Ludwig-Maximillians-Universität
München

Fakultät für Mathematik,Informatik und Statistik

Institut für Statistik

Multivariate Grafiken für PLS Pfadmodelle

Bachelor Arbeit

München, den 26.7.2011

Betreuer: Prof. Dr. Helmut Küchenhoff
Dipl.-Stat. Armin Monecke

vorgelegt von: Peter Mayer

Inhaltsverzeichnis

1 Einleitung **2**

2 Einführung in die Strukturgleichungsmodelle **3**

2.1 Bedeutung von Strukturgleichungsmodellen 3

2.2 Aufbau eines Strukturgleichungsmodell 3

2.2.1 Das Strukturmodell 5

2.2.2 Das Messmodell 6

3 Schätzung der Parameter - Der PLS-Algorithmus **8**

4 Multivariate Grafiken **12**

4.1 Multivariate Grafiken - Übersicht 12

4.2 Multivariate PLS Pfadmodell Grafiken 13

4.2.1 ggplot2 14

4.2.2 Beispiele für multivariate Grafiken in PLS Pfadmodellen . 19

5 Fazit / Ausblick **30**

Anhang **33**

A R-Code **33**

B CD-ROM **51**

Kapitel 1

Einleitung

Ein typisches Problem eines jeden Statistikers ist es eine Antwort auf die Frage zu finden: "Wie stelle ich meine Daten am besten grafisch dar?". Natürlich gibt es bereits eine Vielzahl an vorgefertigten Grafiken, die es dem Anwender ermöglichen seine Daten passend zu visualisieren, es wird aber mit einer zunehmenden Anzahl an Variablen in den Daten immer schwieriger Grafiken zu finden, die den Betrachter die gewünschten Informationen liefern und gleichzeitig noch übersichtlich, sowie einfach zu handhaben sind. Vor diesem Hintergrund rücken jetzt in dieser Arbeit PLS Pfadmodelle in den Fokus. Bei PLS Pfadmodellen bzw. Strukturgleichungsmodellen[1] steht man häufig vor dem beschriebenen Problem die passenden Grafiken zu finden, da in diesen Modellen häufig eine hohe Anzahl an Variablen zu visualisieren sind, die alle direkt oder indirekt in einem Zusammenhang stehen.

In dieser Arbeit wird zum besseren Verständnis von PLS Pfadmodell Grafiken, zuerst die Idee hinter PLS Pfadmodellen erklärt und anschließend werden Beispiele für multivariate Grafiken in Bezug auf PLS Pfadmodellen vorgestellt.

[1]Pfadmodelle sind ein Spezialfall von Strukturgleichungsmodellen

Kapitel 2

Einführung in die Strukturgleichungsmodelle

2.1 Bedeutung von Strukturgleichungsmodellen

Strukturgleichungsmodelle oder kurz SEM (englisch: *Structural Equation Models*) stellen einen Ansatz dar simultane Beziehungen zwischen einer Anzahl von Variablen zu untersuchen. Es sollen also mit Hilfe dieses Modells theoretisch hergeleitete Zusammenhänge anhand von empirischen Daten überprüft werden. Wie in multiplen Regressionsmodellen kann man die Variablen aufgrund ihrer Ursache-Wirkungsbeziehung im SEM in exogene und endogene Variablen unterteilen, aber mit den Unterschied, dass Variablen im SEM exogen und sowohl endogen zugleich sein können. Man spricht dann von *intervenierenden* Variablen oder Moderator Variablen.

Strukturgleichungsmodelle finden vor allem im wirtschafts- sowie sozialwissenschaftlichen Kontext Anwendung. Für beide Wissenschaften typisch spielen neben beobachtbaren Variabeln auch unbeobachtbare bzw. schwer messbare Variablen eine große Rolle in der Datenanalyse. Das SEM berücksichtigt diesen Umstand, indem es die Variablen zusätzlich zu ihrer Ursache-Wirkungsbeziehung auch nach ihrer Beobachtbarkeit in *latente* (unbeobachtbare) und *manifeste* (beobachtbare) Variablen unterteilt.

2.2 Aufbau eines Strukturgleichungsmodell

Strukturgleichungsmodelle bestehen aus zwei Teilen, dem Struktur- und dem Messmodell. Das Strukturmodell bildet die Zusammenhänge zwischen verschieden latenten Variablen in einem Pfaddiagramm ab, wohingegen die Messmodelle die Beziehung zwischen den latenten Variablen und ihren manifesten Variablen wiedergeben. (Nitzl, 2010)

Das folgende Beispiel zum Aufbau eines Strukturgleichungsmodell ist aus Schlittgen 2009, Kap. 19; es illustriert die Beziehung zwischen Ausgaben für Schulen in den USA und dem Erfolg der Schüler. Der Datensatz stammt aus einer Publikation des US-Ministeriums für Erziehung und Wissenschaft aus dem Jahr 1997. Die Leistung wurde mittels der erreichten Scorewerte beim SAT gemessen; dies ist ein College-Eignungstest für Highschool Absolventen. Es wurden folgende Variablen gemessen:

Ausgaben	Aufwendungen pro Schüler im Durchschnitt der täglichen Anwesenheit der Schüler, 1994-1995 (in Tsd US-Dollar)
Verhältnis	Durchschnittliches Schüler-Lehrer-Verhältnis, Herbst 1994
Vergütung	Geschätztes durchschnittliches jährliches Gehalt der Lehrer an diesen Schulen, 1994-1995 (in Tsd US-Dollar)
Teiln	Anteil derjenigen die am SAT teilgenommen haben bzgl. aller, die dazu berechtigt waren.
Verbal	Durchschnittlicher verbaler SAT-Score,1994-1995
Mathe	Durchschnittlicher SAT-Score in Mathematik, 1994-1995
Total	Durchschnittlicher Gesamt-SAT-Score,1994-1995

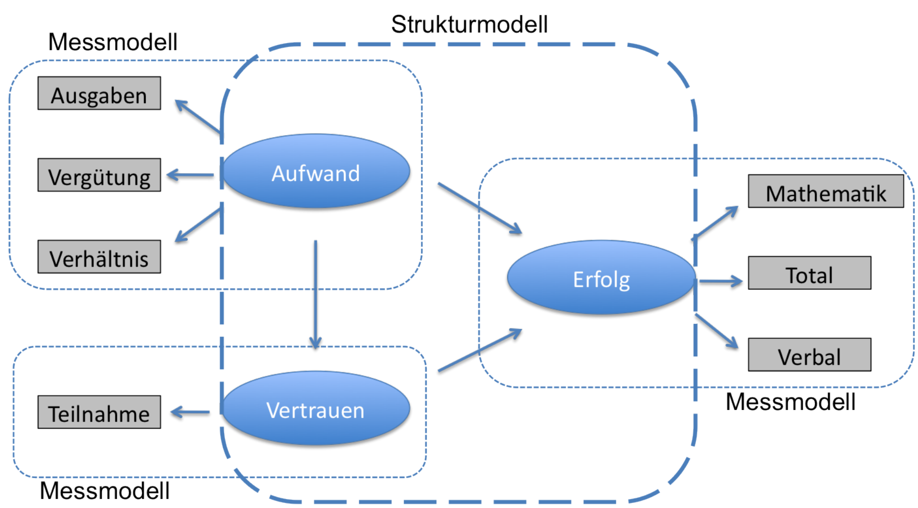

Abbildung 2.1: Strukturgleichungsmodell

Wie man an Abbildung 2.1 erkennt besteht das Strukturmodell bzw. innere Modell aus den drei latenten Variablen Aufwand, Erfolg und Vertrauen. Der Erfolg der Schüler (endogen) hängt von den Variablen Aufwand (exogen) und Vertrauen (exogen) ab. Würde man es bei diesen Einflussgrößen belassen, wäre dies eine typische Ausgangssituation für ein multiples Regressionsmodell. Allerdings hängt die Variable (Selbst-)Vertrauen zusätzlich noch vom Aufwand ab, da davon auszugehen ist, dass bei insgesamt besserer Ausgangsbedingungen auch

das Selbstvertrauen bzgl. Bestehens des Tests steigt. Somit ist die Variable Vertrauen nun intervenierend.
Als nächstes müssen nun geeignete messbare Variablen, auch Indikatoren genannt, gefunden werden, welche die unbeobachtbaren Variablen beschreiben sollen. Dies geschieht in den jeweiligen Messmodellen. In unserem Beispiel würde somit eine Veränderung des Aufwands sich in den Variablen Ausgaben, Verhältnis und Vergütung niederschlagen. Eine Veränderung des Selbstvertrauens hat im Modell einen Einfluss auf die Anzahl der Teilnehmer des Tests und der Erfolg der Schüler wird mit den SAT-Scores messbar gemacht. Die latente bzw. manifesten Variablen werden allgemein in Ellipsen bzw. Rechtecken dargestellt.
Im nächsten Schritt wird nun genauer auf das Struktur- sowie Messmodell und den zugrunde liegenden Gleichungen eingegangen.

2.2.1 Das Strukturmodell

Wir verwenden weiterhin das Strukturgleichungsmodell aus Kapitel 2.2.Wie bereits erwähnt bildet das Strukturmodell den Zusammenhang zwischen den latenten Variblen ab, die *Pfadkoeffizienten* β_{31},β_{32} und β_{21} geben die Stärke der Beziehungen zwischen den 3 Variablen an.

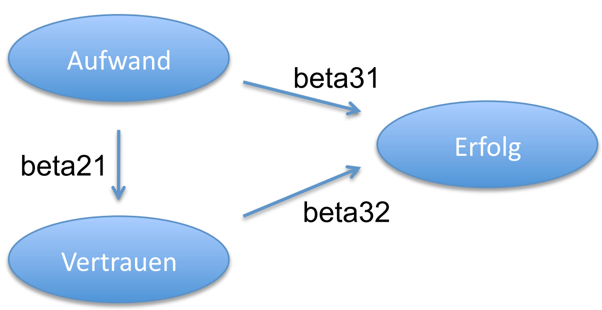

Abbildung 2.2: Strukturmodell

Wenn man davon ausgeht, dass die Beziehungen zwischen den Variablen linear sind, lauten die Modellgleichungen aus unserem Strukturmodell:

$$\begin{aligned} Aufwand &= \zeta_1 \\ Vertrauen &= \beta_{21} * Aufwand + \zeta_2 \\ Erfolg &= \beta_{31} * Aufwand + \beta_{32} * Vertrauen + \zeta_3 \end{aligned}$$

Der Intercept kann bei diesen Gleichungen vernachlässigt werden, da die Zielgrößen latent sind und somit bereits passend skaliert sind. Die Variable Aufwand ist im Strukturmodell exogen, daher ist sie gleich dem Fehlerterm ζ_1. In Matrixschreibweise ausgedrückt ergibt sich dann:

$$\eta = B\eta + \zeta$$

Wobei η für die latenten Variablen (Aufwand,Vertrauen und Erfolg) steht. Die Matrix B gibt die Pfadkoeffizienten zwischen den latenten Variablen wieder. Für den Residuenvektor ζ gilt:

$$E(\zeta) = 0$$

Damit sieht unsere Strukturmodellgleichung wie folgt aus:

$$\begin{pmatrix} Aufwand \\ Vertrauen \\ Erfolg \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \beta_{21} & 0 & 0 \\ \beta_{31} & \beta_{32} & 0 \end{pmatrix} \begin{pmatrix} Aufwand \\ Vertrauen \\ Erfolg \end{pmatrix} + \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \zeta 3 \end{pmatrix}$$

2.2.2 Das Messmodell

Jede latente Variable in einem Strukturgleichungsmodell wird durch manifeste Variablen beschrieben und besitzt somit ein eigenes Messmodell. Außerdem werden Messmodelle noch nach *refliktive* und *formative* Messmodellen unterschieden. Je nachdem in welcher Ursache-Wirkung Beziehung die manifesten und latenten Variablen zueinander stehen, handelt es sich um ein reflektives oder formatives Messmodell. Folgende Grafik verbildlicht diesen Sachverhalt andhand der latenten Variable *Trunkenheit*:

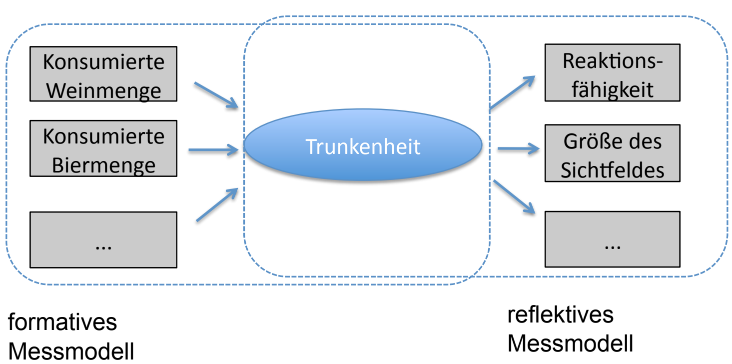

Abbildung 2.3: Messmodelle Trunkenheit; links formativ, rechts reflektiv

In unserem Strukturgleichungsmodell sind alle Messmodelle reflektiv, daher werden wir uns auf reflektive Messgleichungen beschränken. Die Messgleichungen für die latente Variable Erfolg lauten somit:

$$\begin{aligned} Mathematik &= \lambda_{11} Erfolg + \epsilon_1 \\ Total &= \lambda_{12} Erfolg + \epsilon_2 \\ Verbal &= \lambda_{13} Erfolg + \epsilon_3 \end{aligned}$$

λ_{11},λ_{21} und λ_{31} sind die Regressionskoeffizienten, die den Ausmaß des Zusammenhangs zwischen den manifesten Variablen und der latenten Variable anzeigen; ϵ_1,ϵ_2 und ϵ_3 sind die jeweiligen Messfehler der Messgleichungen. Die gebräuchliche Matrixschreibweise für reflektive Messmodelle lautet:

$$y = \Lambda_y \eta + \epsilon$$

y beschreibt die manifeste Variable, η die latente Variable. Die Beziehungen zwischen den latenten und manifesten Variablen wird durch die Koeffizientenmatrix Λ_y ausgedrückt. ϵ ist der Vektor der Messfehler.
Betrachten wir nun das Messmodell für die 2 latenten Variablen Erfolg und Aufwand ergibt sich folgende Matrixgleichung:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} \lambda_{11} & 0 \\ \lambda_{12} & 0 \\ \lambda_{13} & 0 \\ 0 & \lambda_{21} \\ 0 & \lambda_{22} \\ 0 & \lambda_{23} \end{pmatrix} \begin{pmatrix} Erfolg \\ Aufwand \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{pmatrix}$$

Die Variablen y_1, y_2, y_3 bzw. y_4, y_5, y_6 sind die manifesten Variablen von Erfolg bzw. Aufwand.
Es gelten die üblichen Modellannahmen bei Regressionsgleichungen, Messfehler haben Erwartungswert 0 und Faktoren korrelieren nicht mit den Messfehlern:

$$\begin{aligned} E(\epsilon) &= 0 \\ E(\eta\epsilon') &= 0 \end{aligned}$$

Kapitel 3

Schätzung der Parameter - Der PLS-Algorithmus

Die Messmodelle und Strukturmodelle wurden im vorigen Kapitel definiert, nun müssen noch die Koeffizienten im Messmodell und Strukturmodell bestimmt werden. Zur Schätzung der Koeffizienten von Strukturgleichungsmodellen gibt es im Wesentlichen zwei Ansätze. Einmal die kovarianzbasierte Variante[1] und auf der anderen Seite das PLS-Modell (englisch: *Partial Least Square*). Diese Arbeit und die darin vorgestellten Grafiken beziehen sich ausschließlich auf Strukturgleichungsmodelle, die mit Hilfe des PLS Verfahrens geschätzt wurden.
Das PLS - Modell baut auf einen Algorithmus auf, der sukzessiv die Werte der latenten Variablen, den *Scores* $\eta_1, \ldots, \eta_G$ bestimmt und schließlich mittels der KQ-Methode die Koeffizienten schätzt.
Um die Scores bestimmen zu können wird zunächst für jede latente Variable ein *Gewichtungsvektor* w_g ermittelt, der gerade soviele Komponenten hat wie die jeweilige latente Variable Indikatoren. Der Score η_g wird dann als Linearkombination des Gewichtsvektor w_g und seiner Indikatoren berechnet. Die Gewichte werden dabei mit Hilfe des Algorithmus iterativ berechnet. (Schlittgen, 2009)

[1] siehe zum Beispiel Wolf (2010, Kapitel 29)

Im Folgenden wird der PLS Algorithmus Schritt für Schritt nach Schlittgen (2009) vorgestellt. Das untere Schaubild soll zu einem besseren Verständnis beitragen. In den Gleichungen sind die gesuchten Variablen farbig markiert.

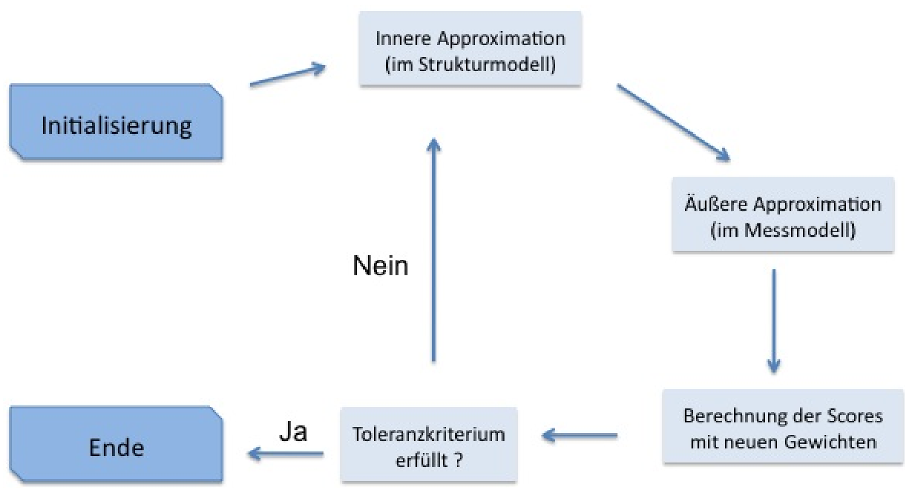

Abbildung 3.1: PLS Algorithmus

- **1. Schritt: Initialisierung**

  Anfangs wird für jede latente Variable g ein Gewicht w_g gewählt, um so die Scores der latenten Variablen als gewichtete Summe der Werte der zugehörigen manifesten Variablen bestimmen zu können. Die Wahl der Gewichte ist in der Initialisierung beliebig, es wird aber vorgeschlagen die Werte +/- 1 zu nehmen, je nachdem ob die erwartete Korrelation von η_g mit der manifesten Variable X_{gi} positiv oder negativ ist.
  Die Scores der latenten Variablen werden nun im ersten Schritt unter Verwendung der Gewichte w_{gi} bestimmt.

  $$\eta_g = f_g X_g w_g \qquad (g = 1, \ldots, G)$$

  Der Faktor f_g wird dabei so gewählt, dass die latente Variable η_g standardisiert ist. Die Varianz von η_g ist also gleich 1.

- **2. Schritt: Innere Approximation**

  Die *innere Approximation* beschreibt die Zusammenhänge der latenten Variablen untereinander. Basis für die Ermittlung der Beziehungen zwischen den latenten Variablen ist die Menge C_g, sie gibt an mit welchen Variablen η_g in Beziehung steht, also direkte Nachbarn sind. Beispielweise hat die Variable *Erfolg* die Menge C_{Erfolg}=$\{Aufwand, Vertrauen\}$. Da im Schritt 1 die latenten Variablen nur über ihre manifesten Variablen bestimmt wurden, wird eine Verbindung der latenten Variablen miteinander dadurch hergestellt, dass η_g ersetzt wird durch eine Umgebungsvariable

$$\eta_g^* = f_g \sum_{h \in C_g} e_{gh} \eta_h$$

Bei dieser *inneren Approximation* werden die Gewichte e_{gh} so gewählt, dass sie je nach der im Strukturmodell unterstellten Ursache-Wirkung Beziehung entweder +1 oder -1 sind. Die einzelnen latenten Variablen werden durch die gewichtete Summen der mit ihr in direkter Beziehung stehenden übrigen latenten Variablen ersetzt. Diese Wahl der inneren Gewichte wird als *Zentroid-Schema* bezeichnet. f_g wird so gewählt, dass η_g^* wieder die Varianz 1 hat.

- **3. Schritt: Äußere Approximation**

  Im ersten Schritt wurden die äußeren Gewichte w_{gi} auf +/- 1 gesetzt. Im nächsten Schritt, der sogenannten *äußeren Approximation* werden die Gewichte w_{gi} mit Hilfe der neuen Scores η_g^* neu bestimmt. An dieser Stelle kann man sich entscheiden, ob man ein reflektives Messmodell oder ein formatives Messmodell zur Schätzung von w_{gi} anwenden will. Wir entscheiden uns für das reflektive Messmodell, die manifesten Variablen gehen also als Regressanden und die latenten Variablen als Regressoren ein:

  $$x_g = \hat{w}_g' \eta_g^*$$

  Die Lösung für $\hat{w}_g'$ ergibt sich durch KQ-Schätzung.

- **4. Schritt: Berechnung der Scores mit neuen Gewichten**

  Die Scores der latenten Variablen werden nun wie in Schritt 1 bestimmt, nur mit den neuen Gewichten $\hat{w}_g$.

  $$\eta_{g\nu} = f_g X_g \hat{w}_g \qquad (g = 1, \ldots, G)$$

  Der Faktor f_g wird wie in Schritt 2 gewählt.

- **5. Schritt: Prüfung des Toleranzkriteriums**

  Es werden die alten Gewichte w_{gi}^{alt} mit w_{gi} verglichen. Unterscheiden sie sich weniger als um einen vorgegebenen Schwellenwert, so wird die Iteration abgebrochen somit sind die $\eta_{g\nu}$ Scores der latenten Variablen. Andernfalls wird zu Schritt 2 übergegangen.

Sind die Scores der latenten Variablen einmal geschätzt, können die Koeffizienten Λ_y, B aus dem Beispiel von Kapitel 1 mit Hilfe der hergeleiteten Modellgleichungen und der linearen Regression geschätzt werden.

Kapitel 4

Multivariate Grafiken

4.1 Multivariate Grafiken - Übersicht

Multivariate Grafiken sind bei der explorativen Analyse von multivariaten Daten unverzichtbar, sie sollen eine Überblick über die Daten liefern, als auch Modelle diagnostizieren und Ausreißer sowie Fehler in den Daten identifizieren. (Carr, 2005)
Es gibt eine Vielzahl von multivariaten Grafiken in *R*, ich hebe zunächst ein paar hervor und gehe später bei den PLS Pfadmodell Grafiken noch einmal genauer auf sie ein. In Klammern stehen die zu den Grafiken geeignete R-Pakete, falls sie noch nicht standardmäßig in R implementiert sind.

- Scatterplot
  - Scatterplot Matrix
    ermöglicht paarweise Zusammenhänge zwischen mehr als zwei metrisch skalierten Variablen zu veranschaulichen.
  - 3D Scatterplot (scatterplot3d)
    R Paket um multivariate Daten in einem dreidimensionalen Koordinatensystem zu visualisieren.
  - Sunflower-Plot
    ist ein Scatterplot, der mit Hilfe von Symbolen (Sonnenblumen) *Overplotting* visualisiert. Falls man einzelne Beobachtungen in einem Scatterplot nicht mehr voneinander unterschriden kann, spricht man von *Overplotting*.
- Parallel-Plot (MASS)
  Parallel Plots sind ein mächtiges Werkzeug um mehr als 3 Variablen darzustellen. Wie der Name schon sagt, wird für jede Variable ein parallele Achse bestimmt.

- Mosaik Plots
  Mosaik Plots stellen Zeilen und Spalten einer Kontigenztafel flächenproportional dar, werden bei kategorialen Daten angewendet.

- Figure Plots
  Grundidee: Eigenschaft einer Figur entspricht einer Variablen

  - Chernoff Gesichter (aplpack)
  - Star Plots
  - Profile Plots

4.2 Multivariate PLS Pfadmodell Grafiken

Das Ziel von Multivariate Grafiken für PLS Pfadmodelle ist es die Beziehungen zwischen den manifesten sowie latenten Variablen anschaulich darzustellen. Um bei den Beispiel aus Kapitel 2 zu bleiben, will man die Variabelen im Mess- und Strukturmodell, insbesondere alle Koeffizienten λ und β, also die Zusammenhänge in Strukturgleichungsmodellen passend visualisieren.

Messmodell für Erfolg:

$$\begin{aligned}
Mathematik &= \lambda_{11} Erfolg + \epsilon_1 \\
Total &= \lambda_{12} Erfolg + \epsilon_2 \\
Verbal &= \lambda_{13} Erfolg + \epsilon_3
\end{aligned}$$

Strukturmodell:

$$\begin{aligned}
\mathit{Aufwand} &= \zeta_1 \\
\mathit{Vertrauen} &= \beta_{21} * \mathit{Aufwand} + \zeta_2 \\
\mathit{Erfolg} &= \beta_{31} * \mathit{Aufwand} + \beta_{32} * \mathit{Vertrauen} + \zeta_3
\end{aligned}$$

Eine Hürde dabei ist die oft sehr hohe Anzahl an Variablen in Strukturgleichungsmodellen, die es für den Benutzern erschweren alle für ihn relevante Informationen über ein Modell in einer oder weniger Grafiken zu veranschaulichen, ohne dabei die Übersicht zu verlieren.
Da in Strukturgleichungsmodellen manifeste Variablen in *Likart Skalen* [1] gemessen werden, geht man häufig davon aus, dass man zwar eine metrische Variable beobachtet aber nur mit einer Ordinalskala misst. Diese Annahme ermöglicht eine umfangreichere Datenanalyse.
Hinsichtlich der explorativen Datenanalyse sollte dem Benutzer auch ermöglicht

[1] ordinale Skala

werden, bisher erstellte Grafiken ohne großen Aufwand an neue Fragestellungen anpassen zu können. Die Benutzerfreundlichkeit sollte also auch sichergestellt sein. Besonders in Bezug auf diesen Aspekt empfiehlt sich die Benutzung des R Pakets *ggplot2* von Hadley Wickham, da der Benutzer hier nicht gezwungen wird aus einem Pool an vordefinierten Grafiken auszuwählen, sondern von Grund auf Grafiken maßgeschneidert auf die jeweiligen Fragestellungen erstellen kann, mit einem verhältnismäßig sehr geringem Aufwand.

4.2.1 ggplot2

Wie bereits erwähnt ist ggplot2 ein mächtiges Werkzeug zur Erstellung von individuell angepassten Grafiken, daher will ich in diesem Teil noch einen kurzen Einblick in die Funktionsweise von ggplot2 geben.
Im Gegensatz zu anderen Pakete in R liegt ggplot2 eine eigene Grammatik zur Erstellung von Grafiken zu Grunde, die *Grammar of Graphics* (Wilkinson, 2005). Kurz zusammengefasst steht hinter der *Grammar of Graphics* die Idee, dass statistische Grafiken eine Abbildung von Daten auf ästhetische Merkmale (Farbe,Größe,Form) von geometrischen Objekten (Linien, Punkte, Balken) sind. Die Grammatik legt fest, dass jede Grafik aus sechs Bausteinen zusammengefügt wird: (Wickham, 2009)

- der **Datensatz**, der in der Grafik visualisiert werden soll, sowie die dazu gehörigen ästhetischen Abbildungen, kurz *Aesthetics*, die beschreiben in welcher Form, Farbe und Größe die Variablen dargestellt werden sollen.

- geometrischen Objekte, kurz **geoms** bestimmen durch welche Art von Objekt (Punkte,Linien,Balken, etc.) die Variablen dargestellt werden.

- statistische Transformationen, kurz **stat** erstellen neue Variablen für eine Grafik, beispielsweise erzeugt stat_density eine Dichtekurve.

- für jede benutzte Aesthetic existiert eine Funktion **scale**, die die Werte aus dem Datenraum gültige Werte im Raum der Aesthetics zuweist. Die Scales kontrollieren das Aussehen der geoms und zugehöriger Legenden, und erstellen Achsen sowie Legenden.

- das Hinzufügen von Koordinatensystemen kurz **cords** ist der letzte Schritt bei der Erstellung von Grafiken. Hier kann bestimmt werden wie die Koordinaten der Dateien in die Ebene übertragen werden sollen.

- außerdem ist es noch möglich Grafiken zu **facet**tieren. Dabei erstellt man aus einem Datensatz Untergruppen und erzeugt für diese Untermengen jeweils die selbe Art von Grafik.

Beispiel

Die Funktionsweise von ggplot2 ist ohne konkrete Beispiele schwer nachzuvollziehen, darum werde ich im Folgenden Schritt für Schritt exemplarisch eine ggplot2 Grafik erstellen.
Der benutzte Datensatz bezieht sich auf den *European Customer Satisfaction Index* Datensatz. Die Variablen *IMAG1,IMAG2* sind manifeste Variablen für die latente Variable *IMAGE*, die Variable *gender* bzw. *nation* bezieht sich auf das Geschlecht bzw. auf die Nationalität.

- ggplot2 wird geladen, Datensatz wird eingelesen, sowie die zu visualisierenden Variablen bestimmt:

```
> library(ggplot2)
> library(semPLS)
> data(ECSImob)
> gender <- sample(c("m","f"),250,replace=TRUE)
> nation <- sample(c("EU","US"),250,replace=TRUE)
> g <- ggplot(data=mobi[,ECSImobi$blocks[[1]]], aes(x=IMAG1,y=IMAG2))
```

- Nun soll ein Scatter Plot erstellt werden, dazu benötigt man das geometrische Objekt Punkt.

```
> g + geom_point()
```

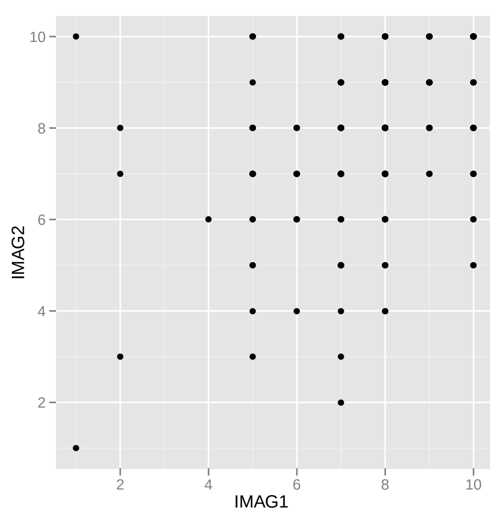

Abbildung 4.1:

- Anhand Abbildung 4.1 lässt sich vermuten dass viele Beobachtungen in einem Punkt überlappen, ggplot2 bietet hier verschiedene Funktionen an um dies zu verhindern. Die Funktion `stat_sum` skaliert die Größe der Punkte nach der Anzahl der Beobachtungen. Außerdem will man noch den Zusammenhang zwischen den zwei manifesten Variablen mit Hilfe von *Smoothing Splines* verdeutlichen:

  ```
  > g + stat_sum(aes(group = 1,size=..n..)) +
  + geom_smooth()
  ```

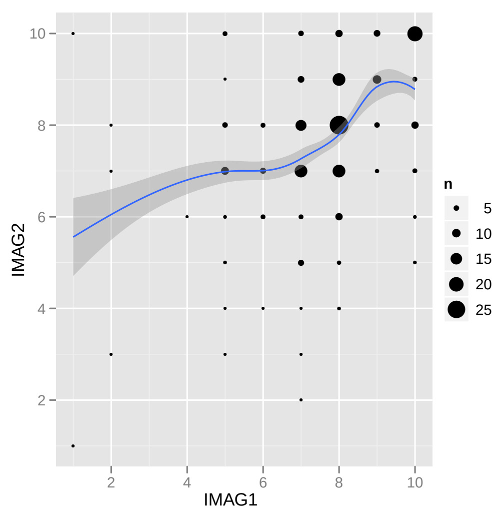

Abbildung 4.2:

- Als nächstes sollen die Beobachtungen nach Geschlecht unterschieden werden, dazu ist die bisher benutzte Funktion `stat_sum` nur bedingt geeignet. Um auch jede Beobachtung eindeutig farblich zuordnen zu können, empfiehlt sich *Jittering* anzuwenden. *Jittering* verhindert Overplotting, indem es auf Beobachtungen im Koordinatensystem einen kleinen zufälligen Wert in x-und y-Richtung addiert.

```
> g + geom_jitter(aes(col=gender))+
+ stat_smooth(aes(col=gender))
```

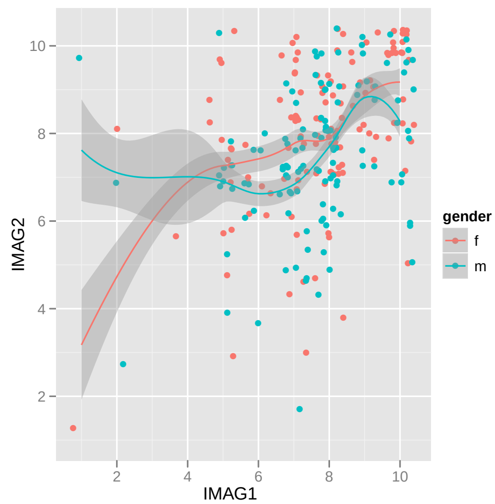

Abbildung 4.3:

- Jetzt soll noch nach der Nationalität anhand der Form der Punkte unterschieden werden:

```
> g + geom_jitter(aes(col=gender,shape=nation)) +
+ stat_smooth(aes(col=gender))
```

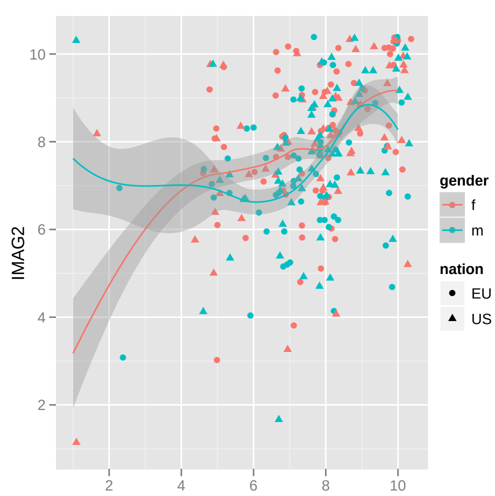

Abbildung 4.4:

- Will man nun noch die Regressionsgeraden getrennt nach Geschlecht und Nationalität betrachten, empfiehlt sich aufgrund der Übersichtlichkeit eine Facettierung:

```
> g + geom_jitter(aes(col=gender))
+ stat_smooth(aes(col=gender)) + facet_grid(.~nation)
```

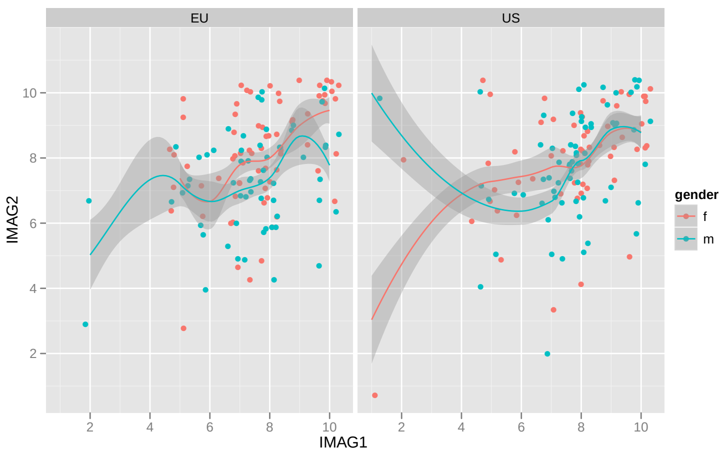

Abbildung 4.5:

- Im letzten Schritt will man nun noch das Erscheinungsbild der Grafik verändern:

```
> g + geom_jitter(aes(col=gender))+stat_smooth(aes(col=gender)) +
+ facet_grid(.~nation)+scale_colour_manual("gender",c("f"="red","m"="blue"))
+ opts(title="Beispiel") + theme_set(theme_bw())
```

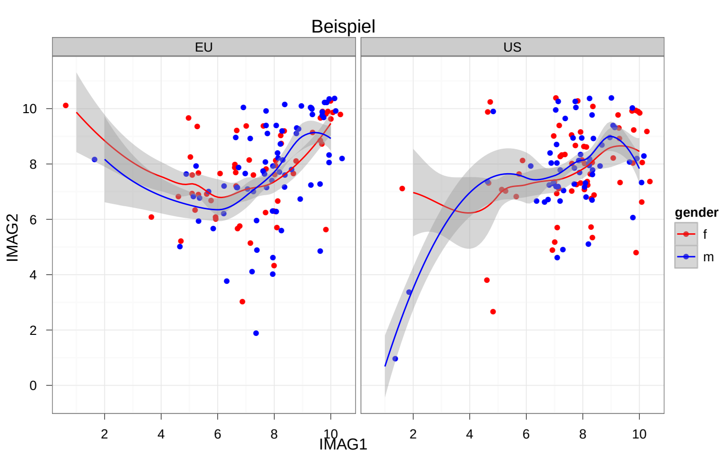

Abbildung 4.6:

An diesem Beispiel wird deutlich, dass ggplot2 es ermöglicht seine Grafik Stück für Stück aufzubauen. Im Gegensatz zu vielen anderen Paketen die statistische Grafiken erstellen, kann man in ggplot2 mit wenigen Befehlen Grafiken nachträglich mit zusätzlichen Informationen anreichern. Dies kann bei der deskriptiven Auswertung von PLS Pfadmodellen ein ungemeiner Vorteil sein.

4.2.2 Beispiele für multivariate Grafiken in PLS Pfadmodellen

In diesem Kapitel werden nun konkrete Beispiele für multivariate Grafiken in PLS Pfadmodellen gegeben. Alle Grafiken beziehen sich wie bereits im vorherigen Kapitel auf den *European Customer Satisfaction Index* Datensatz und wurden mit Hilfe von ggplot2 erstellt.

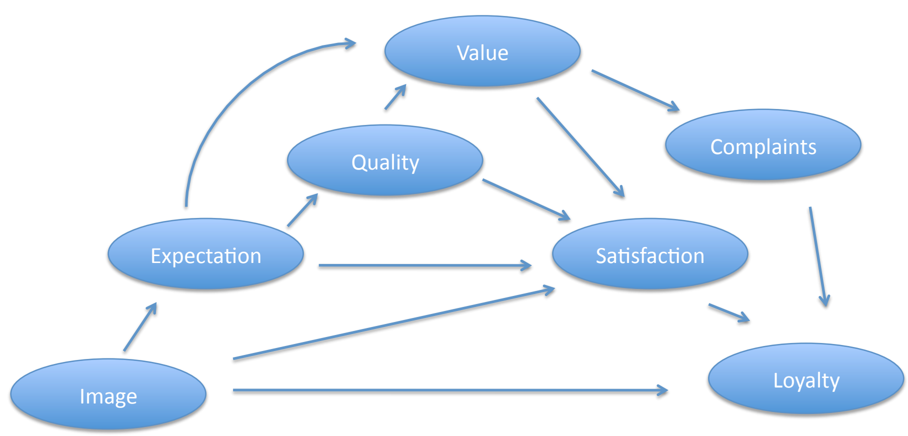

Abbildung 4.7:

Abbildung 4.7 zeigt das Strukturmodell für den ECSI Datensatz, die Messmodelle für jede latente Variable wurden zu Gunsten der Übersichtlichkeit ausgespart. Hinsichtlich der folgenden Grafiken ist es wichtig zu wissen, dass die Koeffizienten im Strukturmodell mit dem griechischen Buchstaben *beta* und die Koeffizienten in den Messmodellen mit dem griechischen Buchtaben *lambda* beginnen.

```
> library(semPLS)
> library(ggplt2)
> data(ECSImobi)
> ecsi <- sempls(model=ECSImobi, data=mobi)
> g <- ggplotmvpairs(ECSImobi,data=ecsi)
> print(g)
```

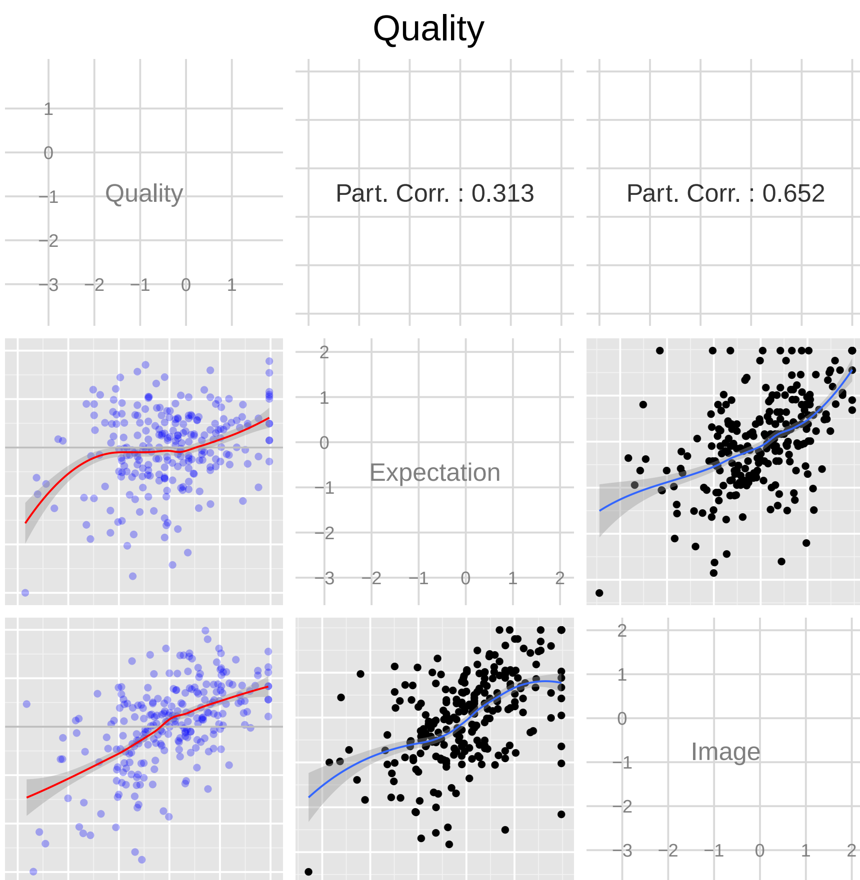

Abbildung 4.8:

Die erste Grafik zeigt einen Pairs Plot, der als Zielvariable die latente Variable *Quality* hat. Auf der Diagonalen werden außer der Zielvariable, der direkte Vorgänger *Expectation* sowie der indirekte Vorgänger *Image* angegeben. Die Plots unterhalb der Variable *Quality* tragen auf der y-Achse die standardisierten Residuen eines linearen Modells ab, das als Zielgröße eben die Variable *Quality* und als Einflussgrößen alle Vorgänger bis auf denjenigen hat, der auf der x-Achse abgetragen wird. So zeigt beispielsweise der Plot im linken unteren Eck die standardisierte Streuung der Residuen des Modells mit *Expectation* als Einflussgröße, in Abhängigkeit von der Variable *Image*. Der Zusammenhang wurde mit Hilfe von

Smoothing Splines (rote Line) verdeutlicht. Die Zahlen in der obersten Zeile des Pairs Plot, geben die partiellen Korrelationskoeffizienten wieder. Die restlichen Plots zeigen die Scatterplots der Nachfolgervariablen.

```
> ecsiBoot <- bootsempls(ecsi, nboot=200, start= "ones " ,verbose=TRUE)
> ggplotdensity(ecsiBoot,pattern="beta",centered= "TRUE")
```

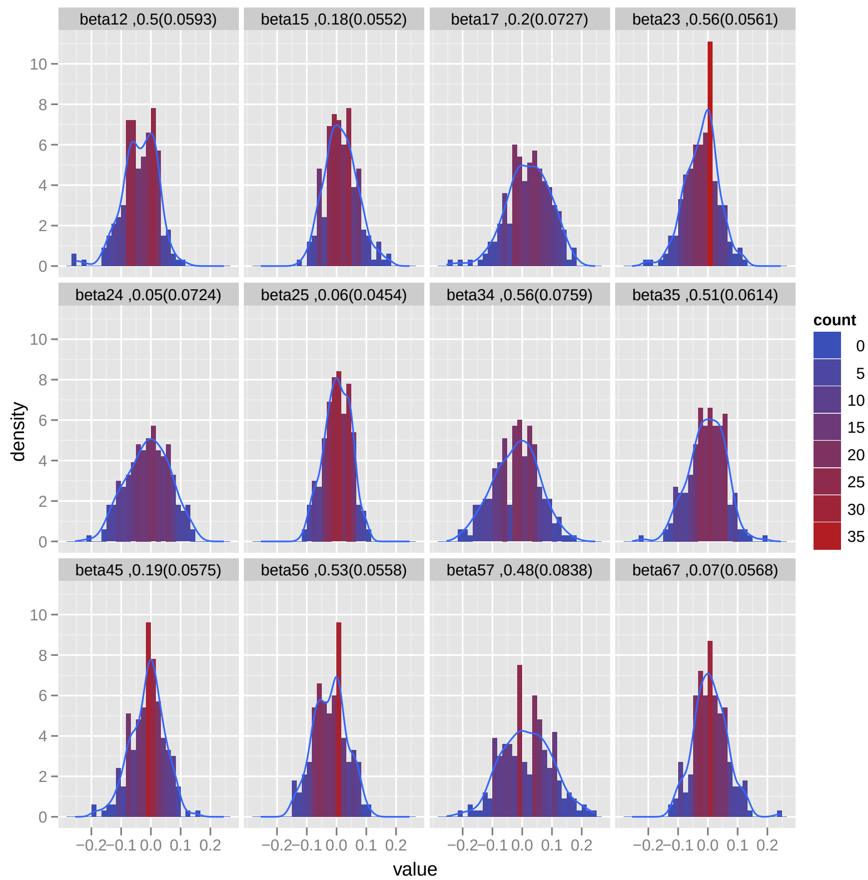

Abbildung 4.9:

Abbildung 4.9 zeigt die Verteilung aller **Pfadkoeffizienten** *beta* im Modell. Um die Verteilungen vergleichbarer zu machen wurden die Verteilungen jeweils auf 0 zentriert, hinter den Pfadkoeffizienten sind der Mittelwert und in Klammern die Standardabweichung angegeben. Die Dichtekurven in den Grafiken wurden mit Hilfe des Gauss-Kerns geschätzt.

```
> ecsi <- sempls(model=ECSImobi, data=mobi)
> ggplotdensity(ecsi,centered="TRUE", use="fscores")
```

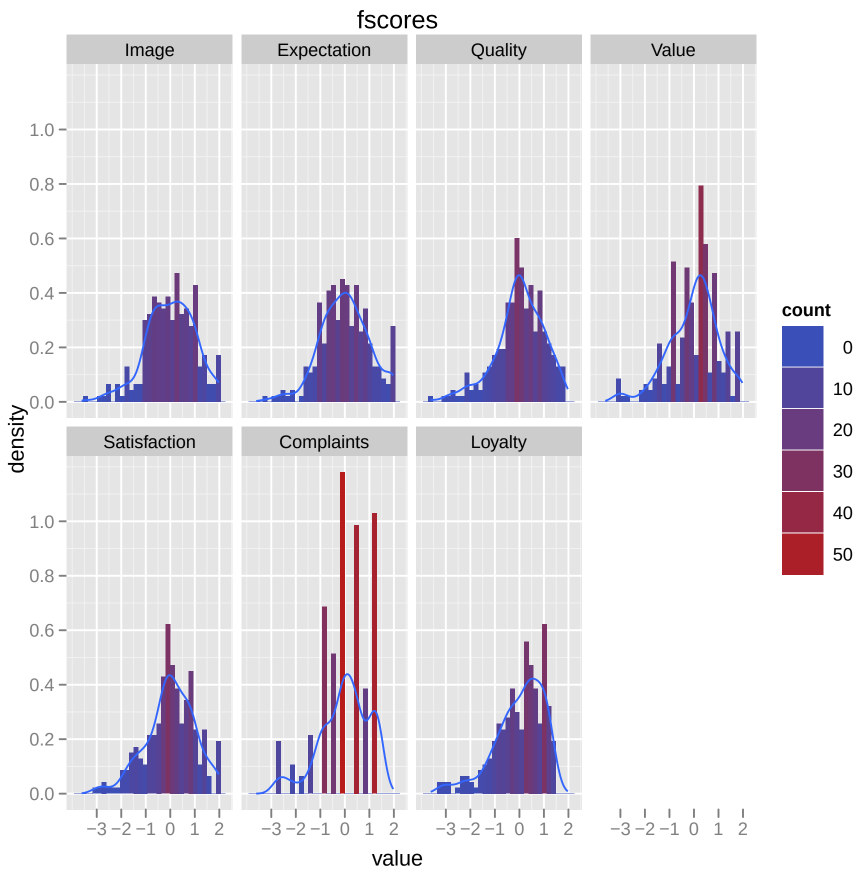

Abbildung 4.10:

In Abbildung 4.10 sieht man die Verteilung der **Scores** der latenten Variablen im Pfadmodell. Der Erwartungswert bzw. die Standardabweichung sind bereits wie in Kapitel 3 erwähnt auf 0 bzw. 1 standardisiert.

```
> ggplotparallel(ecsiBoot,pattern="lam")
```

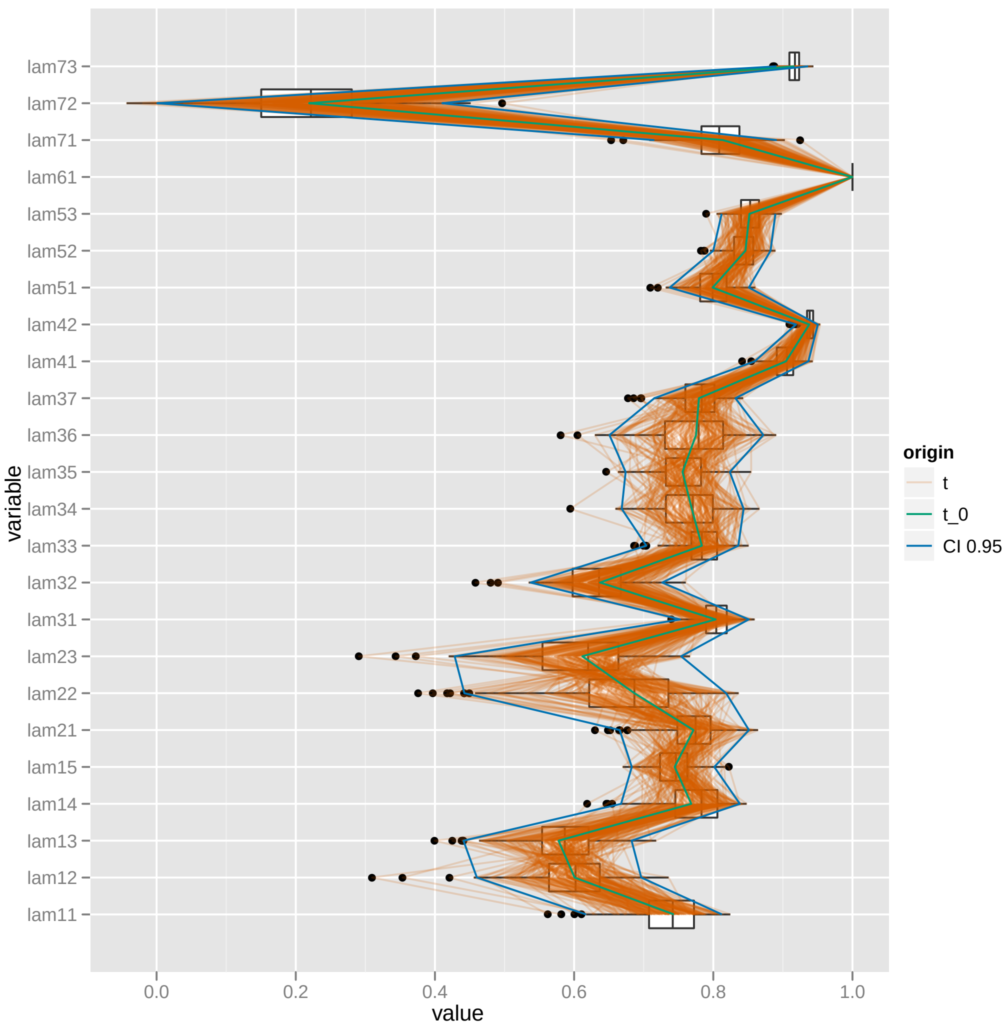

Abbildung 4.11:

In der oben stehenden Abbildung werden die **Koeffizienten aller Messmodells** *lambda* in einem Parallel Plot beschrieben. Die orangefarbenen Linien stehen jeweils für eine Stichprobenziehung aus der Bootstrap Methode. Zusätzlich wurde für jeden Koeffizienten des Messmodells ein Boxplot hinzugefügt.

```
> g <- ggplotmvpairs(ECSImobi, upper="corr")
> print(g)
```

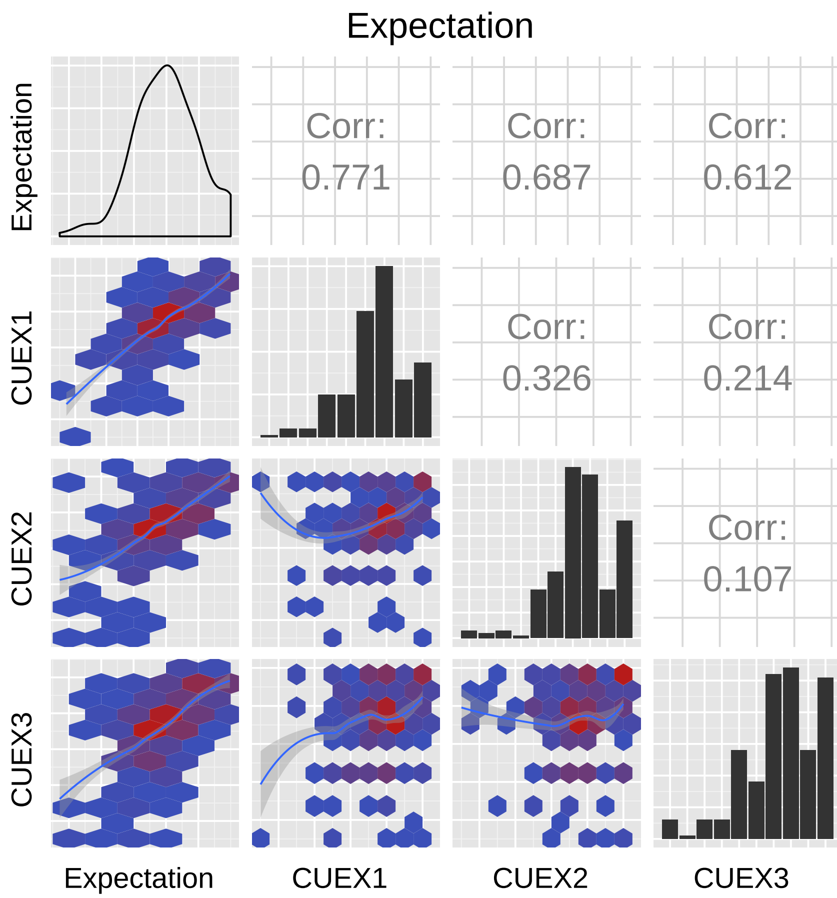

Abbildung 4.12:

Abbildung 4.12 stellt in einer Plot Matrix die latenten sowie manifesten Variablen für ein Messmodell dar, in diesem Fall für das Messmodell der latenten Variable *Expectation*. Die Zusammenhänge zwischen den manifesten Variablen (*CUEX1*,*CUEX2*,*CUEX3*) und der latenten Variable *Expectation* wurden in der linken unteren Hälfte in sogenannten *BinPlots* dargestellt. Dabei wird der Plot in kleine Gebiete (englisch: Bins) aufgeteilt und die Anzahl der Beobachtungen in jedem Bin gezählt. Die Farbe der Bins von blau (wenig) bis rot (viel) gibt die Anzahl der Beobachtungen an. Bei den Regressionsgeraden handelt es sich um *Smoothing Splines*. Die Grafiken auf der Diagonale geben die Verteilungen der latenten bzw. manifesten Variablen wieder. Die obere rechtere Hälfte zeigt die Korrelationskoeffizienten. Die Funktion `ggplotmvpairs` erstellt für jedes Messmodell eine Plot Matrix. Mit dem Parameter `upper` dieser Funktion kann man festlegen, welche Grafiktypen in der oberen rechten Hälfte angezeigt werden sollen.

Im Folgendem wird für jede Plot Matrix exemplarisch ein Grafiktyp betrachtet.

```
> g <- ggplotmvpairs(ECSImobi, upper="jitter")
> print(g)
```

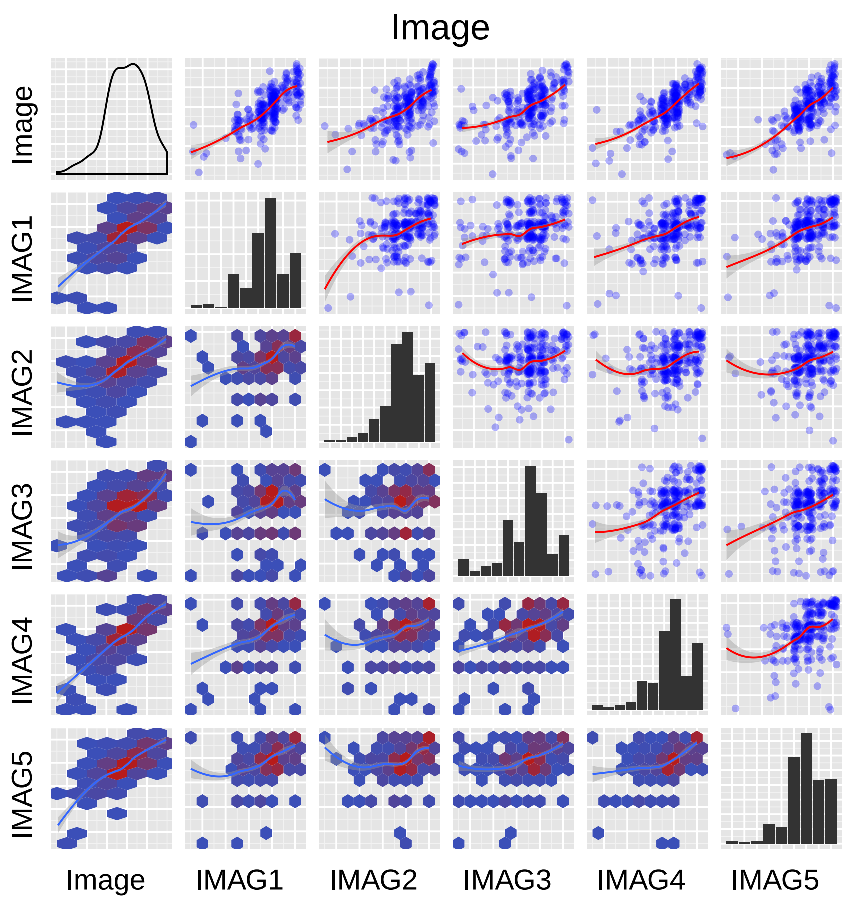

Abbildung 4.13:

In dieser Plot Matrix wird das Messmodell der latenten Variable *Image* dargestellt. Im Gegensatz zur vorherigen Grafik wurden in der rechten oberen Hälfte diesmal die Zusammenhänge mittels eines Jitterplots dargestellt. Zusätzlich wurde von jedem Punkt in der Grafik die Transparenz erhöht ,um einzelne Beobachtungen besser unterscheiden zu können.

```
> g <-ggplotmvpairs(ECSI, upper="cooks")
> print(g)
```

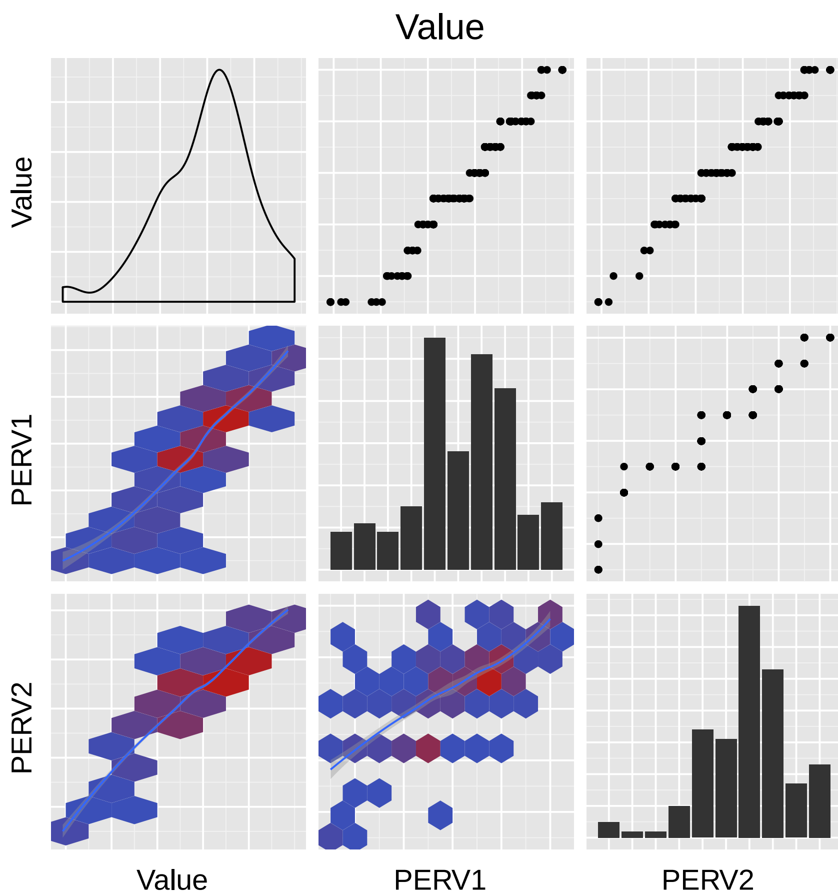

Abbildung 4.14:

Hier wird in der rechten oberen Hälfte paarweise die Verteilungen der Variablen mit Hilfe eines Quantile-Quantile-Plots verglichen.

```
> g <- ggplotmvpairs(ECSI, pattern="he")
> print(g)
```

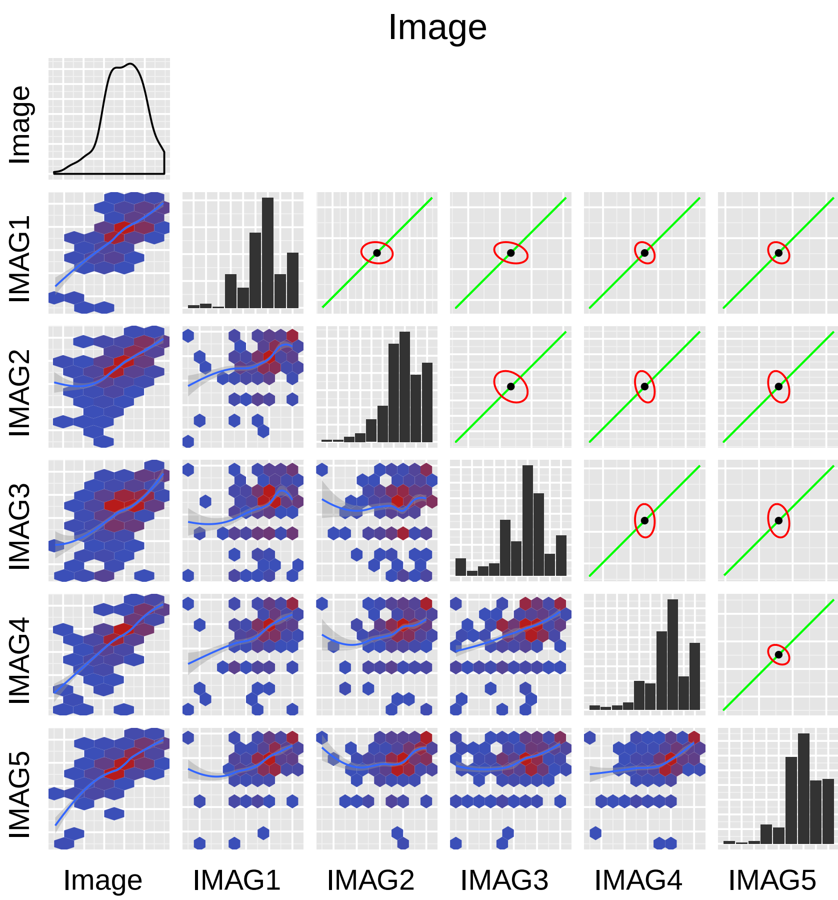

Abbildung 4.15:

Hier sehen wir wieder eine Plot Matrix, anstatt der Quantlie-Qunatile Plots sind in dieser Grafik in der rechten oberen Hälfte sogenannte **HEPlots** abgebildet. HEPlots ermöglichen die Visualisierung von Hypothesentests bei linearen multivariaten Modellen, indem sie die Hypothesen und Error Matrizen als Ellipsen darstellen. Mit Hilfe des folgendem Ausschnitt aus der obigen Plot Matrix wird auf die Interpretationsmöglichkeiten von HEPlots näher eingegangen:

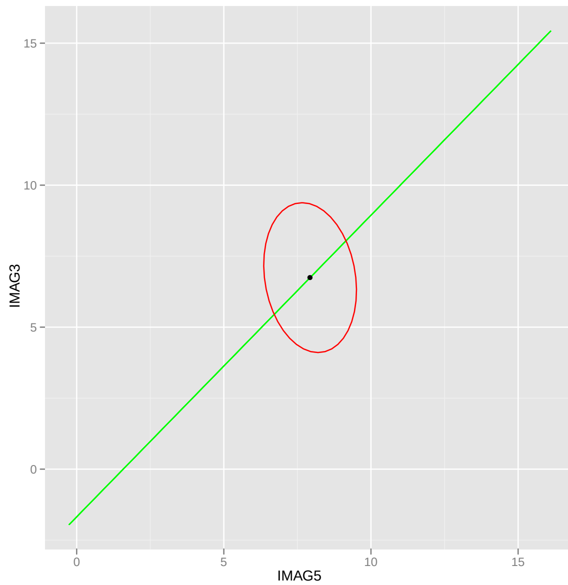

Abbildung 4.16:

Die Idee hinter den HEPlots ist, dass sich wie oben schon erwähnt, in multivariaten linearen Modellen der Form:

$$Y = XB + U$$

die Hyptohese

$$H_0 : LB = 0$$

grafisch darstellen lässt, wobei B die Koeffizientenmatrix, Y die Response Matrix, X die Einflussvariablen und E die Residuen Matrix darstellt. In dem linearen multivariate Modell zu Abbildung 4.16 sind die Zielvariabeln die manifesten Größen *IMAG1,IMAG2,*
IMAG3,IMAG4,IMAG5 und die Einflussgröße ist die die latente Variable *Image*. Es soll getestet werden ob die latente Variable einen signifikanten Einfluss auf die manifesten Variablen hat. Die Matrix L ist also gleich der Identitätsmatrix. Für die Teststatistik benötigt man nun das multivariate Analogon zu der univariaten Streuung der Residuen SSE und Modellstreuung SSM, also die Error Matrix E und die Hypothesen Matrix H:

$$E = (Y - X\hat{B})'(Y - X\hat{B})$$
$$H = (Y - X\hat{B}_0)'(Y - X\hat{B}_0)'$$

wobei $\hat{B}$ den Schätzer für B ohne Restriktion (also unter H_1) und $\hat{B}_0$ den Schätzer für B mit Restriktionen (also unter H_0) bezeichnet.

Die Matrizen von H und E sind in HEPlots genau so skaliert, dass die Ellipse von H (grün) genau dann über die Ellipse von E (rot) hinausragt, wenn die Nullhypothese H_0 zum Signifikanzniveau $\alpha = 0.05$ abgelehnt werden kann. Falls wie in unserem Fall die Hypothese nur einen Freiheitsgrad hat, wird die Ellipse von H zu einer Geraden. Wie man sieht hat die latente Variable auf beide manifeste Variablen einen signifikanten Einfluss.
Zusätzlich gibt die Lage und Form der Ellipse von E an, auf welche manifeste Variable der Einfluss der latenten Variable signifikanter ist. Da der Schatten, den die Ellipse E auf die x-Achse projiziert kleiner ist, als der den die Ellipse auf die y-Achse projiziert, ist der Einfluss der Variable *Image* auf *IMAG5* signifikanter als der Einfluss auf die Variable *IMAG3*. Man kann auch sagen, dass die Ellipse "paralleler" zur y-Achse steht und daher Image signifikanter auf *IMAG5* wirkt. Die Richtung in die die Ellipse von H zeigt, gibt auch die Richtung des Zusammenhangs der manifesten Variablen an, in unserem Fall ist er positiv. Die Ellipsen wurden so in das Koordinatensystem gesetzt, dass ihr Zentrum die Mittelwerte der beiden Ausprägungen wiedergibt.

Kapitel 5

Fazit / Ausblick

Das vorgestellt Paket ggplot2 ist sehr gut geeignet für die Analyse von multivariaten Daten. Sobald man mit der ggplot2 zugrundeliegenden *Grammar of Graphics* vertraut geworden ist, ermöglicht das Paket auf eine sehr unkomplizierte Weise ansprechende multivariate Grafiken zu erstellen. Alle in Kapitel 4.2.2 vorgestellten Grafiken wurden so erstellt, dass der Anwender neue Variablen (wie zum Beispiel in Kapitel 4.2.1: Geschlecht und Nationalität) im Strukturgleichungsmodell mit geringem Aufwand in seine Grafiken visualisieren kann. Zusammenfassend kann man sagen, dass ggplot2 ein sehr mächtiges Werkzeug im Erstellen von benutzerdefinierten Grafiken ist und damit sich eben speziell auch für die Erstellung von mulitvariaten Grafiken in PLS Pfadmodellen eignet. Diese Arbeit stellt nur einen Teil von möglichen multivariaten Grafiken in Pfadmodellen dar, es gibt auch Grafiken, die (bisher) nicht ohne Weiteres mit ggplot2 erstellt werden könne, wie zum Beispiel sogenannte Netzwerk Plots[1] die Korrelationsmatrizen visualisieren (nächste Seite):

[1] R-Paket qgraph

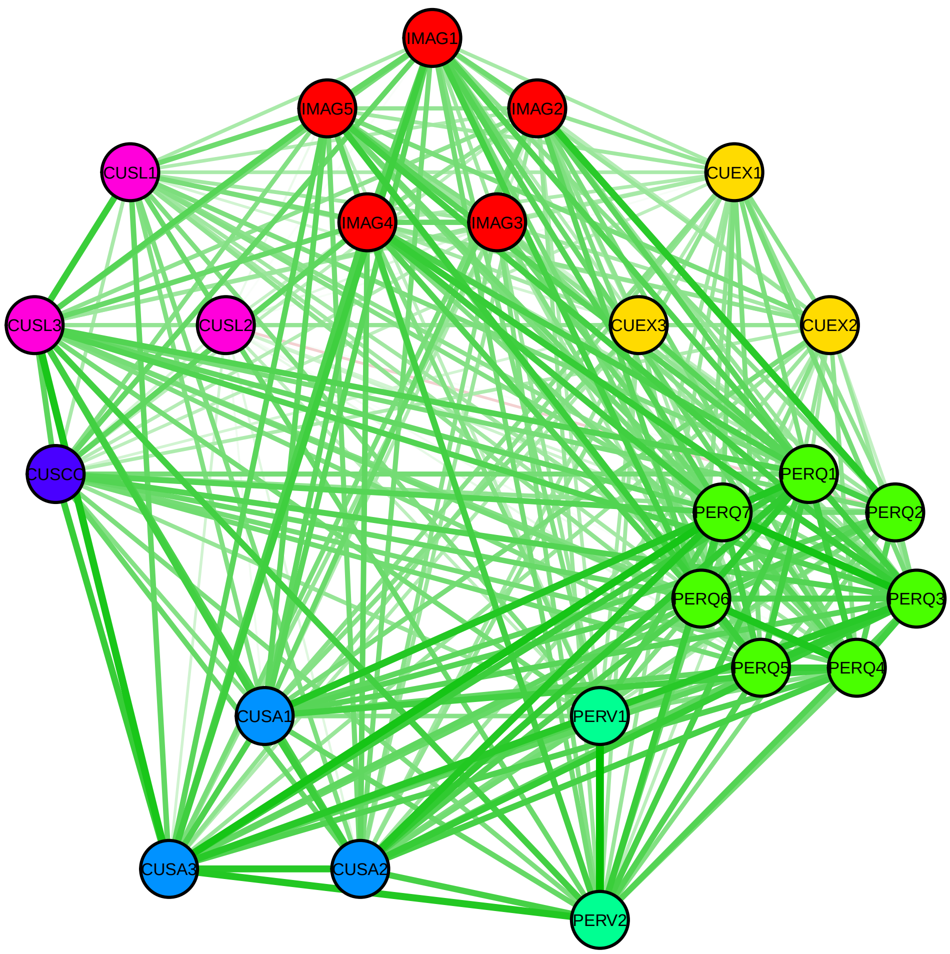

Abbildung 5.1: alle Beziehungen der manifesten untereinander Variablen des ECSI Datensatzes visualisiert in einem Netzwerk Plot

Literaturverzeichnis

Carr, D. B. (2005). *Multivariate Graphics*, Encyclopedia of Biostatistics.

Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D. and Borsboom, D. (2011). *qgraph: Network representations of relationships in data, R package version 0.5.0.*

Fox, J., Friendly, M. and Monette, G. (2010). *heplots: Visualizing Tests in Multivariate Linear Models. R package version 0.9-8.*

Friendly, M. (2007). *HE plots for Multivariate General Linear Models*, number 16.

Monecke, A. (2010). *semPLS R package version 0.8-7.*

Nitzl, C. (2010). *Eine anwendungsorientierte Einführung in die Partial Least Square Methode (PLS)-Methode*, Prof. Dr. K.-W. Hansmann.

Schlittgen, R. (2009). *Multivariate Statistik*, Oldenburger Wissenschaftsverlag.

Schloerke, B., Crowley, J., Cook, D., Hofmann, H., and Wickham, H. (2011). *GGally: Extension to ggplot2.R package version 0.3.0.*

Weiber, R. and Mühlhaus, D. (2010). *Strukturgleichungsmodellierung*, Springer.

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*, Springer New York.

Wilkinson, L. (2005). *The grammar of graphics*, New York, Springer.

Wolf, C. (2010). *Handbuch der sozialwissenschaftlichen Datenanalyse*, VS Verlag.

Anhang A

R-Code

R-Code zur Erstellung der Grafiken in Kapitel 4.2.2:

```
#ECSI Datensatz laden
library(semPLS)
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi)
ecsiBoot <- bootsempls(ecsi, nboot=200, start="ones", verbose=TRUE)


#Funktionen

ggplotdensity<-function (x, data, ...)
UseMethod("ggplotdensity")

ggplotparallel<-function (x, data, ...)
UseMethod("ggplotparallel")




#Verteilungen der Koeffizienten
ggplotdensity.bootsempls <- function(x, data, pattern="lam", subset=NULL,
                                     centered = TRUE , ...){


library(ggplot2)

  ind <- grep(pattern, colnames(x$t))
  ifelse(is.null(subset),
           params <- colnames(x$t)[ind],
           ifelse(is.character(subset), params <- subset,
                  params <- colnames(x$t)[subset])
           )
```

```
    Y <- data.frame(NULL)

    if(centered){
    k<-1
    for(i in params){
        if(round(var(x$t[,i], na.rm=TRUE), digits=4)==0) next
        x_max <- density(x$t[,i])$x[which.max(density(x$t[,i])$y)]
        m<-paste(",",round(x$t0[ind[k]],digits=2),sep="")
        tmp <- data.frame(value=x$t[,i]- x_max , name=paste(i,m))
        Y <- rbind(Y, tmp)
        k<-k+1
    }}

    else{

    for(i in params){
        if(round(var(x$t[,i], na.rm=TRUE), digits=4)==0) next
        tmp <- data.frame(value=x$t[,i], name=i)
        Y <- rbind(Y, tmp)
        }}

Y<<-Y
Y.unst<-unstack(Y)     # Daten transformieren: Beobachtung pro Zeile
mean <- as.vector(apply(Y.unst,2,mean))
sd <- as.vector(apply(Y.unst,2,sd))
Y[,2] <- paste(Y[,2],"(",rep(round(sd,digits = 4),
each=dim(Y.unst)[1]),")",sep="") # Überschrift der Plots werden festgelegt

densityboot<-ggplot(Y)
densityboot+geom_histogram(aes(value, ..density..,fill = ..count..))+
facet_wrap(~ name,ncol = 4)+stat_density(aes(value,ymax=max(Y$value)),
colour = "#3366FF", geom = "line")

}

# Verteilungen der Scores,Residuen, vorhergesagten Werte
# zuerst Funktionen predict.R, residuals.r laden
ggplotdensity.sempls <- function(x, data,
            use=c("fscores", "prediction", "residuals"),scaled=TRUE,...){
library(ggplot2)
      use <- match.arg(use)
    if(use=="fscores")         val <<- x$factor_scores
    else if(use=="prediction") val <<- predict(x)
    else if(use=="residuals")  val <<- residuals(x)

if (!scaled) {
s  <-attr(val,"scaled:scale")
m <-attr(val,"scaled:center")
val<- t(apply(val,1,function(x) {x*s+m}))}
```

```
    Y <- data.frame(NULL)
    exogenous <- exogen(x$model)
    r<-x$model$latent

     for(i in x$model$latent){
        if(i %in% exogenous & use!="fscores") next
        tmp <- data.frame(value=val[,i], name=i)
        Y <- rbind(Y, tmp)

    Y$name <- factor(Y$name, levels=x$model$latent)}


    if(is.null(list(...)$main)){
        main=paste(deparse(substitute(x)), "\n",
                  ifelse(use=="fscores", "factor scores", use))
    }
    if(is.null(list(...)$sub)){
        sub=paste("Exogenous LVs: ", paste(exogenous, collapse=", "))
    }

    densityboot<-ggplot(Y)
    densityboot<-densityboot +
    geom_histogram(aes(value, ..density..,fill = ..count..))  +
    stat_density(aes(value),colour = "#3366FF", geom = "line")
    +facet_wrap(~ name,ncol = 4)
    densityboot+opts(title=use)

 }

# Parallel Plot für Koeffizienten
ggplotparallel.bootsempls <- function(x, data, pattern="lam",
                                      subset=NULL, reflinesAt,..){
  library(ggplot2)
  ifelse(is.null(subset), ind <- grep(pattern, colnames(x$t)),
         ind <- subset)
    lower <- summary(x)$table$Lower
    upper <- summary(x)$table$Upper
    Y <- rbind(x$t, x$t0, lower, upper, deparse.level=0)
    if(!missing(reflinesAt)){
        Y <- rbind(Y, matrix(rep(reflinesAt, each=ncol(x$t)),
                   nrow=length(reflinesAt), byrow=TRUE))
        origin <- c(rep("1resample", x$nboot), "2sample", "3ci", "3ci",
                   rep("4reflines", times=length(reflinesAt)))
        Y <- data.frame(Y, origin)
    }
    else Y <- data.frame(Y, origin=c(rep("1resample", x$nboot),
                                     "2sample", "3ci", "3ci"))

df<-data.frame(id=seq_along(Y$origin),Y[ind],origin = Y$origin)
# Daten transformieren: Beobachtung pro Zeile
dfm<- reshape::melt(df,id.var=c("id","origin"))
```

```
pcp<-ggplot(dfm,(aes(variable,value)))+geom_boxplot()
pcp<- pcp + geom_line(aes(variable,value,group = id,colour = origin))
+ scale_colour_manual("origin",c("1resample"=alpha("#D55E00",0.2),
                              "2sample" = "#009E73","3ci" = "#0072B2"),
breaks =c("1resample","2sample","3ci"),labels= c("t","t_0","CI 0.95"))

pcp + coord_flip()

# aus dem R-Paket semPLS:

predict <- function(object, what=c("LVs", "MVs"), scale=c("original", "scaled"),
                    total=FALSE){
    # total: only for MVs. Include exogenous and endogenous MVs
    #        Exogenous MVs are treated as if they were all reflective.
    what <- match.arg(what)
    scale <- match.arg(scale)
    model <- object$model
    data <- object$data
    # missing factor scores?
    fsMissing <- which(complete.cases(object$factor_scores)==FALSE)
    # mising MV data?
    mvMissing <- which(complete.cases(object$data)==FALSE)
    if(what=="MVs"){
        msep <- function(object, mvPrediction, mvObserved){
            sum((mvPrediction - mvObserved)^2, na.rm=TRUE)/
            (prod(dim(data)) - ncol(object$coeff) - sum(is.na(mvPrediction)))
        }
    }
    # MVs
    if(total & what=="MVs"){
        mv_hat <- matrix(NA, nrow=nrow(data), ncol=ncol(data))
        colnames(mv_hat) <- colnames(data)
        exogen <- exogen(model)
        for(i in endogen(model)){
            block <- model$blocks[[i]]
            m <- attr(data, "scaled:center")[block]
            s <- attr(data, "scaled:scale")[block]
            for(l in 1:length(m)){
                ind <- complete.cases(object$factor_scores[, exogen])
                e <- object$factor_scores[ind, exogen, drop=FALSE] %*%
                     object$total_effects[exogen, i, drop=FALSE] *
                     object$outer_loadings[block[l],i]
                # rescaling
                if(scale=="original") e <- e * s[l] + m[l]
                mv_hat[ind, block[l]] <- e
            }
        }
        for(i in exogen(model)){
            block <- model$blocks[[i]]
            m <- attr(data, "scaled:center")[block]
```

```
        s <- attr(data, "scaled:scale")[block]
        for(l in 1:length(m)){
            ind <- complete.cases(object$factor_scores[, i])
            e <- object$factor_scores[ind, i] *
                 object$outer_loadings[block[l],i]
            # rescaling
            if(scale=="original") e <- e * s[l] + m[l]
            mv_hat[ind, block[l]] <- e
        }
    }
    result <- list(mv_prediction=mv_hat,
                   msep=msep(object, mv_hat, ifelse(scale=="orig",
                  rescale(object), data)))
    return(result)
}



# situation A: complete data
# LVs
if(what=="LVs" & length(fsMissing)==0){
    Y_hat <- object$factor_scores %*% object$path_coefficients
    return(Y_hat)
}
# MVs
if(what=="MVs" & length(fsMissing)==0){
    if(length(mvMissing)==0){
        mv_hat <- object$factor_scores %*%
                  object$path_coefficients %*%
                  t(object$outer_loadings)
        for(i in exogen(model)){
            block <- model$blocks[[i]]
            m <- attr(data, "scaled:center")[block]
            s <- attr(data, "scaled:scale")[block]
            for(l in 1:length(m)){
                e <- object$factor_scores[, i] *
                     object$outer_loadings[block[l],i]
                mv_hat[, block[l]] <- e
            }
        }
        result <- list(mv_prediction=mv_hat,
                       msep=msep(object, mv_hat, data))
    }
    if(length(mvMissing)==0 & scale=="original"){
        rescale <- function(object, data){
            m <- attr(object$data, "scaled:center")
            s <- attr(object$data, "scaled:scale")
            if(missing(data)){
                t(apply(object$data, 1, function(x) {x * s + m}))
            }
            else t(apply(data, 1, function(x) {x * s + m}))
```

```
        }
        mv_hat <- rescale(object, mv_hat)
        result <- list(mv_prediction=mv_hat,
                       msep=msep(object, mv_hat, rescale(object)))
    }
    return(result)
}

# situation B: with missing observations
# LVs
if(what=="LVs" & length(fsMissing)>0){
    Y_hat <- matrix(0, dim(object$factor_scores))
    for(i in endogen(model)){
        yind <- which(i==model$latent)
        Y_hat <- object$factor_scores[, -yind] %*%
                 object$path_coefficients[-yind, i]
    }
    return(Y_hat)
}
# MVs
if(what=="MVs" & length(fsMissing)>0){
    mv_hat <- matrix(NA, nrow=nrow(data), ncol=ncol(data))
    colnames(mv_hat) <- colnames(data)
    pred <- predecessors(model)
    for(i in endogen(model)){
        block <- model$blocks[[i]]
        m <- attr(data, "scaled:center")[block]
        s <- attr(data, "scaled:scale")[block]
        for(l in 1:length(m)){
            ind <- complete.cases(object$factor_scores[, pred[[i]]])
            e <- object$factor_scores[ind, pred[[i]], drop=FALSE] %*%
                 object$path_coefficients[pred[[i]], i, drop=FALSE] *
                 object$outer_loadings[block[l],i]
            # rescaling
            if(scale=="original") e <- e * s[l] + m[l]
            mv_hat[ind, block[l]] <- e
        }
    }
    for(i in exogen(model)){
        block <- model$blocks[[i]]
        m <- attr(data, "scaled:center")[block]
        s <- attr(data, "scaled:scale")[block]
        for(l in 1:length(m)){
            ind <- complete.cases(object$factor_scores[, i])
            e <- object$factor_scores[ind, i] *
                 object$outer_loadings[block[l],i]
            # rescaling
            if(scale=="original") e <- e * s[l] + m[l]
            mv_hat[ind, block[l]] <- e
        }
    }
```

```
        result <- list(mv_prediction=mv_hat,
                       msep=msep(object, mv_hat, ifelse(scale=="orig",
                                                        rescale(object), data)))
        return(result)
    }
}
}
```

aus dem Paket semPLS:

```
residuals <- function(object, what=c("LVs", "MVs"),
                      scale=c("original", "scaled"),total=FALSE){
    what <- match.arg(what)
    scale <- match.arg(scale)
    model <- object$model
    data <- object$data
    # LVs
    if(what=="LVs"){
        res <- object$factor_scores - predict(object, what, scale, total)
    }
    # MVs
    else{
      if(scale=="scaled"){
        pdata <- predict(object, what, scale, total)$mv_prediction
        res <- data - pdata
      }
      else{
        m <- attr(data, "scaled:center")
        s <- attr(data, "scaled:scale")
        data <- t(t(data) * s) + m
        pdata <- predict(object, what, scale, total)$mv_prediction
        res <- data - pdata
      }
    }
    return(res)




}
```

aus dem Paket heplots, modifiziert um mit Hilfe von ggplot2 Error und Hypothesen Ellipsen zeichnen zu können :

```
heplot<-function (mod, terms, hypotheses, term.labels = TRUE, hyp.labels = TRUE,
    err.label = "Error", variables = 1:2, error.ellipse = !add,
    factor.means = !add, grand.mean = !add, remove.intercept = TRUE,
    type = c("II", "III", "2", "3"), idata = NULL, idesign = NULL,
    icontrasts = c("contr.sum", "contr.poly"), imatrix = NULL,
    iterm = NULL, markH0 = !is.null(iterm), manova, size = c("evidence",
        "effect.size"), level = 0.68, alpha = 0.05, segments = 40,
```

```
    center.pch = "+", center.cex = 2, col = palette()[-1], lty = 2:1,
    lwd = 1:2, fill = FALSE, fill.alpha = 0.3, xlab, ylab, main = "",
    xlim, ylim, axes = TRUE, offset.axes, add = FALSE, verbose = FALSE,
    warn.rank = FALSE, ...)
{
    ell <- function(center, shape, radius) {
        angles <- (0:segments) * 2 * pi/segments
        circle <- radius * cbind(cos(angles), sin(angles))
        if (!warn.rank) {
            warn <- options(warn = -1)
            on.exit(options(warn))
        }
        Q <- chol(shape, pivot = TRUE)
        order <- order(attr(Q, "pivot"))
        t(c(center) + t(circle %*% Q[, order]))
    }
    label.ellipse <- function(ellipse, label, col) {
        if (cor(ellipse)[1, 2] >= 0) {
            index <- which.max(ellipse[, 2])
            x <- ellipse[index, 1] + 0.5 * strwidth(label)
            y <- ellipse[index, 2] + 0.5 * strheight("A")
            adj <- c(1, 0)
        }
        else {
            index <- which.min(ellipse[, 2])
            x <- ellipse[index, 1] - 0.5 * strwidth(label)
            y <- ellipse[index, 2] - 0.5 * strheight("A")
            adj <- c(0, 1)
        }
        text(x, y, label, adj = adj, xpd = TRUE, col = col, ...)
    }
    if (!require(car))
        stop("car package is required.")
    if (packageDescription("car")[["Version"]] >= 2)
        linear.hypothesis <- linearHypothesis
    type <- match.arg(type)
    size <- match.arg(size)
    data <- model.frame(mod)
    if (missing(manova)) {
        if (is.null(imatrix)) {
            manova <- Anova(mod, type = type, idata = idata,
                idesign = idesign, icontrasts = icontrasts)
        }
        else {
            if (packageDescription("car")[["Version"]] >= 2)
                manova <- Anova(mod, type = type, idata = idata,
                  idesign = idesign, icontrasts = icontrasts,
                  imatrix = imatrix)
            else stop("imatrix argument requires car 2.0-0 or later")
        }
    }
```

```
if (verbose)
    print(manova)
if (is.null(idata) && is.null(imatrix)) {
    Y <- model.response(data)
    SSPE <- manova$SSPE
}
else {
    if (is.null(iterm))
        stop("Must specify a within-S iterm for repeated measures designs")
    if (is.null(names(manova$P)))
        names(manova$P) <- names(manova$SSPE)
    Y <- model.response(data) %*% manova$P[[iterm]]
    SSPE <- manova$SSPE[[iterm]]
}
if (!is.null(rownames(SSPE))) {
    response.names <- rownames(SSPE)
}
else {
    response.names <- paste("V.", 1:nrow(SSPE), sep = "")
}
p <- length(response.names)
if (!is.numeric(variables)) {
    vars <- variables
    variables <- match(vars, response.names)
    check <- is.na(variables)
    if (any(check))
        stop(paste(vars[check], collapse = ", "), " not among response variables.")
}
else {
    if (any(variables > length(response.names)))
        stop("There are only ", length(response.names), " response variables.")
    vars <- response.names[variables]
}
if (length(variables) != 2) {
    extra <- if (length(variables) == 1)
        "heplot1d()"
    else if (length(variables) == 3)
        "heplot3d()"
    else "pairs()"
    stop(paste("You may only plot 2 response variables. Use",
        extra))
}
if (missing(terms) || (is.logical(terms) && terms)) {
    terms <- manova$terms
    if (!is.null(iterm)) {
        terms <- terms[grep(iterm, terms)]
    }
    if (remove.intercept)
        terms <- terms[terms != "(Intercept)"]
}
n.terms <- if (!is.logical(terms))
```

```
    length(terms)
else 0
n.hyp <- if (missing(hypotheses))
    0
else length(hypotheses)
n.ell <- n.terms + n.hyp
if (n.ell == 0)
    stop("Nothing to plot.")
Y <- Y[, vars]
gmean <- if (missing(data))
    c(0, 0)
else colMeans(Y)
if (missing(xlab))
    xlab <- vars[1]
if (missing(ylab))
    ylab <- vars[2]
dfe <- manova$error.df
scale <- 1/dfe
radius <- sqrt(2 * qf(level, 2, dfe))
col <- he.rep(col, n.ell)
lty <- he.rep(lty, n.ell)
lwd <- he.rep(lwd, n.ell)
fill <- he.rep(fill, n.ell)
fill.alpha <- he.rep(fill.alpha, n.ell)
fill.col <- trans.colors(col, alpha)
fill.col <- ifelse(fill, fill.col, NA)
E.col <- last(col)
H.ellipse <- as.list(rep(0, n.ell))
H.rank <- rep(0, n.ell)
if (n.terms > 0)
    for (term in 1:n.terms) {
        term.name <- terms[term]
        H <- manova$SSP[[term.name]]
        if (!(all(variables %in% 1:nrow(H)))) {
            warning(paste("Skipping H term ", term.name,
              "(size: ", nrow(H), ")", sep = ""))
            next
        }
        H <- H[variables, variables]
        dfh <- manova$df[term.name]
        factor <- if (size == "evidence")
            lambda.crit(alpha, p, dfh, dfe)
        else 1
        H <- H * scale/factor
        if (verbose) {
            cat(term.name, " H matrix (", dfh, " df):\n")
            print(H)
        }
        H.ellipse[[term]] <- ell(gmean, H, radius)
        H.rank[term] <- qr(H)$rank
    }
```

```
if (n.hyp > 0)
    for (hyp in 1:n.hyp) {
        lh <- linear.hypothesis(mod, hypotheses[[hyp]])
        H <- lh$SSPH[variables, variables]
        dfh <- lh$df
        factor <- if (size == "evidence")
            lambda.crit(alpha, p, dfh, dfe)
        else 1
        H <- H * scale/factor
        if (verbose) {
            cat("\n\n Linear hypothesis: ", names(hypotheses)[[hyp]],
              "\n")
            print(lh)
        }
        H.ellipse[[n.terms + hyp]] <- ell(gmean, H, radius)
    }
E <- SSPE
E <- E[variables, variables]
E <- E * scale[1]
E.ellipse <- ell(gmean, E, radius)
H.ellipse$E <- E.ellipse
if (!add) {
    max <- apply(sapply(H.ellipse, function(X) apply(X, 2,
        max)), 1, max)
    min <- apply(sapply(H.ellipse, function(X) apply(X, 2,
        min)), 1, min)
    factors <- data[, sapply(data, is.factor), drop = FALSE]
    if (!is.logical(factor.means)) {
        factor.names <- colnames(factors)
        which <- match(factor.means, factor.names)
        check <- is.na(which)
        if (any(check))
            stop(paste(factor.means[check], collapse = ", "),
              " not among factors.")
        factors <- factors[, which, drop = FALSE]
    }
    if (!is.logical(factor.means) || factor.means) {
        for (fac in factors) {
            means <- aggregate(Y, list(fac), mean)
            min[1] <- min(min[1], means[, 2])
            max[1] <- max(max[1], means[, 2])
            min[2] <- min(min[2], means[, 3])
            max[2] <- max(max[2], means[, 3])
        }
    }
    if (!missing(offset.axes)) {
        range <- max - min
        min <- min - offset.axes * range
        max <- max + offset.axes * range
    }
    xlim <- if (missing(xlim))
```

```
            c(min[1], max[1])
        else xlim
        ylim <- if (missing(ylim))
            c(min[2], max[2])
        else ylim

    }
    H.ellipse$E <- NULL

    hyp.labels <- if (n.hyp == 0)
        NULL
    else if (!is.logical(hyp.labels))
        hyp.labels
    else if (hyp.labels)
        names(hypotheses)
    else rep("", n.hyp)
    if (n.hyp > 0)
        for (hyp in 1:n.hyp) {
            ell <- n.terms + hyp
            polygon(H.ellipse[[ell]], col = fill.col[ell], border = col[ell],
                lty = lty[ell], lwd = lwd[ell])
            label.ellipse(H.ellipse[[ell]], hyp.labels[hyp],
                col = col[ell])
        }
    if (!add && (!is.logical(factor.means) || factor.means)) {
        for (fac in factors) {
            means <- aggregate(Y, list(fac), mean)
            points(means[, 2], means[, 3], pch = 16, xpd = TRUE,
                ...)
            text(means[, 2], means[, 3], labels = as.character(means[,
                1]), pos = 3, xpd = TRUE, ...)
        }
    }
    if (is.logical(markH0) && markH0)
        mark.H0()
    else if (is.list(markH0))
        do.call(mark.H0, markH0)
    names(H.ellipse) <- c(if (n.terms > 0) term.labels, if (n.hyp >
        0) hyp.labels)
    result <- if (!add)
        list(H = H.ellipse, E = E.ellipse, center = gmean, xlim = xlim,
            ylim = ylim, radius = radius)
    else list(H = H.ellipse, E = E.ellipse, center = gmean, radius = radius)
    class(result) <- "heplot"
    invisible(result)
}



he.rep<-function (x, n)
```

```
{
    if (length(x) < 2)
        x <- rep(x, 2)
    x <- c(rep(x[-1], n)[1:n], x[1])
    return(x)
}



last<-function (x)
{
    x[length(x)]
}


lambda.crit<-function (alpha, p, dfh, dfe, test.statistic = c("Roy", "HLT",
    "Hotelling-Lawley"))
{
    test.statistic <- match.arg(test.statistic)
    switch(test.statistic, Roy = Roy.crit(alpha, p, dfh, dfe),
        HLT = HLT.crit(alpha, p, dfh, dfe), `Hotelling-Lawley` = HLT.crit(alpha,
            p, dfh, dfe))
}



Roy.crit<-function (alpha, p, dfh, dfe)
{
    df1 <- max(p, dfh)
    df2 <- dfe - df1 + dfh
    (df1/df2) * qf(alpha, df1, df2, lower.tail = FALSE)
}


trans.colors <- function (col, alpha = 0.5, names = NULL)
{
    nc <- length(col)
    na <- length(alpha)
    if (nc != na) {
        col <- rep(col, length.out = max(nc, na))
        alpha <- rep(alpha, length.out = max(nc, na))
    }
    clr <- rbind(col2rgb(col)/255, alpha = alpha)
    col <- rgb(clr[1, ], clr[2, ], clr[3, ], clr[4, ], names = names)
    col
}


# typische Befehle zum Aufruf und Anpassen einer Grafik aus der Plotmatrix:
# g<-ggplotmvpairs(ECSImobi,dat=ecsi)
# g$manifest$upperlist$Image[[1]]
```

```
# g$manifest$lowerlist$Quality[[4]]
# g$latent$diaglist$Value[[2]], diaglist gibt Resiudenplots wieder
# g$latent$upperlist$Value[[2]], upperlist gibt scatterplots wieder,
# lowerlist part. Korr.
#

# Plotmatrizen werden mit g$latent$plot bzw. g$manifest$plot aufgerufen

# zuerst Funktion heplot.R laden

#---------------------------------- ECSI Datensatz laden --------------------
library(semPLS)
data(ECSImobi)
ecsi <- sempls(model=ECSImobi, data=mobi)
ecsiBoot <- bootsempls(ecsi, nboot=200, start="ones", verbose=TRUE)

ggplotmvpairs <- function(model, ...){
  UseMethod("ggplotmvpairs", model)
}

#---------------------------------- Funktion ---------------------------------
print.gglist <- function(x, ask=TRUE, ...) {par(ask=ask); x$latent$plot}
# für "latente" Plot Matrix
print.gglist <- function(x, ask=TRUE, ...) {par(ask=ask); x$manifest$plot}
#für manifeste" Plot Matrix

ggplotmvpairs.plsm <- function(model, dat, bar = 20,upper=c("jitter",
                                                        "he","corr","qq"), ...){

library(ggplot2)
library(GGally)


diaglist <- list()
upperlist <- list()
lowerlist <- list()
plot<-list()
ggpair<-list()
diag <- 0
up<-0
lower<-0
pos <- position_jitter(width = 0.33,height = 0.33)

upper <- match.arg(upper)

#Erstellen der "manifesten" Plotmatrizen
for (j in model$latent){

Layout <- grid.layout(nrow = length(model$blocks[[j]]), ncol = length(model$blocks[[j]]))

d<-as.data.frame(ecsi$factor_scores[,j])
```

```
colnames(d)<- j
data<-data.frame(d,mobi[,model$blocks[[j]]])
coln <- colnames(data)
ncol <- ncol(data)



z<-1
k<-1

# Plot Matrix wird initialisiert:
plotMatrix<- ggpairs(data, upper = "blank",lower="blank" ,title = j,axisLabels="none")

# Diagonal-Plots werden erzeugt
for(i in 1:ncol){



  if (nrow(table(data[,i]))<bar){

    D <-data.frame(x=factor(data[,i]))
    bp <- ggplot(D,aes(x)) + geom_bar()
  }
  else
{
    bp <- ggplot(data,aes_string(x=coln[i])) + geom_density()
}

plotMatrix<-putPlot(plotMatrix,bp,i,i)
diag[i] <- list(bp)
}
diaglist[[j]]<-diag




while(k<ncol){

for (i in 1:(ncol-k)){

  #Lower Plots werden erstellt
  lm<-ggplot(data,aes_string(x=coln[k],y=coln[k+i])) + stat_binhex(binwidth = c(1,1))
  + geom_smooth(aes_string(x=coln[k],y=coln[k+i]))
  plotMatrix<-putPlot(plotMatrix,lm,i+k,k)
  lower[z] <- list(lm)

  #Upper Plots werden erstellt
  if (upper == "jitter"){

   jit <- ggplot(data,aes_string(x=coln[k+i],y=coln[k]))
   + geom_point(size = 2,position=pos,col=alpha("blue",0.3)) + geom_smooth(col="red")
```

```
    plotMatrix<-putPlot(plotMatrix,jit,k,i+k)
   up[z] <- list(jit)
    z <- z+1
  }


  else if (upper == "corr"){
    cor <- ggally_cor(data, aes_string(x=coln[k],y=coln[k+i]),colour="gender",corSize = 7)
    plotMatrix<-putPlot(plotMatrix,cor,k,i+k)
    up[z] <- list(cor)
     z <- z+1
  }

  else if (upper == "qq"){
           x <- sort(data[,k])
           y <- sort(data[,k+i])
           x <- approx(1L:length(x),x,n=length(y))$y
           q <- ggplot(data.frame(x,y)) + geom_point(aes(x=x,y=y)) #+geom_abline(intercept = 0,slope
           plotMatrix<-putPlot(plotMatrix,q,k,i+k)
           up[z] <- list(q)
            z <- z+1
  }

    }

  k<-k+1}


lowerlist[[j]] <- lower


# heplots werden erstellt
if(upper == "he"){

k <- 2

while(k<ncol){

for (i in 1:(ncol-k)){

  data[j] <- data[j] + rnorm(nrow(data[j]),0,1e-4)
  mod<-lm(as.matrix(data[2:ncol])~ . , data[1])
  he<-heplot(mod,variables=c(coln[k],coln[i+k]),size ="evidence",alpha=0.05)

  H <- as.data.frame((he$H)$`TRUE`)
  E <- as.data.frame(he$E)
  center <- as.data.frame(t(he$center))
  h <- ggplot(data,aes_string(x=coln[i+k],y=coln[k])) + geom_path(data = H,col="green")
  +geom_path(data = E,col="red") +geom_point(data=center)
  up[z] <- list(h)
  plotMatrix<-putPlot(plotMatrix,h,k,i+k)
```

```
  z <- z + 1

}
k<- k+1
}
}
z<<-z
plot[[j]]<-plotMatrix
upperlist[[j]] <- up

}
# Liste für manifeste Grafiken
ggpair$manifest <- list(lowerlist=lowerlist,diaglist=diaglist,upperlist=upperlist,plot=plot)



# Plot Matrizen für latente Variablen
diaglist <- list()
upperlist <- list()
lowerlist <- list()
plot<-list()

diag <- 0
up<-0

LVdat <- as.data.frame(dat$factor_scores)

# Es wird davon ausgegangen, dass die latente Variable mit den meisten Vorgängern
# als Letztes in der Liste (model$latent) steht, die mit den wenigsten am Anfang
for (j in model$latent[3:length(model$latent)]){

  index <- which(model$latent==j)-1     #Anzahl der Vorgänger (direkt und indirekt)
  pred <- model$latent[1:index]         # Vorgänger (direkt und indirekt)
  Layout <- grid.layout(nrow = index, ncol = index)
  plotMatrix<- ggpairs(LVdat[(index+1):1], upper = "blank",lower="blank",title = j)

#Resiualplots werden erstellt:
for (i in 1:index){
  lm <- lm(as.formula(paste(j, "~ -1 + ", paste(pred[-i], collapse=" + ") )),data=LVdat)
  basic <- ggplot(data=lm, aes(y=.stdresid))
  full <- basic %+% fortify(lm,LVdat)
  resplot <- full +aes_string(x=pred[i]) + geom_point(size = 2,col=alpha("blue",0.3))
  + geom_hline(yintercept = 0, col ="grey" )+geom_smooth(col="red")+scale_y_continuous(breaks=-3:2)
  +scale_x_continuous(breaks=-3:1)+xlim(c(-3,2))+ylim(c(-3,2))
  diag[i]<-list(resplot)

 # Part. Korrelation wird berechnet
  lvcor <<- cor(LVdat[,c(pred,j)]) #Korr. Matrix der Zielgröße+Vorgänger
  lvpcor <- cor2pcor(lvcor) #Part. Korr. Matrix der Zielgröße+Vorgänger
  pcor <- lvpcor[index-i+1,index+1] #Part. Korrelation eins Vorgängers zu Zielgröße
  text <- paste("Part. Corr. :",round(pcor,digits=3))
```

```
  corplot<-ggally_text(text,size=5,alpha=0.8)
  lower[i]<- list(corplot)
  plotMatrix<-putPlot(plotMatrix,corplot,1,1+i)



  plotMatrix<-putPlot(plotMatrix,resplot,index-i+2,1)

}
  lowerlist[[j]] <- lower
  diaglist[[j]]<-diag

k<-1
z<-1

#Scatterplots werden erstellt:
while(k<(index)){

  for (i in 1:(index-k)){

    corplotr <- ggplot(data=LVdat,aes_string(x=pred[k],y=pred[k+i]))+geom_point()+geom_smooth()
    corplotl <- ggplot(data=LVdat,aes_string(y=pred[k],x=pred[k+i]))+geom_point()+geom_smooth()

    plotMatrix<-putPlot(plotMatrix,corplotr,index-i-k+2,index-k+2)
    plotMatrix<-putPlot(plotMatrix,corplotl,index-k+2,index-i-k+2)
    z<-z+1
    up[z]<-list(corplotr)
    }

  k<-k+1


 }
  upperlist[[j]]<-up
  plot[[j]]<-plotMatrix
}

ggpair$latent <- list(diaglist=diaglist,upperlist=upperlist,lowerlist=lowerlist,plot=plot)
class(ggpair) <- "gglist"
invisible(ggpair)

}





}
```

Anhang B

CD-ROM

Die beigelegt CD-ROM enthält den kompletten Code zur Erstellung der Grafiken in Kapitel 4.2.2:

- **heplot.R**: wird für ggplotmvapirs.R benötigt, modifizierte Funktion aus dem R-Paket *heplots*
- **ggplot.R**: enthält die Funktionen zur Erstellung der Parallel Plots, sowie der Verteilungsgrafiken
- **ggplotmvpairs.R** : Funktion zur Erstellung aller Plot Matrizen
- **predict.R**: wird für ggplot.R benötigt; aus dem R-Paket *semPLS*
- **residuals.R**: wird für ggplot.R benötigt; aus dem R-Paket *semPLS*

Erklärung

Hiermit bestätige ich, Peter Mayer, dass ich die vorliegende Bachelor Arbeit selbstständig und ohne Benutzung anderer als den angegeben Hilfsmitteln angefertigt habe.

München, den 26. Juli 2011

Peter Mayer