths
- Compute the mean sepal width
- How many rows are in the data frame?
- How many columns?

- Tidyverse is a collection of packages that work together to make R work a bit more like `sql`
- These features make routine data manipulation tasks FAR easier than they are in base R

Installing packages

On your console, run this command:

```
install.packages("tidyverse")
```

You only have to install packages once.

Your first line of code in nearly every script from now on

```
library(tidyverse)
```

The basics of data manipulation: filter()

`filter()` allow us to pick observations based on their values

```
filter(iris, Sepal.Length == 5.1)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	5.1	3.5	1.4	0.3	setosa
## 3	5.1	3.8	1.5	0.3	setosa
## 4	5.1	3.7	1.5	0.4	setosa
## 5	5.1	3.3	1.7	0.5	setosa
## 6	5.1	3.4	1.5	0.2	setosa
## 7	5.1	3.8	1.9	0.4	setosa
## 8	5.1	3.8	1.6	0.2	setosa
## 9	5.1	2.5	3.0	1.1	versicolor

filter() can take many kinds of arguments

```
filter(iris, Sepal.Length<4.5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         4.4         2.9         1.4         0.2   setosa
## 2         4.3         3.0         1.1         0.1   setosa
## 3         4.4         3.0         1.3         0.2   setosa
## 4         4.4         3.2         1.3         0.2   setosa
```

filter() can take many kinds of arguments

```
filter(iris, Sepal.Length>=7.7)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	7.7	3.8	6.7	2.2	virginica
## 2	7.7	2.6	6.9	2.3	virginica
## 3	7.7	2.8	6.7	2.0	virginica
## 4	7.9	3.8	6.4	2.0	virginica
## 5	7.7	3.0	6.1	2.3	virginica

filter() can take more than one argument

```
filter(iris, Species == "versicolor" , Sepal.Length<5.5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1         4.9         2.4         3.3         1.0 versicolor
## 2         5.2         2.7         3.9         1.4 versicolor
## 3         5.0         2.0         3.5         1.0 versicolor
## 4         5.4         3.0         4.5         1.5 versicolor
## 5         5.0         2.3         3.3         1.0 versicolor
## 6         5.1         2.5         3.0         1.1 versicolor
```


filter() can take many kinds of arguments

```
filter(iris, Species != "setosa")
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width  Species
## 1          7.0         3.2         4.7         1.4 versicolor
## 2          6.4         3.2         4.5         1.5 versicolor
## 3          6.9         3.1         4.9         1.5 versicolor
## 4          5.5         2.3         4.0         1.3 versicolor
## 5          6.5         2.8         4.6         1.5 versicolor
## 6          5.7         2.8         4.5         1.3 versicolor
## 7          6.3         3.3         4.7         1.6 versicolor
## 8          4.9         2.4         3.3         1.0 versicolor
## 9          6.6         2.9         4.6         1.3 versicolor
## 10         5.2         2.7         3.9         1.4 versicolor
## 11         5.0         2.0         3.5         1.0 versicolor
## 12         5.9         3.0         4.2         1.5 versicolor
## 13         6.0         2.2         4.0         1.0 versicolor
## 14         6.1         2.9         4.7         1.4 versicolor
## 15         5.6         2.9         3.6         1.3 versicolor
## 16         6.7         3.1         4.4         1.4 versicolor
## 17         5.6         3.0         4.5         1.5 versicolor
## 18         5.8         2.7         4.1         1.0 versicolor
## 19         6.2         2.2         4.5         1.5 versicolor
## 20         5.6         2.5         3.9         1.1 versicolor
## 21         5.9         3.2         4.8         1.8 versicolor
## 22         6.1         2.8         4.0         1.3 versicolor
## 23         6.3         2.5         4.9         1.5 versicolor
## 24         6.1         2.8         4.7         1.2 versicolor
## 25         6.4         2.9         4.3         1.3 versicolor
## 26         6.6         3.0         4.4         1.4 versicolor
```

The basics of data manipulation: arrange()

Arrange reorders rows

```
arrange(iris, Sepal.Width)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.0	2.0	3.5	1.0	versicolor
## 2	6.0	2.2	4.0	1.0	versicolor
## 3	6.2	2.2	4.5	1.5	versicolor
## 4	6.0	2.2	5.0	1.5	virginica
## 5	4.5	2.3	1.3	0.3	setosa
## 6	5.5	2.3	4.0	1.3	versicolor
## 7	6.3	2.3	4.4	1.3	versicolor
## 8	5.0	2.3	3.3	1.0	versicolor
## 9	4.9	2.4	3.3	1.0	versicolor
## 10	5.5	2.4	3.8	1.1	versicolor
## 11	5.5	2.4	3.7	1.0	versicolor
## 12	5.6	2.5	3.9	1.1	versicolor
## 13	6.3	2.5	4.9	1.5	versicolor
## 14	5.5	2.5	4.0	1.3	versicolor
## 15	5.1	2.5	3.0	1.1	versicolor
## 16	4.9	2.5	4.5	1.7	virginica
## 17	6.7	2.5	5.8	1.8	virginica
## 18	5.7	2.5	5.0	2.0	virginica
## 19	6.3	2.5	5.0	1.9	virginica
## 20	5.7	2.6	3.5	1.0	versicolor
## 21	5.5	2.6	4.4	1.2	versicolor
## 22	5.8	2.6	4.0	1.2	versicolor
## 23	7.7	2.6	6.9	2.3	virginica
## 24	6.1	2.6	5.6	1.4	virginica

arrange() can sort ascending and descending

```
arrange(iris, desc(Sepal.Width))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.7	4.4	1.5	0.4	setosa
## 2	5.5	4.2	1.4	0.2	setosa
## 3	5.2	4.1	1.5	0.1	setosa
## 4	5.8	4.0	1.2	0.2	setosa
## 5	5.4	3.9	1.7	0.4	setosa
## 6	5.4	3.9	1.3	0.4	setosa
## 7	5.7	3.8	1.7	0.3	setosa
## 8	5.1	3.8	1.5	0.3	setosa
## 9	5.1	3.8	1.9	0.4	setosa
## 10	5.1	3.8	1.6	0.2	setosa
## 11	7.7	3.8	6.7	2.2	virginica
## 12	7.9	3.8	6.4	2.0	virginica
## 13	5.4	3.7	1.5	0.2	setosa
## 14	5.1	3.7	1.5	0.4	setosa
## 15	5.3	3.7	1.5	0.2	setosa
## 16	5.0	3.6	1.4	0.2	setosa
## 17	4.6	3.6	1.0	0.2	setosa
## 18	4.9	3.6	1.4	0.1	setosa
## 19	7.2	3.6	6.1	2.5	virginica
## 20	5.1	3.5	1.4	0.2	setosa
## 21	5.1	3.5	1.4	0.3	setosa
## 22	5.2	3.5	1.5	0.2	setosa
## 23	5.5	3.5	1.3	0.2	setosa
## 24	5.0	3.5	1.3	0.3	setosa
## 25	5.0	3.5	1.6	0.6	setosa
## 26	4.6	3.4	1.4	0.3	setosa

The basics of data manipulation: select()

`select()` allows us to focus on a subset of columns

```
select(iris, Species, Petal.Length)
```

##	Species	Petal.Length
## 1	setosa	1.4
## 2	setosa	1.4
## 3	setosa	1.3
## 4	setosa	1.5
## 5	setosa	1.4
## 6	setosa	1.7
## 7	setosa	1.4
## 8	setosa	1.5
## 9	setosa	1.4
## 10	setosa	1.5
## 11	setosa	1.5
## 12	setosa	1.6
## 13	setosa	1.4
## 14	setosa	1.1
## 15	setosa	1.2
## 16	setosa	1.5
## 17	setosa	1.3
## 18	setosa	1.4
## 19	setosa	1.7
## 20	setosa	1.5
## 21	setosa	1.7
## 22	setosa	1.5
## 23	setosa	1.0
## 24	setosa	1.7

select() can explicitly drop columns

```
select(iris, -Sepal.Length, -Sepal.Width)
```

##	Petal.Length	Petal.Width	Species
## 1	1.4	0.2	setosa
## 2	1.4	0.2	setosa
## 3	1.3	0.2	setosa
## 4	1.5	0.2	setosa
## 5	1.4	0.2	setosa
## 6	1.7	0.4	setosa
## 7	1.4	0.3	setosa
## 8	1.5	0.2	setosa
## 9	1.4	0.2	setosa
## 10	1.5	0.1	setosa
## 11	1.5	0.2	setosa
## 12	1.6	0.2	setosa
## 13	1.4	0.1	setosa
## 14	1.1	0.1	setosa
## 15	1.2	0.2	setosa
## 16	1.5	0.4	setosa
## 17	1.3	0.4	setosa
## 18	1.4	0.3	setosa
## 19	1.7	0.3	setosa
## 20	1.5	0.3	setosa
## 21	1.7	0.2	setosa
## 22	1.5	0.4	setosa
## 23	1.0	0.2	setosa
## 24	1.7	0.5	setosa
## 25	1.9	0.2	setosa
## 26	1.6	0.2	setosa

rename() renames columns

```
rename(iris, type = Species)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	type
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 6	5.4	3.9	1.7	0.4	setosa
## 7	4.6	3.4	1.4	0.3	setosa
## 8	5.0	3.4	1.5	0.2	setosa
## 9	4.4	2.9	1.4	0.2	setosa
## 10	4.9	3.1	1.5	0.1	setosa
## 11	5.4	3.7	1.5	0.2	setosa
## 12	4.8	3.4	1.6	0.2	setosa
## 13	4.8	3.0	1.4	0.1	setosa
## 14	4.3	3.0	1.1	0.1	setosa
## 15	5.8	4.0	1.2	0.2	setosa
## 16	5.7	4.4	1.5	0.4	setosa
## 17	5.4	3.9	1.3	0.4	setosa
## 18	5.1	3.5	1.4	0.3	setosa
## 19	5.7	3.8	1.7	0.3	setosa
## 20	5.1	3.8	1.5	0.3	setosa
## 21	5.4	3.4	1.7	0.2	setosa
## 22	5.1	3.7	1.5	0.4	setosa
## 23	4.6	3.6	1.0	0.2	setosa
## 24	5.1	3.3	1.7	0.5	setosa
## 25	4.8	3.4	1.9	0.2	setosa
## 26	5.0	3.0	1.6	0.2	setosa

Basics of data manipulation: mutate()

`mutate()` adds new columns, which can be functions of existing columns

```
mutate(iris, Petal.Size = Petal.Length + Petal.Width)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Petal.Size
## 1	5.1	3.5	1.4	0.2	setosa	1.6
## 2	4.9	3.0	1.4	0.2	setosa	1.6
## 3	4.7	3.2	1.3	0.2	setosa	1.5
## 4	4.6	3.1	1.5	0.2	setosa	1.7
## 5	5.0	3.6	1.4	0.2	setosa	1.6
## 6	5.4	3.9	1.7	0.4	setosa	2.1
## 7	4.6	3.4	1.4	0.3	setosa	1.7
## 8	5.0	3.4	1.5	0.2	setosa	1.7
## 9	4.4	2.9	1.4	0.2	setosa	1.6
## 10	4.9	3.1	1.5	0.1	setosa	1.6
## 11	5.4	3.7	1.5	0.2	setosa	1.7
## 12	4.8	3.4	1.6	0.2	setosa	1.8
## 13	4.8	3.0	1.4	0.1	setosa	1.5
## 14	4.3	3.0	1.1	0.1	setosa	1.2
## 15	5.8	4.0	1.2	0.2	setosa	1.4
## 16	5.7	4.4	1.5	0.4	setosa	1.9
## 17	5.4	3.9	1.3	0.4	setosa	1.7
## 18	5.1	3.5	1.4	0.3	setosa	1.7
## 19	5.7	3.8	1.7	0.3	setosa	2.0
## 20	5.1	3.8	1.5	0.3	setosa	1.8
## 21	5.4	3.4	1.7	0.2	setosa	1.9
## 22	5.1	3.7	1.5	0.4	setosa	1.9
## 23	4.6	3.6	1.0	0.2	setosa	1.2
## 24	5.1	3.3	1.7	0.5	setosa	2.2

mutate() can create many new variables at once

```
mutate(iris,  
  Petal.Size = Petal.Length + Petal.Width,  
  Sepal.Size = Sepal.Length + Sepal.Width,  
  Petal.Sepal.Ratio = Petal.Size/Sepal.Size)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Petal.Size
## 1	5.1	3.5	1.4	0.2	setosa	1.6
## 2	4.9	3.0	1.4	0.2	setosa	1.6
## 3	4.7	3.2	1.3	0.2	setosa	1.5
## 4	4.6	3.1	1.5	0.2	setosa	1.7
## 5	5.0	3.6	1.4	0.2	setosa	1.6
## 6	5.4	3.9	1.7	0.4	setosa	2.1
## 7	4.6	3.4	1.4	0.3	setosa	1.7
## 8	5.0	3.4	1.5	0.2	setosa	1.7
## 9	4.4	2.9	1.4	0.2	setosa	1.6
## 10	4.9	3.1	1.5	0.1	setosa	1.6
## 11	5.4	3.7	1.5	0.2	setosa	1.7
## 12	4.8	3.4	1.6	0.2	setosa	1.8
## 13	4.8	3.0	1.4	0.1	setosa	1.5
## 14	4.3	3.0	1.1	0.1	setosa	1.2
## 15	5.8	4.0	1.2	0.2	setosa	1.4
## 16	5.7	4.4	1.5	0.4	setosa	1.9
## 17	5.4	3.9	1.3	0.4	setosa	1.7
## 18	5.1	3.5	1.4	0.3	setosa	1.7
## 19	5.7	3.8	1.7	0.3	setosa	2.0
## 20	5.1	3.8	1.5	0.3	setosa	1.8
## 21	5.4	3.4	1.7	0.2	setosa	1.9
## 22	5.1	3.7	1.5	0.4	setosa	1.9
## 23	4.6	3.6	1.0	0.2	setosa	1.2

Tidyverse uses a special symbol, called a pipe `%>%` to string together commands. `Cmd + shift + m` will make one.

Piping commands

The pipe `%>%` tells R to use `iris` for all commands that follow

```
iris %>%  
  filter(Species == "versicolor")
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	7.0	3.2	4.7	1.4	versicolor
## 2	6.4	3.2	4.5	1.5	versicolor
## 3	6.9	3.1	4.9	1.5	versicolor
## 4	5.5	2.3	4.0	1.3	versicolor
## 5	6.5	2.8	4.6	1.5	versicolor
## 6	5.7	2.8	4.5	1.3	versicolor
## 7	6.3	3.3	4.7	1.6	versicolor
## 8	4.9	2.4	3.3	1.0	versicolor
## 9	6.6	2.9	4.6	1.3	versicolor
## 10	5.2	2.7	3.9	1.4	versicolor
## 11	5.0	2.0	3.5	1.0	versicolor
## 12	5.9	3.0	4.2	1.5	versicolor
## 13	6.0	2.2	4.0	1.0	versicolor
## 14	6.1	2.9	4.7	1.4	versicolor
## 15	5.6	2.9	3.6	1.3	versicolor
## 16	6.7	3.1	4.4	1.4	versicolor
## 17	5.6	3.0	4.5	1.5	versicolor
## 18	5.8	2.7	4.1	1.0	versicolor
## 19	6.2	2.2	4.5	1.5	versicolor
## 20	5.6	2.5	3.9	1.1	versicolor
## 21	5.9	3.2	4.8	1.8	versicolor
## 22	6.1	2.8	4.0	1.3	versicolor
## 23	6.3	2.5	4.9	1.5	versicolor

Piping commands

```
iris %>%  
  mutate(sepal2 = Sepal.Length + 2)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	sepal2
## 1	5.1	3.5	1.4	0.2	setosa	7.1
## 2	4.9	3.0	1.4	0.2	setosa	6.9
## 3	4.7	3.2	1.3	0.2	setosa	6.7
## 4	4.6	3.1	1.5	0.2	setosa	6.6
## 5	5.0	3.6	1.4	0.2	setosa	7.0
## 6	5.4	3.9	1.7	0.4	setosa	7.4
## 7	4.6	3.4	1.4	0.3	setosa	6.6
## 8	5.0	3.4	1.5	0.2	setosa	7.0
## 9	4.4	2.9	1.4	0.2	setosa	6.4
## 10	4.9	3.1	1.5	0.1	setosa	6.9
## 11	5.4	3.7	1.5	0.2	setosa	7.4
## 12	4.8	3.4	1.6	0.2	setosa	6.8
## 13	4.8	3.0	1.4	0.1	setosa	6.8
## 14	4.3	3.0	1.1	0.1	setosa	6.3
## 15	5.8	4.0	1.2	0.2	setosa	7.8
## 16	5.7	4.4	1.5	0.4	setosa	7.7
## 17	5.4	3.9	1.3	0.4	setosa	7.4
## 18	5.1	3.5	1.4	0.3	setosa	7.1
## 19	5.7	3.8	1.7	0.3	setosa	7.7
## 20	5.1	3.8	1.5	0.3	setosa	7.1
## 21	5.4	3.4	1.7	0.2	setosa	7.4
## 22	5.1	3.7	1.5	0.4	setosa	7.1
## 23	4.6	3.6	1.0	0.2	setosa	6.6
## 24	5.1	3.3	1.7	0.5	setosa	7.1
## 25	4.8	3.4	1.9	0.2	setosa	6.8

Using `iris` and the pipe operator `%>%`

- Use `select` to return only the petal variables and flower species
- Use `filter` to return only versicolor observations
- Use `arrange` to sort the data.frame by `Sepal.Length`
- Use `mutate` to compute the sum of petal lengths and widths
- Identify and return a data.frame containing only observations with the smallest observed petal width

Piping many commands

```
iris %>%  
  filter(Species == "versicolor") %>%  
  select(Sepal.Length, Petal.Length, Species) %>%  
  mutate(Sepal.Petal.Ratio = Sepal.Length / Petal.Length)
```

##	Sepal.Length	Petal.Length	Species	Sepal.Petal.Ratio
## 1	7.0	4.7	versicolor	1.489362
## 2	6.4	4.5	versicolor	1.422222
## 3	6.9	4.9	versicolor	1.408163
## 4	5.5	4.0	versicolor	1.375000
## 5	6.5	4.6	versicolor	1.413043
## 6	5.7	4.5	versicolor	1.266667
## 7	6.3	4.7	versicolor	1.340426
## 8	4.9	3.3	versicolor	1.484848
## 9	6.6	4.6	versicolor	1.434783
## 10	5.2	3.9	versicolor	1.333333
## 11	5.0	3.5	versicolor	1.428571
## 12	5.9	4.2	versicolor	1.404762
## 13	6.0	4.0	versicolor	1.500000
## 14	6.1	4.7	versicolor	1.297872
## 15	5.6	3.6	versicolor	1.555556
## 16	6.7	4.4	versicolor	1.522727
## 17	5.6	4.5	versicolor	1.244444
## 18	5.8	4.1	versicolor	1.414634
## 19	6.2	4.5	versicolor	1.377778
## 20	5.6	3.9	versicolor	1.435897
## 21	5.9	4.8	versicolor	1.229167
## 22	6.1	4.0	versicolor	1.525000
## 23	6.3	4.9	versicolor	1.285714

Summarizing data

`summarize()` uses a variety of summary functions over the data (e.g. `mean`, `min`, `max`, `sd`, etc.)

```
iris %>%  
  summarize(mean.pl = mean(Petal.Length))
```

```
##   mean.pl  
## 1    3.758
```

But `summarize()` is more powerful with its buddy, `group_by()`

`group_by()` groups the data by individual groups within the data

```
iris %>%  
  group_by(Species) %>%  
  summarize(mean.pl = mean(Petal.Length))
```

```
## # A tibble: 3 x 2  
##   Species    mean.pl  
##   <fct>      <dbl>  
## 1 setosa      1.46  
## 2 versicolor 4.26  
## 3 virginica  5.55
```

We can use it to count too

```
iris %>%  
  group_by(Species) %>%  
  summarize(n_obs = n())
```

```
## # A tibble: 3 x 2  
##   Species    n_obs  
##   <fct>    <int>  
## 1 setosa      50  
## 2 versicolor  50  
## 3 virginica   50
```


We can use it to count too

```
iris %>%  
  filter(Sepal.Width<3) %>%  
  group_by(Species) %>%  
  summarize(n_obs = n())
```

```
## # A tibble: 3 x 2  
##   Species    n_obs  
##   <fct>      <int>  
## 1 setosa         2  
## 2 versicolor   34  
## 3 virginica    21
```

- Which species has the largest average sepal length?
- Which species has the smallest average petal length?
- What is the average sepal width across all species?
- Which species has the highest number of observations with petal lengths less than the average petal length across all species?