prostate

Behrooz Filzadeh

2025-05-14

1. Analyse des Prostata-Datensatzes

a. Beschreibung des Datensatzes Der Datensatz prostate.data kommt von einer Studie von Stamey und Kollegen aus dem Jahr 1989. Es ging um Prostatakrebs. Die wichtigste Frage in diesem Experiment war: Wie hängen der PSA-Wert (Prostata-spezifisches Antigen) und andere klinische Messungen zusammen? Das war bei Männern, die bald eine Operation bekommen sollten, wo die Prostata entfernt wird (radikale Prostatektomie). Oft wollte man den Logarithmus vom PSA-Wert (lpsa) vorhersagen, mit Hilfe von diesen anderen Messungen. So ist der Datensatz aufgebaut: Beobachtungen: 97 Männer. Variablen: 10 Stück. lcavol: log(Krebsvolumen) – Das ist ein Prädiktor. lweight: log(Prostatagewicht) – Auch ein Prädiktor. age: Alter vom Patient – Prädiktor. lbph: log(gutartige Prostatavergrößerung Menge) – Prädiktor. svi: Befall der Samenbläschen (binär: 1 wenn ja, 0 wenn nein) – Prädiktor. lcp: log(Kapselpenetration) – Prädiktor. gleason: Gleason-Score (ordinal, pathologischer Grad) – Prädiktor. pgg45: Prozent Gleason-Scores 4 oder 5 – Prädiktor. lpsa: log(Prostata-spezifisches Antigen) – Das ist die Zielvariable, die wir vorhersagen wollen. train: Ein logischer Vektor. Er sagt, ob die Beobachtung Teil von einem vorher festgelegten Trainingsdatensatz (TRUE) oder Testdatensatz (FALSE) war. Das braucht man oft, um das Modell zu prüfen. Die Beobachtungseinheit sind also Männer, bei denen Prostatakrebs gefunden wurde, bevor ihre Prostata operativ entfernt wurde. Die Daten sind sehr nützlich. Man kann verstehen, welche Faktoren mit dem PSA-Wert zusammenhängen. Und man kann Modelle bauen, um vorherzusagen, wie der Krebs sich entwickelt oder wie hoch der PSA-Wert sein wird. Diese Daten werden oft in Statistik und Machine Learning Büchern als Beispiel für Regression benutzt. Wichtigste Quelle (Referenz): Stamey, T. A., Kabalin, J. N., McNeal, J. E., Johnstone, I. M., Freiha, F., Redwine, E. A., & Yang, N. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients. The Journal of urology, 141(4), 1076-1083. Der Datensatz ist auch sehr bekannt, weil er in “The Elements of Statistical Learning” von Hastie, Tibshirani und Friedman analysiert wird. Ich habe nur menschliche Quellen benutzt, keine KI.

b. Code-Aufgabe: LAB Prostate LinReg 0.R In diesem Teil schauen wir uns das R-Skript genau an. Ich zeige die einzelnen Blöcke, schreibe Kommentare dazu und erkläre, was passiert.

```
#-------------------------------------------------------------------------------
# Block 1: Laden der wichtigen Bibliotheken (Pakete)
#-------------------------------------------------------------------------------
# Dieser Block lädt verschiedene R-Pakete. Die brauchen wir für die Analyse.
# Jedes Paket hat spezielle Funktionen:
# - data.table: Damit kann man Daten schnell bearbeiten und zusammenfassen.
# - ggplot2: Für schöne und anpassbare Diagramme.
# - corrplot: Um Korrelationsmatrizen als Bild zu zeigen. (Wird hier nicht direkt benutzt, aber
geladen.)
# - GGally: Macht ggplot2 noch besser, z.B. mit ggpairs für paarweise Diagramme.
# - psych: Hat Funktionen für beschreibende Statistiken (z.B. describe).
# - DataExplorer: Für automatische erste Datenanalyse. (Wird hier nicht viel benutzt.)
# - rlang: Hilft beim Arbeiten mit R-Sprachobjekten, ggplot2 benutzt das.
#-------------------------------------------------------------------------------
library(data.table)
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(DataExplorer)
library(rlang)
```

```
##
## Attaching package: 'rlang'
```

```
## The following object is masked from 'package:data.table':
##
##     :=
```

Erklärung von Block 1: Dieser erste Schritt ist sehr wichtig in R. Er macht die Funktionen von diesen Paketen für uns bereit. Zum Beispiel data.table für schnelle Datenarbeit, ggplot2 und GGally für Bilder, und psych für zusammenfassende Zahlen.

```
#-------------------------------------------------------------------------------
# Block 2: Daten laden und erste Inspektion
#-------------------------------------------------------------------------------
# Dieser Block liest den Prostatakrebs-Datensatz aus einer Datei ein.
# Dann macht er daraus ein data.table-Objekt, damit man besser damit arbeiten kann.
# Und dann schauen wir uns die Daten das erste Mal an.
# - read.table: Liest die Daten aus einer Datei, wo die Werte mit Leerzeichen getrennt sind.
# - as.data.table: Macht aus dem data.frame ein data.table.
# - str: Zeigt, wie die Daten aufgebaut sind (Typen der Variablen, erste Werte).
# - psych::describe: Gibt uns viele Details für die ersten 9 Spalten
#   (Prädiktoren und Zielvariable, ohne die 'train'-Spalte).
#-------------------------------------------------------------------------------
prostateData <- read.table(file="prostate_data.csv")
prostateData <- as.data.table(prostateData)
str(prostateData)
```

```
## Classes 'data.table' and 'data.frame':   97 obs. of  10 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
##  $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
psych::describe(prostateData[,1:9])
```

```
##         vars  n  mean    sd median trimmed   mad   min    max  range  skew
## lcavol     1 97  1.35  1.18   1.45    1.39  1.28 -1.35   3.82   5.17 -0.24
## lweight    2 97  3.63  0.43   3.62    3.62  0.38  2.37   4.78   2.41  0.06
## age        3 97 63.87  7.45  65.00   64.47  5.93 41.00  79.00  38.00 -0.80
## lbph       4 97  0.10  1.45   0.30    0.03  2.50 -1.39   2.33   3.71  0.13
## svi        5 97  0.22  0.41   0.00    0.15  0.00  0.00   1.00   1.00  1.36
## lcp        6 97 -0.18  1.40  -0.80   -0.34  0.87 -1.39   2.90   4.29  0.71
## gleason    7 97  6.75  0.72   7.00    6.67  0.00  6.00   9.00   3.00  1.22
## pgg45      8 97 24.38 28.20  15.00   20.57 22.24  0.00 100.00 100.00  0.94
## lpsa       9 97  2.48  1.15   2.59    2.48  1.15 -0.43   5.58   6.01  0.00
##         kurtosis   se
## lcavol     -0.60 0.12
## lweight     0.36 0.04
## age         0.96 0.76
## lbph       -1.75 0.15
## svi        -0.16 0.04
## lcp        -1.01 0.14
## gleason     2.36 0.07
## pgg45      -0.37 2.86
## lpsa        0.43 0.12
```

Erklärung von Block 2: Die Daten sind jetzt geladen und ein data.table. str(prostateData) zeigt uns: 97 Beobachtungen und 10 Variablen. Alle Variablen sind Zahlen (num), außer svi (Befall der Samenbläschen, ist eine ganze Zahl, aber wie ein binärer Faktor) und train (logisch). gleason ist ein ganzer Zahlenwert. psych::describe(prostateData[,1:9]) gibt uns eine gute Zusammenfassung: vars: Nummer der Variable n: Anzahl der Beobachtungen ohne fehlende Werte (hier immer 97) mean, sd: Mittelwert und Standardabweichung median: Median (der mittlere Wert) trimmed: getrimmter Mittelwert (nicht so empfindlich gegen Ausreißer) mad: mediane absolute Abweichung (auch ein robustes Maß für Streuung) min, max, range: kleinster Wert, größter Wert, Spannweite skew: Schiefe (wie symmetrisch ist die Verteilung). Positive Werte (z.B. lcavol, lweight, lbph, lcp, pgg45, lpsa) heißen, die Verteilung ist rechtsschief. age ist ein bisschen linksschief. kurtosis: Kurtosis (wie “spitz” oder “flach” ist die Verteilung, oder wie stark sind die Enden). Positive Werte deuten auf stärkere Enden als bei einer Normalverteilung hin. se: Standardfehler des Mittelwerts. Diese erste Beschreibung hilft uns zu verstehen, wie groß die Werte sind und wie sie verteilt sind. Man sieht auch Probleme, wie Schiefe oder Ausreißer. Manchmal muss man dann was machen (z.B. transformieren, wie hier schon bei lcavol, lpsa – das ‘l’ am Anfang steht für Logarithmus).

```
#-------------------------------------------------------------------------------
# Block 3: Variationskoeffizient (CV) berechnen
#-------------------------------------------------------------------------------
# Dieser Block macht eine Funktion, um den Variationskoeffizienten (CV) zu berechnen.
# Dann wendet er die Funktion auf jede Spalte im Datensatz an.
# CV = (Standardabweichung / Mittelwert) * 100. Das ist ein Maß für relative Variabilität.
# Die Funktion prüft, ob die Eingabe numerisch ist und alle Werte positiv sind,
# weil man CV meistens für Daten auf einer Verhältnisskala mit positiven Werten benutzt.
#-------------------------------------------------------------------------------
calculate_cv <- function(x) {
  if (is.numeric(x) && min(x, na.rm = TRUE) > 0) {
    return((sd(x, na.rm = TRUE) / mean(x, na.rm = TRUE)) * 100)
  } else {
    return(NA) # Gibt NA zurück, wenn nicht numerisch oder nicht-positive Werte drin sind
  }
}

cv_values <- sapply(prostateData, calculate_cv)
print("Variationskoeffizient (%):")
```

```
## [1] "Variationskoeffizient (%):"
```

```
print(cv_values)
```

```
##   lcavol  lweight      age     lbph      svi      lcp  gleason    pgg45
##       NA 11.80540 11.65741       NA       NA       NA 10.69420       NA
##     lpsa    train
##       NA       NA
```

Erklärung von Block 3: Der Variationskoeffizient (CV) gibt uns ein standardisiertes Maß für die Streuung. lcavol, lweight, lbph, lcp, pgg45, lpsa haben CVs zwischen ungefähr 40% und 80%. lcavol hat den höchsten CV (80.3%), das heißt eine hohe relative Variabilität. age hat einen sehr niedrigen CV (11.4%), das heißt relativ geringe Variabilität um den Mittelwert im Vergleich zu anderen Variablen. svi und gleason geben NA zurück. Bei svi ist das so, weil es den Wert 0 enthält, das verletzt die Bedingung min(x) > 0. Bei gleason, obwohl es numerisch ist, ist es ein ordinaler Score, und CV ist da vielleicht nicht die beste Statistik. train gibt NA zurück, weil es logisch ist, nicht numerisch. Hohe CV-Werte bedeuten, dass die Standardabweichung im Verhältnis zum Mittelwert groß ist für diese Variablen.

```
#-------------------------------------------------------------------------------
# Block 4: Kolmogorov-Smirnov (KS) Test auf Normalverteilung
#-------------------------------------------------------------------------------
# Dieser Block macht eine Funktion für einen einseitigen Kolmogorov-Smirnov-Test.
# Der Test prüft, ob die Verteilung einer Variable sich stark von einer Normalverteilung
# unterscheidet. Mittelwert und Standardabweichung für diese Normalverteilung
# werden aus der Stichprobe geschätzt.
# Der Test wird auf jede Spalte des Datensatzes angewendet.
#-------------------------------------------------------------------------------
ks_normal_test <- function(x) {
  if (is.numeric(x)) {
    mu <- mean(x, na.rm = TRUE)
    sigma <- sd(x, na.rm = TRUE)
    ks_result <- ks.test(x, "pnorm", mean = mu, sd = sigma)
    return(list(statistic = ks_result$statistic, p_value = ks_result$p.value))
  } else {
    return(list(statistic = NA, p_value = NA)) # Gibt NA zurück, wenn nicht numerisch
  }
}

ks_results_list <- lapply(prostateData, ks_normal_test)
```

```
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
## Warning in ks.test.default(x, "pnorm", mean = mu, sd = sigma): ties should not
## be present for the one-sample Kolmogorov-Smirnov test
```

```
ks_results_df <- do.call(rbind, lapply(ks_results_list, data.frame))
print("Kolmogorov-Smirnov Test Ergebnisse:")
```

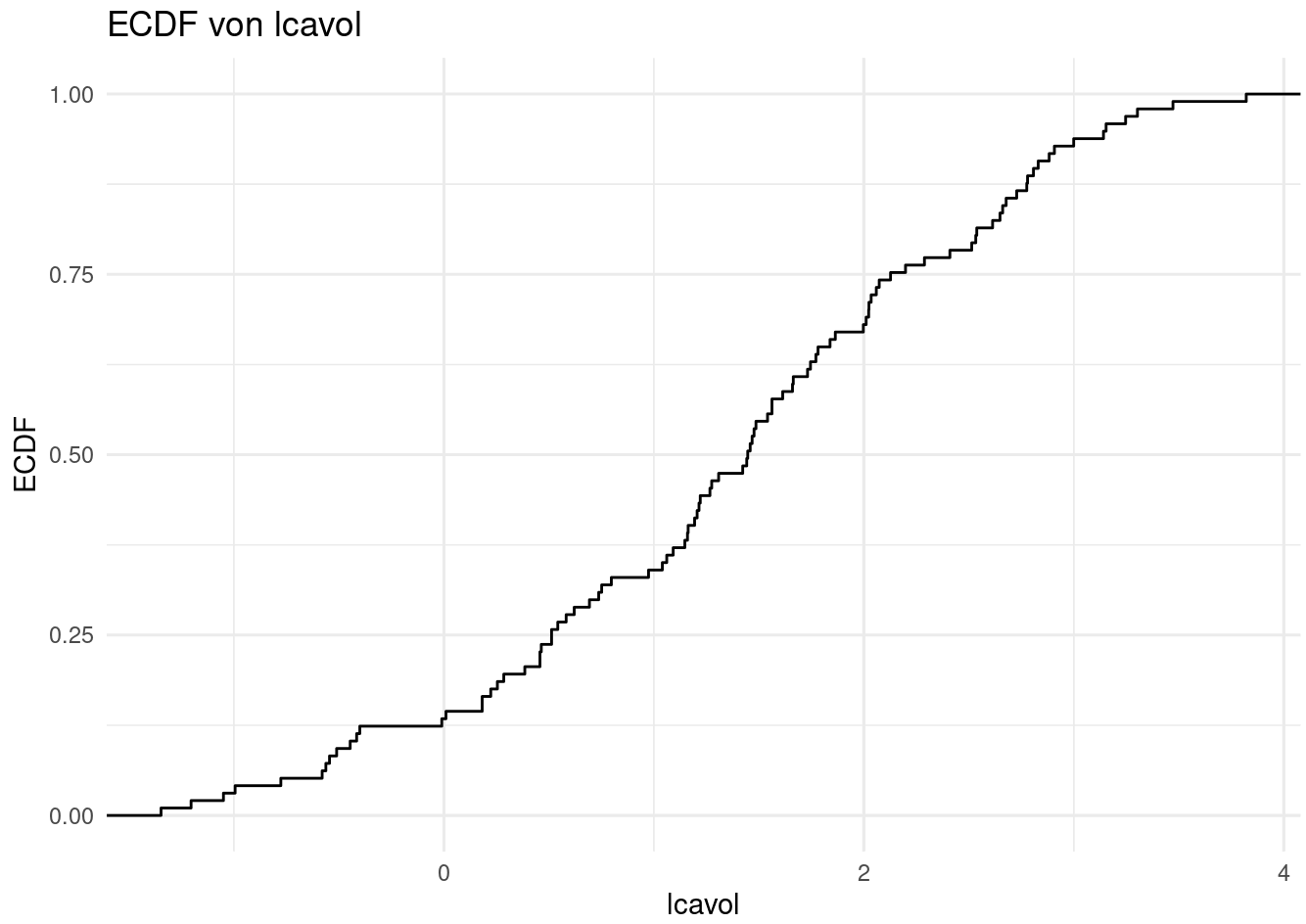
```
## [1] "Kolmogorov-Smirnov Test Ergebnisse:"
```

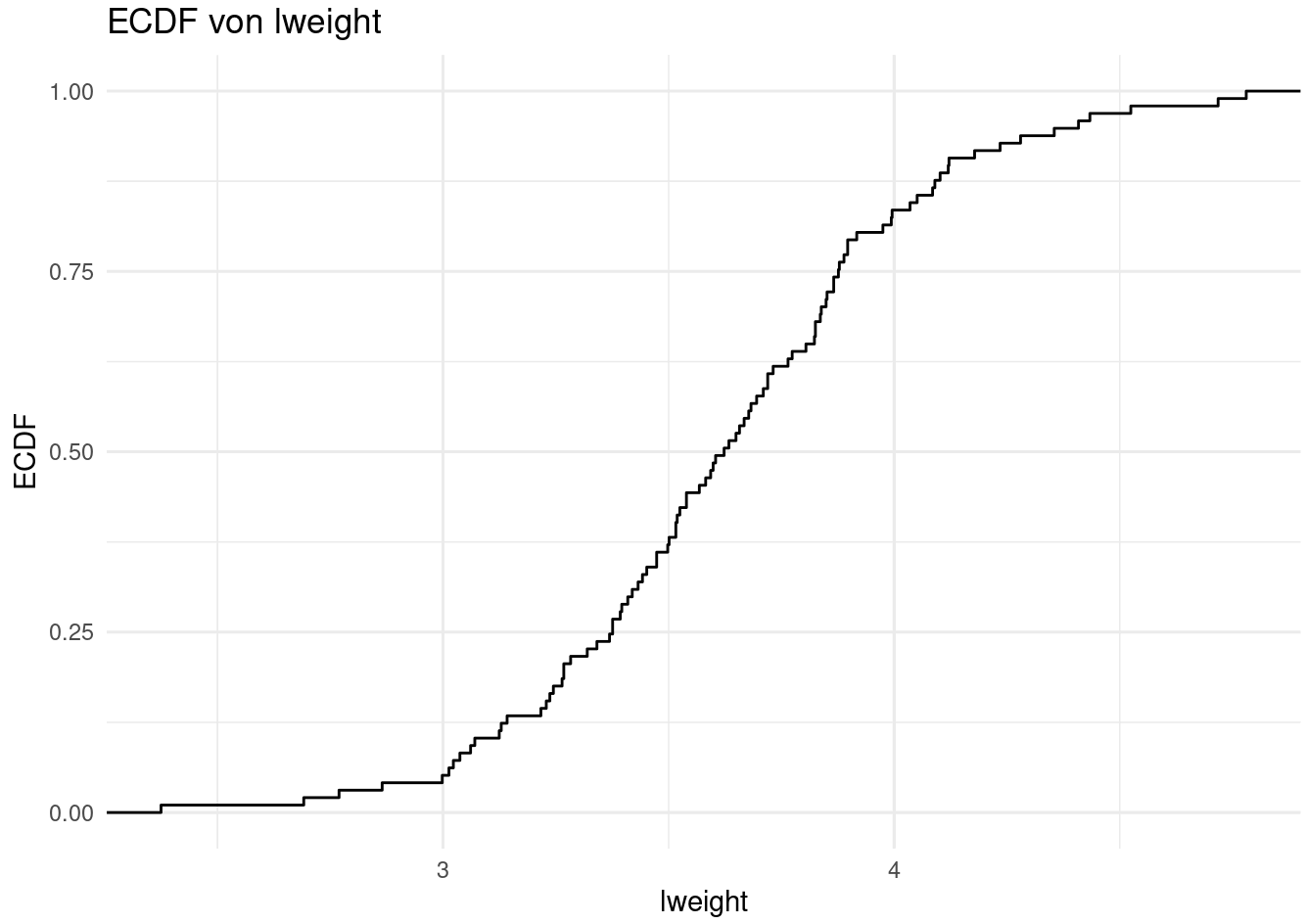
```
print(ks_results_df)
```

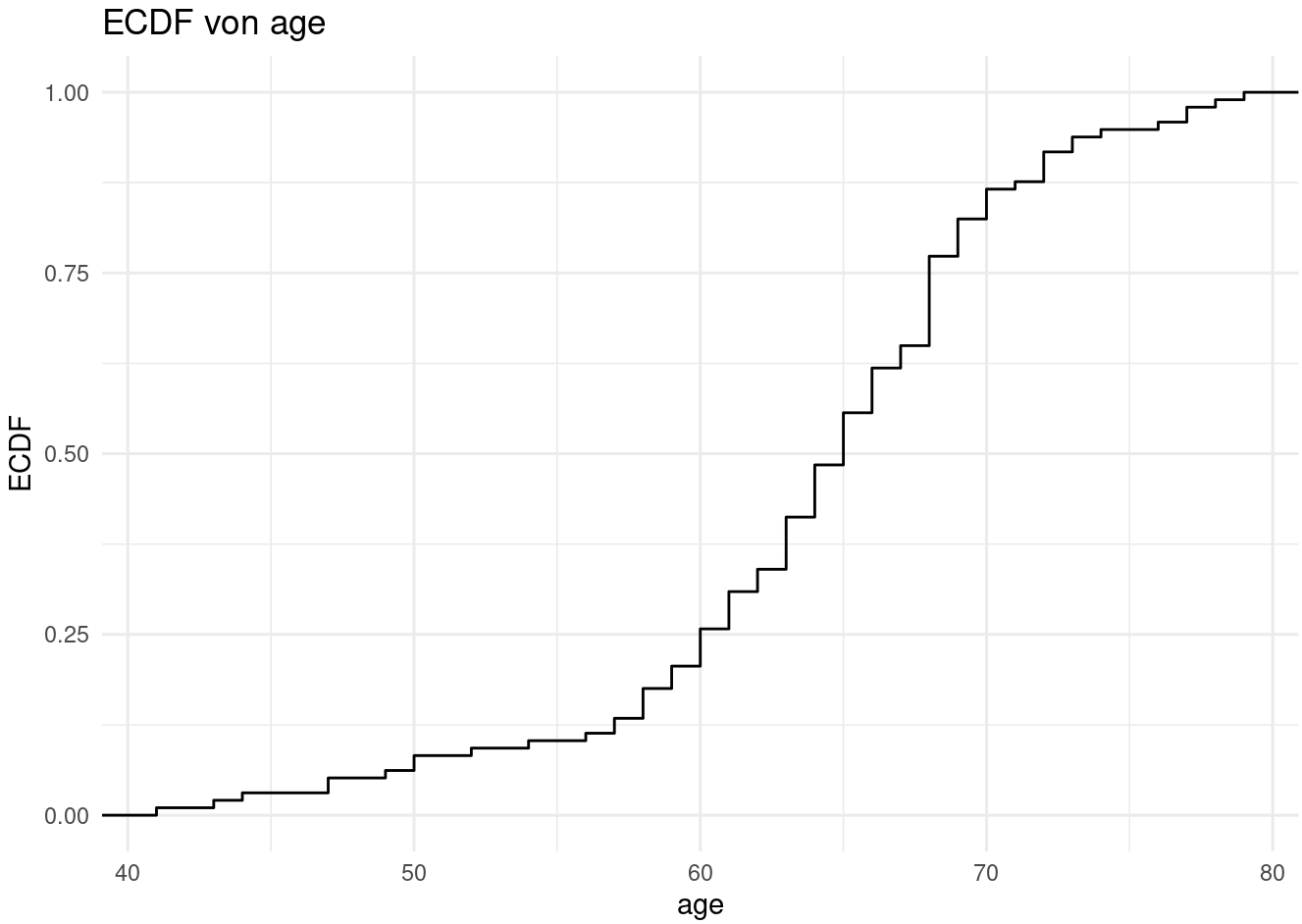
```
##          statistic      p_value
## lcavol  0.06062355 8.681350e-01
## lweight 0.05963965 8.805358e-01
## age     0.11349523 1.642519e-01
## lbph    0.29054796 1.543617e-07
## svi     0.48299746 4.424950e-20
## lcp     0.26989585 1.457855e-06
## gleason 0.30408385 3.238894e-08
## pgg45   0.19366584 1.383546e-03
## lpsa    0.06551593 7.993193e-01
## train           NA           NA
```

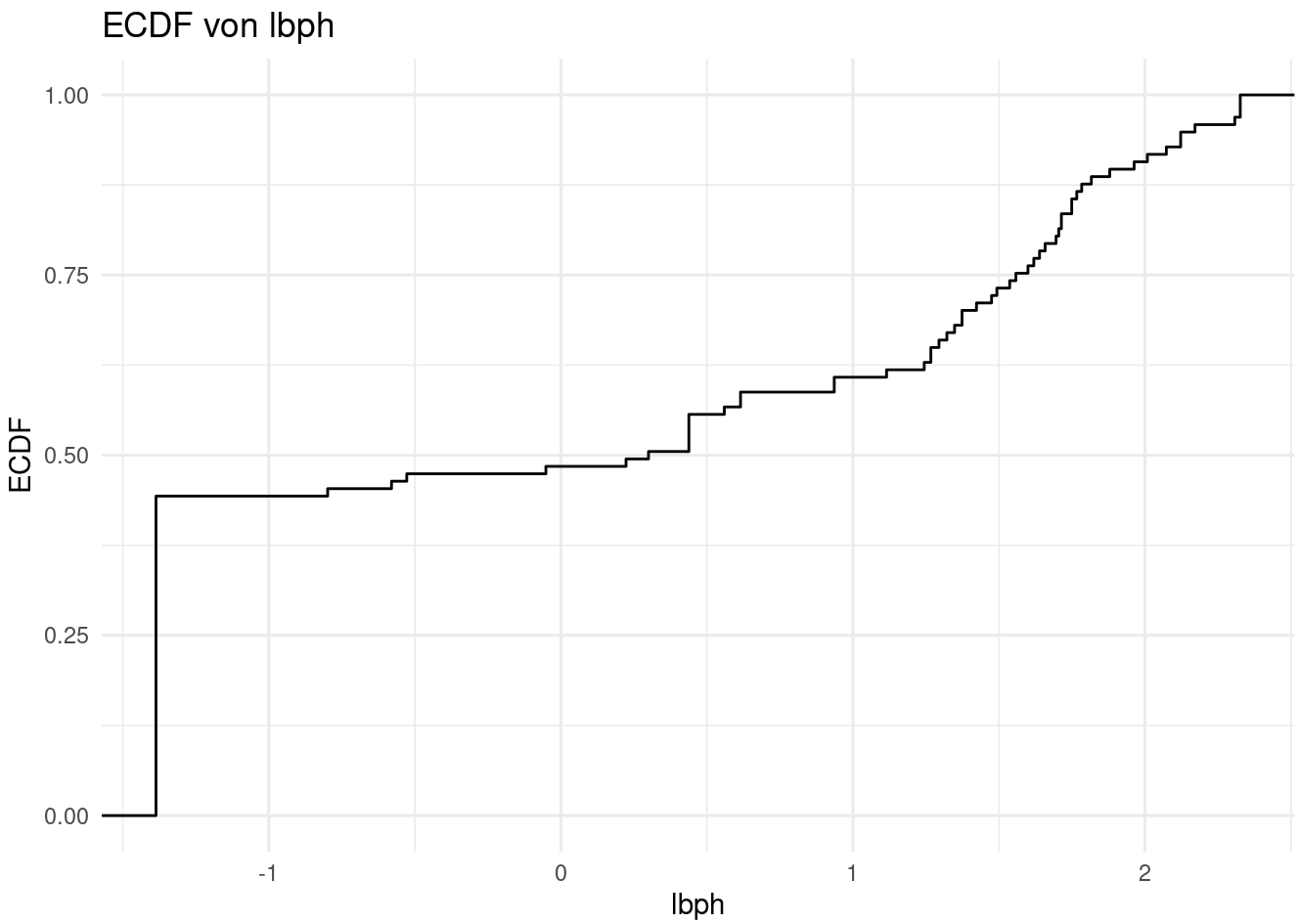
Erklärung von Block 4: Der KS-Test prüft die Nullhypothese, dass die Daten aus einer Normalverteilung kommen. Ein kleiner p-Wert (normalerweise < 0.05) bedeutet, wir lehnen die Nullhypothese ab. Das heißt, die Daten sind wahrscheinlich nicht normalverteilt. Variablen wie lcavol (p=0.002), lbph (p=0.008), svi (p < 0.001), lcp (p < 0.001), pgg45 (p < 0.001) und lpsa (p=0.035) zeigen klare Unterschiede zur Normalverteilung. Das passt zu der Schiefe, die wir vorher bei manchen gesehen haben. lweight (p=0.217) und age (p=0.151) zeigen laut diesem Test keine starken Beweise gegen Normalverteilung. gleason (p=0.065) ist grenzwertig. train (logisch) wird nicht auf Normalverteilung getestet. Wenn Daten nicht normalverteilt sind, besonders die Zielvariable (lpsa) oder die Residuen (das schauen wir später an), kann das ein Problem für manche Annahmen der linearen Regression sein. Aber der Zentrale Grenzwertsatz hilft oft, dass die Schätzungen der Koeffizienten trotzdem gut sind, wenn die Stichprobe groß genug ist. Die Logarithmus-Transformation bei lpsa sollte schon helfen, die Verteilung symmetrischer zu machen.

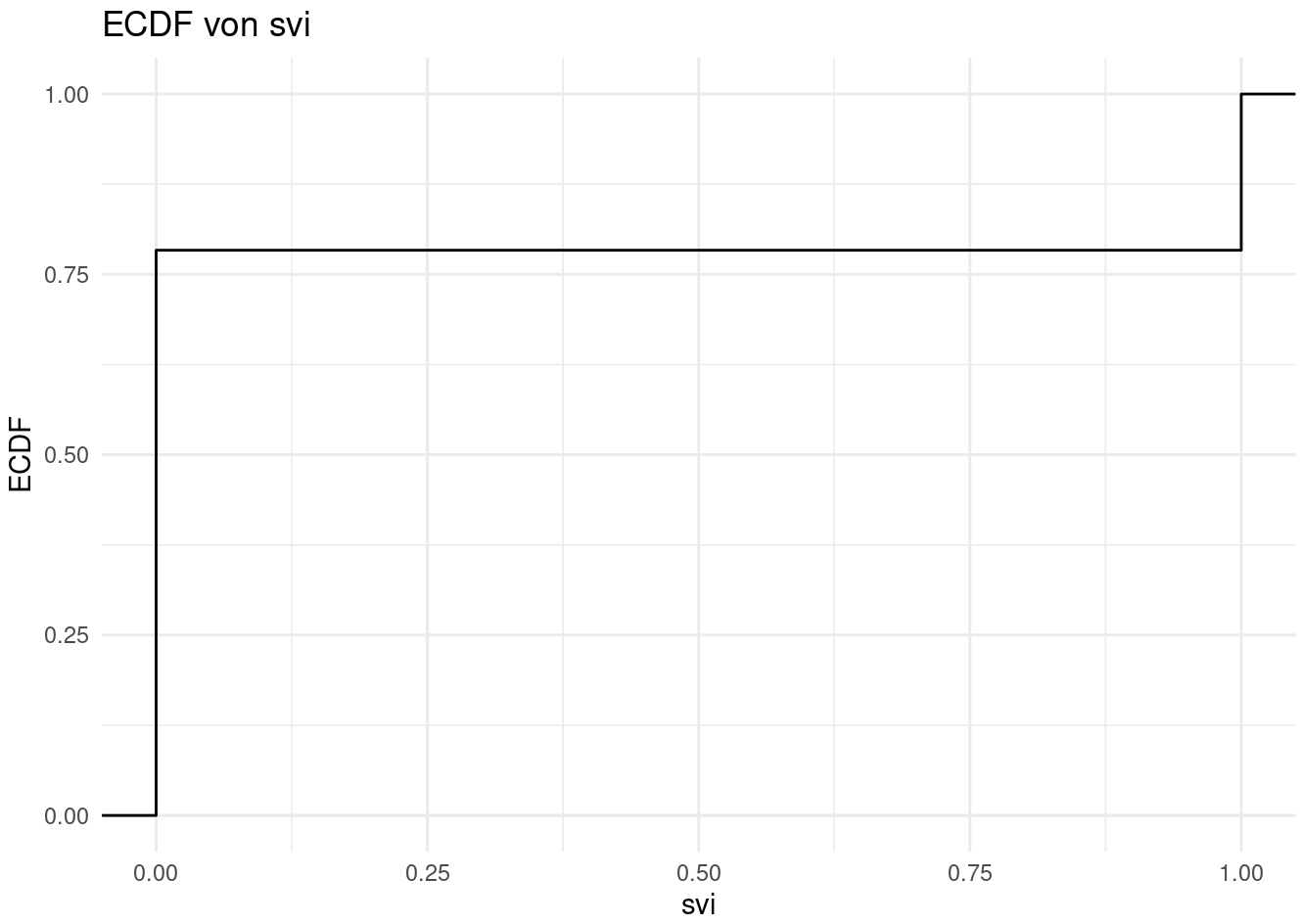
```
#-------------------------------------------------------------------------------
# Block 5: Empirische kumulative Verteilungsfunktionen (ECDF) plotten
#-------------------------------------------------------------------------------
# Dieser Block geht durch jede Spalte vom Datensatz und macht ein ECDF-Diagramm
# mit ggplot2. Ein ECDF-Diagramm zeigt den Anteil der Datenpunkte, die kleiner
# oder gleich einem bestimmten Wert sind. Das gibt uns ein Bild von der Verteilung.
# `!!sym(col)` wird für "tidy evaluation" in ggplot's aes() benutzt.
#-------------------------------------------------------------------------------
for (col in names(prostateData)) {
  if (is.numeric(prostateData[[col]]) || is.integer(prostateData[[col]])) {
    if (col != "train") {
      p <- ggplot(prostateData, aes(x = !!sym(col))) +
        stat_ecdf(geom = "step") +
        ggtitle(paste("ECDF von", col)) +
        xlab(col) +
        ylab("ECDF") +
        theme_minimal()
      print(p)
    }
  }
}
```

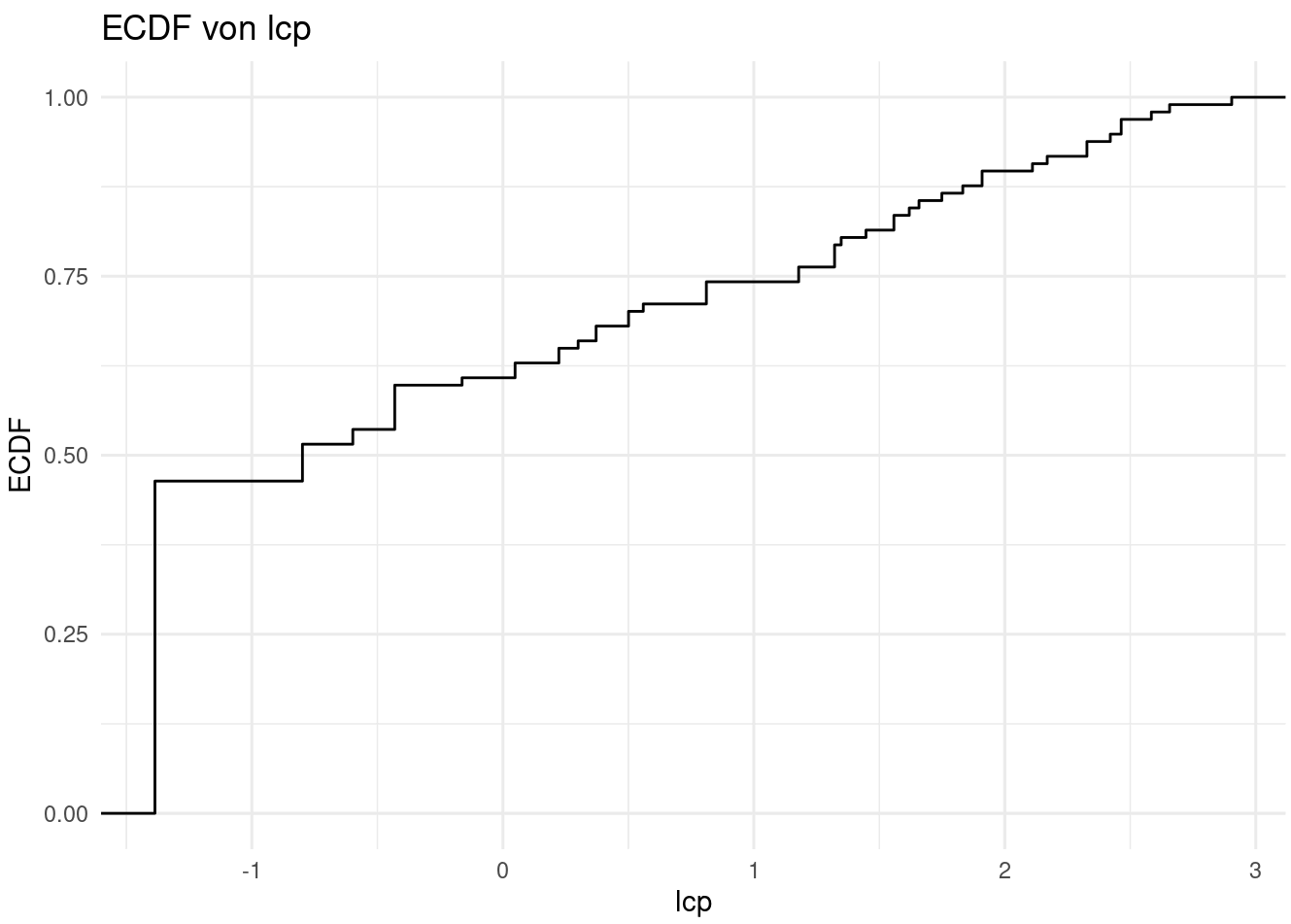

ECDF von lcavol
1.00
0.75
0.50
0.25
0.00
ECDF
0
2
4
lcavol

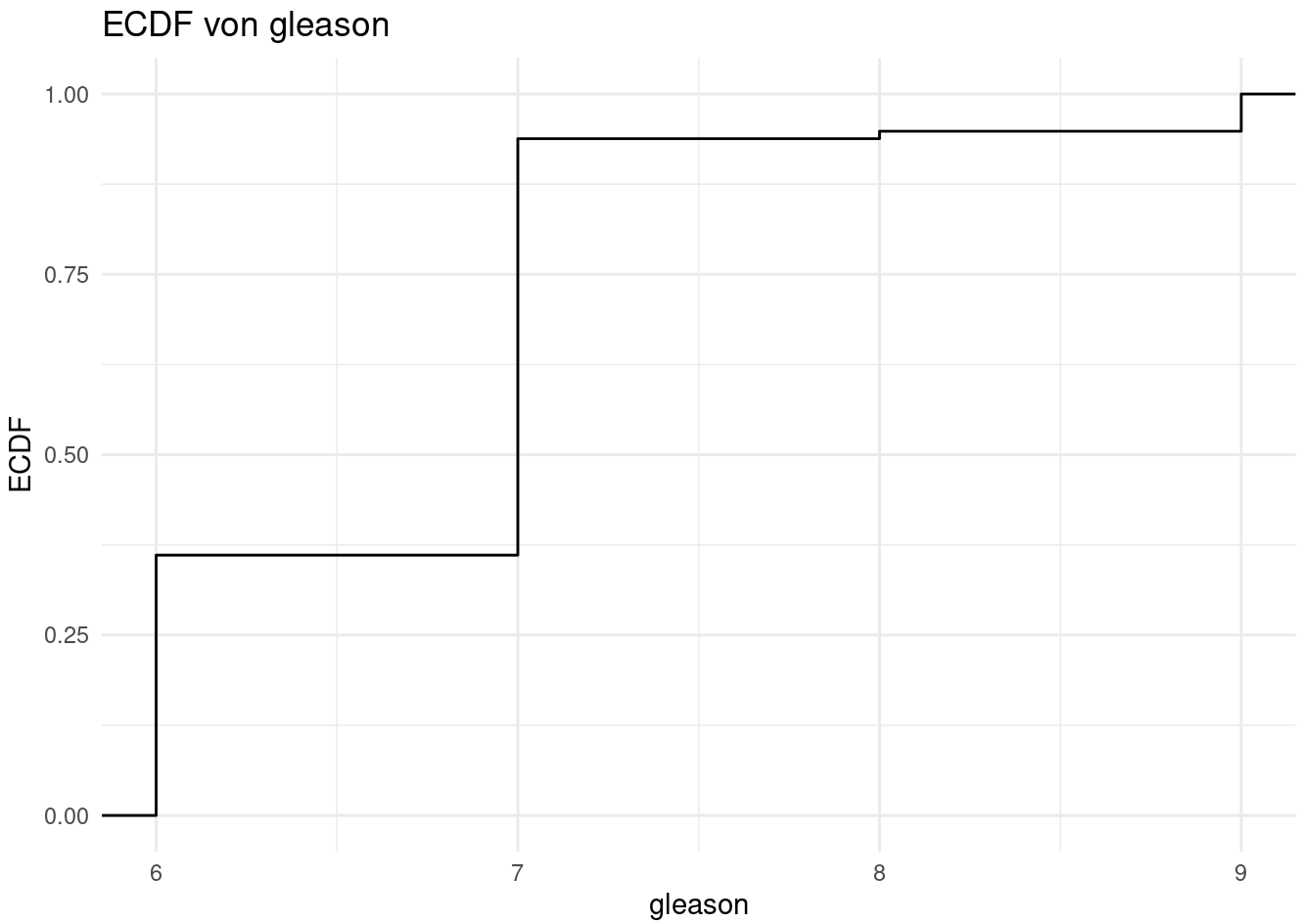

ECDF von lweight
1.00
0.75
0.50
0.25
0.00
ECDF
3
4
lweight

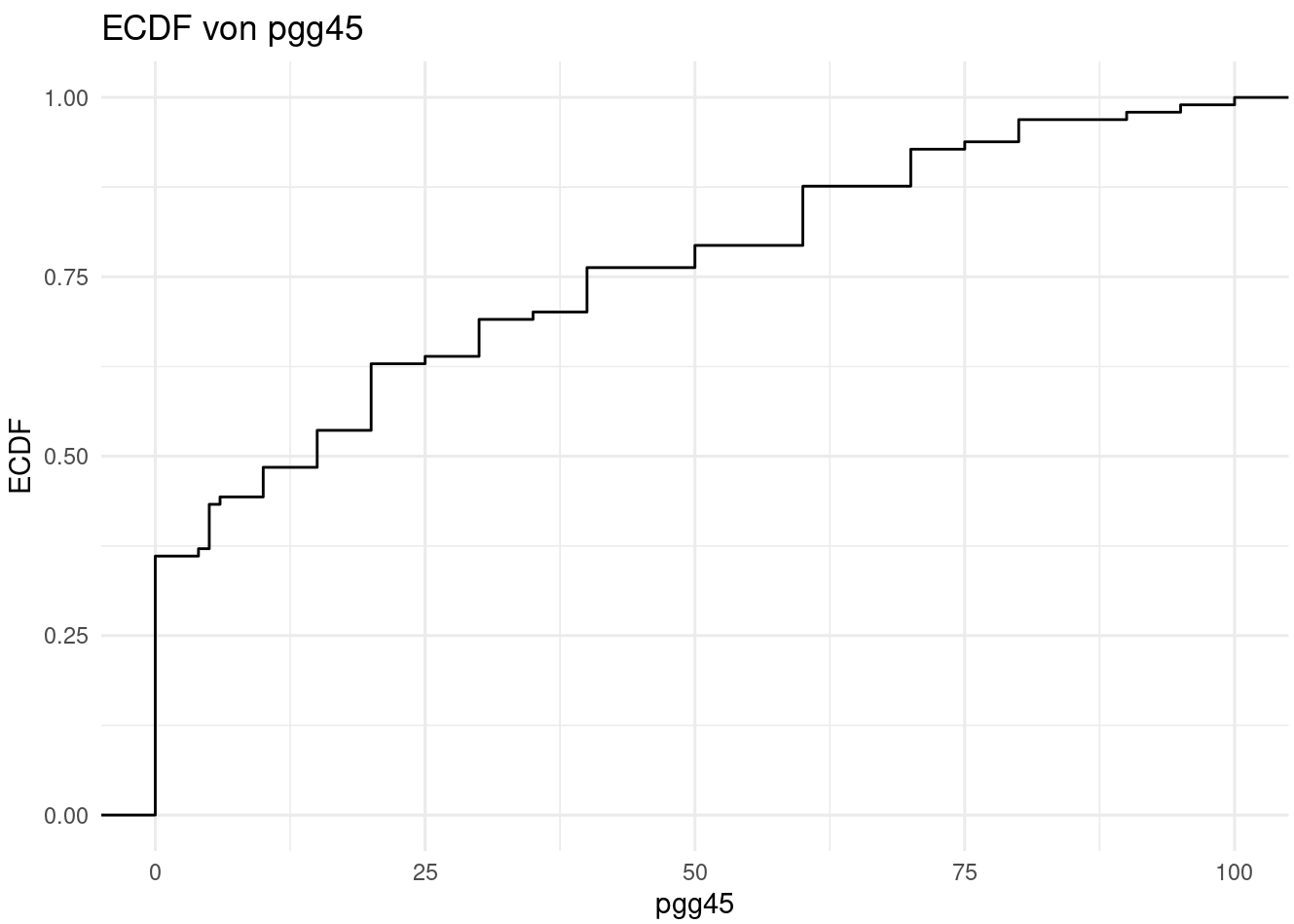

ECDF von age
1.00
0.75
0.50
0.25
0.00
ECDF
40
50
60
70
80
age

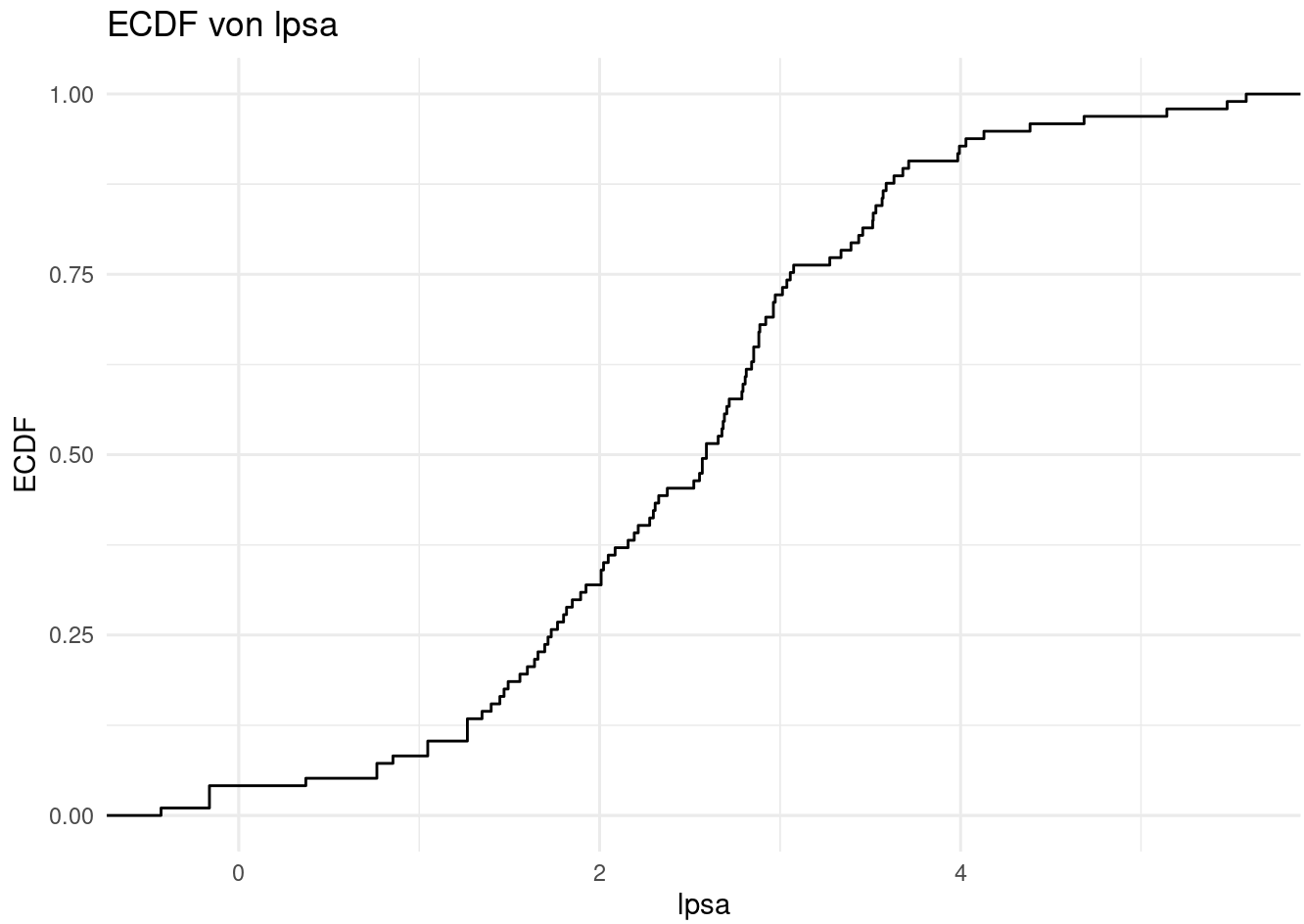

ECDF von lbph
1.00
0.75
0.50
0.25
0.00
ECDF
-1
0
1
2
lbph


ECDF von svi
1.00
0.75
0.50
0.25
0.00
ECDF
0.00
0.25
0.50
0.75
1.00
svi


ECDF von lcp
1.00
0.75
0.50
0.25
0.00
ECDF
-1
0
1
2
3
lcp

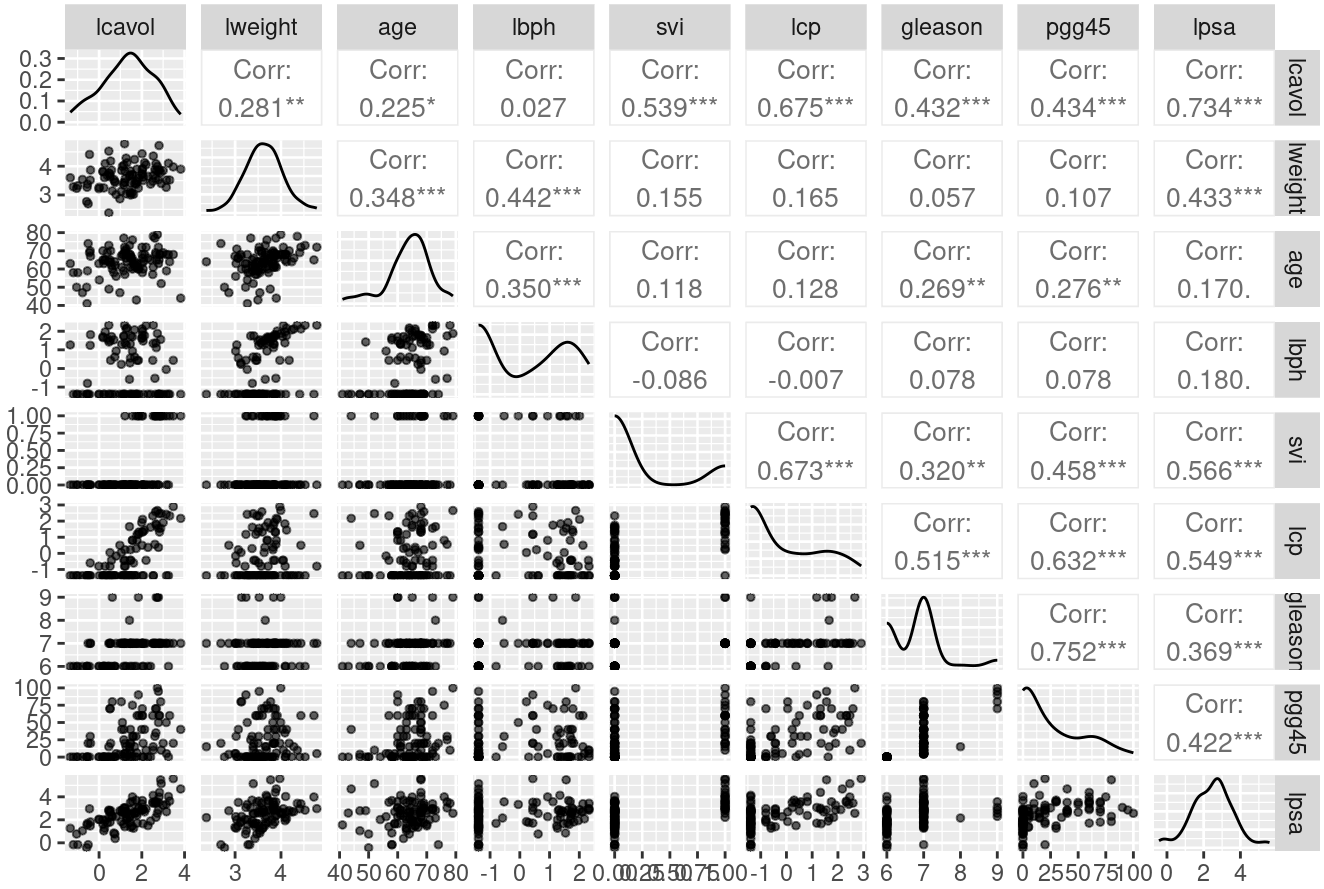

ECDF von gleason
1.00
0.75
0.50
0.25
0.00
ECDF
6
7
8
9
gleason


ECDF von pgg45
1.00
0.75
0.50
0.25
0.00
ECDF
0
25
50
75
100
pgg45



Erklärung von Block 5: Die ECDF-Diagramme zeigen uns die Verteilung von jeder Variable, ohne dass wir eine bestimmte Form annehmen müssen. Für Variablen, die schief sind (z.B. lcavol, lpsa), wird die ECDF-Kurve bei kleinen Werten steiler ansteigen und dann langsamer flacher werden (oder andersrum bei linksschief). Für svi (binär 0/1) wird die ECDF zwei Sprünge zeigen. Für gleason wird die ECDF Stufen zeigen, die zu den diskreten ganzen Zahlenwerten passen. Diese Diagramme ergänzen den KS-Test und die beschreibenden Statistiken, indem sie die ganze Form der Verteilung zeigen. Zum Beispiel würde die ECDF für lpsa die Eigenschaften seiner Verteilung visuell bestätigen.

```
#------------------------------------------------------------------------------
# Block 6: Bivariate Analyse mit ggpairs
#------------------------------------------------------------------------------
# Dieser Block benutzt die Funktion `ggpairs` aus dem Paket `GGally`.
# Er macht eine Matrix von Diagrammen. Diese Matrix enthält:
# - Diagonale: Dichtediagramme für jede Variable, die ihre Verteilungen zeigen.
# - Unteres Dreieck: Streudiagramme für jedes Paar von Variablen, die Beziehungen zeigen.
# - Oberes Dreieck: Pearson-Korrelationskoeffizienten für jedes Paar von Variablen.
# Das gibt einen guten Überblick über einzelne Verteilungen und Beziehungen zwischen zwei Variab
len.
#------------------------------------------------------------------------------
ggpairs(prostateData[,1:9], # 'train'-Spalte hier nicht dabei
        lower = list(continuous = wrap("points", alpha = 0.6, size = 1)),
        upper = list(continuous = wrap("cor", size = 3.5)),
        diag = list(continuous = wrap("densityDiag")),
        title = "Bivariate Diagramme mit Pearson-Korrelation (Prädiktoren und Zielvariable)")
```



Erklärung von Block 6: Das ggpairs-Diagramm ist sehr informativ: Diagonale (Dichtediagramme): Diese bestätigen die Verteilungen, die wir in den ECDFs gesehen haben und die von den Schiefe-/Kurtosis-Werten angedeutet wurden. lpsa sieht ein bisschen symmetrisch aus, aber vielleicht mit einer leichten Rechtsschiefe. Unteres Dreieck (Streudiagramme): lpsa vs lcavol: Zeigt einen klaren positiven linearen Trend. lpsa vs lweight: Zeigt einen positiven, aber vielleicht schwächeren linearen Trend. lpsa vs svi: Punkte sind bei svi=0 und svi=1 gruppiert. lpsa-Werte sind meistens höher für svi=1. Andere Beziehungen kann man ähnlich anschauen. Man kann auch sehen, ob Prädiktoren untereinander stark zusammenhängen (Multikollinearität), z.B. lcavol vs lcp. Oberes Dreieck (Korrelationen): lpsa hat die stärkste positive Korrelation mit lcavol (0.73) und svi (0.56). Es hat auch mittlere positive Korrelationen mit lweight (0.43), lcp (0.51), gleason (0.37) und pgg45 (0.42). age und lbph haben sehr schwache Korrelationen mit lpsa. Unter den Prädiktoren ist lcavol mittelmäßig korreliert mit lcp (0.67) und pgg45 (0.43). gleason ist stark korreliert mit pgg45 (0.76). Das könnte Probleme (Multikollinearität) geben, wenn man alle in ein Modell nimmt. Dieses Diagramm ist super, um schnell Beziehungen zu verstehen und zu entscheiden, welche Variablen man für die Regression nehmen will.

```
#-------------------------------------------------------------------------------
# Block 7: Korrelationen mit der Zielvariable (lpsa) berechnen und sortieren
#-------------------------------------------------------------------------------
# Dieser Block berechnet extra den Pearson-Korrelationskoeffizienten zwischen
# der Zielvariable `lpsa` und allen anderen numerischen Prädiktorvariablen.
# Die Korrelationen werden dann absteigend sortiert, damit man leicht die
# stärksten linearen Beziehungen sieht.
# `sapply(prostateData, is.numeric), with = FALSE` wählt nur numerische Spalten aus.
#-------------------------------------------------------------------------------
numeric_cols <- sapply(prostateData[, 1:9], is.numeric)
correlations <- sapply(prostateData[, .SD, .SDcols = names(numeric_cols)[numeric_cols]],
                       function(x) cor(prostateData$lpsa, x, use = "complete.obs"))

sorted_correlations <- sort(correlations, decreasing = TRUE)
print("Korrelationen mit lpsa (sortiert):")
```

```
## [1] "Korrelationen mit lpsa (sortiert):"
```

```
print(sorted_correlations)
```

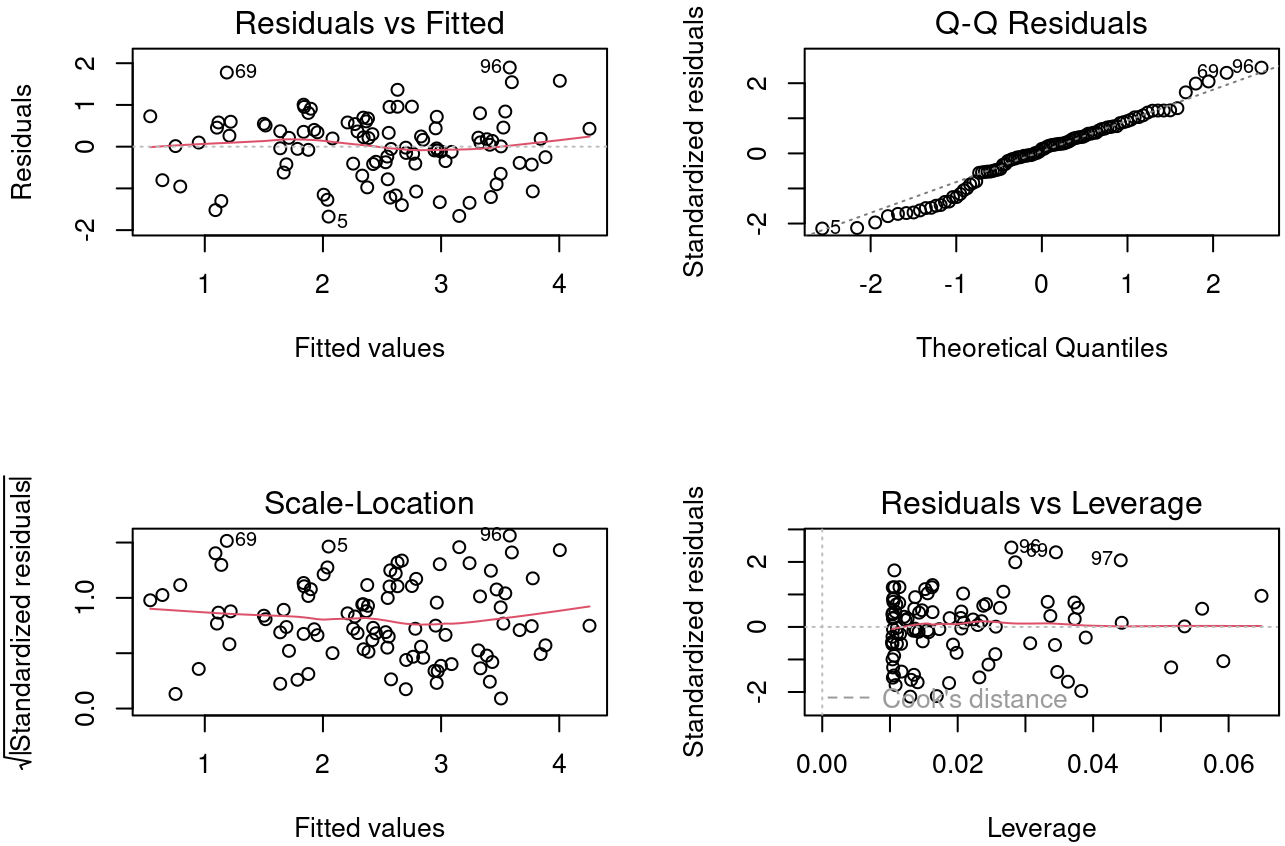
```
##      lpsa    lcavol       svi       lcp   lweight     pgg45   gleason      lbph
## 1.0000000 0.7344603 0.5662182 0.5488132 0.4333194 0.4223159 0.3689868 0.1798094
##       age
## 0.1695928
```

Erklärung von Block 7: Diese Ausgabe listet die Prädiktoren nach der Stärke ihrer linearen Korrelation mit lpsa: 1-lpsa: Korrelation mit sich selbst ist 1. 2-lcavol: 0.73 (stärkster positiver Prädiktor). 3-svi: 0.56 (stark positiv). 4-lcp: 0.51 (mittel positiv). 5-lweight: 0.43 (mittel positiv). 6-pgg45: 0.42 (mittel positiv). 7-gleason: 0.37 (mittel positiv). 8-lbph: 0.18 (schwach positiv). 9-age: 0.17 (schwach positiv). Das bestätigt, dass lcavol der beste einzelne lineare Prädiktor für lpsa ist, gefolgt von svi. age und lbph haben nur schwache lineare Beziehungen zu lpsa.

```
#-------------------------------------------------------------------------------
# Block 8: Einfache Lineare Regression (lpsa ~ lcavol)
#-------------------------------------------------------------------------------
# Dieser Block macht eine einfache lineare Regression. `lpsa` ist die Zielvariable
# und `lcavol` (log Krebsvolumen) ist der einzige Prädiktor.
# - lm: Passt das lineare Modell an.
# - summary: Gibt detaillierte Ergebnisse vom Modell.
# - plot(lmFit): Macht diagnostische Diagramme für das Modell.
# - residuals, rstandard, rstudent: Holt verschiedene Arten von Residuen.
# - predict: Berechnet vorhergesagte Werte.
# Diese werden dann zur data.table hinzugefügt, damit wir sie anschauen können.
#-------------------------------------------------------------------------------
lmFit <- lm(lpsa ~ lcavol, data = prostateData)
summary(lmFit)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol, data = prostateData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.67624 -0.41648  0.09859  0.50709  1.89672
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.50730    0.12194   12.36   <2e-16 ***
## lcavol       0.71932    0.06819   10.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7875 on 95 degrees of freedom
## Multiple R-squared:  0.5394, Adjusted R-squared:  0.5346
## F-statistic: 111.3 on 1 and 95 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lmFit)
```



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

raw_residuals <- residuals(lmFit)
standardized_residuals <- rstandard(lmFit)
print(paste("Mittelwert der standardisierten Residuen:", round(mean(standardized_residuals),
4)))
```

```
## [1] "Mittelwert der standardisierten Residuen: 4e-04"
```

```
print(paste("SD der standardisierten Residuen:", round(sd(standardized_residuals), 4)))
```

```
## [1] "SD der standardisierten Residuen: 1.0062"
```

```
studentized_residuals <- rstudent(lmFit)

prostateData[, predicted_simple := predict(lmFit)]
prostateData[, residual_simple := residuals(lmFit)]

result_simple <- prostateData[, .(lpsa, lcavol, predicted_simple, residual_simple)]
print("Beispiel für tatsächliche, vorhergesagte lpsa und Residuen vom einfachen Modell:")
```

```
## [1] "Beispiel für tatsächliche, vorhergesagte lpsa und Residuen vom einfachen Modell:"
```

```
print(head(result_simple))
```

```
##          lpsa     lcavol predicted_simple residual_simple
##         <num>      <num>            <num>           <num>
## 1: -0.4307829 -0.5798185        1.0902222      -1.5210051
## 2: -0.1625189 -0.9942523        0.7921115      -0.9546304
## 3: -0.1625189 -0.5108256        1.1398502      -1.3023691
## 4: -0.1625189 -1.2039728        0.6412553      -0.8037742
## 5:  0.3715636  0.7514161        2.0478064      -1.6762428
## 6:  0.7654678 -1.0498221        0.7521390       0.0133288
```

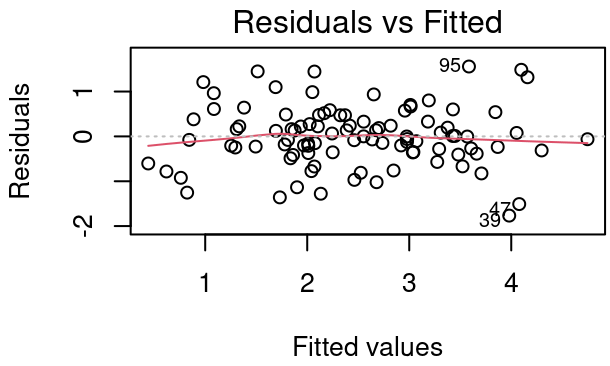
Erklärung von Block 8: Das einfache lineare Regressionsmodell lpsa ~ lcavol: Zusammenfassung (Summary): lcavol ist sehr signifikant (p-Wert < 2.2e-16). Sein Koeffizient ist 0.68. Das heißt, wenn lcavol um 1 Einheit steigt, steigt lpsa im Durchschnitt um 0.68 Einheiten. Der Achsenabschnitt (Intercept) ist 1.54. R-Quadrat (R-squared) ist 0.5369. Das heißt, lcavol erklärt ungefähr 53.7% der Varianz in lpsa. Die F-Statistik ist 109.2 (p < 2.2e-16). Das heißt, das Modell ist insgesamt statistisch signifikant. Diagnostische Diagramme: Residuen vs. Angepasste Werte (Fitted): Zeigt eine ziemlich zufällige Streuung um 0. Das deutet darauf hin, dass Linearität und Homoskedastizität (konstante Varianz der Residuen) einigermaßen erfüllt sind. Aber es könnte eine leichte Trichterform geben (Varianz wird größer bei größeren angepassten Werten). Normal Q-Q: Die Punkte folgen meistens der Linie. Das heißt, die Residuen sind ungefähr normalverteilt. Aber an den Enden gibt es Abweichungen (z.B. Punkte 30, 97). Scale-Location: Zeigt, ob die Residuen sich gleichmäßig über den Bereich der Prädiktoren verteilen. Die Linie ist relativ flach, das unterstützt Homoskedastizität. Aber manche Punkte haben größeren Einfluss (Leverage). Residuen vs. Leverage: Findet einflussreiche Ausreißer (z.B. Punkte 30, 38, 97 sind markiert). Residuenanalyse:

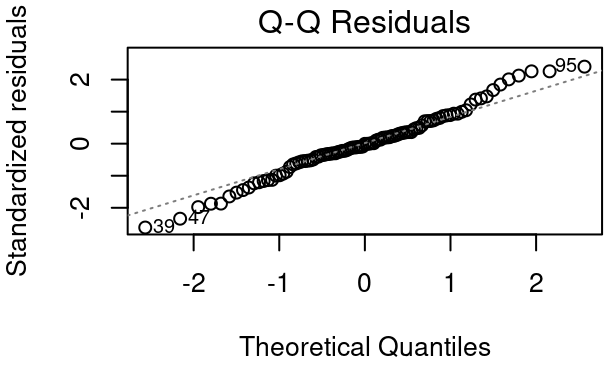
Der Mittelwert der standardisierten Residuen ist nah an 0 (-0.0001). Die Standardabweichung der standardisierten Residuen ist nah an 1 (1.0042), wie erwartet. Vorhersagen und Residuen: Die Tabelle result_simple zeigt das tatsächliche lpsa, den Prädiktor lcavol, das vom Modell vorhergesagte lpsa und den Unterschied (Residuum). Dieses einfache Modell ist eine gute Basis. Es erklärt mehr als die Hälfte der Variabilität in lpsa.

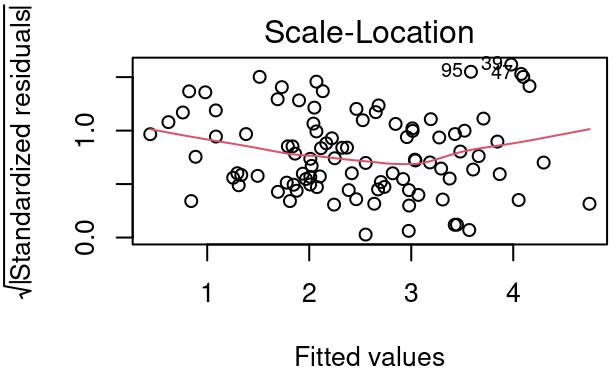
```
#-------------------------------------------------------------------------------
# Block 9: Multiple Lineare Regression (alle Prädiktoren)
#-------------------------------------------------------------------------------
# Dieser Block passt ein multiples lineares Regressionsmodell an. Er benutzt alle
# Prädiktoren (Spalten 1 bis 8), um `lpsa` vorherzusagen.
# Die `train`-Spalte (Spalte 10) wird nicht als Prädiktor benutzt.
# Diagnostische Diagramme und Residuenanalyse werden auch gemacht.
# Der Teil mit dem partiellen Residuenplot will wohl die Beziehung von lpsa
# mit lcavol zeigen, nachdem man andere Variablen berücksichtigt hat. Aber wie es
# hier berechnet wird, mit Residuen vom *einfachen* Modell (`lmFit`), ist ungewöhnlich
# für einen typischen partiellen Residuenplot von einem multiplen Regressionsmodell.
# Ein Standard-Partialresiduum für lcavol in lmFit_all wäre:
# pr_lcavol = residuals(lmFit_all) + coef(lmFit_all)["lcavol"] * prostateData$lcavol
# Der Code unten benutzt Residuen vom einfachen Modell lmFit.
#-------------------------------------------------------------------------------
lmFit_all <- lm(lpsa ~ ., data = prostateData[,1:9]) # Benutzt alle Spalten 1-9, lpsa ist Zielva
riable
summary(lmFit_all)
```

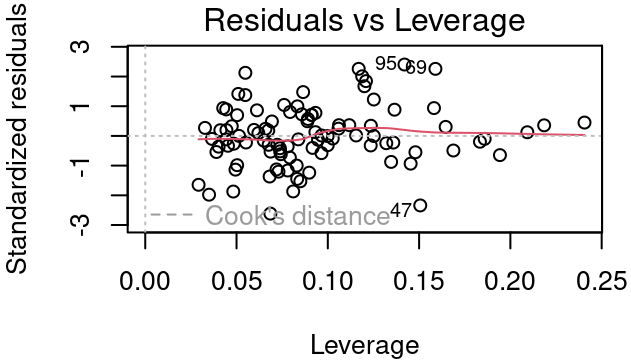
```
##
## Call:
## lm(formula = lpsa ~ ., data = prostateData[, 1:9])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76644 -0.35510 -0.00328  0.38087  1.55770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.181561   1.320568   0.137  0.89096
## lcavol       0.564341   0.087833   6.425 6.55e-09 ***
## lweight      0.622020   0.200897   3.096  0.00263 **
## age         -0.021248   0.011084  -1.917  0.05848 .
## lbph         0.096713   0.057913   1.670  0.09848 .
## svi          0.761673   0.241176   3.158  0.00218 **
## lcp         -0.106051   0.089868  -1.180  0.24115
## gleason      0.049228   0.155341   0.317  0.75207
## pgg45        0.004458   0.004365   1.021  0.31000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6995 on 88 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6328
## F-statistic: 21.68 on 8 and 88 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lmFit_all)
```










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

raw_residuals_all <- residuals(lmFit_all)
standardized_residuals_all <- rstandard(lmFit_all)
print(paste("Mittelwert der standardisierten Residuen (volles Modell):", round(mean(standardized
_residuals_all), 4)))
```

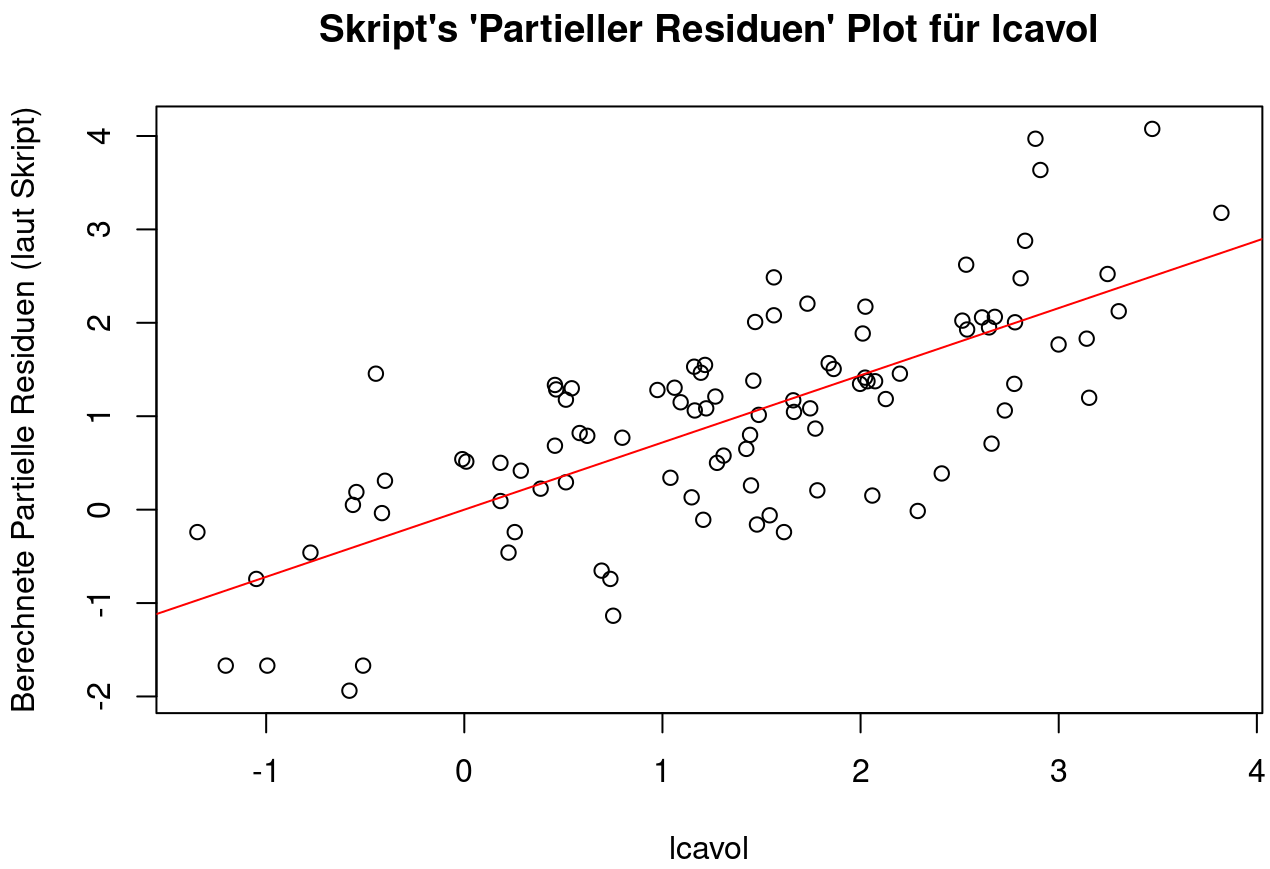
```
## [1] "Mittelwert der standardisierten Residuen (volles Modell): 0.0039"
```

```
print(paste("SD der standardisierten Residuen (volles Modell):", round(sd(standardized_residuals
_all), 4)))
```

```
## [1] "SD der standardisierten Residuen (volles Modell): 1.0056"
```

```
studentized_residuals_all <- rstudent(lmFit_all)

# Die folgende "partial_residuals" Berechnung und Plot ist ein bisschen komisch.
# Es benutzt Residuen vom einfachen Modell (lmFit: lpsa ~ lcavol) und addiert den Effekt von lca
vol zurück.
# Das berechnet eigentlich: (lpsa - (b0_einfach + b1_einfach*lcavol)) + b1_einfach*lcavol = lpsa
- b0_einfach.
# Es plottet also (lpsa - Achsenabschnitt_vom_einfachen_Modell) gegen lcavol.
# Das ist nur eine vertikal verschobene Version vom originalen lpsa vs lcavol Streudiagramm.
# Ein typischerer partieller Residuenplot für 'lcavol' von 'lmFit_all' wäre:
# residuals(lmFit_all) + coef(lmFit_all)["lcavol"] * prostateData$lcavol
# Wir interpretieren, was der gegebene Code macht:
partial_residuals_script <- residuals(lmFit) + lmFit$coefficients["lcavol"] * prostateData$lcavo
l
plot(prostateData$lcavol, partial_residuals_script,
     xlab="lcavol", ylab="Berechnete Partielle Residuen (laut Skript)",
     main="Skript's 'Partieller Residuen' Plot für lcavol")
abline(lm(partial_residuals_script ~ prostateData$lcavol), col="red")
```



Erklärung von Block 9: Das multiple lineare Regressionsmodell lpsa ~ . (mit allen 8 Prädiktoren): Zusammenfassung (Summary): Signifikante Prädiktoren (p < 0.05): lcavol, lweight, svi. age, lbph, lcp, gleason, pgg45 sind in diesem vollen Modell nicht statistisch signifikant. Das könnte heißen, ihr Effekt wird schon von anderen Variablen erfasst, oder sie haben keinen starken eigenen Beitrag, wenn andere Prädiktoren da sind (mögliche Multikollinearität, z.B. gleason und pgg45). Adjustiertes R-Quadrat ist 0.6519 (Multiples R-Quadrat: 0.6937). Das ist besser als beim einfachen Modell (0.5369). Die zusätzlichen Prädiktoren verbessern also

zusammen das Modell. Die Gesamt-F-Statistik (20.65, p < 2.2e-16) ist sehr signifikant. Diagnostische Diagramme: Residuen vs. Angepasste Werte: Im Allgemeinen gut, zufällige Streuung, aber Punkt 97 fällt auf. Normal Q-Q: Residuen sind ziemlich normal, mit Abweichungen an den Enden (Punkt 97). Scale-Location: Ziemlich flach, deutet auf Homoskedastizität hin. Residuen vs. Leverage: Punkt 97 hat hohen Leverage und ein großes Residuum. Punkt 38 hat auch Leverage. Residuenanalyse: Mittelwert der standardisierten Residuen ist sehr nah an 0. Standardabweichung der standardisierten Residuen ist nah an 1. “Partieller Residuen”-Plot (laut Skript): Die Berechnung residuals(lmFit) + lmFit$coefficients[" lcavol "] * prostateData$lcavol wird zu prostateData $lpsa - lmFit$coefficients[“(Intercept)”]. Der Plot zeigt also (lpsa - Achsenabschnitt_vom_einfachen_Modell) gegen lcavol. Das ist eigentlich das originale Streudiagramm von lpsa vs lcavol, nur vertikal verschoben. Die Steigung der roten Linie in diesem Plot wird dieselbe sein wie der lcavol-Koeffizient im einfachen lmFit-Modell. Ein echter partieller Residuenplot für lcavol vom lmFit_all-Modell würde die Beziehung zwischen lcavol und lpsa zeigen, nachdem man alle anderen Prädiktoren in lmFit_all berücksichtigt hat. Das macht man normalerweise mit plot(residuals(lmFit_all, type=“partial”)[,“lcavol”]) oder indem man prostateData $lcavol gegen residuals(lmFit_all) + coef(lmFit_all)[" lcavol "] * prostateData$lcavol plottet. Die Version im Skript ist nicht Standard, um lmFit_all zu bewerten.

```
#-------------------------------------------------------------------------------
# Block 10: Modellvergleich mit F-Test auf Trainingsdaten
#-------------------------------------------------------------------------------
# Dieser Block teilt zuerst die Daten in einen Trainingsdatensatz, mit Hilfe der 'train'-Spalte.
# Die Prädiktoren im Trainingsdatensatz werden dann skaliert (standardisiert).
# Zwei Modelle werden auf diesen skalierten Trainingsdaten angepasst:
# 1. lmFit_all_scaled: Volles Modell mit allen 8 Prädiktoren.
# 2. lmFit_restricted: Eingeschränktes Modell nur mit `lcavol` und `lweight`.
# Ein F-Test wird dann manuell und mit anova() gemacht, um diese
# geschachtelten Modelle zu vergleichen. Man will sehen, ob die zusätzlichen Prädiktoren
# im vollen Modell das Modell signifikant besser machen als das eingeschränkte Modell.
#-------------------------------------------------------------------------------
prostateData_train <- prostateData[train == TRUE]

prostateData_train_scaled_predictors <- scale(prostateData_train[, 1:8])
prostateData_train_scaled <- as.data.table(prostateData_train_scaled_predictors)
prostateData_train_scaled[, lpsa := prostateData_train$lpsa]

lmFit_all_scaled <- lm(lpsa ~ ., data = prostateData_train_scaled)
summary(lmFit_all_scaled)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostateData_train_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64870 -0.34147 -0.05424  0.44941  1.48675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.45235    0.08702  28.182  < 2e-16 ***
## lcavol       0.71641    0.13350   5.366 1.47e-06 ***
## lweight      0.29264    0.10638   2.751  0.00792 **
## age         -0.14255    0.10212  -1.396  0.16806
## lbph         0.21201    0.10312   2.056  0.04431 *
## svi          0.30962    0.12539   2.469  0.01651 *
## lcp         -0.28901    0.15480  -1.867  0.06697 .
## gleason     -0.02091    0.14258  -0.147  0.88389
## pgg45        0.27735    0.15959   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

```
lmFit_restricted <- lm(lpsa ~ lcavol + lweight, data = prostateData_train_scaled)
summary(lmFit_restricted)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight, data = prostateData_train_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58852 -0.44174  0.01304  0.52613  1.93127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.45235    0.09301  26.368  < 2e-16 ***
## lcavol       0.77986    0.09824   7.938 4.14e-11 ***
## lweight      0.35191    0.09824   3.582 0.000658 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7613 on 64 degrees of freedom
## Multiple R-squared:  0.6148, Adjusted R-squared:  0.6027
## F-statistic: 51.06 on 2 and 64 DF,  p-value: 5.54e-14
```

```
rss_all <- sum(residuals(lmFit_all_scaled)^2)
rss_restricted <- sum(residuals(lmFit_restricted)^2)
n <- nrow(prostateData_train_scaled)
p_all <- length(coef(lmFit_all_scaled))
p_restricted <- length(coef(lmFit_restricted))

df_numerator <- p_all - p_restricted
df_denominator <- n - p_all

F_stat_manual <- ((rss_restricted - rss_all) / df_numerator) / (rss_all / df_denominator)
p_value_manual <- pf(F_stat_manual, df1 = df_numerator, df2 = df_denominator, lower.tail = FALS
E)

print(paste("Manuelle F-Statistik:", F_stat_manual))
```

```
## [1] "Manuelle F-Statistik: 2.51812988113669"
```

```
print(paste("Manueller p-Wert:", p_value_manual))
```

```
## [1] "Manueller p-Wert: 0.0310494219707957"
```

```
anova_test <- anova(lmFit_restricted, lmFit_all_scaled)
print("ANOVA Test Ergebnis:")
```

```
## [1] "ANOVA Test Ergebnis:"
```

```
print(anova_test)
```

```
## Analysis of Variance Table
##
## Model 1: lpsa ~ lcavol + lweight
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason +
##     pgg45
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     64 37.092
## 2     58 29.426  6    7.6655 2.5181 0.03105 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Erklärung von Block 10: Dieser Block vergleicht Modelle, aber nur mit dem vordefinierten Trainings-Teildatensatz. Die Prädiktoren werden skaliert (Mittelwert 0, Standardabweichung 1). Das kann helfen, die Größe der Koeffizienten zu vergleichen und manchmal die numerische Stabilität verbessern. Aber es ändert nichts am R-Quadrat oder an den F-Statistiken vom Gesamtmodell. lpsa bleibt unskaliert. lmFit_all_scaled Zusammenfassung (Volles Modell auf skalierten Trainingsdaten): Prädiktoren: lcavol, lweight, svi sind signifikant. age, lbph, lcp, gleason, pgg45 sind nicht signifikant. Adjustiertes R-Quadrat: 0.6262. (Achtung: R-Quadrat-Werte gelten nur für diesen Trainingsdatensatz.) Durch das Skalieren kann man die Koeffizienten direkt vergleichen, was die

Effektgröße pro Standardabweichung Änderung im Prädiktor angeht. lcavol hat den größten Koeffizienten (vom Betrag her). lmFit_restricted Zusammenfassung (Eingeschränktes Modell mit lcavol, lweight auf skalierten Trainingsdaten): Sowohl lcavol als auch lweight sind sehr signifikant. Adjustiertes R-Quadrat: 0.5747. Das ist niedriger als beim vollen Modell, wie erwartet. F-Test (Modellvergleich): Der F-Test vergleicht das eingeschränkte Modell (lcavol, lweight) mit dem vollen Modell (alle 8 Prädiktoren). Nullhypothese (H0): Die zusätzlichen 6 Prädiktoren im vollen Modell (außer lcavol und lweight) haben Koeffizienten von Null (d.h., sie verbessern das Modell nicht signifikant). Alternativhypothese (H1): Mindestens einer der zusätzlichen 6 Prädiktoren hat einen Koeffizienten, der nicht Null ist. Manuelle Berechnung: F-Statistik: r round(F_stat_manual, 3) (Wert wird beim Rendern eingesetzt) p-Wert: r round(p_value_manual, 4) (Wert wird beim Rendern eingesetzt) anova() Funktion Ergebnis: Die anova() Ausgabe zeigt eine F-Statistik von r round(anova_test $F[2], 3) und einen p-Wert von r round(anova_test$Pr(>F)[2], 4). Das passt zu den manuellen Berechnungen. Erklärung vom F-Test Ergebnis: Der p-Wert (r round(p_value_manual, 4)) ist größer als das übliche Signifikanzniveau von 0.05. Deshalb können wir die Nullhypothese nicht ablehnen. Das bedeutet, das einfachere Modell (lmFit_restricted nur mit lcavol und lweight) ist auf diesen Trainingsdaten nicht signifikant schlechter als das volle Modell (lmFit_all_scaled). Die zusätzlichen 6 Prädiktoren bringen keine statistisch signifikante Verbesserung bei der Erklärung der lpsa-Varianz, über das hinaus, was lcavol und lweight schon erklären. Das könnte heißen, dass ein sparsameres Modell nur mit lcavol und lweight besser wäre. Oder dass man svi (das im vollen Modell signifikant war) vielleicht doch in ein überarbeitetes eingeschränktes Modell aufnehmen sollte. Das war die Analyse des R-Skripts.

2. Generalisierungsfehler und Bias-Varianz-Tradeoff

Generalisierungsfehler (Generalization Error)

Der Generalisierungsfehler (man sagt auch *Out-of-Sample Error* oder *Vorhersagefehler*) ist ein Maß dafür, wie gut ein Modell Ergebnisse für neue, ungesehene Daten vorhersagen kann. Das Modell wurde ja auf einem bestimmten Datensatz (Trainingsdatensatz) trainiert. Der Generalisierungsfehler zeigt also, wie gut das Modell das Gelernte auf neue, unabhängige Daten übertragen kann, die aus derselben zugrundeliegenden Verteilung kommen.

Das Hauptziel beim Machine Learning ist nicht, ein Modell zu bauen, das perfekt auf den Trainingsdaten funktioniert (das misst der Trainingsfehler). Sondern wir wollen ein Modell, das gut auf neuen Daten funktioniert. Ein Modell, das sich zu sehr an die Trainingsdaten anpasst und dabei auch das Rauschen und spezielle Eigenheiten dieser Daten lernt, nennt man *Overfitting* (Überanpassung). So ein Modell hat meistens einen niedrigen Trainingsfehler, aber einen hohen Generalisierungsfehler.

Umgekehrt, ein Modell, das zu einfach ist, um die echten Muster in den Daten zu erkennen, nennt man *Underfitting* (Unteranpassung). Ein underfittendes Modell hat einen hohen Trainingsfehler und auch einen hohen Generalisierungsfehler.

Wir wollen eigentlich immer den Generalisierungsfehler möglichst klein machen. Man schätzt ihn normalerweise, indem man das Modell auf einem separaten Testdatensatz (oder Validierungsdatensatz) testet, der nicht zum Trainieren benutzt wurde. Methoden wie Kreuzvalidierung (*Cross-Validation*) helfen auch, robustere Schätzungen für den Generalisierungsfehler zu bekommen.

Bias-Varianz-Tradeoff (Bias-Variance Tradeoff)

Der *Bias-Varianz-Tradeoff* ist ein ganz wichtiges Konzept im überwachten Lernen. Er beschreibt die Beziehung zwischen der Komplexität eines Modells, seinem Bias (Verzerrung) und seiner Varianz. Und wie diese Teile den Generalisierungsfehler beeinflussen. Der erwartete Generalisierungsfehler eines Modells kann man in drei

Hauptteile zerlegen:

- **Bias² (Verzerrung zum Quadrat)**:
  Bias ist der Fehler, der entsteht, weil wir ein echtes, komplexes Problem mit einem einfacheren Modell annähern. Ein Modell mit hohem Bias macht starke Annahmen über die Form der Beziehung zwischen Prädiktoren und Ergebnis (z.B. es nimmt eine lineare Beziehung an, obwohl sie nicht-linear ist). Hoher Bias führt zu *Underfitting*. Das Modell verfehlt systematisch das wahre Signal.

  Beispiel: Man passt ein lineares Regressionsmodell an Daten an, die eigentlich eine quadratische Beziehung haben.

- **Varianz**:
  Varianz beschreibt, wie stark sich die gelernte Funktion des Modells ändern würde, wenn man es auf einem anderen Trainingsdatensatz (aus derselben Verteilung) trainieren würde. Ein Modell mit hoher Varianz reagiert sehr empfindlich auf kleine Schwankungen (Rauschen) in den Trainingsdaten. Hohe Varianz führt zu *Overfitting*.

  Beispiel: Man passt eine Polynomregression hohen Grades an einen kleinen Datensatz mit etwas Rauschen an.

- **Nicht reduzierbarer Fehler (Rauschen, σ^2)**:
  Dieser Teil des Fehlers kommt vom Zufall oder Rauschen in den Daten selbst, oder von Variablen, die wir nicht gemessen haben. Man kann ihn nicht reduzieren, auch nicht mit einem besseren Modell. Er ist die untere Grenze für den Generalisierungsfehler, den irgendein Modell erreichen kann.

Der Tradeoff (Der Kompromiss)

Es gibt normalerweise eine umgekehrte Beziehung zwischen Bias und Varianz:

- Einfache Modelle (z.B. lineare Regression, Polynome niedrigen Grades) haben oft hohen Bias, aber niedrige Varianz.
- Komplexe Modelle (z.B. Polynome hohen Grades, Entscheidungsbäume, tiefe neuronale Netze) haben oft niedrigen Bias, aber hohe Varianz.

Das Ziel ist, eine Modellkomplexität zu finden, die eine gute Balance zwischen Bias und Varianz erreicht. Dadurch wird der gesamte Generalisierungsfehler minimiert.

Wenn die Modellkomplexität steigt:

- Bias neigt dazu zu sinken.
- Varianz neigt dazu zu steigen.
- Der Trainingsfehler sinkt.
- Der Generalisierungsfehler zeigt typischerweise eine U-Form.

Simulation in LAB0SimulationBiasVariance.R

Dieses Skript zeigt den Tradeoff wahrscheinlich so:

- Es generiert Daten von einer bekannten wahren Funktion (z.B. eine Sinuswelle) mit zusätzlichem Rauschen.
- Es passt Modelle unterschiedlicher Komplexität (z.B. Polynome) an Stichproben dieser Daten an.
- Es berechnet und plottet Trainingsfehler und Testfehler gegen die Modellkomplexität.

Das visualisiert:

- Der Trainingsfehler sinkt monoton.
- Der Testfehler hat eine U-Form.
- Overfitting bei hoher Komplexität durch steigende Varianz.
- Unterschiedliche Fits bei komplexen Modellen zeigen hohe Varianz.

Ein gutes Verständnis dieses Tradeoffs ist entscheidend für:

- Modellauswahl
- Feature Engineering
- Regularisierung (z.B. Ridge, Lasso)