Beispiel 11.2. Vergleich verschiedener JS-Modelle zur Schätzung der Überlebensraten von Geckos

Kapitel 11 aus Henle, K., A. Grimm-Seyfarth & B. Gruber: Erfassung und Analyse von Tierpopulationen. Ulmer Verlag

Annegret Grimm-Seyfarth

2025-03-17

In diesem Beispiel werden wir uns mit JS-Methoden beschäftigen. Zunächst werden wir die Schätzungen anhand der im Buch vorgestellten Formeln (Kap. 11.1.2) durchführen. Dann werden wir eine Einführung in die verschiedenen R-Pakete geben und die Schätzungen vergleichen. Dabei wollen wir auch individuelle Parameter mit in die Schätzungen einbeziehen und das am besten passende Modell finden. Dazu nutzen wir AIC-Vergleiche. Zum Abschluss zeigen wir Goodness-of-Fit-Tests.

Wir nutzen dabei ein Beispiel aus einer Langzeiterfassung des australischen, baumbewohnenden Geckos *Gehyra variegata*, der seit 1985 regelmäßig im Kinchega Nationalpark gefangen wird (Henle 1990, Grimm-Seyfarth et al. 2018). Zur besseren Demonstration nutzen wir einen Teildatensatz von 2012 bis 2016, da hier jährlich Ende Februar (gleiche Saison) und mit vergleichbarem Fangaufwand gefangen wurde. Das Fangen folgte einem robusten Versuchsplan. Dabei fungieren die Jahre als Primärperioden (5), innerhalb der es jeweils 6 (2012, 2013) bzw. 7 Sekundärperioden (2014-2016) gab. Im robusten Versuchsplan wurde eine Fanggeschichte (ch) erstellt, bei der, sofern ein Individuum innerhalb des Jahres gefangen wurde, dieses Tier eine "1" zugewiesen bekam - egal wann und wie oft es gefangen wurde (sog. ad hoc Design). Zusätzlich wurden für jedes Tier Eigenschaften aus dem Originaldatensatz (Grimm-Seyfarth et al. 2018) herausgelesen: das Geschlecht (Sex); die Sesshaftigkeit (Residency), bei der Tiere, die immer am selben Baum gefangen wurden, als "Resident" bezeichnet werden und solche, die den Baum wechselten, als "Floater"; die maximale und die mittlere Anzahl Fanggelegenheiten pro Jahr, an denen das Tier beobachtet wurde (Catchability_max und Catchability_mean), die mittlere Masse (average_mass) und die mittlere Kopf-Rumpf-Länge (average_SVL). Der vorliegende Datensatz enthält ausschließlich adulte Tiere, keine juvenilen oder subadulten Tiere. Diese haben eine unterschiedliche Fängigkeit und Überlebenswahrscheinlichkeit und können daher nicht im gleichen JS-Modell berechnet werden. Hierfür wäre ein Altersklassenmodell notwendig, welches in Grimm-Seyfarth et al. (2018) genutzt und erklärt wird.

Die Daten lesen wir zunächst in R ein:

```
UlmerBuch::beispiel.pfad() #Pfad zu den Beispieldaten
```

```
## Der Pfad zu den Beispiel Daten wurde gesetzt auf:  D:/Bernd/R/UlmerBuch/inst/extdata
```

```
gecko <- read.csv2("GV_RI_2012-2016.csv", stringsAsFactors=TRUE)
head(gecko)
```

```
##   Number X2012 X2013 X2014 X2015 X2016    Sex Residency Catchability_max
## 1  R000X     0     1     0     0     0   male  Resident                4
## 2  R004Y     1     0     0     0     0   male  Resident                1
## 3  R00Y0     0     1     0     0     0 female  Resident                5
## 4  R0411     1     1     1     1     1   male   Floater                3
## 5  R063X     1     0     0     0     0 female  Resident                1
## 6  R07Y7     1     1     1     1     1 female   Floater                6
```

```
##   Catchability_mean average_mass average_SVL
## 1               4.0        3.260        4.90
## 2               1.0        2.990        5.00
## 3               5.0        2.360        5.00
## 4               2.2        3.140        5.02
## 5               1.0        3.460        5.00
## 6               4.2        2.914        5.00
```

```
nrow(gecko)
```

```
## [1] 145
```

Im Datensatz sind 145 individuelle Geckos enthalten.

Berechnung anhand der Formeln 11.1-11.10 des Buches

Fangstatistiken:

Bei k = 5 Erfassungsjahren reicht i von 1 bis 5 und j von 1 bis k-1 = 4.

ni: Gesamtzahl aller Tiere, die bei der i-ten Erfassung gefangen/festgestellt wurden (i = 1,. . . ,k) (ni = mi + ui);

```
ch <- gecko[,2:6]
ni <- colSums(ch)
ni
```

```
## X2012 X2013 X2014 X2015 X2016
##    41    59    62    60    55
```

Ri: Anzahl Exemplare aus ni, die nach der i-ten Erfassung (i = 1,. . . ,k-1) wieder freigelassen wurden. Wenn ein Teil der zum Zeitpunkt i gefangenen Exemplare einbehalten wird (Belegentnahme, Verletzung bei Markierung etc.), können zum Zeitpunkt i nur noch Ri < ni Exemplare freigelassen werden; in diesem Datensatz haben wir alle gefangenen Geckos wieder freigelassen, daher ist Ri = ni

```
Ri <- ni
Ri
```

```
## X2012 X2013 X2014 X2015 X2016
##    41    59    62    60    55
```

mi: Anzahl markierter Individuen, die bei der i-ten Erfassung gefangen/festgestellt wurden (i = 1,. . . ,k; m1 = 0);

Hinweis: bei der Auswahl bedeutet “&” das statistische UND und “|” das statistische ODER

```
# Wiederfangvektor
mi <- c(0, sum(ch$X2013[ch$X2012==1]),
        sum(ch$X2014[ch$X2012==1 | ch$X2013==1]),
        sum(ch$X2015[ch$X2012==1 | ch$X2013==1 | ch$X2014==1]),
        sum(ch$X2016[ch$X2012==1 | ch$X2013==1 | ch$X2014==1 | ch$X2015==1]))
mi
```

```
## [1]  0 22 39 32 39
```

```
# Matrix Mij für den Wiederfang
# i sind die Primärparioden 1 bis 5, organisiert in Spalten
# j sind die
mij <- as.data.frame(matrix(nrow=4,ncol=5))
colnames(mij) <- 1:5
```

```
rownames(mij) <- 1:4
mij
```

```
##    1  2  3  4  5
## 1 NA NA NA NA NA
## 2 NA NA NA NA NA
## 3 NA NA NA NA NA
## 4 NA NA NA NA NA
```

```
mij$"1" <- 0
mij[1,2] <- mi[2]
mij[1,3] <- sum(ch$X2014[ch$X2012 == 1 & ch$X2013 == 0])
mij[2,3] <- sum(ch$X2014[ch$X2013 == 1])
mij[1,4] <- sum(ch$X2015[ch$X2012 == 1 & ch$X2013 == 0 & ch$X2014 == 0])
mij[2,4] <- sum(ch$X2015[ch$X2013 == 1 & ch$X2014 == 0])
mij[3,4] <- sum(ch$X2015[ch$X2014 == 1])
mij[1,5] <- sum(ch$X2015[ch$X2012 == 1 & ch$X2013 == 0 & ch$X2014 == 0 & ch$X2015 == 0])
mij[2,5] <- sum(ch$X2016[ch$X2013 == 1 & ch$X2014 == 0 & ch$X2015 == 0])
mij[3,5] <- sum(ch$X2016[ch$X2014 == 1 & ch$X2015 == 0])
mij[4,5] <- sum(ch$X2016[ch$X2015 == 1])

# Überprüfung: Wenn die Matrix richtig erstellt wurde,
# entsprechen die Spaltensummen mi
colSums(mij, na.rm=TRUE) == mi
```

```
##    1    2    3    4    5
## TRUE TRUE TRUE TRUE TRUE
```

Damit haben wir sowohl den Gesamtvektor mi, als auch die Teilwerte für jede Fanggelegenheit j. Es ergibt sich

ri: Anzahl Individuen von Ri, die zum Zeitpunkt i freigelassen und bei einem späteren Fangtermin wiedergefangen wurden (i = 1,. . . ,k-1);

```
ri <- rowSums(mij, na.rm=TRUE)
ri
```

```
##  1  2  3  4
## 26 36 34 36
```

zi: Anzahl Tiere, die vor der Fanggelegenheit i markiert und bei Fanggelegenheit i nicht, jedoch bei einer späteren Fanggelegenheit wiedergefangen wurden (i = 2,. . . ,k-1).

```
zi <- c(NA,
        nrow(ch[ch$X2012 == 1 & ch$X2013 == 0 &
                  (ch$X2014 == 1 | ch$X2015 == 1 | ch$X2016 == 1),]),
        nrow(ch[(ch$X2012 == 1 | ch$X2013 == 1) & ch$X2014 == 0 &
                  (ch$X2015 == 1 | ch$X2016 == 1),]),
        nrow(ch[(ch$X2012 == 1 | ch$X2013 == 1 | ch$X2014 == 1) &
                  ch$X2015 == 0 & ch$X2016 == 1,]),
        NA)
zi
```

```
## [1] NA  4  1  3 NA
```

ui: Anzahl unmarkierter Exemplare, die bei der i-ten Erfassung gefangen wurden (i = 1,. . . ,k);

```r
ui <- ni-mi
ui
```

```
## X2012 X2013 X2014 X2015 X2016
##    41    37    23    28    16
```

Berechnung der Parameter

Mi: Anzahl markierter Tiere, die sich zum Zeitpunkt der i-ten Erfassung in der Population befinden (i = 1,. . . ,k; M1 = 0);

```r
Mi <- mi + (Ri*zi)/ri
Mi
```

```
##    X2012    X2013    X2014    X2015    X2016
##       NA 28.55556 40.82353 37.00000       NA
```

```r
# wir setzten Wert M1 auf 0
Mi[1] <- 0
```

Ni: Gesamtzahl Individuen, die sich zum Zeitpunkt der i-ten Erfassung in der Population befinden (i = 1,. . . ,k);

```r
Ni <- (ni*Mi)/mi
Ni
```

```
##    X2012    X2013    X2014    X2015    X2016
##      NaN 76.58081 64.89894 69.37500       NA
```

```r
VarNi <- Ni*(Ni-ni)*((Mi-mi+Ri)/Mi * (1/ri)-(1/Ri) + (Ni-Mi)/(Ni*mi))
VarNi
```

```
##      X2012      X2013      X2014      X2015      X2016
##        NaN 101.415665   7.406131  30.383301         NA
```

```r
seNi <- sqrt(VarNi)
seNi
```

```
##     X2012     X2013     X2014     X2015     X2016
##       NaN 10.070535  2.721421  5.512105        NA
```

Die Populationsgrößen für 2013 bis 2015 betragen (gerundet) 77, 65 und 69 Individuen mit einem entsprechenden Standardfehler von (gerundet) 10, 3 und 6 Individuen.

phii: Überlebenswahrscheinlichkeit aller Tiere zwischen der i-ten und (i+1)-ten Erfassung (i = 1,. . . ,k-1);

```r
j <- 5-1
phii <- rep(NA,j)
phii
```

```
## [1] NA NA NA NA
```

```r
VarPhii <- rep(NA,j)

for (i in 1:j) {
  phi <- Mi[i+1]/(Mi[i]-mi[i]+Ri[i])
  phii[i] <- phi
  VarPhii[i] <- phi^2 * (((Mi[i+1]-mi[i+1])*(Mi[i+1]-mi[i+1]+Ri[i+1]))/Mi[i+1]^2 *
                           (1/ri[i+1]-1/Ri[i+1])+((Mi[i]-mi[i])/(Mi[i]-mi[i]+Ri[i]))*
                           (1/ri[i]-1/Ri[i]))
```

```
}
phii
```

```
## [1] 0.6964770 0.6227318 0.5797235        NA
```

```
VarPhii
```

```
## [1] 0.0027683731 0.0007796464 0.0010140453           NA
```

```
sePhii <- sqrt(VarPhii)
sePhii
```

```
## [1] 0.05261533 0.02792215 0.03184408         NA
```

Die Überlebensraten betragen von 2012 nach 2013 69,6%, von 2013 nach 2014 62,2% und von 2014 nach 2015 58%; die Standardfehler sind sehr gering.

```
# Gesamtvarianz der Überlebensrate
GesVarPhi <- VarPhii + (phii*(1-phii))/(Mi-mi+Ri)
GesVarPhi
```

```
##       X2012       X2013       X2014       X2015       X2016
## 0.007924393 0.004363430 0.004831511          NA          NA
```

Bi: Gesamtzahl der Exemplare, die zwischen der i-ten und (i+1)-ten Erfassung zur Population neu hinzugekommen sind und zum Zeitpunkt der (i+1)-ten Erfassung sich noch in der Population befinden (i = 1,. . . ,k-1);

```
Bi <- rep(NA,4)

for (i in 1:4) {
  Bi[i] <- Ni[i+1]-phii[i]*(Ni[i]-ni[i]+Ri)
}

Bi
```

```
## [1]       NaN 28.41881 43.92575       NA
```

Von 2013 nach 2014 kommen 28 Individuen hinzu, von 2014 nach 2015 sind es 44 Individuen. Auf die Berechnung der Varianz verzichten wir hier an dieser Stelle.

qi: Sterberate zwischen der i-ten und (i+1)-ten Erfassung (i = 1,. . . ,k-1) (qi = 1 - phii);

```
qi <- 1-phii
qi
```

```
## [1] 0.3035230 0.3772682 0.4202765        NA
```

Die Sterberate beträgt von 2012 nach 2013 30,4%, von 2013 nach 2014 37,7% und von 2014 nach 2015 42,0%.

pi: Fangwahrscheinlichkeit für alle Tiere, die sich bei der i-ten Erfassung in der Population befinden (i = 1,. . . ,k).

```
pi = mi/Mi
pi
```

```
##     X2012     X2013     X2014     X2015     X2016
##       NaN 0.7704280 0.9553314 0.8648649        NA
```

Berechnung mittels verschiedener R-Pakete

Die folgende Übersicht wurde von Laake et al. (2023) zusammengestellt und von uns ergänzt. Die derzeit umfassendste Software für die Analyse von Fang-Wiederfang-Daten ist MARK (White und Burnham 1999). MARK ist ein FORTRAN-Programm zur Anpassung von Fang-Wiederfang-Modellen, die manuell über eine grafische Benutzeroberfläche erstellt werden. RMark (Laake und Rexstad 2008) ist ein R-Paket, das Modelle für MARK mit benutzerdefinierten Formeln konstruiert, um die manuelle Modellerstellung zu ersetzen. Mit RMark und MARK können die meisten derzeit verfügbaren Fang-Wiederfang-Modelle in R angepasst und bearbeitet werden.

Weitere R-Pakete für die Analyse von Fang-Wiederfang-Daten sind FSA (Ogle et al. 2023), Rcapture (Baillargeon und Rivest 2007), mra (McDonald et al. 2005), secr (Borchers und Efford 2008), BTSPAS (Schwarz et al. 2009), SPACECAP (Royle et al. 2009), BaSTA (Colchero, Jones und Rebke 2012) und marked (Laake et al. 2013). FSA beinhaltet das JS-Grundmodell analog zum Programm JOLLY. Rcapture passt geschlossene und offene Modelle in einem log-linearen Rahmen an. Das mra-Paket passt Cormack-Jolly-Seber (CJS) und das geschlossene Huggins-Modell mit einem Regressionsansatz zur Modellspezifikation an. Die Pakete secr und SPACECAP ermöglichen eine räumlich explizite Modellierung geschlossener Fang-Wiederfang-Daten und BTSPAS passt zeitlich geschichtete Petersen-Modelle in einem Bayes'schen Rahmen an. BaSTA schätzt das Überleben mit Kovariaten aus Wiederfang-/Wiederfindungsdaten in einem Bayes'schen Rahmen, wenn viele Individuen ein unbekanntes Alter haben. Spezifikationen für das marked finden sich unten im entsprechenden Kapitel.

In diesem Beispiel konzentrieren wir uns auf die Pakete FSA, marked und RMark. Diese laden wir hier zunächst ein. Außerdem laden wir ein Paket für Goodness-of-fit-Tests ein: R2ucare. Und schließlich laden wir ggplot2 (Wickham 2016) zur graphischen Darstellung der Ergebnisse.

```
# check.packages function: install and load multiple R packages.
# Function from: https://gist.github.com/smithdanielle/9913897
check.packages <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE, type = "source")
  sapply(pkg, require, character.only = TRUE)
}

# benoetigte R pakete
pakete <- c("FSA", "marked", "RMark", "R2ucare", "ggplot2")

# Pruefe und installiere
check.packages(pakete)
```

```
##     FSA  marked   RMark R2ucare ggplot2
##    TRUE    TRUE    TRUE    TRUE    TRUE
```

Weitere Informationen zu den verschiedenen Paketen sowie zur Nutzung des Paketes finden sich hier:

https://cran.r-project.org/web/packages/RMark/RMark.pdf

https://search.r-project.org/CRAN/refmans/FSA/html/mrOpen.html

https://cran.r-project.org/web/packages/FSA/FSA.pdf

https://cran.r-project.org/web/packages/marked/vignettes/markedVignette.html

https://cran.r-project.org/web/packages/R2ucare/vignettes/vignette_R2ucare.html

JS-Model mit der Funktion jolly aus dem R-Paket FSA

Das R-Paket FSA (Ogle et al. 2023) kennen wir bereits aus dem Beispiel 6.7 (Kap. 6.4.3 des Buches), bei dem die Removal Methode vorgestellt wurde. Dieses Paket ermöglicht auch, ein offenes Populationsmodell nach Jolly-Seber zu berechnen.

Es gibt hierfür zwei Varianten: Basierend auf der Fanggeschichte oder basierend auf den Fangstatistiken.

Mittels Fanggeschichte

Dazu, sowie auch für alle folgenden R-Pakete, ist es nötig, die Fanggeschichte als character in eine Spalte zu bringen. Im FSA Packet gibt es dafür die hilfreiche Funktion "capHistSum", die wir in Verbindung mit unserer Fanggeschichte ch anwenden. Anmerkung: Sollte sich in der ersten Spalte noch die ID befinden, gibt es die hilfreiche Ergänzung "cols2use=-1", womit diese erste Spalte aus der Fanggeschichte ausgeschlossen wird.

```
ch1 <- capHistSum(ch)
head(ch1)
```

```
## $caphist
##
## 00001 00010 00011 00100 00101 00110 00111 01000 01010 01100 01110 01111 10000 10100 10101
##    16    10    18    14     1     6     2    13     1    10     6     7    15     2     1
## 10111 11000 11100 11101 11110 11111
##     1    10     2     1     1     8
##
## $sum
##    n  m  R   M  u  v  f
## 1 41  0 41   0 41 15 68
## 2 59 22 59  41 59 23 48
## 3 62 39 62  78 23 28 11
## 4 60 32 60 101 28 60 10
## 5 55 39  0 129 16 55  8
##
## $methodB.top
##     i=1 i=2 i=3 i=4 i=5
## j=1  NA  22   4   0   0
## j=2  NA  NA  35   1   0
## j=3  NA  NA  NA  31   3
## j=4  NA  NA  NA  NA  36
## j=5  NA  NA  NA  NA  NA
##
## $methodB.bot
##   i=1 i=2 i=3 i=4 i=5
## m   0  22  39  32  39
## u  41  37  23  28  16
## n  41  59  62  60  55
## R  41  59  62  60   0
##
## $m.array
##     ni c2 c3 c4 c5 not recapt
## i=1 41 22  4  0  0         15
## i=2 59 NA 35  1  0         23
## i=3 62 NA NA 31  3         28
## i=4 60 NA NA NA 36         24
## i=5 55 NA NA NA NA         55
```

Hier bekommen wir eine Zusammenfassung der Fanggeschichte. Wir sehen nacheinander die wichtigsten Fangstatistiken:

“caphist” - die zusammengefasste Fanggeschichte

“methodB.top” - die mij-Matrix

“methodB.bot” - die Fangstatistiken mi, ui, ni und Ri

“m.array” - die erste Spalte enthält ni, die folgenden (ci) die Wiederfänge bei i = 2, 3. . . , sowie 1-ri. Die Spaltensummen ergeben mi.

Hier können wir auch die von uns erstellen Fangstatistiken einmal vergleichen:

```
mij
```

```
##   1  2  3  4  5
## 1 0 22  4  0  0
## 2 0 NA 35  1  0
## 3 0 NA NA 31  3
## 4 0 NA NA NA 36
```

```
ch1$methodB.top
```

```
##     i=1 i=2 i=3 i=4 i=5
## j=1  NA  22   4   0   0
## j=2  NA  NA  35   1   0
## j=3  NA  NA  NA  31   3
## j=4  NA  NA  NA  NA  36
## j=5  NA  NA  NA  NA  NA
```

```
# stimmen überein
mi == ch1$methodB.bot[1,]
```

```
##  i=1  i=2  i=3  i=4  i=5
## TRUE TRUE TRUE TRUE TRUE
```

```
ui == ch1$methodB.bot[2,]
```

```
## X2012 X2013 X2014 X2015 X2016
##  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
ni == ch1$methodB.bot[3,]
```

```
## X2012 X2013 X2014 X2015 X2016
##  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
Ri == ch1$methodB.bot[4,]
```

```
## X2012 X2013 X2014 X2015 X2016
##  TRUE  TRUE  TRUE  TRUE FALSE
```

Bis hierhin stimmen unsere anhand selbst eingegebener Formeln gerechneten Fangstatistiken mit den hier berechneten überein. Berechnen wir nun die Parameter.

```
ex1 <- mrOpen(ch1, type = c("Jolly"), phi.full = TRUE)
# der komplette Ergebnis-Array:
ex1
```

```
## $df
##      m  n  R  r  z    M M.se    N N.se N.lci N.uci   phi phi.se phi.lci phi.uci    B B.se
## i=1  0 41 41 26 NA   NA   NA   NA   NA    NA    NA 0.695  0.089   0.520   0.869   NA   NA
## i=2 22 59 59 36  4 28.5  2.1 74.3  7.8  59.1  89.5 0.623  0.066   0.494   0.752 18.0  4.5
```

```
## i=3 39 62 62 34  1 40.8  1.2 64.3  2.1  60.2  68.4 0.579  0.069   0.443   0.715 31.1  3.4
## i=4 32 60 60 36  3 36.9  1.9 68.3  4.4  59.7  76.9    NA     NA      NA      NA   NA   NA
## i=5 39 55  0 NA NA   NA   NA   NA   NA    NA    NA    NA     NA      NA      NA   NA   NA
##     B.lci B.uci
## i=1    NA    NA
## i=2   9.1  26.8
## i=3  24.4  37.7
## i=4    NA    NA
## i=5    NA    NA
##
## $type
## [1] "Jolly"
##
## $phi.full
## [1] TRUE
##
## $conf.level
## [1] 0.95
##
## attr(,"class")
## [1] "mrOpen"
```

```
# eine Zusammenfassung der wichtigsten Parameter
summary(ex1,verbose=TRUE)
```

```
## Observables:
##      m  n  R  r  z
## i=1  0 41 41 26 NA
## i=2 22 59 59 36  4
## i=3 39 62 62 34  1
## i=4 32 60 60 36  3
## i=5 39 55  0 NA NA
##
## Estimates (phi.se includes sampling and individual variability):
##        M M.se    N N.se   phi phi.se    B B.se
## i=1   NA   NA   NA   NA 0.695  0.089   NA   NA
## i=2 28.5  2.1 74.3  7.8 0.623  0.066 18.0  4.5
## i=3 40.8  1.2 64.3  2.1 0.579  0.069 31.1  3.4
## i=4 36.9  1.9 68.3  4.4    NA     NA   NA   NA
## i=5   NA   NA   NA   NA    NA     NA   NA   NA
```

```
# Beispiele Parameter einzeln aufrufen
summary(ex1,parm="N")
```

```
##        N N.se
## i=1   NA   NA
## i=2 74.3  7.8
## i=3 64.3  2.1
## i=4 68.3  4.4
## i=5   NA   NA
```

```
summary(ex1,parm=c("N","phi"))
```

```
##        N N.se   phi phi.se
## i=1   NA   NA 0.695  0.089
## i=2 74.3  7.8 0.623  0.066
```

```
## i=3 64.3  2.1 0.579  0.069
## i=4 68.3  4.4    NA     NA
## i=5   NA   NA    NA     NA
```

```
# Berechnung der Konfidenzintervalle
confint(ex1,verbose=TRUE)
```

```
##     N.lci N.uci phi.lci phi.uci B.lci B.uci
## i=1    NA    NA   0.520   0.869    NA    NA
## i=2  59.1  89.5   0.494   0.752   9.1  26.8
## i=3  60.2  68.4   0.443   0.715  24.4  37.7
## i=4  59.7  76.9      NA      NA    NA    NA
## i=5    NA    NA      NA      NA    NA    NA
```

```
# Beispiele für einzelne Parameter
confint(ex1,parm="N")
```

```
##     N.lci N.uci
## i=1    NA    NA
## i=2  59.1  89.5
## i=3  60.2  68.4
## i=4  59.7  76.9
## i=5    NA    NA
```

```
confint(ex1,parm=c("N","phi"))
```

```
##     N.lci N.uci phi.lci phi.uci
## i=1    NA    NA   0.520   0.869
## i=2  59.1  89.5   0.494   0.752
## i=3  60.2  68.4   0.443   0.715
## i=4  59.7  76.9      NA      NA
## i=5    NA    NA      NA      NA
```

Auch hier können wir wieder mit unseren anhand selbst eingegebener Formeln gerechneten Daten vergleichen:

```
# die verbleibenden Fangstatistik-Parameter r und z
ri == ex1$df$r
```

```
## [1] TRUE TRUE TRUE TRUE   NA
```

```
zi == ex1$df$z
```

```
## [1]   NA TRUE TRUE TRUE   NA
```

```
# stimmen alle überein

# weitere Parameter können wir vergleichen, sie werden aber minimal voneinander abweichen:
Mi
```

```
##    X2012    X2013    X2014    X2015    X2016
##  0.00000 28.55556 40.82353 37.00000       NA
```

```
ex1$df$M
```

```
## [1]   NA 28.5 40.8 36.9   NA
```

```
# nahezu identische Werte

Ni
```

```
##    X2012    X2013    X2014    X2015    X2016
```

```
##      NaN 76.58081 64.89894 69.37500       NA
```

```
ex1$df$N
```

```
## [1]   NA 74.3 64.3 68.3   NA
```

```
# geringe Abweichungen

phii
```

```
## [1] 0.6964770 0.6227318 0.5797235        NA
```

```
ex1$df$phi
```

```
## [1] 0.695 0.623 0.579    NA    NA
```

```
# sehr geringe Abweichung
```

In der Anleitung zum FSA Paket findet man folgende Erklärung: Alle Parameterschätzungen erfolgen anhand der Gleichungen 4.6-4.9 aus Pollock et al. (1990) und aus Seite 204 in Seber 2002. Wenn type=„Jolly“, dann werden alle Standardfehler (Quadratwurzel der Varianzen) aus den Gleichungen 4.11, 4.12 und 4.14 in Pollock et al. (1990) berechnet (diese sind anders als die in Seber (2002) - siehe die Anmerkung von Pollock et al. auf Seite 21). Wenn type=„Jolly“ und phi.full=TRUE, dann wird die volle Varianz für den phi-Parameter wie in Gleichung 4.18 in Pollock et al. (1990) angegeben, andernfalls wie in Gleichung 4.13 aus Pollock et al. (1990) verwendet. Bei type=„Jolly“ werden die Konfidenzintervalle erzeugt unter Verwendung der Normaltheorie (d. h. Schätzung +/- z*SE). Unsere vorgestellten Formeln stimmen zumeist mit denen von Pollock et al. (1990) überein, lediglich für phi nutzen wir die Formel nach Seber (2002) (vgl. Formel 11.7 des Buches).

Mittels Fangstatistiken

Liegen bereits die Fangstatistiken vor, können diese auch direkt zur Berechnung genutzt werden. Dazu verbindet man diese zunächst als Matrix und berechnet dann analog wie oben.

```
jolly.bot <- rbind(mi,ui,ni,Ri)
colnames(jolly.bot) <- NULL
rownames(jolly.bot) <- c("m","u","n","R")
jolly.top <- rbind(mij,NA)
jolly.top$"1" <- NA
jolly.top <- as.matrix(jolly.top)
colnames(jolly.top) <- c("s1","s2","s3","s4","s5")
ex2 <- mrOpen(jolly.top,jolly.bot)
summary(ex2,verbose=TRUE)
```

```
## Observables:
##    m  n  R  r  z
## 1  0 41 41 26 NA
## 2 22 59 59 36  4
## 3 39 62 62 34  1
## 4 32 60 60 36  3
## 5 39 55 55 NA NA
##
## Estimates (phi.se includes sampling and individual variability):
```

```
##      M M.se    N N.se   phi phi.se    B B.se
## 1   NA   NA   NA   NA 0.695  0.089   NA   NA
## 2 28.5  2.1 74.3  7.8 0.623  0.066 18.0  4.5
## 3 40.8  1.2 64.3  2.1 0.579  0.069 31.1  3.4
## 4 36.9  1.9 68.3  4.4    NA     NA   NA   NA
```

```
## 5   NA   NA   NA   NA    NA     NA   NA   NA
```

```
confint(ex2,verbose=TRUE)
```

```
##   N.lci N.uci phi.lci phi.uci B.lci B.uci
## 1    NA    NA   0.520   0.869    NA    NA
## 2  59.1  89.5   0.494   0.752   9.1  26.8
## 3  60.2  68.4   0.443   0.715  24.4  37.7
## 4  59.7  76.9      NA      NA    NA    NA
## 5    NA    NA      NA      NA    NA    NA
```

Beide Ansätze produzieren das gleiche Ergebnis. Eine alternative Schreibweise statt mrOpen(jolly.top,jolly.bot) wäre jolly(jolly.top,jolly.bot). Beides kann gleich verwendet werden.

CJS-Modelle mit dem R-Paket marked

Das R-Paket marked (Laake et al. 2013) wurde für Analysen mit markierten Tieren entwickelt, als Gegensatz zum R-Paket unmarked (Fiske und Chandler 2011), welches wir für Occupancy-Modelle vorstellen (Beispiele 4.1 und 4.2). Die Entwickler des Paketes marked schreiben: Der ursprüngliche Anstoß für das Paket war die Implementierung des CJS-Modells unter Verwendung der von Pledger, Pollock und Norris (2003) beschriebenen hierarchischen Likelihood-Konstruktion und die Verbesserung der Ausführungszeiten mit RMark/MARK (White und Burnham 1999; Laake und Rexstad 2008) für die Analyse von großen Datensätze mit vielen zeitlich variierenden individuellen (tierbezogenen) Kovariablen. Anschließend wurde das Jolly-Seber-Modell mit der POPAN-Struktur von Schwarz und Arnason (1996) implementiert, wobei die Idee der hierarchischen Wahrscheinlichkeitskonstruktion auf den Eintritt von Tieren in die Population (Immigration) ausgedehnt wurde, was dem Konzept der Superpopulation entspricht. Zudem wurde eine Bayes'sche Markov-Chain-Monte-Carlo-Implementierung (MCMC) des CJS-Modells hinzugefügt, die auf dem von Albert und Chib (1993) verwendeten Ansatz zur Analyse binärer Daten mit einem Probit-Regressionsmodell basiert.

Wir erstellen als erstes die benötigte Fanggeschichte im Importformat für marked:

```
ch2 <- apply(ch[ ,1:5] , 1 , paste , collapse = "" )
ch2
```

```
##   [1] "01000" "10000" "01000" "11111" "10000" "11111" "01100" "00001" "00100" "00100"
##  [11] "10000" "00001" "00010" "00010" "00001" "00010" "00011" "11110" "11000" "11000"
##  [21] "11111" "10000" "10000" "10000" "10111" "00100" "10000" "11000" "11111" "01100"
##  [31] "10101" "10000" "11100" "10000" "00100" "11000" "00100" "01100" "00011" "01111"
##  [41] "01111" "00100" "00110" "00100" "01000" "01100" "00110" "00011" "00101" "11111"
##  [51] "00011" "11111" "10000" "11100" "00011" "00011" "11000" "00011" "01100" "01010"
##  [61] "11111" "01110" "01000" "10000" "01000" "00100" "00001" "11000" "00011" "00100"
##  [71] "00110" "00011" "00010" "00001" "00011" "00001" "00010" "00010" "00011" "00010"
##  [81] "00001" "00001" "00011" "00011" "00001" "00010" "00011" "00001" "00001" "00001"
##  [91] "00001" "00010" "01111" "01110" "00110" "00110" "11000" "10000" "10000" "01000"
## [101] "01110" "00001" "11000" "00100" "00001" "00100" "01000" "10000" "01110" "00110"
## [111] "00111" "01000" "00100" "00111" "01111" "00011" "01100" "01110" "01100" "01110"
## [121] "00010" "01111" "11111" "11000" "10100" "11101" "01000" "01111" "00100" "01000"
## [131] "10000" "01000" "00011" "10100" "00011" "00001" "01000" "11000" "00011" "01100"
## [141] "01100" "01111" "01100" "01000" "00100"
```

```
gecko2 <- data.frame(ch = ch2,
                     sex = gecko$Sex)
head(gecko2)
```

```
##      ch    sex
## 1 01000   male
## 2 10000   male
```

```
## 3 01000 female
## 4 11111   male
## 5 10000 female
## 6 11111 female
```

Die folgenden Schritte sind zur Vorbereitung der Berechnungen nötig:

```
# zunächst muss RMark entfernt werden, da sich die Befehle ähneln
# (R "weiß" dann nicht, welche Befehle ausgeführt werden sollen)
detach("package:RMark", unload = TRUE)

# Datenprozessierung
# wir nutzen nur das Geschlecht in dieser Variante
gecko.js.proc <- process.data(gecko2, model = "JS",
                              groups = "sex",
                             begin.time=1)
# Designdaten erstellen
gecko.js.ddl <- make.design.data(gecko.js.proc)
```

Folgende Parameter werden geschätzt:

phi(t) - Überlebensrate (apparent survival)

p(t) - Fängigkeit zum Zeitpunkt t

N(super) - Superpopulationsgröße, Gesamtzahl der Individuen, die an der Studie teilnehmen können, d. h., alle Individuen, die sich während der Erfassungsdauer entweder durchgehend oder zeitweise (durch Einwanderung, Geburt bzw. Wechsel in das relevante Altersstadium - in unserem Fall Adult - in das Untersuchungsgebiet kommen oder durch Emigration oder Tod es verlassen haben) im Untersuchungsgebiet aufhalten

pent(t) - Wahrscheinlichkeit, dass zum Zeitpunkt t neue Individuen aus der Superpopulation hinzukommen

JS mit einem Parameter: Geschlecht

Im Folgenden können alle möglichen Formeln für alle Parameter zusammengestellt werden. Dazu schreibt man eine sog. “wrapper”-Funktion.

```
fit.js.gecko.models <- function(){
  # Phi Formeln
  Phi.dot <- list(formula=~1)
  Phi.time <- list(formula=~time)
  # p Formeln
  p.dot <- list(formula=~1)
  p.sex <- list(formula=~sex)
  # pent Formeln (pent Schätzungen summieren immer auf 1 (für jede Gruppe))
  pent.time <- list(formula=~time)
  pent.sex <- list(formula=~sex)
  pent.dot <- list(formula=~1)
  # Nsuper Formeln (NICHT N!)
  N.sex <- list(formula=~sex)
  N.dot <- list(formula=~1)
  cml <- create.model.list(c("Phi","p", "pent", "N"))
  results <- crm.wrapper(cml, data = gecko.js.proc, ddl = gecko.js.ddl,
                         external = FALSE, accumulate = FALSE, hessian = TRUE)

  return(results)
}
```

```
# Alle Modellkombinationen rechnen lassen
gecko.js.models <- fit.js.gecko.models()
```

```
gecko.js.models
```

```
##                                    model npar      AIC  DeltaAIC       weight  neg2lnl
## 5          Phi(~1)p(~1)pent(~time)N(~1)    7 411.2927  0.000000 0.3097923078 397.2927
## 11       Phi(~1)p(~sex)pent(~time)N(~1)    8 413.0074  1.714675 0.1314418323 397.0074
## 6        Phi(~1)p(~1)pent(~time)N(~sex)    8 413.0761  1.783394 0.1270022827 397.0761
## 1             Phi(~1)p(~1)pent(~1)N(~1)    4 413.6786  2.385897 0.0939679210 405.6786
## 12     Phi(~1)p(~sex)pent(~time)N(~sex)    9 414.9921  3.699403 0.0487254079 396.9921
## 3           Phi(~1)p(~1)pent(~sex)N(~1)    5 415.2887  3.995987 0.0420100402 405.2887
## 7           Phi(~1)p(~sex)pent(~1)N(~1)    5 415.3199  4.027183 0.0413598585 405.3199
## 2           Phi(~1)p(~1)pent(~1)N(~sex)    5 415.4284  4.135685 0.0391758181 405.4284
## 17     Phi(~time)p(~1)pent(~time)N(~1)   10 415.5036  4.210905 0.0377297726 395.5036
## 9         Phi(~1)p(~sex)pent(~sex)N(~1)    6 417.0488  5.756092 0.0174241303 405.0488
## 4         Phi(~1)p(~1)pent(~sex)N(~sex)    6 417.0651  5.772403 0.0172826102 405.0651
## 23   Phi(~time)p(~sex)pent(~time)N(~1)   11 417.1363  5.843575 0.0166783992 395.1363
## 18   Phi(~time)p(~1)pent(~time)N(~sex)   11 417.2796  5.986866 0.0155252681 395.2796
## 8         Phi(~1)p(~sex)pent(~1)N(~sex)    6 417.3107  6.018014 0.0152853546 405.3107
## 13         Phi(~time)p(~1)pent(~1)N(~1)    7 417.8711  6.578454 0.0115498817 403.8711
## 10     Phi(~1)p(~sex)pent(~sex)N(~sex)    7 419.0204  7.727674 0.0065017083 405.0204
## 24 Phi(~time)p(~sex)pent(~time)N(~sex)   12 419.1321  7.839448 0.0061483151 395.1321
## 19      Phi(~time)p(~sex)pent(~1)N(~1)    8 419.4166  8.123964 0.0053330324 403.4166
## 15      Phi(~time)p(~1)pent(~sex)N(~1)    8 419.4942  8.201486 0.0051302731 403.4942
## 14      Phi(~time)p(~1)pent(~1)N(~sex)    8 419.6113  8.318617 0.0048384450 403.6113
## 21    Phi(~time)p(~sex)pent(~sex)N(~1)    9 421.1818  9.889161 0.0022063102 403.1818
## 16    Phi(~time)p(~1)pent(~sex)N(~sex)    9 421.2688  9.976115 0.0021124424 403.2688
## 20    Phi(~time)p(~sex)pent(~1)N(~sex)    9 421.4156 10.122918 0.0019629391 403.4156
## 22  Phi(~time)p(~sex)pent(~sex)N(~sex)   10 423.1720 11.879346 0.0008156493 403.1720
##    convergence
## 5            0
## 11           0
## 6            0
## 1            0
## 12           0
## 3            0
## 7            0
## 2            0
## 17           0
## 9            0
## 4            0
## 23           0
## 18           0
## 8            0
## 13           0
## 10           0
## 24           0
## 19           0
## 15           0
## 14           0
## 21           0
## 16           0
## 20           0
```

```
## 22           0
```

Das beste Modell mit dem kleinsten AIC hat konstante Überlebens- und Fangwahrscheinlichkeit, zeitabhängigen Zuwachs (pent) und eine konstante Superpopulationsgröße. Das nächstfolgende Modell mit einem deltaAIC von 1,7 (also im Bereich von möglichen Alternativen, da deltaAIC < 2) hätte eine geschlechtsabhängige Fangwahrscheinlichkeit (p(sex)).

Schauen wir uns die Schätzwerte des zweitbesten Modells (Modell 11) an:

```
gecko.js.models[[11]]
```

```
##
## crm Model Summary
##
## Npar :  8
## -2lnL:  397.0074
## AIC  :  413.0074
##
## Beta
##                     Estimate        se         lcl         ucl
## Phi.(Intercept)   0.5613277 0.1475026  0.27222272  0.85043278
## p.(Intercept)     2.2458061 0.4778278  1.30926358  3.18234869
## p.sexmale        -0.2806844 0.5242580 -1.30823009  0.74686126
## pent.(Intercept) -0.1854738 0.2478647 -0.67128857  0.30034090
## pent.time3       -0.5102670 0.3072576 -1.11249189  0.09195786
## pent.time4       -0.2538672 0.2686945 -0.78050837  0.27277396
## pent.time5       -0.8962219 0.3377424 -1.55819698 -0.23424676
## N.(Intercept)     1.1801902 0.6195904 -0.03420688  2.39458736
```

Diese Schätzwerte liegen auf der Wahrscheinlichkeitsskala (Phi, p auf der logit und pent auf der mlogit Skala) und können daher nicht direkt genutzt werden. Für die tatsächlichen Werte benötigen wir die Funktion "predict".

```
gecko.js.predicted <- predict(gecko.js.models[[11]])
gecko.js.predicted
```

```
## $Phi
##   occ  estimate         se       lcl       ucl
## 1   1 0.6367597 0.03411687 0.5676385 0.7006579
##
## $p
##      sex occ  estimate         se       lcl       ucl
## 1 female   1 0.9042882 0.04135651 0.7873900 0.9601646
## 2   male   1 0.8770862 0.05029990 0.7408957 0.9468294
##
## $pent
##   time occ  estimate         se        lcl       ucl
## 1    2   2 0.2507505 0.03980350 0.18095910 0.3364033
## 2    3   3 0.1505344 0.03324417 0.09621815 0.2277848
## 3    4   4 0.1945310 0.03552416 0.13409798 0.2735936
## 4    5   5 0.1023335 0.02796115 0.05906948 0.1715092
##
## $N
##   estimate       se       lcl      ucl
## 1 3.254993 2.016763 0.9663716 10.96367
```

Die (konstante) Überlebenswahrscheinlichkeit liegt damit bei 63,9%, die konstante Fangwahrscheinlichkeit

bei 88,7%, pent bei 17,3% und die Anzahl nicht markierter Individuen liegt bei 3, die Superpopulationsgröße liegt damit bei 145+3=148.

Die Populationsgröße ist hierbei ein abgeleiteter Parameter.

```
# N.derived besteht aus:
# 5 Fanggelegenheiten
# 4 Schätzungen der Überlebensrate
# Nsuper Schätzung + Anzahl markierter Tiere
# Achtung: Summe aller pent muss 1 sein

N.derived <- data.frame(occ = c(1:5),
                        Phi = c(rep(gecko.js.predicted$Phi$estimate, 4), NA),
                        Nsuper = rep(gecko.js.predicted$N$estimate + nrow(gecko2), 5),
                        pent = c(1-sum(gecko.js.predicted$pent$estimate),
                                 gecko.js.predicted$pent$estimate))

# Spalte für N
N.derived$N <- NA

# Die initale Populationsgröße (N[1]) = Nsuper * (1 - sum(aller anderen pent Schätzungen))
# Dies begründet sich in der link Funktion für die Schätzung von pent.
# Die Summe aller pent Parameter MUSS 1 sein (daher wird eine weniger geschätzt)

N.derived$N[1] <- (N.derived$Nsuper[1] * N.derived$pent[1])

# Alle folgenden Populationsgrößen werden geschätzt durch die
# Anzahl überlebender Tiere als (N[t-1] * Phi[t]),
# dann werden alle Zugänge addiert (Nsuper * pent[t])
for(i in 2:nrow(N.derived)){
  N.derived$N[i] <- (N.derived$N[i-1]*N.derived$Phi[i-1]) +
    (N.derived$Nsuper[i] * N.derived$pent[i])
}

N.derived
```

```
##   occ       Phi  Nsuper      pent        N
## 1   1 0.6367597 148.255 0.3018507 44.75087
## 2   2 0.6367597 148.255 0.2507505 65.67057
## 3   3 0.6367597 148.255 0.1505344 64.13385
## 4   4 0.6367597 148.255 0.1945310 69.67804
## 5   5        NA 148.255 0.1023335 59.53961
```

In marked können keine Standardfehler für die Populationsgröße geschätzt werden. Dies ist jedoch in RMark möglich. Dies schauen wir uns daher unten an.

JS-Modelle mit mehreren individuellen Parametern

Damit hier nicht alle Parameter für alle Berechnungen genutzt werden, macht es Sinn, dies bereits beim Design zu berücksichtigen. So ändern sich Umweltparameter mit der Zeit, die individuellen Parameter jedoch (in dieser Variante) nicht. Nach Grimm-Seyfarth et al. (2018) ist die Überlebenswahrscheinlichkeit der adulten Tiere zwar nicht von Umweltfaktoren abhängig, wohl aber die Populationsgröße, bei der die Sommertemperatur die größte Rolle spielte. Wir nutzen die in Grimm-Seyfarth et al. (2018) angegebenen Sommertemperaturen und fügen sie zu unserem Datensatz hinzu. Für 2012-2016 betragen die Werte 31,48, 34,73, 34,00, 34,20, und 35,05.

```
Tsummer <- matrix(rep(c(31.48, 34.73, 34.00, 34.20, 35.05),each=nrow(gecko2)),ncol=5)
colnames(Tsummer)=paste("Tsummer",1:5,sep="")

gecko3 <- data.frame(ch = ch2,
                     sex = gecko$Sex,
                     residency = gecko$Residency,
                     catchability_mean = gecko$Catchability_mean,
                     mass = gecko$average_mass,
                     svl = gecko$average_SVL)
gecko3=cbind(gecko3,Tsummer)

head(gecko3)
```

```
##      ch    sex residency catchability_mean  mass  svl Tsummer1 Tsummer2 Tsummer3 Tsummer4
## 1 01000   male  Resident               4.0 3.260 4.90    31.48    34.73       34     34.2
## 2 10000   male  Resident               1.0 2.990 5.00    31.48    34.73       34     34.2
## 3 01000 female  Resident               5.0 2.360 5.00    31.48    34.73       34     34.2
## 4 11111   male   Floater               2.2 3.140 5.02    31.48    34.73       34     34.2
## 5 10000 female  Resident               1.0 3.460 5.00    31.48    34.73       34     34.2
## 6 11111 female   Floater               4.2 2.914 5.00    31.48    34.73       34     34.2
##   Tsummer5
## 1    35.05
## 2    35.05
## 3    35.05
## 4    35.05
## 5    35.05
## 6    35.05
```

```
names(gecko3)
```

```
##  [1] "ch"               "sex"              "residency"        "catchability_mean"
##  [5] "mass"             "svl"              "Tsummer1"         "Tsummer2"
##  [9] "Tsummer3"         "Tsummer4"         "Tsummer5"
```

Jetzt müssen wieder alle Daten prozessiert werden und die Designmatrix erstellt werden. Außerdem muss definiert werden, welcher Prozess welchen Parameter (hier meist als Covariaten bezeichnet) nutzt.

```
gecko.proc=process.data(gecko3, model="probitCJS")
# Designdaten erstellen mit statischen (static) und zeitlich variablen covariaten (time.varying)
design.Phi=list(static=c("sex","residency","mass","svl"),
                time.varying=c("Tsummer"))
design.p=list(static=c("sex","catchability_mean","mass","svl"),
              time.varying=c("Tsummer"))
design.parameters=list(Phi=design.Phi,p=design.p)
ddl=make.design.data(gecko.proc,parameters=design.parameters)
names(ddl$Phi)
```

```
##  [1] "id"        "occ"       "time"      "cohort"    "age"       "Tsummer"   "sex"
##  [8] "residency" "mass"      "svl"       "Time"      "Cohort"    "Age"       "order"
```

```
names(ddl$p)
```

```
##  [1] "id"               "occ"              "time"             "cohort"
##  [5] "age"              "Tsummer"          "sex"              "catchability_mean"
##  [9] "mass"             "svl"              "Time"             "Cohort"
## [13] "Age"              "order"
```

Da dies sehr complexe Modelle sind, nutzen wir die MCMC CJS-Modelle (probitCJS) - was wir bereits in den prozessierten Daten spezifiziert haben. Die Argumente burnin und iter kontrollieren hierbeit die Anzahl der burnin Iterationen und die Anzahl der Iterationen nach dem burnin Prozess.

```
model.parameters=list(Phi=list(formula=~Tsummer+sex+residency+mass+svl),
                      p=list(formula=~time+sex+mass+svl+catchability_mean))

MCMCfit=crm(gecko3,model="probitCJS",
             model.parameters=model.parameters,
             design.parameters=design.parameters,
             burnin=1000,iter=5000)
```

```
## Approximate time till completion:  0  minutes
## 10 % completed
## 20 % completed
## 30 % completed
## 40 % completed
## 50 % completed
## 60 % completed
## 70 % completed
## 80 % completed
## 90 % completed
## 100 % completed
```

```
MCMCfit
```

```
##
## crm Model Summary
##
## Npar :  14
##
## Beta
##                              mode        mean         sd    CI.lower    CI.upper
## Phi.(Intercept)        0.25492393  0.99864168 4.59455380  -8.1573633  9.79478111
## Phi.Tsummer           -0.12560588 -0.11996902 0.09513434  -0.3065255  0.06055702
## Phi.sexmale            0.07918908  0.08257537 0.23298539  -0.3665048  0.54083869
## Phi.residencyResident -0.75525177 -0.71723005 0.24926859  -1.2136658 -0.27555823
## Phi.mass               0.41227231  0.50917253 0.47216210  -0.3826659  1.45274767
## Phi.svl                0.37502911  0.37073950 0.80204428  -1.1404658  2.00468785
## p.(Intercept)         -7.97162384 -9.63686393 8.40181030 -25.5752656  7.15160186
## p.time3                1.38553454  1.67633756 0.90611859   0.1563004  3.64093987
## p.time4                0.72385884  1.86971884 2.00137416  -0.6206308  6.61277209
## p.time5                0.75054563  7.26995008 5.78301656  -0.5026649 17.12506953
## p.sexmale             -0.28380813 -0.75111345 0.87924330  -2.4723266  0.72651018
## p.mass                 1.47031246  0.02988033 2.16611858  -3.9546008  3.83423087
## p.svl                  1.91843928  1.65298242 2.38018720  -3.1213896  5.91319315
## p.catchability_mean    0.58579242  1.42458339 1.01150536  -0.2155509  3.33360910
```

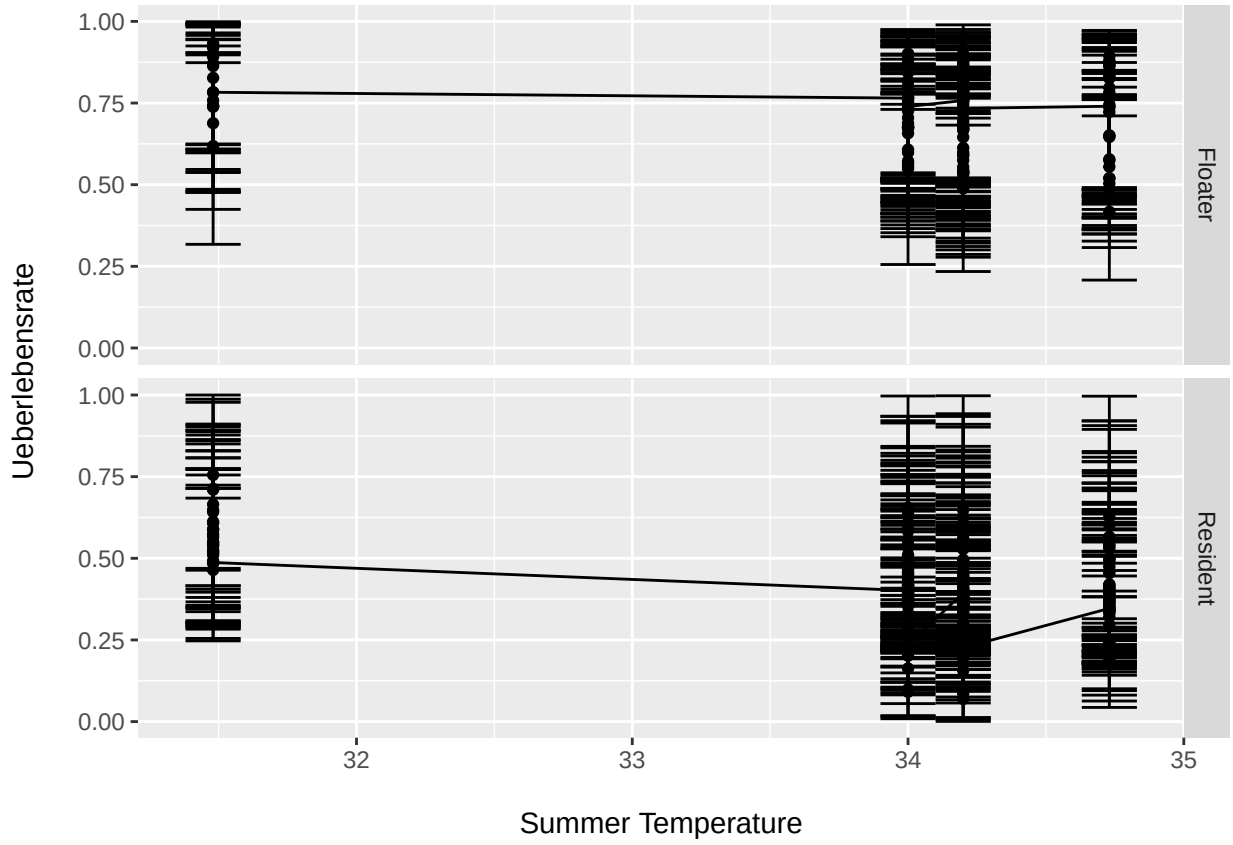
Wir sehen: steigende Sommertemperatur wirkt sich negativ auf das Überleben aus; es gibt kaum Geschlechtsunterschiede; Residents haben eine geringere Überlebensrate als Floater; und steigende Masse und Kopf-Rumpf-Länge erhöhen das Überleben. Hinsichtlich der Nachweiswahrscheinlichkeit sehen wir: sie unterscheidet sich massiv zwischen den Jahren; Männchen haben eine geringere Fängigkeit; schwerere Tiere haben eine geringere Fängigkeit, größere jedoch eine höhere; und es gibt offensichtlich Tiere, die generell häufiger gefangen werden als andere (individuelle Heterogenität - Achtung, dies könnte, wenn sie zu stark ausgeprägt ist, gegen die Modellannahmen verstoßen).

Nun benötigen wir wieder die tatsächlichen Schätzwerte:

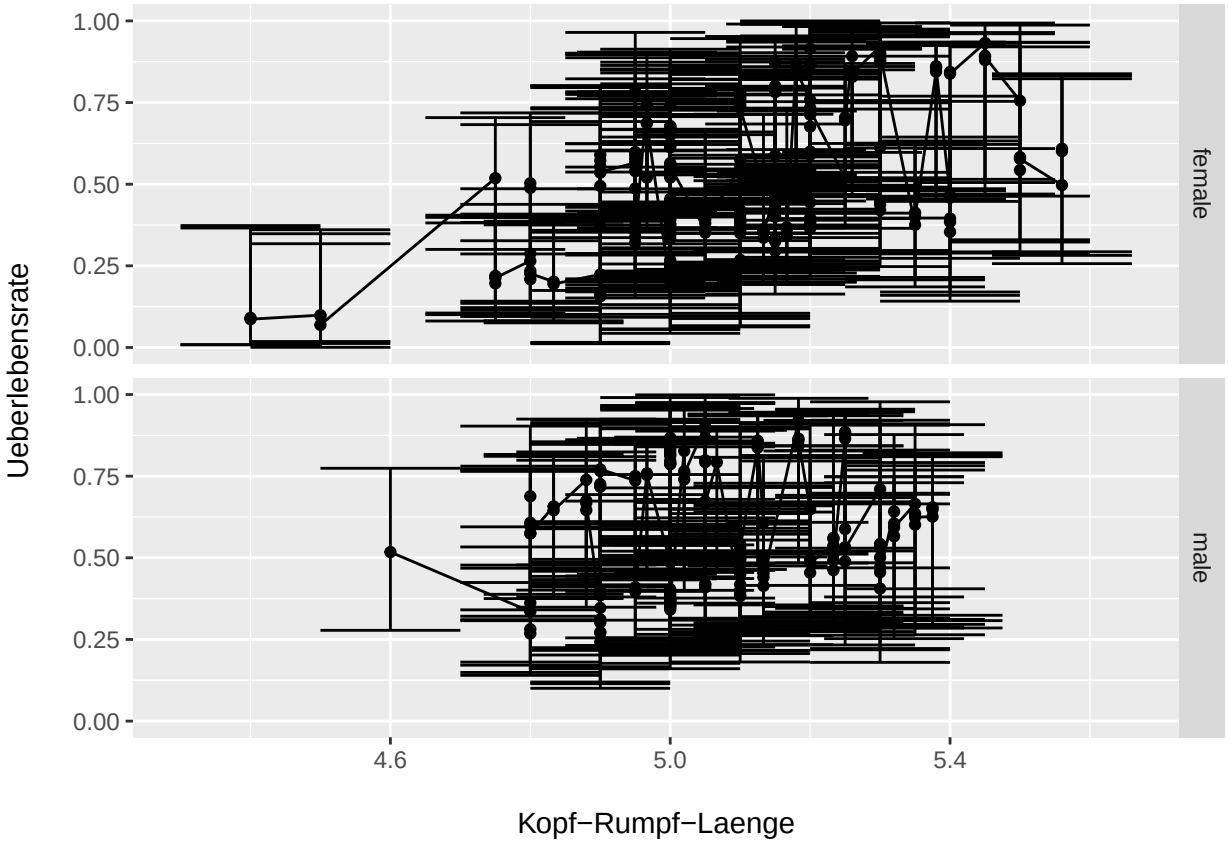
```
reals <- MCMCfit$results$reals
#head(reals)
```

Stellen wir diese einmal graphisch dar (wir benötigen dazu das Paket ggplot2):

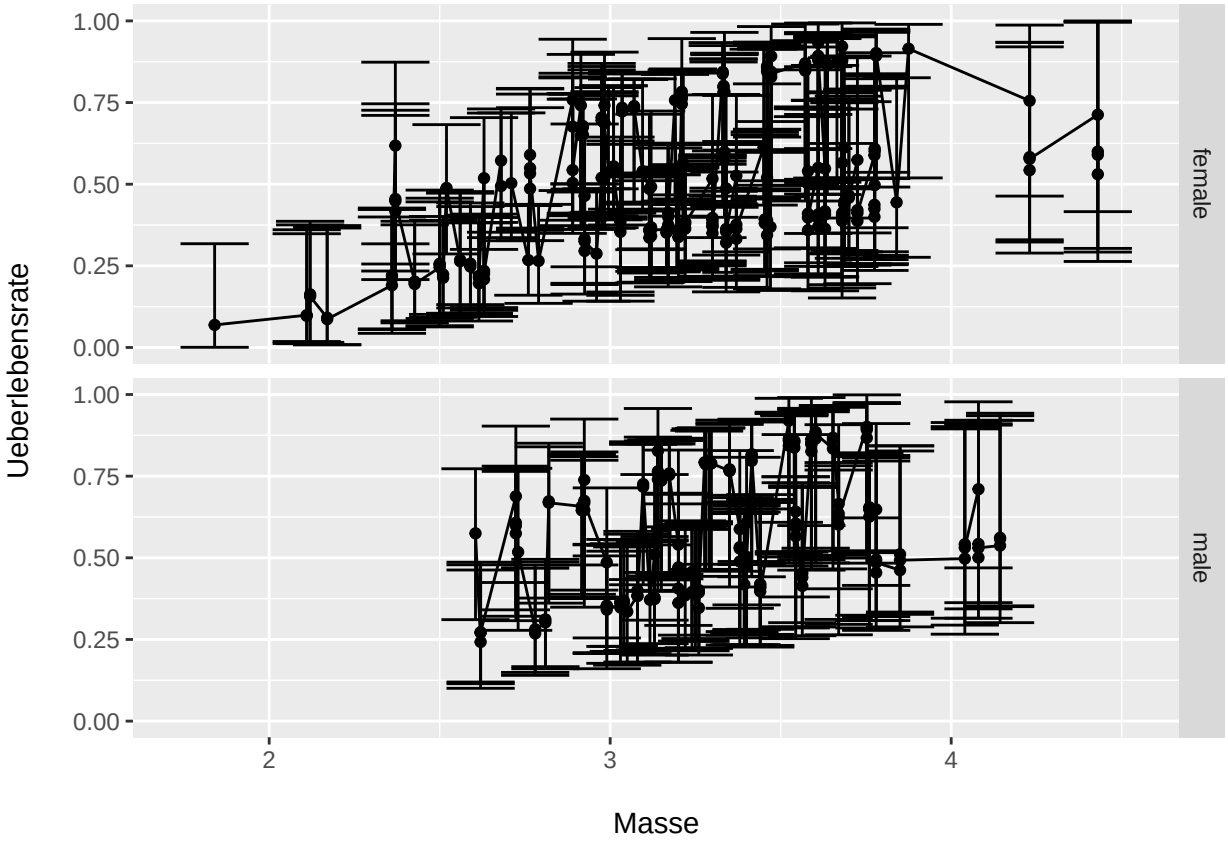
```
# Modell ohne MCMC
ggplot(reals$Phi,aes(Tsummer,estimate.mode,
                     ymin=estimate.CI.lower,ymax=estimate.CI.upper))+
       geom_errorbar(width=0.2)+geom_point()+geom_line()+
       xlab("\nSummer Temperature")+ylab("Ueberlebensrate\n")+facet_grid(residency~.)
```



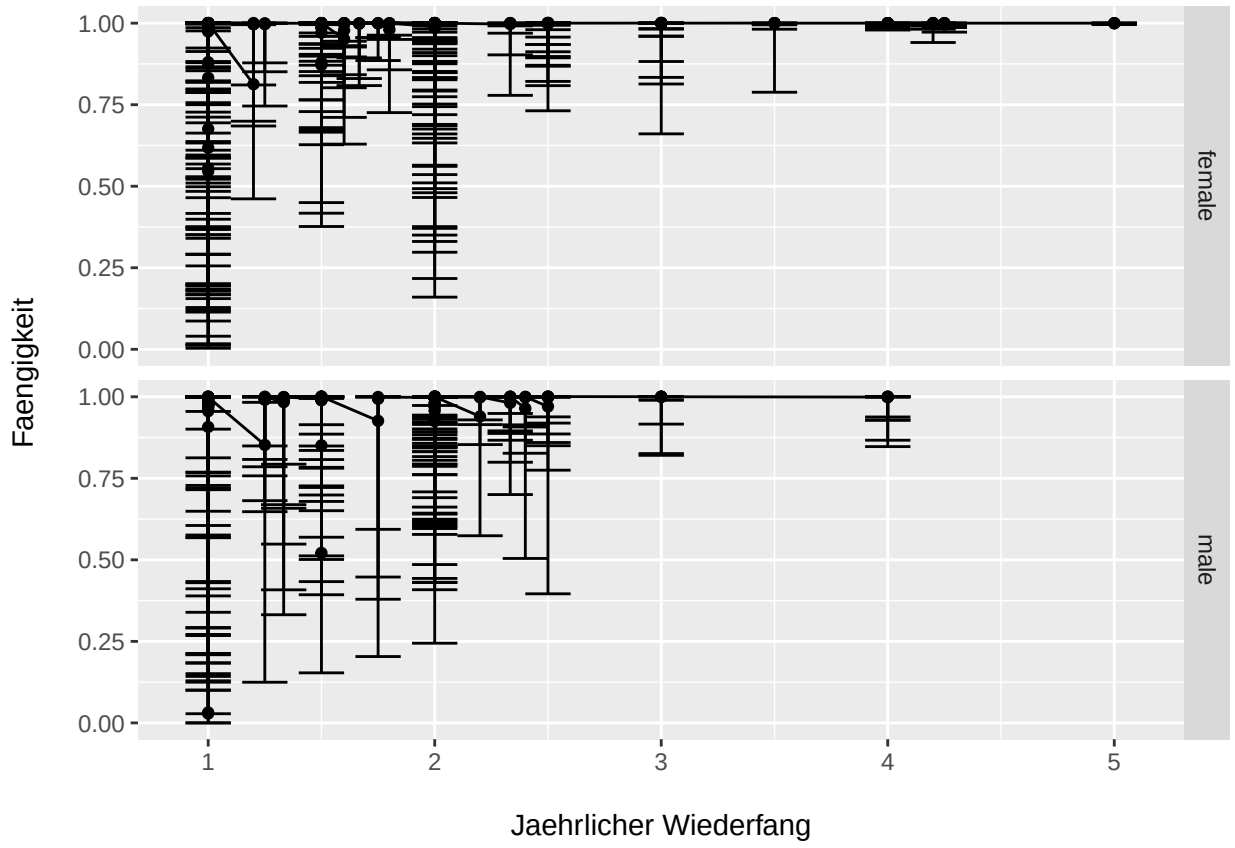

```
ggplot(reals$Phi,aes(svl,estimate.mode,
                     ymin=estimate.CI.lower,ymax=estimate.CI.upper))+
       geom_errorbar(width=0.2)+geom_point()+geom_line()+
       xlab("\nKopf-Rumpf-Laenge")+ylab("Ueberlebensrate\n")+facet_grid(sex~.)
```




```
ggplot(reals$Phi,aes(mass,estimate.mode,
                     ymin=estimate.CI.lower,ymax=estimate.CI.upper))+
       geom_errorbar(width=0.2)+geom_point()+geom_line()+
       xlab("\nMasse")+ylab("Ueberlebensrate\n")+facet_grid(sex~.)
```




```
ggplot(reals$p[order(reals$p$catchability_mean),],aes(catchability_mean,estimate.mode,
                       ymin=estimate.CI.lower,ymax=estimate.CI.upper))+
       geom_errorbar(width=0.2)+geom_point()+geom_line()+
       xlab("\nJaehrlicher Wiederfang")+ylab("Faengigkeit\n")+facet_grid(sex~.)
```



CJS-Modelle mit dem R-Paket RMark

RMark ist eine Sammlung, die analog zum Programm MARK entworfen wurde. Lesern wird hier nahegelegt, für eine detaillierte Beschreibung in das MARK Handbuch (http://www.phidot.org/software/mark/docs/book/) sowie in den User Guide von RMark (https://cran.r-project.org/web/packages/RMark/RMark.pdf) zu schauen.

Obwohl RMark langsamer ist als marked, ist die Berechnung der Konfidenzintervalle besser, speziell für die abgeleitete Populationsgröße. Es nutzt dabei die delta Methode. Dazu wird das POPAN Modell genutzt. Wir müssen wieder das Paket wechseln:

```
detach("package:marked", unload=TRUE)
```

```
##
## Bye-Bye from marked
```

```
library(RMark)
```

Wir bleiben der Einfachheit halber bei unserem ersten Geckobeispiel, gecko2. Die Datenprozessierung muss noch einmal mit diesem Paket durchgeführt werden. Außerden nutzen wir nur das erstbeste Modell, welches $\text{Phi}_{(1)\text{p}(}\text{sex})\text{pent}_{(\text{time})\text{N}(}1)$ war.

```
gecko.rmark.processed <- process.data(gecko2,
                                      model = "POPAN",
                                     group = "sex")
ddl1 <- make.design.data(gecko.rmark.processed)
Phi.dot <- list(formula=~1)
```

```r
p.sex <- list(formula=~sex)
pent.time <- list(formula=~time)
N.dot <- list(formula=~1)

# folgende Formel ähnelt der von marked,
# wir nutzen aber das POPAN modell in RMark
gecko.rmark <- mark(gecko.rmark.processed, model = "POPAN",
                    model.parameters = list(Phi = Phi.dot, p= p.sex,
                                            pent = pent.time, N = N.dot),
                    realvcv = TRUE)
```

```
##
## Output summary for POPAN model
## Name : Phi(~1)p(~sex)pent(~time)N(~1)
##
## Npar :  8
## -2lnL:  397.0074
## AICc :  413.5447
##
## Beta
##                     estimate        se        lcl        ucl
## Phi:(Intercept)   0.5613243 0.1475024  0.2722196  0.8504290
## p:(Intercept)     2.2458138 0.4778289  1.3092691  3.1823584
## p:sexmale        -0.2806856 0.5242592 -1.3082336  0.7468625
## pent:(Intercept) -0.1854767 0.2478651 -0.6712924  0.3003389
## pent:time3       -0.5102605 0.3072574 -1.1124849  0.0919640
## pent:time4       -0.2538599 0.2686945 -0.7805010  0.2727813
## pent:time5       -0.8962055 0.3377406 -1.5581770 -0.2342340
## N:(Intercept)     1.1801685 0.6196032 -0.0342538  2.3945908
##
##
## Real Parameter Phi
##                          1         2         3         4
## Group:sexfemale 0.6367589 0.6367589 0.6367589 0.6367589
## Group:sexmale   0.6367589 0.6367589 0.6367589 0.6367589
##
##
## Real Parameter p
##                          1         2         3         4         5
## Group:sexfemale 0.9042888 0.9042888 0.9042888 0.9042888 0.9042888
## Group:sexmale   0.8770869 0.8770869 0.8770869 0.8770869 0.8770869
##
##
## Real Parameter pent
##                          2         3         4         5
## Group:sexfemale 0.2507493 0.1505346 0.1945314 0.1023346
## Group:sexmale   0.2507493 0.1505346 0.1945314 0.1023346
##
##
## Real Parameter N
##                         1
## Group:sexfemale 88.25492
## Group:sexmale   88.25492
```

```
# Die Schätzwerte der Populationsgrößen bekommt man über die popan.derived Funktion
# Diese enthalten die 95% Konfidenzintervalle über die Delta Methode.
gecko.derived.rmark <- popan.derived(gecko.rmark.processed,
                                     gecko.rmark)$N

gecko.derived.rmark
```

```
##       sex Occasion        N       se      LCL      UCL
## 1  female        1 26.63975 3.706163 19.37567 33.90383
## 2  female        2 39.09296 3.589114 32.05829 46.12762
## 3  female        3 38.17821 3.354413 31.60356 44.75286
## 4  female        4 41.47867 3.674933 34.27580 48.68154
## 5  female        5 35.44345 3.662849 28.26426 42.62263
## 6    male        1 19.09350 2.707308 13.78718 24.39982
## 7    male        2 28.01908 2.656976 22.81141 33.22676
## 8    male        3 27.36346 2.477738 22.50709 32.21982
## 9    male        4 29.72899 2.720001 24.39779 35.06020
## 10   male        5 25.40337 2.694510 20.12213 30.68461
```

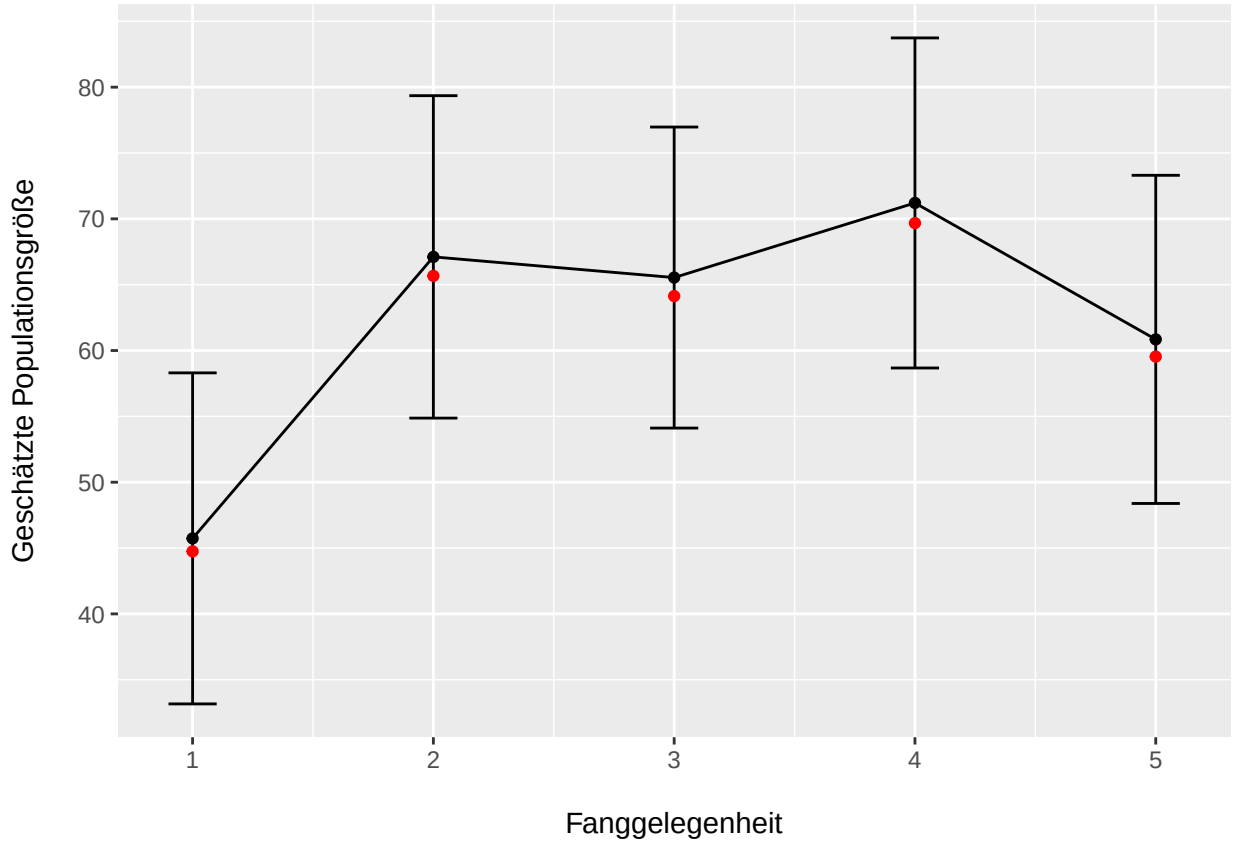
Achtung: Im output werden für alle Parameter Werte für Männchen und Weibchen angegeben, auch wenn diese identisch sind (so wie im Modell gesetzt). Damit ist der Output ähnlich dem aus dem Programm MARK. Das bedeutet aber auch, dass in der abgeleiteten Populationsgrößenschätzung separate Werte für Männchen und Weibchen berechnet werden. Um die Größen vergleichen zu können. müssen wir die Werte addieren.

```
gecko.N.rmark <- gecko.derived.rmark[1:5,3:6] + gecko.derived.rmark[6:10,3:6]
gecko.N.rmark$Occasion <- 1:5
gecko.N.rmark
```

```
##          N       se      LCL      UCL Occasion
## 1 45.73325 6.413471 33.16285 58.30365        1
## 2 67.11204 6.246091 54.86970 79.35438        2
## 3 65.54167 5.832150 54.11065 76.97268        3
## 4 71.20767 6.394935 58.67359 83.74174        4
## 5 60.84682 6.357358 48.38640 73.30724        5
```

Schauen wir uns nun an, wie verschieden die Schätzwerte von JS und POPAN sind:

```
ggplot(gecko.N.rmark,aes(Occasion,N,
                         ymin=LCL,ymax=UCL))+
       geom_errorbar(width=0.2)+geom_point()+geom_line()+
        geom_point(data=N.derived, aes(occ, N, ymin=N,ymax=N), color="red") +
       xlab("\nFanggelegenheit")+ylab("Geschätzte Populationsgröße\n")
```



Die Werte sind sehr ähnlich jedoch nicht identisch. Dies liegt an geringen Abweichungen in den ursprünglichen Schätzwerten. Während sich die Werte für phi (ca. 0,64), pent (0.25, 0.15, 0.19, 0.10) und p (female: 0.90, male: 0.88) praktisch nicht unterscheiden, ist der Wert für Nsuper verschieden (148 vs. 176). Dies führt zu leicht unterschiedlichen Schätzungen der Populationsgrößen.

Goodness-of-fit-test für CJS-Modelle

Wir führen den GoF-Test anhand des Datensatzes gecko2 durch. Diesen Datensatz müssen wir zunächst nach Geschlecht trennen.

```
gecko.hist <- matrix(as.numeric(unlist(strsplit(as.character(gecko2$ch),""))),
                     nrow = length(gecko2$ch),
                     byrow = T)
gecko.freq <- rep(1,nrow(gecko2))
gecko.group <- gecko2$sex
head(gecko.hist)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    1    0    0    0    0
## [3,]    0    1    0    0    0
## [4,]    1    1    1    1    1
## [5,]    1    0    0    0    0
## [6,]    1    1    1    1    1
```

```
# Einteilen in Gruppen (Geschlecht)
mask <- (gecko.group == "female")
```

```
gecko.fem.hist <- gecko.hist[mask,]
gecko.fem.freq <- gecko.freq[mask]
mask <- (gecko.group == "male")
gecko.mal.hist <- gecko.hist[mask,]
gecko.mal.freq <- gecko.freq[mask]
```

Nun können wir die Tests durchführen. Sie stammen aus dem bereits zu Beginn geladenen Packet R2ucare (Gimenez et al. 2018). Man kann die Tests einzeln aufrufen (test3sr, test3sm, test2ct und testcl), oder den Gesamttest mittels "overall_CJS". Test2 testet Fangheterogenität, Test3 testet, ob alle Tiere, die bei Fanggelegenheit i am Leben waren, die gleiche Wahrscheinlichkeit haben, nach Fanggelegenheit i+1 zu überleben. Eine Übersicht über die Tests sowie ein Entscheidungsschema geben wir in Kapitel 11.1.2.3 des Buches.

```
test3sr_females <- test3sr(gecko.fem.hist, gecko.fem.freq)
test3sm_females <- test3sm(gecko.fem.hist, gecko.fem.freq)
test2ct_females <- test2ct(gecko.fem.hist, gecko.fem.freq)
test2cl_females <- test2cl(gecko.fem.hist, gecko.fem.freq)
test.overall_females <- overall_CJS(gecko.fem.hist, gecko.fem.freq)
```

Schauen wir uns die einzelnen Testergebnisse an:

```
test3sr_females
```

```
## $test3sr
##      stat        df     p_val sign_test
##     4.123     3.000     0.248     0.210
##
## $details
##   component  stat p_val signed_test  test_perf
## 1         2  1.34 0.247      -1.158 Chi-square
## 2         3 2.763 0.096       1.662 Chi-square
## 3         4  0.02 0.888      -0.141 Chi-square
```

```
test3sm_females
```

```
## $test3sm
##  stat    df p_val
##     0     1     1
##
## $details
##   component stat df p_val test_perf
## 1         2    0  0     0      None
## 2         3    0  1     1    Fisher
## 3         4    0  0     0      None
```

```
test2ct_females
```

```
## $test2ct
##      stat        df     p_val sign_test
##         0         0         1         0
##
## $details
##   component dof stat p_val signed_test test_perf
## 1         2   0    0     0           0      None
## 2         3   0    0     0           0      None
```

```
test2cl_females
```

```
## $test2cl
##  stat    df p_val
##     0     0     1
##
## $details
##   component dof stat p_val test_perf
## 1         2   0    0     0      None
```

```
test.overall_females
```

```
##                           chi2 degree_of_freedom p_value
## Gof test for CJS model: 4.123                 4    0.39
```

Keiner der Tests ist signifikant. Die Modellannahmen werden also nicht verletzt. Wenn doch ein Test signifikant wäre, beschreiben wir in Kapitel 11.4 bis 11.10, welche Lösungsmöglichkeiten es geben könnte. Entscheiden wir uns für ein Modell, welches beispielsweise Änderungen der Fangwahrscheinlichkeit oder Durchzügler berücksichtigt, wollen wir sicherlich im Anschluss prüfen, ob die Anpassung das Problem behoben hat. Die Gesamtteststatistik (overall_CJS) ist dabei die Summer der vier Einzeltests (Test.3Sr, Test3.Sm, Test2.Ct and Test.Cl). Den Test für Änderungen der Fangwahrscheinlichkeit wäre dann folgendermaßen zu berechnen:

```
# Gesamttest
overall_test <- overall_CJS(gecko.fem.hist, gecko.fem.freq)
# Test für fangabhängige Änderungen von phi
twoct_test <- test2ct(gecko.fem.hist, gecko.fem.freq)
# Gesamtteststatistik - 2CT Teststatistik
stat_tp <- overall_test$chi2 - twoct_test$test2ct["stat"]
# Gesamttest-Freiheitsgrad - 2CT-Freiheitsgrad
df_tp <- overall_test$degree_of_freedom - twoct_test$test2ct["df"]
# p-Wert für Nullhypothese berechnen
pvalue <- 1 - pchisq(stat_tp, df_tp)
# Aufrufen des p-Werts
pvalue
```

```
##      stat
## 0.3896155
```

Die Anpassung würde in diesem Fall gut passen, da der Test nicht signifikant ist.

Literaturverzeichnis

Albert, J.H., & S. Chib. 1993. Bayesian-Analysis of Binary and Polychotomous Reponse Data. Journal of the American Statistical Association 88(422): 669–79.

Baillargeon, S. & L.P. Rivest. 2007. Rcapture: Loglinear models for capture-recapture in R. Journal of Statistical Software 19(5): 1–31.

Borchers, D.L. & M.G. Efford. 2008. Spatially Explicit Maximum Likelihood Methods for Capture-Recapture Studies. Biometrics 64(2): 377–85.

Colchero, F., O.R. Jones, & M. Rebke. 2012. BaSTA: an R package for Bayesian estimation of age-specific survival from incomplete mark-recapture/recovery data with covariates. Methods in Ecology and Evolution 3: 466–70.

Fiske, I.J. & R.B. Chandler. 2011. unmarked : An R Package for fitting hierarchical models of wildlife occurrence and abundance. 43(10): 1–23.

Gimenez, O., J.-D. Lebreton, R. Choquet, & R. Pradel. 2018. R2ucare: An R package to perform goodness-of-fit tests for capture–recapture models. Methods in Ecology and Evolution 9: 1749-1754.

Grimm-Seyfarth, A., J.-B. Mihoub, B. Gruber, & K. Henle. 2018. Some like it hot: from individual to population responses of an arboreal arid-zone gecko to local and distant climate. Ecological Monographs 88(3): 336–352.

Henle, K. 1990. Population ecology and life history of the arboreal gecko Gehyra variegata in arid Australia. Herpetological Monographs 4:30–60.

Laake, J. & E. Rexstad. 2008. RMark – an alternative approach to building linear models in MARK. In: Program MARK: A Gentle Introduction, edited by E. Cooch & G.C. White.

Laake, J.L., D.S. Johnson, & P.B. Conn. 2013. marked: An R package for maximum-likelihood and MCMC analysis of capture-recapture data. Methods in Ecology and Evolution 4: 885-890.

Laake, J.L., D.S. Johnson, & P.B. Conn.2023. marked Package Vignette. https://cran.r-project.org/web/packages/marked/vignettes/markedVignette.html, zuletzt aufgerufen: 18.03.2025.

McDonald, T.L., S.C. Amstrup, E.V. Regehr, & B.F.J. Manly. 2005. Examples. In: Handbook of Capture-Recapture Analysis, edited by S.C. Amstrup, T.L. McDonald, & B.F.J. Manly, 196–265. Princeton, New Jersey USA: Princeton University Press.

Ogle D.H., J.C. Doll, A.P. Wheeler, & A. Dinno 2023. FSA: Simple Fisheries Stock Assessment Methods. R package version 0.9.5, https://CRAN.R-project.org/package=FSA.

Pledger, S., K.H. Pollock, & J.L. Norris. 2003. Open capture-recapture models with heterogeneity: I. Cormack-Jolly-Seber model. Biometrics 59(4): 786–94.

Pollock, K.H., J.D. Nichols, C. Brownie, & J.E. Hines. 1990. Statistical inference for capture-recapture experiments. Wildlife Monographs, 107: 1-97.

Royle, J.A., K.U. Karanth, A.M. Gopalaswamy, & N.S. Kumar. 2009. Bayesian Inference in Camera Trapping Studies for a Class of Spatial Capture-Recapture Models. Ecology 90(11): 3233–44.

Schwarz, C.J. & A.N. Arnason. 1996. A general methodology for the analysis of capture-recapture experiments in open populations. Biometrics 52(3): 860–73.

Schwarz, C.J., D. Pickard, K. Marine, & S.J. Bonner. 2009. Juvenile Salmonid Outmigrant Monitoring Evaluation, Phase II, September 2009. Unpublished Report Prepared for the Trinity River Restoration Program, Weaverville, CA.

Seber, G.A.F. 2002. The Estimation of Animal Abundance. Edward Arnold, second edition (reprinted).

White, G.C. & K.P. Burnham. 1999. Program MARK: survival estimation from populations of marked animals. Bird Study 46: 120–39.

Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag, New York.