Module 2_2 - Functions

Keaton Wilson

Learning Objectives

1. Students will be able to recall some fundamental components of the biology of recirculating aqaculture systems and udnerstand how these components link to the problem at hand - particularly linking environmental tank conditions to physiology which in turn can affect disease rates.

2. Students will practice preliminary data explorations on tank data.

3. Students will be able to create custom functions that perform simple tasks (converting values in celsius to farenheit), with an eye to the utility of this generalizeable programming technique.

4. Students will practice data summary techniques from module 1, particularly tidyverse syntax.

Data Exploration - query to the class? Where did we leave off last time?

So, it does seem like there is something wrong with the fish, given the correlation we found earlier. Let's check out our tank data.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.3.

## v ggplot2 3.2.1     v purrr   0.3.3
## 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_conflicts(
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
fish_tank_data = read_csv("https://tinyurl.com/tbyskxl")
```

```
## Parsed with column specification:
## cols(
##   tank_id = col_double(),
##   species = col_character(),
##   avg_daily_temp = col_double(),
##   num_fish = col_double(),
##   day_length = col_double(),
##   tank_volume = col_double(),
##   size_day_30 = col_double()
## )
```

```
glimpse(fish_tank_data)
```

```
## Observations: 1,000
## Variables: 7
## $ tank_id        <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
## $ species        <chr> "tilapia", "tilapia", "tilapia", "tilapia", "tilapia...
## $ avg_daily_temp <dbl> 23.92704, 23.92969, 24.18485, 23.81712, 24.08202, 23...
## $ num_fish       <dbl> 99, 103, 104, 99, 100, 97, 104, 102, 98, 99, 102, 10...
## $ day_length     <dbl> 10, 9, 11, 11, 9, 10, 9, 11, 10, 11, 8, 10, 9, 10, 9...
## $ tank_volume    <dbl> 399.6131, 400.4007, 400.4813, 400.8997, 400.7822, 40...
## $ size_day_30    <dbl> 2786.305, 2787.769, 2788.661, 2778.243, 2791.451, 27...
```

Discussion of the problem

So here, the scenario, we have this data, and we know what the critical temperature cutoffs are for both species, but the problem is that they're in farenheit… and the values in our data frame are in celsius. The fish biologists we worked with on this were from the UK and US, and for some reason didn't give us information the appropriate format.

Anything below 59ª F for Trout and anything below 65º F for Tilapia is critically low, and can result in suppression of the immune system.

So here is the first challenge, how do we use this information given our current data frame. Ultimately, we want a filtered data frame that just shows us tanks that are below these values… trying to figure out how common it is.

With your groups, brainstorm some approaches you think might work. Don't actually code them out, just diagram, psuedo code or talk through them. As always, be ready to share.

Custom Functions

There are many solutions to this problem, but one that is particularly useful and powerful in R is building your own functions. We've already used functions a ton, and you can write your own… to do absolutely anything you want. This has lot of uses… but the big one here is that we can write a simple function that converts celsius to farenheit and apply it to our entire data set, in just a few lines of code.

```
# Here is the basic syntax

new_sum = function(x,y){
  z = x+y
  return(z)
}

new_sum(2,4)
```

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

Building a custom function for converting celsius to farenheit

In your groups, build a custom function called `c_to_f()` that converts a single value of celsius to farenheit. Check out the web for the equation on how to do this. Be prepared to show your code and run some test values.

```r
c_to_f = function(c = NULL){
  f = (c*1.8)+32
  return(f)
}

c_to_f(c = 40)
```

```
## [1] 104
```

Thinking about iteration and looping

So, we have a great function that we can use to convert our celsius values to farenheit, but how do we apply that to everything… we've built the function for one value, not for a vector of values…

Quickly in your groups, brainstorm some ideas on how you might implement this.

Three main ways to iterate

1. for loops
2. apply functions
3. Re-write your function

Next time we'll cover some options, but for now… the solution is actually really simple. The function we built actually will auto-iterate over a vector… because R is amazing at doing vector stuff. Check it out.

```r
c_to_f(fish_tank_data$avg_daily_temp)
```

```
##    [1] 75.06866 75.07344 75.53273 74.87081 75.34763 75.03976 75.19566 74.88584
##    [9] 75.22817 74.58180 74.60286 75.10135 75.42659 74.43868 75.07003 74.10868
##   [17] 73.91243 74.04529 75.37752 75.81322 74.23724 76.35335 75.22210 74.95958
##   [25] 73.59577 74.37272 77.01639 73.97914 73.64002 74.79769 75.77151 75.29395
##   [33] 76.32355 76.30084 73.65917 75.08868 74.65485 76.12371 75.24456 74.90208
##   [41] 73.92656 73.87968 75.32362 75.95446 74.90440 76.39606 75.10489 75.00112
##   [49] 75.67383 75.59593 75.29187 74.23135 75.13903 74.34478 75.48373 75.73244
##   [57] 75.01129 74.74751 75.79227 75.22892 75.86220 76.02696 75.17718 74.64985
##   [65] 75.12233 74.97072 74.57785 74.16118 76.51640 75.27735 75.01171 74.55144
##   [73] 74.40760 74.82447 74.18681 73.38865 74.93598 75.07566 75.05004 76.10977
##   [81] 75.44423 74.24517 74.12969 76.31287 73.62098 75.82851 75.14306 75.22465
##   [89] 73.45798 76.08465 73.20576 75.28530 74.02459 75.18128 73.99656 76.55444
##   [97] 75.10103 76.68251 75.52818 73.50869 74.13243 74.16180 74.81337 74.55849
##  [105] 74.04315 74.50551 75.54855 76.28985 75.94075 76.10191 74.98830 76.56389
##  [113] 74.17088 73.75512 74.46612 75.46141 74.42692 75.35652 74.15765 76.39262
##  [121] 75.22265 75.48103 75.06226 73.42623 76.18598 75.23258 73.81900 74.69035
##  [129] 74.46276 73.96664 76.03034 74.38874 78.71422 74.57488 73.15312 74.94593
##  [137] 75.23242 75.57135 75.21937 74.49343 74.60302 74.14242 75.33158 75.09711
##  [145] 76.35129 76.97065 74.51571 74.74363 75.78339 74.32596 75.77346 75.10451
##  [153] 74.35163 75.58481 74.34122 75.31775 74.26064 73.83143 76.24213 75.71114
##  [161] 74.10855 74.91389 74.41881 74.63438 74.86558 73.63335 74.63772 74.19117
##  [169] 74.71267 73.62542 74.68553 75.10184 75.34201 75.12246 74.32546 74.82960
##  [177] 74.96482 73.98958 74.63536 74.85512 75.95898 74.83912 74.41450 75.15329
##  [185] 73.67724 74.93358 74.90109 74.44321 73.40447 75.04978 74.40196 75.21041
##  [193] 75.39584 75.32335 75.94932 74.19433 73.89618 76.19637 76.11647 73.09952
##  [201] 76.54116 75.17503 74.06770 75.74848 75.45207 74.51163 75.15618 76.64050
##  [209] 75.26987 74.75185 74.60622 73.18135 73.80891 75.85540 76.24556 75.46012
##  [217] 75.39676 76.00883 73.69178 75.15962 73.50179 74.07712 74.69251 74.54902
##  [225] 75.82253 74.19644 75.03657 74.63192 74.82198 76.05931 75.03612 76.03543
##  [233] 76.05373 75.51768 74.51241 75.10584 73.79411 76.40484 73.98751 75.32488
##  [241] 74.44040 74.21775 75.07750 76.51063 76.18218 75.49442 74.94996 76.63865
##  [249] 74.82869 73.90723 72.99071 75.72896 76.62745 74.83990 75.28909 73.57706
##  [257] 74.57936 75.89376 76.26137 73.34261 76.26609 73.44479 75.33597 75.88790
##  [265] 76.01350 76.91592 75.77646 75.58277 74.81185 75.20471 74.99831 75.74854
##  [273] 74.45451 75.62124 76.17176 74.98068 75.58148 74.95999 74.76504 74.39252
##  [281] 73.77622 75.61536 75.92143 74.95052 74.19221 75.43070 75.99154 75.81348
##  [289] 74.55237 73.72592 75.05003 75.95213 74.21023 76.49035 73.40380 75.75064
##  [297] 74.51645 76.23044 75.67042 74.77203 72.87859 76.67738 75.60747 75.85833
##  [305] 76.16402 74.54852 75.68035 76.07760 73.71479 75.47155 74.11125 75.28818
##  [313] 76.10281 74.59588 73.14439 75.08414 75.12819 76.31395 74.57143 75.70684
##  [321] 76.18617 75.16647 74.81044 75.29700 76.55783 74.75324 75.05570 75.49702
##  [329] 74.53490 74.92832 76.67780 74.09849 74.78049 74.99993 76.42938 75.14287
##  [337] 74.67240 74.64490 75.88134 75.07960 74.39261 74.42437 75.64953 74.57222
##  [345] 73.94743 75.86078 74.54902 75.23797 75.53781 73.11306 75.60329 76.44946
##  [353] 74.68355 73.87409 75.46185 74.21672 74.75086 74.42011 74.75329 75.32138
##  [361] 75.60178 74.81717 73.70599 75.36560 75.62030 76.20829 75.81465 73.43919
##  [369] 73.74066 74.73774 74.47581 75.52285 75.72974 74.11422 73.55972 74.94728
##  [377] 75.04006 74.39188 74.43001 74.68193 75.76313 74.63663 76.29192 76.06990
##  [385] 73.44366 73.56818 73.89183 74.50725 75.65449 75.17035 75.33209 76.80008
##  [393] 74.38054 76.42637 74.06471 74.22131 74.95071 74.80739 76.55847 76.18005
##  [401] 75.82215 74.66133 74.78467 75.15069 75.79097 75.51003 74.85612 73.04281
##  [409] 75.06231 74.58625 75.21446 76.25264 75.61446 75.56094 75.29635 75.32934
##  [417] 74.68199 75.08783 74.47141 76.43279 75.97029 75.40711 76.27346 76.39921
##  [425] 74.98923 74.78867 73.51420 74.35468 76.31662 74.00459 74.99903 76.40887
##  [433] 74.83005 72.01249 72.64739 74.02247 75.04957 75.57715 74.74683 75.51303
##  [441] 75.26121 75.66828 74.91446 74.40417 75.93472 75.84389 75.04321 74.65584
##  [449] 74.67861 75.08569 76.19098 75.64835 75.32916 76.40658 73.89883 75.03004
##  [457] 74.45634 75.11936 74.29703 74.79924 75.95818 75.86880 76.19817 74.67452
##  [465] 75.82918 74.37283 73.73424 75.87979 74.97867 75.00063 74.99499 74.88138
##  [473] 76.08001 74.47208 75.49845 74.63169 74.37831 74.13459 73.04877 73.76465
##  [481] 75.63445 75.11440 77.05584 75.77566 75.05233 73.90281 74.50220 74.19274
##  [489] 74.81783 74.32176 74.59295 74.21393 73.68811 75.90277 75.09540 75.37602
##  [497] 73.65398 74.31490 74.74350 75.86187 74.75234 74.44908 76.66993 75.14033
##  [505] 75.78338 74.40081 74.22994 76.54632 75.48303 75.20050 74.89051 75.71856
##  [513] 74.04224 73.65273 77.22506 74.34776 73.92052 72.94204 76.09007 74.66441
##  [521] 76.37083 75.26039 75.23408 77.00945 74.29329 76.96866 74.10285 75.80514
##  [529] 74.35741 73.99651 76.17991 75.20075 76.45522 74.19901 73.77355 76.39649
##  [537] 76.21359 75.33397 74.96852 75.47322 74.00468 76.08928 75.56288 75.60471
##  [545] 75.23600 76.33729 75.00965 75.28337 74.30434 75.79370 74.85827 74.02043
##  [553] 74.46513 73.32137 75.95334 74.25109 72.86435 75.69668 73.56955 75.33004
##  [561] 74.38635 76.18377 74.74309 75.81109 74.70413 73.86072 75.27685 73.77264
##  [569] 75.10696 74.98350 74.15832 74.00970 73.87604 76.61765 76.24490 75.24894
##  [577] 75.93362 74.17372 75.42614 74.52603 75.40834 74.82029 76.73951 76.35721
##  [585] 74.58558 74.07730 74.97548 75.26811 75.78983 74.61759 75.93185 75.95244
##  [593] 74.93385 75.29618 74.35924 73.81002 76.53651 74.71045 75.36795 75.33585
##  [601] 73.86941 74.86270 74.90173 74.55712 74.56799 75.78132 74.98563 73.66974
##  [609] 75.19133 75.44030 75.53951 73.82612 74.86075 74.11444 74.51970 74.42108
##  [617] 75.66900 75.61061 73.99478 72.47781 75.45778 74.10973 77.10036 75.06917
##  [625] 73.58409 74.82925 74.89987 74.16702 75.55563 74.82586 73.74863 74.97146
##  [633] 74.11528 75.53956 74.59446 74.19679 73.61921 74.92865 73.34163 76.03320
##  [641] 75.56871 75.01896 75.08043 74.77315 74.43954 75.67666 74.32893 75.14663
##  [649] 75.09169 74.46701 74.60279 73.80397 75.51722 74.89953 74.54304 75.66468
##  [657] 75.63073 75.28346 74.47219 75.56254 73.34725 74.77869 74.72986 75.26788
##  [665] 74.95649 74.67573 74.38486 74.24900 73.60979 73.94814 75.44589 74.81516
##  [673] 73.91209 73.90286 75.20908 75.72469 74.13278 74.33985 75.15517 74.32127
##  [681] 75.61784 75.13097 75.11661 75.07181 74.95378 74.76336 73.81861 75.83302
##  [689] 74.64412 74.10918 75.25623 74.38123 74.41393 74.64983 75.26121 75.00537
##  [697] 76.18522 73.66928 75.33972 76.40743 73.88965 75.99408 75.37165 74.31195
##  [705] 75.94235 74.08047 74.49782 73.23096 74.69738 74.51777 77.58466 75.69593
##  [713] 76.08913 75.18766 74.94146 74.50445 73.79997 75.65491 75.30963 74.43652
##  [721] 75.14147 75.41537 74.79329 74.69100 74.21935 74.86011 75.81252 75.96644
##  [729] 75.73786 74.70359 75.07334 74.23036 76.56705 75.08483 75.26529 74.90414
##  [737] 75.33655 75.21456 72.89265 75.43183 75.02827 74.75677 74.93780 74.86776
##  [745] 73.70572 74.81703 75.63186 76.19216 75.46403 75.42614 59.18199 59.52628
##  [753] 58.75540 57.77538 58.43803 58.43565 59.45633 60.34071 58.10654 60.11184
##  [761] 57.64730 59.42821 58.52139 59.27752 58.04651 59.92208 60.05735 59.08057
##  [769] 59.54839 57.95236 59.05165 58.46695 58.26292 58.56199 58.80828 58.11657
##  [777] 59.58289 58.67915 59.04464 60.18565 58.49555 59.86248 58.63172 59.07695
##  [785] 58.62065 60.29892 59.29615 59.94928 59.64518 59.28930 58.61997 57.62515
##  [793] 58.85835 58.64455 59.69957 59.38105 58.13995 59.62304 59.71618 60.26507
##  [801] 59.80362 58.61043 58.21806 61.00382 58.30614 59.98057 59.51958 59.94210
##  [809] 57.31955 58.79972 59.57270 60.28505 58.71264 58.98423 58.43513 58.33030
##  [817] 59.51175 59.18600 59.67096 58.89575 60.14726 58.78072 59.69215 58.64692
##  [825] 57.48630 58.77667 58.69528 58.60696 58.44629 59.84777 58.32359 57.79150
##  [833] 58.82691 58.43635 59.01444 59.06590 60.70609 58.63354 58.63175 58.97484
##  [841] 61.67384 57.58693 58.47500 58.99634 58.64428 56.99346 58.85887 58.37510
##  [849] 58.78753 60.72110 59.22825 59.85375 60.20394 60.07484 58.68679 58.51885
##  [857] 59.16449 58.85927 58.15883 59.52826 58.61809 58.69395 59.10422 60.56987
##  [865] 59.11534 58.55388 59.15738 58.03875 58.89239 58.15619 58.82611 58.12656
##  [873] 58.54933 59.67897 57.79431 58.03083 60.25839 58.25548 58.66550 57.30053
##  [881] 60.00944 58.65266 58.87664 57.73442 59.51437 59.27195 59.69986 59.38470
##  [889] 57.46448 57.93004 59.55861 58.40389 59.96818 60.10439 58.42338 58.57007
##  [897] 60.05032 58.32774 60.13196 58.43425 59.12118 58.78293 58.72593 58.15149
##  [905] 59.04439 57.51281 59.26368 58.76600 58.66419 59.91794 59.24594 60.18335
##  [913] 58.59404 60.03353 58.53172 58.96314 60.99999 57.84410 59.18707 58.62823
##  [921] 57.97564 58.09841 57.80802 59.17464 57.23564 57.19897 58.29943 58.51416
##  [929] 60.03392 58.30749 59.48332 58.84834 58.52797 58.53484 58.51328 58.64311
##  [937] 59.98118 58.64858 59.45344 59.44152 59.22711 60.05601 60.47193 59.41932
##  [945] 58.55411 58.78864 57.77648 60.08416 60.20946 58.48950 58.57124 58.64052
##  [953] 57.72976 59.17489 59.96123 59.10811 58.56501 59.71142 59.78552 58.39409
##  [961] 59.36325 58.72293 61.09714 58.73519 57.45339 59.65728 59.02008 57.48416
##  [969] 57.39791 57.80607 59.65003 59.40106 58.36286 56.92565 59.20587 58.15021
##  [977] 59.01959 60.05027 60.27371 59.42370 58.99054 59.08643 59.44103 58.70893
##  [985] 59.24048 59.42119 58.69374 58.83941 58.59189 58.67367 57.97295 59.03970
##  [993] 58.82814 58.50285 60.04439 58.44347 59.35385 57.33042 59.80184 59.45482
```

```r
# So, we can do something like this
fish_tank_data = fish_tank_data %>%
  mutate(avg_daily_temp_F = c_to_f(avg_daily_temp))
```

```r
fish_tank_data
```

```
## # A tibble: 1,000 x 8
##    tank_id species avg_daily_temp num_fish day_length tank_volume size_day_30
##      <dbl> <chr>            <dbl>    <dbl>      <dbl>       <dbl>       <dbl>
##  1       1 tilapia           23.9       99         10        400.       2786.
##  2       2 tilapia           23.9      103          9        400.       2788.
##  3       3 tilapia           24.2      104         11        400.       2789.
##  4       4 tilapia           23.8       99         11        401.       2778.
##  5       5 tilapia           24.1      100          9        401.       2791.
##  6       6 tilapia           23.9       97         10        400.       2785.
##  7       7 tilapia           24.0      104          9        400.       2787.
##  8       8 tilapia           23.8      102         11        400.       2782.
##  9       9 tilapia           24.0       98         10        400.       2790.
## 10      10 tilapia           23.7       99         11        400.       2787.
## # ... with 990 more rows, and 1 more variable: avg_daily_temp_F <dbl>
```

Filtering based on our new converted column

Ok, so we've solved one problem, but not the main one… who remembers the original question we set out to examine here?

How many tanks are below the temperature cutoffs for each species?

Take the rest of the class period to work on figuring this out (hint: remember the group_by and filter and summary functions we used in module 1?). If you can't finish, work in groups remotely to figure out a solution by the beginning of the next class.

```r
fish_tank_data %>%
  filter(species == "tilapia" & avg_daily_temp_F < 75 |
         species == "trout" & avg_daily_temp_F < 59) %>%
  group_by(species) %>%
  summarize(n = n())
```

```
## # A tibble: 2 x 2
##   species     n
##   <chr>   <int>
## 1 tilapia   376
## 2 trout     137
```