# On Loops

Sander Renes

6-3-2020

| Functions needed |
|---|
| bind_rows() |
| dim() |
| filter() |
| for |
| geom_point |
| gglot() |
| group_by |
| list.files() |
| lm() |
| max() |
| min() |
| mutate() |
| near() |
| paste0() |
| print |
| read_csv() |
| readRDS |
| rename() |
| rnorm() |
| saveRDS() |
| select() |
| seq_along() |
| setwd() |
| stargazer() |
| stargazer() |
| str() |
| summarise() |
| tibble() |
| while |

# Take home Exercises Loops

## exercise 1

in the folder "Weekly data - year 1", you find .csv files with the weekly data on the costs of producing a product.
There is one file per week. Every file contains the data on 5 production days for 2 production lines, A and B, so 10 observations per file, 50 weeks means 50 files.

### exercise 1a

Use the read_csv function to load all of the files in to one Tibble called "production_data_df" with a *for* loop (Hint, suppress output of the read_csv function)

```
#your code goes here


#your code ends here
# uncomment the line below

# dim(production_data_df)

# expected output: [1] 500   5
```

### exercise 1b

now try this with a *while* loop

```
#your code goes here


#your code ends here
# uncomment the line below

# dim(production_data_df)

# expected output: [1] 500   5
```

### exercise 1c

Now use a *seq_along* loop (Hint look at the 'dir' functions)

```
#your code goes here


#your code ends here
# uncomment the line below

# dim(production_data_df)

# expected output: [1] 500   5
```

**exercise 1d**

did you do the first step of writing down the basics? if not, do so now, for all 3 loops.

**answer**

**end answer**

**exercise 1e**

the advantage of the loops are that you do not have to repeat yourself 50 times to load and append. Still these loops are not the same, which of these loops could you re-use most easily for different folders or periods?

**answer**

**end answer**

**exercise 1f**

1) Now what happens if you ask for production_data_df$cost[4]? How can you find the same item using two dimensional indexing on the tibble (hint try asking for the same item in a 2-dimensional list, or extracting it from a numeric vector selected from the tibble/list)

2) What value do you get if you ask production_data_df[1,3] and production_data_df[3,1]? Where do you find them in the tible?

**answer**

**end answer**

## Exercise 2:

Try running your best loop on the folder "Weekly data-year2". Get the data in there correctly and clean out all mistakes: (HINT: check that you have the right number of obs per week for all weeks, and have the right variables for all weeks)

```
# uncomment the line below

# str(production_data_df)

# expected output:
# Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 500 obs. of  5 variables:
#........
```

## exercise 3

0) load the data from exercise 2, and make a separate tibble containing only the data of product A, call it "product_A"

**exercise 3a**

Find the highest and lowest value of cost per unit for product A.

```
#your code goes here
```

**exercise 3b**

Get the lists of weeks in which the cost per unit are equal to the maximum and minimum. What parts of the year are those?

```
# max_week <- filter(product_A, cost_pu==0)## fill in the number your found for the maximum and uncomme
# min_week <- filter(product_A, cost_pu==0)## fill in number your found for the minimum and uncomment
```

**exercise 3c**

The code in q2 has an issue because of the precision of numbers stored in your computer. There are 2 solutions available, either the near() function, or the max() and min() functions. read the help-files of these functions and find the week with the maximum using near, and the minimum using the min() function.

```
#your code goes here
```

Many of the operations you can do in loops, you can also do directly on the vector of data. Vectorization of an operation is a substitute for loops. In the past (and still in some programming languages) vectorization was much faster and more computationally efficient than loops, but the difference is quickly disappearing in R (mostly because R has vectorized most loop operations). For instance, if you want to add the production costs of product A and product B in every week, you could loop over every week to add them together, or you could use a vectorized version. Make a vector (i.e. variable in a tibble) with the production cost of A, a variable with the production cost of B and use a vectorized addition (mutate command) to add them together in a new variable.

**exercise 3d**

Create a tibble with the maximum of per unit production cost per week (i.e. maximum over the five days in the week) for product A, and a variable indicating the week. Use a loop over the weeks in the dataset "Product_A".

```
#your code goes here
```

**exercise 3e**

Create a tibble with the maximum of per unit production cost per week (i.e. maximum over the five days in the week) for product A, and a variable indicating the week. Use the vectorized command summarise() {hint: think groups and group_by}.

```
#your code goes here
```

**exercise 3f**

Plot the two variables you created in d) and e) in a single plot and check that they are the same.

```
#your code goes here
```

## exercise 4

We have not used loops to do much that cannot be done without a loop. Loading the data as in exercise 1 could have been done through existing commands (see the document "planning to load the data.Rmd"). In many statistical analysis, however, using loops can help a lot. For instance in doing subset analysis. In our production data, we might suspect that seasonal patterns exist. If this is the case, the relation between some of our variables should be different from month to month, and that we can check by calculating the same relation – either a correlation of a regression coefficient– every month. In our production costs, other_costs can be split into labor and overhead costs.

"production_data_df2.rds" contains the same data as in the last questions, but now breaks some of the costs down to labor and overhead costs. Overhead is applied in this company via labor costs, but the company is unsure whether the relation between overhead and labor is constant over the year. To test it, the company compared the relation between overhead and labor over the entire year, to the relation found in each of 12 blocks of 4 weeks called "month". Both relations are found by running linear regressions. Note that this is essentially subset-analysis for 12 subsets of data. They used this (terrible) code:

```
# Load the data in "production_data_df2.rds"
 production_data_df2<-readRDS(file = "production_data_df2.rds")

# Fit the overhead rate for the entire period:
lm(overhead ~ labor_cost, data=production_data_df2)
```

Call: lm(formula = overhead ~ labor_cost, data = production_data_df2)

Coefficients: (Intercept) labor_cost
-16.198 2.324

```
#check if it is the same in all months

  data_month1<-filter(production_data_df2, month==1 )
  data_month2<-filter(production_data_df2, month==2 )
  data_month3<-filter(production_data_df2, month==3 )
  data_month4<-filter(production_data_df2, month==4 )
  data_month5<-filter(production_data_df2, month==5 )
  data_month6<-filter(production_data_df2, month==6 )
  data_month7<-filter(production_data_df2, month==7 )
  data_month8<-filter(production_data_df2, month==8 )
  data_month9<-filter(production_data_df2, month==9 )
  data_month10<-filter(production_data_df2, month==10 )
  data_month11<-filter(production_data_df2, month==11 )
  data_month12<-filter(production_data_df2, month==12 )

  linearMod1 <- lm(overhead ~ labor_cost, data=data_month1)  # build linear regression model on month 1
  linearMod2 <- lm(overhead ~ labor_cost, data=data_month2)  # build linear regression model on month 2
  linearMod3 <- lm(overhead ~ labor_cost, data=data_month3)  # build linear regression model on month 3
  linearMod4 <- lm(overhead ~ labor_cost, data=data_month4)  # build linear regression model on month 4
  linearMod5 <- lm(overhead ~ labor_cost, data=data_month5)  # build linear regression model on month 5
  linearMod6 <- lm(overhead ~ labor_cost, data=data_month6)  # build linear regression model on month 6
  linearMod7 <- lm(overhead ~ labor_cost, data=data_month7)  # build linear regression model on month 7
  linearMod8 <- lm(overhead ~ labor_cost, data=data_month8)  # build linear regression model on month 8
  linearMod9 <- lm(overhead ~ labor_cost, data=data_month9)  # build linear regression model on month 9
  linearMod10 <- lm(overhead ~ labor_cost, data=data_month10)  # build linear regression model on month
  linearMod11 <- lm(overhead ~ labor_cost, data=data_month1)  # build linear regression model on month
  linearMod12 <- lm(overhead ~ labor_cost, data=data_month12)  # build linear regression model on month

  print ("month 1")
```

[1] “month 1”

```
print(linearMod1)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month1)

Coefficients: (Intercept) labor_cost
0.6429 2.0178

```
  print ("month 2")
```

[1] “month 2”

```
print(linearMod2)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month2)

Coefficients: (Intercept) labor_cost
0.551 2.055

```
  print ("month 3")
```

[1] “month 3”

```
print(linearMod3)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month3)

Coefficients: (Intercept) labor_cost
-1.413 2.104

```
  print ("month 4")
```

[1] “month 4”

```
print(linearMod4)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month4)

Coefficients: (Intercept) labor_cost
1.827 2.122

```
  print ("month 5")
```

[1] “month 5”

```
print(linearMod5)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month5)

Coefficients: (Intercept) labor_cost
4.610 2.146

```
  print ("month 6")
```

[1] “month 6”

```
print(linearMod6)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month6)

Coefficients: (Intercept) labor_cost
2.621 2.190

```
  print ("month 7")
```

[1] “month 7”

```
print(linearMod7)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month7)

Coefficients: (Intercept) labor_cost
5.623 2.208

```
  print ("month 8")
```

[1] “month 8”

```
print(linearMod8)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month8)

Coefficients: (Intercept) labor_cost
-0.5875 2.2780

```
  print ("month 9")
```

[1] “month 9”

```
print(linearMod9)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month9)

Coefficients: (Intercept) labor_cost
1.042 2.304

```
  print ("month 10")
```

[1] "month 10"

```
print(linearMod10)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month10)

Coefficients: (Intercept) labor_cost
2.792 2.331

```
  print ("month 11")
```

[1] "month 11"

```
print(linearMod11)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month1)

Coefficients: (Intercept) labor_cost
0.6429 2.0178

```
  print ("month 12")
```

[1] "month 12"

```
print(linearMod12)
```

Call: lm(formula = overhead ~ labor_cost, data = data_month12)

Coefficients: (Intercept) labor_cost
2.206 2.407

**exercise 4a**

Please rewrite the code above, feel free to remove the old code to reduce the size of your assignment output, the code above using a loop to make the code easier to read, and remove the mistake in the code.* Bonus assignment (good prep for your thesis), use the stargazer package to organize the output of the regression in (a) clear table(s)

**exercise 4b**

Note that there is also an indicator for weeks, what would have to change in your loop to run the loop over all weeks in stead of the months? (explain, no code required)

**answer**

**end answer**