



PRAHEALTHSCIENCES

How to Write SAS Code Using Excel

Hansjörg Frenzel



- What do we want to accomplish?
- Why bother with VBA?
- How to use VBA?
- Demo
- Conclusions



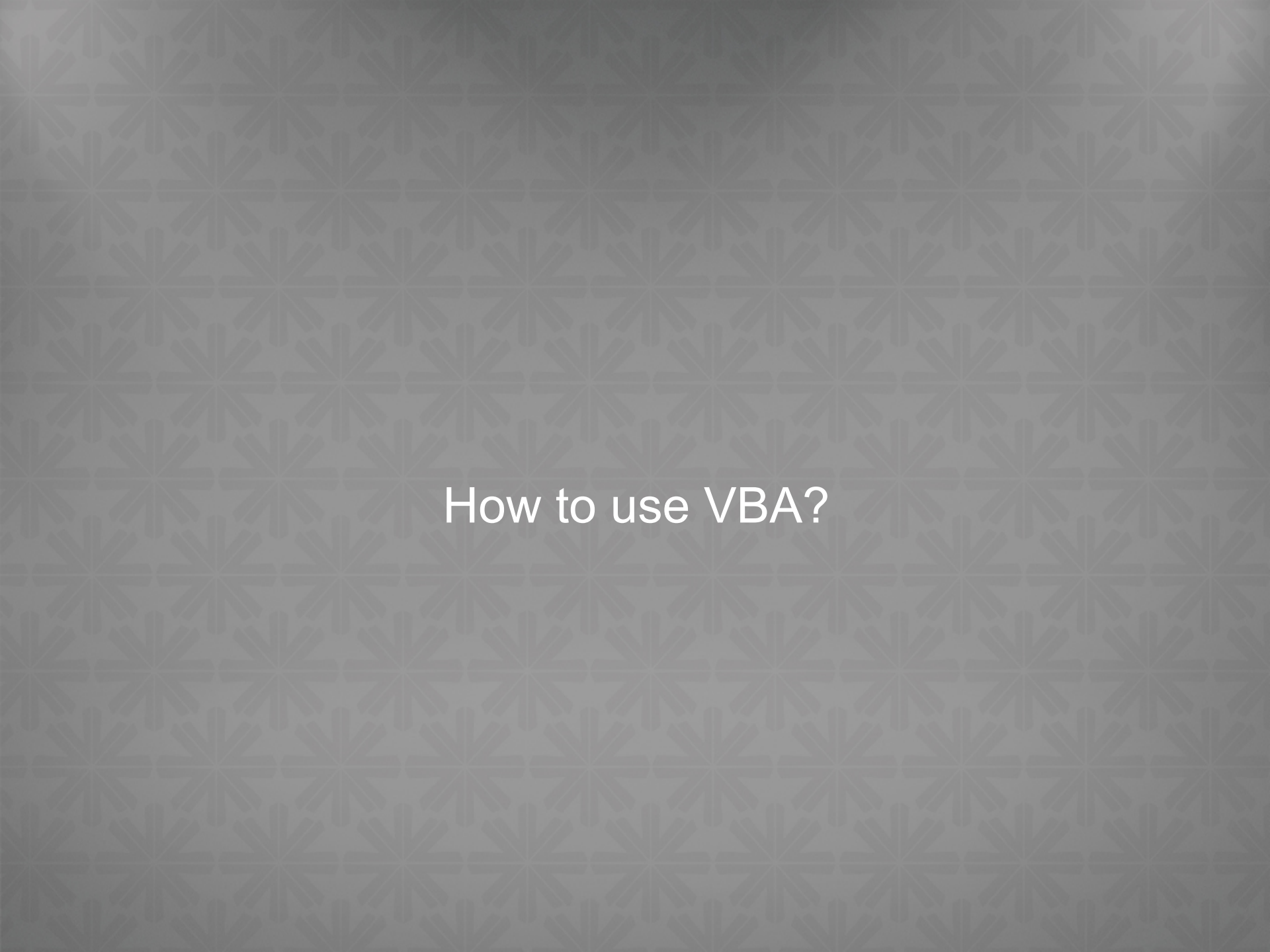
What do we want to accomplish?

- An Excel application which can write simple SAS Code
- A worksheet which stores metadata
- VBA code that creates a SAS program
- Interface elements to execute the VBA code



Why bother with VBA?

- Automate tasks to gain efficiency
- Using Excel because it is almost ubiquitous
- Using VBA because it comes bundled with Excel
- To hone our programming skills



How to use VBA?

Differences between VBA and SAS

- VBA is object oriented / SAS is procedural
- VBA sub-routine based / SAS is almost not
- VBA programs are more complex / SAS uses less branching and explicit looping
- VBA is event driven / SAS is not

Definitions

- **Class:** The "blueprint" of objects. Contains the code that defines values and behaviour of objects
- **Object:** Instance of a class
- **Property:** A variable defined for a class and stored inside objects (sometimes called member in VBA)
- **Method:** a subroutine, defined within a class, for modifying values stored in a property within objects (in VBA, property is a special set of methods)

Objects

Class myObject

Option Explicit

' Properties:

```
Private pProp1 as String  
Private pProp2 as String  
Private pProp3 as Integer
```

' Methods:

```
Property Get strProp1() as String  
    strProp1 = pProp1  
End Property
```

```
Property Let strProp1(strVal1 as String)  
    pProp1 = strVal1  
End Property
```

```
Property Get strProp2() as String  
    strProp2 = pProp2  
End Property
```

```
Property Let strProp2(strVal2 as String)  
    pProp2 = strVal2  
End Property
```

```
Property Get intProp3() as String  
    intProp3 = pProp3  
End Property
```

```
Property Let intProp3(intVal3 as String)  
    pProp3 = intVal3  
End Property
```

' more methods

pProp1 = "Hello"
pProp2 = "World"
pProp3 = 42

myObject1

Dim myObject1 as myObject
Set myObject1 = New myObject
myObject1.intProp3 = 42

pProp1 = "proc"
pProp2 = "PCMP"
pProp3 = 2

myObject3

Dim myObject2 as myObject
Set myObject2 = New myObject
myObject2.intProp3 = 2

pProp1 = "mighty"
pProp2 = "duck"
pProp3 = 99

myObject2

Dim myObject3 as myObject
Set myObject3 = New myObject
myObject1.strProp2 = "duck"

pProp1 = "very mighty"
pProp2 = "mouse"
pProp3 = 104

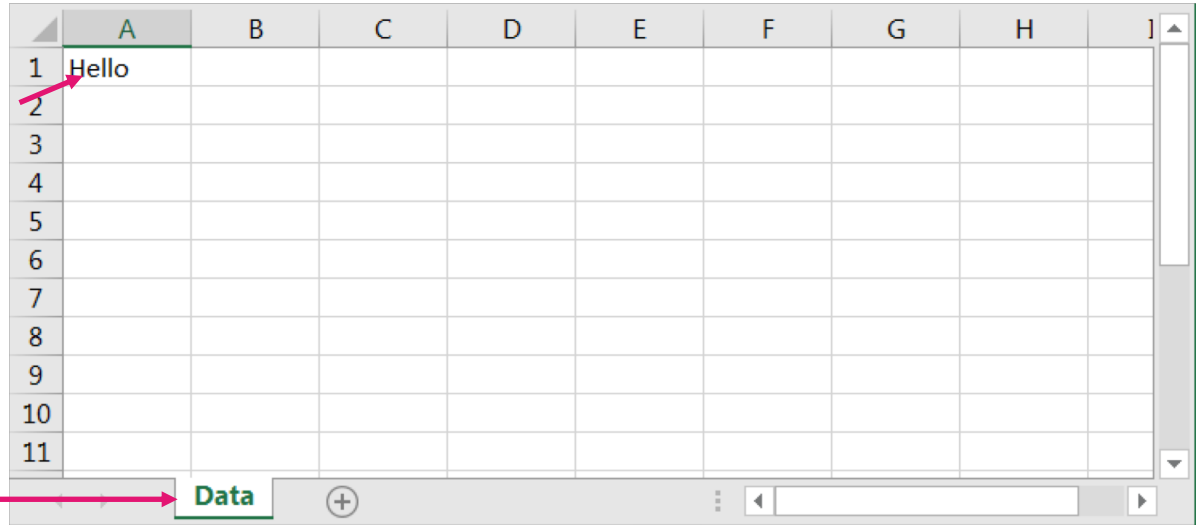
myObject4

Dim myObject4 as myObject
Set myObject4 = New myObject
myObject4.strProp1 = "very mighty"

Objects from Libraries

```
Dim mySheet as Excel.Worksheet  
Set mySheet = ThisWorkbook.Worksheets(1)  
mySheet.Range("A1").Value="Hello"
```

```
mySheet.Name = "Data"
```



Changing the value of properties of library objects
Changes how the objects are displayed to the user

The Application

- Excel Workbook
- One worksheet named "Data" contains our metadata
- A button in the quick access bar starts the VBA code
- Two sub-routines:
 - Extract metadata from worksheet "Data"
 - Write the SAS program

Demo

Conclusions

- You do not always need a SAS macro to be able to write SAS code
 - A viable solution for creating SAS programs using metadata is VBA, if the data are stored within a Microsoft Office® product
 - Other programming languages than SAS do not have to face the same ambiguity than the SAS Macro Compiler with specific characters
 - VBA copes well with changing data types from cell to cell
 - When using MS Office Applications, VBA is a great language to take advantage of lots of features of MS Office
 - We can enhance our toolset with new programming techniques

Questions?