(6) Multivariate Varianzanalyse

Ziel dieser Sitzung ist es, die Durchführung einer einfaktoriellen multivariaten Varianzanalyse an einem Anwendungsbeispiel zu demonstrieren. Dazu betrachten wir die Anlayse von Differenzwerten für BDI-Scores und Glukokortikoidplasmalevel vor und nach der Intervention für drei Gruppen von jeweils 15 Patienti:innen, die unterschiedliche Psychotherapiesettings (face-to-face und online) bzw. eine Wartelistenkontrollbedingung durchlaufen haben.

Wir stellen dazu in Tabelle 1 einen simulierten Beispieldatensatz dar. Die erste Spalte von Tabelle 1 (`COND`) listet das spezifische Therapiesetting (`F2F`: face-to-face, `ONL`: online, `WLC`: waitlist control) der Patient:innen auf. Die zweite Spalte (`dBDI`) listet die entsprechenden BDI-Score-Differenzwerte und die dritte Spalte (`dGLU`) die entsprechenden Glukokortikoidplasmalevel-Differenzwerte für die ersten fünf Patient:innen jeder Studiengruppe auf. In beiden Fällen sollen positive Werte eine Abnahme der Depressionssymptomatik, negative Werte dagegen einer Zunahme der Depressionssymptomatik anzeigen.

```
library(knitr)                                  # knitr für Tabellen
D = read.csv("Multivariate_Varianzanalyse.csv") # Dateneinlesen
kable(D[c(1:5,16:20,31:35),],                   # Datentabelle
    digits = 1,
    align  = "c")
```

Tabelle 1. Differenzwerte für BDI-Score und Glukokortikoidplasmalevel prä/post Intervention von drei Studiengruppen (F2F: face-to-face, ONL: online, WLC: waitlist control) mit jeweils 15 Patienti:innen.

	COND	dBDI	dGLU
1	F2F	11	4.3
2	F2F	10	3.9
3	F2F	12	3.5
4	F2F	7	2.6
5	F2F	10	3.3
16	ONL	6	3.1
17	ONL	8	2.7
18	ONL	7	2.1
19	ONL	8	3.1
20	ONL	11	2.8
31	WLC	-2	0.7
32	WLC	2	1.4
33	WLC	1	1.0
34	WLC	2	0.9
35	WLC	3	1.6

Folgender **R**-Code demonstriert die Auswertung der gruppenbezogenen Deskriptivstatistiken für diesen Datensatz.

```
# studiengruppenspezifische Deskriptivstatistiken
D       = read.csv("Multivariate_Varianzanalyse.csv")                    # Dateneinlesen
m       = 2                                                              # Datendimension von Interesse
p       = 3                                                              # Anzahl Gruppen
k       = 15                                                             # Anzahl Datenpunkte pro Gruppe
Y       = array(dim = c(m,k,p))                                          # Datenarrayinitialisierung
Y[,,1]  = rbind(D$dBDI[D$COND == "F2F"],                                 # F2F dBDI-Werte
                D$dGLU[D$COND == "F2F"])                                 # F2F dGLU-Werte
Y[,,2]  = rbind(D$dBDI[D$COND == "ONL"],                                 # ONL dBDI-Werte
                D$dGLU[D$COND == "ONL"])                                 # ONL dGLU-Werte
Y[,,3]  = rbind(D$dBDI[D$COND == "WLC"],                                 # WLC dBDI-Werte
                D$dGLU[D$COND == "WLC"])                                 # WLC dGLU-Werte
y_bar_i = array(dim = c(m,p))                                            # Stichprobenmittelarray
C_i     = array(dim = c(m,m,p))                                          # Stichprobenkovarianzmatrizenarray
j_k     = matrix(rep(1,k), nrow = k)                                     # 1_{l}
I_k     = diag(k)                                                        # Einheitsmatrix I_l
J_k     = matrix(rep(1,k^2), nrow = k)                                   # 1_{ll}
for (i in 1:p){                                                          # Gruppeniterationen
    y_bar_i[,i] = (1/k)*(Y[,,i] %*% j_k)                                 # Stichprobenmittel \bar{\ups}_i
    C_i[,,i]    = (1/(k-1))*(Y[,,i] %*% (I_k-(1/k)*J_k) %*% t(Y[,,i]))   # Stichprobenkovarianzmatrix C_i
}
```

Folgender **R**-Code visualisiert diese Deskriptivstatistiken in Abbildung 1.

```
# R-Pakete
library(ellipse)
library(latex2exp)

# Abbildungsparameter
par(
    family      = "sans",
    mfcol       = c(1,1),
    pty         = "s",
    bty         = "l",
    lwd         = 1,
    las         = 1,
    mgp         = c(2,1,0),
    xaxs        = "i",
    yaxs        = "i",
    font.main   = 1,
    cex         = 1,
    cex.main    = 1)

# Achsenbeschriftung
cols     = c("Gray20", "Gray50", "Gray80")
plot(NaN, NaN,
    xlim = c(-5,15),
    ylim = c(-1,6),
    xlab = TeX("dBDI"),
    ylab = TeX("dGLU"))

# Datenpunkte und Deskriptivstatistiken
for (i in 1:p){
    points(Y[1,,i], Y[2,,i],
        col = "White",
        bg  = cols[i],
        pch = 21)
    points(y_bar_i[1,i], y_bar_i[2,i],
        col = cols[i],
        pch = 1)
        iso = ellipse(C_i[,,i], level = .6, centre = y_bar_i[,i])
    lines(iso[,1], iso[,2],
        col = cols[i])
}

# Legende
legend("topleft", c("F2F", "ONL", "WLC"),
    pch         = c(21,21,21),
    bg          = cols,
    col         = cols,
    bty         = "n",
    cex         = 1,
    y.intersp   = 1.5)
```

```
# PDF-Speicherung
dev.copy2pdf(
    file        = "./Abbildungen/anwendungsbeispiel.pdf",
    width       = 6,
    height      = 6)
dev.off()
```

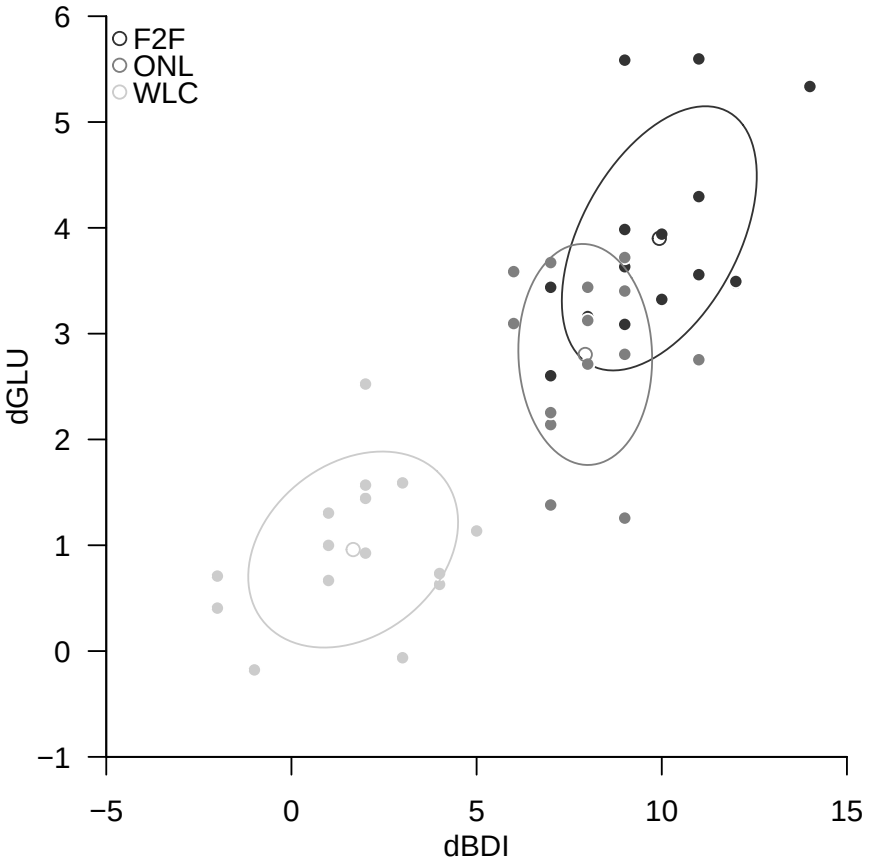

Abbildung 1. Deskriptivstatistiken der Daten des Beispieldatensatzes. Jeder Punkt visualisiert die Daten einer Patient:in.

Zur Parameterschätzung nutzen wir folgende **R**-Funktion.

```
estimate = function(Y){
    # ------------------------------------------------------------------------
    # Diese Funktion evaluiert die Parameterschätzer einer einfaktoriellen
    # multivariaten Varianzanalyse basierend auf einen m x k x p Datensatz Y.
    #
    # Input
    #     Y          : m x k x p Datenarray
    #
    # Output
    #     $mu_hat    : m x p \mu_i Parameterschätzer
    #     $Sigma_hat : m x m \Sigma Parametschätzer
    # ------------------------------------------------------------------------

    # Dimensionsparameter
    d         = dim(Y)                                  # Datensatzdimensionen
    m         = d[1]                                    # Datendimension
    k         = d[2]                                    # Anzahl Datenpunkte pro Gruppe
    p         = d[3]                                    # Anzahl Gruppen

    # Erwartungswertparameterschätzer
    mu_hat_i  = matrix(apply(Y,3,rowMeans), nrow = m)

    # Kovarianzmatrixparameterschätzer
    Sigma_hat = matrix(rep(0,m*m), nrow = m)
    for(i in 1:p){
        for(j in 1:k){
            Sigma_hat = Sigma_hat + (1/(k*p-p))*(Y[,j,i] - mu_hat_i[,i]) %*% t(Y[,j,i] - mu_hat_i[,i])
        }
    }

    # Outputspezifikation
    return(list(mu_hat_i = mu_hat_i, Sigma_hat = Sigma_hat))
}
```

Zur multivariaten Quadratsummenzerlegung nutzen wir folgende **R**-Funktion.

```
sos = function(Y){
    # ------------------------------------------------------------------------
    # Diese Funktion evaluiert die multivariaten Quadratsummen T,B,W einer
    # einfaktoriellen Varianzanalyse basierend auf einen m x k x p Datensatz Y.
    #
    # Input
    #     Y        : m x k x p Datenarray
    #
    # Output
    #     $y_bar   : m x 1 Gesamtmittelwert
    #     $y_bar_i : m x p Gruppenmittelwerte
    #     $T       : m x m total         sum-of-squares Matrix
    #     $B       : m x m between-group sum-of-squares Matrix
    #     $W       : m x m within-group  sum-of-squares Matrix
    # ------------------------------------------------------------------------

    # Dimensionen
    d        = dim(Y)                                  # Datensatzdimensionen
    m        = d[1]                                    # Datendimension
    k        = d[2]                                    # Anzahl Datenpunkte pro Gruppe
    p        = d[3]                                    # Anzahl Gruppen

    # Mittelwerte
    y_bar_i  = matrix(apply(Y,3,rowMeans), nrow = m)   # Gruppenstichprobenmittel
    y_bar    = matrix(rowMeans(y_bar_i),   nrow = m)   # Gesamtstichprobenmittel

    # total sum-of-squares Matrix
    T = matrix(rep(0,m*m), nrow = m)
    for(i in 1:p){
        for(j in 1:k){
            T = T + (Y[,j,i] - y_bar) %*% t(Y[,j,i] - y_bar)
        }
    }

    # between sum-of-squares Matrix
    B = matrix(rep(0,m*m), nrow = m)
    for(i in 1:p){
        B = B + k*(y_bar_i[,i] - y_bar) %*% t(y_bar_i[,i] - y_bar)
    }
```

```
    # within sum-of-squares Matrix
    W = matrix(rep(0,m*m), nrow = m)
    for(i in 1:p){
        for(j in 1:k){
            W = W + (Y[,j,i] - y_bar_i[,i]) %*% t(Y[,j,i] - y_bar_i[,i])
        }
    }

    # Outputspezifikation
    return(list(y_bar_i = y_bar_i, y_bar = y_bar, T = T, B = B, W = W))
}
```

Zur Durchführung der einfaktoriellen multivariaten Varianzanalyse schließlich nutzen wir folgenden, auf der Funktion `sos` ("sums of squares") aufbauenden **R**-Code.

```
# Einlesen und Präprozessierung des Datensatzes
D           = read.csv("Multivariate_Varianzanalyse.csv")                  # Dateneinlesen
m           = 2                                                            # Datendimension von Interesse
p           = 3                                                            # Anzahl Gruppen
k           = 15                                                           # Anzahl Datenpunkte pro Gruppe
n           = p*k                                                          # Gesamtanzahl Datenpunkte
Y           = array(dim = c(m,k,p))                                        # Datenarrayinitialisierung
Y[,,1]      = rbind(D$dBDI[D$COND == "F2F"],                               # F2F dBDI-Werte
                    D$dGLU[D$COND == "F2F"])                               # F2F dGLU-Werte
Y[,,2]      = rbind(D$dBDI[D$COND == "ONL"],                               # ONL dBDI-Werte
                    D$dGLU[D$COND == "ONL"])                               # ONL dGLU-Werte
Y[,,3]      = rbind(D$dBDI[D$COND == "WLC"],                               # WLC dBDI-Werte
                    D$dGLU[D$COND == "WLC"])                               # WLC dGLU-Werte

# Einfaktorielle multivariate Varianzanalyse
alpha_0     = 0.05                                                         # Signifikanzlevel
S           = sos(Y)                                                       # sum-of-squares Matrizen
L           = det(S$W)/det(S$W + S$B)                                      # Wilks' Lambda
w           = n-1-(1/2)*(m+p)                                              # w
t           = sqrt((m^2*(p-1)^2-4)/(m^2+(p-1)^2-5))                        # t
nu_1        = m*(p-1)                                                      # \nu_1
nu_2        = w*t-(1/2)*(m*(p-1)-2)                                        # \nu_2
tau         = ((1-L^(1/t))/L^(1/t))*(nu_2/nu_1)                            # Teststatistik
k_alpha_0   = qf(1-alpha_0,nu_1,nu_2)                                      # kritischer Wert
phi         = as.numeric(tau > k_alpha_0)                                  # Nullhypothesentest
P           = 1-pf(tau,nu_1,nu_2)                                          # Überschreitungswahrscheinlichkeit

# Ausgabe
cat(  "Wilks' Lambda      : ", L,
    "\ntau                : ", tau,
    "\nnu_1               : ", nu_1,
    "\nnu_2               : ", nu_2,
    "\nphi                : ", phi,
    "\nP(tau > tau_tilde) : ", P, "\n")
```

```
Wilks' Lambda      :  0.1569049
tau                :  31.25301
nu_1               :  4
nu_2               :  82
phi                :  1
P(tau > tau_tilde) :  8.881784e-16
```

Im vorliegenden Fall wird die Nullhypothese identischer Gruppenerwartungswertparameter also verworfen. Schließlich validieren wir obige Analyse im Sinne eines Black-Box-Verfahrens mithilfe der **R**-Funktionen `lm()` und `Manova()`.

```
library(car)
D        = read.csv("Multivariate_Varianzanalyse.csv")                     # Dateneinlesen
model    = lm(cbind(D$dBDI,D$dGLU) ~ D$COND, D)                            # Modellspezifikation
Manova(model, test.statistic = "Wilks")                                    # multivariate Varianzanalyse
```

```
Type II MANOVA Tests: Wilks test statistic
       Df test stat approx F num Df den Df    Pr(>F)
D$COND  2    0.1569   31.253      4     82 8.346e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```