Zentrale Funktionen im Überblick

Maike Luhmann & Merle Widlok

Allgemeines

`install.packages("Paket")`: Paket installieren
`library(Paket)`: Paket laden
`search()`: Aktuell geladene Pakete anzeigen
`help()`: Hilfedatei für Paket öffnen
`help.search()`: Nennt Funktionen, die Suchbegriff verwenden
`objects()`: Inhalt des Workspaces
`remove(objekt)`: Entfernt Objekt aus dem Workspace
`setwd("...")`: Arbeitsverzeichnis ändern
`load("...")`: Datei öffnen
`save(objekt, file="...")`: Objekt speichern
`save.image(file="...")`: Workspace speichern
`c(x, y)`: Erstellt einen Vektor
`factor()`: Vektor in Faktor konvertieren
`levels()`: Faktorstufen eines Faktors angeben
`as.numeric(faktor)`: Faktor in Vektor konvertieren
`cbind(x, y)`, `rbind(x, y)`: Einzelne Vektoren werden als Spalten (cbind) bzw. als Zeilen in einer Matrix zusammengefasst
`data.frame(x, y)`: Erstellt einen Data Frame
`nrow()` bzw. `ncol()`: Anzahl der Zeilen bzw. Spalten eines Data Frames
`View()`: Ansicht eines Data Frames
`as.tibble()` [3,4]: Objekt in *Tibble* umwandeln

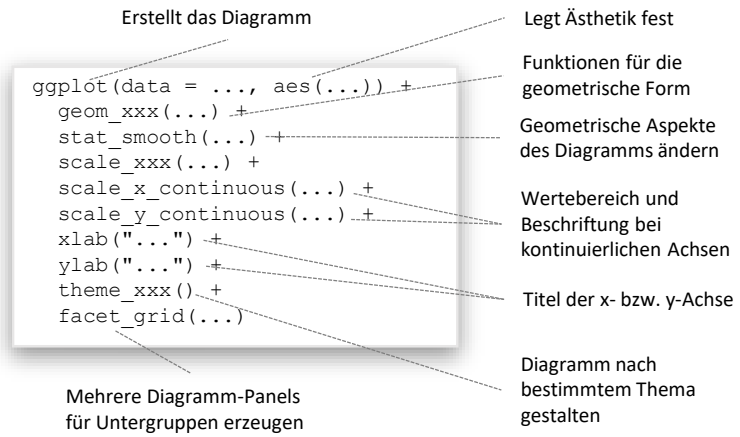
Objekteigenschaften

`class(x)`: Art des Objekts
`names(x)`: Namen der Komponenten des Objekts
`length(x)`: Anzahl der Elemente eines Objekts
`dim(x)`: Anzahl Zeilen eines Objekts
`str(x)`: Struktur des Objekts

Tabellen

`table(x)`: Häufigkeitstabelle
`table(x,y)`: Kontingenztabelle
`addmargins()`: Zeilen- und Spaltensummen zur Kontingenztabelle hinzufügen
`rowSums()`: Zeilensumme der Kontingenztabelle
`colSums()`: Spaltensumme der Kontingenztabelle
`prop.table(tabelle)`: Relative Häufigkeiten
`cumsum(tabelle)`: Kumulierte Häufigkeiten

Grafiken mit ggplot



Klassische Grafikfunktionen

`plot(x, y)`: Flexible Grafikfunktion für Streudiagramme, Boxplots etc.
`barplot(table(x))`: Säulendiagramm
`hist(x)`: Histogramm
`plot(density)`: Kerndichte-Diagramm
`boxplot(x)`: Einfacher Boxplot
`pairs.panels(x)`: Streudiagramm-Matrix
`par(...)`: Allgemeine Grafik-Parameter
`par(mfrow=c(a,b))`: Anzahl der Diagramme pro Zeile (a) und Spalte (b)
`error.bars()`, `error.bars.by()` [2]: Fehlerbalkendiagramm
`afex_plot()` [5]: Fehlerbalkendiagramm
`jpeg(…)`, `pdf(…)`: Diagramme im entsprechenden Dateiformat exportieren

Univariate deskriptive Statistik

`describe()` [2]: Deskriptive Statistiken
`describeBy()` [2]: Deskriptive Statistiken getrennt für bestimmte Gruppen
`which.max()`: Kategorie mit höchstem Wert
`which.min()`: Kategorie mit niedrigstem Wert
`quantile()`: Minimum, Maximum und drei Quantile
`na.rm=TRUE`: Argument bei fehlenden Werten

Klassische Datenaufbereitung

`which()`: Nummern der Zeilen, auf die logische Bedingung zutrifft
`ifelse()`: Definiert logische Bedingung und was passiert, wenn diese erfüllt/nicht erfüllt ist
`is.na()`: logische Abfrage, die fehlende und gültige Werte anzeigt
`rowMeans()` bzw. `rowSums()`: Zeilenmittelwerte bzw. Zeilensummen aller Variablen berechnen
`subset()`: Fälle/Variablen auswählen
`na.omit(daten)`: Fälle mit fehlenden Werten entfernen
`merge(data.frame.1, data.frame.2, by="x")`: Variablen aus zwei Data Frames zusammenfügen

Datenaufbereitung mit *Tidyverse*

`%>%`: Pipe verknüpft verschiedene tidyverse-Befehle
`mutate()` [3]: Variablen verändern/erstellen
`rename()` [3]: Variablen umbenennen
`recode()` [3]: Variablen umkodieren
`select()` [3]: Variablen auswählen
`filter()` [3]: Fälle nach Kriterien auswählen
`drop_na()` [4]: Fälle mit fehlenden Werten ausschließen
`arrange()` [3]: nach Variable sortieren
`bind_rows()` [3]: Data Frames mit unterschiedlichen Fällen zusammenfügen
`inner_join()`, `full_join()`, `left_join()`, `right_join()` [3]: Data Frames mit denselben Personen und unterschiedlichen Variablen zusammenfügen
`pivot_longer()` [4]: ins Long-Format umstrukturieren
`pivot_wider()` [4]: ins Wide-Format umstrukturieren
`group_by()` [3]: folgende tidyverse-Befehle werden getrennt für definierte Untergruppen durchgeführt

Bivariate deskriptive Statistik

`cor()`: Korrelationsmatrix
`cor.test(x,y)`: Signifikanztest für Korrelation
`corr.test(daten)` [2]: Korrelation, *N* und *p*-Werte
`cov()`: Varianz-Kovarianz-Matrix

t-Tests

t.test(av, mu=x): *t*-Test für eine Stichprobe
t.test(av ~ uv): *t*-Test für unabhängige Stichproben
t.test(av1, av2, paired=TRUE): *t*-Test für abhängige Stichproben
effsize::cohen.d() [10]: Effektgröße mit Konfidenzintervall für Mittelwertsvergleiche

Varianzanalyse

aov_car() [5]: verschiedene Varianzanalysen
emmeans() [11]: Multiple Paarvergleiche/Kontraste
pairs() [11]: Paarweise Vergleiche
afex_plot() [5]: Mittelwerte grafisch darstellen
nice() [5]: Varianzanalyse-Tabelle

Regressionsanalyse

lm(y ~ x): Einfache lineare Regression
lm(y ~ x1 + x2): Multiple Regression
anova(model1, model2): *F*-Test für den Vergleich zweier Regressionsmodelle
confint(): Konfidenzintervalle für einzelne Regressionskoeffizienten
fitted(): vorhergesagte Werte des Modells
resid(): Residuen des Modells
predict(): Vorhergesagte Werte für bestimmte Prädiktorvariablen
plot(): Diagramme zur Überprüfung der Modellannahmen
vif() [1]: Variance Inflation Factors für jeden Prädiktor

Spezielle Regressionsmodelle

lm(y ~ x1 * x2): Moderierte Regression
lm(y ~ x + I(x^2)): Regression mit quadratischem Zusammenhang
contrasts(faktor): Zeigt die aktuelle Kodierung eines Faktors an
relevel(): Faktorstufe als Referenzkategorie festlegen
emmeans() [11]: Paarweise Mittelwertsvergleiche
Anova() [1]: Prädiktoren auf Signifikanz testen
emtrends() [11]: Bedingte Regressionskoeffizienten bestimmen
interact_plot() [14]: Bedingte Regressionsgeraden darstellen
johnson_neyman() [14]: Johnson-Neyman-Intervall berechnen und grafisch darstellen
glm(y ~ x, family = binomial): Logistische Regression

Wahrscheinlichkeitsverteilungen

dt(x,df): Dichtefunktion der t-Verteilung für bestimmten Wertebereich x und bestimmte Anzahl an Freiheitsgraden df
qt(p,df): Quantil der t-Verteilung mit df Freiheitsgraden für bestimmte Wahrscheinlichkeit p
pt(q,df): p-Wert für Quantil q der t-Verteilung mit df Freiheitsgraden

Testkonstruktion

fa.parallel(daten) [2]: Parallelanalyse
VSS() [2]: Bestimmung Anzahl der Faktoren
fa(daten, faktoren, methode, rotation) [2]: Faktorenanalyse
alpha(daten) [2]: Itemanalyse

Weitere Tests

leveneTest(av ~ uv) [1]: Levene-Test
chisq.test(): chi^2-Test
wilcox.test(x, mu = y): Wilcoxon-Test
kruskal.test(y ~ x): Kruskal-Wallis-Test

Lineare Strukturgleichungsmodelle

'...' [6]: Spezifikation der Modellgleichungen
sem(...) [6]: Modell schätzen
cfa(...) [6]: konfirmatorische Faktorenanalyse
parameterEstimates (...) [6]: Modellparameter in Tabellenform, auch mit Konfidenzintervallen
fitMeasures(...) [6]: Modellgütemaße
modindices(...): [6] Modification-Indices
semPaths(...) [7]: Pfaddiagramm erstellen

Mehrebenenanalyse

lmer(y ~ Intercept + (x|Gruppe)) [8]: Spezifikation des Modells
performance::icc(...) [12]: Interklassenkorrelation berechnen
lmerTest::lmer() [13]: Freiheitsgrade und p-Werte für Mehrebenenmodell berechenen
dotplot(ranef(...)) [9]: Dotplot gruppenspezifischer Parameter
emtrends(...) [11]: Bedingte Regressionskoeffizienten bestimmen
interact_plot() [14]: Bedingte Regressionsgeraden darstellen

Ergebnisdarstellung

round(objekt, x): Rundet alle Werte eines Objekts auf *x* Nachkommastellen
sort(x): Gibt die Elemente eines Objekts in aufsteigender Reihenfolge wieder
summary(objekt): Fasst den Inhalt eines Objekts angemessen zusammen
write.csv2(tabelle): Speichert eine Tabelle als csv-Datei

Sonstige Funktionen

<- Zuweisungspfeil, erstellt ein neues Objekt
objekt$komponente: Wählt eine Komponente aus einem Objekt aus
object[a,b]: Wählt bestimmte Werte aus einem Objekt aus
scale(x): Variable standardisieren
scale(x, scale=FALSE): Variable zentrieren

Zusätzliche Pakete für die hier aufgeführten Funktionen:

[1] car
[2] psych
[3] dplyr
[4] tidyr
[5] afex
[6] lavaan
[7] semPlot
[8] lme4
[9] lattice
[10] effsize
[11] emmeans
[12] performance
[13] lmerTest
[14] interactions

Maike Luhmann & Merle Widlok