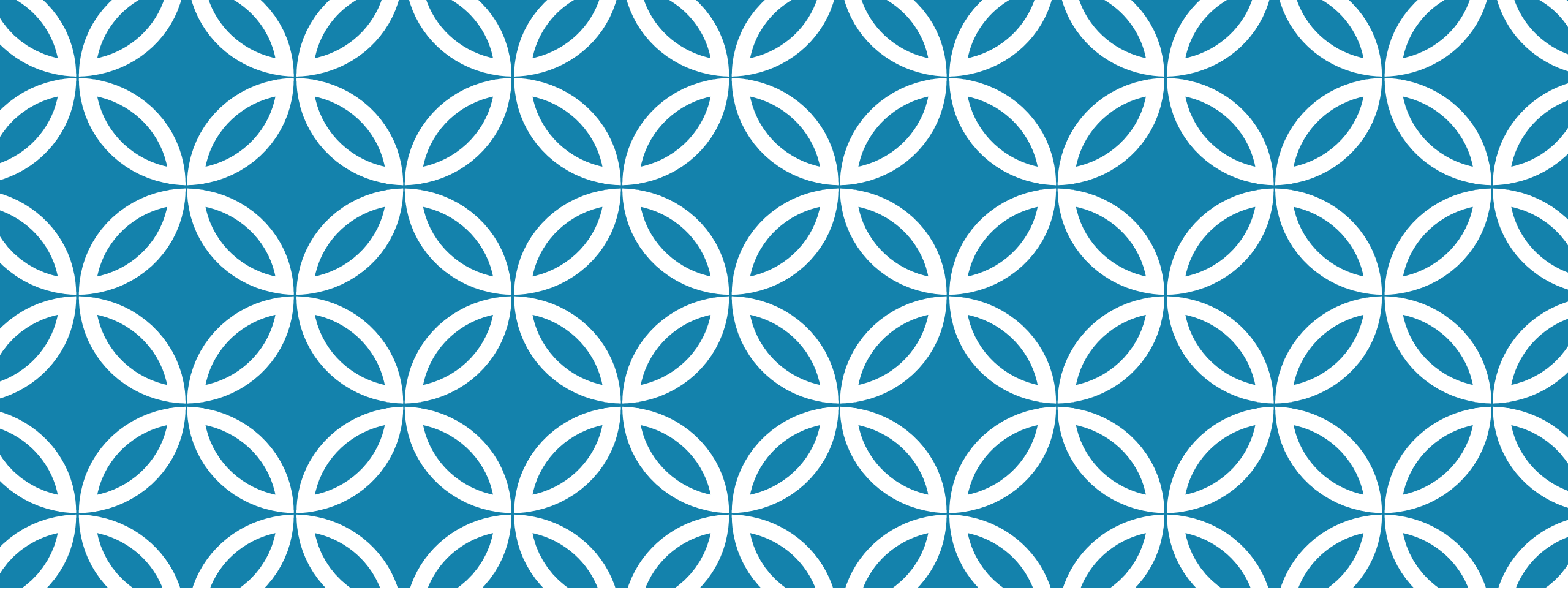

STATISTICS AND MODELLING

mgr Maria Kubara, WNE UW

STATISTICS IN R

Important argument:: na.rm = TRUE to omit the NA (missing value) when calculating the statistics

Function	Description
`str()`	Check structure
`min()`	Smallest value
`max()`	Largest value
`mean()`	Mean
`median()`	Median
`quantile(X, p=0.25)`	Quantile (of given percentage)
`IQR()`	Interquartile range
`sd()`	Standard deviation
`var()`	Variance

MODELS IN R

Depending on the goal of our analysis we can create numerous models in R, using built-in functions or functions coming from additional packages.

Analysis of the models in R is very convenient. We get only one output object, which stores all the meaningful information for the given modelling technique. Usually, the output object will be an R list, which flexible structure is exactly what will come in handy here.

```
> model <- lm(GNP ~ Employed, data = longley)
> model

Call:
lm(formula = GNP ~ Employed, data = longley)

Coefficients:
(Intercept)     Employed
   -1430.48        27.84

>
> summary(model)

Call:
lm(formula = GNP ~ Employed, data = longley)

Residuals:
    Min      1Q  Median      3Q     Max
-39.223 -11.920   0.855  14.882  23.555

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1430.482     89.361  -16.01 2.15e-10 ***
Employed       27.836      1.366   20.37 8.36e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.58 on 14 degrees of freedom
Multiple R-squared:  0.9674,    Adjusted R-squared:  0.965
F-statistic: 415.1 on 1 and 14 DF,  p-value: 8.363e-12
```

LM() OUTPUT STRUCTURE

```
> str(model)
List of 12
 $ coefficients : Named num [1:2] -1430.5 27.8
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "Employed"
 $ residuals    : Named num [1:16] -14.395 -11.499 13.601 11.864 -0.379 ...
  ..- attr(*, "names")= chr [1:16] "1947" "1948" "1949" "1950" ...
 $ effects      : Named num [1:16] -1550.79 378.62 18.61 16.35 3.06 ...
  ..- attr(*, "names")= chr [1:16] "(Intercept)" "Employed" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:16] 249 271 244 273 329 ...
  ..- attr(*, "names")= chr [1:16] "1947" "1948" "1949" "1950" ...
 $ assign       : int [1:2] 0 1
 $ qr           :List of 5
  ..$ qr   : num [1:16, 1:2] -4 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:16] "1947" "1948" "1949" "1950" ...
  .. .. ..$ : chr [1:2] "(Intercept)" "Employed"
  .. ..- attr(*, "assign")= int [1:2] 0 1
  ..$ qraux: num [1:2] 1.25 1.23
  ..$ pivot: int [1:2] 1 2
  ..$ tol  : num 1e-07
  ..$ rank : int 2
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 14
 $ xlevels      : Named list()
 $ call         : language lm(formula = GNP ~ Employed, data = longley)
 $ terms        :Classes 'terms', 'formula'  language GNP ~ Employed
  .. ..- attr(*, "variables")= language list(GNP, Employed)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "GNP" "Employed"
  .. .. .. ..$ : chr "Employed"
  .. ..- attr(*, "term.labels")= chr "Employed"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
```

output is a list

1 Vector

2

3

4

Matrix

List within a list

Integer

Object of class terms & formula within a list

KMEANS() OUTPUT STRUCTURE

```
> clustering <- kmeans(waterNoMiss, 4)
> summary(clustering)
             Length Class  Mode
cluster      2011   -none- numeric
centers        40   -none- numeric
totss           1   -none- numeric
withinss        4   -none- numeric
tot.withinss    1   -none- numeric
betweenss       1   -none- numeric
size            4   -none- numeric
iter            1   -none- numeric
ifault          1   -none- numeric
>
> str(clustering)
List of 9
 $ cluster     : Named int [1:2011] 4 4 3 3 1 3 3 4 2 3 ...
  ..- attr(*, "names")= chr [1:2011] "4" "5" "6" "7" ...
 $ centers     : num [1:4, 1:10] 7.31 6.88 7.04 7.03 200.23 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "1" "2" "3" "4"
  .. ..$ : chr [1:10] "ph" "Hardness" "Solids" "Chloramines" ...
 $ totss       : num 1.5e+11
 $ withinss    : num [1:4] 4.20e+09 5.51e+09 3.65e+09 3.47e+09
 $ tot.withinss: num 1.68e+10
 $ betweenss   : num 1.33e+11
 $ size        : int [1:4] 504 238 560 709
 $ iter        : int 2
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
> clustering$centers
        ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_carbon Trihalomethanes Turbidity Potability
1 7.305184 200.2295 11895.11    7.271365 341.3755     430.5896       14.47270        67.12800  3.932841  0.3908730
2 6.875451 192.2817 38096.99    7.042852 322.2074     418.8249       14.43157        66.64036  3.993359  0.4579832
3 7.042767 196.8145 27246.67    7.058733 329.4239     431.6737       14.37230        65.38228  3.968482  0.3946429
4 7.034988 193.5077 19401.44    7.127359 334.1309     422.1577       14.23965        66.60808  3.989004  0.4005642
```

output is a list

matrix

vectors

inside the centers matrix (could call clustering[[2]] as well)

LAPPLY

More on the *apply family functions on the Advanced Programming in R classes. This is just to signalize the topic. You can always read more in documentation or online tutorials.

***apply family functions allow you to apply a given method over a series of elements – vectors or lists.**

Example:

We want to use mean() function on 2nd, 3rd and 4rd column of our dataset. We can do it in two ways:

1) `mean(data[,2], na.rm=T); mean(data[,3], na.rm=T); mean(data[,4], na.rm=T)`

2) `lapply(data[,2:4], mean, na.rm=T)`

1st approach requires repeating the function call for each vector we want to calculate the average from. 2nd approach uses the lapply (list apply) function to apply the mean function to the vectors creating 2nd, 3rd, and 4th columns of our data. As a result, we will get a list which elements will be consecutive results of the mean calculation. We can also use the na.rm=T parameter, just by adding it after the function name.