R_script per formati da untidy a tidy (con spiegazione dei comandi come Pivot_longer)

FORMATO TIDY -> VARIABILI NELLE COLONNE OSSERVAZIONI NELLE RIGHE
Formato untidy –> variabili multivalore nelle celle oppure il contrario dei dati tidy

```
rm(list=ls())
library(tidyverse)
setwd("~/Desktop/tutto BDA/R scripts, data, and exercise n2")
docenti_wide <- read.csv("istat_docenti_200607.csv")

#passare da un formato untidy ad un formato tidy

docenti_ordinati <- docenti_wide %>%
  pivot_longer(cols = c("Ordinari","Associati","Ricercatori"))
docenti_ordinati

#passare da una tabella untidy a tidy tramite il secondo metodo che prevede mantenere la colonna che presenta le variabili in forma ordinata nel nostro caso Facolta e utilizzeremo il pivotlonger per ordinare tutte le variabili meno quella che è gia ordinata,il risultato sarà lo stesso

docenti_ordinati_2metodo <- docenti_wide %>%
  pivot_longer(cols=-Facolta)

#in seguito all'ordinamento notiamo che nella di default il programma da come nome alle colonne “name", "value", se volessimo rinominare che comandi utilizzeremmo?

docenti_ordinati3 <- docenti_wide%>%
  pivot_longer(cols=-Facolta,names_to = "Ruolo",values_to = "Numero")

#possiamo ragionare in maniera più compatta quindi "smart" raggruppando le tre idee in un unico comando

docenti_ordinati4 <- docenti_wide %>%
  pivot_longer(cols=-Facolta, names_to = "Ruolo",values_to = “Numero")
```

#parte 2 comando pivot_wider

Serve a trasformare le tabelle che sono già in un formato Tidy in un formato Untidy più comprensibile per una determinata analisi

#comando pivot_wider ovvero trasformare tabelle tidy in tabelle con struttura più ampia che lunga

```
rm(list=ls())
library(readr)
istat_imprese_long <- read_csv("istat_imprese_innovazione_2016.csv")
```

#come posso ora renderla una tabella più facile da interpretare? utilizzo il comando pivot wider
#modo1

```
imprese_wide1 <- istat_imprese_long %>%
  pivot_wider(names_from = Innovazione, values_from = Imprese)
```

#supponiamo di voler rinominare alcune di quelle variabili che ora abbiamo tutte in una riga per ragioni di comodità, per rendere la forma ancora più compatta
#utilizziamo il comando mutate (crea nuove colonne partendo dalle variabili esistenti)
#utilizziamo all'interno di mutate la funzione case_when(immagina di avere una tabella con migliaia di righe e di voler rinominare il valore delle variabili su più righe contemporaneamente)

```
imprese_wide2 <- istat_imprese_long %>%
  mutate(
    Innovazione = case_when(
    Innovazione == "Solo di prodotto/processo"~"prod_proc",
    Innovazione  == "Solo organizzativo/di marketing"~"org_mkt",
    Innovazione == "Di prodotto/processo e organizzative/di
marketing"~"tutti_i_tipi")) %>%
  pivot_wider(names_from = Innovazione, values_from = Imprese)
```

#oppure un'altro modo di operare è il seguente per rinominare

```
Imprese_wide3 <- istat_imprese_long %>%
  pivot_wider(names_from=Innovazione, values_from = Imprese)%>%
  rename(add = Addetti,
         prod_proc='Solo di prodotto/processo',
         org_mtk= 'Solo organizzativo/di marketing',
         tutti_tipi ='Di prodotto/processo e organizzative/di marketing')
```

FUNZIONI SEPARATE,UNITE,SEPARATE_ROWS
#funzioni separate(), unite(), separate_rows()------
#carico il file

```
rm(list=ls())
covid_regioni1 <-read_csv("salute_covid_report27.csv")
```

#funzione separate () spacchettare le informazioni all'interno di una cella

```
covid_regioni2 <- covid_regioni1 %>%
  separate('Ind3.2 (Rt puntuale)' , c("Rt","CI","SupInf"), sep =" “)%>%
```

#2°separazione della stessa cella

```
  separate( 'SupInf',c('Inferiore','Superiore'),sep = "-") %>%
```

#cancellazione della parentesi “)” nella cella di nostro interesse tramite la funzione gsub

```
mutate(Superiore = gsub(")","",Superiore))%>%
  select(-CI)
```

RICORDA* gsub è una funzione usata quando vogliamo dire che un simbolo, numero o altro deve essere cancellato perché un di più
Nel nostro caso avevamo bisogno di cancellare la parentesi

#funzione unite(nome della cella in cui vogliamo unire i dati, nome dei dati che vogliamo unire, sep=‘separatore che si vuole utilizzare ‘,
Ricorda per effettuare una concatenazione bisogna mettere il simbolo%>% alla fine di ogni stringa

```
covid_regioni3 <- covid_regioni2%>%
  unite(CI,Inferiore,Superiore,sep = '-')%>%
  unite(Rt_CI,Rt,CI, sep = '--- CI:')
```

#funzione separate_rows in questo esempio abbiamo un dato untidy che presenta valori multipli nelle celle abbiamo quindi necessità di separare questi valori all'interno delle celle per avere un dato tidy che ci permetta di fare osservazioni
#carichiamo il dato che ci interessa

```
rm(list=ls())
progetto <- read_csv('proj_org.csv')
```

#notiamo subito che la tabella partners presenta dati multivalore nelle celle divisi da ";", dobbiamo separarli utilizzando il comanndo separate rows

```
Progetto2 <- progetto%>%
  separate_rows(Partners,sep=';')
```

#abbiamo ottenuto una tabella con i valori ordinati quindi non più untidy ma tidy

Esercizi su pivot_longer e pivot_wider

#esercizio1 (istat_docenti_200607.csv)
#passare da un formato untidy ad un formato tidy secondo i tre metodi
#rinominare i nomi dei valori dele colonne name e value
#eseguire la forma compatta dei tre metodi

```
rm(list=ls())
library(readr)
setwd("~/Desktop/tutto BDA/R scripts, data, and exercise n2")
tabella_docenti1 <- read_csv("istat_docenti_200607.csv")
tabella_docenti1
#passare da un formato untidy a tidy utilizzo la funzione pivot_longer
tabella_docenti_longer1 <- tabella_docenti1%>%
  pivot_longer(cols=c("Ordinari","Associati","Ricercatori"))
#passare da un formato untidy a tidy secondo metodo

tabella_docenti_longer2 <- tabella_docenti1 %>%
  pivot_longer(cols=-Facolta)
#rinomino colonne tramite la funzione mutate case_when
tabella_docenti_longer2 <- tabella_docenti1%>%
  pivot_longer(cols=-Facolta, names_to = "Ruolo",values_to = "Totale")
```

#esercizio2 pivot winder(istat_imprese_innovazione_2016.csv)
#passare da un formato longer ad un formato wider
#rinominare le celle in maniera tale da ottenere una formulazione più compatta secondo i due metodi

```
rm(list=ls())
library(readr)
tabella_imprese1 <- read_csv("istat_imprese_innovazione_2016.csv")
tabella_imprese1

#passare da un formato longer ad un formato wider

tabella_imprese2 <- tabella_imprese1 %>%
  pivot_wider(names_from = Innovazione,values_from = Imprese)

#rinominare le celle in maniera tale da ottenere un formato più

compatto
tabella_imprese2 <- tabella_imprese1 %>%
  mutate(
    Innovazione = case_when(
      Innovazione == "Solo di prodotto/processo" ~ "prod.proc",
      Innovazione == "Solo organizzativo/di marketing" ~ "org.mktg",
      Innovazione == "Di prodotto/processo e organizzative/di marketing" ~
"vari.tipi"))%>%
pivot_wider(names_from = Innovazione,values_from = Imprese)

#altro tipo di rinominazione (rename)

tabella_imprese3 <- tabella_imprese1%>%
          pivot_wider(names_fromInnovazione,values_from=Imprese%>%
  rename(add= "Addetti",
         prod_proc = "Solo di prodotto/processo",
         org_mkt ="Solo organizzativo/di marketing",
         vari_tipi = "Di prodotto/processo e organizzative/di marketing"


#lezione 4 online
#cerchiamo di utilizzare un po' meglio altri comandi che abbiamo già
usato in maniera intuitiva in precedenza
#carichiamo nuovamente il file csv istat-imprese_innovazione_2016.csv

rm(list=ls())

imprese <- read_csv("istat_imprese_innovazione_2016.csv")%>%
  pivot_wider(names_from = Innovazione, values_from = Imprese)

#funzione rename
```

```
imprese <- imprese%>%
  rename(Add = 'Addetti',
         prod_proc ='Solo di prodotto/processo',
         org_mkt = 'Solo organizzativo/di marketing',
         tutti_i_tipi ='Di prodotto/processo e organizzative/di marketing')

#funzione mutate, vogliamo colcolare la proporzione di firme per ogni categoria

imprese <- imprese%>%
  mutate(total_firms=prod_proc + org_mkt + tutti_i_tipi)%>%
  mutate(prod_proc_perc = prod_proc/total_firms,
         org_mkt_perc= org_mkt/total_firms,
         tutti_i_tipi_perc= tutti_i_tipi/total_firms)
#utilizzando il comando mutate abbiamo aggiunto quattro colonne, una in cui avevamo il valore 'total_firms'
#le altre tre sono  i valori percentuali di ogni categoria/tota_firms
#-----------------------------------------------------------------------
#supponiamo ora che dopo aver ottenuto questa tabella non ci servano le prime tre colonne, ma vogliamo presentare solo una tabella con le percentuali
#utilizziamo il comando select
 imprese <- imprese%>%
  select(-prod_proc,-org_mkt,-tutti_i_tipi,-total_firms)

#un'altro modo pre ottenere lo stesso risultato è
imprese <- imprese%>%
  select(Add, ends_with("_perc"))

#-------------------------------------------------
```

#funzioni left_join(), right_join(), inner_join(), full_join()

#supponiamo di leggere due file Scoreboard18 e scorerboard19

```
rm(list=ls())
scoreboard18 <- readxl::read_xlsx('scorebaord_2018.xlsx')
scoreboard19 <- readxl:: read_xlsx('scorebaord_2019.xlsx')

#queste sono le imprese che hanno investito di più in imprese e sviluppo 18/19, seleziono ora solo le colonne che mi servono
```

```
scoreboard18_tidy <- scoreboard18%>%
  select(Company,"R&D intensity (%)",Employees, `World rank`)%>%

#rinominiamo anche i nomi delle colonne della nuova tabella

 rename(Com_= Company,
        R_int_perc_18= 'R&D intensity (%)',
        emp_18 = Employees,
        WR_18 ='World rank')

#facciamo lo stesso per l'altra tabella

scoreboard19_tidy <- scoreboard19%>%
  select(Company,'R&D intensity (%)', Employees,`World rank`)%>%
  rename(Com_=Company,
        R_int_perc_19 = "R&D intensity (%)",
        emp_19= Employees,
        WR_19= 'World rank')

#creiamo ora un nuovo oggetto che unisca tutti i dati delle due tabelle

scoreboard1819_full <- scoreboard18_tidy%>%
  full_join(., scoreboard19_tidy, by = "Com_")

#il punto nella funzione full_join serve a mantenere l'elemento che
abbiamo selezionato prim per fare match con quello che c'è scritto
dopo ovvero Scoreboard19_tidy il risultato sarà una tabella unita
ordinata by com_
#posso ottenere lo stesso risultato anche in questo modo

scoreboard1819_full2 <-
full_join(scoreboard18_tidy,scoreboard19_tidy,by="Com_")

#innerjoin(identifica le righe  secondo la colonna by, nel nostro caso
'Com_ 'che sono comuni alle due tabelle)

scoreboard1819_inner <- scoreboard18_tidy%>%
  inner_join(.,scoreboard19_tidy, by='Com_')

#avremo ora le imprese comuni nei due anni
#possiamo anche calcolare in una nuova colonna quanto hanno perso le
aziende tra il 2018 ed il 2019
```

```
scoreboard1819_inner <- inner_join(scoreboard18_tidy, scoreboard19_tidy,
by="Com_")%>%
```

Filter(), Arrange(),summarise(), group_by()

```
rm(list=ls())
empl <-  read_csv("istat_employment_rate_1992_2019.csv")
empl1 <- empl%>%
  separate(TIME,c("year","quarter"), sep="-")%>%
  filter(is.na(quarter))%>%
  select(Gender,`Age class`,year,Value,Territory)%>%
  filter(Gender != "total")%>%
  filter(`Age class` == "20-64 years")
```

-Attraverso il comando separate abbiamo diviso la colonna time senza le Q
-In seguito abbiamo filtrato solo le colonne che avevano la sigla NA nella colonna quarter in quanto interessati solo ai valori interi
-Abbiamo poi selezionato solo le colonne di nostro interesse tramite il select
-infine abbiamo filtrato i dati in gender e age class interessandoci di più ai maschi e femmine e agli anno dai 20 ai 64

```
empl_puglia <- empl1%>%
  filter(Territory == “Puglia")
```

Abbiamo poi effettuato un altro filtro per capire la situazione della puglia

```
empl_puglia1 <- empl_puglia%>%
  arrange(year,Gender)
```

Attraverso il comando arrange abbiamo disposto i dati per anno e genere

Abbiamo poi effettuato delle operazioni di filtraggio dei dati sul dataset empi Puglia
E attraverso summarise abbiamo creato una nuova colonna con la media dei valori del 2016

```
empl_puglia1 <- empl_puglia%>%
  arrange(year,Gender)
empl_puglia2 <- empl_puglia%>%
  filter(year == 2016)%>%
  summarise(mean_tot2016 = mean(Value))
```

In quest'altro caso abbiamo aggiunto anche l'operazione di deviazione standard nel comando summarise

```
empl_puglia %>%
  filter(year > 2010) %>%
  group_by(Gender) %>%
  summarise(mean_gender_age = mean(Value),
            sd_gender_age   = sd(Value))
```

Abbiamo fatto operazioni simili in questo caso creando un nuovo dataset empl_summary filtrato i dati superiori al 2010,raggruppato per genere e territorio fatto i nostri calcoli statistici e ordinandoli per territorio attraverso arrange .

```
empl_summary <- empl1 %>%
  filter(year > 2010) %>%
  group_by(Gender, Territory) %>%
  summarise(mean_gender_age = mean(Value),
            sd_gender_age   = sd(Value)) %>%
  arrange(Territory)
```