1. **## Aufgabe**  
   Schreiben Sie eine Funktion, mit folgendem Algorithmus, wobei ein beliebiger Vektor als Eingabe verlangt wird.
   1. Zähle die Anzahl verschiedener Werte.
   2. Wenn es nur zwei verschiedene Werte gibt, gebe TRUE zurück, ansonsten FALSE.

* Hinweise:
  + Wählen Sie einen treffenden Namen für Ihre Funktion (nutzen Sie am besten ein konsistentes Namensschema).
  + Wichtigster Tipp: Googeln :-)
  + Verschiedene Werte eines Vektors gibt die Funktion unique() zurück.
  + Vermutlich gibt es schon viele Lösungen (Implementierungen) für diese Funktion. Ist nur als Übung gedacht :-)
* **## Lösung**
* levels ist ein Ausdruck, der nahelegt, dass es sich um *verschiedene* Werte (“Ausprägungen” oder “Stufen”) handelt.
* Alternativ könnte man die Funktion auch so schreiben:
* Das ist expliziter, aber länger.
* Wenn man es genau nimmt, heißt *binär*, dass es nur die Werte 0 und 1 gibt. Das ist ein strengeres Kriterium, wie dass es zwei beliebigen verschiedene Werte gibt.
* Test 1:
* ## [1] TRUE
* ## [1] FALSE
* Test 2:
* Wir wenden unsere neue Funktion mit Tidyverse-Methoden an:
* ## am\_has\_two\_values mpg\_has\_two\_values  
  ## 1 TRUE FALSE
* Bonus!
* Verwenden Sie across() ({dplyr}), um alle Spalten von mtcars mit has\_two\_levels() zu überprüfen.
* ## mpg cyl disp hp drat wt qsec vs am gear  
  ## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE  
  ## carb  
  ## 1 FALSE
* Kann man auch so schreiben:
* ## mpg cyl disp hp drat wt qsec vs am gear  
  ## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE  
  ## carb  
  ## 1 FALSE

1. **## Aufgabe**  
   Schreiben Sie eine Funktion zur Berechnung der Anzahl der fehlenden Werte in einem (numerischen) Vektor!

* Hinweise:
  + Wählen Sie einen treffenden Namen für Ihre Funktion (nutzen Sie am besten ein konsistentes Namensschema).
* **## Lösung**
* Test:
* ## Error in na\_n(x2): Numeric input is needed!
* ## mpg\_na\_n  
  ## 1 1
* BONUS!
* Verwenden Sie across(), um die fehlenden Werte in allen Spalten von mtcars zu zählen.
* Wer schnell ist, der nehme gerne nycflights13::flights (aus dem Paket {nycflights13} oder per CSV-Datei aus geeigneter Stelel aus dem Internet. Meistens geht es schneller, die Daten aus einem Paket zu laden mit data(flights) nachdem man library(nycflights13) geschrieben hat).
* ## mpg cyl disp hp drat wt qsec vs am gear carb  
  ## 1 1 1 1 1 0 0 0 0 0 0 0
* Mit pivot\_longer() ist es häufig übersichtlicher bzw. besser für weitere Bearbeitungsschritte, wie das folgende Beispiel zeigt:
* ## Error: <text>:5:0: unexpected end of input  
  ## 3: pivot\_longer(everything()) %>%   
  ## 4: filter(value > 0)  
  ## ^
* flights:
* ## # A tibble: 1 × 14  
  ## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
  ## <int> <int> <int> <int> <int> <int> <int>  
  ## 1 0 0 0 8255 0 8255 8713  
  ## # … with 7 more variables: sched\_arr\_time <int>,  
  ## # arr\_delay <int>, flight <int>, air\_time <int>,  
  ## # distance <int>, hour <int>, minute <int>