VMerge *custom function* Help

Install this script from the *Google Apps Script* Gallery by using in your spreadsheet:
Tools->Script gallery...>>Miscellaneous (*Search for* **VMerge)->Install**

Type	**Function**	**Syntax**	**Description**
Filter function	Merge data vertically	VMerge(sourceArray1, sourceArray2, ...)	Merge vertically specified data ranges. All ranges must have the same width, i.e. horizontally must be the same number of cells. Returned is a range of cells of the width of the provided source arrays and the height of the combined arrays. Single cells can be merged, as well as embedded arrays, as well as the output of FILTER, QUERY, ImportRange and any other function returning ranges of data. Version 2.11: Solves an issue when all arguments are empty. Version 2.1: This version will automatically skip empty parameters, like FILTER functions that do not return a value when used like this: VMERGE(**IFERROR(**FILTER(...; ...)**)** ; **IFERROR(**FILTER(...; ...)**) ; ...)**

Examples (note in locales using the *decimal comma* the column separator in embedded arrays is a backslash, i.e. use in such locales e.g. {"Col1" \ "Col2"}):

	A	B	C	D	E	F	G	H
1	X	Y	Z	=VMerge(A1; B1; C1; "Q") ⇒	X		=VMERGE(A1:A2 ; B1:B2 ; C1:C2) ⇒	X
2	A	B	C		Y			A

3					Z			Y
4					Q			B
5								Z
6								C
7				=VMerge ({"Col1", "Col2"}; A1:B2) ⇒	Col1	Col2		
8					X	Y		
9					A	B		
10								

	A	B	C	D	E	F	G
1	A	B	C		J	K	L
2	D	E	F		M	N	O
3	G	H	I				
4				=VMerge({"Col1", 2 , TRUE }; A1:C3 ; E1: G2) ⇒	Col1	2	TRUE
5					A	B	C
6					D	E	F
7					G	H	I
8					J	K	L
9					M	N	O
10							
11				=VMerge(IfError(Filter(A1:C;A1:A="A"));IfError(Filter(A1:C;A1:A="Q"))) ⇒	A	B	C
12							

Unresolved issues (see also the unofficial test spreadsheet "Quality test"):
- VMerge (like currently all custom GAS functions) will ***hang*** when a source ranges contains certain (time related) functions, like *NOW(), RAND()*, similar functions and formulas containing such functions.
- cells containing *errors*, and *blank cells* can ***not*** be ***faithfully copied*** due to limitations in Google Apps Script.

- copied cells containing *time* information will - again due to limitations in Google Apps Script - ***receive a sub-millisecond bias***; a possible workaround is to round such copied times using a formula like:
=CEILING(<copied value> ; "00:00:00.001"*SIGN(<copied value>))

GAS source code:
Version 2.12 (*release candidate*)

```
function VMerge() { //Vertical Merge 2.12 (release candidate) - ahab facit 2010
  var maxw=0;
  var l=0;
  var minw=Number.MAX_VALUE;
  var al=arguments.length ;
  for( i=0 ; i<al ; i++){
    if( arguments[i].constructor == Array ) {
      l =arguments[i][0].length ;
      maxw=l>maxw?l:maxw; minw=l<minw?l:minw;
    }
    else if ((arguments[i] != "???") && (arguments[i].length!=0) ){
          l = 1 ;  // literal values count as array with a width of one cell, empty cells are ignored!
          maxw=l>maxw?l:maxw; minw=l<minw?l:minw;
        }
  }

  if( maxw==minw) { // when largest width equals smallest width all are equal /
    var s = new Array();
    for( i=0 ; i<al ; i++){
      if( arguments[i].constructor == Array ) s = s.concat( arguments[i].slice() )
      else if ((arguments[i] != "???") && (arguments[i].length!=0)) s = s.concat( [[arguments[i]]] )
    }
    if ( s.length == 0 ) return null ; else return s
  }
  else return "#N/A: All data ranges must be of equal width!!"
}
```

Version 2.11 (**Current version!**)

```
function VMerge() { //Vertical Merge 2.11 - ahab facit 2010
  var maxw=l=0;
  var minw=Number.MAX_VALUE;
  var al=arguments.length ;
  for( i=0 ; i<al ; i++){
    if( arguments[i].constructor == Array )l =arguments[i][0].length ;
    else if (arguments[i].length!=0) l = 1 ;  /* literal values count as array with a width of one cell, empty cells are ignored! */
    maxw=l>maxw?l:maxw;
    minw=l<minw?l:minw;
  }
  if( maxw==minw) { /* when largest width equals smallest width all are equal */
    var s = new Array();
    for( i=0 ; i<al ; i++){
      if( arguments[i].constructor == Array ) s = s.concat( arguments[i].slice() )
      else if (arguments[i].length!=0) s = s.concat( [[arguments[i]]] )
    }
  if ( s.length == 0 ) return null
   else return s
  }
  else return "#N/A: All data ranges must be of equal width!"
}
```

Version 2.1

```
function VMerge() { //Vertical Merge 2.1 - ahab facit 2010
  var maxw=l=0;
  var minw=Number.MAX_VALUE;
  var al=arguments.length ;
  for( i=0 ; i<al ; i++){
    if( arguments[i].constructor == Array )l =arguments[i][0].length ;
    else if (arguments[i].length!=0) l = 1 ;
         // literal values count as array with a width of one cell, empty cells are ignored!
    maxw=l>maxw?l:maxw;
    minw=l<minw?l:minw;
  }
  if( maxw==minw) { /* when largest width equals smallest width all are equal */
    var s = new Array();
    for( i=0 ; i<al ; i++){
      if( arguments[i].constructor == Array ) s = s.concat( arguments[i].slice() )
      else if (arguments[i].length!=0) s = s.concat( [[arguments[i]]] )
    }
  return s
  }
  else return "#N/A: All data ranges must be of equal width!"
}
```

Version 2.0

```
function VMerge() { //Vertical Merge 2.0 - ahab facit 2010
  var maxw=l=0;
  var minw=Number.MAX_VALUE;
  var al=arguments.length ;
  for( i=0 ; i<al ; i++){
    if(arguments[i].constructor == Array) l =arguments[i][0].length ;
        // literal values count as array with a width of one cell
    else l = 1 ;
    maxw=l>maxw?l:maxw;
    minw=l<minw?l:minw;
  }
 /* when largest width equals smallest width all are equal */
  if( maxw==minw) {
    var s = new Array();
    for( i=0 ; i<al ; i++){
      if( arguments[i].constructor == Array ) s = s.concat( arguments[i].slice() )
      else s = s.concat( [[arguments[i]]] )
    }
  return s
  }
  else return "#N/A: All data ranges must be of equal width!"
}
```

Version 1.0 (Pre-release version ; unreliable - because of floating point calculations - method (harmonic mean = average) to determine equal widths of all ranges)

```
function VMerge() { //Vertical Merge 1.0 - ahab facit 2010
  var l=h=a=0;
  var al=arguments.length ;
  for( i=0 ; i<al ; i++){
    l =arguments[i][0].length ;
    h += 1/l ; a += l ;
  }
  if( al/h==a/al) {
 /* when the harmonic mean (al/h) equals the average (a/al) all widths were equal */
    var s = new Array();
    for( i=0 ; i<al ; i++){
      if( arguments[i].constructor == Array ) s = s.concat( arguments[i].slice() )
      else s = s.concat( [[arguments[i].slice()]] )
    }
  return s
  }
  else return "#N/A: All data ranges must be of equal width!"
}
```