

Misurazione Implicita in Psicologia

Analizzare i dati IAT

Ottavia M. Epifania
ottavia.epifania@unipd.it

Master di II Livello
Psicologia quantitativa. Misurazione, valutazione e analisi di variabili psicosociali


22 Luglio 2022, Padova

Contenuti


- 1 DScoreApp
- 2 implicitMeasures
- 3 Letture

Cosa usiamo

- download  e seguire le istruzioni di installazione

- download  e seguire le istruzioni di installazione



-  per analizzare rapidamente e facilmente i dati (non serve installazione)
- Il pacchetto `implicitMeasures` di R

Disclaimer

Le illustrazioni seguenti assumono che sia stato usato Inquisit per raccogliere i dati

I dati che verranno usati per gli esempi sono disponibili [qui](#)

In entrambi i casi, si tratta dei dati raccolti su 142 partecipanti da uno IAT sul pregiudizio razziale

Sono state raccolte anche misure esplicite (i.e., orientamento politico e atteggiamento verso le persone Bianche e di colore)

Sia la app sia il pacchetto possono essere usati con data set ricavati da altri software

1 DScoreApp

2 implicitMeasures

3 Letture

DScoreApp è la soluzione migliore per calcolare i punteggi IAT in modo rapido e semplice

Pro

- Molto facile da usare
- Documentazione molto chiara e manintainer disponibile ad aiutare
- Si possono ispezionare i risultati durante il loro stesso calcolo
- Si può familiarizzare con la app attraverso un data set "giocattolo" interno alla app stessa

Contro

- I dati vanno preparati con software esterni (e.g., Excel)
- Si può calcolare solo un D score alla volta
- Se si vuole indagare la relazione tra misure implicite ed esplicite bisogna unire manualmente i data set su Excel (o altro)

DScoreApp

DScoreApp

DScoreApp

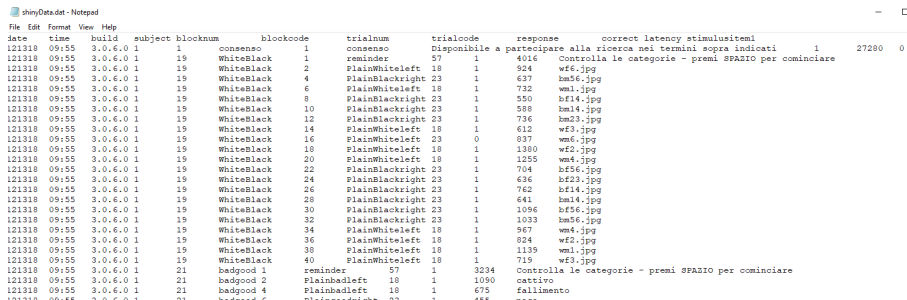
DScoreApp

Il data set deve essere salvato in **csv** e deve essere organizzato in 4 colonne, come segue:

- **participant**: Contiene gli ID dei partecipanti
- **block**: Contiene le etichette che identificano i blocchi dello IAT (pratica e test compatibile, pratica e test incompatibile)
- **latency**: Contiene i tempi di risposta
- **correct**: Contiene le risposte di accuratezza

Il data set

Da Inquisit solitamente si ottiene un file `.dat`:



Si può copia & incollare in un file Excel. Se non si ha la virgola settata di default come separatore di colonne:

Dati → *Testo in colonne* → *Delimitato* → *virgola*

Per l'esercitazione su shiny, usate [questo](#) data set

Si possono cancellare le colonne che non servono

- **date**
- **time**
- **build**

La prima colonna che ci interessa è **blockcode**. Usando la funzione filtro si possono vedere tutti i valori contenuti nella colonna

blockcode

A	B	C	D	E	F	G	H	I	J	K	L
subject	blockn	blockcode	trialnu	trialcod	respon	correct	latency	stimulu	em1		
1	consenso	Disponibil	1	27280	0						
1	reminder	57	1	4016	Controlla le categorie - premi SPAZIO						
2	PlainWhit	18	1	924	wf6.jpg						
4	PlainBlack	23	1	637	bm56.jpg						
6	PlainWhit	18	1	732	wm1.jpg						
8	PlainBlack	23	1	550	bf14.jpg						
10	PlainBlack	23	1	588	bm14.jpg						
12	PlainBlack	23	1	736	bm23.jpg						
14	PlainWhit	18	1	612	wf3.jpg						
16	PlainWhit	23	0	837	wm6.jpg						
18	PlainWhit	18	1	1380	wf2.jpg						
20	PlainWhit	18	1	1255	wm4.jpg						
22	PlainBlack	23	1	704	bf56.jpg						
24	PlainBlack	23	1	636	bf23.jpg						
26	PlainBlack	23	1	762	bf14.jpg						
28	PlainBlack	23	1	641	bm14.jpg						
30	PlainBlack	23	1	1096	bf56.jpg						
32	PlainBlack	23	1	1033	bm56.jpg						
34	PlainWhit	18	1	967	wm4.jpg						

psiColalAtdata.csv 548.3 KB May 17, 2022, 10:01 AM

blockcode

I blocchi che servono sono i blocchi “critici” dello IAT

- PracticeWhitegood
- TestWhitegood
- PracticeWhitebad
- TestWhitebad

blockcode

I blocchi che servono sono i blocchi “critici” dello IAT

- PracticeWhitegood
- TestWhitegood
- PracticeWhitebad
- TestWhitebad

White-Good/Black-Bad Condition
(MappingA)

blockcode

I blocchi che servono sono i blocchi “critici” dello IAT

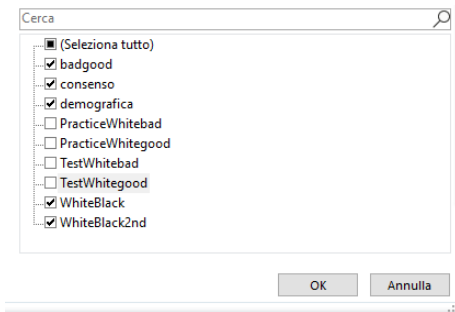
- PracticeWhitegood
- TestWhitegood
- PracticeWhitebad
- TestWhitebad

White-Good/Black-Bad Condition
(MappingA)

Black-Good/White-Bad Condition
(MappingB)

Si selezionano i blocchi di interesse...Eliminadndo tutti gli altri!

Dal filtro sulla colonna **blockcode**, si selezionano tutti i blocchi ~~tranne~~
PracticeWhitegood, TestWhitegood, PracticeWhitebad,
TestWhitebad:



blockcode

Evidenziare e cancellare tutte le righe che rimangono **dopo** l'applicazione del filtro

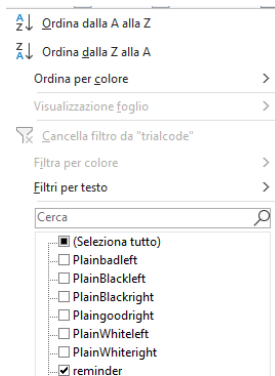
Vedrete sparire tutte le righe... Niente panico!

Togliete il filtro dalla colonna **blockcode**

trialcode

Dopo avere selezionato i blocchi che ci interessano, ci potrebbero ancora essere dei trial che sono parte dell'esperimento ma che non servono per il calcolo del D score (e.g., reminders, instructions)

Mettete un filtro sulla colonna **trialcode** e selezionate tutti i trial non rilevanti:



trialcode

Selezionate tutte le righe che sono rimaste **dopo** che avete applicato il filtro

Cancellatele

(Niente panico)

Togliete il filtro dalla colonna **trialcode**

Ultimi ritocchi al dataset

Togliete ogni filtro rimasto dalle colonne

Togliete tutte le colonne non necessarie (`blocknum`, `trialnum`, `trialcode`, `response`, `stimulusitem1`)

Rinominate tutte le colonne rimaste a seconda delle istruzioni della app:

- `subject` → `participant`
- `blockcode` → `block`
- `latency` → `latency`
- `correct` → `correct`

Il look finale

	A	B	C	D	E
1	participant	block	correct	latency	
2	1	PracticeWhitebad	1	725	
3	1	PracticeWhitebad	1	1052	
4	1	PracticeWhitebad	1	1517	
5	1	PracticeWhitebad	1	767	
6	1	PracticeWhitebad	1	985	
7	1	PracticeWhitebad	1	708	
8	1	PracticeWhitebad	1	689	
9	1	PracticeWhitebad	1	719	
10	1	PracticeWhitebad	1	550	
11	1	PracticeWhitebad	1	1101	
12	1	PracticeWhitebad	1	918	
13	1	PracticeWhitebad	1	812	
14	1	PracticeWhitebad	1	717	
15	1	PracticeWhitebad	1	1028	
16	1	PracticeWhitebad	1	823	
17	1	PracticeWhitebad	1	843	
18	1	PracticeWhitebad	1	764	
19	1	PracticeWhitebad	1	651	
20	1	PracticeWhitebad	1	1076	

Il file **deve** essere salvato in `.csv` con la virgola settata come separatore di colonna

La virgola come separatore di colonna è un dettaglio estremamente importante perché altrimenti la app non funziona ma soprattutto non vi dirà perché non funziona

Importare il data set

The screenshot shows the DScoreApp web interface. On the left, there's a sidebar with the title 'DscoreApp'. Below it, there's a section for 'Example data' with a checkbox for 'Raw IAT dataset'. Underneath, it says 'Choose CSV file' and there's a file selection area with a 'Choose...' button and a message 'No file selected'. Below this, there are two mapping sections: 'Mapping A Practice block label' and 'Mapping A Test block label', each with a dropdown menu and an example value 'e.g. practiceEfficientGood' and 'e.g. testEfficientGood' respectively. There are also 'Mapping B' sections for 'Practice block label' and 'Test block label' with similar dropdowns and examples. At the bottom of the sidebar, there's a 'Previous Data' button and a 'Show info' link. The main content area has three tabs: 'Read Me First', 'D- Score results', and 'Descriptive Statistics'. The 'D- Score results' tab is active, showing a list of links: 'THE D-SCORE SHINY APP', 'IMPORT DATA', 'HOW IT WORKS', 'THE D-SCORE RESULTS PANEL', 'DESCRIPTIVE STATISTICS PANEL', 'WHAT YOU GET', 'REFERENCES', 'CONTACTS', and 'LICENSE'. At the bottom of the main content area, it says 'WAITING FOR DATA'.

Cercate il file nel vostro computer e selezionatelo. Verrò caricato automaticamente

Preparete il dataset (sì, di nuovo)

Example data

☐ Race IAT dataset

Choose CSV file

Browse... dataset.csv

MappingA Practice block label
e.g. practiceWhiteGood
PracticeWhitebad

MappingA Test block label
e.g. testWhiteGood
TestWhitebad

MappingB Practice block label
e.g. practiceWhiteBad
PracticeWhitegood

MappingB Test block label
e.g. testWhiteBad
TestWhitegood

Prepare Data Show info

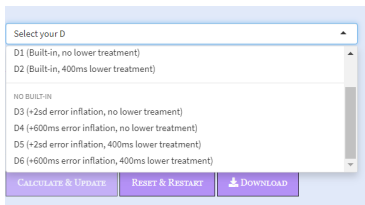
WAITING FOR DATA

Per cambiare l'ordine con cui viene calcolato il D score (i.e., $M(A) - M(B)$ vs. $M(B) - M(A)$) → selezionate le etichette corrispondenti all'ordine che volete seguire

Una volta selezionate le etichette desiderate → click su “Prepare data” e aspettate che appaia il messaggio “Data are ready”

Selezionate il D score

Selezionate il D score che volete calcolare dal drop down menu, click su “Calculate & Update”... ed è fatta! I D score dei partecipanti appariranno a breve nel “Results panel”

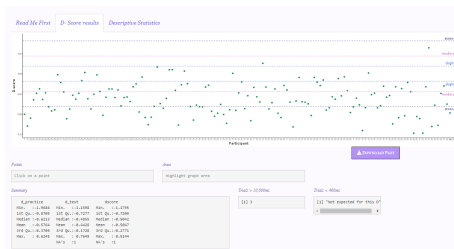


Select your D

- D1 (Built-in, no lower treatment)
- D2 (Built-in, 400ms lower treatment)
- NO BUILT-IN
- D3 (+2sd error inflation, no lower treatment)
- D4 (+600ms error inflation, no lower treatment)
- D5 (+2sd error inflation, 400ms lower treatment)
- D6 (+600ms error inflation, 400ms lower treatment)

CALCULATE & UPDATE RESET & RESTART DOWNLOAD

(default) Results panel



Divertitevi con le rappresentazioni grafiche e le impostazioni

Accuracy deletion

☒ No

☐ Yes (Practice + Test blocks)

Fast participants deletion

☒ No

☐ Yes

Graphic display

☒ Points

☐ Histogram

☐ Density

☐ Histogram + Density

Point Graph

None ▼

Note: Please, read the READ ME FIRST before doing anything

[CALCULATE & UPDATE](#) [RESET & RESTART](#) [DOWNLOAD](#)

Download

Una volta finito → Potete scaricare i risultati in un file `.csv` (Il file ha la virgola come separatore di colonna)

Il nome del file contiene l'etichetta dell'ultimo D score calcolato

Ad esempio, se $D3$ è l'ultimo algoritmo che è stato calcolato, il nome del file sarà: `ShinyAPPDscore3.csv`

1 DScoreApp

2 **implicitMeasures**

3 Letture

implicitMeasures

Pro

- Molto facile da usare (se sapete usare R =))
- Ben documentato e la maintainer è sempre disponibile
- Si possono calcolare diversi D scores insieme
- Calcola il D score anche per il SC-IAT
- Il calcolo del D score ed eventuali altre analisi avvengono tutte nello stesso posto

Contro

- Richiede una medio-buona conoscenza di R

Install & Upload

Installare il pacchetto:

```
> install.packages("implicitMeasures")
```

Caricare il pacchetto:

```
> library(implicitMeasures)
```

Siete pronti/e

Importare il data set

Bisogna importare il file `.dat` ottenuto da Inquisit (il file è disponibile [qui](#)):

```
> data = read.table("IATdata.dat", header=TRUE, sep = "\t")
> head(data)
```

	date	time	build	subject	blocknum	blockcode	trialnum	tr
1	121318	09:55	3.0.6.0	1	1	consenso	1	
2	121318	09:55	3.0.6.0	1	19	WhiteBlack	1	
3	121318	09:55	3.0.6.0	1	19	WhiteBlack	2	PlainW
4	121318	09:55	3.0.6.0	1	19	WhiteBlack	4	PlainBl
5	121318	09:55	3.0.6.0	1	19	WhiteBlack	6	PlainW
6	121318	09:55	3.0.6.0	1	19	WhiteBlack	8	PlainBl

	response
1	Disponibile a partecipare alla ricerca nei termini sopra indicati
2	57
3	18
4	23
5	18

blockcode

```
> table(data$blockcode)
```

badgood	consenso	demografica	PracticeWhite
5985	147	852	2
PracticeWhitegood	TestWhitebad	TestWhitegood	WhiteB
3003	5822	5863	3
WhiteBlack2nd			
3003			

We have a lot of stuff to get rid of. . . .

trialcode

```
> table(data$trialcode)
```

consenso	edu	età	occupazio	
147	142	142	142	
PlainBlackleft	PlainBlackright	Plaingoodright	PlainWhiteleft	Pla
3575	3561	7125	3560	
pol1	pol2	reminder	reminder1	
142	142	857	285	

Preparare il data set

```
> data_clean = clean_iat(  
+   data,                                # nome del data set  
+   sbj_id = "subject", # colonna con gli ID dei soggetti  
+   block_id = "blockcode", # Colonna con le etichette dei blocchi  
+   mapA_practice = "PracticeWhitegood",  
+   mapA_test = "TestWhitegood",  
+   mapB_practice = "PracticeWhitebad",  
+   mapB_test = "TestWhitebad",  
+   latency_id = "latency",  # colonna delle latenze  
+   accuracy_id = "correct", # colonna delle accuratezze  
+   trial_id = "trialcode",  # colonna con le etichette dei trial  
+   trial_eliminate = c("reminder", "reminder1"), # trial da eliminare  
+   demo_id = "blockcode",   # colonna con le etichette dei blocchi  
+   trial_demo = "demografica" # etichette dei trial demografica  
+ )
```

Cosa contiene `data_clean`?

```
> names(data_clean)
```

```
[1] "data_keep"      "data_eliminate" "demo"
```

- `data_keep`: il data set su cui viene calcolato il D (con classe `data.frame`, `iat_clean`)
- `data_eliminate`: I trial che sono stati scartati
- `demo`: Il data set che contiene le informazioni socio-demografiche

Cosa contiene `data_clean`?

```
> names(data_clean)
```

```
[1] "data_keep"      "data_eliminate" "demo"
```

- `data_keep`: il data set su cui viene calcolato il D (con classe `data.frame`, `iat_clean`)
- `data_eliminate`: I trial che sono stati scartati
- `demo`: Il data set che contiene le informazioni socio-demografiche

Se esportate l'oggetto `data_keep` in `.csv`, lo potete usare in `DScoreApp`!

```
> write.table(data_clean[[1]], "cleanIAT.csv",
>               sep = ",",          row.names = FALSE)
```

Calcolare il D score

```
> iat = data_clean[[1]] # selezionare il data set pulito
```

usando la funzione `compute_iat()` e specificando l'algoritmo desiderato:

```
> d3 = compute_iat(iat, # il data set pulito
+                  Dscore = "d3") # l'algoritmo desiderato
```

```
> head(d3[, 1:5]) # prime 5 colonne
```

	participant	n_trial	nslow10000	nfast400	nfast300
1	1	120	0	0.01	0
2	2	120	0	0.03	0
3	3	120	0	0.14	0
4	4	120	0	0.07	0
5	5	120	0	0.00	0
6	6	120	0	0.00	0

```
> head(d3[, 6:10]) # colonna da 6 a 10
```

	accuracy.practice_MappingA	accuracy.practice_MappingB	accuracy.test_MappingA	accuracy.test_MappingB	accuracy.MappingA	accuracy.MappingB
1	1.00	1.00	0.950	0.9833333	0.9833333	0.9833333
2	1.00	0.95	1.000	0.9833333	0.9833333	0.9833333
3	0.95	0.75	0.900	0.9000000	0.9000000	0.9000000
4	1.00	0.95	0.950	0.9666667	0.9666667	0.9666667
5	0.95	1.00	0.925	0.9333333	0.9333333	0.9333333
6	0.95	0.90	0.975	0.9166667	0.9166667	0.9166667

```
> head(d3[, 11:15]) # colonna da 11 a 15
```

	accuracy.MappingB	RT_mean.MappingA	RT_mean.MappingB	mean_practice
1	0.9666667	597.6649	738.7075	
2	0.9833333	598.3324	649.2085	
3	0.8500000	575.2006	721.7637	
4	0.9500000	606.2957	645.4930	
5	0.9500000	849.8184	1011.9773	
6	0.9500000	914.6978	981.7482	

	mean_test_MappingA
1	585.7973
2	589.5736
3	585.2044
4	618.3685
5	695.3284
6	911.1501

```
> head(d3[, 16:19]) # colonna da 16 a 19
```

	mean_practice_MappingB	mean_test_MappingB	d_practice_d3	d_test_c
1	851.5500	682.2862	1.00062841	0.608255
2	840.5754	553.5250	0.62762704	-0.231817
3	973.0089	596.1411	1.16298569	0.055042
4	746.2660	595.1065	0.70572685	-0.162865
5	1134.5500	950.6909	-0.04599615	0.691086
6	1152.3929	896.4259	0.50668795	-0.034255


```
> head(d3[, 20:21]) # colonna 20 e 21
```

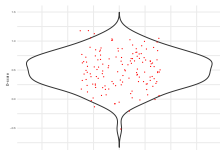
	dscore_d3	cond_ord
1	0.8044421	MappingB_First
2	0.1979047	MappingA_First
3	0.6090143	MappingB_First
4	0.2714307	MappingA_First
5	0.3225454	MappingA_First
6	0.2362163	MappingB_First

```
> head(d3[, 22:23]) # colonna 22 e 23
```

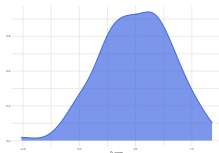
	legendMappingA	legendMappingB
1	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood
2	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood
3	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood
4	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood
5	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood
6	PracticeWhitegood_and_TestWhitegood	PracticeWhitebad_and_TestWhitegood

Distribuzioni

```
> d_density(d3,  
+           graph =  
+           ↪ "violin")
```

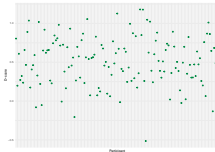


```
> d_density(d3,  
+           graph =  
+           ↪ "density")
```

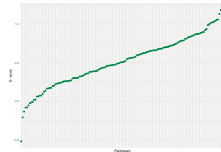


Punti

```
> d_point(d3, x_values =  
  ↪ FALSE)
```



```
> d_point(d3, x_values =  
  ↪ FALSE,  
+         order_sbj =  
  ↪ "D-increasing")
```



Diversi algoritmi allo stesso tempo

```
> dscores = multi_dscore(iat, # data set pulito  
+                          ds = "error-inflation") # quali  
↪ algoritmi
```

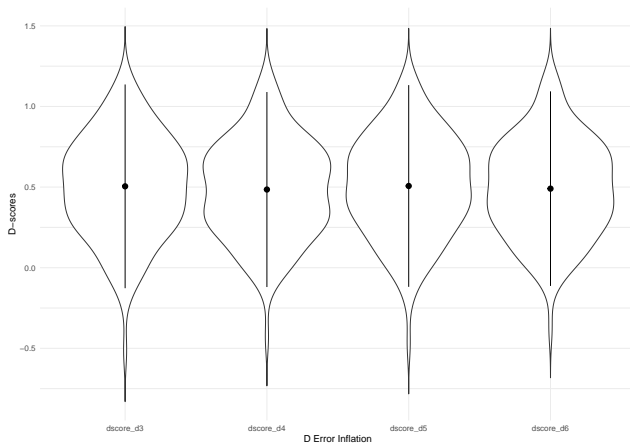
Attenzione!

```
> names(dscores)  
  
[1] "dscores" "graph"
```

```
> head(dscores[[1]])
```

	participant	dscore_d3	dscore_d4	dscore_d5	dscore_d6
1	1	0.8044421	0.7664824	0.7902660	0.75162963
2	2	0.1979047	0.1807094	0.1081013	0.09710327
3	3	0.6090143	0.5433544	0.6836311	0.65357652
4	4	0.2714307	0.2849834	0.4108129	0.43049456
5	5	0.3225454	0.3177612	0.3225454	0.31776123
6	6	0.2362163	0.2714998	0.2362163	0.27149977

```
> dscores[[2]]
```



Ci siamo dimenticati di demo...?

```
> demo_raw = data_clean[[3]] # data set con le info. demografiche
> str(demo_raw)

'data.frame':   852 obs. of  12 variables:
 $ date          : int   121318 121318 121318 121318 121318 121318 121318 121318 121318 121318 121318 121318 ...
 $ time          : chr    "09:55" "09:55" "09:55" "09:55" ...
 $ build         : chr    "3.0.6.0" "3.0.6.0" "3.0.6.0" "3.0.6.0" ...
 $ participant   : int    1 1 1 1 1 1 2 2 2 2 ...
 $ blocknum      : int    57 57 57 57 57 57 57 57 57 57 ...
 $ blockcode     : chr    "demografica" "demografica" "demografica" "demografica" ...
 $ trialnum      : int    1 1 1 1 4 4 1 1 1 1 ...
 $ trialcode     : chr    "sesso" "età" "occupazio" "edu" ...
 $ response      : chr    "Maschio" "21" "stud" "sup" ...
 $ correct       : int    1 1 1 1 1 1 1 1 1 1 ...
 $ latency       : int    19185 19185 19185 19185 28866 28866 24586 24586 ...
 $ stimulusitem1: chr    "0" "0" "0" "0" ...
....
```


Selezioniamo solo le colonne che servono:

```
> demo_raw = demo_raw[, c("participant", "trialcode",  
+                           "response")]  
> str(demo_raw)
```

```
'data.frame':   852 obs. of   3 variables:  
 $ participant: int   1 1 1 1 1 1 2 2 2 2 ...  
 $ trialcode  : chr   "sesso" "età" "occupazio" "edu" ...  
 $ response   : chr   "Maschio" "21" "stud" "sup" ...
```

“Giriamo” il data set

In modo che soggetto sia su una riga singola:

```
> demo <- reshape(demo_raw,
+                 timevar = "trialcode",
+                 idvar = "participant",
+                 direction = "wide")
> str(demo)
```

```
'data.frame':  142 obs. of  7 variables:
 $ participant      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ response.sesso   : chr  "Maschio" "Maschio" "Femmina" "Femmina" ...
 $ response.età     : chr  "21" "31" "21" "21" ...
 $ response.occupazio: chr  "stud" "stud" "stud" "stud" ...
 $ response.edu     : chr  "sup" "magistrale" "sup" "sup" ...
 $ response.pol1    : chr  "3" "3" "3" "2" ...
 $ response.pol2    : chr  "2" "5" "2" "3" ...
 . . . .
```

Sistemiamo il dataset

```
> colnames(demo) <- gsub("response.", '', colnames(demo))
> demo[, c(3, 6:7)] <- apply(demo[, c(3, 6:7)], 2, as.integer)
> str(demo)
```

```
'data.frame':  142 obs. of  7 variables:
 $ participant: int  1 2 3 4 5 6 7 8 9 10 ...
 $ sesso      : chr  "Maschio" "Maschio" "Femmina" "Femmina" ...
 $ età       : int  21 31 21 21 21 20 20 19 20 20 ...
 $ occupazio  : chr  "stud" "stud" "stud" "stud" ...
 $ edu       : chr  "sup" "magistrale" "sup" "sup" ...
 $ pol1      : int  3 3 3 2 4 3 2 3 3 3 ...
 $ pol2      : int  2 5 2 3 4 2 3 2 1 2 ...
....
```

Qualche info

- participant: ID dei partecipanti
- sesso: Sesso dei partecipanti
- età: Età
- occupazio: Occupazione (stud, stud/lav, lavD, lavA, dis)
- edu: Istruzione (sup, magistrale, triennale, dottorato)
- pol1: Atteggiamento verso le persone Bianche/di colore (1 = *Preferisco molto le persone bianche rispetto alle persone di colore*, 6=*Preferisco molto le persone nere rispetto alle persone bianche*)
- pol2: Orientamento politico (1 - *Left wing orientation*, 6 -*Right wing orientation*)

Unire demo with d3

```
> d3complete = merge(d3, # data set con i D score
+                     demo, # data set con le info. demografiche
+                     by = "participant") # id della variabile per
↪ unire
> str(d3complete[17:29])

'data.frame':  142 obs. of  13 variables:
 $ mean_test_MappingB: num  682 554 596 595 951 ...
 $ d_practice_d3      : num  1.001 0.628 1.163 0.706 -0.046 ...
 $ d_test_d3          : num  0.608 -0.232 0.055 -0.163 0.691 ...
 $ dscore_d3          : num  0.804 0.198 0.609 0.271 0.323 ...
 $ cond_ord           : chr   "MappingB_First" "MappingA_First" "MappingB_Second" ...
 $ legendMappingA     : chr   "PracticeWhitegood_and_TestWhitegood" "PracticeWhitebad_and_TestWhitebad" ...
 $ legendMappingB     : chr   "PracticeWhitebad_and_TestWhitebad" "PracticeWhitegood_and_TestWhitegood" ...
 $ sesso              : chr   "Maschio" "Maschio" "Femmina" "Femmina" ...
 $ età                : int   21 31 21 21 21 20 20 19 20 20 ...
 $ occupazio          : chr   "stud" "stud" "stud" "stud" ...
 $ edu                : chr   "sup" "magistrale" "sup" "sup" ...
 $ m1                 : int   2 2 2 2 2 4 2 2 2 2 ...
```

Calcolo delle correlazioni

```
> correlazioni <- data.frame(cor(d3complete[,
+                               c("dscore_d3",
+                               ↪ "pol1", "pol2"))))
> correlazioni <- round(correlazioni, 2)
> correlazioni[upper.tri(correlazioni, diag = TRUE)] <- ""
```

Results are in Table 1:

Table 1: Race IAT correlations

	dscore_d3	pol1	pol2
dscore_d3			
pol1	0		
pol2	0.12	-0.34	

Significatività delle correlazioni

```
> cor.test(~ pol1 + dscore_d3, data = d3complete)
```

Pearson's product-moment correlation

data: pol1 and dscore_d3

t = 0.03199, df = 140, p-value = 0.9745

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

-0.1620959 0.1673563

sample estimates:

cor

0.002703599

1 DScoreApp

2 implicitMeasures

3 Lecture

DScoreApp:

Epifania, O. M., Anselmi, P., & Robusto, E. (2020). Dscoreapp: A shiny web application for the computation of the implicit association test D score. Frontiers in Psychology, 10, 2938. doi: 10.3389/fpsyg.2019.02938

implicitMeasures:

Epifania, O. M., Anselmi, P., & Robusto, E. (2020). Implicit measures with reproducible results: The implicitmeasures package. Journal of Open Source Software, 5(52), 2394. doi: 10.21105/joss.02394