Vertiefende statistische Verfahren

4. Übungsblatt SS 2024

Stefan Kolb, Joachim Waltl

Allgemeine Information

Alle Aufgaben sind mit R zu lössen, wenn nicht explizit anders angegeben. Die Berechnungen sollen nachvollziehbar und dokumentiert sein. Um die vollständige Punktezahl zu erreichen, müssen alle Ergebnisse und Fragen entsprechend interpretiert bzw. beantwortet werden. Code alleine ist nicht ausreichend! Die Abgabe erfolgt über Moodle entsprechend der Abgaberichtlinien als pdf und Rmd File. Bitte inkludieren Sie namentlich alle beteiligten Gruppenmitglieder sowohl im Bericht als auch im Source Code. Die jeweiligen Datensätze die für diese Übung relevant sind finden Sie ebenfalls in Moodle.

Datensatz: Reaven and Miller Diabetes Daten

Verwenden Sie den Datensatz `diabetes_RM.csv`. Der Datensatz enthält fünf Messungen, die an 145 nicht adipösen erwachsenen Patienten durchgeführt wurden, die in drei Gruppen eingeteilt wurden.

Die drei primären Variablen sind die Glukoseintoleranz, die Insulinantwort auf orale Glukose und die Insulinresistenz (gemessen durch die Steady-State-Plasmaglukose, die nach chemischer Suppression der endogenen Insulinsekretion bestimmt wird). Zwei zusätzliche Variablen, das relative Gewicht und die Nüchternplasmaglukose, sind ebenfalls enthalten. Zusammengefasst ergeben sich folgende Prädiktorvariablen:

- `rw`: relatives Gewicht, Verhältnis zwischen aktuellem Gewicht und zu erwartendem Gewicht bei der Körpergröße
- `fpg`: Nüchternglukoselevel im Plasma in mg/dl
- `glucose`: Fläche unter Glukose-Antwort (mg/dl*h) nach 3h oralem Glukosetoleranztest (OGTT)
- `insulin`: Fläche unter der Insulin-Antwort (mg/dl*h) nach OGTT
- `sspg`: Steady-State-Plasmaglukose (mg/dl) als Maß für die Insulinresistenz

```
# Variable Beschreibungen
descriptions <- get_descriptions()
```

Reaven und Miller [ref] wendeten in Anlehnung an Friedman und Rubin (1967) eine Clusteranalyse auf die drei primären Variablen an und identifizierten drei Cluster: “normal”, “chemical” und “overt” diabetsiche Probanden. Die Variable `group` enthält die Klassifizierungen der Probanden in diese drei Gruppen.

1 Diskriminanzanalyse [5P]

Führen Sie eine Diskriminanzanalyse unter Berücksichtigung folgender Punkte durch:

```r
# Datensatz laden
diabetes <- read.csv("diabetes_RM.csv")

# Übersicht
str(diabetes)
```

```
## 'data.frame':    145 obs. of  6 variables:
##  $ rw     : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ fpg    : int  80 97 105 90 90 86 100 85 97 97 ...
##  $ glucose: int  356 289 319 356 323 381 350 301 379 296 ...
##  $ insulin: int  124 117 143 199 240 157 221 186 142 131 ...
##  $ sspg   : int  55 76 105 108 143 165 119 105 98 94 ...
##  $ group  : chr  "normal" "normal" "normal" "normal" ...
```

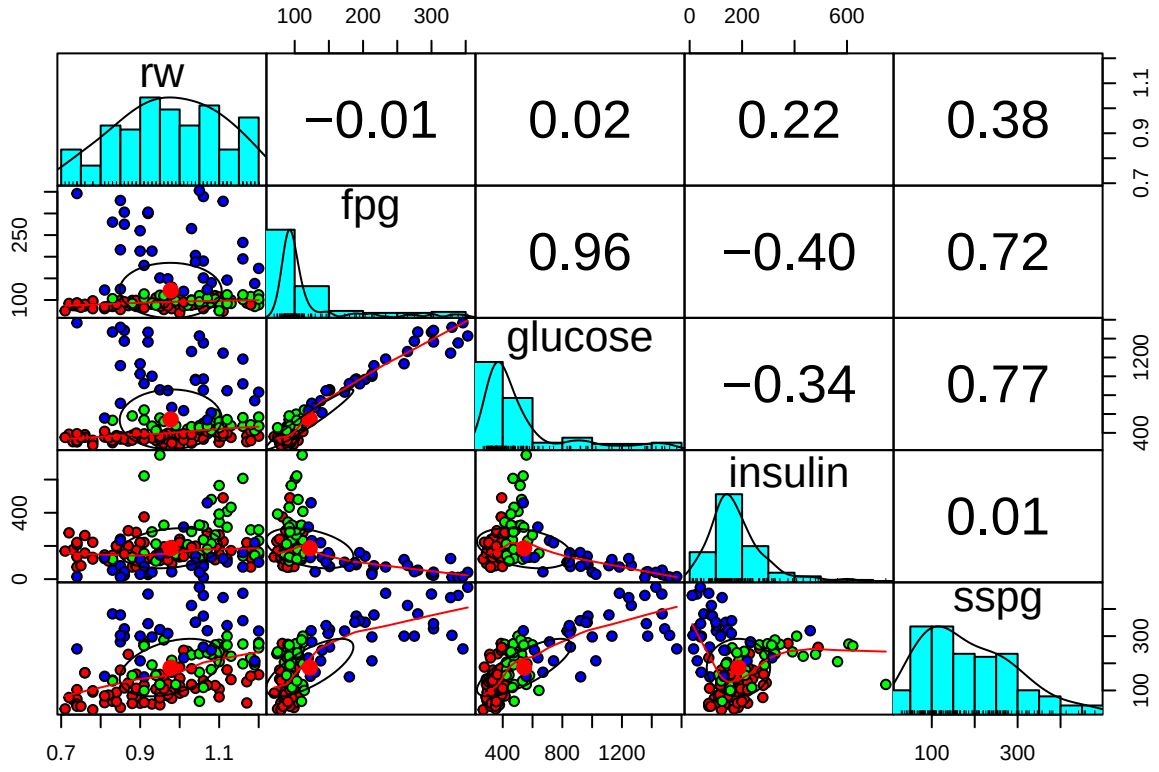
```r
head(diabetes)
```

```
##     rw fpg glucose insulin sspg  group
## 1 0.81  80     356     124   55 normal
## 2 0.95  97     289     117   76 normal
## 3 0.94 105     319     143  105 normal
## 4 1.04  90     356     199  108 normal
## 5 1.00  90     323     240  143 normal
## 6 0.76  86     381     157  165 normal
```

```r
# Outcomevariable als Faktor definieren
diabetes$group <- as.factor(diabetes$group)

# Zusammenfassung
summary(diabetes)
```

```
##        rw              fpg         glucose          insulin
##  Min.   :0.7100   Min.   : 70   Min.   : 269.0   Min.   : 10.0
##  1st Qu.:0.8800   1st Qu.: 90   1st Qu.: 352.0   1st Qu.:118.0
##  Median :0.9800   Median : 97   Median : 413.0   Median :156.0
##  Mean   :0.9773   Mean   :122   Mean   : 543.6   Mean   :186.1
##  3rd Qu.:1.0800   3rd Qu.:112   3rd Qu.: 558.0   3rd Qu.:221.0
##  Max.   :1.2000   Max.   :353   Max.   :1568.0   Max.   :748.0
##       sspg            group
##  Min.   : 29.0   chemical:36
##  1st Qu.:100.0   normal  :76
##  Median :159.0   overt   :33
##  Mean   :184.2
##  3rd Qu.:257.0
##  Max.   :480.0
```

```r
pairs.panels(diabetes[1:5],
             gap = 0,
             bg = c("green","red","blue")[diabetes$group],
             pch = 21)
```



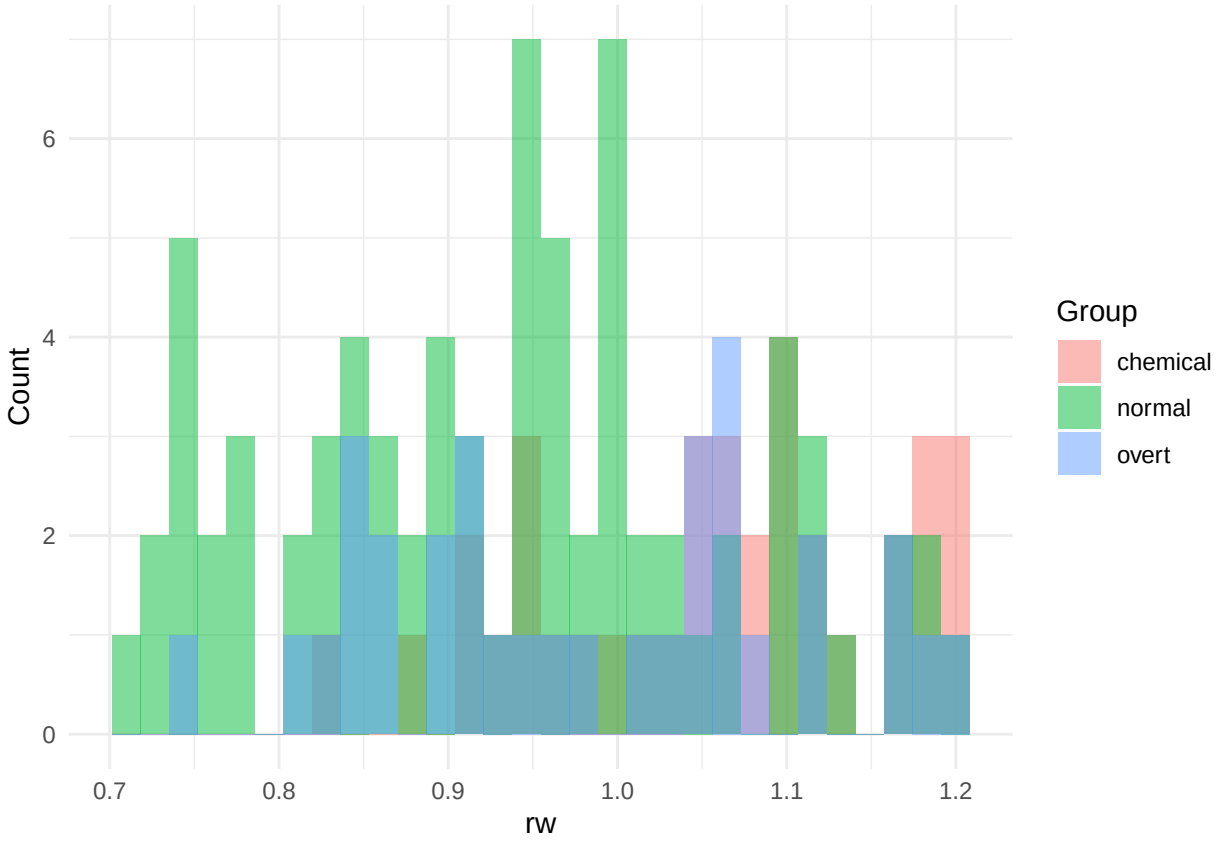
i) Explorative Analyse der Prädiktoren mit Hilfe von Histogrammen. Gibt es Prädiktoren, die bereits eine gute Trennung zwischen den Klassen erlauben?

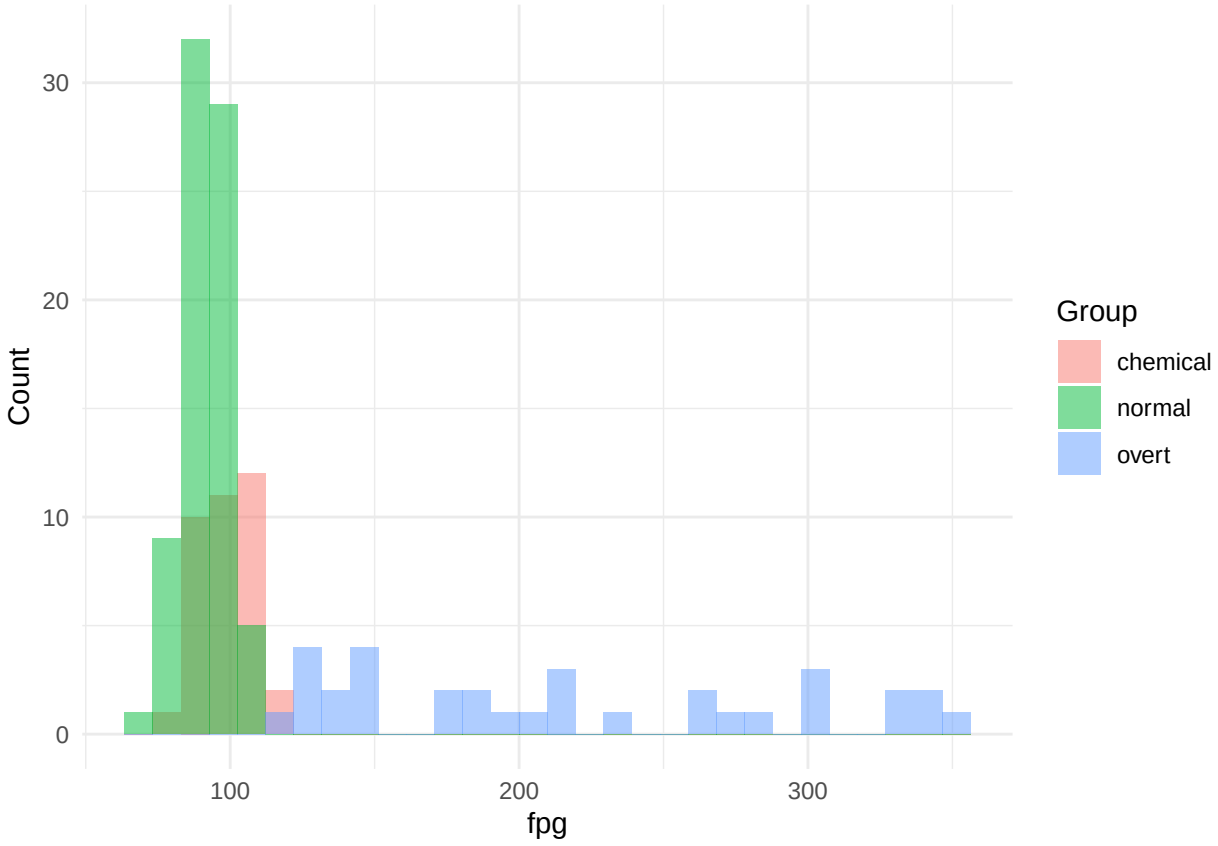
```
# Explorative Analyse

# Mittelwerte aller Gruppe für numerische Prädiktoren berechnen

# Tabelle
aggregate(cbind(glucose,insulin,fpg,sspg,rw)~group,diabetes,mean)
```

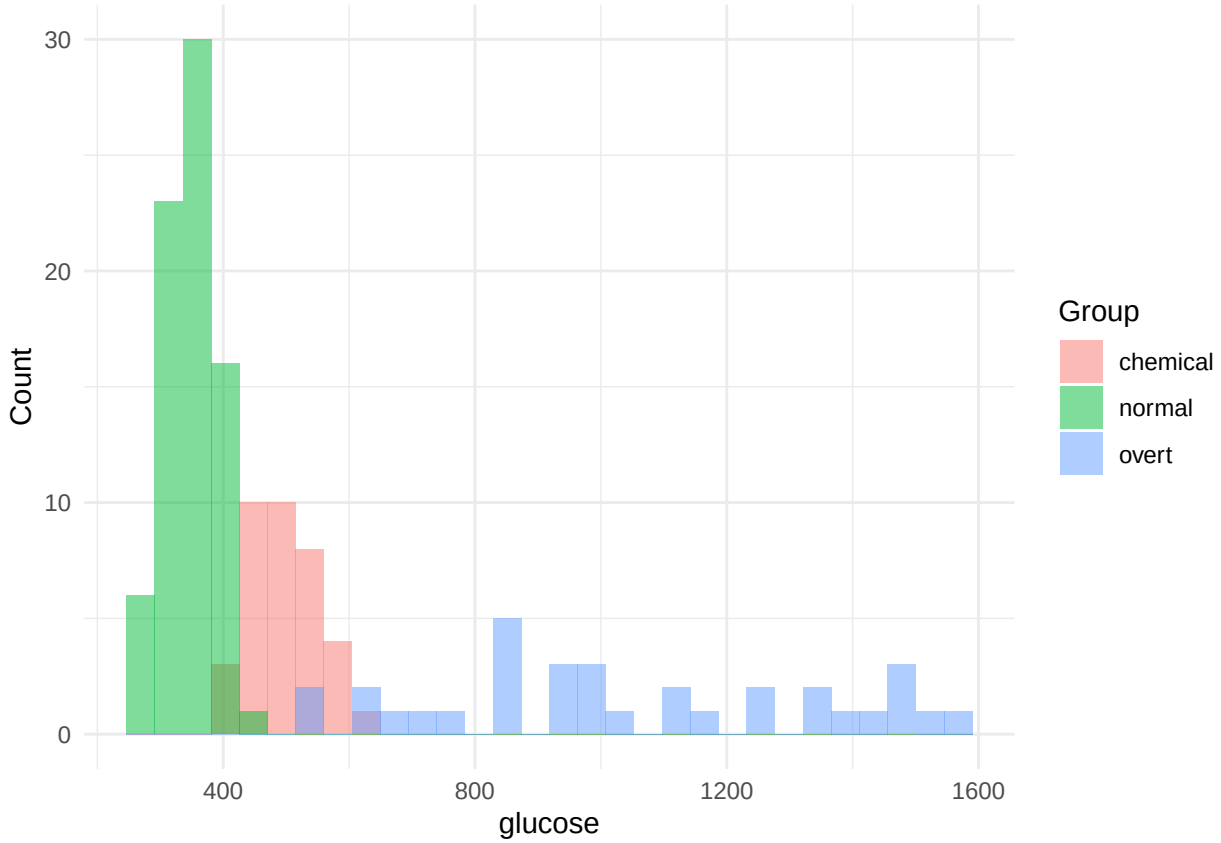
```
##      group   glucose  insulin       fpg     sspg        rw
## 1 chemical  493.9444 288.0000  99.30556 208.9722 1.0558333
## 2   normal  349.9737 172.6447  91.18421 114.0000 0.9372368
## 3    overt 1043.7576 106.0000 217.66667 318.8788 0.9839394
```

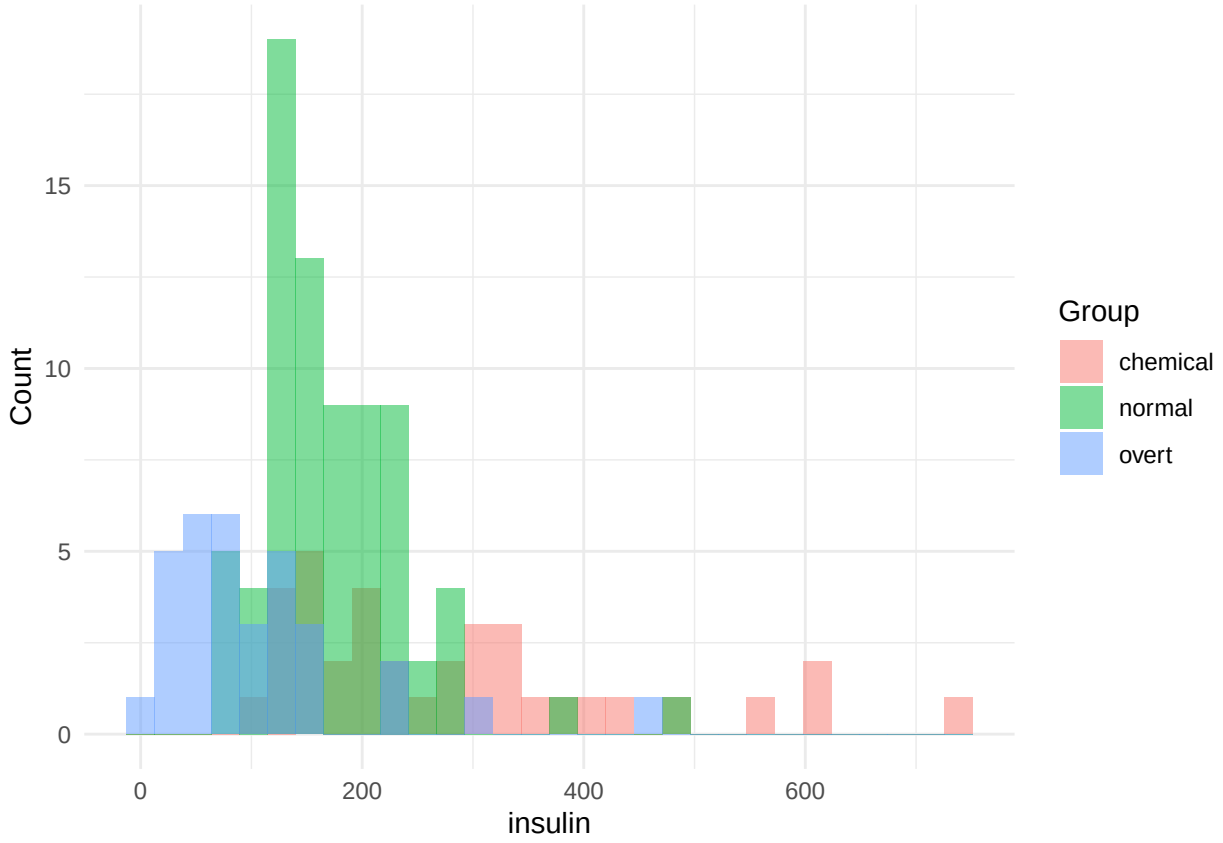
```
# Histogramme
create_histograms(diabetes)
```

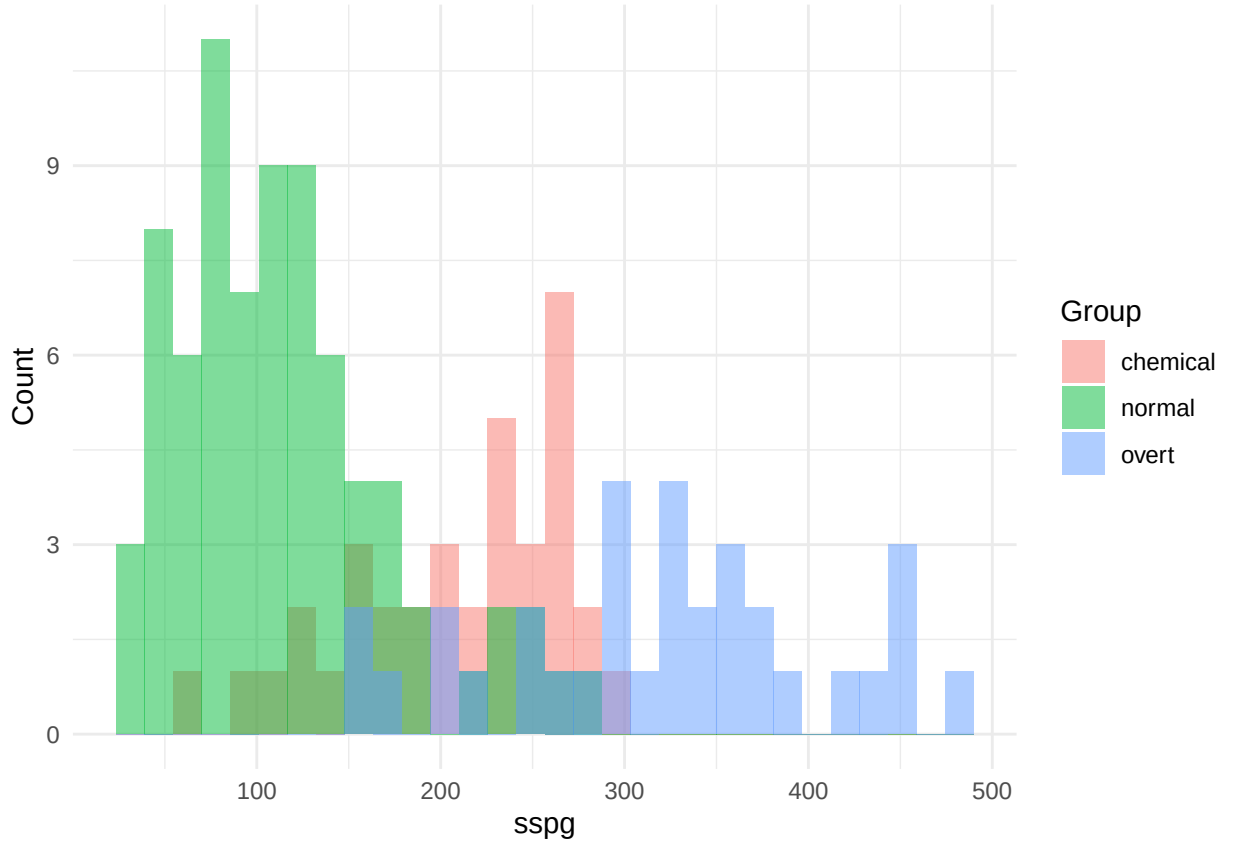
```
## Warning: ‘aes_string()‘ was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with ‘aes()‘.
## i See also ‘vignette("ggplot2-in-packages")‘ for more information.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```


Count
6
4
2
0
0.7
0.8
0.9
1.0
1.1
1.2
rw
Group
chemical
normal
overt


30
20
10
0
Count
100
200
300
fpg
Group
chemical
normal
overt

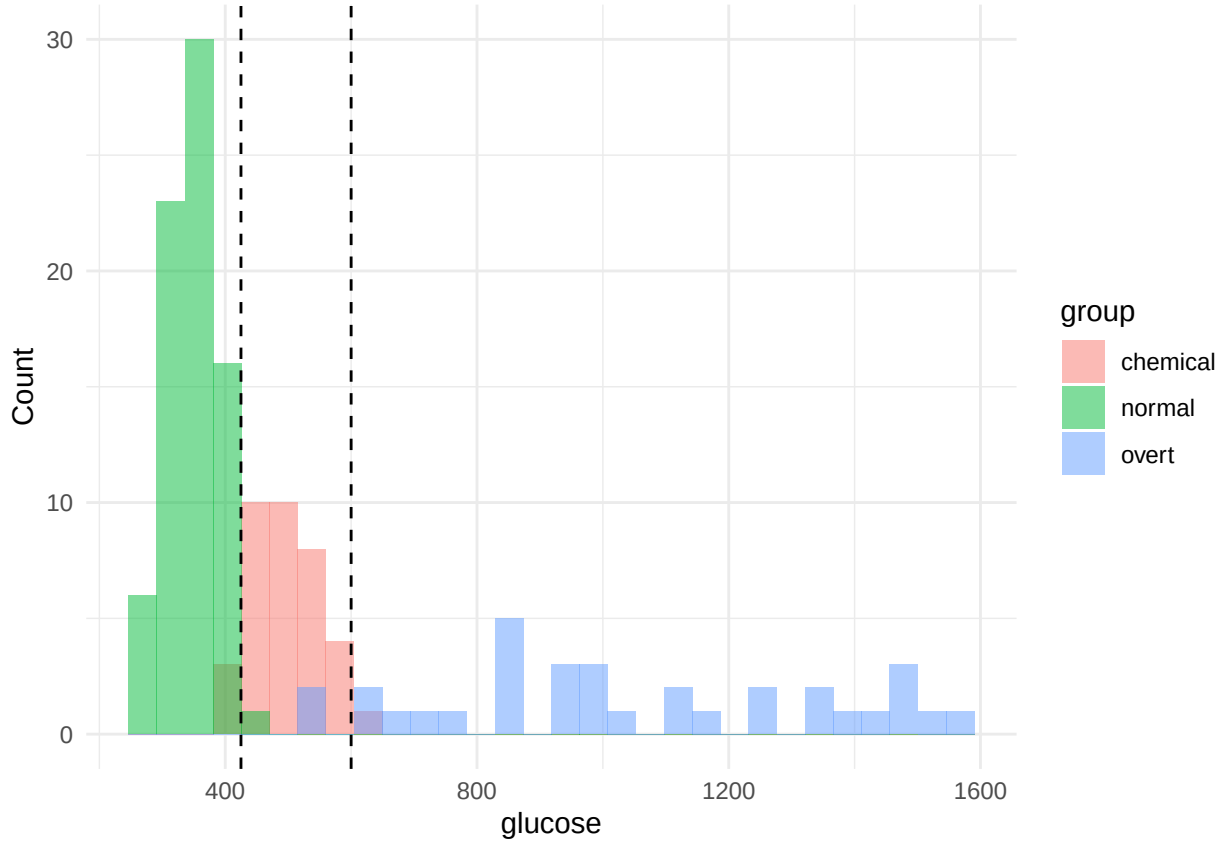

30
20
10
0
Count
400
800
1200
1600
glucose
Group
chemical
normal
overt


Count
15
10
5
0
0
200
400
600
insulin
Group
chemical
normal
overt



Anhand der Histogramme lässt sich kein Prädiktor identifizieren, der eine gute Trennung zwischen den Klassen erlaubt. Am ehesten ist beim Prädiktor ‘glucose’ eine Abgrenzung der Gruppen erkennbar. Zur explorativen Trennung der Gruppen wird daher der Prädiktor ‘glucose’ mit manuell festgelegten Schwellwerten verwendet.

```
# Explorativer Ansatz zur Trennung der Gruppen
# Grenzen anhand des Prädiktors 'glucose'
grenzen <- c(425,600)

ggplot(diabetes, aes(x = glucose, fill = group)) +
    geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
    labs(x = "glucose", y = "Count", fill = "group") +
    # Grenze einfügen
    geom_vline(xintercept = grenzen, linetype = "dashed", color = "black") +
    theme_minimal()
```




```
# Klassifizierung der Gruppen anhand der Grenzen
diabetes$group_K1<- ifelse(diabetes$glucose < grenzen[1], "normal",
                           ifelse(diabetes$glucose < grenzen[2], "chemical",
                                  "overt"))

# Genauigkeit der Klassifizierung
mean(diabetes$group_K1 == diabetes$group)
```

```
## [1] 0.9586207
```

Durch die Klassifizierung anhand des Prädiktors 'glucose' konnte eine Genauigkeit von knapp 96% erreicht werden. Dies deutet darauf hin, dass der Prädiktor 'glucose' tatsächlich bereits eine recht gute Trennung der Gruppen ermöglicht. Im nächsten Schritt wird eine Diskriminanzanalyse (LDA) durchgeführt, um hoffentlich eine noch akuratere Klassifizierung der Gruppen zu erreichen.

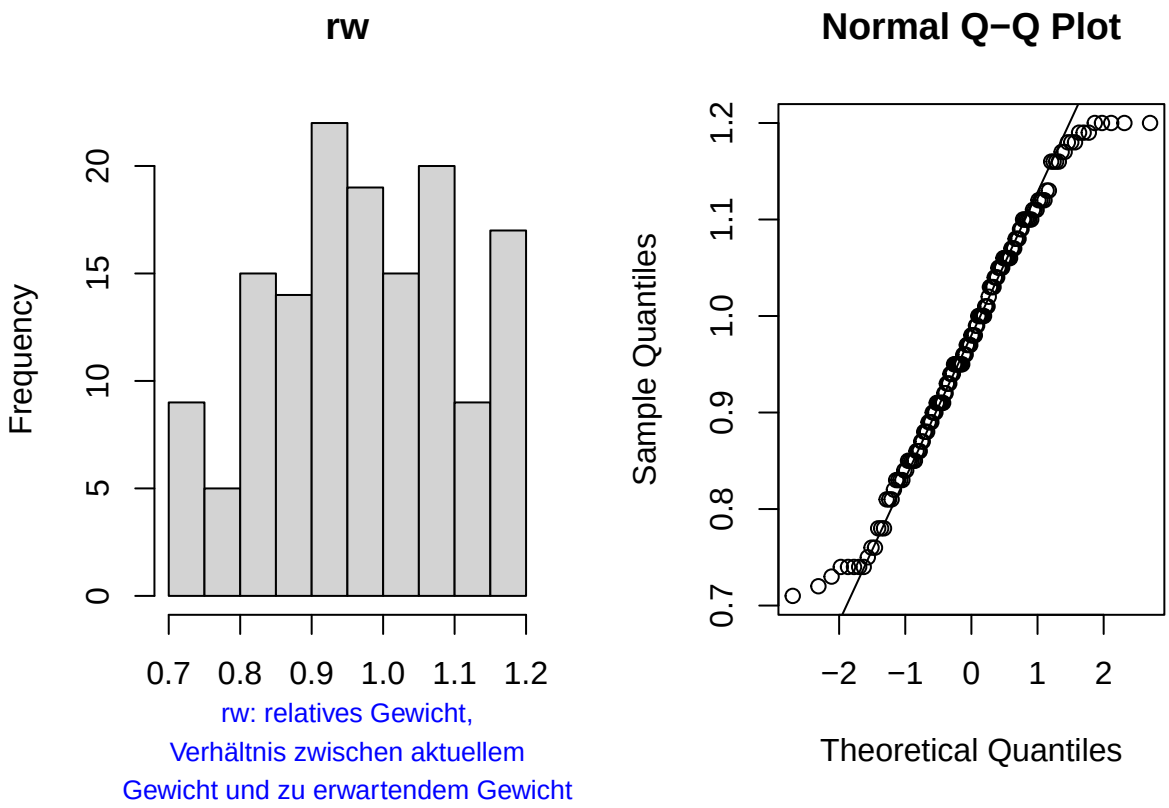
ii) Überprüfen Sie ob die Vorraussetzungen für eine LDA gegeben sind und führen Sie eine Standardisierung der Daten durch.

```
# Überprüfen der Voraussetzungen

# Normalverteilung der Prädiktoren (Visuell u. Konservativ)

print("p-Werte des Shapiro-Wilk-Tests: ")
```

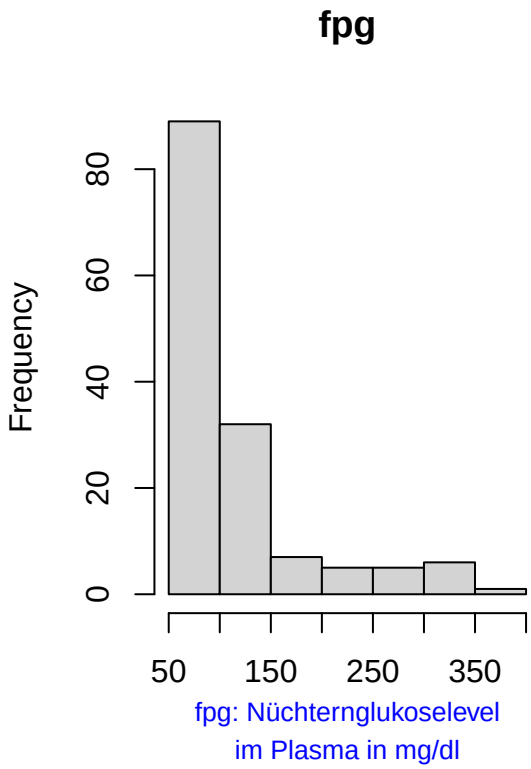
```
## [1] "p-Werte des Shapiro-Wilk-Tests: "
```

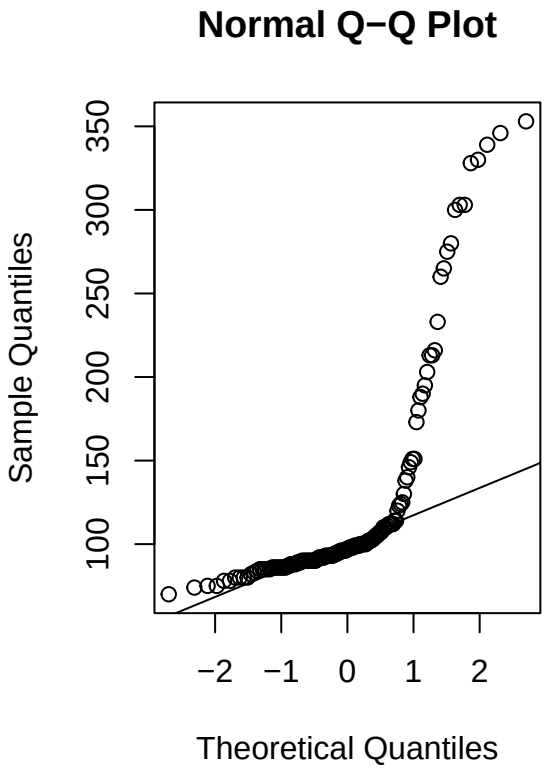
```r
for (i in 1:5) {

  #Shapiro-Wilk-Test
  sw_p <- shapiro.test(diabetes[,i])$p.value
  print(paste("p-Wert für", colnames(diabetes)[i], ":", sw_p))


  par(mfrow = c(1,2))
  # Histogramme
  hist(diabetes[,i], main = colnames(diabetes)[i], xlab = "")
   for (j in 1:length(descriptions[[i]])) {
    mtext(descriptions[[i]][j], side = 1, line = 1 + j, cex = 0.8, col =   "blue")
  }
  # QQ-Plots
  qqnorm(diabetes[,i])
  qqline(diabetes[,i])

}
```

```
## [1] "p-Wert für rw : 0.00852897444259993"
```

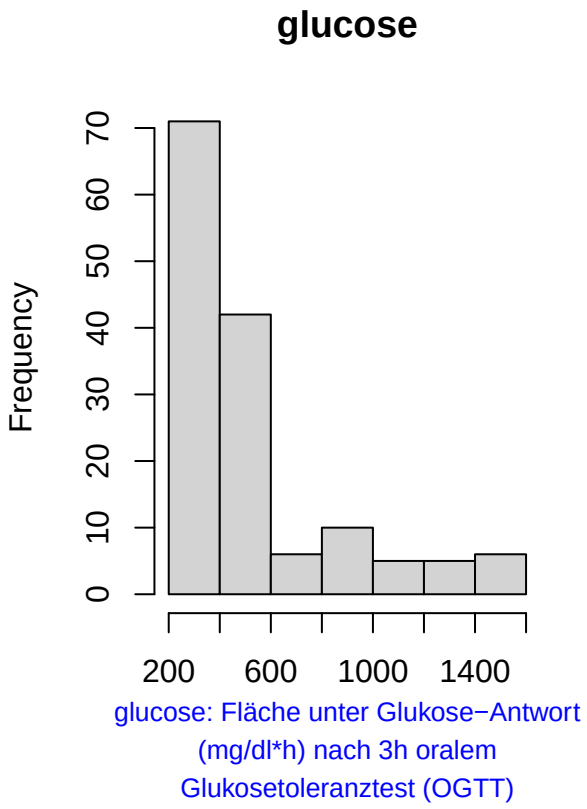


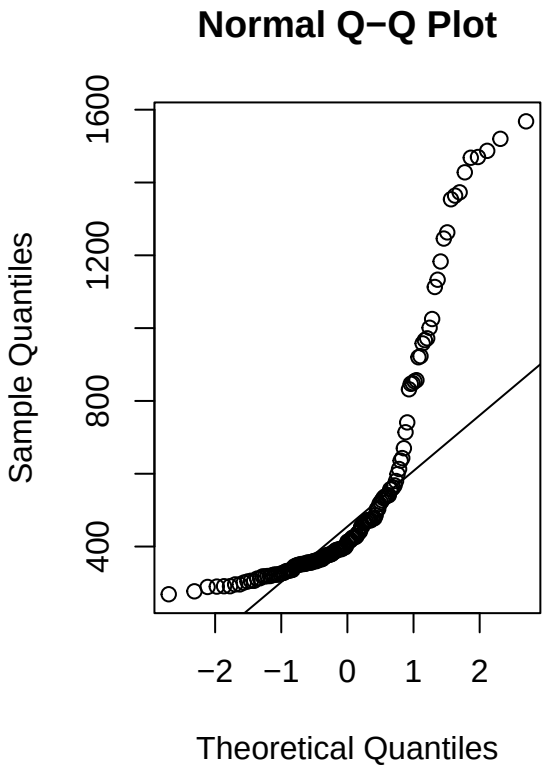

```
## [1] "p-Wert für fpg : 1.2632736568106e-17"
```



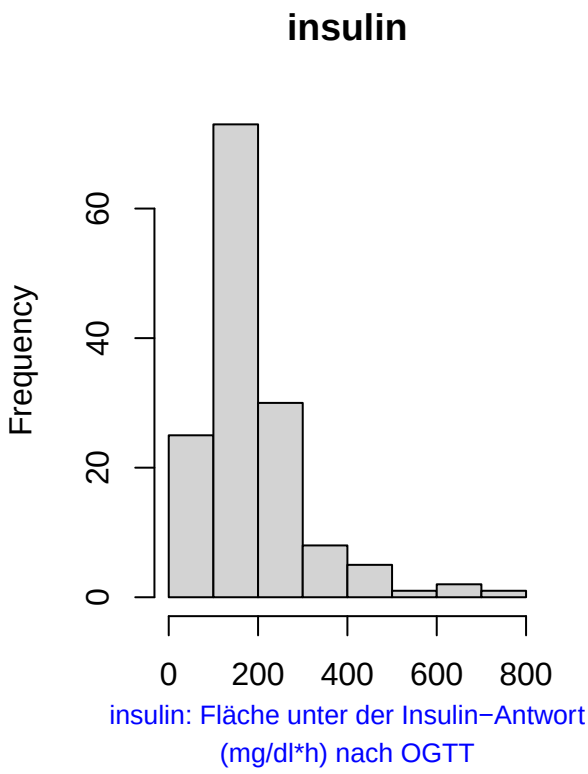


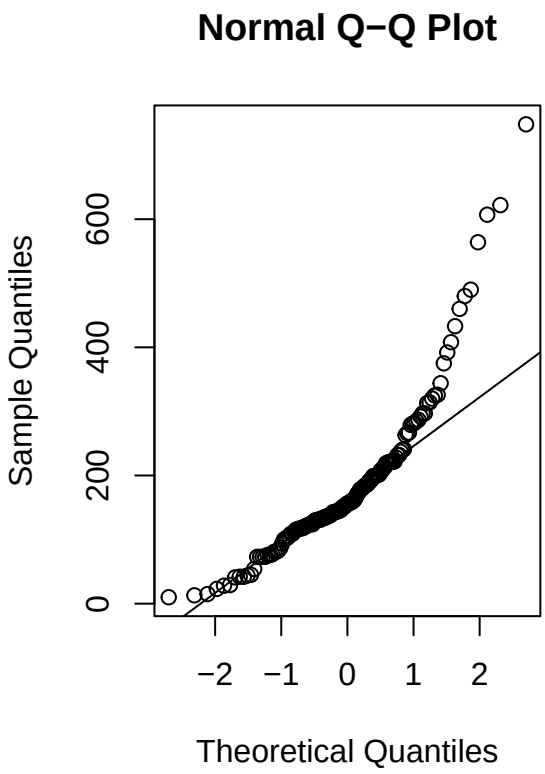

```
## [1] "p-Wert für glucose : 3.15570068501942e-15"
```



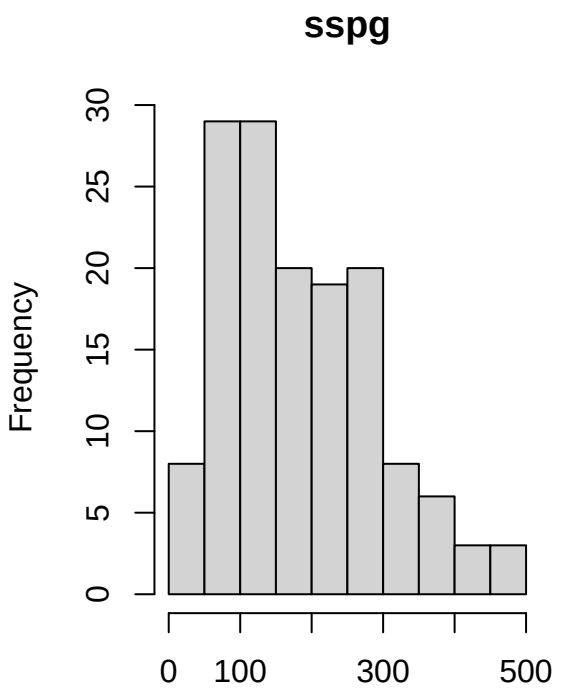



```
## [1] "p-Wert für insulin : 9.70364408387467e-11"
```

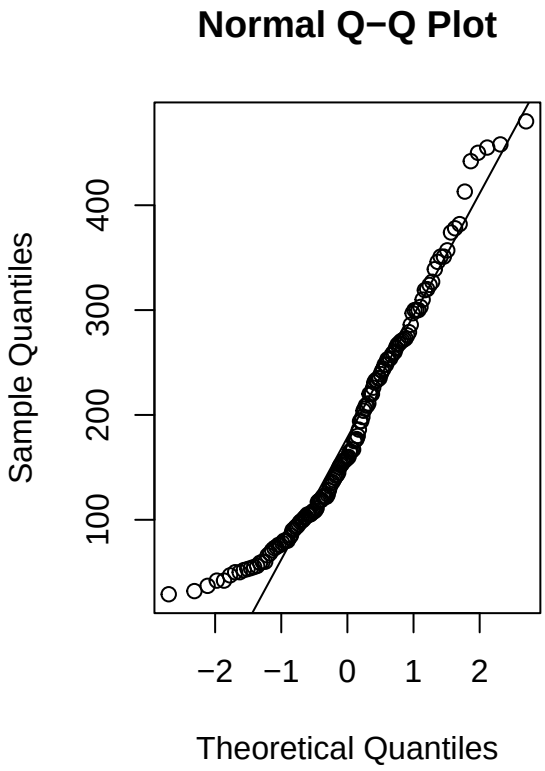





```
## [1] "p-Wert für sspg : 1.24618944438909e-05"
```



sspg: Steady−State−Plasmaglukose (mg/dl) als Maß für die Insulinresistenz



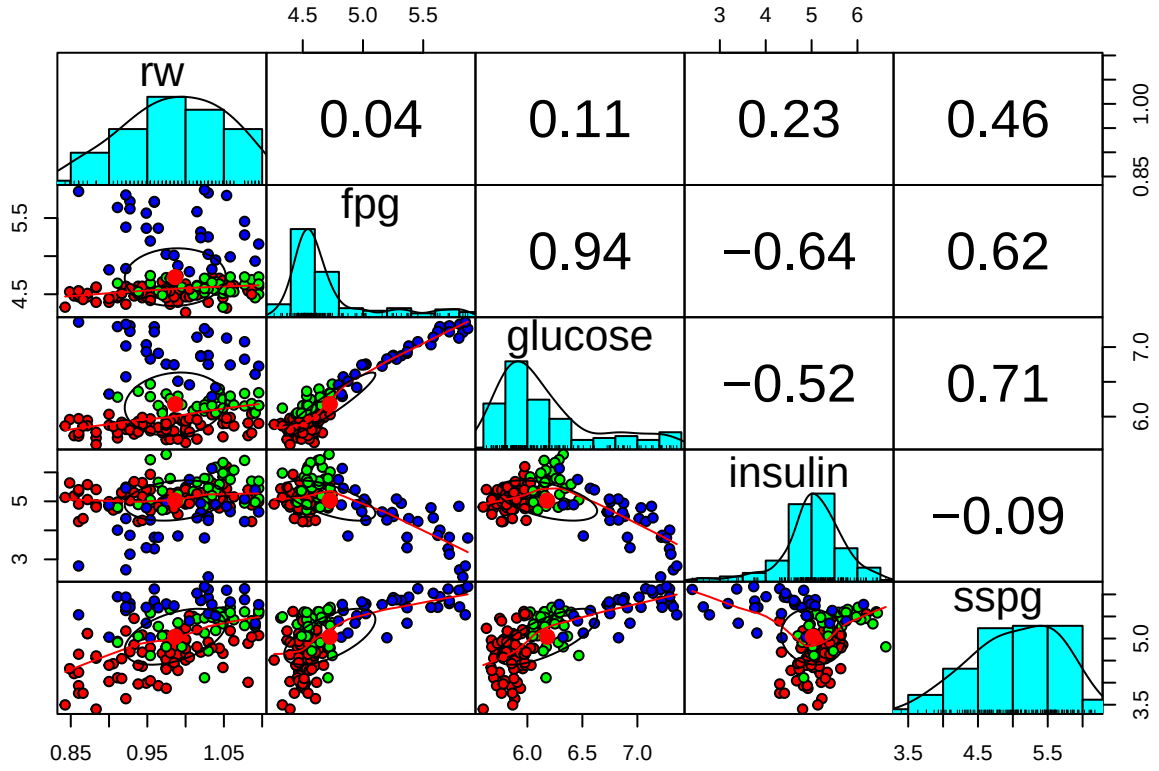

```
# Überprüfung der Multivariaten Normalverteilung
mvn_test <- MVN::mvn(diabetes[, sapply(diabetes, is.numeric)], mvnTest = "hz")
print(mvn_test)
```

```
## $multivariateNormality
##            Test       HZ p value MVN
## 1 Henze-Zirkler 3.272543       0  NO
##
## $univariateNormality
##               Test  Variable Statistic   p value Normality
## 1 Anderson-Darling    rw        0.6311  0.0982      YES
## 2 Anderson-Darling    fpg      21.8353  <0.001      NO
## 3 Anderson-Darling  glucose    15.4117  <0.001      NO
## 4 Anderson-Darling  insulin     5.4330  <0.001      NO
## 5 Anderson-Darling   sspg       2.2421  <0.001      NO
##
## $Descriptives
##           n        Mean     Std.Dev Median    Min    Max   25th   75th
## rw      145   0.9773103   0.1292352   0.98   0.71    1.2   0.88   1.08
## fpg     145 121.9862069  63.9304077  97.00  70.00  353.0  90.00 112.00
## glucose 145 543.6137931 316.9508634 413.00 269.00 1568.0 352.00 558.00
## insulin 145 186.1172414 120.9351584 156.00  10.00  748.0 118.00 221.00
## sspg    145 184.2068966 106.0298632 159.00  29.00  480.0 100.00 257.00
##               Skew   Kurtosis
## rw      -0.1161301 -0.8665719
```

```
## fpg       2.2287483  4.0105224
## glucose   1.7796189  2.1620152
## insulin   1.7960220  4.4484613
## sspg      0.6884820 -0.2335819
```

In den Histogrammen und QQ-Plots ist zu erkennen, dass die Prädiktoren 'glucose', 'fpg' und 'insulin' deutliche Abweichungen von der Normalverteilung aufweisen. Ihre Verteilungen zeigen eine starke Rechtsschiefe und auch in den QQ-Plots sind deutliche Abweichungen von der Normalverteilung ersichtlich, insbesondere in den oberen Quantilen. Auch der Multivariate Normalverteilungstest zeigt, dass die Daten nicht multivariat normalverteilt sind. Die Prädiktoren 'sspg' und 'rw' sind hingegen annähernd normalverteilt.

```
# Transformieren der Daten
diabetes_trans <- diabetes %>%
  mutate(glucose  = log(glucose+1),
         fpg = log(fpg+1),
         sspg  = log(sspg + 1),
         insulin = log(insulin + 1),
         rw = sqrt(rw),
         group = as.factor(group))

# Überprüfung der Transformation
pairs.panels(diabetes_trans[1:5],
             gap = 0,
             bg = c("green","red","blue")[diabetes_trans$group],
             pch = 21)
```



Selbst nach der Transformation der Daten sind die Prädiktoren 'glucose', 'fpg' und 'insulin' noch immer nicht normalverteilt. Die Diskriminanzanalyse wird dennoch durchgeführt, da sie robust gegenüber Verletzungen der Normalverteilung ist.

```r
# Korrelationen
cor_matrix <- cor(diabetes[,1:5])
print(cor_matrix)
```

```
##                   rw          fpg    glucose      insulin        sspg
## rw       1.000000000 -0.008813193  0.0239843  0.222237813 0.384319804
## fpg     -0.008813193  1.000000000  0.9646281 -0.396234858 0.715480192
## glucose  0.023984304  0.964628091  1.0000000 -0.337020435 0.770942459
## insulin  0.222237813 -0.396234858 -0.3370204  1.000000000 0.007914263
## sspg     0.384319804  0.715480192  0.7709425  0.007914263 1.000000000
```

Die Korrelationsmatrix zeigt, dass der Prädiktor 'glucose' eine starke positive Korrelation mit den Prädiktoren 'fpg'(0.96) und 'sspg'(0.77) aufweist.

```r
# Kovarianzen
cov_matrix <- cov(diabetes[,1:5])
print(cov_matrix)
```

```
##                   rw           fpg       glucose       insulin         sspg
## rw       0.01670174 -7.281513e-02  9.824262e-01      3.473373     5.266255
## fpg     -0.07281513  4.087097e+03  1.954606e+04  -3063.463649  4849.905651
## glucose  0.98242625  1.954606e+04  1.004578e+05 -12918.162739 25908.490182
## insulin  3.47337308 -3.063464e+03 -1.291816e+04  14625.312548   101.482519
## sspg     5.26625479  4.849906e+03  2.590849e+04    101.482519 11242.331897
```

```r
# Überprüfung der Homogenität der Kovarianzen
boxM_test <- boxM(diabetes[, sapply(diabetes, is.numeric)], diabetes$group)
print(boxM_test)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  diabetes[, sapply(diabetes, is.numeric)]
## Chi-Sq (approx.) = 396.8, df = 30, p-value < 2.2e-16
```

Der Box-M-Test zeigt, dass die Kovarianzmatrizen der Gruppen nicht gleich sind. Dies bedeutet, dass die Annahme der Homogenität der Kovarianzmatrizen verletzt ist. Streng genommen sind die Voraussetzungen für eine LDA nicht gegeben. Da die LDA in Klassifizierungsaufgaben jedoch robust gegenüber Verletzungen der Normalverteilung und Homogenität der Kovarianzen ist, wird die LDA dennoch durchgeführt.

Um eine bessere Vergleichbarkeit der Prädiktoren zu gewährleisten, werden die Daten standardisiert.

```r
# Standardisierung der Daten (mittels preProcess-Funktion)
diabetes.prePro <- preProcess(diabetes[, sapply(diabetes, is.numeric)], method = c("center", "scale"))
# Anwenden des Preprocessers
diabetes.pre <- predict(diabetes.prePro, diabetes[, sapply(diabetes, is.numeric)])
# Hinzufügen der Gruppe
diabetes.pre$group <- as.factor(diabetes$group)
```

```r
# Alternative: Standardisieren in data.frame (mittels Scale-Funktion)(Transformierte Daten)
diabetes_trans.pre1<-data.frame(rw=scale(diabetes_trans$rw),
                          fpg=scale(diabetes_trans$fpg),
                          glucose=scale(diabetes_trans$glucose),
                          insulin=scale(diabetes_trans$insulin),
                          sspg=scale(diabetes_trans$sspg),
                          group=diabetes_trans$group)
```

iii) Unterteilen sie die gesamten Daten in Trainings- und Test-Daten und führen Sie in der weiteren Folge eine Klassifizierung mit einer LDA durch. Evaluieren Sie die Perfomance der Klassifizierung und stellen Sie die Ergebnisse graphisch dar (Darstellung der Projektionen, Partition Plot).

```r
# Splitten der Daten in Trainings- und Testdaten (80/20)
set.seed(42) # Für Reproduzierbarkeit

trainIndex <- sample(1:145, 0.8 * 145, replace = FALSE)
trainData <- diabetes.pre[ trainIndex,]
testData  <- diabetes.pre[-trainIndex,]

# Splitten der transformierten Daten
set.seed(42)
trainIndex <- sample(1:145, 0.8 * 145, replace = FALSE)
trainData_t <- diabetes_trans.pre1[ trainIndex,]
testData_t <- diabetes_trans.pre1[-trainIndex,]
```

Nachdem die Daten in Trainings- und Testdaten aufgeteilt wurden, wird die LDA durchgeführt.

```r
# LDA (Daten ohne Transformation)
lda_m1 <- lda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
# LDA (Daten mit Transformation)
lda_m2 <- lda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData_t)

    # LDA nur mit Hauptvariablen
    lda_gis <- lda(group ~ glucose + rw + insulin + sspg, data = trainData)
    lda_gis.p <- predict(lda_gis, newdata = trainData)

# Zusammenfassung des Modells
lda_m1
```

```
## Call:
## lda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2586207 0.5431034 0.1982759
##
## Group means:
##             glucose        fpg    insulin       sspg          rw
## chemical -0.1527486 -0.3392158  1.0392574  0.2605534  0.60888705
## normal   -0.6117151 -0.4916385 -0.1072838 -0.6407351 -0.35379707
## overt     1.5311498  1.4345199 -0.6513359  1.2708648  0.08809756
```

```
##
## Coefficients of linear discriminants:
##                 LD1        LD2
## glucose  4.87591983  1.8677167
## fpg     -2.55596630 -1.8956161
## insulin -0.07988361  0.8369985
## sspg     0.25821227 -0.1977281
## rw       0.25603990  0.5437438
##
## Proportion of trace:
##   LD1   LD2
## 0.885 0.115
```

```
lda_m2
```

```
## Call:
## lda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData_t)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2586207 0.5431034 0.1982759
##
## Group means:
##              glucose        fpg     insulin       sspg          rw
## chemical  0.04809973 -0.2918910  0.85407495  0.4198356  0.60784793
## normal   -0.71483447 -0.5589182  0.08244404 -0.6318908 -0.35259786
## overt     1.58306779  1.5494931 -0.85249480  1.0816803  0.09572234
##
## Coefficients of linear discriminants:
##                 LD1         LD2
## glucose  3.48047922  1.93046492
## fpg     -0.59143119 -1.95343959
## insulin  0.08203518  0.76539026
## sspg     0.05858524 -0.09275069
## rw       0.17809750  0.43479094
##
## Proportion of trace:
##    LD1    LD2
## 0.9027 0.0973
```

```
lda_gis
```

```
## Call:
## lda(group ~ glucose + rw + insulin + sspg, data = trainData)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2586207 0.5431034 0.1982759
##
## Group means:
##             glucose          rw    insulin       sspg
## chemical -0.1527486  0.60888705  1.0392574  0.2605534
## normal   -0.6117151 -0.35379707 -0.1072838 -0.6407351
```

```
## overt     1.5311498  0.08809756 -0.6513359  1.2708648
##
## Coefficients of linear discriminants:
##                 LD1        LD2
## glucose -2.18679912 -0.2761453
## rw      -0.17988298 -0.6064187
## insulin  0.03757696 -1.0111732
## sspg    -0.28425918  0.1540434
##
## Proportion of trace:
##    LD1    LD2
## 0.8783 0.1217
```

```
# Klasse der Trainingsdaten vorhersagen
lda_m1.p <- predict(lda_m1, newdata = trainData)
lda_gis.p <- predict(lda_gis, newdata = trainData)
lda_m2.p <- predict(lda_m2, newdata = trainData_t)

head(lda_m1.p$posterior)
```

```
##         chemical       normal        overt
## 49  1.913845e-03 9.980862e-01 5.072943e-11
## 65  9.850652e-01 1.425041e-02 6.844357e-04
## 74  2.722844e-02 9.727715e-01 1.214273e-08
## 122 4.933605e-04 1.436295e-07 9.995065e-01
## 145 3.367075e-11 2.716493e-15 1.000000e+00
## 128 2.003124e-03 9.987091e-08 9.979968e-01
```

```
head(lda_m1.p$class)
```

```
## [1] normal   chemical normal   overt    overt    overt
## Levels: chemical normal overt
```

```
# Genauigkeit des Modells überprüfen
print(paste("Model 1 (alle Parameter):",mean(lda_m1.p$class == trainData_t$group)))
```

```
## [1] "Model 1 (alle Parameter): 0.905172413793103"
```

```
print(paste("Model 2 (trans. Parameter):",mean(lda_m2.p$class == trainData_t$group)))
```

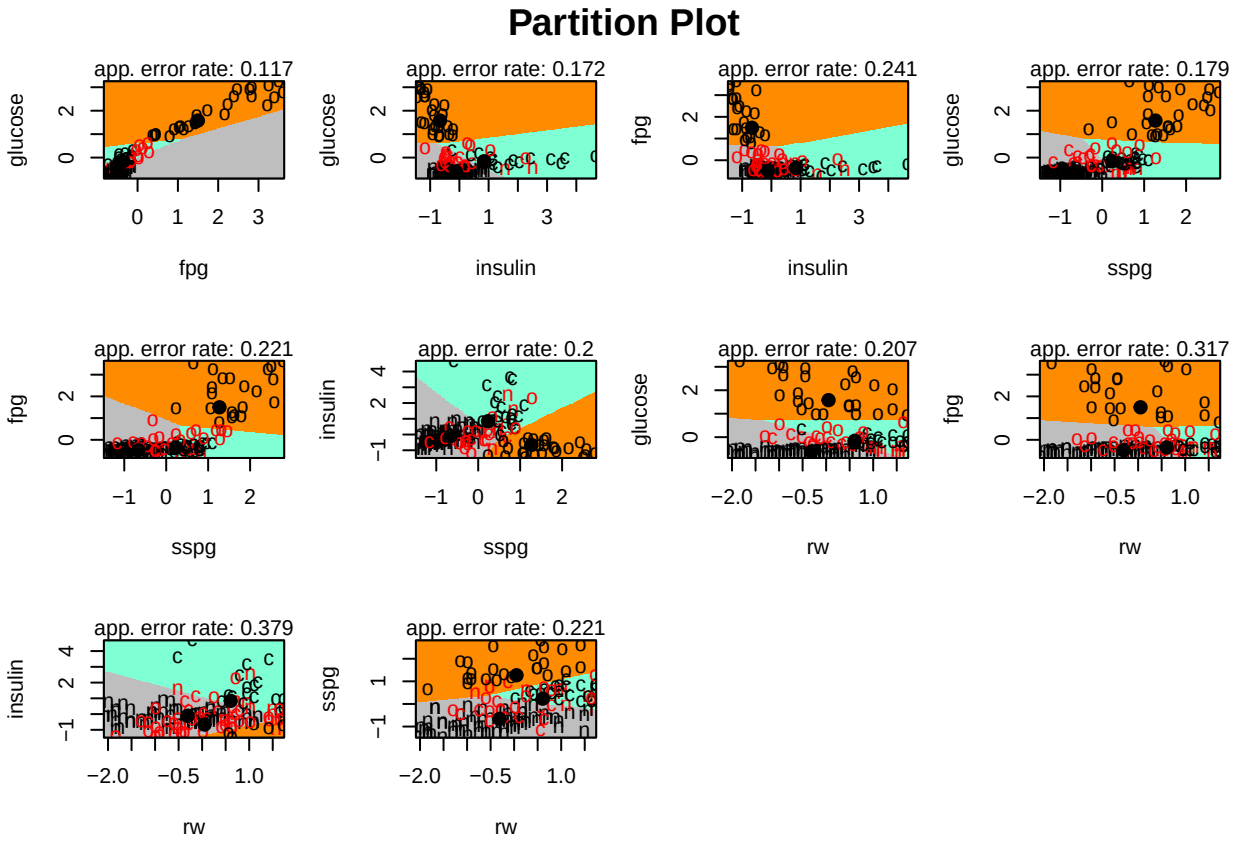
```
## [1] "Model 2 (trans. Parameter): 0.931034482758621"
```

```
print(paste("Model 3 (nur Hauptparameter):",mean(lda_gis.p$class == trainData_t$group)))
```

```
## [1] "Model 3 (nur Hauptparameter): 0.844827586206897"
```

Da durch das Modell mit den transformierten Daten eine höhere Genauigkeit erreicht werden konnte, wird neben dem Model 1 auch dieses Modell für die weitere Analyse und Klassifikation der Testdaten verwendet.

```r
#lda Trennung darstellen
# Partition Plot
partimat(group ~ glucose + fpg + insulin + sspg + rw,
         data = diabetes.pre, method = "lda",
         image.colors = c("aquamarine", "gray", "darkorange"))
```

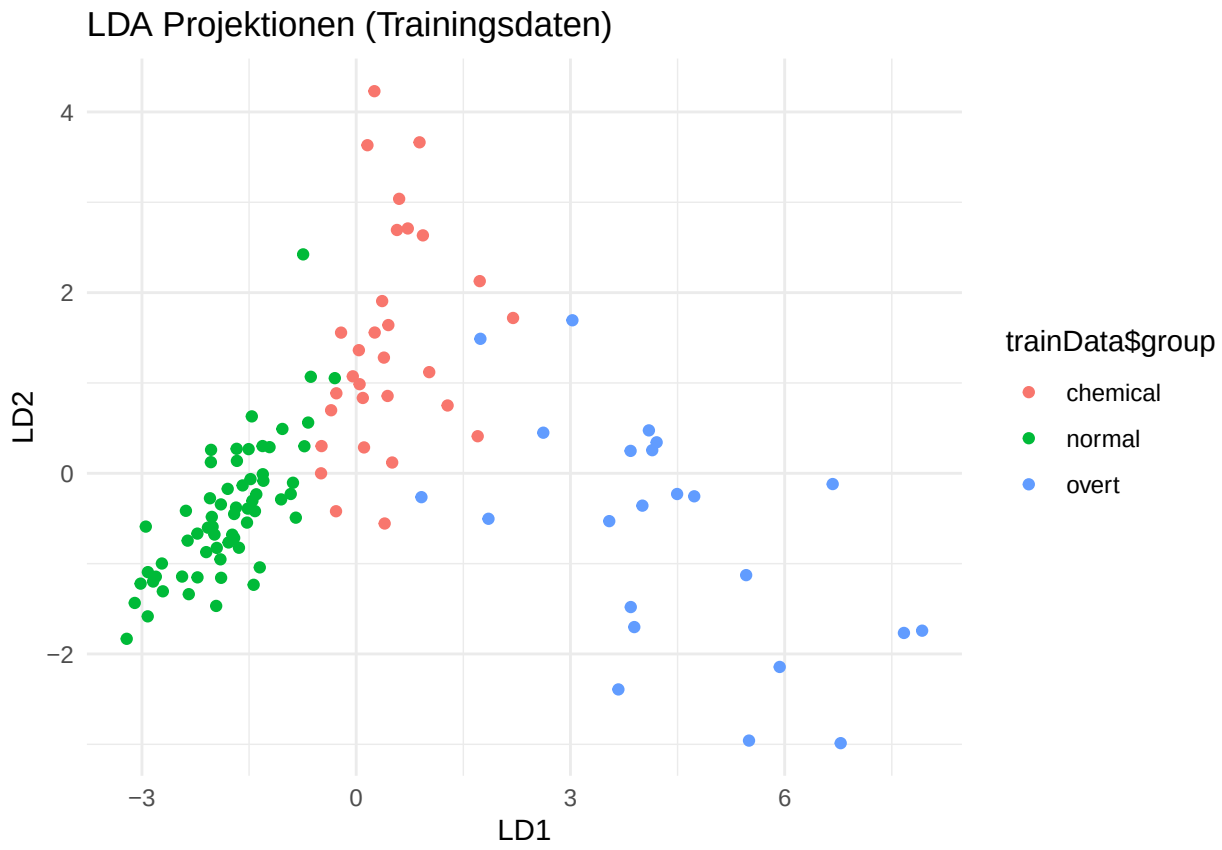



```r
# Projektionen
head(lda_m1.p$x) #LD1 und LD2
```

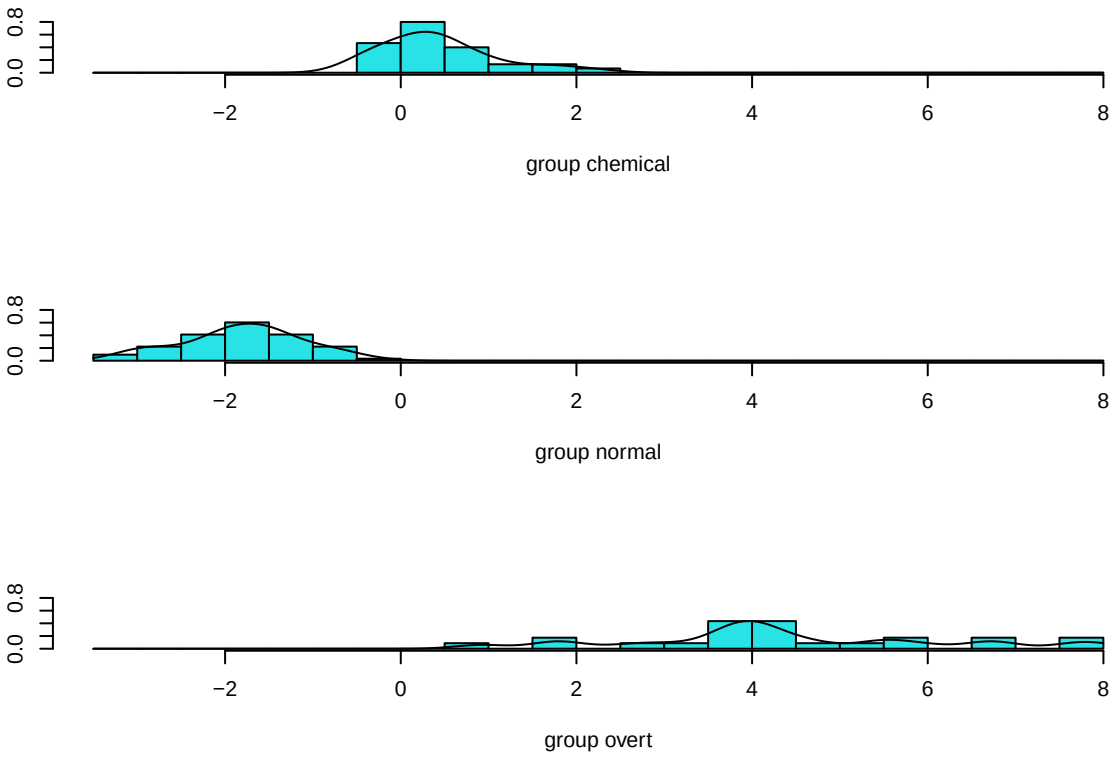
```
##           LD1         LD2
## 49  -2.384426 -0.41510594
## 65   1.022330  1.11994323
## 74  -1.479451 -0.06415764
## 122  4.007336 -0.35796333
## 145  6.784049 -2.98729450
## 128  4.098586  0.47514019
```

```r
# lda_m1.p$x als data.frame
lda_m1.p$x <- as.data.frame(lda_m1.p$x)

#projektionen graphisch darstellen
ggplot(lda_m1.p$x, aes(x = LD1, y = LD2, color = trainData$group)) +
  geom_point() +
  ggtitle("LDA Projektionen (Trainingsdaten)") +
```

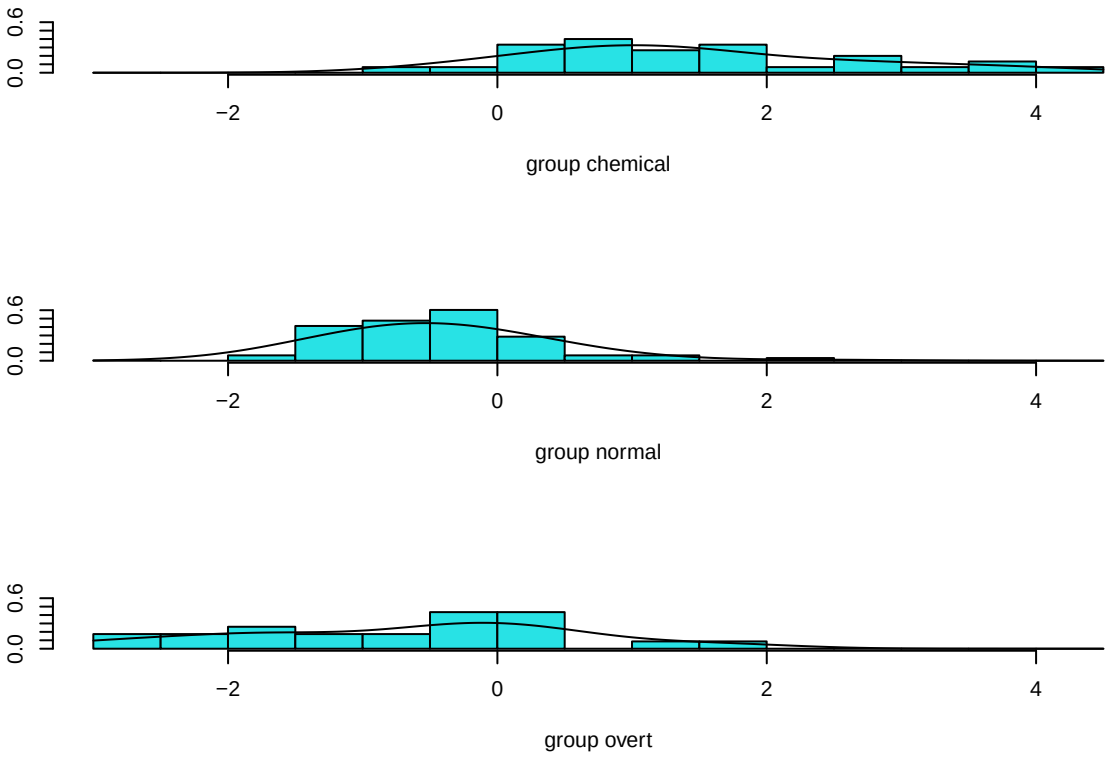
```
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()
```




```
# Histogramme der Projektionen
ldahist(lda_m1.p$x[,1], g = trainData$group,type = "both")
```




```
ldahist(lda_m1.p$x[,2], g = trainData$group, type = "both")
```




```
# Vorhersage auf Testdaten
lda_pred <- predict(lda_m1, newdata = testData)
lda_pred2 <- predict(lda_m2, newdata = testData_t)

head(lda_pred$posterior)
```

```
##        chemical    normal        overt
## 7  0.0023214274 0.9976786 1.226958e-10
## 11 0.0075812773 0.9924187 4.494838e-10
## 12 0.0001244314 0.9998756 3.381976e-12
## 23 0.0042265730 0.9957734 3.851015e-10
## 28 0.0221860832 0.9778139 3.138812e-09
## 37 0.0650780301 0.9349220 4.766249e-09
```

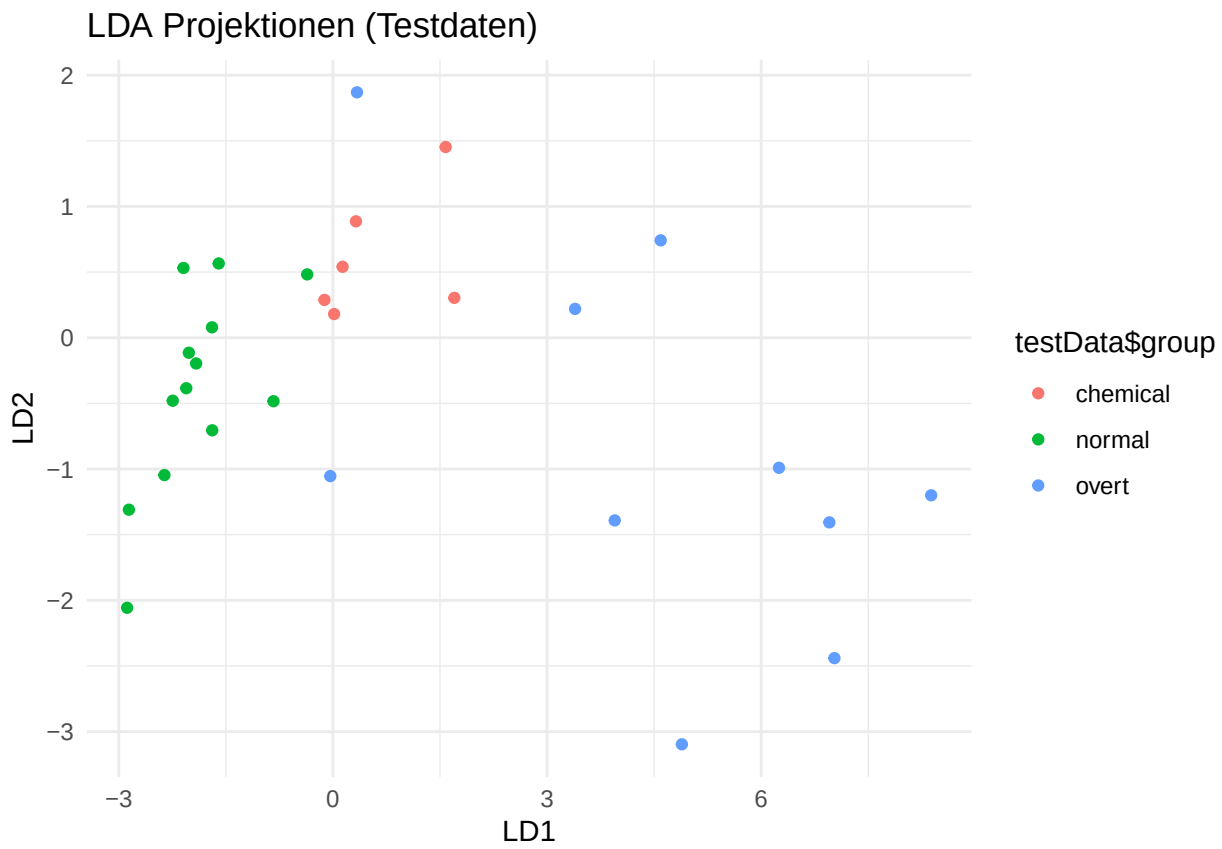
```
head(lda_pred$class)
```

```
## [1] normal normal normal normal normal normal
## Levels: chemical normal overt
```

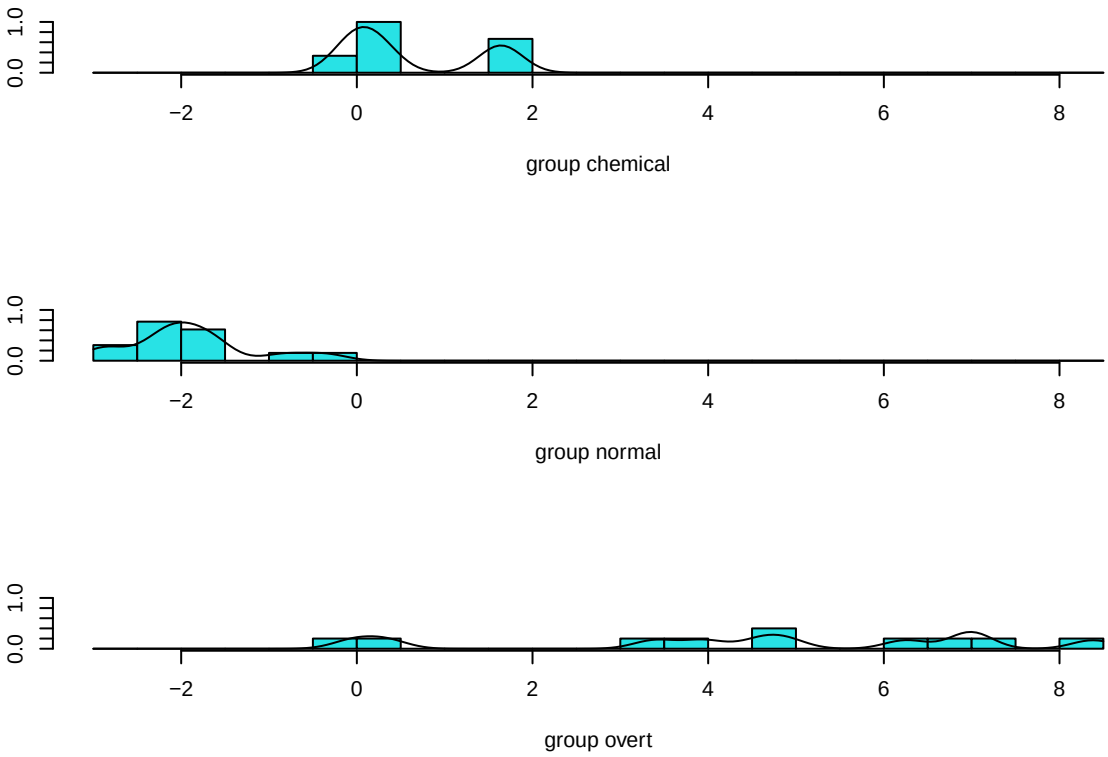
```
# Genauigkeit des Modells überprüfen
ac_lda <-mean(lda_pred$class == testData$group) # 93%
ac_lda_t <-mean(lda_pred2$class == testData_t$group) # 90%

# Projektionen der Testdaten
lda_pred$x #LD1 und LD2
```

```
##             LD1         LD2
## 7   -2.24386940 -0.47959353
## 11  -2.01843419 -0.11404898
## 12  -2.85829320 -1.30950936
## 23  -2.05461844 -0.38458332
## 28  -1.69373155  0.07952616
## 37  -1.59976577  0.56579969
## 39  -1.69076890 -0.70491935
## 45  -2.88284697 -2.05658411
## 51  -2.36094302 -1.04594732
## 52  -2.09346570  0.53144272
## 56  -1.91578422 -0.19597619
## 59   0.01726209  0.18050636
## 62   0.13598549  0.54041313
## 66  -0.11855509  0.28833208
## 67  -0.36001531  0.48249638
## 70  -0.83240309 -0.48334119
## 83   0.32288046  0.88685053
## 91   1.57965994  1.45271176
## 107  1.70127938  0.30362268
## 116  8.38184003 -1.19978015
## 121  3.39277350  0.21975105
## 126  6.95519813 -1.40627326
## 130  4.59255271  0.74175877
## 131  0.33737152  1.86909949
## 133  7.02632877 -2.43999746
## 134 -0.03661093 -1.05386480
## 139  4.88880623 -3.09639799
## 140  6.24966915 -0.99043375
## 143  3.94861736 -1.39131919
```

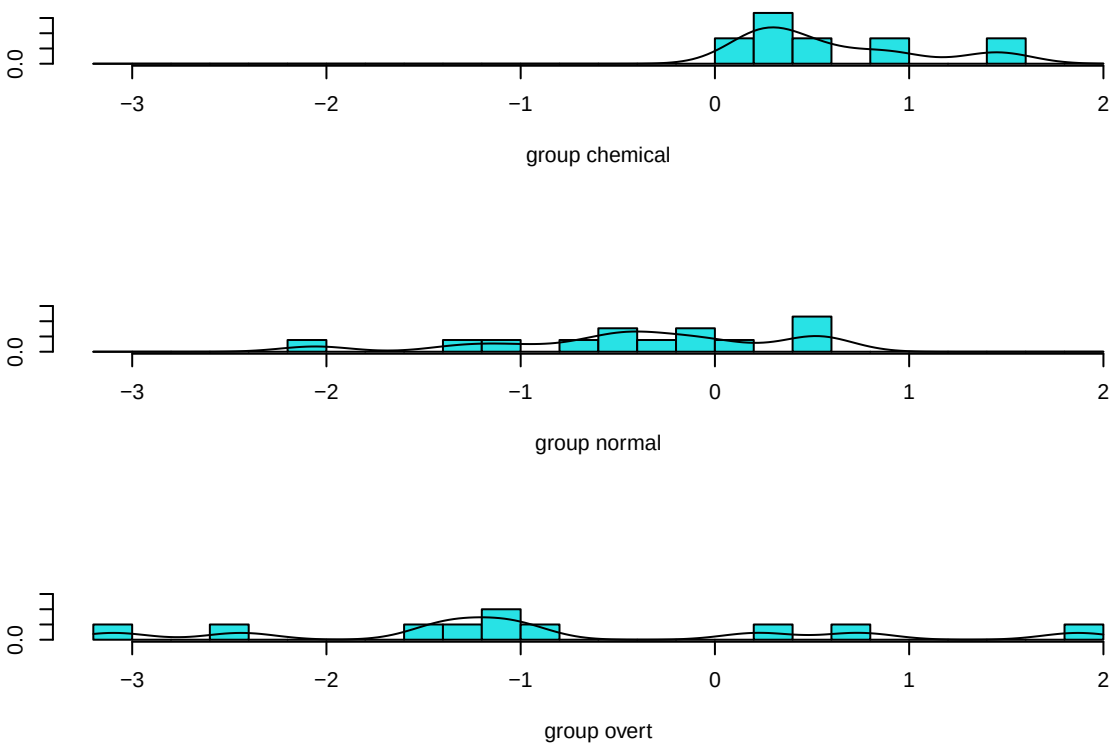
```r
# lda_m1.p$x als data.frame
lda_pred$x <- as.data.frame(lda_pred$x)

#projektionen graphisch darstellen
ggplot(lda_pred$x, aes(x = LD1, y = LD2, color = testData$group)) +
  geom_point() +
  ggtitle("LDA Projektionen (Testdaten)") +
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()
```




```
# Histogramme der Projektionen
ldahist(lda_pred$x[,1], g = testData$group,type = "both")
```




```
ldahist(lda_pred$x[,2], g = testData$group, type = "both")
```



Die Klassifikationsgenauigkeit von Modell 1 beträgt auf den Testdaten 93%. Somit performt das Modell auf den Testdaten etwas besser als auf den Trainingsdaten, da diese nur zu 90% richtig klassifiziert wurden. Erstaunlich ist, dass der anfänglich untersuchte, explorative Ansatz einer Klassifikation über einen einfachen Schwellwert der Glucose-Variable zu einer höheren Klassifikationsgenauigkeit führte.

Das Modell 2, welches die transformierten Daten verwendet, erreicht auf den Testdaten eine Klassifikationsgenauigkeit von 90% und auf den Trainingsdaten eine Genauigkeit von 93%.

iv) Vergleichen Sie unterschiedliche Varianten der Diskriminanzanalyse (QDA, MDA, FDA) hinsichtlich ihrer Klassifikationsgenauigkeit.

```
#quadratische QDA
m_qda <- qda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
m_qda.p <- predict(m_qda, newdata = testData)
ac_qda <- mean(m_qda.p$class == testData$group) # 0.90

#MDA
library(mda)
m_mda <- mda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
m_mda.p <- predict(m_mda, newdata = testData)
ac_mda <- mean(m_mda.p== testData$group) # 0.90

#FDA
m_fda <- fda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
m_fda.p <- predict(m_fda, newdata = testData)
ac_fda <-mean(m_fda.p == testData$group) # 0.93
```

```r
#RDA
m_rda <- rda(group ~ glucose + fpg + insulin + sspg + rw, data = trainData)
m_rda.p <- predict(m_rda, newdata = testData)
ac_rda <- mean(m_rda.p$class == testData$group) # 0.93

# Tabelle
results <- data.frame(Model = c("LDA","LDA_T", "QDA", "MDA", "FDA", "RDA"),
                      Genauigkeit = c(ac_lda, ac_lda_t, ac_qda, ac_mda, ac_fda, ac_rda))
print(results)
```

```
##   Model Genauigkeit
## 1   LDA   0.9310345
## 2 LDA_T   0.8965517
## 3   QDA   0.8965517
## 4   MDA   0.9310345
## 5   FDA   0.9310345
## 6   RDA   0.8620690
```

Sowohl die FDA als auch die RDA erreichen ebenfalls eine Klassifikationsgenauigkeit von 93% auf den Testdaten. Die QDA, MDA sowie die LDA mit transformierten Daten erreichen nur eine Genauigkeit von etwa 90%.

2 Principal Component Analyse [5P]

Führen Sie eine PCA unter Berücksichtigung folgender Punkte durch:

i) Überprüfung der paarweisen Kovarianzen / Korrelationen

```r
# Load data
diabetes <- read.csv("diabetes_RM.csv")

# Overview
str(diabetes)
```

```
## 'data.frame':    145 obs. of  6 variables:
##  $ rw     : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ fpg    : int  80 97 105 90 90 86 100 85 97 97 ...
##  $ glucose: int  356 289 319 356 323 381 350 301 379 296 ...
##  $ insulin: int  124 117 143 199 240 157 221 186 142 131 ...
##  $ sspg   : int  55 76 105 108 143 165 119 105 98 94 ...
##  $ group  : chr  "normal" "normal" "normal" "normal" ...
```

```r
head(diabetes)
```

```
##     rw fpg glucose insulin sspg  group
## 1 0.81  80     356     124   55 normal
## 2 0.95  97     289     117   76 normal
## 3 0.94 105     319     143  105 normal
## 4 1.04  90     356     199  108 normal
## 5 1.00  90     323     240  143 normal
## 6 0.76  86     381     157  165 normal
```

```r
# Calculate variance for each numeric column
variances <- sapply(diabetes[sapply(diabetes, is.numeric)], var)
print(variances)
```

```
##           rw          fpg      glucose      insulin         sspg
## 1.670174e-02 4.087097e+03 1.004578e+05 1.462531e+04 1.124233e+04
```

```r
# Calculate covariance between all pairs of numeric columns
covariances <- cov(diabetes[sapply(diabetes, is.numeric)])
print(covariances)
```

```
##                  rw           fpg       glucose       insulin         sspg
## rw       0.01670174 -7.281513e-02  9.824262e-01      3.473373     5.266255
## fpg     -0.07281513  4.087097e+03  1.954606e+04  -3063.463649  4849.905651
## glucose  0.98242625  1.954606e+04  1.004578e+05 -12918.162739 25908.490182
## insulin  3.47337308 -3.063464e+03 -1.291816e+04  14625.312548   101.482519
## sspg     5.26625479  4.849906e+03  2.590849e+04    101.482519 11242.331897
```

```r
# Select only numeric variables, excluding 'group'
numeric_vars <- sapply(diabetes, is.numeric)
numeric_vars["group"] <- FALSE

# Compute the correlation matrix
cor_matrix <- cor(diabetes[numeric_vars])

# Print the correlation matrix
print(cor_matrix)
```

```
##                   rw          fpg    glucose      insulin        sspg
## rw       1.000000000 -0.008813193  0.0239843  0.222237813 0.384319804
## fpg     -0.008813193  1.000000000  0.9646281 -0.396234858 0.715480192
## glucose  0.023984304  0.964628091  1.0000000 -0.337020435 0.770942459
## insulin  0.222237813 -0.396234858 -0.3370204  1.000000000 0.007914263
## sspg     0.384319804  0.715480192  0.7709425  0.007914263 1.000000000
```

Die Kovarianzmatrix zeigt, dass es Korrelationen zwischen unterschiedlichen Variablen gibt: Nicht-Diagonal-Elemente sind ungleich 0.

ii) Berechnen der PCA und Beurteilung wie viele PCs sinnvoll sind (Eigenwerte und Screeplot).

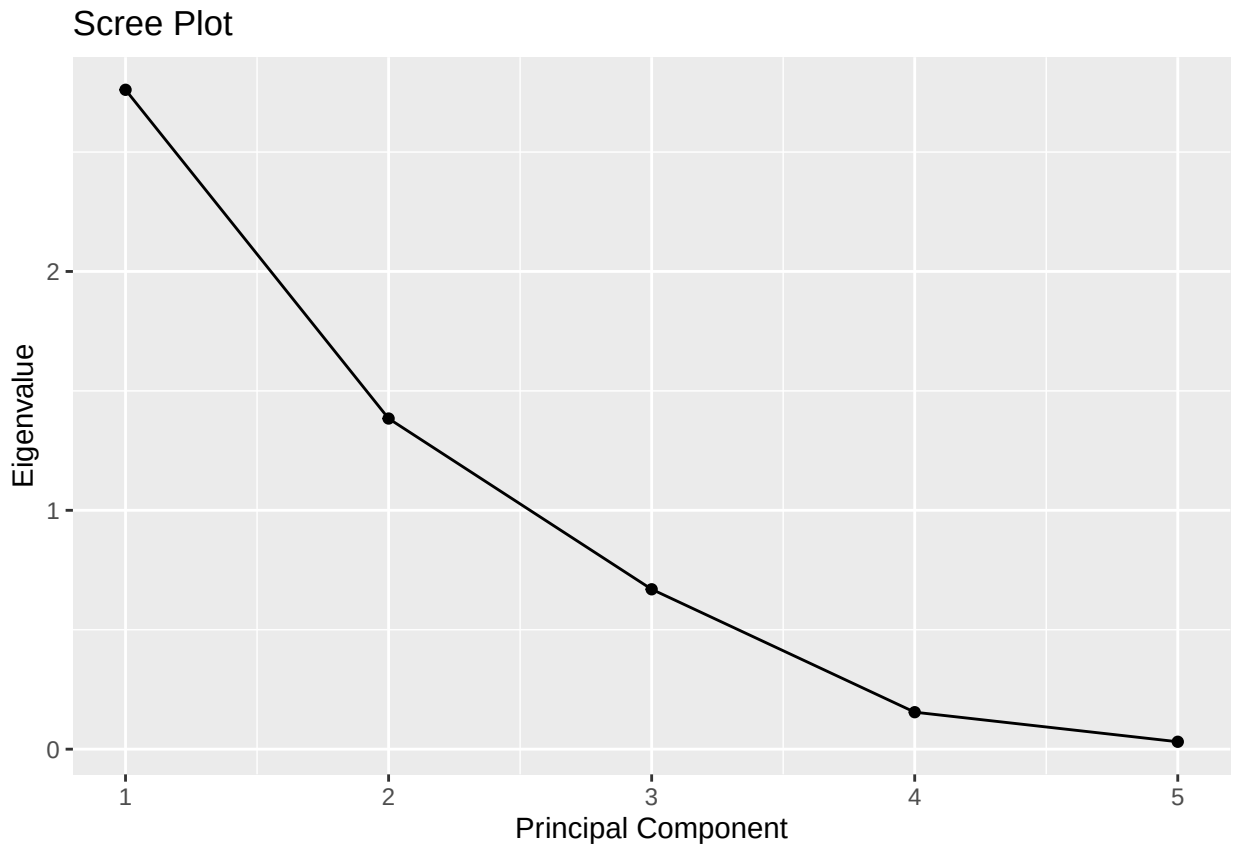
```r
# Perform PCA
pca_result <- prcomp(diabetes[sapply(diabetes, is.numeric)], scale. = TRUE)

# Print summary of the PCA result
print(summary(pca_result))
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5
## Standard deviation     1.6615 1.1766 0.8181 0.39341 0.17589
## Proportion of Variance 0.5521 0.2769 0.1339 0.03095 0.00619
## Cumulative Proportion  0.5521 0.8290 0.9629 0.99381 1.00000
```

```r
# Eigenvalues (variances of the principal components)
eigenvalues <- pca_result$sdev^2
print(eigenvalues)
```

```
## [1] 2.76063291 1.38429713 0.66935968 0.15477222 0.03093807
```

```r
# Scree plot
scree_df <- data.frame(PC = 1:length(eigenvalues), eigenvalues = eigenvalues)
ggplot(data = scree_df, aes(x = PC, y = eigenvalues)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:length(eigenvalues)) +
  labs(title = "Scree Plot", x = "Principal Component", y = "Eigenvalue")
```



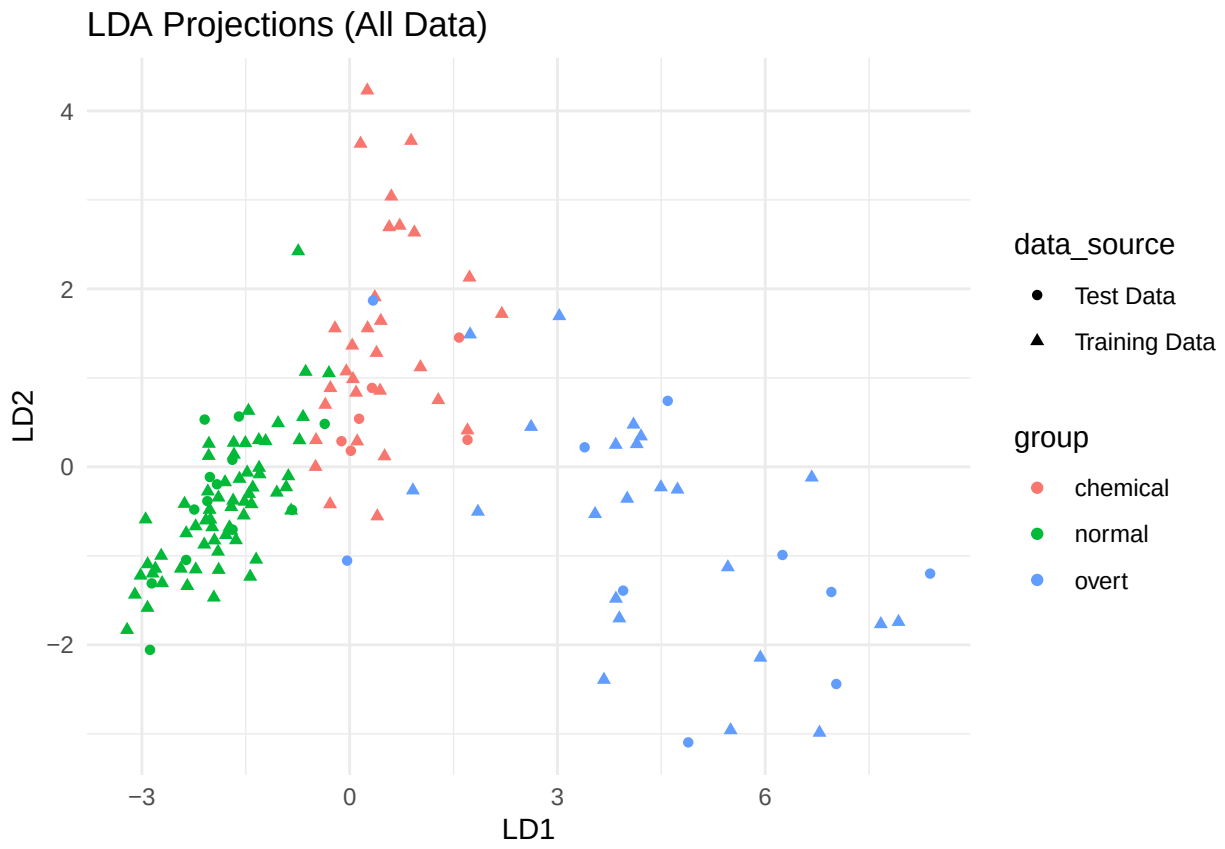
Da PC1 und PC2 bereits 82% der totalen Streuung erklären, können zwei (maximal 3 (96%)) PCs sinnvoll sein.

iii) Transformation der ursprünglichen Daten in ein PC-Koordinatensystem mit zwei PCs. Vergleichen Sie die Darstellung mit dem Ergebnis der LDA (LD1 und LD2 Projektionen).

Im Folgenden werden zuerst die LD-Projektionen der kombinierten Daten aus Training und Test gezeigt.

```r
# Add group information to lda_m1.p$x and lda_pred$x
lda_m1.p$x$group <- trainData$group
```

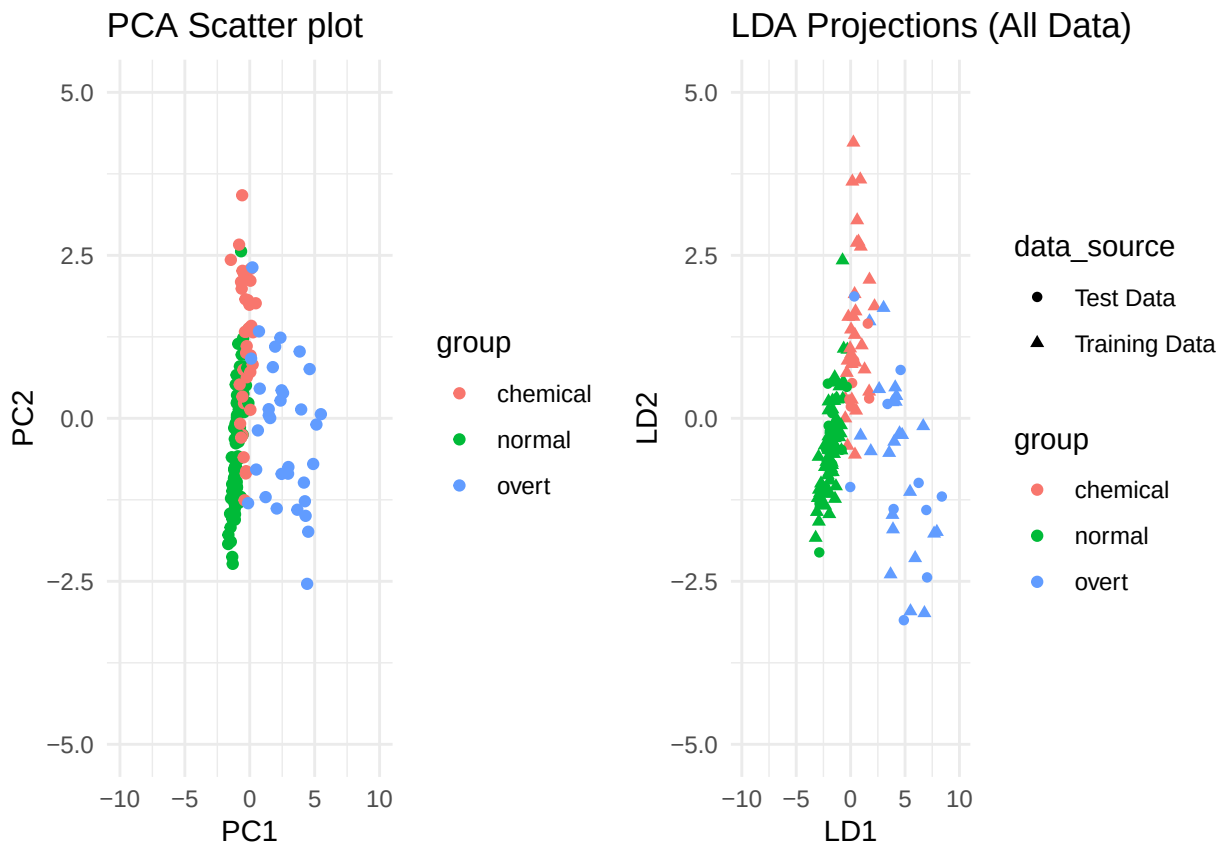
```r
lda_pred$x$group <- testData$group

# Add a new variable to indicate the data source
lda_m1.p$x$data_source <- "Training Data"
lda_pred$x$data_source <- "Test Data"

# Combine the data frames
combined_data <- rbind(lda_m1.p$x, lda_pred$x)

# Create the combined plot
combined_plot <- ggplot(combined_data, aes(x = LD1, y = LD2, color = group)) +
  geom_point(aes(shape = data_source)) +
  ggtitle("LDA Projections (All Data)") +
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()

print(combined_plot)
```



Nun machen wir einen Vergleich zwischen den LD-Projektionen und den PC-Projektionen der gesamten Daten.

```r
# Perform PCA
pca_result <- prcomp(diabetes[sapply(diabetes, is.numeric)], scale. = TRUE)

# Extract the scores of the first two PCs
```

```r
PC1_scores <- pca_result$x[,1]
PC2_scores <- pca_result$x[,2]

# Create a new data frame with the scores of the first two PCs
transformed_data <- data.frame(PC1 = PC1_scores, PC2 = PC2_scores)

# Add group information to the transformed data
transformed_data$group <- diabetes$group


# First plot (PCA scatter plot)
plot1 <- ggplot(transformed_data, aes(x = PC1, y = PC2, color = group)) +
  geom_point() +
  xlim(-10, 10) +
  ylim(-5, 5) +
  xlab("PC1") +
  ylab("PC2") +
  ggtitle("PCA Scatter plot") +
  theme_minimal()

# This plot is commented out because it only shows the training data

# Seconde plot (LDA scatter plot)
# plot2 <- ggplot(lda_m1.p$x, aes(x = LD1, y = LD2, color = trainData$group)) +
#   geom_point() +
#   xlim(-10, 10) +
#   ylim(-5, 5) +
#   ggtitle("LDA Projections (Training Data)") +
#   xlab("LD1") +
#   ylab("LD2") +
#   theme_minimal()

# Second plot (LDA scatter plot)
plot2 <- ggplot(combined_data, aes(x = LD1, y = LD2, color = group)) +
  geom_point(aes(shape = data_source)) +
  xlim(-10, 10) +
  ylim(-5, 5) +
  ggtitle("LDA Projections (All Data)") +
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()

grid.arrange(plot1, plot2, ncol = 2)
```

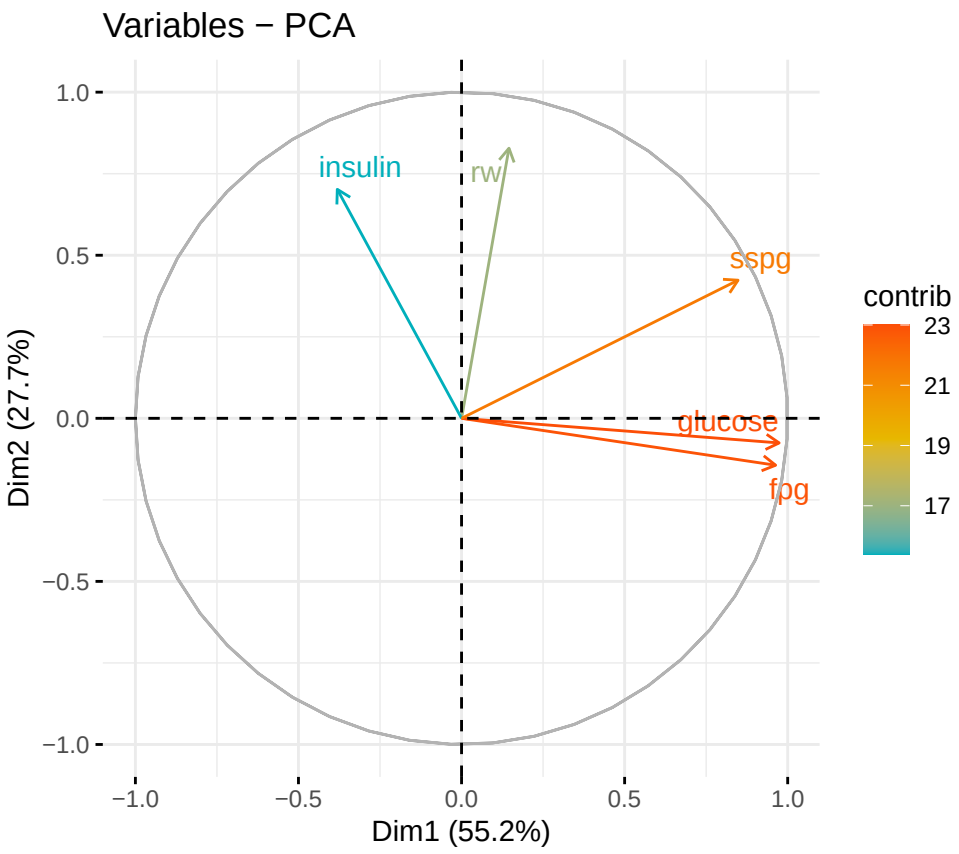



```
# Print the transformed data
# print(transformed_data)
```

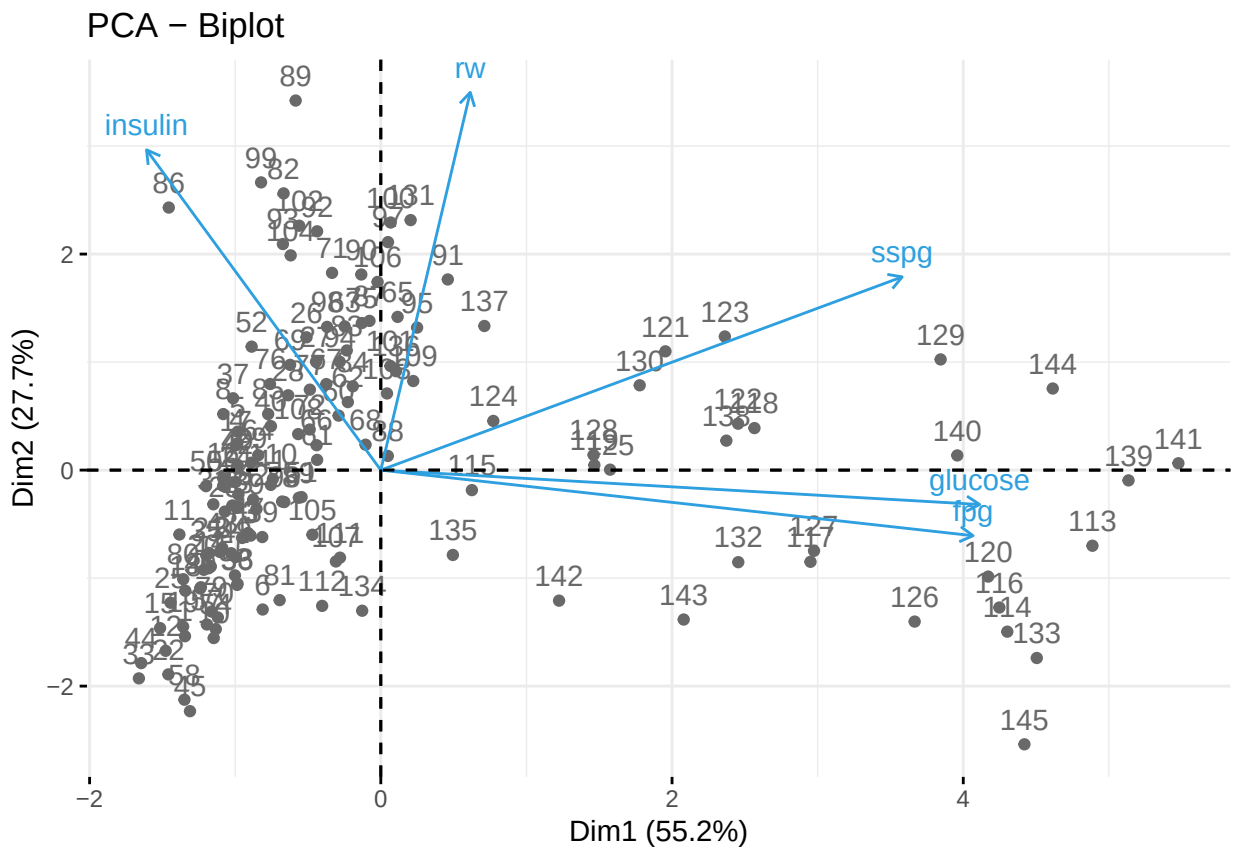
Die Plots lassen ein ähnliches Muster erkennen, mit dem Unterschied dass die PC2-Projektionen deutlich weniger Varianz aufweisen als die LD2-Projektionen. In iiii) wird sich zeigen, dass sich dieser Befund damit deckt, dass die LDA der PC-Projektionen eine geringere Genauigkeit erreicht als die LDA der ursprünglichen Daten. Dies ist einleuchtend, wenn man bedenkt, dass die LDA für das Auffinden der LDs die Labelinformationen nutzen kann, während die PCA lediglich die Varianz der Daten berücksichtigt.

iii) Stellen Sie den Correlation Circle und Biplot graphisch dar. Welche Information liefert diese Darstellung?

```
# Perform PCA
pca_result <- prcomp(diabetes[sapply(diabetes, is.numeric)], scale. = TRUE)

# Plot the correlation circle
fviz_pca_var(pca_result, col.var="contrib", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel =
```




```
# Plot the biplot
fviz_pca_biplot(pca_result, col.var="#2E9FDF", col.ind="#696969")
```



Correlation Circle:

Die Normalprojektionen der Vektoren auf die PC-Achsen zeigen die Korrelation zwischen den Prädiktoren und den jeweiligen PCs im Einzelnen. Die Länge der normierten Vektoren (0 bis 1) zeigt die Stärke der Korrelation zwischen einer Variable und den (zwei) PCs im Gesamten. Da die ersten zwei PC-Dimensionsn etwa 82% der totalen Streuung erklären, liegen die Längen der Vektoren hinreichend nahe bei 1. Am geringsten ist die Korrelation zwischen den Variablen rw und insulin und den PCs im Gesamten, da sie die kürzesten Vektoren haben.

Biplot:

Es ist gut erkennbar, dass die Streuung der PC1-Projektionen (PC1 erfasst 55% der totalen Streuung) stärker ist als die Streuung der PC2-Projektionen (PC2 erfasst 28% der totalen Streuung). Die Vektoren der Variablen glucose und fpg zeigen in Richtung der PC1-Achse, was für eine starke Korrelation zwischen diesen Variablen und PC1 spricht. Dasselbe gilt für die Vektoren der Variablen insulin und rw für die PC2.

iv) Beurteilen Sie die Qualität und Beiträge der Variablen auf die PCs.

Die Variablen fpg, glucose und sspg tragen am stärksten zu den PCs im Gesamten bei. glucose und fpg tragen am stärksten zu PC1 bei. Bei fpg, glucose, sspg und rw besteht eine positive Korrelation zu PC1. Die Variablen insulin und rw korrelieren positiv mit PC2 und tragen am stärksten zu PC2 bei. Wie gut zu erkennen ist, korreliert insulin negativ mit PC1, während fpg und glucose leicht negativ mit PC2 korrelieren.

v) Wiederholen Sie die LDA von Aufgabe 1 unter Verwendung der PCs zur Klassifizierung. Achten Sie auf die Verwendung der gleichen Trainings- und Test-Daten und vergleichen Sie die Performance. Vergleichen Sie weiters die Performance der LDA mit den Variablen `glucose` und `fpg` mit der PCA+LDA mit zwei PCs.

Diskriminanzanalyse mit PC-Projektionen

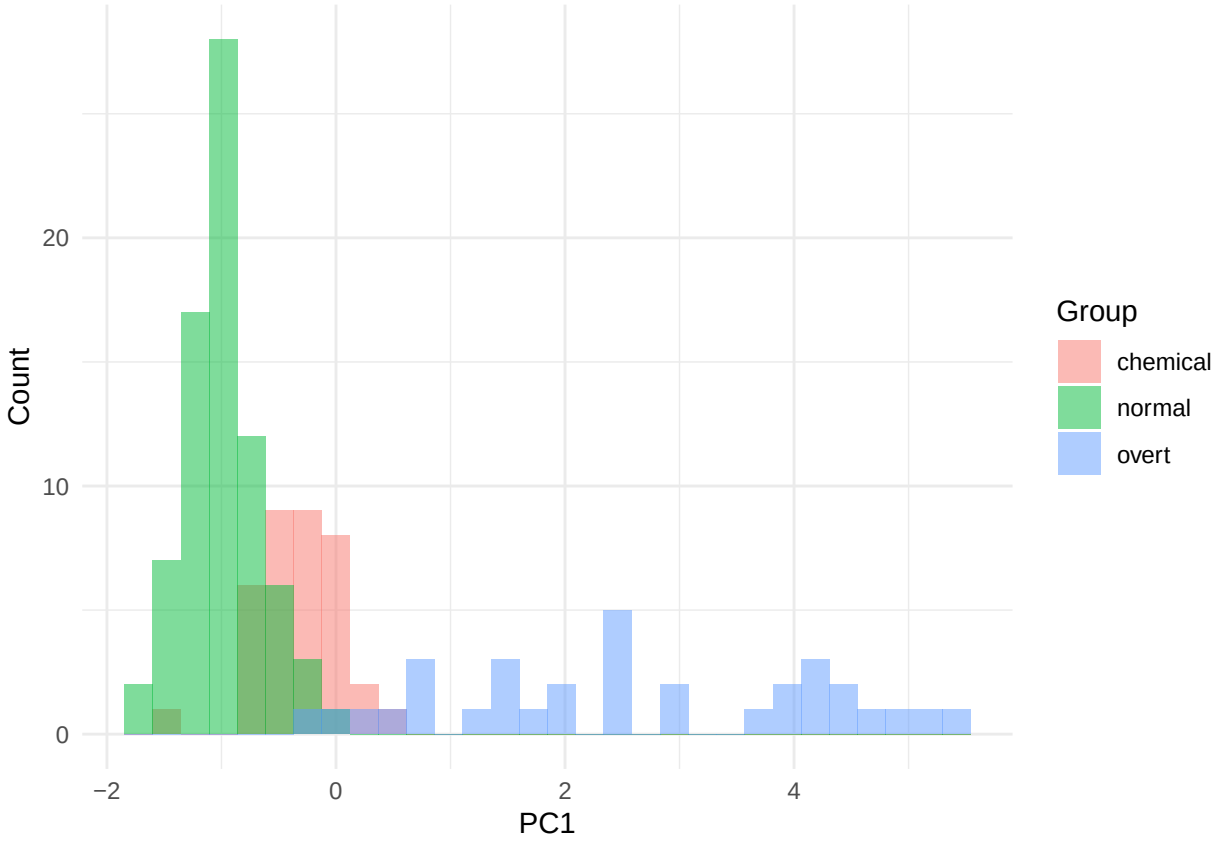
Führen Sie eine Diskriminanzanalyse unter Berücksichtigung folgender Punkte durch:

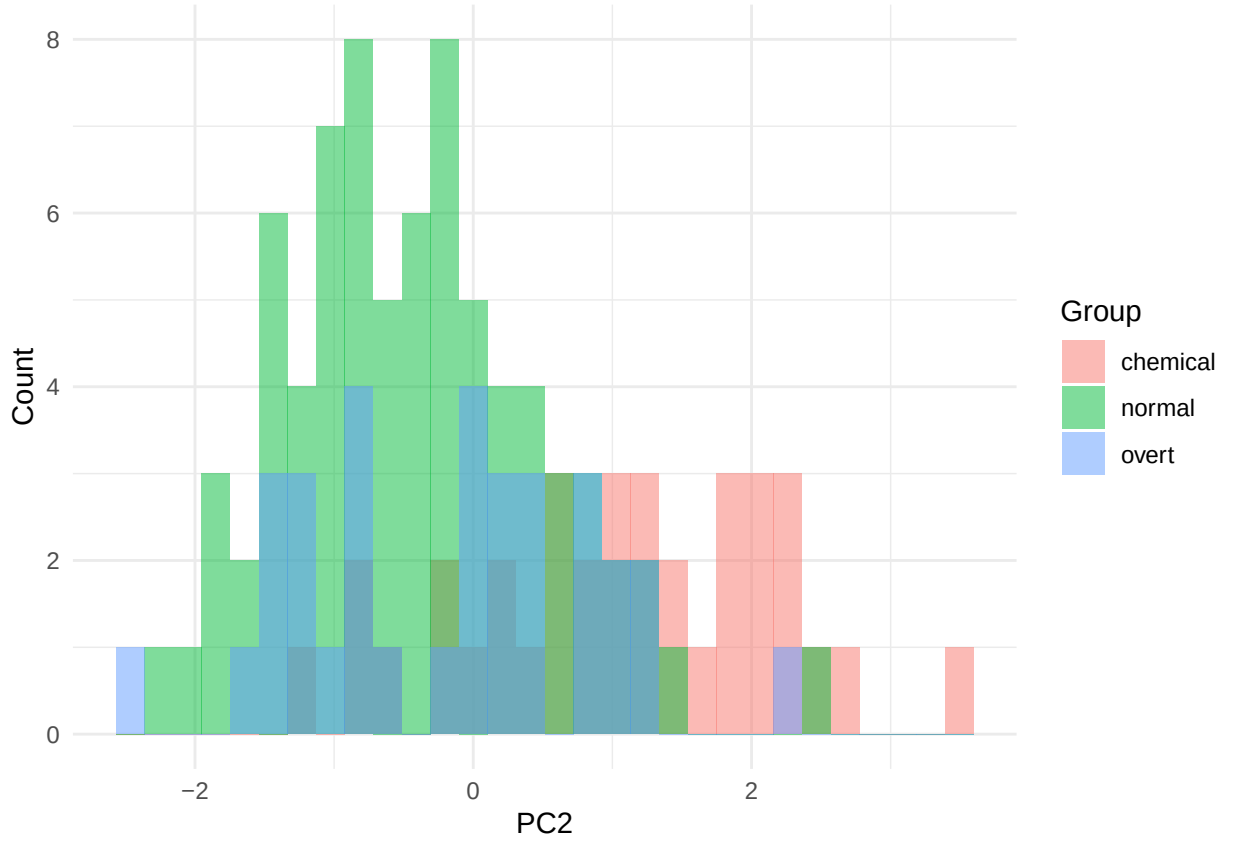
i) Explorative Analyse der Prädiktoren mit Hilfe von Histogrammen. Gibt es Prädiktoren, die bereits eine gute Trennung zwischen den Klassen erlauben?

```
# Datensatz laden
diabetes <- read.csv("diabetes_RM.csv")

# Perform PCA
pca_result <- prcomp(diabetes[sapply(diabetes, is.numeric)], scale. = TRUE)

# Extract the scores of the first two PCs
PC1 <- pca_result$x[,1]
PC2 <- pca_result$x[,2]

# Create a new data frame with the scores of the first two PCs
pca_scores <- data.frame(PC1 = PC1, PC2 = PC2)

# Add labels to PCA scores
pca_scores$group <- diabetes$group

# Assuming `pca_scores` is a data frame with your PCA scores
# and `group` is a factor variable indicating the group of each observation

# Calculate means of principal components for each group
aggregate(cbind(PC1, PC2) ~ group, pca_scores, mean)
```

```
##      group        PC1        PC2
## 1 chemical -0.3182846  1.0678079
## 2   normal -0.9767612 -0.4249652
## 3    overt  2.5967301 -0.1861735
```

```
# Create histograms
# You would need to define or adapt the `create_histograms()` function to work with PCA scores
create_histograms(pca_scores)
```


Count
20
10
0
−2
0
2
4
PC1
Group
chemical
normal
overt



Anhand der Histogramme lässt sich erkennen, dass PC2 nicht als Trennungskriterium geeignet ist, da die Gruppen sich stark überlappen. Allerdings lässt ich die "overt" Gruppe bei PC1 gut von den anderen Gruppen abgrenzen.

ii) Überprüfen Sie ob die Vorraussetzungen für eine LDA gegeben sind und führen Sie eine Standardisierung der Daten durch.

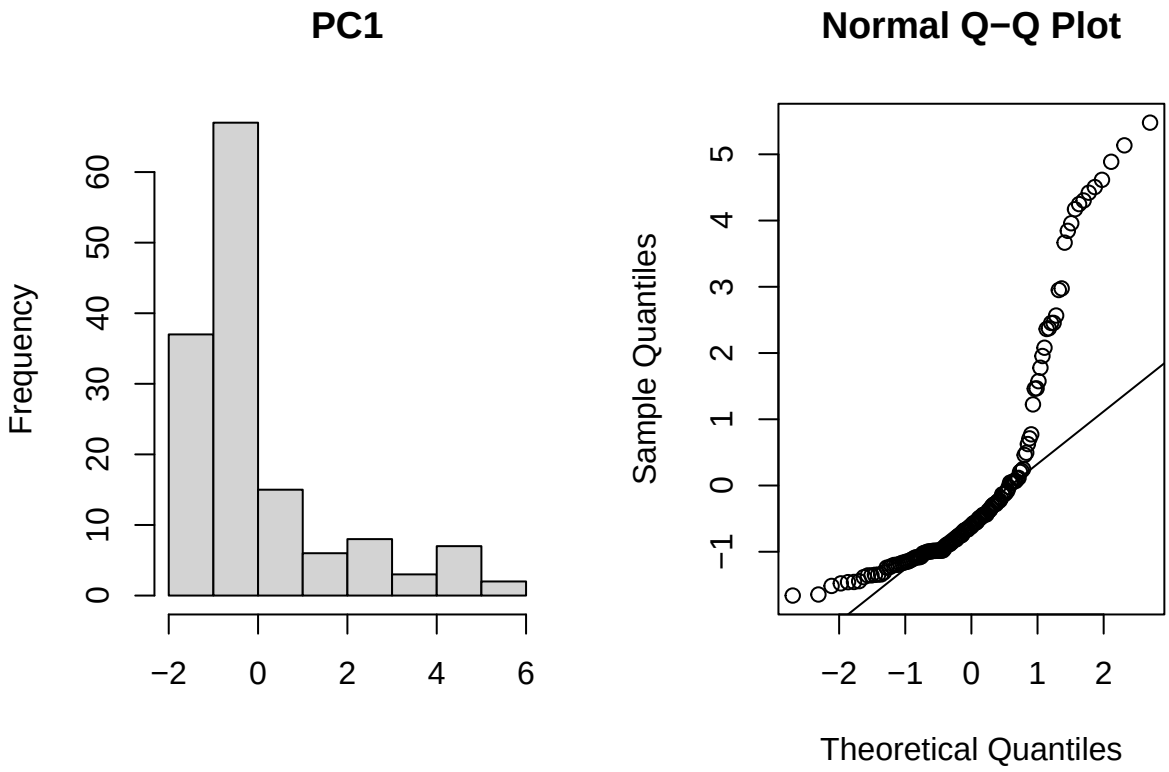
```
# Überprüfen der Voraussetzungen

descriptions <- get_descriptions()

# Normalverteilung der Prädiktoren (Visuell u. Konservativ)

print("p-Werte des Shapiro-Wilk-Tests: ")
```

```
## [1] "p-Werte des Shapiro-Wilk-Tests: "
```

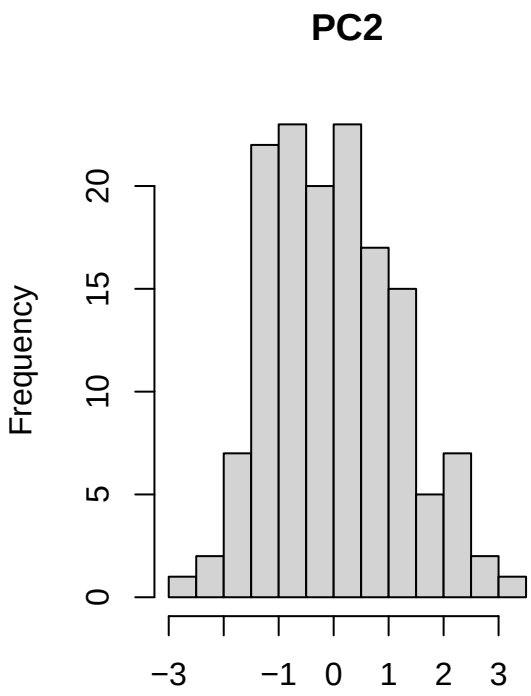
```
for (i in 1:2) {

  #Shapiro-Wilk-Test
  sw_p <- shapiro.test(pca_scores[,i])$p.value
  print(paste("p-Wert für", colnames(pca_scores)[i], ":", sw_p))


  par(mfrow = c(1,2))
```

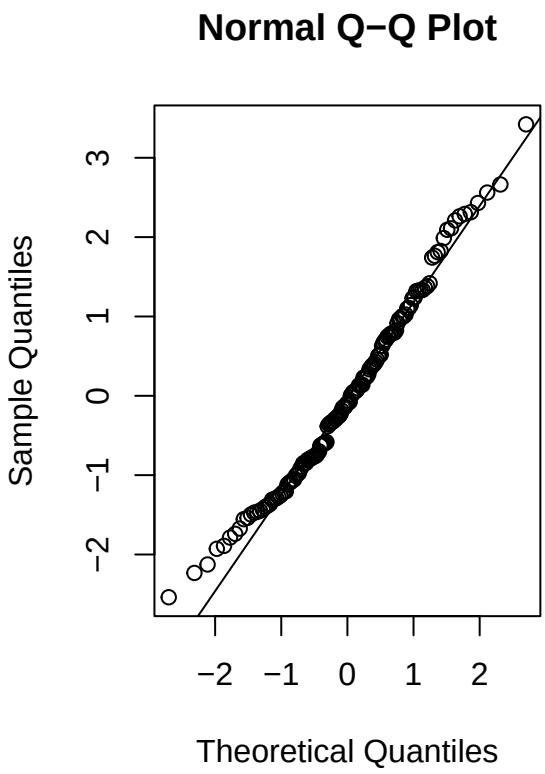
```r
  # Histogramme
  hist(pca_scores[,i], main = colnames(pca_scores)[i], xlab = "")

  # QQ-Plots
  qqnorm(pca_scores[,i])
  qqline(pca_scores[,i])

}
```

```
## [1] "p-Wert für PC1 : 1.79294257940679e-14"
```




```
## [1] "p-Wert für PC2 : 0.0770158068612695"
```



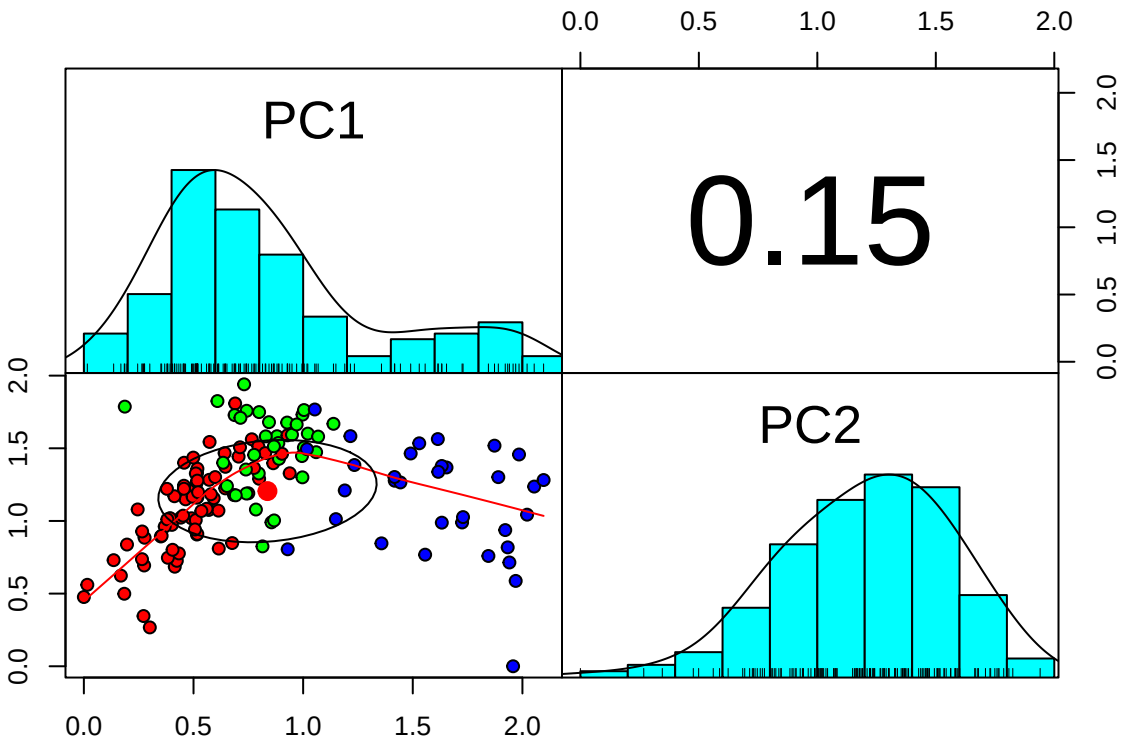



```r
# Überprüfung der Multivariaten Normalverteilung
mvn_test <- MVN::mvn(pca_scores[, sapply(pca_scores, is.numeric)], mvnTest = "hz")
print(mvn_test)
```

```
## $multivariateNormality
##            Test       HZ p value MVN
## 1 Henze-Zirkler 9.335833       0  NO
##
## $univariateNormality
##               Test  Variable Statistic   p value Normality
## 1 Anderson-Darling    PC1      13.8293  <0.001      NO
## 2 Anderson-Darling    PC2       0.6730  0.0773      YES
##
## $Descriptives
##       n         Mean  Std.Dev      Median       Min      Max       25th
## PC1 145 6.679882e-17 1.661515 -0.61898939 -1.661322 5.477520 -1.0016630
## PC2 145 2.875603e-16 1.176562 -0.09526639 -2.538729 3.421412 -0.8523993
##           75th      Skew   Kurtosis
## PC1 0.06753672 1.7279335  2.0178483
## PC2 0.78514678 0.3818425 -0.3630788
```

In den Histogrammen und QQ-Plots ist zu erkennen, dass die PC2-Projektionen normalverteilt sind (p-Wert>0.05), allerdings sind die PC1-Projektionen nicht normalverteilt (p-Wert nahezu 0).

```r
# Transformieren der Daten
pca_scores_trans <- pca_scores %>%
  mutate(PC1 = log(PC1 - min(PC1) + 1),
         PC2 = log(PC2 - min(PC2) + 1),
         group = as.factor(group))

# Überprüfung der Transformation
pairs.panels(pca_scores_trans[,1:2],
             gap = 0,
             bg = c("green","red","blue")[pca_scores_trans$group],
             pch = 21)
```



Darüber hinaus zeigt der multivariate Normalverteilungstest, dass die Daten nicht multivariat normalverteilt sind.

Auch nach der Log-Transformation der Daten sind die PC1-Projektionen nicht normalverteilt. Die LDA wird dennoch durchgeführt, da sie robust gegenüber Verletzungen der Normalverteilung ist.

```r
# Korrelationen
cor_matrix <- cor(pca_scores[,1:2])
print(cor_matrix)
```

```
##               PC1           PC2
## PC1 1.000000e+00 1.451026e-16
## PC2 1.451026e-16 1.000000e+00
```

```r
# Kovarianzen
cov_matrix <- cov(pca_scores[,1:2])
print(cov_matrix)
```

```
##              PC1          PC2
## PC1 2.760633e+00 2.836574e-16
## PC2 2.836574e-16 1.384297e+00
```

Die Korrelations/Kovarianzmatrix zeigt erwartungsgemäß, dass die Nicht-Diagonal-Elemente bei 0 liegen. Das ist so, weil die PC-Vektoren eine orthogonale Basis bilden und daher nicht miteinander korrelieren.

```r
# Überprüfung der Homogenität der Kovarianzen
boxM_test <- boxM(pca_scores[, sapply(pca_scores, is.numeric)], pca_scores$group)
print(boxM_test)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  pca_scores[, sapply(pca_scores, is.numeric)]
## Chi-Sq (approx.) = 209.41, df = 6, p-value < 2.2e-16
```

Der Box-M-Test zeigt, dass die Kovarianzmatrizen der Gruppen nicht gleich sind. Dies bedeutet, dass die Annahme der Homogenität der Kovarianzmatrizen verletzt ist. Streng genommen sind die Voraussetzungen für eine LDA nicht gegeben. Da die LDA in Klassifizierungsaufgaben jedoch robust gegenüber Verletzungen der Normalverteilung und Homogenität der Kovarianzen ist, wird die LDA dennoch durchgeführt.

PC-Projektionen von standardisierten Daten sollten nicht nochmal standardisiert werden (weil es unnötig ist und zu verzerrten Ergebnissen führen kann).

iii) Unterteilen sie die gesamten Daten in Trainings- und Test-Daten und führen Sie in der weiteren Folge eine Klassifizierung mit einer LDA durch. Evaluieren Sie die Perfomance der Klassifizierung und stellen Sie die Ergebnisse graphisch dar (Darstellung der Projektionen, Partition Plot).

```r
# Splitten der Daten in Trainings- und Testdaten (80/20)
set.seed(42) # Für Reproduzierbarkeit

# Splitten der Daten
trainIndex <- sample(1:145, 0.8 * 145, replace = FALSE)
trainData <- pca_scores[ trainIndex,]
testData  <- pca_scores[-trainIndex,]

# Splitten der transformierten Daten
set.seed(42)
trainIndex <- sample(1:145, 0.8 * 145, replace = FALSE)
trainData_t <- pca_scores_trans[ trainIndex,]
testData_t <- pca_scores_trans[-trainIndex,]
```

Nachdem die Daten in Trainings- und Testdaten aufgeteilt wurden, wird die LDA durchgeführt.

```r
# LDA (Daten ohne Transformation)
lda_m1 <- lda(group ~ PC1 + PC2, data = trainData)
# LDA (Daten mit Transformation)
lda_m2 <- lda(group ~ PC1 + PC2, data = trainData_t)

# Zusammenfassung des Modells
lda_m1
```

```
## Call:
## lda(group ~ PC1 + PC2, data = trainData)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2586207 0.5431034 0.1982759
##
## Group means:
##                  PC1        PC2
## chemical -0.3381896  1.1937017
## normal   -0.9767583 -0.4443185
## overt     2.5343187 -0.1427610
##
## Coefficients of linear discriminants:
##             LD1         LD2
## PC1 -1.30381483  0.03586655
## PC2  0.00268035 -1.02766797
##
## Proportion of trace:
##    LD1    LD2
## 0.8594 0.1406
```

```r
lda_m2
```

```
## Call:
## lda(group ~ PC1 + PC2, data = trainData_t)
##
## Prior probabilities of groups:
##  chemical    normal     overt
## 0.2586207 0.5431034 0.1982759
##
## Group means:
##                 PC1      PC2
## chemical 0.8281705 1.524097
## normal   0.5017716 1.086904
## overt    1.6017178 1.172978
##
## Coefficients of linear discriminants:
##            LD1        LD2
## PC1 -4.6363864  0.2116696
## PC2  0.7408854 -3.3710881
##
## Proportion of trace:
##    LD1    LD2
## 0.9087 0.0913
```

```r
# Klasse der Trainingsdaten vorhersagen
lda_m1.p <- predict(lda_m1, newdata = trainData)
lda_m2.p <- predict(lda_m2, newdata = trainData_t)

head(lda_m1.p$posterior)
```

```
##         chemical       normal        overt
## 49  1.636720e-01 8.363191e-01 8.877390e-06
## 65  8.600618e-01 1.386255e-01 1.312720e-03
## 74  1.097520e-01 8.902387e-01 9.332582e-06
## 122 1.387930e-03 1.110215e-04 9.985010e-01
## 145 9.017279e-10 1.526906e-09 1.000000e+00
## 128 1.033351e-01 3.726833e-02 8.593966e-01
```

```r
head(lda_m1.p$class)
```

```
## [1] normal   chemical normal   overt    overt    overt
## Levels: chemical normal overt
```

```r
# Genauigkeit des Modells überprüfen
print(paste("Model 1 (PCs):",mean(lda_m1.p$class == trainData$group)))
```

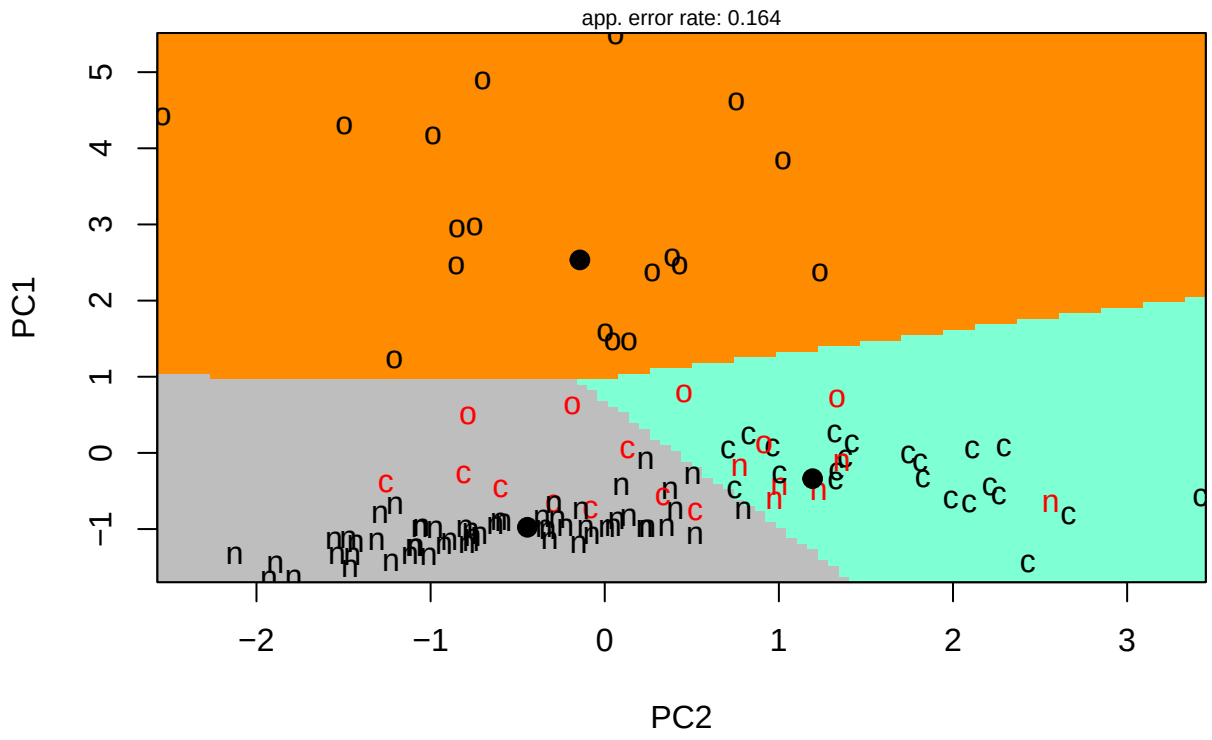
```
## [1] "Model 1 (PCs): 0.836206896551724"
```

```r
print(paste("Model 2 (trans. PCs):",mean(lda_m2.p$class == trainData_t$group)))
```

```
## [1] "Model 2 (trans. PCs): 0.827586206896552"
```

Da durch das Modell mit den log-transformierten Daten keine höhere Genauigkeit erreicht werden konnte, wird nur Model 1 für die weitere Analyse und Klassifikation der Testdaten herangezogen.

```r
#lda Trennung darstellen

# Group variable to factor
trainData$group <- as.factor(trainData$group)


# Partition Plot
partimat(group ~ PC1 + PC2,
         data = trainData, method = "lda",
         image.colors = c("aquamarine", "gray", "darkorange"))
```



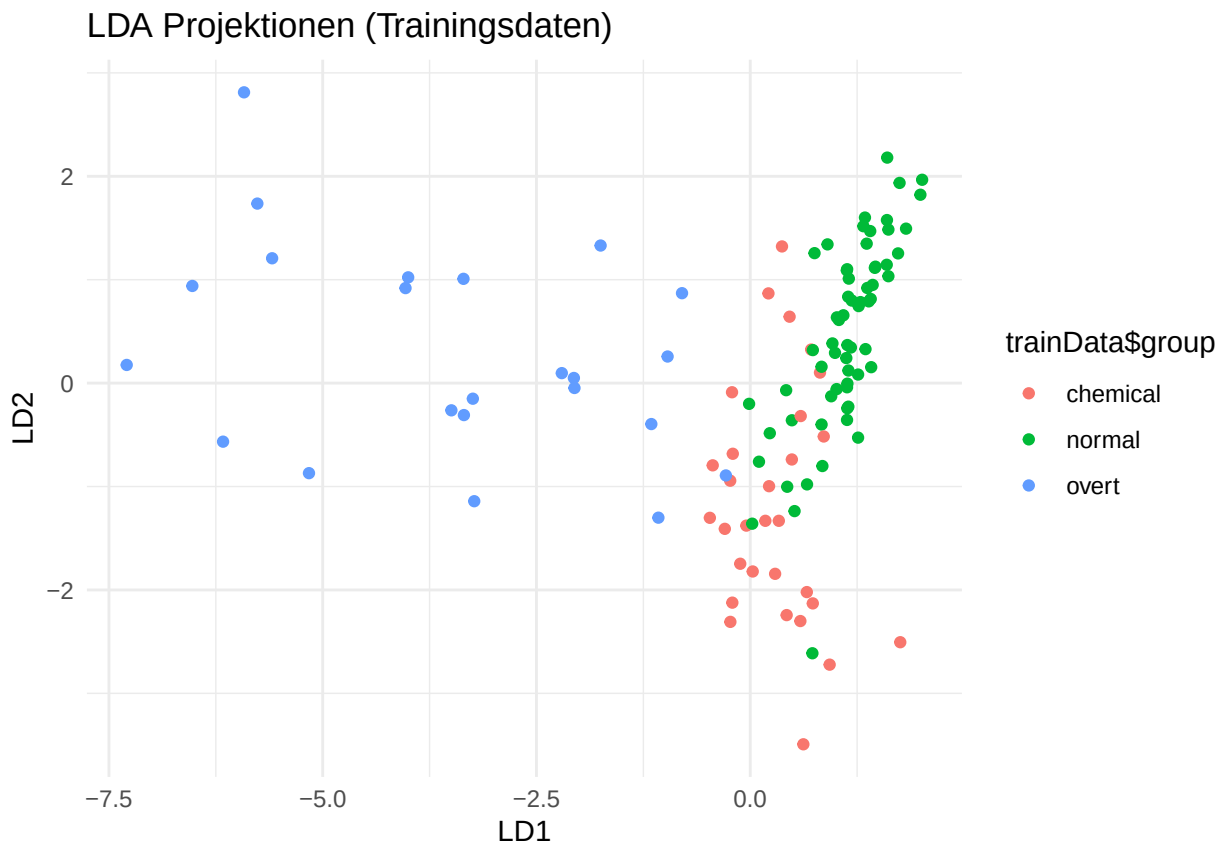

```
# Projektionen
lda_m1.p$x #LD1 und LD2
```

```
##             LD1          LD2
## 49   1.13206040 -0.040502523
## 65  -0.29721638 -1.409219122
## 74   1.12342623  0.242116497
## 122 -3.34864237 -0.309190659
## 145 -5.92018417  2.811819284
## 128 -2.05565534 -0.047103444
## 47   1.03639295  0.610613433
## 24   1.14815058  0.121813851
## 71   0.29092327 -1.844187740
## 100 -0.23254082 -2.308914786
## 89   0.62104089 -3.492729530
## 110  0.81557767  0.101853978
## 20   1.43162191  0.949580131
## 114 -5.76428899  1.735962377
## 111  0.21284000  0.867486563
## 41   0.83396415  0.158060417
## 135 -0.79788783  0.869675007
## 27   0.43237176 -1.002324424
## 36   1.13323551  1.101310977
## 101 -0.23432952 -0.943782029
## 95  -0.47213867 -1.302537377
## 109 -0.43901923 -0.795146654
```

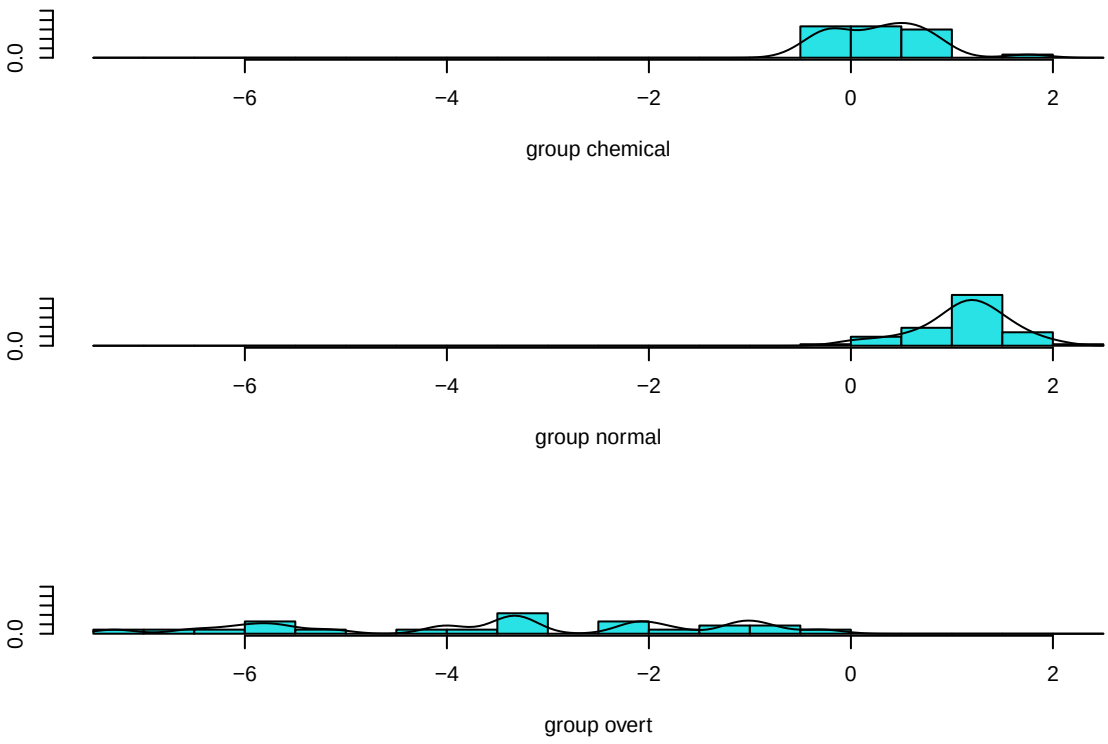
```
## 5    1.13199733 -0.356105830
## 84   0.10213384 -0.759988716
## 34   1.18738439  0.798359580
## 92   0.42579983 -2.243275821
## 104  0.66174648 -2.020823794
## 3    1.01336712  0.635931660
## 58   1.60197980  2.180570635
## 97  -0.20942874 -2.122887230
## 42   1.13616423 -0.005183853
## 138 -3.24398896 -0.150745687
## 30   1.46418206  1.125984718
## 43   1.26747772  0.744694711
## 15   1.82195599  1.493663767
## 22   1.74646364  1.935986468
## 117 -4.00027250  1.022740280
## 8    1.26121817 -0.527344327
## 127 -4.03162259  0.918882060
## 68  -0.01360652 -0.200632538
## 86   1.75444021 -2.505681853
## 18   1.59642607  1.144339601
## 120 -5.59160702  1.207252810
## 69   0.66422363 -0.979303284
## 4    1.13507470 -0.243215643
## 98   0.33424191 -1.331485675
## 50   1.41434557  0.154117361
## 88  -0.21299056 -0.088463809
## 87  -0.04554581 -1.377984657
## 141 -7.29213090  0.175703920
## 26   0.51819863 -1.237591369
## 6    0.90373808  1.341561377
## 94   0.22015517 -0.996759447
## 2    1.28870695  0.783422612
## 118 -3.49517320 -0.263238865
## 21   1.38578191  0.790329366
## 103 -0.20421356 -0.683264999
## 124 -1.15679676 -0.395934546
## 10   1.34146583  1.601023299
## 40   0.83417951 -0.400491062
## 123 -3.22781764 -1.142030178
## 33   2.01025799  1.966389461
## 96   0.71243212  0.324596949
## 73   1.08928275  0.655739909
## 29   1.01035085 -0.057899592
## 76   0.84386325 -0.801933760
## 9    0.96124399  0.383796377
## 35   1.40928676  0.813794830
## 16   1.26125905  0.082854332
## 102  0.58615219 -2.300930663
## 129 -5.16039089 -0.870736571
## 132 -3.35363138  1.008353342
## 119 -2.06252881  0.049624689
## 80   1.61573609  1.033741424
## 55   1.15274048  1.010306017
## 105  0.46021730  0.642176406
```

```
## 90   0.02901114 -1.821433104
## 57   1.40312570  1.470170395
## 115 -0.96645518  0.257280250
## 82   0.72674532 -2.612275426
## 13   1.13602266  0.367981163
## 53   1.12931208  1.090820786
## 54   1.32325508  1.516994589
## 19   1.61569640  1.484741624
## 32   0.99122798  0.292510182
## 77   0.48673690 -0.738490096
## 63   0.17669955 -1.331598111
## 81   0.75193577  1.256809487
## 17   1.14736347 -0.227745983
## 44   1.98999644  1.821606546
## 79   1.36170169  1.348502282
## 72   0.48708156 -0.359413202
## 112  0.37138386  1.321712152
## 48   1.14533872  0.835147325
## 108  0.59031678 -0.318649909
## 136 -0.28474205 -0.892864248
## 38   1.17722329  0.344316022
## 1    1.59871622  1.576665319
## 144 -6.16536328 -0.566296179
## 14   1.36903520  0.919976085
## 85   0.85955221 -0.516241883
## 137 -1.07428365 -1.301243856
## 60   0.22867927 -0.484403201
## 125 -2.20373623  0.096635119
## 142 -1.74996231  1.330427466
## 99   0.92872124 -2.722472874
## 25   1.72878218  1.254415714
## 61   0.41990444 -0.068637601
## 106 -0.11699981 -1.746710233
## 46   1.45744686  1.114665064
## 93   0.73163435 -2.130269631
## 64   0.94701126 -0.127297270
## 31   1.34791561  0.328666135
## 75   0.02287957 -1.359279210
## 113 -6.52460884  0.939410730
## 78   0.73207081  0.319209164
```

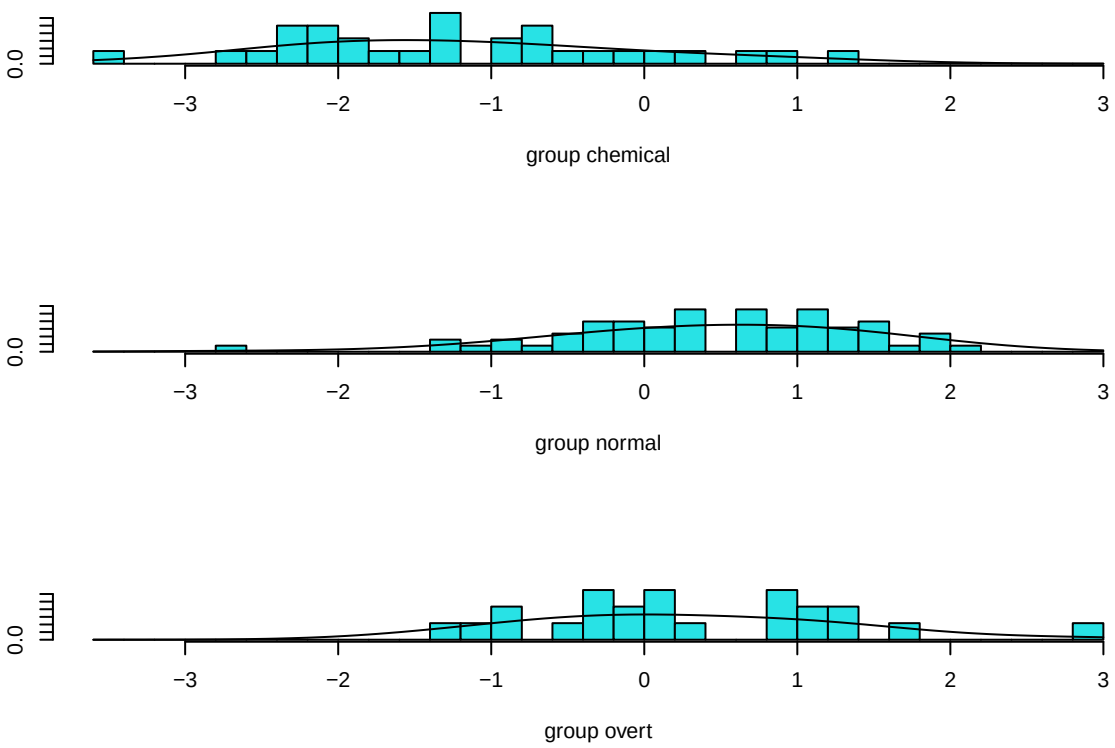
```r
# lda_m1.p$x als data.frame
lda_m1.p$x <- as.data.frame(lda_m1.p$x)

#projektionen graphisch darstellen
ggplot(lda_m1.p$x, aes(x = LD1, y = LD2, color = trainData$group)) +
  geom_point() +
  ggtitle("LDA Projektionen (Trainingsdaten)") +
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()
```




```
# Histogramme der Projektionen
ldahist(lda_m1.p$x[,1], g = trainData$group,type = "both")
```




```
ldahist(lda_m1.p$x[,2], g = trainData$group, type = "both")
```



Die Klassifikationsgenauigkeit von Modell 1 beträgt auf den Testdaten 75%. Somit performt das Modell auf den Testdaten schlechter als auf den Trainingsdaten, da diese zu 84% richtig klassifiziert wurden.

```
# Prediction on test data
lda_pred <- predict(lda_m1, newdata = testData)

# Check the head of the posterior probabilities and predicted classes
head(lda_pred$posterior)
```

```
##       chemical    normal        overt
## 7  0.092569688 0.9074213 9.047516e-06
## 11 0.041586557 0.9584126 8.385591e-07
## 12 0.006252636 0.9937470 4.099159e-07
## 23 0.078902834 0.9210918 5.375382e-06
## 28 0.455490758 0.5444578 5.139728e-05
## 37 0.351735379 0.6482582 6.398181e-06
```

```
head(lda_pred$class)
```

```
## [1] normal normal normal normal normal normal
## Levels: chemical normal overt
```

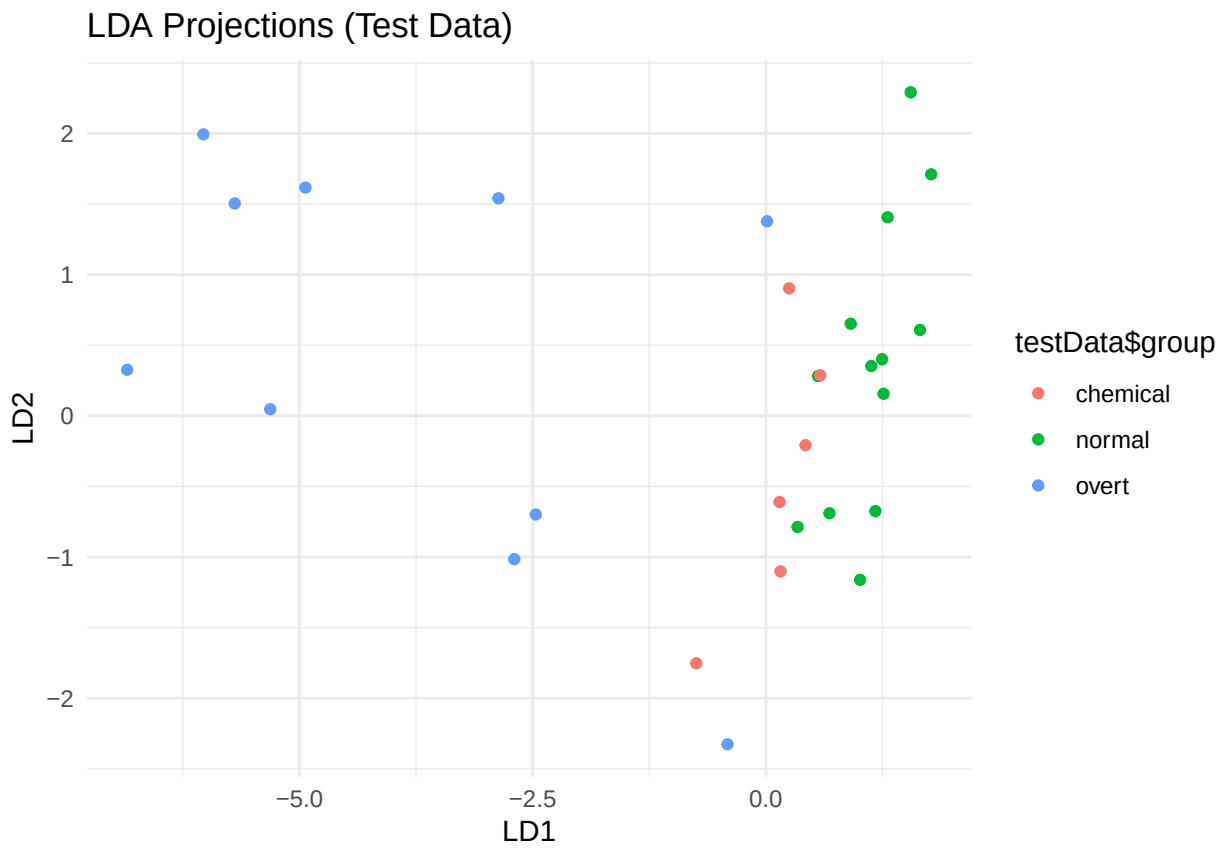
```
# Check the accuracy of the model
ac_lda <- mean(lda_pred$class == testData$group) # 75%
ac_lda
```

```
## [1] 0.7586207
```

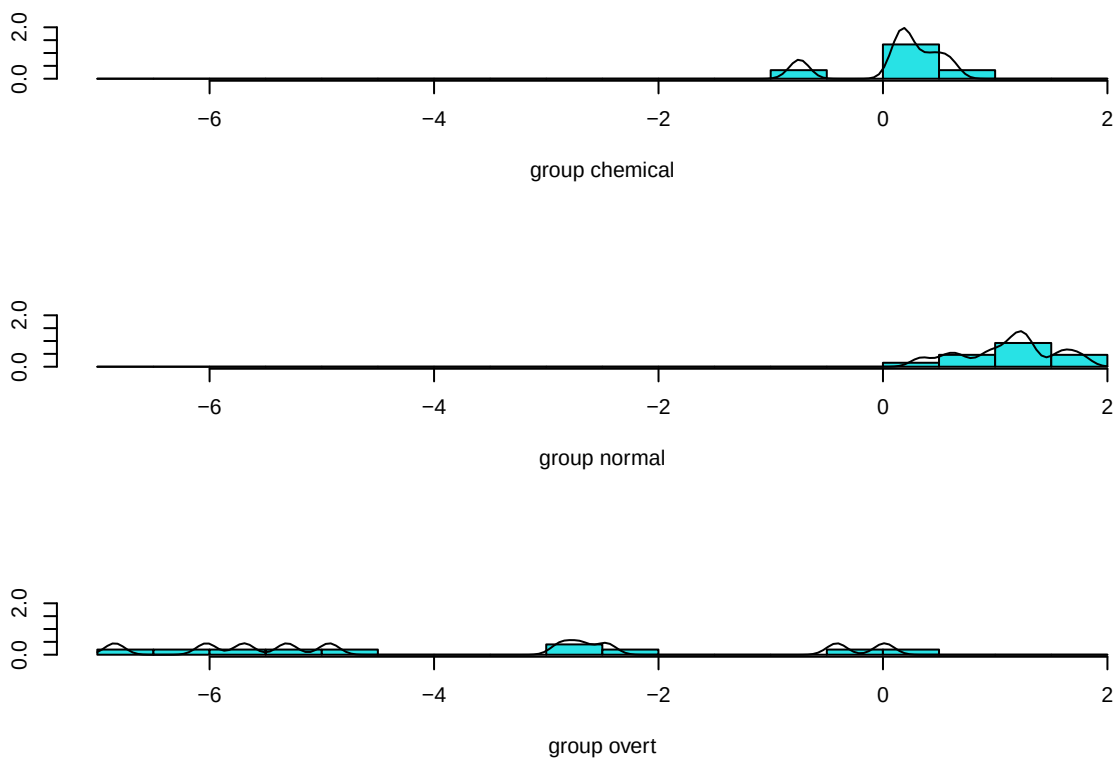
```
# Projections of the test data
lda_pred$x #LD1 and LD2
```

```
##             LD1         LD2
## 7    1.12992672  0.35292772
## 11   1.65130890  0.60771325
## 12   1.77125637  1.71081962
## 23   1.24502853  0.40072952
## 28   0.68072529 -0.69029608
## 37   1.17347548 -0.67529606
## 39   0.90916417  0.65214212
## 45   1.55209894  2.29125296
## 51   0.55842120  0.28183266
## 52   1.00910754 -1.16195903
## 56   1.26129348  0.15621588
## 59   0.58269295  0.28740853
## 62   0.14625365 -0.61079421
## 66   0.42526474 -0.20828330
## 67   0.33963421 -0.78726439
## 70   1.30477439  1.40633535
## 83   0.15694055 -1.10217049
## 91  -0.74578000 -1.75291909
## 107  0.24923501  0.90281159
## 116 -5.69279179  1.50413261
## 121 -2.69624465 -1.01540201
## 126 -4.93443160  1.61705912
## 130 -2.46646696 -0.69878455
## 131 -0.41150593 -2.32673717
## 133 -6.02896885  1.99353412
## 134  0.01270394  1.37778715
## 139 -6.84589513  0.32639633
## 140 -5.31206236  0.04672522
## 143 -2.86642882  1.54070666
```

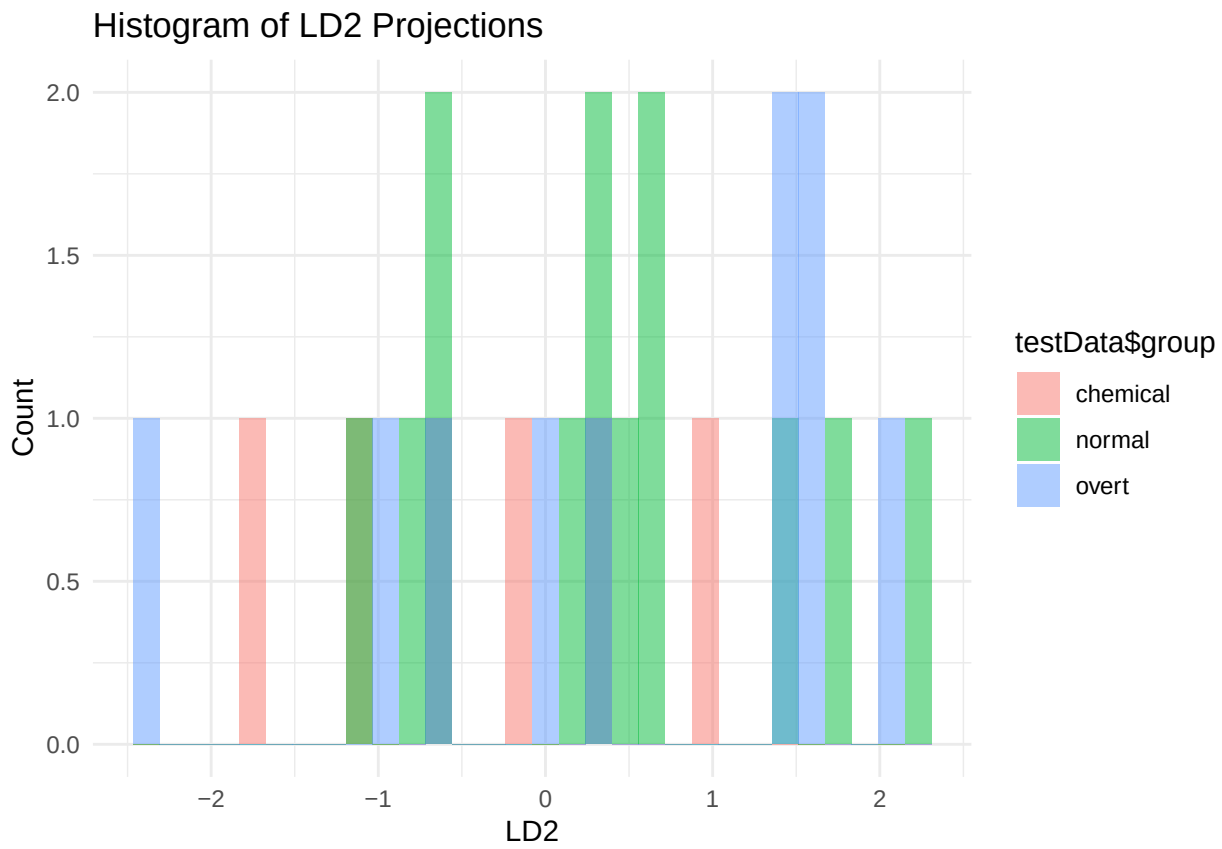
```
# Convert lda_m1.p$x to a data.frame
lda_pred$x <- as.data.frame(lda_pred$x)

# Graphical representation of the projections
ggplot(lda_pred$x, aes(x = LD1, y = LD2, color = testData$group)) +
  geom_point() +
  ggtitle("LDA Projections (Test Data)") +
  xlab("LD1") +
  ylab("LD2") +
  theme_minimal()
```




```
# Histograms of the projections
ldahist(lda_pred$x[,1], g = testData$group, type = "both")
```




```r
# ldahist(lda_pred$x[,2], g = testData$group, type = "both")

# The ldahist method is not suitable LDA2 because there are too few unique observations in the groups.
# Adjusting the bin width parameter h couldn't solve the problem.
# Therefore, we us ggplot instead.
ggplot(lda_pred$x, aes(x=LD2, fill=testData$group)) +
  geom_histogram(position="identity", alpha=0.5, bins=30) +
  theme_minimal() +
  xlab("LD2") +
  ylab("Count") +
  ggtitle("Histogram of LD2 Projections")
```



iv) Vergleichen Sie unterschiedliche Varianten der Diskriminanzanalyse (QDA, MDA, FDA) hinsichtlich ihrer Klassifikationsgenauigkeit.

```
# QDA
m_qda <- qda(group ~ PC1 + PC2, data = trainData)
m_qda.p <- predict(m_qda, newdata = testData)
ac_qda <- mean(m_qda.p$class == testData$group)

# MDA
library(mda)
m_mda <- mda(group ~ PC1 + PC2, data = trainData)
m_mda.p <- predict(m_mda, newdata = testData)
ac_mda <- mean(m_mda.p == testData$group)

# FDA
m_fda <- fda(group ~ PC1 + PC2, data = trainData)
m_fda.p <- predict(m_fda, newdata = testData)
ac_fda <- mean(m_fda.p == testData$group)

# RDA
m_rda <- rda(group ~ PC1 + PC2, data = trainData)
m_rda.p <- predict(m_rda, newdata = testData)
ac_rda <- mean(m_rda.p$class == testData$group)

# Table
```

```r
results <- data.frame(Model = c("QDA", "MDA", "FDA", "RDA"),
                      Accuracy = c(ac_qda, ac_mda, ac_fda, ac_rda))
print(results)
```

```
##   Model  Accuracy
## 1   QDA 0.7931034
## 2   MDA 0.8275862
## 3   FDA 0.7586207
## 4   RDA 0.7586207
```

Nur die FDA erreicht eine Klassifikationsgenauigkeit von etwa 83% auf den Testdaten wie die LDA. Alle anderen Modelle sind unterlegen.

Bonus: Iris Datensatz

In diesem Abschnitt wird die Diskriminanzanalyse auf dem Iris-Datensatz durchgeführt. Der Iris-Datensatz enthält 150 Beobachtungen von Iris-Blumen. Jede Beobachtung enthält die Länge und Breite des Kelchblattes und des Kronblattes. Die Blumen gehören zu einer von drei Arten: Setosa, Versicolor oder Virginica.

Ziel der Aufgabe ist es, für einen neuen vermessenen Datensatz von Iris-Blumen die Art der Blume vorherzusagen.

Dafür wurden folgende Daten erhoben:

```r
new_data <- data.frame(Sepal.Length = c(7.2, 6.5),
                       Sepal.Width = c(3.2, 2.8),
                       Petal.Length = c(5.5, 5),
                       Petal.Width = c(2, 2))
new_data
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          7.2         3.2          5.5           2
## 2          6.5         2.8          5.0           2
```

Verwenden Sie ein auf LDA basierendes Klassifizierungsmodell um die vorliegende Iris Klasse zu bestimmen. Welche Spezies liegt hier vor? Ist das Ergebnis plausibel?

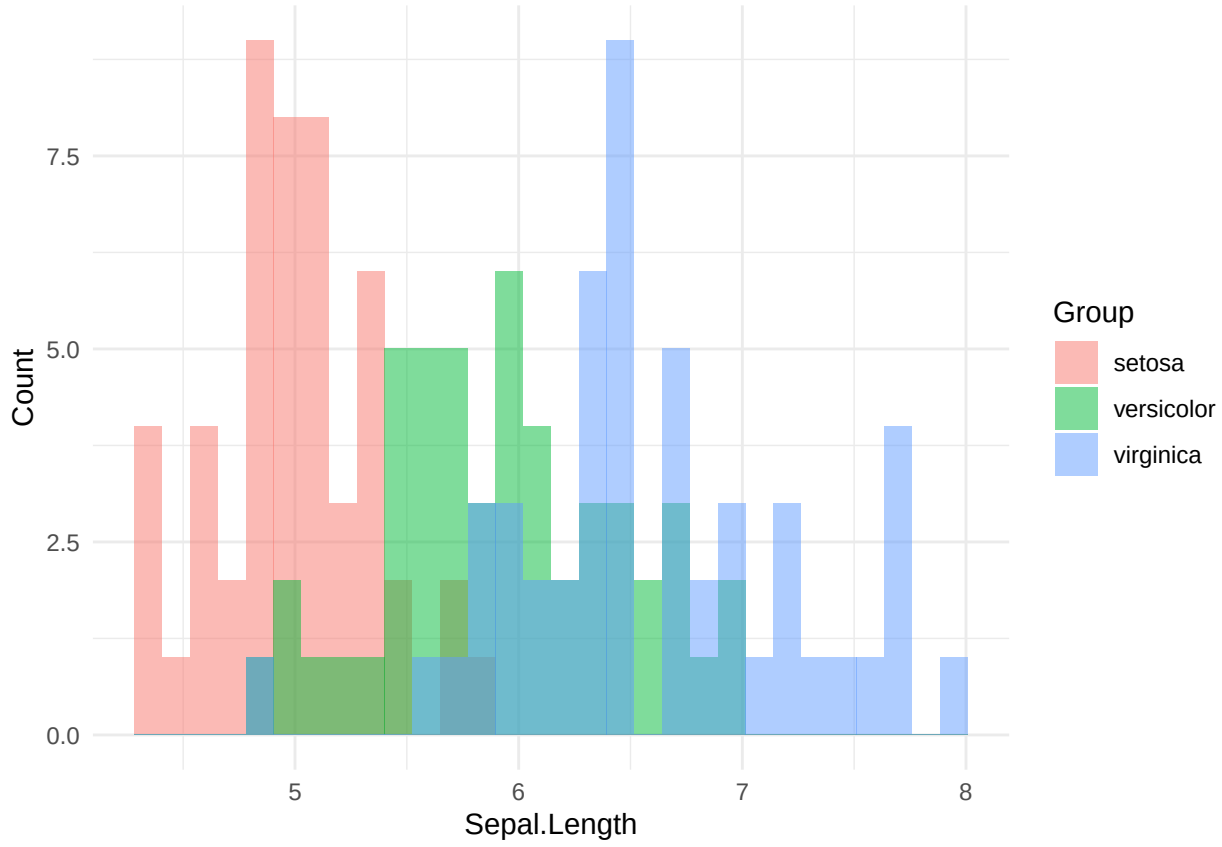
```r
# Iris data
data(iris)

str(iris)
```

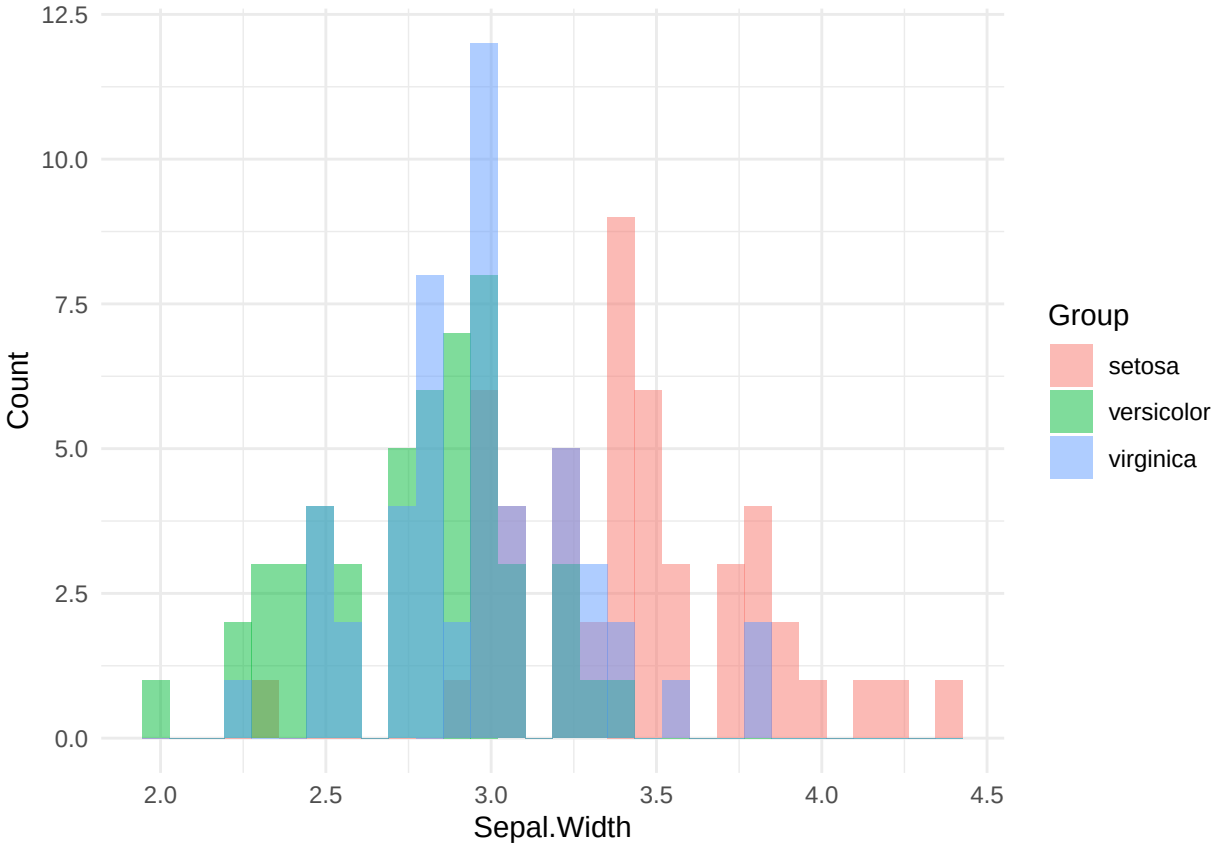
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

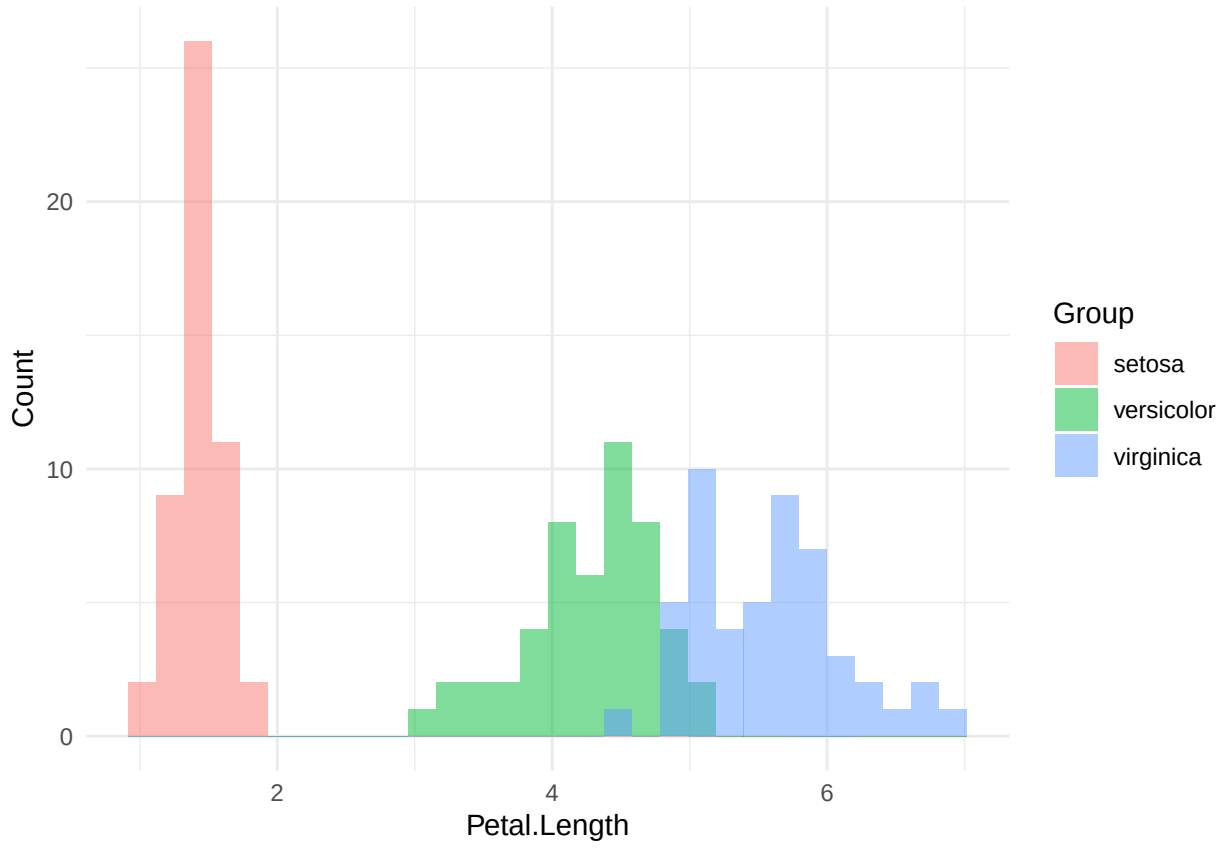
```r
# Species in group umbenennen
iris$group <- as.factor(iris$Species)

# Tabelle
aggregate(cbind(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) ~ Species, data = iris, mean)
```

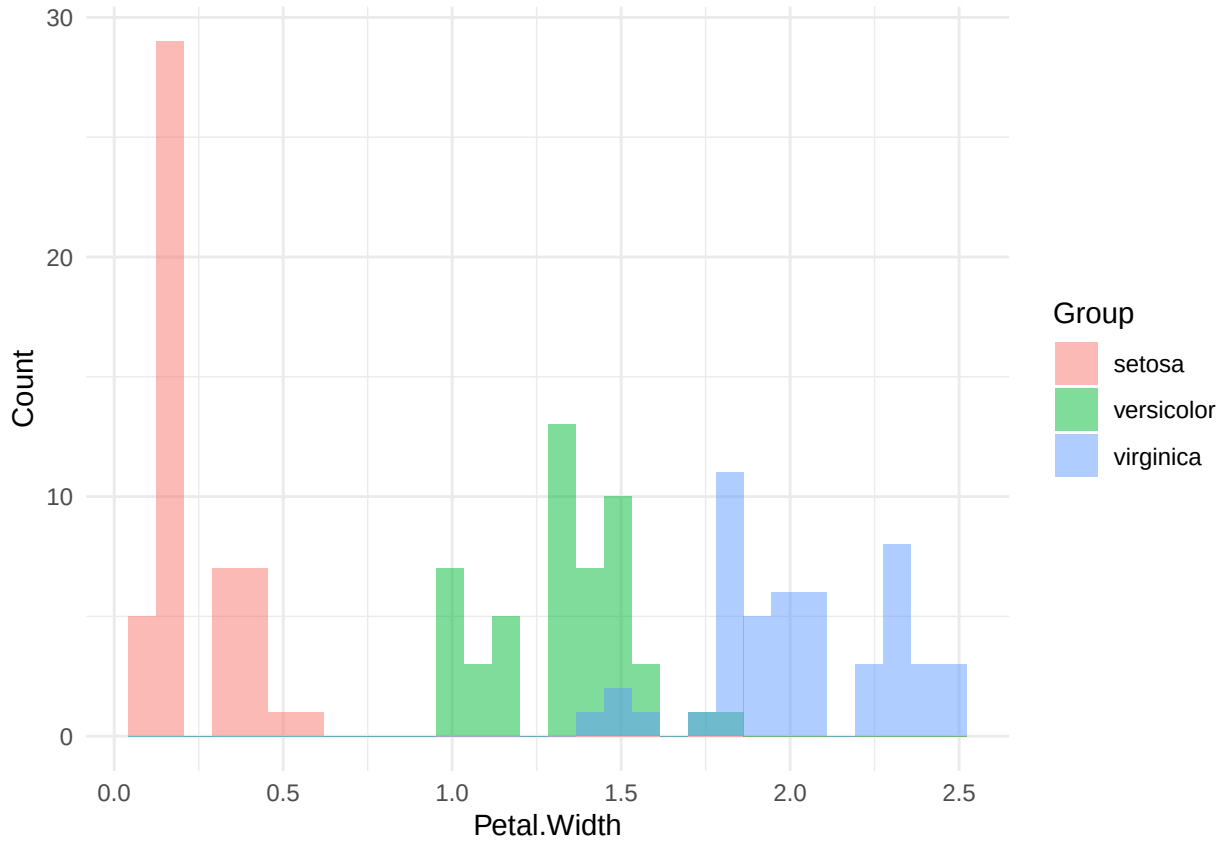
```
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

```r
# Histogramme
create_histograms(iris)
```

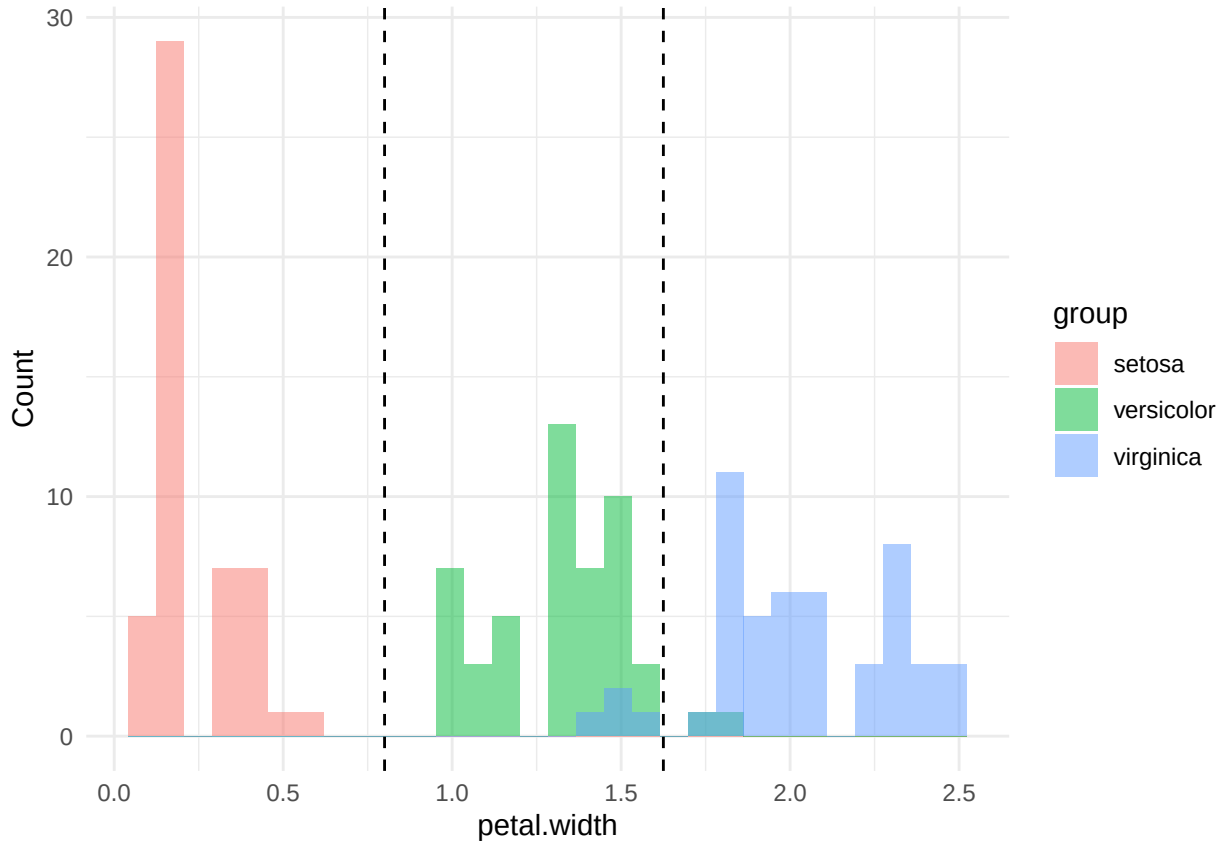



12.5
10.0
7.5
5.0
2.5
0.0
Count
2.0
2.5
3.0
3.5
4.0
4.5
Sepal.Width
Group
setosa
versicolor
virginica


Count
20
10
0
2
4
6
Petal.Length
Group
setosa
versicolor
virginica




```r
# Explorativer Ansatz zur Trennung der Gruppen
# Grenzen anhand des Prädiktors 'petal.width'
grenzen <- c(0.8,1.625)

ggplot(iris, aes(x = Petal.Width, fill = group)) +
    geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
    labs(x = "petal.width", y = "Count", fill = "group") +
    # Grenze einfügen
    geom_vline(xintercept = grenzen, linetype = "dashed", color = "black") +
    theme_minimal()
```




```
# Explorative Klassifizierung der Gruppen
iris$group_K1<- ifelse(iris$Petal.Width < grenzen[1], "setosa",
                       ifelse(iris$Petal.Width < grenzen[2], "versicolor", "virginica"))

iris$group_K1 <- as.factor(iris$group_K1)

# Genauigkeit der Klassifizierung
mean(as.factor(iris$group_K1) == iris$group)
```

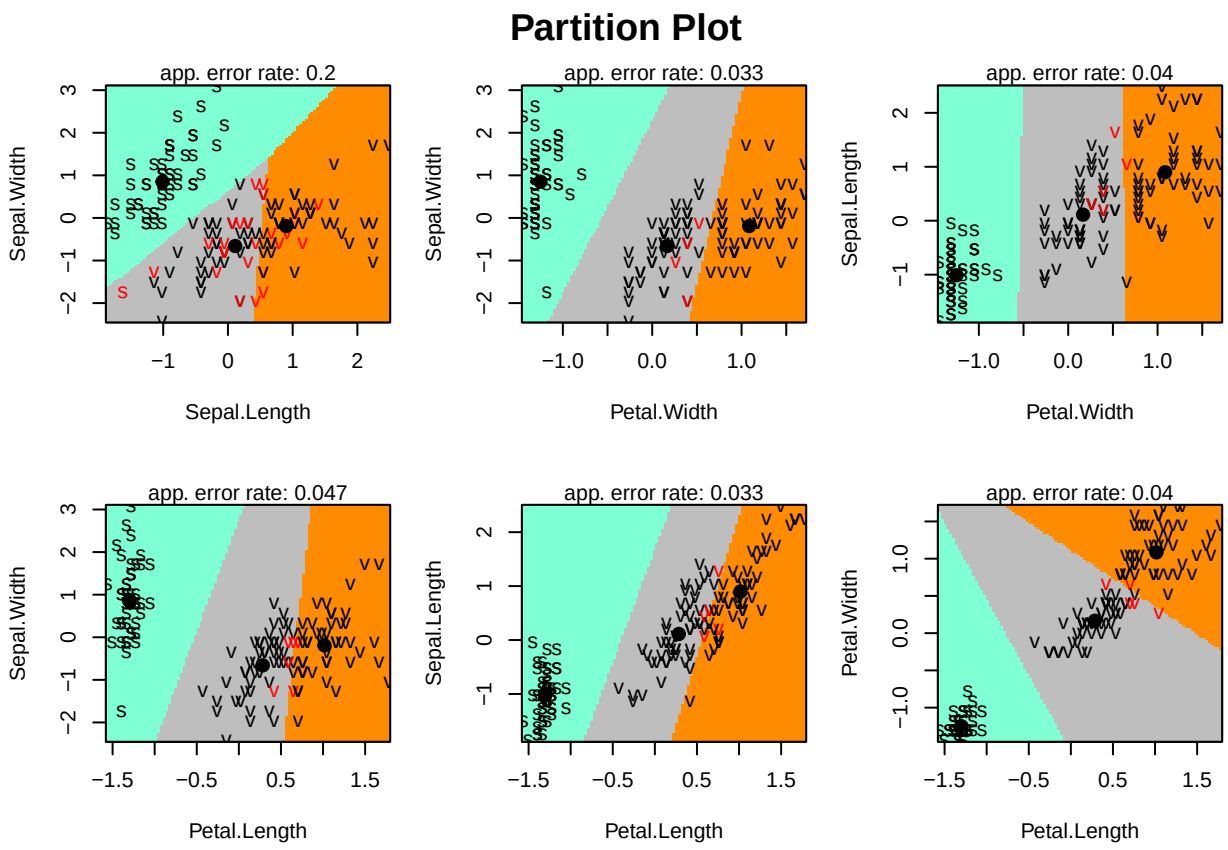
```
## [1] 0.96
```

```
# Explorative Klassifizierung der neuen Daten
group_K1 <- ifelse(new_data$Petal.Width < grenzen[1], "setosa",
                           ifelse(new_data$Petal.Width < grenzen[2], "versicolor", "virginica"))



# standardize and split the data
iris_pp <- preProcess(iris[,1:4], method = c("center", "scale"))
iris_std <- predict(iris_pp, iris[,1:4])
iris_std$group <- iris$group

# Train-Test-Split
set.seed(42)
trainIndex <- sample(1:nrow(iris), 0.8 * nrow(iris))
```

```r
trainData <- iris_std[trainIndex,]
testData <- iris_std[-trainIndex,]

# LDA
m_lda <- lda(group ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = trainData)
m_lda
```

```
## Call:
## lda(group ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
##     data = trainData)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3416667  0.3250000  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa      -1.03021897   0.8029613   -1.3025920  -1.2534554
## versicolor   0.05914305  -0.6962851    0.2634552   0.1657064
## virginica    0.92887107  -0.2061030    1.0236231   1.0766546
##
## Coefficients of linear discriminants:
##                      LD1        LD2
## Sepal.Length  0.8464478 -0.1405916
## Sepal.Width   0.5421108 -0.9192285
## Petal.Length -3.9916320  1.4922810
## Petal.Width  -2.2117046 -1.8848725
##
## Proportion of trace:
##    LD1    LD2
## 0.9924 0.0076
```

```r
# Partition Plot
library(klaR)
partimat(group~Sepal.Width+Sepal.Length+Petal.Width+Petal.Length,
         data = iris_std,method="lda",
         image.colors =c("aquamarine","gray","darkorange"))
```




```
# Klassifikation der Testdaten
m_lda.p <- predict(m_lda, newdata = testData)
head(m_lda.p$posterior)
```

```
##    setosa   versicolor    virginica
## 7       1 6.775719e-18 8.964411e-37
## 11      1 4.024961e-23 5.492232e-44
## 12      1 6.419658e-18 4.599576e-37
## 19      1 5.354076e-22 3.815704e-42
## 23      1 6.545836e-24 4.548179e-45
## 28      1 4.044905e-21 2.040512e-41
```

```
head(m_lda.p$class)
```

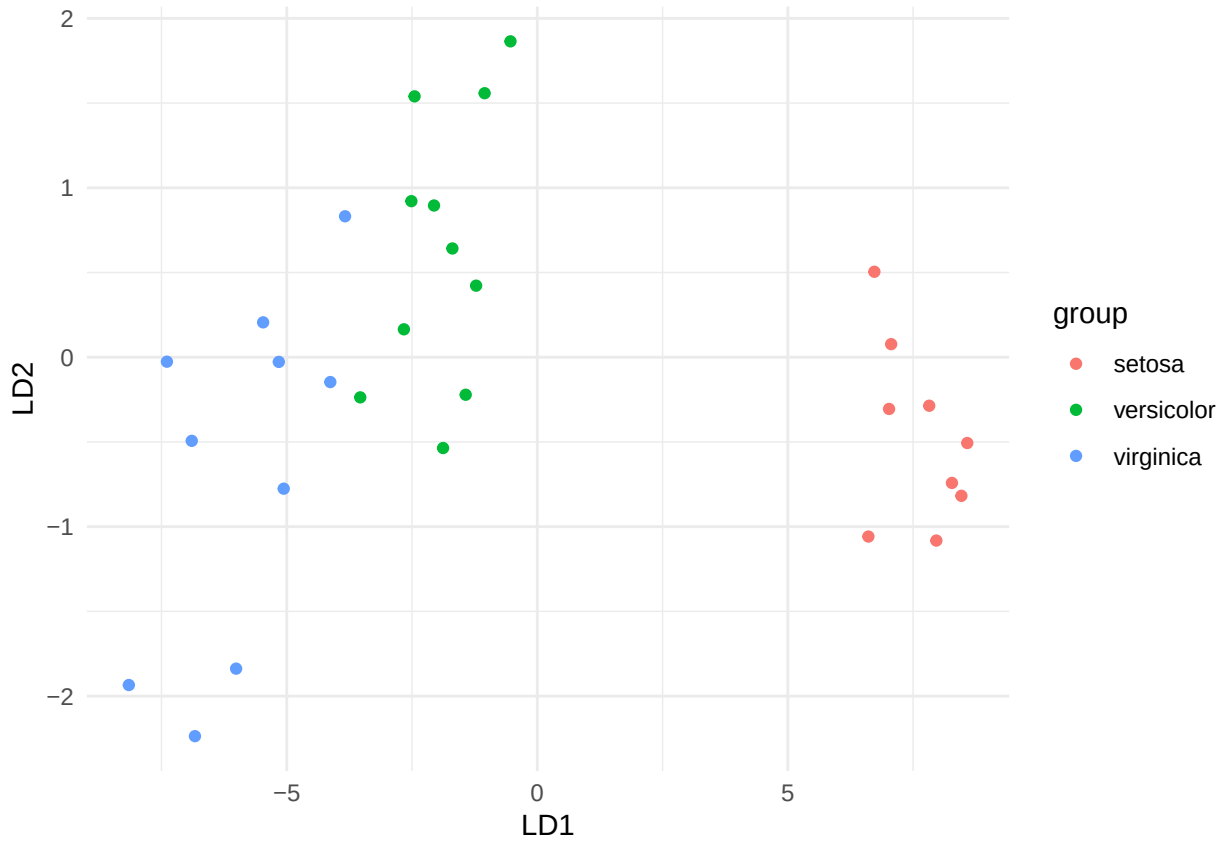
```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
# Genauigkeit
ac_lda <- mean(m_lda.p$class == testData$group)

# Standardisierung der neuen Daten
new_data_std <- predict(iris_pp, new_data)

# Vorhersage der neuen Daten
```

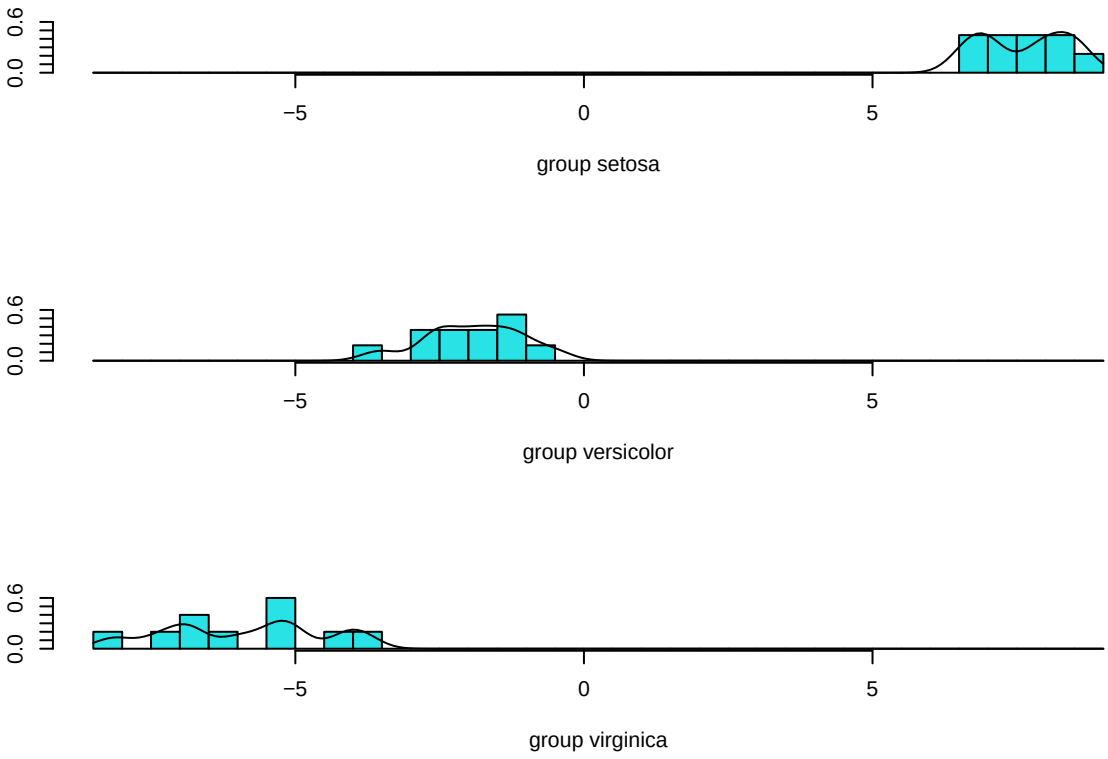
```r
new_data$group_Klda <- predict(m_lda, newdata = new_data)$class
iris_pred <- predict(m_lda, newdata = new_data_std)

# Add highest Probability of class membership to new_data
new_data$prob <- apply(iris_pred$posterior, 1, max)

# Projektionen der Testdaten
m_lda.p$x #LD1 und LD2
```

```
##           LD1         LD2
## 7    7.020306 -0.30531332
## 11   8.275234 -0.74201479
## 12   7.062672  0.07707998
## 19   7.963876 -1.08205965
## 23   8.463683 -0.81796253
## 28   7.822043 -0.28626436
## 37   8.580936 -0.50626771
## 45   6.608165 -1.05840287
## 46   6.727244  0.50431771
## 51  -1.428770 -0.22146584
## 52  -1.880015 -0.53594592
## 56  -2.512737  0.92105279
## 59  -1.694500  0.64188536
## 70  -1.051065  1.55807946
## 75  -1.220590  0.42223932
## 78  -3.533009 -0.23697887
## 79  -2.662021  0.16465803
## 82  -0.535268  1.86416772
## 91  -2.449652  1.53955047
## 95  -2.060983  0.89532535
## 101 -8.151203 -1.93466599
## 106 -7.391533 -0.02636203
## 112 -5.471463  0.20568647
## 116 -6.010224 -1.83792395
## 117 -5.158191 -0.02763265
## 127 -4.130785 -0.14664344
## 133 -6.895916 -0.49345220
## 134 -3.836437  0.83182532
## 137 -6.832202 -2.23641829
## 148 -5.060160 -0.77579824
```

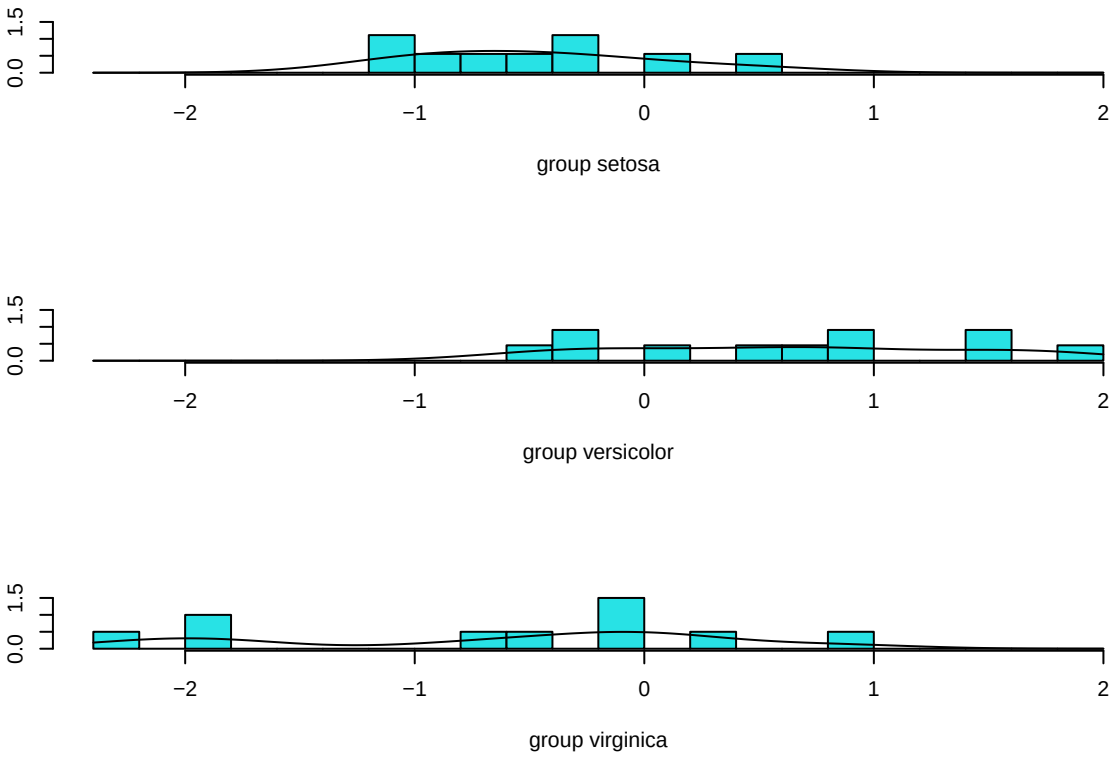
```r
# lda_m1.p$x als data.frame
m_lda.p$x <- as.data.frame(m_lda.p$x)

# projektionen graphisch darstellen
ggplot(m_lda.p$x, aes(x = LD1, y = LD2, color = testData$group)) +
    geom_point() +
    labs(x = "LD1", y = "LD2", color = "group") +
    theme_minimal()
```




```
# Histogramme der Projektionen
ldahist(m_lda.p$x[,1], g = testData$group,type = "both")
```




```
ldahist(m_lda.p$x[,2], g = testData$group, type = "both")
```




```
# Ergebnisse
print(new_data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width group_Klda      prob
## 1          7.2         3.2          5.5           2  virginica 0.9941781
## 2          6.5         2.8          5.0           2  virginica 0.9922348
```

Antwort: Die vorliegenden Daten entsprechen der Spezies ‘virginica’. Das Ergebnis ist plausibel, da die Wahrscheinlichkeit für die Spezies ‘virginica’ bei beiden Beobachtungen über 99% liegt. Die Genauigkeit der Klassifikation auf dem Testdatensatz beträgt über 96%.