Praktikum zu Grundlagen der Statistik

Praktikumsbericht

Parviz Moskalenko

Universität Marburg

Studiengang: Wirtschaftsinformatik

WiSe 2022/2023

Inhaltsverzeichnis

Tag 1: Daten und Plots

Aufgabe 1: Datensatz OECD

a - Laden Sie sich die Datei oecdM.csv herunter

In dieser Aufgabe habe ich den Datensatz von ILIAS herruntergeladen und mittels read.table() - Funktion in einen Data-Frame formatiert. Dazu übergibt man der read.table()-Funktion, den Namen beziehungsweise den Pfad, wo das CSV-Dokument gespeichert ist, ob das Dokument einen header hat, welches Trennzeichen verwendet wurde und welches Zeichen benutzt wurde für die Darstellung der Dezimalzahlen. Die Zeichen findet man einfach heraus, indem man die CSV-Datei einmal mit einem Editor öffnet und einfach nachschaut welche Zeichen genutzt wurden. Den Data-Frame speichere ich in eine Variable namens **data**.

b - Berechnen Sie die Mittelwerte und Varianzen der einzelnen Variablen mit dem geeigneten apply Befehl

Um die Mittelwerte und Varianzen der einzelnen Variablen vom oecdM-Datensatz zu berechnen schaue ich mir erst einmal mit der head()-Funktion die ersten paar Daten des Datensatzes an. Ich sehe, dass die Länder in der ersten Spalte, namens „X“ und dessen dazugehörigen Variablen in den nachfolgenden Spalten stehen. Um nun die Varianzen und Mittelwerte zu berechnen nutze ich die **lapply-Funktion(),** mit der ich über den Data-Frame iterieren kann, um somit für jede Variable, für die es möglich, den Mittelwert und die Varianz zu berechnen. Also übergibt man der **lapply()-Funktion** den Data-Frame, die Funktion welche man auf jede einzelne Variable anweden möchte und als letztes den Parameter **na.rm = TRUE**, was hierbei wichtig ist, da unser Datensatz lücken enthält. Ich sage der Funktion hiermit, dass sie alle NA-Werte vor der Berechnung löschen soll und dann die Berechnung durchführen soll. Als letztes gebe ich noch die Liste der Mittelwerte und Varianzen aus.

c - Überprüfen Sie, ob die Niederlande in der Länderliste des Datensatze auftaucht. Gibt es auch einen Eintrag für China?

Um herauszufinden, ob die Niederlande oder China in dem Datensatz sind , kann man mittels einem eingebauten R-Befehls „**%in%**“ , welche einen boolschen Wert zurückgibt, gucken ob ein bestimmter Wert in einer Liste vorhanden ist. Es wird **TRUE** zurückgegeben, wenn es wahr ist und **FALSE**, wenn es falsch ist. Wir schauen also nun mit dem Befehl, ob der String der Niederlande und von der China in der Liste der Ländernamen vorhanden ist, wobei wir die Liste der

Ländernamen erhalten, wenn wir auf die erste Spalte des Data-Frames zugreifen. Man sieht, dass die Niederlande dabei sind, China dagegen ist jedoch nicht vorhanden.

d - In welchem Land waren die meisten Jugendlichen mindestens zweimal betrunken? Wie hoch ist der maximale Prozentsatz?

Um das Land zu finden, nutzt man die **which.max()-Funktion**, womit man den Index erhält, an welchem der Wert der höchste ist und speichert diesen Wert in einer **indexAlkohol** Variablen. Mit dieser Information kann man auf den Data-Frame zugreifen, indem man den auf die Spalte der **indexAlkohol** schaut, welches ja das Land mit dem höchsten Alkoholkonsum ist. Man erhält Dänemark als das Land mit dem höchsten Alkoholwert und als Wert erhält man 24,8.

e - In welchem Land ist die Säuglingssterblichkeit am geringsten? Wie hoch ist sie in diesem Land?

Man führt die selbe Prozedur wie in der Aufgabe d. durch, wobei man diesmal ja den niedrigsten Wert haben möchte, weshalb man hier die **which.min()-Funktion** nutzt. Als Ergebnis erhält man Island mit dem Wert 2,3.

f - In welchen Ländern ist der Prozentsatz an Jugendlichen, die sich regelmäßig bewegen, kleiner als der Durchschnitt?

Dafür braucht man zuerst einmal den Durchschnitt der Bewegung, welchen man mittels der **mean()-Funktion** ermitteln kann. Den Durchschnitt speichert man wieder in einer Variable namens **averageMovement**, da man diesen Wert bei der Berechnung benötigt. Nun will man einen Vektor erzeugen, der alle Indexe der Länder erhält, die unter dem Durchschnitt liegen, dies kann man mit der **which()-Funktion** realisieren. Man übergibt der Funktion die Spalte mit den Werten der Bewegungen und fragt ab, welche Werte der Spalte kleiner sind als **averageMovement**. Man erhält einen Vektor mit den Indexen, welchen ich **indexBelowAverage** nenne. Um nun die Ländernamen auszugeben, greift man nur auf die Data-Frame-Zeilen mit den gespeicherten Indexen von i**ndexBelowAverage** zu.

Aufgabe 2: Häufigkeiten und Stripcharts

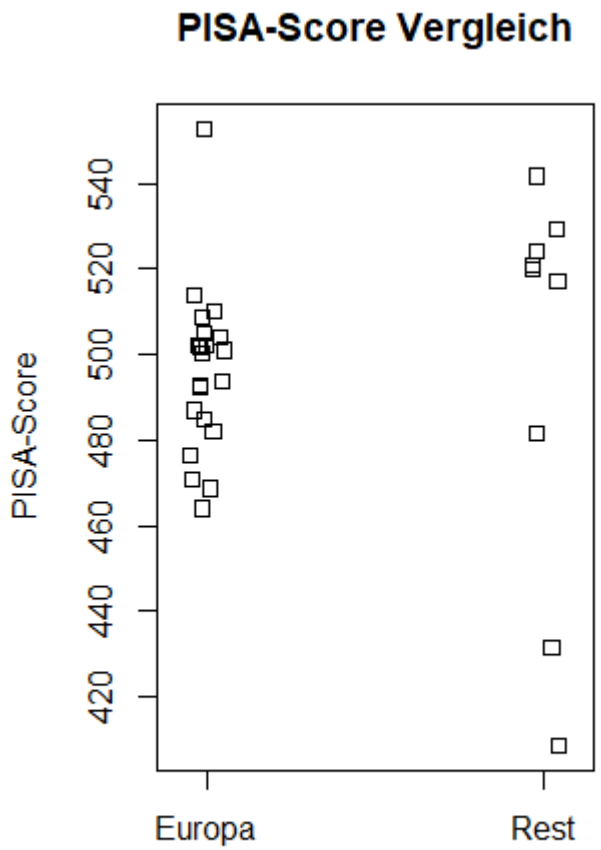
a - Wieviele Länder im Datensatz oecdM gehören zu Europa, wieviele zum Rest der Welt? Stellen Sie das Ergebnis in einem Kuchendiagramm dar und verwenden Sie dazu die Farben grün und blau

Um diese Aufgabe zu bewältigen, nutzt man die **split()-Funktion**, welche einem einen gegebenen Data-Frame nach einer Variable aufteilt, wobei man danach mehrere Data-Frames erhält, die dann nach der angegebenen Variable unterteilt sind. Also erstellet man nun einen neuen Data-Frame namens **splitDataRegion** mit der Funktion und übergibt als „Teilungsparameter“ die **Geo**-Variable, da man ja den Data-Frame nach dieser Variable aufteilen möchte, in Länder die zu Europa gehören und Länder welche nicht zu Europa gehören. Damit man nun die jeweiligen Anzahlen erhält, schaut man einfach wie viele Zeilen diese Teil-Data-Frames enthalten, bzw. in anderen Worten: Wie viele Länder sind Teil Europas und wie viele sind vom Rest der Welt? Dies kann man mittels der **nrow()-Funktion** machen, die einem die Anzahl der Zeilen eines Data-Frames zurückgibt. Wir führen also nun die **nrow()-Funktion** auf beide Teil-Data-Frames durch und speichern die Anzahlen in zwei Variablen namens **belongsEurope** und **belongsRest**. Das Kuchendiagramm erstellen wir mit der **pie()-Funktion** und übergeben dabei die beiden Variablen mit den Anzahlen als Vektor, die dazugehörigen Labelnamen „Rest der Welt“ und „Europa“ verknüpfen wir mit der **paste()-Funktion** zu einem String der den Labelnamen an sich enthält, ebenso wie die Anzahl. Diese übergeben wir auch als Vektor, an den Parameter **labels**. Um die Farben zuzuweisen, übergeben wir der **pie()-Funktion** noch den Parameter **col** mit den Werten **„blue“** und **“green“** als Vektor um die Farben zuzuweisen. Zuletzt fügen wir noch einen Titel hinzu, mit Parameter **main**.
Wir erhalten:

Also bestehen die Daten aus 21 Europäischen Ländern und 9 Ländern aus dem Rest der Welt.

b - Visualisieren Sie die Variable Lesen, getrennt nach dem Faktor Geo, in einem vertikalen Stripchart. Welche Aussage können Sie mit diesem Stripchart treffen?

Um einen Stripchart zu erstellen nutzen ich die **stripchart()-Funktion** . Da ich die Variable **Lesen** getrennt nach dem Faktor **Geo** visualisieren möchte, kann ich dies mit der Formel **data$Lesen ~ data$Geo** der **stripchart()-Funktion** übergeben. Dies bedeutet anders gesagt, dass man die erste Variable getrennt nach den Parametern der zweiten Variable teilen möchte. Um den Stripchart noch etwas schöner darzustellen, übergebe ich noch die Namen der Gruppen als Vektor, die Darstelungsethode **„jitter“** und die Beschriftungen. Ebenso möchte ich die Vertikale Ansicht erhalten und übergebe deshalb den Parameter **vertical** mit dem Wert **TRUE**. Es kommt raus:



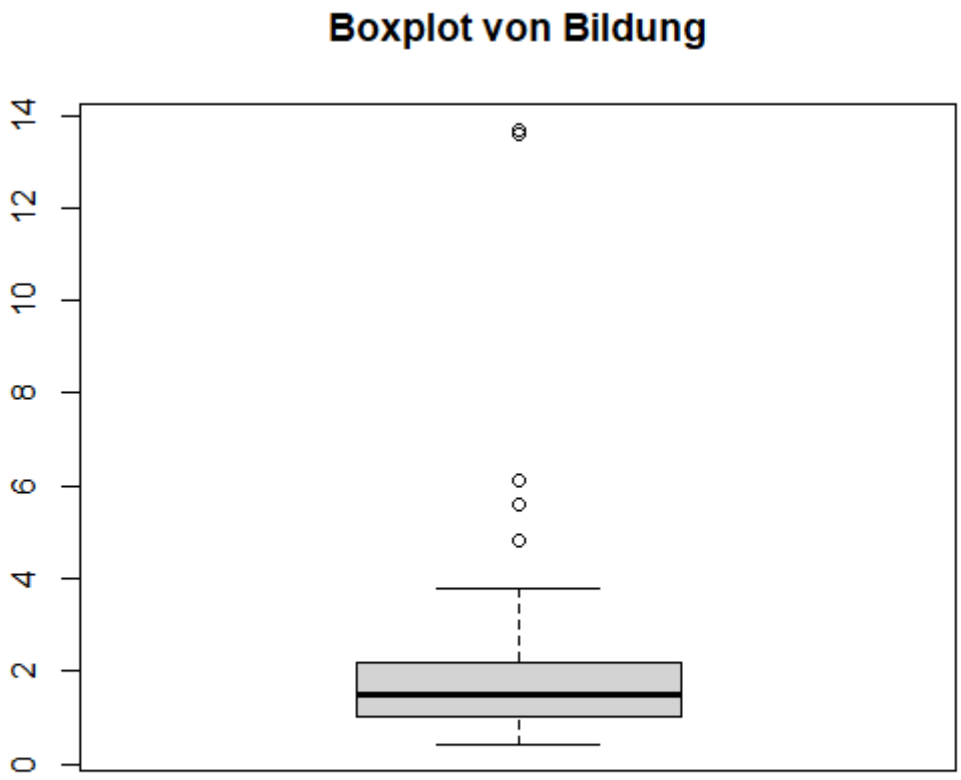
Es fällt auf, dass die Länder vom Rest der Welt im Vergleich zu Europa entweder zum Großteil besser als die Europäischen Länder sind, es aber auch Aussreiser nach unten gibt welche wesentlich

schlechter als die Europäischen Länder sind. Die Varianz vom Rest der Welt ist deutlich größer als die Varianz von den Europäischen Ländern. Die Europäischen Ländern sind alle relativ zentriert um einen Punkt herum, mit einem starken Aussreiser nach ganz oben. Aus Interesse wollte ich wissen welches Land dieser starke Aussreiser ist und habe mir mal das Land mit dem Max-Pisa-Score ausgegeben. Das Land ist Finnland, mit einem Score von 552,7.

Aufgabe 3: Quantile und Plots

a - Erstellen Sie einen Boxplot für die Variable "Bildung". Was fällt Ihnen auf?

Um einen Boxplot der Variable Bildung zu erstellen nutzt man die **boxplot()-Funktion** und übergibt die Variable Bildung. Das sieht dann wie folgt aus:



Man sieht, dass der Anteil an Kindern, die ohne eine Grundausstattung für Bildung auskommen müssen, einen Median von ungefähr 2% hat. Die mittleren 50% der Daten befinden sich ungefähr zwischen etwas über 0% und 4%. Es gibt ein paar schwache Aussreisser und zwei sehr starke Aussreisser die bei ca. 14% liegen. Somit hat der Großteil der Länder einen relativ geringen Anteil

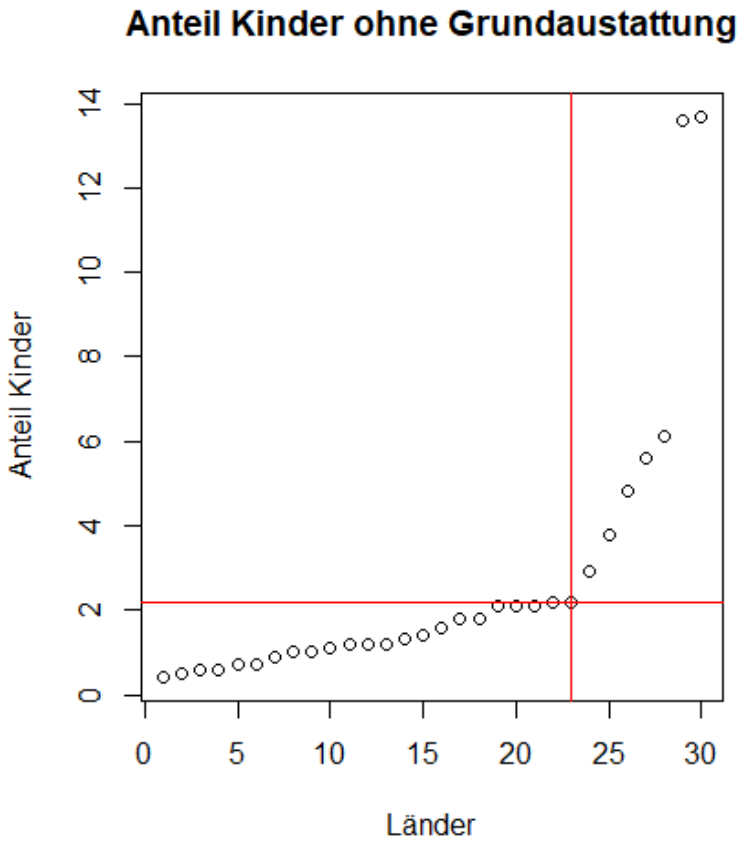
von Kindern, die ohne eine Grundausstattung auskommen müssen, die restlichen weichen aber stark davon ab.

b - Untermauern Sie die Beobachtung aus Aufgabe (a) durch Berechnung einiger Quantile mit Hilfe der Funktion quantile()

Hierfür übergibt man der **quantile()-Funktion** die Variable Bildung und den Parameter **na.rm = TRUE**, da man ja wieder die nicht vorhandenen Daten vor der Berechnung entfernen möchte. Man erhält, dass der Wert in 75% der Daten unter 2,2% liegt und nur in den restlichen 25% der Daten über 2,2% bis 13,7%. Dies untermauert die Aussage, dass der Großteil der Ländern einen relativ geringen Anteil hat, der Rest jedoch einen stark davon abweichenden Anteil, nämlich bis zu 13,7%.

c - Stellen Sie zudem die aufsteigend geordneten Werte der Variable Bildung mit Hilfe der Funktion plot() als Kurve dar

Um die aufsteigend geordneten Werte der Variable Bildung darzustellen, muss man erstmal die Werte der Bildung aufsteigend sortieren. Um dies durchzuführen nutze ich die **lapply()-Funktion** in Verbindung mit der **sort-Funktion()** und übergebe der **lapply-Funktion** die Liste mit den Werten von Bildung. Die Rückgabe wird in einer Variable namens **orderedEducation** gespeichert. Da man danach die geordneten Werte hat, kann man diese in einer Grafik darstellen, mittels der **plot()-Funktion**, welcher man **orderedEducation** übergibt. Da man in der vorherigen Aufgabe herausgefunden hat, dass 75% der Daten unter 2,2% liegen, wollte ich diese Grenze nochmal im Plot verdeutlichen und habe mittels der **abline()-Funktion** eine Linie in den Plot gezeichnet, welche die 75% der Daten Grenze darstellt. (75% der Daten von 30 Daten = 0,75*30 = 22,5 bzw. ab 23) Ebenso wollte ich die Grenze von 2,2% in den Plot zeichnen, was ich ebenfalls mit gleicher Funktion gemacht habe.



d - Begründen Sie anhand Ihrer Beobachtungen, dass das 75% Quantil der Daten einen guten Trennpunkt zwischen Ländern mit “guter” und “schlechter” Grundausstattung für Bildung darstellt

Anhand des Plots kann man noch sehr gut sehen, dass 75% der Länder im Bereich von 2,2% liegen und die restlichen 25% der Daten stark von den 75% abweichen. Ebenso sieht man, dass die unteren 75% fast eine Gerade bilden, woran man erkennen kann, dass diese Länder sich nicht stark voneinander verschieden sind, wobei man dies von den oberen 25% der Daten nicht behaupten kann. Ab der 75% Grenze fängt der Anteil stark an zu wachsen , was fast einer Exponentiell steigenden Funktion ähnelt. Anhand des Plots kann man also gut erkennen, dass das 75% Quantil eine gute Trennlinie zwischen „gut“ und „schlecht“ darstellt.

Tag 2: Statistische Test, Regression, ANOVA

Aufgabe 4: Annahmen des t-Tests

a - Ziehen Sie 100 exponential-verteilte Zufallszahlen mit dem Parameter λ = 0.1 und speichern Sie diese in dem Objekt X1. Erstellen Sie analog ein Objekt X2 von 100 Zufallszahlen, für die Sie erneut 100 exponential-verteilte Zufallszahlen HX2 (H wie Hilfsvektor) mit dem Parameter λ = 0.1 ziehen und anschließend alle Elemente von 20 subtrahieren. Ein Element i des Vektors X2 berechnet sich also als X2[i] = 20−X2[i]

Um zwei Vektoren mit Exponentiell verteilen Zufallszahlen zu erstellen nutzt man die **rexp()-Funktion**. Dieser Funktion übergibt man die Anzahl, wie viele Zahlen generiert werden sollen und den Parameter **rate** mit 0,1. Um nun die Elemente des Vektors X2 zu berechnen, wendet man mittels einer for-Schleife die gegebene Vorschrift für jedes Element an.

b - Führen Sie den t−Test durch, um zu untersuchen, ob die beiden Objekte unterschiedliche Mittelwerte besitzen

Da man bei Zufall reproduzierbar sein möchte setzten wir einen **seed** = 2023. Um nun zu untersuchen, ob die beiden Objekte verschiedene Mittelwerte besitzen, wendet man die **t.test()-Funktion** an, und übergibt beide Objekte mit Komma getrennt, da man hier keine gepaarten Daten hat. Das Konfidenzlimit beträgt 95%, wenn man dazu keine Angabe macht. Man erhält als p-Wert = 0,1439 (mit seed = 2023) , was nicht kleiner als alpha = 0,05 ist ⇒ H_0 nicht ablehnbar.

H_0 = Differenz der Durchschnitte ist gleich 0. Dies bedeutet, dass die Objekte mit 95%iger Sicherheit gleiche Mittelwerte besitzen.

c - Führen Sie den Wilcoxon-Rangsummen-Test durch, um zu untersuchen, ob die beiden Objekte unterschiedliche Mediane besitzen

Um den Test durchzuführen nutzt man die **wilcox.test()-Funktion** und übergibt als Parameter die beiden Objekte. Man erhält als p-Wert = 0,002908 (mit seed = 2023), was kleiner als alpha = 0,05 ist ⇒ H_0 ablehnbar ⇒ H_1 annehmen, wobei H_1 = unterschiedliche Mediane.

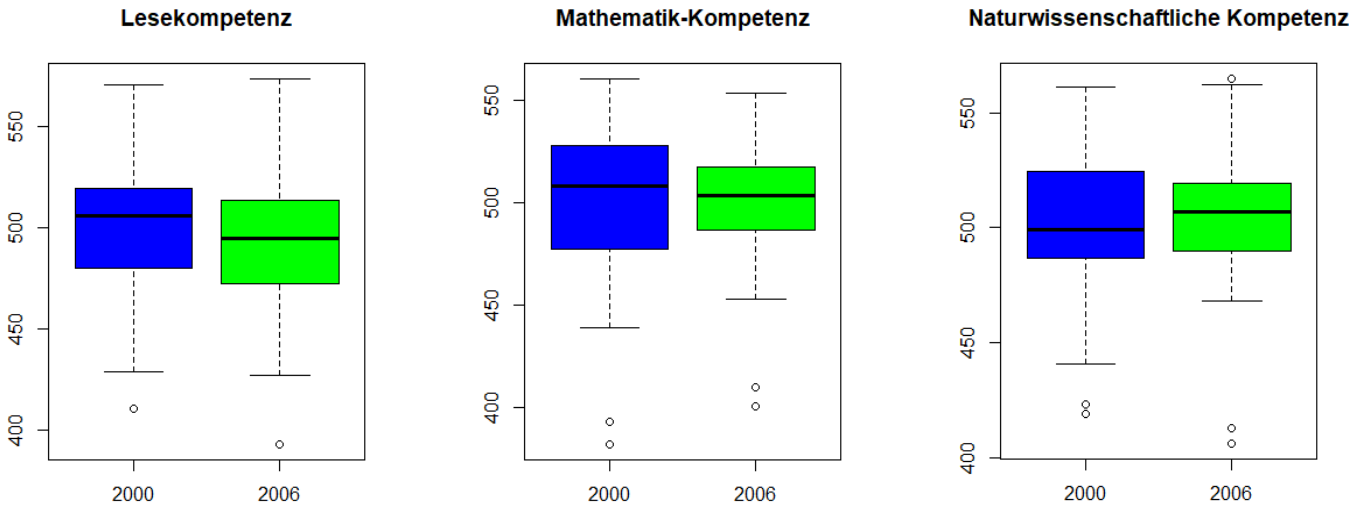
Aufgabe 5: Testen an PISA-Daten

a - Laden Sie den Datensatz PISA.csv von der Homepage herunter und lesen sie ihn ein

Den Datensatz habe ich mittels read.table Funktion eingelesen, als Trennzeichen wurde das Komma genutzt und als Dezimalzeichen der Punkt.

b - Untersuchen Sie deskriptiv (Boxplots aller 6 Variablen), ob sich die drei PISA-Scores des Jahres 2006 im Vergleich zum Jahr 2000 verändert haben. (Gehen Sie hierbei und im weiteren nicht näher auf irgendwelche Geschlechtsunterschiede ein)

Mittels **boxplot()-Funktion** habe ich die Boxplots der verschiedenen Variablen erstellt.



Lesekompetenz: Man sieht, dass die Mediane ein wenig voneinander abweichen, jedoch nicht gravierend, wobei der des Jahres 2006 etwas geringer ausfällt. Es gibt einen Aussreiser bei beiden Jahren, dieser fällt auch ungefähr gleich aus. Der Interquartilsbereich ist im Jahre 2006 ein klein wenig nach unten verschoben. Lower- und Upperfence sehen fast identisch aus. Ich denke daher, dass sich der PISA-Score nicht wesentlich verändert hat.

Mathematik-Kompetenz: Man sieht, dass der Interquartilsbereich von 2006 viel größer ausfällt, aus der von 2006, wobei die Mediane aber fast identisch sind. Lower- und Upperfence fallen im Jahre 2006 auch höher, bzw. niedriger aus als im Jahre 2006. Die 2 Aussreisser vom Jahre 2006 sind ein wenig nach oben gerutscht, was ebenso für eine Verbesserung der Kompetenz spricht. Es sieht im Insgesamten so aus, dass die Kompetenz „konzentrierter“ um einen Punkt herum liegt. Deshalb vermute ich, dass sich der PISA-Score verändert hat.

Naturwissenschaftliche Kompetenz: Man sieht, dass der Median im Jahr 2006 etwas höher angesiedelt ist als der Median im Jahr 2000, was bereits für eine Verbesserung der Kompetenz spricht. Ebenso hat sich die Lowerfence im Jahr 2006 deutlich nach oben verschoben. Dahingegen gibt es zwei schlechte Aussreisser, welche denen aus dem Jahr 2000 ähneln, jedoch liegen diese im Jahr 2006 etwas weiter unten und weisen einen etwas größeren Unterschied untereinander auf. Außerdem gibt es einen „guten“ Aussreiser, welcher vorher im Jahr 2000 nicht vorhanden war, diese ist aber nur ganz leicht über dem Upperfence. Der Interquartilsbereich hat sich verkleinert. Der Upperfence liegt bei beiden Jahren ungefähr auf dem gleichen Niveau. Ingesamt sieht es so aus, dass sich der PISA-Score der Naturwissenschaftlichen Kompetenz verbessert hat.

c - Untersuchen Sie mit einem geeigneten Test, ob sich die drei PISA-Scores signifikant verändert haben

Man muss vorher die Variablen auf Varianzhomogenität und auf Normalverteilung testen um den richtigen Test auswählen zu können, da verschiedene Tests verschiedene Anforderungen haben.

Test auf Normalverteilung:

Mittels Shapiro-Wilk-Test habe ich auf Normalverteilung getestet. Die **shapiro.test()-Funktion** bietet die Möglichkeit den Test in R durchzuführen. (H_0 = Normalverteilung liegt vor) Dabei kam folgendes raus:

R00 – p = 0.07609 ⇒ normalverteilt **&** R06 – p = 0.7889 ⇒ normalverteilt

M00 – p = 0.005166 ⇒ **nicht** normalverteilt **&** M06 – p = 0.006494 ⇒ **nicht** normalverteilt

S00 – p = 0.2752 ⇒ normalverteilt **&** S06 – p = 0.0009737 ⇒ **nicht** normalverteilt

Danach wollte ich auf Varianzhomogenität testen, wobei ich dafür die Variablen R00 und R06, die normalverteilt sind, mittels F-Test untersuche (**var.test()-Funktion**). Da dieser Test aber sensitiv gegenüber Abweichungen der Normalverteilung ist, muss ich für die anderen Variablen einen anderen Test nutzen. Dafür nutze ich den Ansari-Bradley Test (**ansari.test()-Funktion**).

R00 – R006 = p = 0.6979 ⇒ Varianzhomogenität gegeben

M00 – M006 = p = 0.03509 ⇒ Varianzhomogenität **nicht** gegeben

S00 – S006 = p = 0.2861 ⇒ Varianzhomogenität gegeben

Somit darf man für die Variable **R** den t.test verwenden, für die anderen beiden Variablen muss man den Wilcoxon-Rangsummen-Test (**wilcox.test()-Funktion**) nutzen.
Dabei sehen wir, dass keiner der PISA-Scores sich signifikant verändert hat, mit einem Konfidenzlimit von 95%.

Aufgabe 6: Produktionskontrolle bei Hustensaft

Laut Produktbeschreibung enthält der Hustensaft Mikasolvan 40g/Liter des Wirkstoffes Halsruhe. Die Produktion des Saftes wird angehalten, wenn die Konzentration des Wirkstoffes deutlich zu niedrig ist. Um die Produktion zu überprüfen, wurden 9 Flaschen zufällig entnommen und die Konzentration von Halsruhe gemessen. Die Messwert finden Sie in der Datei Hustensaft.csv. Begründen diese Daten einen Produktionsstopp?

Um herauszufinden, ob die Messwerte einen Produktionsstopp begründen, habe ich mich für den Wilcox-Test entschieden, da wir nur 9 Daten vorliegen haben und man somit genaue parameterfreie Tests bevorzugen sollte. Also habe ich der **wilcox.test()-Funktion** die Daten des Datensatzes übergeben und mein $\mu = 40$ gesetzt, da die Konzentration des Wirkstoffes 40g/Liter betragen sollte. Da wir hier testen wollen, ob der Wirkstoff deutlich zu **niedrig** ist, wollen wir einen einseitigen Test durchführen, weswegen wir als Alternativhypothese „**less**“ angeben.
Als Konfidenzlimit setzte ich 95% fest. Der p-Wert beträgt $0{,}08203 > 0{,}05 \Rightarrow H_0$ nicht ablehnbar, was bedeutet, dass die Konzentration **nicht** signifikant von 40g/Liter abweicht, mit einem Konfidenzlimit von 95%.

Aufgabe 7: Lineare Regression

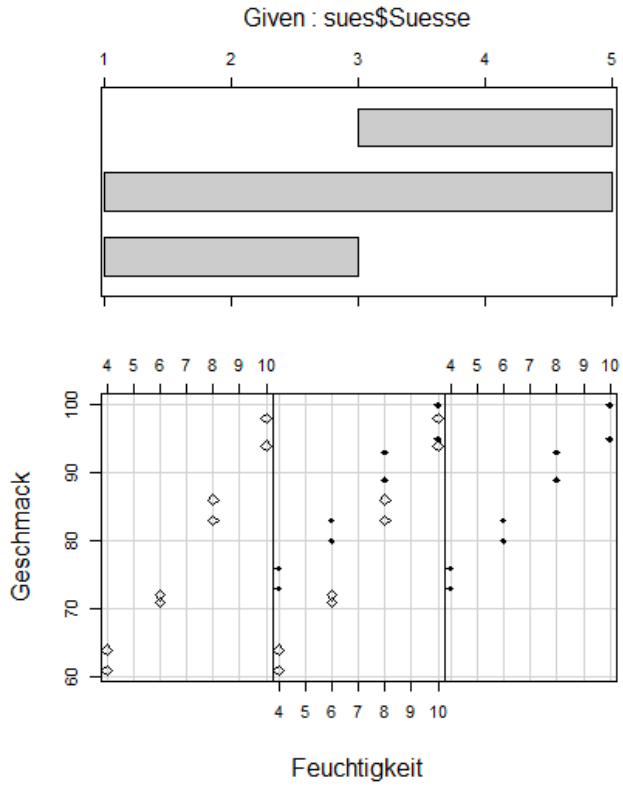
a - Laden Sie den Datensatz Suess.csv und speichern Sie ihn in einem Dataframe sues ab

Ich habe den Datensatz per **read.table()-Funktion** eingelesen und in einem Dataframe sues abgespeichert.

b - Benutzen Sie die Funktion coplot() um einen Plot von Geschmack abhängig von der Feuchtigkeit der Süßigkeit, bedingt auf den Süßegrad zu erstellen. Benutzen Sie für eine bessere Darstellung die Optionen pch = c(5,18), rows = 1 und columns = 3. Gibt es bei gegebener Feuchtigkeit einen Einfluss der Süße auf den Geschmack?

Um einen Plot vom Geschmack abhängig von der Feuchtigkeit der Süßigkeit, bedingt auf den Süßegrad zu erstellen, muss man der **coplot()-Funktion** eine Formel übergeben, die wie folgt lautet: **Geschmack ~ Feuchtigkeit | Suesse**, was genau soviel bedeutet wie: Zeige mir den

Geschmack in Abhängigkeit der Feuchtigkeit bedingt auf den Süßegrad. Als weitere Parameter habe ich die Parameter der Aufgabenstellung übernommen und noch die x- und y-Achsenbeschriftung hinzugefügt. Es kommt raus:



Man sieht einen deutlichen Einfluss der Süße auf den Geschmack, bei gegebener Feuchtigkeit, da der Geschmack ungefähr linear ansteigt, woran man auch erkennt, dass sie korrelieren.

Die schwarze Punkten beschreiben die Feuchtigkeit (x-Achse) mit Süßegrad 4, die unausgefüllten Punkte beschreiben die Feuchtigkeit mit Süßegrad 2. Anhand des Plots kann man sehr gut erkennen, dass die höhere Süße (4) einen positiven Einfluss auf den Geschmack hat. Dies erkennt man unter anderem an den schwarzen Punkten, die immer ein wenig weiter oben sind , als die unausgefüllten Punkte ⇒ mehr Geschmack.

Gegen „Ende“ bzw. dort wo die Feuchtigkeit den Wert 10 erreicht, schwindet der Unterschied allmählich, ist aber trotzdem noch vorhanden. Also hat die Bedingung der Süße einen positiven Effekt, in Kombination mit der Feuchtigkeit, auf den Geschmack.

c - Fitten Sie das Modell $Geschmack_i = \beta0 + \beta1 \cdot Feuchtigkeit_i + \beta2 \cdot Suessei + \varepsilon i$

Um das Modell zu fitten, führt man eine lineare Regression mit der **lm()-Funktion** durch. Als abhängige Variable übergibt man den Geschmack und als unabhängige Variablen übergibt man die Feuchtigkeit + den Geschmack. Dieses Modell nenne ich **linear.model** und lasse mir im Anschluss eine Zusammenfassung des Modells ausgeben. Heraus kommt das Modell:

$Geschmack_i = 37{,}525 + Feuchtigkeit_i * 4{,}8 + Suesse_i * 3{,}75 + \varepsilon_i$, mit $R^2 = 0{,}9462$

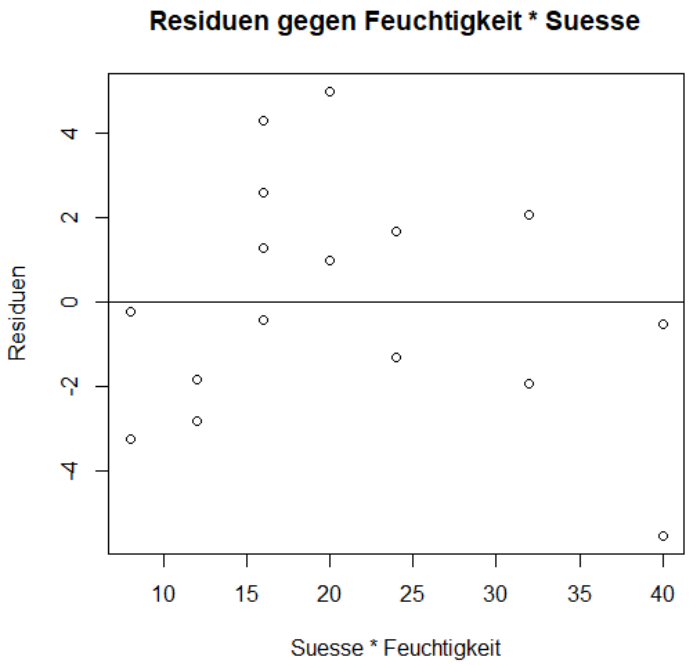
Da das Bestimmtheitsmaß R^2 fast 1 beträgt, erklärt das Modell die Varianz fast vollständig.

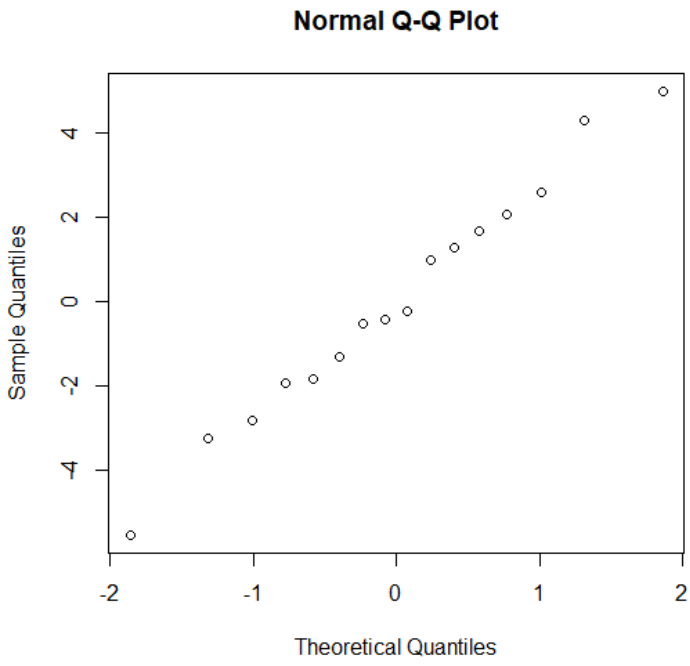
d - Plotten Sie die Residuen gegen Feuchtigkeit · Suesse

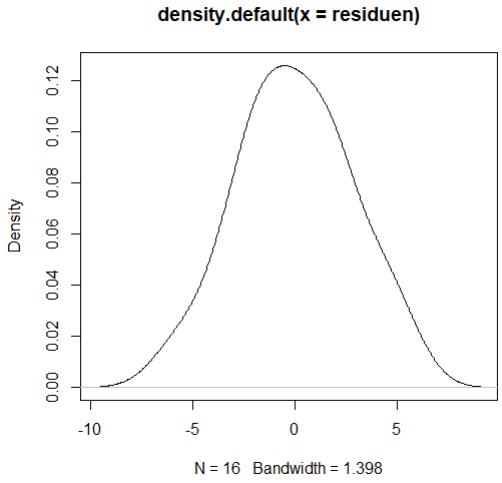
Um die Residuen zu plotten, muss man zuerst eine Liste der Residuen für jeden Wert erstellen. Dies kann man mithilfe der **resid()-Funktion** machen, wobei man der Funktion das **linear.model**

übergibt. Da man nun die Liste der Residuen hat, übergibt man der **plot()-Funktion** als x-Achse die Liste der Süße mal die Liste der Feuchtigkeit und als y-Achse die Liste der Residuen. Als nächstes habe ich noch eine Gerade in den Plot gezeichnet, mithilfe der **abline()-Funktion**. Dazu fügt man noch ein paar Beschriftungen hinzu und man erhält folgenden Plot.

Man sieht, dass die Residuen, bzw. die Fehler, nicht groß sind. Wären die Punkte alle auf der Geraden bei 0, so wäre das Modell perfekt, da es somit keine Fehler hätte bei der Vorhersage des Geschmacks. Man könnte eine umgedrehte Parabel, als Form der Punkte, erahnen. Somit sind die Punkte zu Beginn der Suesse*Feuchtigkeit meist im negativen Bereich. Mit Anstieg der Suesse*Feuchtigkeit, werden die Residuen größtenteils positiv und verlaufen dann wieder gegen Ende ins Negative. Um die Normalverteilung der Residuen, die gefordert ist bei der Nutzung von linearen Modellen, genauer bewerten zu können, habe ich nochmal einen QQ-Plot und einen Dichteplot der Residuen erstellt, an dem man besser sehen kann, wie die Residuen verteilt sind. Man sieht, das die Residuen normalverteilt sind.







e - Bestimmen Sie nun die Parameter des Modells mit Interaktionsterm Geschmacki = β0 + β1 · Feuchtigkeiti + β2 · Suessei + β3 · (Feuchtigkeiti · Suessei) + εi

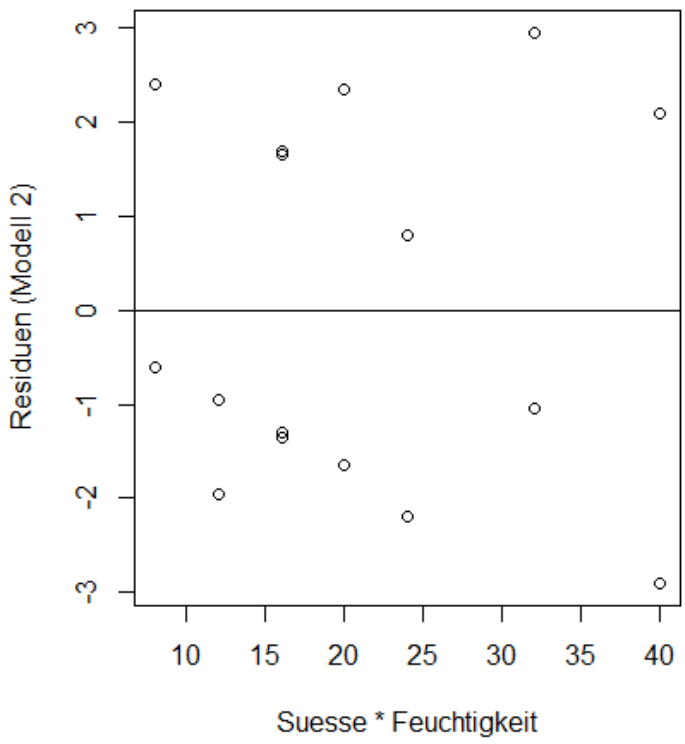
Um dieses Modell zu erstellen, nutze ich erneut die **lm()-Funktion**, mit den gleichen Parametern wie bei dem vorherigen Modell, mit dem einzigen Unterschied, dass ich noch die Feuchtigkeit*Suesse als unabhängige Variable hinzugefügt habe. Das Modell nenne ich **new.linear.model.** Ich lasse mir ebenso erneut eine Zusammenfassung des Modells mithilfe der **summary()-Funktion** ausgeben. Als Term erhalte ich:

$Geschmack_i = 19{,}15 + 7.425*Feuchtigkeit_i + 9.875*Suesse_i + (-0.875)*Feuchtigkeit_i*Suesse_i + \varepsilon_i$

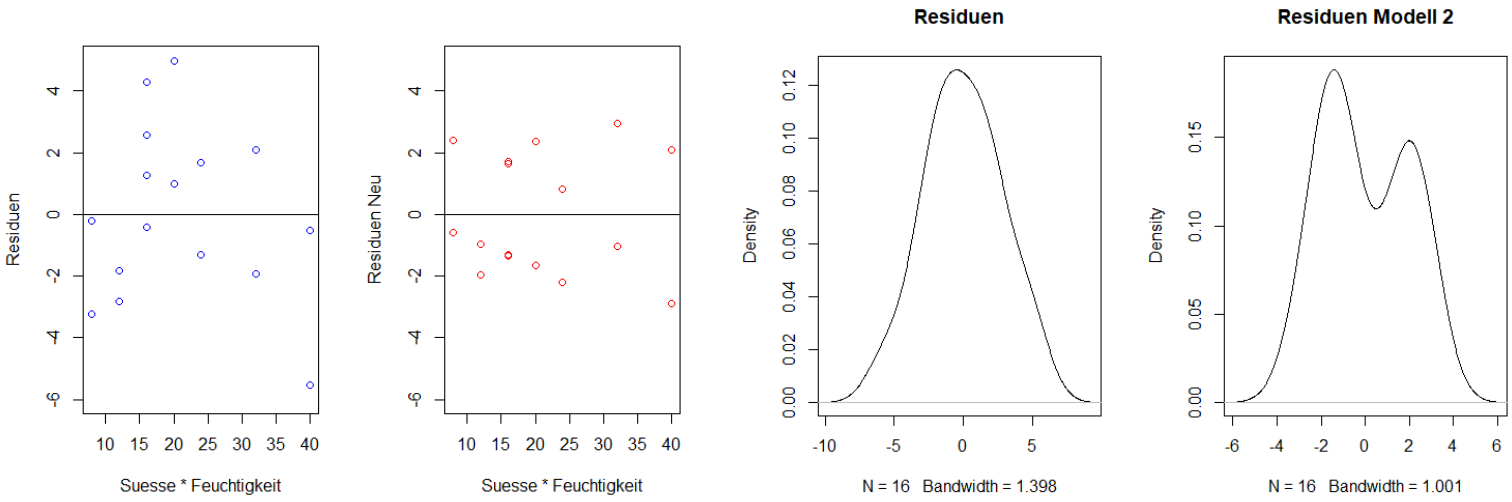
mit einem R^2 von 0,9742.

f - Verbessert sich die Anpassung des Modells an die Daten? Ist der Koeffizient β3 der Interaktion signifikant von 0 verschieden? Plotten Sie erneut die Residuen gegen Feuchtigkeit· Suesse und vergleichen Sie die Ergebnisse mit dem Plot aus Aufgabe (d)

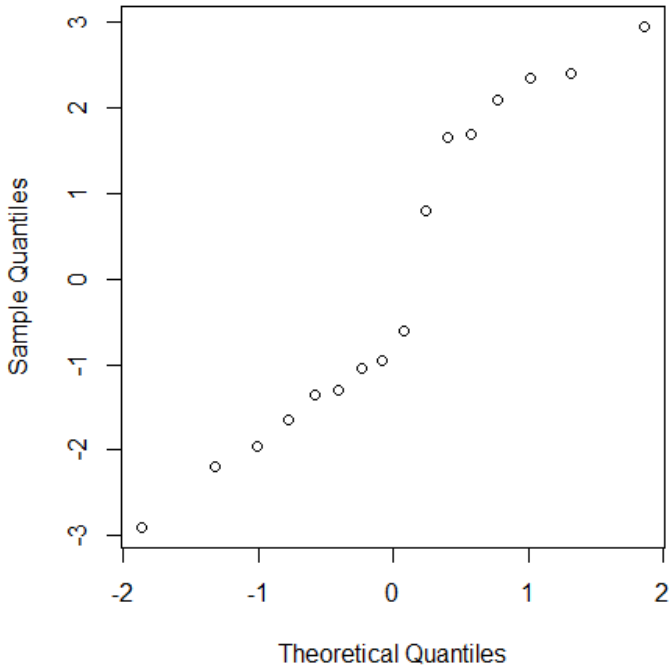
Das Bestimmtheitsmaß des neuen Modells ist um 0,028 besser, als des des vorherigen Modells. Um den neuen Plot zu erstellen, muss ich erneut eine Liste von den Residuen erstellen, was ich wie bei dem vorherigen Modell mache. Dabei kommt raus:



In diesem Plot kann man eine Zick-Zack Kurve der Residuen feststellen, wobei Sie relativ hoch starten, dann fallen bis ungefähr 25, wonach sie wieder ansteigen bis zu ca. 33 und danach wieder fallen. Auffällig sind auch die Residuen bei ca. 15, da sie fast aufeinander liegen. Ebenso erkennt man, anhand der Skalierung der y-Achse, dass die Residuen im allgemeinen kleiner geworden sind, da das vorherige Modell Residuen jenseits der 4 hatte. Ingesamt würde ich also sagen, dass das neue Modell besser als das vorherige Modell ist. Schauen wir uns aber nochmal die Verteilung der Residuen an und vergleichen beide Residuen Plots nebeneinander:



Man erkennt anhand der Dichtefunktion gut, dass die Residuen des neuen Modells an den Rändern zwar normalverteilt sind, in der Mitte hat die Dichtefunktion aber einen starken Knick, wodurch sie nicht annähern normalverteilt sind, woraus folgt, dass man dieses Modell eigentlich nicht benutzen darf (laut Skript). Dies sieht man ebenfalls gut an dem QQ-Plot:



Man sieht deutlich, dass die Residuen nicht normalverteilt sind, da sie nicht auf einer Geraden liegen.

Der Koeffizient **ß3** ist signifikant von 0 verschieden, wenn man davon ausgeht, dass signifikant heißt: Über 0,5 oder unter 0,5. Er beträgt = -0,875. Insgesamt ist das zweite Modell also besser als Modell 1, man sollte es aber nicht benutzen, da die Residuen nicht normalverteilt sind.

Aufgabe 8: Korrelationen

In dieser Aufgabe müssen Sie eine eigene Untersuchung durchführen überlegen. Verwenden Sie dazu die Datenbank der EU und laden sich dort mindestens zwei Datensätze A und B

herunter (z.B Geburtenrate und Anzahl der Verkauften PKWs), welche eine Größe in Abhängigkeit der Jahreszahl angeben. Stellen Sie sicher, dass zumindest einer der Datensätze nicht nur eine Aufschlüsselung nach Jahreszahl sondern auch über die verschiedenen Länder der EU beinhaltet

Für die eigene Untersuchung habe ich mir überlegt, die Korrelation zwischen der Anzahl an Flugpassagieren und dem jährlichen Einkommen der Bürger von Deutschland anzuschauen. Dafür habe ich mir von der Datenbank der EU folgende Datensätze ausgesucht:

International intra-EU air passenger transport by reporting country and EU partner country: https://ec.europa.eu/eurostat/databrowser/view/AVIA_PAINCC__custom_4871302/default/table?lang=en

Mean and median income by age and sex – EU-SILC and ECHP surveys: https://ec.europa.eu/eurostat/databrowser/view/ILC_DI03__custom_4871298/default/table?lang=en

Da dies beides große Datensätze mit, für meinen Fall, viel nicht relevanter Informationen sind, habe ich mir bereits vorher auf der Internetseite die unnötigen Informationen aus der Tabelle entfernt (zum Beispiel die Daten für die Jahre vor 2008 und nach 2019) und mir danach die Datensätze heruntergeladen. Ich habe nur die Daten von 2008 bis 2019 gewählt, weil die Daten davor Lücken aufweisen und die Daten danach von Corona starkt beeinflusst worden.

Somit bestanden meine Datensätze nur noch aus:

Datensatz A : Median-Einkommen pro Jahr in € von 2008-2019 für alle Länder der EU (Kaufkraft bereinigt). Dabei habe ich mich für das Median-Einkommen entschieden, weil ich nicht wollte, dass Ausreißer eine Auswirkung auf den Wert haben

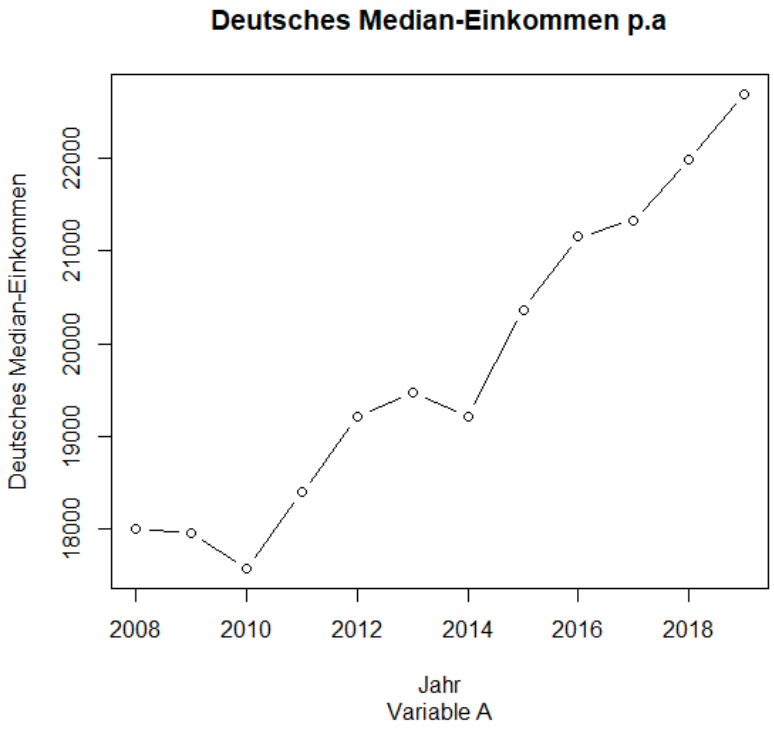
Datensatz B : Anzahl der Flugpassagiere (bei Abflug) pro Jahr von 2008-2019 für alle Länder der EU (Die Anzahl beschreibt die Flugpassagiere die bei Abflug an Board sind, wobei der Flug auch wieder innerhalb der EU landet)

Die heruntergeladenen Tabellen habe ich noch ein wenig in Excel umstrukturiert, um das Einlesen in R einfacher zu gestalten. Zum Schluss sahen meine

a - Visualisieren Sie beide Größen A und B in Abhängigkeit der Jahreszahl

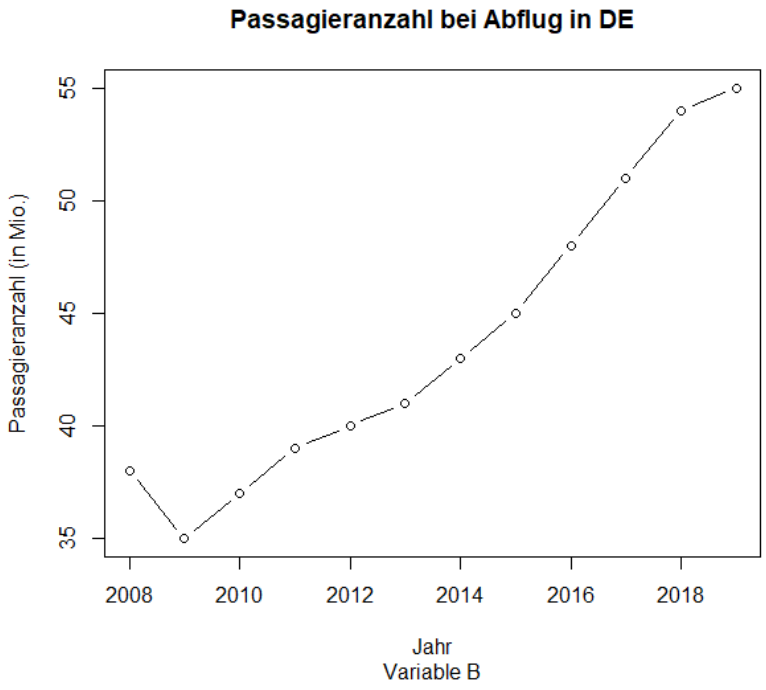
Um die Daten zu visualisieren, habe ich erst einmal beide Datensätze eingelesen. Den Datensatz A nenne ich **flug** und den Datensatz B nenne ich **einkommen**. Danach lasse ich mir die ersten paar Einträge mittels der **head()-Funktion** für beide Datensätze ausgeben. Die Zeile vom **einkommen** mit den Werten von Deutschland, speichere ich mir in einer neuen Variable namens **gerEinkommen**. Um die nun die Daten für das deutsche Median-Einkommen zu visualisieren, nutze ich die **plot()-Funktion**, wobei ich als x-Werte die Jahreszahlen von 2008 bis 2019 übergebe

und als y-Werte meine **gerEinkommen** Variable. Als weitere Parameter übergebe ich noch die Achsenbeschriftungen, Titel und die Darstellungsart, um eine „kurvenähnliche“ Form zu visualisieren. Ich erhalte dabei folgenden Plot:



Man sieht, dass das deutsche Median-Einkommen fast stetig wächst, mit zwei Ausnahmen, nämlich um die Jahre 2010 (Wirtschaftskriese) und 2014 (keinen Grund gefunden). Insgesamt sieht der Graph so aus, als würde er eine Treppe steigen.

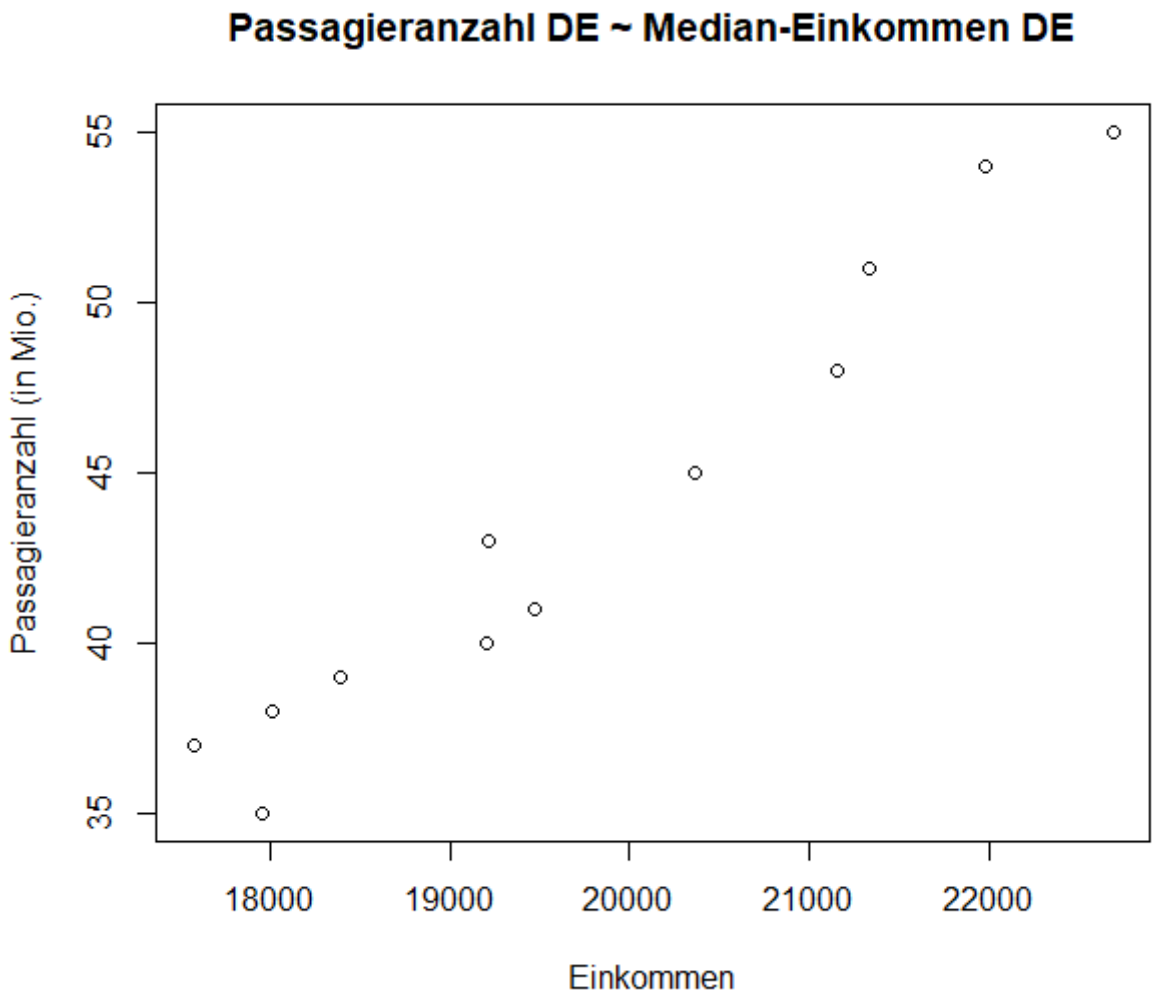
Als nächstes möchte ich die Anzahl der Flugpassagiere in Abhängigkeit der Jahreszahl visualisieren. Dazu erstelle ich wieder eine Variable namens **gerFlug**, welche die Flugdaten zu Deutschland enthält. Da es bei dem Plot, aufgrund zu großer Zahlen, zu einer unschönen Darstellung der y-Achse kommt, nutze ich die **prettyNum()-Funktion**, welche mir meine Zahlen im Millionenbereich zu Zahlen im Zehnerbereich umformt. (z.B. 30.000.000 → 30). Heraus kommt der Plot der links zu sehen ist. Hierbei erkennt man ebenso einen Fall um 2009 (Wirtschaftskriese). Davon abgesehen, sieht der Graph annähernd gerade aus und ist nach 2009 streng monoton steigend.



b - Visualisieren Sie die Abhängigkeit ihrer Größen A und B von einander und berechnen Sie die Korrelation und die Signifikanz ihrer Korrelation

Um herauszufinden, ob die Korrelation signifikant ist, lege ich ein Konfidenzlimit von 95% fest.

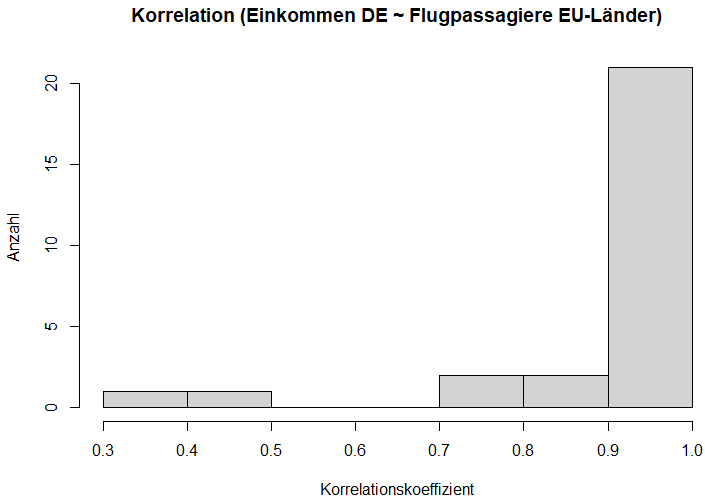
Um die Korrelation zwischen meinen Variablen zu visualisieren, übergebe ich der **plot()-Funktion** für die x-Werte, die **gerEinkommen** Variable und für die y-Werte, die **gerFlug** Variable. Dabei kommt folgender Plot heraus:



Man erkennt anhand des Plots eine stark positive Korrelation, der beiden Variablen, da die Punkte die Form einer Geraden mit positiver Steigung skizzieren. Um den genauen Korrelationskoeffizienten zu berechnen, nutze ich die **cor.test()-Funktion**, und übergebe ihr als Parameter die beiden Variablen. Bei dem Test kommt heraus, dass der Korrelationskoeffizient den Wert **0,978065** hat, was nach dem vorherigen Plot keine Überraschung darstellt. Der p-Wert der Korrelation beträgt **0,00003855**. Dies bedeutet, dass diese Korrelation hoch signifikant und positiv ist.

c - Bestimmen Sie die Länder welche die größte und die kleinste Korrelation mit Deutschland der Größe A bzw. B aufweisen. Visualisieren Sie dies

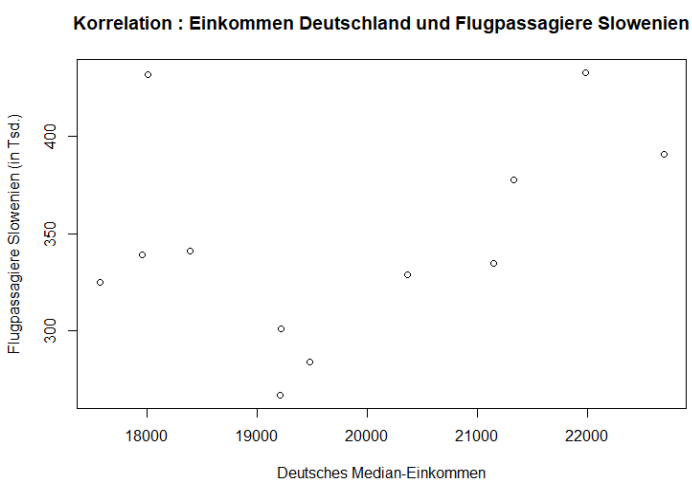
Da meine beiden Datensätze die Daten aller EU-Länder beinhalten, musste ich mich entscheiden, welche Variable ich fest auf Deutschland setzte. Dabei habe ich mich dafür entschieden, die Korrelationen zwischen dem deutschen Einkommen und den Flugpassagieren der EU-Länder zu bestimmen. Dafür erstelle ich einen Vektor namens **correlations**, in dem ich alle Werte der Korrelationen speichern möchte. Ich erstelle eine for-Schleife, welche über die gesamten Länder iteriert, und für jedes Land einen Test für die Korrelation durchführt. Der Wert jedes Tests wird daraufhin in **correlations** gespeichert. Nun erstelle ich ein Histogramm mit der **hist()-Funktion**, um mir anzuschauen, wie die Korrelationskoeffizienten ausgefallen sind. Dabei kommt folgendes Histogramm raus:



Man sieht, dass ein Großteil der EU-Länder (hinsichtlich der Anzahl der Flugpassagiere) eine stark positive Korrelation zwischen 0,9 und 1 mit dem deutschen Median-Einkommen haben. Nur einige wenige Länder weisen eine schwach positive Korrelation (< 0,5) auf. Auffällig ist auch, dass es keine negative Korrelation gibt.

Um herauszufinden, welches Land das mit der größten/kleinsten Korrelation zum Einkommen in Deutschland aufweist, nutze ich die **which.max()-Funktion** bzw. **which.min()-Funktion**. Dabei speichere ich die Werte in zwei index-Variablen, um danach damit auf meinen **flug** Dataframe zuzugreifen. Die index-Variablen geben mir die Spalte an, an der die Korrelation am geringsten/höchsten war. Das Land, bei welchem der Korrelationskoeffizient am geringsten ist, ist Slowenien. Das Land mit dem größten Korrelationskoeffizienten ist Frankreich. Als Nächstes führe ich zwei Korrelations-Tests mit den beiden Ländern durch.

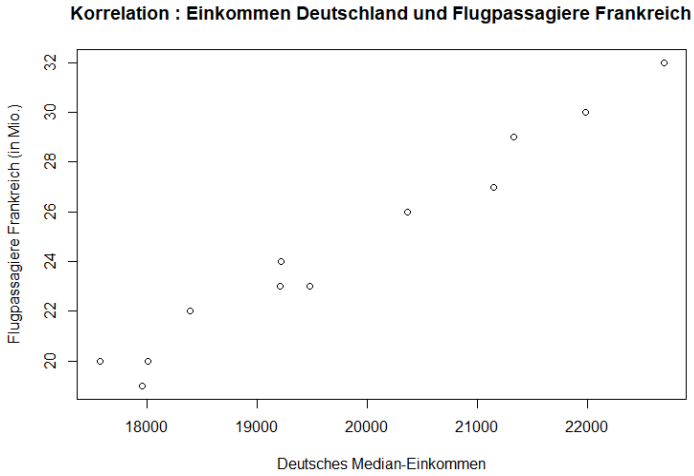
Korrelationskoeffizient mit Slowenien = **0,3535538** , p-Wert = 0,2596 , nicht signifikant



Korrelationskoeffizient mit Frankreich = **0,9873359** , p-Wert = 0,000002511 , hoch signifikant

Folgend die Visualisierungen der Abhängigkeiten:

Bei genauem Hinsehen kann man eine „V-Form“ der Punkte erahnen, was für eine Korrelation sprechen würde, trotz dessen, dass der Korrelationskoeffizient einen geringen Wert anzeigt, ganz einfach weil der Korrelationskoeffizient nur lineare Zusammenhänge untersucht, bzw. anzeigt. Dies ist aber nur eine Vermutung.



Man sieht eine stark positiv lineare Korrelation der Variablen. Dies bestätigt auch der Korrelationskoeffizient.

Tag 3: Multivariate Verfahren

Aufgabe 9: Deskriptives

a - Lesen Sie den SEPCTF Datensatz mit Hilfe des load()-Befehls ein und überprüfen Sie welche Information das RData-File enthält und die Dimension der Daten SPECTF

Den SEPCTF kann man einfach einlesen, indem man der **load()-Funktion** den Speicherpfad der Datei übergibt. Mithilfe der **head()-Funktion** lasse ich mir die ersten paar Zeilen des SPECTF-Datensatzes ausgeben und sehe, dass der Datensatz über 45 Spalten verfügt, welche von **X0** bis **X44** reichen. Über die **nrow()-Funktion** lasse ich mir die Anzahl der Zeilen ausgeben, welche 267 beträgt. Somit scheint alles in Ordnung zu sein.

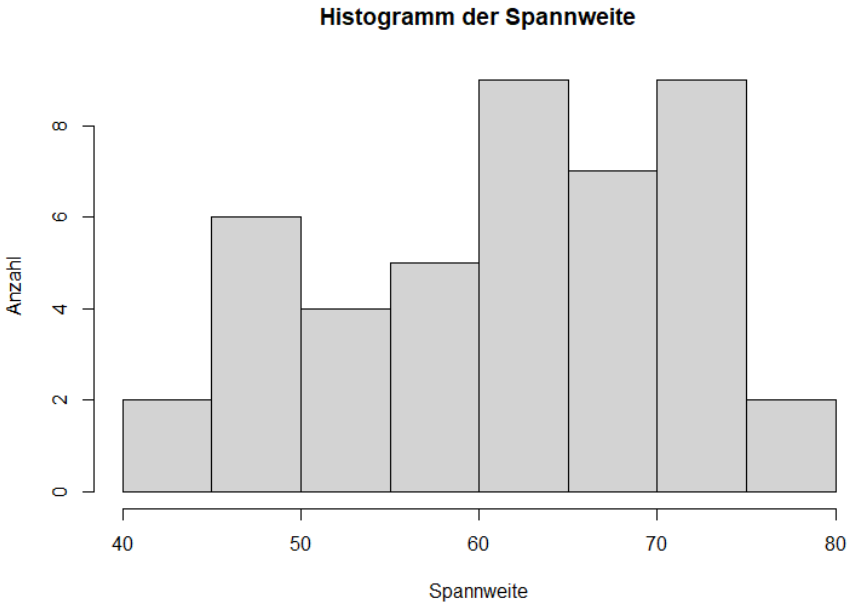
b - Berechnen Sie für jeden Parameter die Spannweite (Maximum-Minimum). Betrachten Sie die Summary und erstellen Sie ein Histogramm der Spannweiten

Um die Spannweite für jeden $\mathbf{X_i}$ Parameter zu berechnen, habe ich mit einer for-Schleife über jeden $\mathbf{X_i}$ Parameter iteriert und die Spannweite wie folgt berechnet: $\mathbf{span = max(X_i) – min\ (X_i)}$

Die einzelnen Ergebnisse speichert man in einem dafür vorher angelegten Vektor. Danach habe ich mir mithilfe der **summary()-Funktion** eine Zusammenfassung von den gespeicherten Spannweiten

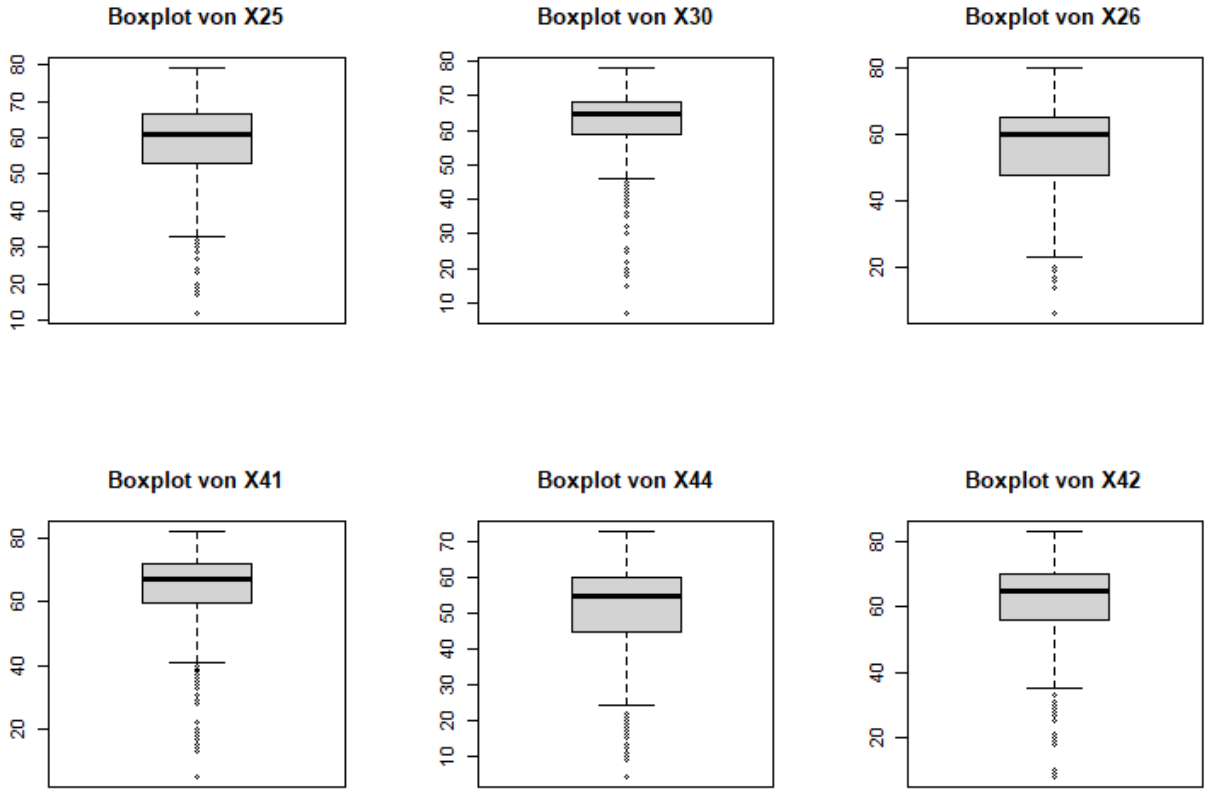
ausgeben lassen. Dabei sieht man, dass die minimale Spannweite 41 beträgt und die maximale Spannweite 77 beträgt. Der Median ist 64 und der Mittelwert ist 62,18. Der **hist()-Funktion** übergibt man den Vektor mit den Spannweiten und erhält:

Anhand des Histogramms kann man erkennen, dass die Spannweite 60<x<65 und 70<x<75 ,die am häufigsten vorkommenden Spannweiten sind. Die Spannweiten sehen nicht normalverteilt aus. An den Rändern der Spannweite ist die Häufigkeit geringer als sonst.



c - Berechnen Sie für jeden Parameter die Varianz. Greifen Sie die 6 Parameter mit der höchsten Varianz heraus, erstellen Sie für diese je einen Boxplot und fassen Sie die Einzelboxplots in einer Graphik zusammen

Um für jeden Parameter die Varianz zu berechnen, nutzt man die dafür vorgesehene **lapply()-Funktion**, in Kombination mit der **var()-Funktion**. Die Rückgabe davon speichere ich in einer Variable **varList**, in welcher nun alle Varianzen vorzufinden sind. Um die 6 größten Varianzen zu kriegen, muss ich die Liste erst einmal sortieren, damit die 6 größten Varianzen an der 6 letzten Stellen stehen. Dafür extrahiert man erst einmal die Namen der jeweiligen Komponenten mithilfe der **names()-Funktion**. Um die Indexe der sortierten Werte zu erhalten nutzt man die **order()-Funktion** und die **unlist()-Funktion**. Da man die **order()-Funktion** nicht auf eine Liste Variable vom Typ „list“ anwenden kann, muss man die Liste erst mal umwandeln mithilfe der **unlist()-Funktion**, daraufhin kann man dann die Funktion anwenden und man erhält die Indexe in sortierter Reihenfolge. Nun kann man die **varList** mithilfe der Variable der Namen und der Variable der Indexe neu anordnen. Man erhält eine Liste, mit aufsteigend sortierten Varianzen. Danach gebe ich mir die letzten 6 Werte, also die 6 größten Werten, aus und erstelle zu jedem der Werte einen Boxplot mithilfe der **boxplot()-Funktion**, wobei ich davor mit der **par()-Funktion** und dem Parameter **mfrow = c(2,3)** festlege, dass ich die nächsten 6 Plots in einer 2 X 3 Ansicht haben möchte. Danach setzte ich den Parameter wieder auf **c(1,1)**, um die zukünftigen Plots wieder einzeln zu plotten. Heraus kommt:



Hierbei hat X25 die kleinste von der 6 größten Varianzen und X42 die größte Varianz. Man erkennt an den Boxplots, dass alle ihren Median *ungefähr* bei 60 haben. Ausreißer scheint es nur unter dem Lower-fence zu geben. Der Upper-fence liegt bei allen Variablen so ca. bei 80, bis auf X44, wo er ungefähr bei 70 liegt. Der Lower-Fence liegt für alle Variablen zwischen 20 bis 45.

Aufgabe 10: Logistische Regression, Kreuzvalidierung und ROC-Kurven

b - Implementieren Sie eine 10-fache Kreuzvalidierung mittels einer for-Schleife für den SPECTF Datensatz. Innerhalb dieser Schleife soll eine logistische Regression anhand des der Trainingsdaten (Zielvariable ~ Parameter), sowie eine Vorhersage anhand der Testdaten gemacht werden

Zuerst lade die drei Bibliotheken „**caret**“, „**pROC**“ und „**ROCR**“, mit der **library()-Funktion**, wobei diese mir helfen werden, bei der Kreuzvalidierung und den nächsten Aufgaben und setzte einen seed mit dem Wert 123, um reproduzierbar zu sein. Um eine logistische Regression durchführen zu können , muss man die binäre Zielvariable in einen Faktor umwandeln und dessen Level angeben. Also wandle ich die Variable **X0** in einen Faktor um mit der **factor()-Funktion**. Da man Trainings und Testdaten haben möchte kann man die **createDataPartition()-Funktion** nutzen,

um die Daten zum Beispiel in ein 80/20 Training/Test Verhältnis zu unterteilen. Dabei gibt das **p** das Verhältnis an, welches ich hier mit 0,8 gewählt habe. Die Funktion gibt eine Liste von Indexen zurück, mit denen man dann den SPECTF Datensatz in 2 zufällig gemischte Teildatensätze unterteilen kann. Somit erstelle ich mithilfe der Indexe zwei Teil-Daten-Mengen welche ich **dataToTrain** und **dataToTestLater** nenne. Da ich eine 10-fache Kreuzvalidierung mittels einer for-Schleife implementieren möchte, nutze ich die **createFolds()-Funktion** als Hilfe, welche mir anhand der Parameter die ich übergebe, die übergebenen Daten (**dataToTrain**) in 10 gleich große Teildatenmengen unterteilt. Vor der for-Schleife erstelle ich noch eine Liste names **results**, in welcher in meine Ergebnisse der Vorhersage speichern möchte. Da eine 10-fache Kreuzvalidierung gefragt ist, iteriert meine Schleife von 1 bis 10.

In jeder Schleife werden nun die Daten mit denen das Modell erstellt wird und die Daten mit denen das Modell trainiert wird neu vergeben. Dafür habe ich zwei Variablen erstellt, namens **train_fold** und **test_fold**, wobei die erste die Daten enthält, womit das Modell trainiert wird und die zweite Variable die Daten enthält, mit denen das Modell danach getestet wird. Die beiden Variablen enthalten sozusagen immer das Gegenteil voneinander, wobei die „Gesamtdaten“ von **dataToTrain** stammen.

Als nächstes wird die logistische Regression durchgeführt, mithilfe der **glm()-Funktion**, anhand der **train_fold** Daten jeder Iterierung. Um danach Vorhersagen für das Modell anhand den Testdaten zu erstellen, nutzt man die **predict()-Funktion**, welcher man das erstellte Modell und die Testdaten bzw. **test_fold** übergibt. Diese Vorhersage speichere ich in einer Variable, welche ich **preds** nenne, und füge die entstandenen Vorhersagen, sowie die wahren Werte, meiner **results** Liste hinzu.

c - Berechnen Sie mit Hilfe der R-package pROC und ROCR die ROC-Kurve, die Konfidnezintervalle und den AUC der Vorhersage aus Aufgabe (b)

Um die ROC-Kurve zu berechnen, nutze ich die **roc()-Funktion**, welche vom Paket pROC stammt. Der Funktion übergibt man die wahren und vorhergesagten Werte, welche beide in **results** gespeichert sind, wobei diese dann die ROC-Kurve berechnet. Die Daten werden mithilfe der **lapply()-Funktion** und der **unlist()-Funktion** zu einem Faktor, bzw. zu einem Vektor umgewandelt, wobei man am Ende wieder alle Daten von allen Iterationen vereint hat. Anhand dieser Daten berechnet die **roc()-Funktion** die ROC-Kurve. Um den AUC (Area under the Curve) zu berechnen, nutze ich die **auc()-Funktion** von dem Paket pROC und übergebe ihr meine **roc** Variable, welche meine ROC-Kurve darstellt. Die Funktion berechnet somit den AUC-Wert, welchen ich in einer Variable namens **auc** speichere. Das 95% Konfidenzintervall berechne ich

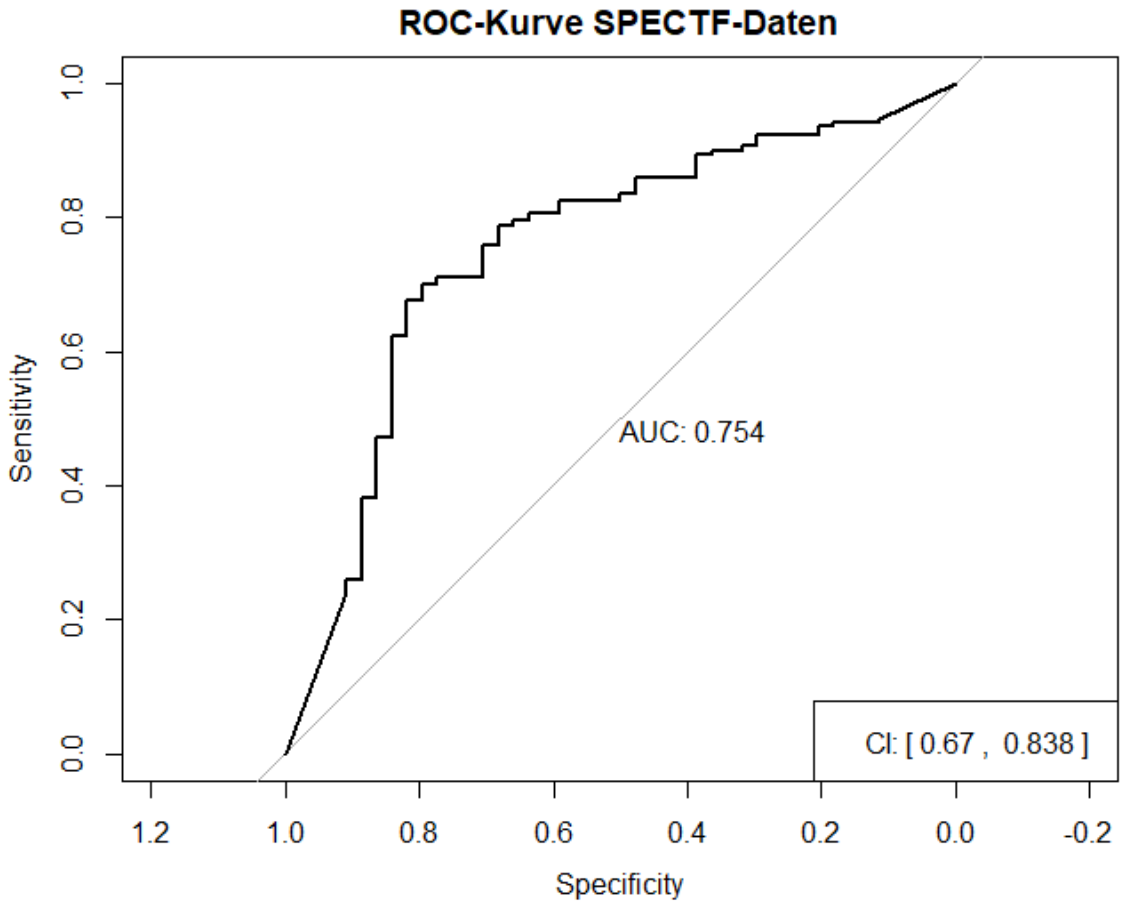
mithilfe der **ci.auc()-Funktion**, welche ebenso vom Paket pROC stammt, indem ich dieser Funktion ebenso meine **roc** Variable übergebe. Meine Ergebnisse lauten: (für seed = 123)

AUC = 0,754

95% Konfidenzintervall = [0,67 , 0,838]

d - Plotten Sie die ROC-Kurve und machen Sie im gleichen Plot den AUC-Wert und das Konfidenzintervall sichtbar. Speichern Sie den Plot in einem PDF-File ROC SPECTF.pdf

Um die ROC-Kurve zu plotten, nutze ich die **plot()-Funktion**, welcher ich meine **roc** Variable übergebe, mit dem weiteren Parameter **print.auc = TRUE**, womit mir die Funktion direkt meinen AUC-Wert in dem Plot sichtbar macht. Um das Konfidenzintervall noch einzufügen, nutze ich die **legend()-Funktion**, womit man eine Legende in einem Plot erstellen kann. Ich möchte das Konfidenzintervall unten rechts sichtbar machen, weshalb ich der Funktion als Argument „bottomright“ übergebe. Um das Konfidenzintervall hinzuzufügen, übergibt man der **paste()-Funktion** , welche mehrere Sub-Strings zu einem String zusammenfügt, die einzelnen Werte des **ci** Objektes. Somit kommt am Ende der Plot, welcher links zu sehen ist, raus. Um diesen Plot zu speichern nutzt man die **pdf()-Funktion**, welcher man einen String übergibt, wie man die Dateien benennen möchte. Dies tut man, bevor man den Plot an sich erstellt. Wenn der Plot fertig gestellt wurde, nutzt man noch die **dev.off()-Funktion**, welche den Plot dann im Working-Directory speichert.



Die ROC-Kurve hat einen AUC-Wert von 0,754, wobei 1 ein perfektes Modell wäre und 0,5 ein Modell was raten würde, somit ist das Modell besser als raten, aber nicht perfekt.

Anhang

Die .R Datei befindet sich mit in der .zip-Datei.

Parviz Moskalenko