TECHNISCHE UNIVERSITÄT
CHEMNITZ

# Permutierte isotonische Regression

## Ein statistisches Testverfahren für State-Trace-Analysis

**Fakultät für Human- und Sozialwissenschaften**

**Institut für Psychologie**

**Professur für Forschungsmethodik und Evaluation**

**Abschlussarbeit im Masterstudiengang Psychologie**

Verfasser: Lukas Michael Rommel

Matrikelnummer: 536361

Studiengang: Master Psychologie

Mailadresse: lukas.rommel@psychologie.tu-chemnitz.de

Erstgutachter: Dr. Johannes Titz

Zweitgutachter: Dr. Matthias Beggiato

Datum der Abgabe: 17.06.2024

**Danksagung**

Zu Beginn dieser Arbeit möchte ich die Gelegenheit nutzen mich bei all jenen Menschen zu bedanken, die mich beim Schreiben meiner Masterarbeit unterstützt haben. Besonderer Dank gilt dabei meinem Betreuer Johannes Titz, welcher mich zu dieser Arbeit inspirierte. Johannes hatte jeder Zeit ein offenes Ohr für mich und stand mir stets mit wertvollen Ratschlägen zur Seite. Durch sein Engagement und seine lockere Art hätte ich mir keinen besseren Betreuer vorstellen können. Genauso danke ich meiner Partnerin Michelle Süß und meinem besten Freund Paul Strien, die mich auch in stressigen Zeiten immer wieder auf andere Gedanken gebracht haben. Zu guter Letzt möchte ich meiner Familie danken, auf deren Hilfe und moralische Unterstützung ich jederzeit zählen konnte.

**Vorbemerkung**

Zur Gestaltung dieser Arbeit habe ich mich weitestgehend an das Publication Manual der American Psychological Association (Seventh Edition, 2020) gehalten. Mit dem Ziel die Übersichtlichkeit und Verständlichkeit zu verbessern, habe ich jedoch einige Gestaltungsmerkmale leicht abgeändert (Abbildungen im Fließtext, Zeilenabstand 1,5).

**Zusammenfassung**

*State-Trace-Analysis* (STA) ist eine aufstrebende Methode zur Untersuchung der Fragestellung: Wie viele latente Prozesse bestimmen die Performanz kognitiver Fähigkeiten? Statistische Testverfahren im Rahmen der STA existieren jedoch erst seit Kurzem und in geringer Zahl. Ein neuartiges sowie vielversprechendes dieser Verfahren ist *Permuted Isotonic Regression for state-trace* (PIRST; Benjamin et al., 2019) - ein nonparametrischer Algorithmus basierend auf isotonischer Regression. In dieser Arbeit werden Anwendungsprobleme des Verfahrens offengelegt und aufbauend darauf eine Weiterentwicklung vorgestellt, welche diese Probleme adressiert. Diese beinhaltet einen typischen Signifikanztest mit *p*-Wert. Das erweiterte PIRST wird anschließend im Rahmen einer umfassenden Simulationsstudie genauer betrachtet. Dabei werden Stärken und Schwächen des Verfahrens über ROC-, Sensitivitäts- und Spezifitätsanalysen evaluiert. Abschließend werden konkrete Anwendungsempfehlungen gegeben sowie eine Code-Vorlage für die Statistiksoftware *R* bereitgestellt, welche die direkte Anwendung des Verfahrens ermöglicht.

## Inhaltsverzeichnis

## Abbildungsverzeichnis



## Tabellenverzeichnis

## 1. Einleitung

Seit Beginn der kognitiv-psychologischen Forschung beschäftigt sich eine Vielzahl wissenschaftlicher Bestrebungen mit der Frage, aus welchen Prozessen sich verschiedene mentale Leistungen zusammensetzen. Dabei unterscheiden sich psychologische Theorien nicht nur bezüglich der Art dieser latenten Prozesse, sondern auch bezüglich ihrer Anzahl. Ein klassisches Beispiel, um den Sachverhalt zu verdeutlichen, ist das Wiedererkennungsgedächtnis (*recognition memory*). In Experimenten zur Wiedererkennungsleistung werden Versuchspersonen gebeten sich ein Set von Bildern einzuprägen, anschließend werden sie angewiesen die eingeprägten Bilder aus einem größeren Set wiederzuerkennen. Eine Vielzahl von Theoriemodellen postuliert, dass die Wiedererkennungsleistung durch eine Kombination der Prozesse *Abruf* (die konkrete Erinnerung an die Episode des Lernens) und *Familiarität* (ein von der Lernepisode unabhängiges "bekannt vorkommen") determiniert wird, da "familiäre" Stimuli besser wiedererkannt werden als "nicht-familiäre" (z. B. Jacoby, 1991; Mandler, 1980; Wixted, 2007). Diese Theorien werden auch *dual-process-theories* genannt. Dem entgegen stehen sogenannte *single-process-theories,* welche diese Trennung ablehnen und von einem einzelnen Gesamtprozess (*memory-signal-strength)* ausgehen (Dunn, 2008). Gegensätzliche Theorien wie diese sind es, die Forschungsfelder in Debatten zwischen *single-process-models* und *dual/multiple-process-models* bringen und die Frage aufwerfen, welche Sichtweise nun die plausiblere ist. Derartige Debatten finden sich beispielsweise auch in den Forschungsbereichen *free recall* (Brainerd & Reyna, 2010), *reasoning* (Evans, 2003; Heit & Rotello, 2010), *visual perception* (Goodale & Milner, 1992), *perceptual category learning* (Ashby et al., 1998) und *reading aloud* (Coltheart et al., 2001).

Die methodischen Werkzeuge, welche es ermöglichen solche Fragestellungen rigoros zu untersuchen, werden in der Praxis jedoch nur selten verwendet. Eine veraltete Vorgehensweise, an der noch in vielen Lehrbüchern festgehalten wird, ist die Suche nach *funktionellen Dissoziationen* (Wagenmaker et al., 2012). Hierfür werden in der Regel zwei abhängige Variablen (AVs, z. B. zwei Leistungsmaße) herangezogen, welche in der Theorie dasselbe messen (z. B. *recognition memory*). Anschließend überprüft man in einem experimentellen Design, ob die Manipulation einer unabhängigen Variable (UV; z. B. hohe vs. niedrige *Familiarität*) zu ähnlichen oder unterschiedlichen Ergebnissen auf den AVs führt. Zeigen die Probanden in einem Maß verbesserte Ergebnisse und in dem anderen Maß gleichbleibende Ergebnisse, wird dies folglich als Evidenz für mehrere latente Prozesse gedeutet. Die Idee dabei ist, aufzuzeigen, dass die UV mindestens einen latenten Prozess

beeinflusst, der zur Performanz in einem Maß, jedoch nicht zur Performanz in dem anderen Maß beiträgt und es demnach mehrere Prozesse geben muss (*Einzeldissoziation)*. Um Befunde zu untermauern, wird meistens versucht eine zweite UV zu finden, deren Manipulation genau umgekehrte Effekte auf den AVs anzeigt *(Doppeldissoziation)*.

Obwohl das Konzept auf den ersten Blick nachvollziehbar erscheint, ist das Schlussfolgern aus Dissoziationen, egal welcher Art, eine problematische Methode. Dafür gibt es, wie von Dunn und Kirsner (1988) verdeutlicht, zwei Gründe. Ersterer ist, dass ein *single-process-model* bei derartigen Beobachtungen nicht ausgeschlossen werden kann. Gründe dafür sind Boden-, Plateau- oder Deckeneffekte in der Beziehung zwischen latenten Prozessen und AVs. So können experimentelle Rahmenbedingungen (z. B. Lernzeit bei Gedächtnisaufgaben) Performanz-Level erzeugen, bei denen eine AV weniger auf die Manipulation einer UV reagiert als die andere (z. B., wenn Performanz nahe Perfektion ist). Eine Dissoziation kann demnach anwesend oder nicht anwesend sein, je nachdem welches Performanz-Level man durch seine experimentellen Rahmenbedingungen vorgibt. Der zweite Grund ist, dass selbst wenn keine Dissoziationen beobachtet werden, ein *dual process model* nie ausgeschlossen werden kann. Dies liegt daran, dass Dissoziationen nur unter der Voraussetzung selektiven Einflusses beobachtbar sind. Diese Annahme besagt, dass jede UV genau einen latenten Prozess und jeder latente Prozess genau eine AV beeinflusst. Ist diese nicht gegeben und Dissoziationen werden niemals beobachtet, dann ist die Schlussfolgerung auf ein *single-process-model* nicht zwangsweise richtig.

Glücklicherweise existiert eine validere Analysemethode für derartige Fragestellungen – *State-Trace-Analysis* (STA). Diese wird im nächsten Abschnitt genauer beschrieben. Die STA ist der psychologischen Forschung größtenteils unbekannt, was sehr wahrscheinlich daran liegt, dass für die STA erst seit kurzem statistische Testverfahren zur Verfügung stehen. Das Verfahren was sich vorrangig etabliert hat, nennt sich *conjoint monotonic regression* (CMR, Kalish et al., 2016). CMR besitzt nicht nur eine umfangreiche theoretische Fundierung, sondern enthält ebenfalls einen psychologie-typischen Signifikanztest mit *p*-Wert (Nullhypothese – ein latenter Prozess, Alternativhypothese – mehrere Latente Prozesse).

Drei Jahre nach der Entwicklung von CMR stellten Benjamin et al. (2019) ein Verfahren vor, welches sich in einer umfangreichen Simulationsstudie dominant gegenüber CMR zeigte. Dieses nannten die Autoren *Permuted isotonic regression for state-trace* (PIRST). PIRST basiert im Gegensatz zu CMR ausschließlich auf einer Effektgröße und enthält keine Form von Signifikanztest. Problematischer Weise zeigte sich in den Simulationsexperimenten, dass der Nullwert dieser Effektgröße (ein latenter Prozess)

systematisch variieren kann. Ist der jeweilige Nullwert also, anders wie bei einem Simulationsexperiment, nicht bekannt, kann nicht genau entschieden werden, ob ein Effekt bedeutsam/signifikant vom Nulleffekt abweicht (mehrere Prozesse). Diese Arbeit zielt darauf ab diese Limitation aufzuheben. So wird eine Weiterentwicklung des Verfahrens vorgestellt, diese anschließend anhand einer Simulationsstudie validiert und abschließend über einen Code innerhalb der Statistiksoftware *R* zur freien Anwendung bereitgestellt.

## 2. Theorie

### 2.1 State-Trace-Analysis

Das Konzept der *State-Trace-Analysis* (STA) wurde 1978 erstmalig von Bamber eingeführt. Dieser definierte die STA als Methode, um die funktionalen Beziehungen zwischen UVs und AVs, welche durch Latente Prozesse (LVs) vermittelt werden, zu analysieren. Ausgangspunkt der STA ist der *State-Trace-Plot*, ein zweidimensionales Koordinatensystem, in dem zwei AVs eines experimentellen Designs an x- und y-Achse gegeneinander abgetragen werden. Die AV-Messwertpaare einzelner experimenteller Bedingungen werden innerhalb dieses Koordinatensystems als Punkte abgebildet. Die einzelnen Bedingungen setzen sich dabei aus den Ausprägungen von zwei konzeptuell unterschiedlichen UVs zusammen. Die erste dieser UVs wird *Trace-Variable* genannt und dient zur Manipulation des Performanz-Levels. Um sicherzustellen, dass eine Erhöhung/Verringerung der *Trace-Variable* auch zu Erhöhung/Verringerung in der Performanz führt, sollte die *Trace-Variable* in monotoner Beziehung zu beiden AVs stehen. Die zweite UV wird *Konditions-* bzw. *Dimensionsvariable* genannt. Diese dient zur Manipulation eines separaten latenten Prozesses, welcher in Theorie nicht durch die *Trace-Variable* beeinflusst wird. Angenommen der Untersuchungsgegenstand ist wie im obigen Beispiel *recognition memory*, dann könnte ein *State-Trace-Design* so aussehen, dass man die zwei Konditionen *hohe Familiarität* versus *geringe Familiarität* mit der *Trace-Variable* Lernzeit über zehn unterschiedlich lange Lernzeit-Bedingungen auskartiert. Abbildung 1 visualisiert, wie sich ein solcher *State-Trace-Plot* abzeichnen könnte.

Die Logik der STA ist dabei nun die Folgende: Angenommen eine einzelne latente Variable (LV) bestimmt die Performanz auf den zwei AVs - dann lässt sich jeder Punkt auf $AV_1$ und $AV_2$ als Funktion von LV darstellen und es gibt zu jedem Wert auf LV einen einzelnen zugehörigen Wert auf $AV_1$ und $AV_2$. Somit existiert zu jedem Wert $LV_i$ genau ein Wertepaar ($AV_{1,i}$, $AV_{2,i}$), was bedeutet, dass alle Bedingungspunkte im *State-Trace-Plot* auf einer einzelnen Funktionskurve liegen (Abbildung 2). Unter der Voraussetzung, dass die

**Abbildung 1**

*Hypothetischer State-Trace-Plot mit Zehn Messungen pro Kondition*

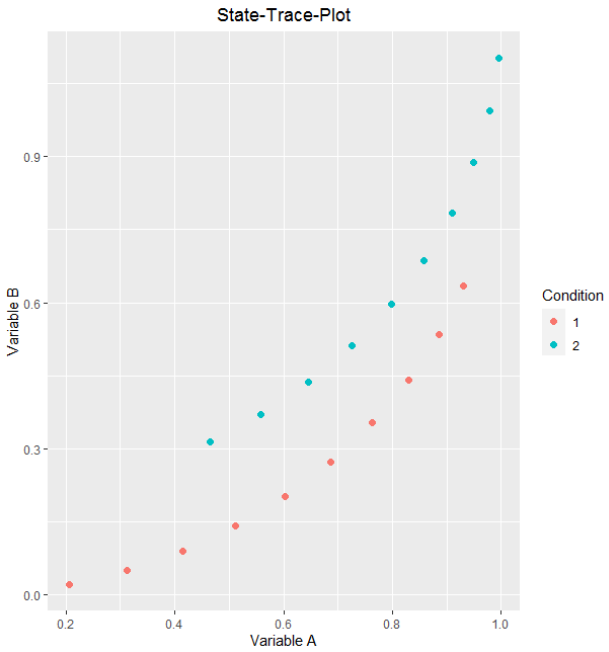
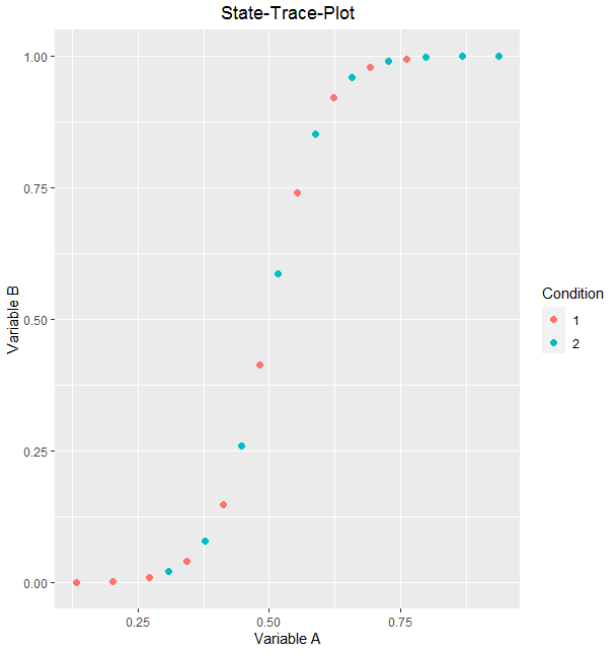



**Abbildung 2**

*Hypothetischer State-Trace-Plot bei einem latenten Prozess*

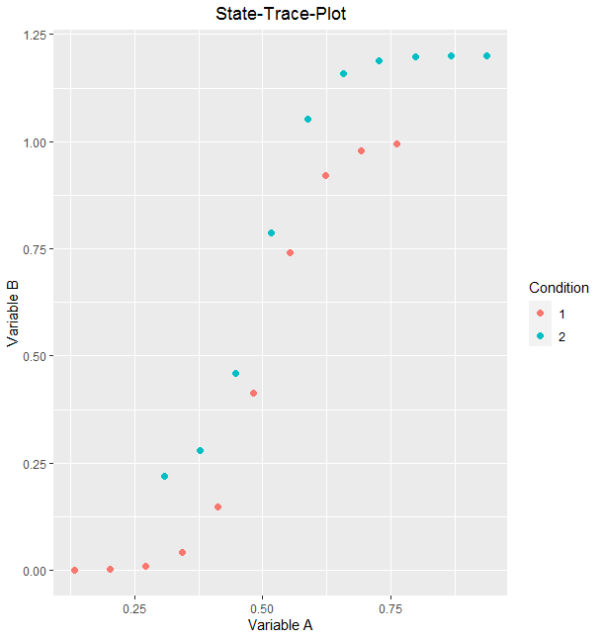



*Trace-Variable* in monotoner Beziehung zu beiden AVs steht, ist diese Funktionskurve dabei selbst stets monoton. Lässt sich demnach keine monotone Kurve erzeugen, auf der alle Bedingungspunkte liegen, dann kann niemals ein Prozess die Struktur der Daten verursacht haben. In diesem Fall beeinflusst die *Konditionsvariable* einen separaten Prozess, sodass

mehrere monotone Funktionskurven (eine pro Ausprägung der *Konditionsvariable*) zugrunde liegen (Abbildung 3).

**Abbildung 3**

*Hypothetischer State-Trace-Plot bei zwei latenten Prozessen*




### *2.1.1 Limitationen*

Wie im vorherigen Abschnitt beschrieben, orientiert sich die STA an der Differenzierung zwischen ein und mehreren monotonen *State-Trace-Funktionen*. Da jede AV allerdings Messfehlern unterliegt, reicht die visuelle Analyse jedoch nur in seltenen Fällen aus, um ein eindeutiges Urteil zu fällen. Die STA unterliegt hierbei einer besonderen Problematik: Die übliche Vorgehensweise zur Reduktion von Messfehlern, Mittelwertbildung über mehrere Versuchspersonen, erhält nicht zwangsweise Monotonie (Prince et al., 2012). So kann aus verschiedenen monotonen *State-Trace-Funktionen* eine nicht monotone und aus nicht monotonen *State-Trace-Funktionen* eine monotone gemittelt werden (Abbildung 4). Die einzige rigorose Möglichkeit Rauschen durch Aggregation zu reduzieren, besteht im Mitteln der Daten derselben Versuchsperson. Diese sollte über mehrere Messungen dieselbe *State-Trace-Funktion* besitzen. Eine weitere Schwierigkeit in der Interpretation von *State-Trace-Plots* besteht darin, dass monotone Funktionen unter informationsleeren Bedingungen möglich sind (Benjamin et al. 2019; Prince et al., 2012). Abbildung 5 visualisiert, wie bei einer ungünstig gewählten *Trace-Variable* ein Plot entstehen kann, in dem zwar alle Punkte durch eine monotone Funktion verbunden werden können, man jedoch keine Aussage treffen kann, ob es mehrere Funktionen gibt. An diesem Beispiel wird deutlich, dass eine

**Abbildung 4**

*Monotonie und Mittelwertbildung*

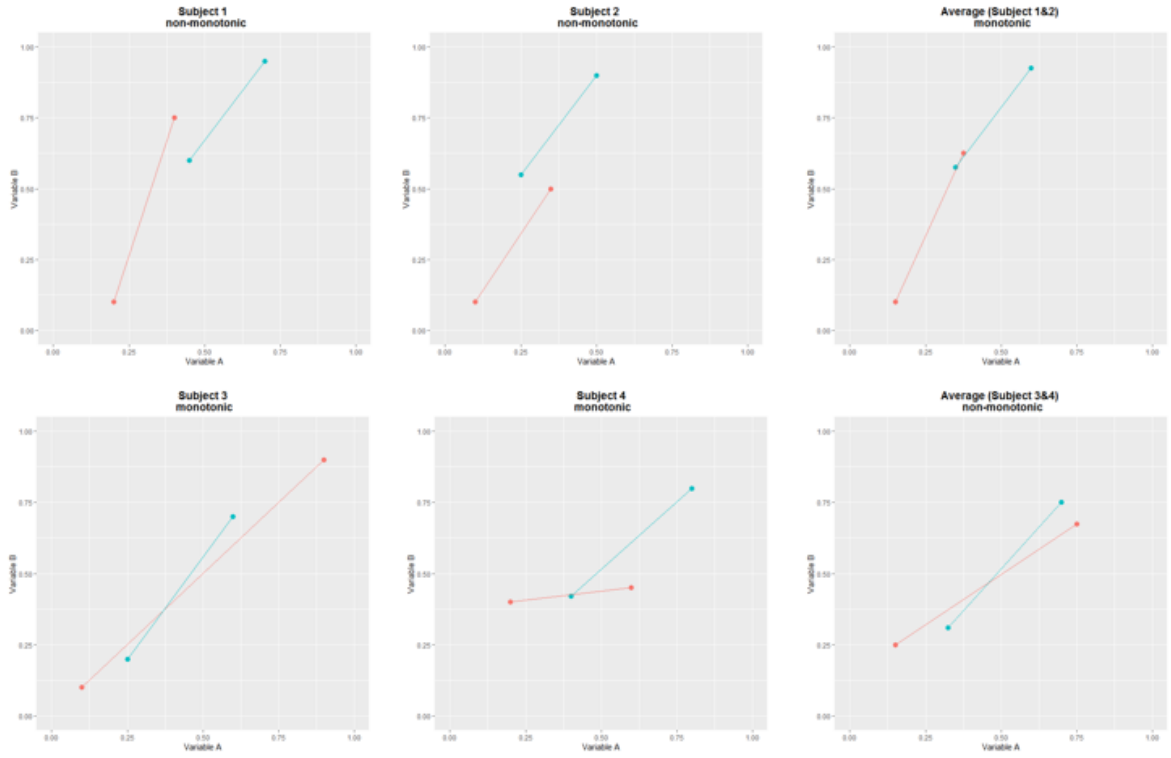
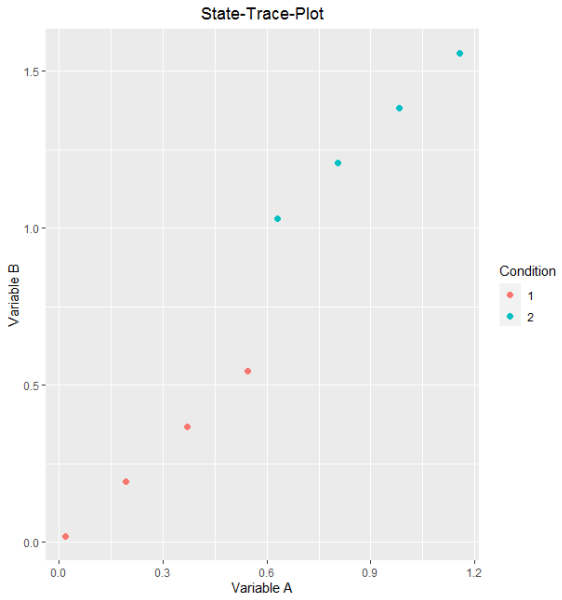



**Abbildung 5**

*State-Trace-Plot ohne Überlappung*




Überlappung der Konditionen (auf x- und/oder y-Achse) eine große Rolle in der Analyse von *State-Trace-Plots* spielt und unbedingt einbezogen werden muss.

### *2.1.2 Isotonische Regression in der State-Trace-Analysis*

Eine Alternative zur rein visuellen Analyse ist die Zuhilfenahme isotonischer Regression. Eine isotonische Regressionskurve wird als die monotone Funktionskurve mit der kleinsten quadrierten Abweichung zwischen den Daten und der gefitteten Funktion definiert (Dunn & Kalish, 2018; Newell & Dunn, 2008). Isotonische Regression wurde von Kalish und Kollegen (2016) für den Einsatz in der STA vorgeschlagen und seitdem unabhängig weiterentwickelt. Eine dieser Weiterentwicklung ist das meistgenutzte Testverfahren der STA – *Conjoint monotonic regression* (CMR, Dunn & Kalish, 2018). CMR funktioniert dabei folgendermaßen: Zu Beginn werden anhand der Daten des *State-Trace-Plots* zwei Modelle erstellt. Im ersten Modell (*multiprocess* - oder *partial order model*) wird für jede der zwei AVs eine individuelle isotonische Regressionskurve gefittet. Im zweiten Modell (*monotonic model*) wird ähnlich vorgegangen, jedoch wird eine Ordnungsbeschränkung auf beiden Variablen definiert. Diese besagt, dass die Regressionsreihenfolge der einen Variable auch der der anderen Variable entsprechen muss, sodass eine monotone Beziehung der AVs forciert wird. Das erstere Modell liefert dabei immer einen besseren oder gleichguten Fit im Vergleich zum zweiten. Der Unterschied zwischen beiden Fits wird anschließend genutzt, um entweder Zuspruch für die *multiple-process-Hypothese* oder für die *single-process-Hypothese* zu generieren. Das Verfahren beinhaltet einen Signifikanztest mit klassischem *p*-Wert. Die dafür notwendige Nullverteilung wird über ein doppeltes Bootstrap-Verfahren generiert. Dies hat zur Folge, dass die Anwendung von CMR eine Varianz und somit aggregierte Daten benötigt.

## 2.2 PIRST (Permutierte isotonische Regression für State-Trace)

2019 präsentierten Benjamin und Kollegen ein weiteres Verfahren, welche isotonische Regression zur Analyse von *State-Trace-Plots* nutzt. Dieses nennen die Autoren PIRST – Permutierte isotonische Regression für State-Trace. Der PIRST-Algorithmus vereint dabei isotonische Regression mit einem Permutationsansatz. Permutiert wird dabei über die Konditionslabels der einzelnen Punkte im *State-Trace-Plot*. Die zentrale Idee ist dabei die Folgende: Angenommen hinter den Daten eines *State-Trace-Plots* liegt in Wahrheit eine einzelne monotone Funktion (*single process*) und es werden für jede Kondition isotonische Regressionskurven gefittet. Gäbe es nun keine Messfehler in den Daten, dann sollte es egal sein, wie man die Konditionslabels der Punkte vertauscht. Die Residuen der Konditionskurven wären für jede Permutation Null, da die einzelnen Konditionen in diesem Fall nur bestimmen, welchen Abschnitt der monotonen Funktion man betrachtet (Abbildung 6). Angenommen dem entgegen liegt hinter jeder Kondition eine separate monotone Funktion

(*dual/multiple process*). Dann führt ein Vertauschen der Konditionslabels nie zu einem besseren Fit (Abbildung 7).

**Abbildung 6**

*Einfluss von Permutation der Konditionslabels auf isotonische Regression (single process)*

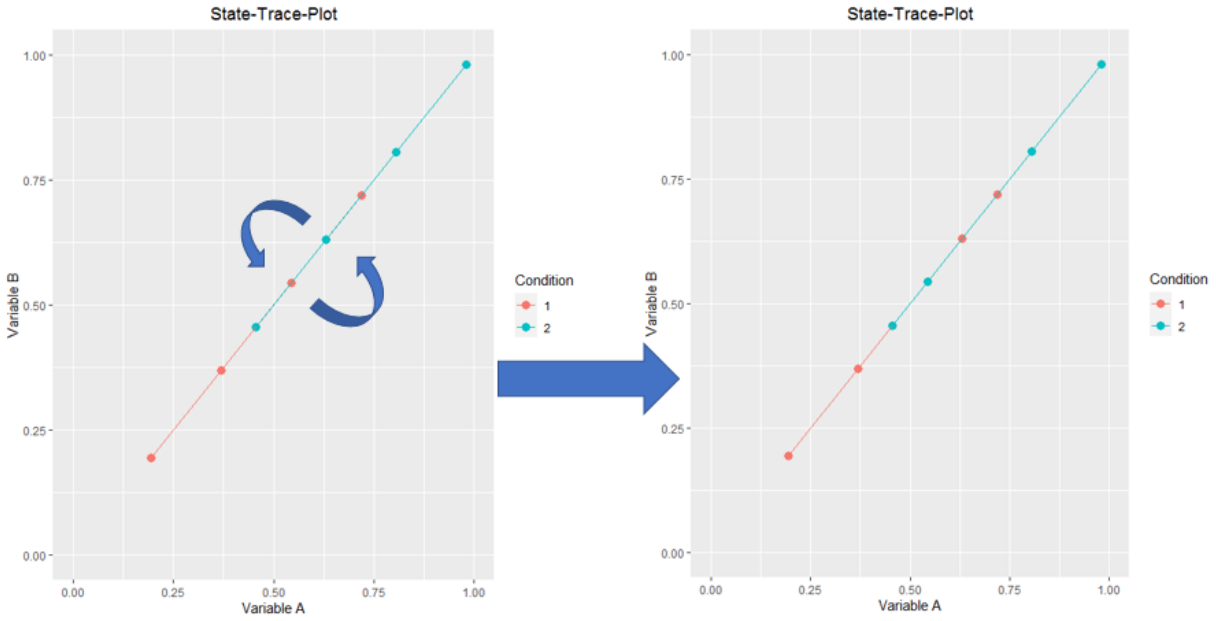
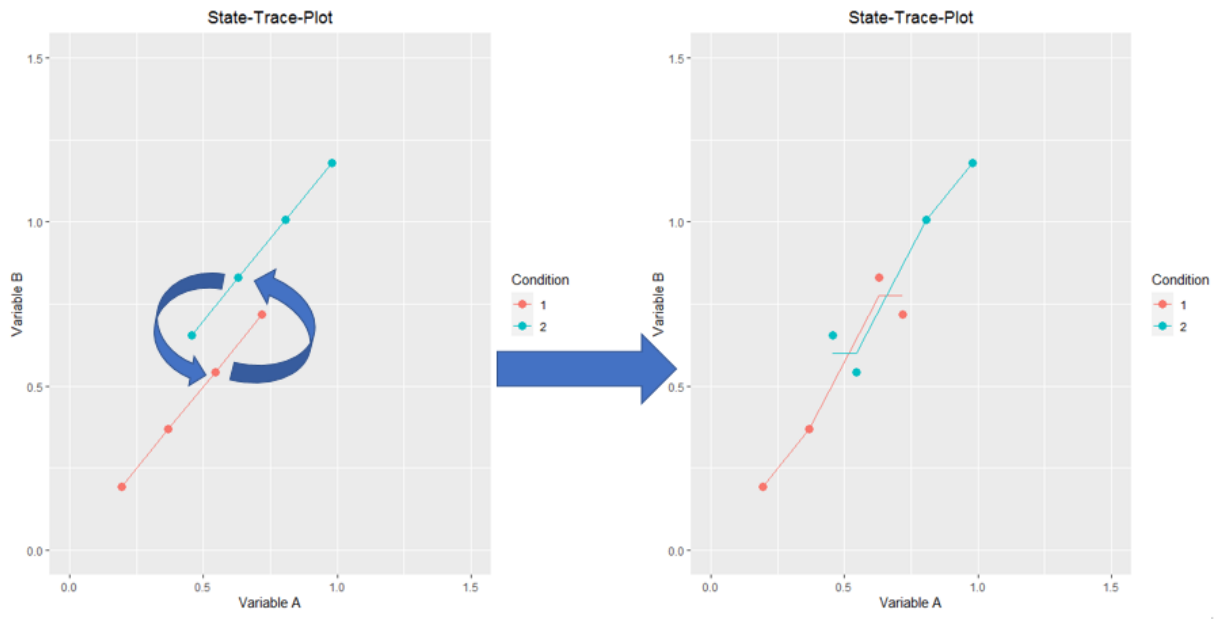



**Abbildung 7**

*Einfluss von Permutation der Konditionslabels auf isotonische Regression (dual process)*




Liegt nun in einem realistischen Setting (mit zufälligen Messfehlern) der Fit des originalen Datensatzes nah an der Mitte der Verteilung der Fits der permutieren Datensätze, wäre dies starke Evidenz für eine einzelne latente Variable. Ist der originale Fit jedoch deutlich besser als der Großteil der Verteilung (je nach gewähltem Konfidenzkriterium), wäre dies im Gegenzug starke Evidenz für mehrere latente Variablen. Demnach ist die Teststatistik

so aufgebaut, dass für jede Versuchsperson der Anteil der permutierten Datensätze bestimmt wird, in denen der Fit schlechter bzw. der Fehler größer war ($P_{greater}$). Anschließend wird daraus der Mittelwert gebildet (*Mean* $P_{greater}$). Liegt dieser Wert nahe .50 gilt dies als Evidenz für einen einzelnen Prozess, da der originale Fit nur mit Zufallswahrscheinlichkeit besser ist. Liegt der Wert deutlich jedoch über .50 wird dies als Evidenz für mehrere Prozesse gedeutet, da der originale Fit überzufällig häufig besser ist. Die Autoren merken an, dass die Werte durch sogenannte *Ties* (permutierter Fit = originaler Fit) auch deutlich unter .50 liegen können. Diese Fälle werden jedoch immer als Evidenz für einen einzelnen latenten Prozess angesehen, da Permutation den Fit nicht überzufällig häufig verschlechtert. Der PIRST-Algorithmus soll im nächsten Abschnitt genauer beschrieben werden.

### *2.2.1 Der Algorithmus*

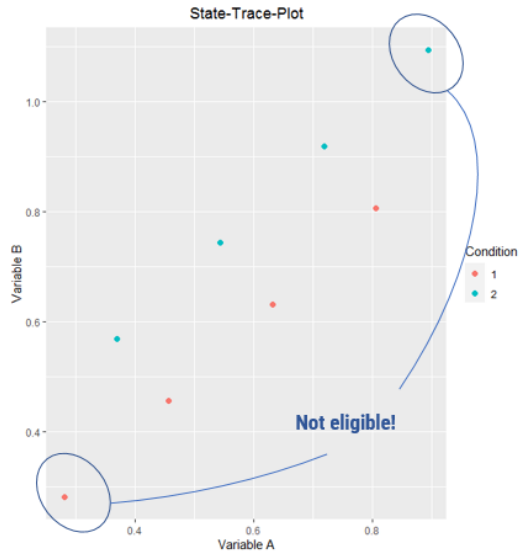
PIRST nach Benjamin et al. (2019) berechnet für jede Versuchsperson die Effektgröße $P_{greater}$ und mittelt diese am Ende über alle Versuchspersonen (anstatt die Ursprungsdaten zu mitteln). Das bedeutet, der eigentliche Algorithmus wird je nach Größe der Stichprobe $N$ auch $N$-mal durchgeführt. Dieser funktioniert dabei folgendermaßen: Man betrachtet die Daten des *State-Trace-Plots* jener Person (2 AV's, $n$ Konditionen sowie $k$ Ausprägungen der *Trace-Variable* = $2 \times (n \times k)$ Messwerte = $n \times k$ Datenpunkte) und rechnet, wie im letzten Abschnitt beschrieben, monotone Regressionen für jede der $n$ Konditionen. Anschließend wird der ungewichtete quadrierte Fehler (*SSE*) der Regressionskurven berechnet und zusammenaddiert.

Der nächste Schritt des Algorithmus besteht darin permutierbare Punkte zu identifizieren. Ein Punkt ist mit einer anderen Kondition permutierbar (*eligible*), sobald er entweder auf der x- oder y-Achse beidseitig von mindestens einem Punkt der anderen Kondition flankiert wird. Ein Punkt ist nicht mit einer anderen Kondition permutierbar, sobald er auf x- und y-Achse größer/kleiner als alle anderen Punkte der Kondition im *State-Trace-Plot* ist (Abbildung 8). Dieser Prozess sorgt dafür, dass Permutationen nicht in den Bereichen stattfinden, in denen die jeweiligen Konditionen auch nicht überlappen. So werden die Permutationen aussortiert, welche das Ergebnis nicht beeinflussen. Dies hängt mit demselben Nicht-Überlappungs-Problem zusammen, welches bereits im Abschnitt zu Limitationen genannt wurde.

Nach diesem Schritt beginnt der Permutationsprozess. Hierbei werden die Konditionslabels der permutierbaren Punkte zufällig neu verteilt. Die Anzahl der Punkte pro Kondition wird dabei konstant gehalten. Ist diese Verteilung dieselbe wie im Originaldatensatz, wird der Prozess wiederholt. Anschließend wird der *SSE* neu berechnet und mit dem originalen *SSE* verglichen. Dieser Vorgang wird, je nach Rechenkapazität, in hoher

**Abbildung 8**

*Nicht-permutierbare Punkte im State-Trace-Plot*




Anzahl wiederholt. Schlussendlich wird die Proportion der Permutationen berechnet, in der der *SSE* größer war als der originale *SSE* - $P_{greater}$.

Die Autoren verzichten danach auf eine konkrete Entscheidungsregel für *single*- oder *multipe/ dual process.* Sie nennen lediglich, dass bei einer einzelnen Funktion und zufälligem Messfehler das durchschnittliche $P_{greater}$ nahe .50 liegt und dieser Wert bei mehreren Funktionen deutlich nach oben abweichen sollte. Diese grobe Heuristik lässt sich zwar auf die Mehrheit, jedoch nicht alle Fälle anwenden. Wie im übernächsten Abschnitt deutlich wird, kann das gemittelte $P_{greater}$ auch bei einer einzelnen monotonen Funktion systematisch über diesem Wert liegen.

### *2.2.2 PIRST vs. CMR*

Um das Verfahren zu validieren, verglichen Benjamin et al. (2019) PIRST und CMR innerhalb einer groß angelegten Simulationsstudie mithilfe von Receiver-operating-characteristic (ROC). Bei sogenannten ROC-Kurven werden die *richtigen Positiven* (korrekt identifizierte multiple Prozesse) gegen die *falschen Alarme* (falsch diagnostizierte Einzelprozesse) über verschiedene Entscheidungskriterien (*mean* $P_{greater}$ von 1 bis 0 bzw. α von 0 bis 1 Schritten von 0.01) des jeweiligen Verfahrens abgetragen. Die Fläche unter der sich ergebenden Kurve (*area under curve*, kurz AUC) sagt anschließend aus, wie gut das Verfahren differenzieren kann bzw. ob überhaupt ein Entscheidungskriterium existiert, unter welchem das Verfahren angemessen performt. Ein AUC von 1 spricht dafür, dass ein Entscheidungskriterium mit perfekter Performanz existiert und ein AUC von .50 dafür, dass bestenfalls Performanz im Zufallsbereich erreicht werden kann. PIRST überzeugte dabei nicht

nur mit vielversprechenden AUC-Werten, diese fielen sogar für den Großteil der Simulationen höher aus als für CMR.

### *2.2.3 Die Problematik mit PIRST*

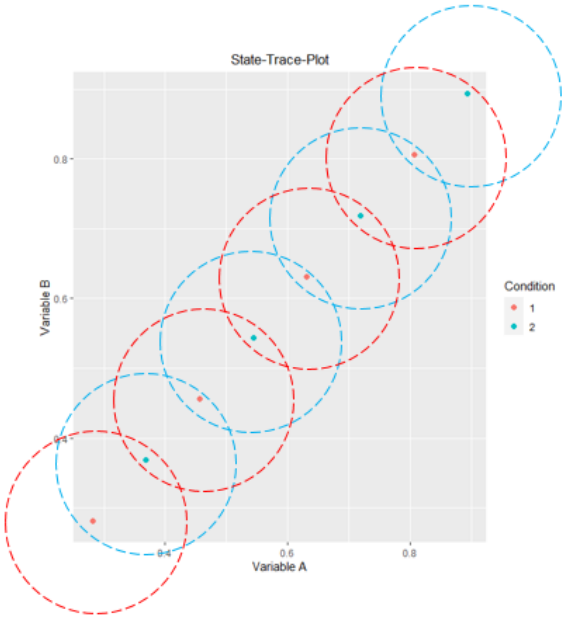
Wie in den ROC-Kurven von Benjamin et al. (2019) zur Geltung kommt, ist PIRST in der Lage sehr genau zwischen einer und mehreren Funktionskurven zu differenzieren. Das bedeutet mathematisch formuliert: Die *Mean* $P_{greater}$-Verteilungen von einer und mehreren Kurven überlagern sich gering bis gar nicht, wenn alle anderen Parameter konstant gehalten werden. Dies sagt jedoch noch wenig darüber aus, ob PIRST an realen Daten effektiv genutzt werden kann. Im Folgenden soll erläutert werden, weshalb die Anwendung in der Praxis problematisch ist.

Um PIRST effektiv anzuwenden, muss auf *mean* $P_{greater}$ ein geeignetes Entscheidungskriterium zwischen 0 und 1 festgelegt werden. Dabei gilt es *richtige Positive* zu maximieren und *falsche Alarme* zu minimieren. Dies erweist sich als kompliziert, da sich entgegen den Annahmen von Benjamin und Kollegen in den Simulationsergebnissen zeigte, dass *Mean* $P_{greater}$ auch bei einer einzelnen monotonen Funktionskurve systematisch über dem empfohlenen Richtwert .50 liegen kann. Ob es zu solchen Abweichungen kommt, hängt von der (wahren) Lage der Bedingungspunkte des *State-Trace-Plots* und der Stärke des Rauschens ab. Dies kann an einem einfachen Beispiel verdeutlicht werden, welches durch Abbildung 9 visualisiert wird. Angenommen die wahren Bedingungspunkte eines *State-Trace-Plots* liegen verteilt wie in Abbildung 9. Ist das Rauschen in diesem Falle so gering, dass es hauptsächlich zu Monotonie-Verletzungen mit unmittelbaren Nachbarbedingungen (in diesem Falle nur Bedingungen der anderen Kondition) kommt, ist der Fit der Permutationen überzufällig häufig schlechter als der originale Fit. So zeigt sich in den Simulationen von Benjamin et al. (2019), dass Nulleffekte von *Mean* $P_{greater}$ = .70 auftreten können, ohne dass dies zufällige Ausreißer sind. Nicht weniger problematisch ist die Tatsache, dass *Mean* $P_{greater}$ durch die oben beschriebenen *Ties* auch deutlich unter .50 liegen kann, wenn mehrere Funktionskurven vorliegen. Demnach ist der Richtwert .50 aus gleich zwei Gründen problematisch. Insbesondere gestaltet es sich durch diese Anomalien schwierig, wenn gar unmöglich, eine andere Entscheidungsregel (z. B. *dual-process* ab *mean* $P_{greater}$ = .70) zu formulieren.

Hinter diesem Gedanken wird schnell klar, dass die vielversprechenden ROC-Kurven und zugehörigen AUC-Werte aus Benjamin et al. (2019) zwar ein Indikator für das diagnostische Potential von PIRST sind, jedoch weniger über den praktischen Nutzen des Verfahrens aussagen. Um dieses Potential voll auszuschöpfen, ist die Konstruktion eines

**Abbildung 9**

*Beispiel für ein State-Trace-Design mit systematisch erhöhtem Mean* $P_{greater}$




Anmerkung. Die unterbrochenen Kreise stellen die durch Rauschen entstehenden Abweichungsbereiche der einzelnen Bedingungen dar.

Nullmodels notwendig, was die Datenstruktur (Lage-Rauschen-Interaktionen und *Ties)* erhält. Anhand eines solchen Modells könnte, unter Wahl einer geeigneten Konfidenz (z. B. 99 % Sicherheit), entschieden werden, ob ein gefundener Effekt bedeutsam ist. Das, was PIRST demnach für die praktische Anwendung fehlt, ist eine Form von Signifikanztest.

## 2.3 Ziele dieser Arbeit

Das Hauptziel dieser Arbeit ist es, PIRST für die Praxis nutzbar zu machen und der STA ein weiteres und vielleicht sogar (im Vergleich zu CMR) valideres Analyseverfahren zur Verfügung zu stellen. Dementsprechend wird im Folgenden ein Signifikanztest vorgestellt, der das bisherige Verfahren erweitert. Dieser wird anschließend an realitätsnahen Simulationsdaten ähnlich derer von Benjamin et al. (2019) validiert.

Ein weiteres Ziel dieser Arbeit ist es Forschende zum Einstieg in die STA zu motivieren. Benjamin et al. (2019) stellten keinerlei Softwarepakete zur Anwendung von PIRST zur Verfügung, sodass Forschende das Verfahren auf eigene Experimentaldaten anwenden können. Im Rahmen dieser Arbeit wird deshalb zusätzlich eine Code-Vorlage innerhalb der freien Statistiksoftware *R* bereitgestellt, welche eine schnelle und unkomplizierte Anwendung des erweiterten PIRST möglich macht.

# 3. Methode

## 3.1 Konstruktion von Nullmodel, Nullverteilung und Signifikanztest

Bei einem Signifikanztest wird überprüft, wie wahrscheinlich ein gefundener Effekt ist, unter der Annahme, dass die Nullhypothese der Realität entspricht. Dafür wird ein statistisches Modell unter der Nullhypothese aufgebaut, in dem der wahre Wert dem Nulleffekt entspricht und mit einer empirisch ermittelten Varianz um diesen Wert schwankt. Die Verteilung der Effektgröße, die sich aus dem Nullmodel ergibt, erlaubt dann eine Wahrscheinlichkeitsaussage für den gefundenen Effekt. Für den PIRST-Signifikanztest wird dementsprechend ein Nullmodel benötigt, aus dem die notwendige Nullverteilung für *Mean* $P_{greater}$ generiert wird. Dies erweist sich als komplex, da *Mean* $P_{greater}$ aus $P_{greater}$ berechnet wird, $P_{greater}$ sich aus der monotonen Beziehung der AV-Messwert-Paare mehrerer Bedingungen (Kondition × Ausprägung *Trace-Variable*) ergibt und jede Bedingung wiederum eine eigene Varianz aufweist.

PIRST nach Benjamin et al. (2019) wurde primär für within-Designs konzipiert, da der Algorithmus den *State-Trace-Plot* jeder Person einzeln betrachtet und separat ein $P_{greater}$ berechnet. Für den Signifikanztest bedeutet dies, dass auch für jede Person ein eigenes Sub-Nullmodel erstellt werden muss. Within-Daten mit einer Messung pro Bedingung (Kondition × Ausprägung *Trace-Variable*) reichen dafür jedoch nicht aus, da aus diesen keine individuelle Varianz berechnet werden kann. So werden für einen Signifikanztest Daten benötigt, in denen zu jeder experimentellen Bedingung mehrere Messwerte pro Person vorhanden sind.

Hinter diesem Gedanken werden die individuellen Nullmodelle folgendermaßen konstruiert: Im ersten Schritt werden Abweichungsverteilungen für jede einzelne Bedingung erstellt. Die Werte aus diesen Verteilungen sind die x/y-Abweichungspaare der einzelnen Messungen vom Mittelwert der Bedingung (Abbildung 10). Diese Verteilungen dienen als Varianzquelle für das jeweilige Sub-Nullmodel. Im zweiten Schritt werden die Daten des *State-Trace-Plots* so verändert, dass sie der Nullhypothese (ein einzelner latenter Prozess liegt hinter den Daten) entsprechen. Dafür wird eine monotone Beziehung zwischen den zwei Abhängigen Variablen über alle Bedingungsmittelwerte erzwungen. Diese wird durch eine vereinfachte Form des CMR-Algorithmus realisiert, welcher Werte auf x- und y-Achse korrigiert. So befinden sich nach dieser Korrektur alle Wertepaare im *State-Trace-Plot* einer Person auf einer einzelnen monotonen Funktionskurve (Abbildung 11). Die Daten dieses *State-Trace-Plots* simulieren die wahren Werte der einzelnen Bedingungen unter der Hypothese, dass es nur einen latenten Prozess gibt. Kombiniert mit der Varianz der

**Abbildung 10**

*Abweichungsverteilung für eine zufällige Bedingung bei 3 Messwiederholungen*

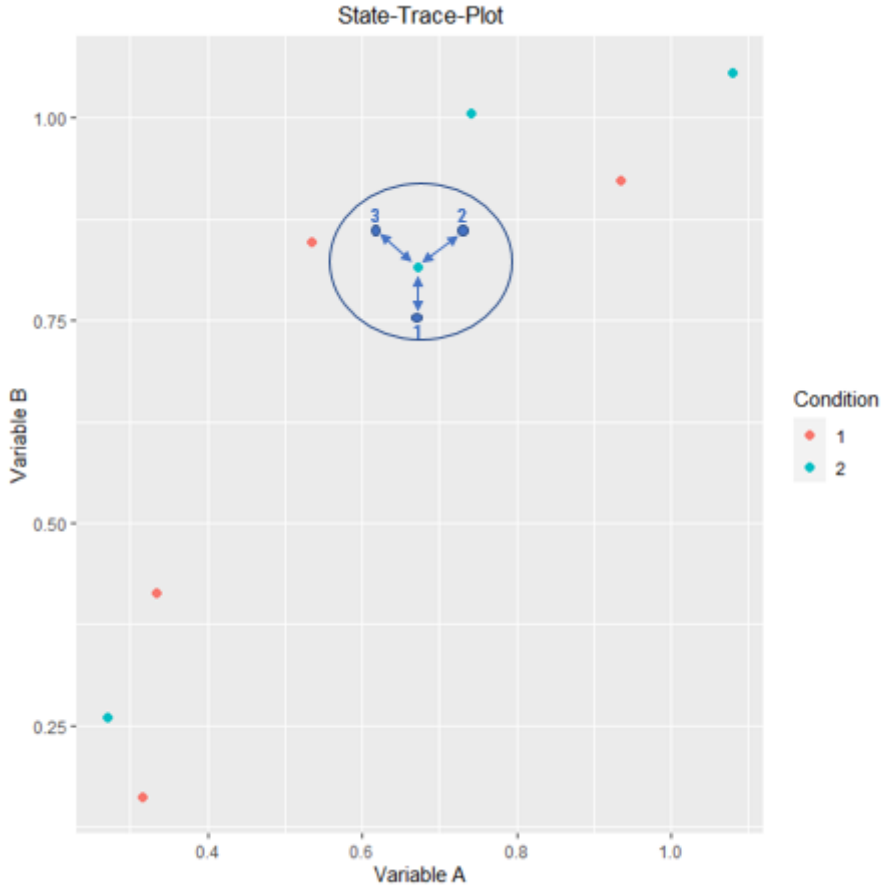
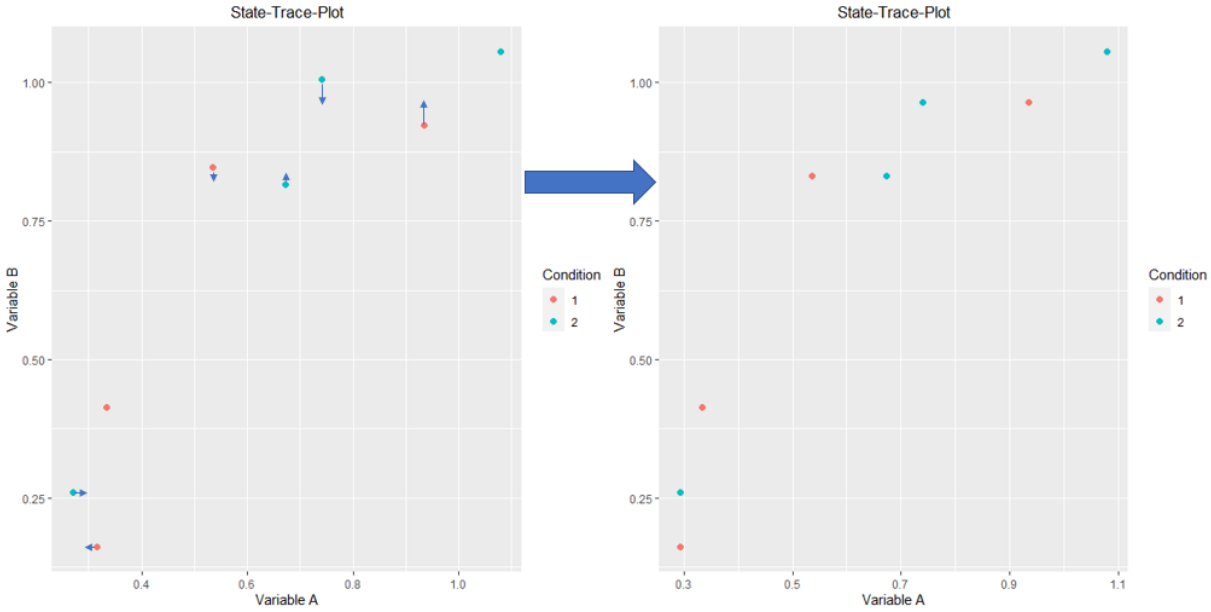



**Abbildung 11**

*Korrektur durch den CMR- Algorithmus*



Abweichungsverteilungen entsteht das individuelle Nullmodell.

Die Nullverteilung von *Mean* $P_{greater}$ wird anhand einer Bootstrap-Prozedur generiert. Das eigentliche Bootstrapping findet dabei auf Bedingungsebene von $P_{greater}$ statt. Je nachdem wie viele Messwiederholungen für eine Person im Originaldatensatz pro Bedingung vorliegen, wird auch diese Anzahl an Messwiederholungen pro Bedingung aus dem Nullmodel generiert. Dabei wird für jede Messwiederholung zufällig ein x/y-Abweichungspaar aus der Abweichungsverteilung der jeweiligen Bedingung gezogen. Anschließend wird der arithmetische Mittelwert aus diesen Abweichungen gebildet und auf die x/y-Werte des jeweiligen Monotonie-korrigierten Bedingungspunktes aufaddiert. Diese Prozedur ist äquivalent zum Erzeugen und Mitteln der Daten mehrerer Monotonie-korrigierter *State-Trace-Plots*. Nachdem dies für jede Bedingung erfolgt ist, erhält man die Daten des einer Person gegeben der Nullhypothese. Dies wird für jede Person im Datensatz wiederholt, um einen Nulleffekt-Datensatz zu erzeugen und anschließend *Mean* $P_{greater}$ zu berechnen. Dieser Prozess wird abhängig von der gewählten Bootstrap-Stichprobengröße $n$-mal wiederholt, sodass man eine Nullverteilung für *Mean* $P_{greater}$ erhält.

Anhand dieser Verteilung wird ein klassischer $p$-Wert für den gefundenen Effekt berechnet und unter Wahl einer geeigneten Konfidenz ein Signifikanztest durchgeführt. Da keine Verteilungsannahmen getroffen werden müssen, ist PIRST sowie seine Erweiterung vollständig nonparametrisch.

## 3.2 Simulationsanalysen

Um das erweiterte PIRST an kompatiblen *State-Trace-Daten* auszuprobieren und die Performanz zu überprüfen, wurden Simulationsdaten im Stil einer typischen Monte-Carlo-Studie erzeugt. Der Fokus lag dabei einerseits darauf, die Simulationen so realitätsnah wie möglich zu gestalten und andererseits darauf, eine besonders große Spanne an Variation experimenteller Rahmenbedingungen abzudecken. Für den Simulationsalgorithmus sowie für das Simulationsdesign wurde sich dafür zu großen Teilen am Vorgehen von Benjamin et al. (2019) orientiert. Beides soll im Folgenden genauer erläutert werden.

### *3.2.1 Simulationsalgorithmus*

Um eine hohe Plausibilität der Simulationen zu ermöglichen, wurde der datenerzeugende Algorithmus so gestaltet, dass alle relevanten Rahmenbedingung eines *State-Trace-Experiments* unabhängig voneinander kontrolliert werden können. Diese sind Anzahl der Konditionen, Anzahl der Bedingungen pro Kondition, Anzahl der Messungen pro Bedingung, Überlappung, Form der Funktionskurven, Interaktionseffekt (räumliche Trennung

der Funktionskurven), Rauschen sowie Anzahl der Versuchspersonen. Die detaillierte Funktionsweise wird im Folgenden erklärt:

Schritt 1: Zu Beginn jeder Simulation eines *State-Trace-Plots* wird eine einzelne lineare Funktionskurve erzeugt. Die Anzahl der Punkte auf dieser Kurve setzt sich aus der Anzahl der Konditionen und Anzahl der Bedingungen pro Kondition zusammen. Diese sind im Simulationsalgorithmus frei variierbar. Die Punkte werden anschließend gleichmäßig im *State-Trace-Raum* verteilt.

Schritt 2: Die Form der Funktionskurve wird angepasst. Diese ist zwischen linear, konkav und sigmoidal variierbar, um unterschiedliche *State-Trace-Funktionen* testen zu können.

Schritt 3: Die Bedingungspunkte der einzelnen Konditionen werden so angeordnet, dass ein frei wählbarer Bereich zwischen den Konditionen überlappt. Dies beeinflusst die Anzahl der permutierbaren Punkte.

Schritt 4: Um einen oder mehrere latente Prozesse zu simulieren, wird eine Interaktionsvariable mit variierbarem Wert eingeführt. Diese macht es möglich eine monotonen Funktionskurve (*single process*) oder mehrere monotone Funktionskurven (*multiple process*) zu erzeugen. Der Wert dieser Variable wird dafür als Konstante auf alle y-Punktwerte einer vorher ausgewählten Kondition aufaddiert. Wenn der Wert auf null gesetzt wird, liegen die Punkte aller Konditionen auf einer monotonen Funktionskurve, wie es bei einem latenten Prozess der Fall wäre. Ist dieser nicht Null, erzeugt dies unterschiedliche Kurven für unterschiedliche Konditionen (versetzt in Richtung der y-Achse) und simuliert somit mehrere latente Prozesse. Je höher der Wert der Interaktionsvariable gewählt wird, desto weiter liegen die Kurven auseinander bzw. desto größer ist der zu findende Effekt.

Schritt 5: Gaußsches Rauschen mit einer frei wählbaren Standardabweichung wird den Datenpunkten auf beiden Achsen hinzugefügt. Dies simuliert den zufälligen Messfehler bei der Erhebung beider AVs in einer Experimentalsituation.

Schritt 6: Der Prozess wird wiederholt, je nachdem wie viele Messwiederholungen pro Person vorgenommen werden sollen.

### *3.2.2 Implementierung*

**3.2.2.1 Generieren der Punkte.** Wie bei Benjamin et al. (2019), ist der Startpunkt eine einzelne lineare Funktion *S*, welche *N* Datenpunkte generiert:

$$S_i = 0{,}7\frac{i}{N} + 0{,}15$$

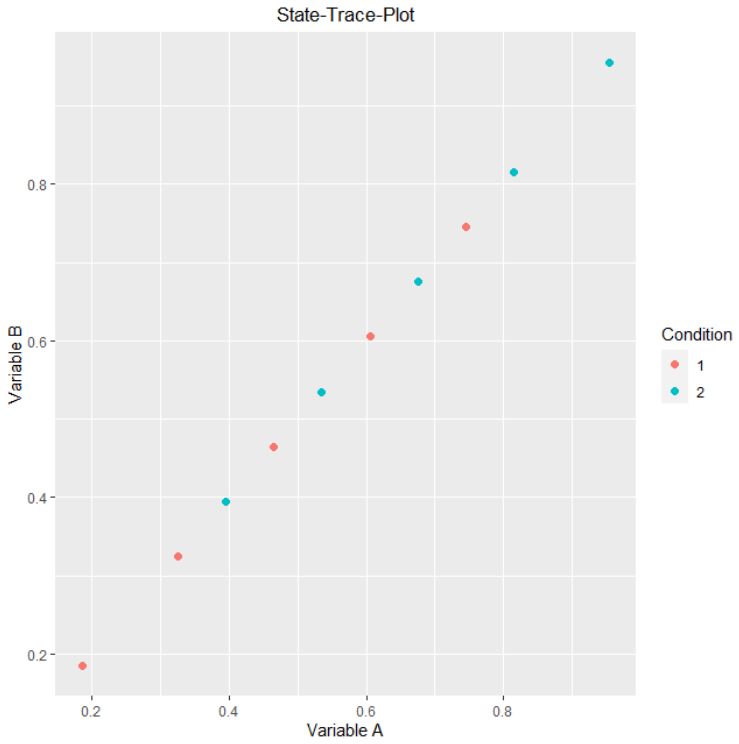
*S* ist dabei so gewählt, dass die Punkte gleichmäßig im Bereich von 0 bis 1 verteilt werden. Die einzelnen *N* Punkte der *m* Konditionen werden aus *S* erzeugt, wofür eine konditionsspezifische Konstante aufaddiert wird:

$$A_{i,C} = S_i - (0{,}35 - 0{,}7 * \frac{C-1}{m} * \frac{p-1/m}{N})$$

*C* steht dabei für die jeweilige State-Kondition und *p* bestimmt die Überlappung zwischen den Konditionen. Bei zwei Konditionen bedeutet der Wert von *p*, dass genau *p* Punkte pro Kondition nicht permutierbar sind (bei mehr Konditionen muss *p* anders adjustiert werden). Ein Beispiel für zwei Konditionen, fünf Punkte pro Kondition und $p = 2$ ist in Abbildung 12 visualisiert.

**Abbildung 12**

*Ausgangspunkt eines State-Trace-Plots mit vier nicht permutierbaren Punkten*




**3.2.2.2 Form der State-Trace-Funktion.** Wenn eine lineare latente Funktion simuliert werden soll, wird keine weitere Transformation von $A_{i,C}$ vorgenommen. In diesem Fall gilt:

$$X_{i,C} = A_{i,C}$$

$$Y_{i,C} = A_{i,C}.$$

Ist eine konkave Funktion gewünscht, werden die Werte von *A* auf den Bereich eines Kreises von 270° bis 360° transformiert. Für konkav aufwärts:

$$Y_{i,C} = \sin\left(\left(90 * A_{i,C} + 270\right) * \frac{\pi}{180}\right) + 1$$

$$X_{i,C} = \cos\left(\left(90 * A_{i,C} + 270\right) * \frac{\pi}{180}\right)$$

Für konkav abwärts:

$$Y_{i,C} = \cos\left(\left(90 * A_{i,C} + 270\right) * \frac{\pi}{180}\right)$$

$$X_{i,C} = \sin\left(\left(90 * A_{i,C} + 270\right) * \frac{\pi}{180}\right) + 1$$

Im Unterschied zum Simulationsalgorithmus von Benjamin und Kollegen wurde zusätzlich die Möglichkeit einer sigmoidalen (S-Förmigen) Funktion hinzugefügt. Diese imitiert die in der STA oftmals diskutierten Boden- und Deckeneffekte und ist daher besonders relevant im Hinblick auf die praktische Anwendung von PIRST. Die Transformation in eine sigmoidale Funktion ergibt sich über:

$$Y_{i,C} = 0{,}5 * (1 + \tanh\,(A_{i,C} * 10 - 5))$$

$$X_{i,C} = A_{i,C}$$

Bzw. für eine diagonal gespiegelte sigmoidale Funktion:

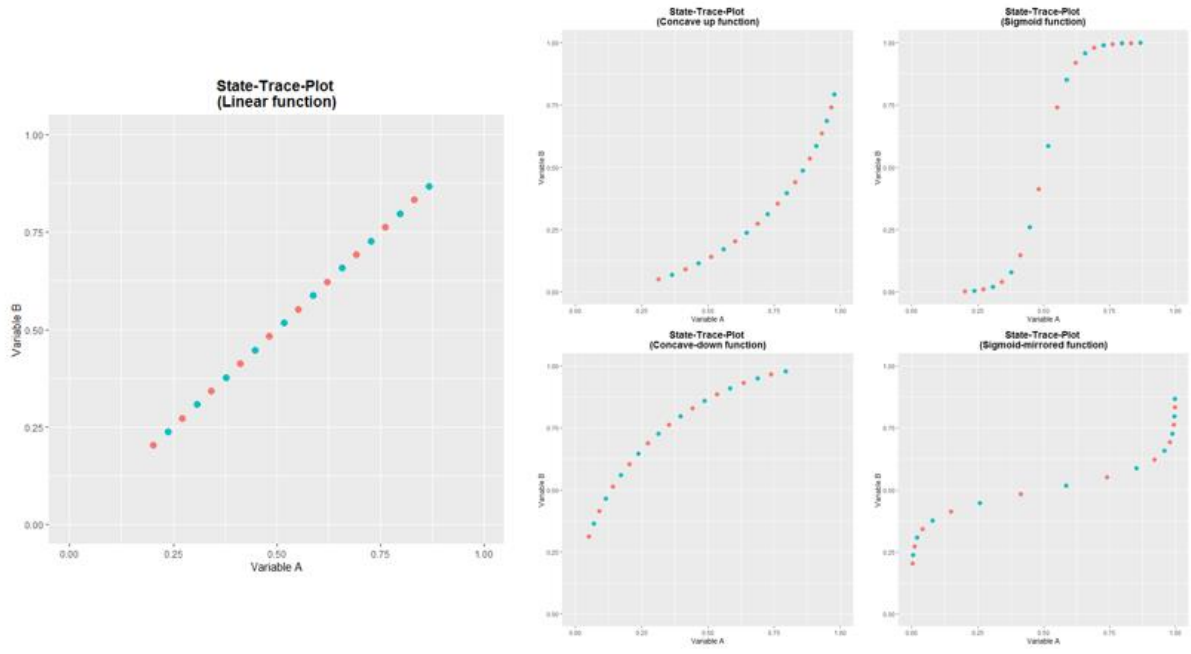
$$Y_{i,C} = A_{i,C}$$

$$X_{i,C} = 0{,}5 * (1 + \tanh(A_{i,C} * 10 - 5))$$

Alle Funktionsformen werden in Abbildung 13 aufgeführt.

**Abbildung 13**

*Formen von State-Trace-Funktionen*



**3.2.2.3 Interaktionskonstante und Rauschen.** Bis zu dieser Stelle des Simulationsalgorithmus liegen alle Punkte auf einer latenten Funktion. Dies kann, wie oben beschrieben, über den Wert der Interaktionskonstante verändert werden. Dafür muss diese auf einen anderen Wert als Null gesetzt werden. Im Unterschied zu Benjamin und Kollegen hat jede Kondition eine eigene Interaktionskonstante, die verändert werden kann. Diese Veränderung wurde in Anpassung an die variierbare Zahl von Konditionen vorgenommen.

Zum Ende wird ein normalverteiltes Rauschen mit dem Mittelwert Null und einer frei wählbaren Standardabweichung auf jedes $X_{i,C}$ und $Y_{i,C}$ aufaddiert, um die Simulation eines *State-Trace-Plots* abzuschließen.

### *3.2.3 Simulationsdesign*

Um eine möglichst hohe Realitätsnähe zu gewährleisten, müssen die simulierten experimentellen Parameter auch in realitätsnahen Bereichen gewählt werden (oder zumindest einen referenziellen Rahmen bieten). Wie bereits erwähnt, wurde sich hierbei ebenfalls, wenn auch mit Anpassungen, am Vorgehen von Benjamin et al. (2019) orientiert.

Bei der Anzahl der Konditionen wurde sich auf den konstanten Wert zwei beschränkt. Dies liegt daran, dass es bereits ohne Variation der Konditionen eine große Vielzahl von Kombinationen anderer Simulationsparameter gibt, die es vorrangig zu untersuchen gilt. Bei der Anzahl der Performanz-Bedingungen (Bedingungen pro Kondition) wurde sich für die Werte vier, acht und 15 entschieden. Höhere Werte, wie bei Benjamin et al. wurden in dieser Studie außer Acht gelassen, um die Simulationen in einem zeitlich umsetzbaren Setting zu halten. Zur Verdeutlichung: Bei drei vollständigen Messwiederholungen, zwei Konditionen und 15 Bedingungen pro Kondition gäbe es zu einer Versuchsperson bereits 180 Messwerte (90 pro AV). Bei der Überlappung wurde sich für eine hoch-Kategorie (ca. 90%) und eine niedrig-Kategorie (ca. 50%) entschieden, um *State-Trace-Plots* mit hohem sowie geringem Informationsgehalt zu simulieren. Für diese Wahl gibt es zwei Gründe. Ersterer besteht darin, dass in der Planung von *State-Trace-Designs* immer eine hohe Überlappung angestrebt werden sollte und auch in großen Teilen durch umfassende theoretische Vorarbeit sichergestellt werden kann (Prince et al. 2012). Zweiterer ist, dass bei zu geringer Überlappung die Aussagekraft der Daten allgemein in Frage zu stellen ist und die Ausprägungen der *Trace-Variable* anders gewählt werden sollten. Dementsprechend stellt die Wahl einen plausiblen Referenzrahmen mit einerseits optimaler und andererseits gerade noch akzeptierbarer Voraussetzung dar. So ergeben sich bei vier Bedingungen pro Kondition $p_{high}$ = 1 und $p_{low}$ = 2, bei acht Bedingungen pro Kondition $p_{high}$ = 1 und $p_{low}$ = 4 sowie bei 15 Bedingungen pro Kondition $p_{high}$ = 2 und $p_{low}$ = 7. Für die Standardabweichung des Gauß-

Rauschens wurden die Werte $n_{small}$ = 0.10, $n_{medium}$ = 0.20 und $n_{large}$ = 0.40 ausgewählt. So macht die Standardabweichung 10%, 20% oder 40% des *State-Trace-Raumes* aus. Diese Werte wurden dem Simulationsdesign von Benjamin et. al (2019) entnommen und zeigen sich verglichen mit Standardabweichungen vergangener *State-Trace-Experimente* (z. B. Prince et al. (2012), Kalish et al. (2016)) als realistisch. Die Werte der Interaktionskonstante $I_{Null}$ = 0.00, $I_{small}$ = 0.10, $I_{medium}$ = 0.15 und $I_{large}$ = 0.20 wurden ebenfalls aus dem Design von Benjamin und Kollegen übernommen. Hinter dem Gedanken, dass diese Werte im engeren Sinne den Mittelwertsunterschied der Kurven darstellen und sich in Relation zum Rauschen psychologietypische Effektgrößen von *d* = 0.25-2.00 ergeben, ist die Plausibilität dieser Werte ebenfalls gewährleistet. Die Form der *State-Trace-Funktion* variiert in den Simulationen über alle implementierten Möglichkeiten (linear, konkav-aufwärts und sigmoidal), um ein möglichst breites Spektrum an vorstellbaren Funktionen abzudecken. Für die Anzahl der Messwiederholungen pro Bedingung wurden die Werte drei, fünf und zehn gewählt. Diese Variation soll aufzeigen wie viele Messwiederholungen mindestens notwendig sind, um genaue Ergebnisse zu generieren. Die Anzahl an Versuchspersonen wurde, statt auf 100 wie bei Benjamin et al. (2019), auf den festen Wert 30 gesetzt. Diese Anpassung wurde vorgenommen, um die Anzahl der Gesamtmesswerte, welche durch die Messwiederholungen stark ansteigen, in etwa gleich zu halten. Insgesamt ergibt sich demnach ein Umfang von 648 verschiedenen Simulationsexperimenten mit einer Stichprobengröße von je 30 Probanden.

### *3.2.4 Performanz-Kriterien*

Um Aussagen über die Differenzierungsfähigkeit des erweiterten PIRST treffen zu können, werden AUC-Werte berechnet. Die zugehörigen ROC-Kurven werden diesmal jedoch, ähnlich wie für CMR in Benjamin et al. (2019), über α-Entscheidungskriterien erstellt. So können spezifische Bedingungen identifiziert werden, unter denen das Verfahren zufriedenstellend performen kann. Zusätzlich werden die Sensitivität (Anteil der *tatsächlichen Positiven,* die *korrekt* als solche erkannt werden) und Spezifität (Anteil der *tatsächlichen Negativen,* die *korrekt* als solche identifiziert werden) für α = .05 erfasst. Diese erlauben in Kombination mit den AUC-Werten ein Urteil darüber, ob sich ein α von 5% unter geeigneten Bedingungen auch als geeignetes Entscheidungskriterium herausstellt oder angepasst werden muss.

## 4. Ergebnisse

Im Folgenden werden die Ergebnisse für *Mean* $P_{greater}$, AUC-Werte, Sensitivität und Spezifität über die 648 Kombinationen der Simulationsparameter vorgestellt. Da es sich schwierig gestaltet jeden Wert einzeln zu betrachten, werden allgemeine Muster in den Daten

fokussiert. Einzelne Kennwerte können den jeweiligen Tabellen entnommen werden, welche jeweils mit aufgeführt sind.

## 4.1 Mean $P_{greater}$

AUC-Werte, Sensitivität und Spezifität werden durch die Verteilungen von *Mean* $P_{greater}$ bestimmt. Um ein besseres Verständnis zu schaffen, werden demnach zuerst die Ergebnisse von *Mean* $P_{greater}$ vorgestellt (Tabelle 1-3).

### *4.1.1 Einfluss der Interaktionskonstante*

Über annähernd alle Simulationen zeigt eine Erhöhung der Interaktionskonstante auch eine eindeutige Erhöhung von *Mean* $P_{greater}$. So ist *Mean* $P_{greater}$ im selben Simulationsexperiment größer, wenn die Interaktionskonstante von Null auf einen höheren Wert gesetzt wird. Je stärker die Interaktionskonstante erhöht wird, desto weniger überlappen die Konfidenzintervalle der Verteilungen von Effekt und Nulleffekt. Diese intendierte Auswirkung zeigt sich stärker bei wenig Rauschen, mehr Bedingungen pro Kondition sowie mehr Messungen pro Bedingung. Die Anzahl der Bedingungen pro Kondition zeigt sich als stärkster Faktor darin, wie groß die Differenz von Effekt und Nulleffekt ausfällt.

Beläuft sich der Wert der Interaktionskonstante auf $I = 0$ (Eine zugrundeliegende monotone Funktionskurve) befindet sich wie auch bei Benjamin et al. (2019) der Großteil der Werte von *Mean* $P_{greater}$ bei ca .50. Dennoch lässt sich auch in dieser Studie beobachten, dass es vereinzelt systematische Abweichungen von diesem Wert gibt, sodass die Werte von *Mean* $P_{greater}$, auch wenn kein Effekt in den Daten vorliegt, deutlich über oder unter .50 liegen können. Die Beobachtung, dass sich unter steigender Interaktionskonstante auch steigende Werte von *Mean* $P_{greater}$ ergeben, wird davon jedoch nicht beeinflusst.

### *4.1.2 Einfluss von Rauschen*

Bei der Variation des Rauschens fällt Folgendes auf: Je stärker das Rauschen innerhalb einer Simulation ansteigt, desto stärker nähern sich die Werte von *Mean* $P_{greater}^{Effekt}$ und *Mean* $P_{greater}^{Null}$ einander an. Es gibt dabei keine eindeutigen Hinweise darauf, dass die Werte von *Mean* $P_{greater}$ für dieselbe Simulation mehr oder weniger stark streuen, wenn das Rauschen variiert wird. Dies wird bei Betrachtung der 95%-Konfidenzintervalle deutlich.
Zusätzlich wird die im Theorieteil verdeutlichte Problematik der von .50 abweichenden Nulleffekte an einer Vielzahl von Simulationen sichtbar. So nehmen Nulleffekte Werte bis zu .70 an, wenn Rauschen und Lage der Bedingungspunkte so im Verhältnis stehen, dass es hauptsächlich zu Monotonie-Verletzungen zwischen Bedingungen kommt. Ebenfalls lassen sich Werte nahe Null beobachten, wenn das Rauschen so gering ist, dass es generell kaum zu Monotonie-Verletzungen kommt.

## Tabelle 1

*Mean P-Greater - 3 Messungen pro Bedingung*

**Low Noise**

| Interaction | Size | Low Overlap 0 | Low Overlap S | Low Overlap M | Low Overlap L | High Overlap 0 | High Overlap S | High Overlap M | High Overlap L |
|---|---|---|---|---|---|---|---|---|---|
| Linear | 15 | .47 [.37, .57] | .83 [.75, .90] | .95 [.91, .98] | .98 [.96, .99] | .51 [.37, .60] | .88 [.80, .93] | .98 [.95,.99] | .99 [.99, 1] |
| | 8 | .56 [.44, .66] | .73 [.62, .81] | .84 [.72, .91] | .92 [.86, .95] | .60 [.49, 68] | .80 [.71, .87] | .91 [.84, .96] | .97 [.93, .99] |
| | 4 | .39 [.28, .51] | .47 [.38, .57] | .55 [.43, .65] | .61 [.49, .71] | .50 [.39, .59] | .61 [.49, .70] | .69 [.60, .76] | .77 [.68, 83] |
| Concave | 15 | .48 [.36, .57] | .80 [.70, .87] | .91 [.84, .94] | .97 [.91, .99] | .49 [.40, .59] | .83 [.74, .89] | .95 [.89, .98] | .99 [.97, .99] |
| | 8 | .54 [.44, .66] | .71 [.59, .80] | .81 [.74, .87] | .88 [.83, .94] | .55 [.46, .65] | .72 [.62, .82] | .84 [.75, .91] | .92 [.86. .96] |
| | 4 | .38 [.47, .28] | .46 [.33, .55] | .53 [.43, .64] | .58 [.48, .69] | .46 [.36, .55] | .55 [.44, .66] | .64 [.54, .73] | .71 [.61, .69] |
| Sigmoid | 15 | .41 [.29, .52] | .56 [.46, .66] | .67 [.55, .77] | .78 [.69, .84] | .46 [.37, .56] | .66 [.55, .75] | .81 [.71, .88] | .90 [.83, .95] |
| | 8 | .48 [.37, .58] | .55 [.46, .64] | .61 [.51, .73] | .68 [.55, .76] | .54 [.43, .65] | .66 [.56, .76] | .76 [.67, .85] | .84 [.73, .91] |
| | 4 | 45 [.33, .54] | .41 [.31, .50] | .44 [.33, 53] | .47 [.40, .54] | .52 [.42, .60] | .52 [.40, .61] | .56 [.43, .63] | .62 [.51, .71] |

**Medium Noise**

| Interaction | Size | Low Overlap 0 | Low Overlap S | Low Overlap M | Low Overlap L | High Overlap 0 | High Overlap S | High Overlap M | High Overlap L |
|---|---|---|---|---|---|---|---|---|---|
| Linear | 15 | .40 [.30, .52] | .67 [.55, .76] | .79 [.66, .88] | .89 [.81, .94] | .49 [.35, .58] | .70 [.58, .79] | .83 [.72, .91] | .93 [.87, .97] |
| | 8 | .47 [.34, .60] | .61 [.51, .69] | .69 [.54 .79] | .78 [.70, .86] | .51 [.41,.60] | .62 [.49, .71] | .72 [.62, .81] | .82 [.75, .90] |
| | 4 | .48 [.33, .58] | .51 [.38, .63] | .57 [.43, .69] | .61 [.48, .72] | .54 [.42, .65] | .58 [.48, .67] | .62 [.50, .71] | .69 [.58, .80] |
| Concave | 15 | .48 [.37, .59] | .70 [.59, .82] | .80 [.69, .89] | .88 [.79, .95] | .47 [.36, .58] | .68 [.56, .79] | .82 [.73, .89] | .91 [.85, .96] |
| | 8 | .49 [.33, .59] | .62 [.52, .72] | .70 [.60, 78] | .76 [.63, .85] | .49 [.38, .61] | .60 [.49, .70] | .69 [.59, .77] | .78 [.67, .85] |
| | 4 | .46 [.32, .58] | .52 [.40, .62] | .56 [.45, .70] | .60 [.48, .70] | .50 [.39, .59] | .55 [.44, .64] | .58 [.49, .68] | .63 [.54, .73] |
| Sigmoid | 15 | .41 [.29, .50] | .51 [.41, .60] | .59 [.47, .71] | .67 [.57, .77] | .45 [.35, .55] | .53 [.41, .62] | .62 [.49, .71] | .72 [.60, .80] |
| | 8 | .43 [.33, .57] | .48 [.38, .59] | .53 [.40, .66] | .59 [.49, .69] | .47 [.36, .57] | .52 [.36, .61] | .58 [.50, .67] | .64 [.53, .73] |
| | 4 | .45 [.32, .55] | .45 [.34, .54] | .46 [.36, .57] | .47 [.34, .59] | .50 [.37, .60] | .50 [.41, .61] | .53 [.41, .65] | .57 [.45, .68] |

**High Noise**

| Interaction | Size | Low Overlap 0 | Low Overlap S | Low Overlap M | Low Overlap L | High Overlap 0 | High Overlap S | High Overlap M | High Overlap L |
|---|---|---|---|---|---|---|---|---|---|
| Linear | 15 | .48 [.35, .57] | .63 [.53, .74] | .72 [.57, .82] | .80 [.70, .89] | .49 [.38, .57] | .60 [.51, .70] | .68 [.52, .79] | .78 [.66, .86] |
| | 8 | .46 [.32, .57] | .56 [.45, .66] | .61 [.49, .72] | .68 [.54, .79] | .50 [.36, .58] | .55 [.41, .64] | .60 [.49,.72] | .66 [.57, .75] |
| | 4 | .47 [.36, .58] | .51 [.34, .60] | .55 [.40, .66] | .57 [.41, .69] | .50 [.38, .61] | .53 [.43,.64] | .55 [.44,.65] | .59 [.47, .72] |
| Concave | 15 | .52 [.42, .62] | .67 [.50, .79] | .75 [.65, .82] | .81 [.69, .89] | .48 [.35, .58] | .59 [.48, .71] | .68 [.58, .78] | .76 [.66, .84] |
| | 8 | .49 [.37, .58] | .59 [.49, .69] | .64 [.51, .76] | .69 [.55, .80] | .49 [.39, .61] | .54 [.41, .63] | .58 [.42, .69] | .65 [.53, .74] |
| | 4 | .47 [.33, .57] | .53 [.37, .64] | .55 [.42, .67] | .57 [.45, .69] | 48 [.34, .58] | .51 [.40, .60] | .55 [.44, .65] | .57 [.45, .68] |
| Sigmoid | 15 | .51 [.39, .63] | .61 [.49, .73] | .67 [.56, .79] | .73 [.60, .83] | .45 [33,.55] | .52 [.43, .63] | .58 [.45, .67] | .63 [.54, .70] |
| | 8 | .47 [.33, .61] | .52 [.43, .64] | .56 [.46, .68] | .61 [.49, .71] | .46 [.35, .56] | .48 [.35, .58] | .52 [.41, .62] | .56 [.44, .64] |
| | 4 | .45 [.35, .55] | .46 [.35, .58] | .47 [.34, .58] | .50 [.36, .62] | .46 [.36, .55] | .47 [.34, .57] | .50 [.38, .60] | .50 [39, .62] |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist *Mean* $P_{greater}$

## Tabelle 2

*Mean P-Greater - 5 Messungen pro Bedingung*

**Low Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .51 [.41, .61] | .90 [.84, .95] | .98 [.96, .99] | .99 [.99, 1] | .54 [.44, .63] | .94 [.90, .98] | .99 [.97, 1] | .99 [.99, 1] |
| | 8 | .59 [.48, .70] | .79 [.71, .88] | .90 [.85, .93] | .95 [.91, .98] | .67 [.55, .76] | .87 [.78, .92] | .95 [.91, .98] | .98 [.97, .99] |
| | 4 | .28 [.19, .39] | .42 [31, .50] | .50 [.43, .58] | .56 [.47, .65] | .36 [.26, .46] | .59 [.52, .66] | .68 [.62, .74] | .74 [.69, .79] |
| Concave | 15 | .51 [.40, .63] | .86 [.80, .92] | .96 [.92, .98] | .98 [.97, .99] | .50 [.36, .57] | .89 [.82, .94] | .98 [.94, .99] | .99 [.99, 1] |
| | 8 | .56 [.46, .67] | .75 [.66, .84] | .86 [.78, .92] | .93 [.88, .96] | .57 [.48, .66] | .79 [.69, .86] | .90 [.85, .95] | .96 [.93, .98] |
| | 4 | .27 [.19, .36] | .41 [.29, .49] | .51 [.43, .60] | .55 [.44, .65] | .38 [.26, .47] | .56 [.46, .65] | .66 [.57, .72] | .72 [.64, .78] |
| Sigmoid | 15 | .42 [.34, .51] | .61 [.51, .72] | .76 [.65, .84] | .85 [.75, .91] | .47 [.36, .57] | .74 [.60, .82] | .88 [.79, .93] | .96 [.92, .99] |
| | 8 | .53 [.42, .62] | .59 [.49, .68] | .68 [.58, .76] | .75 [.65, .82] | .61 [.51, .71] | .76 [.64, .86] | .84 [.75, .91] | .90 [.8, .95] |
| | 4 | .41 [.31, .51] | .38 [.27, .46] | .43 [.35, .51] | .44 [.37, .53] | .48 [.38, .55] | .48 [.40, .55] | .52 [.43, .60] | .59 [.51, .65] |

**Medium Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .42 [.32, .52] | .71 [.60, .80] | .86 [.75, .92] | .94 [.90, .98] | .51 [.40, .62] | .76 [.68, .84] | .90 [.84, .96] | .97 [.93, .99] |
| | 8 | .50 [.41, .60] | .65 [.55, .75] | .76 [.66, .83] | .83 [.76, .90] | .55 [.45, .65] | .69 [.59, .79] | .79 [.69, .87] | .88 [.81, .94] |
| | 4 | .47 [.36, .56] | .52 [.37, .64] | .58 [.47, .70] | .63 [.46, .75] | .56 [.46, .66] | .61 [.50, .73] | .66 [.54, .76] | .73 [.65, .83] |
| Concave | 15 | .48 [.36, .56] | .73 [.61, .81] | .83 [.73, .91] | .91 [.82, .96] | .48 [.36, .58] | .73 [.61, .81] | .87 [.79, .93] | .95 [.88, .98] |
| | 8 | .49 [.37, .59] | .65 [.53, .74] | .73 [.61, .82] | .80 [.69, .88] | .50 [.40, .59] | .65 [.53, .73] | .75 [.67, .82] | .83 [.75, .90] |
| | 4 | .45 [.35, .56] | .51 [.41, .62] | .56 [.42, .66] | .60 [.45,.71] | .51 [.40, .59] | .55 [.46, .56] | .61 [.51, .69] | .67 [.57, .75] |
| Sigmoid | 15 | .41 [.32, .51] | .52 [.41, .60] | .61 [.52, .69] | .69 [.58, .79] | .46 [.37, .55] | .56 [.47, .67] | .67 [.55, .76] | .79 [.70, .86] |
| | 8 | .44 [.33, .56] | .49 [.41, .59] | .54 [.41, .64] | .60 [.49, .71] | .49 [.34, .58] | .55 [.43, .66] | .62 [.51, .73] | .71 [.62, .78] |
| | 4 | .46 [.34, .56] | .44 [.33, .55] | .45 [.32, .56] | .46 [.32, .57] | .53 [.40, .64] | .53 [.43, .61] | .54 [.44, .63] | .59 [.49, .69] |

**High Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .43 [.30, .53] | .63 [.45, .75] | .74 [.63, .83] | .82 [.69, .91] | .48 [.38, .60] | .63 [.50, 73] | .73 [.60, .85] | .83 [.75, .91] |
| | 8 | .45 [.34, .55] | .56 [.45, .70] | .64 [.51, .73] | .70 [.57, .80] | .50 [.39, .62] | .57 [.45, .68] | .63 [.51, .72] | .71 [.60, .81] |
| | 4 | .48 [.37, .60] | .52 [.40, .62] | .56 [.41, .70] | .59 [.48, .71] | .53 [.44, .64] | .54 [.43, .65] | .58 [.42, .72] | .62 [.53, .73] |
| Concave | 15 | .49 [.37, .59] | .67 [.55, .78] | .76 [.66, .87] | .83 [.73, .89] | .49 [.38, .60] | .62 [.50, .73] | .72 [.58, .79] | .82 [.88, .70] |
| | 8 | .47 [.35, .58] | .59 [.49, .68] | .65 [.51, .77] | .72 [.62, .82] | .50 [.38, .60] | .55 [.44, .65] | .62 [.49, .72] | .68 [.58, .80] |
| | 4 | .47 [.32, .58] | .53 [.38, .65] | .55 [.43, .65] | .60 [.49, .70] | .49 [.38, .60] | .52 [.41, .64] | .56 [.43, .66] | .60 [.49, .69] |
| Sigmoid | 15 | .45 [.34, .60] | .57 [.45, .69] | .63 [.52, .72] | .70 [.59, .79] | .45 [.34, .55] | .50 [.40, .60] | .57 [.46, .66] | .65 [.51, .73] |
| | 8 | .45 [.33, .57] | .51 [.39, .60] | .54 [.42, .64] | .58 [.46, .69] | .46 [.31, .56] | .49 [.36, .56] | .53 [.43, .62] | .57 [.45, .66] |
| | 4 | .44 [.32, .55] | .44 [.35, .54] | .45 [.35,.56] | .48 [.37, .60] | .47 [.37, .57] | .48 [.38, .58] | .50 [.38, .62] | .52 [.42, .62] |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist *Mean* $P_{greater}$

**Tabelle 3**

*Mean P-Greater - 10 Messungen pro Bedingung*

**Low Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .58 [.50, .67] | .96 [.92, .98] | .99 [.98, 1] | 1 [.99, 1] | .60 [.48, .71] | .98 [.96, .99] | 1 [.99, 1] | 1 [1, 1] |
| | 8 | .61 [.53, .69] | .85 [.79, .90] | .94 [.90, .97] | .98 [.96, .99] | .74 [.63, .82] | .94 [.90, .98] | .98 [.97, .99] | .99 [.99, .1] |
| | 4 | .10 [.04, .16] | .36 [.29, .42] | .43 [.38, .48] | .47 [.40, .52] | ,13 [.05, .19] | .55 [.49, .61] | .65 [.61, .68] | .68 [.64, .72] |
| Concave | 15 | .56 [.45, .67] | .93 [.89, .97] | .99 [.97, .99] | .99 [.99, 1] | .54 [.44, .63] | .95 [.90, .98] | .99 [.98, 1] | .99 [.99, 1] |
| | 8 | .57 [.47, .66] | .81 [.74, .86] | .90 [.86, .94] | .96 [.93, .98] | .64 [.54, .73] | .86 [.79, .91] | .95 [.90, .98] | .98 [.97, .99] |
| | 4 | .13 [.05, .22] | .34 [.27, .43] | .45 [.37, .52] | .48 [.43, .55] | .23 [.13, .30] | .52 [.59, .44] | .68 [.62, .74] | .71 [.62, .79] |
| Sigmoid | 15 | .47 [.36, .55] | .70 [.58, .79] | .84 [.77, .91] | .93 [.88, .96] | .51 [.40, .58] | .85 [.75, .91] | .96 [.91, .99] | .99 [.97, 1] |
| | 8 | .56 [.45, .66] | .65 [.56,.73] | .75 [.66, .82] | .82 [.74, .88] | .69 [.59, .77] | .90 [.83, .95] | .94 [.89, .98] | .96 [.92, .99] |
| | 4 | .33 [.24, .43] | .35 [.28, .41] | .40 [.37, .47] | .41 [.36, .48] | .39 [.31, .48] | .43 [.37, .47] | .46 [.41, .50] | .53 [.47, .59] |

**Medium Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .46 [.32, .55] | .81 [.72, .89] | .93 [.88, .97] | .98 [.95, .99] | .51 [.38, .63] | .85 [.75, .92] | .97 [.94, .99] | .99 [.98, 1] |
| | 8 | .54 [.42, .64] | .72 [.61, .79] | .82 [.74, .90] | .90 [.82, .96] | .59 [.50, .69] | .77 [.67, .85] | .89 [.81, .93] | .95 [.92, .98] |
| | 4 | .43 [.32, .53] | .49 [.40, .59] | .57 [.46, .65] | .62 [.53, .73] | .53 [.42, .62] | .61 [.50, .70] | .70 [.61, .76] | .77 [.69, .83] |
| Concave | 15 | .48 [.35, .58] | .78 [.69, .87] | .90 [.83, .95] | .96 [.92, .98] | .49 [.36, .69] | .81 [.72, .89] | .94 [.90, .97] | .98 [.95, .99] |
| | 8 | .53 [.41, .63] | .69 [.59, .77] | .79 [.70, .87] | .86 [.79, .92] | .53 [.39, .63] | .70 [.60, .79] | .83 [.73, .89] | .91 [.86, .95] |
| | 4 | .40 [.31, .52] | .47 [.36, .56] | .54 [.43, .64] | .60 [.50, .69] | .51 [.36, .61] | .56 [.45, .64] | .64 [.57, .74] | .69 [.61, 78] |
| Sigmoid | 15 | .42 [.30, .53] | .55 [.43, .63] | .65 [.56, .72] | .75 [.65, .83] | .46 [.35, .55] | .63 [.54, .72] | .77 [.65, .84] | .88 [82, .94] |
| | 8 | .47 [.37, .55] | .53 [.42, .63] | .59 [.47, .68] | .65 [.51, .74] | .53 [.44, .61] | .64 [.51, .76] | .74 [.66, .82] | .82 [.73, .88] |
| | 4 | .47 [.35, .59] | .43 [.34, .52] | .45 [.34, .54] | .47 [.34, .56] | .54 [.44, .63] | .52 [.42, .62] | .56 [.45, .65] | .62 [.52, .69] |

**High Noise**

| | | Low Overlap | | | | High Overlap | | | |
|---|---|---|---|---|---|---|---|---|---|
| Interaction | Size | 0 | S | M | L | 0 | S | M | L |
| Linear | 15 | .41 [.31, .60] | .54 [.51, .78] | .78 [.67, .87] | .88 [.79, .94] | .48 [.39, .58] | .67 [.55, .76] | .81 [.71, .89] | .91 [.82, .97] |
| | 8 | .46 [.37, .56] | .61 [.47, .70] | .68 [.58, .79] | .75 [.64, .83] | .52 [.44, .62] | .62 [.49, .71] | .70 [.79, .58] | .79 [.69, .88] |
| | 4 | .48 [.36, .59] | .53 [.38, .64] | .56 [.43, .68] | .61 [.50, .74] | .53 [.39, .67] | .58 [.48, .67] | .62 [.50, .72] | .67 [.57, .76] |
| Concave | 15 | .48 [.37, .60] | .69 [.56, .77] | .79 [.70, .88] | .87 [.77, .94] | .48 [.36, .59] | .67 [.56, .77] | .79 [.69, .87] | .89 [.82, .94] |
| | 8 | .48 [.35, .60] | .61 [.52, .70] | .69 [.57, .78] | .75 [.66, .84] | .49 [.37, .60] | .59 [.47, .69] | .67 [.56, .74] | .76 [.64, .86] |
| | 4 | .46 [.34, .57] | .50 [.36, .60] | .55 [.44, .68] | .59 [.46, .68] | .51 [.41, .63] | .54 [.44, .65] | .58 [.45, .66] | .62 [.52, .74] |
| Sigmoid | 15 | .42 [.31, .52] | .51 [.40, .61] | .58 [.45, .68] | .67 [.54, .78] | .45 [.33, .57] | .52 [.44, .61] | .60 [.51, .71] | .70 [.59, .79] |
| | 8 | .43 [.33, .55] | .48 [.38, .57] | .53 [.41, .64] | .58 [.46, .67] | .46 [.36, .54] | .51 [.41, ,59] | .56 [.46, .66] | .62 [.53, .71] |
| | 4 | .45 [.31, .54] | .44 [.33, .58] | .46 [.35, .57] | .46 [.31, .57] | .49 [.37, .60] | .51 [.37, .62] | .52 [.41, .65] | .55 [.45, .64] |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist *Mean* $P_{greater}$

### *4.1.3 Einfluss der Bedingungen pro Kondition*

Die Anzahl der Bedingungen pro Kondition zeigt unterschiedliche Auswirkungen auf *Mean* $P_{greater}$ an, je nachdem ob ein Effekt in den Daten vorliegt oder nicht. Ist die Interaktionskonstante größer als Null, dann führt eine Zunahme der Bedingungen pro Kondition zu einer starken Erhöhung der Werte von *Mean* $P_{greater}$. Befindet sich kein Effekt in den Daten, gibt es keine derartigen Veränderungen. Auffällig sind lediglich erneut die genannten Simulationen mit systematisch abweichendem *Mean* $P_{greater}$. Bei diesen führt eine Zunahme der Bedingungen pro Kondition scheinbar zunächst zu einer Erhöhung (von vier auf acht Bedingungen), anschließend wieder zu einer leichten Abnahme (von acht auf 15 Bedingungen) von *Mean* $P_{greater}$. Diese Beobachtung tritt jedoch nicht direkt aufgrund der Variation der Bedingungsanzahl auf. Dies liegt lediglich daran, dass der Simulationsalgorithmus die Bedingungspunkte so im *State-Trace-Raum* verteilt, dass es mehr oder weniger zu oben genannten Lage-Rauschen-Interaktionen kommt.

### *4.1.4 Einfluss von Überlappung*

Eine Variation der Überlappung zeigt, verglichen mit den anderen Parametern, wenig Einfluss auf *Mean* $P_{greater}$. Zwei zentrale Befunde sind jedoch zu nennen. Erstens scheinen die Werte bei weniger Überlappung insgesamt kleiner auszufallen. Zweitens kann geringe Überlappung in Kombination mit wenig Bedingungen pro Kondition und mittleren bis hohem Rauschen experimentelle Bedingungen erzeugen, in denen *Mean* $P_{greater}$, selbst wenn ein Effekt in den Daten vorhanden ist, nicht größer ausfällt, als wenn kein Effekt in den Daten vorliegt. Diese Fälle sind rar, verlangen jedoch eine Erklärung.

### *4.1.5 Einfluss von Messwerten pro Bedingung*

Die Variation der Messwerte pro Bedingung zeigt vorhersagbarerweise umgekehrte Auswirkungen zur Variation des Rauschens. Wird die Anzahl der Messwiederholungen pro Bedingung erhöht, distanzieren sich Werte von *Mean* $P_{greater}^{Effekt}$ und *Mean* $P_{greater}^{Null}$ voneinander.

### *4.1.6 Einfluss der Form der Funktionskurve*

Die Form der Funktionskurve zeigt keinen auffälligen Einfluss auf die Werte von *Mean* $P_{greater}$, wenn sich kein Effekt in den Daten befindet. Befindet sich jedoch ein Effekt in den Daten unterscheiden sich die Werte der unterschiedlichen Funktionskurven, wenn die übrigen Simulationsparameter gleich gehalten werden, sehr stark. So steigen die Werte von *Mean* $P_{greater}$ unter zunehmender Interaktionskonstante bei einer sigmoidalen Funktionskurve deutlich weniger an als bei einer linearen oder konkaven Funktionskurve. Somit braucht es

bei einer sigmoidalen Funktionskurve größere Effekte, um eindeutig zwischen Effekt und Nulleffekt zu differenzieren.

#### *4.1.7 Zusammenfassung der Ergebnisse für Mean $P_{greater}$*

Der PIRST-Algorithmus zeigt sich für die Simulationsdaten funktional. Liegen den Daten mehrere Funktionskurven zugrunde, dann ist *Mean* $P_{greater}$, wenn alle anderen Simulationsparameter gleichgehalten werden, im Durchschnitt größer, als wenn eine einzelne Funktionskurve den Daten zugrunde liegt. Je weiter die Kurven räumlich getrennt liegen, je geringer das Rauschen in den Daten ist, je mehr Bedingungen pro Kondition vorhanden sind und je mehr Messwiederholungen pro Bedingung vorgenommen wurden, desto größer wird dieser Unterschied. Wie auch in der Studie von Benjamin et. al (2019) ist es nicht möglich ein allgemeines Kriterium festzulegen, ab dem man von Evidenz für mehrere latente Prozesse sprechen kann, da *Mean* $P_{greater}$ auch bei einer Funktionskurve systematisch über oder unter .50 liegen kann. In Tabelle 3 im Abschnitt “Low Noise“ sind dabei mehrere Beispiele zu finden, an denen die Problematik deutlich wird.

### 4.2 AUC-Werte

Im Folgenden werden die Ergebnisse der AUC-Werte als zentrales Performanz-Kriterium dieser Arbeit betrachtet. Die vollständigen Ergebnisse der AUC-Werte sind in Tabelle 4-6 aufgeführt.

#### *4.2.1 Einfluss der Interaktionskonstante*

Die Variation der Interaktionskonstante beeinflusst die AUC-Werte folgenderweise: Je höher die Interaktionskonstante, desto mehr nähert sich der AUC-Wert der jeweiligen Simulation dem optimalen Wert 1 (Alle *p*-Werte der Simulationen mit Effekt sind kleiner als die *p*-Werte derselben Simulationen ohne Effekt) an. Diese Annäherung erfolgt schneller bei wenig Rauschen, mehr Bedingungen pro Kondition sowie mehr Messungen pro Bedingung. Dies zeigt, dass der Signifikanztest zunächst funktioniert, wie intendiert.

#### *4.2.2 Einfluss von Rauschen*

Der Einfluss des Rauschens auf die AUC-Werte lässt sich in einem Satz zusammenfassen: Je stärker das Rauschen, desto kleiner werden die AUC-Werte bei gleicher Interaktionskonstante und desto schwerer wird es zwischen Effekt und Nulleffekt zu differenzieren. Eine hohe Anzahl an Bedingungen pro Kondition sowie Messwerten pro Bedingung wirken dem jedoch bedeutsam entgegen. So zeigen dies Simulationen, dass selbst wenn Messfehler im Vergleich zum vorliegenden Effekt sehr groß ausfallen, ein AUC-Wert von .80 und höher mit realistischem Messaufwand fast immer erreicht werden kann.

**Tabelle 4**

*AUC - 3 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .98 | 1 | 1 | .97 | 1 | 1 | .91 | .97 | .99 | .91 | .97 | 1 |
| | 8 | .92 | .99 | 1 | .98 | .99 | 1 | .88 | .96 | .99 | .86 | .98 | .99 | .80 | .90 | .98 | .65 | .84 | .97 |
| | 4 | .85 | .96 | .98 | .94 | 1 | 1 | .61 | .80 | .89 | .68 | .79 | .94 | .64 | .77 | .81 | .65 | .71 | .79 |
| Concave | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .96 | .99 | 1 | .97 | 1 | 1 | .86 | .96 | .98 | .83 | .98 | 1 |
| | 8 | .96 | .99 | .99 | .97 | 1 | 1 | .85 | .97 | .99 | .84 | .98 | .99 | .81 | .89 | .94 | .69 | .79 | .95 |
| | 4 | .81 | .96 | .99 | .87 | .99 | 1 | .71 | .80 | .88 | .70 | .80 | .92 | .71 | .77 | .82 | .60 | .71 | .78 |
| Sigmoid | 15 | .9 | .98 | 1 | .98 | 1 | 1 | .79 | .93 | .98 | .75 | .92 | .99 | .79 | .9 | .96 | .72 | .88 | .95 |
| | 8 | .76 | .92 | .98 | .90 | .98 | .99 | .67 | .81 | .92 | .68 | .85 | .94 | .64 | .76 | .85 | .60 | .70 | .82 |
| | 4 | .53 | .78 | .91 | .63 | .84 | .94 | .58 | .60 | .68 | .51 | .64 | .81 | .59 | .59 | .66 | .54 | .66 | .66 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist der AUC-Wert.

**Tabelle 5**

*AUC - 5 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | .99 | 1 | 1 | .95 | 1 | 1 | .93 | .99 | 1 |
| | 8 | .97 | .98 | .98 | .95 | .96 | .96 | .90 | .98 | 1 | .92 | .99 | 1 | .85 | .96 | .99 | .76 | .89 | .98 |
| | 4 | .97 | .99 | 1 | 1 | 1 | 1 | .70 | .86 | .94 | .67 | .84 | .96 | .65 | .75 | .83 | .55 | .70 | .81 |
| Concave | 15 | .99 | .99 | .99 | 1 | 1 | 1 | .98 | 1 | 1 | .99 | 1 | 1 | .93 | .99 | .99 | .89 | .98 | 1 |
| | 8 | .96 | .99 | .99 | .98 | 1 | 1 | .90 | .95 | .98 | .95 | 1 | 1 | .85 | .93 | .98 | .70 | .87 | .95 |
| | 4 | .94 | 1 | 1 | .99 | 1 | 1 | .72 | .86 | .92 | .68 | .86 | .97 | .67 | .73 | .83 | .61 | .72 | .83 |
| Sigmoid | 15 | .98 | 1 | 1 | 1 | 1 | 1 | .80 | .94 | .99 | .83 | .98 | 1 | .82 | .92 | .97 | .67 | .82 | .95 |
| | 8 | .76 | .95 | .98 | .95 | .99 | .99 | .63 | .76 | .89 | .74 | .85 | .98 | .65 | .71 | .82 | .60 | .74 | .82 |
| | 4 | .68 | .95 | .98 | .89 | .95 | .99 | .50 | .58 | .60 | .51 | .55 | .78 | .53 | .60 | .67 | .55 | .65 | .72 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist der AUC-Wert.

**Tabelle 6**

*AUC - 10 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | .98 | .98 | .98 | 1 | 1 | 1 | 1 | 1 | 1 | .99 | 1 | 1 | .98 | 1 | 1 | .97 | .99 | .99 |
| | 8 | .98 | .98 | .98 | .93 | .93 | .93 | .93 | .99 | .99 | .97 | .99 | .99 | .90 | .97 | .99 | .86 | .96 | 1 |
| | 4 | 1 | 1 | 1 | 1 | 1 | 1 | .80 | .95 | .98 | .85 | .99 | 1 | .70 | .75 | .89 | .67 | .80 | .91 |
| Concave | 15 | .99 | .99 | .99 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | .95 | .99 | .99 | .97 | 1 | 1 |
| | 8 | .99 | 1 | 1 | .98 | .98 | .98 | .91 | .99 | 1 | .96 | 1 | 1 | .89 | .97 | .99 | .86 | .96 | .99 |
| | 4 | .99 | 1 | 1 | 1 | 1 | 1 | .80 | .93 | .98 | .68 | .94 | .98 | .63 | .77 | .89 | .60 | .76 | .86 |
| Sigmoid | 15 | .98 | .99 | 1 | .99 | 1 | 1 | .85 | .97 | .99 | .97 | 1 | 1 | .78 | .89 | .97 | .79 | .93 | .98 |
| | 8 | .92 | .99 | .99 | .97 | .98 | .98 | .73 | .84 | .94 | .88 | .99 | .99 | .63 | .77 | .85 | .74 | .85 | .96 |
| | 4 | .96 | .99 | .99 | 1 | 1 | 1 | .50 | .66 | .76 | .56 | .72 | .90 | .50 | .55 | .56 | .60 | .64 | .74 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist der AUC-Wert.

### *4.2.3 Einfluss der Bedingungen pro Kondition*

Der Einfluss der Bedingungen pro Kondition auf die AUC-Werte zeigt sich im Vergleich zu den anderen Parametern besonders stark. Je mehr Bedingungen pro Kondition verwendet werden und der *State-Trace-Raum* auskartiert wird, desto mehr nähern sich die AUC-Werte dem Wert 1 an. Während die AUC-Werte bei vier Bedingungen pro Kondition

sehr stark abhängig von den übrigen Simulationsparametern sind, befinden sich bei 15 Bedingungen pro Kondition der größte Teil der AUC-Werte im Bereich zwischen .90 und 1. Hierbei existieren einzelne Ausreißer, jedoch selbst unter ungünstigsten Bedingungen (hohes Rauschen, kleiner Effekt, wenig Überlappung, wenig Messungen pro Bedingung) fallen deren Werte nicht unter .70.

#### *4.2.4 Einfluss von Überlappung*

Die Ergebnisse zum Einfluss der Überlappung auf die AUC-Werte lassen keine allgemeinen Aussagen zu. Die Daten zeigen, dass die AUC-Werte bei hoher Überlappung in Kombination mit kleinem bis mittlerem Rauschen insgesamt höher ausfallen. Nimmt das Rauschen jedoch zu, verschwindet dieser positive Einfluss der Überlappung. In einigen Simulationen mit starkem Rauschen kehrt sich dieser Effekt sogar um, sodass geringe Überlappung eher wünschenswerte Einflüsse auf die Performanz anzeigt. Im Vergleich zu den anderen Simulationsparametern zeichnet sich dieser Einfluss jedoch eher unauffällig ab.

#### *4.2.5 Einfluss der Messwerte pro Bedingung*

Der Einfluss der Messwerte pro Bedingung auf die AUC-Werte zeigt sich wie auch bei *Mean* $P_{greater}$ invers zum Einfluss des Rauschens. Wird die Anzahl der Messwerte pro Bedingung erhöht, dann nähern sich die AUC-Werte zunehmend dem optimalen Wert 1 an, da somit indirekt das Rauschen in den Daten reduziert wird.

#### *4.2.6. Einfluss der Form der Funktionskurve*

Die Auswirkungen der Form der Funktionskurve auf die AUC-Werte zeigen sich ähnlich zu den Ergebnissen für *Mean* $P_{greater}$. So fallen die AUC-Werte, unter Gleichhaltung aller übrigen Parameter, bei einer sigmoidalen Funktionskurve deutlich geringer aus als bei einer linearen oder konkaven Funktionskurve. Es gibt demnach große Unterschiede in der Differenzierungsfähigkeit des PIRST-Signifikanztests für unterschiedliche Formen von Funktionskurven.

#### *4.2.7 Zusammenfassung der Ergebnisse der AUC-Werte*

Das erweiterte PIRST zeigte für die Simulationen eine große Reichweite von AUC-Werten (.50 – 1). Der Großteil der Werte liegt jedoch im Bereich zwischen ca. .80 und 1. Als größter Einfluss auf die Differenzierungsfähigkeit des Algorithmus zeigt sich die Anzahl der Bedingungen pro Kondition. Während bei vier Bedingungen pro Kondition mittleres bis starkes Rauschen die Differenzierungsfähigkeit massiv beeinträchtigen kann, zeigt sich der Algorithmus bei acht und 15 Punkten sehr robust gegen Messfehler. Befindet sich eine Vielzahl von Bedingungen auf Boden- oder Deckenplateaus (siehe Sigmoidale Funktion), wirkt sich dies negativ auf die AUC-Werte aus.

### 4.3 Sensitivität

Als nächstes werden die Sensitivitätswerte (α = .05) betrachtet, um Aussagen über die Power (Wahrscheinlichkeit einen Effekt zu finden, wenn er vorhanden ist) abzuleiten. Die einzelnen Sensitivitätswerte der Simulationen sind in Tabelle 7-9 aufgeführt.

**Tabelle 7**

*Sensitivität - 3 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .70 | .96 | 1 | .61 | .98 | 1 | .26 | .70 | .88 | .12 | .47 | .90 |
| | 8 | .83 | 1 | 1 | .90 | 1 | 1 | .23 | .62 | .95 | .22 | .76 | .99 | .07 | .14 | .55 | .04 | .14 | .38 |
| | 4 | .24 | .75 | .91 | .70 | .98 | 1 | .18 | .44 | .67 | .28 | .46 | .89 | .01 | .14 | .17 | .07 | .09 | .22 |
| Concave | 15 | .97 | 1 | 1 | .99 | 1 | 1 | .63 | .92 | .99 | .55 | .97 | 1 | .44 | .83 | .94 | .10 | .47 | .79 |
| | 8 | .80 | .99 | 1 | .78 | 1 | 1 | .25 | .77 | .90 | .19 | .63 | .92 | .14 | .32 | .53 | .04 | .11 | .46 |
| | 4 | .20 | .52 | .81 | .31 | .86 | .99 | .15 | .29 | .51 | .14 | .26 | .57 | .10 | .15 | .15 | .02 | .10 | .15 |
| Sigmoid | 15 | .19 | .68 | .99 | .77 | .97 | 1 | .03 | .21 | .52 | .05 | .35 | .75 | .07 | .20 | .37 | .05 | .11 | .18 |
| | 8 | .15 | .47 | .82 | .52 | .92 | .99 | .01 | .09 | .28 | .03 | .20 | .44 | .03 | .03 | .10 | .01 | .03 | .04 |
| | 4 | .16 | .58 | .80 | .25 | .62 | .88 | .07 | .10 | .18 | .09 | .17 | .46 | 0 | .02 | .06 | .01 | .07 | .04 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Sensitivität (α = .05).

**Tabelle 8**

*Sensitivität - 5 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .88 | .99 | 1 | .96 | 1 | 1 | .37 | .83 | .94 | .26 | .70 | .99 |
| | 8 | 1 | 1 | 1 | 1 | 1 | 1 | .52 | .91 | 1 | .65 | .96 | 1 | .19 | .39 | .64 | .08 | .34 | .74 |
| | 4 | .06 | .32 | .65 | .46 | .98 | 1 | .21 | .45 | .75 | .34 | .68 | .94 | .02 | .09 | .28 | .12 | .20 | .34 |
| Concave | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .78 | 1 | .99 | .83 | 1 | 1 | .52 | .86 | .96 | .33 | .78 | .97 |
| | 8 | .91 | 1 | 1 | .97 | 1 | 1 | .42 | .75 | .95 | .43 | .91 | 1 | .14 | .42 | .67 | .09 | .28 | .56 |
| | 4 | .01 | .43 | .73 | .41 | .94 | 1 | .14 | .25 | .54 | .18 | .40 | .83 | .09 | .16 | .23 | .05 | .15 | .28 |
| Sigmoid | 15 | .53 | .98 | 1 | .96 | 1 | 1 | .06 | .39 | .78 | .23 | .77 | .99 | .05 | .08 | .38 | .04 | .11 | .45 |
| | 8 | .38 | .82 | .96 | .93 | 1 | 1 | .04 | .08 | .31 | .13 | .36 | .85 | .01 | .04 | .10 | .01 | .08 | .17 |
| | 4 | .06 | .57 | .85 | .12 | .35 | .77 | .03 | .12 | .19 | .14 | .14 | .54 | 0 | .02 | .03 | .02 | .08 | .07 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Sensitivität (α = .05).

**Tabelle 9**

*Sensitivität - 10 Messungen pro Bedingung*

| | | Low Noise | | | | | | Medium Noise | | | | | | High Noise | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | | Low Overlap | | | High Overlap | | |
| Interaction | Size | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L | S | M | L |
| Linear | 15 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | .54 | .97 | 1 | .53 | .99 | 1 |
| | 8 | 1 | 1 | 1 | 1 | 1 | 1 | .85 | 1 | 1 | .96 | 1 | 1 | .33 | .66 | .87 | .34 | .70 | .97 |
| | 4 | .06 | .19 | .18 | .46 | 1 | 1 | .16 | .56 | .82 | .41 | .91 | 1 | .12 | .22 | .46 | .19 | .38 | .71 |
| Concave | 15 | 1 | 1 | 1 | 1 | 1 | 1 | .96 | 1 | 1 | 1 | 1 | 1 | .63 | .93 | .99 | .63 | .99 | 1 |
| | 8 | 1 | 1 | 1 | 1 | 1 | 1 | .64 | 1 | 1 | .82 | .99 | 1 | .27 | .69 | .88 | .24 | .61 | .92 |
| | 4 | .01 | .25 | .58 | .18 | 1 | 1 | .05 | .30 | .68 | .22 | .73 | .94 | .07 | .17 | .29 | .11 | .25 | .46 |
| Sigmoid | 15 | .91 | 1 | 1 | 1 | 1 | 1 | .22 | .74 | .95 | .67 | .99 | 1 | .03 | .18 | .39 | .12 | .42 | .82 |
| | 8 | .71 | .99 | 1 | 1 | 1 | 1 | .11 | .30 | .71 | .57 | .97 | 1 | 0 | .06 | .13 | 0 | .16 | .45 |
| | 4 | .38 | 1 | 1 | .52 | .72 | .79 | .08 | .27 | .44 | .11 | .25 | .64 | .04 | .11 | .06 | .13 | .11 | .21 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Sensitivität (α = .05).

### *4.3.1 Einfluss der Interaktionskonstante*

Der Einfluss der Interaktionskonstante auf die Sensitivität des Verfahrens lässt sich relativ einfach beschreiben. Je größer die Interaktionskonstante, also der gesuchte Effekt, desto mehr nähert sich die Sensitivität dem optimalen Wert 1 an (alle Effekte werden erkannt). Diese Annäherung erfolgt schneller, je weniger Rauschen in den Daten vorliegt, mehr Bedingungen pro Kondition vorhanden sind und mehr Messungen pro Bedingung vorgenommen werden.

### *4.3.2 Einfluss von Rauschen*

Der Einfluss des Rauschens auf die Sensitivität zeichnet sich in den Daten sehr eindeutig ab. Je höher das Rauschen, desto geringer wird die Sensitivität. Dieser Einfluss zeigt sich im Hinblick auf die praktische Anwendung sehr gravierend. So gibt es Simulationen, in denen eine Erhöhung des Rauschens von gering auf hoch die Wahrscheinlichkeit einen Effekt (bei $\alpha = .05$) zu finden, von 100% auf 0% verringern kann. Dieser Problematik kann durch eine erhöhte Anzahl von Bedingungen pro Kondition sowie Messungen pro Bedingung stark entgegengewirkt werden.

### *4.3.3 Einfluss der Bedingungen pro Kondition*

Die Anzahl der Bedingungen pro Kondition hat, wie bereits angerissen, einen großen Einfluss auf die Sensitivitätswerte. Generell gilt: Je mehr Bedingungen pro Kondition vorhanden sind, desto höher wird die Wahrscheinlichkeit einen vorhandenen Effekt zu finden. Während bei Simulationen mit vier Bedingungen pro Kondition die Sensitivitätswerte unter zunehmendem Rauschen stark abfallen, zeigen sich 15 Bedingungen pro Bedingung, auch wenn nicht unbeeinflusst, deutlich resilienter gegenüber Messfehlern.

### *4.3.4 Einfluss von Überlappung*

Die Ergebnisse zum Einfluss der Überlappung auf die Sensitivität lassen wie bei den AUC-Werten keine allgemeinen Aussagen zu. Auch hier fallen die Werte bei hoher Überlappung in Kombination mit kleinem bis mittlerem Rauschen insgesamt höher aus. Nimmt das Rauschen jedoch zu, verschwindet dieser positive Einfluss. Die Daten weisen darauf hin, dass eine geringe Überlappung bei starkem Rauschen sogar wünschenswerte Einflüsse haben kann und Sensitivitätsabfälle abschwächt.

### *4.3.5 Einfluss der Messwerte pro Bedingung*

Die Anzahl der Messwerte pro Bedingung steht im positiven Zusammenhang mit der Sensitivität. Je höher die Anzahl, desto mehr nähern sich die Sensitivitätswerte dem optimalen Wert 1 an. Besonders bei mittlerem bis starkem Rauschen zeigt sich eine erhöhte Anzahl von Messwerten pro Bedingung als starker Schutzfaktor gegen Sensitivitätsabfälle.

### *4.3.6 Einfluss der Form der Funktionskurve*

Die Auswirkungen der Form der Funktionskurve auf die Sensitivität zeigen sich erneut ähnlich zu vorherigen Ergebnissen. So fallen die Sensitivitätswerte, unter Gleichhaltung aller übrigen Parameter, bei einer sigmoidalen Funktionskurve deutlich kleiner aus als bei einer linearen oder konkaven Funktionskurve. Besonders in Kombination mit starkem Rauschen kommt es zu gravierenden Sensitivitätsabfällen, sodass es hier annähernd unmöglich wird kleine Effekte zu finden.

### *4.3.7 Zusammenfassung der Ergebnisse der Sensitivität*

Die Sensitivitätswerte für $\alpha = .05$ zeigen über die Simulationen eine große Reichweite an Werten (0 – 1). Dabei hängt es hier von vielen Faktoren ab, ob eine hohe Sensitivität (z. B. 80%) erreicht wird. Prinzipiell gilt: Je kleiner der Effekt, desto mehr Bedingungen pro Kondition und mehr Messungen pro Bedingung werden benötigt, um diesen mit hoher Sicherheit zu finden. Funktionskurven in sigmoidaler Form zeigen sich in Kombination mit starkem Rauschen als größte Problematik in Bezug auf die Sensitivität.

## 4.4 Spezifität

Im folgenden Abschnitt werden die Spezifitätswerte ($\alpha = .05$) betrachtet, um Aussagen darüber abzuleiten, mit welcher Wahrscheinlichkeit ein Nulleffekt auch als dieser erkannt wird. Die einzelnen Spezifitätswerte der Simulationen sind in Tabelle 10-12 aufgeführt.

### *4.4.1 Einfluss von Rauschen*

Der Einfluss des Rauschens auf die Spezifitätswerte zeigt sich insgesamt so, dass stärkeres Rauschen höhere Spezifitätswerte erzeugt. Dieses Phänomen scheint jedoch nicht allein vom Rauschen abzuhängen. Je stärker Lage-Rauschen-Interaktionen in den Daten vorliegen (und der Nulleffekt von *Mean* $P_{greater}$ in einer Simulation systematisch über .50 liegt), desto tiefer liegen auch die Spezifitätswerte. Dies erzeugt auffällige Ausreißer. Da diese Ausnahmefälle, aufgrund der Implementation des Simulationsalgorithmus, vermehrt in Simulationen mit geringem Rauschen auftreten, ist der alleinige Einfluss des Rauschens dadurch nicht eindeutig absehbar.

### *4.4.2 Einfluss der Bedingungen pro Kondition*

Die Anzahl der Bedingungen pro Kondition zeigt über die Simulationen keine eindeutigen Einflussmuster auf die Spezifität. Acht Bedingungen pro Kondition scheinen zwar insgesamt die geringsten Spezifitätswerte zu erzeugen, Ursache dafür ist jedoch nicht der Simulationsparameter selbst. Da der Simulationsalgorithmus die Bedingungspunkte gleichmäßig im normierten *State-Trace-Raum* verteilt, wird durch diese Ausprägung des

**Tabelle 10**

*Spezifität - 3 Messungen pro Bedingung*

| | | Low Noise | | Medium Noise | | High Noise | |
|---|---|---|---|---|---|---|---|
| | | Low Overlap | High Overlap | Low Overlap | High Overlap | Low Overlap | High Overlap |
| Interaction | Size | 0 | 0 | 0 | 0 | 0 | 0 |
| Linear | 15 | .99 | .99 | .99 | 1 | 1 | 1 |
| | 8 | .86 | .89 | .97 | .99 | 1 | 1 |
| | 4 | .97 | .96 | .91 | .89 | .97 | .99 |
| Concave | 15 | 1 | .99 | .97 | .99 | .95 | 1 |
| | 8 | .94 | .97 | .99 | 1 | .98 | .99 |
| | 4 | .98 | .98 | .92 | .95 | .99 | .99 |
| Sigmoid | 15 | 1 | .99 | .99 | .99 | .98 | 1 |
| | 8 | .98 | .94 | .98 | .99 | 1 | 1 |
| | 4 | .84 | .88 | .95 | .96 | .98 | 1 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Spezifität (α = .05).

**Tabelle 11**

*Spezifität - 5 Messungen pro Bedingung*

| | | Low Noise | | Medium Noise | | High Noise | |
|---|---|---|---|---|---|---|---|
| | | Low Overlap | High Overlap | Low Overlap | High Overlap | Low Overlap | High Overlap |
| Interaction | Size | 0 | 0 | 0 | 0 | 0 | 0 |
| Linear | 15 | .94 | .94 | .99 | .98 | 1 | 1 |
| | 8 | .77 | .63 | .95 | .91 | 1 | .99 |
| | 4 | 1 | 1 | .94 | .84 | .98 | .94 |
| Concave | 15 | .93 | .98 | .99 | 1 | .97 | .98 |
| | 8 | .90 | .90 | .96 | 1 | .98 | .97 |
| | 4 | 1 | 1 | .98 | .98 | .99 | .94 |
| Sigmoid | 15 | 1 | .95 | 1 | .99 | 1 | 1 |
| | 8 | .91 | .80 | .97 | .99 | 1 | .98 |
| | 4 | .95 | 1 | .94 | .86 | .99 | .99 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Spezifität (α = .05).

**Tabelle 12**

*Spezifität - 10 Messungen pro Bedingung*

| | | Low Noise | | Medium Noise | | High Noise | |
|---|---|---|---|---|---|---|---|
| | | Low Overlap | High Overlap | Low Overlap | High Overlap | Low Overlap | High Overlap |
| Interaction | Size | 0 | 0 | 0 | 0 | 0 | 0 |
| Linear | 15 | .78 | .82 | .98 | .97 | 1 | .99 |
| | 8 | .90 | .49 | .89 | .88 | .98 | .98 |
| | 4 | 1 | 1 | .98 | .96 | .96 | .91 |
| Concave | 15 | .88 | .91 | .99 | .98 | .97 | .98 |
| | 8 | .91 | .87 | .92 | .98 | .97 | .98 |
| | 4 | 1 | 1 | .98 | .94 | .98 | .94 |
| Sigmoid | 15 | .97 | .94 | .96 | .98 | .99 | .96 |
| | 8 | .90 | .73 | .99 | .95 | .99 | .98 |
| | 4 | .99 | 1 | .87 | .89 | .99 | .96 |

Anmerkung. Umso heller die Farbe ausfällt, desto größer ist die Spezifität (α = .05).

Parameters die Lage der Punkte so angepasst, dass oben genannte Interaktionen begünstigt werden.

### *4.4.3 Einfluss der Überlappung*

Die Überlappung zeigt sich über die Simulationen ebenfalls ohne übergreifende Einflüsse. Interessant ist jedoch, dass eine geringe Überlappung die starken Sensitivitätsabfälle oben genannter Ausnahmefälle bedeutsam abschwächt. Somit ergibt sich ein zweiter Fall in dem eine geringere Überlappung wünschenswerte Effekte aufzeigt.

### *4.4.4 Einfluss der Messwerte pro Bedingung*

Die Anzahl der Messwerte pro Bedingung zeigt bezüglich der Spezifität, wie auch in vorherigen Ergebnissen, inverse Einflüsse zum Rauschen auf. So sinken die Spezifitätswerte unter Erhöhung der Messwerte pro Bedingung insgesamt ab. Dieser Effekt zeigt sich jedoch, bis auf bei genannten Ausnahmefällen, relativ gering. Für den Großteil der Werte führt eine Erhöhung von drei auf zehn Messwerte nur zu Spezifitätsverlusten, die den einstelligen Prozentbereich nicht übersteigen.

### *4.4.5 Einfluss der Form der Funktionskurve*

Die Form der Funktionskurve zeigte keine übergreifenden Einflüsse auf die Spezifität. Lineare, konkave sowie sigmoidale Funktionskurven weisen ähnliche Spezifitätswerte auf. Lediglich oben genannte Ausnahmefälle mit systematisch erhöhtem *Mean* $P_{greater}$ zeigen bei einer linearen Funktionskurve deutlich höhere Spezifitätsverluste an. Diese Beobachtung wird jedoch nicht direkt durch eine lineare Kurvenform verursacht, sondern durch ihre Implementierung im Algorithmus. So wird der Abstand der Bedingungspunkte so gewählt, dass er die mehrfach genannte Lage-Rauschen-Interaktion begünstigt.

### *4.4.6 Zusammenfassung der Ergebnisse der Spezifität*

Insgesamt lässt sich sagen, dass die Spezifitätswerte für $\alpha = .05$ sehr hoch ausfallen. Fast alle Werte, bis auf einzelne Ausreißer, liegen im Bereich von .90-1. Diese Ausreißer entstehen nur dann, wenn Lage-Rauschen-Interaktionen außergewöhnlich stark auftreten. Das Rauschen steht im positiven Zusammenhang mit der Spezifität. Übrige Parameter scheinen kaum Einfluss auf die Spezifitätswerte zu haben.

## 5. Diskussion

Diese Arbeit verfolgte zwei übergeordnete Ziele. Das erste Ziel bestand darin, ein theoretisch fundiertes Testverfahren im Rahmen der *State-Trace-Analysis* (STA) zu entwickeln, welches valide Aussagen über die Anzahl leistungsbestimmender Prozesse in kognitiven Fähigkeiten generiert. Realisiert wurde dies durch eine Erweiterung des von Benjamin et al. (2019) vorgestellten Permutationsansatz PIRST. Dieser erwies sich in früheren Simulationen als hochgradig differenzierungsfähig zwischen ein und mehreren latenten Prozessen, zeigte jedoch Anwendungsprobleme durch eine ungeeignete Teststatistik.

Um die Problematik aufzuheben, wurde der Ansatz mit einem Signifikanztest verknüpft, der unter einer selbst gewählten Fehlerwahrscheinlichkeit (z. B. α = .05) eine objektive Entscheidung für oder gegen die Nullhypothese (nur ein latenter Prozess bestimmt die Performanz) generiert. Das zweite Ziel dieser Arbeit bestand in der Validierung des Testverfahrens. Umgesetzt wurde dies durch die Ergebnisanalyse von 648 simulierten Experimenten. In diesen wurden verschiedene experimentelle Parameter systematisch und in möglichst realistischen Wertebereichen variiert und Performanz-Kennwerte wie AUC-Werte, Sensitivität und Spezifität berechnet.

## 5.1 Die Performanz des erweiterten PIRST

Im Folgenden werden die einzelnen Performanz-Kriterien AUC-Werte, Sensitivität und Spezifität final zusammengefasst und interpretiert. Anschließend sollen aufkommende Problematiken diskutiert und aufbauend darauf ein Gesamturteil über die Performanz getroffen werden.

### *5.1.1 Zusammenfassung und Interpretation der Performanz-Kriterien*

Die AUC-Werte zeigen, wie im Ergebnisteil beschrieben, über alle Simulationen eine große Reichweite (.50 – 1), wobei der deutlich überwiegende Teil der Werte zwischen .80 und 1 liegt. Dies signalisiert insgesamt eine gute grundlegende Differenzierungsfähigkeit. Es gibt zwei Aspekte, welche primär zur Abnahme der AUC-Werte führen. Diese sind starkes Rauschen bzw. große Messfehler (SD > ca. 20% des *State-Trace-Raumes*), sowie eine überwiegende Anzahl von Messungen innerhalb weniger Plateaus (siehe sigmoidale Funktion). Während eine dieser Gegebenheiten geringere Einbußen in der Performanz verursacht, führt die Kombination oft zu gravierenden Verlusten. Das erklärt sich dadurch, dass Plateaus, aufgrund von Stauchung der Werte auf mindestens einer Achse, dazu führen, dass Monotonie-Verletzungen innerhalb einer Kondition einfacher auftreten können. Auf diese Weise werden die negativen Auswirkungen des Rauschens verstärkt. Eine großflächige und detaillierte Auskartierung des *State-Trace-Raums* erweist sich als starker Schutzfaktor gegen beide Gegebenheiten. So wurden bei 15 Punkten pro Kondition unter ungünstigsten übrigen Parametern keine AUC-Werte unter .70 verzeichnet. So ist eine hohe allgemeine Differenzierungsfähigkeit unter der Voraussetzung geringer Messfehler und wenig Plateau-messungen gegeben.

Die Sensitivitätswerte für $\alpha = .05$ zeigen sich über die Simulationen hinweg sehr unterschiedlich (0 - 1). Ob eine hohe Sensitivität (ca. 80%) erreicht wird, hängt ebenfalls von den zwei oben genannten Aspekten ab. Die Sensitivitätswerte zeigen sich dabei vergleichsweise anfällig. Starkes Rauschen und/oder viele Messungen in Plateaubereichen

erschweren das Finden mehrerer Funktionskurven in hohem Maße (größtenteils Sensitivitätswerte unter .50 bei kleinen und mittleren Interaktionskonstanten). Der Problematik kann mit einer erhöhten Anzahl von Bedingungen pro Kondition sowie einer erhöhten Zahl von Messungen pro Bedingung teilweise entgegengewirkt werden. Das erweiterte PIRST zeigt sich dementsprechend unter der Voraussetzung geringer Messfehler und wenig Plateaumessungen mit einem Signifikanzkriterium von $\alpha = .05$ als sensitiv.

Die Spezifitätswerte für $\alpha = .05$ fallen insgesamt sehr hoch aus. Fast alle Werte, bis auf einzelne Ausreißer, liegen im Bereich von .90 - 1. Besagte Ausreißer treten dann auf, wenn es zu Lage-Rauschen-Interaktionen kommt, und *Mean* $P_{greater}$ außergewöhnlich stark von .50 abweicht. Für Abweichungen bis ca. .65 zeigt sich das erweiterte PIRST jedoch resilient. Dies beschreibt sehr präzise in welchem Ausmaß der Signifikanztest mit der Problematik, welche zu dieser Arbeit geführt hat, umgehen kann – zu großen Teilen, jedoch nicht vollständig. Selbst unter der Konstruktion eines Nullmodells können die Auswirkungen von Lage-Rauschen-Interaktionen nicht vollständig umgangen werden. Spezifität ist demnach unter der Voraussetzung gegeben, dass Lage-Rauschen-Interaktionen nicht zu stark auftreten.

### *5.1.2 Umgang mit Rauschen, Plateaus und Lage-Rauschen-Interaktionen*

Aus den Ergebnissen leitet sich ab, dass *State-Trace-Daten* drei Problematiken beinhalten können, welche die Performanz von PIRST einschränken. Diese sind starke Messfehler, Messungen auf Plateaus und übermäßige Lage-Rauschen-Interaktionen. Im Folgenden soll darauf eingegangen werden, ob und wie mit diesen Problematiken im *State-Trace-Design* umgegangen werden kann.

Existieren keine messgenauen AVs, dann gibt es zwei Möglichkeiten, um mit starken Messfehlern umzugehen. Der Einfluss der Messfehler im *State-Trace-Raum* ist immer relativ zum Abstand der Bedingungspunkte. So kann über die *Trace-Variable* versucht werden Abstände zu optimieren. Ist dies nicht möglich, dann besteht die Antwort auf die Messfehlerproblematik schlichtweg in mehr Bedingungen pro Kondition und mehr Messungen pro Bedingung. Plateaus können ebenfalls auf diese Weise adressiert werden, dies gestaltet sich jedoch unökonomisch. Ideal wäre es, zu wissen, in welchen Performanz-Bereichen derartige Plateaus liegen und dann in anderen Performanz-Bereichen zu messen. Um an solche Informationen zu gelangen, kann in vergangenen *State-Trace-Experimenten* des eigenen Forschungsbereichs recherchiert werden. Prince et al. (2012) demonstrierten beispielsweise, wie Informationen vergangener *State-Trace-Experimente* das Design neuaufgesetzter Experimente deutlich verbessern können. Existieren keine vorangegangenen

Versuche, besteht dennoch die Möglichkeit mithilfe theoretischer Ableitungen oder Pilotstudien Plateaus zu lokalisieren.

Um mit der Problematik der Lage-Rauschen-Interaktionen umzugehen, ergeben sich ebenfalls mehrere Möglichkeiten. Eine dieser Möglichkeiten besteht darin das Problem schlichtweg zu ignorieren. Das erweiterte PIRST zeigt sich, solange diese Interaktionen nicht zu stark auftreten, sehr resilient gegenüber diesen. Damit Interaktionen von $Mean\ P_{greater} > .65$ entstehen, müssen sehr spezifische Bedingungen vorliegen, die in der Realität unwahrscheinlich sein sollten. Des Weiteren besteht die Möglichkeit das Problem direkt im *State-Trace-Design* zu adressieren. Wie von Prince et al. (2012) verdeutlicht, kann die Lage der Bedingungspunkte durch intensive theoretische Vorarbeit stark beeinflusst werden, um z. B. Überlappung zu erzeugen. So wäre ein Ansatz, den Abstand von Bedingungspunkten innerhalb einer Kondition so zu regulieren, dass eine Art A-A-B-B-A-A-B-B-Anordnung anstatt der üblicherweise angestrebten A-B-A-B-Anordnung (Kondition 1 – A, Kondition 2 - B) der Bedingungen entlang einer Achse entsteht. So wäre jede Bedingung einmal von derselben und einmal von einer anderen Kondition flankiert, was der Interaktion entgegenwirken sollte. Ziel ist es dabei nicht Interaktionen dieser Art vollständig zu verhindern, diese müssen lediglich in einem akzeptablen Rahmen gehalten werden.

So bleibt abschließend zu sagen, dass all diese Problematiken adressierbar und größtenteils auch nicht PIRST-spezifisch sind. Ein qualitativ hochwertiges *State-Trace-Experiment*, unabhängig davon wie es ausgewertet wird, setzt in jedem Fall theoretische Vorarbeit sowie eine Auseinandersetzung mit derartigen Problemen voraus.

### *5.1.3 Gesamturteil*

Das erweiterte PIRST ist ein statistisches Testverfahren, das auf der Grundlage von Experimentaldaten, welche verschiedene Voraussetzungen erfüllen, valide Aussagen über die Anzahl latenter Prozesse in kognitiven Performanzen generiert. Diese Voraussetzungen können durch eine detaillierte Planung des experimentellen Designs (hohe Überlappung, wenig Messungen in einzelnen Plateaubereichen, keine zu starken Lage-Rauschen-Interaktionen) und/oder erhöhten Messaufwand (Bei Erwartung vergleichsweise kleiner Effekte bzw. großer Messfehler) stets gewährleistet werden. Unter gegebenen Voraussetzungen ist $\alpha = .05$ ein differenzierungsstarkes Entscheidungskriterium. Somit zeigt das erweiterte PIRST vergleichbare Eigenschaften mit anderen erfolgreichen statistischen Testverfahren der Psychologie. Dementsprechend ist das Verfahren als performant sowie für die Forschungspraxis geeignet einzustufen.

### 5.2 Methodische Kritikpunkte

Die Validierung des erweiterten PIRST fundiert auf den Ergebnissen von Simulationsexperimenten. Da Simulationen die Realität trivialerweise nicht zu 100 Prozent wiedergeben können, unterliegen die Ergebnisse dieser Arbeit einigen Limitationen. Der erste und offensichtlichste Punkt betrifft die Form der Funktionskurven. In den Simulationen wurden lineare, konkave so wie sigmoidale Funktionskurven betrachtet. Diese decken zwar ein breites Spektrum an plausiblen Funktionsformen ab, theoretisch sind aber unendlich viele andere monotone Funktionen vorstellbar, welche man untersuchen könnte. Ein weiterer Punkt ist das Rauschen. Die Standardabweichung des Gauß-Rauschens ist innerhalb eines Simulationsexperiments für jeden Bedingungspunkt und auf beiden Achsen gleich. Dies muss, besonders im Hinblick auf Boden und Deckeneffekte, nicht zwangsweise der Realität entsprechen. Der letzte Punkt ist die fehlende Variation zwischen den Versuchspersonen. Die Form der Funktionskurve sowie die Verhältnisse verschiedener Bedingungen auf beiden Achsen sind innerhalb einer Simulation für alle Versuchspersonen identisch. Dies auf die Realität zu übertragen, wäre eine gewagte Annahme, wie die Problematik der Mittelwertbildung über Personen verdeutlicht (Prince et al. 2012). All dies sind relevante Details. Da aber das Simulationsdesign dieser Arbeit bereits einen Umfang von 648 verschiedenen Experimenten aufweist und jede weitere Variation die Anzahl verdoppelt, wurde wie auch bei Benjamin et al. (2019) zunächst grundlegenden Parametern Beachtung geschenkt.

Zusätzlich gibt es zwei Kritikpunkte auf anderen methodischen Ebenen. Ersterer bezieht sich auf die Bootstrap-Stichprobenanzahl innerhalb der durchgeführten Signifikanztests. Diese wurde auf den Wert $n = 100$ festgelegt, welcher im Rahmen derartiger Signifikanztests (in der Regel mindestens 1000) als sehr klein zu bewerten ist (Chernik, 2011). Zweiterer bezieht sich auf die Wiederholungen der Simulationsexperimente, aus welchen die Verteilungen von *Mean* $P_{greater}$ generiert worden sind. Diese wurden ebenfalls nur auf den Wert $n = 100$ beschränkt, was im Rahmen von Simulationsstudien einen genauso geringen Wert darstellt. Beide Kritikpunkte haben zur Folge, dass die Ergebnisse dieser Arbeit (Tabelle 1-12) leichte Ungenauigkeiten aufweisen. Dies erklärt auch die Ausnahmefälle, in denen die Mittelwerte von *Mean* $P_{greater}$ unter ansteigender Interaktionskonstante geringfügig abnehmen (z. B. Tabelle 2 unter “Medium Noise“ und “Sigmoid“). Dennoch war die Entscheidung für derartig geringe Anzahlen notwendig, um den Rechenaufwand der Simulationen in einem umsetzbaren Rahmen zu halten. Die heute verfügbaren Implementationen isotonischer Regression erlauben zwar eine schnelle Berechnung des

besten monotonen Modells (Busing, 2022), die Kombination des aufwendigen PIRST-Algorithmus mit einem großen Simulationsdesigns bringt diese jedoch an ihre Grenzen.

## 5.3 Anwendungsempfehlungen

Wer sich für eine Datenanalyse mit PIRST entschließt, muss bezüglich Permutationsanzahl, Bootstrap-Stichprobenanzahl, Signifikanzkriterium und Platzierung der AVs auf den Achsen eine Entscheidung treffen. In diesem Abschnitt werden konkrete Empfehlungen dafür gegeben.

### *5.3.1 Permutationsanzahl*

Die Anzahl der gezogenen Permutationen kann prinzipiell nie zu hoch sein, allerdings macht es wenig Sinn 1000 Permutationen zu verwenden, wenn z. B. nur 70 Permutationen möglich sind. Dies verlangsamt den Algorithmus nur unnötigerweise und limitiert gegebenenfalls die Wahl anderer Parameter. Eine gute Heuristik ist die Folgende: Für bis zu vier Bedingungen pro Kondition wählt man 100 Permutationen, andernfalls erhöht man den Wert auf 1000. Wie einige Simulationstests zeigten, verbesserten mehr Permutationen die Genauigkeit der Ergebnisse in kaum erkennbarer Weise.

### *5.3.2 Bootstrap-Stichprobenanzahl*

Die Bootstrap-Stichprobenanzahl im Rahmen des Signifikanztests kann ebenfalls nie zu hoch gewählt werden. Ist diese jedoch zu klein, schwankt möglicherweise der berechnete *p*-Wert. Dies kann sich ungünstig auf die Interpretation des Ergebnisses auswirken. Ist ein Effekt in Realität signifikant aber der *p*-Wert liegt eng am gewählten Signifikanzkriterium, dann kann ein Effekt aufgrund solcher Schwankungen fälschlicherweise abgelehnt werden. Ein allgemein guter Richtwert für derartige Signifikanztests sind 1000 Samples; für sehr hohe Genauigkeit sollten 10000 gewählt werden (Chernik, 2011).

### *5.3.3 Signifikanzkriterium*

Die Ergebnisse dieser Arbeit zeigen, dass ein $\alpha$ von 5% ein gutes, jedoch nicht immer das beste Entscheidungskriterium darstellt. Dies zeigt sich daran, dass die Sensitivitätswerte für $\alpha = .05$ deutlich anfälliger für Messfehler sind als AUC-Werte und Spezifität. So ist dieses Entscheidungskriterium insgesamt eher als konservativ einzuschätzen und ein höher gewähltes $\alpha$ sollte in vielen Fällen die differenzierungsstärkere Wahl sein. Dies erklärt sich darin, dass die Funktionskurve des Nullmodels so erzeugt wird, dass sie randweise die Monotonie-Bedingung erfüllt. Dadurch werden Monotonie-Verletzungen häufiger und die Abweichungen von $Mean\ P_{greater}$ stärker. Werden kleine Effekte erwartet, ist es demnach eine gut begründete Entscheidung höhere $\alpha$-Werte zu nutzen. So ist die Empfehlung je nach erwarteter Effektstärke ein $\alpha$ von 5% oder 10% zu wählen.

### *5.3.4 Platzierung der AVs auf den Achsen*

Das erweiterte PIRST nutzt, außer für die Konstruktion des Nullmodels, einfache isotonische Regression entlang der x-Achse. Da beide AVs jedoch in Theorie dasselbe messen und logisch betrachtet keine ein Prädiktor der anderen ist, stellt sich die Frage wie die Variablen auf den Achsen platziert werden sollten. Die beste Empfehlung ist, sich hierfür an den Simulationen zu orientieren und die Variable, entlang derer der Konditionseffekt erwartet wird, auf der y-Achse zu platzieren. Alternativ können auch beide Varianten gerechnet und ein zusammengesetztes Ergebnis berichtet werden.

## 5.4 Fazit

In Rahmen dieser Arbeit wurde ein neuartiges statistisches Testverfahren entwickelt, mit welchem sich Theorien über die Anzahl latenter Prozesse innerhalb verschiedener kognitiver Fähigkeiten überprüfen lassen. Die Validität des Verfahrens wurde über 648 realitätsnahe Simulationsexperimente sichergestellt. Mit dem Abschluss dieser Arbeit wurden konkrete Empfehlungen für die Auswertung des Verfahrens sowie die Datenerhebung bei geplanter Anwendung gegeben. Eine Code-Vorlage zur praktischen Anwendung mit der Statistiksoftware *R* und die zugehörige Nutzungsanleitung befindet sich in Anhang 1.

Für die Forschung besteht der nächste Schritt darin, die Frage zu beantworten, wie gut das erweiterte PIRST im Vergleich zu konkurrierenden Testverfahren performt. Unter Betrachtung der Ergebnisse von Benjamin et al. (2019), lässt sich zwar bereits eine optimistische Antwort erahnen, dies bedarf jedoch weiteren Untersuchungen.

# 6. Literaturverzeichnis

Ashby, F. G., Alfonso-Reese, L. A., Turken, A. U., & Waldron, E. M. (1998). A neuropsychological theory of multiple systems in category learning. *Psychological Review, 105*(3), 442–481.

Bamber, D. (1979). State-trace analysis: A method of testing simple theories of causation. *Journal of Mathematical Psychology, 19*(2), 137–181.

Benjamin, A. S., Griffin, M. L., & Douglas, J. A. (2019). A nonparametric technique for analysis of state-trace functions. *Journal of Mathematical Psychology*, *90*, 88-99.

Brainerd, C. J., & Reyna, V. F. (2010). Recollective and nonrecollective recall. *Journal of memory and language*, *63*(3), 425-445.

Busing, F. M. T. A. (2022). Monotone Regression: A Simple and Fast O(n) PAVA Implementation. *Journal of Statistical Software, Code Snippets*, *102*(1), 1–25.

Chernick, M. R. (2011). *Bootstrap methods: A guide for practitioners and researchers*. John Wiley & Sons.

Coltheart, M., Rastle, K., Perry, C., Langdon, R., & Ziegler, J. (2001). Drc: A dual route cascaded model of visual word recognition and reading aloud. *Psychological Review, 108*(1), 204–256.

Dunn, J. C. (2008). The dimensionality of the remember-know task: a state-trace analysis. *Psychological review*, *115*(2), 426.

Dunn, J. C., & Kalish, M. L. (2018). *State-trace analysis*. New York: Springer

Dunn, J. C., & Kirsner, K. (1988). Discovering functionally independent mental processes: The principle of reversed association. *Psychological Review, 95*(1), 91–101.

Evans, J. S. B. T. (2003). In two minds: Dual-process accounts of reasoning. *Trends in Cognitive Sciences, 7*(10), 454–459.

Goodale, M. A., & Milner, A. D. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences, 15*(1), 20–25.

Heit, E., & Rotello, C. M. (2010). Relations between inductive reasoning and deductive reasoning. *Journal of Experimental Psychology: Learning, Memory and Cognition, 36*(3), 805–812.

Jacoby, L. L. (1991). A process dissociation framework: Separating automatic from intentional uses of memory. *Journal of Memory and Language, 30*(5), 513–541.

Kalish, M. L., Dunn, J. C., Burdakov, O. P., & Sysoev, O. (2016). A statistical test of the equality of latent orders. *Journal of Mathematical Psychology, 70*, 1–11

Loftus, G. R. (1978). On interpretation of interactions. *Memory & Cognition, 6*(3), 312–319.

Loftus, G. R., Oberg, M. A., & Dillon, A. M. (2004). Linear theory, dimensional theory, and the face-inversion effect. *Psychological Review, 111*(4), 835–863.

Mandler, G. (1980). Recognizing: The judgment of previous occurrence. Psychological Review, 87(3), 252–271.

Newell, B. R., & Dunn, J. C. (2008). Dimensions in data: Testing psychological models using state-trace analysis. *Trends in Cognitive Sciences, 12*(8), 285–290.

Prince, M., Brown, S., & Heathcote, A. (2012). The design and analysis of state-trace experiments. *Psychological Methods, 17*(1), 78–99.

Prince, M., Hawkins, G., Love, J., & Heathcote, A. (2012). An R package for state-trace analysis. *Behavior Research Methods, 44*(3), 644–655.

Wagenmakers, E. J., Krypotos, A. M., Criss, A. H., & Iverson, G. (2012). On the interpretation of removable interactions: A survey of the field 33 years after Loftus. *Memory & Cognition, 40*(2), 145–160.

Wixted, J. T. (2007). Dual-process theory and signal-detection theory of recognition memory. *Psychological Review, 114*(1), 152–176.

## Anhang 1

Um PIRST nach Benjamin et al. (2019) auszuführen, wird die Funktion *pirst()* benötigt. Für den vollständigen PIRST-Algorithmus mit Signifikanztest muss *pirst_test()* benutzt werden.

- *pirst()*

**Input 1**: Dataframe der Form:

| | X | Y | Condition | Performance-Level | Measurement | Person |
|---|---|---|---|---|---|---|
| 1 | 0.1924646 | 0.3463719 | 1 | 1 | 1 | 1 |
| 2 | 0.6956952 | 0.3765527 | 1 | 2 | 1 | 1 |
| 3 | 0.6875184 | 0.7199673 | 1 | 3 | 1 | 1 |
| 4 | 0.8655014 | 0.7370147 | 1 | 4 | 1 | 1 |
| 5 | 0.3025038 | 0.1557383 | 2 | 1 | 1 | 1 |
| 6 | 0.3124431 | 0.5848850 | 2 | 2 | 1 | 1 |
| 7 | 0.7427404 | 0.7933778 | 2 | 3 | 1 | 1 |
| 8 | 0.7991143 | 0.8969737 | 2 | 4 | 1 | 1 |

**Input 2:** Anzahl der Permutationen.

Beispiel:

```
1. Test <- pirst(data = STA_data,
2.               nperms = 1000)
```

- *pirst_test()*

**Input 1**: Dataframe wie bei *pirst()*

**Input 2:** Anzahl der Permutationen

**Input 3:** Anzahl der Bootstrap-Stichproben für den Signifikanztest

**Input 4**: Konfidenz

Beispiel:

```
1. Test <- pirst_test(data = STA_data,
2.                    nperms = 1000,
3.                    nboot = 1000,
4.                    conf = 0.95)
```

### Code-Vorlage PIRST

```
 1. #Aggregates X and Y over repeated measures of one person
 2. aggregate <- function(data) {
 3.   newx <- c()
 4.   newy <- c()
 5.   newcondition <- c()
 6.   newtrace <- c()
 7.   newperson <- c()
 8.   datanew <- c()
 9.   for (i in 1:max(data$Person)) {
10.     for (j in 1:max(data$Condition)) {
11.       for (h in 1:max(data$Trace)) {
```

```
        newx <-  mean(data[data$Person == i &
                           data$Condition == j &
                           data$Trace == h, ]$X, na.rm = TRUE)
        newy <-  mean(data[data$Person == i &
                           data$Condition == j &
                           data$Trace == h, ]$Y, na.rm = TRUE)
        newcondition <- j
        newtrace <- h
        newperson <- i
        newrow <- cbind(newx, newy, newcondition, newtrace, newperson)
        datanew <- rbind.data.frame(datanew, newrow)
      }
    }
  }
  names(datanew) <- c("X", "Y", "Condition", "Trace", "Person")
  datanew
}

# Identifies eligible points
find_eligible_points <- function(data) {
  names(data) <- c("X", "Y", "Condition", "Trace")
  # Create a dataframe of eligibilityvalues for the data
  eligibilityframe <-
    as.data.frame(matrix(1, nrow = nrow(data), max(data$Condition)))
  eligibilitynames <- c()
  for (i in 1:(max(data$Condition))) {
    eligibilitynames[i] <- paste0("eligibility ", i)
  }
  names(eligibilityframe) <- eligibilitynames
  data <- cbind(data, eligibilityframe) # Add it to the data
  # In the next process the algorithm identifies all conditions  a point is
  # eligible with and adds that to the dataframe
  data <-
    data[order(data$Condition, data$X), ] # needed for next function
  for (p in 1:max(data$Condition)) {
    for (q in 1:max(data$Condition)) {
      for (i in 1:(nrow(data))) {
        if ((data$Condition)[i] == p) {
          data[i, p + 4] <- 1
        } else if (data$Condition[i] == q) {
          if ((data$X[i] >= max(data$X[data$Condition == p]) &&
               data$Y[i] >= max(data$Y[data$Condition == p])) ||
              data$X[i] <= min(data$X[data$Condition == p]) &&
              data$Y[i] <= min(data$Y[data$Condition == p])) {
            data[i, p + 4] <- 0
          }
        }
      }
    }
  }
  data
}

# Identifies the Conditions, a point is eligible with (for ncons > 2)
combfinder <- function(data) {
  eligibilityframe <- data[, 5:(4 + max(data$Condition))]
  eligibilitycombs <- dplyr::distinct(eligibilityframe)
  combination <- seq(1, nrow(eligibilitycombs), 1)
  eligibilitycombs$combination <- combination
  eligibility <- c(rep(0, nrow((data))))
  for (i in 1:nrow(eligibilitycombs)) {
    for (j in 1:(nrow(data))) {
      if (all(data[j, 5:(4 + max(data$Condition))] == eligibilitycombs[i,
1:max(data$Condition)]) == TRUE) {
        eligibility[j] <- eligibilitycombs$combination[i]
      }
    }
  }
  data$eligibility <- eligibility # Every point gets a specific eligibility
  data                            # decoding-number
}
```

```
82.
83. # Identifies if data is not completely overlapping (ncons >2) or not overlapping at all
84. find_cuts <- function(data) {
85.   cuts <- data
86.   result <- 0
87.   # Starts at condition one and checks from conditions to condition if there is
88.   # any overlap to find out if there is a cut between conditions
89.   for (j in 1:max(cuts$Condition)) {
90.     for (i in 1:nrow(cuts)) {
91.       if ((cuts[i, 4 + 1] == 1) && (cuts$Condition[i] != 1)) {
92.         cuts[, 4 + j][cuts$Condition == cuts$Condition[i]] = 1
93.         cuts$Condition[i] = 1
94.       }
95.     }
96.   }
97.   if (all(cuts$Condition == data$Condition) == TRUE) {
98.     result <- 1 # No permutations possible
99.   } else if (all(cuts$Condition == rep(1, nrow(cuts))) == TRUE) {
100.    result <- 0 # No cuts, overlap of all conditions
101.  } else {
102.    result <- 2 # At least one condition is isolated
103.  }
104.  result
105. }
106.
107. # Runs extremely fast isotonic regression
108. fastiso <- function (y,x) {
109.   if (anyDuplicated (x) == 0) {
110.     switchback <- seq_along(y)
111.     isomat <- cbind(y,x,switchback)
112.     isomat <- isomat[Rfast::Order(isomat[, 2]),]
113.     isomat[, 1] <- monotone::monotone(isomat[, 1])
114.     isomat <- isomat[Rfast::Order(isomat[, 3]),]
115.     y <- isomat[, 1]
116.     y
117.   } else {#tie handling, because monotone does not have one (extremly few cases)
118.     isoreg <- isotone::gpava(x,y)
119.     y <- isoreg[["x"]]
120.     y
121.   }
122. }
123.
124. # Runs isotonic regression with x-axis a predictor and adds values to data
125. isoreg <- function(data) {
126.   Y_isoreg <- c()
127.   for (i in 1:max(data$Condition)) {
128.     isoreg <- fastiso(data$Y[data$Condition == i],
129.                               data$X[data$Condition == i])
130.     Y_isoreg <- c(Y_isoreg, isoreg)
131.   }
132.   data$Y_isoreg <- Y_isoreg
133.   data
134. }
135.
136. # Calculates SSE for the original data
137. SSE_calc <- function(data) {
138.   SSE_gather <- c()
139.   for (i in 1:max(data$Condition)) {
140.     SSE_hold <- sum((data$Y_isoreg[data$Condition == i] -
141.                        data$Y[data$Condition == i]) ^ 2)
142.     SSE_gather[i] <- SSE_hold
143.   }
144.   SSE <- sum(SSE_gather)
145.   SSE
146. }
147.
148. # Shuffles conditionlabels from points of the same eligibilitycombination randomly
149. shuffle_more <- function(data) {
150.   new_perm <- data[c("Condition","eligibility")]
151.   for (i in 1:max(data$eligibility)) {
152.     if (length(new_perm$Condition[new_perm$eligibility == i]) > 1) {
```

```
153.       shuffle <- sample(new_perm$Condition[new_perm$eligibility == i])
154.       new_perm$Condition[data$eligibility == i] <- shuffle
155.     }
156.   }
157.   data$Condition <- new_perm$Condition
158.   data
159. }
160.
161. # Calculates SSE of a permutation without adding any values to a dataframe
162. isoreg_ssepermcalc <- function(data) {
163.   SSE_gather <- c()
164.   for (i in 1:max(data$Condition)) {
165.     iso_reg <- fastiso(data$Y[data$Condition == i],data$X[data$Condition == i])
166.     SSE_gather[i] <- sum((iso_reg - data$Y[data$Condition == i]) ^ 2)
167.   }
168.   SSE <- sum(SSE_gather)
169.   SSE
170. }
171.
172. # Combines shuffle_more and isoreg_ssepermcalc to permutate and get SSE in one step
173. permutate <- function(data, nperms) {
174.   cont_shuffle <-
175.     data$Condition # For shuffling again if permutation = original
176.   SSE_all <- c(rep(0, nperms))
177.   for (i in 1:nperms) {
178.     data <- shuffle_more(data)
179.     count <- 0
180.     while ((all(data$Condition == cont_shuffle) == TRUE) && (count < 1000) ) {
181.       data <- shuffle_more(data) # Bugfix for a very rare case of overlap but no
182.       count <- count + 1         # possible permuations where the algorithm hangs up
183.     }
184.     if (count < 1000){
185.       SSE_all[i] <- isoreg_ssepermcalc(data)
186.     } else {   # In this rare case, there are no possible permutations
187.       SSE_all <- c(rep(NaN, nperms))
188.       message("BUT no permutations are possible, this subject will not have impact on the
Test-statistic")
189.       break
190.     }
191.   }
192.   SSE_all
193. }
194.
195. ## HEART OF THE ALGORITHM
196. # Combines all previous functions to a PIRST for a single case
197. pirst_sc <- function(data, nperms) {
198.   data <- find_eligible_points(data)
199.   data <- combfinder(data)
200.   data <- isoreg(data)
201.   SSE_compare <- SSE_calc(data)
202.   info <- find_cuts(data)
203.   if (info == 1) {
204.     message(
205.       "Because of no overlap, this subject will not have impact on the Test-statistic!"
206.     ) # No Overlap. Bad!
207.     SSE <- NaN
208.   } else if (info == 2) {
209.     message("Data is NOT completely overlapping!")
210.     SSE <- permutate(data, nperms) # Just to inform experimenter
211.   } else {
212.     message("Data is completely overlapping.")
213.     SSE <- permutate(data, nperms) # Best case, everything is fine
214.   }
215.   Proportion_greater <- sum(SSE > SSE_compare) / length(SSE)
216.   Proportion_equal <- sum(SSE == SSE_compare) / length(SSE)
217.   Proportion_smaller <- sum(SSE < SSE_compare) / length(SSE)
218.   Statistic <-
219.     list(Proportion_greater,
220.          Proportion_equal,
221.          Proportion_smaller,
222.          SSE,
```

```
223.          SSE_compare,
224.          data)
225.   names(Statistic) <-
226.     c(
227.       "Proportion greater",
228.       "Proportion equal",
229.       "Proportion smaller",
230.       "All SSE's",
231.       "SSE original",
232.       "Data"
233.     )
234.   Statistic
235. }
236.
237. # Applies pirst_sc to every person and calculates mean p-greater and other statistics
238. pirst <- function(data, nperms= 100) {
239.   names(data) <-
240.     c("X", "Y", "Condition", "Trace", "Measurement", "Person")
241.   SSE_greater <- c(rep(0, max(data$Person)))
242.   SSE_equal <- c(rep(0, max(data$Person)))
243.   SSE_smaller <- c(rep(0, max(data$Person)))
244.   SSE_original <- c(rep(0, max(data$Person)))
245.   output_data <- as.data.frame(c())
246.   data <- aggregate(data)
247.   for (i in 1:max(data$Person)) {
248.     calc_simdata <- data[data$Person == i, ]
249.     calc_simdata <- calc_simdata[, -5]
250.     message("For subject ", i)
251.     get_sc_data_greater <- pirst_sc(calc_simdata, nperms)
252.     SSE_greater[i] <- get_sc_data_greater[["Proportion greater"]]
253.     SSE_equal[i] <- get_sc_data_greater[["Proportion equal"]]
254.     SSE_smaller[i] <- get_sc_data_greater[["Proportion smaller"]]
255.     SSE_original[i] <- get_sc_data_greater[["SSE original"]]
256.     output_data <- rbind(output_data, get_sc_data_greater[["Data"]])
257.   }
258.   SSE_data <- as.data.frame(cbind(SSE_greater, SSE_equal, SSE_smaller, SSE_original))
259.   SSE_data <- na.omit(SSE_data)
260.   Prop_greater <- mean(SSE_data$SSE_greater)
261.   Prop_equal <- mean(SSE_data$SSE_equal)
262.   Prop_smaller <- mean(SSE_data$SSE_smaller)
263.   SSE_original_mean <- mean(SSE_data$SSE_original)
264.   Prop_greater_all <- SSE_data$SSE_greater
265.   SSE_original_all <- SSE_data$SSE_original
266.   Person <- data$Person
267.   output_data <- cbind(output_data, Person)
268.   Statistic <-
269.     list(
270.       Prop_greater,
271.       Prop_equal,
272.       Prop_smaller,
273.       SSE_original_mean,
274.       Prop_greater_all,
275.       SSE_original_all,
276.       output_data
277.     )
278.   names(Statistic) <-
279.     c(
280.       "Mean P greater",
281.       "Mean P equal",
282.       "Mean P smaller",
283.       "Mean of original SSE's",
284.       "Proportion of SSE's greater original SSE",
285.       "All original SSE's",
286.       "Data"
287.     )
288.   Statistic
289. }
```

## Code-Vorlage PIRST-Signifikanztest

```
# Calculates isotonic regressions for X and Y-values for a specific order
# Needed for easy_cmr
combined_isoreg_fit <- function(data) {
  isoY <- fastiso(data$Y, data$Order)
  isoX <- fastiso(data$X, data$Order)
  ErrorY <- sum((isoY - data$Y) ^ 2)
  ErrorX <- sum((isoX - data$X) ^ 2)
  fit <- ErrorY + ErrorX
  fit
}

# Runs CMR for every condition seperately
# Needed for easy_cmr
preorder <- function(data) {
  for (i in 1:max(data$Condition)) {
    getpreordered_X <- fastiso(data$X[data$Condition == i],
                               data$Trace[data$Condition == i])
    getpreordered_Y <- fastiso(data$Y[data$Condition == i],
                               data$Trace[data$Condition == i])
    data$X[data$Condition == i] <- getpreordered_X
    data$Y[data$Condition == i] <- getpreordered_Y
  }
  data
}

# Runs an easy cmr-Version with a sorting-algorithm
easy_cmr <- function(data) {
  data <- preorder(data)
  data$Order <- rep(0, nrow(data))
  cmr_dat <- data[data$Condition == 1, ]
  cmr_dat <- cmr_dat[order(cmr_dat$Condition, cmr_dat$Trace), ]
  cmr_dat$Order <- seq(1, nrow(cmr_dat))
  # Searches for the best fitting spot of a point of an ordered condition
  # in another ordered condition and continues until the whole condition is
  # sorted in
  for (i in 2:max(data$Condition)) {
    shorten_search <- 0.5
    for (j in 1:max(data$Trace)) {
      best_order <- matrix(0, nrow = nrow(cmr_dat) + 1, ncol = 2)
      for (k in seq(from = shorten_search,
                    to = max(cmr_dat$Order) + 1,
                    by = 1)) {
        newrow <- data[data$Condition == i & data$Trace == j, ]
        newrow$Order[1] <- k
        cmr_dat_plus <- cmr_dat
        cmr_dat_plus <- dplyr::bind_rows(cmr_dat_plus, newrow)
        fit <- combined_isoreg_fit(cmr_dat_plus)
        best_order[k + 0.5, 1] <- fit
        best_order[k + 0.5, 2] <- k
      }
      if (shorten_search > 0.5) {
        best_order <- as.matrix(best_order[-c(1:(shorten_search - 0.5)), ])
      }
      # Sometimes the best_order matrix get automatically transformed into a
      # vector, this fixes indices and continues the algorithm
      if (ncol(best_order) > 1) {
        placement <- best_order[which.min(best_order[, 1]), 2]
      } else {
        placement <- best_order[2, 1]
      }
      newrow$Order <- placement
      cmr_dat <- rbind(cmr_dat, newrow)
      cmr_dat <- cmr_dat[order(cmr_dat$Order), ]
      cmr_dat$Order <- seq(1, nrow(cmr_dat))
      shorten_search <-
        cmr_dat$Order[cmr_dat$Condition == i & cmr_dat$Trace == j] + 0.5
    }
  }
  isoY <- fastiso(cmr_dat$Y, cmr_dat$Order)
```

```
  isoX <- fastiso(cmr_dat$X, cmr_dat$Order)
  cmr_dat$X <- isoX
  cmr_dat$Y <- isoY
  cmr_dat
}

# Get the deviation-distribution of every point for every person
# Uses the raw data and the pirst(data,...)-output
get_dist <- function(data, test) {
  datanew <- c()
  for (i in 1:max(data$Person)) {
    for (j in 1:max(data$Condition)) {
      for (h in 1:max(data$Trace)) {
        distx <- data[data$Person == i &
                        data$Condition == j &
                        data$Trace == h, ]$X -
          test[["Data"]][test[["Data"]]$Person == i &
                           test[["Data"]]$Condition == j &
                           test[["Data"]]$Trace == h, ]$X

        disty <-  data[data$Person == i &
                         data$Condition == j &
                         data$Trace == h, ]$Y -
          test[["Data"]][test[["Data"]]$Person == i &
                           test[["Data"]]$Condition == j &
                           test[["Data"]]$Trace == h, ]$Y

        newcondition <- rep(j, length(distx))
        newtrace <- rep(h, length(distx))
        newperson <- rep(i, length(distx))
        newrow <-
          cbind(distx, disty, newcondition, newtrace, newperson)
        datanew <- rbind.data.frame(datanew, newrow)
      }
    }
  }
  names(datanew) <-
    c("distX", "distY", "Condition", "Trace", "Person")
  datanew
}

# Creates the nullmodel and the nulleffect-Dataframe
create_null <- function(test) {
  # First create the nullmodel
  testdata <- test[["Data"]]
  nulldata <- c()
  for (i in 1:max(testdata$Person)) {
    nulldata_part <- easy_cmr(testdata[testdata$Person == i,])
    nulldata <- rbind(nulldata, nulldata_part)
  }
  nulldata <- cbind.data.frame(
    nulldata$X,
    nulldata$Y,
    nulldata$Condition,
    nulldata$Trace,
    rep(1, nrow(nulldata)),
    nulldata$Person
  )
  names(nulldata) <-
    c("X", "Y", "Condition", "Trace", "Measurement", "Person")
  nulldata
  # Nullmodel is complete, now deviations can be added
}
create_null_plus_dev <- function(nulldata, dist) {
  for (i in 1:max(dist$Person)) {
    for (j in 1:max(dist$Condition)) {
      for (h in 1:max(dist$Trace)) {
        randx <- dist[dist$Person == i &
                        dist$Condition == j &
                        dist$Trace == h,]$distX
        randy <- dist[dist$Person == i &
```

```
                        dist$Condition == j &
                        dist$Trace == h,]$distY
        randxmean <- 0
        randymean <- 0
        for (k in 1:min(length(randx), length(randy))) {
          randdev <- sample(1:min(length(randx), length(randy)), 1)
          randxmean <- randxmean + randx[randdev]
          randymean <- randymean + randy[randdev]
        }
        randxmean <- randxmean / min(length(randx), length(randy))
        randymean <- randymean / min(length(randx), length(randy))
        nulldata[nulldata$Person == i &
                   nulldata$Condition == j &
                   nulldata$Trace == h,]$X <-
          nulldata[nulldata$Person == i &
                     nulldata$Condition == j &
                     nulldata$Trace == h, ]$X +
          randxmean
        nulldata[nulldata$Person == i &
                   nulldata$Condition == j &
                   nulldata$Trace == h,]$Y <-
          nulldata[nulldata$Person == i &
                     nulldata$Condition == j &
                     nulldata$Trace == h,]$Y +
          randymean
      }
    }
  }
  nulldata
}

# Create null-distribution by drawing nulleffect-dataframes and collect mean p-greater
create_null_dist <- function(test, dist, ndraws, nperms = 100) {
  null_dist <- c()
  null_without_dev <- create_null(test)
  for (i in 1:ndraws) {
    nulldata <- create_null_plus_dev(null_without_dev, dist)
    null_test <- pirst(nulldata, nperms)
    null_dist[i] <- null_test[["Mean P greater"]]
  }
  null_dist
}

# Combines previous functions
# Runs a significance test for the found effect given the null-distribution
pirst_conf <- function(data, test, ndraws, nperms = 100, conf = 0.95) {
  dist <- get_dist(data, test)
  nulldist <- create_null_dist(test, dist, ndraws, nperms)
  upper <- round(conf * length(nulldist))
  nulldist <- sort(nulldist)
  upperbound <- nulldist[upper]
  p <- sum(nulldist > test[["Mean P greater"]]) / length(nulldist)
  significance <- FALSE
  if (upperbound < test[["Mean P greater"]]) {
    significance <- TRUE
  }
  Significancetest <-
    list(test[["Mean P greater"]], upperbound, significance, p, conf, nulldist)
  names(Significancetest) <-
    c("Mean P greater",
      "Upper",
      "Significance",
      "p-Value",
      "Confidence",
      "Null-Distribution")
  Significancetest
}

# All together: the final test-function
pirst_test <- function(data, nperms = 100, nboot = 100, conf = 0.95){
  test <- pirst(data, nperms)
```

```
  Significancetest <- pirst_conf(data = data,
                                 test = test,
                                 ndraws = nboot,
                                 nperms = nperms,
                                 conf = conf)
  Significancetest
}
```

## Anhang 2

Um einen Simulationsdatensatz zu erzeugen kann die Funktion *pirstsim()* genutzt werden. Die Simulationsparameter werden folgendermaßen angepasst:

**Input 1**: Anzahl der Bedingungen pro Kondition

**Input 2**: Anzahl der Konditionen

**Input 3**: Anzahl der Punkte pro Kondition die **nicht** überlappen

**Input 4**: Trennung der Konditionskurven entlang der y-Achse

**Input 5**: Standardabweichung des Gauß-Rauschens

**Input 6**: Form der Funktionskurve (0- linear, 1- konkav, 2- sigmoidal)

**Input 7**: Anzahl der Messwiederholungen pro Person

**Input 8**: Anzahl der Versuchspersonen

Beispiel:

```
Data <- pirstsim(npercon = 4,
                 ncons = 2,
                 overlap = 1,
                 interactsize = c(0, 0.1),
                 noise = 0.1,
                 curve = 2,
                 nmeasures = 3,
                 cases = 30)
```

### Code-Vorlage

```
# STEP 1
## Create data for a single ST-plot for one person
pirstsim_sc <- function(npercon,      # Number of points per condition
                        ncons,        # Number of conditions
                        overlap,      # Points overlapping
                        interactsize, # Single or multiple process model, vector
                        noise,        # Gaussian noise
                        curve) {      # Linear, Concave, Sigmoid
  ## Producing the linear funtion
  npoints <- npercon * ncons
  Condition <- c()
  for (i in 1:ncons) {
    Condition <- c(Condition, rep(i, npercon))
  }
  p <- overlap
  M <- matrix(nrow = ncons, ncol = npercon)
  # Producing the points
  for (i in 1:ncons) {
    for (j in 1:npercon){
      M[i, j] <- 0.7 * (j / npercon) + 0.15
      M[i, j] <- 0.7 * (j / npercon) + 0.15
    }
  }
  X <- c()
  for (i in 1:nrow(M)){
    X <- c(X, M[i, ])
  }
  # Evenly strechtch points out
  if (ncons == 1) {
    correct <- ncons+1
  } else {
    correct <- ncons
  }

```

```
  for (i in 1:npoints) {
    X[i] <- X[i] + (-0.35 + 0.7 * (Condition[i] - 1) / (correct - 1)) * ((p - 1 / ncons) / npercon)
  }
  ## Linear, curve, curve down, sigmoid or mirrored-sigmoid cases
  if (curve == 0) {
    # Linear
    Y <- X
    X <- X
  } else if (curve == 1) {
    # Transforming to Concave Up
    Y <- sin((90 * X + 270) * pi / 180) + 1
    X <- cos((90 * X + 270) * pi / 180)
  } else if (curve == -1) {
    # Transforming to Concave Down
    Y <- cos((90 * X + 270) * pi / 180)
    X <- sin((90 * X + 270) * pi / 180) + 1
  } else if (curve == 2) {
    Y <- 0.5 * (1 + tanh(X * 10 - 5))
    X <- X
  }else if (curve == -2) {
    Y <- X
    X <- 0.5 * (1 + tanh(X * 10 - 5))
  } else {
    warning("invalid Curve input")
  }
  ## Adding the Interaction constant to Conditions
  for (i in 1:npoints) {
    Y[i] <- Y[i] + interactsize[Condition[i]]
  }
  ## Adding noise
  Y <- Y + rnorm(npoints, 0, noise)
  X <- X + rnorm(npoints, 0, noise)
  ## Combine to a dataframe
  Trace <- rep(seq(1, npercon, 1), ncons)
  simdata <- as.data.frame(cbind(X, Y, Condition, Trace))
  simdata
}

#STEP 2
##Repeat pirstsim_sc for multiple measures on one Person
pirstsim_rep_sc <- function(npercon,
                            ncons,
                            overlap,
                            interactsize,
                            noise,
                            curve,
                            nmeasures){
  measurements <- c()
  for (i in 1: nmeasures) {
    m <- pirstsim_sc(npercon, ncons, overlap, interactsize, noise, curve)
    m$Measurement <- rep(i, nrow(m))
    measurements <- rbind(measurements, m)
  }
  measurements
}

# STEP 3
## Create a full STA data-Set of multiple persons with multiple measures
pirstsim <- function(npercon = 4,
                     ncons = 2,
                     overlap = 1,
                     interactsize = c(0, 0),
                     noise = 0,
                     curve = 0,
                     nmeasures = 10,
                     cases = 100) {

  if (length(interactsize) != ncons) {
    stop("Number of conditions must match length(interactsize)")
  }
```

```
105.   exp_simdata <- c()
106.   for (i in 1:cases) {
107.     simdata_part <- pirstsim_rep_sc(npercon, ncons, overlap, interactsize, noise, curve,
    nmeasures)
108.     simdata_part$Person <- c(rep(i, npercon * ncons))
109.     exp_simdata <- rbind(exp_simdata, simdata_part)
110.   }
111.   exp_simdata
112. }
```

TECHNISCHE UNIVERSITÄT CHEMNITZ

Zentrales Prüfungsamt

Selbstständigkeitserklärung

| | |
|---|---|
| Name: | Bitte beachten: |
| Vorname: | 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein. |
| geb. am: | |
| Matr.-Nr.: | |

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum:..... Unterschrift: .....

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.