[[R language]]

Slide usate a lezione:

Course Log

1. Hasty tour inside R
2. READR
3. Tidy
4. DPLYR
5. ggplot2
6. Elo - chess

[[R Guide|R Guide markdown file]]

Definizioni

Media

> Rapporto tra la somma dei dati numerici e il numero dei dati.

Gli *outlayers* influenzano molto la media, ma è generalmente pià accurata per rappresentare il valore medio di un insieme di dati. ### Mediana > Valore centrale tra i dati numerici disposti in ordine crescente o decrescente.

In caso di numero pari di dati, la median è data dalla media aritmetica dei due valori centrali. La presenza di *outlayers* (=valori distanti dalla media) non influenza necessariamente la mediana.

Moda

> Valore che si presenta con maggiore frequenza

Quindi, se ogni valore è presente una volta, non esiste una moda.

Data structures

> [!info]- Atomic Vectors > Sequence of elements with same type. Vector index starts at 1, not 0
>
> `c(1, 3, 5)`, `c(TRUE, FALSE, TRUE)` ### Indexing: `primes = c(2, 3, 5, 7)` `primes[2]` -> `[1] 3` **Coercion** = gli elementi dei vettori devono avere lo stesso tipo quindi quando combinati vengono convertiti nel tipo più flessibile. In ordine di flessibilità, sono: 1. logical 2. integer 3. double 4. character ### Factors > vectors that can contain only predefined values. used to store categorical variables. (ex. sex)

> [!info]- List > sequence of elements that might have different types
>
> `l = list(thing = "hat", size = 3, female = TRUE)` `l[1]` = a sublist containing the first element of the list `l[[1]]` = the first element of the list.

> [!info]- Matrix > 2-dimensional vector: same type and same length. `M[1, ]` = first row use `cbind` and `rbind` to add columns and rows to a matrix. ### Matrix operations - Element-wise sum `M+N` - Element-wise product `M*N` - matrix product `M %*% N*` - Matrix transpose `t(M)` - matrix inverse - linear systems - matrix spectrum - eigenvalues

> [!info]- Data Frame > List of vectors (columns). It's like a database table: > - Each column has a name and contains elements of the same type. > - Different columns have the same length and may have different types.
>
> ```
> name = c("John", "Enri")
> age = c("3", "20")
> bros = data.frame(name, age)
> ```
>
> Data frames can be nested.

[[R programming]]

loops

`for`, **`while`** loops. `ifelse` statements Most of the times tasks can be performed by applying functions, avoiding loops. (pipes) ## Functions Can be standard (`log`) or custom made. ### Functionals can be recursive. Can be applied to each element of a vector: `lapply(df, mean)` -> apply mean to each column of dataframe, return a list. `sapply(df, mean)` -> apply mean to each col of dataframe, return atomic vector. `apply(mtx, 1, mean)` -> apply mean to each row of matrix, return atomic vector.

Plot [[R visualization]]

- Barplot
  - [[Pasted image 20230911200943.png|300]]
- Histogram
  - [[Pasted image 20230911201008.png|300]]
- BoxPlot
- [[Pasted image 20230911201657.png|300]]
  - Per visualizzare la distribuzione dei dati. Include dati come mediana, variabilità, presenza di outlier e simmetria dati.
  - Asse orizzontale che rappresenta la variabile di interesse
  - Il box copre il secondo e terzo quartile dei dati. La linea nel box rappresenta la mediana.
    - I quartili dividono l'insieme dei dati in 4 parti uguali. ciascun quartile contiene un quarto dei dati. Il primo quartile (Q1) contiene il primo 25% dei dati. Sono importanti perchè forniscono una misura della dispersione e posizione dei dati in un set.
  - i "baffi", le due linee sopra e sotto il box, rappresentano l'intervallo interquartile *IQR* ovvero la distanza tra il secondo e terzo quartile. Quindi si estendono fino al massimo e minimo dei dati all'interno di un certo intervallo.
  - Outlier: i dati che cadono fuori dai "baffi" sono considerati Outlier. Solitamente sono dati anomali o errori.
  - Simmetria dei dati: se la mediana è al centro del box e i baffi sono uguali in distanza dal box, i dati sono distribuiti simmetricamente.

Data import [[R import]]

Tibbles

> Tibbles are data frames with some improvements. They are defined in the `tibble` package.

Some of the improvements include: - Printing is more user friendly - Column types are preserved - Subsetting returns a vector, instead of a dataframe with a single column which can be confusing. - Automatic col and row names

`tidyr` : tidy data

- Variable = quantity or quality that you can measure
- Value = state of a variable when you measure it
- Observation = set of measurements of variables made under similar conditions. [[Pasted image 20230909015838.png|600]]

Gathering

`gather(table4a, "1999", "2000", key = "year", value = "cases")` [[Pasted image 20230909015951.png|500]]

Spreading

[[Pasted image 20230909020144.png|500]]

Separating

[[Pasted image 20230909020239.png|500]]

Uniting

[[Pasted image 20230909020312.png|500]]

`dplyr` : Data manipulation [[Dplyr]]

Unary verbs

- `select()` pick variables based on names (it's a filter on columns of the data frame)
- `filter()` pick case based on their values (it's a filter on **rows** of the data frame)
- `mutate()` add new variable in function of existing data
- `arrange()` change the order of the rows
- `group_by()` partition rows of data into groups defined by the value of some variables.
- `summarise()` reduces multiple values down to a single summary.

Binary verbs

when many tables contribute to analysis, tools to combine them exist: - mutating join: add new variables to one table from matching rows in another - filtering joins: filter observation from one table based on whether or not they match observation on another table - set operations. combine observations in dataset as if they where set elements. ### Joins - **mutating joins** add new variables to one table from matching rows in another - **inner join** includes observations that match in both tables - **outer join** (left, right, full) includes also observations that do not match in one of the tables - **filtering joins** filter observations from one table from matching rows in another - **semi-join** filter observations from one table based on whether they match an observation in the other table - **anti-join** filter observations from one table based on whether they do not match an observation in the other table

Pipes

> powerful tool for clearly expressing a sequence of multiple operations.
>
> ```
> foo_foo %>%
> hop(through = forest) %>%
> scoop(up = field_mouse) %>%
> bop(on = head)
> ```

ggplot2

```
ggplot(data = df)+
    geom_obj(mapping = aes())
```