Warum umständlich, wenn es auch einfacher geht?

Unentdecktes Verbesserungspotenzial in den eigenen Excel®-Dateien

DGVFM
DEUTSCHE GESELLSCHAFT FÜR VERSICHERUNGS- UND FINANZMATHEMATIK e.V.

DAV
DEUTSCHE AKTUARVEREINIGUNG e.V.

Institut für Finanz- und Aktuarwissenschaften ifa

Stefan Graf

e-Herbsttagung von DAV und DGVFM, November 2020

Warum umständlich, wenn es auch einfacher geht?

Einführung

Die **Themenfelder der Aktuare** sind vielfältig, z.B. hinsichtlich

- Kapitalanlage, Pricing bzw. Produktentwicklung, Reporting, Risikomanagement, …

und erfordern die Entwicklung entsprechender **aktuarieller Methoden** und insbesondere deren **quantitative Umsetzung**.

- Eine quantitative Umsetzung kann grundsätzlich auf Basis verschiedener Ansätze bzw. Systeme erfolgen:
  - Einsatz von extern entwickelten Softwarelösungen (z.B. aktuarieller Projektionssoftware)
  - Entwicklung eigenständiger Lösungen mittels einer Programmiersprache der Wahl (z.B. Java, C#, VB.NET, Python, R, …)
  - Entwicklung eines „Spreadsheets" unter Verwendung eines Tabellenkalkulationsprogramms, und das heißt in den allermeisten Fällen **Microsoft Excel®**.

Ziel dieses Vortrags ist es

- nicht zu adressieren, wann der Einsatz welches Systems angemessen ist, sondern anzuerkennen, dass aktuarielle Methoden häufig mittels Excel® umgesetzt werden,
- um dann Hinweise und Anregungen zu geben, was bei Überarbeitung bestehender oder neu zu erstellender Spreadsheets beachtet werden kann bzw. eher vermieden werden sollte.

➔ Damit können die stetig wachsenden Anforderungen sicherer, schneller und zielgerichteter bearbeitet werden und es bleibt mehr Zeit, die Ergebnisse zu validieren und zu interpretieren.

  - vgl. hierzu auch im Vortrag adressiertes Beispiel

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

Fazit

Institut für Finanz-und Aktuarwissenschaften

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

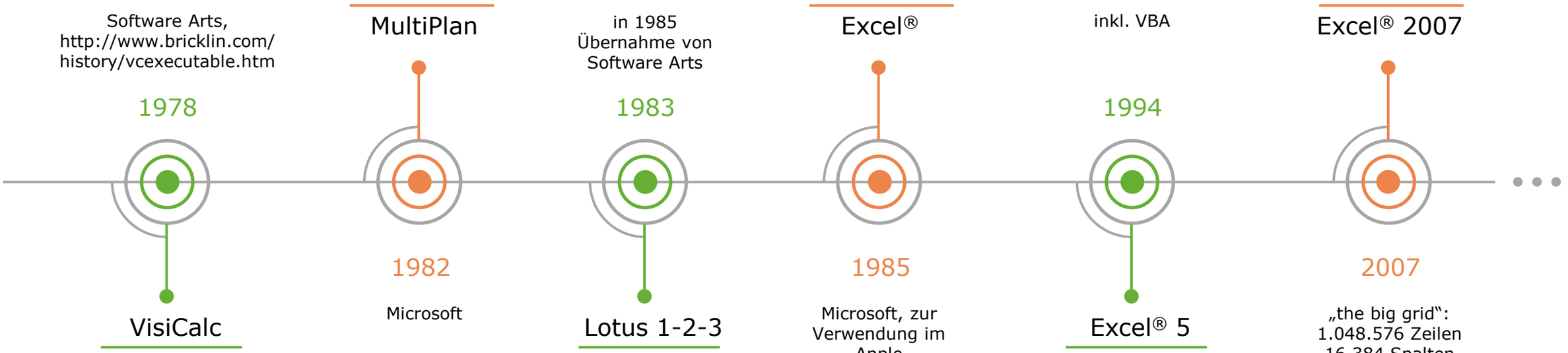
Ein grober Überblick

Die erste Version von Excel® wurde entwickelt …

- für den Apple Macintosh.

„recalc or die"

Aber der Reihe nach



Und danach …

- Im Oktober 2014 stellt IBM den Support für Lotus 1-2-3 ein.
- Mittlerweile sind auch freie Alternativen (z.B. LibreOffice, OpenOffice) verfügbar.
  - Excel® ist dennoch auf absehbare (dauerhafte?) Zeit „die" Lösung für Spreadsheets im Bereich Business-Anwendungen.

Zur Historie der Tabellenkalkulation: Von VisiCalc zu Excel®

Funktionsweise eines Spreadsheets

VisiCalc und (die ersten Versionen von) Lotus 1-2-3

- berechnen das vollständige Tabellenblatt neu, sobald sich eine Zelle im Tabellenblatt geändert hat.

Excel® führt das sogenannte „**intelligent recalc**" ein, d.h.

- es werden nicht mehr alle, sondern nur noch die „relevanten" Zellen neu berechnet.

➔ massiver Performancegewinn

Wie kommt man aber zu den relevanten Zellen bzw. was macht intelligent recalc?

- Zunächst wird ein sogenannter Abhängigkeitsbaum („dependency tree") ermittelt (iteratives Vorgehen).
  - Dieser fasst Abhängigkeiten (Vorgänger bzw. Nachfolger) der im Spreadsheet implementierten Zellen zusammen.
- Auf Basis dieser Abhängigkeiten wird bei Veränderungen im Spreadsheet der Berechnungsablauf („calculation chain") ermittelt und die Berechnungen werden dann entsprechend durchgeführt.

➔ **Nur Zellen, die im aktuellen Durchlauf neu berechnet werden müssen, werden auch wirklich neu berechnet.**

- Ausnahme (Auszug):
  - Volatile Aktionen (z.B. Autofilter, Öffnen einer CSV-Datei, Einfügen/Löschen von Zeilen, Spalten) führen ebenfalls zur Neuberechnung des gesamten Spreadsheets.
  - Sogenannte volatile Funktionen werden stets neu berechnet ➔ Abhängige Zellen werden damit auch neu berechnet.

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

Fazit

Institut für Finanz-und Aktuarwissenschaften

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Warum ist gerade Excel® so beliebt?

Warum ist Excel® so beliebt und weit verbreitet?

- sehr flexibel und (prinzipiell) für jede Anwendung geeignet
- geringe Einstiegshürde für Benutzer
  - Excel® ist im Prinzip Standardsoftware.
  - Jeder kann sofort produktiv werden.
- (gefühlt) äußerst transparent
  - keine Black-Box: Man sieht sofort, was man bekommt.
    - Auch wenn die zugrundeliegenden Formeln manchmal beliebig intransparent sind.
- große Community
  - Für fast jedes Problem findet man online eine (manchmal auch gute) Lösung.

Und die Schattenseiten davon …

- Quelle: http://www.eusprig.org/horror-stories.htm
- Oktober 2020 (Großbritannien)
  - Excel: Why using Microsoft's tool caused Covid-19 results to be lost.
- August 2020
  - Human genes renamed as Microsoft Excel reads them as dates.
- Juni 2019 (Südafrika)
  - *"[…] which ruled that the actuary alone was responsible for R40m in overpayments made to exiting members of the fund. The overpayments, made over a number of years, stemmed from a "hard-coding error" in the spreadsheet system the actuary had been using."*

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Anwendungsmöglichkeiten

Excel® ermöglicht (unglaublich) vielfältige Anwendungen, z.B. hinsichtlich

- Buchhaltung bzw. Controlling
- Budgetplanung und Business Cases
- Verwendung als Datenbank (und zugehörige Datenanalyse)
- Nutzung als Oberfläche für gänzlich andere Funktionalitäten (in Verbindung mit VBA)
- oder auch zum Musik machen
  - „Drum Machine in Excel®", vgl. https://www.youtube.com/watch?v=To2JIXGoYzA
- …

Ein Auszug aus aktuariellen Anwendungen

- Kapitalanlage
  - Erstellung von ESGs, Pricing von Optionen, …
- Pricing bzw. Produktentwicklung
  - Analyse von Produktideen bzw. Entwicklung von Rechnungsgrundlagen, Durchführung von Profitabilitätsanalysen, Test von Rechenkernen, …
- Reporting und Risikomanagement
  - Aufbereitung zugehöriger Inputdaten bzw. Durchführen von ALM-, Solveny II- oder Planungsrechnungen
- …

Welche **gemeinsamen Anforderungen** ergeben sich bei einer Umsetzung der aktuariellen Anwendungen?

- **Transparenz und Performance** der Berechnungen bzw. (Daten-)aufbereitungen (insbesondere vor dem Hintergrund der Wartbarkeit bzw. Weitergabe eines Tools)
- **Datenhaltung** externer Input-Daten, interner Parameter und entsprechendem Output
- **Automatisierung** sowohl innerhalb einer Aufbereitung als auch bei mehreren ähnlichen Aufbereitungen

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

Fazit

Institut für Finanz-und Aktuarwissenschaften

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Einfache, aber effektive Empfehlungen

Transparenz und Performance

Dos

- ein paar Grundregeln
  - Programmierung von links oben nach rechts unten
  - Colour-Coding
    - nicht je bunter desto besser, sondern
    - gezielter Einsatz von farbigen Kennzeichnungen
      - für Input-Parameter,
      - bei sich ändernden Formeln (z.B. zum Jahresstichtag oder für andere Formeln bei Erstbelegung und Fortschreibung einzelner Größen)
      - für wichtige Ergebnisse
  - Gezielte Verwendung von absoluten („A1“) und relativen Bezügen („A1“) erleichtert das korrekte automatische Ausfüllen von Formeln.

Don'ts

- Konstanten („magic numbers“) in Formeln verwenden

  ➔ stattdessen: Konstanten auf einen separaten Zellbereich verlagern.

- Ausgeblendete Zellen bzw. Arbeitsblätter oder unsichtbarer Inhalt (weiße Schrift auf weißem Hintergrund)

  ➔ stattdessen: Gruppierungen vornehmen und damit ggf. ausblenden.

- duplizierte Formeln in einer Zelle (oft bei Wenn-Dann-Bedingungen anzutreffen)
  - WENN(„Komplexe Abfrage“ erfüllt Bedingung, „Komplexe Abfrage“, Sonst_Wert)

  ➔ stattdessen: „Komplexe Abfrage“ auf separate Zelle verlagern

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Einfache, aber effektive Empfehlungen

Transparenz und Performance

Dos

- Formatierung von Formeln in Zellen
  - Setzen von Leerzeichen oder Zeilenumbrüchen (ALT + ENTER) in Formeln verbessert die Leserlichkeit
- **Regelmäßig unnötige Zellen / Berechnungen entfernen!**
  - Prüfung wie viele Zellen eigentlich aktuell vom Arbeitsblatt verwendet werden (STRG + ENDE) und dann ggf. überflüssige Zellen entfernen.
  - Häufig wird in einem Spreadsheet schnell noch eine Auswertung für einen aktuellen Zweck implementiert.
    - Nächstes Jahr ist diese Auswertung aber nicht mehr notwendig, im Spreadsheet aber (dann dauerhaft) enthalten.

    ➔ Man muss sich trauen, auch wieder etwas zu löschen!
- **Interne Checks zur Validierung der Berechnungen einführen.**

Don'ts

- Verwendung von **volatilen Funktionen**, z.B.
  - offensichtlich: ZUFALLSZAHL(), HEUTE(), JETZT()
  - weniger offensichtlich: INDIREKT(), BEREICH.VERSCHIEBEN(), ZELLE()
  - gar nicht offensichtlich:
    - SUMMEWENN(), falls Ranges nicht vollständig angegeben wurden (z.B. SUMMEWENN(A1:A4;"=1"; B1)
    - bedingte Formatierungen

  ➔ stattdessen: In den allermeisten Fällen kann dieselbe Fragestellung auch ohne Verwendung einer volatilen Funktion erreicht werden, z.B.
    - BEREICH.VERSCHIEBEN() oder INDIREKT() kann häufig mittels INDEX() oder einer Kombination aus INDEX() und VERGLEICH() ersetzt werden.

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Einfache, aber effektive Empfehlungen

Transparenz und Performance, Datenhaltung

Dos

- Was zusammengehört, steht auch zusammen.
  - Daten zum Input, zu Berechnungen und zum Output in zusammenhängenden Blöcken sammeln

Don'ts

- Input aus externen Datenquellen (meist anderen Spreadsheets) durch manuelles Kopieren übernehmen

Maybes

➔ stattdessen Verwendung von Verknüpfungen, aber falls möglich nicht auf eine konkrete (operative) Zelle verweisen.

- Was wenn diese beim nächsten Mal nicht mehr mit dem gewünschten Zielwert belegt ist bzw. verschoben wurde?
- besser über INDEX() / VERGLEICH() oder ein geeignetes Schnittstellenblatt zugreifen

- **ABER**: Zu viele Verknüpfungen führen oft zu operativen Problemen.
  - Manchmal lohnt eine alternative Art der Datenhaltung (z.B. in Form einer Datenbank, XML, JSON, …), die dann im Spreadsheet direkt verwendet werden kann.

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Aber es gilt auch: Nicht immer gibt es nur schwarz oder weiß

Transparenz und Performance, Datenhaltung, Automatisierung

Maybes

- Verwendung von Namensdefinitionen
  - Formeln und Berechnungen werden bei sparsamer Dosierung durch den Einsatz von Namen häufig transparenter.
  - ABER: Hat man zu viele Namen vergeben, so wird es häufig unübersichtlich und meist sind nach einiger Zeit viele der Namen kaputt („#BEZUG!“).
  - Für dynamische Charts, d.h. bei Charts deren Datengrundlage sich dynamisch an Vorgaben im Spreadsheet (z.B. bzgl. einer Projektionsdauer) anpassen, sind entsprechende Namensbereiche oft hilfreich.
    - auch hier gilt: Verzicht auf volatile Funktionen, d.h. lieber INDEX() / VERGLEICH() statt BEREICH.VERSCHIEBEN()
- Verwendung von Multithreading
  - Seit Excel® 2007 sind parallelisierte Berechnungen durch Nutzung mehrerer Threads möglich.
    - Oft sind Berechnungen dadurch performanter.
  - Das kommt aber auf die Struktur des Abhängigkeitsbaums an und sollte im Zweifel stets getestet werden.
    - Bei zu vielen verfügbaren Threads kann die Berechnung auch deutlich langsamer werden.

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Die wichtigste Punkte kurz zusammengefasst und in einem Beispiel angewendet

Zusammenfassung

- Klare, übersichtliche Datenhaltung unterstützt durch geeignetes Colour-Coding schafft Transparenz, führt zu Stabilität der Berechnungen und erleichtert die Weitergabe von Spreadsheets.
- Vermeidung von volatilen Funktionen verbessert die Performance (oft maßgeblich).

Ein Beispiel: Überarbeitung eines Spreadsheets zur stochastischen Projektion von Altersvorsorgeprodukten

- Ausgangspunkt
  - In einer CSV-Datei liegen Realisierungen mit stochastischen Entwicklungen von Aktien und Zinsen.
  - Im Spreadsheet werden diese Realisierungen verarbeitet und damit unterschiedliche Altersvorsorgeprodukte simuliert.
  - Rechenzeit: grob 20 Minuten für eine Berechnung mit 1.000 Szenarien
- (Automatisches) Screening nach volatilen Funktionen ergibt eine Vielzahl verwendeter volatiler Funktionen und einige nicht mehr funktionierende externe Verknüpfungen (insbesondere in Namen).

➔ Durchgeführte Anpassungen:

  - Alle volatilen Funktionen und bedingten Formatierungen entfernt und durch nicht volatile Funktionen ersetzt.
  - Nicht mehr funktionierende Verknüpfungen in Namen entfernt.

Fortsetzung folgt…

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

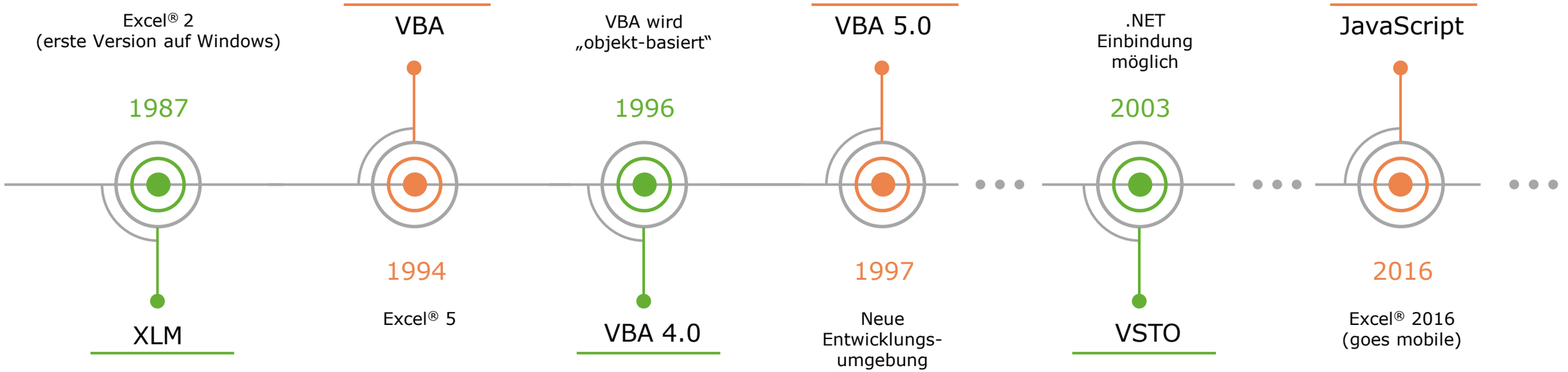
Fazit

Institut für Finanz-und Aktuarwissenschaften

Noch mehr Funktionalität durch VBA und Co

Erweiterung der Arbeitsblattfunktionen durch „Makros“: Ein grober Überblick

Der Funktionsumfang von Excel® kann durch einige zusätzliche (Skript-)Sprachen erweitert werden.



- XLM Makros funktionieren heute immer noch, insbesondere in der Variante nach Excel® 4.
- Visual Basic for Applications (VBA) stellt zusätzliche Funktionalitäten zur Verfügung und ist Bestandteil vieler Anwendungen.
  - wurde aber seit 2010 nicht mehr wesentlich weiterentwickelt
- Visual Studio Tools for Office (VSTO) bzw. Excel-DNA (freie Alternative) erlaubt Integration des **.NET-Framework**.
- JavaScript / TypeScript erweitert das Repertoire und wird voraussichtlich in Office 365 (web) zum Standard werden.

Noch mehr Funktionalität durch VBA und Co

Makros: Anwendungen und Beliebtheit von VBA

Mögliche Anwendungen von Makros

- Erweiterung von Funktionalitäten auf dem Arbeitsblatt (sog. UDF, User Defined Function)
  - VBA ist bereits sehr mächtig. Durch Integration von .NET bestehen sehr viele Erweiterungsmöglichkeiten.
- Automatisierung bestehender Aufgaben, z.B.
  - Wiederholung gewisser Berechnungen einer Arbeitsmappe für unterschiedliche Modelpoints / Parametersets
  - oder Steuerung extern durchzuführender Berechnungen
- Entwicklung völlig eigenständiger Programme
  - Excel® ist hier nur die „Entwicklungsumgebung“ für eine Applikation, die in VBA geschrieben ist.
- …

Warum ist VBA so beliebt?

- sehr niedrige Einstiegshürde
  - keine separate Installation notwendig ➔ (ALT + F11) führt bereits zum Code
  - einfach zu lernen bzw. einfach zu schreiben
- Der Compiler verzeiht einem (leider) vieles, z.B. müssen Variablen nicht explizit deklariert werden.
- **Der Makro-Rekorder!**
  - Auch ohne VBA-Kenntnis kann mit dem Makro-Rekorder Code mit komplexer Funktionalität erzeugt werden.
- große Community
  - Für fast jedes Problem findet man online eine (manchmal auch gute) Lösung.

Auch bei der Entwicklung von Makros können hinsichtlich Transparenz und Performance, Datenhaltung und Automatisierung gewisse Dos, Don'ts and Maybes beachtet werden.

- Es folgen ein paar Anregungen …

Noch mehr Funktionalität durch VBA und Co

Anregungen zu Umsetzungen mit VBA

Transparenz und Performance, Datenhaltung und Automatisierung

Dos

- ein paar einfache Grundregeln
  - Deklaration von Variablen erzwingen durch Verwendung von „Option Explicit"
  - explizite Typendefinition von Variablen durchführen
    - Dim i As Integer, j As Integer
    - Nicht: Dim i, j As Integer
      - Damit ist j eine Integer, i aber eine Variant.
- Redundanz vermeiden und keine globalen Variablen verwenden
- lieber Interaktion einfacher Funktionalitäten statt Implementierung einer umfangreichen Funktionalität
  - Weniger umfangreicher Code ist i.A. robuster und kann besser getestet bzw. wiederverwendet werden.

Don'ts

- Konstanten („magic numbers") innerhalb von Methoden / Funktionen verwenden

  ➔ stattdessen: globale Konstanten definieren
- Verwendung mittels Makro-Rekorder aufgezeichneten Codes ohne Nachbearbeitung
  - Oft findet man bei aufgezeichnetem Code eine Verwendung von „Selection".
  - Diese ist in den meisten Fällen nicht notwendig, oft fehleranfällig und meistens schlecht für die Performance.

  ➔ stattdessen: korrekten Zugriff („per google") verwenden
- Kommunikation mit dem Arbeitsblatt mittels fester Zellbezüge, z.B. Worksheets(„Tabelle1").Range(„A1")

  ➔ stattdessen: Verweis auf Namen

Noch mehr Funktionalität durch VBA und Co

Anregungen zu Umsetzungen mit VBA

Transparenz und Performance, Datenhaltung und Automatisierung

Dos

- bei Automatisierung sich wiederholender Aufgaben
  - Performancesteigerung mittels
    - Application.ScreenUpdating = False
    - ggf. Umschaltung auf manuelles Berechnen Application.Calculation=xlCalculationManual und verwenden von Application.Calculate
  - Transparenz durch Darstellung des aktuellen Berechnungsstands mittels der Application.StatusBar
- „Housekeeping"
  - Bei Ende oder Abbruch der Berechnungen stelle ursprünglichen Zustand der Arbeitsmappe wieder her, insbesondere
    - Application.ScreenUpdating = True
    - Ausgangszustand von Application.Calculation

Don'ts

- zu häufige Kommunikation mit dem Arbeitsblatt, beispielsweise innerhalb von Simulationen

  ➔ stattdessen: Daten (z.B. Ergebnisse) in VBA halten und dann „am Ende in einem Rutsch" (als Matrix) auf das Arbeitsblatt schreiben
  - führt häufig zu großem Performancegewinn
- Vermeidung von „copy & paste", z.B. bei der Einbindung externer Daten (z.B. CSVs, Spreadsheets)

  ➔ stattdessen: Daten in die Arbeitsmappe einbinden (mittels UDF) oder Verknüpfung (aber sparsam) verwenden
  - Damit werden die externen Daten Teil des Abhängigkeitsbaums und dies steigert im Allgemeinen die Performance.

Noch mehr Funktionalität durch VBA und Co

Anregungen zu Umsetzungen mit VBA

Transparenz und Performance, Datenhaltung und Automatisierung

Maybes

- Auslagerung häufig verwendeter Funktionalitäten in ein Add-In?
  - Hat man mehrmals quasi denselben Code in separaten Arbeitsmappen implementiert (z.B. „die selbe Monte-Carlo-Schleife“) bietet es sich an, derartigen Code in ein Add-In auszulagern. Damit kann Codereproduktion vermieden werden.
  - ➔ Dazu sollte der ins Add-In aufgenommene Code allerdings entsprechend abstrahiert sein, damit er auch für ähnliche Aufgaben wiederverwendet werden kann.

Erstellung von Add-Ins erfolgt grundsätzlich mittels folgender Methoden:

- VBA („xlam“)
  - ➔ einfacher Zugang, da im Prinzip nur Erstellung einer entsprechenden Arbeitsmappe
- C# oder VB.NET („xll“)
  - ➔ komplexer (aber mächtiger), da Visual Studio benötigt wird
  - ➔ Entwicklung mit modernen Programmiersprachen möglich
- TypeScript / HTML / CSS (ab Excel® 2016)
  - ➔ u.U. der zukünftige Weg zur Erstellung von Add-Ins (seit 2020 ist ein zugehöriger „Makro-Rekorder“ verfügbar).

Noch mehr Funktionalität durch VBA und Co

Anregungen zu Umsetzungen mit VBA

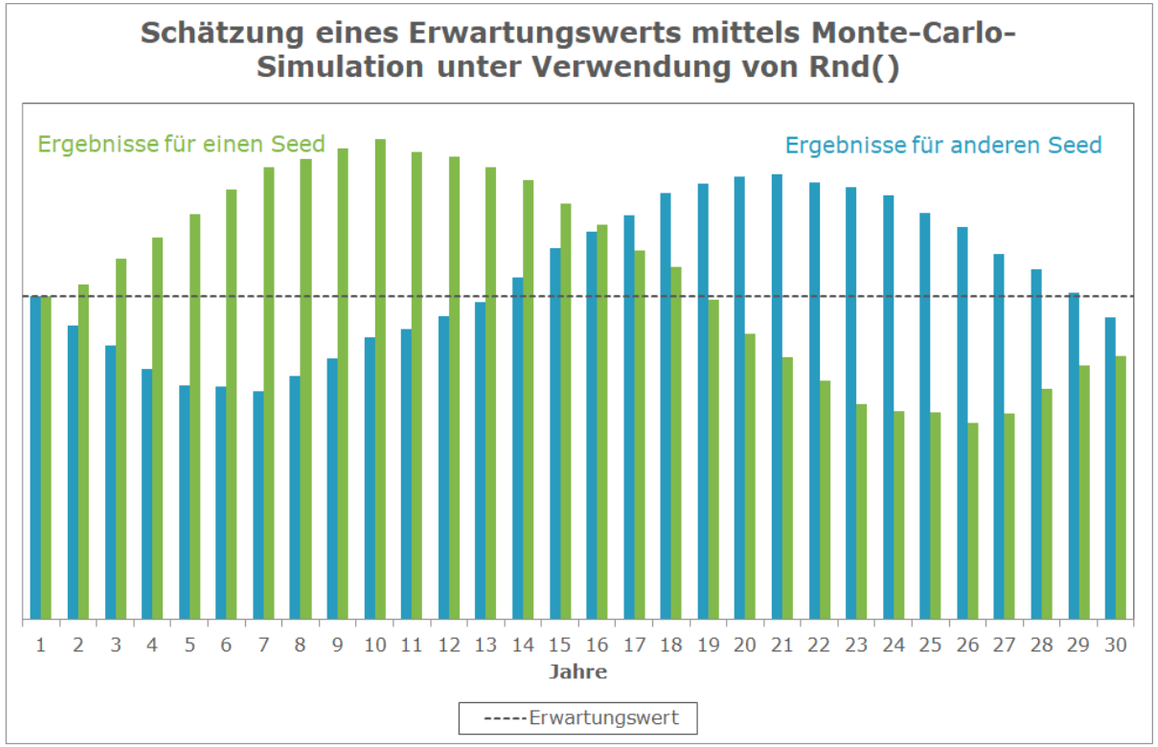
Maybes

- Verwendung des Zufallszahlengenerators „Rnd()“ in VBA
  - *„Thankfully, while Rnd() is an **extremely unsophisticated RNG**, it is at least easy to use and seed”*
  - Quelle: http://blog.richpollock.com/2014/08/randomness-in-excel/

Weitere akademische Arbeiten

- L'Ecuyer, P und Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4): 22.
- Mélard, G. (2014). On the accuracy of statistical procedures in Microsoft Excel 2010. *Computational statistics*, 29(5): 1095-1128

Auswirkungen in der Praxis



Der Einsatz von Rnd() sollte zumindest wohl überlegt sein oder es werden stattdessen alternative (auch in VBA verfügbare) Zufallszahlengeneratoren verwendet.

Noch mehr Funktionalität durch VBA und Co

Die wichtigste Punkte kurz zusammengefasst und in einem Beispiel angewendet

Zusammenfassung (VBA)

- Explizite Deklarationen und übersichtliche Methoden verbessern die Transparenz und Wartbarkeit des Codes.
- Eingeschränkte Kommunikation mit dem Arbeitsblatt und Vermeidung von „copy & paste" tragen zur Steigerung der Performance bei.

Zurück zum Beispiel: Überarbeitung eines Spreadsheets zur stochastischen Projektion von Altersvorsorgeprodukten

- Weitere durchgeführte Anpassungen
  - Anbindung des bisher durch „copy & paste" eingebundenen CSV-Files mittels einer Arbeitsblattfunktion (UDF)
  - Reduktion der Kommunikation mit dem Arbeitsblatt auf im Wesentlichen zwei „Hits" (Input, Output)

➔ Reduktion der Rechenzeit auf grob 6 Minuten (von ursprünglich 20) für eine Berechnung

  - Damit bleibt mehr Zeit, die **Ergebnisse zu interpretieren und zu validieren** und auch mal die eine oder andere zusätzliche Sensitivität zu analysieren.

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

Fazit

Institut für Finanz-und Aktuarwissenschaften

Warum umständlich, wenn es auch einfacher geht?

Fazit

Die **Themenfelder der Aktuare** sind vielfältig, z.B. hinsichtlich

- Kapitalanlage, Pricing bzw. Produktentwicklung, Reporting, Risikomanagement, …

und die Umsetzung quantitativer Methoden findet häufig mittels Excel®-Spreadsheets statt.

- Nichtsdestotrotz ist Excel® nicht immer die einzig richtige (und dauerhafte) Lösung für jede Fragestellung.

In diesem Vortrag haben wir Hinweise und Anregungen gegebenen

- was bei neu aufzusetzenden Spreadsheets beachtet werden kann, um die Transparenz und Performance der Berechnungen zu steigern,
- welche Überarbeitungen mit **in aller Regel begrenztem Aufwand** möglich sind, um existierende (historisch gewachsene) Spreadsheets zu verbessern.

Durch die Erhöhung der Qualität der Spreadsheets gewinnen die Mitarbeiter an Sicherheit mit deren Umgang, Kopfmonopole werden reduziert und regelmäßig auftretende Berechnungen können schneller durchgeführt werden.

- Dadurch bleibt mehr **Zeit für die Analyse und Validierung der Ergebnisse**.

Warum umständlich, wenn es auch einfacher geht?

Agenda

Zur Historie der Tabellenkalkulation: Und alles begann mit VisiCalc

Beliebtheit von Excel® und (aktuarielle) Anwendungen

Dos, Don'ts and Maybes (auf dem Arbeitsblatt)

Noch mehr Funktionalität durch VBA und Co

Fazit

Institut für Finanz-und Aktuarwissenschaften

Kontaktdaten

Beratungsangebot

Institut für Finanz- und Aktuarwissenschaften

Kontaktdaten

Dr. Stefan Graf
+49 (731) 20644-258
s.graf@ifa-ulm.de

Beratungsangebot

Life

Produktentwicklung
Biometrische Risiken
Zweitmarkt

Non-Life

Produktentwicklung und Tarifierung
Schadenreservierung
Risikomodellierung

Health

Aktuarieller Unternehmenszins
Leistungsmanagement

Actuarial Consulting

Solvency II ▪ Embedded Value ▪ Asset-Liability-Management
ERM ▪ wert- und risikoorientierte Steuerung ▪ Data Analytics

Projektmanagement ▪ Markteintritt ▪ Bestandsmanagement ▪ strategische Beratung

Actuarial Services

aktuarielle Großprojekte ▪ aktuarielle Tests
Überbrückung von Kapazitätsengpässen

Research

Aus- und Weiterbildung

… weitere Informationen unter **www.ifa-ulm.de**