Beispiel 10.13. Populationsgrößenschätzung in R am Beispiel des baumbewohnenden Geckos *Gehyra variegata* aus dem Kinchega Nationalpark, Australien

Kapitel 10 aus Henle, K., A. Grimm-Seyfarth & B. Gruber: Erfassung und Analyse von Tierpopulationen. Ulmer Verlag

Annegret Grimm-Seyfarth

2025-04-26

Im Kinchega Nationalpark befinden sich zwei räumlich getrennte, benachbarte Geckopopulationen, welche 2016 an sieben (Auwaldgebiet, RI) bzw. sechs (Feldstation, Station) Tagen in Folge mittels Fang-Markierung-Wiederfang untersucht wurden. Wir wissen, dass es keine konstante Fangwahrscheinlichkeit gibt (Beispiel 10.5), sondern individuelle Heterogenität vorliegt (Henle 1990b, Grimm et al. 2014, Grimm-Seyfarth et al. 2018). Anhand des Studiendesigns – Fang am Ende der Reproduktionssaison (d. h., keine neuen Individuen durch Geburten), nur wenige Tage andauernde Primärperiode (d. h., keine Zu- und Abwanderungen sowie vernachlässigbare Mortalität erwartet) – gehen wir von einer geschlossenen Population aus. Während die Feldstation räumlich ohnehin geschlossen ist, was Zu- und Abwanderung in wenigen Tagen unwahrscheinlich macht, untersuchten wir in der Auwaldpopulation zusätzlich 19 umliegende Bäume, um wandernde Individuen zu entdecken. Dabei werden Individuen, die mindestens einmal im Kerngebiet erfasst werden, dem Kerngebiet hinzugeschlagen, während Individuen, die ausschließlich an den 19 umliegenden Bäumen entdeckt werden, nicht zur Population gerechnet werden. Somit können wir in unserem Design bereits sicherstellen, dass die Population geschlossen ist. Dieses Beispiel wurde bereits im Kapitel 9 des Buches, Beispiele 9.1 und 9.2 genutzt und auf Geschlossenheit geprüft. Weiterhin wurde die Populationsgröße bereits mittels des R-Pakes secr (Efford 2025) und verschiedenen Methoden berechnet. Daher verzichten wir auf eine Wiederholung der Berechnung mit diesem Paket. Hier zeigen wir die Populationsgrößenschätzung mittels der Pakete RMark (Laake 2013), Rcapture (Rivest & Baillargeon 2022) und CARE1 (Hsieh 2012).

```
# 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("RMark", "Rcapture", "CARE1")#

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

```
##    RMark Rcapture    CARE1
##     TRUE     TRUE     TRUE
```

Achtung, aktuell muss das Paket CARE1 direkt von der Website geladen werden, es ist nicht in CRAN verfügbar. Bitte hier die aktuellste Version downloaden: https://cran.r-project.org/src/contrib/Archive/CARE1/ Dann muss das Paket aus dem Verzeichnis installiert werden.

Weitere Informationen zur Nutzung der Pakete finden sich hier:

https://cran.r-project.org/package=RMark

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

CARE1: https://drive.google.com/file/d/1f7RoM8mkxa2HFmk0v1L_zfYySOWkG-Gj/view

Einlesen der Fangdaten

Wir laden zunächst die Fangdaten ein. In diesem Beispiel nutzen wir ausschließlich die Daten aus der Auwaldpopulation.

```
# Daten von Auwaldgebiet (RI) einlesen
ch.RI <- read.csv2("extdata/GV_RI_2016_capture_history2.csv", header = FALSE, row.names = NULL)
ch.RI <- as.matrix(ch.RI)
```

Schauen wir uns die daten einmal an:

```
head(ch.RI)
```

```
##      V1 V2 V3 V4 V5 V6 V7
## [1,]  0  1  0  0  0  1  1
## [2,]  1  1  1  1  1  1  0
## [3,]  1  0  0  0  0  0  0
## [4,]  0  0  1  0  0  0  1
## [5,]  1  0  0  1  0  0  1
## [6,]  0  0  0  1  0  0  0
```

Wir sehen für jede Fanggelegenheit 1 bis 7 die Fanggeschichte.

```
dim(ch.RI)
```

```
## [1] 108   7
```

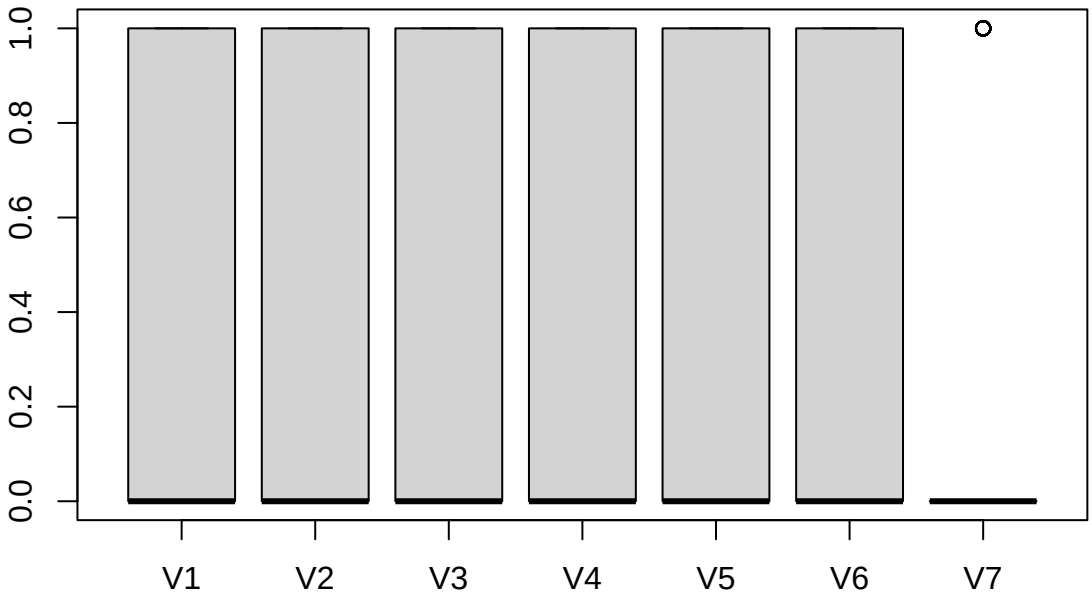
Es gibt 108 Individuen.

```
colSums(ch.RI)
```

```
## V1 V2 V3 V4 V5 V6 V7
## 29 31 32 31 34 33 20
```

Es wurden meist annähernd gleich viele Tiere gefangen, lediglich am letzten Fangtag sind es weniger.

```
boxplot(ch.RI)
```



Generell scheint die Fängigkeit am letzten Tag geringer zu sein. Eventuell ist es besser, den letzten Tag dann wegzulassen. Wir prüfen dies später.

Deskriptive Statistik

Mittels R-Paket Rcapture kann man sehr gut sich die deskriptive Statistik der Fangdaten anzeigen lassen:

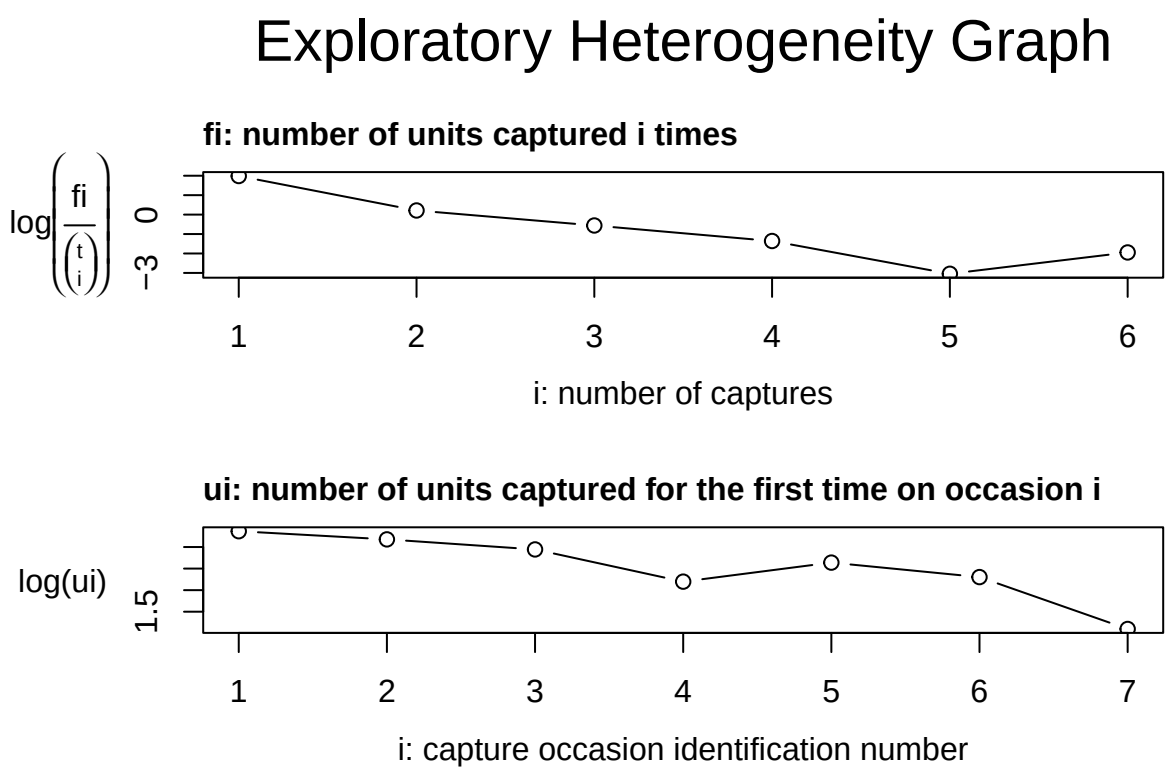
```
RI.desc <- descriptive(ch.RI)
RI.desc
```

```
##
## Number of captured units: 108
##
## Frequency statistics:
##        fi  ui  vi  ni
## i = 1  51  29  10  29
## i = 2  26  24   9  31
## i = 3  20  19  10  32
## i = 4   9   9  14  31
## i = 5   1  14  17  34
## i = 6   1  10  28  33
## i = 7   0   3  20  20
## fi: number of units captured i times
## ui: number of units captured for the first time on occasion i
## vi: number of units captured for the last time on occasion i
```

```
## ni: number of units captured on occasion i
```

Wir sehen hier die Fangfrequenzen fi (vgl. Kapitel 10.2 des Buches), die jeweilige Anzahl erstgefangener Tiere ui, die jeweilige Anzahl letztgefangener Tiere vi, und die Gesamtzahl Tiere der Sekundärperiode ni. Stellen wir das grafisch dar (Achsenbeschriftungen werden automatisch erstellt und sind daher in Englisch):

```
plot(RI.desc)
```



Populationsgrößenschätzung in Rcapture

Starten wir mit den Populationsgrößenschätzungen mit dem R-Paket Rcapture. Hier gibt es folgende Möglichkeiten:

closedp-Funktionen: Anpassung verschiedener loglinearer Modelle für Abundanzschätzungen;

closedpCI-Funktionen: Anpassung eines angepassten loglinearen Modells und Berechnung eines Konfidenzintervalls für die Abundanzschätzung;

closedpMS.t: passt verschiedene hierarchische loglineare Modelle für eine Modellselektion;

closedp.bc: führt Verzerrungskorrekturen an den Abundanzschätzungen aus angepassten loglinearen Modellen durch;

closedp.Mtb: passt das Modell Mtb an, das von keiner anderen Funktion angepasst werden kann.

Modelle vergleichen

Beginnen wir von oben:

```
RI.closedp <- closedp.t(ch.RI)
RI.closedp
```

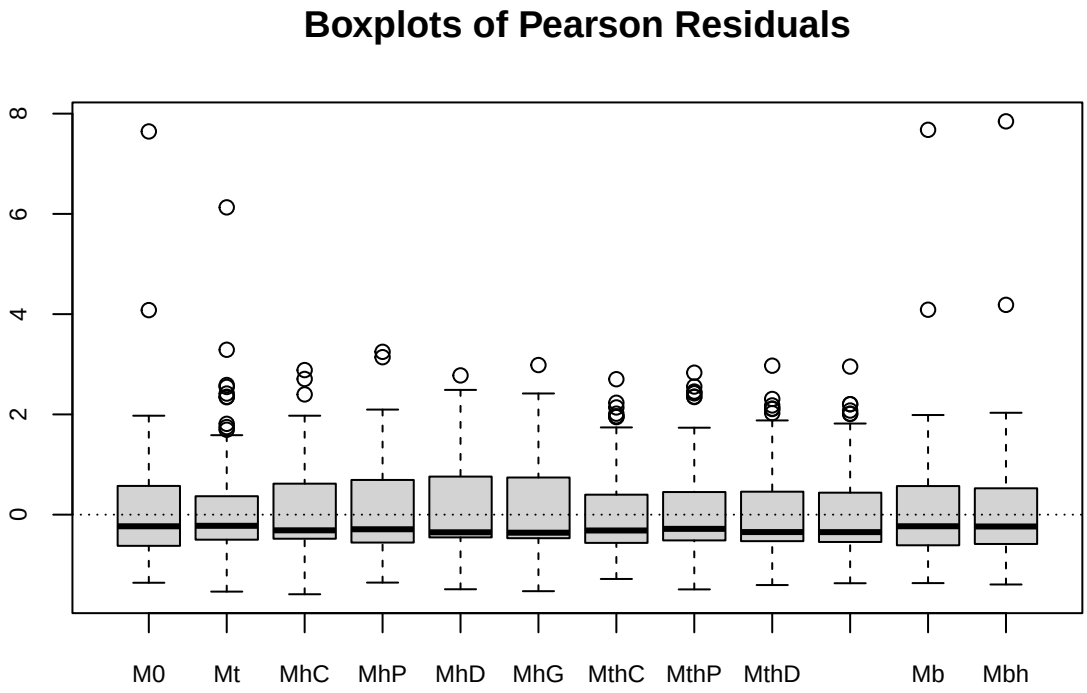
```
##
## Number of captured units: 108
##
## Abundance estimations and model fits:
##                abundance stderr deviance  df     AIC     BIC infoFit
## M0                 127.5    6.1  113.988 125 233.683 239.047      OK
## Mt                 127.2    6.1  107.849 119 239.544 261.001      OK
## Mh Chao (LB)       149.4   15.2  104.830 123 228.524 239.253      OK
## Mh Poisson2        132.9    7.8  110.415 124 232.110 240.156      OK
## Mh Darroch         151.4   17.2  107.780 124 229.475 237.521      OK
## Mh Gamma3.5        176.8   35.3  106.992 124 228.687 236.733      OK
## Mth Chao (LB)      149.2   15.1   98.437 117 234.132 260.953      OK
## Mth Poisson2       132.7    7.8  104.125 118 237.820 261.959      OK
## Mth Darroch        151.5   17.2  101.410 118 235.105 259.244      OK
## Mth Gamma3.5       177.5   35.7  100.612 118 234.307 258.446      OK
## Mb                 126.8    9.4  113.980 124 235.675 243.721      OK
## Mbh                124.9   18.2  113.850 123 237.544 248.273      OK
```

Folgende Modelle werden mittels AIC-Tabelle verglichen (für Abkürzungen empfehlen wir die Kapitel 9 und 10 des Buches): M0, Mt, Mh bzw. Mth Moment estimator (Chao 1987), verschiedene Mh/Mth Poisson- und Gammaverteilungen (beschrieben in Rivest and Baillargeon 2007), und Mh bzw. Mth Modelle von Darroch et al. (1993). Den niedrigsten AIC Wert weist in diesem Beispiel das Modell Mh von Chao (1987) auf, dicht gefolgt von Mh von Darroch et al. (1993).

Graphisch darstellen

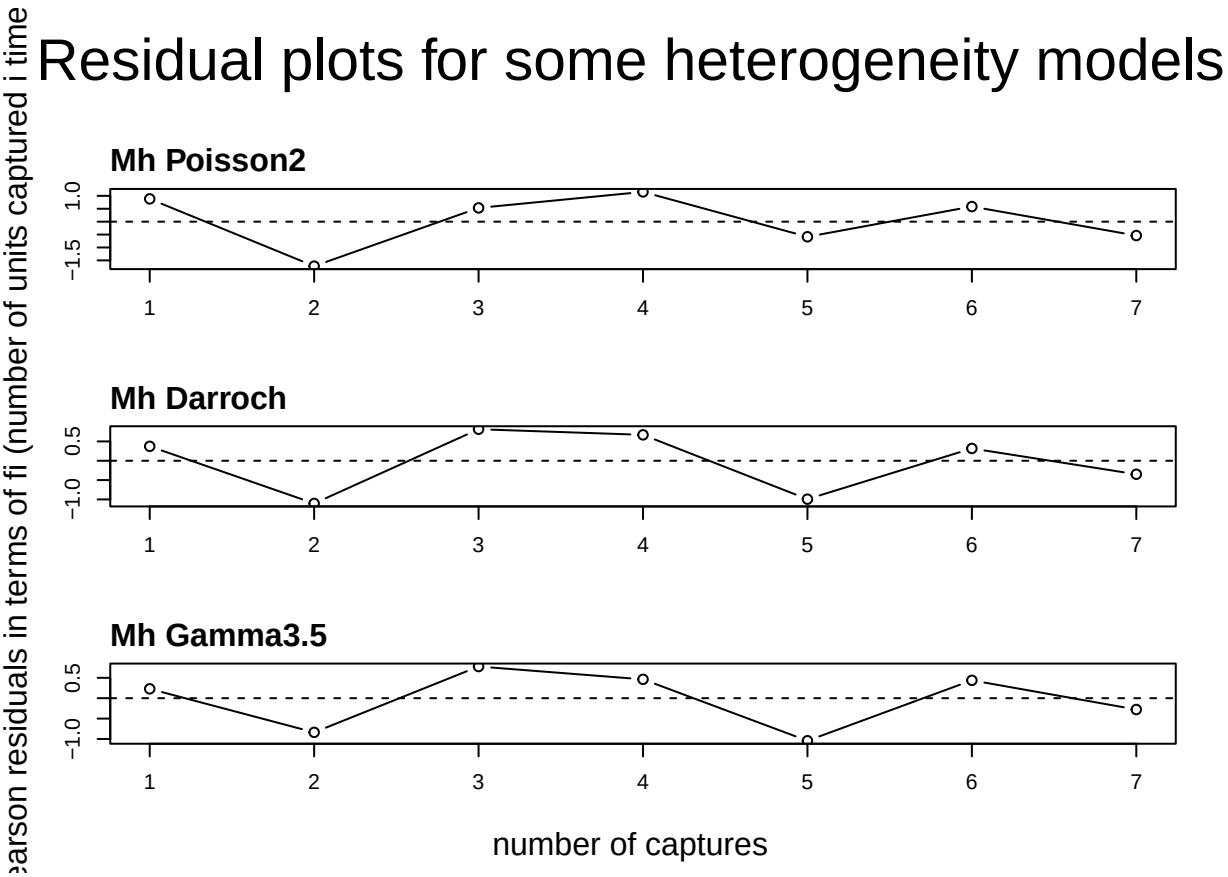
Die Funktion boxplot.closedp erzeugt Boxplots der Pearson-Residuen der angepassten loglinearen Modelle, die konvergiert haben.

```
boxplot(RI.closedp)
```



Die Funktion plot.closedp erzeugt Scatterplots der Pearson-Residuen in Form von fi (Fangfrequenzen) für die heterogenen Modelle Mh Poisson2, Mh Darroch und Mh Gamma3.5, sofern sie konvergieren. Achsenbeschriftungen werden automatisch erstellt und sind daher Englisch.

```
plot(RI.closedp)
```



Konfidenzintervalle und Berechnungen des ausgewählten Modells

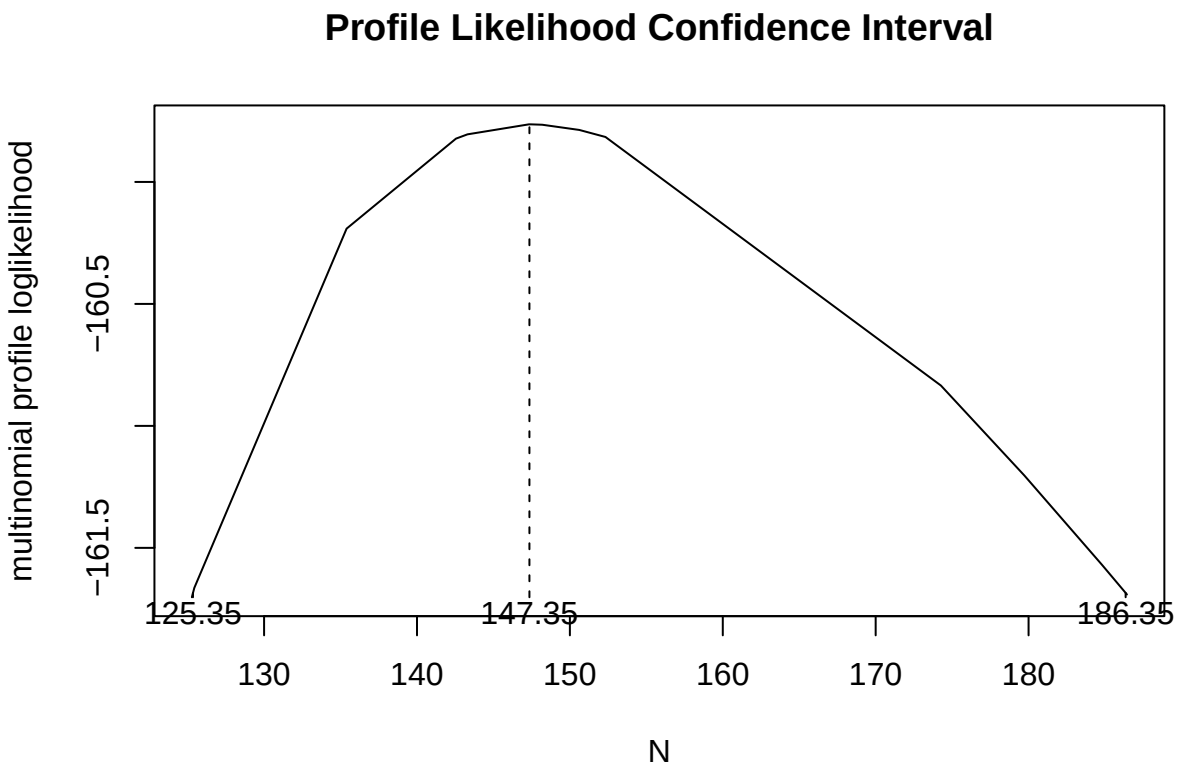
Wir wählen in jedem Fall ein Mh Modell aus. Starten wir mit dem Moment Estimator.

```
CI <- closedpCI.t(ch.RI, m = "Mh", h = "Chao")
CI
```

```
##
## Number of captured units: 108
##
## Poisson estimation and model fit:
##               abundance stderr deviance  df     AIC     BIC infoFit
## Mh Chao (LB)     149.4   15.2   104.83 123 228.524 239.253      OK
##
## Multinomial estimation, 95% profile likelihood confidence interval:
##               abundance infCL supCL infoCI
## Mh Chao (LB)     147.4 125.3 186.3     OK
```

Nun wissen wir, dass die Populationsgröße bei 147 Tieren liegt und mit 95% zwischen 125 und 186. Dieses Konfidenzintervall können wir uns noch graphisch anschauen (erneut automatische Achsenbeschriftungen):

```
plotCI(CI)
```



Vergleichen wir die Zahlen noch mit dem zweitbesten Modell von Darroch et al. (1993):

```
CI2 <- closedpCI.t(ch.RI, m = "Mh", h = "Darroch")
CI2
```

```
##
## Number of captured units: 108
##
## Poisson estimation and model fit:
##            abundance stderr deviance  df     AIC     BIC infoFit
## Mh Darroch     151.4   17.2   107.78 124 229.475 237.521      OK
##
## Multinomial estimation, 95% profile likelihood confidence interval:
##            abundance infCL supCL infoCI
## Mh Darroch     148.9 124.7 193.6     OK
```

Es werden nahezu identische Werte berechnet, das Konfidenzintervall ist lediglich etwas breiter.

Populationsgrößenschätzungen in CARE1

Aus der Fanggeschichte muss zunächst eine für CARE1 nutzbare Importdatei kreiert werden. Dazu gibt es die Funktion *as.record*.

```
RI.CARE <- as.record(ch.RI)
head(RI.CARE)
```

```
## 0000001 0000010 0000011 0000100 0000101 0000110
##       3       9       1       8       2       4
```

Bei diesem Befehl werden alle möglichen Fanggeschichten erstellt und jeweils aufaddiert, wie oft diese spezielle Fanggeschichte vorkommt.

Modelle aufrufen

In CARE1 werden alle Modelle zur Populationsgrößenschätzung mit einem Befehl aufgerufen. Die Ausführung dauer einen Moment.

```
CARE1.print(RI.CARE)
```

```
## (1) NUMBER OF IDENTIFIED CASES IN EACH LIST:
## n1 n2 n3 n4 n5 n6 n7
## 29 31 32 31 34 33 20
##
## (2) ESTIMATES BASED ON ANY PAIR OF SAMPLES:
##      Petersen Chapman se cil ciu
## pa12      128     119 30  81 206
## pa13      133     123 31  84 213
## pa14      112     106 24  75 176
## pa15       99      94 18  71 147
## pa16      137     126 32  86 220
## pa17      145     125 41  76 251
## pa23      124     116 27  82 194
## pa24      107     101 21  74 163
## pa25      132     123 29  86 207
## pa26      114     108 23  78 174
## pa27      103      95 24  65 169
## pa34       90      87 15  67 132
## pa35       99      95 17  73 145
## pa36      132     124 29  87 207
## pa37      107      98 25  67 175
## pa45      132     123 29  86 207
## pa46      102      98 19  73 153
## pa47      124     111 32  72 209
## pa56       94      91 15  71 135
## pa57      113     104 27  71 186
## pa67      132     118 34  76 223
##
##
## Note1: Refer to Seber(1982,pages 59 and 60) for Petersen estimator
##        and Chapman estimators as well as s.e formula.
## Note2: A log-transformation is used is used to obtain the confidence
##        interval so that the lower limit is always greater than the
##        number of ascertained. Refer to Chao(1987,Biometrics,43,783-791)
##        for the construction of the confidence interval.
```

```
##
## (3) SAMPLE COVERAGE APPROACH:
##          M       D  Chat est se cil ciu
## Nhat-0 108 100.714 0.761 132  7 122 152
## Nhat   108 100.714 0.761 160 24 129 232
## Nhat-1 108 100.714 0.761 150 17 128 198
##
## Parameter estimates:
##          u1   u2   u3   u4   u5   u6   u7  r12  r13  r23  r14  r24  r34  r15
## Nhat-0 0.22 0.23 0.24 0.23 0.26 0.25 0.15 0.03 0.00 0.07 0.18 0.24 0.47 0.34
## Nhat   0.18 0.19 0.20 0.19 0.21 0.21 0.13 0.24 0.20 0.29 0.42 0.49 0.77 0.62
## Nhat-1 0.19 0.21 0.21 0.21 0.23 0.22 0.13 0.17 0.13 0.21 0.33 0.40 0.66 0.52
##         r25  r35  r45   r16  r26  r36  r46  r56   r17  r27  r37  r47  r57  r67
## Nhat-0 0.00 0.34 0.00 -0.03 0.16 0.00 0.29 0.42 -0.09 0.28 0.24 0.07 0.17 0.00
## Nhat   0.21 0.61 0.21  0.17 0.40 0.21 0.56 0.71  0.10 0.54 0.50 0.29 0.41 0.21
## Nhat-1 0.14 0.52 0.14  0.10 0.32 0.14 0.47 0.60  0.03 0.45 0.41 0.21 0.32 0.14
##
##
## Definitions for the sample coverage approach:
## M: number of individuals ascertained in at least one list.
## D: the average of the number of invididuals listed in the combination
##    of any two lists omitting the other one.
## C^: sample coverage estimate, see Equation (14) of Chao and Tsay(1998).
##     est: population size estimate.
## se: estimated standard error of the population size estimation based on
##     bootstrap replications.
## cil: 95% confidence interval lower limit(using a log-transformation).
## ciu: 95% confidence interval upper limit(using a log-transformation).
## Nhat: Population size estimate for sufficiently high sample coverage
##       cases, see Equation (20) of Chao and Tsay (1998).
## Nhat-1: One-step population size estimate for low sample coverage cases;
##         see Equation (2.21) of Chao et al. (1996). This estimator is
##         suggested for use when the estimated se of Nhat is relatively large.
## u1,u2,u3: estimated mean probabilities depending on the estimate of N.
## r12,r13,r23 etc.: estimated coefficient of covariation(CCV) depending on the
##                   estimate of N.
##
```

Der Output besteht aus drei Teilen. Der erste Teil der Ausgabe gibt die Anzahl der identifizierten Individuen in jeder der sieben Sekundärperioden (hier Listen genannt) an. Der zweite Teil enthält die Petersen- und Chapman-Schätzungen zusammen mit dem Standardfehler (s.e.) und den Konfidenzintervallen (cil - unteres, ciu - oberes) für jedes Paar von Sekundärperioden. Diese Schätzungen können als vorläufige Analyse verwendet werden, um mögliche Abhängigkeiten zwischen Sekundärperioden zu erkennen.

Im dritten Teil werden die Sample Coverage Populationsgrößenschätzer (Chao et al. 2001, Chao and Tsay 1998) zusammen mit den zugehörigen Statistiken vorgestellt. Der geschätzte Stichprobenerfassungsgrad oder Überschneidungsanteil (Sample Coverage) beträgt C = 76,1 % (Chat in der Ausgabe), was als hoch angesehen werden kann. Der Durchschnitt (über alle Listen) der sich überschneidenden Fälle ist D = 100,71. Wenn man von der Unabhängigkeit der Stichproben ausgehen würde, ergäbe sich eine geschätzte Populationsgröße von N0 = D/C = 132 (Nhat-0 in der Ausgabe), mit einem Bootstrap s.e. von 8 auf der Grundlage von 1000 Bootstrap-Replikationen. Die untere Grenze des 95%-Konfidenzintervalls (cil) beträgt 121 und die obere Grenze (ciu) beträgt 153. Achtung: Selbst bei gleichen Eingabedaten sind der Bootstrap-Schätzwert und das Konfidenzintervall bei wiederholten Durchläufen von CARE1 verschieden aufgrund von Variationen bei der Wiederholungsstichprobe in den Bootstrapverfahren. Da jedoch die Sample Coverage hoch ist und auch der Koeffizient der Kovariation (CCV, dargestellt als r12, r13 etc.) häufig relativ hoch scheint, liegt

wohl individuelle Heterogenität vor und der N0-Schätzer ist bedeutungslos. Beachten wir nun die Sample Coverage und nutzen entsprechend das Modell Nhat, bekommen wir eine Populationsgröße von 160 mit einem s.e. von 23 und einem 95%-Konfidenzintervall zwischen 130 und 228. Dies liegt nur unwesentlich höher als der Moment estimator und die Schätzung nach Darroch, wobei sich alle Konfidenzintervalle überlappen und Schätzungen in den jeweils anderen Konfidenzintervallen enthalten sind.

Für weitere Analysen empfehlen wir Chao 2015 sowie Chao & Yang 2006 (die Desktop-Version von CARE - diese hatte neben den Sample Coverage Schätzern auch die Estimating Equations enthalten).

Populationsgrößenschätzungen in RMark

Anders als in Rcapture und CARE1 werden in RMark nicht eine Reihe bekannter Schätzer gerechnet, sonder über eine Parameter-Index-Matrix (PIM). Möchte man mit MARK diese Modelle rechnen, würde man ebenfalls das Programm CAPTURE innerhalb von MARK aufrufen, andernfalls nutzt man auch in MARK das PIM-Design. Daher zeigen wir hier nur relativ einfache Anwendungen. Gute Anwendungsbeispiele findet man, neben dem User-guide, hier:

https://www.montana.edu/rotella/documents/502/lab07RMark.html

https://pdixon.stat.iastate.edu/stat534/RMark/Intro.pdf

Fanggeschichte einlesen

Leider funktioniert die Fanggeschichte nicht so, wie wir sie für Rcapture und CARE1 einlesen. Zum Einlesen gibt es zwei Varianten: (1) die .inp Datei einlesen, die auch für MARK genutzt wird, mittels *convert.inp()*, oder (2) die reine Fanggeschichte einlesen mittels *import.chdata()*. Wir nutzen hier exemplarisch die zweite Version. Wichtig ist hier, dass die erste Zeile als *ch* bezeichnet wird

```
ch.RI2 <- import.chdata("extdata/GV_RI_2016_capture_history.txt")
head(ch.RI2)
```

```
##        ch
## 1 0100011
## 2 1111110
## 3 1000000
## 4 0010001
## 5 1001001
## 6 0001000
```

M0-Modell

Ein einfaches Modell ohne Parameter (also ein M0-Modell) würde man wie folgt aufrufen:

```
f0 <- list(formula=~1)
m0 <- mark(ch.RI2, model="Closed", model.parameters=list(p=f0, c=f0))
```

```
##
## Output summary for Closed model
## Name : p(~1)c(~1)f0(~1)
##
## Npar :  3
```

```
## -2lnL:  67.5095
## AICc :  73.54142
##
## Beta
##                 estimate        se       lcl        ucl
## p:(Intercept)  -1.120564 0.2150623 -1.542087 -0.6990423
## c:(Intercept)  -1.183170 0.1131677 -1.404978 -0.9613610
## f0:(Intercept)  2.826393 0.5164896  1.814074  3.8387128
##
##
## Real Parameter p
##          1         2         3         4         5         6         7
##  0.2459066 0.2459066 0.2459066 0.2459066 0.2459066 0.2459066 0.2459066
##
##
## Real Parameter c
##          2         3         4         5         6         7
##  0.2344828 0.2344828 0.2344828 0.2344828 0.2344828 0.2344828
##
##
## Real Parameter f0
##         1
##  16.88445
```

mark() ruft eigentlich 5 Funktionen nacheinander auf, um die Daten zu verarbeiten, die Designmatrix zu erstellen, das Modell laufen lassen (durch Schreiben einer .inp-Datei auf die Festplatte und anschließendes Ausführen von MARK, das 3 Ausgabedateien erzeugt), dann das Sammeln und Organisieren dieser Ausgabe als R-Objekte. Der Output enthält Infos zum genutzten Modell, Anzahl Parameter und AICc, beta-Schätzungen (Link-Skala) und die rücktransformierten (backtransformed) Koeffizienten (als zeitspezifische Werte).

Daten für verschiedene Modelle vorbereiten

Wie wir es aus anderen RMark Beispielen kennen, können wir auch die Daten zuerst prozessieren, dann verschiedene Modelle in einem wrapper schreiben und schließlich alle ausführen und vergleichen lassen. Dabei kann man hier auch Kovariablen einbeziehen, was wir hier aber nicht tun. Es könnte aber z. B. die Temperatur sich auf die Nachweiswahrscheinlichkeit auswirken.

```
# Daten prozessieren
RI.pr <- process.data(ch.RI2, begin.time = 1, model = "Closed")

# Standarddesign erstellen
RI.ddl <- make.design.data(RI.pr)

# Möchte man M(bh) Modelle rechnen, benötigt man zudem eine separate Zeitspalte
# 1. Fanggelegenheit 0, später 1
# das müssen wir für p und c anpassen
RI.ddl$p$t2 <- 0
RI.ddl$p$t2 = ifelse(RI.ddl$p$Time == 0, 0, 1)
RI.ddl$c$t2 <- 0
RI.ddl$c$t2 <- ifelse(RI.ddl$c$Time == 0, 0, 1)
```

```
# schauen wir die Daten an
RI.ddl
```

```
## $p
##   par.index model.index group time Time c t2
## 1         1           1     1    1    0 0  0
## 2         2           2     1    2    1 0  1
## 3         3           3     1    3    2 0  1
## 4         4           4     1    4    3 0  1
## 5         5           5     1    5    4 0  1
## 6         6           6     1    6    5 0  1
## 7         7           7     1    7    6 0  1
##
## $c
##   par.index model.index group time Time c t2
## 1         1           8     1    2    0 1  0
## 2         2           9     1    3    1 1  1
## 3         3          10     1    4    2 1  1
## 4         4          11     1    5    3 1  1
## 5         5          12     1    6    4 1  1
## 6         6          13     1    7    5 1  1
##
## $f0
##   par.index model.index group age time Age Time
## 1         1          14     1   0    1   0    0
##
## $pimtypes
## $pimtypes$p
## $pimtypes$p$pim.type
## [1] "all"
##
##
## $pimtypes$c
## $pimtypes$c$pim.type
## [1] "all"
##
##
## $pimtypes$f0
## $pimtypes$f0$pim.type
## [1] "all"
```

Verschiedene Modelle erstellen

Hier richten wir die Strukturen für 'p' und 'c' ein. Wir verwenden die Optionen „share=TRUE“ oder „share=FALSE“ in jeder der Strukturen, um anzugeben, ob „p“ und „c“ dieselben Spalten der Designmatrix teilen sollen oder nicht. Obwohl dies nicht für alle der unten aufgeführten Strukturen notwendig ist, wird eine Kovariate „c“ zu den Designmatritzen hinzugefügt, wobei c=0 ist für Zeilen, die zum Parameter „p“ gehören, und c=1 für Zeilen, die zum Parameter „c“ gehören. Das ist hilfreich, denn es gibt uns die Möglichkeit, einige der additiven Strukturen zu erstellen, an denen wir interessiert sein könnten. Wir können dann die Kovariate „c“ in Formelanweisungen verwenden, wenn wir das möchten. Aber wir müssen diese Kovariate nicht einbeziehen, wenn wir das nicht wollen (siehe z. B. die p.dot-Struktur). Hinweis: formula(=~time) gibt die Zeit als Faktor an. Die Zeit muss nicht im Datenframe enthalten sein. Jede Spalte der Erfassungshistorie

wird als separate Zeit behandelt (Mt). formula(=~Time) [großgeschrieben] gibt die Zeit als Regression an: f(param) = b0 + b1*Time (Mt mit steigender Nachweiswahrscheinlichkeit)

```
# Funktion zum Ausführen einer Reihe von Modellen für phi und für p
run.RI <- function() {

  # verschiedene Modellformeln erstellen, Parameter definieren
  p.dot = list(formula =  ~ 1, share = TRUE)
  p.time = list(formula =  ~ time, share = TRUE)
  p.Time <- list(formula = ~ Time, share=TRUE)
  p.time.behav.add = list(formula =  ~ time + c, share = TRUE)
  p.dot.behav = list(formula =  ~ 1, share = FALSE)
  p.bh.2p = list(formula =  ~ t2, share = FALSE)

  # Erstellen konkurrierender Modelle basierend auf den Strukturen für 'p' und 'c'
  RI.model.list = create.model.list("Closed")

  # HINWEIS: Wenn die Ausgabe für die einzelnen Modelle gezeigt werden soll,
  # entfernt man ', output=FALSE' nach 'ddl=caps.ddl'.

  RI.results = mark.wrapper(RI.model.list,
                            data = RI.pr,
                            ddl = RI.ddl,
                            output=FALSE)

  # Tabelle und Modellliste ausgeben
  return(RI.results)
}
```

Modelle laufen lassen

Hinweis: zum Überprüfen der einzelnen Modelle entfernt man oben im mark.wrapper den Befehl ‘, output=FALSE’. Wir lassen uns hier die Modellvergleichstabelle anzeigen.

```
RI.results <- run.RI()
```

Schauen wir uns die Modellvergleichstabelle an.

```
RI.results
```

```
##                   model npar     AICc DeltaAICc      weight Deviance
## 2        p(~1)c()f0(~1)    2 71.59017  0.000000 0.465281385 118.6213
## 5     p(~Time)c()f0(~1)    3 72.78631  1.196139 0.255845317 117.8014
## 3      p(~1)c(~1)f0(~1)    3 73.54142  1.951248 0.175391134 118.5565
## 1     p(~t2)c(~1)f0(~1)    4 75.32030  3.730129 0.072065673 118.3141
## 4     p(~time)c()f0(~1)    8 77.61713  6.026955 0.022854886 112.4714
## 6 p(~time + c)c()f0(~1)    9 79.58088  7.990710 0.008561604 112.3866
```

Eine umfassendere Version beinhaltet auch die einzelnen Parametereinstellungen aus dem PIM-Design sowie das Modell, wie wir es benannt haben:

```
model.table(RI.results)
```

```
##           p  c f0                 model npar     AICc DeltaAICc      weight
## 2        ~1    ~1        p(~1)c()f0(~1)    2 71.59017  0.000000 0.465281385
## 5     ~Time    ~1     p(~Time)c()f0(~1)    3 72.78631  1.196139 0.255845317
## 3        ~1 ~1 ~1      p(~1)c(~1)f0(~1)    3 73.54142  1.951248 0.175391134
## 1       ~t2 ~1 ~1     p(~t2)c(~1)f0(~1)    4 75.32030  3.730129 0.072065673
## 4     ~time    ~1     p(~time)c()f0(~1)    8 77.61713  6.026955 0.022854886
## 6 ~time + c    ~1 p(~time + c)c()f0(~1)    9 79.58088  7.990710 0.008561604
##   Deviance
## 2 118.6213
## 5 117.8014
## 3 118.5565
## 1 118.3141
## 4 112.4714
## 6 112.3866
```

Auch eine Zusammenfassung der einzelnen Modelle können wir anschauen:

```
summary(RI.results[[2]])
```

```
## Output summary for Closed model
## Name : p(~1)c()f0(~1)
##
## Npar :  2
## -2lnL:  67.57424
## AICc :  71.59017
##
## Beta
##                 estimate        se       lcl        ucl
## p:(Intercept)  -1.170141 0.1009125 -1.367930 -0.9723524
## f0:(Intercept)  2.927100 0.3251139  2.289877  3.5643236
##
##
## Real Parameter p
##          1         2         3         4         5         6         7
##  0.2368295 0.2368295 0.2368295 0.2368295 0.2368295 0.2368295 0.2368295
##
##
## Real Parameter c
##          2         3         4         5         6         7
##  0.2368295 0.2368295 0.2368295 0.2368295 0.2368295 0.2368295
##
##
## Real Parameter f0
##        1
##  18.6734
```

```
summary(RI.results[['p.Time']])
```

```
## Output summary for Closed model
## Name : p(~Time)c()f0(~1)
```

```
##
## Npar :  3
## -2lnL:  66.7544
## AICc :  72.78631
##
## Beta
##                   estimate        se        lcl        ucl
## p:(Intercept)  -1.0807509 0.1409077 -1.3569300 -0.8045718
## p:Time         -0.0409687 0.0451870 -0.1295352  0.0475978
## f0:(Intercept)  2.9886306 0.3246001  2.3524145  3.6248468
##
##
## Real Parameter p
##          1         2         3         4         5         6         7
##  0.2533639 0.2456925 0.2381792 0.2308253 0.2236319 0.2165995 0.2097285
##
##
## Real Parameter c
##          2         3         4         5         6         7
##  0.2533639 0.2456925 0.2381792 0.2308253 0.2236319 0.2165995
##
##
## Real Parameter f0
##         1
##  19.85847
```

Auf die Populationsgröße kommt man schließlich folgendermaßen:

```
RI.results[['p.Time']]$results$derived
```

```
## $`N Population Size`
##   estimate       se     lcl      ucl
## 1 127.8585 6.446061 118.679 144.9284
```

In diesem Modell würde die Populationsgröße auf 128 geschätzt werden, mit einem 95% Konfidenzintervall von 119 bis 145.

Schauen wir uns das M0 parallel an:

```
RI.results[['p.dot']]$results$derived
```

```
## $`N Population Size`
##   estimate       se      lcl      ucl
## 1 126.6734 6.070984 118.0324 142.7572
```

Wir erhalten ganz ähnliche Schätzungen, alle unterschätzen die von uns oben bestimmte Populationsgröße, da sie die individuelle Heterogenität nicht beachten. Dafür kann man in RMark die Modelle von Pledger (2000) oder Huggins (1989, 1991) nutzen. Dies machen wir hier noch einmal mit einer wrapper Funktion und vergleichen nochmals mit dem M0 und Mt Modell.

Individuelle Heterogenität hinzufügen

```
run.RI.het=function(){
  # Parameter und Modelle definieren
  p.dotshared=list(formula=~1,share=TRUE)
  p.timeshared=list(formula=~time,share=TRUE)
  p.time.c=list(formula=~time+c,share=TRUE)
  p.timemixtureshared=list(formula=~time+mixture,share=TRUE)
  p.mixture=list(formula=~mixture)

  # Ausgewählte Modelle laufen lassen
  # Standard Closed Modelle
  # konstant p=c
  RI.closed.m0 = mark(ch.RI2, model="Closed",
                      model.parameters=list(p=p.dotshared),delete=TRUE)
  # p konstant, c konstant, aber verschieden
  RI.closed.m0c = mark(ch.RI2, model="Closed",delete=TRUE)
  # zeitlich verschiedene p=c
  RI.closed.mt = mark(ch.RI2,model="Closed",
                      model.parameters=list(p=p.timeshared),delete=TRUE)

  # HetClosed Modelle (Mixtures)
  # 2 mixtures Mh2
  RI.closed.Mh2 = mark(ch.RI2, model="HetClosed",
                       model.parameters=list(p=p.mixture),delete=TRUE)
  # Mth2 - p zeitlich verschieden; 2 mixture (additiv)
  RI.closed.Mth2.additive = mark(ch.RI2, model="FullHet",
                                 model.parameters=list(p=p.timemixtureshared),
                                 adjust=TRUE,delete=TRUE)

  # Huggins Modelle
  # p=c zeitlich konstant
  RI.huggins.m0 = mark(ch.RI2, model="Huggins",
                       model.parameters=list(p=p.dotshared),delete=TRUE)
  # p konstant, c konstant, aber verschieden; Standardmodell für Huggins
  RI.huggins.m0.c = mark(ch.RI2,model="Huggins",delete=TRUE)
  # Huggins Mt
  RI.huggins.Mt = mark(ch.RI2,model="Huggins",
                       model.parameters=list(p=p.timeshared),
                       adjust=TRUE,delete=TRUE)

  # Huggins Modelle für individuelle Heterogenität
  # Mh2 - p verschieden für mixtures
  RI.huggins.Mh2 = mark(ch.RI2,model="HugHet",
                        model.parameters=list(p=p.mixture),delete=TRUE)
  # Huggins Mth2 - p zeitlich verschieden; mixture additiv
  RI.huggins.Mth2.additive = mark(ch.RI2,model="HugFullHet",
                                  model.parameters=list(p=p.timemixtureshared),
                                  adjust=TRUE,delete=TRUE)

  # Modelltabelle ausgeben
  return(collect.models() )
}

# Modelle aufrufen
```

```
RI.results.het = run.RI.het()
```

Schauen wir uns diese Ergebnisse an:

```
RI.results.het
```

```
##                                 model npar      AICc  DeltaAICc      weight
## 3            pi(~1)p(~mixture)f0(~1)    4  68.58580   0.000000 0.729613874
## 1                     p(~1)c()f0(~1)    2  71.59017   3.004375 0.162443132
## 2                   p(~1)c(~1)f0(~1)    3  73.54142   4.955623 0.061234096
## 5  pi(~1)p(~time + mixture)c()f0(~1)   10  74.45762   5.871823 0.038729598
## 4                   p(~time)c()f0(~1)    8  77.61713   9.031330 0.007979299
## 8                   pi(~1)p(~mixture)    3 862.36209 793.776299 0.000000000
## 6                           p(~1)c()    1 866.80846 798.222659 0.000000000
## 10       pi(~1)p(~time + mixture)c()    9 868.21709 799.631291 0.000000000
## 7                         p(~1)c(~1)    2 868.81111 800.225310 0.000000000
## 9                        p(~time)c()    7 872.81402 804.228226 0.000000000
##    Deviance
## 3  111.5796
## 1  118.6213
## 2  118.5565
## 5  105.2094
## 4  112.4714
## 8  907.3772
## 6  915.8502
## 10 901.0228
## 7  915.8422
## 9  909.7113
```

Auch hier ist wieder ein klassisches Mh Modell am besten. Schauen wir an, welches Modell das war (model.table funktioniert nicht, wenn verschiedene Modelltypen aufgerufen werden).

```
names(RI.results.het)
```

```
##  [1] "RI.closed.m0"             "RI.closed.m0c"
##  [3] "RI.closed.Mh2"            "RI.closed.mt"
##  [5] "RI.closed.Mth2.additive"  "RI.huggins.m0"
##  [7] "RI.huggins.m0.c"          "RI.huggins.Mh2"
##  [9] "RI.huggins.Mt"            "RI.huggins.Mth2.additive"
## [11] "model.table"
```

Das Modell RI.closed.Mh2, also Pledgers Mixture Modell, hat am besten abgeschnitten. Schauen wir uns die Populationsgrößenschätzung an:

```
RI.results.het[['RI.closed.Mh2']]$results$derived
```

```
## $`N Population Size`
##   estimate       se      lcl      ucl
## 1 205.3179 152.4339 118.9788 970.6389
```

Die geschätzte Populationsgröße liegt bei 205 und mit 95% Wahrscheinlichkeit zwischen 119 und 971. Damit hat Pledgers Mixed Modell erneut ein sehr breites Konfidenzintervall und neigt womöglich zu einer leichten Überschätzung der tatsächlichen Populationsgröße (vgl. Grimm et al. 2014).

Literaturverzeichnis

Burnham, K.P., Overton, W.S. 1978. Estimating the size of a closed population when capture probabilities vary among animals. Biometrika 65: 625–633.

Chao, A. 1987. Estimating the population size for capture–recapture data with unequal catchability. Biometrics 43: 783–791.

Chao, A. 2015. Capture-recapture for human populations. Wiley StatsRef: Statistics Reference Online, https://drive.google.com/file/d/1f7RoM8mkxa2HFmk0v1L_zfYySOWkG-Gj/view.

Chao, A., Tsay, P.K. 1998. A sample coverage approach to multiple-system estimation with application to census undercount. Journal of the American Statistical Association 93: 283–293.

Chao, A., Yang, H.-C. 2006. User guide for program CARE-2. https://drive.google.com/file/d/1VyJfK4N3syYBk-WY411m84WKvnu2OuW4/view.

Chao, A., Tsay, P.K., Lin, S.H., et al. 2001. The applications of capture-recapture models to epidemiological data. Statistics in Medicine 20: 3123–3157.

Darroch, S.E., Fienberg, G., Glonek, B. and Junker, B. 1993 A three sample multiple capture-recapture approach to the census population estimation with heterogeneous catchability. Journal of the American Statistical Association 88: 1137–1148.

Dorazio, R.M., Royle, J. A. 2003. Mixture models for estimating the size of a closed population when capture rates vary among individuals. Biometrics 59: 351–364.

Efford, M.G. 2025. secr: Spatially explicit capture-recapture models. R package version 5.2.1. https://CRAN.R-project.org/package=secr

Grimm, A., Gruber, B., Henle, K. 2014. Reliability of different mark-recapture methods for population size estimation tested against reference population sizes constructed from field data. Plos One 9: e98840.

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

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

Hsieh, T. 2012. CARE1: Statistical package for population size estimation in capture-recapture models. R package version 1.1.0, https://CRAN.R-project.org/package=CARE1.

Huggins, R.M. 1989. On the statistical analysis of capture-recapture experiments. Biometrika 76: 133-140.

Huggins, R.M. 1991. Some practical aspects of a conditional likelihood approach to capture experiments. Biometrics 47: 725-732.

Laake, J.L. 2013. RMark: An R Interface for Analysis of capture-recapture data with MARK. AFSC Processed Rep 2013-01, 25p. Alaska Fish. Sci. Cent., NOAA, Natl. Mar. Fish. Serv., 7600 Sand Point Way NE, Seattle WA 98115.

Lee, S.-M., Chao, A. 1994. Estimating population size via sample coverage for closed capture-recapture models. Biometrics 50: 88–97.

Otis, D.L., Burnham, K.P., White, G.C., Anderson, D.R. 1978. Statistical inference from capture data on closed animal populations. Wildlife Monographs 62: 1–135.

Pledger, S. 2000. Unified maximum likelihood estimates for closed capture-recapture models using mixtures. Biometrics 56: 434–442.

Rivest, L.P. and Baillargeon, S. 2007. Applications and extensions of Chao's moment estimator for the size of a closed population. Biometrics 63(4): 999–1006.

Rivest, L., Baillargeon, S. 2022. Rcapture: Loglinear Models for Capture-Recapture Experiments. R package version 1.4-4, https://CRAN.R-project.org/package=Rcapture.