P2 - Simulering af Data

Gruppe B2-19

2020-05-20

Contents

Chapter 1

Introduktion

1.1 Forord

Projektet er udarbejdet af gruppe B2-19 fra Datavidenskab 2. semester på Aalborg Universitet. Projketperioden løb fra den 3. februar 2020 til den 27. maj 2020. Gruppen blev vejledt af Mikkel Meyer Andersen. Det overordnede tema i projektet er "Fra Data til Videnskab" med fokus på statistisk inferens ved hjælp af simuleringer. Projektets kildekode kan findes på følgende link til GitHub.

1.2 Synopsis

Projektet har haft til formål at afdække, hvilke alternativer der er til klassiske inferensmetoder, hvis disses antagelser ikke er overholdt.

Først beskrives en klassisk inferensmetode, t-testen, og hvordan denne kan bruges til hypotesetests. Dernæst undersøges, hvordan tilsyneladende tilfældige tal kan genereres ved hjælp af algoritmer og hvordan programmeringssproget R kan udnyttes, til at udføre disse simuleringer hurtigt og simpelt. Herefter undersøges det, ved hjælp af simuleringer, om resultaterne er troværdige, hvis stikprøverne ikke er normalfordelte, når der arbejdes med en uparret t-test. Bagefter redegøres der for to alternativer til t-testen der ikke antager normalfordelte stikprøver, permutationer og bootstrap.

Der blev fundet frem til at det ikke kan antages, at en t-test på en ikke-normalfordelt stikprøve altid giver retvisende resultater. Desuden blev det fundet frem til, at permutationstesten er mere præcis end bootstrap, hvis der udelukkende skal foretages en hypotesetest. Derudover blev det konkluderet, at, ud af de tre undersøgte bootstrap-metoder til konfidensintervaller, er dækningsgraden af T-metoden den mest præcis.

1.3 Abstract

The aim of the project was to identify alternatives to classical inference methods if the assumptions of these were not met.

First, a classical inference method, the t-test, and how it can be used for hypothesis testing, is described. Next, it is examined how seemingly random numbers can be generated by algorithms, and how the programming language R can be utilized to perform these simulations quickly and simply. Then, by means of simulations, it is examined whether the results are credible if the samples are not normally distributed when working with an unpaired t-test. Afterwards, two alternatives to the t-test that do not assume normally distributed samples, permutations and bootstrap, are presented.

It was discovered that it cannot be assumed that a t-test on a non-normal random sample always produces true results. In addition, it was found that the permutation test is more accurate than bootstrap if only a hypothesis test is to be performed. Furthermore, it was concluded that, out of the three bootstrap methods examined for confidence intervals, the coverage of the T method is the most accurate.

1.4 Anvendte pakker

I dette projekt er følgende pakker blevet anvendt.

1) `bookdown`, [Xie, 2020]. Denne pakke muliggør at skrive hele rapporter i R ved hjælp af Markdown, og er derfor brugt til udarbejdelsen af hele rapporten.

2) `mosaic`, [Pruim et al., 2020]. Denne pakke indeholder flere funktionaliteter, der benyttes i statistike analyser. Pakken anvendes i samtlige afsnit i rapporten.

3) `wPerm`, [Weiss, 2015]. Denne pakke indeholder en funktion, der kan udføre hypotesetest ved hjælp af permutationstest. Pakken anvendes i afsnit 5.3.

4) `boot`, [Canty and Ripley, 2020]. Denne pakke indeholder funktioner, der benyttes til at oprette bootstrap-stikprøver og til at udarbejde bootstrap-konfidensintervaller. Pakken anvendes i afsnit 6.2 og 6.4.4.

1.5 Læsevejledning

- `pakke::funktion`.
- Ikke-parametrisk bootstrap.
- Samme seed, `31415`, i alle rapportens afsnit.
- Engelsk version af decimal- og tusindtalsseperator
- Antal bootstraps / permutationer // reps kontra computerkraft

Chapter 2

Problemanalyse

2.1 Statistikkens udvikling

Ordet *statistik* stammer tilbage fra det latinske *statisticum collegium* ("statsrådgiver") og det italienske *statista* ("statsmand" eller "politiker"), [Dictionary, 2020]. At statistik netop stammer derfra, giver god mening, under anskueelse af betydningen af disse to ord og den tidlige anvendelse af statistik. I takt med udviklingen af suverænitetsstaterne, steg behovet for at registrere befolkningen og dennes tilhørselsforhold. Derfor anvendte statsrådgivere og statsmænd statistik til at beskrive staten, særligt demografien. Senerere blev dette udvidet til at indsamle flere informationer, og ligeledes at analysere og fortolke disse ved hjælp af statistik, [Danske, 2020].

To statistikere, som ligger fundamentet til den statistiske arbejdsform, der bruges i dag er Karl Pearson og Ronald Fisher. Karl Pearson var interesseret i at udvikle matematiske metoder til at studere biologisk arv og evolution. Gennem den interesse udsprang en række bidrag til statistiske analyse metoder. Pearson opfandt korrelationskoefficienten (R^2) som bruges til at vise, hvor godt en regressionsmodel passer noget givent data. Udover det, opfandt han også χ^2-testen, som er en metode der bruges til at teste at observerede værdier stemmer overens med de forventede værdier, [Porter, 2020], [Matematikcenter, 2020].

Ronald Fisher introducerede princippet om randomisering. Det siger, at et eksperiment skal gentages på et antal kontrolgrupper, og de elementer der bruges i eksperimentet skal tilfældigt udvælges fra hele populationen. Dette gjorde data forventningsret, som mindsker variationen i et eksperiment. Udover det har Fisher introduceret analyse af varians, også kaldet ANOVA (fra engelsk analysis of variance). Denne model bruges til at analysere forskellen mellem gruppemiddelværdier i en stikprøve, [of Encyclopaedia Britannica, 2020].

Yderligere har udviklingen af computeren været med til at gøre anvendelsen af komplicerede statistiske beregninger hurtigere, mere præcise og mere tilgængelige. Anvendelsesområderne for statistik har ligeledes udviklet sig, fra i begyndelsen at være noget staten anvendte til styring af økonomi og befolkningsindblik, til stort set at være repræsenteret i alle større hverv i dag. Den moderne definition af statistik kan beskrives som evnen til at drage konklusioner om generelle tilfælde, *populationer*, på baggrund af enkelte tilfælde, *stikprøver*, [Agresti and Finlay, 2014, s. 4].

2.2 Statistik

Statistik opdeles overordnet i to kategorier, deskriptiv statistisk og statistisk inferens.

Deskriptiv statistik

Det følgende afsnit er baseret på [AAU, 2020a] og [Agresti and Finlay, 2014, s. 4-5]

Deskriptiv statistik drejer sig om at opsummere data, således at informationerne heri anskueliggøres og forståelsen af data styrkes, uden at fordreje eller miste den oprindelige information. Dette kan eksempelvis gøres ved at beregne middelværdien og spredningen, da det giver et bedre overblik, end at skulle overskue samtlige datapunkter. I nogle tilfælde kan det være tilstrækkeligt at lave en grafisk fremstilling af data, eventuelt i kombination med middelværdi og spredning.

Statistisk inferens

Det følgende afsnit er baseret på [AAU, 2020b], [AAU, 2020c] og [Agresti and Finlay, 2014, s. 4-5].

På baggrund af informationen, som udtrykkes af deskriptiv statistik, er det således muligt ved hjælp af statistisk inferens at kommme med antagelser og drage konklusioner.

I statistisk inferens differentieres der mellem to metoder, estimering og hypotesetest. Når der estimeres på baggrund af en population, bruges stikprøven til at beskrive en ukendt del af populationen. Det kan eksempelvis være gennemsnitsindkomst hos en befolkning, μ, for hvilken der findes et estimat $\hat{\mu}$, som bruges til at beskrive μ. Dette betegnes som et punktestimat, og vil oftest suppleres med et intervalestimat. Årsagen til dette er, at punktestimater er tilfældige, og derfor ændrer sig fra stikprøve til stikprøve. Da punktestimaters sandsynlighed for at være korrekt derfor er lig 0, tilstræbes det at anvende et intervalestimat, hvor det kan siges at μ med 95% sikkerhed ligger, fremfor et punktestimat. Dette kaldes for et konfidensinterval.

Den anden form for statistisk infernes er hypotesetest, som beskrives i det efterfølgende afsnit.

2.3 Hypotesetest

I det følgende afsnit introduceres hypotesetests, samt hvorledes disse anvendes til at drage konklusioner for en population, ved at opstille to hypoteser.

En hypotesetest baserer sig på det videnskabelige princip om falsificering. Der opstilles en indledende formodning om en population, kaldet nulhypotesen H_0, og en alternativ, modsat hypotese H_1. Er den indledende formodning ikke korrekt, må den alternative hypotese være gældende.

Ved en hypotesetest undersøges, hvorvidt der er en difference mellem observerede værdier og forventede værdier, hvis H_0 er sand.

Sandsynligheden for, at der er en difference, er stor, eftersom der arbejdes på en stikprøve og ikke selve populationen, og derfor benyttes et mål for, hvornår differencen er *for* stor, kaldet signifikansniveauet, α.

Når nulhypotesen og den alternative hypotese er blevet opstillet, kan stikprøvens resultat sammenlignes med det forventede resultat under nulhypotesen, ved hjælp af en teststørrelse. Teststørrelsen kan blandt andet bestemmes som antallet af standardafvigelser, den observerede værdi, $\hat{\theta}$, ligger fra den forventede værdi θ_0, i retning af den alternative hypotese, [Editor, 2015].

At $\hat{\theta}$ ligger mere end 3 standardafvigelser fra θ_0, er højst usandsynligt, da $\hat{\theta}$ i så fald er en outlier i populationen. I et sådan tilfælde er θ_0 højst sandsynligt ikke populationens korrekte værdi, hvorved nulhypotesen kan forkastes.

Derudover benyttes testtørrelsen til at udregne p-værdien, som er sandsynligheden for at få en teststørrelse, der er lige så eller mere ekstrem, hvis H_0 er sand.

Værdien af teststørrelsen påvirker p-værdien på den måde, at når teststørrelsen bliver mere ekstrem, falder p-værdien. Jo mindre p-værdien er, des mindre stoles på H_0, og hvis p-værdien er mindre end signifikansniveauet, α, forkastes H_0. Er p-værdien derimod større end α, er der ikke belæg for at forkaste H_0 - dette betyder dog ikke, at H_0 givetvis er sand.

En illustration af teststørrelsens betydning ved en normalfordeling kan ses på Figur 2.1.

FIXME: (Mere forklaring til denne figur)

Normalt arbejdes der med et signifikansniveau på 5%, $\alpha = 0.05$. Dog er der intet fast signifikansniveau og det kunne lige såvel være 10% eller 1%. Betydningen heraf diskuteres kort sidst i afsnittet under fejltyper, [Team, 2020].

2.3.1 Hypotesetest for middelværdier

I dette afsnit gennemgås fremgangsmåden for, hvordan en hypotesetest kan bruges til at bestemme middelværdien for en population. I dette afsnit kaldes en sådan hypotesetest for en t-test. Afsnittet er skrevet på baggrund af [AAU, 2020d].

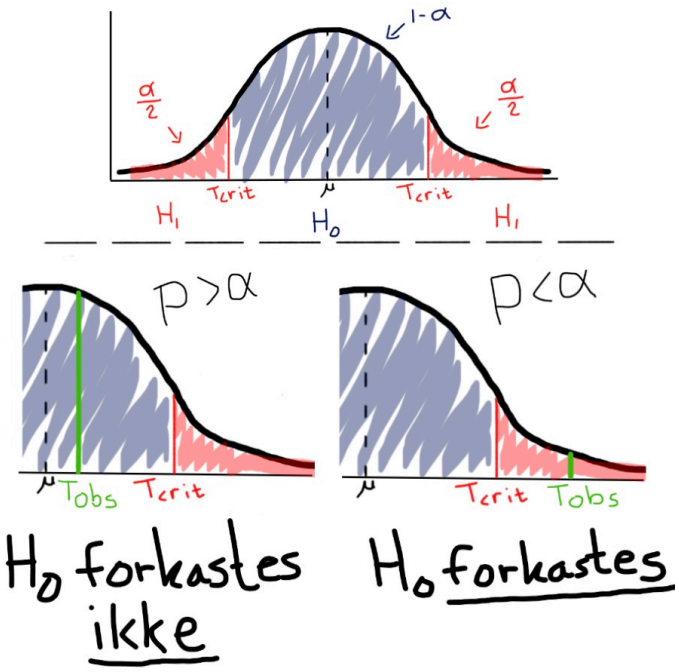

Figure 2.1: Teststørrelsens indflydelse på nulhypotesen

Forud for gennemførelsen af en t-test, er der nogle antagelser, som skal være opfyldt, for at t-testen ikke giver misvisende resultater.

1. Stikprøven skal være repræsentativ for populationen.
2. Variablen skal være kvantitativ.
3. Stikprøveudtagning skal være udført med tilfældighed.
4. Populationen skal være normalfordelt.

Herefter er fremgangsmetoden som beskrevet i afsnit 2.3.

Eksempel

Der vil nu vises et eksempel på en t-test. Figur 2.2 viser en stikprøve af `10` observationer med en middelværdi på `0.0371563`, udtaget fra en standard normalfordelt population.

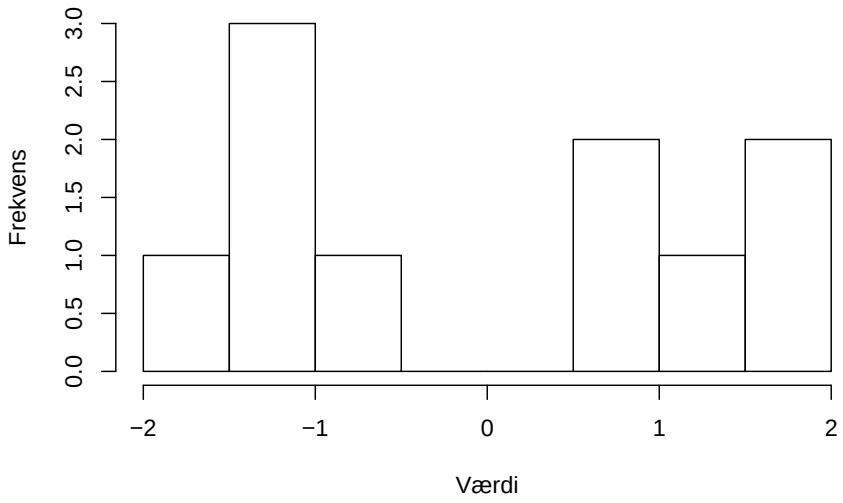

Figure 2.2: Histogram over 10 simulerede standard normalfordelte tal.

I kodestykket nedenfor gennemgås den beskrevne fremgangsmåde for en t-test. I dette eksempel er $H_0 : \mu = 0$, og $H_1 : \mu \neq 0$.

```
forventet_middelvaerdi <- 0

stik <- rnorm(n = 10, mean = 0, sd = 1)
middelvaerdi <- mean(stik)
standardafvigelse <- sd(stik)
```

```
estimeret_standardfejl <- standardafvigelse/sqrt(10)

t_obs <- (abs(middelvaerdi-forventet_middelvaerdi))/
            estimeret_standardfejl

p_ensidet <-  1 - pdist("t", q = t_obs, df = 10-1, plot = FALSE)

p_tosidet <- 2 * p_ensidet
p_tosidet
```

```
## [1] 0.941182
```

Eftersom p-værdien er `0.941182` $> \alpha = 0.05$, forkastes H_0 ikke. Havde p-værdien derimod været mindre end 0.05 ville det medføre, at H_0 forkastes og det ville formodes, at H_0 ikke er korrekt for populationen.

2.3.2 Fejltyper

Der er risiko for to primære fejl når der foretages en hypotesetest, som illustreret på figur 2.3. Den første, type-I fejl, er hvor H_0 forkastes, men i virkeligheden er sand, og den anden, type-II fejl, er hvor H_0 accepteres, men i realiteten er falsk.

En af de primære årsager til disse typer fejl er, hvor signifikansniveauet bliver sat, da dette i nogle tilfælde har stor betydning for, hvorvidt en hypotese bliver forkastet eller ej.

Sandsynligheden for type-I fejl er lig med det valgte signifikansniveau, α. Sandsynligheden for type-II fejl, β, er derimod ikke let at præcisere. Dog er der stor sandsynlighed for type-II fejl, hvis den virkelige sandhed er tæt på nulhypotesen. Er den virkelige sandhed derimod langt fra nulhypotesen, vil sandsynligheden for type-II fejl være tilsvarende lille. Ligeledes har stikprøvens størrelse indflydelse, eftersom meget data mindsker risikoen for type-II fejl, hvor der er større risiko ved en mindre mængde data. Andelen af type-II fejl benyttes til at udregne hypotesetestens styrke, $1-\beta$. Styrken beskriver andelen af gange, hvor H_1 korrekt bliver forkastet, [Frydenberg, 2001].

Det vil påvises, hvilken betydning små differencer i middelværdien samt stikprøvestørrelsen har for andelen af type-II fejl.

```
##                          Stikprøvestørrelsen, n =
## Differencen i middelværdi     5    10    25    50   200  1000 10000
##                     0.001 0.963 0.945 0.952 0.951 0.956 0.967 0.948
##                     0.1   0.955 0.955 0.934 0.938 0.831 0.403 0.000
##                     1     0.719 0.426 0.079 0.003 0.000 0.000 0.000
##                     2     0.248 0.013 0.000 0.000 0.000 0.000 0.000
##                     5     0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

		Virkelighed	
		H_0	H_1
Konklusion	H_0	Korrekt	Type-II fejl
	H_1	Type-I fejl	Korrekt

Figure 2.3: Tabel over fejltyper

Tabellen viser, at både stikprøvestørrelsen samt differencen i middelværdien, har stor betydning for andelen af af type-II fejl. Er differencen i middelværdierne 0.001 fremgår det, at stikprøvestørrelsen mellem 5 og $10,000$ ikke giver signifikante forskelle, hvilket medfører, at stikprøvestørrelsen skal være meget større for at mindske andelen af type-II fejl. Derudover mindskes risikoen for type-II fejl som stikprøvestørrelsen øges, hvilket også er gældende hvis differencen i middelværdierne forøges.

2.4 Problemformulering

1) Hvorledes kan simuleringsstudier bidrage til at afdække problemer med klassiske inferensmetoder, når deres antagelser ikke er overholdt?

2) Hvilke alternativer er der til klassiske inferensmetoder, såsom t-tests og konfidensintervaller, når deres antagelser ikke er overholdt, og hvor robuste er alternativerne?

Chapter 3

Simulering

I dette afsnit, vil der redegøres for lidt baggrundsviden omkring simulering, og hvordan tilsyneladende tilfældige tal kan genereres ved hjælp af algoritmer. Derefter vil der ses på, hvordan programmeringssproget R kan udnyttes, til at udføre disse simuleringer hurtigt og simpelt.

En definition på simulering er;

> A situation or event that seems real but is not real, used especially in order to help people deal with such situations or events. - Cambridge Dictonary, [Dictonary, 2020].

Ud fra definitionen, er formålet altså ved simuleringer at efterligne virkeligheden, så de analyser der gøres på baggrund af simuleringerne, kan bruges i virkeligheden når lignende situationer opstår.

Brancher hvor simuleringer er et yderst vigtigt redskab, er i motorsporten. I Formel 1 benytter holdene sig af simulatorer, hvor de genskaber bilerne og derved kan teste nye dele inden de producerer dem i virkeligheden for at spare ressourcer, [F1, 2020].

3.1 Pseudorandom number generator

I dette afsnit introduceres pseudo-tilfældige talgeneratorer, PRNG'er (fra engelsk *pseudorandom number generators*), og hvorledes disse kan bringes i anvendelse i forbindelse med simuleringer. I de tilhørende underafsnit beskrives først en type af PRNG kaldet lineær kongruens generator, og dernæst en metode til at omdanne uniformt fordelte talt til standard normalfordelte tal kaldet Box-Muller-transformation.

En computer fungerer ved, at den modtager et input, der bearbejdes af en algoritme, som derefter returnerer et output. Der findes ingen algoritmer, som

er i stand til at generere faktisk tilfældige tal. Grunden til dette er, hvis der gives det samme input, til den samme algoritme, vil resultatet være det samme output som tidligere, fordi en computer fungerer på baggrund af matematik og derfor er deterministisk. Det er dog muligt ved hjælp af en beregningsmodel at skabe en illusion af ægte tilfældighed. En sådan model kaldes for en PRNG.

En PRNG beskrives ved nedenstående karakteristika, [Oxford, 2015].

1. Et input, eller *seed*, på baggrund af hvilket algoritmen beregner et pseudo-tilfældigt output. Herefter fortsætter algoritmen rekursivt, hvor det forrige output anvendes som nyt input.

2. Cyklussen, som beskriver den talserie, PRNG'en beregner.

3. Perioden, som beskriver hvor mange repetitioner algoritmen gennemløber, før cyklussen gentages. Jo kortere perioden er, des mere gennemskuelig vil algoritmen fremstå.

4. Fordelingen af cyklussen. Som det ses på figur 3.1, kan fordelingen af cyklussen være jævn, hvilket viser en ligelig fordeling af cyklussen. Ses der derimod et mønster eller tendens i fordelingen, vil algoritmen ikke være forventningsret, og anvendeligheden formindskes.

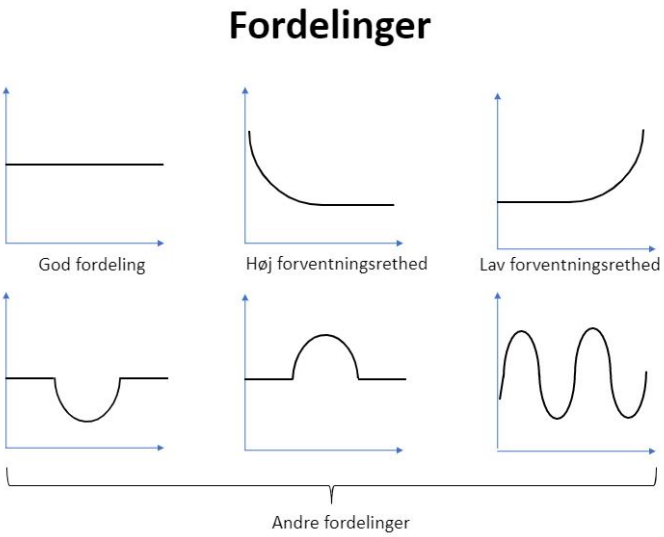

Figure 3.1: PRNG fordelinger

I takt med udviklingen af de teknologiske hjælpemidler er der opstået mere effektive algoritmer til PRNG. En af de mere kendte og hyppigt anvendte algoritmer er lineær kongruens generator, som gennemgåes i det følgnede afsnit.

3.1.1 Lineær kongruens generator

Dette afsnit beskriver, hvorledes lineær kongruens generatorer, LCG (fra engelsk *linear congruential generator*), kan generere en cyklus af tilsyneladende tilfældige tal, som er uniformt fordelt, ved valg af passende parametre. Det følgende afsnit er primært baseret på kilden [Radziwill, 2015].

LCG danner en cyklus af tal ved iteration, og kræver kun få parametre. Helt specifikt er algoritmen givet ved

> **Definition 3.1.** Givet $a, c, m \in \mathbb{Z}$, hvor $0 < m, 0 < a$ og $0 \leq c \leq m$, genererer algoritmen lineær kongruens generator en talsekvens, $X = [X_0, X_1, \dots, X_n]$, givet ved $X_i = (a \cdot X_{i-1} + c) \mod m$.

Algoritmen fungerer således: Der startes ved et *seed*, angivet med X_0, som bliver multipliceret med faktoren, a, og konstantleddet, c, adderes. Derefter tages modulus m af værdien.

I figuren nedenfor kan ses tre eksempler på LCG. De to første eksempler har samme parametre, men forskelligt *seed*.

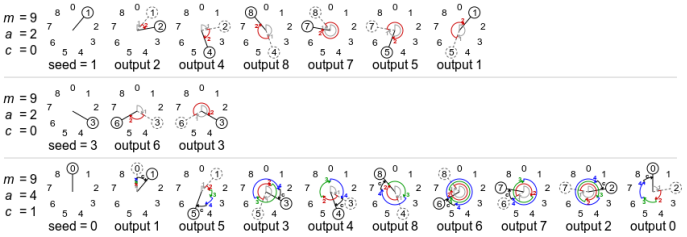

Figure 3.2: Tre eksempler på algoritmen lineær kongruens generator. De to første eksempler har samme parametre, men forskelligt seed. [@lincongenvis]

I teorien er parametrene arbitrære, men i praksis bruges nogle værdier oftere end andre. Begrundelsen for dette er, at der findes værdier for parametrene, der vil returnere en åbentlyst ikke-tilfældig cyklus. Et eksempel på dette er ved parametrene $m = 64, a = 33$ og $c = 12$. Efter et antal iterationer vil der ses et mønster i værdierne, og det vil være åbenlyst, at denne cyklus i virkeligheden ikke er tilfældig. Dette eftervises i det følgende eksempel.

Eksempel

De følgende simuleringer er baseret på kilden [Trumbo, 2005], som også går mere i dybden med hvordan, der undersøges om de givne parametre genererer tilsyneladende tilfældige cyklusser.

Først defineres en funktion for LCG-algoritmen, som tager de nødvendige parametre for algoritmen som inputs. Derudover angives også et argument, n, der fortæller hvor mange iterationer algoritmen skal foretage.

```
lcg <- function(modulus, faktor, tilvaekst, seed, iterationer) {

  r <- c()
  r[1] <- seed

  for (i in 1: (iterationer-1)) {
  r[i + 1] <- (faktor * r[i] + tilvaekst) %% modulus

}
 return(r)
}
```

Det følgende er et eksempel på en LCG-cyklus med en periode på 16.

```
lcg1 <- lcg(modulus = 64, faktor = 33, tilvaekst = 12, seed = 57,
            iterationer = 20)
lcg1
```

```
##  [1] 57 37 17 61 41 21  1 45 25  5 49 29  9 53 33 13 57 37 17 61
```

På figur 3.3 plottes denne cyklus. Det er åbenlyst at cyklussen ikke er tilfældig, da den danner et synligt mønster. Selv, hvis antallet af gange, LCG køres igennem, ændres, vil mønstret ikke ændres.

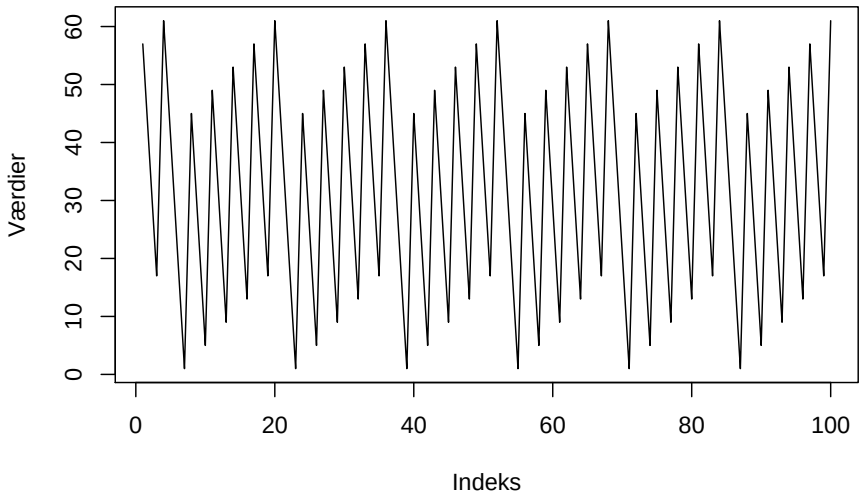

Figure 3.3: Et eksempel på en ikke-tilfældig LCG-cyklus.

På figur 3.4 vises et eksempel på, hvordan LCG-algoritmen kan bruges til at lave en uniform fordeling i intervallet $[0, 1]$. Når cyklussen er oprettet, deles hvert element deri med den valgte modulus, hvilket gør, at hvert element bliver indeholdt i $[0, 1]$.

```
lcg2 <- lcg(modulus = 86436, faktor = 1093, tilvaekst = 0,
            seed = 12, iterationer = 1000)

lcg2_uniform <- lcg2/86436
```

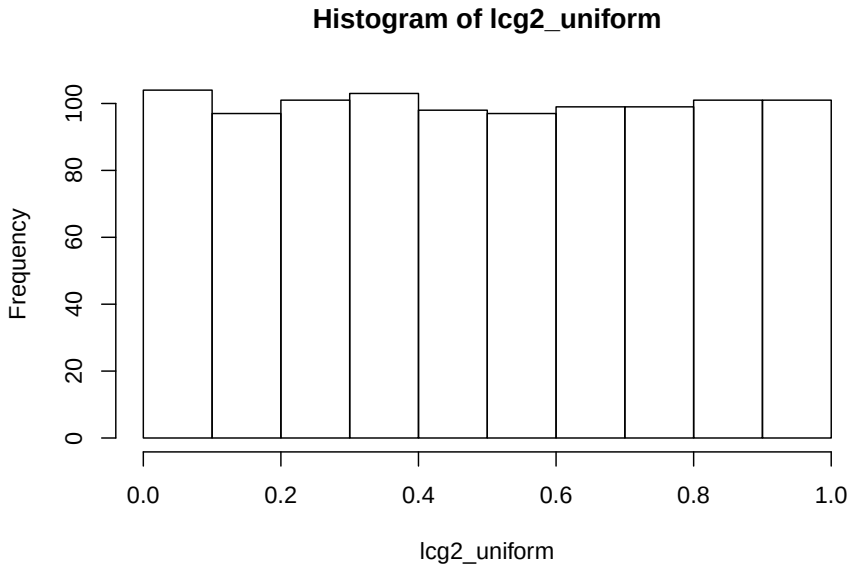

Figure 3.4: Et eksempel på en LCG-cyklus, der er uniformt fordelt.

Her kan det også nævnes, at hvis den cyklus, algoritmen returnerer, plottes i et 2d-punktplot, som på figur 3.5, vil der fremgå en mere tilfældig fordeling, dog vil der stadig kunne ses et mønster.

Afslutningsvist vil der vises et eksempel, hvor der returneres en cyklus, der ser tilfældig ud. Denne cyklus kan ses på figur 3.6.

```
lcg3 <- lcg(modulus = 86436, faktor = 1093, tilvaekst = 18257,
            seed = 12, iteration = 100)
```

3.1.2 Box-Muller-transformation

I dette afsnit beskrives Box-Muller-transformation, hvilket er en metode til at generere standard normalfordelte tal ud fra uniformt fordelte tal. Dette gøres for

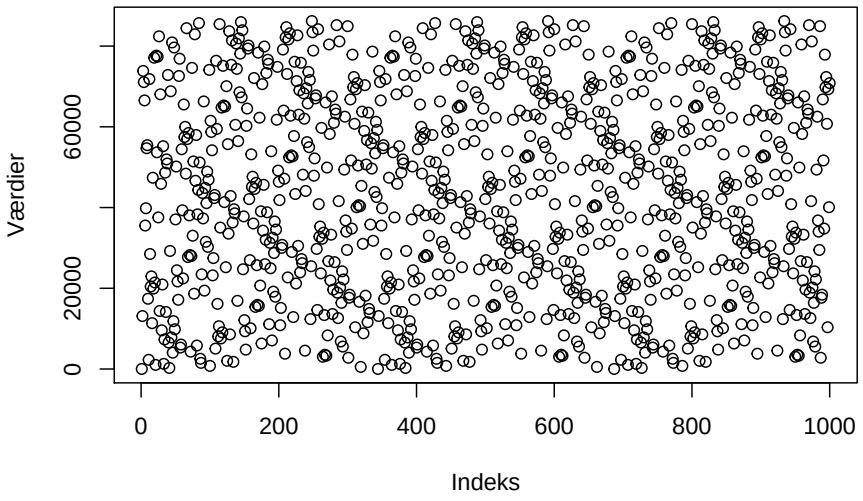

Figure 3.5: En 2d-graf af en LCG-cyklus, hvor der ses et svagt mønster.

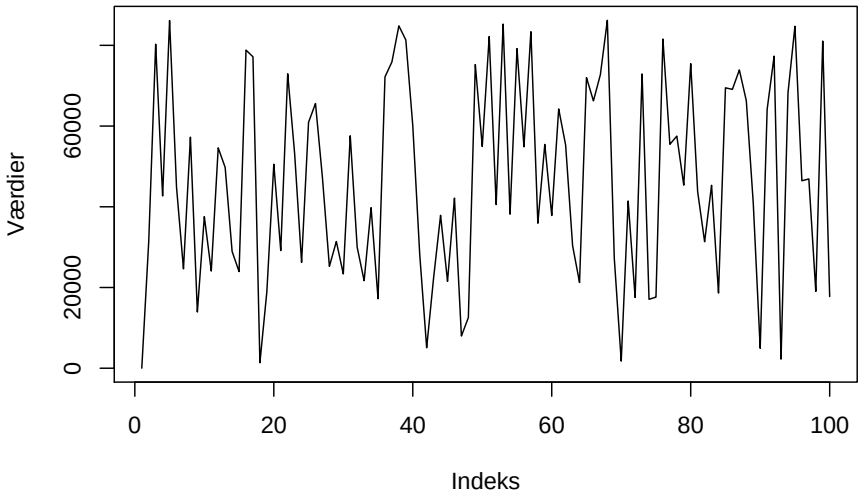

Figure 3.6: Et eksempel på en tilsyneladende tilfældig LCG-cyklus.

at belyse, hvordan en computer kan generere tilsyneladende tilfældige tal, der er normalfordelte. Afsnittet er skrevet med inspiration fra [Weisstein, 2020a]. Metoden beskrives konkret i nedenstående sætning.

Sætning 3.1. *Box-Muller-transformation*

Antag, at U_1 og U_2 er uafhængige stokastiske variabler, der begge er uniformt fordelt på intervallet $[0, 1]$. Lad

$$Z_1 = \sqrt{-2\ln U_1}\cos(2\pi U_2) \ \wedge \ Z_2 = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$

Så er Z_1 og Z_2 uafhængige stokastiske variabler, der er standard normalfordelte.

Eksempel

I dette afsnit oprettes to normalfordelinger ved hjælp af Box-Muller-transformationen i R.

Først simuleres to stokatiske variabler, `U1` og `U2`, hvor U1 $\sim$ unif$(0, 1)$ og U2 $\sim$ unif$(0, 1)$.

```
U1 <- runif(n = 100000, min = 0, max = 1)
U2 <- runif(n = 100000, min = 0, max = 1)
```

Disse to stokatiske variabler benyttes nu til at oprette de to påstået standard normalfordelte stokatiske variabler, `Z1` og `Z2`.

```
# Opretter Z1 vha. formlen for Box-Muller
Z1 <- sqrt(-2*log(U1))*cos(2*pi*U2)
# Opretter Z2 vha. formlen for Box-Muller
Z2 <- sqrt(-2*log(U1))*sin(2*pi*U2)
```

Efterprøvning i R

I det følgende afsnit efterprøves de forventede karakteristika af de to stokastiske variabler Z_1 og Z_2, der blev oprettet ved hjælp af Box-Muller-transformationen.

Først undersøges deskriptivt, hvilken fordeling `Z1` og `Z2` har, ved at oprette et boksplot af de to stokastiske variabler, som ses i figur 3.7.

Det ses på boksplottet, at der er en indikation på normalfordeling med henvisning til de fire krav et boksplot af en normalfordeling opfylder. De fire krav er:

1) Idet en normalfordeling er symmetrisk fordelt omkring dens middelværdi, er medianen og middelværdien lig hinanden.

2) Grundet normalfordelingens symmetri, er øvre og nedre kvartil lige langt fra medianen.

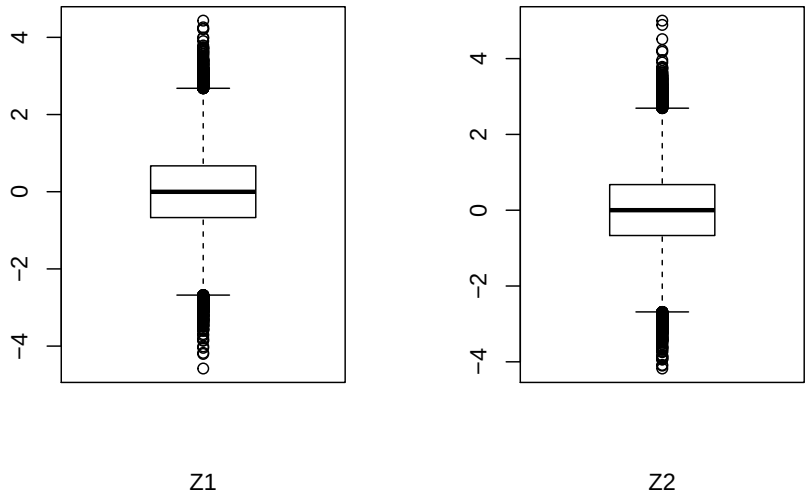

Figure 3.7: Et boksplot over Z1 og Z2

3) På grund af symmetrien i en normalfordeling, er der lige mange outliers over øvre kvartil som under nedre kvartil.

4) Da en outlier i et boksplot identificeres som data, der ligger mere end $1.5 \cdot \text{IQR}$ udover nedre og øvre kvartil, medfører det, at data, der ligger mere end 2.68 standardafvigelser fra middelværdien er outliers. Dermed følger, at 0.7% af observationerne ligger som outliers.

Punkterne 2 og 3 tjekkes ved at betragte boksplottet, hvor der ikke umiddelbart synes at være noget, der modbeviser en normalfordeling af `Z1` og `Z2`.

For at tjekke punkt 1, beregnes median (`median`) og middelværdi (`mean`) i kodestykket nedenfor.

```
mean_Z1 <- mean(Z1)
median_Z1 <- median(Z1)

mean_Z2 <- mean(Z2)
median_Z2 <- median(Z2)
```

Dette giver henholdsvis en middelværdi og median for `Z1` på $-7.9683876 \times 10^{-4}$ og -0.0023288, samt for `Z2` på 0.0014972 og 6.8522143×10^{-4}. De meget lave værdi stemmer overens med forventningen om, at `Z1` og `Z2` er standard normalfordelte.

For at tjekke punkt 4, beregnes andelen af outliers i hvert boksplot i kodestykket nedenfor.

```
OutVals1 <- boxplot(Z1, plot = FALSE)$out # Outliers i Z1
outliers_Z1 <- length(OutVals1) # Tæller antal outliers i Z1

OutVals2 <- boxplot(Z2, plot = FALSE)$out # Outliers i Z2
outliers_Z2 <- length(OutVals2) # Tæller antal outliers i Z2

# Andelen af outliers i Z1 og Z2
outliers_andel_Z1 <- outliers_Z1/length(Z1)
outliers_andel_Z2 <- outliers_Z2/length(Z2)
```

Dette giver en andel af outliers i `Z1` på `0.0074` og i `Z2` på `0.00745`. Dette svarer til cirka $0,7$ % outliers i både `Z1` og `Z2`, hvilket stemmer overens med punkt 4.

Der er altså ikke noget evidens imod, at `Z1` og `Z2` skulle være normalfordelt. Tværtimod underbygger beregningen af deres middelværdi, at de er *standard* normalfordelte.

Hernest kigges der på histogrammerne for `Z1` og `Z2`. På figur 3.8 sammenlignes histogrammerne for henholdsvis `Z1` og `Z2` med densiteten for standard normalfordeling.

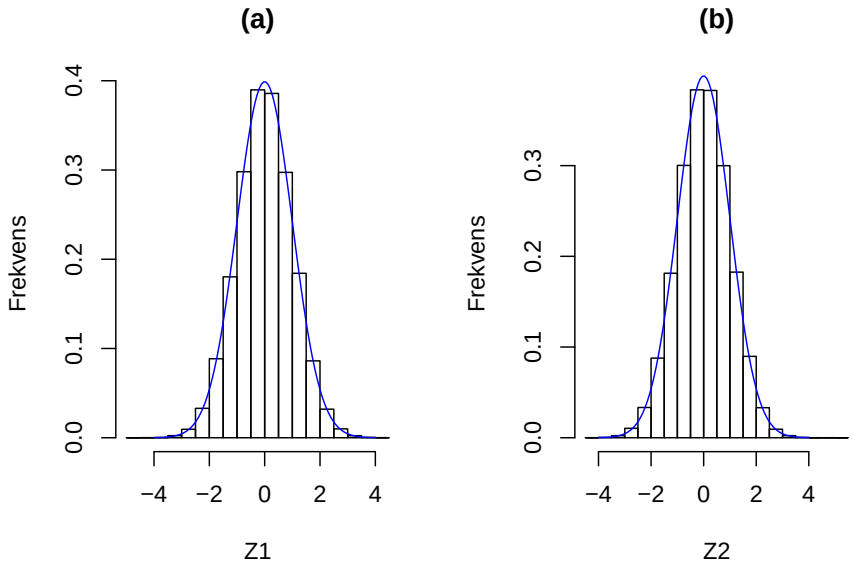

Figure 3.8: Sammenligning af histogrammerne for henholdsvis Z1 (a) og Z2 (b) med densiteten for standard normalfordeling.

Det ses på histogrammerne, at de tilnærmelsesvist er normalfordelte, samt at de har en middelværdi på ≈ 0 og en standardafvigelse på ≈ 1, og derved er `Z1` og `Z2` tilnærmelsesvist standard normalfordelte.

Desuden er det også en konsekvens af Box-Muller-transformationerne, at `Z1` og `Z2` er uafhængige. Dette undersøges på figur 3.9.

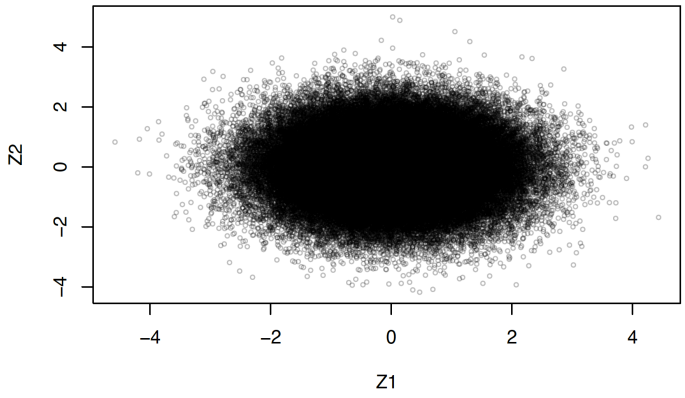

Figure 3.9: Graf for uafhængigheden mellem Z1 og Z2.

Udover at eftervise uafhængigheden af `Z1` og `Z2` grafisk, kan det også undersøges ved hjælp af funktionen `cor` i R, som giver en værdi for korrelationen af de to talsekvenser. Korrelationen af to variabler udregnes ved formlen

$$cor(X, Y) = \frac{cov(X, Y)}{sd(X) \cdot sd(Y)}$$

Hvor *cov* angiver kovariansen og *sd* angiver standardafvigelsen.

Korrelation fortæller, hvor stor en lineær sammenhæng der er mellem to variabler. Denne værdi ligger i intervallet $[-1, 1]$, hvor -1 påviser en perfekt negativ lineær sammenhæng, 1 påviser en perfekt positiv lineær sammenhæng, og 0 viser, at der ingen lineær sammenhæng er.

```
korrelation <- cor(Z1, Z2)
```

Et resultat på afrundet `-0.0021`, hvilket næsten er 0, viser altså også at der ingen lineær korrelation er mellem `Z1` og `Z2`, hvilket giver en indiktion for, at de er uafhængige.

3.2 Simulering i R

I dette afsnit gives eksempler på, hvorledes der kan udføres simuleringer af forskellige fordelinger, således disse fordelinger kan anvendes til statistisk inferens. Afsnittet er primært baseret på kilden [Peng, 2019].

I dataanalysen vil de viste fordelinger bruges som populationer, og stikprøver udtages så ved hjælp af forskellige funktioner. Idéen med denne fremgangsmåde er, at der derved udtages stikprøver fra teoretisk uendelige fordelinger.

Nogle af de fordelinger, der kan simuleres i R, er normal-, binomial-, uniforme-, beta- og gammafordelinger. Alle disse fordelinger har forskellige karateristika, som påvirker hvilke værdier de kan antage. Der vil nu vises eksempler på, hvordan disse fordelinger simuleres i R, og hvordan resultatet vil se ud.

3.2.1 Normalfordeling

I R simuleres en normalfordeling ved funktionen `rnorm`. At en stokastisk variabel, X, er normalfordelt med middelværdi μ og standardafvigelse σ skrives $X \sim N(\mu, \sigma)$. Fordelingen på figur 3.10 bliver genereret ud fra en middelværdi på 0 og en standardafvigelse på 1.

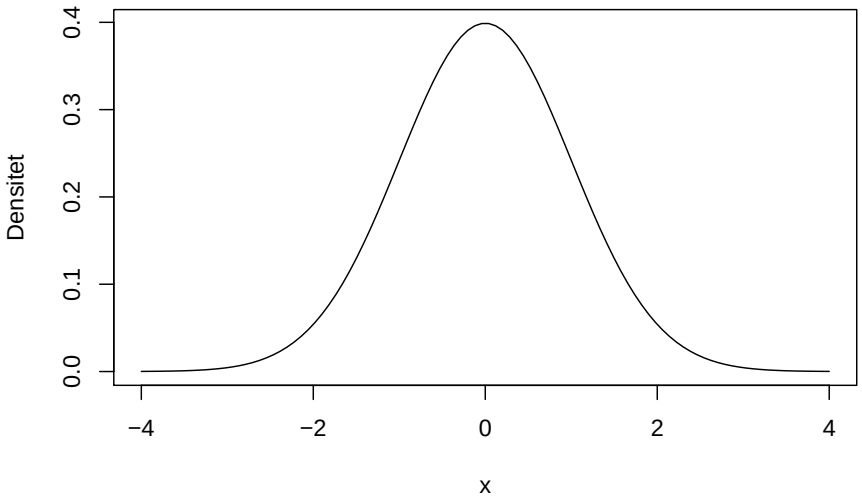

Figure 3.10: Densitetskurven for en standard normalfordeling.

3.2.2 Uniform fordeling

En uniform fordeling er defineret ud fra et interval, som der skal genereres et givent antal værdier indenfor. Alle disse værdier har lige stor sandsynlighed for at blive genereret. Jo større n er, des mere uniformt bliver histogrammet. At en stokastisk variabel, X, er uniformt fordelt i intervallet (a, b) skrives $X \sim \text{unif}(a, b)$. På figur 3.11 ses densitetskurven for en uniform fordeling.

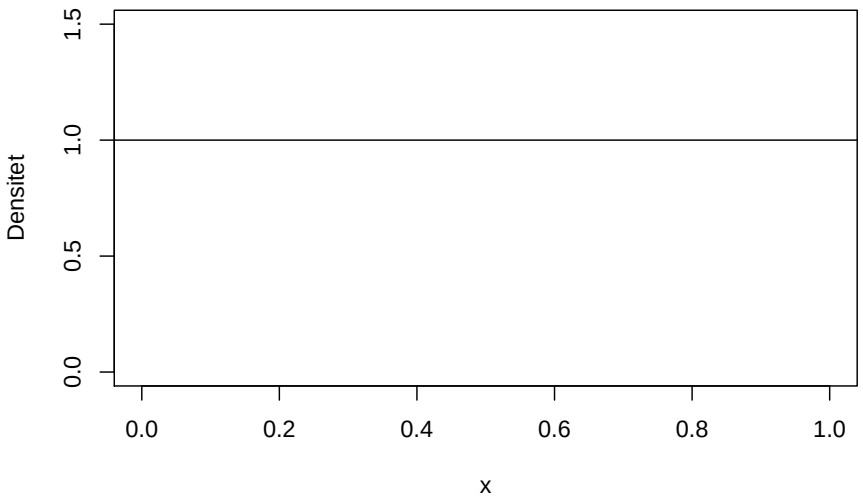

Figure 3.11: En uniform fordeling, hvor alle tal har lige stor sandsynlighed for at optræde i intervallet [0 , 1].

3.2.3 Binomialfordeling

En binomialfordeling bliver genereret ud fra sandsynligheder med to udfald, succes eller fiasko. I den binomiale fordeling gives en *size* som er størrelsen af et eksperiment, samt en sandsynlighed for succes, angivet med *prob*. For hver gentagelse af eksperimentet optælles andelen af successer, som minimalt kan være 0 og maksimalt 1. At en stokastisk variabel, X, er binomal fordelt med n antal eksperimenter og en sandsynlighed for sucess på p, skrives $X \sim \text{binom}(n, p)$. Andelen af successer fra hvert eksperiment illustreres herefter i et histogram.

På figur 3.12 vises densitetskurven af en binomialfordeling.

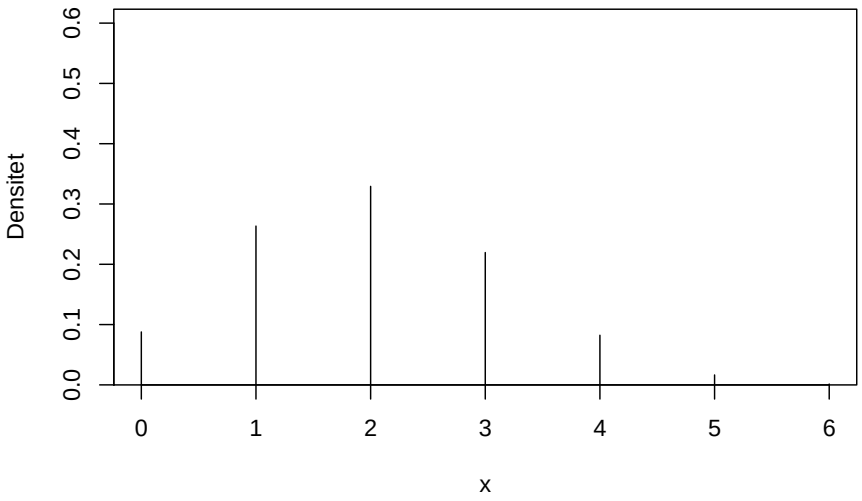

Figure 3.12: En binomialfordeling med size = 6 og en sandsynlighed for succes på prob = 1/3.

3.2.4 Skæve fordelinger

En skæv fordeling er kendetegnet ved, at størstedelen af observationerne er samlet omkring visse værdier, mens de resterende observationer fordeler sig ud til kun én af siderne. De resterende observationer kaldes for "halen" af fordelingen, og alt efter retningen af dem, kaldes den skæve fordeling for "venstreskæv" elller "højreskæv".

3.2.4.1 Betafordeling

En skæv fordeling kan være en betafordelingAt en stokastisk variabel, X er betafordelt med parametrene α og β skrives $X \sim Beta(\alpha, \beta$. I en betafordeling angiver $\alpha - 1$ andelen af sucesser, og $\beta - 1$ angiver andelen af fiaskoer, [Kim, 2020].

En venstreskæv og en højreskæv betafordeling vil nu simuleres og vises på figur 3.13, hvor $\alpha = 8$ og $\beta = 2$ for den venstreskæve, mens $\alpha = 2$ og $\beta = 8$ for den højreskæve.

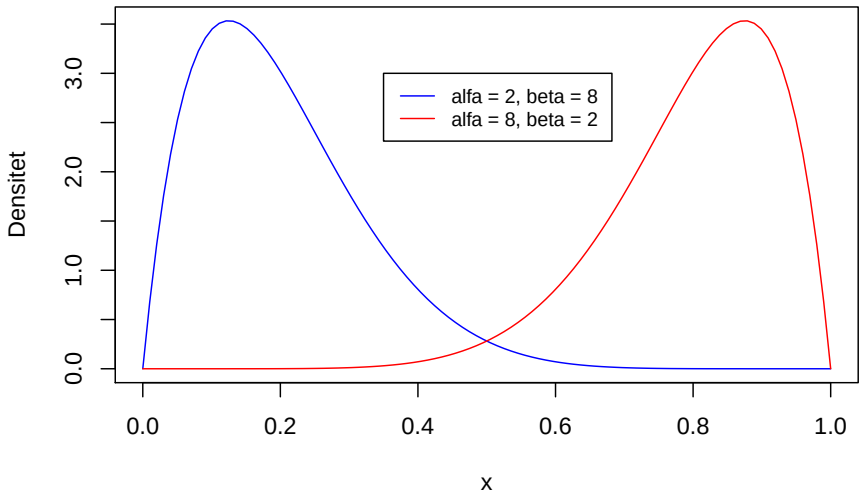

Figure 3.13: To betafordelinger med henholdsvis alfa = 8 og beta = 2, og alfa = 2 og beta = 8.

3.2.4.2 Gammafordeling

Gammafordelingen er ligeledes en skæv fordeling. Funktionaliteten af gammafordelingen er at forudsige ventetiden til den α'te begivenhed. Tæthedsfunktionen for gammafordelingen, er beskrevet med to forskellige parametersæt, henholdsvis $(k,\ \theta)$ og $(\alpha,\ \beta)$. I denne rapport vil der fremefter blive anvendt parameterne $(\alpha,\ \beta)$. Tæthedsfuktionen er givet ved $f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$, hvor $\alpha > 0$ er *shape*, $\beta > 0$ er *rate* og $x \in (0, \infty)$. At en stokastisk variabel, X, er gammafordelt med *shape* α og *rate* β, skrives $X \sim Gamma(\alpha, \beta)$, [Kim, 2019]. På figur 3.14 er vist tre forskellige højreskæve gammafordelinger.

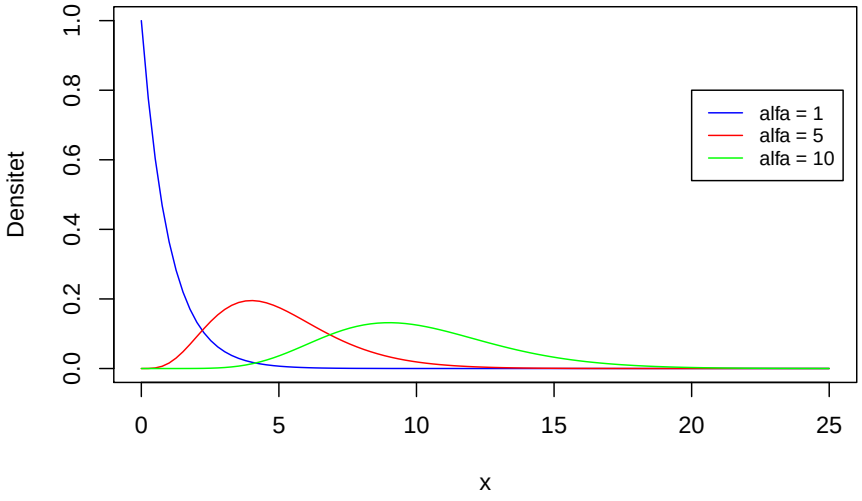

Figure 3.14: Tre gammafordelinger hvor beta = 1 og alfa på henholdsvis 1, 5 og 10.

3.2.5 Repetitioner

I nogle statistiske metoder er det nødvendigt at arbejde med flere stikprøver end kun én enkelt. Med funktionen `replicate` i R, kan der genereres adskillige nye stikprøver ud fra den oprindelige. Processen med *replicate* vil nu vises. Først ses en stikprøve med tre observationer.

```
##               Oprindelig stikprøve
## Observation 1           -0.3268578
## Observation 2            1.0451809
## Observation 3            0.3082858
```

Replicate-funktionen vil nu bruges. I alt replikeres der fem gange, hvilket betyder, at der bliver oprettet fem nye stikprøver. I forbindelse med denne process anvendes funktionen `sample`, som er funktionen der *resampler*, enten med eller uden tilbagelægning fra stikprøven. I dette eksempel bruges funktionen med tilbagelægning. Fra den oprindelige stikprøve udtrækkes en observation, som indsættes i en ny stikprøve. Observationeren bliver derefter lagt tilbage i den oprindelige stikprøve, dette medfører at den samme observation kan udtages flere gange. Grunden til dette er, at stikprøven anses for at være repræsentativ for population, og to observationer af den samme værdi derfor ikke er utænkelig.

```
ny_stik <- replicate(n = 5, {
  sample(x = stik, size = 3, replace = TRUE)
})
ny_stik
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,]  1.0451809 -0.3268578  1.0451809  1.0451809  0.3082858
## [2,] -0.3268578  1.0451809  1.0451809  1.0451809 -0.3268578
## [3,]  0.3082858  1.0451809 -0.3268578 -0.3268578 -0.3268578
```

Funktionen har nu resamplet fem nye repræsentative stikprøver udfra den oprindelige. Denne metode og dens anvendelser vil der kigges nærmere på kapitel 6.

Chapter 4

T-test for skæve stikprøver

I de tilfælde, hvor det ikke er muligt at overholde alle de pågældende antagelser, som beskrevet i afsnit 2.3.1, kan det ikke med sikkerhed antages, at resultaterne er retvisende. I dette afsnit vil det vises, hvad der kan ske, hvis stikprøverne ikke er normalfordelte, når der arbejdes med en uparret t-test.

I dette eksempel benyttes betafordelingen, se figur 3.13, og gammafordelingen, se figur 3.14, til at udføre en uparret t-test.

Nulhypotesen er $H_0 : \mu_1 - \mu_2 = 0$ og den alternative hypotese er $H_1 : \mu_1 - \mu_2 \neq 0$. Denne nulhypotese undersøges ved hjælp af en uparret t-test. Der udtages en stikprøve fra hver af de viste fordelinger, som der udføres to-sidet uparret t-test på, ved hjælp af den indbyggede funktion `t.test`.

```
Stik1 <- rbeta(n = 100, shape1 = 8, shape2 = 2)
Stik2 <- rgamma(n = 100, shape = 1, rate = 2)
t_test <- t.test(Stik1, Stik2, alternative = "two.sided",
                 mu = 0, conf.level = 0.95)
```

Udfra t-testen fås et konfidensinterval på [`0.262, 0.4413`]. Forskellen mellem populationernes middelværdier vil ligge i dette interval med 95% sikkerhed, ifølge t-testen. Dækningsgraden af et konfidensinterval udarbejdet fra en uparret t-test kan undersøges ved at trække nye stikprøver fra populationerne, i alt 10,000 gange, og hver gang oprette et nyt konfidensinterval. Den sande dækningsgrad er andelen af gangene, forskellen mellem populationernes middelværdier er indeholdt i konfidensintervallerne.

```
# Middelværdien for en betafordeling er alfa/(alfa+beta)
middel_stik1 <- 8/(8+2)
# Middelværdien for en gammafordeling er shape*(shape/rate)
middel_stik2 <- 1*(1/2)
```

```r
sand_dif <- abs(middel_stik1 - middel_stik2)

daekningsgrad <- replicate(n = 10000, {
  x1 <- rbeta(n = 20, shape1 = 8, shape2 = 2)
  x2 <- rgamma(n = 20, shape = 1, rate = 2)

  t_test <- t.test(x1, x2, alternative = "two.sided",
                   conf.level = 0.95)
  konf_interval <- t_test$conf.int

  # Tjekker, om den sande difference ligger i konfidensintervallet
  konf_interval[1L] <= sand_dif & konf_interval[2L] >= sand_dif
})

tf <- table(daekningsgrad)
tf
```

```
## daekningsgrad
## FALSE  TRUE
##   794  9206
```

Det fremgår fra tabellen, at dækningsgraden af konfidensintervallerne er `92.06`%. Dette stemmer ikke overens med antagelsen om, at konfidensintervallet har en dækningsgrad på 95%. Det kan derfor ikke antages, at en t-test på en ikke-normalfordelt stikprøve altid giver retvisende resultater.

Chapter 5

Permutationer

I afsnit 2.3.1 blev der redegjort for, hvilke antagelser, der skal være opfyldt, for at en t-test giver retvisende resultater. Ligeledes blev der i afsnit 4 vist hvorledes en t-tests resultater kan være misvisende, hvis stikprøverne ikke er normalfordelte. I dette kapitel vil der redegøres for en løsning til dette problem. Hvis antagelserne ikke er opfyldte, og en t-test derfor ikke kan anvendes, er det muligt at anvende permutationer.

Givet en ordnet liste, $X = [x_1, x_2, \dots, x_n]$, er en permutation, π, en omarrangering af listens elementer, $X_\pi = [x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)}]$. Antallet af mulige permutationer af en ordnet liste, X, af længde n, er givet ved $n!$, [Weisstein, 2020b].

Som eksempel, er en mulig permutation, π_1, af $A = [1, 2, 3]$ givet ved $A_{\pi_1} = [2, 1, 3]$, og der er i alt $|A|! = 3! = 1 \cdot 2 \cdot 3 = 6$ mulige permutationer af listen.

Ved hjælp af permutationer er det muligt at udføre en hypotesetest. En sådan hypotesetest kaldes for en permutationstest.

5.1 Permutationstest

I dette afsnit gives en forklaring af, hvad en permutationstest er. Afsnittet er skrevet på baggrund af [Chihara and Hesterberg, 2019b, side 54].

Antag, at en tilfældig stikprøveudtagning af en population kan udtrykkes ved $X = [x_1, x_2, \dots, x_n]$. Der opstilles en nulhypotese H_0 og en alternativ hypotese, H_1, der er relevante at undersøge. Der kan så vælges en teststørrelse at undersøge for at måle evidensen imod H_0. Værdien af denne teststørrelse er givet ved S_{obs} for stikprøven.

En permutationstest går ud på at forsøge at måle evidens imod H_0, som i en almindelig t-test. Dette gøres ved at beregne teststørrelsen for samtlige permutationer af X. Denne mængde af permutationer er givet ved $\Pi_X =$

$\{X_{\pi(1)}, X_{\pi(2)}, ... X_{\pi(M)}\}$, hvor M er antallet af mulige permutationer af X. Teststørrelsen for den i'te permutation benævnes S_{π_i}.

Selve beregningen af teststørrelsen afhænger af de specifikke hypoteser, men det er et krav, at store værdier for S_{π_i} indikerer evidens imod H_0. Desuden er det kun nulhypoteser om ingen forskel, der kan testes ved hjælp af permutationer - eksempelvis kan evidensen imod $H_0 : \mu_1 \neq \mu_2$ ikke beregnes, [Chihara and Hesterberg, 2019a, s. 48].

P-værdien for en tosidet test beregnes som antal gange, den numeriske værdi af permutationsteststørrelsen, $|S_{\pi_i}|$, er større end den observerede teststørrelse, S_{obs}, divideret med antallet af mulige permuationer, M. P-værdien beregnes på tilsvarende vis for en ensidet test, hvor $|S_{\pi_i}|$ erstattes med S_{π_i}, og uligheden erstattes med det tilsvarende passende. **FIXME** (Skulle vi ikke have en kilde på den formel og fremgangsmåde?)

$$p = \frac{\#(|S_{\pi_i}| \geq S_{obs}) + 1}{M + 1}$$

Permutationstest har den fordel, at den antager meget lidt omkring stikprøven. I særdeleshed antager den hverken tilfældig stikprøveudtagning, eller at populationen, fra hvilken stikprøven stammer, er normalfordelt. Desuden er permutationstest især smart, når fordelingen af teststørrelsen er ukendt.

Der er dog visse ulemper ved en permutationstest. For det første, skal det antages, at den del af data, der permuteres, er ombyttelig. For det andet er det ikke alle teststørrelser, der kan undersøges ved hjælp af en permutationstest - for eksempel vil middelværdien af samtlige permutationer være lig hinanden for en test for en enkelt stikprøve. For det tredje bliver antallet af mulige permutationer hurtigt meget stor, når størrelsen af stikprøven stiger. Hvis der for eksempel er $n = 10$ datapunkter i stikprøven, vil der være $10! = 3,628,800$ mulige permutationer af stikprøven.

Netop på grund af ombytteligheden af stikprøven fungerer permutationstests. Hvis stikprøven er ombyttelig, vil hver eneste permutation af stikprøven, inklusiv stikprøven selv, være lige sandsynlige. Dette medfører, at hvis H_0 er sand, vil enhver difference imellem S_{obs} og S_{π_i} være lille og tilfældig. Hvis værdien for S_{π_i} på den måde konkluderes at være væsentlig anderledes end S_{obs}, tyder det på, at H_0 skal forkastes.

Det store antal af permutationer afhjælpes ved kun at beregne S_{π_i} for et passende antal, tilfældigt udvalgte, permutationer af stikprøven, og et acceptabelt resultat vil stadig blive opnået.

5.2 Uparret permutationstest

Der vil nu vises et eksempel på, hvordan permutationstest kan anvendes som en hypotesetest. Antag, at det ønskes undersøgt, om middelværdien af to stikprøver

er signifikant forskellige. Den opstillede nulhypotese bliver $H_0 : \mu_1 - \mu_2 = 0$, og den alternative hypotese bliver $H_1 : \mu_1 - \mu_2 \neq 0$.

Der vil blive udtrukket to stikprøver, vist på figur 5.1, fra den samme betafordeling.

```
stik1 <- rbeta(n = 100, shape1 = 8, shape2 = 2)
stik2 <- rbeta(n = 100, shape1 = 8, shape2 = 2)
```

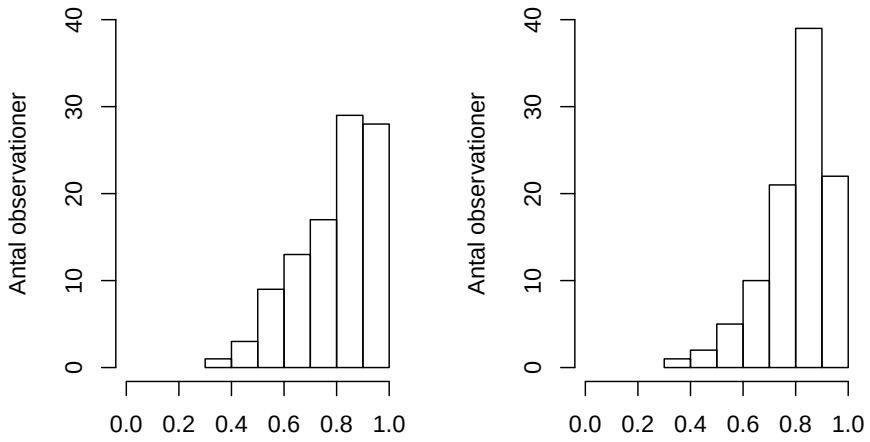

Figure 5.1: To stikprøver udtrukket fra den samme betafordeling med parametrene alfa = 8 og beta = 2.

Teststørrelsen vil i dette tilfælde være forskellen mellem stikprøvernes middelværdi, $S_{obs} = |\widehat{\mu}_1 - \widehat{\mu}_2|$. Den observerede teststørrelse ses at være `0.0076034`. Hvorvidt denne forskel er signifikant, kan nu undersøges ved at beregne forskellen i middelværdi for samtlige permutationer af stikprøverne. Observationerne fra hver stikprøve vil blive forenet i en enkelt stikprøve, som der vil permuteres udfra.

I nedenstående kode udføres der 100 permutationer på den forenede stikprøve. Den forenede stikprøve har en størrelse på $N = 200$. I hver permutation udtrækkes de første 100 observationer til at være den permuterede stikprøve `permuteret_stik1` , mens de sidste 100 observationer bliver til den permuterede stikprøve `permuteret_stik2`. Derefter udregnes den absolutte forskel i middelværdien mellem de nye permuterede stikprøver, i alt 100 gange, en for hver permutation. Denne difference kaldes den permuterede teststørrelse, S_π.

```
S_obs <- abs(mean(stik1) - mean(stik2))
forenet_stik <- c(stik1, stik2)

S_pi <- c()
for(i in 1:100){
  permuteret_forenet_stik <- sample(forenet_stik, size = (2 * 100))
  permuteret_stik1 <- head(permuteret_forenet_stik, 100)
  permuteret_stik2 <- tail(permuteret_forenet_stik, 100)

  S_pi[i] <- abs(mean(permuteret_stik1) - mean(permuteret_stik2))
}
head(S_pi, 3)
```

```
## [1] 0.018010076 0.006666829 0.012069227
```

I outputtet *head(S_pi, 3)* ses et eksempel på tre af de permuterede teststørrelser. På figur 5.2 illusteres fordelingen af de permuterede teststørrelser.

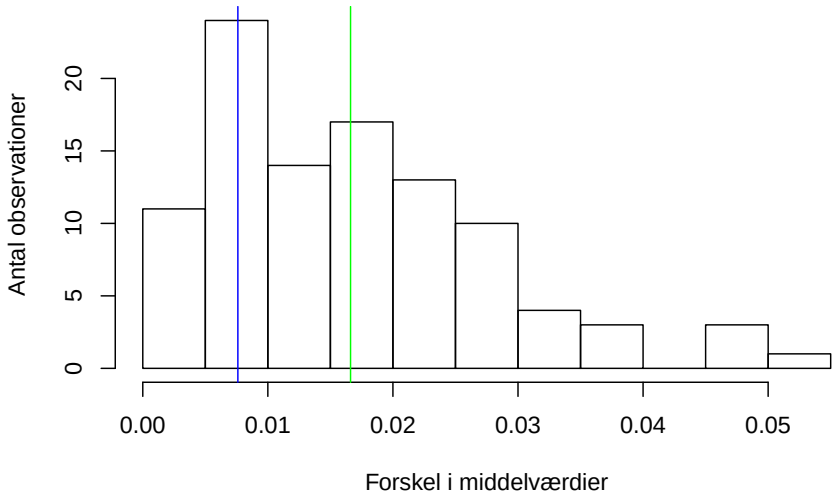

Figure 5.2: Permutationsfordelingen af teststørrelserne. Den blå linje viser den observerede forskel i middelværdierne, og den grønne viser middelværdien for samtlige permuterede teststørrelser.

Nu kan p-værdien beregnes, ved at udregne hvor mange gange de permuterede teststørrelser er større end den observerede teststørrelse. Denne værdi divideres derefter med antallet af permutationer. Dette gøres i nedenstående kodestykke.

```
antal_ekstreme_vaerdier <- table(abs(S_pi) >= S_obs)[2]

p_vaerdi <- unname(antal_ekstreme_vaerdier+1)/(length(S_pi)+1)
p_vaerdi
```

```
## [1] 0.7425743
```

Med en p-værdi på `0.7425743`, er der ikke noget belæg for at forkaste nulhypotesen.

Type-I fejl

Antallet af type-I fejl, der opstår i en uparret permutationstest kan undersøges ved at se, hvor mange gange H_0 forkastes, selvom H_0 er sand. Antallet af type-I fejl bør svare til det valgte signifikansniveau, [Agresti and Finlay, 2014, s. 159-160].

I koden nedenfor bestemmes antallet af type-I fejl for en uparret permutationstest.

```
konf_niveau <- 0.95
reps <- 1000
n <- 100
type_1_fejl <- replicate(n = reps, {
  stik1 <- rbeta(n, shape1 = 8, shape2 = 2)
  stik2 <- rbeta(n, shape1 = 8, shape2 = 2)
  samlet_stik <- c(stik1, stik2)

  S_obs <- abs(mean(stik1) - mean(stik2))
  S_pi <- c()

  for(i in 1:(n*2)){
    permuteret_samlet_stik <- sample(samlet_stik, size = (2 * n))
    permuteret_stik1 <- head(permuteret_samlet_stik, n)
    permuteret_stik2 <- tail(permuteret_samlet_stik, n)

    S_pi[i] <- abs(mean(permuteret_stik1) - mean(permuteret_stik2))
  }

  antal_ekstreme_vaerdier <- table(abs(S_pi) >= S_obs)[1]

  p_vaerdi <- (200-sum(antal_ekstreme_vaerdier)+1)/(length(S_pi)+1)
  type1fejl <- p_vaerdi > 1 - konf_niveau
  return(type1fejl)
})

fejltable <- table(type_1_fejl)
fejltable
```

```
## type_1_fejl
## FALSE  TRUE
##    56   944
```

I alt forkastes H_0 fejlagtigt i 5.6% af tilfældene, hvilket stemmer overens med det valgte signifikansniveau.

5.3 Parret permutationstest

Ligesom permutationstest kan bruges til at lave en uparret hypotesetest, kan den også bruges til en parret. Fremgangsmåden er den samme, bortset fra, at det er hvert koordinatpar (x_i, y_i) i stikprøven $Z = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$, der permuteres.

I nedenstående kode, vises et eksempel på en parret permutations-test for forskel i middelværdi, der gør brug af pakken `wPerm`.

```
reps <- 100
n <- 100
permutationer <- 1000

p_reps <- replicate(n = reps, {
  stik1 <- rgamma(n, shape = 10, rate = 2)
  stik2 <- rgamma(n, shape = 4, rate = 13)

  test <- wPerm::perm.paired.loc(x = stik1, y = stik2,
                                 parameter = mean,
                                 alternative = "two.sided",
                                 R = permutationer)

  p_vaerdi <- test$p.value
})

p_vaerdi_parret <- mean(p_reps)
p_vaerdi_parret
```

```
## [1] 0.002
```

Med en p-værdi på 0.002 kan nulhypotesen forkastes, da der er evidens for, at den ikke er korrekt.

Chapter 6

Bootstrap

I dette kapitel vil den teoretiske del af bootstrap blive beskrevet.

Der findes forskellige bootstrap-metoder, som varierer på forskellige punkter. Valget af bootstrap-metode afhænger af den individuelle situation, hvor der skal udføres statistisk inferens. Der gøres opmærksom på, at i den resterende del af rapporten, vil ordet bootstrap henvise til den ikke-parametriske bootstrap-metode. Ikke-parametrisk bootstrap, er når der ikke sættes specifikke antagelser eller en præcis model for populationen, når undersøgelsen udføres. Derimod antages det, at en stikprøve er repræsentativ for hele populationen, [Berrar, 2019, side 3].

Der vil i kapitlet blive undersøgt, hvordan bootstrap kan anvendes i praksis til at udregne standardfejl for en estimator, hvordan hypotesetest kan udføres ved hjælp af bootstrap og, hvordan bootstrap kan benyttes til at angive konfidensintervaller. Til hver anvendelse vil der også blive givet et eksempel på metoden.

6.1 Ikke-parametrisk bootstrap

Bootstrap er en resampling-metode, der bruges til at generere yderligere stikprøver, kaldet bootstrap-stikprøver, ud fra en given stikprøve, hvor målet er at udføre statistisk inferens for en valgt teststørrelse. For eksempel kan bootstrap give et indblik i tendenser for teststørrelsen, såsom standardfejlen og forventningsrethed, eller udregne konfidensintervaller. Der gøres opmærksom på, at bootstrap ikke kan bruges til at få et bedre estimat for parameteren, da bootstrap-fordelingen er centreret omkring stikprøvens estimat, se figur 6.1, for eksempel middelværdien $\hat{\mu}$, og ikke populationens middelværdi, μ, [Chihara and Hesterberg, 2019a, s. 114].

Hver bootstrap-stikprøve har samme størrelse som stikprøven. Bootstrap oper-

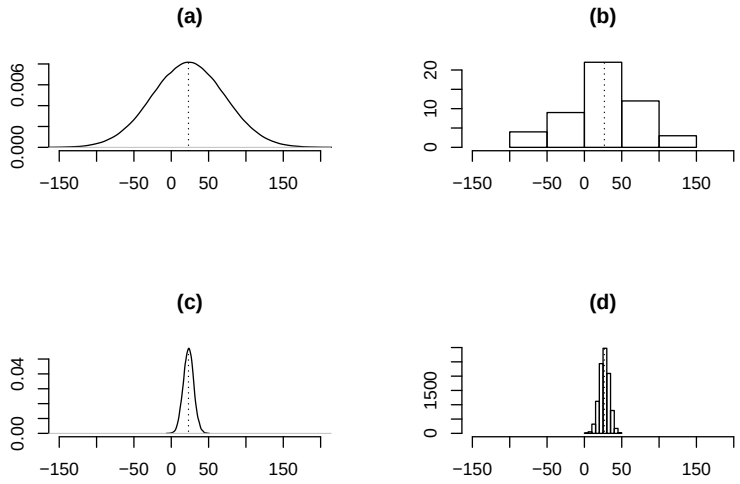

Figure 6.1: (a) Populationens fordeling, N(23, 49). (b) Fordelingen på en stikprøve af størrelsen 50. (c) Den teoretiske fordeling af stikprøvens middelværdi. (d) Fordelingen af bootstrap-stikprøvernes middelværdi. De stiplede linjer representerer middelværdien, [@MathStat, side 108].

erer med tilbagelægning, så der er en sandsynlighed for, at et givent datapunkt bliver udtaget mere end en gang. Samtidig er der en sandsynlighed for, at et datapunkt slet ikke bliver udvalgt. Det er relevant at undersøge, hvor mange af de oprindelige observationer, som i gennemsnit medtages i nye bootstrap-stikprøver, og ligeledes, hvor mange, som udelades.

Sandsynligheden for, at en specifik observation ikke udtages én gang fra de oprindelige n observationer, er $1 - 1/n$, og sandsynligheden for, at denne observation ikke udtages n gange er $P() = (1 - 1/n)^n$. Når stikprøvestørrelsen, n, går mod uendeligt gælder, at $(1 - 1/n)^n \overset{n\to\infty}{=} 1/e \approx 0.368$. Derfor vil en bootstrap-stikprøve af tilpas stor størrelse indeholde $\approx 63.2\%$ observationer fra den oprindelige stikprøve, og udelade $\approx 36.8\%$, [Wicklin, 2017].

I alt bliver der genereret B bootstrap-stikprøver, som der hver især udføres statistisk inferens på. Med den computerkraft der er tilgængelig i dag, anbefales der af kilden, [Marin, 2018], mindst $10,000$ resamples, derved $B \geq 10,000$, for at få et nøjagtigt estimat. Grunden til, at der ikke genereres et endnu større antal bootstrap-stikprøver end de $10,000$ er, at bootstrap-stikprøven genereres ud fra den obseverede data. Et større B vil derfor ikke medføre yderligere information om populationen, men vil dog medvirke til et mere præcist estimat, [Marin, 2018, 10:20].

Fordelen ved bootstrap er, at selvom der kun er én tilgængelig stikprøve fra den underliggende population, er der stadig mulighed for at estimere stikprøvefordelingen, uden at der kræves yderligere stikprøver fra populatonen. Dette skyldes netop antagelsen om, at stikprøven skal være repræsentativ for populationen.

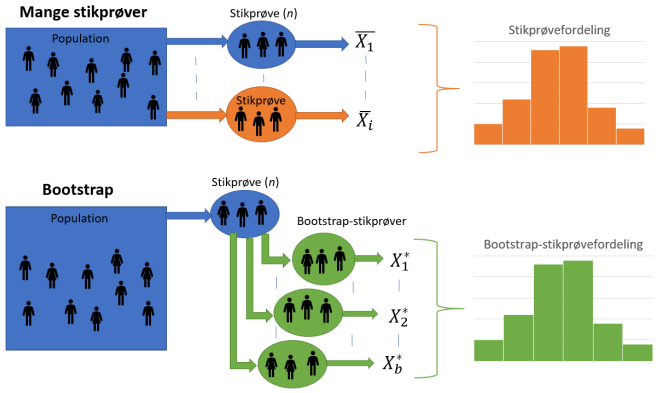

Figure 6.2: Her er illustreret forskellen mellem at finde den teoretiske stikprøvefordeling ved hjælp af mange stikprøver fra populationen (orange), og måden hvorpå stikprøvefordelingen kan findes ved hjælp af kun én stikprøve, der udføres bootstrap på (grøn).

Der er to hovedårsager til at benytte bootstrap, som beskrevet i [Marin, 2018].

For det første, hvis stikprøven ikke er tilpas stor, og stikprøvefordelingen derfor heller ikke kan antages at være normalfordelt. For det andet, hvis metoden til at beregne teststørrelsens standardfejl er teoretisk avanceret. Eksempelvis er standardfejlen for middelværdien nem at løse, $\widehat{\text{se}}(\hat{\mu}) = \frac{S}{\sqrt{n}}$, mens det ikke er tilfældet, hvis det i stedet er afstanden mellem to percentiler, der estimeres.

6.2 Bootstrap-standardfejl

Der vil i dette afsnit beskrives, hvordan standardfejl af en stikprøve kan udregnes ved hjælp af bootstrap. Udregnes standardfejlen på denne måde, kaldes den for bootstrap-standardfejlen. I det efterfølgende afsnit vil standardfejlen inddrages i beregningen af konfidensintervallet for en bootstrap-stikprøve. Det følgende afsnit er primært skrevet på baggrund af [Yen, 2019].

Standardafvigelsen for en estimator beskrives som estimatorens standardfejl. Standardfejlen er et udtryk for, hvor stor en afvigelse der er fra populationens parameter til stikprøvens estimat. Jo mindre standardfejlen er, desto mindre er afvigelsen mellem estimatet og parameteren. Som udgangspunkt vil en stikprøves estimat aldrig være lig populationens parameter, fordi der ved udtagning af en stikprøve, i hvert tilfælde vil være variabilitet. Standardfejlen er et mål for denne variabilitet.

Som eksempel vil standardfejlen for estimatet af middelværdien, $\hat{\mu}$, være $\text{se}(\hat{\mu}) = \frac{\sigma}{\sqrt{n}}$.

Når der arbejdes med data udover det teoretiske, vil standardafvigelsen for populationen, σ, altid være ukendt. Derfor bruges stikprøvens estimat for standardafvigelsen, S til at beregne den estimerede standardfejl, $\widehat{\text{se}}$.

Som eksempel vil den estimerede standardfejl for estimatet af middelværdien, $\hat{\mu}$, være $\widehat{\text{se}}(\hat{\mu}) = \frac{S}{\sqrt{n}}$, hvor $S = \sum_{i=1}^{n} \frac{(x_i - \hat{\mu})^2}{n-1}$ er stikprøvens standardafvigelse for middelværdien, og n er størrelsen på stikprøven.

Dog er dette ikke altid ligetil i virkeligheden. Oftest er der ikke tilstrækkelige informationer om populationen eller fordeligen af denne. Samtidig kræver det, at der er nogle specifikke krav, som er opfyldt. Disse problemer kan undgås ved at benytte bootstrap til at estimere bootstrap-standardfejlen, givet ved nedstående formel.

$$se(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}_b^* - \bar{\theta})^2}$$

Hvor $\hat{\theta}$ er stikprøvens estimat for den ønskede parameter, B er antal bootstrap-stikprøver, $\hat{\theta}_b^*$ er estimatet for den b'te bootstrap-stikprøve og $\bar{\theta} = (\frac{1}{B})\sum_{b=1}^{B}\hat{\theta}_b^*$, [Bartlett, 2013].

Eksempel

I kodestykket nedenfor udtages en tilfældig stikprøve fra en standard normalfordelt population. Først udregnes standardfejlen for stikprøven. Derefter laves $10,000$ bootstrap-stikprøver, som benyttes til at beregne stikprøvens bootstrap-standardfejl. Til sidst sammenlignes disse to estimater.

```
n <- 1000

stik <- rnorm(n, mean = 0, sd = 1)
SD_stik <- sd(stik)
SE_stik <- SD_stik/sqrt(n)

B <- 10000

middel_funk <- function(stik, i){mean(stik[i])} # Estimator

# Bootstrap-stikprøver
Boot_stik <- boot::boot(data = stik, statistic = middel_funk, R = B)
# Standardfejlen for bootstrap-stikprøverne
SE_boot <- sd(Boot_stik$t)

# Procentvis afvigelse mellem normal standardfejl og
# bootstrap-standardfejl
diff_procent <- ((abs(SE_stik - SE_boot))/SE_stik) * 100
```

Fra eksemplet fås, at bootstrap-standardfejlen for stikprøven er $se(\hat{\theta})$ =0.0325, hvilket approksimerer stikprøvens standardfejl på se($\hat{\mu}$) =0.0325, med en forskel i dette tilfælde på 0.064%. Hvis der opstår en situation, hvor der ikke er en simpel måde, hvorpå det er muligt at udregne standardfejlen, kan bootstrap-standardfejlen altså udnyttes, da en forskel på 0.064% ikke er signifikant.

6.3 Bootstrap-hypotesetest

Som nævnt i afsnit 2.3.1, skal visse antagelser være opfyldt, for at garantere korrektheden af resultaterne af en t-test, og det efterfølgende resultatet af overtrædelsen af disse antagelser, blev vist i afsnit 4.

Når disse antagelser ikke er opfyldt, kan bootstrap anvendes til at udføre hypotesetest, og i så fald kaldes det i det følgende for en bootstrap-test. I følgende to afsnit gennemgås først fremgangsmåden for en uparret bootstrap-test, og dernæst fremgangsmåden for en parret bootstrap-test.

6.3.1 Uparret bootstrap-test

I dette afsnit benyttes bootstrap til at lave en uparret hypotesetest på skæve stikprøver.

Lad to uafhængige uparrede stikprøver, $X = [x_1,\ x_2,\ ...,\ x_n]$ og $Y = [y_1,\ y_2,\ ...,\ y_m]$, hvor $X \wedge Y \sim \text{Beta}(2, 8)$, med ens varians være givet på figur 6.3.

```
n <- 20
m <- 50

stik1 <- rbeta(n, shape1 = 2, shape2 = 8)
stik2 <- rbeta(m, shape1 = 2, shape2 = 8)
```

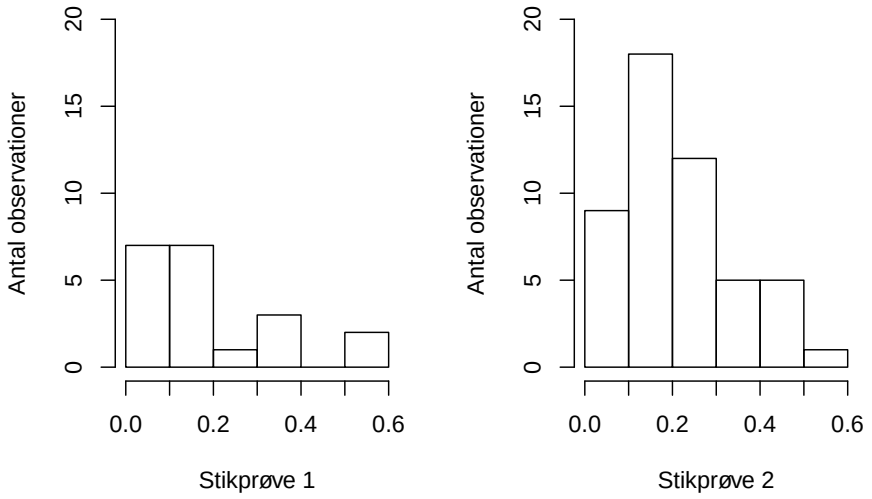

Figure 6.3: To uafhængige uparrede stikprøver

Så opstilles der en nulhypotese, $H_0 : \mu_{_X} - \mu_{_Y} = 0$, hvor $\mu_{_X}$ og $\mu_{_Y}$ er de sande middelværdier for populationerne, hvorfra stikprøverne blev udtrukket og en alternativ hypotese, $H_1 : \mu_{_X} - \mu_{_Y} \neq 0$, samt et signifikansniveau, $\alpha = 0.05$.

På baggrund af forskellen i de to stikprøvers middelværdi, er det muligt at udregne en teststørrelse, $t_{obs} = \frac{(\bar{X}-\bar{Y})-(\bar{X}_0-\bar{Y}_0)}{((s^2_{\bar{X}}/n)+(s^2_{\bar{Y}}/m))}$, hvor $\bar{X}_0$ og $\bar{Y}_0$ middelværdien for X og Y under H_0, og s_i er standardafvigelsen for i.

```
t_obs <- ((mean(stik1) - mean(stik2)) - (0)) /
           (sqrt(((sd(stik1))^2 / n)) +
             ((sd(stik2))^2 / m))
t_obs
```

```
## [1] -0.6280534
```

For at udføre bootstrap-testen, forenes de to stikprøver til en samlet stikprøve med størrelsen $n + m$ observationer. Derefter laves en bootstrap-stikprøve af $n + m$ observationer fra den samlede stikprøve. De første n indgange i den samlede stikprøve er bootstrap-stikprøven for X, og kaldes X^*. De resterende m indgange er bootstrap-stikprøven for Y, og kaldes Y^*. Til sidst udregnes bootstrap-teststørrelsen $t^* = \frac{(\bar{X}^* - \bar{Y}^*) - (\bar{X} - \bar{Y})}{((s^2_{X^*}/n) + (s^2_{Y^*}/m))}$. I alt beregnes der B bootstrap-teststørrelser, [Myung, 2015]. Et eksempel på dette udregnes i nedenstående kodestykke.

```
bootstraps <- 1000
boot_t <- replicate(n = bootstraps, {

    # Samlede bootstrap-stikprøve
    boot <- sample(x = c(stik1, stik2), replace = TRUE)

    boot_x <- boot[1 : n] # Bootstrap-stikprøve 1
    boot_y <- boot[(n + 1) : (n + m)] # Bootstrap-stikprøve 2

    boot_test <- ((mean(boot_x) - mean(boot_y)) -
                    (mean(stik1) - mean(stik2))) /
                      (sqrt(((sd(boot_x))^2 / n)) +
                        ((sd(boot_y))^2 / m))
})
head(boot_t, 3)
```

```
## [1] -1.5835004  0.8703555 -0.3314820
```

Ovenfor ses tre af bootstrap-teststørrelserne, og fordelingen af dem kan ses på på figur 6.4, hvor den observerede teststørrelse er markeret med en blå linje, og bootstrap-teststørrelsernes middelværdi er markeret med en grøn linje.

Herefter kan p-værdien udregnes som ved permutationstest i afsnit 5.1.

```
antal_ekstreme <- abs(boot_t) >= t_obs

p_vaerdi <- (sum(antal_ekstreme) + 1) / (bootstraps + 1)
p_vaerdi
```

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

Med en p-værdi på `1`, kan H_0 ikke forkastes, da der ikke er evidens for at middelværdierne for de to populationer er forskellige. Dette resultat stemmer overens med figur 6.4, da bootstrap-teststørrelsernes middelværdi er tæt på den observerede teststørrelse.

Eksempel

Den udførte test kan ved hjælp af R's funktion `replicate`, gentages mange gange, for at undersøge om resultatet vil blive det samme, ved mange

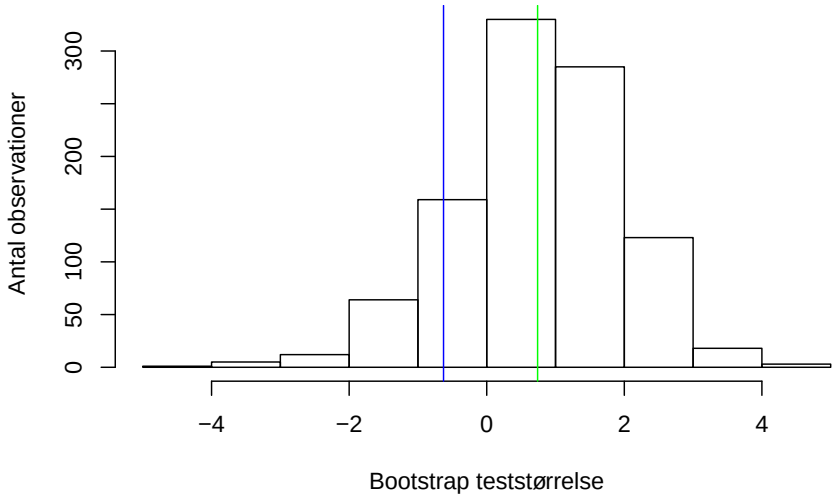

Figure 6.4: Fordelingen af bootstrap-teststørrelserne, hvor den blå linje markerer den observerede teststørrelse, og den grønne linje markerer fordelingens middelværdi.

stikprøveudtagninger. Fremgangsmåden for den uparrede bootstrap-test er den samme. Den bliver blot gentaget 100 gange, og middelværdien af de 100 p-værdier findes.

```
res <- replicate(n = 100, {
  stik1 <- rbeta(n, shape1 = 2, shape2 = 8)
  stik2 <- rbeta(m, shape1 = 2, shape2 = 8)
  t_obs <- ((mean(stik1) - mean(stik2)) - (0)) /
              sqrt(((sd(stik1))^2 / n)) + ((sd(stik2))^2 / m)

  boot_t <- replicate(n = bootstraps, {

    boot <- sample(c(stik1, stik2), replace = TRUE)

    boot_x <- boot[1 : n]
    boot_y <- boot[(n+1) : (n+m)]
    boot_test <- ((mean(boot_x) - mean(boot_y)) -
                    (mean(stik1) - mean(stik2))) /
                      sqrt(((sd(boot_x))^2 / n)) +
                        ((sd(boot_y))^2 / m)
  })

  antal_ekstreme <- abs(boot_t) >= t_obs

  p_vaerdi <- (sum(antal_ekstreme)+1) / (bootstraps + 1)
})

p_vaerdi_uparret <- mean(res)
p_vaerdi_uparret
```

```
## [1] 0.7864935
```

Der ses, at gennemsnittet af p-værdierne er lig `0.7865`, hvilket er større end signifikansniveauet. Dermed er der ikke evidens nok til at forkaste nulhypotesen.

6.3.2 Parret bootstrap-test

I dette afsnit benyttes bootstrap til at lave en parret hypotesetest på skæve stikprøver.

Lad to parrede stikprøver være givet, $X = [x_1,\ x_2,\ ...,\ x_n]$ og $Y = [y_1,\ y_2,\ ...,\ y_n]$. Der oprettes et tredje datasæt, Z, som består af differencerne mellem x_i og y_i, $Z = [x_1 - y_1,\ x_2 - y_2,\ ...,\ x_n - y_n] = [z_1,\ z_2,\ ...,\ z_n]$. Ved hjælp af det nye datasæt er det muligt at udregne teststørrelsen, $t_{obs} = \frac{\bar{Z} - \mu_0}{\hat{SE}(Z)}$.

Så opstilles der en nulhypotese, $H_0 :\ \mu_Z = 0$, hvor μ_Z angiver den sande middelværdi for Z, og en alternativ hypotese, $H_1 :\ \mu_Z \neq 0$, samt et signifikansniveau, $\alpha = 0.05$.

Der udtages en bootstrap-stikprøve for Z, som betegnes Z^*. På baggrund af bootstrap-stikprøven kan der nu udregnes B nye teststørrelser, $t_i^* = \frac{\bar{Z}_i^* - \bar{Z}}{\hat{SE}(Z_i^*)}$, hvor Z_i^* er den i'te bootstrap-stikprøve, og $i = 1, 2, \ldots, B$, [Chihara and Hesterberg, 2019a, s. 106, 124-127, 246].

Eksempel

I nedenstående kode, vises et eksempel på en parret bootstrap-test, hvor stikprøverne ikke er fra den samme fordeling.

```
n <- 15

p_reps <- replicate(n = 100, {
  stik1 <- rgamma(n, shape = 10, rate = 2)
  stik2 <- rgamma(n, shape = 4, rate = 13)
  stik_diff <- stik1 - stik2
  obs_t <- (mean(stik_diff)-0)/(sd(stik_diff)/sqrt(n))

  boot_t <- c()
  for(i in seq(1, bootstraps)){
    boot <- sample(stik_diff, n, replace = TRUE)
    boot_t[i] <- (mean(boot)-mean(stik_diff))/(sd(boot)/sqrt(n))
  }

  antal_ekstreme <- abs(boot_t) >= obs_t

  p_vaerdi <- (sum(antal_ekstreme) + 1) / (bootstraps + 1)
})

p_vaerdi_parret <- mean(p_reps)
p_vaerdi_parret
```

```
## [1] 0.001028971
```

Det ses, at p-værdien er lig `0.001029`, hvilket er mindre end signifikansniveauet. Her vil nulhypotesen forkastes, da der er evidens for at differencen ikke er 0.

Type-I fejl

Antallet af type-I fejl, der opstår i en parret bootstrap-test kan undersøges ved at se, hvor mange gange H_0 forkastes, selvom H_0 er sand. Antallet af type-I fejl bør svare til det valgte signifikansniveau, [Agresti and Finlay, 2014, s. 159-160].

I koden nedenfor bestemmes antallet af type-I fejl for en parret bootstrap-test.

```
n <- 20
konf_niveau <- 0.95
sand_middel <- 0
```

```
type_1_fejl_boot <- replicate(n = 500, {
  stik1 <- rgamma(n, shape = 8, rate = 2)
  stik2 <- rgamma(n, shape = 8, rate = 2)
  stik_diff <- stik1 - stik2
  obs_t <- (mean(stik_diff)-sand_middel)/(sd(stik_diff)/sqrt(n))

  boot_t <- c()
  for(i in seq(1, bootstraps)){
    boot <- sample(stik_diff, n, replace = TRUE)
    boot_t[i] <- (mean(boot)-mean(stik_diff))/(sd(boot)/sqrt(n))
  }

  antal_ekstreme <- abs(boot_t) >= obs_t

  p_vaerdi <- (sum(antal_ekstreme)+1) / (bootstraps + 1)

  andel <- p_vaerdi > 1 - konf_niveau
})

type_1_parret <- table(type_1_fejl_boot)
type_1_parret
```

```
## type_1_fejl_boot
## FALSE  TRUE
##    18   482
```

I alt forkastes H_0 fejlagtigt i 3.6% af tilfældene, hvilket ikke stemmer overens med det valgte signifikansniveau.

Selvom det virker godt, at bootstrap laver færre type-I fejl end signifikansniveauet antyder, er det ikke nødvendigvis en fordel. Hvis der laves en hypotesetest, hvor det forventede antal type-I fejl er lig signifikansniveauet, men det faktiske antal er noget andet, kan det lede til forkerte konklusioner.

6.4 Bootstrap-konfidensintervaller

I dette afsnit beskrives, hvorledes et konfidensinterval kan beregnes ved hjælp af bootstrap. Der vises tre forskellige metoder, kaldet henholsvis percentil-, basic- og T-metoden. Til sidst sammenlignes de tre metoders dækningsgrader ved forskellige stikprøvestørrelser.

Et 95% konfidensinterval på en normalfordelt estimator, Θ, kan udregnes ved $\text{KI} = [\hat{\theta} - 1.96 \cdot \text{se}(\hat{\theta}),\ \hat{\theta} + 1.96 \cdot \text{se}(\hat{\theta})]$, [Orloff and Bloom, 2014]. Hvis ikke fordelingen er kendt, er det ikke muligt at udregne et konfidensinterval således. Her er det i stedet muligt at benytte bootstrap til at udregne et konfidensinterval, hvilket kan gøres ved hjælp af forskellige metoder.

6.4.1 Percentilmetoden

En intuitiv fremgangsmåde til at bestemme et $(1-\alpha)100\%$ konfidensinterval ved percentiler, er at bruge det $(B(\frac{\alpha}{2}))$'te og $(B(1-\frac{\alpha}{2}))$'te percentil, hvor B er antallet af bootstrap-repetitioner. Lad $\Theta^* = [\vartheta_1, \ \ldots, \ \vartheta_B]$ være en sorteret liste af bootstrap-estimater. Derved er formlen for et konfidensinterval på baggrund af percentiler:

$$\text{KI}_p = \left[q_{(\alpha/2)}, \ q_{(1-\alpha/2)}\right]$$

Hvor q_i er det i'te percentil i Θ^*. [Lau et al., Unknown]

Eksempel

I kodetstykket nedenfor udtages en tilfældig stikprøve fra en standard normalfordeling. Herefter laves $10,000$ bootstrap-stikprøver, som middelværdien udregnes på. Til sidst sorteres middelværdierne i en liste.

```
data <- rnorm(100, mean = 0, sd = 1)

n <- length(data)
bootstraps <- 10000

bootstrap_fordeling <- replicate(n = bootstraps, {
  x <- mean(sample(data, size = n, replace = TRUE))
})

SortedBoots <- sort(bootstrap_fordeling)
```

Dernæst vælges konfidensniveauet til $\alpha = 0.05$, og konfidensintervallet ved hjælp af percentiler beregnes i nedenstående kodestykke.

```
KI_niveau <- 0.95
alpha <- 1 - KI_niveau

NedrePercentil <- round(bootstraps * alpha/2)
OevrePercentil <- round(bootstraps * (1-alpha/2))

KI_Percentil <- c(Nedre = SortedBoots[NedrePercentil],
                  Oevre = SortedBoots[OevrePercentil])

KI_Percentil
```

```
##       Nedre       Oevre
## -0.08445604  0.32382861
```

Percentilmetoden giver et konfidensinterval på $\text{KI}_p = [-0.084 \ , \ 0.324]$.

6.4.2 Basic-metoden

Denne type konfidensinterval ud fra bootstrap er også kendt som *reversed percentile interval*, der benytter nedenstående formel til udregning af konfidensintervaller.

$$\text{KI}_b = \left[2\hat{\theta} - q_{(1-\alpha/2)},\ 2\hat{\theta} - q_{(\alpha/2)}\right]$$

Hvor q_i er det i'te percentil i Θ^* og $\hat{\theta}$ er middelværdien af stikprøven, [Duke, Unknown]. Hermed følger udledningen af basic-metoden:

Proof. Fra percentilmetoden kendes følgende konfidensinterval

$$0.95 \approx P[q_{(\alpha/2)} \leq \hat{\theta}^* \leq q_{(1-\alpha/2)}]$$

Der trækkes $\hat{\theta}$ fra på alle sider af uligheden og der ganges igennem med -1, så uligheden vendes om

$$0.95 = P[q_{(\alpha/2)} - \hat{\theta} \leq \hat{\theta}^* - \hat{\theta} \leq q_{(1-\alpha/2)} - \hat{\theta}]$$

$$0.95 = P[\hat{\theta} - q_{(\alpha/2)} \geq \hat{\theta} - \hat{\theta}^* \geq \hat{\theta} - q_{(1-\alpha/2)}]$$

Differencen $\theta - \hat{\theta}$ er tilnærmelsevist den samme som $\hat{\theta} - \hat{\theta}^*$, hvilket nu indsættes i uligheden

$$0.95 \approx P[\hat{\theta} - q_{(\alpha/2)} \geq \theta - \hat{\theta} \geq \hat{\theta} - q_{(1-\alpha/2)}]$$

Dernæst lægges $\hat{\theta}$ til på alle sider af uligheden, og dermed fås konfidensintervallet

$$0.95 = P[2\hat{\theta} - q_{(\alpha/2)} \geq \theta \geq 2\hat{\theta} - q_{(1-\alpha/2)}]$$ □

Eksempel

I kodestykket nedenfor beregnes middelværdien af stikprøven og benyttes i formlen.

```
theta_hat <- mean(data)

KI_Basic <- c(Nedre = 2*theta_hat-SortedBoots[OevrePercentil],
              Oevre = 2*theta_hat-SortedBoots[NedrePercentil])

KI_Basic
```

```
##       Nedre       Oevre
## -0.08905211  0.31923255
```

Basic-metoden giver et konfidensinterval på $\text{KI}_b = [$`-0.089 , 0.319`$]$.

6.4.3 T-metoden

Denne type bootstrap-konfidensinterval, benytter nedenstående formel.

$$\text{KI}_t = \left[\hat{\theta} - t^*_{(1-\alpha/2)} \cdot \hat{\text{se}}(\theta),\ \hat{\theta} - t^*_{(\alpha/2)} \cdot \hat{\text{se}}(\theta)\right]$$

Hvor $\hat{\theta}$ er middelværdien af stikprøven, $t^* = \frac{\hat{\vartheta}-\hat{\theta}}{\hat{\text{se}}(\hat{\vartheta})}$ og $\hat{\vartheta}$ er middelværdien for Θ^*, [Lau et al., Unknown].

Beviset for formlen til konfidensintervallet for T-metoden, følger samme metode som i afsnittet for basic-metoden.

Eksempel

I kodestykket forneden beregnes t^* for det nedre og øvre percentil. Dernæst beregnes konfidensintervallet ud fra formlen.

```
T_Nedre <- (SortedBoots[OevrePercentil]-theta_hat) /
  (sd(bootstrap_fordeling))

T_Oevre <- (SortedBoots[NedrePercentil]-theta_hat) /
  (sd(bootstrap_fordeling))

KI_T <- c(Nedre = theta_hat - T_Nedre * (sd(data)/sqrt(n)),
               Oevre = theta_hat - T_Oevre * (sd(data)/sqrt(n)))
KI_T
```

```
##       Nedre       Oevre
## -0.09330652  0.32339224
```

T-metoden giver et konfidensinterval på $\text{KI}_t = [-0.093\ ,\ 0.323]$.

6.4.4 Dækningsgrader

I dette afsnit vil der undersøges, hvor præcise metoderne for at udregne konfidensintervallerne med bootstrap er. Dette gøres ved at udregne dækningsgraden af konfidensintervallerne som de forskellige metoder har produceret.

Som det første oprettes en funktion, der udregner middelværdien for stikprøven, og tager en stikprøve og et indeks som *input*.

```
# Estimator for middelværdi
meanFunc <- function(stik, i)
{    middel <- mean(stik$data[i])
     n <- length(i)
     varians <- (n-1) * var(stik$data[i]) / n^2
     c(middel, varians)
}
```

Det næste der udregnes er et konfidensinterval. Konfidensintervallet beregnes ved hjælp af funktionen `boot.ci`, der kommer fra pakken `boot`.

Der bliver først lavet 100 konfidensintervaller på baggrund af normalfordelte stikprøver. Så undersøges der om middelværdien af populationen er i konfidensintervallet. Hvis det er tilfældet, vil outputtet være `TRUE`, hvis ikke vil det være `FALSE`. Herefter indsættes disse output i en matrix.

Til sidst beregnes dækningsgraden som andelen af de intervaller, der indeholder den sande middelværdi. Der laves 100 konfidensintervaller for hver stikprøvestørrelse. Stikprøvestørrelsen er angivet i en sekvens fra 4 til 100 med spring på 4. Denne proces udføres for de tre metoder, percentil, T og basic.

Her udføres processen for beregning af dækningsgraden for alle tre metoder.

```
sand_middel <- 0

matriks_p <- matrix(ncol = 25, nrow = 100)
vector_p <- c()

matriks_b <- matrix(ncol = 25, nrow = 100)
vector_b <- c()

matriks_t <- matrix(ncol = 25, nrow = 100)
vector_t <- c()

for(n in seq(4, 100, 4)){
  res <- replicate(n = 100,{
    stik <- data.frame(data = rnorm(n))
    boot_stik <- boot::boot(data = stik, statistic = meanFunc,
                            R = 100)
    interval <- boot::boot.ci(boot.out = boot_stik,
                              conf = KI_niveau,
                              type = c("perc", "basic", "stud"))

    # Returnerer nedre og øvre grænse for konfidensintervallerne
    interval_p <- interval$percent[4:5]
    interval_b <- interval$basic[4:5]
    interval_t <- interval$student[4:5]

    # Undersøger om den sande middelværdi er i konfidensintervallet
    tf_p <- interval_p[1]<=sand_middel & interval_p[2]>=sand_middel
    tf_b <- interval_b[1]<=sand_middel & interval_b[2]>=sand_middel
    tf_t <- interval_t[1]<=sand_middel & interval_t[2]>=sand_middel
    return(c(tf_p, tf_b, tf_t))
  })
```

```
  # Dækningsgraden afhængigt af stikprøvestørrelsen
  matriks_p[,n/4] <- res[1,]
  vector_p <- append(vector_p, mean(matriks_p[,n/4]))

  matriks_b[,n/4] <- res[2,]
  vector_b <- append(vector_b, mean(matriks_b[,n/4]))

  matriks_t[,n/4] <- res[3,]
  vector_t <- append(vector_t, mean(matriks_t[,n/4]))
}

# Dækningsgraden for percentilmetoden
vector_p[1:5]
```

```
## [1] 0.78 0.84 0.89 0.89 0.96
```

I *vector_p* ses middelværdierne for dækningsgraden af 100 konfidensintervaller genereret ud fra percentilmetoden. Den første middelværdi er for stikprøvestørrelsen $n = 4$, den næste er for stikprøvestørrelsen $n = 8$ osv.

Dækningsgraderne for de tre metoder illustreres på figur 6.5.

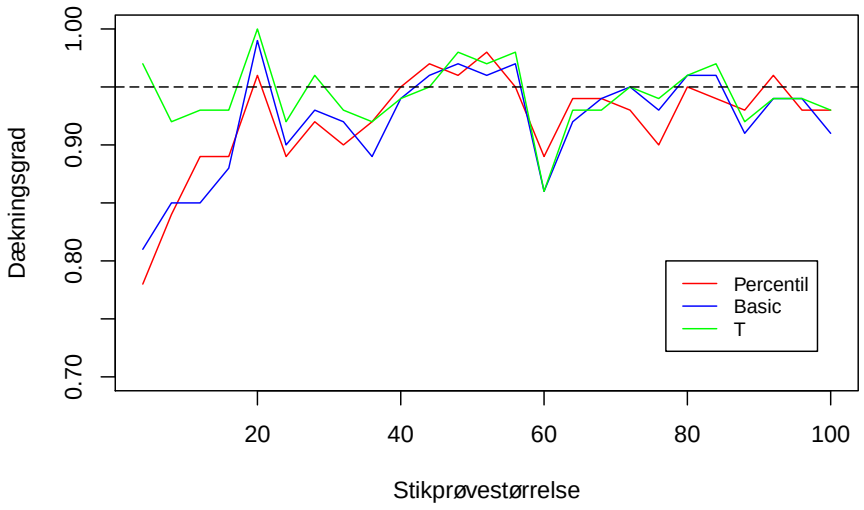

Figure 6.5: Dækningsgraden for de tre bootstrap-konfidensintervaller. På x-aksen ses stikprøvestørrelsen, mens på y-aksen ses dækningsgraden af konfidensintervallet på baggrund af den givne stikprøvestørrelse.

Her tyder det på, at T-metoden præsterer bedre ved mindre stikprøver end både percentilmetoden og basic-metoden. Ved tilpas stor stikprøvestørrelse opnår percentil- og basic-metodens konfidensinterval begge den samme dækningsgrad som T-metoden, og generelt kan der ses, at percentil- og basic-metoden har cirka den samme dækningsgrad. Både percentil- og basic-metoden ser ud til at have større udsvingninger end T-metoden. Desuden ser alle tre metoders dækningsgrader ud til at stabilisere sig omkring 95%, når stikprøvestørrelsen bliver tilstrækkelig stor.

Chapter 7

Diskussion

I dette afsnit vil nogle relevante resultater og opdagelser fra rapporten diskuteres.

En væsentlig faktor, som ville have forbedret projektet samt analyserne, er betydeligt bedre computerkraft. I større analyser, hvor antal gentagelser har været essentielle for præcise resultater, har projektet været nødsaget til at reducere disse gentagelser, da computerkraften ikke var tilstrækkelig og derved medført timelange simuleringsanalyser. Computere med nødvendige ressourcer til at gennemløbe disse analyser, ville have hjulpet til at højne samt præcisere resultaterne. Dette giver sig bland andet til udtryk i rapporten ved antallet af bootstrap-stikprøver der bliver beregnet. Selvom der i afsnit 6.1 anbefales et antal bootstrap-stikprøver på $10,000$, er der i rapporten valgt at reducere antallet af bootstrap-stikprøver i nogle tilfælde af hensyn til computerkraft og beregningstid. Dette kunne give anledning til større usikkerhed i de opnåede resultater, da et for lille antal bootstrap-stikprøver kan medføre, at fordelingen af bootstrap-stikprøvernes estimater ikke tilnærmer sig fordelingen af stikprøveestimatet jævnfør figur @ref{fig:fig-boot-fordeling}.

Der blev i rapporten undersøgt andelen af type-I fejl, for både permutationstest og bootstrap-test, tilsvarende blev dog ikke undersøgt for type-II fejl. Type-II fejl er nært umulige at afdække da det afhænger af mange variabler såsom stikprøvestrørrelsen og populationens parametre. Derfor blev andelen af type-II fejl ikke undersøgt yderligere, da det kun ville have givet information for de specifike tilfælde, hvilket ikke giver mening i den teoretiske tilgang som rapporten beskæftiger sig med.

I kapitel 4 blev det vist, at en uparret t-tests konfidensinterval for ikke-normalfordelte stikprøver kunne anses for at være misvisende i forhold til dets dækningsgrad. En af antagelserne for t-testen er netop at populationen, som stikprøverne (eller stikprøven) stammer fra, er normalfordelt, som i dette tilfælde ikke blev overholdt. En observation, der blev gjort i forbindelse med

undersøgelsen af den uparret t-test var, at for at få ikke holdbare resultater, krævede det to forholdsvist skæve, men ikke sammenlignelige, stikprøver. Derfor blev der også udtrukket stikprøver fra to forskellige fordelinger, henholdsvis en beta- og gammafordeling.

Holdbarheden af t-testens konfidensintervaller er robust på trods af, at populationerne ikke er normalfordelte. T-testen viser sig altså at være meget robust overfor overtrådte antagelser, og dette må skyldes den centrale grænseværdisætning, CLT (fra engelsk *central limit theroem*).

CLT medfører, at fordelingen af stikprøvemiddelværdierne tilnærmelsesvist ligner en normalfordeling. Selv ved relativt lave stikprøvestørrelser, $n = 5$, er dette aktuelt, [Agresti and Finlay, 2014, side 93, figur 4.15]. At komme ud for en situation, hvor der estimeres middelværdier som i denne rapport, uden at CLT vil have en indflydelse, er usandsynligt. Dette er en fordel i forhold til t-testen, da den netop antager en normalfordeling.

Var stikprøverne i kapitel 4 i stedet udtrukket fra to fordelinger, der lignede hinanden, ville dækningsgraden have været forholdsvis tæt på de 95% (se eksempel 1).

Eksempel 1

I dette eksempel påvises dækningsgraden for to stikprøver udtrukket fra to forskellige betafordelinger ved hjælp af en t-test. Den ene stikprøve er udtrukket fra en betafordeling med $\alpha = 8$ og $\beta = 2$. Den anden stikprøve er udtrukket fra en betafordeling med $\alpha = 2$ og $\beta = 8$.

```
## Daekningsgrad
## FALSE  TRUE
##   568  9432
```

For to stikprøver udtrukket fra hver deres betafordeling, en venstreskæv og en højreskæv, ses en dækningsgrad på konfidensintervallet fra t-testen på `94.32`%, hvilket er passende tæt på de antaget 95%.

Chapter 8

Konklusion

I dette afsnit vil der konkluderes på den viden der er opnået gennem rapporten.

I afsnittet der omhandlede "t-test for skæve stikprøver" blev det konkluderet, på baggrund af den resulterende dækningsgrad på `92.06`%, at det ikke kan antages, at en t-test på en ikke-normalfordelt stikprøve altid giver retvisende resultater.

Der blev herefter fundet frem til to altenative metoder til t-testen, som kunne lave konfidensintervaller og hypotesetests, hvis antagelserne ikke var overholdt. De to var henholdsvis bootstrap-metoden og permutationstesten.

Permutationstesten løste problemet med, at stikprøverne ikke var normalfordelte, og at der så kunne udføres en retvisende hypotesetest. Det fremgik, at andelen af type-I fejl, `5.6`%, stemte overens med det valgte signifikansniveau, 5%.

Hypotesetest ved hjælp af bootstrap fremgik at have færre type-I fejl end antaget, hvor H_0 fejlagtigt forkastes i `3.6`% af tilfældene. Permutationstesten er altså mere præcis end bootstrap, hvis der udelukkende skal foretages en hypotesetest.

Bootstrap-metoden løste det samme problem og det viste sig, at kunne lave konfidensintervaller ved hjælp af percentilmetoden, basic-metoden og T-metoden. Disse metoder viste, at dækningsgraden af T-metoden er den mest robuste metode. Dog når stikprøvestørrelsen er tilpas stor, er det lige meget, hvilken metode der bruges, fordi de alle stort set har den samme dækningsgrad ifølge figur 6.5. Det kan konkluderes, at bootstrap-metoden kan give retvisende resultater, når antagelserne ikke er overholdt.

Det kan derfor siges, at simuleringsmetoderne bootstrap og permutation, kan afhjælpe problemer der opstår ved t-tests, med ikke-normalfordelte stikprøver.

Chapter 9

Perspektivering

Simuleringsstudier medfører mange retninger projektet kunne have inddraget, såvel som mere dybdegående analysemetoder. Dette er dog undladt, blandt andet, på baggrund af manglende computerkraft samt begrænset tid.

Det kunne have været interesseant at undersøge om permutationer kunne bruges til at lave konfidensintervaller og sammenligne dækningsgraden af disse med bootstrap.

Da det blev undersøgt, hvordan bootstrap kunne benyttes til at beregne standardfejl blev der kun givet et eksempel på standardfejlen for middelværdi. Her kunne det have været relevant at lave udregningen for en estimator, hvis standardfejl ikke umiddelbart let kunne beregnes.

I rapporten benyttes udelukkende ikke-parametrisk bootstrap. Der findes dog også parametrisk bootstrap, som skiller sig ud fra den ikke-parametriske bootstrap, ved at den antager at stikprøven stammer fra en specifik fordeling, F, der afhænger af en specifik parameter, θ. Stikprøven har da punktestimatet $\widehat{\theta}$, og det er derved muligt at simulere bootstrap-stikprøver fra fordelingen, $F(\widehat{\theta})$. Punktestimatet, $\widehat{\theta}$, estimeres oftest ved hjælp af *maximum likelihood estimation* (MLE). Det kunne have været interessant at undersøge, hvordan begge typer bootstrap ville håndtere samme stikprøver og sammenligne resulaterne. For bedst muligt at kunne anvende parametrisk bootstrap, anvendes MLE til at estimere parameteren, θ. Derfor er en forståelse af MLE en forudsætning, der også skulle have været inddraget i projektet, for at kunne have anvendt parametrisk bootstrap.

Bibliography

The Applied Statistics Team AAU. Asta - lektion 1.1, 2020a. URL https://asta.math.aau.dk/course/asta/2020-1/std/lecture/1-1.

The Applied Statistics Team AAU. Asta - lektion 1.2, 2020b. URL https://asta.math.aau.dk/course/asta/2020-1/std/lecture/1-2.

The Applied Statistics Team AAU. Asta - lektion 1.3, 2020c. URL https://asta.math.aau.dk/course/asta/2020-1/std/lecture/1-3.

The Applied Statistics Team AAU. Asta - lektion 2.1 hypothesis test, 2020d. URL https://asta.math.aau.dk/course/asta/2020-1/std/lecture/2-1?file=B/lecture-B-1.html#(1).

Alan Agresti and Barbara Finlay. *Statistical Methods for the Social Sciences*. Pearson, Edinburgh Gate, England, 4th edition, 2014. ISBN 978-1-292-02166-9.

Jonathan Bartlett. The miracle of the bootstrap, 2013. URL https://thestatsgeek.com/2013/07/02/the-miracle-of-the-bootstrap/.

Daniel Berrar. Introduction to the non-parametric bootstrap, 2019. URL https://www.researchgate.net/publication/332553015_Introduction_to_the_Non-Parametric_Bootstrap.

Angelo Canty and Brian Ripley. *boot: Bootstrap Functions (Originally by Angelo Canty for S)*, 2020. URL https://CRAN.R-project.org/package=boot. R package version 1.3-25.

Laura M Chihara and Tim C. Hesterberg. *Mathematical Statistics with Resampling and R*. John Wiley and Sons, Inc, Hoboken, NJ, 2nd edition, 2019a. ISBN 9781119416524.

Laura M. Chihara and Tim C. Hesterberg. *Mathematical Statistics with Resampling and R*. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA, 2nd edition, 2019b. ISBN 9781119416524.

Den Store Danske. Statistik, 2020. URL http://denstoredanske.dk/Samfund,_jura_og_politik/Samfund/Samfund_og_statistik/statistik.

Online Etymology Dictionary. Statistics, 2020. URL https://www.etymonline.com/word/statistics?fbclid=IwAR24xfhlEU8QlZQWFlc8zdzbupnl-HDFTIdr8y4HwH637e5yP1XO9Lh3dqs.

Cambridge Dictonary. Simulation, 2020. URL https://dictionary.cambridge.org/dictionary/english/simulation.

Stat Duke. Bootstrap confidence intervals, Unknown. URL http://www2.stat.duke.edu/~banks/111-lectures.dir/lect13.pdf.

Minitab Blog Editor. Understanding hypothesis tests: Why we need to use hypothesis tests in statistics, 2015. URL https://blog.minitab.com/blog/adventures-in-statistics-2/understanding-hypothesis-tests-why-we-need-to-use-hypothesis-tests-in-statistics.

Mercedes AMG F1. Simulator, 2020. URL https://careers.mercedesamgf1.com/facilities/simulator/.

Morten Frydenberg. Epidemiologi og biostatistik, 2001. URL https://www.biostat.au.dk/teaching/praegrad/for%C3%A5r2001/forelaesninger/biostatistik/uge2torsdag.pdf.

Aerin Kim. Gamma distribution — intuition, derivation, and examples, 2019. URL https://commons.wikimedia.org/w/index.php?curid=38637545.

Aerin Kim. Beta distribution — intuition, examples, and derivation, 2020. URL https://towardsdatascience.com/beta-distribution-intuition-examples-and-derivation-cf00f4db57af.

Sam Lau, Joey Gonzalez, and Deb Nolan. Principles and techniques of data science, Unknown. URL https://www.textbook.ds100.org/ch/18/hyp_studentized.html.

Mike Marin. Bootstrapping and resampling in statistics with example| statistics tutorial # 12 |marinstatslectures, 2018. URL https://www.youtube.com/watch?v=O_Fj4q8lgmc&list=PLqzoL9-eJTNDp_bWyWBdw2ioA43B3dBrl&index=2&t=1s.

Matematikcenter. Chi i anden-test, 2020. URL https://www.webmatematik.dk/lektioner/matematik-b/statistik/chi-i-anden-test.

Jay Myung. Bootstrap hypothesis testing, 2015. URL https://faculty.psy.ohio-state.edu/myung/personal/course/826/bootstrap_hypo.pdf.

The Editors of Encyclopaedia Britannica. Sir ronald aylmer fisher, 2020. URL https://www.britannica.com/biography/Ronald-Aylmer-Fisher.

Jeremy Orloff and Jonathan Bloom. Bootstrap confidence intervals, 2014. URL https://ocw.mit.edu/courses/mathematics/18-05-introduction-to-probability-and-statistics-spring-2014/readings/MIT18_05S14_Reading24.pdf.

The Numerical Algorithms Group Ltd Oxford. Nag toolbox chapter introduction — random number generators, 2015. URL https://www.nag.co.uk/numeric/mb/nagdoc_mb/manual_25_1/html/g05/g05intro.html.

Roger D. Peng. R programming for data science, 2019. URL https://bookdown.org/rdpeng/rprogdatascience/simulation.html.

Theodore M. Porter. Karl pearson, 2020. URL https://www.britannica.com/biography/Karl-Pearson.

Randall Pruim, Daniel T. Kaplan, and Nicholas J. Horton. *mosaic: Project MOSAIC Statistics and Mathematics Teaching Utilities*, 2020. URL https://CRAN.R-project.org/package=mosaic. R package version 1.6.0.

Nicole Radziwill. A linear congruential generator (lcg) in r, 2015. URL https://qualityandinnovation.com/2015/03/03/a-linear-congruential-generator-lcg-in-r/#comments.

ASTA Team. Hypothesis test, 2020. URL https://asta.math.aau.dk/course/asta/2020-1/std/lecture/2-1?file=B/lecture-B-1.pdf.

Bruce E. Trumbo. Congruential generators of pseudorandom numbers, 2005. URL http://www.sci.csueastbay.edu/~btrumbo/Stat3401/Hand3401/CongGenIntroB.pdf.

Neil A. Weiss. *wPerm: Permutation Tests*, 2015. URL https://CRAN.R-project.org/package=wPerm. R package version 1.0.1.

Eric W. Weisstein. Box-muller transformation, 2020a. URL https://mathworld.wolfram.com/Box-MullerTransformation.html.

Eric W. Weisstein. Permutation, 2020b. URL https://mathworld.wolfram.com/Permutation.html.

Rick Wicklin. The average bootstrap sample omits 36.8 URL https://blogs.sas.com/content/iml/2017/06/28/average-bootstrap-sample-omits-data.html.

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2020. URL https://CRAN.R-project.org/package=bookdown. R package version 0.18.

Lorna Yen. An introduction to the bootstrap method, 2019. URL https://towardsdatascience.com/an-introduction-to-the-bootstrap-method-58bcb51b4d60.