CT5132/CT5148 Week 11 Exercises

James McDermott

University of Galway

Week 11 Exercises

The exercises and solutions are extracted from the lecture slides.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------- tidyverse
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts ---------------------------- tidyverse_confli
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Exercises (R Basics)

1. Write the Factorial function in R, eg `fact(5)` gives 120.
2. Given `x <- "John"`, calculate the length in characters of `x`. Use `nchar()`.
3. Given `xs <- c("John", "Paul", "George", "Ringo")`, calculate the length of each name, using vectorisation (not a for-loop).
4. Calculate whether each name is shorter than 5 characters.
5. Index `xs` to keep just the names shorter than 5 characters.
6. Write a function which unit-norms a vector, ie normalises it so that the vector length equals 1. Eg `unit_norm(c(10, 10, 10, 10))` gives `0.5 0.5 0.5 0.5`.
7. Write a function which standardises a vector, ie gets the *z*-score, ie maps it to have mean 0 and standard deviation 1. Eg `z_score(c(10, 6, 12, 12))` gives `0.0000000 -1.4142136  0.7071068 0.7071068`.

Solutions (R Basics)

```r
fact <- function(n) { # Exercise 1
  if (n <= 1) {
    1 # remember, no return statement!
  } else {
    n * fact(n-1)
  }
}
fact(5)
```

```
## [1] 120
```

```r
x <- "John"
nchar(x) # Exercise 2
```

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

```r
xs <- c("John", "Paul", "George", "Ringo")
nchar(xs) # Exercise 3
```

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

```r
nchar(xs) < 5 # Exercise 4
```

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

```r
xs[nchar(xs) < 5] # Exercise 5
```

```
## [1] "John" "Paul"
```

```
unit_norm <- function(x) { # Exercise 6
  x / sqrt(sum(x**2))
}
unit_norm(c(10, 10, 10, 10))
```

```
## [1] 0.5 0.5 0.5 0.5
```

```
z_score <- function(x) { # Exercise 7
  (x - mean(x)) / sd(x)
}
z_score(c(10, 6, 12, 12))
```

```
## [1]  0.0000000 -1.4142136  0.7071068  0.7071068
```

Exercises (Tidy Data)

1. Recall our experiment on running time for sorting an array of different sizes. The original data (before we added extra columns) is available in `data/sort_times_original.csv`. Read it in to a `tibble`. (You might need to set the working directory first.)
2. Use `glimpse` to take a look. What types do the columns have?
3. In what way is this *not* tidy data? Use `gather` to fix it. Hint: the result should have shape 50 x 3 with columns `n`, `run_number`, `run_time`.
4. It would be nicer if `run_number` was just an integer, eg 0, instead of `run0`. Use `separate` to split it into two parts. Hint: use `into=c("dummy", "run_number")`.
5. Look again at the result. We don't need that "dummy" column. Use `NA` to omit it. Hint: see `?separate` for help on `into`.
6. Look again – `run_number` is still not an integer! Fix this. Hint: `separate` can guess the correct type to convert to, but see `?separate` again to see how to ask it to.
7. Write it to a file `data/sort_times_tidy.csv` using `write_csv()`.

Solutions (Tidy Data)

Exercise 1

```
d <- read_csv("data/sort_times_original.csv")
```

```
## Parsed with column specification:
## cols(
##   n = col_double(),
##   run0 = col_double(),
##   run1 = col_double(),
##   run2 = col_double(),
##   run3 = col_double(),
##   run4 = col_double()
## )
```

Exercise 2

```r
glimpse(d) # All columns of type `dbl`, which is ok
```

```
## Observations: 10
## Variables: 6
## $ n    <dbl> 1e+06, 2e+06, 3e+06, 4e+06, 5e+06, 6e+06, 7e+0
## $ run0 <dbl> 0.09924603, 0.19706607, 0.30280304, 0.4454867
## $ run1 <dbl> 0.1099961, 0.1945050, 0.3008888, 0.5314040, 0.
## $ run2 <dbl> 0.1018548, 0.2033260, 0.3165970, 0.4161611, 0.
## $ run3 <dbl> 0.1004527, 0.1933441, 0.3653409, 0.4889781, 0.
## $ run4 <dbl> 0.1140921, 0.2565329, 0.3853610, 0.4850140, 0.
```

Exercise 3

```
d <- gather(d, key="run_number", value="run_time",
            run0, run1, run2, run3, run4)
```

Exercise 4

```
separate(d, run_number,
         into=c("dummy", "run_number"), sep=3)
```

```
## # A tibble: 50 x 4
##           n dummy run_number run_time
##       <dbl> <chr> <chr>         <dbl>
##  1  1000000 run   0            0.0992
##  2  2000000 run   0            0.197
##  3  3000000 run   0            0.303
##  4  4000000 run   0            0.445
##  5  5000000 run   0            0.584
##  6  6000000 run   0            0.771
##  7  7000000 run   0            1.54
##  8  8000000 run   0            0.982
##  9  9000000 run   0            1.24
## 10 10000000 run   0            1.38
## # ... with 40 more rows
```

Exercise 5

```r
separate(d, run_number,
         into=c(NA, "run_number"), sep=3)
```

```
## # A tibble: 50 x 3
##           n run_number run_time
##       <dbl> <chr>         <dbl>
##  1  1000000 0            0.0992
##  2  2000000 0            0.197
##  3  3000000 0            0.303
##  4  4000000 0            0.445
##  5  5000000 0            0.584
##  6  6000000 0            0.771
##  7  7000000 0            1.54
##  8  8000000 0            0.982
##  9  9000000 0            1.24
## 10 10000000 0            1.38
## # ... with 40 more rows
```

Exercise 6

```
d <- separate(d, run_number,
              into=c(NA, "run_number"), sep=3,
              convert=TRUE)
d
```

```
## # A tibble: 50 x 3
##           n run_number run_time
##       <dbl>      <int>    <dbl>
##  1  1000000          0   0.0992
##  2  2000000          0   0.197
##  3  3000000          0   0.303
##  4  4000000          0   0.445
##  5  5000000          0   0.584
##  6  6000000          0   0.771
##  7  7000000          0   1.54
##  8  8000000          0   0.982
##  9  9000000          0   1.24
## 10 10000000          0   1.38
```

Exercise 7

```
write_csv(d, "data/sort_times_tidy.csv")
```

Exercises (dplyr)

- Exercise 1: Our sort times data is available in tidy format as sort_times_tidy.csv. Use group_by and summarise to get the mean and the standard deviation for each n, and then for each run_number.
- A dataset of characters in *Star Wars* is available as dplyr::starwars. Exercise 2: Find all the human females. Exercise 3: Find the characters who are human *or* Wookiee. Exercise 4: Find the shortest character. Hint: recall we might need na.rm. Exercise 5: Add a new column called BMI giving the body mass index, where the formula is BMI $= m/h^2$ for mass m in kg and height h in metres. https://en.wikipedia.org/wiki/Body_mass_index. Exercise 6: Which character has the highest BMI?

Solutions (`dplyr`)

Exercise 1

```
d <- read_csv("data/sort_times_tidy.csv")
```

```
## Parsed with column specification:
## cols(
##   n = col_double(),
##   run_number = col_double(),
##   run_time = col_double()
## )
```

```
d %>% group_by(n) %>%
  summarise(mean_run_time=mean(run_time),
            sd_run_time=sd(run_time))
```

```
## # A tibble: 10 x 3
##          n mean_run_time sd_run_time
##      <dbl>         <dbl>       <dbl>
##  1 1000000         0.105     0.00654
##  2 2000000         0.209     0.0269
##  3 3000000         0.334     0.0387
```

Exercise 1

Notice that the mean and stddev for $n = 7$ million are anomalously high. One way this could occur is if our computer had a spike in CPU usage during the experiment, e.g. due to a browser loading a video.

Exercise 1

```
d %>% group_by(run_number) %>%
  summarise(mean_run_time=mean(run_time),
            sd_run_time=sd(run_time))
```

```
## # A tibble: 5 x 3
##   run_number mean_run_time sd_run_time
##        <dbl>         <dbl>       <dbl>
## 1          0         0.754       0.512
## 2          1         0.644       0.368
## 3          2         0.604       0.353
## 4          3         0.648       0.369
## 5          4         0.678       0.416
```

No major anomalies this time.

Exercise 2

```r
sw <- dplyr::starwars
# human females
sw %>% filter(species == "Human", gender == "female")
```

```
## # A tibble: 9 x 13
##   name  height  mass hair_color skin_color eye_color birth_
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <
## 1 Leia~    150    49 brown      light      brown
## 2 Beru~    165    75 brown      light      blue
## 3 Mon ~    150    NA auburn     fair       blue
## 4 Shmi~    163    NA black      fair       brown
## 5 Cordé    157    NA brown      light      brown
## 6 Dormé    165    NA brown      light      brown
## 7 Joca~    167    NA white      fair       blue
## 8 Rey       NA    NA brown      light      hazel
## 9 Padm~    165    45 brown      light      brown
## # ... with 5 more variables: homeworld <chr>, species <chr
```

Exercise 3

```
# human or Wookiee
sw %>% filter(species == "Human" | species == "Wookiee")
```

```
## # A tibble: 37 x 13
##    name   height  mass hair_color skin_color eye_color birth
##    <chr>   <int> <dbl> <chr>      <chr>      <chr>
##  1 Luke~     172    77 blond      fair       blue
##  2 Dart~     202   136 none       white      yellow
##  3 Leia~     150    49 brown      light      brown
##  4 Owen~     178   120 brown, gr~ light      blue
##  5 Beru~     165    75 brown      light      blue
##  6 Bigg~     183    84 black      light      brown
##  7 Obi-~     182    77 auburn, w~ fair       blue-gray
##  8 Anak~     188    84 blond      fair       blue
##  9 Wilh~     180    NA auburn, g~ fair       blue
## 10 Chew~     228   112 brown      unknown    blue
## # ... with 27 more rows, and 5 more variables: homeworld <c
```

Exercise 4

```
sw %>% filter(height == max(height, na.rm=TRUE))
```

```
## # A tibble: 1 x 13
##   name   height  mass hair_color skin_color eye_color birth_
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <
## 1 Yara~    264    NA none       white      yellow
## # ... with 5 more variables: homeworld <chr>, species <chr
## #   vehicles <list>, starships <list>
```

Exercise 5

```r
# NB convert height from cm to metres before squaring
BMI <- function(h, m) {m / (h / 100)^2}
sw <- sw %>% mutate(bmi=BMI(height, mass))
```

Exercise 6

```
sw %>% filter(bmi == max(bmi, na.rm=TRUE))
```

```
## # A tibble: 1 x 14
##   name  height  mass hair_color skin_color eye_color birth_
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <
## 1 Jabb~    175  1358 <NA>       green-tan~ orange
## # ... with 6 more variables: homeworld <chr>, species <chr
## #   vehicles <list>, starships <list>, bmi <dbl>
```