Einführung in die Datenwissenschaft mit Excel

Christian Glahn

Inhaltsverzeichnis

Vorwort

Work in Progress

Copyright

Die PDF-Version liegt hier zum Download.

Teil I.

Ausgangslage und Vorbereitung

1. Einleitung

> ⚠ Work in Progress

1.1. Ziele

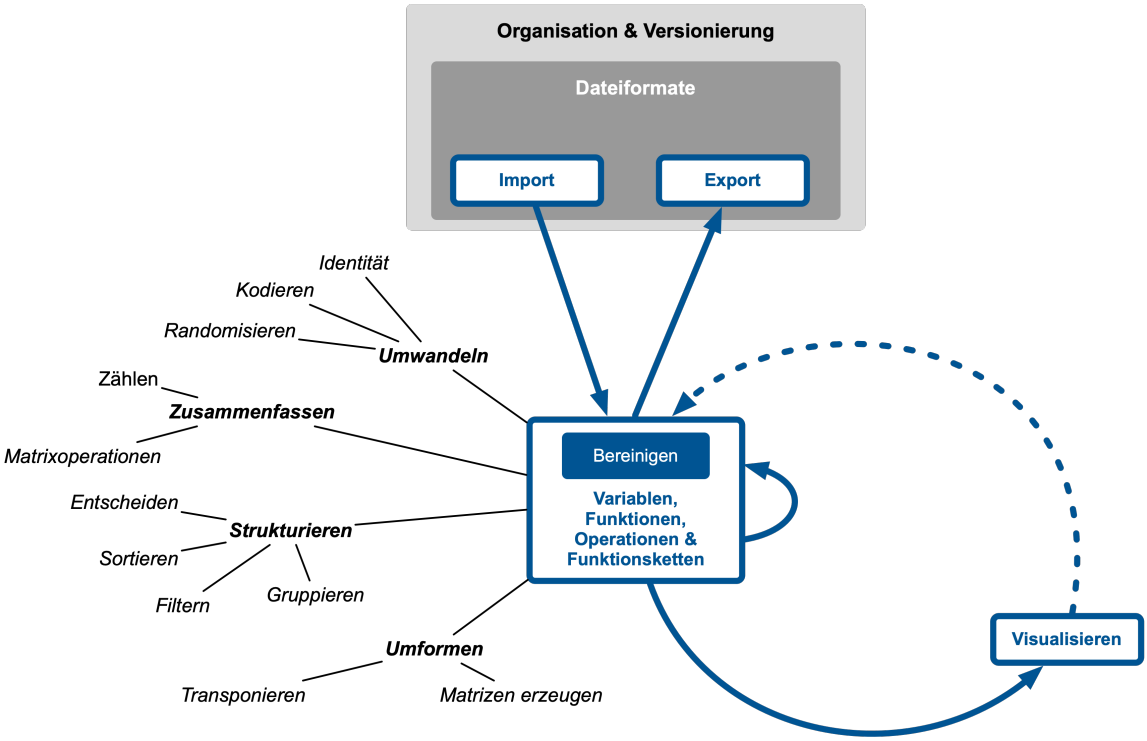


Abbildung 1.1.: Ziele und Themen

1.2. Organisation dieses Buchs

2. Excel Versionen

Excel ist die Tabellenkalkulation der Microsoft Office Umgebung. Die erste Version wurde 1985 veröffentlicht und wird seitdem ständig weiterentwickelt. Aktuell existieren mehrere Versionen von Excel nebeneinander.

- Excel365
- Excel für Desktop-Computer ohne Microsoft365
- Excel als Web-Anwendung
- Excel für Mobilgeräte

Die verschiedenen Versionen unterscheiden sich in der Funktionalität.

Die aktuelle Version ist **Excel365** und muss auf einem Desktop-Computer oder Laptop installiert werden. Diese Version ist Teil des Microsoft365 Abonnements. Diese Version wird kontinuierlich weiterentwickelt und erhält regelmässig neue Funktionen, welche die Arbeit erleichtern.

Teil des Microsoft365 Abonnements sind die Web- und die Mobilversionen von Excel. Beide Versionen sind in der Funktion leicht eingeschränkte Versionen von Excel365. Die Mobilversion ist für die Arbeit auf einem Smartphone oder Tablet optimiert. Die Web-Version benötigt einen Web-Browser und kann auf jedem Gerät mit Internetzugang verwendet werden. Der Unterschied zwischen Excel365 und den wird immer kleiner und die meisten Konzepte in diese Buch lassen sich auch in Excel im Web umsetzen.

Excel's Desktop Version ohne Abonnement wurde auf dem Stand von Juni 2019 eingefroren. Diese Version wird nur noch mit Sicherheitsupdates versorgt. Neue Funktionen sind in dieser Version nicht verfügbar. Diese Version ist noch recht weit verbreitet und lässt sich nur schwer von der aktuellen Version unterscheiden, denn tatsächlich handelt es sich bei Excel365 und der Desktop-Version ohne Abonnement um die selbe Anwendung. Fehlt das Abonnement, dann sind die neuen Funktionen nicht verfügbar und werden nicht angezeigt oder ausgeführt. Falls diese Version aktiv ist und eine Arbeitsmappe, die mit einer Version mit Abonnement erstellt wurde, öffnet, dann erfolgt eine Fehlermeldung und die entsprechenden Funktionen werden blockiert.

Dieses Buch setzt die aktuelle Version von Excel365 voraus. In den folgenden Kapiteln ist also immer Excel365 gemeint, wenn schlicht von Excel geschrieben wird. Die meisten Beispiele und Konzepte lassen sich in der Desktop-Version *ohne Abonnement* nicht umsetzen. Viele Beispiele lassen sich mit der Web- oder Mobilversion nachvollziehen, diesen Versionen fehlt allerdings der Importer für Daten aus anderen Dateien, der in Kapitel 7 behandelt wird und für viele Beispiele Voraussetzung ist.

Ist die richtige Excel Version installiert

Mit einem einfachen Test lässt sich überprüfen, ob eine geeignete Excel Version installiert ist:

1. Eine beliebige Zelle mit der Maus ausgewählt.
2. Es werden die drei Zeichen '`=HS` eingegeben.

Nun sollte der Funktionsvorlschag für die Funktion `HSTAPELN` unterhalb der Eingabe erscheinen. Wird kein Funktionsvorschlag angezeigt, dann ist eine zu alte Version von Excel installiert.
Die Eingabe wird mit `ESC` abgebrochen.
Erscheint kein Funktionsvorschlag, dann muss ein zweiter Test erfolgen.

1. Eine beliebige Zelle mit der Maus ausgewählt.
2. Es werden die Zeichen `=seq` eingegeben.

Nun sollte der Funktionsvorlschag für die Funktion `SEQUENZ` unterhalb der Eingabe erscheinen. In diesem Fall muss nur ein kleines Update von Microsoft365 installiert werden.
Erscheint kein Funktionsvorschlag, dann ist eine veraltete Version von Excel installiert. Diese muss zuerst vollständig deinstalliert werden, bevor die neue Version von Microsoft365 installiert werden kann. Beim vollständigen Deinstallieren leider müssen auch alle Einstellungen **aller** Microsoft365 Anwendungen entfernt werden.

2.1. Excel-Versionen und Betriebssysteme

Excel365 unterscheidet sich leicht für MacOS und Windows. Im Rahmen dieses Buchs stellt das keine Einschränkung dar, denn die Unterschiede sind gering.

Die auffälligsten Unterschiede betreffen den Importer und die bedingte Formatierung.

- Der Importer ist unter Windows etwas flexibler gestaltet als unter MacOS. Dieser Unterschied betrifft den Inhalt dieses Buchs nicht.
- Die *bedingte Formatierung* wird unter MacOS traditionell mit einen völlig anderen Dialog konfiguriert wird als unter Windows.

MacOS vs. Windows

In den (wenigen) Fällen, in denen sich die Excel-Versionen auf den beiden Betriebssystemen unterscheiden, wird der Unterschied durch eine *MacOS vs. Windows*-Box - wie diese hier - gesondert hervorgehoben. Das gleiche gilt für die Eigenheiten, die nur auf einem der beiden Betriebssyeme vorkommen.

MacOS

Soll unter MacOS Excel365 eine alte Version ohne Abonnement ersetzen, dann muss die alte Version zuerst deinstalliert werden. Sind zusätzlich andere Office-Programme von Microsoft ohne Abonnement installiert, dann müssen diese ebenfalls deinstalliert werden. Beim Deinstallieren werden die Konfigurationsdateien nicht gelöscht. Diese *müssen* manuell gelöscht werden. Erst jetzt kann die neue Version von Microsoft365 installiert werden.

2.2. Excel und Sprachen

Excel365 ist in vielen Sprachen verfügbar. Die Sprache wird von Excel vom Betriebssystem übernommen. Dieses Buch bezieht sich auf die deutschsprachige Excel Version mit Schweizer Regionseinstellungen. Alle Beispiele lassen sich in allen anderen Sprachen nachvollziehen, wenn die Werte und Funktionsnamen entsprechend angepasst werden.

Warnung

Die Funktionsnamen unterscheiden sich stark zwischen den einzelnen Sprachen. Oft ist es unmöglich, die Funktionsnamen durch eine Übersetzung des Funktionsnamen aus einer anderen Sprache abzuleiten.
Beispielsweise heisst der englische Funktionsname `OFFSET()` (deutsch: *Versatz*) auf Deutsch `BEREICH.VERSCHIEBEN()` (engl. *move range*).

MacOS vs. Windows

Unter MacOS kann die Sprache von Excel365 ausschliesslich über die *Systemeinstellungen* geändert werden. Die Spracheinstellung wird vom Betriebssystem übernommen. Dadurch verändert sich die Sprache aller Programme, die auf dem Computer installiert sind. Wird die Sprache von MacOS geändert, dann muss der Computer neu gestartet werden, damit die Änderung wirksam wird. Dieser Schritt verändert jedoch nicht nur die Sprache von Menus sondern auch die Namen vieler Verzeichnisse. Deshalb wird dieser Schritt nicht empfohlen.
Unter Windows kann die Sprache nur für Excel365 geändert werden. Dazu muss in Excel unter *Datei* > *Optionen* > *Sprache* geändert werden. Die gewünschte Sprache muss unter *Office-Anwendungssprache* als *bevorzugt* markiert werden. Die Änderung wird erst wirksam, nachdem Excel beendet und neu gestartet wurde.

Die Spracheinstellung haben keinen Einfluss auf die Funktion von Excel. Die Spracheinstellung bestimmt nur die Sprache der Benutzeroberfläche. Zur Benutzeroberfläche gehören die Menüs, die Dialoge, die Hilfetexte und die **Funktionsnamen**. Wird eine Arbeitsmappe in einer Excel mit einer anderen Spracheinstellung geöffnet, dann werden die Funktionsnamen automatisch übersetzt. Es ist jedoch unmöglich die Funktionsnamen in der ursprünglichen Sprache der Arbeitsmappe einzugeben.

3. Bedienoberfläche

3.1. Arbeitsblätter

Der Tabellenbereich ist das zentrale Element von Excels Arbeitsoberfläche. Dabei handelt es sich um den Datenbereich, der sich deutlich durch das beschriftete Gitterraster im Zentrum des Excel-Fensters abhebt und den Grossteil der Arbeitsoberfläche einnimmt. Über dieses Raster wird primär mit Daten und Funktionen interagiert. Der Tabellenbereich zeigt einen Ausschnitt des aktuellen Arbeitsblatts.

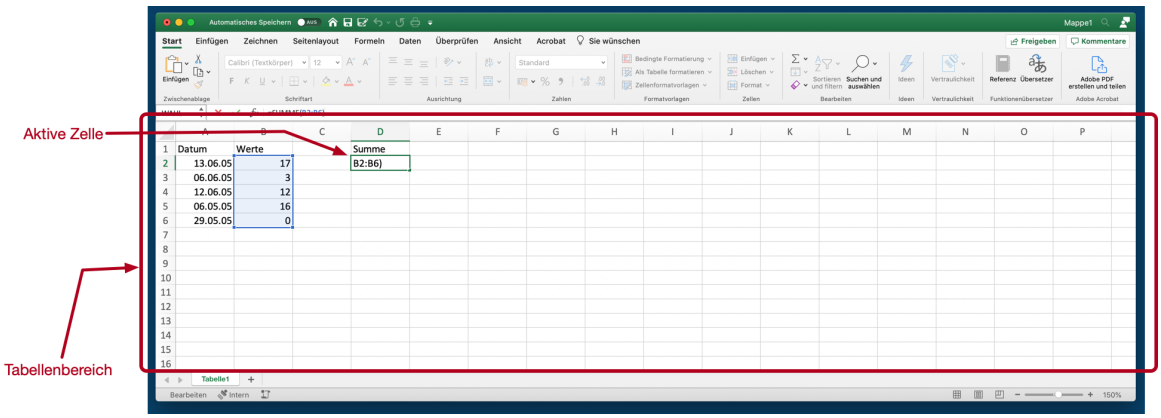


Abbildung 3.1.: Das Excel Arbeitsfenster

Definition 3.1. Ein Excel Arbeitsblatt ist ein Gitterraster, das aus Zellen besteht. Jede Zelle kann einen Wert enthalten.

Definition 3.2. Eine **Zelle** ist ein Feld im Raster eines Arbeitsblatt.

Excel zeigt durch eine Hervorhebung an, welche Zelle aktuell markiert ist. Diese **Markierung** kann mit der Maus oder mit den Pfeiltasten bewegt werden. Wenn nur eine Zelle markiert ist, dann entspricht die Markierung der **aktiven Zelle**. Eine Markierung kann auch mehrere Zellen umfassen. In diesem Fall wird die *aktive Zelle* durch eine helle Hervorhebung gekennzeichnet, während der Rest der Markierung ausgegraut ist. Alle Eingaben mit der Tastatur beziehen sich immer auf die aktive Zelle.

Oben und Links vom Tabellenbereich findet sich die Beschriftung für die Spalten und Zeilen. Spalten werden mit Buchstaben benannt, Zeilen werden mit Zahlen benannt. Daraus ergibt sich ein Koordinatensystem, mit dem jede Zelle eindeutig adressiert wird.

Definition 3.3. Die Koordinaten einer Zelle sind in der **Zelladresse** kodiert.

Im Abschnitt 6.4 werden die verschiedenen Arten der Adressierung von Zellen und Zellbereichen detailliert beschrieben.

3.2. Arbeitsblattreiter

Direkt unter dem Tabellenbereich zeigt Excel *Register* mit den Namen der Arbeitsblätter. Diese Register zeigen alle Arbeitsblätter einer Excel-Arbeitsmappe. Wenn eine neue, leere Arbeitsmappe erstellt wird, dann enthält diese nur ein Arbeitsblatt mit dem Namen `Tabelle 1`. Mit dem `Plus`-Symbol neben dem **Arbeitsblattregistern** können zusätzliche Arbeitsblätter in der Arbeitsmappe erzeugt werden.

3.3. Menübalken

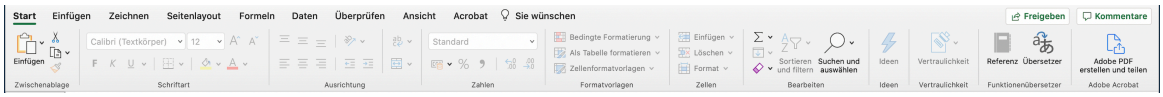


Abbildung 3.2.: Menubalken

Am oberen Fensterrand von Excel befindet sich der **Menübalken**, der im Excel Jargon "Ribbon" heisst. Der Menübalken ist in verschiedene Kategorien unterteilt und sortiert so die verschiedenen interaktiven **Kommandos** von Excel.

Definition 3.4. Ein **Kommando** ist eine *interaktive Anweisung* an Excel, eine bestimmte Aktion auszuführen.

Kommandos werden immer interaktiv ausgelöst und lassen sich meist nicht durch Excel-Funktionen ersetzen.

Für die tägliche Arbeit sind die folgenden Kategorien des Menübalkens am wichtigsten:

- Start
- Einfügen
- Formeln
- Daten
- Tabelle

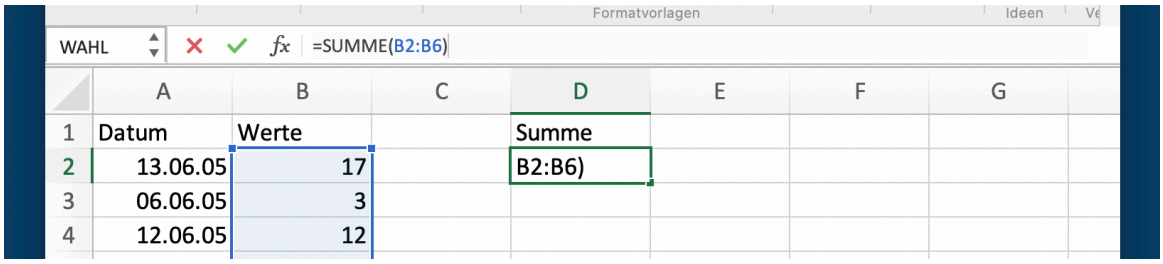


Abbildung 3.3.: Die Excel Funktionsleiste

3.4. Funktionsleiste

Die Funktionsleiste unterstützt die Arbeit mit der *aktiven Zelle*. Die Funktionsleiste wird verwendet, wenn mit speziellen Werten oder mit Komplexen Formeln gearbetiet wird.

Die Funktionsleiste hat 6 Bedienfelder.

Ganz links ist das **Adressfeld**. Das *Adressfeld* zeigt uns immer die Adresse der *aktiven Zelle* an. Wenn wir dieses Feld anklicken, dann können wir eine andere Adresse eintragen. In diesem Fall springt die Markierung für die aktive Zelle an die entsprechende Adresse.

Daneben befinden sich zwei unscheinbare Dreiecke. Dieses Bedienfeld verbirgt eine Liste der *benannten Bereiche* in der aktuellen Arbeitsmappe. Bei einer leeren Arbeitsmappe ist diese Liste leer.

Die nächsten vier Bedienfelder gehören zusammen.

Ganz rechts wird der Inhalt der aktuellen Zelle angezeigt. Bei einer leeren Tabelle ist dieser Bereich selbstverständlich leer. Wenn dieser Bereich mit der Maus ausgewählt wird, dann erscheint ein senkrechter, blinkender Strick, der signalisiert, dass Werte oder Formeln eingegeben werden können. Wenn nun etwas eingegeben wird, dann erscheint diese Eingabe auch in der aktiven Zelle.

Ist die Eingabe abgeschlossen, dann kann diese mit dem grünen Haken oder mit der Eingabetaste bestätigt werden.

Ist bei der Eingabe ein Fehler passiert, dann lässt sich die Eingabe mit einem Mausklick auf das rote X oder mit der `ESC`-Taste abbrechen. Die Eingabe wird dann auf den ursprünglichen Zustand zurückgesetzt.

Wird die aktive Zelle nicht bearbeitet, dann sind der Hacken und das X ausgegraut und können nicht verwendet werden.

Direkt neben dem Eingabefeld mit dem Inhalt der aktiven Zelle ist das Bedienfeld für den *Formelgenerator*. Der Formelgenerator ist ein Hilfsdialog, der die Suche nach geeigneten Formeln unterstützt. Der Formelgenerator unterscheidet sich von Menübalken *Formeln* durch eine kurze Erklärung über die Arbeitsweise einer ausgewählten Funktion.

3.5. Statusleiste

Am unteren Rand von Excels Anwendungsfenster findet sich die Statusleiste. Die Statusleiste hat drei zentrale Funktionen:

- Zoom der Tabellenanzeige,
- Status von Excel und Fehlermeldungen und
- Schnellauswertung eines Zellbereichs.

3.5.1. Zoom der Tabellenanzeige

Der aktuelle Zoom der Arbeitsmappe und die Zoom-Einstellung finden sich ganz rechts in der Statusleiste. Dabei wird der aktuelle Zoom als ein Prozentwert dargestellt. Über den "Schieberegler" links daneben lässt sich der Zoom anpassen.

Wenn der Zoom-Wert angeklickt wird, dann erscheint ein kleiner Dialog, über den Zoom-Wert als Prozentzahl direkt eingegeben kann. Das ist praktisch, wenn für eine Präsentation oder zum Arbeiten ein bestimmter Zoom-Wert bevorzugt wird.

> **Achtung**
>
> Die Zoom-Einstellung muss für jedes Arbeitsblatt separat eingestellt werden und wird beim nächsten Speichern zusammen mit der Arbeitsmappe gesichert.

3.5.2. Status und Fehlermeldungen

Auf der linken Seite der Statusleiste ist der Status Bereich. Normalerweise wird an dieser Stelle den Wert `Bereit` angezeigt. Wenn eine Zelle bearbeitet wird, dann erscheint der Status `Eingeben` oder `Zeigen`.

Wenn einen Fehler bei der Eingabe einer Formel auftritt, dann wird neben dem Status eine **Fehlermeldung** eingeblendet. Diese unscheinbare Fehlermeldung wird immer dann wichtig, wenn eine Formel oder ein Algorithmus nicht zu rechnen scheint. In solchen Fällen hilft ein Blick auf den linken Rand der Statusleiste bei der Fehlersuche. Bei Fehlern zeit die Statusleiste einen Hinweis auf den Fehler sowie die Adresse der Zelle, an der Excel den Fehler vermutet.

Tritt ein Fehler auf, dann bricht Excel die Berechnung der gesamten Arbeitsmappe *sofort* ab und ignoriert alle Formeln, die nach der fehlerhaften Zelle berechnet werden müssten, selbst wenn die Formeln unabhängig voneinander wären. Damit Excel wieder arbeitet, müssen zuerst alle Fehler behoben werden.

3.5.3. Schnellauswertung eines Zellenbereichs

Im mittleren Bereich der Statusleiste ist normalerweise leer. Hier versteckt sich eine sehr nützliche Funktion zur schnellen Zusammenfassung der Werte in einem Bereich: Werden mehrere Zellen markiert, dann zeigt Excel den Mittelwert, die Summe und die Anzahl der ausgewählten Zellen an.

3.5.4. Anpassen der Statusleiste

Wenn Sie mit der rechten Maustaste (Win) bzw. mit Control-Klick (Mac) auf die Statusleiste klicken, dann erhalten Sie eine Auswahlliste der Funktionen, die in der Statusleiste angezeigt werden sollen.

So lassen sich zusätzliche Funktionen in der Statusleiste aktivieren oder störende Elemente deaktivieren.

> **Warnung**
>
> Diese Einstellungen beziehen sich ausschliesslich auf die Excel-Installation und werden nicht zusammen mit der Arbeitsmappe gespeichert!

4. Excel Jargon

Die Dokumentation von Excel verwendet eine Reihe von Begriffen, die von den Begriffen der Datenwissenschaften und der Informatik abweichen. Dieses Kapitel enthält ein Glossar dieser Begriffe.

Arbeitsmappe: Bezeichnet eine Excel-Datei.

Arbeitsblatt: Als Arbeitsblatt wird in der Regel eine Tabelle in einer Excel-Arbeitsmappe bezeichnet.

Adresse: Koordinaten einer Zelle oder eines Bereichs auf einem Arbeitsblatt.

Aktive Zelle: Die Zelle, die für das Bearbeiten ausgewählt ist. Ist nur eine Zelle markiert, dann entspricht die aktive Zelle der Markierung. Ist ein Bereich markiert, dann ist die aktive Zelle, die weiss hervorgehobene Zelle.

Bereich: Eine Auswahl von Zellen auf dem gleichen Arbeitsblatt.

dynamischer Bereich oder **dynamisches Feld** oder **dynamisches Array**: Ein Bereich mit Werten, der durch eine Formel erzeugt wird.

Bezug: Adresse eines Bereichs, um den Wert bzw. die Werte an der Adresse zu erhalten.

Externe Daten: Tabellen, die aus einer anderen Datei oder einer Datenbank eingebunden wurden.

Formel: Aufruf von Funktionen in einer Zelle.

Funktion: Bestandteil einer Formel. Funktionen werden über Parameter gesteuert.

Kommando: Ein Kommando ist eine Interaktion, mit der Excel eine Aktion auslöst. Aktionen ähneln Funktionen, können aber nicht als Formel ausgedrückt werden.

Parameter: Wert, der an eine Funktion übergeben wird. Ein Parameter kann direkt als Wert, indirekt als Adresse oder als Ergebnis einer anderen Funktion übergeben werden.

Pivot-Tabelle: Eine Kreuztabelle, mit der Daten aus mehreren Spalten eines Arbeitsblatts zusammengefasst werden können. Eine Pivot-Tabelle kann Daten von mehreren Arbeitsblättern und Tabellen zusammenfassen.

Ribbon: Menüleiste von Excel.

Tabelle: Eine Tabelle ist eine zwei-dimensionale Datenstruktur für strukturierte Daten. Eine Tabelle besteht aus Spalten und Zeilen.

Zelle: Ein Feld in einer Tabelle. Eine Zelle besteht immer aus einem Werteanteil und einem Formelanteil. Normalerweise wird der Werteanteil angezeigt.

Teil II.

Datenquellen

5. Daten sammeln

5.1. Daten mit Formularen sammeln

Die direkte Eingabe von Daten ist nur für kleine Datensätze mit wenigen Merkmalen geeignet. Bei grösseren Datensätzen ist die direkte Eingabe oft zu aufwändig und fehleranfällig. Für die strukturierte Dateneingabe von grösseren Datensätzen werden Formulare verwendet.

Bei der Dateneingabe über Formulare werden die Daten mithilfe von *Fragen* mit vorgegebenen Antworttypen gesammelt. Der Antworttyp bestimmt, welche Werte für die Frage zulässig sind.

> **Hinweis**
>
> In vielen Formularsystemen wird der *Antworttyp* als ***Fragetyp*** bezeichnet, obwohl die Frage immer in Text-Bildform präsentiert wird und der *Typ* durch die unterschiedlichen Antwortmöglichkeiten bestimmt wird.

Durch das Festlegen des Antworttyps wird gleichzeitig der Wertebereich des gemessenen Merkmals definiert. Dadurch ergibt sich beim Erstellen eines Formulars das Datenschema automatisch.

Excel hat kein eigenes Formularsystem. Stattdessen kann Microsoft Forms verwendet werden. Microsoft Forms ist ein Online-Formulardienst, der in Microsoft365 integriert ist. Forms sammelt die Eingaben mit einem Formular und speichert die Daten in einer Excel Arbeitsmappe. Diese Formulare können über einen Web-Link geteilt werden und im Web-Browser ausgefüllt werden. Dadurch kann das Datensammeln auf verschiedene Personen verteilt werden.

> **Hinweis**
>
> Ein Formular kann in Microsoft Forms ein **Quiz** oder ein **Formular** sein. Ein Quiz ist ein Formular, bei dem die Antworten mit einer Punktzahl bewertet werden und ein Feedback erhalten können. Ein Formular fokussiert sich auf die reine Dateneingabe.

Microsoft Forms unterstützt die folgenden Antworttypen:

- Offene Antworttypen
  - Offene Texteingabe

  - Datumseingabe
  - Datei hochladen

- Geschlossene Antworttypen
  - Einfachauswahl
  - Mehrfachauswahl
  - Rangfolge
  - Likert-Skala
  - Bewertung
  - Net Promoter Score

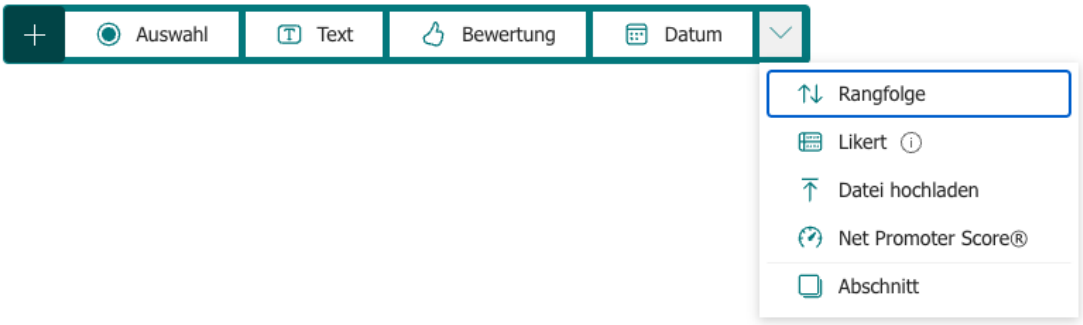


Abbildung 5.1.: MS Forms Fragetypmenu

Jede dieser Varianten hat die folgenden gemeinsamen Optionen:

- Eine einleitende Frage oder Anweisung.
- Ein optionaler Untertitel oder eine Beschreibung. (*Untertitel*)
- Die Möglichkeit eine mathematische Formel zur Validierung des Wertebereichs anzugeben (*Mathematik*).
- Das erzwingen einer Antwort (*Erforderlich*).
- Eine Verzweigung zu einer anderen Frage, wenn eine bestimmte Antwort gegeben wird (*Verzweigung*).

Die Unterschiede ergeben sich in den Antwortmöglichkeiten.

5.1.1. Offene Texteingabe (*Text*)

Die offene Texteingabe erlaubt einzeilige und mehrzeilige (*lange*) *offene Antworten* möglich sind. Damit können un- oder semistrukturierte Daten erfasst werden. Diese Antwortmöglichkeit lässt sich auf Zahlenwerte und Zahlenintervalle einschränken.

5.1.2. Datumseingabe (*Datum*)

Diese Eingabe ist eine Variante der *offenen Texteingabe* zur Eingabe eines Datums. Diese Eingabe wird durch die Option ergänzt, das Datum über eine Kalenderdarstellung auszuwählen.

Das eingegebene Datum wird als Zahl gespeichert und als Datum dargestellt.

5.1.3. Datei hochladen

Die zweite Variante für *offene Antworten* ist das Hochladen von Dateien. Mit diesem Antworttyp lassen sich Bilder, Videos oder Tonaufnahmen leicht erfassen.

5.1.4. Einfachauswahl (*Auswahl*)

Die Einfachauswahl ist ein Single-Choice-Antworttyp. Es sind nur die vorgegebenen Antwortmöglichkeiten möglich, von denen nur eine ausgewählt werden kann. Die Antwortmöglichkeiten können als Text oder als Bild angegeben werden. Die Antwortmöglichkeiten werden immer in einer Liste angezeigt. Dabei kann zwischen einem Auswahlmenu als *Dropdown*-Liste und frei anwählbaren Antwortmöglichkeiten (*Optionen*) gewählt werden.

Die Reihenfolge der Antwortmöglichkeiten kann durch die Option `Optionen in zufälliger Reihenfolge` für jedes Formular zufällig angeordnet werden. Diese Option ist sinnvoll, wenn die Reihenfolge der Antwortmöglichkeiten die Antwort beeinflussen könnte. Soll ein Formular mehrfach durch die gleichen Personen ausgefüllt werden, dann sollte diese Option nicht aktiviert werden.

Das Ergebnis der Einfachauswahl ist eine Zeichenkette der ausgewählten Option.

5.1.5. Mehrfachauswahl (*Auswahl*)

Die Mehrfachauswahl erlaubt die Auswahl mehrerer Antwortmöglichkeiten, indem die Option `Mehrere Antworten` aktiviert wird. Dieser Antworttyp ist ansonsten identisch mit der Einfachauswahl.

Das Ergebnis der Mehrfachauswahl ist eine Zeichenkette mit den ausgewählten Optionen. Wurden mehrere Antworten ausgewählt, dann werden die Optionen durch ein Semikolon getrennt.

5.1.6. Rangfolge

Die Rangfolge erlaubt es, dass die angebotenen Antwortmöglichkeiten in eine Reihenfolge gebracht werden können. Bei diesem Antworttyp müssen die Werte sortiert werden und die Sortierung wird als Antwort gespeichert.

Bei der Darstellung werden die Antwortmöglichkeiten in einer Liste angezeigt und für jedes Formular neu sortiert.

Das Ergebnis der Rangfolge ist eine Zeichenkette mit der Reihung der Optionen. Die Optionen werden durch ein Semikolon getrennt. Im Ergebnis lässt sich die Rangfolge nicht von einer Mehrfachauswahl unterscheiden.

5.1.7. Likert-Skala (*Likert*)

Der Antworttyp *Likert* setzt eine Fragebatterie mit Likert-Skalen um. Darüber lässt sich ein *semantisches Differential* erheben.

Die Anworten dieser Fragebatterie werden wie mehrere Einfachauswahlen als Zeichenketten gespeichert. Deshalb muss für diesen Antworttyp immer eine Kodierungstabelle für die richtige Reihenfolge der Werte dokumentiert werden.

5.1.8. Bewertung

Die Bewertung ist eine Variante des Antworttyps *Likert-Skala*. Mit der Bewertung wird über eine festgelegte Skala der gewünschte Wert abgefragt. Optional können für die beiden Extremwerte Beschriftungen angegeben werden.

Das Ergebnis wird als Zahl gespeichert. Die Zahl entspricht dem Wert der ausgewählten Option beginnend bei 1.

5.1.9. Net Promoter Score

Der Typ Net Promoter Score ist eine spezielle elfstufige Liker-Skala, die zwischen zwei frei wählbaren Extremwerten festgelegt ist. Die offiziell als *Net Promoter Score* (Reichheld, 2003) bezeichnete Eingabeform ist eine verzerrte Likert-Skala mit dem Mittelpunkt bei `7.5`.

Der Net Promoter Score wird immer als Ganzzahl im Intervall $0 \leq s \leq 10$ zurückgegeben.

> **Warnung**
>
> Der Net Promoter Score hat im Management-Umfeld eine grosse Verbreitung gefunden. Die Verwendung dieser Skala ist aber umstritten (Keiningham et al., 2007). Speziell die proklamierte Aussagekraft lässt sich nicht belegen (Fisher & Kordupleski, 2019; Grisaffe, 2007). Von der verbreiteten isolierten Verwendung des Net Promoter Score nach der Methode von Reichheld (2003) wird deshalb abgeraten.

Wird diese Eingabe als Likert-Skala verwendet, dann besteht ausser bei der Darstellung kein Unterschied zur *Einfachauswahl* oder der *Bewertung*.

5.1.10. Darstellung der Ergebnisse in Excel

Die Ergebnisse der Formulare werden in einer Excel-Arbeitsmappe als Tabelle gespeichert. Die Antworten sind in der Spalte `ID` sequenziell durchnummeriert. Zusätzlich enthält die Tabelle zwei sog. *Zeitstempel* mit Datum und Uhrzeit des Beginns (`Startzeit`) und des Endes (`Fertigstellungszeit`) der Datenerfassung eines Formulars. Optional können die Namen und E-Mail-Adressen der ausfüllenden Person miterfasst werden.

Nach diesen Basisinformation folgen die einzelnen Antworten für die Formularelemente. Die Werte in der gleichen Zeile entsprechen dabei den Werten aus dem gleichen Formular. Die Spaltenüberschriften sind Standardmässig der Fragetext.

5.2. Daten direkt eingeben

Bei der direkten Eingabe von Daten muss zuerst die Ergebnistabelle manuell erstellt werden, um damit die Daten zu sammeln.

Damit die Daten einheitlich erfasst werden, benötigt die Tabelle ein Schema. In Excel wird diese Schema über die Tabellenüberschriften definiert und kann bei einer späteren Verarbeitung weiter angepasst werden (s. Kapitel 7).

Ein einfaches Datenschema umfasst nur die Spaltenüberschriften, die erfasst werden sollen. Dazu müssen die Tabellenüberschriften definiert werden. Die Tabellenüberschriften werden in der ersten Zeile der Tabelle eingetragen. Überschriften für weitere Merkmale können zu einem späteren Zeitpunkt noch ergänzt werden. Anschliessend werden die Überschriften markiert und es wird für diesen Bereich eine **Tabelle** eingefügt. Hierbei muss die Option `Tabelle hat Überschriften` aktiviert werden (s. Abbildung 5.2).

Daten werden der Tabelle hinzugefügt, indem in der Zeile unterhalb der Tabelle neue Werte eingetragen werden. Die Tabellenstruktur mit dem Schema wird automatisch erweitert.

5.2.1. Schemadefinition durch Datenüberprüfung

Die einfache Schemadefinition stellt nicht sicher, dass die Werte einheitlich abgelegt werden. Das kann später die Datenerfassung und die Datenverarbeitung behindern. Wurde die Datentabelle mit einem Überprüfungsschema erstellt, dann können die Werte in die Tabelle eingetragen und bei der Eingabe validiert werden.

Die Schemadefinition durch Datenüberprüfung erlaubt die Wertebereiche der gemessenen Merkmale vorab festzulegen. Dadurch lassen sich potentielle Fehler bereits bei der Dateneingabe vermeiden. Hierzu ist es notwendig, die **Wertebereiche** und die zugehörige **Datenklasse** einer Spalte zu bestimmen.

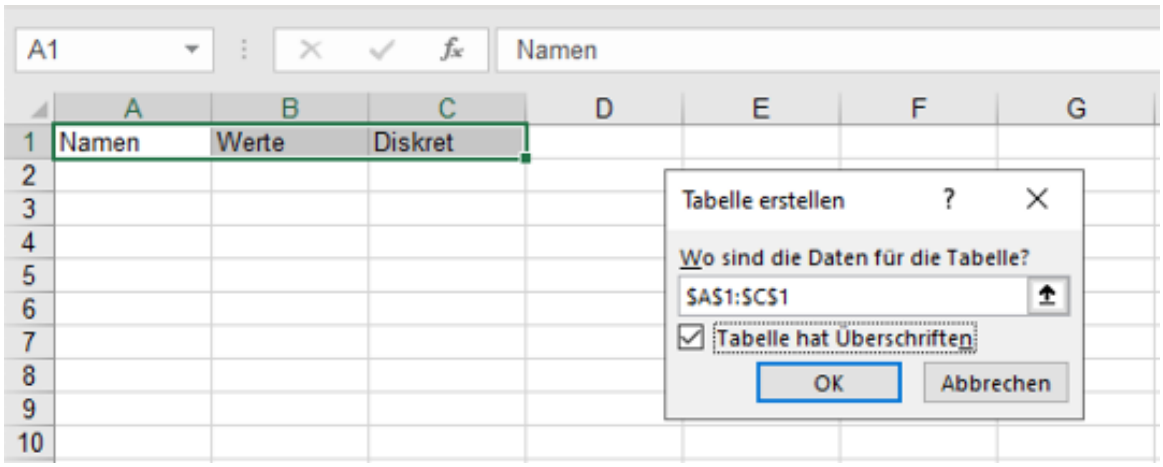


Abbildung 5.2.: Tabelle erstellen

Praxis

Das Erstellen eines Schemas zur Dateneingabe ist in Excel zwar aufwändig, dennoch sollte dieser Schritt nicht übersprungen werden, weil die Datenüberprüfung die Datenqualität verbessert und die Datenverarbeitung erheblich vereinfacht.

In Excel müssen Vektoren durch die Option *Datenüberprüfung* definiert werden. Die Option *Datenüberprüfung* findet sich im Ribbon unter *Daten* (s. Abbildung 5.3) im Abschnitt *Datentools* (s. Abbildung 5.4).

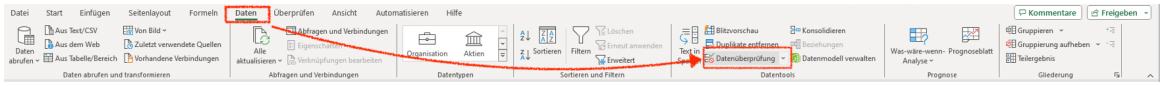


Abbildung 5.3.: Ribbon Datenüberprüfung

Abbildung 5.4.: Datentools Details

Die Schemadefinition erfolgt in drei zusätzlichen Schritten:

2. Es wird eine *Platzhalterzeile* eingefügt, indem die Zelle unterhalb der ersten Überschrift ausgewählt wird. Die Platzhalterzeile wird benötigt, um die Datenüberprüfung zu definieren.

3. Für jeden Vektor wird mit der Option *Datenüberprüfung* eine Datenüberprüfung für den gewünschten Wertebereich definiert.
4. Die Platzhalterzeile wird gelöscht.

5.2.2. Funktionsweise der Datenüberprüfung

Die Datenüberprüfung ist eine Funktion, die auf einen Bereich angewendet wird, wobei der Bereich eine einzelne Zelle, einer Zeile oder eine Spalte sein kann. Für das Schema einer Tabelle wird die Datenüberprüfung für eine Spalte eingerichtet.

Die Datenüberprüfung wird Zellenweise konfiguriert, wobei die erste Zelle eines Bereichs die Referenzzelle ist. Die Datenüberprüfung wird auf die Referenzzelle angewendet und diese Konfiguration wird auf alle Zellen des Bereichs übertragen und automatisch so angepasst, dass die aktuelle Zelle überprüft wird. Dabei sind ein paar Besonderheiten zu beachten.

- Die Datenüberprüfung muss mit konstanten Adressen konfiguriert werden. Es muss also die doppelte Dollar-Adressierung (z.B. `$A$2`) verwendet werden.
- Wird nicht die Referenzzelle als Adresse verwendet, dann erfolgt die Datenüberprüfung mithilfe der Werte der angegebenen Adresse. In den folgenden Zellen wird dann der relative Versatz zur Referenzzelle verwendet.
- Konstante Werte müssen in der Datenüberprüfung explizit angegeben und nicht über Bezüge verwiesen werden.

Excel bringt einige vordefinierte Überprüfungen mit, die über die Dropdown-Liste ausgewählt werden können. Die meisten Überprüfungen sind für Zahlenintervalle definiert. Die Standardeinstellung `Jeden Wert` ist gleichbedeutend mit keiner Überprüfung, weil alle Werte als Eingabe akzeptiert werden. Generelle Überprüfungen von Datentypen sind nicht vorgegeben und müssen über die Option `Benutzerdefiniert` definiert werden.

> **Hinweis**
>
> Wird die Datenüberprüfung auf eine Zelle in einer Tabelle angewendet und in der jeweiligen Spalte existieren noch keine anderen Werte, dann wird die Datenüberprüfung für die gesamte Spalte übernommen. Stehen in der Tabelle bereits Werte, dann müssen zuerst alle Werte in der Spalte ausgewählt werden, bevor die Datenüberprüfung eingerichtet wird. Neue Werte übernehmen die Datenüberprüfung der vorangehenden Zeile.

5.2.3. Datenüberprüfung für Zahlen, Zeichenketten oder Wahrheitswerte

Eine generelle Überprüfung für die fundamentalen Datentypen ist in Excel nicht vorgesehen. Solche Überprüfungen müssen über die Option `Benutzerdefiniert` festgelegt werden.

Im Eingabefeld Formel kann eine beliebig komplexe Formel stehen. Die Formel muss einen Wahrheitswert zurückgeben. Wird der Wahrheitswert `WAHR` zurückgegeben, dann wird der

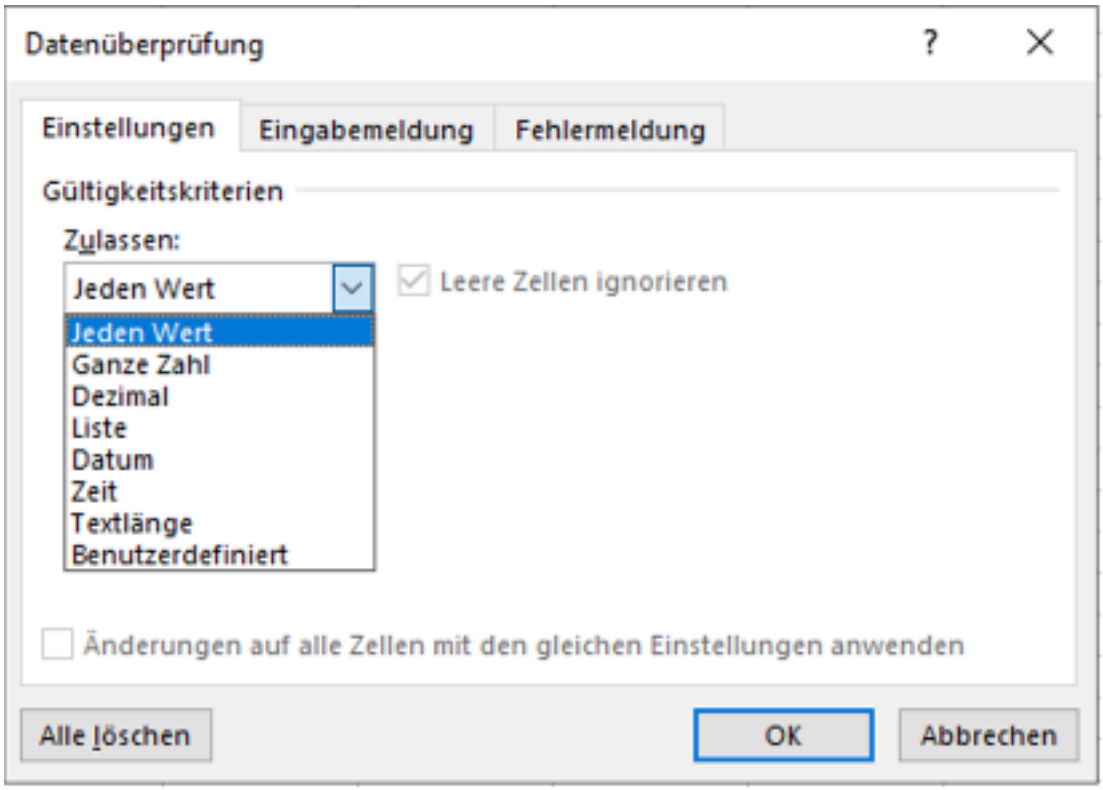


Abbildung 5.5.: Dialog Datenüberprüfung mit allen Überprüfungsoptionen

Wert akzeptiert. Wird der Wahrheitswert `FALSCH` zurückgegeben, dann wird der Wert nicht akzeptiert.

Um die fundamentalen Datentypen zu überprüfen, können die Informationsfunktionen `ISTZAHL()`, `ISTTEXT()` und `ISTLOG()` verwendet werden (s. Tabelle 5.1). Diese Funktionen geben `WAHR` zurück, wenn der Wert dem angegebenen Datentyp entspricht. Ansonsten geben sie `FALSCH` zurück.

Tabelle 5.1.: Überprüfungsausdrücke für die fundamentalen Datentypen

Datentyp	Überprüfungsausdruck
Zahlen	`=ISTZAHL($A$2)`
Zeichenketten	`=ISTTEXT($A$2)`
Wahrheitswerte	`=ISTLOG($A$2)`

5.2.4. Datenüberprüfung für ganze Zahlen

Die Überprüfung ganzer Zahlen ist in Excel über die Option `Ganze Zahl` möglich. Diese Option überprüft, ob der Wert eine ganze Zahl ist. Diese Option kann aber nicht direkt für alle ganze Zahlen eingesetzt werden. Um alle zulässigen ganzen Zahlen zu erfassen muss unter der Option `Daten` der Punkt `größer als` ausgewählt werden. Anschliessend muss als `Minimum` der Wert `-9.99999999999999E+307` eingetragen werden (s. Abbildung 5.6). Dieser Wert ist der kleinste Wert, der in Excel als (ganze) Zahl erfasst werden kann (Microsoft Support, 2023a).

5.2.5. Datenüberprüfung für ordinalskalierte Wertebereiche

Für ordinalskalierte Wertebereiche sollten die Werte am Besten als Ganzzahlen erfasst werden. Damit zu einem späteren Zeitpunkt diese Zahlen den eigentlichen Werten zugeordnet werden können, wird eine Zuordnungstabelle benötigt. Diese Zuordnungstabelle wird in einer separaten Tabelle erstellt. Diese Tabelle besteht aus zwei Spalten: Dem Zahlenwert und dem zugeordneten Wert. Die Ordnung der Werte ergibt sich aus der Ordnung der zugeordneten Zahlen.

> **!** Wichtig
>
> Häufig muss bei ordinalskalierten Wertebereichen die Option einer nicht zutreffenden Antwort berücksichtigt werden. In diesem Fall muss eine zusätzlicher Zahlenwert ausserhalb der regulären Ordnung definiert werden. Dieser Wert wird dann der nicht zutreffenden Antwort (z.B. `NA` oder `nicht zutreffend`) zugeordnet.

Damit der Wertebereich korrekt eingeschränkt wird, muss die Option `Liste` verwendet werden. Die `Quelle` entspricht der Spalte mit den zugeordneten Zahlenwerten in der Zuordnungstabelle. Die Option `Zellendropdown` kann deaktiviert werden.

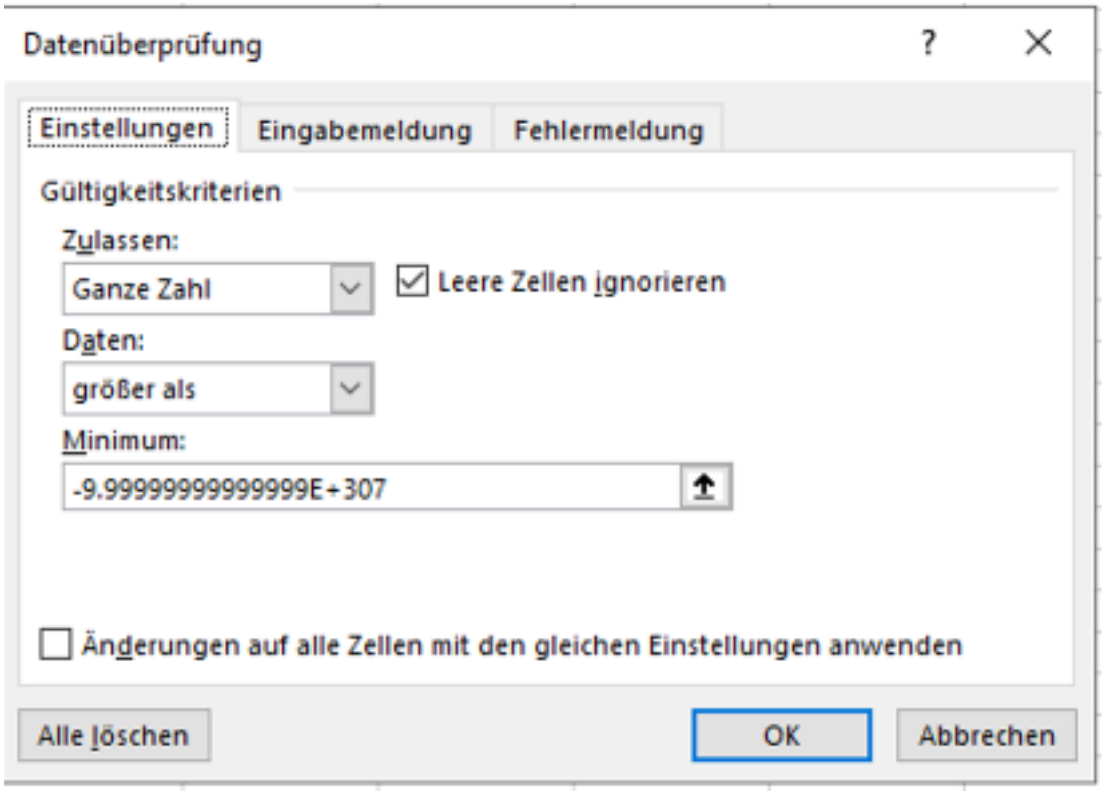


Abbildung 5.6.: Dialog Datenüberprüfung für alle ganzen Zahlen

> **Achtung**
>
> Die Datenüberprüfung kann nicht mit Tabellenadressen umgehen. Deshalb müssen die gültigen Werte als absolute Arbeitsblattadressen angegeben werden (s. Abbildung 5.7).

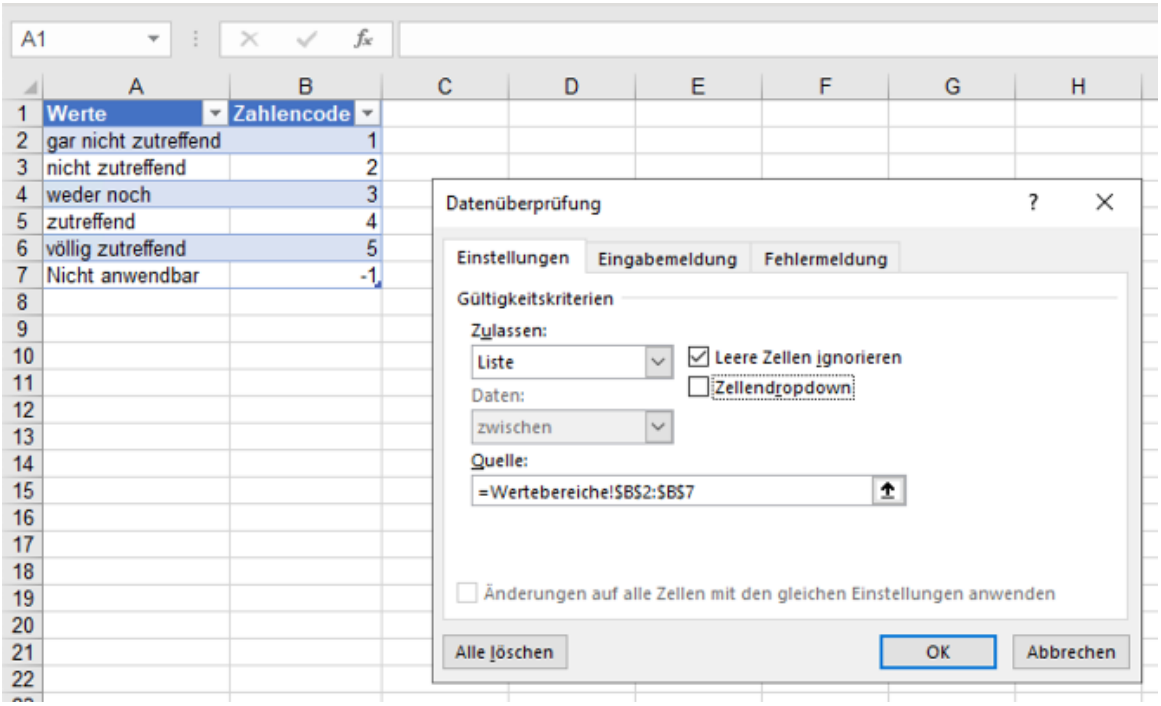


Abbildung 5.7.: Datenüberprüfung mit Zuordnungstabelle für ordinalskalierte Wertebereiche

5.2.6. Datenüberprüfung für nominalskalierte Wertebereiche

Die Überprüfung nominalskalierter Wertebereiche wird in Excel durch die Option `Liste` realisiert. Die Option `Liste` erlaubt die Auswahl von Werten aus einer Liste in der gleichen Arbeitsmappe. Diese Liste ist einfach ein Bereich mit Werten, eine Spalte in einer anderen Tabelle oder eine benannte Liste (s. Kapitel 8). Die Liste muss in einer Spalte definiert werden, weil die Datenüberprüfung nur auf Spalten angewendet werden kann. Deshalb ist es nicht möglich, eine zeilenorientierte Liste für die Datenüberprüfung anzuwenden.

Für nominalskalierte Wertebereiche werden alle zulässigen Werte in einer Spalte festgelegt. Anschliessend wird im Dialog Datenüberprüfung unter `Quelle` der Bereich mit den zulässigen Werten angegeben.

> **Tipp**
>
> Im Gegensatz zu ordinalskalierten Wertebereichen, sollten nominalskalierte Wertebereiche nicht als Zahlen kodiert werden, sondern als Zeichenketten erfasst werden.

Bei der Dateneingabe wird anschliessend ein Auswahlliste mit den zulässigen Werten, so dass die Werte nicht mehr direkt eingetippt werden müssen. Diese Auswahlliste kann unterbunden werden, wenn die Option `Zellendropdown` deaktiviert wird.

6. Datentypen

6.1. Fundamentale Datentypen

Excel kennt sechs fundamentale Datentypen.

- Zahlen
- Zeichenketten
- Wahrheitswerte
- Fehlerwerte
- Formeln
- leere Zelle

6.1.1. Zahlen

Zahlen werden in Excel immer als Gleitkommazahlen behandelt. Excel kennt keine ganzen Zahlen. Wenn eine Zahl in eine ganze Zahl umgewandelt wird (z.B. durch die Funktion `GANZZAHL`), wird lediglich der Nachkommaanteil der Zahl auf `0` gesetzt.

Manche Excel Funktionen arbeiten nur mit ganzen Zahlen. In diesen Fällen wird der Nachkommaanteil der Zahl automatisch abgeschnitten.

Durch die Verwendung von Gleitkommazahlen können in Excel Zahlen mit einer Genauigkeit von 15 signifikanten Stellen dargestellt werden. Werden zwei Zahlen addiert, und das Ergebnis mehr als 15 signifikante Stellen hätte, werden alle Stellen ab der 15. signifikanten Stelle abgeschnitten. Excel versucht diese Fehler möglichst zu vermeiden.

Der Datentyp *Zahlen* wird in Excel mit der Funktion `ISTZAHL()` geprüft. Die Funktion `ISTZAHL()` liefert `WAHR`, wenn der Wert ein Zahl ist, und sonst `FALSCH`.

6.1.2. Zeichenketten

Zeichenketten heissen in Excel **Text**. Zeichenketten werden von Excel automatisch gewählt, wenn kein anderer Datentyp für eine Eingabe erkannt wurde. Um die automatische Erkennung zu verhindern, muss der Apostroph-Dekorator (s. Abschnitt 6.2.1) verwendet werden.

Im *Formelmodus* müssen Zeichenketten in doppelte Anführungszeichen eingeschlossen werden. Im Wertemodus müssen Zeichenketten meist nicht besonders markiert werden.

Der Datentyp *Zeichenkette* wird in Excel mit der Funktion `ISTTEXT()` geprüft. Die Funktion `ISTTEXT()` liefert `WAHR`, wenn der Wert eine Zeichenkette ist, und sonst `FALSCH`.

Besondere Zeichenketten sind Adressen bzw. im Excel Jargon **Bezüge**. Die Funktion `ISTBEZUG()` prüft, ob eine Zeichenkette eine gültige Excel Adresse ist. Die Funktion `ISTBEZUG()` liefert `WAHR`, wenn die Zeichenkette eine Adresse ist, und sonst `FALSCH`.

6.1.3. Wahrheitswerte

Wahrheitswerte heissen in Excel `WAHR` und `FALSCH`. Wahrheitswerte heissen im Excel Jargon **logische Werte**. Während der Werteeingabe werden Wahrheitswerte unabhängig von der Gross- und Kleinschreibung automatisch erkannt.

Im Formelmodus dürfen Wahrheitswerte *nicht* in Anführungszeichen eingeschlossen werden, weil sonst die Symbole als Zeichenkette behandelt werden.

Der Datentyp *Wahrheitswerte* wird in Excel mit der Funktion `ISTLOG()` geprüft. Die Funktion `ISTZLOG()` liefert `WAHR`, wenn der Wert ein Wahrheitswert ist, und sonst `FALSCH`.

6.1.4. Fehlerwerte

Fehlerwerte ist in Excel ein besonderer Datentyp, um Fehler zu signalisieren. Fehler beginnen immer mit einem Gatter (`#`), das von einem sog. *Fehlerbezeichner* gefolgt wird. In der Regel werden Fehlerwerte automatisch erzeugt, es ist aber möglich, Fehlerwerte auch manuell einzugeben. Bei der manuellen Eingabe von Fehlerwerten werden nur die gültigen Fehlerbezeichner akzeptiert. Andere Symbolfolgen werden als Zeichenketten interpretiert.

Excels gültige Fehlerwerte sind `#NV`, `#NULL!`, `#WERT!`, `#BEZUG!`, `#DIV/0!`, `#ZAHL!`, `#NAME?`, `#KALK!`, `#ÜBERLAUF!`, `#DATEN_ABRUFEN!` und `#ZAHL!`.

Der Wert `#NV` wird in Excel verwendet, um einen ungültigen Wert zu verweisen. Dieser Wert ist nicht gleichbedeutend mit fehlenden Werten.

Der Datentyp *Fehlerwert* wird in Excel mit der Funktion `ISTFEHLER()` geprüft. Die Funktion `ISTFEHLER()` liefert `WAHR`, wenn der Wert ein Fehlerwert ist, und sonst `FALSCH`. Um einen bestimmten Fehlerwert zu prüfen, kann die Funktion `FEHLER.TYP()` mit dem entsprechenden Fehlerbezeichner als Argument verwendet werden. Diese Funktion gibt einen eindeutigen Zahlenwert für den Fehler zurück. Ist der Wert kein Fehlerwert, dann wird der Fehler `#NV` zurückgegeben.

6.1.5. Formeln

Formeln sind ein eigener Datentyp, die mit dem Gleich-Dekorator (`=`) beginnen. Eine Formel besteht immer aus einem Wert, einem Funktionsaufruf oder einer Kombination von Werten und Funktionsaufrufen, die durch Operatoren verknüpft wurden.

Der Datentyp *Formel* wird in Excel mit der Funktion `ISTFORMEL()` geprüft. Die Funktion `ISTFORMEL()` liefert `WAHR`, wenn der Wert eine Formel ist, und sonst `FALSCH`.

Formeln sind in Excel ein besonderer Datentyp, denn eine Zelle mit einer Formel hat immer zwei Datentypen. Der erste Datentyp ist der Datentyp *Formel*, der zweite Datentyp ist der Datentyp des Formelergebnisses.

6.1.6. Leere Zellen

Leere Zellen sind Zellen, die keinen Wert enthalten. Leere Zellen zeigen fehlende Werte in Excel an.

Die **leere Zelle** ist für Excel ein eigener Datentyp und wird mit der Funktion `ISTLEER()` geprüft. Die Funktion `ISTLEER()` liefert `WAHR`, wenn der Wert eine leere Zelle ist, und sonst `FALSCH`.

> **Warnung**
>
> Eine leere Zelle und eine leere Zeichenkette lassen sich mit dem Auge nicht unterscheiden. Die Funktion `ISTLEER()` liefert für eine leere Zeichenkette `FALSCH` zurück, weil die Zelle einen Wert enthält.

Leere Zellen als Ergebnis einer Formel werden von Excel in die Zahl `0` umgewandelt. Im Kapitel Abschnitt 8.2 wird gezeigt, wie sich leere Zellen von Zellen mit dem Wert `0` unterscheiden lassen. Im Abschnitt 8.4.1 wird auf leere Zellen als Funktionsergebnisse ausführlich eingegangen.

6.2. Dekoratoren

Definition 6.1. Ein **Dekorator** ist Sprachelement einer Programmiersprache, mit dem die normale Interpretation von Symbolen verändert werden kann.

Excel hat zwei Dekoratoren, als erstes Symbol einer Zelle stehen müssen. D.h. es dürfen auch keine Leerzeichen vor einem Dekorator stehen. Die beiden Dekoratoren sind:

- Der Apostroph-Dekorator (`'`)
- Der Gleich-Dekorator (`=`)

6.2.1. Apostroph-Dekoraktor

Der Apostroph-Dekorator (`'`) erzwingt, dass die nachfolgenden Symbole als Zeichenkette interpretiert werden müssen.

Mithilfe des Apostroph-Dekorators wird die automatische Typerkennung für die laufende Eingabe deaktiviert. Auf diese Weise lassen sich Zeichenketten eingeben, die nur aus Ziffern bestehen oder Zeichenketten, die Datentumswerten ähneln.

Der Apostroph-Dekorator muss zwingend verwendet werden, wenn auf eine Zeichenkette eine der folgenden Bedingungen zutrifft.

- Die Zeichenkette beginnt mit einem Gleichheitszeichen (`=`), einem Pluszeichen (`+`), einem Minuszeichen (`-`) oder einem Prozentzeichen (`%`).
- Die Zeichenkette besteht nur aus Ziffern, dem Dezimaltrenner (`.`) oder dem Tausendertrenner (`‘`) enthält
- Die Zeichenkette ähnelt der wissenschaftliche Notation für Zahlen mit und ohne Vorzeichen für den Exponenten. Z.B. `5.E3`.
- Die Zeichenkette ähnelt einem Datum.
- Die Zeichenkette entspricht einen Fehlerwert, unabhängig von der Schreibweise. Z.B. `#nv`.
- Die Zeichenkette entspricht einem Wahrheitswert (`WAHR` oder `FALSCH`) unabhängig von der Schreibweise.

Das Apostroph kann entfallen, wenn die aktive Zelle vorher bereits als Datentyp `Text` formatiert wurde. In diesem Fall verlieren alle Symbole ihre spezielle Bedeutung. In solche Zellen können keine Formeln eingegeben werden.

6.2.2. Gleich-Dekorator

Der Gleich-Dekorator (`=`) zeigt an, dass die nachfolgenden Symbole als Formel interpretiert werden müssen.

Bei der Eingabe des Gleich-Dekorators wechselt Excel für die laufende Eingabe in den Formelmodus. Mit beenden der Eingabe mit der Eingabetaste wird der Formelmodus wieder verlassen und das Ergebnis der eingegebenen Formel wird in der Zelle angezeigt.

6.3. Komplexe Datenstrukturen

Excel kennt zwei komplexe Datenstrukturen:

- Bereiche
- Tabellen

Excels komplexe Datenstrukturen müssen einen fundamentalen Datentyp haben und können keine komplexen Datenstrukturen schachteln.

6.3.1. Bereiche: Vektoren und Matrizen

Excels Grundstruktur ist das Rechteck, dass durch die markierten Zellen entsteht. Dieses Rechteck heisst im Excel Jargon ein **Bereich**.

Obwohl Excel vektorähnliche Bereiche kennt, ist es sinnvolle sich Excels Bereiche immer als Matrizen vorzustellen.

- Ein Bereich mit nur einer Zelle ist eine 1x1- oder 0-dimensionale Matrix.
- Ein Bereich mit nur einer Zeile oder einer Spalte ist für Excel eine 1xn- oder 1-dimensionale Matrix. Diese speziellen Matrizen werden in der Regel als *Vektoren* bezeichnet.
- Ein Bereich mit mehreren Zeilen und Spalten ist eine nxm- oder 2-dimensionale Matrix.

Funktionen können Bereiche als Ergebnis haben. In diesem Fall werden die restlichen Werte zeilen und spaltenweise unterhalb bzw. rechts von der entsprechenden Formel ausgegeben.

6.3.2. Tabellen

Tabellen sind benannte Bereiche, die Vektoren enthalten. Tabellen bestehen aus Spalten, in denen die Werte als Vektoren stehen. Excel Tabellen haben immer Überschriften.

Excel kennt zwei Arten von Tabellen:

- Wertetabellen
- Pivot-Tabellen.

Definition 6.2. Wertetabellen sind Listen, die Listen mit gleicher Länge schachteln.

Wertetabellen entsprechen ungefähr einem *Datenrahmen* (engl. *data-frame*) in anderen Programmiersprachen. Der Unterschied zu einem Datenrahmen ist, dass Excels Wertetabellen *keine Vektoren* erzwingen. Deshalb können in der gleichen Spalte einer Wertetabelle verschiedene Datentypen gemischt werden.

Definition 6.3. Pivot-Tabellen sind ein Werkzeug zum *interaktiven Zusammenfassen* von Daten.

Pivot-Tabellen erleichtern einfache Datenanalysen. Pivot-Tabellen entsprechen einer tabellarischen Darstellung der Daten. Ihre Funktion ist auf wenige analytische Funktionen beschränkt und die dargestellten Werte lassen sich nur umständlich weiterverarbeiten.

Pivot-Tabellen sind Arbeitsmappenelemente aber ***keine Datenstrukturen***. Deshalb werden Pivot-Tabellen in diesem Buch ähnlich einer Visualisierung als *Darstellungsform* behandelt.

6.4. Adressierung von Datenstrukturen

Definition 6.4. Ein **Bezug** ist die Adresse eines Bereichs.

In Excel gibt es zwei Arten von Bezügen:

- Arbeitsblattadressen
- Tabellenadressen

6.4.1. Arbeitsblattadressen

Definition 6.5. Eine **Arbeitsblattadresse** enthält die Koordinaten eines Bereichs auf einem Arbeitsblatt.

Arbeitsblattadressen besteht aus drei Teilen:

- Arbeitsblattname
- Bereichsbeginn
- Bereichsende

Der Bereichsbeginn verweist immer auf die linke obere Zelle des Bereichs. Der Bereichsende verweist immer auf die rechte untere Zelle des Bereichs. Eine Zelle wird immer durch den Spaltenindex (Buchstabe) und den Zeilenindex (Zahl) identifiziert. Werden die Koordinaten des Bereichsbeginns und des Bereichsendes bei der Eingabe vertauscht, dann wird der Bereich automatisch korrigiert.

Beispiel 6.1 (Arbeitsblattadresse)**.**

```
Tabelle1!A1:C3
```

Beispiel 6.1 verweist auf den Bereich mit drei Spalten und drei Zeilen auf dem Arbeitsblatt `Tabelle1` beginnend mit der Zelle `A1` und endend mit der Zelle `C3`.

Oft werden Arbeitsblattadressen nicht vollständig sondern gekürzt angegeben. Es gibt zwei Möglichkeiten, um Arbeitsblattadressen zu kürzen:

- Werden Bereiche auf dem gleichen Arbeitsblatt adressiert, dann kann der Arbeitsblattname weggelassen werden.
- Wird ein Bereich mit nur einer Zelle adressiert, dann wird das Bereichsende weggelassen.

Weil Arbeitsblattadressen von vielen interaktiven Excelkommandos verwendet werden, gibt es zwei Arten von Arbeitsblattadressen:

- Relative Adressen
- Absolute Adressen

Die Art der Adresse legt fest, wie ein interaktives Kommando mit einer Adresse umgehen soll. Die populärste interaktive Funktion ist das *Autoauffüllen*. Dabei wird eine Zelle mit einer Formel interaktiv auf einen Bereich von Zellen übertragen.

> **Warnung**
>
> Das Autoauffüllen ist eine einfache und beliebte Methode, um Formeln in Excel auf verschiedene Werte wiederholt anzuwenden. Bis 2019 war das Autoauffüllen die einzige Möglichkeit für die Datentransformation.
> Die relative und absolute Adressierung ist eine wichtige Voraussetzung für das Autoauffüllen. Leider ist das Autoauffüllen auch die Ursache für viele Fehler beim Umgang mit Excel.

> **Praxis**
>
> In Excel365 kann das Autoauffüllen durch vektorisierte Funktionen fast vollständig ersetzt werden. Dadurch lassen sich viele Excel-typische Fehler vermeiden. Dadurch ist die Unterscheidung zwischen der relativen und absoluten Adressierung nicht mehr so wichtig.

6.4.1.1. Relative Adressen

Definition 6.6. Eine **relative Adresse** ist eine Adresse eines Bereichs, die *veränderlich* ist.

Relative Adressen werden in Excel von interaktiven Funktionen, wie dem *Autoauffüllen* verwendet, um die Adressen automatisch anzupassen. Eine relative Adresse wird relativ zur aktuellen Zelle angepasst.

Ein Beispiel für eine relative Adresse ist `A1`. Diese Adresse bezeichnet die Zelle, die sich in der ersten Zeile und der ersten Spalte auf dem aktuellen Arbeitsblatt befindet. Wird die adressierte Zelle interaktiv nach unten aufgefüllt, dann wird die Adresse automatisch zu `A2`, `A3`, usw. angepasst. Wird die adressierte Zelle nach rechts aufgefüllt, dann wird die Adresse automatisch zu `B1`, `C1`, usw. angepasst.

6.4.1.2. Absolute Adressen

Definition 6.7. Eine **absolute Adresse** ist eine Adresse eines Bereichs, die ganz oder teilweise *unveränderlich* ist.

Der unveränderliche Teil einer Arbeitsblattadresse wird mit einem Dollarzeichen (`$`) markiert. Dieser Teil der Adresse wird bei der Anpassung der Adresse nicht verändert. So lassen sich Adressen angeben, die durch interaktive Kommandos nicht verändert werden.

Ein Beispiel für eine absolute Adresse ist `$A$1`. Diese Adresse bezeichnet die Zelle, die sich in der ersten Zeile und der ersten Spalte auf dem aktuellen Arbeitsblatt befindet. Wird die adressierte Zelle interaktiv horizontal oder vertikal aufgefüllt, dann wird die Adresse nicht angepasst.

Auf diese Weise lassen sich konstante Werte in Formeln einbauen.

6.4.2. Tabellenadressen

Spalten und einzelne Werte können über die Tabellenadressierung abgefragt werden (Microsoft Support, 2023b). Das Ergebnis einer solchen Adressierung ist immer ein *dynamisches Feld* bzw. ein *dynamischer Bereich* (s. Abschnitt 8.1.1).

> **Hinweis**
>
> Jede Tabellenadresse kann auch als Arbeitsblattadresse dargestellt werden. Umgekehrt ist dies nicht möglich.

Eine Tabellenadresse besteht aus zwei Teilen:

- Dem Tabellennamen
- Dem Spaltennamen

Beispiel 6.2 (Tabellenadresse)**.**

```
Tabelle1[Spalte1]
```

Das Beispiel 6.2 verweist auf die Spalte `Spalte1` der Tabelle `Tabelle1`.

> **Wichtig**
>
> Die Namen von Tabellen sind *unabhängig* den Arbeitsblattnamen. In der gleichen Arbeitsmappe darf jeweils nur eine Tabelle und nur ein Arbeitsblatt mit dem gleichen Namen existieren. Es ist aber möglich, dass eine Tabelle und ein Arbeitsblatt den gleichen Namen haben. Das kann zu Verwirrungen führen, denn die Adresse `Tabelle1[Spalte1]` und die Adresse `Tabelle1!A:A` verweisen nicht zwingend auf die gleiche Spalte, denn eine Tabelle muss nicht auf einem Arbeitsblatt mit dem gleichen Namen stehen!

Tabellenadressen können auch gekürzt werden:

- Werden Spalten in der gleichen Tabelle angesprochen, dann kann der Tabellenname weggelassen werden.
- Soll die gesamte Tabelle angesprochen werden, dann kann der Spaltenname weggelassen werden.

6.4.2.1. Tabellenbereiche

Um mehrere Spalten einer Tabelle anzusprechen kann der Bereichsoperator (`:`) wie bei der Arbeitsblattadressierung verwendet werden. Zusätzlich müssen die Spalten in ein weiteres Paar eckiger Klammern eingeschlossen werden.

Beispiel 6.3 (Tabellenbereich)**.**

```
Tabelle1[[Spalte1]:[Spalte3]]
```

Beispiel 6.3 verweist auf alle Spalten zwischen `Spalte1` und `Spalte3` der Tabelle `Tabelle1`.

> **Warnung**
>
> Werden zu einem späteren Zeitpunkt neue Spalten zwischen den adressierten Spalten hinzugefügt, dann werden die neuen Spalten automatisch in die Adressierung einbezogen.

> **Wichtig**
>
> Es ist nicht möglich mehrere Spalten einer Tabellen *gleichzeitig* gezielt zu adressieren, wenn diese nicht unmittelbar nebeneinander stehen. Diese Adressierung ist unmöglich, weil Excel nur rechteckige Bereiche adressieren kann und solche Spalten keinen *rechteckigen Bereich* bilden.
> Die Adressierung aus Beispiel 6.3 kann nicht angepasst werden, so dass nur `Spalte1` und `Spalte3` adressiert werden, ausser die Spalten `Spalte1` und `Spalte3` stehen direkt nebeneinander.

6.4.2.2. `Diese Zeile`-Operator

Einzelne Zeilen einer Tabelle können mit dem `Diese Zeile`-Operator adressiert werden. Der `Diese Zeile`-Operator wird mit dem Wert `[#Diese Zeile]` oder mit `@` angegeben. Dieser Operator reduziert die Tabelle auf eine einzelne Zeile und wählt anschliessend die gewünschte Spalte aus. `Diese Zeile` bedeutet für Excel die aktuelle Zeile des Arbeitsblatts.

Beispiel 6.4 (`Diese Zeile`-Operator für eine Tabellenzelle)**.**

```
= Tabelle[@Spalte]
```

oder

```
= @Tabelle[Spalte]
```

Der Operator ist speziell für die Verwendung in Formeln in Tabellenzellen gedacht. Obwohl der Operator in allen Formeln eingesetzt werden kann, sollte der Einsatz auf Formeln beschränkt werden, die sich in einer Tabelle befinden.

> **Warnung**
>
> Der `Diese Zeile`-Operator kann nicht auf Tabellen angewendet werden, um eine ganze Zeile zu adressieren. Es darf immer nur eine einzelne Zelle adressiert werden. Die Adressierung `@Tabelle` ist also ungültig.

> **Warnung**
>
> Der `Diese Zeile`-Operator ist immer relativ zum Arbeitsblatt und nicht relativ zur aktuellen Tabelle. Es muss nur der Zeilenindex gleich sein, die Zielzelle kann sich auf einem anderen Arbeitsblatt befinden.

Wird der `Diese Zeile`-Operator in Tabellen unterschiedlicher Länge verwendet, dann können zwei Fälle eintreten:

1. Ist die adressierte Tabelle kürzer als die adressierende Tabelle, dann wird für die Zeilen, die keine Entsprechung in der adressierten Tabelle haben, der Fehler `#WERT!` zurückgegeben.
2. Ist die adressierte Tabelle länger als die adressierende Tabelle, dann werden alle überzähligen Zeilen *ignoriert*.

> **Praxis**
>
> Der `Diese Zeile`-Operator kann nur auf eine einzelne Zelle in der gleichen Zeile auf einem Arbeitsblatts angewendet werden. Wegen dieser Einschränkung des `Diese Zeile`-Operators sollten alle Tabellen so gestaltet werden, dass sie in der ersten Zeile des Arbeitsblatts beginnen. Dadurch kann der `Diese Zeile`-Operator in allen Tabellen verwendet werden.

6.4.2.3. Überschriften adressieren

Eine Excel Tabelle hat immer Spaltenüberschriften. Diese Überschriften können ebenfalls über die Tabellenadressierung adressiert werden. Dazu wird als Spaltenname der Wert `#Kopfzeilen` verwendet.

Beispiel 6.5 (Kopfzeilen einer Tabelle adressieren)**.**

```
Tabelle1[#Kopfzeilen]
```

Um gezielt Kopfzeilen zu adressieren, kann die Adressierung aus Beispiel 6.5 mit der Adressierung aus Beispiel 6.3 kombiniert werden (Beispiel 6.6).

Beispiel 6.6 (Kopfzeilen einer Tabelle adressieren)**.**

```
Tabelle1[[#Kopfzeilen];[Spalte1]:[Spalte3]]
```

6.4.2.4. Absolute Adressierung in Tabellen

Ein Tabellenbereich ist immer absolut adressiert. Wird aber nur eine einzelne Spalte einer Tabelle adressiert, dann wird die Adresse relativ adressiert. Damit beim Autoauffüllen diese Adresse nicht verändert wird, muss die Adresse als Bereich angegeben werden.

Beispiel 6.7 (Absolute Tabellenadressierung von einer Spalte)**.**

```
Tabelle1[[Spalte1]:[Spalte1]]
```

Wird eine Formel mit dieser Adresse interaktiv aufgefüllt, dann wird die Adresse nicht verändert.

7. Daten importieren

Viele Excel Arbeitsmappen kombinieren Berechnungen und Daten. Excel kann allerdings auch Daten aus anderen Quellen importieren. Quellen können andere Dateien, Datenbanken oder Web-APIs sein und in verschiedenen Formaten vorliegen. Dafür stellt Excel Parser für verschiedene Dateiformate bereit, damit die Daten importiert werden können.

Daten werden korrekt mit dem Kommando **Daten abrufen (Power Query)** importiert. Das Kommando ist in der Gruppe **Daten** im Abschnitt **Daten abrufen und transformieren** zu finden. Das Kommando *Daten abrufen* startet die sog. **Power Query** Umgebung. In dieser Umgebung können Daten aus verschiedenen Quellen importiert und vor der Bereitstellung korrigiert werden.

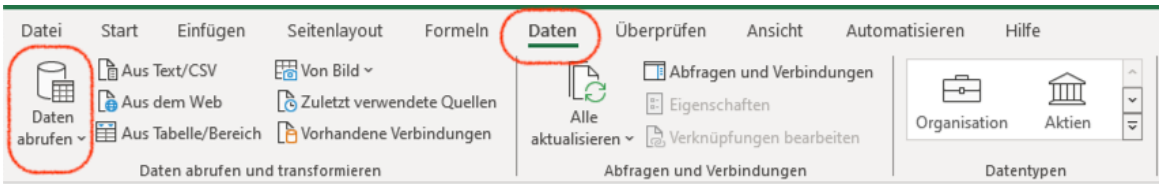


Abbildung 7.1.: Menubalken Power Query - Kommando Daten abrufen

Power Query

Power Query ist eine Umgebung für den Datenimport, die von verschiedenen Microsoft Produkten verwendet wird. Die Umgebung ist in Excel, Power BI, Power Apps und Power Automate verfügbar. Power Query stellt eine einheitliche Schnittstelle für den Datenimport bereit und basiert auf einer *Import-Beschreibungssprache*. Mit diese Sprache lassen sich Daten für die Arbeit vorbereiten, so dass viele Datenbereinigungsschritte in Excel Arbeitsmappen entfallen können.

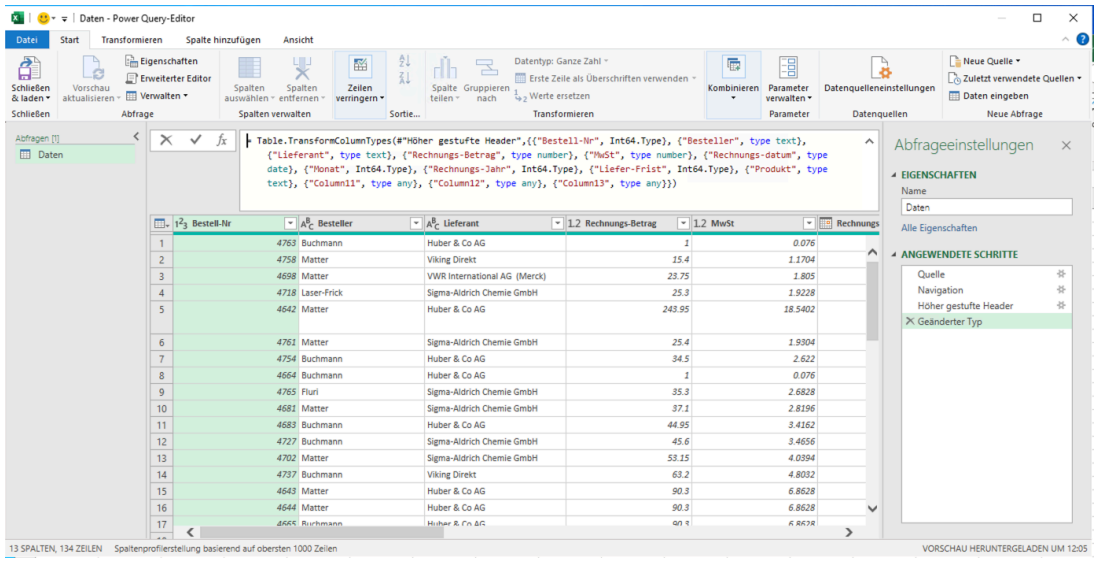


Abbildung 7.2.: Power Query Editor Fenster

Zu den zentralen Funktionen von Power Query gehören:

- Quellenmanagement
- Überschriftenerkennung
- Schemaerkennung und -transformation
- Entfernen von Duplikaten und ungültigen Werten
- Vektorisierung von Daten
- Kombinieren von Daten aus verschiedenen Quellen

Warnung

Viele tabellarische Dateiformate lassen sich direkt mit Excel öffnen. Das sollte nur mit Excel Arbeitsmappen erfolgen. Bei anderen Dateiformaten kann das direkte öffnen zu Datenverlusten oder Datenfehlern führen. Diese lassen sich in Excel nur umständlich korrigieren.
Ausserdem lassen sich direkt geöffnete Dateien nicht mehr erweitern, was die Datenerhebung erschwert.

Das Kernprinzip von *Power Query* ist das Verbinden einer Datenquelle mit einer Arbeitsmappen. Entsprechend heissen importierte Daten im Excel Jargon **Datenverbindungen**. Die Datenquelle wird dabei nicht in die Arbeitsmappe übernommen, sondern nur eine Verbindung zu der Datenquelle hergestellt. Dadurch kann eine Datenquelle in mehreren Arbeitsmappen verwendet werden und sich ändern, ohne dass die Arbeitsmappe angepasst werden muss.

> **Merke**
>
> Das Ergebnis eines Imports ist immer eine *Tabelle*.

7.1. Datenverbindungen herstellen

Eine Datenverbindung wird über das Kommando `Daten abrufen (Power Query)` ausgelöst. Das Kommando ist in der Gruppe **Daten** im Abschnitt **Daten abrufen und transformieren** zu finden. Das Kommando bereitet den Import durch Power Query vor, indem zuerst das Dateiformat und anschliessend die zugehörige Datenquellen ausgewählt wird. Nach diesen Schritten kann Excel die Daten `Laden` oder die `Daten transformieren` (s. Abbildung 7.3).

> **Praxis**
>
> Die Daten sollten nur in Ausnahmefällen unkontrolliert geladen werden. Die Daten sollten vor dem Importieren in Power Query überprüft und transformiert werden. Dadurch wird sichergestellt, dass die Daten korrekt importiert werden.

Wird die Option `Daten transformieren` ausgewählt, öffnet sich der Power Query Editor. In diesem Editor können die Daten überprüft und transformiert werden. Das Transformieren der Daten umfasst vier Arten der Transformation:

1. Datentypanpassungen
2. Spaltenauswahl
3. Datentrennung
4. Daten kombinieren

Für jede Transformation wird ein Schritt in Power Query angelegt. Die Schritte werden in der Reihenfolge der Ausführung angezeigt. Die Reihenfolge der Schritte kann verändert werden, indem die Schritte mit der Maus verschoben werden. In der Mitte des Power Query Fensters wird eine Datenvorschau angezeigt, die die Auswirkungen der Transformationen zeigt.

Standardmässig werden drei bzw. bei Excel Arbeitsmappen vier Schritte in Power Query automatisch konfiguriert:

1. `Quelle`
    - Bei Excel Arbeitsmappen folgt dem Schritt `Quelle` der Schritt `Navigation` zur Auswahl des Arbeitsblattes.
2. `Höherstufen`
3. `Geänderter Typ` (Windows) bzw. `geänderter Spaltentyp` (MacOS)

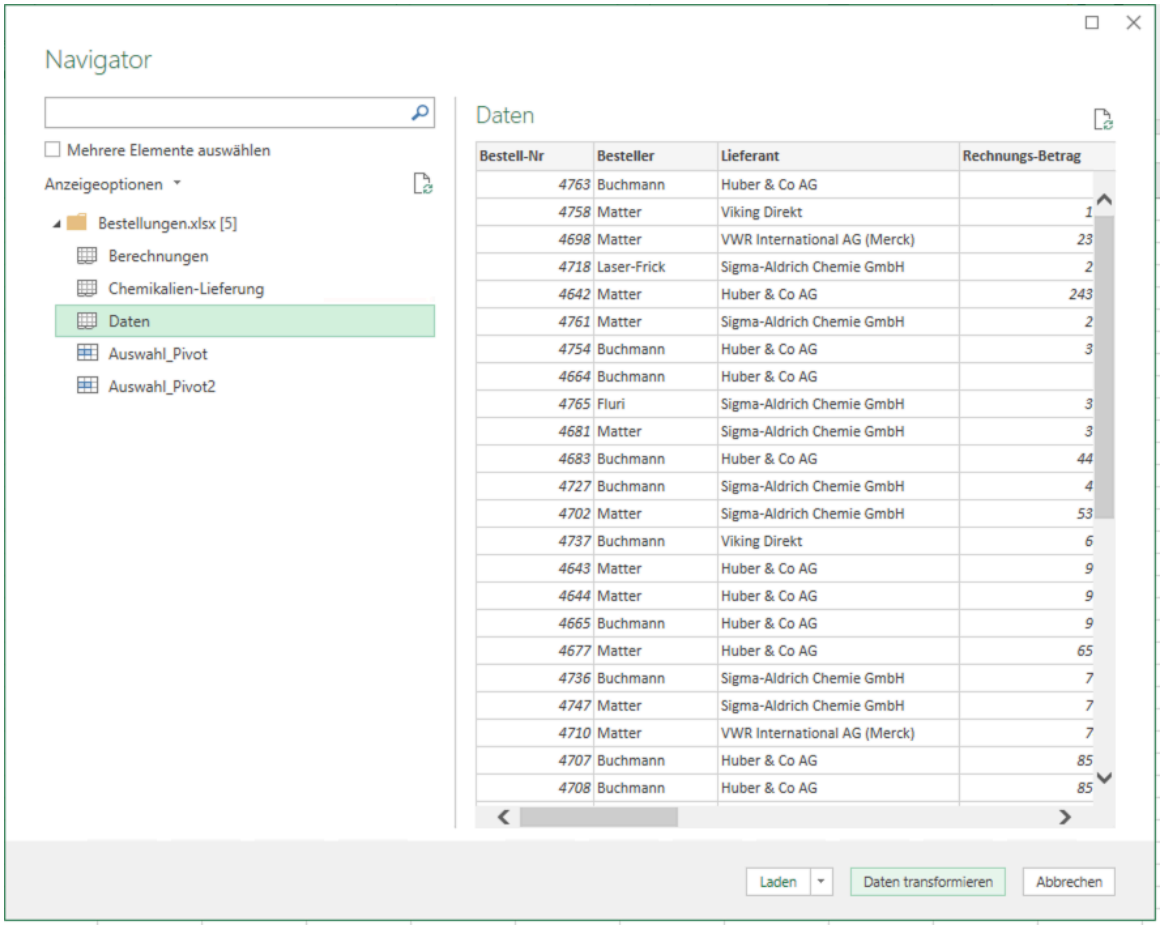


Abbildung 7.3.: Laden oder Transformieren-Auswahl beim Import einer Excel Arbeitsmappe

Der Schritt `Geänderter Typ` enthält die Schemadefinition. Power Query versucht das Schema automatisch zu erkennen. Das funktioniert besonders bei Zahlenwerten nicht immer zuverlässig. Deshalb muss das Schema immer kontrolliert und gegebenenfalls angepasst werden.

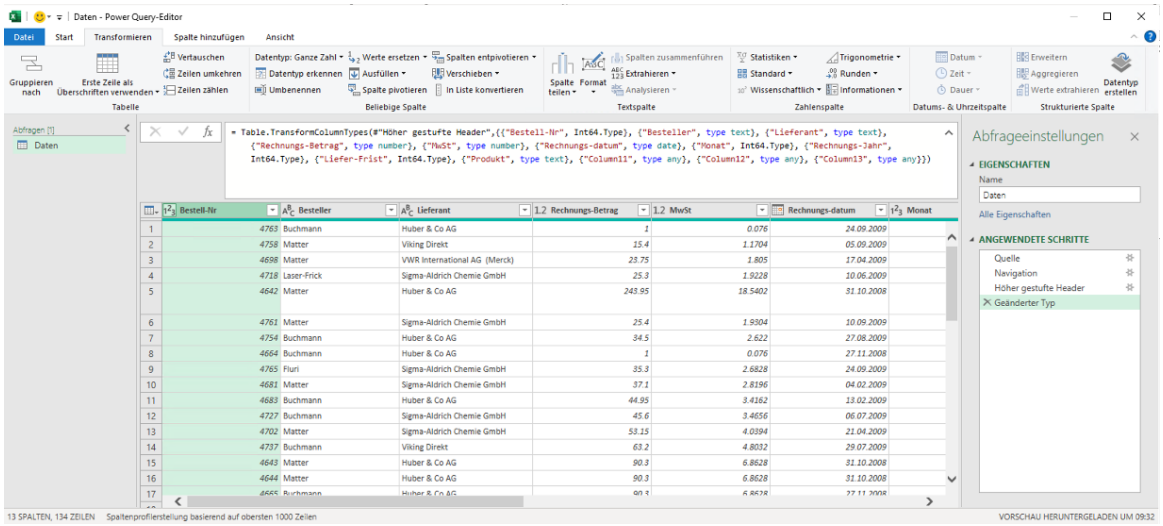


Abbildung 7.4.: Schemaanpassung mit Power Query

Nachdem das Schema kontrolliert wurde, können die Daten nach Excel importiert werden. Dazu wird das Kommando `Schließen & laden` in Power Query Menukategorie `Start` ausgeführt.

Nach einem Datenimport liegen die Daten in der Arbeitsmappe als *Tabelle* vor. Diese Daten sind eine Kopie der Daten in der Datenquelle. Dadurch kann die Arbeitsmappe unabhängig von der Datenquelle verwendet und geteilt werden.

> **Achtung**
>
> Excel betrachtet jeden Datenimport als Sicherheitsproblem. Daher werden Datenverbindungen beim Öffnen einer Arbeitsmappe standardmässig deaktiviert. Beim Öffnen einer Arbeitsmappe mit einer Datenverbindung wird eine Warnung angezeigt. Um mit den importierten Daten arbeiten zu können, muss die Datenverbindung *aktiviert* werden.
> Wird die Datenverbindung nicht aktiviert, werden Funktionen die auf die Daten zugreifen nicht ausgeführt, sondern nur die Ergebnisse der letzten Ausführung angezeigt.

7.2. Datenschema anpassen

Die einfachste Möglichkeit zur Anpassung des Datenschemas ist die Verwendung der Menukategorie `Transformation`. Dieses Menu zeigt alle verfügbaren Transformationen an. Die Transformationen sind in sechs Kategorien gegliedert:

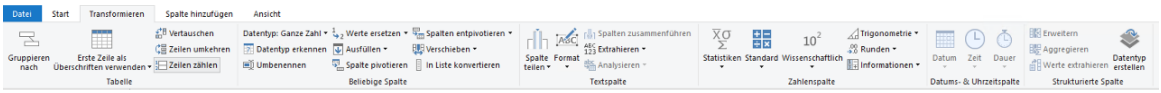


Abbildung 7.5.: Power Query Menu Transformation

1. `Tabelle`
2. `Beliebige Spalte`
3. `Textspalte`
4. `Zahlenspalte`
5. `Datums- & Uhrzeitspalte`
6. `Strukturierte Spalte`

Für das Datenschema sind die folgenden Transformationen wichtig:

- `Datentyp`
- `Umbenennen`
- `Verschieben`
- `Spalte teilen`

> **Hinweis**
>
> Die meisten Transformationen gehören konzeptionell zu späteren Kapiteln.
>
> - `Gruppieren nach` und `Werte ersetzen` werden im Kapitel 13 behandelt.
> - `Vertauschen` wird im Kapitel 12 behandelt.
> - `Zellen zählen`, `Ausfüllen` und alle Transformationen in den Kategorien `Zahlenspalten`, `Datums- & Uhrzeitspalten` und `Strukturierte Spalten` werden im Kapitel 11 behandelt.
> - `Spalte pivotisieren` und `Spalte entpivotisieren` werden im Kapitel 14 behandelt.
> - `Format`, `Analysieren` und `Extrahieren` (Kategorie `Textspalte`) werden im Kapitel 9 behandelt.
>
> In diesen Kapiteln werden die Transformationen von Power Query separat hervorgehoben.

7.2.1. Datentyp anpassen

Die wichtigste Transformation ist die Anpassung des Datentyps. Der Datentyp bestimmt, wie die Daten interpretiert werden. Die Interpretation der Daten bestimmt, welche Operationen auf die Daten angewendet werden können.

> **!** Wichtig
>
> Die Datentypen müssen im Schritt `Geänderter Typ` (Windows) bzw. `geänderter Spaltentyp` (MacOS) angepasst werden.

Power Query zeigt neben den Spaltenüberschriften mit einem Symbol den Datentyp der Spalte an. Tabelle 7.1 zeigt die Symbole für die verschiedenen Datentypen. Zusätzlich wird der Datentyp im Menubalken neben dem Kommando `Datentyp` angezeigt. Die Datentypen sind in vier Kategorien unterteilt:

- Zahlen
- Datum und Uhrzeit
- Zeichenketten
- Wahrheitswerte

Im Gegensatz zu Excel behandelt Power Query Datum und Uhrzeit als eigenen Datentyp. Power Query muss diese Unterscheidung machen, weil in den Datenquellen diese Werte als Zeichenketten kodiert vorliegen. Dabei handelt es sich um *strukturierte Daten*. Power Query hat für diese Daten spezielle *Parser*, die die Zeichenketten aus der Datenquelle in Zahlenwerte für Excel umwandeln.

Tabelle 7.1.: Power Query Datentypen mit Symbolen

Datentyp	Symbol	Kategorie	Bemerkungen
Text		Zeichenketten	
TRUE/FALSE		Wahrheitswerte	
Dezimalzahl		Zahlen	
Ganze Zahl		Zahlen	
Prozentzahl		Zahlen	Dezimalzahl mal `100`
Währung		Zahlen	Dezimalzahl mit Währungsformatierung
Binärzahl		Zahlen	Ganzzahl
(Zahl) mit Gebietsschema		Zahlen	Das Gebietsschema legt den Tausender- und Dezimaltrenner fest
Datum/Uhrzeit		Datum und Uhrzeit	Zeitstempel als formatierte Dezimalzahl

Datentyp	Symbol	Kategorie	Bemerkungen
Datum		Datum und Uhrzeit	Ganzzahl mit Datumsformatierung
Uhrzeit		Datum und Uhrzeit	Dezimalzahl zwischen 0 und 1 mit Zeitformatierung
Dauer		Datum und Uhrzeit	Dezimalzahl als Zeitdauer
Datum/Uhrzeit/Zeitz		Datum und Uhrzeit	Internationalisierter Zeitstempel als formatierte Dezimalzahl, die Zeitzone **muss** in den Werten kodiert sein, sonst wird +1 angenommen.

Der Datentyp einer Spalte lässt sich mit dem Kommando `Datentyp` anpassen. Wird auf das Kommando im Menubalken geklickt, öffnet sich eine Auswahlliste, aus der der richtige Datentyp ausgewählt werden kann (Abbildung 7.6). Wird der Datentyp geändert, dann erfolgt eine Abfrage (s. Abbildung 7.7), ob der Datentyp im aktuellen Arbeitsschritt ersetzt werden soll (`Aktuelle ersetzen`) oder ob ein neuer Arbeitsschritt eingefügt werden soll (`Neuen Schritt hinzufügen`). Hier kann normalerweise `Aktuelle ersetzen` ausgewählt werden.

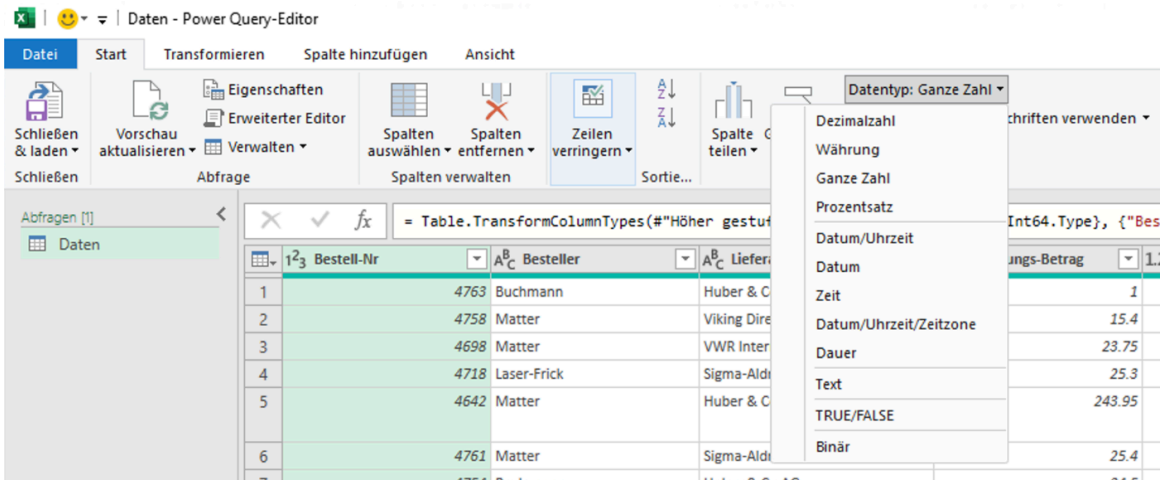


Abbildung 7.6.: Änderung des Datentyps einer Spalte

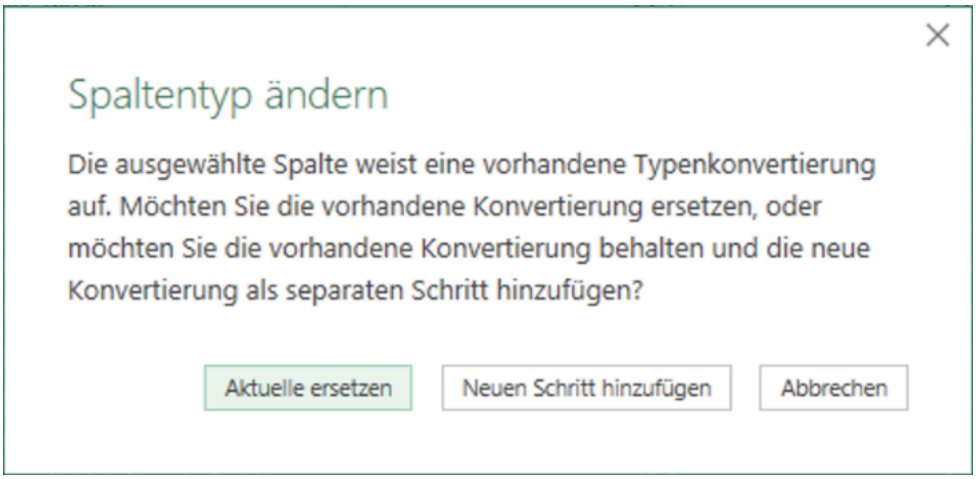


Abbildung 7.7.: Abfrage vor dem Ändern eines Datentyps

MacOS vs. Windows

Gelegentlich müssen Zahlen an das richtige Gebietsschema angepasst werden, damit die Werte korrekt eingelesen werden. Das ist immer dann notwendig, wenn das Dezimaltrennzeichen in der zu importierenden Datei vom aktuellen Gebietsschema abweicht. Unter MacOS können gebietsspezifische Datenformate sowohl über den Menübalken Transformieren als auch über die Spaltenüberschrift erreicht und angepasst werden. Unter Windows ist die Option `mit Gebietsschema` nur hinter dem Datentyp-Icon über die Spaltenüberschrift erreichbar. Dazu wird mit der rechten Maustaste auf die Spaltenüberschrift geklickt und anschliessend das Untermenü `Typ ändern` ausgewählt. Dort findet sich ganz unten der Punkt `Mit Gebietsschema...`. Alternativ lässt sich dieses Menü auch erreichen, indem mit der linken Maustaste auf das Datentyp-Icon links neben der Spaltenüberschrift geklickt wird. Beide Optionen funktionieren auch unter MacOS.

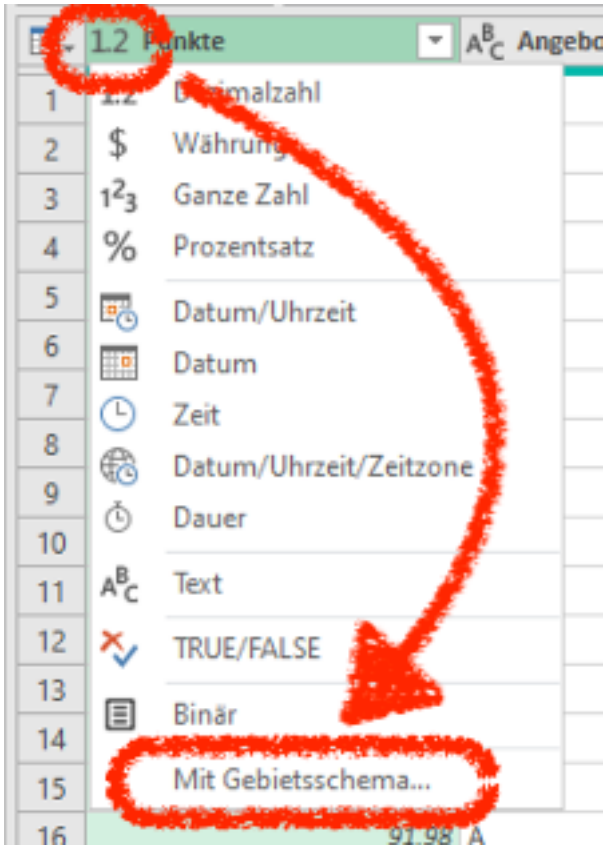


Abbildung 7.8.: Gebietsschema Datentyp in PowerQuery über das Datentypmenü der Spaltenüberschrift

Merke

Beim Import mit Gebietsschema gilt es folgendes zu beachten:

- Wenn Zahlen mit Komma als Dezimaltrennung importiert werden sollen, dann muss eine Region ausgewählt werden, in der das Komma verwendet wird (z.B. Deutschland oder Frankreich).
- Wenn Zahlen mit Punkt als Dezimaltrennung importiert werden sollen, dann muss eine Region ausgewählt werden, in der ein Punkt verwendet wird (z.B. Schweiz oder Vereinigte Staaten).

7.2.2. Spalten umbenennen

Falls einzelne Spaltenüberschriften einer Datenquelle unhandlich sind, lassen sich diese Spalten umbenennen. Dazu wird das Kommando `Umbenennen` verwendet. Das Kommando ist im Menuband `Transformieren` im Abschnitt `Beliebige Spalte` zu finden. Das Komman-

do öffnet einen Dialog, in dem der neue Name der Spalte eingegeben werden kann. Die Überschrift wird in einem eigenen Arbeitsschritt geändert.

Spalten lassen sich auch durch einen Doppelklick auf den Spaltennamen umbenennen.

Die Spaltenüberschriften müssen für die Datenstruktur *eindeutig* sein. Wird eine Spaltenüberschrift doppelt verwendet, dann gibt Power Query eine Fehlermeldung aus.

> **Praxis**
>
> Das Kommando `Umbenennen` sollte nur verwendet werden, wenn nur einzelne Spalten umbenannt werden müssen. Falls mehrere Spalten umbenannt werden müssen, sollte dieser Schritt als Transformation in Excel durchgeführt werden.

7.2.3. Spalten verschieben

In Excel können nur zusammenhängende Bereiche einer Tabelle mit einer Adressierung gemeinsam angesprochen werden. Diese Reihenfolge ist immer dann hilfreich, wenn Matrix-Operationen mit diesen Daten ausgeführt werden sollen. Falls die Spalten einer Datenquelle in einer ungünstigen Reihenfolge vorliegen, lassen diese sich mit einem Kommando unter `Verschieben` richtig positionieren.

7.2.4. Spalte teilen

Manche Daten fassen mehrere Werte in der einer Spalte zusammen. Falls die Werte als strukturierte Daten vorliegen, kann diese Spalte mit dem Kommando `Spalte teilen` in mehrere Spalten aufgeteilt werden. Dabei werden drei Varianten unterschieden.

1. Die Separator-Trennung verwendet Trennzeichen zum Abgrenzen der Werte.
2. Die Positions-Trennung trennt Festkodierungen mit festen Positionen in einzelne Spalten.
3. Die Längen-Trennung trennt Festkodierungen, deren Werte immer die gleiche Anzahl von Symbolen haben.

Die Separator-Trennung kann ein festes Trennzeichen oder ein Trennungsmuster verwenden. Die möglichen Trennungsmuster sind:

- Beim Wechsel zwischen kleinen und grossen Buchstaben.
- Beim Wechsel zwischen grossen und kleinen Buchstaben.
- Beim Wechsel zwischen Ziffern und Nicht-Ziffern.
- Beim Wechsel zwischen Nicht-Ziffern und Ziffern.

Beim trennen von Spalten dürfen nicht semistrukturierte Daten getrennt werden, weil diese nicht für alle Werte die gleiche Struktur haben.

7.3. Spalten auswählen oder entfernen

Zwei besondere Transformationen sind `Spalten auswählen` und `Spalten entfernen`. Diese Transformationen werden im Menuband `Start` angeboten. Mit diesen Transformationen lassen sich nicht benötigte Spalten aus den Daten entfernen. Dadurch werden die Daten in Excel übersichtlicher.

Diese Funktion kommt bei Excel-Arbeitsmappen häufig zum Einsatz, wenn nach Ende der Daten zusätzliche leere Spalten folgen. Diese Spalten können mit `Spalten entfernen` entfernt werden.

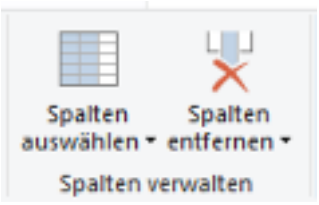


Abbildung 7.9.: Power Query Kommandos zum auswählen oder entfernen von Spalten (Menu `Start`)

Beide Kommandos verändern die Datenstruktur, indem Spalten entfernt werden. Das Kommando `Spalten auswählen` lässt die ausgewählten Spalten in der Datenstruktur und entfernt alle anderen. Das Kommando `Spalten entfernen` entfernt nur die ausgewählten Spalten.

> **Achtung**
>
> Eine spezielle Variante von `Spalten auswählen` ist das Kommando `In Liste konvertieren` im Menuband `Transformieren`. Dieses Kommando entfernt alle bis auf die ausgewählte Spalte. Dieses Kommando ändert jedoch die Spaltenüberschrift und sollte deshalb nicht verwendet werden.

7.4. Daten aktualisieren

Die Daten können sich aber ändern, z.B. weil ein Formular ausgefüllt wurde oder eine Datenbank aktualisiert wurde. Dadurch ändern sich die Daten in der Datenquelle. In solchen Fällen ist die Kopie in der Arbeitsmappe nicht mehr aktuell. Um die Daten zu aktualisieren, muss die Datenverbindung aktualisiert werden.

Dazu wird das Kommando **Alle aktualisieren** verwendet. Das Kommando ist im Menu **Daten** im Abschnitt **Verbindungen** zu finden. Das Kommando aktualisiert alle Datenverbindungen in der Arbeitsmappe entsprechend der Importspezifikation.

Es werden nur die importierten Daten aktualisiert. Wurde die Tabelle durch Formeln erweitert, werden diese Formeln *nicht* gelöscht, sondern auf die neuen Daten erweitert.

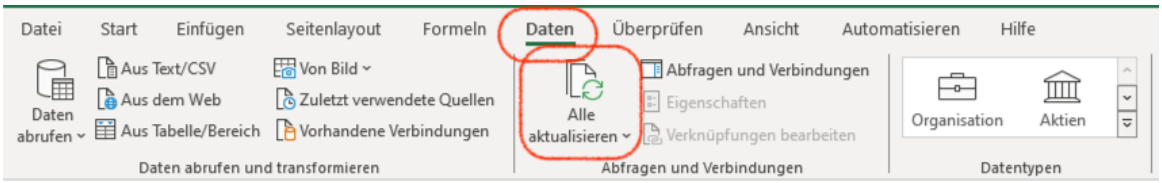


Abbildung 7.10.: Menubalken Daten - Kommando Alle aktualisieren

> **!** Wichtig
>
> Die importierte Struktur darf **nicht verändert** werden. Das bedeutet, dass innerhalb der Struktur keine Spalten hinzugefügt oder gelöscht werden dürfen. Neue Spalten für Formeln müssen rechts von der importierten Datenstruktur der Tabelle hinzugefügt werden.

7.5. Datenverbindung anpassen

Gelegentlich ändert sich der Ort einer Datenquelle. Sehr häufig tritt diese Situation ein, wenn eine Datei in einen anderen Ordner verschoben wird. In solchen Fällen muss die Datenverbindung angepasst werden. Dazu muss in Power Query die Datenquelle angepasst werden.

1. Powerquery öffnen
2. Einstellungen im Schritt "Quelle" auswählen (s. Abbildung 7.11)
3. Pfad zur Datenquelle anpassen

Nachdem die richtige Datei ausgewählt wurde, sollte immer das Schema kontrolliert werden, ob die Änderung keine Auswirkungen auf das Schema hat. Wird dieser Schritt übersprungen, kann es zu Fehlern beim Import kommen und im schlimmsten Fall werden die Formeln in der Arbeitsmappe durch eine Fehlerfortpflanzung zerstört.

Abschliessend kann wieder das Kommando **Schliessen & Landen** ausgeführt werden.

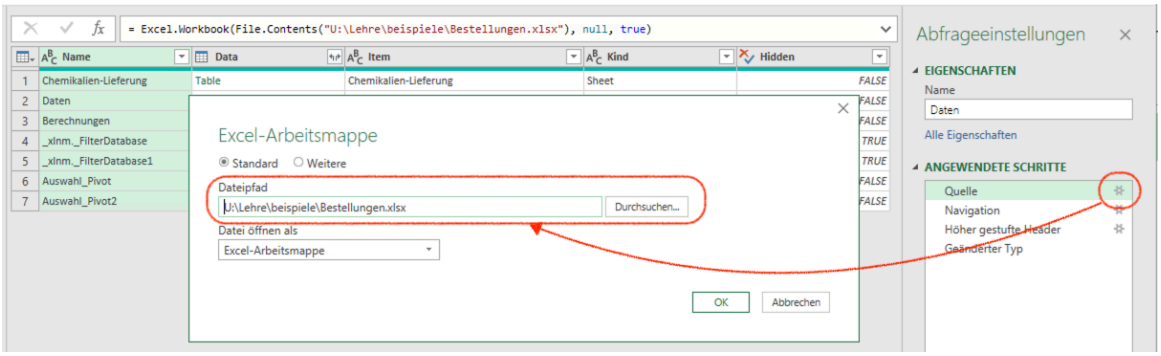


Abbildung 7.11.: Power Query: Quelle Anpassen (Ausschnitt)

Alle Datenverbindungen werden auch im Dialog **Abfragen & Verbindungen** (s. Abbildung 7.12) angezeigt. Der Dialog ist in der Gruppe **Verbindungen** im Abschnitt **Daten** zu finden. Der Dialog zeigt alle Datenverbindungen an, die in der Arbeitsmappe verwendet werden.

Abbildung 7.12.: Menubalken Daten - Abfragen & Verbindungen

> **MacOS vs. Windows**
>
> Unter Windows lassen sich die Datenquellen im Dialog **Abfragen & Verbindungen** anpassen. Die Windows Version von Excel von dort immer den Power Query Editor. Die Version für MacOS macht das nicht. Um Änderungen an einer Abfrage vorzunehmen, muss die Abfrage in Power Query über das Kommando `Daten abrufen` geöffnet werden (s. Abbildung 7.13).

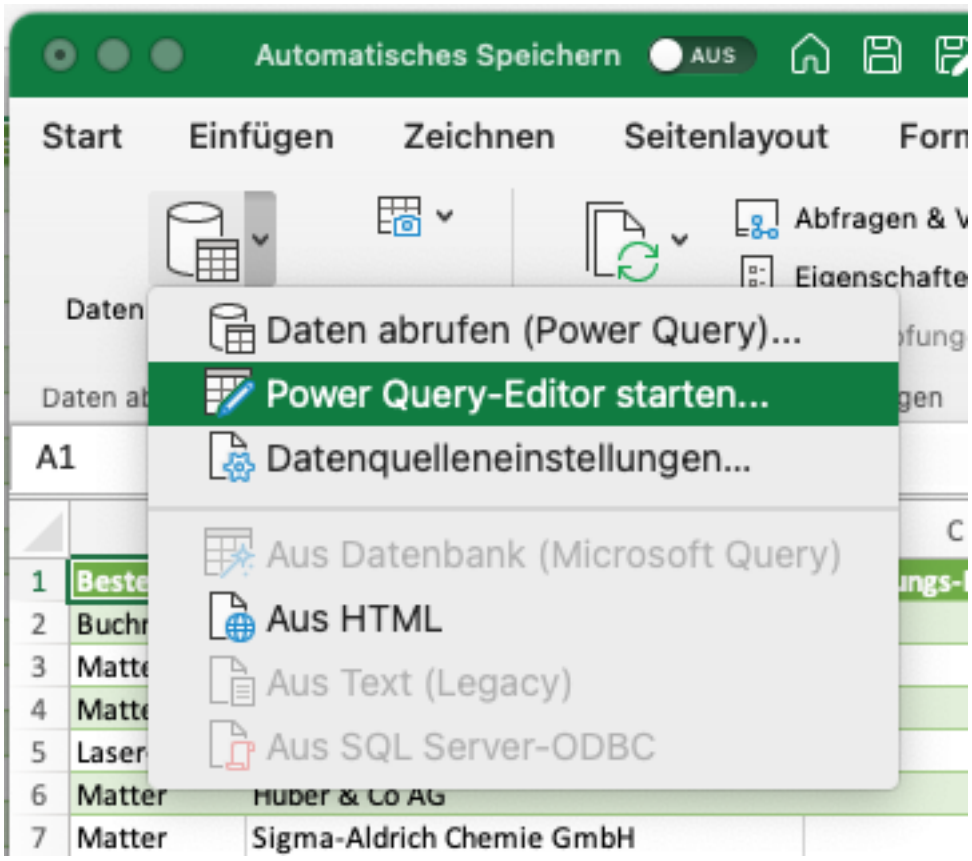


Abbildung 7.13.: Power Query unter MacOS starten

Teil III.

Mathematik der Daten

8. Variablen, Funktionen und Operatoren

8.1. Variablen

Excel kennt keine strikte Unterscheidung zwischen Konstanten und Variablen. Grundsätzlich sind alle Werte in Excel Konstanten, weil direkt eingegebene oder durch eine Formel erzeugte Werte nicht durch andere Formel verändert werden können.

> ⚠ Warnung
>
> In Excel lassen sich Zellen und Bereiche *sperren*. Damit lassen sich die Werte wie Konstanten behandeln. Diese Sperre ist Teil der *Formatierung* einer Zelle und muss für das gesamte Arbeitsblatt festgelegt werden. Die Sperre kann nicht für einzelne Zellen oder Bereiche festgelegt werden.
> Die Sperre eines Arbeitsblatts wird im Menuband `Überprüfen` mit dem Kommando `Blatt schützen` aktiviert. Richten Sie bei diesem Kommando **kein** Passwort ein.

Ist ein Arbeitsblatt gesperrt, sind die Werte auf diesem Arbeitsblatt unveränderlich und verhalten sich wie Konstanten. Ist ein Arbeitsblatt nicht gesperrt, dann verhalten sich die Werte wie Variablen.

Bezeichner unterliegen in Excel gewissen Beschränkungen.

1. Bezeichner dürfen nur aus Buchstaben, Zahlen, Dollarzeichen, Punkten und Unterstrichen (_) bestehen.
2. Bezeichner dürfen keine bestehenden Funktionsnamen sein.
3. Bezeichner dürfen keine gültigen Adressen im A1-Format sein.
4. Bezeichner dürfen keine gültigen Adressen im Z1S1-Format sein.

Beispiel 8.1 (Ungültiger Bezeichner)**.** Der Bezeichner `x1` ist ungültig, weil dieser identisch mit der Adresse `X1` wäre. Wird Bezeichner um einen Unterstrich zu `x_1` erweitert, dann gibt es keine Kollision mit der Adresse mehr und der Bezeichner ist gültig.

8.1.1. Dynamische Bereiche

Formeln können mehr als einen Wert verarbeiten und mehr als einen Ergebnis liefern. Solche Formeln müssen nur in die linke obere Ecke eines Bereichs eingetragen werden. Excel erkennt automatisch, dass die Formel auf einen Bereich angewendet werden soll und erzeugt die entsprechende Formel für alle Zellen des Bereichs. Das Ergebnis einer solchen Formel ist ein *dynamischer Bereich*.

Definition 8.1. Vektorisieren heisst das Erzeugen eines dynamischen Bereichs aus einem statischen Bereich.

Beispiel 8.2 (Vektorisieren eines statischen Bereichs)**.**

```
=A1:A10
```

Im Gegensatz zu einem normalen Bereich, ist bei einem dynamischen Bereich nur die linke obere Zelle bekannt. Um trotzdem alle Zellen eines solchen Bereichs zu adressieren, wird die Gatter- (`#`) Notation verwendet.

Das Beispiel 8.2 erzeugt einen Bereich mit 10 Zellen. Die Formel wird in die linke obere Zelle des Bereichs eingetragen. Die Formel wird z.B. in die Zelle `B1` eingetragen. Anschliessend können die Werte im Bereich `B1:B10` über die Gatter-Notation adressiert werden: `B1#`.

Der Vorteil des Vektorisierens ist, dass der Bereich durch das Einfügen neuer Zeilen vergrössert werden kann, ohne dass die nachfolgenden vektorisierten Formeln angepasst werden müssen.

> **Praxis**
>
> Weil Tabellen automatisch vektorisiert werden, ist es einfacher Werte in einer Tabelle zu erfassen bzw. als Tabelle zu importieren (s. Kapitel 7) und anschliessend über die Tabellenadressierung auf die Werte zu verweisen.

> **Merke**
>
> Tabellenadressierungen auf eine Spalte oder einen Spaltenberech sind immer Vektorisiert.

Beispiel 8.3 (Vektorisieren von Tabellenspalten)**.**

```
= Tabelle1[Spalte1]
```

Diese Formel vektorisiert die Spalte mit dem Namen `Spalte1` aus der Tabelle `Tabelle1`. Angenommen, dass diese Formel in Zelle `A1` des aktuellen Arbeitsblattes steht, dann kann anschliessend auf den Vektor über die Gatter-Notation zugegriffen werden:

```
=A1#
```

8.1.2. Benannte Bereiche

Im Funktionsbalken wird ganz links die Adresse der aktuellen Zelle angezeigt. Wenn in dieses Feld geklickt wird, dann kann der Adresse ein Name zugewiesen werden. Auf diese Weise kann eine Zelle oder ein Bereich benannt werden. Der Name eines Bereichs darf nur einmal in einer Arbeitsmappe existieren. Dafür kann in jeder Zelle der Arbeitsmappe dieser Namen als Adresse verwendet werden. Dazu ist es nicht notwendig, den die genaue Arbeitsblatt- und Zelladresse zu kennen.

Verweist ein benannter Bereich mit einer Zelle auf den Anfang eines dynamischen Bereichs, dann kann der gesamte dynamische Bereich mit der Gatter-Notation referenziert werden.

Auf diese Weise lassen sich oft adressierte Zellen oder Bereiche benennen. Dadurch kann die Adresse für Verwendung in Formeln abstrahiert werden.

Die Verwendung von benannten Bereichen wird weiter vereinfacht, dass bei der Eingabe von Formeln die Namen der benannten Bereiche als Vorschlag angezeigt werden.

Beispiel 8.4 (Adressierung eines dynamischen benannten Bereichs)**.**

```
=umsatz#
```

Benannte Zellen helfen, auf wichtige Zellen in einer Arbeitsmappe zuzugreifen.

> **i** Merke
>
> Ein benannter Bereich entspricht einer Variablen in anderen Programmiersprachen.

> **!** Wichtig
>
> Eine benannte Zelle oder ein benannter Bereich sind immer absolut referenziert.

> **i** Hinweis
>
> Wenn einem benannten Bereich Zellen hinzugefügt werden, dann wird der benannte Bereich um diese Zellen erweitert.

> **i** Hinweis
>
> Eine Tabelle ist ein spezieller benannter Bereich, der als Ganzes oder in Teilen referenziert werden kann (s. Abschnitt 6.4.2).

8.2. Funktionen

Excel kann nur durch Funktionen und Operatoren *programmiert* werden.

Excel hat wenige vordefinierte Operatoren, wobei die Operatoren keine direkte Entsprechung als Funktion haben. Die einzige Ausnahmen sind der Potenzoperator (`^`) und der Textverkettungsoperator (`&`). Der Potenzoperator ist funktional gleich mit der `POTENZ()`-Funktion. Der Textverkettungsoperator entspricht der Funktion `TEXTKETTE()` mit zwei Argumenten.

> **Merke**
>
> Die mit Excel möglichen Programme sind durch die Operatoren und die vordefinierten Funktionen beschränkt.

Excel **Kommandos** können nur durch Interaktion mit den Excel Menus oder Dialogen ausgeführt werden. Diese Funktionalität steht für die Progammierung oft nicht zur Verfügung.

> **Makros**
>
> Neben den Funktionen und Kommandos existieren in Excel noch *Makros*. Mit Makros können neue Kommandos und Funktionen programmiert werden. Makros folgen aber nicht den Regeln von Formeln, weil sie in einer anderen Programmiersprache geschrieben werden.
> Makros unterliegen nicht den Einschränkungen von Excel-Funktionen. Diese Freiheit ist gleichzeitig ein Fluch, denn Makros sind ein Sicherheitsrisiko und Excel präsentiert entsprechende Warnungen, wenn Makros in einer Arbeitsmappe gefunden wurde.
> Aktuelle Bestrebungen von Microsoft zielen darauf ab, Makros langfristig durch Funktionen zu ersetzen. Ein Teil dieser Bestrebungen ist die Einführung von `LAMBDA()`-Funktionen (s. Abschnitt 8.5).

> **Merke**
>
> Excel hat keine Identitätsfunktion. Die Identitätsfunktion wird durch eine Formel simuliert, die nur eine Adresse enthält. Solche Formeln werden für das *Vektorisieren* (Definition 8.1) von Bereichen eingesetzt.

8.2.1. Generatoren in Excel

Excel hat zwei Generatorfunktionen:

- Die `SEQUENZ()`-Funktion
- Die `ZUFALLSMATRIX()`-Funktion

Beide Generatoren erfordern die Anzahl der Zeilen und Spalten, für die Werte generiert werden sollen. Soll nur ein Spaltenvektor erzeugt werden, muss nur die Anzahl der Zeilen angegeben werden.

Die `SEQUENZ()`-Funktion erzeugt eine Sequenz von Werten ausgehend vom ersten Wert (`Anfang`). Die Schrittweite legt die Abstände zwischen den einzelnen Werten fest. Standardmässig ist die Schrittweite mit `1` festgelegt.

> **Merke**
>
> Wird die Schrittweite mit `0` festgelegt, dann wird der Anfangswert für die angegebene Anzahl von Spalten und Zeilen wiederholt.

Die Funktion `ZUFALLSMATRIX()` erzeugt Zufallswerte in einem vorgegebenen Intervall. Diese Funktion kann reelle Zahlen oder ganze Zahlen generieren. Der Wertebereich der generierten Zahlen kann durch einen Minimal- und einem Maximalwert eingeschränkt werden. Wird kein Minimal- und Maximalwert angegeben, dann werden Werte im Intervall $0 < z < 1$ generiert. Werden Ganzzahlen für dieses Intervall angefordert, dann werden die Werte `0` und `1` erzeugt.

8.3. Substitution

Excel kennt zwei Formen der Substitution.

1. Die Substitution über Funktionspfade
2. Die Substitution mit der Funktion `LET()`

Beide Substitutionsformen sind fast gleichwertig. Funktionspfade eigenen sich besonders gut für die Entwicklung von und zur Fehlersuche in komplexen Formeln. `LET()` erlaubt es, mehrere Arbeitsschritte effizient in einer Formel zusammenzufassen.

8.3.1. Substitution über Funktionspfade

Eine komplexe Operation lassen sich in Excel durch Substitution über Funktionspfade vereinfachen.

Definition 8.2. Ein **Funktionspfad** sind Formeln, die sich über ihre Adressen aufeinander beziehen.

Bei einem Funktionspfad werden die substituierten Funktionen als Formeln in separate Zellen geschrieben. Die Adresse der jeweiligen Formel wird als Substitution für die Funktion eingesetzt. Sind die Formeln in einem benannten Bereich, dann kann der Name des Bereichs zur Substitution verwendet werden.

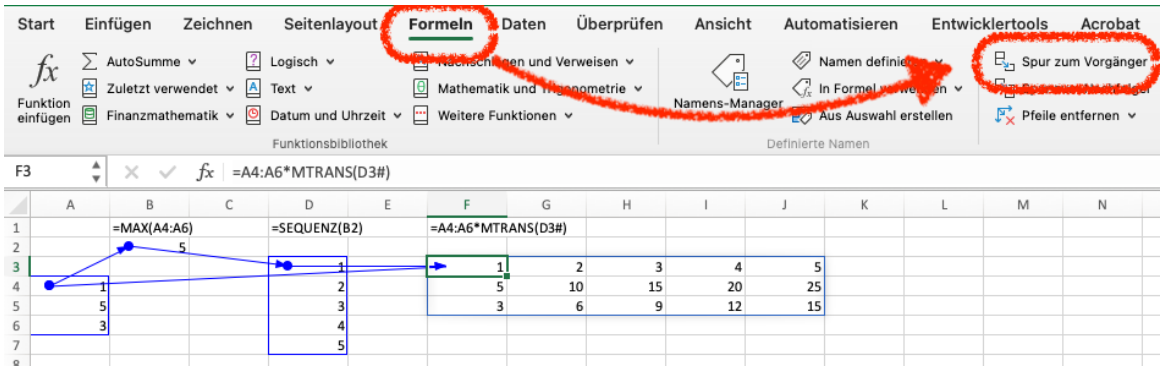


Abbildung 8.1.: Funktionspfad

Funktionspfade können mithilfe des Kommandos `Spur zum Vorgänger` im Menü `Formeln` sichtbar gemacht werden (Abbildung 8.1).

> **Praxis**
>
> Um komplexe Formeln in bestehenden Arbeitsmappen zu verstehen, hilft das Zerlegen dieser Formeln in Funktionspfade. Dabei kann eine tabellarische Organisation helfen, wiederkehrende Operationen leichter zu erkennen.

8.3.2. Substitution mit `LET()`

Excels `LET()`-Funktion erlaubt das Vereinfachen komplizierter Formeln durch *Variablen*. Diese *Variablen* existieren nur im Kontext der `LET()`-Funktion und können nicht ausserhalb dieser Funktion verwendet werden.

Eine Variable in der `LET()`-Funktion entspricht einer Substitution eines Teilausdrucks einer Formel.

Beispiel 8.5 (`LET()`-Funktion zur Substitution)**.**

```
=LET(
    Daten; 'Unbearbeitete Daten'!A:F;
    DatenFeld; INDEX(
        Daten;
        sequenz(ZEILEN(Daten));
        sequenz(1; SPALTEN(Daten))
      );
    WENN(ISTLEER(DatenFeld);#NV;DatenFeld)
 )
```

In Beispiel 8.5 wird der referenzierte Bereich und der Aufruf der `INDEX()` substituiert. Die Substitution wird durch die Variablen `Daten` und `DatenFeld` realisiert.

Beispiel 8.6 (Formel ohne Substitution)**.**

```
=WENN(ISTLEER(INDEX(
        'Unbearbeitete Daten'!A:F;
        sequenz(ZEILEN('Unbearbeitete Daten'!A:F));
        sequenz(1; SPALTEN('Unbearbeitete Daten'!A:F))
      ));
      #NV;
      INDEX(
        'Unbearbeitete Daten'!A:F;
        sequenz(ZEILEN('Unbearbeitete Daten'!A:F));
        sequenz(1; SPALTEN('Unbearbeitete Daten'!A:F))
      )
 )
```

Die beiden Beispiele veranschaulichen, wie mit der `LET()`-Funktion mehr als eine Substitution umgesetzt wird, um eine komplexe Formel in überschaubare Teilschritte zu zerlegen und so stark zu vereinfachen.

> Praxis
>
> Wird eine Funktion in einer Formel mit den gleichen Parametern mehrfach aufgerufen, dann sollte diese Funktion **immer** mit `LET()` substituiert werden. Substituierte Berechnungen werden nur einmal für die Substitution durchgeführt und anschliessend wird nur das Ergebnis verwendet. Nicht substituierte Funktionen werden bei jedem Vorkommen neu ausgeführt.

8.4. Funktionsketten

Funktionsketten werden in Excel über Substitutionen erzeugt. Gerade bei Matrizen und anderer Transformationen von Datenstrukturen ist die Verwendung von Funktionspfaden aus zwei Gründen unhandlich:

1. Wenn die Ergebnisse mehrerer Arbeistsschritte auf einem Arbeitsblatt dargestellt werden, wird die Übersichtlichkeit behindert.
2. Weil Funktionspfade separate Formeln verwenden werden die Ergebnisse für jede Formel serialisiert, was nicht immer zu den erwarteten Ergebnissen führt.

Weil Excel keine Operatoren für die Funktionsverkettung bereitstellt, werden Funktionsketten **immer** mithilfe der Funktion `LET()` erzeugt.

8.4.1. LET() und leere Zellen

Normalerweise werden leere Zellen als Ergebnis einer Funktion durch `0` ersetzt. Dieses Konvertierung findet erst bei der Darstellung des Ergebnisses statt. Innerhalb einer Funktionskette werden leere Zellen als leere Zellen weitergereicht, solange keine Aggregation vorgenommen wird. Es ist deshalb möglich in einer Funktionskette eine Entscheidung mit `ISTLEER()` für den Fall einer leeren Zelle zu treffen.

Eine Excel-Operation **muss** einen Wert als Ergebnis einer **Formel** haben. Wird ein nicht vorhandener Wert (d.h. leere Zelle) in einem Ergebnis einer Formel gefunden, dann wird dieser Wert automatisch in den Wert `0` konvertiert. Diese Umwandlung passiert jedoch erst *nachdem* die Operation abgeschlossen ist und Excel das Ergebnis auf dem Arbeitsblatt darstellt.

Dieses Verhalten hat zur Folge, dass solange eine Operation nicht abgeschlossen ist, die nicht vorhandenen Werte in ihrer ursprünglichen Form erhalten bleiben. Es ist also möglich undefinierte Werte mit `ISTLEER()` zu prüfen.

Die ursprünglichen Daten können unvollständig sein und enthalten dann leere Zellen an den entsprechenden Zellen. Diese fehlenden Werte als `0` darzustellen, kann zu verzerrten Ergebnissen führen. Deshalb sollten solche Werte mit dem *Fehler* `#NV` (lies: *Nicht Vorhanden*) markiert werden. Dieser Fehlerwert wird nicht automatisch in den Wert `0` umgewandelt, so dass die fehlenden Werte korrekt berücksichtigt werden können.

Diese Umwandlung nutzt aus, dass Excel leere Zellen als Ergebnis von *Funktionen* zulässt, aber nicht als Ergebnis von *Formeln*. Entsprechend kann das folgende Funktionsmuster verwendet werden.

Die beiden Vektoren `G1#` und `H1#` sind Hilfsvektoren, die Sequenzen für die Zeilen- und Spaltenindizes der Datenstruktur `A:F` enthalten.

Der *logische Ausdruck* prüft, ob ein Feld mit dem Index `G1#` und `H1#` im Stichprobenobjekt leer ist. Falls das Feld in den unbearbeiteten Daten leer ist, dann wird der Wert `#NV` als Ergebnis zurückgegeben. Sonst soll der Wert im Feld übergeben.

In dieser Operation wird die Funktion `INDEX()` zwei Mal mit den gleichen Parametern aufgerufen. Das ist unpraktisch, weil die Operation an zwei Stellen angepasst müsste, wenn die Daten mehr oder weniger Spalten haben. Besser wäre es, wenn das Zwischenergebnis der `INDEX()`-Funktion aus der Operation herausgelöst wird und über eine Funktionsverkettung eingebunden wird. Das ist aber nicht möglich, weil Excel bei diesem Zwischenschritt die fehlenden Werte in `0` ändert, sodass anschliessend der logische Ausdruck immer `FALSCH` liefern würde.

Mittels der `LET()` Funktion wird das Ergebnis dieses Zwischenschritts in einer temporären *Variablen* gespeichert. Gegenüber der normalen Funktionsverkettung durch Funktionspfade hat diese Strategie den Vorteil, dass für Excel die Operation nicht abgeschlossen ist und deshalb die fehlenden Werte *noch nicht* in den Wert `0` umgewandelt werden. Der logische Ausdruck in der `WENN()`-Funktion kann also `WAHR` ergeben, wenn in den Daten ein Wert fehlt.

Ausserdem muss die Indizierung für eine Position nur einmal durchgeführt werden, was bei komplexen Formeln die Übersichtlichkeit erhöht und die Ausführung beschleunigt.

> **Warnung**
>
> Die Namen in der `LET()` unterliegen den gleichen Beschränkungen, wie alle anderen Bezeichner (Abschnitt Kapitel 8.1).

Die ursprüngliche Formel lässt sich also dahingehend vereinfachen, dass der Aufruf der `INDEX()`-Funktion "ausgeklammert" und in der Hilfsvariablen `Feld` gespeichert wird.

Daraus ergibt sich die Lösung als Funktionskette.

```
=LET(Feld; INDEX('Unbearbeitete Daten'!A:F;A2#;B1#);
     WENN(
        ISTLEER(Feld);
        #NV;
        Feld
     )
 )
```

Diese Lösung entspricht ungefähr der Funktionskette Formel 8.1.

$$Index() \triangleright Wenn() \tag{8.1}$$

Damit wird der Aufruf der `WENN()`-Funktion vereinfacht, weil nur noch die Hilfsvariable `Feld` übergeben müssen. Diese Variable enthält die Daten für die Funktionsverkettung, so dass eine zusätzliche Arbeitsblattadresse nicht notwendig ist.

8.5. Funktionen selbst definieren

In Excel können eigene Funktionen mit der Funktion `LAMBDA()` erzeugt werden. Die `LAMBDA()`-Funktion erzeugt die Funktion aus ihren Parametern. Dabei ist der letzte Parameter immer der Funktionskörper. Die restlichen Parameter sind die Parameter der Funktion. Es können max 253 Parameter angegeben werden, wobei keine optionalen oder vorbelegten Parameter möglich sind. Beispiel 8.7 zeigt eine einfache Funktionsdefinition mit zwei Parametern.

Beispiel 8.7 (Funktionsdefinition mit zwei Parametern ohne Ausführung)**.**

```
= LAMBDA(a; b; a + b)
```

> ⚠ Warnung
>
> Mit Excels `LAMBDA()`-Funktion können nur Funktionen durch Verkettung anderer Funktionen erzeugt werden.

Für Excel sind Funktionen kein darstellbarer Datentyp. Wird also eine Funktion mit `LAMBDA()` definiert, ohne dass sie unmittelbar ausgeführt wird, zeigt Excel den Fehler `#KALK!` an. Dieser Fehler kann dadurch vermieden werden, indem die Funktion unmittelbar ausgeführt wird (s. Beispiel 8.8).

Beispiel 8.8 (Funktionsdefinition mit zwei Parametern mit sofortiger Ausführung)**.**

```
= LAMBDA(a; b; a + b)(1;2)
```

> ⚠ Warnung
>
> Der Funktionskörper muss nicht alle Parameter verwenden. Wird eine Funktion aufgerufen, dann **müssen** alle Parameter angegeben werden, selbst wenn diese im Funktionskörper nicht verwendet werden.

8.5.1. Map-Reduce und `LAMBDA()`

Weil eine mit `LAMBDA()` erzeugte Funktion als Formel direkt ausgeführt werden muss, ergibt sich kein Vorteil gegenüber normalen Excel-Formeln. Die Funktion wird jedoch zum Erstellen von Callbacks benötigt. Die Hauptanwendung sind Operationen, die mit jedem Wert eines Bereichs durchgeführt werden sollen. Weil Excel normalerweise keine Schleifen zulässt, müssen solche Operationen über die Logikfunktionen `MAP()`, `NACHZEILE()`, `NACHSPALTE()`, `REDUCE()` und `SCAN()` erzeugt werden.

Alle Map-Funktionen erwarten als ersten Parameter einen Bereich mit Werten und als letzten Parameter eine Funktionsdefinition mit `LAMBDA()`.

- `MAP()` erzeugt eine Schleife, welche den Callback für jedes Element genau einmal aufruft. Es ist möglich, mehrere Bereiche anzugeben. Die Parameteranzahl der Callback muss der Anzahl der angegebenen Bereiche entsprechen.
- `NACHSPALTE()` erzeugt eine Schleife, die eine Funktion für jede Spalte eines Bereichs aufruft. Der Callback darf nur einen Parameter haben. Der Parameter enthält eine Liste mit allen Werten der aktuellen Zeile.
- `NACHZEILE()` erzeugt eine Schleife, die eine Funktion für jede Zeile eines Bereichs aufruft. Der Callback darf nur einen Parameter haben.

`MAP()`, `NACHSPALTE()` und `NACHZEILE()` behandeln die einzelnen Durchläufe der Schleife als *unabhängig*. Im Gegensatz dazu sind die Durchläufe einer `REDUCE()` oder `SCAN()`-Schleife immer vom voherigen Durchlauf *abhängig*. `REDUCE()` und `SCAN()` erfordern als ersten Parameter einen Initialwert, der als Argument für den ersten Aufruf des Callbacks verwendet wird.

- `REDUCE()` erzeugt eine Schleife, die für jedes Element eines Bereichs den Callback ausführt. Diese Funktion erhält als ersten Parameter, das Callback-Ergebnis des vorherigen Durchlaufs und als zweiten Parameter. Das Ergebnis von `REDUCE()` ist das Ergebnis des letzten Aufrufs der Schleifenfunktion.
- `SCAN()` ist eine Variante von `REDUCE()`. Während `REDUCE()` nur das letzte Ergebnis der Callbacks ausgibt, erzeugt `SCAN()` einen Vektor mit allen Ergebnissen der Callback-Aufrufe.

> **Warnung**
>
> Die Map-Reduce-Funktionen dürfen nur einzelne Werte erzeugen. Datenstrukturen sind auch dann nicht erlaubt, wenn sie alle die gleiche Länge haben und sich in einen rechteckigen Bereich zusammenfügen liessen. Diese Beschränkung gilt auch, wenn die Funktionsergebnisse als Zwischenschritt an eine Kombinationsfunktion für Vektoren (z.B. `VSTAPELN()`) verknüpft werden. Erzeugt ein Callback eine Datenstruktur, dann erzeugt die jeweilige Funktion den Fehlerwert `#KALK!`.
> Einzig die Funktion `REDUCE()` darf Datenstrukturen erzeugen; weil diese Funktion nur ein Ergebnis haben kann.
> Diese Einschränkung bedeutet, dass die Funktionen `NACHSPALTE()` und `NACHZEILE()` die Identitätsfunktion **nicht** als Callback akzeptieren.

8.5.2. Index-Schleifen mit `MATRIXERSTELLEN()`

Während die Funktionen der Map-Reduce-Familie Werte voraussetzen, ist dies nicht immer möglich. Die Funktion `MATRIXERSTELLEN()` erzeugt eine Schleife über zwei Iteratoren, für die Dimensionen der gewünschten Matrix. Der Callback für die Funktion erfordert deshalb 2 Parameter für die beiden Indizes.

> **Praxis**
>
> Die Funktion `MATRIXERSTELLEN()` kann meistens durch das ***äussere Produkt*** (Kapitel 12) ersetzt werden.

Der Callback für `MATRIXERSTELLEN()` darf nur zwei Parameter haben, denen die aktuellen Index-Werte zugewiesen werden. Daraus ergibt sich, dass im Funktionskörper nur diese beiden Parameter bereitgestellt werden. Komplexere Anwendungen lassen sich mit *Closures* erzeugen. Dazu wird eine Funktion mit LAMBDA() erzeugt, die zusätzliche Werte oder Datenstrukturen als Parameter unterstützt. Diese Funktion erzeugt anschliessend den Callback für `MATRIXERZEUGEN()`. Der Callback ist also ein Closure der erzeugenden Funktion.

Dadurch kann der Callback die Werte und Datenstrukturen der erzeugenden Funktion beim Aufruf durch die Funktion `MATRIXERZEUGEN()` ebenfalls verwenden (s. Beispiel 8.9).

Beispiel 8.9 (Matrix mit Closure erstellen)**.** Im Bereich `A1:A4` stehen beliebige Zeichenketten.

```
=MATRIXERSTELLEN(5;3;
    LAMBDA(namen;
           LAMBDA(a;b;
                  ZEILENWAHL(namen;
                             REST(a+b;
                                  ZEILEN(namen))+1)
           )
    )(A1:A4)
 )
```

8.5.3. Neue Funktionen festlegen

Neben Schleifen können mit `LAMBDA()` neue Funktionen erzeugt werden. Dazu muss in einem Arbeitsblatt ein Name mit der Funktionsdefinition erzeugt werden. Namen werden über das Menüband `Formeln` mit dem Kommando `Namen definieren` erzeugt. Dieses Kommando öffnet einen Dialog, über welchen ein Name definiert werden kann(Abbildung 8.2). In diesem Dialog müssen die beiden Felder `Name` und `Bezieht sich auf` ausgefüllt werden.

Der `Name` ist der Bezeichner der neuen Funktion. Hier dürfen keine Namen vorhandener Funktionen oder benannter Bereiche verwendet werden. Das Feld `Bezieht sich auf` muss eine LAMBDA-Formel mit der Funktionsdefinition beinhalten. Als `Bereich` sollte immer `Arbeitsmappe` ausgewählt sein, weil sonst die Funktion auf ein einziges Arbeitsblatt beschränkt wäre. Zur Dokumentation sollte im Feld `Kommentar` zusätzlich eine Kurzbeschreibung der Funktion angegeben werden. Leider zeigt Excel diesen Kommentar nicht als Kurzhilfe für den Funktionsnamen an.

Nachdem eine Funktion einem Namen zugewiesen wurde, kann dieser Name wie jede andere Funktion in Formeln verwendet werden (Abbildung 8.3).

Als Funktionskörper können beliebige Excel Operationen vorkommen. Es bietet sich jedoch an, für komplexere Funktionen den Funktionskörper mit `LET()` einzuleiten. Dadurch lassen sich einzelne Arbeitsschritte leichter isolieren und verketten.

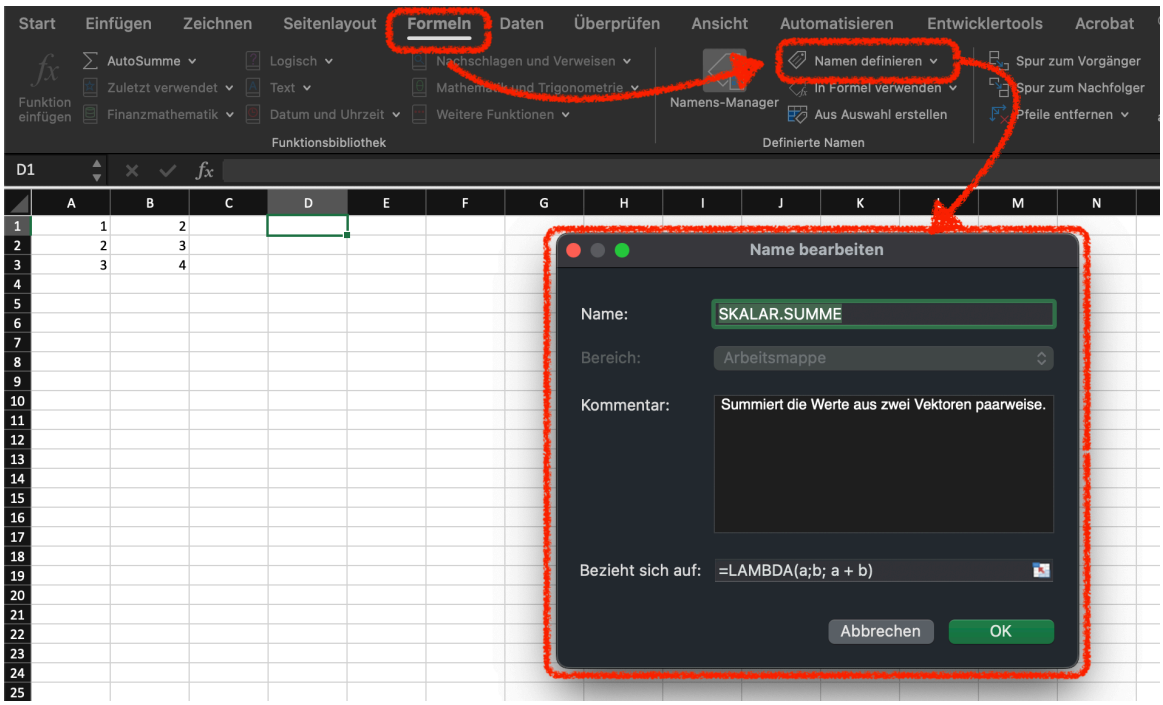


Abbildung 8.2.: Funktionsdefinition mit `LAMBDA()`

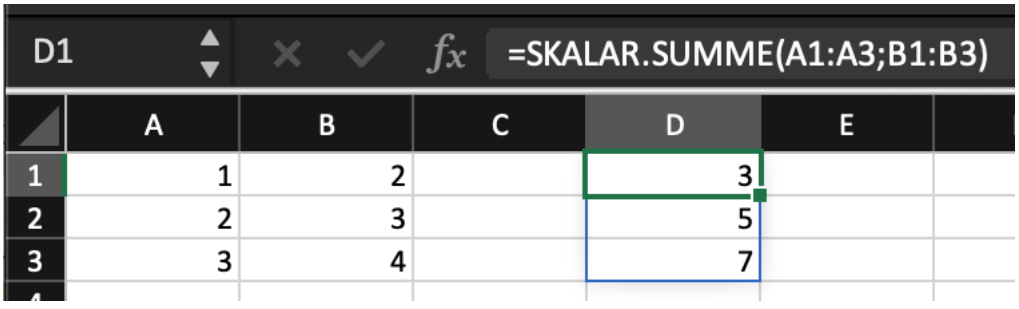


Abbildung 8.3.: Eigene Funktion anwenden

9. Zeichenketten

Bisher wurden Zeichenketten als Werte behandelt. In diesem Abschnitt geht es darum, wie Zeichenketten gesäubert, bearbeitet und aufgeteilt werden.

> **Merke**
>
> Wenn Daten als Zeichenketten vorliegen, dann handelt es sich immer um **diskrete Daten**.

Tabelle 9.1.: Die wichtigsten Zeichenkettenfunktionen von Excel

Name	Excel
Länge	`LÄNGE()`
Teilketten verknüpfen	`TEXTKETTE()`/`TEXTVERKETTEN()`
Teilzeichenkette finden	`FINDEN()`/`SUCHEN()`
In Grossbuchstaben umwandeln	`GROSS()`
In Kleinbuchstaben umwandeln	`KLEIN()`
Nur ersten Buchstaben als Grossbuchstabe	`GROSS2()`
Leerzeichen bereinigen	`GLÄTTEN()`
Nicht-druckbare Zeichen entfernen	`SÄUBERN()`
Teilkette extrahieren (linksseitig)	`LINKS()`, `TEXTVOR()`
Teilkette extrahieren (rechtsseitig)	`RECHTS()`,`TEXTNACH()`
Teilketten extrahieren (alle)	`TEXTTEILEN()`
Teilkette zwischen zwei Positionen extrahieren	`TEIL()`
Zeichenkette ersetzen	`WECHSELN()`, `ERSETZEN()`

9.1. Die leere Zeichenkette

Ein besonderer Fall ist die *leere Zeichenkette*. Die leere Zeichenkette wird oft als Platzhalter genutzt. Die leere Zeichenkette ist das *neutrale Element* für die Verknüpfung von Zeichenketten mit `TEXTKETTE()` oder `TEXTVERKETTEN()`.

In Excel lässt sich die leere Zeichenkette von der leeren Zelle nur unterscheiden, indem die Formel betrachtet wird oder die Zelle mit `ISTLEER()` (`FALSCH`) und `ISTTEXT()` (`WAHR`) überprüft wird.

Die leere Zeichenkette wird in Excel nur als Funktionsparameter durch doppelte Anführungszeichen eingerahmt. Soll eine leere Zeichenkette als Wert in eine Zelle eingegeben werden, dann ist ein einfacher Apostroph (') einzugeben.

Beispiel 9.1 (Leere Zeichenkette in einer Excel-Formel)**.**

```
=WENN(1 = 1; ""; "Fehler")
```

> **Merke**
>
> Wenn in Excel eine leere Zeichenkette als Wert in eine Zelle eingetragen werden soll, dann wird ein einfaches Anführungszeichen als Wert eingegeben.

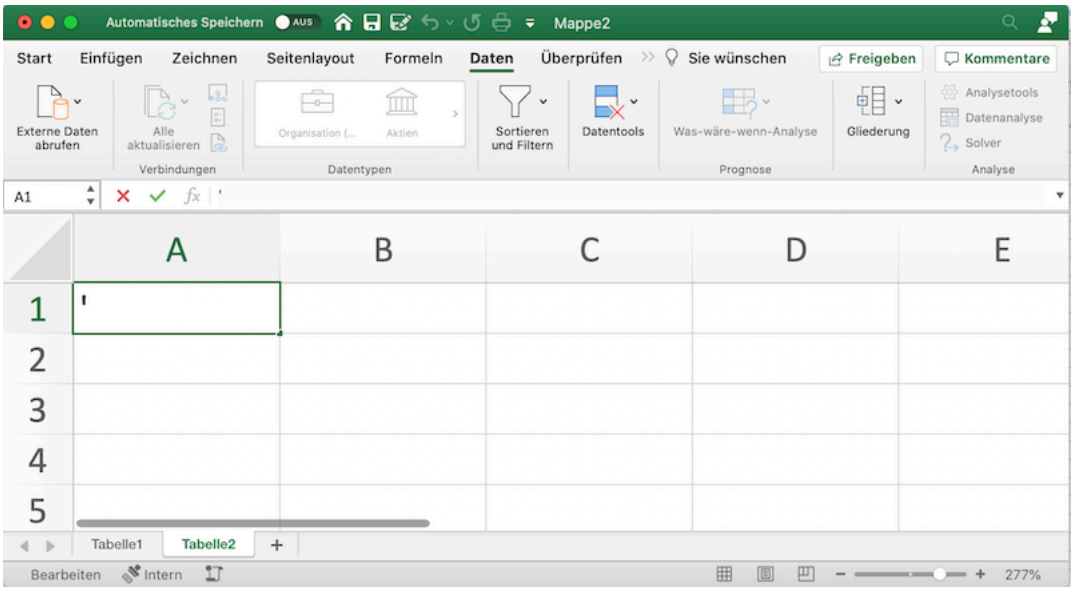


Abbildung 9.1.: Leere Zeichenkette als Zellenwert in Excel

9.2. Nicht-druckbare Zeichen

In Excel werden die nicht-druckbaren Zeichen für die Darstellung und für Vergleiche entfernt, jedoch werden die nicht-druckbaren Zeichen bei der Länge und beim Extrahieren berücksichtigt. In Excel kann mit der `IDENTISCH()`-Funktion geprüft werden, ob zwei Zeichenketten die gleiche Symbolfolge beinhalten. .

> **MacOS vs. Windows**
>
> Excel unter Windows stellt nicht-druckbare Zeichen als Kästchen dar, Excel für MacOS zeigt diese Zeichen nicht an.

Zu den nicht-druckbaren Zeichen gehören auch Leerzeichen, Tabulatoren und Zeilenumbrüche. Diese speziellen nicht-druckbaren Zeichen sind nur erkennbar, wenn sie von druckbaren Zeichen umgeben sind.

Deutlich wird das an den folgenden Zeichenketten:

- `Hallo`
- `Hal<0x07>lo`, wobei das Symbol `0x07` für einen Piepton steht
- `Hal<0x08>lo`, wobei das Symbol `0x08` für einmal Rückwärtslöschen steht.

Diese drei Zeichenketten haben in Excel die Längen 5, 6 und 6. Excel stellt alle drei Zeichenketten als "Hallo" dar. Ausserdem werden die Zeichenketten als gleich ausgewertet.

Excel entfernt über die Funktion `SÄUBERN()` alle nicht-druckbare Zeichen aus einer Zeichenkette.

9.3. Zeichenketten trennen

9.3.1. Einzelne Symbole extrahieren

In Excel lassen sich die einzelnen Symbole einer Zeichenkette mit der folgenden Formel extrahieren:

```
=TEIL("Daten und Information";
      SEQUENZ(LÄNGE("Daten und Information"));
      1)
```

Diese Formel hat drei Funktionsaufrufe.

1. Die Funktion `LÄNGE()` bestimmt die Anzahl der Symbole in der Zeichenkette.
2. Mit der Funktion `SEQUENZ()` werdem alle Positionen der Symbole durchnummeriert.
3. Mit `TEIL(Zeichenkette; Sequenz; 1)` wird ein Teil der Zeichenkette extrahiert, wobei für jede Position der Sequenz aus Schritt 2 eine Teilzeichenkette mit der Länge 1 erzeugt wird.

9.3.2. Zeichenketten vor und nach einem Trenner erhalten

Die Funktion `LINKS()` und `RECHTS()` geben eine Teilzeichenkette mit einer festen Anzahl von Zeichen zurück. Die Funktion `LINKS()` zählt die Anzahl der Zeichen vom Beginn der Zeichenkette und `RECHTS()` vom Ende her.

Beispiel 9.2 (Zeichen links bis und mit der 5. Position zurückgegeben)**.**

```
= LINKS("Daten und Information"; 5)
```

Weil `RECHTS()` vom Ende her zählt, kann nicht die gleiche Position wie bei `LINKS()` verwendet werden. Dazu muss die Position von der Gesamtlänge der Zeichenkette abgezogen werden (s. Beispiel 9.3).

Beispiel 9.3 (Zeichen rechts von der 10. Position zurückgegeben)**.**

```
= RECHTS(
    "Daten und Information";
    LÄNGE("Daten und Information")-10)
```

Etwas eleganter arbeiten die beiden Funktionen `TEXTVOR()` und `TEXTNACH()`. Diese Funktionen suchen nach einer Teilzeichenkette und ergeben die Zeichen vor bzw. nach dieser Zeichenkette. Ein zusätzlicher Parameter ermöglicht die Suche solange zu wiederholen, bis die gesuchte Zeichenkette genau so oft gefunden wurde.

Beispiel 9.4 (`TEXTVOR()` vor dem ersten Treffer)**.**

```
= TEXTVOR("Daten und Information"; " ")
```

Beispiel 9.5 (`TEXTNACH()` ab dem zweiten Treffer)**.**

```
= TEXTNACH("Daten und Information"; " "; 2)
```

Um einen Wert zwischen zwei Treffern zu extrahieren, lassen sich die beiden Funktionen verketten. Diese Technik bietet sich immer dann an, wenn ein Wert zwischen zwei *unterschiedlichen* Markierungen extrahiert werden soll

Beispiel 9.6 (Text zwischen zwei unterschiedlichen Teilketten extrahieren)**.**

```
= TEXTVOR(TEXTNACH("Daten und Information"; "ten "); " Info")
```

9.3.3. Festkodierte Werte trennen

Eine festkodierte Datenstruktur ist eine Zeichenkette, die Werte an festgelegten Positionen mit konstanten Längen enthält. Diese Daten lassen sich mit der Funktion `TEIL()` extrahieren.

Die Funktion hat drei Argumente:

- Die Zeichenkette, aus der die Daten extrahiert werden sollen.
- Die Position, an der die Daten beginnen.
- Die Länge der Daten als Anzahl von Symbolen.

Beispiel 9.7 (IBAN in Land, Prüfziffer, Bankkennung und Kontonummer trennen)**.** Die IBAN ist eine festkodierte Datenstruktur. Die IBAN enthält die Länderkennung, die Prüfziffer, die Bankkennung und die Kontonummer. Die Länge der einzelnen Daten ist konstant und die Position der Felder ist festgelegt.

Feld	Position	Länge
Land	1	2
Prüfziffer	3	2
Bankkennung	5	5
Kontonummer	10	Länge der IBAN - 10

Die (ungültige) IBAN `CH12BANK1002135135` kann mit der `TEIL()`-Funktion in die einzelnen Felder zerlegt werden. Dazu müssen zuerst die Positionen und Längen der Felder erstellt werden. Dazu werden die Positionen und Längen der Felder untereinander geschrieben.

A	B	C	D
1	3	5	10
2	2	5	`= LÄNGE(IBAN_Nummer) - 10`

Diese Werte werden als Vektoren der Funktion `TEIL()` übergeben.

```
= TEIL(IBAN_Nummer; A1:D1; A2:D2)
```

Die Funktion `TEIL()` gibt die einzelnen Felder als Vektor zurück. Das Ergebnis ist `{"CH"; "12"; "BANK1"; "002135135"}`. Hier muss berücksichtigt werden, dass die einzelnen Felder weiterhin Zeichenketten sind.

9.3.4. Zeichenketten mit einem Separator trennen

Sind Werte in einer Zeichenkette durch einen Separator getrennt, dann lassen sich die Werte mit der Funktion `TEXTTEILEN()`. Die Funktion kann eine Zeichenkette entlang von zwei Trennzeichen trennen. Dabei erzeugt die Funktion eine Matrix, wobei die Spalten durch den ersten Separator und die Zeilen durch den zweiten Separator bestimmt werden.

Beispiel 9.8 (`TEXTTEILEN()` mit festem Trennzeichen)**.**

```
= TEXTTEILEN("Daten und Information", " ")
```

> **!** Wichtig
>
> Die Funktion `TEXTTEILEN()` kann nur auf einen Wert angewandt werden. Wird ein Zeichenkettenvektor der Funktion übergeben, wird aus jeder Zeichenkette immer nur der erste Wert getrennt und als Ergebnis ausgegeben. Excel verhindert so, dass durch das Teilen mehrerer Zeichenketten die Ergebnisse einen nicht eindeutig grossen rechteckigen Bereich mit Datentyp Zeichenkette erzeugen könnten.

Dieses Problem wird durch einen Trick umgangen: Dazu wird ein zweiter Separator gewählt, der nicht in den Daten vorkommt und sich vom ursprünglichen Separator unterscheidet. Existiert bereits ein Zeilentrennzeichen in den Daten, wird dieses als zweiter Separator verwendet. Anschliessend werden alle Zeichenketten mit diesem zweiten Separator als Trennzeichen mit `TEXTVERKETTEN()` verkettet. Abschliessend wird die gesamte Zeichenkette mit dem ursprüngliche Separator als Spalten- und dem zweiten Separator als Zeilentrennzeichen getrennt. Dabei werden alle nicht vorhandenen Werte in einer Zeile durch `#NV` angezeigt.

Beispiel 9.9 trennt einen Zeichenkettenvektor mit Werten, die durch ein Leerzeichen getrennt sind. Alle Werte sind Buchstaben oder Ziffern, aber keine Satzzeichen. Deshalb kann das Komma (`,`) als Zeilentrennzeichen verwendet werden. Damit der Fehlerwert `#NV` nachfolgende Operationen nicht behindert, wird dieser durch die leere Zeichenkette ersetzt.

Beispiel 9.9 (Einen Zeichenkettenvektor entlang eines Trennzeichens teilen.)**.**

```
=WENNFEHLER(
    TEXTTEILEN(
        TEXTVERKETTEN(",";; Zeichenketten);
        " ";
        ",");
    ""
)
```

9.4. Suchen und Ersetzen

Eine wichtige Operation für Zeichenketten ist das Suchen-und-Ersetzen. Das Suchen-und-Ersetzen lässt sich als eine spezielle Technik zur Mustererkennung vorstellen. Dabei steht eine Anwendung im Vordergrund: Die Korrektur und Vereinheitlichung von Zeichenketten. Anstatt solche Daten umständlich über eine Benutzeroberfläche zu korrigieren, sollte das Bereinigen von Zeichenketten besser automatisch erfolgen. Excel stellt dazu die Funktion `WECHSELN()` zur Verfügung. Diese Funktion ersetzt entweder alle Vorkommnisse einer Teilzeichenkette oder nur ihr n-tes Auftreten.

Beispiel 9.10 (Alle Vorkommnisse einer Zeichenkette ersetzen)**.**

```
= WECHSELN(
    "Daten oder Information oder Wahrscheinlichkeit";
    "oder";
    "und")
```

Beispiel 9.11 (Erstes Auftreten einer Zeichenkette ersetzen)**.**

```
= WECHSELN(
    "Daten oder Information oder Statistik";
    "oder";
    "und";
    1)
```

9.4.1. Löschen von Teilzeichenketten

Teilzeichenketten lassen sich löschen, indem die fragliche Zeichenkette durch die leere Zeichenkette ersetzt wird.

Beispiel 9.12 (Teilzeichenkette entfernen)**.**

```
= WECHSELN("Daten und Information"; "und"; "")
```

9.4.2. Zeichenketten durch Ersetzungen vereinheitlichen

Bevor das eigentliche Suchen-und-Ersetzen starten kann, sollten die betreffenden Zeichenketten bereinigt werden. Die einfachste Bereinigung ist das Entfernen von überschüssigen Leerzeichen. Dazu dient die Excel Funktion `GLÄTTEN()`. `GLÄTTEN()` entfernt alle Leerzeichen am Anfang und Ende einer Zeichenkette. Alle wiederholten Leerzeichen werden mit `GLÄTTEN()` zu einem einzelnen Leerzeichen zusammengefasst.

Eine zweite häufig verwendete Bereinigung ist die Transformation auf Kleinbuchstaben oder Grossbuchstaben. Hierzu dienen die drei Funktionen `GROSS()`, `GROSS2()` und `KLEIN()`. Die Vereinheitlichung der Schreibweise ist ein wichtiges Werkzeug um Zeichenketten mit unterschiedlichen Schreibweisen zu vereinheitlichen. Beim Bereinigen sollte eine der verfügbaren Varianten gewählt und konsequent für die Vereinheitlichung verwendet werden. Damit wird sicher gestellt, dass alle Ersetzungen unabhängig von der Schreibweise erfolgen.

Für das eigentliche Ersetzen wird ein Suchvektor und einen Ersetzenvektor erzeugt. Die beiden Vektoren enthalten Paare aus Suchmuster und Ersetzung. Diese Paare sind geordnet und werden nacheinander ausgeführt.

Praxis

Satzzeichen sollte immer durch Leerzeichen und nicht durch eine leere Zeichenkette ersetzt werden. Die überzähligen Leerzeichen können anschliessend mit `GLÄTTEN()` entfernt werden. Dadurch ist gesichert, dass nicht versehentlich Elemente zusammengefügt werden. Nachdem alle Sonderzeichen entfernt wurden, sollte eine Zeichenkette noch einmal von überschüssigen Leerzeichen bereinigt werden.

10. Aussagenlogik

Dieses Kapitel befasst sich mit der Anwendung der Aussagenlogik in Excel. Es werden die logischen Operatoren NICHT, UND, ODER und XODER behandelt. Auf dieser Grundlage werden die wichtigsten Vergleichsoperation zum Formulieren logischer Ausdrücke vorgestellt. Das Kapitel schliesst mit der Anwendung von Fallunterscheidungen.

Es werden die folgenden Funktionen behandelt: `NICHT()`, `UND()`, `ODER()`, `XODER()`, `WENN()`, `WENNS()`, `ERSTERWERT()`, `WENNFEHLER()`, `FEHLER.TYP()`, `ISTFEHLER()`, `XVERWEIS()`, `XVERGLEICH()`, `IDENTISCH()`, `FILTER()`, `SORTIEREN()`, `SORTIERENNACH()` und `SPALTENWAHL()`

10.1. Wahrheitswerte in Excel

Wie im Kapitel Datentypen bereits erwähnt, kennt Excel den Datentyp der Wahrheitswerte. Diese Wahrheitswerte können entweder den Wert `WAHR` oder `FALSCH` haben. Eine Operation, die Wahrheitswerte ergibt wird als *logischer Ausdruck* bezeichnet. Weil logische Ausdrücke für viele Funktionen und Operationen notwendig sind, wandelt Excel die Werte anderer Datentypen bei Bedarf um. Dabei gelten die folgenden Regeln:

- `0` und die *leere Zelle* entspricht dem Wert `FALSCH`.
- Alle Zahlen ungleich `0` entsprechen dem Wert `WAHR`.
- Alle Zeichenketten inklusive der *leeren Zeichenkette* entsprechen dem Wert `WAHR`.
- Fehlerwerte bleiben unverändert.

Werden Wahrheitswerte in mathematischen Operationen und Funktionen verwendet, dann konvertiert Excel den Wert `FALSCH` in `0` und den Wert `WAHR` in `1` um.

Werden Wahrheitswerte als Parameter an Zeichenkettenfunktionen übergeben, dann werden die Wahrheitswerte in die entsprechende Zeichenkette umgewandelt. Aus dem Wert `WAHR` wird also die Zeichenkette `"WAHR"` und aus dem Wert `FALSCH` wird die Zeichenkette `"FALSCH"`.

> **Warnung**
>
> Wahrheitswerte werden in den verschiedenen Sprachversionen von Excel in der eingestellten Sprache angegeben. Beim Wechsel zwischen verschiedenen Excel-Sprachversionen werden die Wahrheitswerte automatisch korrekt angezeigt. Die Um-

wandlung in Zeichenketten erfolgt dann in der jeweiligen Sprache. Deshalb sollte die Verwendung von in Zeichenketten konvertierten Wahrheitswerten in nachgelagerten Funktionen vermieden werden.

10.2. Aussagenlogische Operationen

Für Wahrheitswerte existieren spezielle Operationen, um die Regeln der *Aussagenlogik* bzw. der *Boole'schen Algebra* abzubilden. Diese Operationen verknüpfen Wahrheitswerte und haben Wahrheitswerte als Ergebnis. Die vier Grundoperationen `NICHT` ($\neg$), `UND` ($\wedge$), `ODER` ($\vee$, "inklusives Oder") und `XODER` ($\oplus$, "entweder-oder") sind in Excel als Funktionen verfügbar.

Die Funktion `NICHT()` wandelt einen Wert in den jeweils den anderen Wahrheitswert um. Falls anstelle eines Wahrheitswerts ein anderer Datentyp übergeben wurde, gelten die oben angegebenen Regeln für die Umwandlung.

Die Funktionen `UND()`, `ODER()` und `XODER()` sind *Aggregatoren*. Das bedeutet, dass Sie alle Werte in dem angegebenen Bereichen zusammenfassen. Das ist oft nicht das gewünschte Verhalten. Deshalb muss bei der Arbeit mit Vektoren auf die Boolesche Arithmetik zurückgegriffen werden, um logische Ausdrücke richtig auszuwerten.

Zum Beispiel sollen für die folgenden Werte *paarweise* der logische Ausdruck $a \wedge b$ ausgewertet werden, so dass für alle Wertepaare der richtige Wahrheitswert ermittelt wird.

A	B
WAHR	FALSCH
FALSCH	WAHR
WAHR	WAHR
FALSCH	WAHR
WAHR	WAHR
FALSCH	FALSCH

Die Formel `= UND(A1:A6; B1:B6)` liefert den Wert `FALSCH` zurück, weil nicht alle Werte im *gesamten Bereich* von `A1:A6` und `B1:B6` gleich `WAHR` sind. Es gibt keine Funktion und keinen eigenen logischen Operator zur paarweisen logischen Verknüpfung dieser beiden Bereiche. Deshalb werden in Excel oft logische Ausdrücke in der Boole'schen Arithmethik eingesetzt, um nur die Werte aus den gleichen Datensätzen miteinander zu vergleichen. Diese Schreibweise ist immer dann notwendig, wenn logische Ausdrücke sich auf die einzelnen Datensätze beziehen.

Die Formel `= A1:A6 * B1:B6` hat die Werte `{0;0;1;0;1;0}` zum Ergebnis. Um Wahrheitswerte zu erhalten, kann noch auf die Ungleichheit mit `0` geprüft werden. Dazu wird die Formel wie folgt ergänzt: `= (A1:A6 * B1:B6) <> 0`. Das Ergebnis ist nun `{FALSCH; FALSCH;`

WAHR; FALSCH; WAHR; FALSCH}. Dieser Schritt ist in der Praxis selten notwendig, weil für die meisten Operationen Zahlenwerte implizit als Wahrheitswerte behandelt werden.

Weil Excel alle Werte ungleich 0 als WAHR interpretiert, können die Operationen UND() mit * und die Operation ODER() mit + direkt ersetzt werden. Hierbei ist darauf zu achten, dass das nummerische Ergebnis dieser Addition oder Multiplikation *ausschliesslich* als Wahrheitswert von Bedeutung ist.

Die Operation XODER() entspricht der Ungleichheit <>. Dabei muss allerdings darauf geachtet werden, das dieser Vergleich als Ersatz für XODER() *ausschliesslich* für Wahrheitswerte bzw. 0 und 1 erlaubt ist. Für die oben gezeigten Werte ergibt die Formel = (A1:A5 <> B1:B5) die Werte {WAHR; WAHR; FALSCH; WAHR; FALSCH; FALSCH}. Das Ergebnis ist deshalb sichergestellt, weil alle Vergleichswerte 0 oder 1 sind. Werden jedoch auch andere Vergleichswerte zugelassen, dann liefert die Formel = (A1:A6 <> B1:B6) die Werte {WAHR; WAHR; WAHR; WAHR; WAHR; WAHR}. Dieses Verhalten zeigt das folgende Beispiel.

A	B
1	0
0	2
1	2
0	1
3	2
0	0

Damit das richtige Ergebnis erzeugt wird, müssen die Werte in Wahrheitswerte konvertiert werden. Dazu muss die Formel durch Vergleiche ungleich 0 erweitert werden:

```
= (A1:A6 <> 0) <> (B1:B6 <> 0)
```

Diese Formel liefert die gewünschten Werte {WAHR; WAHR; FALSCH; WAHR; FALSCH; FALSCH}.

Die folgende Tabelle zeigt logischen Operatoren und die zugehörigen Terme für die Boole'sche Arithmetik.

Operator	Boole'sche Operation	Vereinfachter Operator
$\neg$	1 - a	NICHT(a)
$\wedge$	a * b	a * b
$\vee$	a + b - a * b	a + b
$\oplus$	a + b - 2 * a * b oder (a - b) ^ 2	(a <> b)

10.3. Vergleiche

Eine besondere Art von logischen Ausdrücken sind *Vergleiche*. Ein Vergleich prüft das Verhältnis zweier Werte zueinander. Excel kennt die üblichen Vergleichsoperatoren, die jeweils einen Wahrheitswert zurückliefern.

In Excel werden die Vergleichsoperatoren wie folgt geschrieben:

- `>` (grösser als)
- `<` (kleiner als)
- `>=` (grösser oder gleich)
- `<=` (kleiner oder gleich)
- `=` (gleich)
- `<>` (ungleich)

Excel's Vergleichsoperatoren sind Datentypen sensitiv. Das bedeutet, dass die Operatoren Datentypen vor dem Vergleich *nicht* angleichen. Der folgende Vergleich ergibt also `FALSCH`.

```
= 3 = "3"
```

> **Merke**
>
> Ziffern sind keine Zahlen!

Weil die Operationen `*`, `+` und `-` normalerweise vor den Vergleichsoperatoren ausgeführt werden, müssen alle Vergleiche eines logischen Ausdrucks für die Boole'sche Arithmetik in Klammern gesetzt werden.

10.3.1. Der ∈-Operator mit XVERWEIS

Mithilfe der Funktion `XVERGLEICH()` kann der ∈-Operator aus der *Mengenlehre* in Excel für logische Ausdrücke bereitgestellt werden. Mit dieser Funktion `XVERWEIS()` können sowohl der ∈ als auch der ∉-Operator mit `XVERWEIS()` abgebildet werden.

Für die folgenden Beispiele verwenden wir die Werte:

A	B
Suchkriterium	Suchbereich
4	1
7	3
	4
	8

- ∈-Operator: `XVERWEIS(A2:A3; B2:B5; B2:B5 = B2:B5; FALSCH)`
- ∉-Operator: `XVERWEIS(A2:A3; B2:B5; B2:B5 <> B2:B5; WAHR)`

Der Trick besteht darin, dass die *Rückgabematrix* durch einen Vergleich aus dem Suchbereich erzeugt wird. Dadurch wird die Rückgabematrix mit den gleichen Wahrheitswerten für alle Werte im Suchbereich gefüllt. Die erste Formel ergibt deshalb `{WAHR; FALSCH}` und die zweite Formel `{FALSCH; WAHR}`, weil der Wert `4` im Suchbereich vorkommt und der Wert `7` nicht. Diese Werte können direkt in logischen Ausdrücken verwendet werden.

10.3.2. Zeichenketten vergleichen

Die Vergleichsoperatoren zeigen die Unterschiede zweier Zeichenketten bezüglich der alphabetischen Sortierung an. Die "kleinste" Zeichenkette ist die leere Zeichenkette. Gross- und Kleinschreibung wird bei Zeichenkettenvergleichen nicht unterschieden.

Weil Excel für Vergleiche die nicht-druckbaren Zeichen mit Ausnahme des Leerzeichens und des Tabulators ignoriert, gibt der Vergleichsoperator `= WAHR` auch für Zeichenketten zurück, die unterschiedliche nicht-druckbare Zeichen enthalten. Das gleiche Problem entsteht beim Vergleich von unterschiedlicher Gross- und Kleinschreibung. Um auch diese Unterschiede zu erkennen, müssen wir die Funktion `IDENTISCH()` verwenden. Diese Funktion vergleicht die Zeichenketten Zeichen für Zeichen und liefert nur dann `WAHR` zurück, wenn die Zeichenketten exakt gleich sind.

Beispiel 10.1. ## Anwendung der `IDENTISCH()` Funktion als Vergleichsoperator.

```
= WENN(IDENTISCH("A"; "a"); "Gleich"; "Ungleich")
```

10.4. Komplexe logische Ausdrücke und Datenstrukturen

Excel hat zwar Funktionen für die logischen Operatoren *Und* (`UND()`), Oder (`ODER()`) und *Exklusives Oder* (`XODER()`), diese Funktionen haben aber den Nachteil, dass sie nur Werte zusammenfassen können. In der Praxis werden jedoch oft *Datenstrukturen* als Variablen für logische Ausdrücke verwendet. Diese sollen durch die logischen Operatoren verknüpft und nicht zusammengefasst werden.

Um logische Verknüpfungen für Datenstrukturen zu realisieren, müssen die logischen Operatoren mit der Boole'schen Arithmetik umgesetzt werden. Dazu werden die logischen Operatoren durch die entsprechenden arithmetischen Operatoren ersetzt.

Beispiel 10.2. Für die folgenden Werte soll der folgende logische Ausdruck geprüft werden.

$$a \wedge b \vee (c < 10) \wedge d$$

A	B	C	D
WAHR	5	21	17
FALSCH	3	5	1
WAHR	0	10	2
FALSCH	1	11	3

Für diesen Ausdruck werden die Werte `{WAHR; WAHR; FALSCH; FALSCH}` erwartet.

Die naive Umsetzung `=ODER(UND(A1:A4;B1:B4);UND(C1:C4;D1:D4))` hat als Ergebnis den Wert `WAHR`.

Die Formel `= A1:A4 * B1:B4 + (C1:C4 < 10) * D1:D4` liefert die Werte `{5; 1; 0; 0}` diese lassen sich durch einen Vergleich in die entsprechenden Wahrheitswerte konvertieren. Diese Konversion ist nur notwendig, wenn der Ausdruck nicht als logischer Ausdruck an eine Funktion übergeben wird.

```
= (A1:A4 * B1:B4 + (C1:C4 < 10) * D1:D4) <> 0
```

Diese Formel ergibt die erwarteten Werte `{WAHR; WAHR; FALSCH; FALSCH}`.

Die Ausnahme von dieser Regel ist Verwendung der logischen Funktionen als Tabellenfunktion. In diesem Fall können die Werte zeilenweise auch von den logischen Funktionen verarbeitet werden.

Beispiel 10.3. Werden die Werte aus Beispiel 10.2 als Tabelle konvertiert und mit `BeispielTabelle` benannt, dann können die logischen Funktionen in einer *gleich langen Tabelle* mit der *gleichen Startzeile* auch als Tabellenfunktionen verwendet werden. In diesem Fall werden die Werte zeilenweise verarbeitet.

```
=ODER(
      UND(BeispielTabelle[@a];BeispielTabelle[@b]);
      UND(BeispielTabelle[@c] < 10; BeispielTabelle[@d])
 )
```

Diese Formel ergibt die erwarteten Werte `{WAHR; WAHR; FALSCH; FALSCH}`.

10.5. Fälle unterscheiden

Logische Ausdrücke eigenen sich besonders gut, um *Fallunterscheidungen* zu formulieren, weil ein logischer Ausdruck immer nur zwei Werte als Ergebnis haben kann. Es gibt also für jeden logischen Ausdruck immer nur zwei unterscheidbare Fälle.

Excel hat zwei zentrale Funktionen für Fallunterscheidungen: `WENN()` und `WENNS()`. Die Funktion `WENN()` ist eine *einfache* Unterscheidung, die Funktion `WENNS()` unterstützt *mehrfache* Unterscheidungen. In anderen Programmiersprachen wird in diesem Zusammenhang auch von *Verzweigungen* gesprochen.

10.5.1. WENN

Die Funktion `WENN()` ist eine *einfache* Fallunterscheidung. Einfach bedeutet hier, dass die beiden Fälles *eines* logischen Ausdrucks unterschieden werden. Entsprechend hat die Funktion `WENN()` drei Parameter:

1. Der auszuwertende logische Ausdruck.
2. Das Ergebnis falls der logische Ausdruck `WAHR` ergibt.
3. Das Ergebnis falls der logische Ausdruck `FALSCH` ergibt.

Das Ergebnis für den Fall, dass der logische Ausdruck `FALSCH` ergibt, ist optional. Fehlt dieser Parameter, dann wird der Wert `FALSCH` zurückgegeben.

Das Verhalten dieser Funktion lässt sich mit den Wahrheitswerten als logischer Ausdruck direkt überprüfen:

```
= WENN(WAHR; "Guten Tag"; "Auf Wiedersehen")
```

Weil der logische Ausdruck in diesem Fall `WAHR` ist, wird der zweite Parameter als Ergebnis zurückgegeben. Die Formel gibt also den Wert `"Guten Tag"` zurück.

Wird der logische Ausdruck auf `FALSCH` geändert, dann liefert die Formel den Wert `"Auf Wiedersehen"`.

```
= WENN(FALSCH; "Guten Tag"; "Auf Wiedersehen")
```

Lassen wir den dritten Parameter weg, dann wird der Wert `FALSCH` zurückgegeben.

```
= WENN(FALSCH; "Guten Tag")
```

Ausser der Fallunterscheidung hat `WENN()` keine weiteren Eigenschaften. Deshalb wird diese Funktion in der Praxis oft mit anderen Funktionen kombiniert. Das kann mit der Funktion `WENN()` selbst geschehen. In diesem Fall wird von geschachtelten Fallunterscheidungen gesprochen.

Zur Veranschaulichung dient das folgende Beispiel:

	A
1	4

	A
2	7

Eine Fallunterscheidung soll prüfen, ob die Werte in `A1:A2` Werte gleich `1`, `3`, `4` oder `8` sind. Falls das der Fall ist, soll der zugehörige Zahlwert als Zeichenkette ausgegeben werden. Falls das nicht der Fall ist, soll der Wert `Ungültig` zurückgegeben werden. Als geschachtelte `WENN()`-Funktion lässt sich diese Fallunterscheidung wie folgt formulieren:

```
= WENN(A1 = 1; "Eins";
       WENN(A1 = 3; "Drei";
            WENN(A1 = 4; "Vier";
                 WENN(A1 = 8; "Acht";
                      "Ungültig"))))
```

Eine solche geschachtelte Fallunterscheidung wird als *Entscheidungsbaum* bezeichnet.

10.5.2. WENNS

Die Funktion `WENN()` ist eine *einfache* Fallunterscheidung. In vielen Excel-Arbeitsmappen existieren geschachtelte Aufrufe von `WENN()`-Funktionen. Diese Aufrufe machen die Formeln nicht nur schwer lesbar, sondern auch fehleranfällig und ineffizient. Deshalb sollten geschachtelte Fallunterscheidungen unbedingt vermieden werden. Mit der Funktion `WENNS()` lassen sich geschachtelte Fallunterscheidungen vermeiden, indem alle Fallunterscheidungen in einem einzigen Funktionsaufruf zusammengefasst werden.

> **Merke**
>
> Geschachtelte Fallunterscheidungen mit `WENN()` unbedingt vermeiden!

Die Funktion `WENNS()` erwartet Parameterpaare, bestehend aus einem logischen Ausdruck und dem Ergebnis, falls dieser logische Ausdruck `WAHR` ergibt. Die Funktion kann bis zu 127 Parameterpaare verarbeiten, so dass sich auch sehr komplexe Fallunterscheidungen mit dieser Funktion abbilden lassen.

Beispiel 10.4. Das folgende Beispiel zeigt die Verwendung der Funktion `WENNS()` für die geschachtelte Fallunterscheidung aus dem Abschnitt `WENN`.

```
= WENNS(A2:A3 = 1; "Eins";
        A2:A3 = 3; "Drei";
        A2:A3 = 4; "Vier";
        A2:A3 = 8; "Acht")
```

Das Beispiel bildet aber noch nicht die vollständige Fallunterscheidung ab. Es fehlt noch der Fall, dass keiner der logischen Ausdrücke `WAHR` ergibt. Leider kann `WENNS()` ausschliesslich logische Ausdrücke mit ihren Ergebnissen verbinden.

> **Merke**
>
> `WENNS()` kann nur logische Ausdrücke mit ihren `WAHR`-Ergebnissen verbinden.

Anders als bei `WENN()` gibt es keine direkte Möglichkeit, ein Ergebnis festzulegen, falls alle logische Ausdrücke `FALSCH` ergeben. Um ein solches Verhalten zu erzeugen, wird ausgenutzt, dass die Funktion `WENNS()` immer einen wahren logischen Ausdruck mit einem Ergebnis verknüpft. Weil die logischen Ausdrücke in der Reihenfolge ausgewertet werden, wie sie in der Funktion angegeben sind, muss der letzte logische Ausdruck alle Fälle abdecken, die von keinem anderen der vorangegangen logischen Ausdrücke akzeptiert wurden. Der einfachste logische Ausdruck, der immer wahr ist, ist der Wahrheitswert `WAHR`. Deshalb wird dieser Wert als letzter logischer Ausdruck für `WENNS()` verwendet.

Mit diesem Wissen lässt sich das Beispiel mit `WENNS()` vervollständigen:

```
= WENNS(A2:A3 = 1; "Eins";
        A2:A3 = 3; "Drei";
        A2:A3 = 4; "Vier";
        A2:A3 = 8; "Acht";
        WAHR; "Ungültig")
```

Diese Formel prüft die Werte in `A2:A3` auf Gleichheit mit den Werten `1`, `3`, `4` und `8`. Für diese Zahlen wird die zugehörige Zahlwert als Zeichenkette ausgegeben. Falls keiner dieser Werte gefunden wird, wird der Wert `Ungültig` zurückgegeben.

Die Fallunterscheidung mit `WENNS()` endet beim ersten logischen Ausdruck, der `WAHR` ergibt. Die Funktion prüft der Reihe nach alle angegebenen logischen Ausdrücke. Sobald einer dieser Ausdrücke `WAHR` ist, wird der zugehörige Ergebniswert ausgegeben und die Funktion wird beendet. Diese Eigenschaft begründet, dass die logischen Ausdrücke nur die Fälle prüfen müssen, die von den vorangegangenen logischen Ausdrücken nicht abgedeckt wurden.

Beispiel 10.5 (Fallunterscheidung mit `WENNS()` vereinfachen)**.** Gegeben ist die Formel mit geschachtelten Entscheidungen.

```
= WENN(J2>=O2;
    (WENN(J2>L2;
          0;
          WENN(J2<=L2;
               WENN((J2>N2)*(J2>=O2);
                    (K2+((M2-K2)/(N2-L2))*(J2-L2));
                    WENN((J2<=N2)*(J2<O2);
```

```
                        (M2+((O2-M2)/(O2-N2))*(J2-N2))
                ))
        )
    ));
    100)
```

Diese Formel ist aus zwei Gründen übermässig komplex.

1. Die Fallunterscheidung mit `WENN()` ist geschachtelt.
2. Es existieren *redundante* Fallunterscheidungen.

Bevor die Fallunterscheidung mit `WENNS()` vereinfacht wird, werden die redundanten Fallunterscheidungen entfernt. Das betrifft die zweite (`J2 > L2`) und die vierte Fallunterscheidung (`J2 > N2`). Im jeweiligen `FALSCH`-Fall wird der gegenteilige logische Ausdruck geprüft. Das ist in diesem Fall unnötig, weil die äussere Fallunterscheidung diesen Fall bereits abdeckt. Werden die redundanten logischen Ausdrücke und unnötige Klammern entfernt, dann ergibt sich die folgende wesentlich einfachere Formel.

```
= WENN(J2>=O2;
    WENN(J2>L2;
          0;
          WENN((J2>N2)*(J2>=O2);
              K2+(M2-K2)/(N2-L2)*(J2-L2);
              M2+(O2-M2)/(O2-N2)*(J2-N2)
          )
    );
    100)
```

Die äusserste Fallunterscheidung hat für den Fall `WAHR` eine geschachtelte `WENN()`-Funktion und im Fall `FALSCH` ein einfaches Ergebnis. Das ist für `WENNS()` unhandlich, so dass die äusserste Fallunterscheidung durch Umkehrung des logischen Ausdrucks umgestellt wird.

```
= WENN(J2<O2;
       100;
       WENN(J2>L2;
            0;
            WENN((J2>N2)*(J2>=O2);
                 K2+(M2-K2)/(N2-L2)*(J2-L2);
                 M2+(O2-M2)/(O2-N2)*(J2-N2)
            )
       )
  )
```

Nun lassen sich die vier unterschiedlichen Fälle gut erkennen und mit `WENNS()` abbilden. Daraus ergibt sich die folgende Formel.

```
= WENNS(J2<O2; 100;
        J2>L2; 0;
        (J2>N2)*(J2>=O2); K2+(M2-K2)/(N2-L2)*(J2-L2);
        WAHR; M2+(O2-M2)/(O2-N2)*(J2-N2)
  )
```

Diese Formel ist wesentlich einfacher zu lesen und zu verstehen. Beim Durchgehen der Fälle fällt auf, dass ein Teilausdruck des dritten Falls das Gegenteil des ersten Falls ist. Diese Bedingung wurde bereits im ersten Fall geprüft und würde sie nicht gelten, dann wäre die Formel bereits beendet worden. Deshalb können bereits geprüfte Teilausdrücke in den nachfolgenden Ausdrücken weggefallen. Dadurch wird nicht nur die Verschachtelung, sondern auch die Komplexität der logischen Ausdrücke vereinfacht.

```
= WENNS(J2<O2; 100;
        J2>L2; 0;
        J2>N2; K2+(M2-K2)/(N2-L2)*(J2-L2);
        WAHR;  M2+(O2-M2)/(O2-N2)*(J2-N2)
  )
```

Diese Formel hat noch den Makel, dass der letzte Fall `WAHR` keine Konstante abbildet. Besser wäre es, wenn der zweite und der letzte Fall vertauscht wären, so dass der Wert `0` der letzte Wert ist. Dazu müssen die logischen Ausdrücke umorganisiert werden. Bei der Umorganisation ist die Reihenfolge der logischen Ausdrücke zu beachten: Die letzten beiden Fälle sind nicht unabhängig vom logischen Ausdruck `J2>L2`. Beim Umorganisieren darf diese Abhängigkeit nicht verloren gehen.

```
= WENNS(J2<O2; 100;
        (J2<=L2)*(J2>N2); K2+(M2-K2)/(N2-L2)*(J2-L2);
        J2<=L2; M2+(O2-M2)/(O2-N2)*(J2-N2)
        WAHR; 0
  )
```

Diese Formel ist deutlich einfacher und weniger Fehleranfällig als die ursprüngliche Formel mit geschachtelten `WENN()`-Funktionen. Es lassen sich auch weitere Fälle hinzufügen, ohne dass die Formel komplexer wird. Dabei ist zu beachten, dass diese Fälle *vor* dem Fall `WAHR` angegeben werden müssen.

10.5.3. Nicht erreichbare Entscheidungen

Ein besonderes Problem sind Entscheidungen, die zwar definiert aber nie erreicht werden können. Solche Entscheidungen sind immer *redundant*. Eine nicht erreichbare Entscheidung kann nur dann auftreten, wenn eine vorangegangene Entscheidung bereits den geprüften Fall abdeckt. Ergibt ein solcher logischer Ausdruck `Falsch`, dann wird eine spätere Entscheidung für den gleichen Fall im Falsch-Zweig des Entscheidungsbaums ebenfalls `Falsch` ergeben. Der Wahr-Zweig dieser Entscheidung kann damit nie erreicht werden.

Beispiel 10.6 (Nicht erreichbare Entscheidung)**.**

```
=WENNS( A1 > 5; "Sehr gut";
        A1 > 3; "Genügend";
        A1 > 4; "Gut";
        A1 <= 3; "Ungenügend")
```

In Beispiel 10.6 kann nie das Ergebnis “Gut” erzeugt werden, weil der zweite logische Ausdruck (A1 $>$ 3) alle Werte “maskiert”, die durch den dritten logischen Ausdruck (A1 $>$ 4) als “Gut” markiert werden müssten. “Ungenügend” würde trotzdem angezeigt werden, wenn der Wert in A1 entweder 1, 2 oder 3 ist.

> **!** Wichtig
>
> Eine nicht erreichbare Entscheidung ist kein technischer Fehler, sondern ein logischer Fehler.

Im Beispiel 10.6 kann die Entscheidung `A1 > 4` nicht erreicht werden, weil das vorherige und allgemeinere Kriterium `A1 > 3` für die gleichen Werte zutrifft.

> **i** Merke
>
> Es müssen immer die spezielleren Kriterien vor den allgemeineren Kriterien geprüft werden.

Nicht erreichbare Entscheidungen lassen sich durch das Formale prüfen der logischen Ausdrücke leicht erkennen. Dazu werden logischen Ausdrücke und die zugehörigen Wertebereiche für den Wahr- und Falsch-Fall untereinander aufgeschrieben. Ein logischer Ausdruck kann einen Wertebereich nur dann abdecken, wenn dieser eine Teilmenge des Wertebereichs der aktuellen logischen Verzweigung ist.

Tabelle 10.7.: Formale Prüfung der logischen Ausdrücke aus Beispiel 10.6

Rang	logischer Ausdruck	Wahr-Fall	Falsch-Fall
1	`A1 > 5`	`A1 > 5`	`A1 <= 5`
2	`A1 > 3`	`A1 > 3`	`A1 <= 3`
3	`A1 > 4`	`A1 > 4`	`A1 <= 4`
4	`A1 <= 3`	`A1 <= 3`	`A1 > 3`

Weil die Funktion `WENNS()` verwendet wird, ist der Wertebereich für einen logischen Ausdruck durch die Falsch-Fälle der logischen Ausdrücke mit niedrigerem Rang abgedeckt.

Für Rang 3 muss wegen dieser Tabelle der logischen Ausdruck in Formel 10.1 gelten. Dieser Ausdruck kann jedoch nie Wahr ergeben, weil der gleiche Wert in Variable $A1$ nicht kleiner oder gleich 3 und gleichzeitig grösser als 4 sein kann.

$$
\begin{aligned}
& (A1 <= 5) \wedge \quad (A1 <= 3) \wedge (A1 > 4) \\
\Leftrightarrow \quad & (A1 <= 3) \wedge (A1 > 4) \\
\Leftrightarrow \quad & \text{Falsch}
\end{aligned}
\tag{10.1}
$$

10.5.4. ERSTERWERT

Die Funktion `ERSTERWERT()` bildet einen Spezialfall von `WENNS()` ab: Es wird bei allen logischen Ausdrücken ein Vergleich auf Gleichheit des *Suchkriteriums* mit verschiedenen Referenzwerten durchgeführt. In diesem Fall können die logischen Ausdrücke mit `ERSTERWERT()` stark vereinfacht werden. Das lässt sich am ersten Beispiel im Abschnitt `WENNS` veranschaulichen.

Weil alle logischen Ausdrücke die Gleichheit über den gleichen Adressbereich prüfen, kann die Operation mit der Funktion `ERSTERWERT()` vereinfacht wie folgt werden.

```
= ERSTERWERT(A2:A3;
             1; "Eins";
             3; "Drei";
             4; "Vier";
             8; "Acht";
             "Ungültig")
```

10.5.5. XVERWEIS zur Fallunterscheidung

Die Funktion `XVERWEIS()` ist als Excels Version des $\in$-Operators bereits bekannt. Die Funktion kann auch als Alternative zur Funktion `ERSTERWERT()` verwendet werden. In diesem Fall werden als Rückgabematrix keine Wahrheitswerte, sondern die Ergebnisse der Fallunterscheidung angegeben.

Der Vorteil dieser Anwendung ist, dass die Fallunterscheidung nicht mehr auf die Anzahl der Parameterpaare beschränkt ist und die Parameterpaare zum Zeitpunkt der Formelerstellung auch nicht bekannt sein müssen.

Das folgende Beispiel zeigt die Umsetzung des Beispiels aus dem Abschnitt `ERSTERWERT` mit `XVERWEIS()`. Dazu wird zuerst eine Tabelle mit den Vergleichswerten und den zugehörigen Ergebnissen erstellt.

C	D
1	Eins
3	Drei
4	Vier
8	Acht

Diese Referenztabelle stell in Spalte `C` die sog. Suchmatrix und in Spalte `D` die sog. Rückgabematrix bereit. Mit diesen Werten lässt sich die Fallunterscheidung wie folgt abbilden.

```
= XVERWEIS(A2:A3; C2:D5; D2:D5; "Ungültig")
```

Ein weiterer Vorteil von `XVERGLEICH()` gegenüber `ERSTERWERT()` sind Vergleiche mit den Operatoren `=`, `<=` oder `>=`. Diese Vergleiche lassen sich über den fünften Parameter von `XVERWEIS()` konfigurieren. Dabei steht der Wert `0` für die Gleichheit, der Wert `-1` für kleiner oder gleich und der Wert `1` für grösser oder gleich. Die Vergleiche sind immer so organisiert, dass der linke Operand dem Suchkriterium entspricht und der rechte Operand dem Wert in der Suchmatrix. Bei einem Treffer wird der Wert aus der Rückgabematrix zurückgegeben. Gibt es keinen Treffer für den Vergleich wird der Wert aus dem vierten Parameter `wenn_nicht_gefunden` geliefert.

10.5.6. Anwendungshilfe für Fallunterscheidungen

Die Anwendung der verschiedenen Fallunterscheidungsfunktionen hängt von verschiedenen Kriterien ab. Diese sind hier zusammengefasst:

Die Funktion `WENN()` wird immer dann eingesetzt, wenn ein logischer Ausdruck geprüft werden muss und nur die beiden Fälle dieses Ausdrucks unterschieden werden müssen.

Die Funktion `WENNS()` wird immer dann eingesetzt werden, wenn mehrere logische Ausdrücke geprüft werden müssen. Die logischen Ausdrücke können dabei beliebig komplex sein und sich auf verschiedene Daten und Bereiche beziehen.

Die Funktion `ERSTERWERT()` wird immer dann eingesetzt, wenn die Gleichheit des Suchkriteriums mit wenigen Referenzwerten überprüft werden soll. Die Suchkriterien sind für alle Vergleiche identisch.

Die Funktion `XVERWEIS()` wird immer dann eingesetzt, wenn ein Vergleich auf Gleichheit, Kleiner-oder-Gleich oder Grösser-oder-Gleich durchgeführt werden muss. Die Suchkriterien und die Vergleichsoperatoren sind für alle Vergleiche identisch.

Die Funktion `XVERWEIS()` muss anstatt von `ERSTERWERT()` verwendet werden, wenn die Referenzwerte des Vergleichs zum Zeitpunkt der Formelerstellung noch nicht bekannt sind oder leicht änderbar bleiben sollen.

10.6. Filtern

Excel bietet die Funktion `FILTER()` zum Filtern von Daten. Diese Funktion erzeugt einen Ergebnisbereich mit den Werten, die durch den angegebenen logischen Ausdruck ausgewählt wurden.

> **Warnung**
>
> In Excel können in Tabellen und Pivot-Tabellen für einzelne Vektoren Werte für die *Darstellung* "gefiltert" werden. Dabei verwendet Excel nicht die Filter Funktion, sondern *blendet* einzelne Datensätze *aus*. Dadurch können die Ergebnisse nachgereihter Operationen nicht mit den dargestellten Werten zusammenpassen, weil nicht-dargestellte Werte weiterhin Teil der Daten sind und bei Berechnungen weiterhin mitberücksichtigt werden.

Das Ergebnis der Funktion `FILTER()` ist ein dynamischer Bereich mit den ausgewählten Werten. Im Gegensatz zu Tabellen-Filter sind die nicht dargestellten Werte nicht mehr Teil der Daten. Deshalb sind die Ergebnisse der `FILTER()`-Funktion konsistent mit den Ergebnissen der nachgereihten Operationen.

Definition 10.1. Excels `FILTER()`-Funktion wählt aus einem Vektor die Werte aus, für die ein **Auswahlvektor** den Wert `WAHR` oder einen Wert, der dem logischen `WAHR` entspricht.

Beispiel 10.7 (Filtern mit Excel)**.** Gegeben sind die folgende Werte in den Spalten `A` und `B`. Die `Filter()`-Funktion steht an Adresse `D2`.

	A	B	C	D
1	Basel	WAHR		`=FILTER(A1:A5; B1:B5)`
2	Genf	FALSCH		Basel
3	Lugano	FALSCH		Zug
4	Zug	WAHR		Zürich
5	Zürich	WAHR		

Weil die Werte in Spalte `B` vom Datentyp Wahrheitswert sind, kann > dieser Vektor zur Auswahl der Städtenamen in Spalte `A` verwendet werden.

Anstelle eines Vektors mit Wahrheitswerten wird meistens ein Vergleich als zweiter Parameter übergeben. Dieser Vergleich muss einen Vektor erzeugen, der genauso lang ist, wie der Vektor im ersten Parameter. Solche Vektoren werden dynamisch erzeugt, indem ein Vergleich entweder den Vektor selbst oder einen benachbarten Vektor verwendet.

	A	B	C	D
1	Basel	deutsch		`=FILTER(A1:A5; B1:B5 = "deutsch")`
2	Genf	französisch		Basel
3	Lugano	italienisch		Zug
4	Zug	deutsch		Zürich
5	Zürich	deutsch		

10.6.1. Excel Filter und logische Operationen

Weil die `FILTER()`-Funktion immer über Vektoren arbeitet, können die logischen Funktionen nicht verwendet werden, weil sie keine Vektoren erzeugen. Deshalb *muss* der logische Ausdruck des Filters als Boole'sche Arithmetik formuliert werden (Kapitel 10.4).

> **Hinweis**
>
> Die Funktion `NICHT()` ist kein Aggregator und kann mit der `FILTER()`-Funktion kombiniert werden.

Um mit komplexen logischen Ausdrücken in Filtern zu verwenden, müssen wir die logischen Operatoren durch ihre *arithmetische Schreibweise* ersetzen.

Tabelle 10.11.: Beispiel eines komplexen logischen Ausdrucks mit `FILTER()`

	A	B	C	D	E
1	*Name*	*Sprache*	*Einwohner:innen*		*Formel*
2	Basel	deutsch	173863		`=FILTER(A2:A6;(B2:B6="deut > 100000))`
3	Genf	französisch	203856		Basel
4	Lugano	italienisch	62315		Zürich
5	Zug	deutsch	30934		
6	Zürich	deutsch	421878		

10.7. Selektieren

Sehr häufig liegen umfangreiche Daten mit vielen Vektoren vor. Soll sich eine Analyse auf einzelne Vektoren beschränken, dann sollen, analog zum Filtern von Datensätzen, nur diese Vektoren ausgewählt werden.

Definition 10.2. Das Filtern von Vektoren wird als **selektieren** bezeichnet.

Weil die Vektoren einer Stichprobe in der Regel benannt sind, werden Vektoren über ihre Namen *selektiert*.

> **Merke**
>
> Die Vektorennamen einer Stichprobe haben besondere Eigenschaften:
>
> 1. Vektorennamen sind immer von Datentyp `Zeichenkette`.
> 2. Vektorennamen einer Stichprobe bilden einen **Vektor**.
> 3. Die Vektorennamen einer Stichprobe sind **eindeutig**.

Die dritte Eigenschaft ist nicht ganz offensichtlich, denn in einer manuell eingegebenen zwei-dimensionalen Struktur kann eine Überschrift mehrfach verwendet werden. Sobald eine solche Struktur in eine Excel-Tabelle umgewandelt wird, erzwingt Excel eindeutige Vektorennamen.

Aus diesen Eigenschaften folgt, dass die Auswahl von Vektoren durch die Eigenschaften von Zeichenketten unterstützt wird. Wir können zur Auswahl die folgenden Operationen verwenden:

- Identischer Vektorname
- Vektorname beginnt mit einer bestimmten Zeichenkette
- Vektorname endet mit einer bestimmten Zeichenkette
- Vektorname enthält an einer beliebigen Position eine bestimme Zeichenkette

Diese Operationen lassen sich als logische Ausdrücke formulieren, wodurch sich komplexere Selektoren umsetzen lassen.

Der einfachste Weg zum Vektoren adressieren ist die Verwendung des Vektornamens über die Tabellenadressierung. Dabei wird der Vektorname in eckige Klammern gesetzt (s. Abschnitt 6.4.2). Die Tabellenadressierung ist jedoch auf vollständige Namen und auf zusammenhängende Bereiche beschränkt.

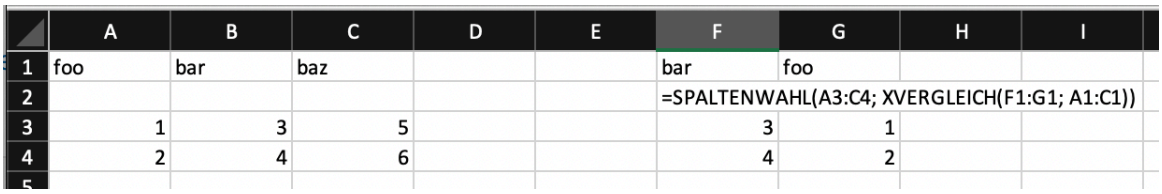
Die Verwendung einer Selektor-Funktion zur Auswahl von Vektoren ist nicht auf Tabellen beschränkt, sondern kann mit beliebigen tabellarischen Strukturen angewendet werden. In Excel wird eine Selektor Funktion durch die Funktionskette eines Filters mit der Funktion `SPALTENWAHL()` erreicht.

Neben der Tabellenadressierung bietet Excel die Funktion `SPALTENWAHL()`, um Vektoren aus einem Bereich auszulesen. Diese Funktion benötigt aber die Position der gewünschten Spalte, denn oft sollen aber Vektoren über Namen oder Namensteile ausgewählt werden. Dafür kommt die Funktion `XVERGLEICH()` zur Anwendung. Mit `XVERGLEICH()` erhalten wir die Position eines gesuchten Werts in einem angegebenen Bereich.

Die Idee hinter dem hier beschriebenen Ansatz ist, dass wir herausfinden, wo unser gewünschter Vektor in der Stichprobe steht. Anschliessend wählen wir alle Werte an dieser Position mit der Funktion `Spaltenwahl()` aus.

> **Hinweis**
>
> Die Funktion `XVERGLEICH()` ähnelt der Funktion `XVERWEIS()` indem wir einen Wert in einem Vektor suchen können. Anstelle eines Referenzwerts aus einem anderen Vektor liefert `XVERGLEICH()` nur die Position des gesuchten Werts zurück. Falls ein Wert mehrfach vorkommt, dann gibt die Funktion nur die *erste* Position zurück.

Der Funktion `XVERGLEICH()` können mehrere Suchwerte übergeben werden, für welche die Positionen bestimmt werden.



	A	B	C	D	E	F	G	H	I
1	foo	bar	baz			bar	foo		
2						=SPALTENWAHL(A3:C4; XVERGLEICH(F1:G1; A1:C1))			
3	1	3	5			3	1		
4	2	4	6			4	2		

Abbildung 10.1.: Beispiel für Vektorenselektion

Der Algorithmus zum Selektieren von Vektoren ist durch die folgenden Schritte definiert:

1. Wir *vektorisieren* nur die Vektornamen auf einem neuen Arbeitsblatt mit der Identitätsfunktion ab Adresse `B1`. Zur Veranschaulichung nenne ich dieses Arbeitsblatt `Stichprobendaten`.
2. Wir *vektorisieren alle* Stichprobendaten mittels der Identitätsfunktion auf dem gleichen Arbeitsblatt ab Adresse `B2`.
3. Auf einem neuen Arbeitsblatt geben wir in der ersten Zeile die Vektorennamen ab Adresse `A1` ein, die wir auswählen möchten. In diesem Beispiel wird angenommen, dass 3 Vektoren ausgewählt werden sollen.
4. Wir wählen die einzelnen Vektoren mit der folgenden Formel an der Adresse A2 aus.

Beispiel 10.8 (Selektion von Vektoren in Excel)**.**

```
=SPALTENWAHL(Stichprobendaten!$B$2#;
             XVERGLEICH(A1:C1; Stichprobendaten!$B$1#))
```

Der Vorteil dieser Strategie ist, dass die Selektion individuelle, nicht-zusammenhängende Vektoren selektieren kann und nicht auf Tabellen beschränkt ist.

10.8. Sortieren

Excel kennt zwei Funktionen zum Sortieren:

- `SORTIEREN()`
- `SORTIERENNACH()`

Die Funktion `SORTIEREN()` sortiert einen Bereich zeilen- oder spaltenweise. Für allgemeine Sortierungen nach mehreren Vektoren stellt Excel die Funktion `SORTIEREN()` zur Verfügung.

> **Hinweis**
>
> **Excels `SORTIERENNACH()`-Funktion** kann einen Bereich zeilen- oder spaltenweise sortieren. Diese Funktion hat vier Parameter:
>
> - `Matrix` - der zu sortierende Bereich, der *keine* Matrix sein muss.

- `Sortierindex` - die Spalten- oder Zeilennummer, nach der sortiert werden soll. Standardmässig wird die erste Spalte bzw. die erste Zeile angenommen.
- `Sortierreihenfolge` - legt die Sortierreihenfolge fest. `1`, um aufsteigend und `-1`, um absteigend zu sortieren.
- `nach_Spalte` - Ein Wahrheitswert, ob die Spalten oder die Zeilen sortiert werden sollen. `WAHR` bedeutet, dass die Spalten (horizontal) sortiert werden sollen. `FALSCH` bedeutet, dass die Zeilen (vertikal) sortiert werden sollen. Standardmässig wird zeilenweise sortiert.

Die Funktion ermöglicht es, mehrere Vektoren auf einmal nach **mehreren** gemeinsamen Kriterien zu sortieren. Dazu müssen zuerst die Sortierkriterien identifiziert werden.

Die Funktion `SORTIERENNACH()` deckt den Spezialfall ab, wenn die Sortierung nach einem externen Kriterium erfolgen soll. Während `SORTIEREN()` erfordert, dass die Sortierindizes im zu sortierenden Bereich enthalten sind, können die Sortierindizes bei `SORTIERENNACH()` an einer beliebigen Stelle in der Arbeitsmappe liegen.

10.8.0.1. Schritt 1: Sortierkriterien festlegen.

Die Sortierkriterien sind durch die Werte im Sortierindex festgelegt, nach denen sortiert werden soll. Der Sortierindex ist ein Vektor mit einem Wert für eine Zeile bzw. Spalte der Sortiermatrix. Entlang der Werte im Sortierindex wird die Sortiermatrix sortiert.

In Excel können die Vektoren mit den Sortierkriterien an einer beliebigen Position in einer Arbeitsmappe liegen. Dabei müssen zwei Bedingungen erfüllt sein:

1. Der Sortierindex und die Sortiermatrix müssen die gleiche Länge haben.
2. Die Sortierindex und die Sortiermatrix müssen die gleiche Orientierung haben.

10.8.0.2. Schritt 2:

Im zweiten Schritt werden die zu sortierenden Vektoren ausgewählt.

In R wird dieser zweite Schritt automatisch auf die vorgegebene Stichprobe angewandt. In Excel können wir zusammenhängende Vektoren als “Matrix” an die `SORTIERENNACH()`-Funktion übergeben. Hängen die Vektoren nicht direkt zusammen, dann müssen mehrere Sortieroperationen mit den gleichen Referenzen auf die Sortierreferenzen durchgeführt werden.

In Excel wird die Sortierrichtung als `Sortierreihenfolge` bezeichnet und als separater Parameter für das jeweilige Sortierkriterium angegeben. Dabei steht `1` für die aufsteigende Sortierung und `-1` für die absteigende Sortierung.

10.9. Rezepte

10.9.1. Fehlerwerte abfangen

Viele Excel-Funktionen geben einen Fehlerwert zurück, falls die Funktion kein gültiges Ergebnis ermitteln kann. Weil sich diese Fehlerwerte in Operationen fortpflanzen, müssen diese Werte durch einen geeigneten regulären Wert ersetzt werden. Das kann mit der folgenden Entscheidung erreicht werden.

```
= WENN(ISTFEHLER(A1); 0; A1)
```

Diese Operation ersetzt alle Fehlerwerte durch den Wert `0` und lässt alle anderen Werte unverändert.

Weil diese Entscheidung sehr oft vorkommt, gibt es die Funktion `WENNFEHLER()`, mit der die gleiche Operation einfacher ausgedrückt werden kann.

```
= WENNFEHLER(A1; 0)
```

10.9.2. Eine Zahl für genau eine Bedingung zurückgeben

Ein häufiger Spezialfall für Unterscheidungen ist die Auswahl von *Zahlen*, die genau **einen** logischen Ausdruck erfüllen. Solche Unterscheidungen geben im `FALSCH`-Fall `0` und im anderen Fall die gesuchte Zahl zurück. In diesem Spezialfall kann der Zielwert ohne Umweg über die `WENN()`-Funktion mit dem logischen Ausdruck multipliziert werden. Der logische Ausdruck liefert `1` für `WAHR` und `0` für `FALSCH`. Die Multiplikation mit `0` liefert immer `0`. Die Multiplikation mit `1` liefert den Zielwert.

Das Beispiel gibt für die folgenden Werte alle Zahlen zurück, die grösser als 10 *und* kleiner als 20 sind.

A
13
5
17
20
12
2
29
11
7
32

Normalerweise würde diese Entscheidung durch die folgende Operation abgebildet:

```
= WENN((A1:A10 > 10) * (A1:A10 < 20); A1:A10; 0)
```

> **Warnung**
>
> Diese spezielle Fallunterscheidung sollte auf Korrektheit überprüft werden, wenn im `WAHR`-Fall der Wert `0` erlaubt ist. In diesem Fall wird der Wert `0` nicht vom logischen Ausdruck unterschieden.

Weil alle Werte Zahlen sind, handelt es sich um den Spezialfall, dass der `WAHR`-Wert eine Zahl und der `FALSCH`-Fall eine 0 ist. Für diesen Fall lässt sich die Formel vereinfachen, indem die gesuchten Werte mit dem logischen Ausdruck multipliziert werden:

```
= A1:A10 * (A1:A10 > 10) * (A1:A10 < 20)
```

Das Ergebnis beider Formeln sind die Werte `{13;0;17;0;12;0;0;11;0;0}`.

Damit dieses Rezept funktioniert, müssen *alle* Teile des logischen Ausdrucks genau die Werte FALSCH oder WAHR bzw. 0 oder 1 zurückgeben. Das ist notwendig, weil nur das neutrale Element die Zielwerte unverändert lässt. Eine direkte Übergabe von Zahlen im logischen Ausdruck verfälscht das Ergebnis, weil nicht mit dem *neutralen Element* gerechnet wird.

Soll für einen logischen Ausdruck nur der Wert 1 oder 0 zurückgegeben werden, dann kann der Rückgabebereich am Anfang der Formel weggelassen werden. Es wird dann nur der logische Ausdruck angegeben. Die Formel `= (A1:A10 > 10) * (A1:A10 < 20)` hat die Werte `{1;0;1;0;1;0;0;1;0;0}` als Ergebnis.

10.9.3. Fehlerwerte vergleichen

Fehlerwerte können nicht direkt mit den Vergleichsoperatoren verglichen werden, weil Excel immer den ersten gefundenen Fehlerwert als Ergebnis einer Operation zurückgibt. Deshalb müssen Fehlerwerte zuerst in normale Werte konvertiert werden. Damit verschiedene Fehlerwerte miteinander verglichen werden können, müssen die verschiedenen Fehlerwerte zuerst in eindeutige Zahlen umgewandelt werden. Das übernimmt die Funktion `FEHLER.TYP()`. Diese Zahlen können anschliessend wie gewohnt weiter verarbeitet werden.

Das folgende Beispiel weist den gegebenen Fehlerwerten eine Fehlermeldung zu:

A
#NV
#WERT!
3

Die folgende Formel liefert die Werte `{"Fehler: #NV"; "Fehlerhafter Wert", "Kein Fehler"}`.

```
= WENNS(FEHLER.TYP(A1:A2) = 7; "Fehler: #NV";
        FEHLER.TYP(A1:A2) = 3; "Fehlerhafter Wert";
        WAHR; "Kein Fehler")
```

Weil alle Vergleiche die Gleichheit überprüfen, kann die Formel mit der Funktion `ERSTERWERT()` vereinfacht werden. Die Formel lautet dann:

```
= ERSTERWERT( WENNFEHLER(FEHLER.TYP(A1:A3); 0);
              3; "Fehlerhafter Wert";
              7; "Fehler: #NV";
              "Kein Fehler")
```

Für diesen Schritt muss die Operation mit der Funktion `WENNFEHLER()` erweitert werden, weil die Funktion `FEHLER.TYP()` einen Fehler ausgibt, wenn der übergebene Wert kein Fehlerwert ist. Weil die Fehlertypen mit Werten grösser `0` durchnummeriert sind, bietet sich für reguläre Werte der Wert `0` an.

10.9.4. Filtern und Summen

Bis Juli 2020 mussten die Funktionen `SUMMEWENN()` oder `SUMMEWENNS()` verwendet werden, um Daten nach Kriterien zu summieren. Diese Funktionen haben allerdings den Nachteil, dass keine echten logischen Ausdrücke verwendet werden können. Seit Juli 2020 steht die `FILTER()`-Funktion zur Verfügung. Dadurch können echte logische Ausdrücke als Filterkriterien eingesetzt werden. Gleichzeitig hat sich die Bedeutung der Funktionen `SUMMEWENN()` und `SUMMEWENNS()`geändert (s. Kapitel 13).

Die folgende Formel sollten wir jetzt mit der Filter-Technik umschreiben.

Beispiel 10.9 (Summenwenn durch Filter-Summe ersetzen)**.** Alt wurde geschrieben:

```
= SUMMEWENN(B2#; "< 0")
```

Neu ist die gleiche Funktion etwas ausführlicher:

```
= SUMME( FILTER(B2#; B2# < 0) )
```

Diese Schreibweise hat den Vorteil, dass die einzelnen Schritte nach dem Prinzip der Problemzerlegung getrennt werden und separat untersucht werden können. Das war in der alten Schreibweise nur indirekt möglich.

Das gleiche Prinzip gilt auch für `ZÄHLENWENN()` und `ZÄHLENWENNS()` anwenden. In diesem Fall wird entweder die Funktion `ANZAHL()` oder die Funktion `ANZAHL2()` als Aggregator eingesetzt.

Der grösste Vorteil ist aber, dass mit dieser Technik **beliebige** EXCEL-Aggregatoren mit gefilterten Daten eingesetzbar sind und nicht mehr auf die vordefinierten Aggregatoren eingeschränkt sind. Mit dem Filtern wird es ausserdem möglich, andere logische Ausdrücke als nur *einen* direkte Vergleich oder, im Fall von `SUMMEWENNS()` oder `ZÄHLENWENNS()`, mit *Und* verknüpfte Vergleiche durchzuführen.

> ⚠ Warnung
>
> Wenn wir komplexe logische Ausdrücke mit Excels `FILTER()`-Funktion verwenden wollen, dann **müssen** wir für die logischen Operatoren die *arithmetische Schreibweise für die logischen Ausdrücke* verwenden!

11. Vektoroperationen

Vektoren sind zusammengesetzte Datenstrukturen, die Werte vom gleichen Datentyp darstellen. Vektoren werden in Excel entweder als *Bereiche* (z.B. `A1:A5`), als *Arrays* (z.B. `B1#`) oder als *Tabellenspalten* (z.B. `Tabelle1[[Spalte1]]`) adressiert.

> ⚠ Wichtig
>
> Excel erzwingt das Datentyp-Kriterium von Vektoren nicht, so dass Vektoren strenggenommen Listen sind.

Falls der gleiche Datentyp in einem Vektor nicht vorausgesetzt werden kann, müssen die Daten für die Verarbeitung vorbereitet werden. Das kann beispielsweise dadurch geschehen, dass der gewünschte Datentyp (oder Wertebereich) durch **filtern** eingeschränkt wird.

> ℹ Merke
>
> Excel kennt keine abstrakten Vektoren, sondern unterscheidet zwischen Zeilen- und Spaltenvektoren.

Definition 11.1. Als **Orientierung** eines Vektors wird die Anordnung der Werte bezeichnet.

Für viele Vektoroperationen existiert deshalb eine Funktion für Spaltenvektoren und eine für Zeilenvektoren.

Um Vektoren von der einen zur anderen Orientierung zu überführen, dient die Funktion `MTRANS()`. Soll eine *bestimmte* Orientierung für beliebige Vektoren erzwungen werden, dann kann die Funktion `ZUSPALTE()` zum Erzwingen eines Spaltenvektors oder die Funktion `ZUZEILE()` zum Erzwingen eines Zeilenvektors verwendet werden.

Beispiel 11.1 (Erzwingen eines Spaltenvektors)**.**

```
= ZUSPALTE(A1:D1)
```

11.1. Vektorlänge

Jeder Vektor hat eine Länge, wobei diese Länge nicht zwingend der Anzahl der Werte entsprechen muss. Deshalb sollte die Länge eines Vektors mit Excel *nicht* mit der Funktion `ANZAHL()` bzw. `ANZAHL2()` bestimmt werden, sondern über die Funktionen `ZEILEN()` und `SPALTEN()`. Der Vorteil dieser beiden Funktionen ist, dass sie die Vektorlänge auch dann korrekt bestimmen, wenn einzelne Werte im Vektor nicht vorhanden sind.

Die Funktionen `ZEILEN()` und `SPALTEN()` geben die Anzahl der Zeilen respektive Spalten für den übergebenen Bereich zurück. Für einen Spaltenvektor gibt die Funktion `ZEILEN()` die Länge des Vektors zurück und die Funktion `SPALTEN()` hat den Wert `1`. Die Funktion `SPALTEN()` gibt die Vektorlänge von Zeilenvektoren zurück. Entsprechend hat in diesem Fall die Funktion `ZEILEN()` immer den Wert `1` als Ergebnis.

> **Merke**
>
> Die kleinste Excel Struktur ist die Zelle. Deshalb werden **keine** Vektoren der Länge `0` unterstützt! Es ist unmöglich, Vektoren der Länge `0` durch Funktionen zu erzeugen!

Aus den beiden Werten der Funktionen `ZEILEN()` und `SPALTEN()` kann die Orientierung eines Vektors mit der Operation `ZEILEN(vektor) - SPALTEN(vektor)` bestimmt werden. Ist das Ergebnis grösser `0` handelt es sich um einen Spaltenvektor und ist er kleiner `0` handelt es sich um einen Zeilenvektor.

> **Praxis**
>
> Zur Feststellung der Orientierung muss in der Regel nur die erste Bedingung geprüft werden.

> **Merke**
>
> Vektoren, auf die mit der Tabellen-Adressierung zugegriffen wird, sind **immer** Spaltenvektoren.

11.2. Konstante Vektoren

Konstante Vektoren sind Vektoren, die als Konstante in einer Formel eingegeben werden und nur konstante Werte enthalten. Ein konstanter Vektor wird durch geschweifte Klammern eingerahmt und kann Zahlen, Zeichenketten, Wahrheits- und Fehlerwerte enthalten (Beispiel 11.2).

> **Merke**
>
> Konstante Vektoren sind **immer** Spaltenvektoren.

Beispiel 11.2 (Konstanter Zahlenvektor der Länge 3)**.**

```
= {3; 2; 1}
```

Konstante Vektoren dürfen keine Referenzen auf Adressen, Bezeichner, Formeln usw. enthalten (Beispiel 11.3). Ungültige konstante Vektoren erzeugen einen `#WERT!`-Fehler

Beispiel 11.3 (Ungültiger konstanter Vektor)**.**

```
= {1; A2; 3}
```

11.3. Wertereferenzierung

Die Referenzierung auf einzelne Werte erfolgt durch die Funktionen `ZEILENWAHL()` für Spaltenvektoren bzw. `SPALTENWAHL()` für Zeilenvektoren. Der erste Parameter ist immer der Vektor. Mit dem zweiten Parameter wird der Index des gewünschten Werts angegeben. Der zweite Parameter kann als Vektor angegeben werden. In diesem Fall werden alle angegebenen Indizes als Vektor zurückgegeben.

> **Merke**
>
> Werden mehrere Indizes angegeben, so werden die Werte in der Reihenfolge der angegebenen Indizes zurückgegeben.

> **Achtung**
>
> Bei der Wertereferenzierung dürfen nur gültige Indizes verwendet werden. Werden mehrere Indizes angegeben und mindestens einer von ihnen ist ungültig, dann werden keine Werte. In solchen Fällen ist das Ergebnis ein `#WERT!`-Fehler.

11.3.1. Rezept: Wertepaare tauschen

Liegen Werte in Wertepaaren vor, dann können die Werte mit der Wertereferenzierung vertauscht werden:

Beispiel 11.4 (Werte eines Spaltenvektors der Länge 2 vertauschen)**.**

```
= ZEILENWAHL(A1:A2; {2; 1})
```

11.4. Sequenzen

Sequenzen werden mithilfe der Generatorfunktion `SEQUENZ()` erzeugt. Die Funktion hat vier Parameter. Die ersten beiden Parameter der Funktion zeigen die Anzahl der Zeilen und Spalten an, über welche sich die Sequenz erstrecken soll. Der dritte Parameter erwartet den Startwert der Sequenz und mit dem letzten Parameter wird die Schrittweite der Sequenz angegeben.

Beispiel 11.5 (Sequenz von geraden Zahlen erzeugen)**.**

```
=SEQUENZ(10; 1; 0; 2)
```

Nur der erste Parameter der Funktion `SEQUENZ()` muss angegeben werden. In diesem Fall wird für alle anderen Parameter der Wert `1` angenommen.

Beispiel 11.6 (Sequenz der Länge 10 als Spaltenvektor erzeugen)**.**

```
=SEQUENZ(10)
```

Zwei spezielle Sequenzen sind der Nullvektor und der Einsvektor. In beiden Vektoren wiederholt sich der Wert an jeder Position. Die zugehörige Sequenz hat also die Schrittweite `0`.

Beispiel 11.7 (Einsvektor der Länge 10 erzeugen)**.**

```
=SEQUENZ(10; 1; 1; 0)
```

Analog zum Einsvektor in Beispiel 11.7 wird der Nullvektor erzeugt, wobei der dritte Parameter den Wert `0` haben muss.

11.4.1. Rezept: Vektor umkehren

Die Wertereihenfolge eines beliebigen Vektors kann durch eine umgekehrte Sequenz umgekehrt werden.

Beispiel 11.8 (Vektorwerte umkehren)**.**

```
= ZEILENWAHL(
    A1:A10;
    SEQUENZ(ZEILEN(A1:A10); 1; ZEILEN(A1:A10); -1)
)
```

11.5. Konkatenation

Die Konkatenation verbindet zwei Vektoren zu einem neuen **Vektor**, wobei die Reihenfolge der Werte erhalten bleibt. Deshalb können in Excel nur Vektoren mit gleicher Orientierung konkateniert werden.

Zum Konkatenieren von Spaltenvektoren dient die Funktion `VSTAPELN()` (für "vertikal stapeln"). Zum Konkatenieren von Zeilenvektoren dient die Funktion `HSTAPELN()` (für "horizontal stapeln").

Die beiden Funktionen `HSTAPELN()` und `VSTAPELN()` prüfen die Orientierung von Vektoren *nicht*! Werden zwei Vektoren mit unterschiedlicher Orientierung mit diesen Funktionen "gestapelt", entsteht eine Matrix, wobei die Vektoren in der Reihenfolge der Argumentübergabe als Zeile oder Spalte gezeigt werden. Die restlichen Positionen der Matrix werden mit dem Fehlerwert `#NV` belegt.

11.6. Transformationen

Das Ergebnis einer Transformationsoperation ist **immer** ein **Vektor** mit der **gleichen Länge** wie der ursprüngliche Vektor. Dazu wird eine Operation für jeden Wert eines Vektors ausgeführt.

Typische Funktionen für Transforamtionen sind beispielsweise:

- `POTENZ()`
- `REST()`
- `NICHT()`
- `ABS()`
- `TEIL()`

> **Merke**
>
> Beliebige Transformationen lassen sich mit `MAP()` umsetzen.

11.6.1. Skalartransformation

Bei Transformationen mit einem Skalar, wird die Transformation für jeden Wert des Vektors und dem Skalar durchgeführt.

Beispiel 11.9 (Verdoppeln von Werten durch Skalartransformation)**.**

```
= 2 * A2:A5
```

Diese Operation entspricht den folgenden Einzeloperationen.

```
= 2 * A2
= 2 * A3
= 2 * A4
= 2 * A5
```

11.6.2. Vektortransformation

Bei der Vektortransformation wird eine Operation paarweise auf die Werte an der gleichen Position von zwei Vektoren angewendet.

> **Merke**
>
> Damit das Ergebnis einer Vektor-Transformation in Excel wieder einen Vektor ergibt, müssen beide Vektoren die *gleiche Orientierung* **und** die *gleiche Länge* haben. D.h. es können nur Zeilenvektoren mit Zeilenvektoren bzw. Spaltenvektoren mit Spaltenvektoren transformiert werden

> **Merke**
>
> Alle Excel-Operatoren können für Vektortransformationen verwendet werden.

Beispiel 11.10 (Werte paarweise addieren)**.**

```
= A1:A5 + B1:B5
```

Diese Operation entspricht den folgenden Teiloperationen:

```
= A1 + B1
= A2 + B2
= A3 + B3
= A4 + B4
= A5 + B5
```

11.6.3. Bedingungen als Transformationen

Spezielle Transformationen sind Bedingungen. Dabei wird ein Vektor erzeugt, dessen Werte durch einen oder mehrere logische Ausdrücke bestimmt werden.

Excel wertet eine Bedingung über Vektoren für jeden Wert des Vektors separat aus. Die Bedingung in Beispiel 11.11 wertet die Bedinung für die Werte an `A2`, `A3`, `A4` und `A5` sowohl im logischen Ausdruck als auch in den Alternativen separat aus.

Beispiel 11.11 (Bedingte Transformation)**.**

```
= LET(vektor; A2:A5; WENN( vektor > 200; vektor; -1))
```

Das entspricht den folgenden Operationen.

```
= WENN( A2 > 200; A2; -1)
= WENN( A3 > 200; A3; -1)
= WENN( A4 > 200; A4; -1)
= WENN( A5 > 200; A5; -1)
```

Die Verwendung von `LET()` dient nur zum Verdeutlichen, dass der gleiche Vektor sowohl im logischen Ausdruck als auch in der Option *Falls Wahr* verwendet wird.

Weil die logischen Funktionen `UND()`, `ODER()` und `XODER()` *Aggregatoren* sind, können sie nicht für Vektortransformationen verwendet werden, sondern **müssen** in Transformationen durch die entsprechenden arithmetischen Operationen ersetzt werden (Beispiel 11.12).

Beispiel 11.12 (Bedingte Transformation mit einem mit Und verknüpften logischen Ausdruck)**.**

```
= LET(vektor; A2:A5; WENN( (vektor > 200)*(vector < 300); vektor; -1))
```

Die Operation entspricht den folgenden Einzeloperationen:

```
= WENN( UND(A2 > 200; A2 < 300); A2; -1)
= WENN( UND(A3 > 200; A3 < 300); A3; -1)
= WENN( UND(A4 > 200; A4 < 300); A4; -1)
= WENN( UND(A5 > 200; A5 < 300); A5; -1)
```

Praxis

Einzeloperationen über Vektoren sollten in der Praxis **immer** durch die entsprechenden Vektoroperationen ersetzt werden. Nur so lassen sich typische Excel-Fehler systematisch vermeiden und die Reaktionszeit von komplexen Arbeitsmappen deutlich beschleunigen.

11.6.4. Rezept: Beliebige Werte wiederholen

Excel fehlt eine Funktion zum Wiederholen von beliebigen Werten. Mithilfe von Sequenzen kann diese Funktionalität leicht nachgebaut werden. Dazu wird eine Sequenz der gewünschten Länge erzeugt und anschliessend für jeden dieser Werte der Wiederholungswert zurückgegeben (Beispiel 11.13). Der Wiederholungswert ist dabei ein Skalar, der für jeden Wert des erzeugten Sequenzvektors zurückgegeben wird.

Beispiel 11.13 (Zeichenkette `Daten` zehn Mal wiederholen)**.**

```
= WENN(SEQUENZ(10); "Daten")
```

11.7. Aggregationen

Beim Aggregieren werden die Werte eines Vektors durch eine Aggregationsoperation zusammengefasst.

Dass Ergebnis von Aggregationen sind Skalare **oder** Vektoren mit einer Länge von *maximal der Länge* des ursprünglichen Vektors.

Häufig verwendete Funktionen zum Aggregieren sind:

- `SUMME()`
- `PRODUKT()`
- `ANZAHL()`
- `MAX()`
- `MIN()`
- `MITTELWERT()`

> **i** Merke
>
> Beliebige Aggregationen lassen sich mit `REDUCE()` umsetzen.

11.7.1. Filtern als Aggregation

Das Filtern ist eine spezielle Aggregation, bei der das Ergebnis nur die Werte enthält, auf welche ein logischer Ausdruck zutrifft. Das Ergebnis einer Filter-Operation kann maximal die gleiche Länge haben, wie der zu filternde Vektor. Die Aggregationsfunktion zum Filtern wertet den logischen Ausdruck zur Auswahl der Werte aus.

11.7.2. Rezept: Laufende Summe

Eine laufende Summe oder *kumulative Summe* liefert für jeden Wert eines Vektors die Summe des Werts mit den Vorgängerwerten. **Diese Operation ist eine spezielle Aggregation.** Excel verfügt keine eigene Funktion für diese Aggregation. Stattdessen kann sie mit Listing 11.1 erzeugt werden.

Listing 11.1 Kumulative Summe

```
= SCAN(0; A2:A100; LAMBDA(accu; wert; accu + wert))
```

Um die laufende Summe herzuleiten, muss sich die Aggregation der normalen Summe als Reduktion mit `REDUCE()` veranschaulicht werden (s. Listing 11.2)

Listing 11.2 Summen-Aggregation

```
= REDUCE(0; A2:A100; LAMBDA(accu; wert; accu + wert))
```

Beim Reduzieren mit `REDUCE()` muss immer ein Initialwert angegeben werden, dem der zu aggregierende Vektor folgt. Abschliessend wird die Aggregationsfunktion angegeben. Beim Reduzieren wird die Aggregationsfunktion für jeden Wert im Vektor nacheinander ausgeführt. Eine Aggregationsfunktion hat zwei Parameter: Der erste Parameter ist das Ergebnis der Aggregationsfunktion für die vorangegangenden Werte, wobei für den ersten Wert der Initialwert verwendet wird. Der zweite Parameter ist der Wert an einer Position im Vektor.

In der Praxis ist die Funktion `SUMME()` dem Reduzieren mit `REDUCE()` immer vorzuziehen. Für die kumulative Summe fehlt aber eine entsprechende Funktion. Für diese Aggregation kann die Funktion `SCAN()` anstelle von `REDUCE()` verwendet werden. `SCAN()` erzeugt einen Vektor mit den Teilergebnissen einer Aggregationsfunktion. Dieser aggregierte Vektor hat die gleiche Länge wie der ursprüngliche Vektor.

11.7.3. Kombinierte Transformation und Aggregation

Sehr häufig werden Vektortransformationen und Aggregationen kombiniert. Zur Veranschaulichung dient hier die Subtraktion von zwei Werten in einem Zahlenvektor. Dazu wird ein Vektor der Länge `2` angenommen. Der erste Wert in diesem Vektor soll von dessen zweiten Wert abgezogen werden. Excel hat allerdings keine Funktion zur Subtraktion, sondern nur die Funktion `SUMME()`. Für die Subtraktion müssen die Werte so angepasst werden, dass diese Werte summiert werden können. Dazu wird eine Vektortransformation in Form einer Multiplikation mit dem konstanten Vektor `{-1; 1}` durchgeführt (Listing 11.3).

Nun ist sichergestellt, das der erste Wert vom zweiten Wert abgezogen wird. Hier gilt zu beachten, dass die Subtraktion unabhängig von den Werten definiert wird. Diese Vorbereitung ermöglicht es, den transformierten Vektor als Parameter für die Summenfunktion zu verwenden. Die vollständige Lösung ist in diesem Fall die folgende Excel Formel:

Listing 11.3 Vorbereitende Vektortransformation

```
= A1:A2 * {-1; 1}
```

Listing 11.4 Subtraktion als kombinierte Vektortransformation und Aggregation

```
= SUMME(A1:A2 * {-1; 1})
```

11.8. Zählen

Beim Zählen werden zählbare Einheiten von nicht-zählbaren Einheiten getrennt und die Anzahl der zählbaren Einheiten bestimmt. Die einfachste Form des Zählens ist das Bestimmen der Länge eines Vektors. Dabei wird die Anzahl der Werte des Vektors *gezählt*.

Alle Zählen-Operationen bestehen entsprechend immer aus zwei Schritten:

- Einer Transformation zum Identifizieren der zählbaren Einheiten
- Einer Aggregation zum eigentlichen Zählen.

Die Funktionen `ANZAHL()` zählt nur Werte vom Datentyp Zahl. Wahrheitswerte, Zeichenketten, Fehler und leere Zellen werden von dieser Funktion ignoriert. Die Funktion `ANZAHL2()` zählt die nicht leeren Zellen in einem Bereich unabhängig vom Datentyp. Diese Funktion zählt auch Fehlerwerte mit.

11.8.1. Zählen durch Summieren

Beim Zählen durch Summieren werden die zählbaren Elemente eines Vektors mithilfe eines logischen Ausdrucks **transformiert**. Daraus ergibt sich ein Vektor aus Wahrheitswerten. Dieser Vektor wird in einem zweiten Schritt mit der `SUMME()`-Funktion **aggregiert**.

Auf diese Weise lassen sich gezielte Häufigkeiten bestimmen.

> **i** Merke
>
> Vektoren, die nur die Werte `0` und `1` bzw. `FALSCH` und `WAHR` beinhalten, legen das Zählen durch Summieren nahe.

Beispiel 11.14 (Zählen durch Summieren)**.**

```
= SUMME(data_ab[Punkte] < 100)
```

11.8.2. Zählen durch Filtern

Beim Zählen durch Filtern werden die zählbaren Einheiten durch eine Filter-Aggregation isoliert. Die Anzahl der zählbaren Einheiten entspricht immer der Länge des gefilterten Vektors.

Beispiel 11.15 (Zählen durch Filtern)**.**

```
= ANZAHL2(FILTER(data_ab[Punkte] < 100))
```

> **Merke**
>
> Zählen durch Filtern sollte nur dann eingesetzt werden, wenn der Ergebnisvektor für weitere Operationen benötigt wird. Ist dies nicht der Fall, ist das Zählen durch Summieren effizienter.

11.8.3. Zählen durch Nummerieren

Beim Zählen durch Nummerieren werden die zu zählenden Werte eines Vektors nummeriert. Anschliessend wird der Maximal-Wert der Nummerierung bestimmt. Diese Technik nutzt aus, dass beim Nummerieren im Hintergrund eine Sequenz mit der Schrittweite `1` verwendet wird. Das letzte zählbare Element hat erhält so automatisch die Anzahl aller Element als Nummer.

Beispiel 11.16 (Zählen durch Nummerieren)**.**

```
= MAX(SEQUENZ(ZEILEN(daten_ab[[Punkte]])))
```

> **Merke**
>
> Zählen durch Nummerieren sollte nur verwendet werden, wenn die Nummerierung entweder bereits vorliegt oder für weitere Operationen benötigt wird.

12. Matrix-Operationen

> **Merke**
>
> Alle Bereiche mit mehr als einer Zelle werden unabhängig von den vorliegenden Datentypen in Excel als *Matrix* bezeichnet.

Wie auch bei Vektoren übernimmt Excel keine Überprüfung der Bedinungen, ob tatsächlich eine Matrix in einem Bereich vorliegt.

12.1. Matrizen erstellen

In Excel können Matrizen auf drei Arten erstellt werden.

1. Durch direkte Eingabe
2. Durch das Überführen aus der Vektorform
3. Durch eine Operation mit einem Zeilen und einem Spaltenvektors
4. Erzeugen durch eine Generatorfunktion

Bei der direkten Eingabe werden die Werte direkt in einen Bereich in Excel eingegeben. Am einfachsten wird dazu der Bereich der Matrix mit der Maus markiert und anschliessend werden die Werte eingegeben und mit der Eingabetaste oder der Tabulatortaste bestätigt. Excel bewegt die Markierung für die aktive Zelle im markierten Bereich automatisch an die nächste freie Position: Wird mit der Eingabetaste bestätigt, dann bewegt Excel die Eingabemarkierung *spaltensweise* durch den markierten Bereich. Wird die Eingabe mit der Tabulatortaste bestätigt, dann bewegt Excel die Eingabemarkierung *zeilenweise* durch den markierten Bereich.

Liegt eine *Vektorform* einer Matrix vor, dann kann sie mit den Funktionen `ZEILENUMBRUCH()` oder `SPALTENUMBRUCH()` in eine Matrix mit einer festen Spalten- bzw. Zeilenzahl erzeugt werden. Neben dem Ausgangsvektor erwarten diese Funktionen die Länge einer Zeile bzw. einer Spalte. Damit ein Vektor in eine Matrix konvertiert werden kann, muss die Länge des Vektors durch die Zeilen- bzw. Spaltenlänge teilbar sein. Erfüllt ein Vektor dieses Kriterium nicht, dann füllt Excel die Positionen bis zur vollständigen Zeile bzw. Spalte mit dem Wert `#NV` auf.

Beispiel 12.1 (Matrix aus einer Sequenz erzeugen)**.**

```
=ZEILENUMBRUCH(SEQUENZ(12); 3)
```

Jede Transformation mit einem Spalten- und einem Zeilenvektor erzeugt in Excel eine Matrix (s. Abschnitt **??**). Dabei werden die Werte über Kreuz ausgewertet, so dass jede Position der Matrix einer paarweisen Operation entspricht. Die dimensionalität der so erzeugten Matrix ist immer die Zeilenanzahl des Spaltenvektors und die Spaltenanzahl des Zeilenvektors.

> **Merke**
>
> Operationen über einem Spalten- und einem Zeilenvektor entsprechen dem äusseren Produkt. Das äussere Produkt ist in Excel nur für Vektoren definiert!

Mit den beiden Generatorfunktionen `SEQUENZ()` und `ZUFALLSMATRIX()` lassen sich Werte in matrizenartigen Bereichen erzeugen. Dazu wird die gewünschte Zeilen- und Spaltenanzahl als die ersten beiden Argumente übergeben. Die beiden Funktionen füllen anschliessend den angegebenen Bereich mit Werten.

12.2. Matrixdimensionen

Die Dimensionen einer Matrix ergibt sich aus den Funktionen `ZEILEN()` und `SPALTEN()`. Beide Werte müssen getrennt bestimmt werden.

12.3. Matrixwerte referenzieren

Die Werte einer Matrix werden mit der Funktion `INDEX()` referenziert. Diese Funktion hat drei Parameter.

1. Die Matrix
2. Der Zeilenindex des gesuchten Werts
3. Der Spaltenindex der gesuchten Werts.

Für den Zeilen- und den Spaltenindex können Vektoren angegeben werden, um mehrere Werte auf einmal zu referenzieren.

Beispiel 12.2 (Werte der Matrix Diagonalen referenzieren)**.**

```
= INDEX(A1:D4; SEQUENZ(4); SEQUENZ(4))
```

Weil diese Operation die gleiche Sequenz zwei Mal erzeugt, lässt sie sich mit `LET()` vereinfachen.

```
=LET(
    diagonalenIndex; SEQUENZ(4);
    INDEX(A1:D4; diagonalenIndex; diagonalenIndex)
)
```

Um einzelne Spalten bzw. Zeilen zu referezieren sollten die Funktionen `SPALTENWAHL()` bzw. `ZEILENWAHL()` verwendet werden, weil sie nur einen Indexwert benötigen.

12.4. Vektorform

Die Vektorform einer Matrix wird in Excel durch die Funktionen `ZUSPALTE()` bzw. `ZUZEILE()` erzeugt. Die beiden Funktionen unterscheiden sich nur durch die Orientierung des Ergebnisvektors. Die zeilenweise Vektorform wird standardmässig von beiden Funktionen erzeugt. Um eine spaltenweise Vektorform zu erhalten, kann der dritte Parameter `scan_by_column` auf `WAHR` gesetzt werden.

> **Praxis**
>
> Enthält eine Matrix fehlende Werte, die durch den Fehlerwert `#NV` markiert sind, lassen sich diese in der Vektorform direkt entfernen, indem der zweite Parameter auf `2` (Fehlerwerte ignorieren) bzw. `3` (Leere Zellen und Fehlerwerte ignorieren) gesetzt wird. Dabei muss beachtet werden, sich die Matrix anschliessend nicht mehr aus dem Vektor reproduzieren lässt.

12.5. Matrizen vergleichen

Zwei Matrizen sind genau dann gleich, wenn sie gleiche Anzahl von Zeilen und Spalten sowie an jedem Index den gleichen Wert haben. Intuitiv bietet sich der paarweise Vergleich der Indizes an. Existieren nicht alle Indizes in der jeweils anderen Matrix, füllt Excel die fehlenden Indizes mit einem Fehler auf. Das führt dazu, dass die logische Verknüpfung in diesen Fällen einen Fehler zurückgibt. Deshalb ist es notwendig, dass zusätzlich gprüft wird, ob das Ergebnis dieser Operation einen Fehlerwert ergibt (Beispiel 12.3).

Beispiel 12.3 (Formel zum Vergleich von zwei Matrizen)**.**

```
= WENNFEHLER(
    UND(
        Matrix1!A1# = Matrix2!A1#
    );
    FALSCH
)
```

Ein zweiter praktischer Vergleich ist das Überprüfen einer quadratischen Matrix. Dabei wir nur eine Matrix geprüft. Weil die Werte in diesem Fall keine Bedeutung haben ist die Formel sehr einfach (Beispiel 12.4).

Beispiel 12.4 (Überprüfen einer quadratischen Matrix)**.**

```
= ZEILEN(Matrix1!a1#) = SPALTEN(Matrix1!A1#);
```

12.6. Matrix Operationen

12.6.1. Matrizen Transponieren

Matrizen werden mit der Funktion `MTRANS()` transponiert. Dabei werden die Zeilen- und Spaltenindizes getauscht.

Beispiel 12.5 (Eine Matrix transponieren)**.**

```
= MTRANS(Matrix1!A1#)
```

12.6.2. Skalar- und Vektoroperationen

Alle Skalaroperationen werden von Excel wie erwartet ausgeführt: die Operation wird mit dem Skalar für jeden Wert der Matrix ausgeführt.

Bei Vektoroperationen ist die Orientierung des Vektors für das Verhalten der Operation bestimmend.

- Spaltenvektoren werden spaltenweise mit der Matrix verknüpft. Dabei werden die Werte mit dem jeweils gleichen Zeilenindex zusammengefasst.
- Zeilenvektoren werden zeilenweise mit der Matrix verknüpft. Dabei werden die Werte mit dem jeweils gleichen Spaltenindex zusammengefasst.

Beispiel 12.6 (Orientierung bei der Verknüpfung eines Vektors und einer Matrix.)**.**

```
= SEQUENZ(4) + SEQUENZ(4; 4)
```

Obwohl die folgende Operation sehr ähnlich aussieht, hat sie ein anderes Ergebnis.

```
= SEQUENZ(1; 4) + SEQUENZ(4; 4)
```

In beiden Fällen gilt, dass der Vektor genauso viele Zeilen bzw. Spalten haben muss, wie die verknüpfte Matrix, so dass für die Verknüpfungsoperation beide Operanden vorhanden sind. Fehlt einer der Operanden, dann ist die Verknüpfung für diesen Index *nicht definiert*, so dass Excel für diese Positionen einen `#NV`-Fehler erzeugt.

Definition 12.1. Excel führt eine **unvollständige** bzw. **partielle Vektoroperation** durch, wenn nicht alle Indizes entweder im Vektor oder in der Matrix vorhanden sind.

Beispiel 12.7 (Orientierung bei der Verknüpfung eines Vektors und einer Matrix.)**.**

```
= SEQUENZ(5) + SEQUENZ(4; 4)
```

> ⚠ Achtung
>
> Für Skalar- und Vektoroperationen sind nur Operatoren und Funktionen zulässig, die einen **Skalar** zum Ergebnis haben. Grundsätzlich sind alle Arithmetischen und Vergleichsoperatoren sowie der Textketten-Operator `&` für Skalar- und Vektoroperationen geeignet.
> **Manche** Operationen erzeugen einen Fehler, andere geben bei überschüssigen Werten den jeweils ersten Index aus.

12.6.3. Kreuzprodukt/Matrixprodukt

Das Kreuzprodukt (oder *innere Matrixprodukt*) ist die Entsprechung zur Multiplikation von zwei Skalaren. Diese Operation wird in Excel durch die Funktion `MMULT()` ausgeführt. Das Kreuzprodukt ist nicht transitiv. Deshalb ist die Reihenfolge der Operanden wichtig, denn für das Kreuzprodukt muss die Spaltenlänge der linken Matrix mit der Zeilenlänge der rechten Matrix übereinstimmen. Ist diese Bedinung nicht gegeben, dann erzeugt Excel einen `#WERT!`-Fehler.

Das Ergebnis des Kreuzprodukts ist eine Matrix mit der Zeilenlänge der linken und der Spaltenlänge der rechten Matrix.

Weil das Kreuzprodukt die Summe mit der Multiplikation verbindet, ergeben sich interessante Vereinfachungen, wenn die Werte einer Matrix nur die Werte `0` und `1` sind (s. Kapitel 12.8.1 und Kapitel 12.8.3).

Beispiel 12.8 (Die Funktion `SUMMENPRODUKT()` als Kreuzprodukt)**.** Die Funktion `SUMMENPRODUKT()` zwischen zwei Vektoren lässt sich auch als Kreuzprodukt formulieren:

```
= MMULT(SEQUENZ(1;4); SEQUENZ(4))
```

Das Ergebnis dieser Operation ist eine 1×1-Matrix (bzw. ein Skalar), weil die linke Matrix nur eine Zeile und die rechte Matrix nur eine Spalte hat.

12.6.4. Das äussere Produkt

In Excel ist das äussere Produkt nur für Vektoren (bzw. `1,n`- oder `m,1`-Matrizen) definiert. Dabei müssen die Vektoren unterschiedliche Orientierungen haben. Das Ergebnis ist dann eine $m \times n$-Matrix, wobei m die Länge des Spaltenvektors und n die Länge des Zeilenvektors ist.

Beispiel 12.9 (Multiplikationstabelle für das kleine Einmaleins erstellen)**.**

```
= SEQUENZ(10) * SEQUENZ(1;10)
```

Das äusser Produkt ist nicht auf die Multiplikation beschränkt, sondern für beliebige Operationen, wobei die Operation einen *Skalar* zum Ergebnis haben muss. Dieser Wert wird am Kreuzungsindex der Ergebnismatrix eingesetzt.

Diese Eigenschaft wird dazu ausgenutzt, um Matrizen mit speziellen Eigenschaften zu konstruieren (Abschnitt 12.8.2).

> **Achtung**
>
> Werden beliebige Matrizen auf die gleiche Weise verknüpft, führt Excel eine paarweise Verknüpfung aus, die für alle nicht exisiterenden Index-Paare den Fehlerwert `#NV` erzeugt (Abschnitt 12.6.5).

Wird ein Vergleich als Operator für das äussere Produkt verwendet, dann ist das Ergebnis der Operation ein Wahrheitswert. Um diesen Wahrheitswert in eine Zahl umzuwandeln, muss eine arithmetische Operation nachgereiht werden. Hierzu ist die Multiplikation mit `1` oder die Addition mit `0` üblich.

Beispiel 12.10 (Äusseres Produkt zum Erzeugen einer Dreiecksmatrix)**.**

```
= 1 * (SEQUENZ(4) <= SEQUENZ(1;4))
```

12.6.5. Paarweise Operationen

Werden beliebige Matrizen miteinander verknüpft, dann führt Excel eine paarweise Verknüpfung aus. D.h. ein Operator oder eine Funktion wird paarweise für die Werte mit den gleichen Indizes ausgeführt. Dabei handelt es sich um eine Verallgemeinerung der Vektoroperation (Abschnitt 12.6.2), wobei wie auch bei den Vektoroperationen die Operation nur ausgeführt werden kann, wenn für beide Operanden in der jeweiligen Matrix ein Wert existiert.

Bei paarweisen Operationen mit zwei Matrizen müssen beide Matrizen die gleichen Dimensionen haben. Ist diese Bedingung nicht gegeben, führt Excel eine *partielle Verknüpfung* (Definition 12.1) aus.

> **Merke**
>
> Zwischen den paarweisen Operationen und dem äusseren Produkt besteht kein Zusammenhang. Für Vektoren bzw. $1 \times n$-Matrizen führt Excel immer das äussere Produkt aus, für alle anderen Matrizen paarweise Operationen.

12.7. Mathematische Hilfsoperationen

Neben den bereits vorgestellten Operationen verfügt Excel über drei weitere Funktionen für ***quadratische Matrizen***.

- `MEINHEIT()`
- `MINV()`
- `MDET()`

12.7.1. Identitätsmatrix erzeugen

Die Funktion `MEINHEIT()` erzeugt eine Identitätsmatrix mit der angegebenen Zeilen-/Spaltenanzahl. Die Identitätsmatrix ist eine quadratische Diagonalmatrix mit den Wert `1` entlang der Diagonalen und dem Wert `0` an allen anderen Positionen.

Beispiel 12.11 (Einheitsmatrix der Grösse 5)**.**

```
= MEINHEIT(5)
```

12.7.2. Determinante

Eine wichtige Kennzahl für quadratische Matrizen ist die Determinante. Sie wird mit der Funktion `MDET()` berechnet.

Beispiel 12.12 (Bestimmung der Determinante einer Matrix)**.**

```
=MDET(SPALTENUMBRUCH(SEQUENZ))
```

12.7.3. Inverse Matrix

Für viele Berechnungen ist die Inverse A^{-1} einer quadratischen Matrix A notwendig. Falls eine Matrix invertierbar ist, gibt die Funktion `MINV()` die Inverse zurück. Diese Funktion löst die Gleichung $A \times A^{-1} = I$ und bestimmt die Inverse Matrix A^{-1}.

Beispiel 12.13 (Inverse einer 4×4-Matrix)**.**

```
=MINV(
    ZEILENUMBRUCH(
        SEQUENZ(16);
        4
    )
)
```

Ist die Matrix nicht invertierbar, dann gibt die Funktion den Fehler `#ZAHL!` zurück. In diesem Fall ist die Determinante gleich `0`.

Beispiel 12.14 (Nicht-invertiertbare 4×4-Matrix)**.**

```
=MINV(
    SPALTENUMBRUCH(
        SEQUENZ(16);
        4
    )
)
```

Wird der Funktion keine quadratische Matrix übergeben, dann hat diese Funktion einen `#WERT!` zum Ergebnis.

12.8. Rezepte

Die folgenden Rezepte Nutzen das Kreuzprodukt und das äussere Produkt.

12.8.1. Zeilen- und Spaltensummen

Es sollen die Summen für jede Zeile einer Matrix (oder Tabelle) berechnet werden und das Ergebnis soll ein Vektor sein.

> **Hinweis**
>
> Für die Excel-Formeln wird angenommen, dass eine Matrix an der Adresse `M1` vektorisiert vorliegt. In der Praxis muss den entsprechenden Parametern die korrekte Adresse der Matrix übergeben werden.

12.8.1.1. Lösung

```
= MMULT(M1#; SEQUENZ(SPALTEN(M1#);1;1;0))
```

> **Praxis**
>
> Diese Operation ist bei der Arbeit mit Excel wichtig, weil diese Operation in Excel effizienter ist, als zeilenweise Aggregationen über vektorisierte oder nicht-verktorisierte Daten.

12.8.1.2. Erklärung

Um Zeilensummen für eine Matrix zu erstellen, nutzen wir die Eigenschaften des Kreuzprodukts aus und multiplizieren eine Matrix mit dem Einsvektor. Das Ergebnis dieser Operation ist ein Vektor, der in jeder Zeile die Summe der Werte aus der entsprechenden Zeile der Matrix enthält.

Das Kreuzprodukt ist nur für Matrizen definiert, wenn die erste (linke) Matrix A gleich viele Spalten hat, wie die Zeilenanzahl der zweiten (rechten) Matrix B. Die Ergebnismatrix des Kreuzprodukts hat gleich viele Zeilen, wie die erste Matrix und gleich viele Spalten wie die zweite. Für jede Position c_{ij} der Ergebnismatrix C gilt dann die folgende Formel:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Um die Zeilensummen einer Matrix A zu erhalten, müssen wir die Matrix B entsprechend konstruieren.

1. Matrix B darf nur eine Spalte haben, damit die Ergebnismatrix nur eine Spalte hat.
2. Matrix B darf die Werte in Matrix A nicht verändern.
3. Matrix B muss gleich viele Zeilen haben, wie Matrix A Spalten hat.

Diese Anforderungen 1 und 2 werden durch den Einsvektor mit `n`-Zeilen erfüllt. Dabei nutzen wir aus, dass ein Spaltenvektor das gleiche wie eine Matrix mit einer Spalte ist.

Die Anforderungen 3 können wir über die Spaltenzahl der Matrix bestimmen und erstellen dann einen Einsvektor mit entsprechender Zeilenanzahl.

Daraus ergibt sich für den Einsvektor v_e und der Matrix an Adresse `M1` die Formel:

```
=SEQUENZ(SPALTEN(M1#); 1; 1; 0)
```

Weil der Einsvektor an jeder Position das neutrale Element der Multiplikation enthält, vereinfacht sich die Formel für das Kreuzprodukt stark:

$$c_{i1} = \sum_{k=1}^{n} a_{ik}$$

Bei dieser Formel ist zu beachten, dass die 1 in c_{i1} für die einzige Spalte des Einsvektors steht und i für die Zeilen in der Matrix A. Daraus wird deutlich, dass wir nun die Zeilensummen für die Matrix A erhalten.

Der Einsvektor muss bei dieser Operation als die rechte Matrix eingesetzt werden, sodass gilt:

$$A \times v_e$$

Als Excel-Formel wird das wie folgt ausgedrückt:

```
=MMULT(M1#; SEQUENZ(SPALTEN(M1#); 1; 1; 0))
```

Aus dieser Lösung lässt sich auch die Spaltensumme ableiten: Für die Spaltensumme muss das neutrale Element der Multiplikation in die erste Matrix wandern. Dazu wird ein *Zeilenvektor* erstellt und mit diesem das Kreuzprodukt mit der Matrix gebildet. Weil in diesem Fall die Spaltensumme gebildet werden soll, kommen die Werte für die Spalten aus der rechten Matrix. Deshalb müssen wir das Kreuzprodukt wie folgt bilden:

$$v_e \times B$$

Daraus ergibt sich die folgende Excel-Formel:

```
= MMULT(SEQUENZ(1; ZEILEN(M1#); 1; 0); M1#)
```

> **Praxis**
>
> Bei der Zeilen- und Spaltensumme handelt es sich um ein sogenanntes *Muster*, dass an verschiedenen Stellen verwendet werden kann. Ein *Muster* ist im Kern eine *Funktion*, die *parametrisiert* werden kann.

Wir können dieses Muster mit der Funktion `LET()` (zum Exkurs über die `LET()`-Funktion) so anpassen, dass wir den Bereich für unsere Matrix nur einmal anpassen müssen.

Daraus ergeben sich die folgenden Excel-Formeln:

- Zeilensumme: `= LET(Matrix; M1#; MMULT(Matrix; SEQUENZ(SPALTEN(Matrix); 1; 1; 0)))`
- Spaltensumme: `= LET(Matrix; M1#; MMULT(SEQUENZ(1; ZEILEN(Matrix); 1; 0); Matrix))`

12.8.2. Dreieckmatrizen erzeugen

Es soll eine quadratische Dreiecksmatrix mit n-Zeilen erzeugt werden.

12.8.2.1. Lösung

Die Anzahl der Zeilen befindet sich in diesem Beispiel an Adresse `A1`.

Obere Dreiecksmatrix

```
= 1 * (SEQUENZ(A#) <= SEQUENZ(1; A1#))
```

Untere Dreiecksmatrix

```
= 1 * (SEQUENZ(A#) >= SEQUENZ(1; A1#))
```

12.8.2.2. Erklärung

Bei einer Dreiecksmatrix steht unterhalb bzw. oberhalb der Diagonalen der Wert 1 und sonst der Wert `0`. Gelegentlich (wie in der obigen Lösung) wird die Diagonale zu den mit `1` belegten Positionen hinzugezählt.

Die Lösung nutzt aus, dass Excel bei unterschiedlich orientierten Vektoren das äussere Produkt bildet und so eine Matrix erzeugt. Die Basis für diese Operation bilden zwei Sequenzen, die über kreuz verglichen werden.

Die Sequenzen können dabei als Nummerierungen der Matrixpositionen verstanden werden. Der Vergleich ergibt jeweils bis zur Diagonalen `FALSCH` und ab der Diagonalen `WAHR`, weil die Nummerierung der Positionen ab der Diagonalen *gleich* ist.

Die Multiplikation mit `1` ist notwendig, damit die Matrix Zahlen und keine Wahrheitswerte enthält.

12.8.3. Kumulative Summe

Es soll die kumulative Summe eines Vektors gebildet werden.

12.8.3.1. Lösung

Für Spaltenvektoren:

```
=LET(
    VLAENGE; ZEILEN(A1#);
    MMULT(
        A1#;
        1 * (SEQUENZ(VLAENGE) >= SEQUENZ(1; VLAENGE))
    )
)
```

Für Zeilenvektoren:

```
=LET(
    HLAENGE; SPALTEN(A1#);
    MMULT(
        1 * (SEQUENZ(HLAENGE) <= SEQUENZ(1; HLAENGE));
        A1#
    )
)
```

12.8.3.2. Erklärung

Die kumulative Summe dreht die Logik der Zeilen- bzw. Spaltensumme um. In diesem Fall wird eine Matrix erzeugt, die für jede Position im Vektor anzeigt, welche Vektorwerte summiert werden sollen. Entsprechend darf diese Matrix nur die Werte `0` oder `1` enthalten, da die Werte im Ausgangsvektor nicht verändert werden dürfen.

Für die kumulative Summe müssen alle Werte bis und mit der aktuellen Position summiert werden. Eine Dreiecksmatrix leistet genau diesen Zweck: Für Spaltenvektoren zeigt die untere Dreickesmatrix zeilenweise an, welche Positionen für die kumulative Summe berücksichtigt werden müssen.

Weil die Diagonale der Dreiecksmatrix die gleiche Länge hat wie der Ausgangsvektor, ist sichergestellt, dass der Ergebnisvektor die gleiche Länge wie der Ausgangsvektor hat.

Dieser Trick funktioniert nur, wenn die das neutrale Element der Multiplikation in der Dreiecksmatrix verwendet wird.

12.8.4. Co-Occurence Matrizen

Es sollen die gemeinsam auftretenden Werte in zwei **diskretskalierten Vektoren** gezählt werden und in einer *Kreuztabelle* gegenübergestellt werden.

12.8.4.1. Lösung

Für Spaltenvektoren:

```
=MMULT(
    1 * (EINDEUTIG(A1#) = MTRANS(A1#));
    1 * (MTRANS(EINDEUTIG(B1#)) = B1#)
)
```

12.8.4.2. Erklärung

Im ersten Schritt werden für jeden Vektor separat die Positionen der eindeutigen Werte in einer Matrix markiert. Die eindeutigen Werte sind die Werte vorkommenden Werte des Wertebereichs. Diese Werte werden mit der Funktion `EINDEUTIG()` bestimmt. Das Ergebnis des ersten Schritts ist eine Matrix, die nur die Werte `1` und `0` enthält. Der Wert `1` ist an der Position gesetzt, an der ein Wert des Wertebeichs im Ausgangsvektor gefunden wurde.Dazu dient die folgende Formel:

```
= 1 * (EINDEUTIG(A1#) = MTRANS(A1#))
```

Diese Formel lässt sich auf beide Ausgangsvektoren anwenden. Dadurch ergeben zwei Matrizen mit gleich vielen Spalten, die nur die Werte `1` und `0` enthalten.

Damit die gemeinsamen Vorkommnisse gezählt werden, wird das Kreuzprodukt verwendet. Dabei wird ausgenutzt, dass das Kreuzprodukt die Produkte der Zeilen- und Spaltenwerte summiert. Weil beide Matrizen nur die Werte `0` und `1` enthalten, entspricht diese Operation dem *Zählen durch Summieren.*

Diese Operation zählt nur das gemeinsame Auftreten, weil in beiden Matrizen sichergestellt ist, dass in jeder Spalte genau eine `1` vorkommt, werden nur die Werte an den gleichen Positionen in den Ausgangsvektoren gezählt.

Damit dieser Trick funktioniert, muss eine der beiden Matrizen transponiert werden, bevor das Kreuzprodukt gebildet werden kann. Daraus ergibt sich die folgende Formel.

```
= MMULT(
    1 * (EINDEUTIG(A1#) = MTRANS(A1#));
    MTRANS(1 * (EINDEUTIG(B1#) = MTRANS(B1#)))
)
```

An dieser Formel ist wenig elegant, dass für den zweiten Parameter zwei Mal transponiert wird. Dieser Teilterm lässt sich dadurch vereinfachen, indem die zweite Transponierung bereits bei der Erzeugung der Matrix vorweggenommen wird, denn es gilt:

```
MTRANS(1 * (EINDEUTIG(B1#) = MTRANS(B1#)))
```

ist equivalent mit

```
1 * (MTRANS(EINDEUTIG(B1#)) = B1#)
```

Wird diese Vereinfachung in die obige Formel eingesetzt, dann ergibt sich die Formel der Lösung.

Das Ergebnis ist eine sog. Kontingenztabelle, die die Häufigkeiten der gemeinsam auftretenden Werte anzeigt.

13. Indizieren und Gruppieren

13.1. Indizieren

Es werden drei Arten von Indizes unterschieden:

1. Der **Primärindex**, mit dem ein einzelner Datensatz eindeutig *identifiziert* werden kann.
2. **Fremdschlüssel** sind Sekundärindizes für *Querverweise* auf eine zweite Datenstruktur (eine sog. *Indextabelle* oder engl. *Lookup-Table*).
3. **Gruppenindizes** sind Sekundärindizes zur Identifikation von Datensätzen mit *gemeinsamen Eigenschaften*.

Weil ein Index Werte über einen Datensatz enthält, gehört ein Index zum jeweiligen Datensatz und wird über einen *Indexvektor* in einer Stichprobe abgebildet.

13.1.1. Hashing eines Primärindex

In Excel muss zum Durchnummerieren die `SEQUENZ()`-Funktion verwendet werden. Das erreichen wir mit der folgenden Operation: `=SEQUENZ(ZEILEN(DatenBereich))`, wobei `DatenBereich` eine Excel-Adresse sein muss. Weil mit Excel keine leeren Tabellen existieren, führt diese Sequenz in Excel zu keinen Fehlern. Wegen dieser Eigenschaft muss ein entsprechender Bereich mindestens einen Umfang von einem Datensatz haben.

> **Hinweis**
>
> Diese Eigenschaft gilt auch für Tabellen, die nur aus Überschriften bestehen. Damit die Formel zum Nummerieren eingegeben werden kann, muss mindestens ein Datensatz existeren, weil sonst keine Zelle für die Formel *in* der Tabelle existiert.

13.1.2. Hashing eines Sekundärindex

Ein Sekundärindex kennzeichnet mehrere Datensätze mit gleichen Eigenschaften. Die einfachste Hashing-Funktion für Sekundärindizes ist `WENNS()`. Diese Funktion stellt über die Bedingungen sicher, dass alle Hashes einen Datensatz die Merkmale eindeutig auswerten.

Soll ein Sekundärindex über Zahlen gebildet werden, werden die Ganzzahldivision und der Modulo-Operator häufig als Hashing-Funktion eingesetzt. Die Ganzzahldivision fasst aufeinanderfolgende Werte zu einem Wert zusammen. Der Modulo-Operator weist aufeinanderfolgenden Werte unterschiedlichen Werten zu.

> **Praxis**
>
> Excels Funktion für die Ganzzahldivision ist eigentlich die Funktion `QUOTIENT()`. Diese Funktion kann leider nur mit einzelnen Werten und nicht mit Vektoren oder Matrizen verwendet werden. Deshalb ist die Funktion `QUOTIENT()` für die Praxis nur bedingt tauglich. Stattdessen muss die Ganzzahldivision in Excel wie folgt *simuliert* werden.

```
= GANZZAHL( A1:A9 / 3 )
```

Beispiel 13.1 (Tage und Wochen bestimmen)**.** Gegegeben ist eine Sequenz in `A1` Startend bei `1`. Diese Sequenz bildet die *Werktage* eines Projekts ab. Der erste Tag ist ein Montag. Feiertage werden nicht berücksichtigt. Aus dieser Sequenz sollen Wochentage und Wochen ermittelt werden.

Die Wochennummern im Projekt wird mithilfe der Ganzzahldivision ermittelt.

```
= GANZZAHL(A1# / 5)
```

Die Wochentage werden mit dem Modulo-Operator bestimmt.

```
=  REST(A1#; 5)
```

13.1.3. Indizieren mit einer Referenztabelle

Oft müssen Werte in andere Werte übersetzt werden, wobei die Kodierung keiner mathematischen Logik folgt. In diesem Fall werden sog. Referenz- oder Kodierungstabellen für das Indizieren verwendet. Sehr oft werden diese Tabellen unabhängig von der Datenerhebung erstellt.

Eine Referenztabelle hat mindestens zwei Spalten, wobei per Konvention die erste Spalte die möglichen Werte vor dem Indizieren. Diese Werte können in den Daten auftreten. Alle folgenden Spalten enthalten Kodierungen des ursprünglichen Werts.

> **Warnung**
>
> Enthält eine Referenztabelle mehr als eine Kodierung, dann müssen die Werte spaltenweise kodiert werden.

Beispiel 13.2 (Likert-Skala kodieren)**.** Eine Liker-Skala ist ein Messinstrument zur Meinungs- oder Empfindungserhebung. Eine Likert Skala wird in der Regel zwischen zwei Extremfeststellungen erfasst, die später als Zahlen kodiert ausgewertet werden.

Solche Skalen erfordern eine Referenztabelle, weil die Ordnung der Werte sich nicht aus der alphabetischen Reihenfolge der Zeichenketten ergibt. Für die Auswertung müssen die Zeichenketten in Zahlen kodiert werden. Es ist üblich die Kodierung von Likert-Skalen in Referenztabellen zu dokumentieren. Die folgende Tabelle zeigt eine fünfstufige Liker-Skala mit den entsprechenden Zahlenwerten für die Auswertung.

Wert	Zahl
Trifft gar nicht zu	-2
Trifft eher nicht zu	1
Unentschieden	0
Trifft eher zu	1
Trifft voll und ganz zu	2

Die Kodierung erfolgt mit der Funktion `XVERWEIS()`.

```
= XVERWEIS(Tabelle1[Aussage_Likert];
           KodierungLikert[Wert];
           KodierungLikert[Zahl])
```

13.2. Gruppieren

Das Gruppieren war in Excel auf die folgenden Funktionen beschränkt:

- `ZÄHLENWENN()`
- `ZÄHLENWENNS()`
- `SUMMEWENN()`
- `SUMMEWENNS()`
- `MINWENNS()`
- `MAXWENNS()`
- `MITTELWERTWENN()`
- `MITTELWERTWENNS()`

> **!** Wichtig
>
> Die Funktionen `ZÄHLENWENN()`, `ZÄHLENWENNS()`, `SUMMEWENN()` und `SUMMEWENNS()` wurden bis 2019 zum Erzeugen von bedingten Aggregationen verwendet. Mit der Einführung dynamischer Felder und der `FILTER()`-Funktion lassen sich bedingte Aggregationen einfacher und flexibler durch Funktionsverkettungen von FILTER() mit einer

beliebigen Aggregationsfunktion erreichen.
Seit dieser Version hat sich die Bedeutung der Funktionen geändert. Die Funktionen werden neu als Gruppierungsfunktionen verwendet.
Mit der Einführung der `GRUPPIERENNACH()` Funktion wurden gruppierte Operation flexibler gestaltet (siehe dazu Kapitel 13.3).

Merke

Wenn für eine Funktion eine WENN und eine WENNS-Variante existieren, dann sollte für *gruppierte Aggregationen* **immer** die WENNS-Variante verwendet werden.

Achtung

Die beiden Funktionen `WENN()` und `WENNS()` sind **keine** Gruppierungsfunktionen, sondern Unterscheidungen. Trotz ähnlicher Namen, dürfen sie nicht mit den Gruppierungsfunktionen verwechselt werden.

Die WENNS-Varianten sind für alle Aggregationen bezüglich ihrer Parameter konsistent, so dass die Syntax leichter zu merken ist. Die beiden Funktionen `MINWENNS()` und `MAXWENNS()` veranschaulichen diese Empfehlung, weil für diese Funktionen keine WENN-Variante existiert.

Eine Gruppierungsoperation erfolgt in drei Schritten:

1. Erzeugen der Gruppierungsindizes.
2. Ermitteln der Gruppierungen
3. Ausführen der Gruppierungsfunktion für alle Gruppierungen

Beispiel 13.3 zeigt diese drei Schritte. Gegeben sind Werte zwischen 1 und 25, die in drei Bereiche organisiert werden sollen. Es soll die Anzahl der Werte in den einzelnen Bereichen ermittelt werden. Die Werte liegen in einer Tabellenstruktur vor.

Der Gruppierungsindex ist ein Merkmal und sollte gemeinsam mit den Daten gespeichert und versioniert werden. Dieser Sekundärindex strukturiert die Daten in einer Tabelle oder einem Bereich.

Beispiel 13.3 (Gruppiertes Zählen)**.** Die Reihenfolge der Schritte muss eingehalten werden!

Schritt 1: Sekundärindex erstellen (in `A1`)

```
= WENNS( Tabelle1[Werte] > 18; "gross";
         Tabelle1[Werte] > 10; "mittel";
         WAHR; "klein" )
```

Dieser Schritt kann entfallen, wenn ein Sekundärindex bereits vorhanden ist.

Schritt 2: Gruppierungen ermitteln (in C1)

```
= EINDEUTIG(A1#)
```

Schritt 3: Zählen (in D1)

```
= ZÄHLENWENNS( A1#; C1# )
```

Beim Zählen wird der Gruppierungsindex direkt verwendet. Die anderen Gruppierungsfunktionen haben einen zusätzlichen Parameter für den Vektor über den die Operation ausgeführt werden soll.

13.2.1. Mehrere Sekundärindizes

> **!** Wichtig
>
> Die WENNS-Varianten können mehrere Sekundärindizes verknüpfen. Dabei werden die vorkommenden *Permutationen* der Hash-Werte als Index verwendet. Alle Kriterienvektoren müssen dafür **gleich lang** sein und die Permutationen abbilden.

Um alle *vorkommenden* Permutationen mehrerer Sekundärindizes zu ermitteln, müssen die Vektoren einen Bereich bilden. Für diese Bereich ermittelt die Funktion `EINDEUTIG()` dann jede vorkommende Zeile. Dieser Trick funktioniert jedoch nur wenn alle Sekundärindizes nebeneinander stehen, was oft nicht der Fall ist. Die Funktion `HSTAPELN()` löst dieses Problem, weil sie mehrere Vektoren zu einem Bereich zusammenfügen kann. Beispiel 13.4 zeigt die Vorgehensweise für drei Vektoren.

Beispiel 13.4 (Vorkommende Permutationen mehrerer Sekundärindizes ermitteln in `A1`)**.**

```
= LET(bereich; HSTAPELN(
                Tabelle1[Name];
                Tabelle1[Durchlauf];
                Tabelle1[Gruppe] );
      EINDEUTIG(bereich)
)
```

Diese Permutationstabelle kann nun mit den Gruppierungsfunktionen verwendet werden. Beispiel 13.5 zeigt die Verwendeung der Gruppierungen aus Beispiel 13.4.

Beispiel 13.5 (Gruppieren mit mehreren Indizes)**.**

```
= MITTELWERTWENNS(
    Tabelle1[Zeit];
    Tabelle1[Name];       SPALTENWAHL(A1#; 1);
    Tabelle1[Durchlauf]; SPALTENWAHL(A1#; 2);
    Tabelle1[Gruppe];     SPALTENWAHL(A1#; 3)
)
```

13.3. Die GRUPPIERENNACH()-Funktion

Mit der Einführung der Funktion GRUPPIERENNACH() erlaubt Excel beliebige Gruppierungen, wodurch gruppierte Aggregationen weiter vereinfacht wurden. Die GRUPPIERENNACH()-Funktion fasst den Schritt der Eindeutigen Werte im Indexvektor und die nachfolgende gruppierte Aggregation zusammen. Das Ergebnis liefert immer die eindeutigen Werte des Gruppierungsindex zusammen mit den jeweiligen Aggregationen als Ergebnis zurück.

> **Praxis**
>
> Die Syntax der Funktion GRUPPIERENNACH() ist deutlich einfacher als die älteren Gruppierungsfunktionen.

Der erste Parameter die GRUPPIERENNACH()-Funktion ist immer der Indexvektor, der zweite Parameter ist immer der Wertevektor, über welchen aggregiert werden soll. Der dritte Parameter ist eine Aggregationsfunktion, die als *Funktionsreferenz* übergeben werden muss. Das bedeutet, dass hinter dem Funktionsnamen ***keine*** Klammern und auch keine Parameterliste stehen darf.

Die GRUPPIERENNACH() Funktion fügt unter den gruppierten Ergebnissen normalerweise eine Total-Zeile für ein Gesamtergebnis ein. Diese zusätzliche Zeile kann durch den Wert 0 als fünften Parameter unterdrückt werden. Der vierte Parameter übernimmt die Überschiften in das Ergebnis, falls diese ebenfalls übergeben wurden. Dieser Parameter hat in vielen Fällen keine Bedeutung und wird deshalb meist ausgelassen.

Tabelle 13.2 stellt die Syntax der älteren und neueren Gruppierungsfunktionen gegenüber.

Tabelle 13.2.: Gegenüberstellung alte und neue Gruppierungsstrategien

Alte Syntax	Neue Syntax mit GRUPPIERENNACH()
=ZÄHLENWENNS(daten[index]; EINDEUTIG(daten[index]))	GRUPPIERENNACH(daten[index]; daten[werte]; ZEILEN;; 0)
=SUMMEWENNS(daten[werte]; daten[index]; EINDEUTIG(daten[index]))	GRUPPIERENNACH(daten[index]; daten[werte]; SUMME;; 0)
=MINWENNS(daten[werte]; daten[index]; EINDEUTIG(daten[index]))	GRUPPIERENNACH(daten[index]; daten[werte]; MIN;; 0)

Alte Syntax	Neue Syntax mit `GRUPPIERENNACH()`
`=MAXWENNS(daten[werte]; daten[index]; EINDEUTIG(daten[index]))`	`GRUPPIERENNACH(daten[index]; daten[werte]; MAX;; 0)`
`=MITTELWERTWENNS(daten[werte]; daten[index]; EINDEUTIG(daten[index]))`	`GRUPPIERENNACH(daten[index]; daten[werte]; MITTELWERT;; 0)`

> **Merke**
>
> Damit eine Gruppierung durchgeführt werden kann, muss auch mit `GRUPPIERENNACH()` ein oder mehrere Indexvektoren erzeugt werden, falls noch keine Indexvektoren für die Daten existieren.

Die Gruppierungsstrategie mit `GRUPPIERENNACH()` hat den Vorteil, dass auch nicht vorgegebene Aggregationen durchgeführt werden können. Das Beispiel 13.6 zeigt die gruppierte Aggregation des Medians, die ohne `GRUPPIERENNACH()` nur umständlich umgesetzt werden kann.

Beispiel 13.6 (Gruppierter Median)**.**

```
= GRUPPIERENNACH(data_ab[Angebot]; data_ab[Interesse]; MEDIAN;; 0)
```

> **Merke**
>
> Anders als andere Excel-Funktionen führen Fehlerwerte nicht zu einem einzelnen Fehlerwert im Ergebnis. Im Index-Vektor werden Fehlerwerte als eigene Gruppe behandelt. Erst innerhalb einer Gruppe führen Fehlerwerte im Wertevektor zu einem Fehler.

13.3.1. Gruppieren über mehrere Indizes

Um über mehrere Indizes gleichzeitig zu indizieren, müssen diese in den Daten einen Bereich bilden. Alle Indexvektoren müssen also unmittelbar nebeneinander liegen. Anschliessend wird dieser Bereich als Index der Funktion `GRUPPIERENNACH()` übergeben. In diesem Fall werden alle vorhandenen Permutationen der Indexvektoren als Index für die Gruppierungen verwendet.

Das Beispiel 13.7 zeigt die Anwendung über mehrere aufeinanderfolgende Indizes in einer Tabelle.

Beispiel 13.7 (Gruppieren über mehrere Indizes)**.**

```
= GRUPPIERENNACH(data_ab[[Interesse]:[Bedeutung]]; data_ab[Punkte]; SUMME;; 0)
```

14. Daten formen

> ⚠ Work in Progress
>
> Dieses Kapitel wird aktuell überarbeitet und erweitert. Fehlerhinweise und Korrekturen sind willkommen.

Speziell für die Datenvisualisierung, erfordert Excel das Umformen von Daten. Dabei kann es notwendig werden, die Daten von der Normalform in eine Breitform umzuformen. Auch für andere Operationen kann es sinnvoll sein, Daten eine definierte Form zu geben. Zu diesen Operationen gehören z.B. gruppierte Aggregationen.

14.1. Die `PIVOTMIT()`-Funktion

Seit 2024 enthält Excel die Funktion `PIVOTMIT()`. Diese Funktion erlaubt das Überführen von Daten in eine Kreuztabelle. Diese Funktion hat vier verpflichtende Parameter:

1. `row_fields` (Zeilenwerte)
2. `col_fields` (Spaltenwerte)
3. `values` (Aggregationswerte)
4. `function` (Aggregationsfunktion)

Diese Funktion ähnelt der den klassischen Pivot-Tabellen stark: Die beiden ersten Parameter liefern Gruppierungswerte, über die eine Zusammenfassung erstellt werden soll. Mit dem Parameter `values` werden die Werte festgelegt, die zusammengefasst werden sollen. Der vierte Parameter legt die Aggregationsfunktion fest, dabei kann jede Aggregrationsfunktion verwendet werden.

=PIVOTMIT(data_ab[Interesse];data_ab[Bedeutung]; data_ab[Punkte];MITTELWERT)							
	1	2	3	4	5	6	Gesamt
1	204.46	225.44	261.88				231.624
2	169.34	222.345714	177.4175	220.795	306.76		201.794167
3	209.45	232.71875	208.349	209.93625	222.22	230.434286	218.698947
4	244.33	198.9175	152.511667	198.932353	211.177143	249.586667	200.211538
5		259.785	301.05	164.816667	193.5825	180.125	195.706842
6	177.2325	189.79	174.95		258.72	128.67	190.957
7	139.04						139.04
Gesamt	190.70375	222.434583	193.487857	198.097714	214.777	219.286154	205.051691

Abbildung 14.1.: Ergebnis der `PIVOTMIT()`-Funktion mit `MITTELWERT()` als Aggregator

Die Standardausgabe generiert Zeilen- und Spaltensummen. Diese Summen sind für eine Breitform unhantlich und lassen sich durch die optionalen 6. und 8. Parameter unterdrücken, wenn diese Parameter auf den Wert 0 gesetzt werden. Die Zeilensummen werden durch den 6. Parameter und die Spaltensummen werden durch den 8. Parameter beeinflusst.

=PIVOTMIT(data_ab[Interesse];data_ab[Bedeutung]; data_ab[Punkte];MITTELWERT;;0;;0)						
	1	2	3	4	5	6
1	204.46	225.44	261.88			
2	169.34	222.345714	177.4175	220.795	306.76	
3	209.45	232.71875	208.349	209.93625	222.22	230.434286
4	244.33	198.9175	152.511667	198.932353	211.177143	249.586667
5		259.785	301.05	164.816667	193.5825	180.125
6	177.2325	189.79	174.95		258.72	128.67
7	139.04					

Abbildung 14.2.: Ergebnis der `PIVOTMIT()`-Funktion ohne Zeilen- und Spaltensummen

14.1.1. `PIVOTMIT()` und fehlende Werte

Tritt ein Zeilen-und-Spaltenwertepaar nicht auf, kann `PIVOTMIT()` keine Werte aggregieren. Anders als andere Excel-Funktionen werden solche fehlenden Werte nicht durch den Wert 0 ersetzt, sondern durch eine leere Zeichenkette. Dieses Verhalten lässt sich nicht anpassen, sondern muss durch eine zusätzliche Operation korrigiert werden.

```
= LET(breitform; PIVOTMIT(
                    data_ab[Interesse];
                    data_ab[Bedeutung];
                    data_ab[Punkte];
```

```
                    MITTELWERT;;
                    0;;
                    0);
      WENN(breitform = ""; 0; breitform)
)
```

> **!** Achtung
>
> Der Wert 0 darf **nicht** eingesetzt werden, wenn eine Breitform erzeugt werden soll, weil sonst Werte erzeugt werden, die ursprünglich nicht vorhanden waren.

14.1.2. Eine Breitform mit `PIVOTMIT()` erzeugen.

Wird als Aggregatorfunktion die Funktion `SUMME()` verwendet und jedes Wertepaar, dass sich aus den ersten beiden Parametern ergibt, dann ist diese Aggregation mit der Identifitätsfunktion funktional gleich.

```
= PIVOTMIT(Zeilenindex; Spaltenindex; Werte; SUMME;; 0;; 0)
```

> **!** Achtung
>
> Die Funktion `PIVOTMIT()` erfordert eine *Aggregationsfunktion*. Diese Funktion stellt sicher, dass mehrfach vorkommende Wertepaare zu einem eindeutigen Ergebnis führen. Dieses Verhalten führt dazu, dass die `PIVOTMIT()`-Funktion **nicht** immer **umkehrbar** ist, sobald die vorkommenden Wertepaare nicht eindeutig sind.

> **i** Merke
>
> Eine Langform kann nur in eine Breitform umgeformt werden, wenn alle Wertepaare der Indexvektoren **eindeutig** sind. Eine solche Langform liegt dann in ihrer **Normalform** *für die beiden Indexvektoren* vor.

14.1.3. Breitform ohne Überschriften erhalten

Weil die Funktion `PIVOTMIT()` auch Zeilen- und Spaltenüberschriften erzeugt, lassen sich die einzelnen Vektoren nicht direkt mit `SPALTENWAHL()` extrahieren. Leider lassen sich die Spaltenüberschriften nicht unterbinden.

Damit die Überschriften nicht in den Daten vorliegen, müssen diese nach der Transformation entfernt werden:

```
= LET(breitform; PIVOTMIT(
                    Zeilenindex;
                    Spaltenindex;
                    Werte;
                    SUMME;;
                    0;;
                    0);
      zeilenwerte; ZEILEN(breitform) - 1;
      spaltenwerte; SPALTEN(breitform);
      wertezeilen; SEQUENZ(zeilenwerte; 1; 2);
      wertespalten; SEQUENZ(1;spaltenwerte; 1);
      INDEX(breitform; wertezeilen; wertespalten)
)
```

Um auch die Werte des Zeilenindex zu entfernen, muss die Spaltenanzahl ebenfalls verringert werden.

```
= LET(breitform; PIVOTMIT(
                    Zeilenindex;
                    Spaltenindex;
                    Werte;
                    SUMME;;
                    0;;
                    0);
      zeilenwerte; ZEILEN(breitform) - 1;
      spaltenwerte; SPALTEN(breitform) - 1;
      wertezeilen; SEQUENZ(zeilenwerte; 1; 2);
      wertespalten; SEQUENZ(1; spaltenwerte; 2);
      INDEX(breitform; wertezeilen; wertespalten)
)
```

> **!** Achtung
>
> Intuitiv bietet sich die Funktion `BEREICH.VERSCHIEBEN()` ebenfalls an. Diese Funktion erfordert jedoch eine Zellbezug, so dass diese Funktion nicht in Operationen mit `LET()` und `LAMBDA()` eingesetzt werden kann.

14.2. Vektoren auf ihre Normalform prüfen

Zwei Indexvektoren liegen genau dann in ihrer Normalform vor, wenn alle Wertepaare (bzw. Tupel) eindeutig sind, d.h. genau einmal vorkommen. Damit eine Breitform der Daten für diese Indexvektoren erzeugt werden kann, muss gelegentlich geprüft werden, ob diese auch tatsächlich in ihrer Normalform vorliegen.

Gegeben seien die folgenden Daten im Bereich A1:C6:

Index 1	Index 2	Werte
a	e	1
b	f	2
c	g	3
d	e	4
a	e	5

Die Indexvektoren liegen in `Index 1` und `Index 2`, also in `A2:A6` bzw. `B2:B6`.

Dazu werden alle eindeutigen Wertepaare mit `EINDEUTIG()` erzeugt. In diesem Beispiel wird diese Operation in Zelle `D2` durchgeführt.

```
= EINDEUTIG(A2:B6)
```

Der Bereich `D2#` enthält jetzt alle eindeutigen Wertepaare der Indexvektoren.

Die Normalform ist also dann gegeben, wenn jedes eindeutige Wertepaar höchstens einmal auftritt. Das liesse sich mit der folgenden Formel überprüfen, die als Ergebnis `WAHR` hat, wenn die Indexvektoren in ihrer Normalform vorliegen.

```
= 1 = MAX(NACHZEILE(
    D2#;
    LAMBDA(zeile;
        ZEILEN(
            FILTER(
                A2:A6;
                (SPALTENWAHL(zeile;1)=A2:A6) *
                (SPALTENWAHL(zeile;2)=B2:B6)
            )
        )
    )
))
```

Diese Prüfung ist recht komplex und lässt sich stark vereinfachen, wenn die Voraussetzungen für die Normalform berücksichtigt werden. Dazu wird die Eigenschaft der Normalform ausgenutzt: Weil eine Normalform erfordert, dass die Wertepaare der Indexvektoren höchstens einmal vorkommen dürfen, muss das Ergebnis der Funktion `EINDEUTIG()` die gleiche Länge haben, wie die ursprünglichen Vektoren. Weil beide Indexvektoren aus der Definition des Datenrahmens die gleiche Länge haben müssen, lässt sich die Normalform mit der folgenden Operation prüfen.

```
= ZEILEN(A2:A6) = ZEILEN(D2#)
```

Diese Prüfung lässt sich in eine Operation zusammenfassen.

```
= ZEILEN(A2:A6) = ZEILEN(EINDEUTIG(A2:B6))
```

Ergibt diese Operation `WAHR`, dann kann die Breitform ohne Informationsverlust erzeugt werden.

14.3. Die `INDEX()`-Funktion

Die Funktion `PIVOTMIT()` hat keine unmittelbare Umkehrfunktion. Mithilfe der `INDEX()`-Funktion lässt sich eine Umkehrfunktion konstruieren. Diese Funktion wurde bereits im letzten Abschnitt zum entfernen der Werte der Indexvektoren eingesetzt.

Die `INDEX()`-Funktion ist eine sog. Referenzfunktion, mit der auf Werte mittels ihrer Position in einem Vektor oder einer Matrix zugegriffen werden kann. Die `INDEX()`-Funktion erwartet drei Parameter:

1. Die zu referenzierende Matrix
2. einen gültigen Zeilenindex, und
3. einen gültigen Spaltenindex.

Der Zeilen- und Spaltenindex sind Ganzzahlen, die sich auf die Position der Werte beziehen. Auf diese Weise lassen sich Werte aus einem Vektor oder einer Matrix extrahieren. Die beiden Indexwerte beziehen sich dabei relativ zur linken oberen Zelle des übergebenen Bereichs, wobei die Referenzzelle einen Zeilen- und Spaltenindex von 1 hat. Aus dieser Spezifikation ergibt sich, dass die `INDEX()`-Funktion keine negativen Indexwerte akzeptiert.

14.3.1. `INDEX()` mit Vektoren als Indexparameter

Die INDEX()-Funktion erlaubt Vektoren für den Zeilen- und Spaltenindex. Auf diese Weise lassen sich mehrere Werte auf einmal referenzieren. Liegen die Vektoren in gleicher Orientierung vor, dann bildet die Funktion Indexpaare. Die Operation `= INDEX(A1:C5; {1; 2; 4}, {1; 2; 2})` wählt beispielsweise die Werte der Adressen `A1`, `B2` und `B4` aus.

Liegen die Indizes in verschiedenen Orientierungen vor, dann bildet die `INDEX()`-Funktion eine Indexmatrix mit allen Permutationen der angegebenen Indizes. Die Werte werden dann als Matrix angeordnet. Z.B. die Operation `=INDEX(A1:D4; {1;2;4}; ZUZEILE({1;2;2}))` wählt die Werte an `A1`, `A2`, `A4`, `B1`, `B2`, `B4` sowie nocheinmal `B1`, `B2` und `B4` aus. Diese Dopplung entsteht durch die doppelte Angabe des Werts `2` im Spaltenindex. Auf den ersten Blick erscheint dieses Verhalten unpraktisch, im folgenden Abschnitt wird genau diese Eigenschaft ausgenutzt.

Die Funktion INDEX() behält die Orientierung der Parametervektoren bei. Diese Eigenschaft lässt sich ausnutzen, um eine Verkettung mit `MTRANS()` zu vermeiden. Die Operation `=MTRANS(INDEX(A1:D4; {1;2;4}; ZUZEILE({1;2;2})))` ist mit der Operation `=INDEX(A1:D4; ZUZEILE({1;2;4}); {1;2;2})` funktional gleich.

14.3.2. Die Umkehrung von `PIVOTMIT()` konstruieren

Beim Umformen von der Breitform in die Langform müssen nicht nur die Werte als Vektor dargestellt werden, sondern auch die beiden Indexvektoren wiederhergestellt werden. Entsprechend reicht der naive Aufruf von `ZUSPALTE()` nicht.

Analog zur Eingabe von PIVOTMIT() soll die Umkehrung drei Vektoren als Ergebnis erzeugen. Damit die Operation eine Umkehrung von PIVOTMIT() ist, ist der Ausgangswert der Umkehrung eine Matrix mit Zeilen- und Spaltenüberschriften.

> **Hinweis**
>
> Im folgenden wird der Bezeichner `datenbereich` in Excel-Formeln verwendet, dieser muss gegen die konkrete Adresse der in Breitform vorliegenden Daten ersetzt werden. Das Gleiche gilt für andere Bezeichner, die sich auf vorangegangene Berechnungen beziehen.

Für die Umkehrung wird ausgenutzt, dass die Langform die höchstens soviele Datensätze wie Werte in der Ausgangsstruktur haben kann. Es gilt also $n_z \cdot n_s$. Weil die Datenstruktur beschriftet ist, müssen die Zeilen- und Spaltenumfänge der Datenstruktur um 1 reduziert werden. Daraus ergibt die folgenden Excel-Operation für den Umfang der Ergebnisdatenstruktur.

```
= (ZEILEN(datenbereich) - 1) * (SPALTEN(datenbereich) - 1)
```

Als nächstes werden drei Indexvektoren als Spaltenvektoren erzeugt:

1. Der Indexvektor für die Zielposition im Ergebnis.
2. Der Indexvektor für die jeweilige Zeile
3. Der Indexvektor für die jeweilige Spalte

Der Indexvektor für die Zielposition ergibt sich direkt aus dem gerade bestimmten Umfang. Diesen Vektor bezeichnen wir hier als *Zielindex*.

```
= SEQUENZ(zielumfang)
```

oder explizit

```
= SEQUENZ((ZEILEN(datenbereich) - 1) * (SPALTEN(datenbereich) - 1))
```

Dieser Indexvektor enthält alle Positionen der Werte im Zielvektor. Aus diesem Vektor lassen sich die beiden Indexvektoren bestimmen.

```
// Zeilenindex
= GANZZAHL((zielindex - 1)/(SPALTEN(datenbereich) - 1)) + 2

// Spaltenindex
= REST(zielindex - 1; SPALTEN(datenbereich) - 1) + 2
```

In beiden Termen ist die Addition mit 2 notwending, weil zuerst der Indexbeginn bei 1 korrigiert (`+ 1`) und anschliessend die Überschriftsspalte bzw. -Zeile im Index berücksichtigt (`+ 1`) werden muss. Diese beiden Korrekturen korrespondieren mit den `- 1` in gleichen Term.

Liegt eine Tabelle in einer Breitform ohne Beschriftungen vor, dann müssen die Indizes ohne Korrekturen berechnet werden:

```
// Zielindex
= SEQUENZ(ZEILEN(datenbereich) * SPALTEN(datenbereich))

// Zeilenindex
= GANZZAHL((zielindex - 1)/(SPALTEN(datenbereich))) + 1

// Spaltenindex
= REST(zielindex - 1; SPALTEN(datenbereich)) + 1
```

Mit dem Zeilen- und dem Spaltenindex lassen sich nun die drei Ausgangsvektoren aus dem Datenbereich rekonstruieren:

```
// Zeilenwerte
= INDEX(datenbereich; zeilenindex; 1)

// Spaltenwerte
= INDEX(datenbereich; 1; spaltenindex)

// Werte
= INDEX(datenbereich; zeilenindex; spaltenindex)
```

Weil `PIVOTMIT()` auch Werte für nicht vorhandene Indexpaare erzeugt, müssen diese Indexpaare abschliessend aus dem Ergebnis entfernt werden:

```
// Für die Zeilenwerte
= FILTER(zeilenwerte; werte <> "")

// Für die Spaltenwerte
= FILTER(Spaltenwerte; werte <> "")

// Für die Werte
= FILTER(werte; werte <> "")
```

> **Tip**
>
> Falls die drei Vektoren unmittelbar nebeneinander stehen, können die Operationen auch zusammengefasst werden:
>
> `= FILTER(zeilenwerte:werte; werte <> "")`

Diese Operationen lassen sich in der folgenden Excel-Formel zusammenfassen:

```
= LET(
    zeilenumfang; ZEILEN(datenbereich) - 1;
    spaltenumfang; SPALTEN(datenbereich) - 1;
    zielumfang; zeilenumfang * spaltenumfang;
    zielindex; SEQUENZ(zielumfang) - 1;
    zeilenindex; GANZZAHL(zielindex / spaltenumfang) + 2;
    spaltenindex; REST(zielindex; spaltenumfang) + 2;
    werte; INDEX(datenbereich; zeilenindex; spaltenindex);
    zwischenergebnis; HSTAPELN(
        INDEX(datenbereich; zeilenindex; 1);
        INDEX(datenbereich; 1; spaltenindex);
        werte
    );
    FILTER(zwischenergebnis; werte <> "")
)
```

> **Hinweis**
>
> Die Funktion `HSTAPELN()` wird in dieser Formel dazu verwendet, eine zusammenhängende Datenstruktur für die abschliessende `FILTER()`-Operation zu erzeugen.

14.4. `SPALTENWAHL()` und `ZEILENWAHL()`

Die beiden Funktionen `SPALTENWAHL()` und `ZEILENWAHL()`, sind Vereinfachungen der `INDEX()`-Funktion.

Die Funktion `SPALTENWAHL()` wählt ganze Spalten aus. Beispielsweise wählt der Aufruf von `SPALTENWAHL(A1:D17; 2)` die zweite Spalte des Bereichs `A1:D17` aus. Man erhält dann die Werte im Bereich `B1:B17`. Diese Funktion entspricht der folgenden Operation mit `INDEX()`.

```
= INDEX(
    A1:D17;
    SEQUENZ(ZEILEN(A1:D17));
```

```
    2
)
```

Analog verhält sich die Funktion `ZEILENWAHL()`. Sie wählt die angegebene Zeile aus einem Bereich aus. Für den Bereich `A1:D17` wählt der Aufruf `ZEILENWAHL(A1:D17; 7)` die Werte im Bereich `A7:D7` aus. Dieser Aufruf entspricht der folgenden Opeation mit `INDEX()`.

```
= INDEX(
    A1:D17;
    7;
    SEQUENZ(1; SPALTEN(A1:D17))
)
```

Ähnlich wie die `INDEX()`-Funktion unterstützen beide Funktionen die Auswahl mehrerer Spalten bzw. Zeilen. Dazu muss entsprechend ein Vektor mit den gewünschten Zeilen- bzw. Spaltennummern übergeben werden.

> **Praxis**
>
> Die Funktionen `SPALTENWAHL()` und `ZEILENWAHL()` sind der Funktion `INDEX()` vorzuziehen, wenn nur Spalten oder Zeilen in einer Operation extrahiert werden müssen. Operationen, die `SPALTENWAHL()` und `ZEILENWAHL()` verketten, sollten jedoch vermieden werden. In solchen Fällen ist die Funktion `INDEX()` meist besser geeignet, weil dann die Funktionsverkettung entfällt.

Teil IV.

Deskriptive Datenanalyse

15. Daten beschreiben

15.1. Universelle Kennwerte

Die universellen Kennwerte von Datenrahmen werden mit Excel auf die Gleicheweise bestimmt, wie die Dimensionen einer Matrix (s. Kapitel 12):

Es werden die Zeilen und Spalten der Daten bestimmt. Beim Stichprobenumfang müssen ggf. ungültige Datensätzte gefiltert werden. Hierzu hilft meist ein Sekundärindex, mit dem die gültigen Datensätze markiert werden.

15.2. Variablenumfänge

Die Variablenumfänge ist durch die Anzahl der gültigen Werte gegeben. Die Funktion `ANZAHL()` ignoriert sowohl die leeren Zellen als auch Fehlerwerte, Wahrheitswerte und Zeichenketten. Sie eignet sich damit zur Bestimmung des Variablenumfangs für Zahlenvektoren.

Für alle anderen Vektoren bietet sich die Funktion `ANZAHL2()` an. Diese Funktion ignoriert jedoch keine Fehlerwerte. Deshalb müssen für Zeichenkettenvektoren fehlenden Werte, die mit `#NV` markiert wurden, gefiltert werden, bevor der Vektorumfang bestimmt wird.

15.3. Ungültige Werte entfernen

Wird diese Operation über mehrere Adressen verteilt, wandelt Excel die leeren Zellen in den Wert 0 um. Deshalb müssen zuerst alle leeren Zellen und anschliessend alle Fehlerwerte gefiltert werden.

Beispiel 15.1 (Entfernen ungültiger Werte)**.**

```
= FILTER(
    Tabelle[Vektor];
    NICHT(ISTLEER(Tabelle[Vektor]) + ISTFEHLER(Tabelle[Vektor]))
)
```

Erst über diese bereinigten Daten darf der Vektorumfang bestimmt werden. Weil durch das Entfernen der ungültigen Werte die Vektorlänge identisch mit dem Vektorumfang ist, kann der Vektorumfang auch mittels der Funktion `ZEILEN()` bzw. `SPALTEN()` bestimmt werden.

Ist der Datentyp eines Vektors festgelegt, dann lässt sich das Filtern ungültiger Werte stark vereinfachen. Dazu wird die Typenerkennung mit der entsprechenden `IST`-Funktion als Filter eingesetzt.

Beispiel 15.2 (Entfernen ungültiger Werte durch erzwingen des Datentyps Zeichenkette)**.**

```
= FILTER(
    Tabelle[Vektor];
    ISTTEXT(Tabelle[Vektor])
)
```

Diese Strategie hat den Vorteil, dass auch alle Werte mit falschen Datentyp als ungültig erkannt und entfernt werden.

> **Merke**
>
> Ungültige Werte müssen auch für alle Lagemasse entfernt werden.

15.4. Lagemasse

15.4.1. Median

Der Median wird mit der Funktion `MEDIAN()` bestimmt. Dabei muss beachtet werden, dass diese Funktion ***ausschliesslich*** mit Rohdaten verwendet werden **muss**. Werden aggregierte Werte dieser Funktion übergeben, liefert diese Funktion den Median der aggregierten Werte und nicht den gesuchten Wert.

Ordinalskalierte Daten können auch als Zeichenketten vorliegen. Diese Werte müssen zuerst in Zahlen kodiert werden, um den Median bestimmen zukönnen.

Beispiel 15.3 (Bestimmen des Medians)**.**

```
=MEDIAN(
    FILTER(
        Tabelle[Vektor];
        ISTZAHL(Tabelle[Vektor])
    )
)
```

15.4.2. Mittelwert

Der Mittelwert wird mit der Funktion `MITTELWERT()` bestimmt. Dieser Kennwert darf nur für metrisch-skalierte Daten bestimmt werden. Diese Daten liegen immer als Zahlen vor, weshalb diese Werte nicht kodiert werden müssen.

Beispiel 15.4 (Bestimmen des Medians)**.**

```
=MEDIAN(
    FILTER(
        Tabelle[Vektor];
        ISTZAHL(Tabelle[Vektor])
    )
)
```

15.5. Streumasse

15.5.1. Bandbreite

Die Bandbreite ergibt sich aus der Differenz zwischen dem kleinsten und dem grössten gemessenen Wert. Die Bandbreite wird nur als eigenständiger Wert angegeben, um ähnliche Bandbreiten in unterschiedlichen Wertebereichen anzuzeigen. Sonst wird nur das Minimum mit `MIN()` und das Maximum mit `MAX()` für den Vektor bestimmt.

15.5.2. Quartiele

Die Quartile werden mit der Funktion `QUARTILE.INKL()` bestimmt. Dazu wird der bereinigte Vektor als erster Parameter übergeben und anschliessend der das jeweilige Quartil. Das untere Quartil wird mit dem Wert `1` als zweiten Parameter berechnet und das obere Quartil mit dem Wert `3`.

Wie beim Median können die Werte für die Quartile mit Zahlenwerten berechnet werden.

Die Funktion `QUARTILE.INKL()` erlaubt es, die Minimalwert, den Maximalwert und den Median mit einem Aufrug zu bestimmen. Dazu wird als zweiter Parameter eine Sequenz von `0-4` übergeben.

- Der Wert `0` steht für das Minimum,
- Der Wert `1` für das untere Quartil,
- Der Wert `2` für den Median,
- Der Wert `3` für das obere Quartil und
- Der Wert `4` steht für das Maximum.

Beispiel 15.5 (Bandbreite, Median und Quartile mit `QUARTILE.INKL()` bestimmen.)**.**

```
=QUARTILE.INKL(
    FILTER(
        Tabelle[Vektor];
        ISTZAHL(Tabelle[Vektor])
    );
    SEQUENZ(1; 5; 0)
)
```

15.5.3. Varianz und Standardabweichung

Die Varianz der gemessenen Werte eines Vektors werden mit der Funktion `VAR.S()` bestimmt. Die Standardabweichung der gemessenen Werte wird mit der Funktion `STABW.S()` bestimmt. Excel kennt für die echte Varianz und die echte Standardabweichung die Funktionen `VAR.P()` udn `STABW.P()`, diese werden für die Beschreibung gemessener Daten nicht verwendet.

15.5.4. Interquartilsabstand

Excel kennt keine Funktion für den Interquartilsabstand. Dieser muss aus den beiden Quartilen mit der Formel aus Beispiel 15.6 berechnet werden.

Beispiel 15.6.

```
= Quartil3 - Quartil1
```

15.5.5. Mittlere Absolute Abweichung (MAD)

Die mittlere absolute Abweichung muss in EXCEL ebenfalls händisch berechnet werden.

Beispiel 15.7.

```
=LET(
    Werte; FILTER(
        Tabelle[Vektor];
        ISTZAHL(Tabelle[Vektor])
    );
    MEDIAN(
        ABS(
            Werte - MEDIAN(Werte)
        )
    )
)
```

16. Daten visualisieren

16.1. Diagramme erstellen

> **Warnung**
>
> Excels Diagrammtypen erfordern, dass die Daten in einem bestimmten Format vorliegen. Das notwendige Format hängt vom Diagrammtyp ab. Es lassen sich deshalb nicht alle Visualisuerungen aus den gleichen Datenstrukturen erzeugen.

16.1.1. Datenvorbereitung

Die verschiedenen Diagrammtypen erfordern unterschiedliche Datenformate. Deshalb müssen die Daten für die Visualisierung vorbereitet und in eine geeignete Form gebarcht werden.

Alle Werte, die visualisiert werden sollen, müssen einen Bereich bilden. Dieser Bereich kann innerhalb einer Tabelle liegen oder dynamische Felder umfassen.

Excel organisiert die Werte für die Darstellung in *Datenreihen*. Je nach Visualisierung besteht eine Datenreihe aus einer Überschrift und einer, zwei oder drei Spalten mit Werten.

> **Praxis**
>
> Es vereinfacht die Arbeit, wenn die Spalten einer Datenreihe möglichst nebeneinander geschrieben werden. Alternativ können die gleichen Teilspalten organisiert werden. Letzteres bietet sich immer dann an, wenn die Werte in die Breitform *transponiert* werden müssen.

Erlaubt ein Diagramm horizontale oder vertikale Achsbeschriftungen, dann werden diese normalerweise in den Spalten *vor* den eigentlichen Werte für den Darstellungsbereicht positioniert.

16.1.2. Diagrammerstellung

Um ein Diagramm zu erstellen müssen die vorbereiteten Daten markiert werden. Excel Diagramme können *nicht* mit dynamischen Feldern umgehen, so dass *alle* darzustellenden Werte markiert werden müssen.

Ausser den Spaltenüberschriften sollten keine weiteren Daten markiert werden. Dazu gehören auch die Werte, die für die Beschriftung der X-Achse benötigt werden. Abbildung 16.1 zeigt eine solche Markierung.

	A	B	C	D
1	Gruppe	x	y	
2	2	1	12	
3	3	8	16	
4	4	34	36	
5	5	14	10	
6	6	10	3	
7				
8				

Abbildung 16.1.: Markierung der zu visualisierenden Werte

> **!** Achtung
>
> Oft liegen Daten als Tabellen oder Dynamische Felder vor. Verändert sich die Anzahl der Werte einer solchen Datenstruktur, betrifft diese Änderung **nicht** den markierten Wertebereich. Liegen neue Werte ausserhalb des markierten Bereichs für ein Diagramm, dann werden diese Werte nicht dargestellt. In solchen Fällen muss der Bereich für die Darstellung nachträglich erweitert werden.

Nachdem wie zu visualisierenden Werte markiert wurden, kann das eigentliche Diagramm eingefügt werden. Dazu muss aus dem Menübalken `Einfügen` im Abschnitt `Diagramme` (Abbildung 16.2) die gewünschte Darstellung ausgewählt werden.

Abbildung 16.2.: Menübalken Einfügen/Diagramme

Die Auswahl erzeugt das gewünschte Diagramm für die markierten Daten auf dem aktuellen Arbeitsblatt.

16.1.3. Diagrammbearbeitung

Sobald ein Diagramm erstellt wurde, kann es über das Menüband `Diagrammentwurf` angepasst werden. Dieses Menüband wird nur angezeigt, wenn ein Diagramm ausgewählt wurde.

Abbildung 16.3.: Menüband Diagrammentwurf

> **Praxis**
>
> Für die Datenvisualisierung sind das Untermenü `Diagrammelement hinzufügen` und das Kommando `Daten auswählen` zentral, weil über sie die Darstellung gesteuert wird. Daneben wird das Farbschema über das Menü `Farben ändern` gesteuert.

Das Untermenu `Diagrammelement hinzufügen` erlaubt es, einzelne Diagrammelement zur Visualisierung hinzuzufügen **oder zu entfernen**. Die möglichen Diagrammelemente hängen vom jeweiligen Diagrammtyp ab.

Das Untermenu `Schnelllayout` bietet Vorlagen für die einzelnen Diagrammelemente. Diese Vorlagen dienen oft als Basis für die weitere Verfeinerung mit dem Untermenü `Diagrammelement hinzufügen`.

Mit dem Untermenü `Farbenändern` können die im Diagramm verwendeten Farben angepasst werden. In diesem Menü finden sich verschiedene Farbpaletten. Diese Farbpaletten werden durch das aktuelle Farbschema der Arbeitsmappe bestimmt.

Im Bereich `Diagrammformatvorlagen` kann die allgemeine Darstellung angepasst werden. Diese Einstellungen sollten nur angepasst werden, um ein Diagramm für eine besondere Präsentation vorzubereiten. Normalerweise werden über diesen Bereich keine Änderungen vorgenommen.

Das Kommando `Zeile/Spalte tauschen` ist bei guter Vorbereitung der Daten nicht notwendig. Dieses Kommando ändert für eine Visualisierung die Verwendung von Spalten in die Verwendung von Zeilen. Dieses Kommando ist entsprechend nur Notwendig, wenn die Daten zeilenweise anstatt spaltenweise vorbereitet wurden.

> **Warnung**
>
> Manche Visualisierungen verwenden ein komplexes internes Datenmodell, so dass die mehrfache Anwendung des Kommandos `Zeile/Spalte tauschen` nicht zwingend zur ursprünglichen Darstellung führt.

Mit `Daten auswählen` werden die Datenreihen und Quellen für Achsbeschriftungen in der Arbeitsmappe festgelegt. Dieses Kommando öffnet den Dialog `Datenquelle auswählen`, mit dem die Daten den einzelnen Darstellungselementen zugewiesen werden. Mit diesem Kommando können die Datenreihen korrigiert werden, wenn die automatische Erkennung von Excel nicht das gewünschte Ergebnis erzielt hat.

> **MacOS vs. Windows**
>
> Das Kommando `Daten auswählen` heisst unter MacOS `Daten markieren`.

Das Untermenü `Diagrammtyp ändern` erlaubt es, den Typ eines Diagramms zu verändern, ohne den Diagrammbereich zu verändern. Weil die verschiedenen Diagrammtyp recht unterschiedliche Anforderungen an die Datenorganisation haben, sollte hier nur Diagrammtypen innerhalb der gleichen Gruppe ausgewählt werden. Beispielsweise könnte so ein Balkendiagramm in ein Säulendiagramm geändert werden.

Das Kommand `Diagramm verschieben` erlaubt es, ein Diagramm auf einem eigenen Arbeitsblatt zu platzieren. Dieses Kommando ist nur dann sinnvoll, wenn eine Arbeitsmappe zur Präsentation der Daten verwendet wird.

16.1.4. Dialog `Datenquelle auswählen`

Über das Kommando `Daten markieren` wird der Dialog `Datenquelle auswählen` (Abbildung 16.4) geöffnet, über den die Daten den Darstellungselementen zugewiesen werden.

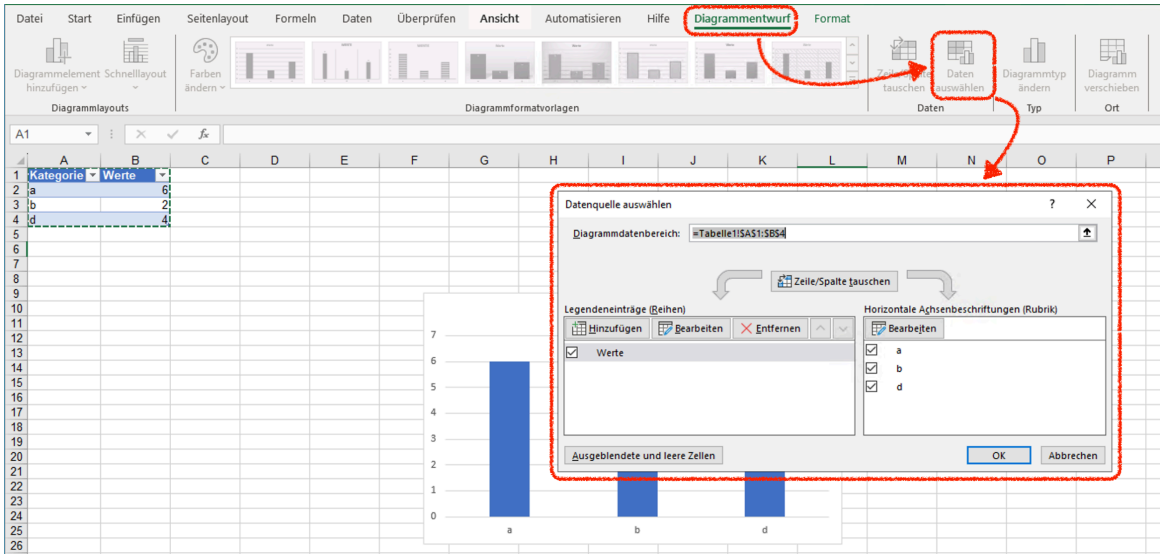


Abbildung 16.4.: Dialog Datenquelle auswählen (Windows)

Durch einen Doppelklick auf eine Datenreihe öffnet sich ein zweiter Dialog, um eine Datenreihe zu konfigurieren (Abbildung 16.5). Über diesen Dialog sollten die Datenreihen konfiguriert werden, damit sicher gestellt wird, dass die Datenreihen auf die richtigen Werte verweisen.

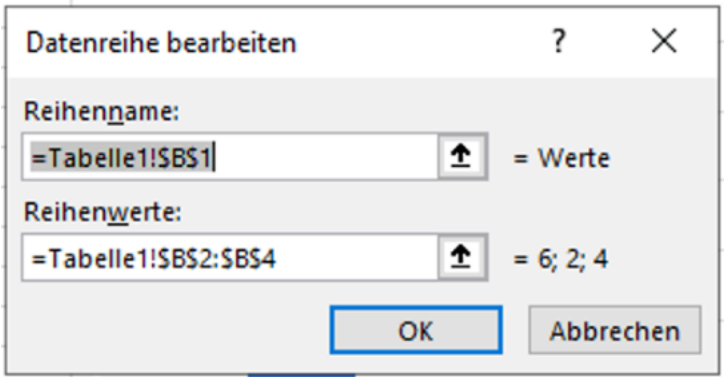


Abbildung 16.5.: Datenreihe konfigurieren (Windows)

MacOs vs. Windows

Unter MacOS ist die gleiche Funktion etwas anders angeordnet.

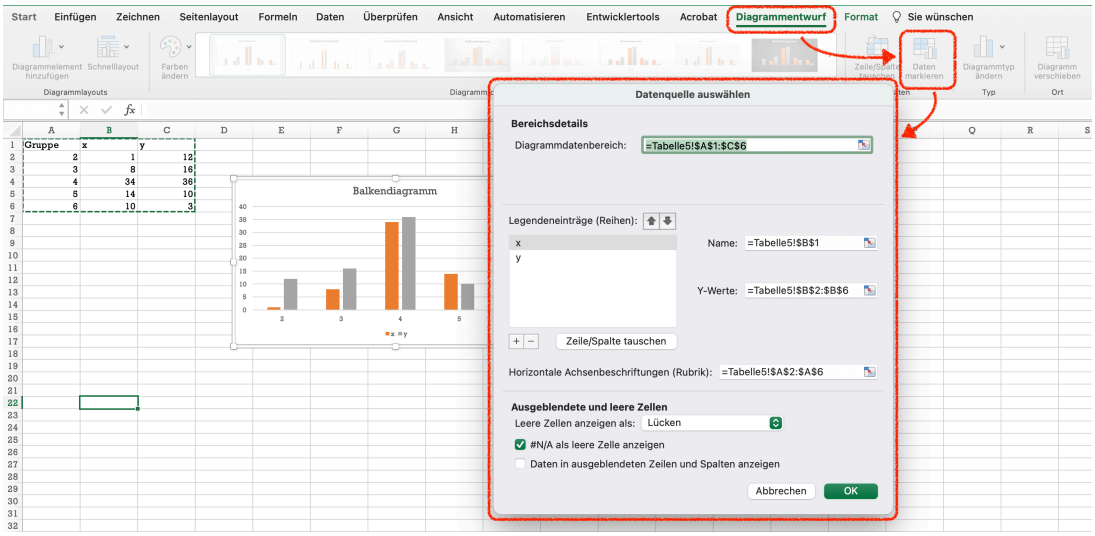


Abbildung 16.6.: Dialog Datenquelle auswählen (MacOS)

Der Dialog besteht aus drei Teilen:

- Der Teil `Bereichdetails` zeigt den Datenbereich des Diagramms an. Der angezeigte Bereich sollte **nicht** in diesem Teil verändert werden.
- Der Teil `Legendeneinträge (Reihen)` weist Spalten mit Werten den Datenreihen des Diagramms zu. In diesem Teil werden die verwendeten Datenbereiche angepasst, die im Teil `Bereichdetails` zusammengefasst angezeigt werden.
- Im Teil `Ausgeblendete und leere Zellen` wird die Behandlung von leeren Zellen und `#NA`-Werten festgelegt. In diesem Teil sind nur selten Anpassungen notwendig.

16.1.5. Diagramme formatieren

Die Darstellungselemente von Diagrammen können über den **Formatierungsbereich** angepasst werden. Über den Formatierungsbereich lassen sich die alle Details eines Diagramms anpassen.

MacOs vs. Windows

Zur Formatierung der Datenreihen wird der Formatierungsbereich benötigt. Dieser wird unter Windows über das Kommando `Auswahl formatieren` geöffnet. Die wichtigsten Formatierungen sind die Datenreihenformatierungen. Je nach Diagrammtyp lassen sich in dieser Rubrik Formatierungen in Abhängigkeit zu den abgebildeten Daten justieren.

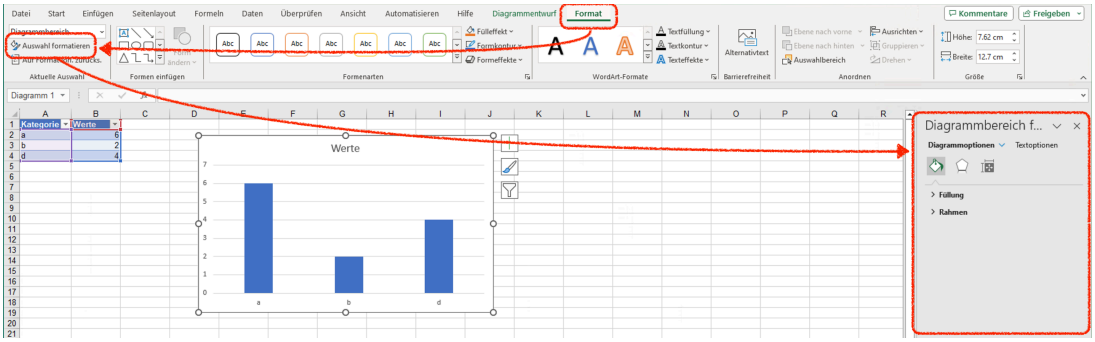


Abbildung 16.7.: Formatierungsbereich öffnen (Windows)

Unter MacOS wird der Formatierungsbereich über das Kommando `Formatierungsbereich` im Menüband `Format` geöffnet.

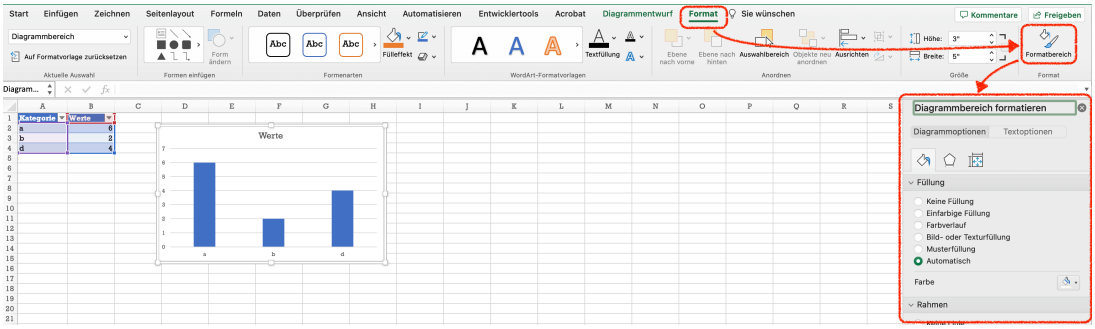
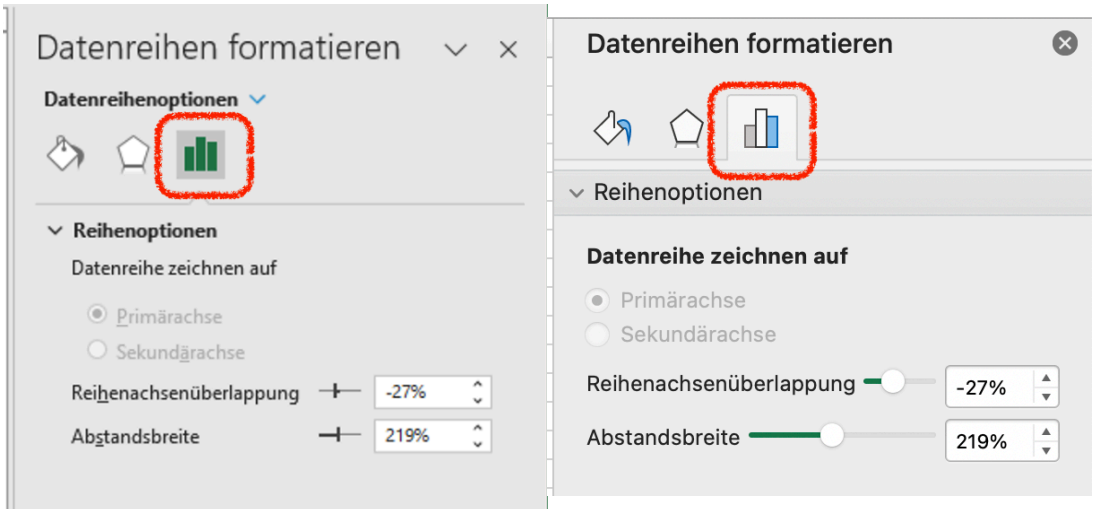


Abbildung 16.8.: Formatierungsbereich öffnen (MacOS)

Nach dem Öffnen des Formatierungsbereichs können die einzelnen Darstellungselemente gezielt formatiert werden. Immer wenn ein Darstellungselement sich auf Datenreihen bezieht, lassen sich zuätzliche Formatierungen unter der Rubrik `Reihenoptionen` konfigurieren. Diese Optionen werden unter MacOs und Windows leicht unterschiedlich dargestellt (Abbildung 16.9)



(a) Windows

(b) MacOS

Abbildung 16.9.: Datenreihen formatieren

16.1.6. Diagramme exportieren

Um ein Diagramm zum Einbetten in andere Programme bereitzustellen, muss das Diagramm aus Excel exportiert werden. Dazu ein Rechtsklick auf das Diagramm öffnet ein *Kontextmenü*, in welchem sich der Punkt `Als Grafik speichern ...` findet (Abbildung 16.10).

> **MacOS vs. Windows**
>
> Das Kommando zum Speichern von Diagrammen heisst unter MacOS `Als Bild speichern ...` und unter Windows `Als Grafik speichern ...`.

Anschliessend erscheint ein Dialog zum Speichern der Diagrammdatei (Abbildung 16.11). Standardmässig bietet dieser Dialog als `Dateityp` das Grafikformat `PNG` an, was in den meisten Fällen gewählt werden sollte. Alternativ können Diagramme in den Formaten `JPEG`, `GIF`, `PDF`, `BMP` und `SVG` exportiert werden. Hier sind für die Praxis nur die beiden Formate `PDF` und `SVG` von Bedeutung.

> **Praxis**
>
> Das `PDF`-Format sollte gewählt werden, wenn eine Grafik einzeln ausgedruckt werden soll.

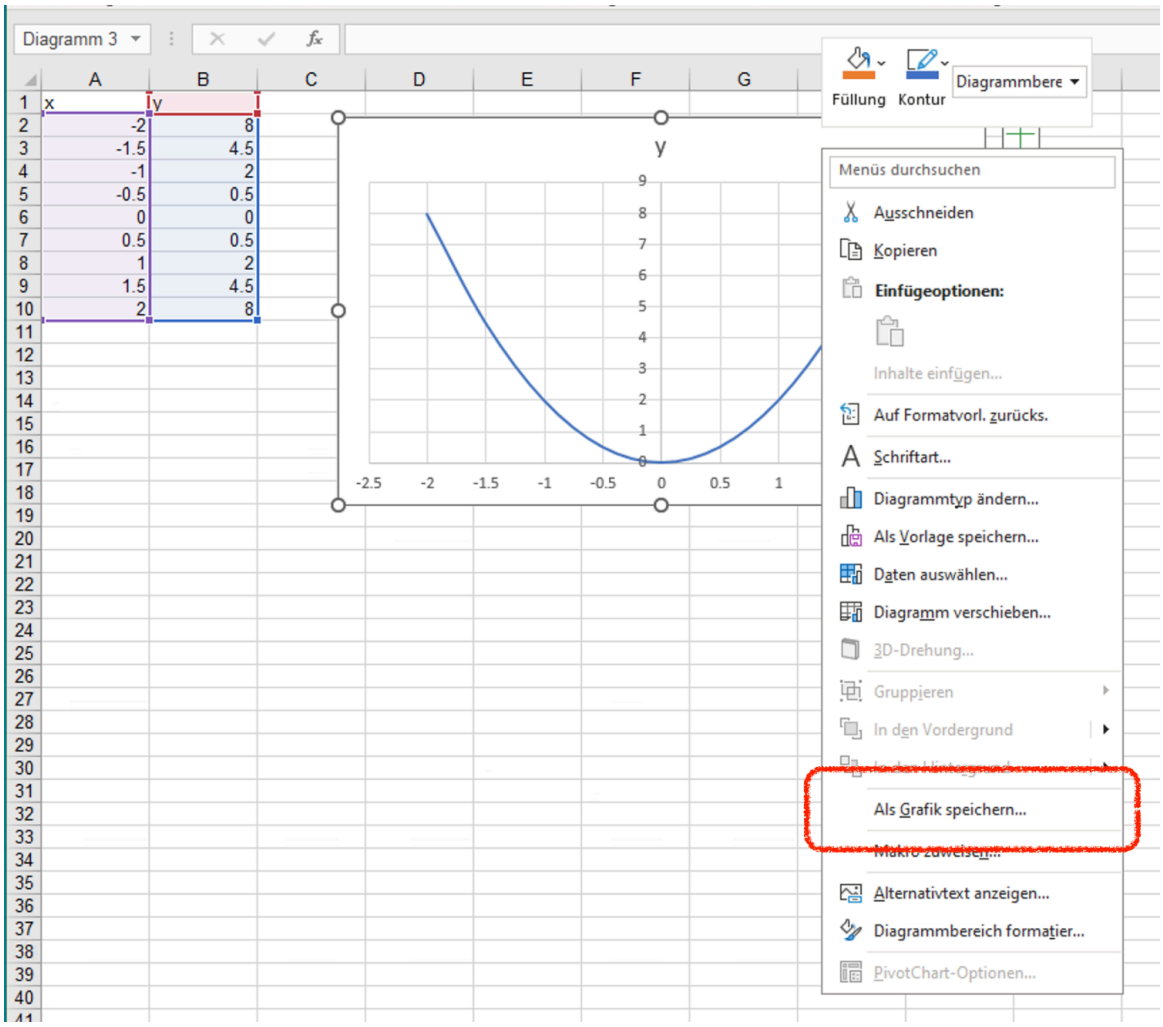


Abbildung 16.10.: Kontextmenu zum Exportieren eines Diagramms

> **Praxis**
>
> Das `SVG`-Format sollte gewählt werden, wenn das Diagramme grossformtig oder in hoher Qualität auf Web-Seiten veröffentlicht werden soll.

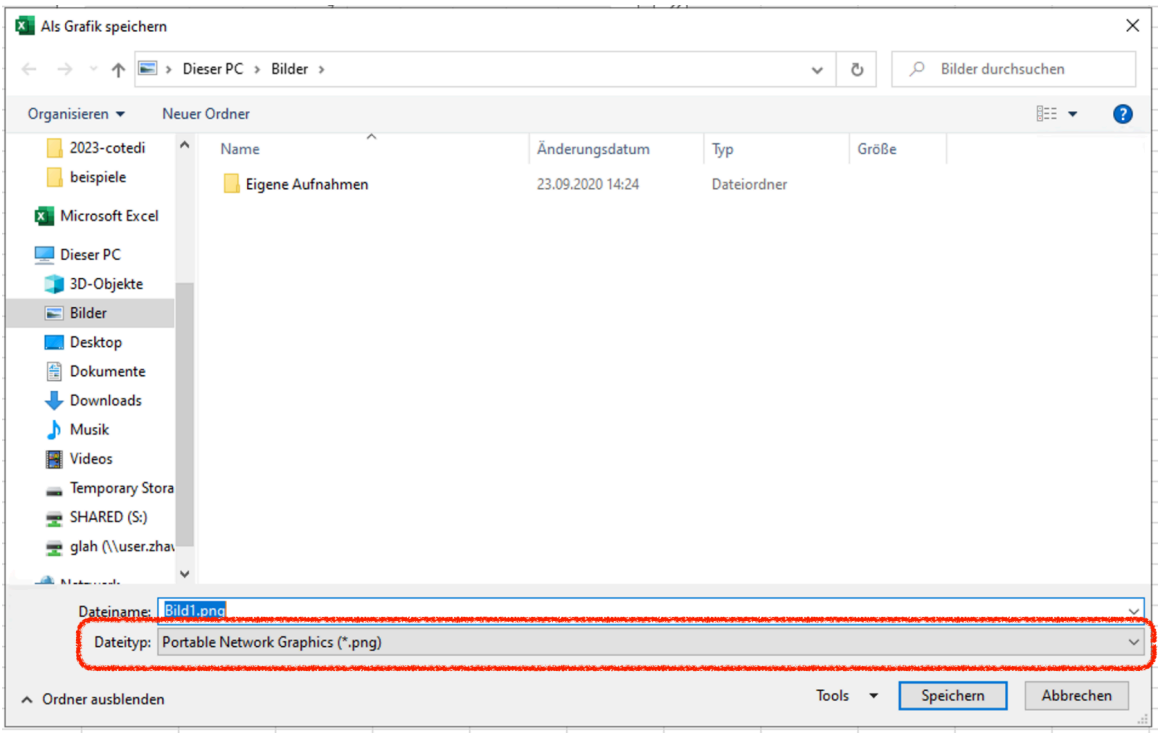


Abbildung 16.11.: Auswahl zum Speichern eines Diagramms

16.2. Diagrammtypen

Alle Excel Diagramme visualisieren Werte vom Datentyp **Zahl**. Um andere Datentypen zu visualisieren, müssen diese zuerst in Zahlen kodiert werden.

16.2.1. Balkendiagramme

Excel unterscheidet zwischen Balken- oder Säulendiagrammen. Technisch unterscheiden sich die beiden Diagrammtype nur durch die Orientierung der Balken. Im Folgenden werden beide Diagrammtypen als Balkendiagramme bezeichnet.

> **Merke**
>
> Für ein Balkendiagramm entspricht jeder Wert einem Balken und der Wert bestimmt dessen Grösse.

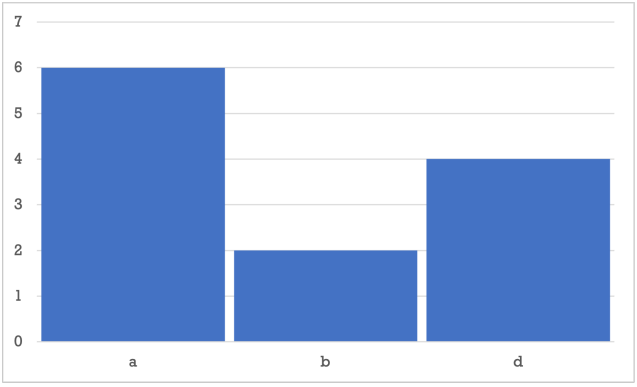
Die Anzahl der Werte entspricht der Anzahl der Balken. Daraus folgt, dass ein Balkendiagramm immmer diskrete Daten auf der X- und Zahlenwerte auf der Y-Achse abbildet.

Balkendiagramme können keine Werte zusammenfassen. Deshalb müssen Daten vor der Visualisierung als Balkendiagramm aggregiert werden. Normalerweise erfolgt ein gruppiertes Zählen mit `ZÄHLENWENNS()`.

Die Werte können als Zeile oder als Spalte angegeben werden. Wird ein zweiter Vektor mit Textwerten angegeben, dann übernimmt Excel diese Werte als Beschriftung der X-Achse.

Kategorie	Werte
a	6
b	2
d	4

(a) Daten



(b) Diagramm

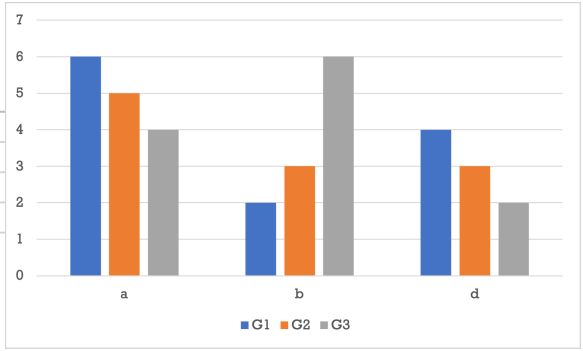
Abbildung 16.12.: Vorbereitete Datentabelle mit Balkendiagramm

> **Merke**
>
> Für Balkendiagramme mit zwei oder mehr Gruppen, **müssen** die Daten in der *Breitform* vorliegen.

Gruppierte Daten werden farblich voneinander abgehoben. Die Farben können über das Farbschema der Arbeitsmappe über das Kommando `Farben ändern` angepasst werden.

Kategorie	G1	G2	G3
a	6	5	4
b	2	3	6
d	4	3	2

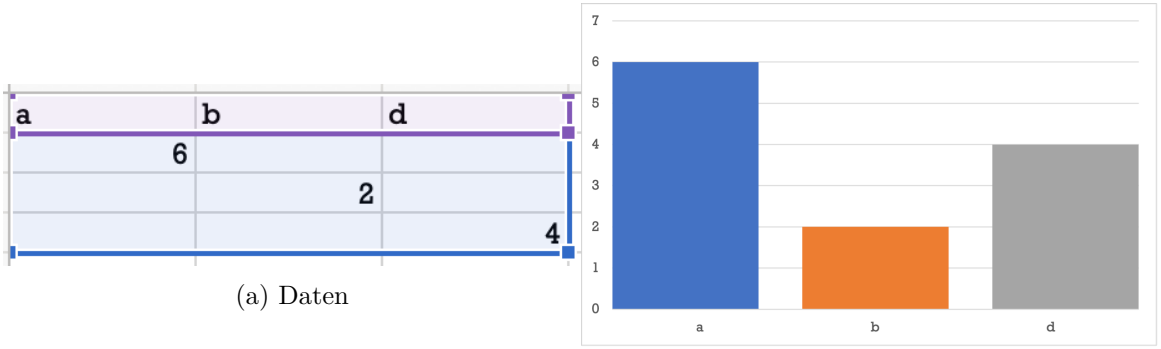
(a) Daten



(b) Diagramm

Abbildung 16.13.: Gruppierte Datentabelle mit Balkendiagramm

Sollen einzelne Kategorien farblich hervorgehoben werden, dann müssen die Werte gruppiert organisiert werden, wobei jeder Wert in einer eigenen Gruppe geführt wird (Abbildung 16.14).



(a) Daten

(b) Diagramm

Abbildung 16.14.: Datentabelle und Balkendiagramm mit farblich abgehobenen Kategorien.

16.2.1.1. Darstellung optimieren

Bei der Darstellung von Balkendiagrammen verwendet Excel immer Standardeinstellungen. Diese nutzen die Fläche des Diagramms nicht unbedingt optimal: Auf der Y-Achse wird immer eine Hauptbeschriftung oberhalb des grössten Wers eingefügt und die Abstände zwischen den Balken sind grösser als die Balken (Abbildung 16.15).

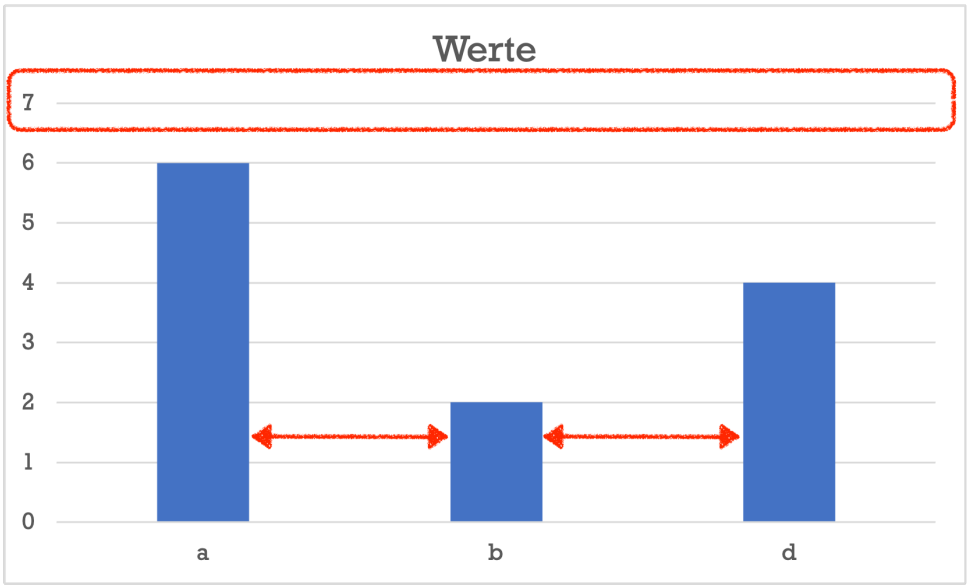


Abbildung 16.15.: Balkendiagramm mit Standardeinstellungen

Die Y-Achse kann immer auf den Maximalwert begrenzt werden. Dazu wird die Achsbeschriftung mit der Maus ausgewählt. Anschliessend können in den Achsenoptionen die Grenzen

der Y-Achse festgelegt werden. Hier sollte das Maximum auf den grössten dargestellten Wert gesetzt werden (s. Abbildung 16.16).

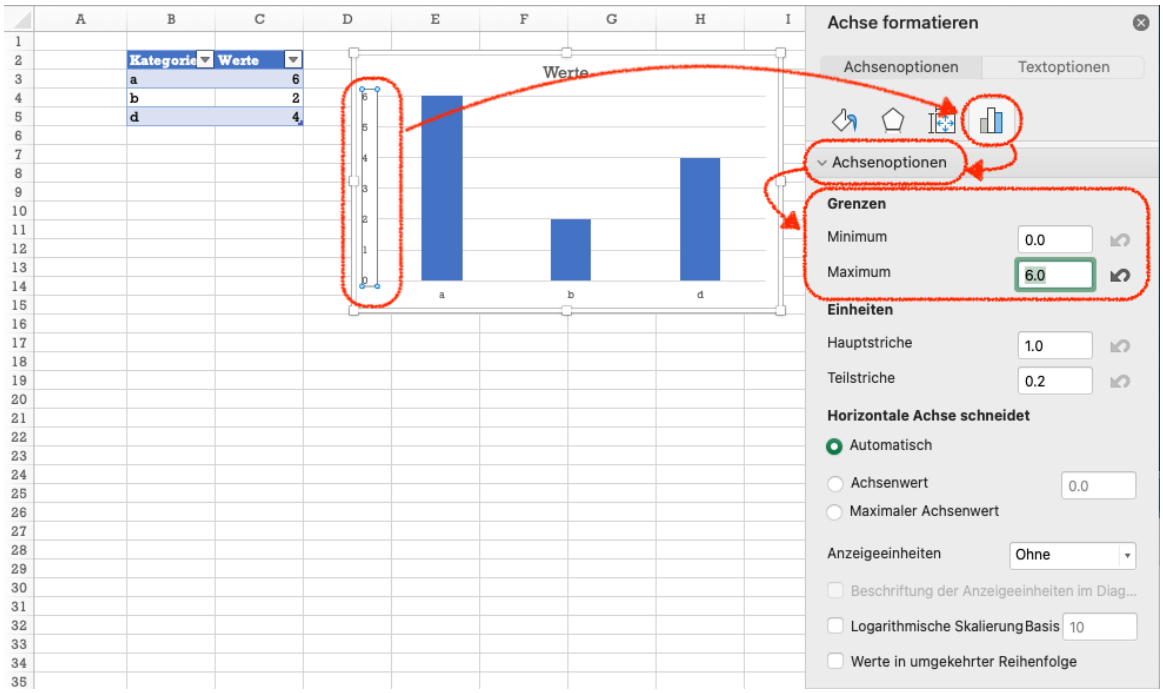


Abbildung 16.16.: Konfiguration der Achseneinstelleungen

> **Merke**
>
> Für Balkendiagramme sollte das Minimum der Y-Achse immer `0.0` sein.

Die Abstände zwischen den Balken werden über die Formatierung der Balken gesteuert. Dazu muss ein Balken mit der Maus ausgewählt werden. Die einzige Einstellungskategorie für die Datenformatierung sind die `Reihenoptionen`, wo sich die beiden Einstellungen `Reihenüberlappungen` und `Abstandsbreite` finden (s. Abbildung 16.17).

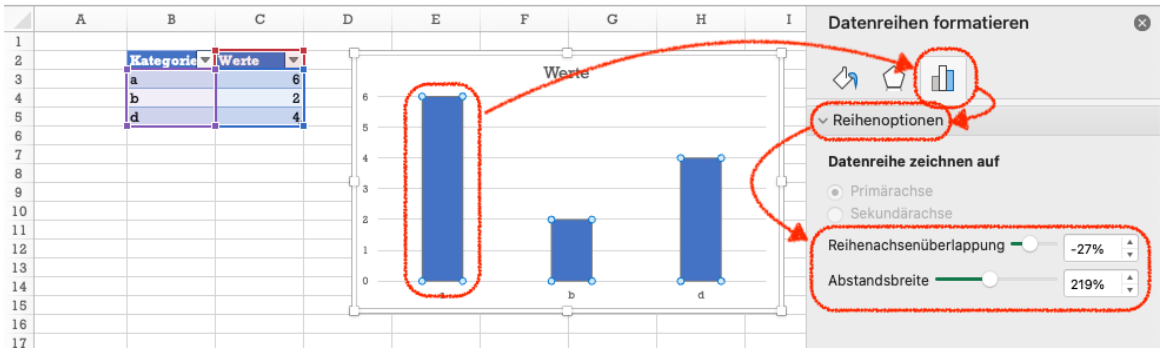


Abbildung 16.17.: Balkenbreite und -abstände konfigurieren

Abbildung 16.18 zeigt die Bedeutung der beiden Optionen.

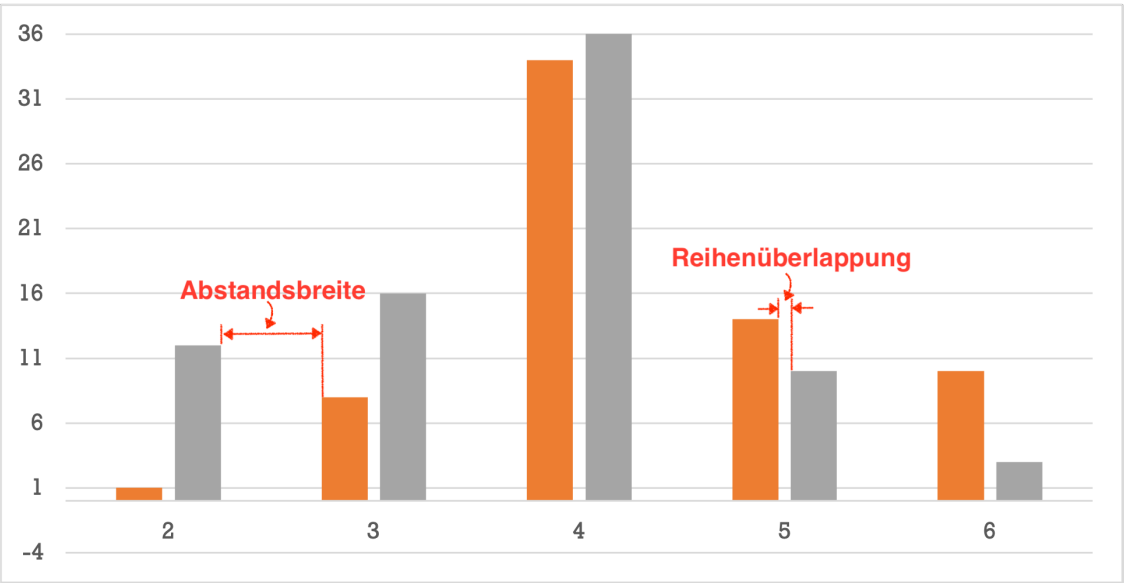


Abbildung 16.18.: Unterschied zwischen Reihenüberlappung und Abstandsbreite

Die Option `Reihenüberlappung` legt die Abstände ziwschen Balken in der gleichen Kategorie fest. Die Abstände werden in Prozent der Überlappung angegeben. 0% bedeutet keine Überlappung ohne Abstand. 100% bedeutet vollständige überlappung und -100% bedeutet einen Abstand einer Balkenbreite.

Die Option `Abstandsbreite` legt die Abstände zwischen den Kategorien fest. Die Abstände werden in Prozent der Balkenbreite angegeben.

> **Achtung**
>
> Die Überlappung und die Balkenbreite wird auch dann berücksichtigt, wenn ein Balken nicht dargestellt wird!

Um farblich voneinander unterschiedene Kategorien mit gleicher Balkenbreite wie in Abbildung 16.14 zu erzeugen, muss die `Reihenüberlappung` auf `100%` gesetzt werden. Anschliessend können die Abstände mit der Option `Abstandsbreite` kontrolliert werden. Abbildung 16.14 verwendet eine `Abstandsbreite` von `10%`.

16.2.2. Spezielle Balkendiagramme

Excel bietet zwei spezielle Formen von Balkendiagrammen: *Trichterdiagramme* und *Wasserfall*-Diagramme. Beide Diagramme sind *eindimensional*.

Trichterdiagramme stellen die Balken horizontal zentriert dar, wobei nur positive Werte zulässig sind. Deshalb haben Excels Trichterdiagramme *keine* Beschriftung der X-Achse (Abbildung 16.19).

> **Merke**
>
> Trichterdiagramme unterscheiden sich von anderen Balkendiagrammen nur in der Anordnung der Balken.

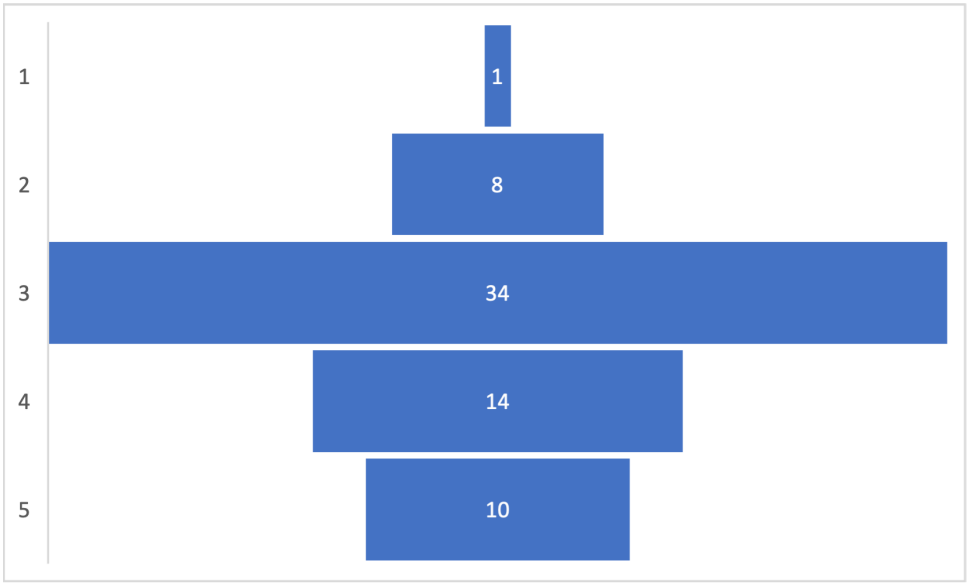


Abbildung 16.19.: Trichterdiagramm mit fünf Werten

> **Praxis**
>
> Weil Trichterdiagrammen keine beschriftete X-Achse haben und die Anordnung der Balken schwerer zu dekodieren ist, sollten diese Diagramme **nicht** verwendet werden.

Wasserfalldiagramme stellen die Balken so dar als Folge dar. Diese Diagramme stellen positive und negative Werte gerichtet nebeneinander. Der Start des von links nach rechts gelesenene nächste Balken beginnt bei dem Wert, an dem der vorherige Balken endete. Bei positiven Werten befindet sich der Endpunkt über dem Startpunkt, bei negativen Werten darunter (Abbildung 16.20).

> **Merke**
>
> Wasserfalldiagramme werden zur Visualisierung von Veränderungen über die Zeit verwendet.

Bei Wasserfalldiagrammen kann die **Beschriftung** der X-Achse über ein zweite Datenspalte mitgegeben werden. Bei Trichterdiagrammen können auf die gleiche Weise die Beschriftungen der Y-Achse angepasst werden. In beiden Fällen müssen die Beschriftungen in der Spalte *vor* den Werten gespeichert sein.

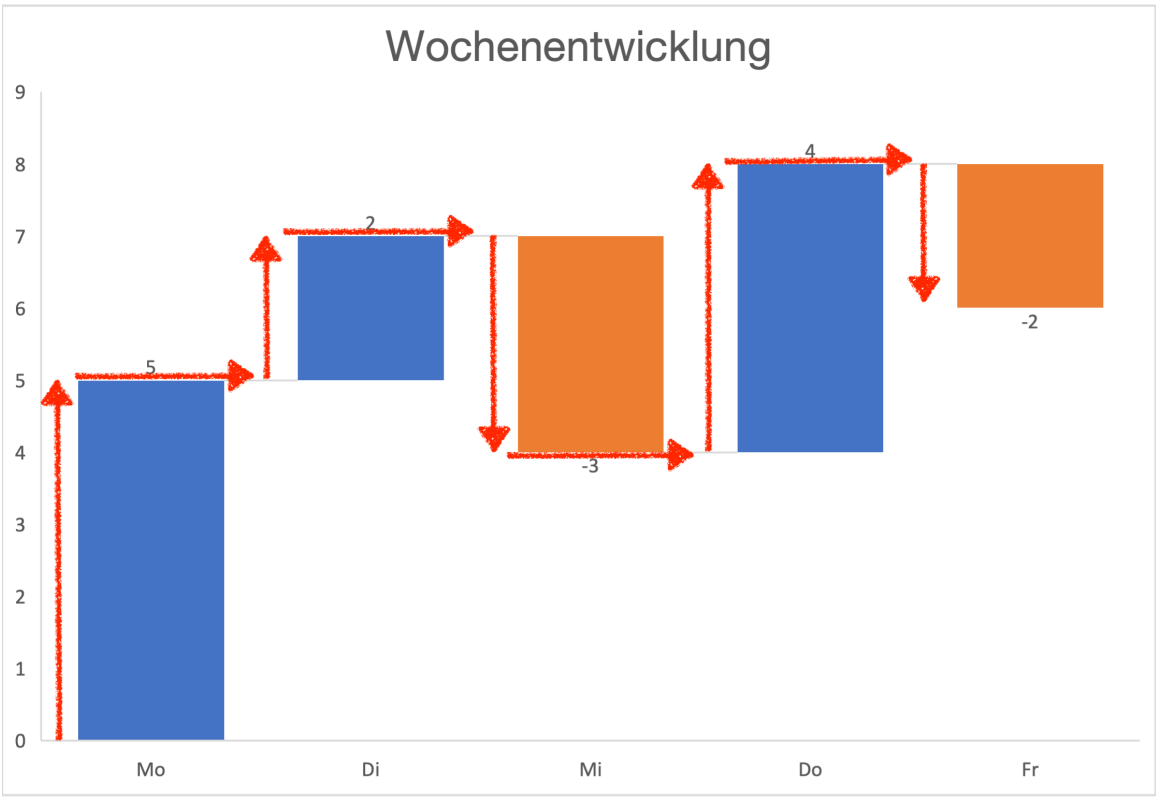


Abbildung 16.20.: Wasserfalldiagramm mit Leserichtungen

16.2.3. Histogramm

> **Merke**
>
> Ein Histogramm ist eine Visualisierung der Werteverteilung als Balkendiagramm.

Histogramme visualisieren *kontinuierliche* bzw. metrischskalierte Wertebereiche, indem zwischen dem kleinsten und dem grössten Wert gleichmässige Intervalle gebildet werden und anschliessend die Anzahl der Werte in den Intervallen bestimmt wird.

> **Merke**
>
> Für ein Excel-Histogramm müssen die Werte vor der Visualisierung **nicht** aggregiert werden.

Ein Histogramm wird oft über einen Vektor erstellt. Werden die Daten so vorbereitet, dass unter den Werten keine anderen Daten stehen, dann kann die *gesamte Spalte* markiert werden, indem auf den Spaltenbuchstaben mit der Maus geklickt wird. Anschliessend kann das Histogramm in die Arbeitmappe eingefügt werden (Abbildung 16.21)

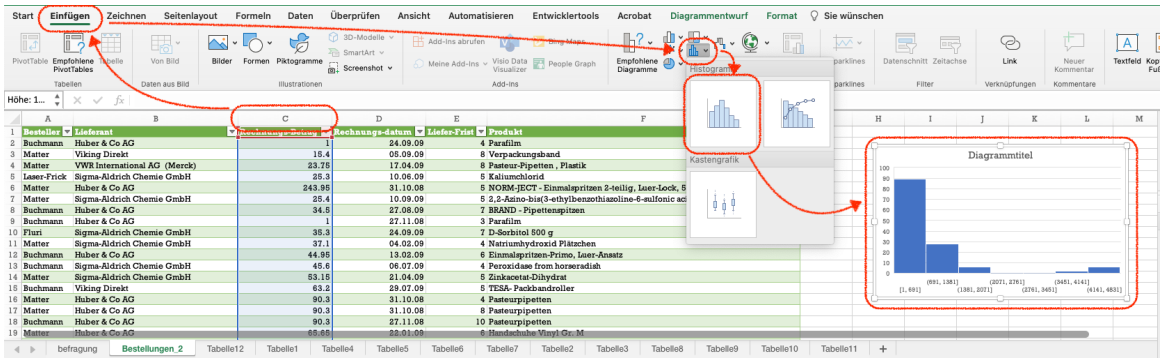


Abbildung 16.21.: Histogramm erstellen

Ein Excel-Histogramm ist **immer** ein eindimensionales Balkendiagramm. Werden Werte aus mehreren Spalten übergeben, dann fliessen alle Werte in das Histogramm ein.

16.2.3.1. Histogramme optimieren

> **Praxis**
>
> Im Gegensatz zu Balkendiagrammen wird die Y-Achse von Histogrammen in der Regel **nicht** angepasst.

Excel versucht die Intervalle für ein Histogramm aus dem vorliegenden Wertebereichen automatisch zu ermitteln. Dazu werden zwischen dem minimalen und maximalen Werten

gleichgrosse Intervalle *proportional* zur Anzahl der Werte gewählt. Das bedeutet, dass Excel grössere Intervalle wählt, wenn weniger Werte vorhanden sind. Das ist nicht immer gewünscht oder die Intervallgrenzen liegen ungeeignet.

Die Intervalle eines Histogramms lassen sich über die Formatierung kontrollieren. Dazu wird mit der Maus ein Balken angeklickt und anschliessend die Datenreihenoptionen ausgewählt.

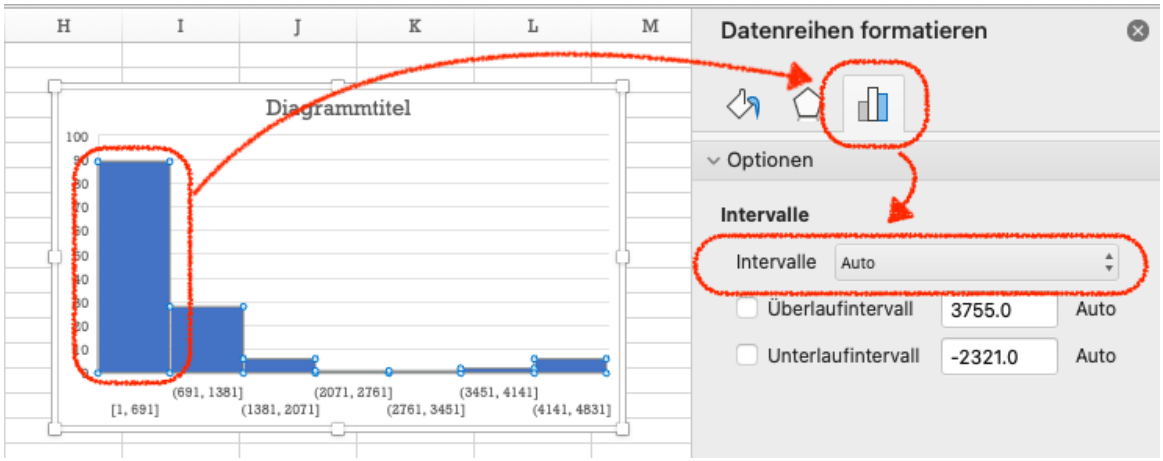


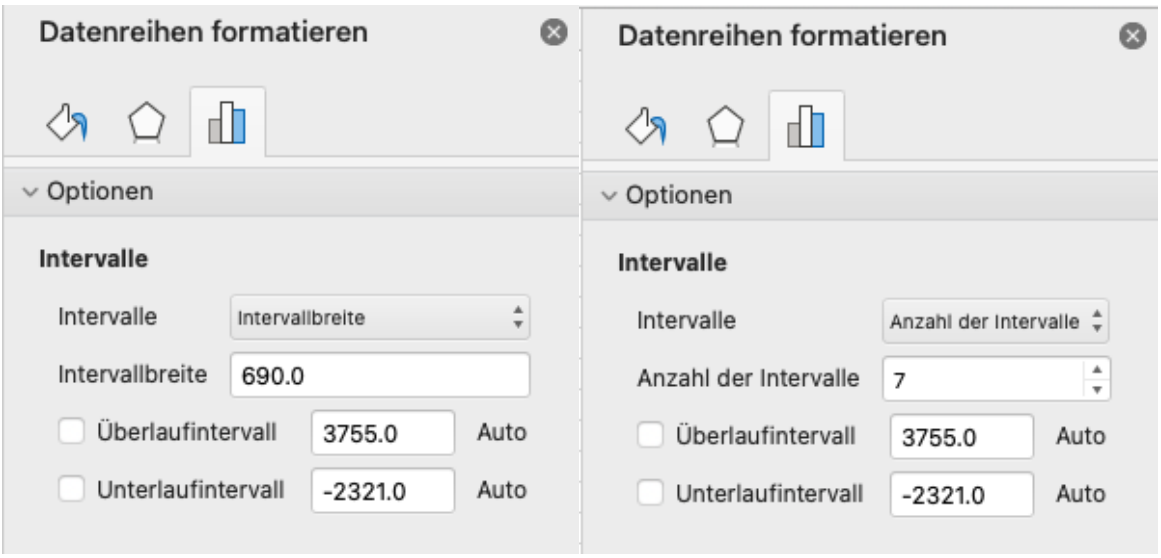
Abbildung 16.22.: Histgrammintervalle konfigurieren

Die Option `Intervalle` steht nach dem Erstellen des Histogramms auf `Auto`. Hier stehen die beiden Optionen `Invervallbreite` und `Anzahl der Intervalle` zur Auswahl.

Mit `Intervallbreite` wird der **Abstand** zwischen zwei Intervallen angegeben (Abbildung 16.23a). Die Intervallgrenzen werden durch den kleinsten und grössten Wert definiert. Wird beispielsweise die Intervallbreite mit 500 angegeben und der kleinste Wert ist 33, dann sind die Intervallgrenzen 33, 533, 1066 usw. Diese Grenzen sind nicht immer intuitiv, besonders wenn Histogramme zum Vergleich von Daten verwendet werden. Mit der Option `Unterlaufintervall` kann die erste Intervallgrenze festgelegt werden. Es bietet sich an, das Unterlaufintervall genauso breit zu machen, wie die Intervallbreite.

Mit `Anzahl der Intervalle` wird die Anzahl der Balken im Histogramm festgelegt (Abbildung 16.23b). Excel berechent aus diesem Wert einen geeigneten Intervallabstand.

Die beiden Optionen `Überlaufintervall` und `Unterlaufintervall` ermöglichen es die Ober- bzw. Untergrenze der regulären Intervalle zu definieren. Die beiden Intervalle entsprechen dem ersten bzw. letzen Balken im Histogramm. Diese beiden Intervalle werden für die Intervallbreite **nicht** berücksichtigt, zählen aber zur Anzahl der Intervalle. Diese Intervalle können verwendet werden, wenn unterhalb der Grenze des Unterlaufintervalls bzw. oberhalb der Grenze des Überlaufintervalls vereinzelt “Ausreisser” mit grösseren Abständen als die Intervallbreite auftreten.



(a) Intervallbreite (b) Intervallanzahl

Abbildung 16.23.: Einstellungsmöglichkeiten für Histogrammintervalle

Praxis

Für die erste Sichtung neuer Daten ist es leichter ein Histogramm über die Anzahl der Intervalle zu definieren. Wird anschliessend auf `Intervallbreite` gewechselt, zeigt Excel die verwendete Intervallbreite an. So lassen sich die Intervalle einfach feinjustieren.

16.2.3.2. Histogramme für diskrete Daten

In der Datenreihenformatierung findet sich zusätzlich die Option `Nach Kategorie`, die keine weiteren Konfigurationsmöglichkeiten bietet. In diesem Fall akzeptiert Excel einen Vektor mit diskreten und einen mit numerischen Daten. Wird diese Option für die Intervalle gewählt, dann bilden die eindeutigen diskreten Werte Kategorien. Über diese Kategorien **summiert** Excel den numerischen Vektor, um die Höhe der Balken zu bestimmen. Das Ergebnis ist in den meisten Fällen kein Histogramm im eigentlichen Sinn, weil es eine Summe anstatt einer Anzahl anzeigt.

Wird als numerischer Vektor der Einsvektor (Kapitel 11)verwendet, dann entspricht die Summe der einzelnen Kategorien der Anzahl der Werte in der Kategorie. Ein solcher Einsvektor wird mit der Listing 16.1 erzeugt.

Listing 16.1 Einsvektor für Kategorien aus einer Tabelle

```
= 1 * (Tabelle1[disktrekeWerte] == Tabelle1[diskreteWerte])
```

Auf diese Weise lässt sich in Excel ein Histogramm für diskrete Daten erstellen (Abbildung 16.24).

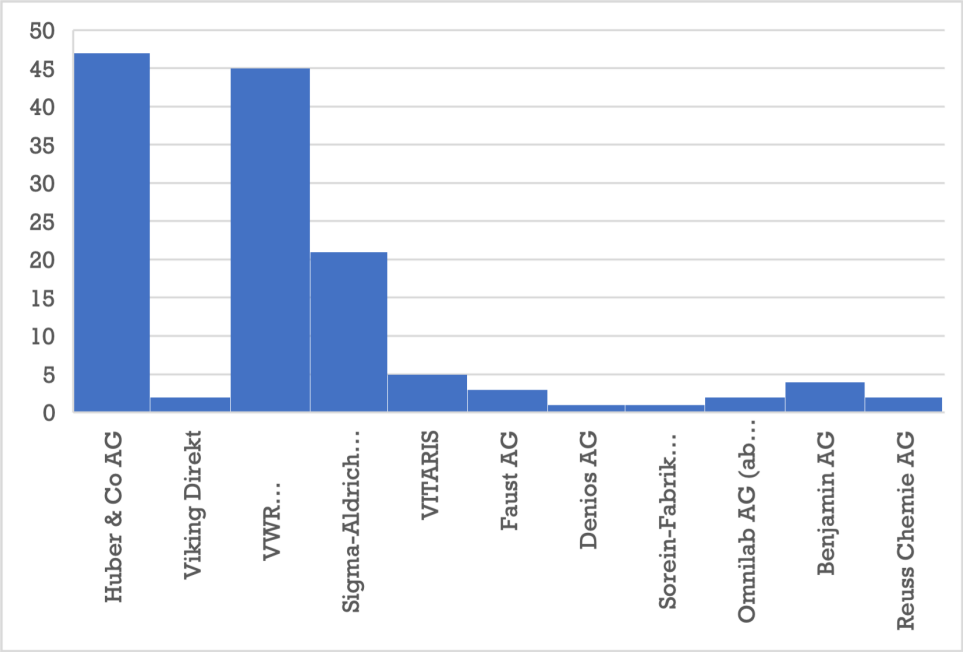


Abbildung 16.24.: Beispiel eines Histogramms diskreter Daten

Im Fall von *nominalskalierten Werten* ist es oft wünschenswert die Balken in der Reihenfolge der Häufigkeiten zu organisieren. Das ist mit der Histogrammvisualisierung von Excel **nicht** möglich. Für solche Visualisierungen muss auf ein reguläres Balkendiagramm zurückgegriffen werden und die Daten vor der Darstellung entsprechend aufbereitet werden.

16.2.4. Box-Plot

Boxplots oder *Kastendiagramme* dienen der Visualisierung von Verteilungen. Im Gegensatz zu den festen Intervallen von Histogrammen, mit denen Häufigkeiten bestimmt werden, wird für Boxplots von **festen Häufigkeiten** ausgegangen. Diese Darstellung erlaubt es, auch sehr unterschiedliche Verteilungen miteinander zu vergleichen.

Weil Boxplots auch für ordinalskalierte Wertebereiche erstellt werden können, sind sie ein flexibles Werkzeug zur Visualisierung von Verteilungen.

> **Merke**
>
> Für Boxplots **dürfen** Werte **nicht** aggregiert werden.

Werden einem Boxplot aggregierte Werte übergeben, dann bestimmt Excel die Intervallgrenze für die aggregierten Werte und nicht die ursprüngliche Verteilung.

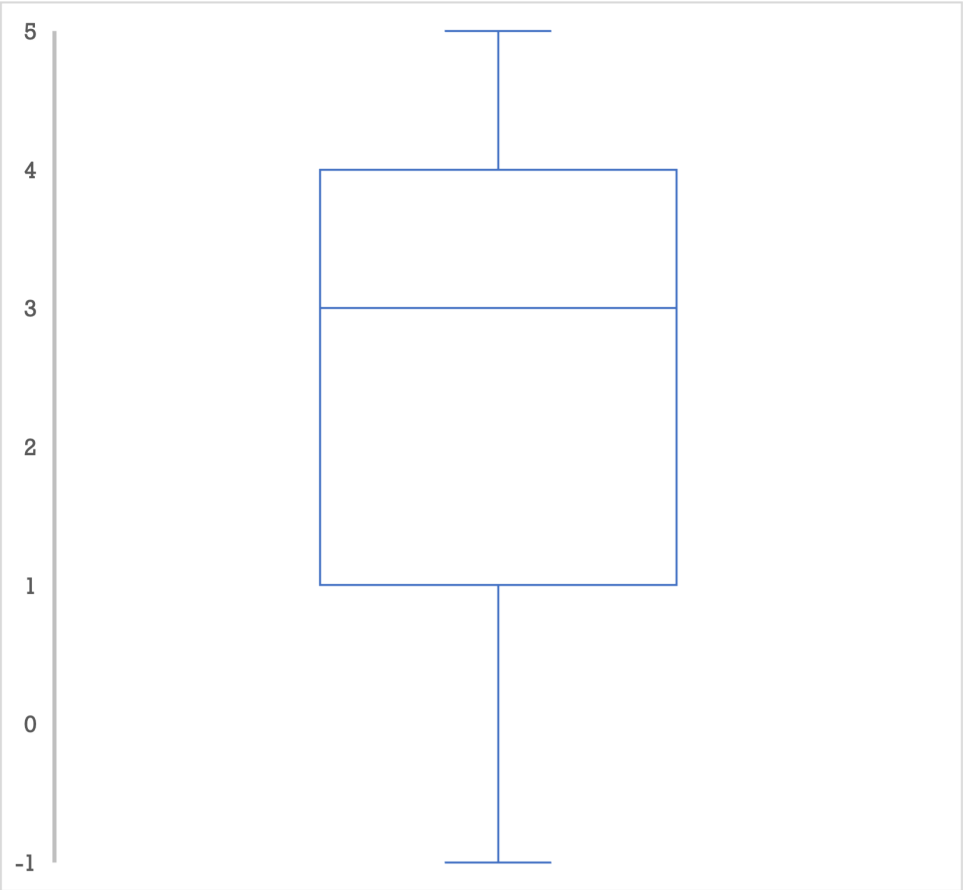


Abbildung 16.25.: Beispiel eines Boxplots

In der Regel wird eine ganze Arbeitsblattspalte ausgewählt, um alle relevanten Werte zu markieren. Anschliessend wird aus der Diagrammkategorie `Statistik` der Diagrammtyp `Kastendiagramm ausgewählt.`

Excel kann gruppierte Bloxplots erstellen. Dazu muss ein zweiter Vektor mit diskreten Werten angegeben werden. Excel gruppiert dann die Werte entlang des zweiten Vektors.

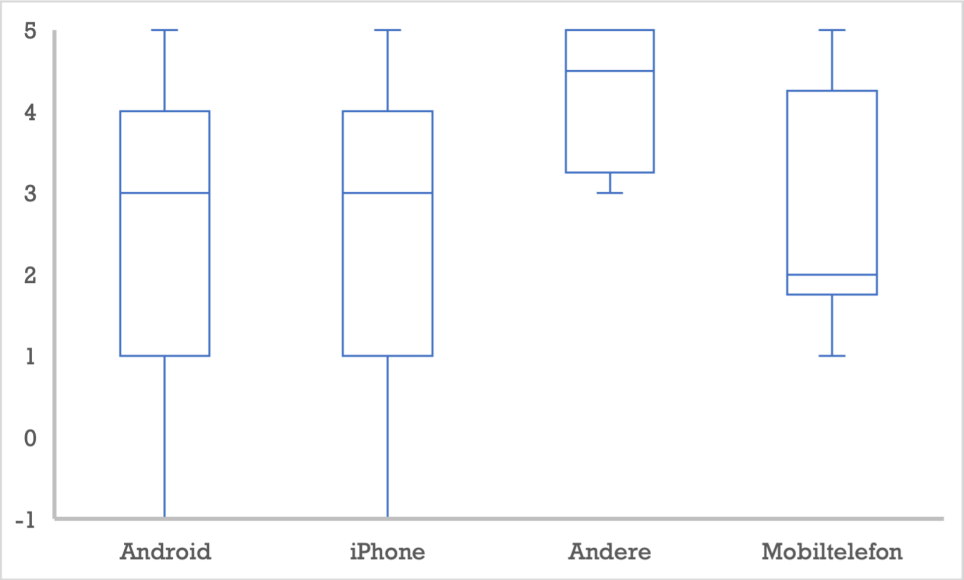


Abbildung 16.26.: Beispiel eines gruppierten Boxplots

16.2.4.1. Boxplots optimieren

Wie auch bei Balkendiagrammen fügt Excel zusätzliche Werte auf der Y-Achse ein. Diese Werte sind nur für offene Wertebereiche zulässig. Für geschlossene Wertebereiche müssen diese zusätzlichen Werte entfernt werden. Dazu wird die Beschriftung der Y-Achse ausgewählt und in der Datenreihenformatierung unter Optionen angepasst (Abbildung 16.27). Damit die Werte auch tatsächlich übernommen werden, muss die automatische Feststellung des Wertebereichs deaktiviert werden.

Die Breite der Box kann über die Datenformatierung der X-Achse festgelegt werden (Abbildung 16.28). Die Option `Abstandsbreite` gibt den Abstand der Box zur nächsten Box als Vielfaches der Boxbreite an. Der Abstand zum Diagrammrand ist die Hälfte des Abstands zur nächsten Box. Aus den Abständen berechnet Excel die Breite der Boxen.

- Ein Wert von `1.0` bedeutet, dass der Abstand genauso breit wie die Box ist.
- Ein Wert von `0.5` bedeutet, dass der Abstand halbso breit wie die Box ist. Dadurch wird die Box breiter.
- Ein Wert von `2.0` bedeutet, dass der Abstand doppelt so breit wie die Box ist. Dadurch wird die Box schmaler.

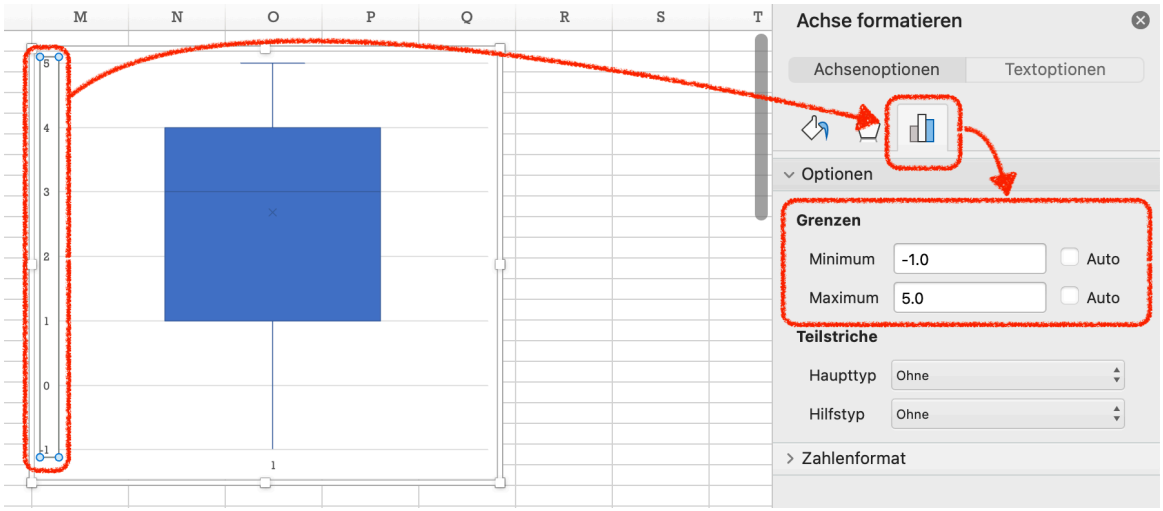


Abbildung 16.27.: Wertebereich der Y-Achse beim Boxplot anpassen

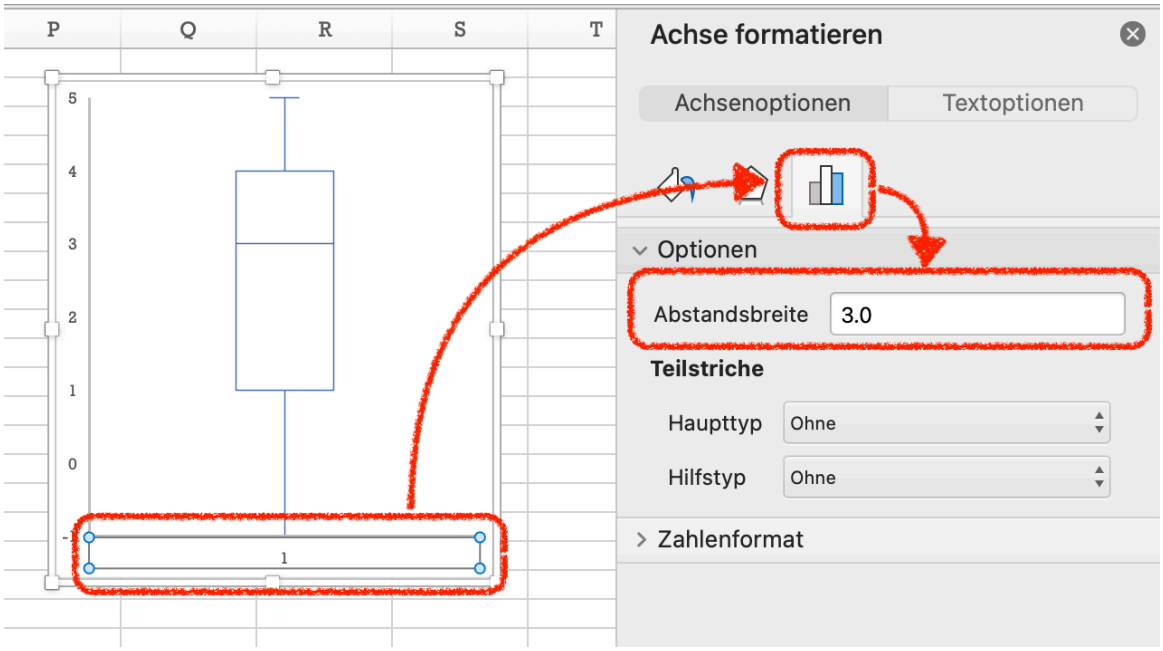


Abbildung 16.28.: Konfiguration der Breite eines Boxplots

> **Achtung**
>
> Werten mehrere Vektoren gleichzeitig als Boxplot dargestellt, werden die Boxen unmittelbar nebeneinander dargestellt. Diese Abstände können nicht konfiguriert werden (Abbildung 16.29)

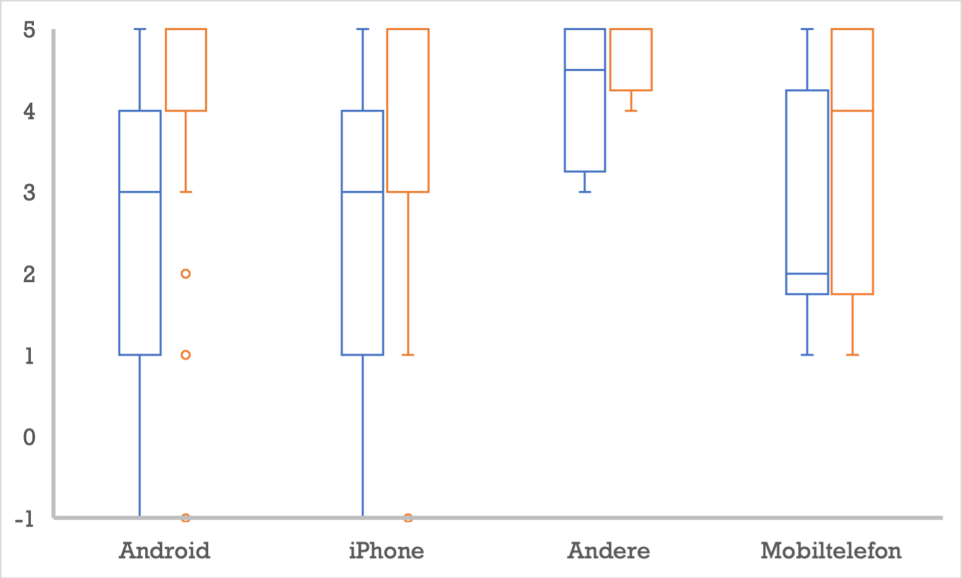


Abbildung 16.29.: Gruppierter Boxplot mit mehreren Vektoren (Blau und Orange)

16.2.5. Punktdiagramm

Ein Punktdiagramm stellt zwei Vektoren mit *kontinuierlichen Daten* gegenüber. Ein Vektor wird der X-Achse zugewiesen, der andere der Y-Achse. Entsprechend besteht jede Datenreihe für ein Punktdiagramm **immer** aus zwei Vektoren.

Werden mehr als zwei Vektoren für ein Punktdiagramm markiert, dann verwendet Excel standardmässig die äusserst linke Spalte *immer* für die Werte auf der X-Achse. Um dieses Verhalten zu ändern, müssen die Datenreihen unter `Daten auswählen` angepasst werden.

> **Praxis**
>
> Sollen mehrere Gruppen von Wertepaaren in einem Punktdiagramm dargestellt werden, sollten die Gruppen dem Punktdiagramm schrittweise als separate Datenreihen hinzugefügt werden.

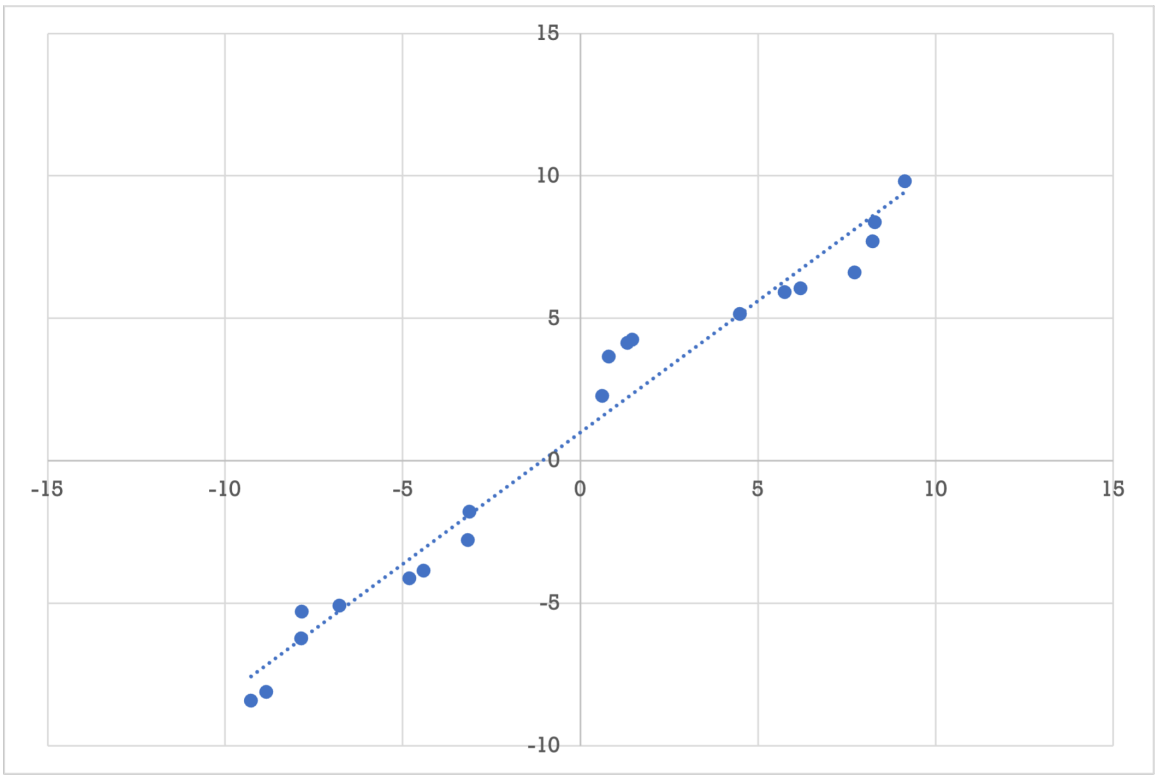


Abbildung 16.30.: Beispiel eines Punktdiagramms mit einer Ausgleichsgeraden

16.2.5.1. Ausgleichsgeraden

Ausgleichsgeraden heissen in Excel *Trendlinien*. Drei Arten von Trendlinien können direkt in ein Punktdiagramm eingebettet werden.

- Linear
- Lineare Prognose
- Gleitender Durchschnitt

Von diesen drei Trendlinienarten ist nur `Linear` eine Ausgleichsgerade im eigentlichen Sinn. Die `Lineare Prognose` ist identisch mit der Ausgleichsgeraden und unterscheidet sich nur dadurch, dass die X-Achse einen etwas grösseren positiven Wertebereich erhält.

Zusätzlich können die folgenden Ausgleichslinien konfiguriert werden.

- Exponentiell
- Logarithmisch
- Polynomisch
- Potenz

> **Warnung**
>
> Exponentielle, logarithmische und Potenzierte Ausgleichslinien sind nur für geeignete Wertebereiche ($\mathbb{R}^+$) zulässig

Die Trendlinie `Gleitender Durchschnitt` setzt voraus, dass die Werte auf der X-Achse aufsteigend sortiert sind. Diese Trendlinie wird meist nur dann verwendet, wenn die Werte auf der X-Achse Zeitangaben sind.

Eine Ausgleichsgerade wird über den Menübalken `Diagrammentwurf` das Untermenu `Diagrammelement hinzufügen` unter `Trendlinien` die Option `Linear` gewählt.

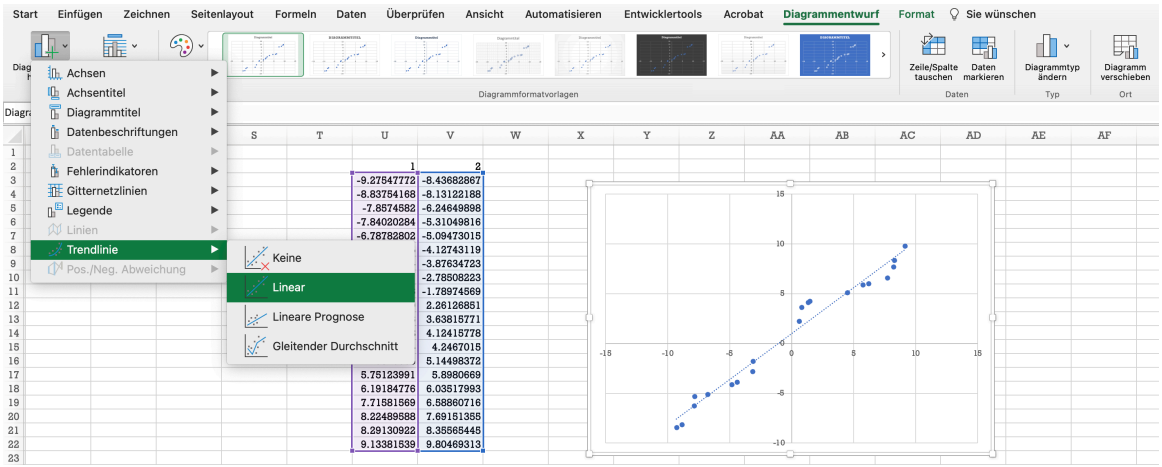


Abbildung 16.31.: Einfügen einer Ausgleichsgeraden

Eine Ausgleichsgerade kann angepasst werden, indem die Linie angelickt wird und unter `Trendlinie formatieren` die Trendlinienoptionen verändert werden. In diesem Dialog können auch die weiteren Ausgleichslinien konfiguriert werden (Abbildung 16.32).

16.2.6. Jitter-Diagramm

Excel stellt für diskrete Werte keine eigene Visualisierung für den Vektorenvergleich bereit. Eine direkte Anwendung des Punktdiagramms führt zum Problem, dass die Punkte für jedes Wertepaar genau übereinander liegen. So lassen sich die Verteilungen in der Visualisierung nicht ablesen. Um ein Jitter-Diagramm zu erzeugen, müssen die Werte vor der Visualisierung leicht angepasst werden, so dass die Punkte nicht exakt übereinander liegen. Für diese Anpassung müssen die Werte beider Vektoren als Zahlen vorliegen (s. Kapitel 13). Weil die Werte diskret sind, kann der Bereich zwischen den Werten genutzt werden. Grundsätzlich steht der Bereich bis 0.5 Abstand vom jeweiligen Wert für das Jittering zur Verfügung. Weil dadurch aber die Punkte nebeneinanderliegender Werte nicht mehr korrekt zugeordnet werden können, sollte zwischen zwei Werten eine *Pufferzone* vorgesehen werden, in der keine Punkte vorkommen dürfen. Dadurch wird eine visuelle Grenze zwischen den Werten erzeugt.

> **Wichtig**
>
> Bei Jitter-Diagrammen sollten zusätzlich angeführt werden, dass die Punkte von den tatsächlichen Werten abweichen.

> **Praxis**
>
> Die Pufferzone sollte bei ganzzahligen Werten .4 breit sein, damit eine ausreichend grosse visuelle Grenze wahrgenommen werden kann. Daraus ergibt sich ein Bereich von ± 0.3 um den tatsächlichen Wert für den Darstellungsbereich.

Für ein Jitter-Diagramm müssen alle Werte zufällig vom tatsächlichen Wert ein klein wenig verschoben werden. Damit vermieden wird, dass zu viele Werte exakt übereinander dargestellt werden, wird also für jeden Wert in den Daten ein eigener Zufallswert bestimmt. Dieser Wert muss im Intervall zwischen -0.3 und 0.3 liegen (s. Praxis-Box).

Der zweite Parameter ist immer `2`, weil eine X- und eine Y-Achse vorliegt. Anschliessend werden diese Verschiebungen auf die tatsächlichen Werte addiert. Diese neuen Werte bilden die Grundlage für das Diagramm, dass wie oben beschrieben als Punktdiagramm erstellt wird.

Beispiel 16.1 (Pufferwerte addieren)**.** Dieses Beispiel nimmt an, dass die beiden Vektoren an den Adressen A1 und B1 beginnen.

Im ersten Schritt werden die Werte für den Darstellungsbereich um den ursprünglichen Wert erzeugt.

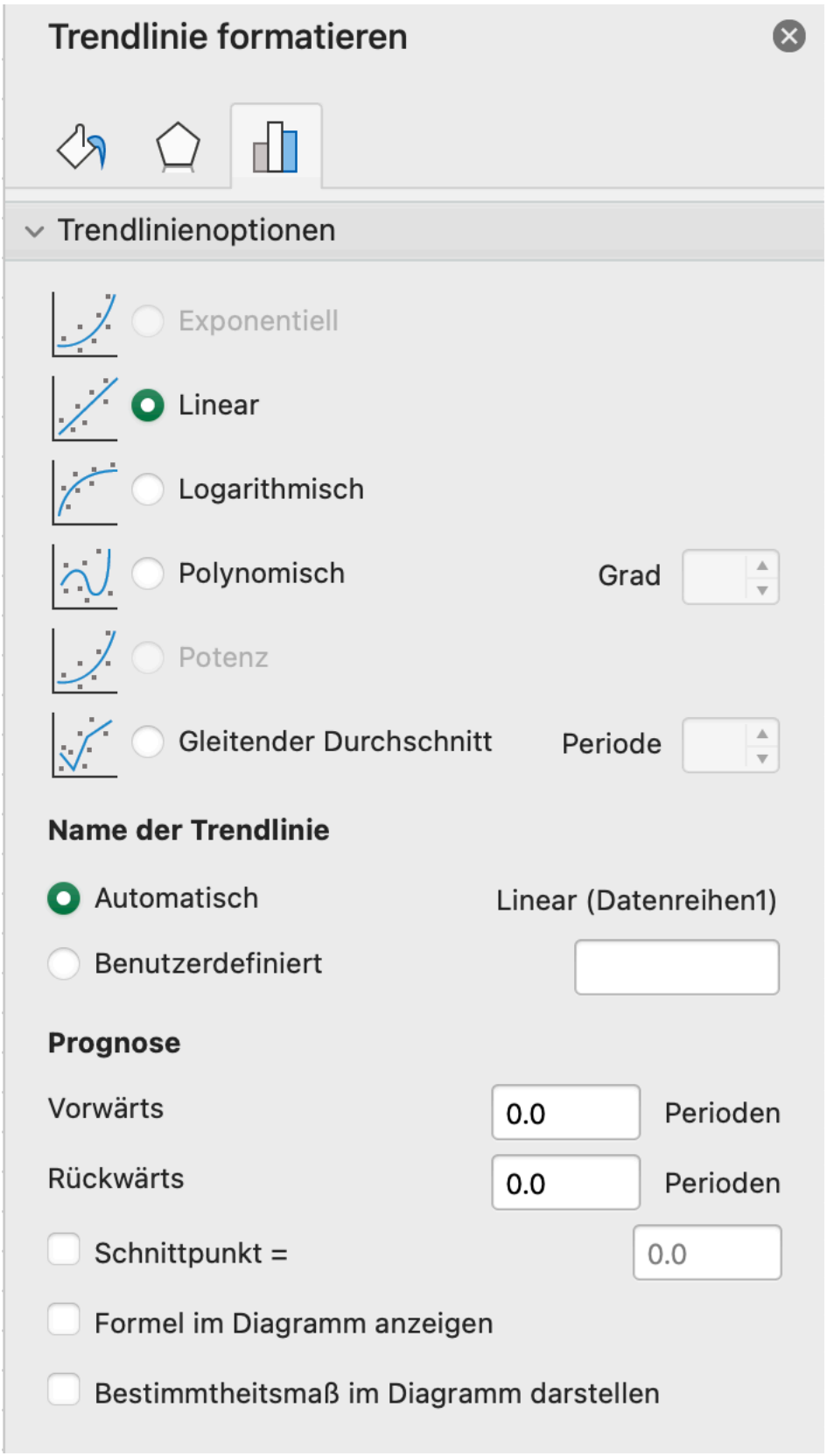


Abbildung 16.32.: Konfigurationsmöglichkeiten für Ausgleichsgeraden

```
= ZUFALLSMATRIX(zeilen(A1#); 2; -0.3; 0.3)
```

Im zweiten Schritt werden diese Verschiebungen auf die Werte addisert.

```
= A1#:B1# + D1#
```

Das Ergebnis ist ein Bereich, der visualisiert werden kann.

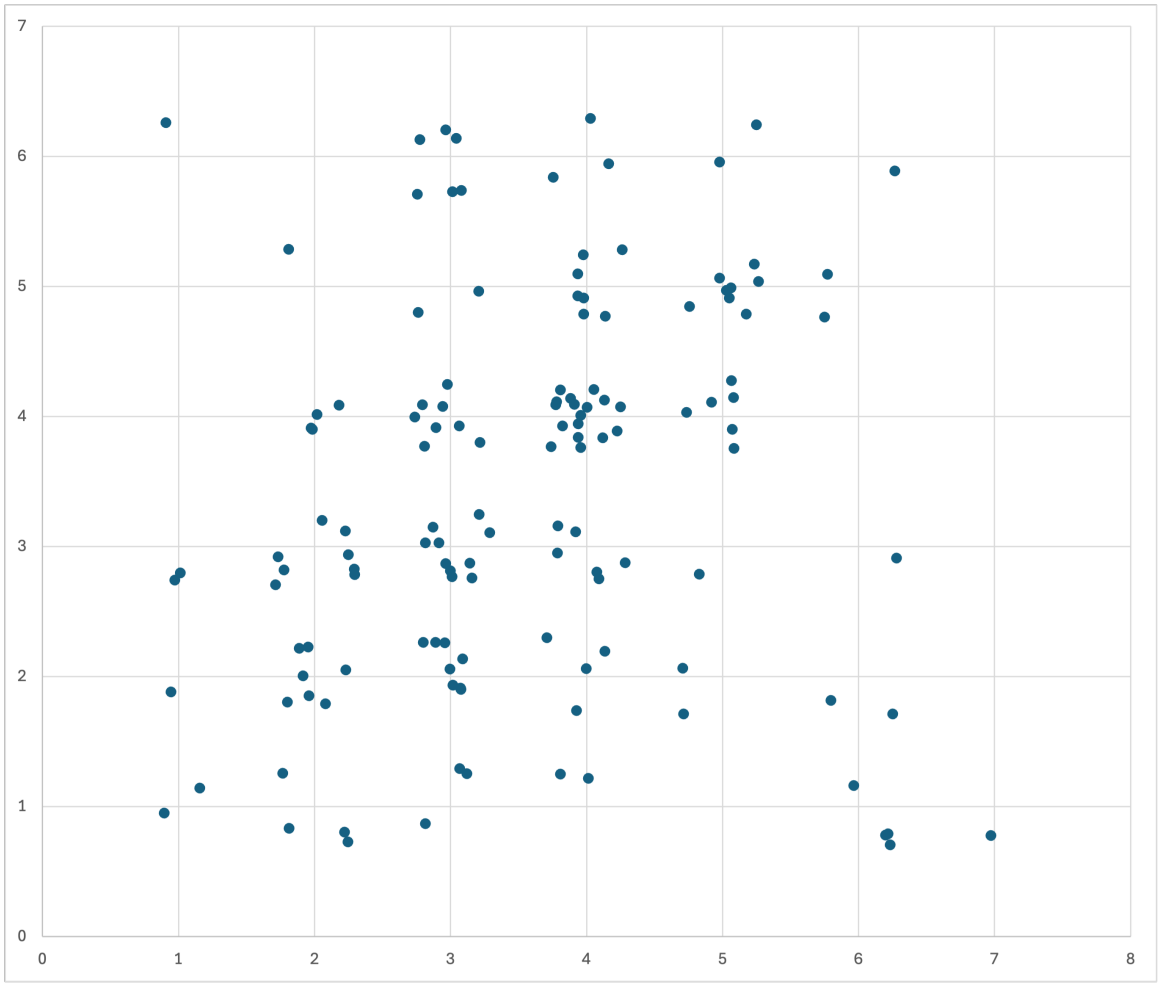


Abbildung 16.33.: Beispiel für ein Jitter-Diagramm

16.2.7. Blasendiagramm

> **Merke**
>
> Blasendiagramme sind Punktdiagramme mit drei dargestellten Merkmalen.

Für Blasendiagramme gelten die gleichen Optionen und Konfigurationen, wie für Punktdiagramme.

Weil drei Merkmale kodiert werden, müssen die Daten in drei Spalten vorliegen. Dabei wird die linke Spalte für die X-Achse verwendet. Die mittlere Spalte wird für die Y-Achse verwendet. Die rechte Spalte enthält die Werte für die Grösse der Punkte. Prinzipiell lassen sich die Werte nachträglich noch umorganisieren, oft ist es aber einfacher, die Spalten vorher anzuordnen.

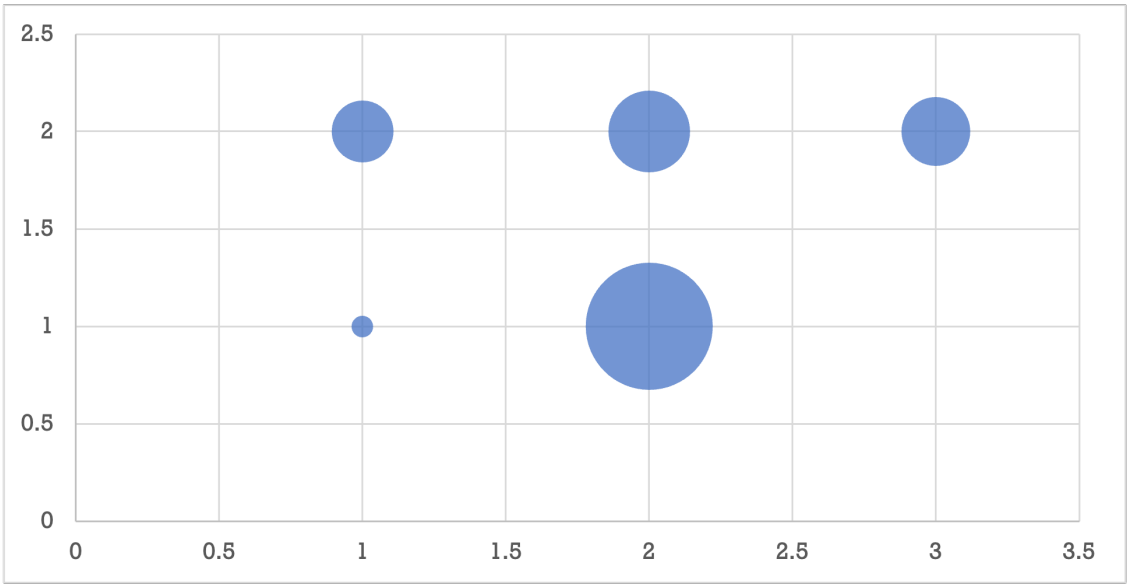


Abbildung 16.34.: Beispiel eines Blasendiagramms

Eine Datenreihe eines Blasendiagramms besteht aus drei Merkmalen.

In Blasendiagrammen lassen sich die Grössenwerte auf zwei Arten abbilden:

1. Über den Durchmesser der Kreise.
2. Über die Fläche der Kreise.

Wird der Durchmesser gewählt, werden die Kreise für grössere Werte schneller grösser als bei der Einstellung, über die Fläche zu kodieren. Die Kodierung über den Durchmesser ist sinnvoll, wenn nur ein kleiner Wertebereich (3-5) mit dicht zusammenliegenden Werten dargestellt werden muss. In allen anderen Fällen sollte die Kodierung über die Fläche der Kreise erfolgen.

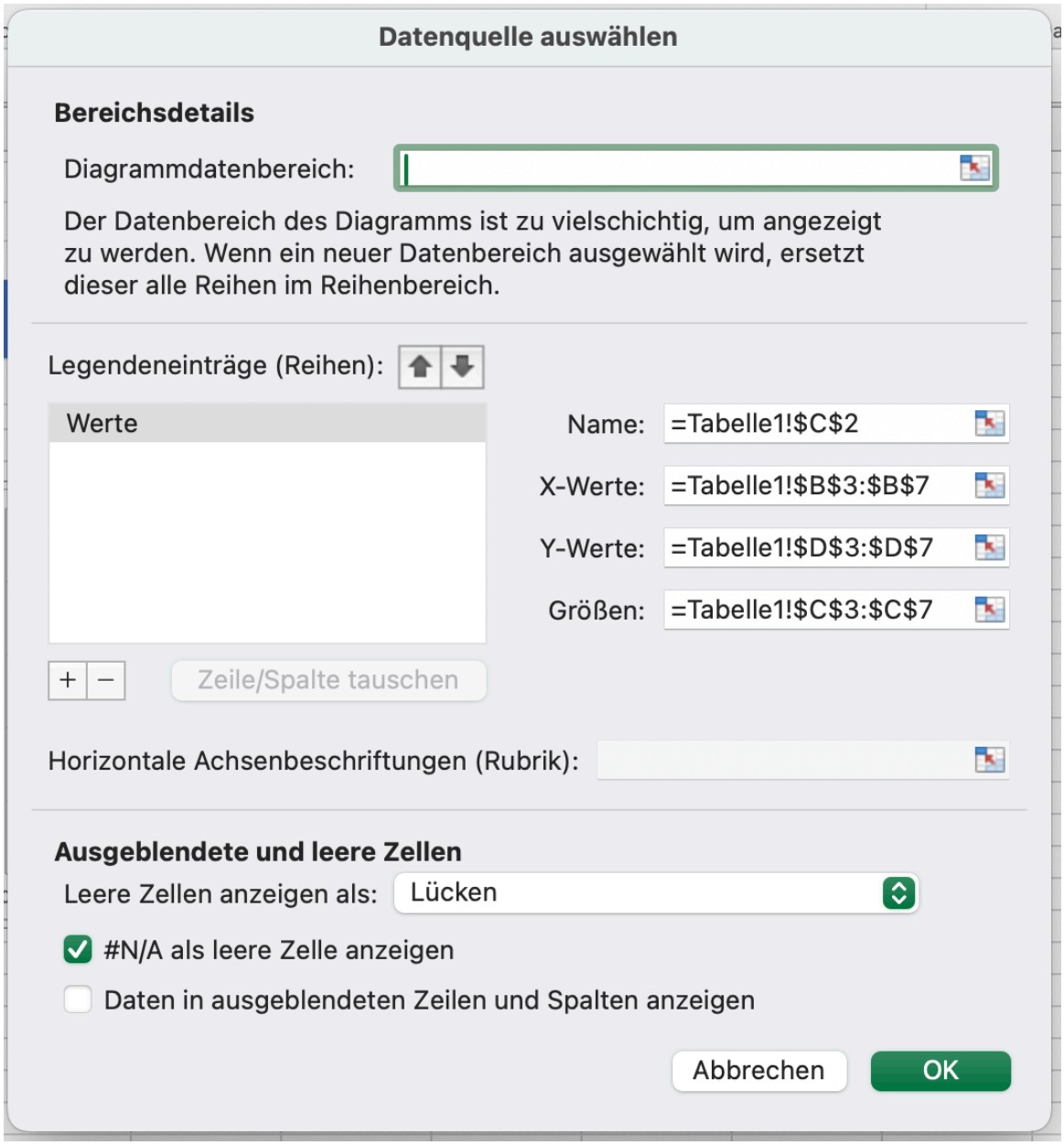


Abbildung 16.35.: Konfiguration der Datenreihen eines Blasendiagramms (MacOS)

Die Blasengrösse wird konfiguriert, indem zuerst auf einen Kreis mit der Maus geklickt wird und anschliessend unter Datenreihen formatieren unter `Reihenoptionen` die Strategie für die Blasengrösse angepasst wird (Abbildung 16.36)

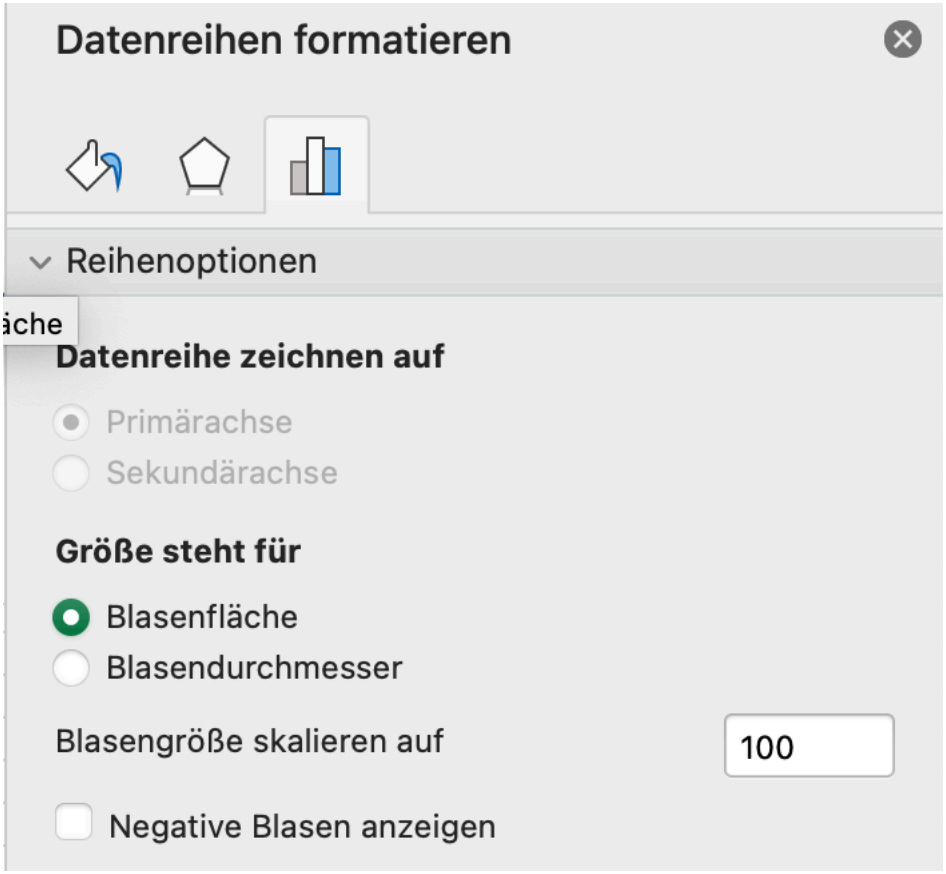


Abbildung 16.36.: Konfiguration der Blasengrösse

16.2.8. Linien-, Kreis- und Donutdiagramme

Linien-, Kreis- und Donutdiagramme sind in Excel Varianten von Balkendiagrammen: Sie verwenden das gleiche Format für die Datenreihen und werden ansonsten gleich konfiguriert. Wie bei Balkendiagrammen, wird die X-Achse als eine diskrete Datenskalierung behandelt. Die Daten können also nominal- oder ordinalskaliert sein.

> Praxis
>
> Kreis- und Donutdiagramme sind identisch mit Balkendiagrammen, wenn alle Werte **positiv** sind. Für mehr als drei Werte können die meisten Menschen sich diese Diagramme oft nicht richtig interpretieren. Deshalb sollten Kreis- und Donutdiagramme möglichst nur zur Darstellung (extremer) Mengenverhältnisse verwendet werden. Grundsätzlich ist ein Balkendiagramm einem Kreis- oder Donutdiagramm vorzuziehen.

(s. Abbildung 16.37 und Abbildung 16.38)

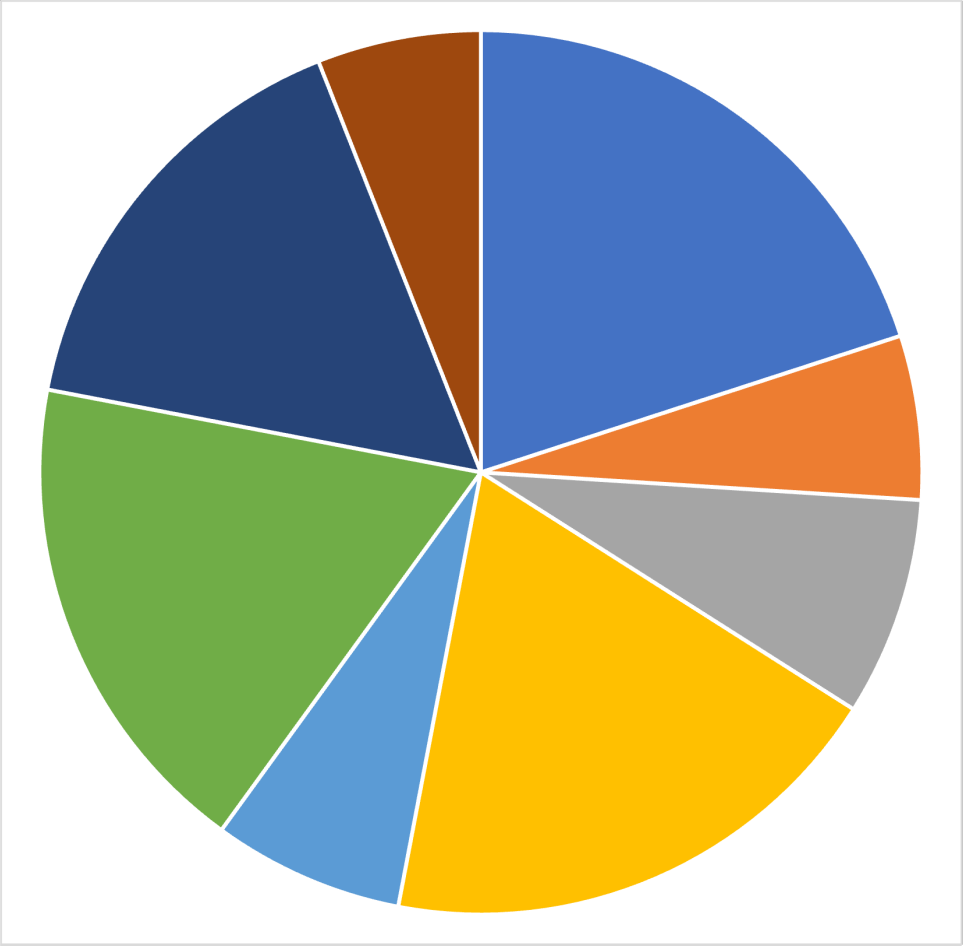


Abbildung 16.37.: Kreisdiagramm mit ähnlich grossen Werten

Excels **Liniendiagramme** zeichnen für jeden Wert auf der Y-Achse einen Punkt und verbinden die Punkte mit geraden Linien.

Achtung

Weil die X-Achse immer diskrete Daten abbildet, dürfen Liniendiagramme nicht mit Punktdiagrammen mit *interpolierten Linien* verwechselt werden.

Praxis

Excels Liniendiagramme sollten nur für sog. Paralleldiagramme verwendet werden, in denen mehrere diskrete Merkmale mit dem gleichen Wertebereich gegenübergestellt werden. Durch die Linien werden Unterschiede in den Ausprägungsprofilen sichtbar.

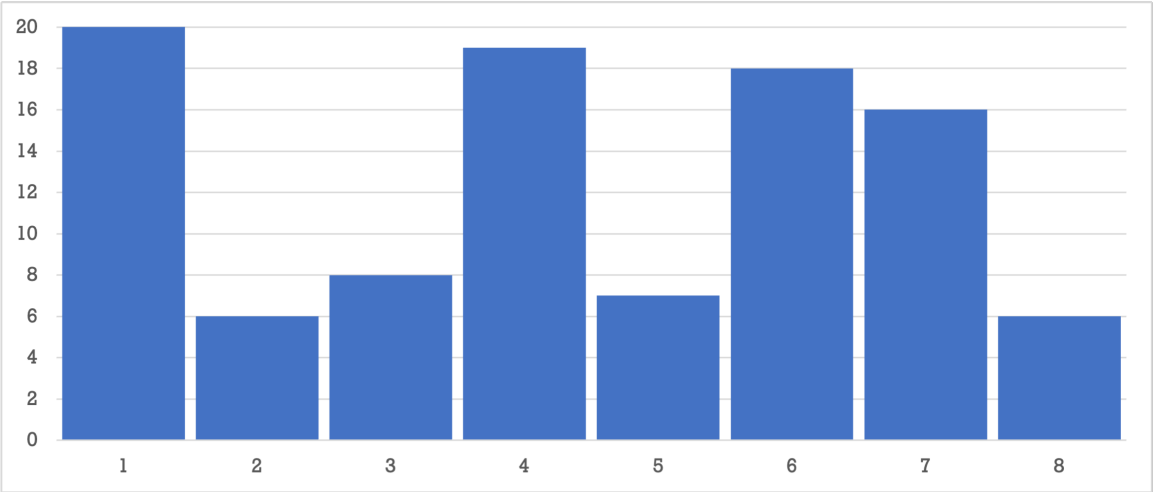


Abbildung 16.38.: Balkendiagramm für die gleichen Werte aus Abbildung 16.37

Abbildung 16.39 zeigt ein solches Paralleldiagramm. Dieses Beispiel macht die Grenzen dieses Diagrammtyps sichbar, weil Excel sich überlagernde Linien nicht nebeneinander darstellen kann.

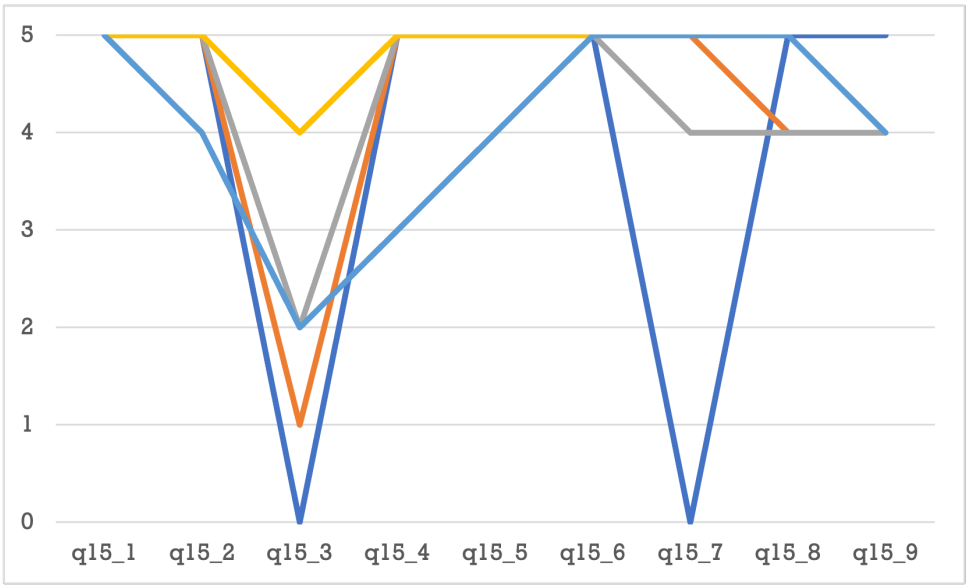


Abbildung 16.39.: Anwendung eines Liniendiagramms als Paralleldiagramm

Eine Variante von Paralleldiagrammen sind sog. Netzdiagramme (bzw. Spider-Web-Diagramme).

> **Merke**
>
> Netzdiagramme ordnen die Kategorien der X-Achse sternförmig an.

In Excel finden sich Netzdiagramme unter der Kategorie `Wasserfalldiagramme`. Diese Diagramme werden oft zum Vergleich von Merkmalsprofilen verwendet, z.B. als Ergänzung einer Stärken- und Schwächen-Analyse (SWOT-Analyse)

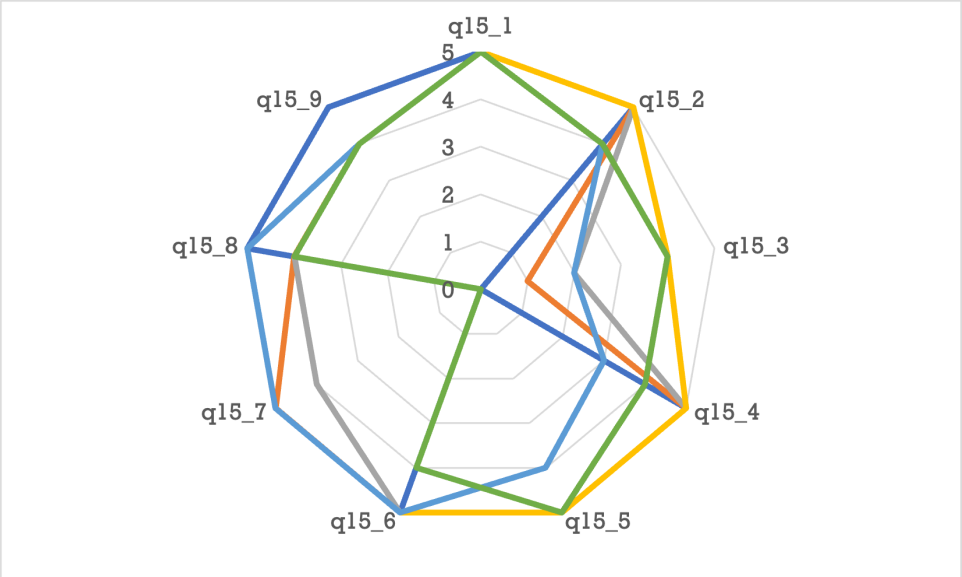


Abbildung 16.40.: Netzdiagramm mit den gleichen Daten wie Abbildung 16.39

16.3. Mathematische Funktionen visualisieren

> **Merke**
>
> Mathematische Funktionen werden immer über **Punktdiagramme** oder **Punkte mit interpolierten Linien** dargestellt.

Excel kann mathematische Funktionen nicht direkt visualisieren. Stattdessen müssen die Funktionen für einen gewünschten Wertebereich berechnet werden. Dazu werden Punkte auf der X-Achse ausgewählt und berechnet. Zwischen diesen Punkten berechnet Excel die Funktion nicht, sondern verbindet diese Punkte nur durch eine Linie. Diese Linie muss keine Gerade sein. Stattdessen versucht Excel eine Kurve so durch einen Punkt zu legen, sodass die Kurve keine Ecke hat. Die Folge dieses Verbinden von Punkten ist, dass je weniger Werte auf der X-Achse ausgewählt werden, desto ungenauer wird die Funktionsdarstellung.

Um eine mathematische Funktion mit Excel zu visualisieren, muss eine Wertetabelle erstellt werden. Diese Tabelle wird in den folgenden Schritten erstellt.

1. Die Unter- und Obergrenze des Darstellungsbereichs auf der X-Achse werden festgelegt.
2. Aus diesen Intervallgrenzen wird die Differenz gebildet, woraus sich die Länge des Intervalls ergibt.
3. Die Anzahl der Punkte in diesem Intervall festgelegt.
4. Die Länge des Intervalls wird durch die Anzahl der Punkte geteilt. Daraus ergibt sich der Abstand der Punkte im Intervall.
5. Es wird eine ***Sequenz*** mit einer Länge der Anzahl der Punkte plus Eins gebiltet. Der Startwert dieser Sequenz ist die Untergrenze des Darstellungsbereichs. Die Schrittweite entspricht dem Abstand der Punkte Abstand der Punkte.
6. Die mathematische Funktion wird als Formel in die Spalte rechts neben der Sequenz eingegeben. Die Werte der Sequenz werden als Argumente der mathematischen Funktion verwendet.

> **Hinweis**
>
> Die Sequenz für die Wertetabelle muss die Anzahl der Punkte *plus Eins* enthalten, damit sowohl die Untergrenze als auch die Obergrenze des Darstellungsbereichs als Argumente verwendet werden.

Sollen mehrere Funktionen über das gleiche Intervall dargestellt werden, dann wird der letzte Schritt für jede zusätzliche Funktion wiederholt werden. Das Intervall der *Funktionsargumente* bildet immer die erste Wertespalte. Anschliessend wird die gesamte Wertetabelle visualisiert.

Diese Wertetabelle kann als Punktdiagramm dargestellt werden. Dazu wird die gesamte Wertetabelle markiert und als **Punkte mit interpolierten Linien** visualisiert.

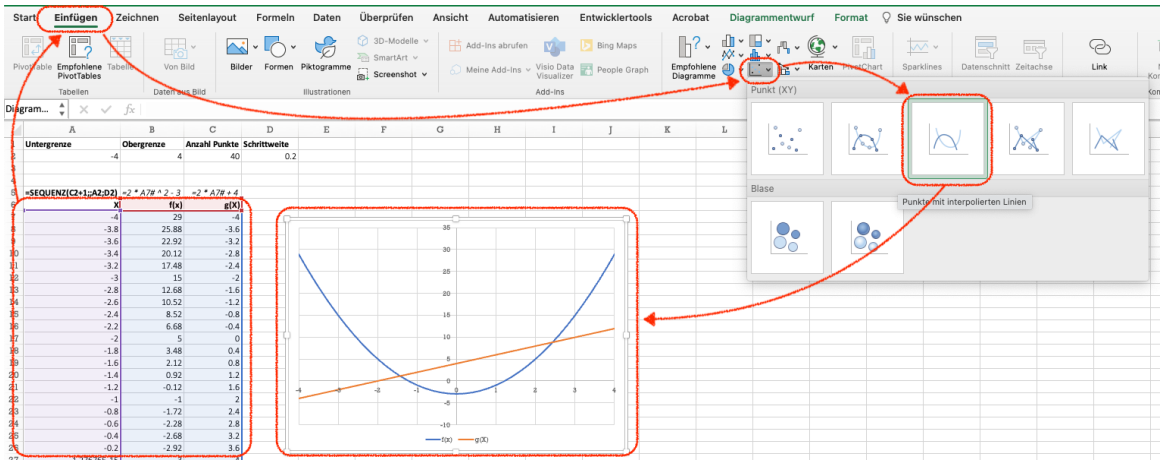


Abbildung 16.41.: Beispiel der Visualisierung der Funktionen $f(x) = 2x^2 - 3$ und $g(x) = 2x + 4$

Referenzen

Fisher, N. I., & Kordupleski, R. E. (2019). Good and bad market research: A critical review of Net Promoter Score. *Applied Stochastic Models in Business and Industry*, *35*(1), 138–151. https://doi.org/10.1002/asmb.2417

Grisaffe, D. B. (2007). *Questions about the ultimate question: conceptual considerations in evaluating Reichheld's Net Promoter Score (NPS). 20.*

Keiningham, T. L., Cooil, B., Andreassen, T. W., & Aksoy, L. (2007). A Longitudinal Examination of Net Promoter and Firm Revenue Growth. *Journal of Marketing*, *71*(3), 39–51. https://doi.org/10.1509/jmkg.71.3.039

Microsoft Support. (2023a). *Excel specifications and limits.* https://support.microsoft.com/en-gb/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3

Microsoft Support. (2023b). *Using structured references with Excel tables.* https://support.microsoft.com/en-gb/office/using-structured-references-with-excel-tables-f5ed2452-2337-4f71-bed3-c8ae6d2b276e

Reichheld, F. F. (2003). The One Number You Need to Grow. *Harvard Business Review.*