Einleitung, Überblick, Vorbereitungen

Worum geht es in diesem Modul?

Das Modul **"Angewandtes Informationsmanagement"** beschäftigt sich mit einem zentralen Element wirtschaftlichen Handelns, letztlich mit einem zentralen Element des menschlichen Handelns überhaupt: dem Entscheidungsprozess, genauer: seiner empirischen Basis. Die Fragen: *wie sollte in der gegebenen Situation entschieden werden?* und *wie wird typischerweise in einer solchen Situation entschieden?* berühren den Kern des Wirtschaftens. Basis für angemessenes Entscheiden sollte nach heutigem Verständnis in rationaler Weise verarbeitete Information sein[1]. **Information** wiederum ist nach rationalem Verständnis *evidenzbasiert*, gründet sich letztendlich also auf **Daten**[2].

Dass die jeweils vorliegenden Daten und damit die aus diesen gewonnene Information unvollständig, potenziell fehlerhaft oder verzerrt ist, stellt ebenfalls eine Grundtatsache des Wirtschaftens dar: *decision under uncertainty* bedeutet, dass Entscheidungen getroffen werden müssen, obwohl die eigentlich erforderliche Grundlage vollständiger und richtiger Information nicht vorliegt (oder noch nicht vorliegt, oder auch: prinzipiell nicht vorliegen kann).

Dieses Modul verfolgt den Weg von den Daten - ihrer Entstehung oder Erhebung, ihrer Speicherung und Verarbeitung - über die Information - durch grafische, statistische oder in anderer Weise algorithmische Verarbeitungsschritte aus den Daten erzeugt - bis hin zum Informationsmanagement, insbesondere zu den rationalen Entscheidungsprozessen[3] auf der Grundlage der dann zur Verfügung stehenden Information.

Voraussetzungen und Lernziele

Hauptsächliches Lernziel ist das Erlangen von *hands on*-**Erfahrung im Umgang mit** entscheidungsrelevanten **Daten** und der aus diesen Daten erzeugten **Information**. Es geht darum, die in der Praxis tatsächlich angewandten Methoden, Logiken und Techniken kennenzulernen, mit denen heutzutage Daten gespeichert, ausgewertet und zu Information verarbeitet werden. *Hands on* heißt in diesem Zusammenhang, dass nicht auf die Vermittlung theoretischer Grundlagen fokussiert wird - obwohl es ohne eine theoretische Basis natürlich niemals geht -, sondern auf das exemplarische Kennenlernen tatsächlich umgesetzter, einfacher Beispiele.

Darüberhinaus fassen wir das sehr wichtige Lernziel **algorithmisches Denken** ins Auge, insbesondere die Beziehungen algorithmischen Denkens zum Problemfeld Information und Entscheidung. Die Fähigkeit zu algorithmischem Denken ist über den eigentlichen Anwendungsbereich, eben das Ent-

[1] Das war nicht immer so. In vergangenen Zeiten war es normativ, vor schwerwiegenden Entscheidungen, etwa im Kriegsfalle, *Auguren* zu befragen, *Orakel* in Anspruch zu nehmen usw. Die Bedeutung dieser aus den modernen Entscheidungsprozessen verbannten Vorgehensweisen für den Entscheidungsträger früherer Epochen kann gar nicht hoch genug eingeschätzt werden. So galt das *Orakel von Delphi*, die wichtigste Weissagungsstätte der Antike, sogar als Mittelpunkt der Welt.

[2] Auch dies mag uns heutzutage selbstverständlich erscheinen, ist aber auch wieder Resultat einer geschichtlichen Entwicklung. In vielen Religionen gilt zudem nach wie vor, dass *geoffenbarten* Wahrheiten hohe Bedeutung zukommen kann.

[3] Eine sich unmittelbar anschließende, sehr spannende Frage ist, ob der Mensch *überhaupt* in der Lage ist, Informationen rational, umfassend und korrekt zu bewerten. Die Wirtschaftspsychologie identifiziert zahllose sogenannte *kognitive Dissonanzen*, mit denen sie begründet, dass diese Frage grundsätzlich verneint werden muss. Das führt in das Feld der *Behavioral Finance* mit hochinteressanten und -relevanten Ergebnissen an der Grenze zwischen Ökonomie und Psychologie - und leider aus dem hier gesteckten Rahmen hinaus. Einzig bei der Interpretation probabilistischer Information im Rahmen des *Bayesian Thinking* werden wir kurz auf diese Thematik zu sprechen kommen.

werfen von Algorithmen, hinaus in vielerlei Hinsicht sehr bedeutsam. Algorithmisch denken können heißt

> in der Lage sein, ein **komplexes Problem** in seine **Bestandteile** zu zerlegen, um dann ein sukzessives Vorgehen zur Problemlösung entwerfen zu können.

Algorithmisches Denken ist präzise, klar und genau. Durch das Einüben dieser Zugangsweise zu Problemen erwirbt man sich breit anwendbare Lösungskompetenzen.

Prinzipien und Vorgehensweisen

Um die Bedeutung von Daten und von daraus erzeugter Information wirklich zu **verstehen** und nicht nur oberflächlich zur Kenntnis zu nehmen, ist es sinnvoll und förderlich, selbst Hand anzulegen: das heißt, mittels praktischer Übungen selbständig auszuprobieren, wie man Daten sauber abspeichern und wieder auslesen kann, wie man mit statistisch gestützten Grafiken große Datenmengen zu kognitiv verarbeitbarer Information machen kann, wie man mit probabilistisch zu interpretierender Information umgehen sollte oder auch, wie man Geodaten so aufbereitet, dass raumbezogene Information daraus entsteht.

Es gibt heute natürlich **kommerzielle Computerprogramme** und Programmpakete, die den Entscheidungsträger und sein Team in diesen Aufgaben, oder in Teilen davon, effizient unterstützen. Die Qualität einer Entscheidung hängt aber aller Erfahrung nach sehr stark davon ab, ob der Entscheidungsträger sich über die Hintergründe im Klaren ist, die ihn zu dieser Entscheidung veranlasst haben. Man kann auf einer nach allen Regeln der UX designten Grafik einer BI-Anwendung nur dann weitreichende Entscheidungen fußen lassen, wenn man wirklich verstanden hat, unter welchen Bedingungen die dargestellten Werte und Verläufe gelten, welche Modellannahmen vorher gemacht wurden, welche Fehlergrenzen gelten, wie volatil die Information ist usw.[4]

> Die Einzelinformation allein ist für den Entscheider wertlos, wenn er nicht beurteilen kann, auf welcher fachlichen Basis, mittels welcher Methoden und unter welchen technischen Einschränkungen sie entstanden ist.

Gerade beim Thema Daten und Information heißt “verstehen” deshalb einmal mehr: **selber machen**! im Sinne von Herumprobieren, Ergebnisse betrachten, Schlüsse ziehen, Verändern, Testen, um diesen Zyklus dann wieder und wieder zu durchlaufen. Auch wenn die Arbeit des Entscheidungsträgers später nicht darin bestehen wird, selbst Auswerteskripte zu schreiben oder Datenbanken anzulegen, ist ein Hands On-Verständnis dieser Themen gleichwohl eine gute Grundlage für ein erfolgreiches Management des Teams und gleichzeitig des kompetent-kritischen Einsatzes seiner Ergebnisse im Entscheidungsprozess.

Um dieses sowohl tiefere wie auch praxisbezogenere Verständnis zu erreichen, verzahnen wir in diesem Modul drei Ebenen:

- Die **fachliche** Ebene: in welchem Teilprozess des Wirtschaftens befinden wir uns? Wer sind die Beteiligten, welche Aufgaben und Kompetenzen haben sie, welche Informationen brauchen sie für ihre Aufgaben zu welchem Zeitpunkt?

[4] Die jüngste *Finanzkrise* liefert zahllose Beispiele, wie nicht verstandene, also nicht kritisch hinterfragte Information in Entscheidungsprozesse eingebunden wurden und zu schweren, zum Teil katastrophalen Fehlentscheidungen geführt haben. Es genügt z.B. **nicht**, die Ratingnote einer ABS-Tranche zur Kenntnis zu nehmen! Es iat auch zwingend erforderlich, die Modellannahmen verstanden zu haben, auf deren Basis sie entstanden ist. In der kommenden Finanzkrise werden wir diesen Effekt mit Sicherheit erneut sehen.

- Die **methodische** Ebene: mit welchen Daten haben wir es zu tun? Woher stammen sie, wie wurden sie erhoben, wie vertrauenswürdig sind sie? Welche mathematischen und statistischen Verfahren müssen wir anwenden, um die auf der fachlichen Ebene benötigten Informationen aus diesen Daten zu erzeugen? Welcher Art sind die dadurch möglichen Aussagen und ihre Grenzen?
- Die **technischen** Fragestellungen: wie sind die Daten abgelegt, wie sind sie geschützt, wie können wir auf sie zugreifen? Wie ist das System aufgebaut, welches die Informationserzeugung realisiert? Wie sind die mathematischen und statistischen Methoden programmiertechnisch umgesetzt, wie geht das System technisch mit den Daten um, und was bedeutet dies für die Art der von dem System gelieferten Aussagen?

In diesem Modul werden wir fachliche Themen und mathematische Methoden gleichzeitig diskutieren, programmierend umsetzen, mit vielen Grafiken und selbsterzeugten Diagrammen anschaulich machen und mittels dieser Schleifen den individuellen Lernprozess moderieren. Die **Grundidee** hinter dieser Vorgehensweise ist, dass sich fachliche, methodische und technische Aspekte gegenseitig so stark durchdringen, dass man sie nicht für sich genommen studieren *kann*. Wie im Abschnitt "Voraussetzungen und Lernziele" deutlich formuliert, sind Programmierkenntnisse jedoch ausdrücklich *nicht* Voraussetzung für dieses Modul, auch nicht für die Hands On-Fallstudien. Kann das funktionieren, können wir die skizzierte Grundidee trotzdem umsetzen? Das geht - indem wir nicht *from scratch* programmieren, sondern mit für dieses Modul didaktisch aufbereiteten fertigen oder halbfertigen Programmskripten arbeiten. Wie das vor sich gehen soll, ist im Abschnitt "Vorbereitungen" weiter unten erklärt.

Aufbau des Moduls

Wie im vorangegangenen Abschnitt skizziert, werden in diesem Modul die inhaltlichen Fragestellungen mit den angewandten Methodiken und deren technischem Unterbau verbunden. Dies geschieht *nicht* so, dass wir zuerst einen Programmier-Crashkurs vorausschicken, anschließend einen mathematisch-methodischen Theorieteil bringen und dann zum Schluss eine Vorlesung über Entscheidungstheorie. Vielmehr werden alle drei Ebenen miteinander verzahnt. Mit Ausnahme der ersten beiden Abschnitte, in denen wir erstens das Problemfeld umreißen und zweitens Bekanntschaft mit der Programmiersprache Python machen, werden in allen Abschnitten Hands On-Übungen eingestreut sein, mittels derer das fachliche Problem vermittels konkreter Umsetzungen abgebildet wird.

1. Kapitel: Erste Schritte mit Python

- Einführung. Ein weiteres kleines Beispiel.
- Data Frames
- Fehlende Daten
- Auswertungen von Data Frames.

2. Kapitel: Datenmanagement 1 - Daten und Information

- Einleitung. Daten und Information
- Die Bedeutung der Datenintegrität
- Spreadsheets vs. Datenbanken?
- Rechtliche Anforderungen

- Fehlende und fehlerhafte Daten
- Datenbanken-Einsatz in der Praxis

3. Kapitel: Datenmanagement 2 - Das relationale Modell

- Einleitung - Technische Vorbereitungen
- Relationen
- Datenbanken anlegen und abfragen
- Eine einfache Datenbank selbst aufsetzen

4. Kapitel: Datenmanagement 3 - Speichern und Bereitstellen von Daten

- Einleitung - Technische Vorbereitungen
- Eine einfache Schnittstelle selbst bauen

5. Kapitel: Monte-Carlo-Verfahren

- Vorbemerkungen
- Präzision mittels Zufall?

6. Kapitel: Zufall vs. Struktur

- Vorbemerkungen
- Mustererkennung
- Aktien- und Indexcharts

7. Kapitel: Geodaten

- Technische Vorbereitungen
- Geodaten: raumbezogene Information

8. Kapitel: Kryptographie

- Einleitung
- Technische Vorbereitungen
- Grundlegende Begriffe
- Cäsaren und monoalphabetische Codes
- Monoalphabetische Codes brechen - Vorbereitungen
- Monoalphabetische Codes brechen - Statistiken
- Public Key-Verschlüsselung - Die Idee
- Public Key-Verschlüsselung - Das Untersummenproblem
- Public Key-Verschlüsselung - Die Idee
- Public Key-Verschlüsselung - Der Algorithmus

9. Kapitel: Fallen der Informationserzeugung

- Die Base Rate Fallacy
- Bayesian Thinking
- Der p-Wert und seine Schwierigkeiten

Vorbereitungen

Zum Erlernen der grundlegenden algorithmischen Fähigkeiten, die wir für den Aufbau wirklich funktionierender kleiner Anwendungen brauchen, ist das tatsächliche Arbeiten mit einer Programmiersprache essenziell. So wie man das Fechten nicht durch die Lektüre von Renzo Nostinis "Scherma di Fioretto" erlernen kann und das Kochen nicht durch noch so ausdauerndes Schauen von Fernseh-Kochsendungen, so gilt auch für das algorithmische Denken:

> Algorithmisches Denken erlernt man nicht durch Lesen.

Nun würde es den uns eingeräumten zeitlichen Rahmen (und mit guten Gründen auch die Geduld vermutlich der meisten Teilnehmer dieses Moduls) mehr als erschöpfen, wenn wir ein ausgebautes Curriculum der Kunst des Programmierens hier ausbreiten würden. Stattdessen orientieren wir uns tatsächlich an den angesprochenen Kochsendungen: der Teilnehmer des Kurses steht, das ist unverzichtbar, selbst am Herd, bekommt aber im Sinne des "*mise en place*" der französischen Restaurantküchen vieles präpariert und vorbereitet in Schüsselchen daneben gestellt. Stück für Stück können wir so lernen, ein kompliziertes Menü zu kochen, auch wenn wir den einen oder anderen Bestandteil zunächst einmal unverstanden hinnehmen müssen. Mit der Erfahrung kommt dann auch die zunehmende Selbständigkeit.

Ein für diese Zwecke nachgerade ideales Medium ist das **Jupyter Notebook**. Es handelt sich dabei um ein Dokumentensystem von sogenannten "Notebooks", basierend auf dem Datenformat `JSON`, welches mehreres zugleich kann:

- Die Erstellung (und auch das Teilen z.B. über GitHub) von Dokumenten in einem gewöhnlichen Webbrowser,
- das einfache Formatieren von Text und auch von mathematischen Formeln,
- sowie, und darauf zielen wir ab, das Schreiben und *interaktive Ausführen* von Computerprogrammen eingebettet in den Text, einschließlich der vom Programm erzeugten Grafiken.

Damit bietet sich das Jupyter Notebook für die Erstellung von Projektarbeiten an (insbesondere auch bei gemeinsamem Arbeiten), weil die textlichen Ausarbeitungen, der zugehörige Programmcode, die Berechnungsergebnisse und die grafischen Auswertungen eine geschlossene Einheit bilden. Für die Handreichungen dieses Kurses nutzen wir ebenfalls Notebooks. So kann der Leser die "*mise en place*"-Schüsselchen unmittelbar in seine eigenen Hände nehmen, verändern, seine Änderungen ausprobieren und ihre Auswirkungen direkt beobachten usw. Auch der hier vorliegende Einführungstext ist im übrigen nichts anderes als ein Notebook respektive sein PDF-Download.

Wer bereits Programmiererfahrung hat und gerne "seine" IDE (*Integrated Development Environment*, Integrierte Entwicklungsumgebung) weiterbenutzen möchte, kann dies problemlos mit dem Notebook kombinieren.

Die Programmiersprache, derer wir uns im Notebook bedienen wollen, ist **Python**, eine universelle interpretierte höhere Programmiersprache (das Jupyter Notebook unterstützt jedoch noch zahlreiche weitere Sprachen). Sie stellt in Form sogenannter Pakete sehr viele, vor allem auch statistische

Funktionalitäten zu Verfügung, die unmittelbar genutzt werden können - auch eine Form des "*mise en place*". Vielleicht hat der eine oder andere Kursteilnehmer Erfahrungen mit der Statistik-Skriptsprache "*R*" gesammelt. In diesem Fall wird er sich, wenn wir zur Benutzung des Python-Pakets `pandas` mit seinen Data Frames kommen, stark an *R* erinnert fühlen. Im übrigen könnten wir alles, was wir im Zuge des Aufbaus unseres Ratingverfahrens programmieren, ebensogut in *R* erstellen. Die Wahl von Python ist in diesem Sinne eher eine Frage des Geschmacks. Aber auch der verwöhnte Geschmack wird sich überzeugen lassen von der hohen Qualität der Auswertungen und Grafiken, die Python ermöglicht, und die denen von *R* in nichts nachsteht.

Was ist zu tun, damit jeder Kursteilnehmer zu seinem Jupyter kommt, die Notebooks der Handreichungen lesen und ausführen und eigene Notebooks erstellen kann? Eine bequeme Möglichkeit, einen Python-Interpreter zu installieren, führt über den Download der Anaconda-Distribution (https://www.anaconda.com/products/individual). Man verfügt dann über Python 3, hunderte von Paketen (die also nicht eigens nachgeladen werden müssen), eine gut ausgestattete IDE namens "Spyder" und eben über das Jupyter Notebook. Auf der Website finden sich Installationsanleitungen für die gängigen Systeme (https://docs.anaconda.com/anaconda/install/). Firefox, Chrome und Safari arbeiten sicher problemlos mit Jupyter zusammen; wer einen anderen Browser bevorzugt, möge diesen ausprobieren.

Der Umgang mit Notebooks ist nicht schwierig. Es gibt Zellen, die für die Eingabe von Python-Code gedacht sind, und Zellen, die Text als "Markdown" erwarten, einer einfachen Beschreibungssprache, die zur Dokumentation von Programmcode entwickelt wurde. Sehr praktisch für die Erstellung wissenschaftlicher Papiere ist, dass LaTeX direkt innerhalb der Markdown-Zellen eingebunden werden kann. Auch kann ein PDF-Download mit einem Klick via LaTeX erfolgen - wie beim vorliegenden Dokument. Ein Export direkt nach LaTeX ist natürlich ebenfalls möglich, wenn eigene Style Files zur Anwendung kommen sollen. Das vorliegende Einführungsdokument zum Beispiel besteht ausschließlich aus Markdown-Zellen. Man schreibt in das Notebook, indem man einfach eine Zelle hinzufügt und festlegt, ob es sich um eine Python- oder um eine Markdown-Zelle handeln soll. Gute Einführungen und Tutorials für die Benutzung des Jupyter Notebook sind über seine `Help`-Schaltfläche in der Menüzeile direkt abrufbar. Im Netz findet sich zusätzlich eine Legion von Tutorials, Handbücher, Erklär-Videos usw., etwa kostenlos, aber einschreibepflichtig bei udemy: https://www.udemy.com/course/jupyter-notebook-for-beginners/ Recht brauchbar als "*cheat sheet*" für die ersten Schritte mit dem frisch installierten Jupyter ist auch das von IBM: https://www.ibm.com/support/knowledgecenter/SSHGWL_1.2.3/analyze-data/markd-jupyter.html.

Auch bei anaconda selbst, unter https://anaconda.cloud/tutorials/ kann man ein gutes, etwa 15minütiges Video zur Einführung anschauen, muss zuvor aber einen Account anlegen.