Olav Gravir Imenes
OsloMet – storbyuniversitetet
Vibeke Bjarnø
OsloMet – storbyuniversitetet
Ove Edvard Hatlevik
OsloMet – storbyuniversitetet

DOI: https://doi.org/10.5617/adno.8180

Lærerstudenters bruk av Monte Carlo-simulering for å løse oppgaver om sannsynlighet: En analyse av 16 gruppebesvarelser

Sammendrag

Programmering er tatt inn i den nye læreplanen for matematikk i grunnskolen i Norge. Det betyr at lærerstudenter har behov for å få erfaring med å løse matematiske problemer gjennom programmering. I matematikkfaget på lærerutdanninga for 1.–7. trinn ble det laga et undervisningsopplegg som omfatta opplæring i sannsynlighet og Monte Carlo-simulering med programmering i Excel med Visual Basic for Applications (VBA). Arbeidskravet innebar bruk av Monte Carlo-simulering for å løse Chevalier de Méré-problemet og Monty Hall-problemet. I etterkant av studentenes arbeid ble det utforma en NSD-godkjent studie. Utvalget i denne studien er 16 studentgruppers besvarelser på arbeidskravet knytta til dette undervisningsopplegget innen dataprogrammering i matematikkfaget. Et funn fra studien er at småfeil kan skape store problemer ettersom mange studenter ikke klarer å vurdere hvor fornuftige de svarene programmet gir er. I tillegg gir manglende systematikk feilsvar. Men i de tilfellene der studentene klarer å programmere rett, hjelpes de til å løse Chevalier de Méré-problemet. Vi finner også at studentene kan få hjelp av manuell Monte Carlo-simulering for å løse Monty Hall-problemet, gitt at denne gir tallverdier som ligger nært forventningsverdien ($p = 2/3$), mens i de tilfellene hvor tallverdiene ligger langt unna forventningsverdien kan det virke forvirrende. Det er fordeler og ulemper med både manuell og digital Monte Carlo-simulering, og det ser ut til at lærerstudenter kan ha nytte av å løse oppgaver ved hjelp av begge metoder. For å få det beste læringsutbyttet er det avgjørende at læreren velger gode og relevante oppgaver, som gjør at studentene både ser nytten av simuleringa, og også har en viss mulighet til å kontrollere svaret, slik at ikke tilfeldighet under simuleringa og programmeringsfeil bidrar til forvirring.

Nøkkelord: lærerstudenter, sannsynlighet, programmering, Monte Carlo-simulering, Chevalier de Méré-problemet, Monty Hall-problemet

Student teachers' use of Monte Carlo simulation to solve probability problems: An analysis of 16 group answers

Abstract

Programming is included in the new curriculum for mathematics in Norwegian primary school. This means that student teachers need to gain experience in solving mathematical problems through programming. As a part of the subject of mathematics in the general teacher education program (grades 1–7), training in probability and Monte Carlo simulation with programming in Excel with Visual Basic for Applications (VBA) was included. A mandatory assignment involved the use of Monte Carlo simulation to solve the Chevalier de Méré problem and the Monty Hall problem. Following the students' work, an NSD-approved study was designed. The sample in this study is 16 student teacher groups' answers to the assignment related to programming and probability in the subject of mathematics. One finding from the study is that small errors may create major problems as some student teachers are not able to assess how sensible the answers that the program provides are. In addition, the lack of systematics gives incorrect answers. But in those cases where student teachers are able to program correctly, they are helped to solve the Chevalier de Méré problem. We also find that student teachers may get help from manual Monte Carlo simulation to solve the Monty Hall problem, given that it gives numerical values that are close to the expected value ($p = 2/3$), while in those cases where the numerical values are far from the expected value it may seem confusing. There are pros and cons to both manual and digital Monte Carlo simulation, and it seems that student teachers can benefit from solving problems using both methods. In order to get the best learning outcome, it is crucial that the teacher chooses good and relevant tasks, which means that the students both see the benefit of the simulation, and also have a certain opportunity to check the answer, so that randomness in the simulation and programming errors do not confuse.

Keywords: student teachers, probability, programming, Monte Carlo simulation, Chevalier de Méré problem, Mony Hall problem

Bakgrunn og problemstilling

Høsten 2017 starta implementeringa av de 5-årige grunnskolelærerutdanningene i Norge (KD, 2016a, 2016b). Formålet var å sette lærerstudentene i stand til å møte framtidas læreplaner i skolen. Statsminister Erna Solberg uttalte at forståelse for koding og teknologi skal inn i læreplanene allerede på barneskolen. Det innebærer en økt satsing på å gi alle elever opplæring i teknologi og programmering, for eksempel «å forstå og håndtere algoritmisk tenkemåte og programmering» (Regjeringen, 2017). I de nye læreplanene for grunnskolen, *Fagfornyelsen*, er dette fulgt opp i flere fag, deriblant matematikk (Udir, 2019a). Under digitale ferdigheter presiseres det at en måte å utforske og løse matematiske problemer på er ved bruk av regneark, programmering med mer. Læreren og elevene skal gjennom skoletida være i dialog om elevers utvikling i programmering. Det skal legges grunnlag for denne type tenkning helt fra tidlige trinn,

samt at programmering er nedfelt i egne kompetansemål fra 6. til og med 10. trinn i grunnskolen. Følgende to kompetansemål er relevante for grunnskolelærerutdanninga for 1.–7. trinn:

> Etter 6. trinn skal elevene kunne bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre.
> Etter 7. trinn skal elevene kunne bruke programmering til å utforske data i tabeller og datasett.

Matematikklærere må både være i stand til å vise elever ulike løsninger på statistiske problemer og kunne demonstrere enkle prinsipper for programmering (Udir, 2019a). Alle studenter på grunnskolelærerutdanninga for 1.–7. trinn har matematikk som fag, og det er derfor viktig at de får erfaring med å anvende programmering for å løse matematikkoppgaver.

Denne artikkelen er basert på data fra et prosjekt som kobla dataprogrammering til matematikkfaget på lærerutdanninga for 1.–7. trinn. Bakgrunnen for prosjektet var ønsket om å gi studentene et verktøy for å finne svar på spørsmål fra matematikkfaget som det er vanskelig for studentene å løse analytisk. Det ble valgt å anvende *Monte Carlo-simuleringer* for å lære programmering og simulering. Monte Carlo-simulering er et verktøy for å gi svar på problemer i sannsynlighet som er kompliserte å løse eksakt. Man gir en datamaskin sannsynlighetene eller sannsynlighetsfordelinga for enkeltdelene av problemet, og i stedet for at man regner ut løsninga, gjør datamaskinen forsøk der utfallene er tilfeldige, men basert på sannsynlighetene eller sannsynlighetsfordelinga for de enkelte hendelsene. Det å lære seg Monte Carlo-simuleringer kan gi studenter flere fordeler for å lære seg og forstå statistikk (Sigal & Chalmers, 2016; Weltman & Tokar, 2019). Vi antar at det å skrive litt programkode også kan gi studentene mulighet til å få et bedre bilde av hvordan man kan angripe problemløsingsoppgaver, samtidig som de får mulighet til å bruke sin digitale kompetanse praktisk. For eksempel, i stedet for å regne ut sannsynligheten for å få en dobbeltsekser hvis man kaster to terninger, så kan man programmere datamaskinen til å kaste to terninger tusen ganger. Vi gir datamaskinen sannsynligheten for å slå 6 i et tilfeldig terningkast, som er 1/6, men den trenger ikke å få vite noe mer, for eksempel det at sannsynlighetene skal multipliseres sammen når man gjør to uavhengige forsøk. Poenget er her at datamaskinen teller opp antall dobbeltseksere, som for eksempel kan være 28 av 1000 ganger, og antall ganger det ikke blir dobbeltseksere, i stedet for å bruke mer avanserte regneoperasjoner. Forløperen til Monte Carlo-simulering ble først utvikla og brukt i forbindelse med det amerikanske atombombeprosjektet under og etter andre verdenskrig, ettersom det først da var noen som var villige til å betale for tilstrekkelig sterke elektromekaniske datamaskiner, og seinere elektroniske (Eckhardt, 1987; Metropolis, 1987). Monte Carlo er et kodeord siden prosjektet var hemmelig. Monte Carlo-simuleringer kan i prinsippet gjennomføres manuelt, for eksempel ved bruk av terninger, eller automatisk med programmering. Vi antar at det å gjøre denne

typen simuleringer manuelt kan bidra til at studentene enklere forstår hva Monte Carlo-simulering er og bidra til at studentene forstår hvor arbeidskrevende det er å gjennomføre tilstrekkelig mange repetisjoner til å få nok informasjon for å finne løsninga på oppgaven.

Sannsynlighet er et av de områdene som i løpet av de siste tre tiårene har kommet inn i læreplanen i matematikk i Norge. Sannsynlighet kom inn i læreplanen (L97) gjeldende fra 1997 (KUF, 1996) og ble videreført i læreplanen (LK06) som kom i 2006 (Udir, 2006). Et poeng med sannsynlighetsundervisninga er at elever og studenter skal kunne vurdere sannsynligheter i dagliglivet, og da vil dette gjerne ta form som uoppstilte problemer. For mange elever og studenter kan det generelt være vanskelig å løse uoppstilte matematikkproblemer eller problemer formulert med ord (word problems) (Daroczy er al., 2015). Prosjektet omfatta to problemer. Problem 1 er om man trenger 24 eller 25 kast med to terninger for å ha over 50 % sannsynlighet for å få minst én dobbeltsekser. Dette er kjent som Chevalier de Méré-problemet (Derriennic, 2003; Maxwell, 1999; Ore, 1960). Det er vanskelig å løse problemet intuitivt, og manuell kasting av terninger er tidkrevende. Simulering ved hjelp av dataprogrammering ga lærerstudentene mulighet for å undersøke hva som skjedde når de lot dataprogrammet utføre mange tusen kast av to terninger. Nærmere forklaring av Chevalier de Méré-problemet kommer etter delen om teoretiske perspektiv og tidligere forskning.

Problem 2 er et sannsynlighetsparadoks. Bør man endre det opprinnelige valget når man får ny informasjon? Studentene blir presentert for et valg som allerede er gjort, men får mer informasjon enn den som gjorde valget. Ut fra den nye informasjonen, så fikk studentene som oppgave å vurdere om de ville endre det opprinnelige valget. Sannsynlighetsparadokset vi eksemplifiserte dette med er kjent som Monty Hall-problemet (Selvin, 1975; Shermer, 2009; Sprenger, 2010). Det er et kontra-intuitivt problem, og derfor kan muligens Monte Carlo-simulering ha bidratt til at lærerstudentene fikk en forståelse av hvordan problemet kan løses. Nærmere forklaring av Monty Hall-problemet kommer etter delen om teoretiske perspektiv og tidligere forskning.

Formålet med artikkelen er å undersøke hvordan lærerstudenter konkluderer ved løsing av problemer i sannsynlighet med og uten Monte Carlo-simulering. Faglærers retting av besvarelser, før forskningsprosjektet, ga en antakelse om at programmeringsfeil eller manglende systematikk hadde betydning for løsning. På bakgrunn av dette er det utforma to problemstillinger for studien:

1. Hvordan influerer programmeringsfeil eller manglende systematikk ved bruk av Monte Carlo-simulering på løsninga av Chevalier de Méré-problemet?
2. Hvordan konkluderer studentgrupper under løsninga av Monty Hall-problemet før de lærer Monte Carlo-simulering, og etter? Er det en forskjell, og i hvilken grad blir studentene hjulpet av Monte Carlo-simulering?

Teoretiske perspektiver og tidligere forskning

Forskning påpeker at mange studenter oppfatter emner med sannsynlighet og statistikk som en samling formler de skal pugge (Cheong et al., 2019). Ren memorering kan gå på bekostning av det å utvikle forståelse for faget og det å være opptatt av hva som læres (Gilje et al., 2018). Wood (2005) mener at det å bruke simulering kan bidra til studentaktive læringsformer gjennom praktisk arbeid med statistiske problemer, og at dette igjen kan bidra til utvikling av studenters forståelse av hvordan man kan løse noen statistiske problemer. Sigal og Chalmers (2016, s. 136) beskriver hvordan «computer-driven activities is an effective way to help students crystallize their knowledge about abstract statistical concepts». De mener at praktiske øvelser, for eksempel gjennom Monte Carlo-simulering, gir studentene bedre muligheter for utvikling av forståelse sammenliknet med pugg av metoder og teorier. Det å utvikle forståelse av hva som læres er også et sentralt poeng innen aktuelle perspektiver på dybdelæring (Gilje et al., 2018).

Et spørsmål er om programmering kan bidra til dybdelæring i matematikk – dybdelæring forstått som å lære noe så godt at det fremmer sammenhenger og kan brukes i nye situasjoner (Udir, 2019c). «Dybdelæring innebærer at elevene gradvis utvikler sin forståelse av begreper og sammenhenger innenfor et fag eller på tvers av fag» (KD, 2015–2016, s. 33). Blant kjennetegnene er at elevene ser på underliggende prinsipper (Sawyer, 2014, s. 5) og kan overføre det de har lært fra en situasjon til en annen (KD, 2015–2016, s. 33). Slik sett kan muligens oppnådd dybdelæring signaliseres av at elevene kan overføre det de lærer om programmering av en oppgave til en annen, samt se sammenhengen mellom de svarene de får når en oppgave løses analytisk versus når oppgaven løses ved hjelp av for eksempel Monte Carlo-simulering, som er en numerisk metode. Det er flere matematiske tankeprosesser som ender opp i algoritmer. I skolen ønsker man å la elevene selv få komme fram til algoritmen på bakgrunn av sin forståelse. Dette vil tilsvare å utvikle programmene, men ikke omvendt; med andre ord er det viktige at elevene selv utvikler programmene, ikke at de får dem levert, for at de skal få dybdelæring. Samtidig er det som kalles testing og feilsøking (debugging) en av de viktigste problemløsende ferdighetene i programmering (Lye & Koh, 2014, s. 58) og noe man kan lære mye av (Lye & Koh, 2014). Lærere bør ha en ekstra kompetanse i dette, ettersom de skal hjelpe elever til å finne ut hvorfor programmene deres ikke virker hvis det er gjort en feil. Å kunne foreta en kodegjennomgang (code review) er en måte å avdekke feil og utføre kvalitetssikring på koding (Indriasari et al., 2020). Kodegjennomgang regnes som en viktig ferdighet for programutviklere, fordi kodegjennomgang kan bidra til bedret kvalitet på koding (Rigby & Bird, 2013).

I matematikk snakker en gjerne om matematisk intuisjon. Poincaré (1905) beskriver intuisjon og logikk i matematikk som motsatte tendenser, eller tenkning, men skriver samtidig at begge typer tenkning er nødvendige for vitenskapelig

framgang. Burton (1999, s. 31) intervjuet 70 matematikere, og de fleste av dem så på intuisjon som en nødvendig komponent for å utvikle kunnskap, uten at de var presise med hva de mente med intuisjon. Noen sidestilte intuisjon med innsikt, mens andre mente det var ulike begreper (Burton, 1999, s. 28). Niss og Jensen (2002, s. 64) tenker ikke på matematisk intuisjon, som de også tenker på som matematisk kreativitet, som en selvstendig kompetanse, men som en kombinasjon av flere av de andre kompetansene. Samtidig ser det ut som at Niss og Jensen tenker seg en mye klarere sammenheng mellom formelle resonnementer og intuisjon enn det Poincaré gjør, ettersom de tenker seg at resonnementskompetanse består i å kunne tenke ut og gjennomføre formelle resonnementer på bakgrunn av intuisjon (Niss & Jensen, 2002, s. 93). Matematisk intuisjon vil vi dermed ikke si at er presist definert, men det kan sies å være det å ha en god forståelse av hvilke løsingsmetoder man kan angripe et gitt problem med, og en evne til å vurdere om svaret en får er rimelig, samt å se sammenhenger mellom de forskjellige temaene. Det kan muligens se ut som om begrepet dybdelæring innen matematikk kan sees på som en formalisering av matematisk intuisjon.

Manuelle kontra programmerte simuleringer

Det kan også være noen utfordringer knytta til bruk av Monte Carlo-simulering, for eksempel når det gjelder overføring av prinsipper til den praktiske simuleringssituasjonen og når det gjelder hvordan studentene skal utføre simuleringa. Studenter kan oppleve utfordringer knytta til både manuell simulering og simulering med programmering. Når det gjelder simulering med programmering, må studentene beherske enkel programmering; i dette tilfellet ble regneark brukt. Når det gjelder manuell simulering, må studentene gjøre de manuelle simuleringene mange nok ganger, noe som raskt blir svært utfordrende.

Scherer et al., (2019) har gjennomført en metastudie av forskning på erfaringer med dataprogrammering. De finner og analyserer 105 aktuelle studier. De mener at det er kognitive fordeler knytta til å lære dataprogrammering og at dette bidrar til diskusjon om å forstå dataprogrammering som en form for problemløsing.

Misoppfatninger knytta til sannsynlighet

Innen sannsynlighet er det flere vanlige misoppfatninger. Her redegjøres det for de som er mest relevante for denne undersøkelsen. En vanlig misoppfatning er å tenke seg alle utfall som like sannsynlige, uten å basere seg på hva som ligger bak hvert utfall (Fischbein & Schnarch, 1997). Videre er estimering av betinget sannsynlighet et problem, spesielt hvis det kommer flere opplysninger etter hverandre i kronologisk rekkefølge (Fischbein & Schnarch, 1997). Dette gir seg utslag i at nye opplysninger som kommer til etter at det opprinnelige valget er tatt, som for eksempel døra som åpnes i Monty Hall-problemet, ikke tillegges vekt. En annen ting er å addere sannsynligheter der det ikke skal gjøres, for eksempel å anslå at det er 1/6+1/6 sjanse for å få minst én sekser ved kast av to terninger (Watson, 2005).

Misoppfatninger i programmering

Når vi ser på vanlige misoppfatninger blant nybegynnere i programmering, fokuserer vi på feil som gjør at studentene ikke får opp noen direkte feilmeldinger, men får et feil svar. Dermed ser vi ikke vanlige syntaksfeil, som faktisk er de mest vanlige (Qian & Lehman, 2017). Løkkekonstruktet er vanskelig for studenter, og det er blant annet misforståelser om hvilke linjer som vil bli repetert (Qian & Lehman, 2017, s. 1–5; Sleeman et al., 1986). For eksempel, hvis det er flere linjer i en løkke, kan noen studenter tro at det bare er den linja som endrer det som står på skjermen som gjentas. Når funksjoner brukes, kan studentene bli forvirret over hvor parameterverdien kommer fra og hvor dataene skrives til (Qian & Lehman, 2017, s. 1–5; Ragonis & Ben-Ari, 2005). Mange tror at et program er delvis korrekt hvis noe av koden er skrevet riktig (Kolikant & Mussai, 2008; Lister et al., 2006; Qian & Lehman, 2017, s. 1–7).

Beskrivelse av arbeidskrav om sannsynlighet med simulering

ProTed – Senter for fremragende lærerutdanning ved Institutt for lærerutdanning og skoleforskning ved Universitetet i Oslo, fikk i 2016 i oppdrag fra Kunnskapsdepartementet å lede et prosjekt med mål om å finne gode eksempler på studieintensive tiltak i grunnskolelærerutdanningene (ProTed, 2016). Prosjektet *STIL – Studentaktiv læring i lærerutdanningene* ble initiert. OsloMet var en av de deltakende lærerutdanningsinstitusjonene, og prøvde blant annet ut simulering i matematikkfaget med førsteårsstudenter. Det ble laga et undervisningsopplegg som omfatta opplæring i sannsynlighet, Monte Carlo-simulering med programmering i Excel med Visual Basic for Applications (VBA). Studentene fikk et notat med en introduksjon til Monte Carlo-simulering. Dette inneholdt relevante eksempler på programmering i Excel, slik at studentene kunne finne inspirasjon. Men ingen av eksemplene inneholdt konkret løsning av Chevalier de Méré- og Monty Hall-problemene. Det ble også utforma et arbeidskrav til studentene (se utdrag i vedlegg). Arbeidskravet skulle gjennomføres som en gruppeoppgave. Det var faglærer som retta arbeidskravet med godkjent/ikke godkjent.

Arbeidskravet omfatta to ulike problemer innen sannsynlighet. Arbeidskravet hadde oppgaver og spørsmål knytta til Chevalier de Méré-problemet og Monty Hall-problemet. Kravet var utforma slik at studentene skulle finne løsning med bruk av intuisjon, altså nesten uten regning, løsning gjennom manuell simulering og med programmering ved bruk av Excel. Det å skrive kode er i utgangspunktet proseduralt, og spesielt innen Monte Carlo-simulering kan man ofte omformulere problemet på en standard måte til programkode. Likevel kreves det en viss relasjonell forståelse for å planlegge hvordan man skal skrive koden og hva som vil være mest hensiktsmessig for å løse det konkrete problemet. I tillegg kan instrumentell tilnærming til oppgavene gjøre at man ikke oppdager programmeringsfeil. Det som særlig krever relasjonell forståelse, er å kunne vurdere

svarene. Dermed kan man si at selve programmeringsjobben til en viss grad er instrumentell, mens svarvurderinga er relasjonell. Svarvurdering var en viktig del av arbeidskravet.

Chevalier de Méré-problemet

Den ene delen av arbeidskravet omfatta Chevalier de Méré-problemet. I 1654 utvekslet Pierre de Fermat og Blaise Pascal en rekke brev der de diskuterte sannsynlighetsproblemer den franske hasardspilleren Chevalier de Méré hadde spurt om (Derriennic, 2003; Maxwell, 1999; Ore, 1960). Blant disse var hvor mange kast man trenger med to terninger for å ha over 50 % sannsynlighet for å få minst én dobbeltsekser. Var det 24 eller 25? Chevalier de Méré inngikk veddemål om at han vinner (det vil si får tilbake det dobbelte av innsatsen) hvis han får minst én dobbeltsekser på 24 kast med to terninger. Ved tap mister han innsatsen. Men han oppdaget at han tapte på det i lengden. For å løse denne oppgaven kan man se på den komplementære hendelsen å ikke få noen dobbeltsekser i det hele tatt på for eksempel 24 kast. For hvert kast er det kun én av 36 muligheter som gir dobbeltsekser, og dermed er det på hvert kast $\frac{35}{36}$ sjanse for ikke å få dobbeltsekser. Denne sjansen må multipliseres med seg selv 24 ganger siden vi kaster de to terningene 24 ganger, og dermed har vi regna ut sjansen for ikke å få noen dobbeltseksere i det hele tatt. Sammen med sannsynligheten for å få minst én dobbeltsekser må dette bli 1, og vi har dermed at sannsynligheten for minst én dobbeltsekser på 24 kast er $1 - \left(\frac{35}{36}\right)^{24} \approx 0{,}4914$. For 25 kast er sannsynligheten for minst én dobbeltsekser $1 - \left(\frac{35}{36}\right)^{25} \approx 0{,}5055$. En vanlig misoppfatning er å tro at sannsynligheten er 24·1/36 = 2/3. I det lange løp er det slik at man på enkelte runder med spillet kaster dobbeltsekser flere ganger i løpet av de 24 kastene. Man vinner selvsagt da den runden, men de ekstra dobbeltsekserne hjelper ikke for de rundene man ikke fikk noen dobbeltseksere i det hele tatt, og dermed tapte. Misoppfatninga om at det er 2/3 sannsynlighet tilsvarer at man har telt disse ekstra dobbeltsekserne med, og fordelt dem utover alle de gangene man ellers ville tapt. Man kan dermed på en måte si at problemet viser viktigheten av å ta hensyn til at tilfeldigheter ikke er jevnt fordelt.

Monty Hall-problemet

Den andre delen av arbeidskravet omfatta Monty Hall-problemet. Dette problemet ble sendt inn til American Statistician av Steve Selvin (1975). I sin mer moderne form går det ut på at talkshow-verten Monty Hall viser programgjesten tre dører og forklarer at bak den ene er det en bil, mens bak hver av de to andre er det en geit. Gjesten skal få velge en dør, og vinner det som er bak døra. Etter at gjesten velger en dør, åpner Monty Hall en av de andre dørene og viser at bak denne døra er det en geit. Gjesten får så muligheten til å bytte dør. Bør han bytte dør (gitt at han ønsker å vinne bilen)? (Shermer, 2009)

En viktig forutsetning for å løse problemet er antakelsen om at Monty Hall uansett må åpne en dør, slik at han ikke kan lure gjesten med bare å åpne en dør når gjesten har valgt riktig dør første gang. Det er noen vanlige feilsvar i Monty Hall-problemet. Det ene er at å åpne en dør med en geit bak ikke endrer sannsynligheten. Et annet feilsvar er å tenke at siden det bare er to dører igjen å velge mellom etter at verten har åpnet en av dørene, så er sannsynligheten for å vinne bilen 1/2.

Når gjesten velger første gang er det 1/3 sjanse for at han velger døra med bilen, og 2/3 sjanse for at han velger en dør med en geit. Anta nå at han valgte en geit. Da vil han, når Monty Hall åpner døra med den gjenværende geita, få bilen hvis han bytter dør. Dermed er det 2/3 sjanse for å vinne bilen hvis han bytter dør. I 1/3 av tilfellene vil han bytte vekk fra bilen, nemlig i de tilfellene hvor han starta med å velge døra med bilen bak.

Om utvikling av oppgavene som inngikk i arbeidskravet

Det ble formulert tilstrekkelig vanskelige oppgaver til at det var hensiktsmessig for studentene å bruke manuell simulering eller programmering; samtidig kunne de, om de ville, regne ut analytisk. Oppgavene kunne løses numerisk med forholdsvis enkel programmering med funksjoner i Excel med programmeringsspråket VBA som er tilgjengelig i regnearket Excel i Windows-versjonen. Det var tilstrekkelig at studentene kunne bruke noen utvalgte kommandoer som HVIS og FOR, samt bruk av makroer i VBA. Her fikk vi inn vilkår i form av HVIS, og løkker i form av makroene. Dette er relevant for lærere siden det er spesifikt nevnt i den nye læreplanen (Udir, 2019b). Studentene fikk opplæring i Excel og VBA fra faglærere med kompetanse i programmering, mens matematikklærerne hadde ansvaret for nødvendig opplæring i matematikk. Det ble også laga et notat som forklarte Monte Carlo-metoden og som inkluderte eksempler på tilsvarende problemer.

Chevalier de Méré-problemet skulle i arbeidskravet løses ved hjelp av Monte Carlo-simulering programmert i regneark uten at studentene ble tvunget til å gjøre det for hånd, men de hadde på forhånd fått det enklere problemet å finne ut sannsynligheten for å få minst én sekser på fire terningkast. Her kunne de sammenlikne resultatet fra simuleringa med resultatet fra utregninga for hånd, og på den måten oppdage eventuelle systematiske feil ved hjelp av en enkel oppgave.

Når studentene gjorde Monty Hall-problemet skulle de, før de gjorde simuleringa, gi argumenter for å bytte eller ikke bytte dør, mens de etter simuleringa skulle vurdere om de ville endre det de tidligere hadde skrevet.

Chevalier de Méré-problemet er forholdsvis enkelt, men det krever forståelse av komplementære hendelser. Monty Hall er et såkalt sannsynlighetsparadoks laga for å forvirre den som skal løse oppgaven. Ved å utvikle et arbeidskrav der begge inngår, slik det ble gjort i denne studien, har man to forskjellige typer problemer som studentene kan se på i arbeid med sannsynlighetsoppgaver med og uten programmering.

Faglærers vurdering av studentgruppenes besvarelser av arbeidskravene

Når det gjelder vurdering av gruppebesvarelsene knytta til Chevalier de Méré-problemet, vurderte faglærer i matematikk om studentene hadde programmert riktig eller galt, og påpekte også hva de studentene som ikke hadde fått rett svar kunne gjøre for å få det rett. Videre vurderte faglærer hvordan studentene begrunnet konklusjonen angående antall nødvendige kast.

Når det gjelder vurdering av gruppebesvarelsene av Monty Hall-problemet, vurderte faglærer om besvarelsene hadde riktig konklusjon før studentene gjorde Monte Carlo-simuleringa manuelt. Faglærer vurderte også hvilke konklusjoner studentene trakk etter simulering, og i hvilken grad de endra sine konklusjoner etter simuleringa. Faglærer la blant annet vekt på argumentasjon for å bytte dør og eventuell forklaring på hvorfor svaret ble endra.

Metode

Utvalg

Det var to ulike kull som gjennomførte arbeidskravet. Et halvt år etter at siste kull hadde gjennomført undervisning og innlevert arbeidskrav ønska forfatterne av artikkelen å forske på arbeidskrav og studentenes besvarelser. Vi utvikla en delstudie og denne inngår i det større NSD-godkjente prosjektet *ILD – Integrerte læringsarenaer gjennom digitalisering* (Bjarnø et al., 2018) som var OsloMet sitt delprosjekt under STIL. Godkjenninga fra NSD innebærer at studentbesvarelsene kan forskes på når alle studentene på en gruppe har gitt sitt samtykke til at gruppebesvarelsen kan brukes.

Det var fem klasser i hvert kull med lærerstudenter som fikk undervisning og som besvarte arbeidskrav om sannsynlighet med simulering. Vi valgte å la to klasser fra siste kull inngå i studien; dette for å begrense mengden data som skulle analyseres, samtidig som det sikra variasjon i besvarelsene. De to klassenes arbeidskrav ble retta av læreren som hadde utvikla arbeidskravet, og som er medforfatter av denne artikkelen, og som da også har gitt tillatelse til å bruke rettingene av besvarelsene som datakilde.

I de to klassene var det totalt 87 studenter fordelt på 23 grupper. Men for at gruppebesvarelsen skal analyseres, måtte alle studentene i den aktuelle gruppa gi sitt samtykke. Signert samtykkeerklæring er mottatt fra samtlige deltakere i 16 grupper, noe som utgjør 60 studenter.

Datakilder og dataanalyse

Denne studien har to datakilder: 1) de innleverte besvarelsene fra 16 studentgrupper, og 2) faglærers retting av besvarelsene.

Studien bruker en kvalitativ tilnærming med gjennomgang og kategorisering av studentbesvarelsene. Hensikten er å besvare de to problemstillingene.

Analyse av studentbesvarelser og faglærers rettinger er basert på en strukturert tilnærming til å identifisere kategorier i de kvalitative dataene (Thagaard, 1993, 2013). Vi har valgt å gjennomføre en systematisk reduksjon av innhold i data ved å identifisere temaer og trekke ut meningsfulle tolkninger av disse (Roller, 2019). Hver gruppebesvarelse med tilhørende rettinger er å anse som et utskrevet kasus. Kasusene ble koda og ført inn i to matriser, en for hver oppgave (se tabell 1 og 2 i resultatdelen). Hver gruppebesvarelse ble tildelt en rad, mens deloppgavene er å finne i hver sin kolonne. For å unngå å miste relevant informasjon mens analysen pågikk, ble vurdering av studentbesvarelsene koda med korte beskrivelser framfor tall. I selve analysen samarbeida forfatterne konkret med å se etter interessante funn i de koda matrisene, for så å veksle til de utskrevne kasusene, og derigjennom sikre en helhetlig og felles forståelse av analysen. En slik veksling er svært viktig for å unngå å miste informasjon når et kvalitativt materiale er koda på en kvantitativ måte (Thagaard, 1993). En slik måte å arbeide på, bidrar til å styrke studiens reliabilitet og å forebygge tilfeldig og systematisk feilkoding av besvarelser. I resultatdelen er deler av analyseprosessen nærmere illustrert gjennom utvalgte eksempler.

Validitet

Det er ikke et tilfeldig utvalg som inngår i studien. Det er en av fire lærere som retta disse arbeidskravene og som underviste klassene i temaet som er inkludert i studien. Studien inkluderer likevel så mange som 60 av 200 studenter på kullet. Disse avgrensningene innebærer at det ikke uten videre kan generaliseres funn fra analysene til å gjelde alle lærerstudenter ved studiestedet. Analysene kan likevel gi en forståelse av hva studentene får til, og hva de opplever som problematisk. Disse funnene belyses ut fra aktuell teori og annen forskning.

Når det gjelder analyser for å identifisere kategorier i kvalitative data (Roller, 2019), har forfatterne diskutert seg fram til og blitt enige om kategorier som er gyldige beskrivelser av funn i datasettet.

Det kan hende at noen studenter kjente til problemene fra før. Det er en feilkilde som vi dessverre ikke har mulighet til å kontrollere for. Siden arbeidskravet var et gruppearbeidskrav, var det tilstrekkelig at én person på gruppa enten forstår eller har sett Monty Hall-problemet før, for at gruppa skal få rett konklusjon før de starter simuleringa.

Resultat

I resultatdelen presenteres dataene på en strukturert måte ved tabeller. Representative eksempler på studentsvar vises eller forklares for de enkelte kategoriene.

Programmering (Chevalier de Méré-problemet)

Faglærer vurderte arbeidskravene ut fra om studentene hadde feil i sine programmer, samt i hvilken grad eventuell feilaktig konklusjon på oppgaven hang sammen med om studentene hadde feil i programmeringa. I tabell 1 er hver gruppe G1–G16 plassert i en rad. De to øverste radene inneholder tekst som beskriver aktuelle løsninger på de to oppgavene.

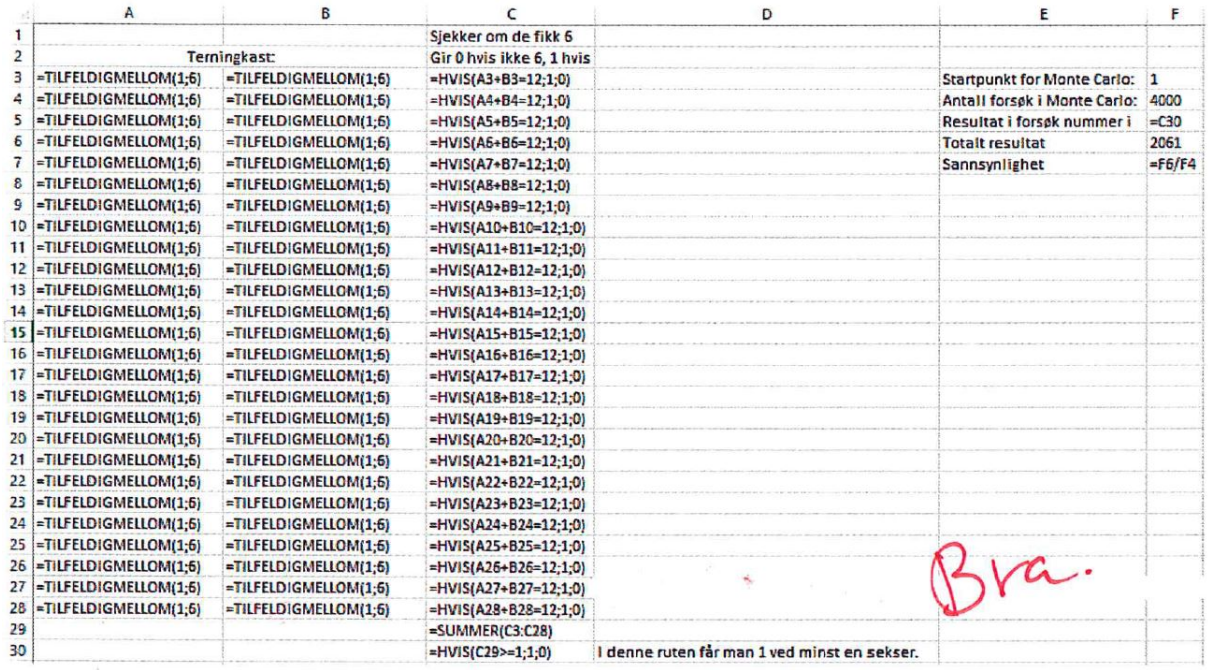
Tabell 1. Oversikt over hvordan studentgruppene har løst Chevalier de Méré-problemet (se beskrivelse av arbeidskrav i vedlegg)

	Studentenes programmering var i det vesentligste uten feil		**Feil i studentenes programmering**	
	25 kast gir sannsynlighet over 50 % for minst én dobbeltsekser (rett svar, markert med X)	Feil svar pga. tilfeldigheter eller manglende systematikk	Feil i begge deloppgaver	Feil i kun én deloppgave
G1		24 kast må til		
G2–G3	X			
G4				Klarer ikke å bruke det rette svaret til å se at det andre svaret må være galt1
G5		26 kast må til		
G6			Liten feil, men svært gal konklusjon pga. denne feilen	
G7		28 kast må til		
G8		26 kast må til		
G9–G11	X			
G12				Klarer ikke å bruke det rette svaret til å se at det andre svaret må være galt2
G13–G16	X			

Faglærer vurderte at 13 av 16 studentgrupper hadde løst selve programmeringsdelen på en tilfredsstillende måte (se kolonnene med overskriften «Studentenes programmering var i det vesentligste uten feil» i tabell 1). Konkret vil det si at de hadde skrevet kode som viser at de hadde forstått problemet. Et eksempel på rett løsning er gruppe G16, se figur 1. Det er en forhåndsprogrammert makro ved hjelp av tallene i celle F3–F5 som gir resultatet i celle F6, den sørger i dette tilfellet for at man får nye tilfeldige tall i kolonne A og B 4000 ganger (tallet i celle F4), og teller opp antall enere man i løpet av de 4000 forsøkene har fått i celle C30.

[1] Galt ved programmering av sannsynlighet for dobbeltsekser på 24 kast med to terninger. Riktig ved programmering av antall kast for å få sannsynlighet større enn ½ for minst én dobbeltsekser.
[2] Riktig ved programmering av sannsynlighet for dobbeltsekser på 24 kast med to terninger. Galt ved programmering av antall kast for å få sannsynlighet større enn ½ for minst én dobbeltsekser.

Figur 1. Eksempel på riktig løsning av Chevalier de Méré-problemet

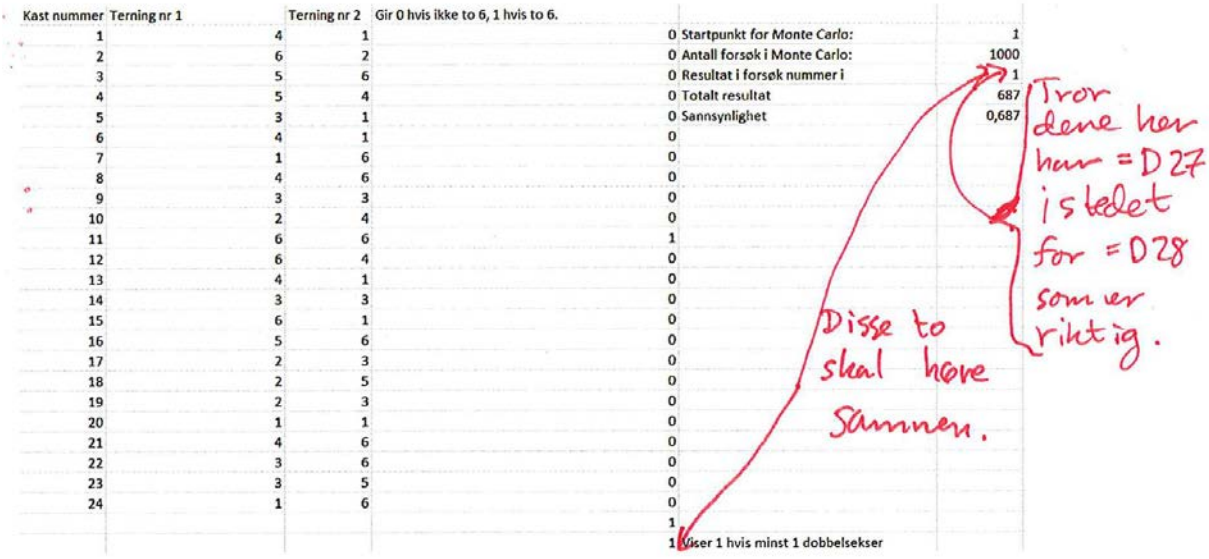


	A	B	C	D	E	F
1			Sjekker om de fikk 6			
2	Terningkast:		Gir 0 hvis ikke 6, 1 hvis			
3	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A3+B3=12;1;0)		Startpunkt for Monte Carlo:	1
4	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A4+B4=12;1;0)		Antall forsøk i Monte Carlo:	4000
5	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A5+B5=12;1;0)		Resultat i forsøk nummer i	=C30
6	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A6+B6=12;1;0)		Totalt resultat	2061
7	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A7+B7=12;1;0)		Sannsynlighet	=F6/F4
8	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A8+B8=12;1;0)			
9	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A9+B9=12;1;0)			
10	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A10+B10=12;1;0)			
11	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A11+B11=12;1;0)			
12	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A12+B12=12;1;0)			
13	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A13+B13=12;1;0)			
14	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A14+B14=12;1;0)			
15	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A15+B15=12;1;0)			
16	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A16+B16=12;1;0)			
17	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A17+B17=12;1;0)			
18	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A18+B18=12;1;0)			
19	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A19+B19=12;1;0)			
20	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A20+B20=12;1;0)			
21	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A21+B21=12;1;0)			
22	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A22+B22=12;1;0)			
23	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A23+B23=12;1;0)			
24	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A24+B24=12;1;0)			
25	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A25+B25=12;1;0)			
26	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A26+B26=12;1;0)			
27	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A27+B27=12;1;0)			
28	=TILFELDIGMELLOM(1;6)	=TILFELDIGMELLOM(1;6)	=HVIS(A28+B28=12;1;0)			
29			=SUMMER(C3:C28)			
30			=HVIS(C29>=1;1;0)	I denne ruten får man 1 ved minst en sekser.		

Bra.

Tre studentgrupper (G4, G6 og G12) hadde en feil. Feilen var at når de utvida regnearket for å inkludere flere kast, og på den måten forskjøv cella med resultatet fra hvert enkelt Monte Carlo-forsøk, så endra de ikke samtidig hvilken celle selve Monte Carlo-scriptet henta data fra. Se figur 2 for eksempel på hva gruppe G4 gjorde.

Figur 2. Eksempel på programmeringsfeil i resultatcella



Kast nummer	Terning nr 1	Terning nr 2	Gir 0 hvis ikke to 6, 1 hvis to 6.			
1	4	1		0	Startpunkt for Monte Carlo:	1
2	6	2		0	Antall forsøk i Monte Carlo:	1000
3	5	6		0	Resultat i forsøk nummer i	1
4	5	4		0	Totalt resultat	687
5	3	1		0	Sannsynlighet	0,687
6	4	1		0		
7	1	6		0		
8	4	6		0		
9	3	3		0		
10	2	4		0		
11	6	6		1		
12	6	4		0		
13	4	1		0		
14	3	3		0		
15	6	1		0		
16	5	6		0		
17	2	3		0		
18	2	5		0		
19	2	3		0		
20	1	1		0		
21	4	6		0		
22	3	6		0		
23	3	5		0		
24	1	6		0		
				1		
				1	Viser 1 hvis minst 1 dobbelsekser	

Tror denne her burde = D27 istedet for = D28 som er riktig.

Disse to skal høre sammen.

De tre studentgruppene med denne programmeringsfeilen konkluderte også feil på Chevalier de Méré-problemet. De klarte heller ikke å vurdere at svaret var langt utenfor det som var rimelig å forvente. Gruppe G6 skrev at det var 66 % sjanse for minst én dobbeltsekser på 24 kast, mens gruppe G4 mente det var 69 %. Gruppe G4 skrev videre at det var omtrent 50 % sjanse for minst én dobbeltsekser

på 26 kast. Gruppe G12 hadde også en annen feil, og fikk samme svar uansett om de la til kast. Dette skyldtes at de uansett bare tok med de 24 første kastene i summeformelen sin når de skulle finne om det ble dobbeltsekser. Dermed hjalp det dem ikke når de la til ekstra kast nedover i regnearket sitt.

Fire av de 13 studentgruppene (G1, G5, G7 og G8) som i det alt vesentligste hadde programmert riktig, fant likevel ikke riktig svar på oppgaven. To av dem (G5 og G8) konkluderte med at man måtte opp i 26 kast eller flere for å få over 50 % sannsynlighet for minst én dobbeltsekser, noe som er et eller flere kast for mye. Eksempelvis skrev gruppe G5: «Ved 24 kast ble sannsynligheten for å få minst én dobbeltsekser 45 %. Vi økte først med ett kast og fikk 49 % sannsynlighet for dobbeltsekser. Da vi økte til 26 kast fikk vi en sannsynlighet på 53 %, altså større enn 1/2.» Gruppe G8 skrev: «Dette gjør at man i realiteten ikke burde inngå dette veddemålet med 25 kast, men heller på 26 kast.» Gruppe G7 hadde 28 kast som svar. Gruppe G1 konkluderte med at 24 kast var tilstrekkelig, ett kast for lite.

Argumentasjon (Monty Hall-problemet)

Når det gjelder løsning av Monty Hall-problemet, var to elementer sentrale ved faglærers vurdering. Det var 1) hvor godt studentbesvarelsene argumenterte for svaret sitt og i hvilken grad de hadde rett svar før gruppene gikk i gang med å gjøre Monte Carlo-simuleringa manuelt, og 2) om studentgruppene trakk riktige konklusjoner i etterkant av den manuelle simuleringa med kopper og papirlapper i 25 simuleringer. Det har betydning hvilken sannsynlighet som indikeres gjennom manuell simulering, og siden det er forholdsvis få forsøk så er det ofte man får tall som ligger langt unna en sannsynlighet på 2/3. I tabell 2 vises resultatene fra gruppene G1–G16. Det er en rad for hver gruppe. Den øverste raden henviser til hvilken oppgave det gjelder, mens vi har klassifisert løsningene i den neste raden. I cellene brukes X, eller en kommentar der det er mer hensiktsmessig, for å vise hvilken kategori svaret til den enkelte gruppa kan plasseres i.

Det første vurderingspunktet hos faglærer var hvorvidt studentgruppene argumenterte for 2/3 vinstsjanse hvis dør byttes (rett svar) i forkant av den manuelle simuleringa, eller om de argumenterte for andre sannsynligheter. Det viste seg at ti grupper argumenterte riktig før de gjennomførte den manuelle simuleringa. De resterende seks gruppene (G3, G5, G7, G12, G14 og G15) argumenterte for andre svar. Av disse hadde tre den vanlige misoppfatninga om at sjansen for gevinst var 50 % ved bytte av dør. For eksempel skriver gruppe G12: «Det at programlederen viser at det er en geit bak den ene døra, gir det 1/2 dels sjanse for å vinne bilen. Deltakeren vil da ha to ulike utfallsrom. Det ene utfallsrommet vil da være å vinne en geit, og det andre vil være å vinne en bil. […] Det vil være like sannsynlig at bilen er bak begge dørene, og det vil derfor ikke spille noen rolle hvis han flytter seg eller ikke.» Gruppe G1 argumenterte først for 50 %, og så i neste deloppgave, før de hadde gjennomført den manuelle simuleringa, for 2/3.

Tabell 2. Oversikt over studentgruppenes besvarelser ved manuell simulering (Monty Hall-problemet)

	Før manuell simulering: Vurdering av sjanse for å vinne bilen ved å bytte dør			**Resultat av manuell simulering**		**Etter manuell simulering: Vurdering av sjanse for å vinne bilen ved å bytte dør**			
	2/3 sjanse	Likt, 1/2 sjanse	Annet	Resultat nær forvent-nings-verdi (p=2/3)	Resultat langt unna forvent-nings-verdi	Beholder argument for 2/3 med støtte i simulering	Hadde galt svar før simulering, og endrer ikke etter simulering	Endrer til rett svar etter simulering	Annet
G1	Argu-menterer for dette i punkt b	Argu-menterer først for dette i punkt a			X				Simuleringa peker i motsatt retning av det teoretiske
G2	X				X				Tallene de fikk kunne fått dem til å tvile i simuleringa
G3		X		X				X	
G4	X			X		X			
G5			Mage-følelse	X			X		
G6	X			X		X			
G7			Mage-følelse	X				Endret uten å bruke simuleringa	
G8	X			X		X			
G9	X			X		X			
G10	X			X		X			
G11	X				X	X			
G12		X		X				X	
G13	X			X		X			
G14			Feilen 1/3+1/2 =1	X				X	
G15		X		Grense-tilfelle			X		
G16	X			X		X			

Det første vurderingspunktet hos faglærer var hvorvidt studentgruppene argumenterte for 2/3 vinstsjanse hvis dør byttes (rett svar) i forkant av den manuelle simuleringa, eller om de argumenterte for andre sannsynligheter. Det viste seg at ti grupper argumenterte riktig før de gjennomførte den manuelle simuleringa. De resterende seks gruppene (G3, G5, G7, G12, G14 og G15) argumenterte for andre svar. Av disse hadde tre den vanlige misoppfatninga om at sjansen for gevinst var 50 % ved bytte av dør. For eksempel skriver gruppe G12: «Det at programlederen viser at det er en geit bak den ene døra, gir det 1/2 dels sjanse for å vinne bilen. Deltakeren vil da ha to ulike utfallsrom. Det ene utfallsrommet vil da være å vinne en geit, og det andre vil være å vinne en bil. […] Det vil være like sannsynlig at bilen er bak begge dørene, og det vil derfor ikke spille noen rolle hvis han flytter seg eller ikke.» Gruppe G1 argumenterte først for 50 %, og så i neste deloppgave, før de hadde gjennomført den manuelle simuleringa, for 2/3.

Det andre vurderingspunktet var hvilke forklaringer studentgruppene ga etter manuell simulering. Det var 12 grupper som hadde riktige forklaringer etter manuell simulering, mens to grupper (G5 og G15) hadde mangelfulle eller feilaktige forklaringer. To av gruppene (G1 og G2) registrerte at resultatet de fikk i simuleringa er problematisk. G2 skriver at «dette gjør at man kan begynne å tvile på teorien, og da heller velge mageføIelsen».

Når vi sammenlikner studentgruppene før og etter manuell simulering, viser det seg at 11 grupper beholdt den samme type argumentasjon angående Monty Hall-problemet, mens fire grupper skifta mening om løsning i etterkant av den manuelle simuleringa.

Det var studentgruppene G3, G7, G12 og G14 som endra fra feilaktig argumentasjon før manuell simulering til riktig argumentasjon etter manuell simulering. Felles for disse gruppene var at de fikk tallverdier som stemte godt med teoretisk forventningsverdi gjennom simulering. Eksempelvis skriver gruppe G12 at «utfra våre resultater og Monty Hall sitt problem vil det helt klart lønne seg å bytte dør» og gruppe G14 at «etter å ha gjennomført forsøket ser det ut som det lønner seg å bytte dør. Det er større sannsynlighet å velge en dør med geit bak fordi det er to geiter og en bil. Dermed 2/3 sjanse for å velge dør med geit bak. Om vi da antar at man velger en dør med geit bak og Monty Hall velger bort en geit, er du garantert en bil.»

Diskusjon

I denne delen diskuterer vi mulige svar på de to problemstillingene, samt at vi kommer inn på mulige implikasjoner.

Drøfting av programmering

Den første problemstillinga tar opp hvordan programmeringsfeil og manglende systematikk influerer på løsninga av Chevalier de Méré-problemet. Det var tre studentgrupper som hadde feil i programmeringsdelen. Feilen var at når de utvida regnearket for å inkludere flere kast, og på den måten forskjøv cella med resultatet fra hvert enkelt Monte Carlo-forsøk, så endra de ikke samtidig hvilken celle selve Monte-Carlo-scriptet henta data fra (eksempel på dette i figur 2). Det finnes ulike forklaringer på hva en slik feil skyldes. Den kan komme av at studentene tenker at Excel korrigerer cellenummer uansett hvordan du setter inn ekstra celler, noe som ikke er tilfelle. Det kan også være slik at studentene ikke har tenkt over hvilke linjer som er inkludert i løkka (Sleeman et al., 1986). Det er også en forvirring om hvor parameterverdien kommer fra (Ragonis & Ben-Ari, 2005). En annen forklaring kan være slurv eller unøyaktighet, og at de ikke har tenkt godt nok over hvordan det de gjør av endringer interagerer med programkoden som allerede er skrevet. Programmet fungerer dermed mest som en maskin med ukjent innhold.

Hos de tre studentgruppene som hadde gjort en liten programmeringsfeil, fungerte programmet tilsynelatende likevel. Siden studentene opplevde at kodene fungerte, det vil si at de fikk et tallresultat og ingen feilmelding, fant studentene ingen grunn til å gjennomføre feilsøking. Det var derfor vanskelig for disse studentgruppene å oppdage og korrigere programmeringsfeilen før innlevering. På den andre sida var tallene de fikk ut såpass langt unna det som burde forventes, og for to av gruppene så langt unna svaret på en nesten lik oppgave, at de burde ha forstått at noe var feilprogrammert selv uten et feilvarsel.

Feil som derimot fører til at programmet ikke får kjørt, oppdages mye lettere før innlevering, og kan da korrigeres. Dette forklarer at de feil som ble avdekka i lærers retting, ikke var feil som hindra programmet i å kjøre, men ga feil resultat i beregninga av sannsynlighet.

Det var altså 13 studentgrupper som hadde programmert i det alt vesentligste riktig. Fire av disse gruppene (G1, G5, G7 og G8) klarte likevel ikke å finne riktig svar på oppgaven. En gruppe konkluderte med ett kast for lite (24 kast). Når det gjelder denne studentgruppa, er det vanskelig å vite hva årsaka er. En mulig forklaring kan være at studentene har misforstått hva de har regna ut, og for eksempel tenkt at de skal komme så nær 50 % som mulig.

To av gruppene (G5 og G8) konkluderte med at man måtte opp i 26 kast for å få over 50 % sannsynlighet for minst én dobbeltsekser. En annen studentgruppe (G7) testet med 28 kast, og fant ut at det ved 28 kast var om lag 55 % sannsynlighet for å få en dobbeltsekser. De valgte da å ikke arbeide videre for å sjekke om færre kast enn 28 fortsatt ga en sannsynlighet på over 50% for å få en dobbeltsekser. De burde prøvd seg fram med å redusere antall kast for å se om dette fortsatt gir over 50 % sannsynlighet for dobbeltsekser. Det tyder på at de rett og slett tenkte at så snart de hadde funnet noe som var over 50 % sannsynlighet, så var de i mål med oppgaven. Dette viser manglende systematisk tenkning, ettersom det i en slik simuleringsoppgave ved hjelp av datamaskinen ville vært mest naturlig, ut ifra matematisk tradisjon, å prøve seg fram med flere og flere kast.

En utfordring med flerfaglige oppgaver (matematikk og digital kompetanse/ IKT) er at for å gi god veiledning bør faglærer ha en viss kompetanse i begge fagfelt. Det kan også være vanskelig for studentene å sette sammen informasjon fra to eller flere fagfelt i lærerutdanninga.

Drøfting av argumentasjon

Den andre problemstillinga tar opp hvordan studentgruppene konkluderer om løsninga av Monty Hall-problemet før de tar i bruk Monte Carlo-simulering.

Det var 10 av 16 studentgrupper som hadde riktig argumentasjon for løsning av Monty Hall-problemet før de gjennomførte den manuelle simuleringa, mens det var 14 studentgrupper som hadde riktige forklaringer etter å ha gjennomført manuell simulering, hvorav to diskuterte problemene med at den manuelle simuleringa indikerte at de hadde tatt feil. Det var fire studentgrupper (G3, G7,

G12 og G14) som endra fra feil argumentasjon før manuell simulering til riktig argumentasjon etter manuell simulering. Felles for disse gruppene var at de fikk tallverdier nær virkelig forventningsverdi gjennom simulering. Dermed skapes en kognitiv konflikt mellom simulering og den første argumentasjonen. Studentene vil da i teorien være åpne for at de må endre argumentasjonen sin ettersom de oppdager at det opprinnelige argumentet var feil (Brekke, 2002).

Manuell simulering eller simulering gjennom programmering

En fordel med bruk av manuell simulering kan være at det er mer transparent for studentene å se og forstå hva de gjør og hva som skjer under simuleringa. Imidlertid viste studentbesvarelsene at 25 manuelle forsøk var for lite til at alle gruppene fikk realistiske tallverdier som ikke avvek for mye fra forventningsverdien 2/3. Dette kan være et argument for at manuell Monte Carlo-simulering ikke bør brukes i for stor grad, og at det kan være mer fordelaktig med programmering hvor det er mulig å få veldig mange forsøk/repetisjoner. På den annen side unngikk samtidig studentene programmeringsfeil ved å utføre manuell Monte Carlo-simulering. Det var trolig mye tydeligere for studentene hva som skjedde under simuleringa, også for de studentene som ikke forsto programmeringa. Dette kan være et argument for at studenter også skal lære manuell simulering, selv om det som regel ikke gir mange nok forsøk til at resultatene kan stoles på. Gruppe G15 er litt spesiell, i og med at studentene der tolka resultatene sine fra den manuelle simuleringa til å indikere at de kunne beholde sitt feilaktige argument, selv om tallene ikke ligger alt for langt unna forventningsverdien.

En fare ved bruk av dataprogrammer i læring er at studentene kan oppfatte programmene som en «svart boks» (Nabb, 2010). Studentene putter inn tall og får noe ut igjen uten å få noen forståelse av hva som har skjedd. Spesielt synes det her å være tilfelle med de gruppene som enten programmerte feil, eller fikk feilsvar i Chevalier de Méré-problemet. De vurderte i for liten grad om svaret de fikk kunne være realistisk. Uansett er Chevalier de Méré-problemet umulig å løse på en god måte med manuell Monte Carlo-simulering. Selv ved 1000 simuleringer på en datamaskin, som tilsvarer å kaste 48000 terninger, vil man få grupper som konkluderer feil fordi de er uheldige med de tilfeldige tallene datamaskinen gir ut. Slike resultater kan man også få ved riktig programmering.

Vi mener studentene kan ha fordel av å løse oppgaver med manuell Monte Carlo-simulering, samt gjøre oppgaver der man programmerer. Ved førstnevnte får man en tydeligere forståelse av hva Monte Carlo-simulering er, mens ved sistnevnte får man inn et nyttig verktøy for å kunne vurdere hva som kan være riktig svar på et sannsynlighetsspørsmål. Imidlertid er det et stort problem med Monte Carlo-simulering at den som oftest ikke forteller deg hvorfor intuisjonen din er feil, bare at den er feil. Samtidig, det å se svært tydelig at du har tatt feil i intuisjonen din vil skape en kognitiv konflikt, og dermed vil du kanskje være mer åpen for ny kunnskap om sannsynlighet etterpå (Brekke, 2002). Samtidig er det et problem at simuleringa kan gi feilsvar, og dermed skape en kognitiv konflikt

som ikke skulle vært skapt. Det viktigste må være at studentene får en bedre forståelse i å vurdere svar, og flere mulige verktøy for å skape svarforslag slik at man kan sammenlikne ulike svar fra forskjellige metoder. Det er på denne måten en kan skape relasjonell forståelse. Det kan alltid være studentgrupper som ønsker å bli fort ferdige med arbeidskravet. De kan være villige til å levere inn et arbeidskrav de vet inneholder småfeil. Det kan være de tenker at faglærer uansett kommer til å kommentere dette, og at det ikke er avgjørende for å få godkjent, og dermed anser det som «bra nok». Det kan for eksempel tenkes at studentene leverer inn en besvarelse hvor de har ulike svar fra egne refleksjoner og manuell simulering, men at de vurderer det til å «koste» for mye tid og krefter å avdekke feil og rette dem opp.

Programmering og statistikk i lærerutdanninga

Studien viser at lærerstudenter kan lykkes med programmering knytta til løsing av statistiske problemer som ikke lar seg løse intuitivt. Det kan bidra til at studenter og elever ser at programmering i matematikkfaget ikke kun gjøres for programmeringas skyld. Det er da viktig å også hjelpe studentene til å koble programmering i statistikk og sannsynlighet til utvikling av forståelse av hvordan en går fram for å løse problemer og å vurdere realismen i de aktuelle løsnings-forslagene.

Det er mulig at programmeringsforståelse kan bidra til å løse et statistisk problem som Monty Hall. Ved programmering gjennom Monte Carlo-simulering er det en mulighet å la datamaskinen først velge hvilken dør bilen er bak og deretter la datamaskinen velge hvilken dør gjesten stiller seg foran. Så kan programmet finne ut hva som skjer hvis gjesten bytter dør. Da kan man ha en HVIS-funksjon som sier: `HVIS Dør.Gjest=Dør.Bil SÅ 0 ELLERS 1` , ettersom man jo bytter vekk fra denne. Ved å se at dette er den enkleste kodelinja, så kan det tenkes at den som programmerer oppdager at det riktige er å tenke at det er bare 1/3 sjanse for å få geit hvis du bytter, ettersom dette er samme sjanse som for å få bilen i utgangspunktet.

Det er også et spørsmål om hvilket programmeringsspråk man skal benytte i skolen, nå som programmering skal innføres. Flere argumenterer for å bruke spesialutviklede programmeringsspråk, for eksempel Scratch (Resnick et al., 2009), utvikla kun til skolebruk, ettersom dette hevdes å være det pedagogisk riktige, mens vi her bruker et programmeringsspråk som brukes kommersielt. Bærland og Gilje (2017) kaller programmering «den nye latin», og diskuterer om programmering er det nye fremmedspråket, og på denne måten sammenlikner de det å lære et programmeringsspråk med å lære et «snakket» språk. Hvis vi sammenlikner med fremmedspråkene, har man for eksempel valget mellom å lære elevene esperanto eller engelsk. Esperanto er et konstruert, kunstig språk, laga for å være lett å lære, med grammatisk regelmessighet og ingen spesialregler (Auld, 1987, s. 26, 27, 32). Videre skrives det akkurat som det snakkes. Det er et pedagogisk fornuftig språk. Engelsk er derimot et ekte språk med mange

grammatiske unntak og spesialregler, og skrives også ganske forskjellig fra hvordan det uttales. Så pedagogisk sett kunne det være enklere for elevene å lære esperanto enn engelsk. På den andre sida lærer vi elevene heller opp i engelsk fordi det er mer brukt, og man tenker at det er like godt å lære det vanskelige språket med en gang. Slik kan man også tenke i programmering. Da bør man bruke et programmeringsspråk som er laga for å kunne brukes i seinere utdanning og arbeidsliv. Vi brukte Excel med VBA, men kunne for eksempel ha brukt Python eller noe annet. Vihavainen et al. (2014) argumenterer også med at studenter like godt kan lære et programmeringsspråk som brukes i arbeidslivet.

Det er også et spørsmål hva man skal kunne si er forutsetninger for å kunne programmere. En måte å se på koding og programmering på, er å se på det å kunne kode som en liten, men grunnleggende del av det å kunne programmere. På samme måte som at det å kunne regne er en forutsetning for å kunne forstå matematikk selv om det kun er en liten del av matematikkfaget (Udir, 2017).

Oppsummering og konklusjon

Funn tyder på at Monte Carlo-simulering kan hjelpe studenter til å få en bedre forståelse av hva som er rett svar på kompliserte og kontra-intuitive sannsynlighetsoppgaver. Men samtidig fordrer dette at de i stor grad programmerer riktig eller får tall nær forventningsverdi på den manuelle Monte Carlo-simuleringa. Arbeidskravet inneholdt oppgaver som er litt for vanskelige til å løses for hånd, og det var derfor ikke mulig for studentene å vurdere svaret de får i Monte Carlo-simuleringa. Samtidig så vi at to grupper fikk vesensforskjellige svar på oppgave 1c og 1d (se vedlegg), omtrent 2/3 og omtrent ½. Disse to svarene burde ikke vært så veldig langt unna hverandre, og dermed burde studentene kunne konkludert med at de hadde en feil. For de fleste studentene kan Monte Carlo-simulering hjelpe i forståelsen, men manglende programmeringsferdigheter og uflaks med datamaskinens tilfeldige tall kan bidra til misoppfatninger i sannsynlighet. Hvis det hadde blitt gitt vesentlig enklere oppgaver ville ikke studentene nødvendigvis ha sett nytten av programmering, da de kunne ha løst dem analytisk. En viktig oppgave i forberedelsene til en lærerutdanner der studenter skal lære seg programmering, blir å finne eller lage oppgaver der de ser nytten av programmering. En bør også diskutere med lærerstudentene hva som er gode oppgaver i en slik situasjon slik at de får større forutsetninger for å bruke egna oppgaver i framtida med sine elever.

Videre vil vi argumentere for at det vil være fornuftig å velge programmeringsspråk som faktisk kan brukes seinere, og ikke bare i en skolesetting. I dette prosjektet fikk mange studentgrupper til programmeringa. Samtidig var det utarbeida et notat som omhandla introduksjon til Monte Carlo-simulering inkludert eksempler på programmering i Excel-ark, slik at de i forkant av arbeidskravet fikk se på eksempler som ikke var så forskjellige fra de som skulle løses, for å hente

inspirasjon. Ettersom det stort sett bare var forskjellige HVIS-kommandoer som ble brukt, er det nok slik at de har fått mer kunnskap om matematikk enn programmering etter gjennomføring av dette konkrete arbeidskravet. Samtidig er det jo slik at hvis programmering skal inn i matematikken, så bør den inn fordi den i enkelte tilfeller kan føre til økt forståelse av matematikk, og ikke utelukkende være der for programmeringas skyld. Slik sett bør programmering vurderes inn i flere fag, og brukes der de kan løse problemer som det ikke er så enkelt å løse manuelt.

Om forfatterne

Olav Gravir Imenes er førsteamanuensis i matematikk ved OsloMet – storby-universitetet. Han har doktorgrad i matematikk fra Universitetet i Oslo innafor fagfeltet ikke-kommutativ algebraisk geometri. Han har også forska og publisert arbeider innafor matematikkdidaktikk med vekt på tverrfaglighet, og bidratt til lærebøker i grunnskolelærerutdanningene.
Institusjonstilknytning: Institutt for grunnskole- og faglærerutdanning, OsloMet – storbyuniversitetet, Postboks 4, St. Olavs plass, 0130 Oslo, Norge.
E-post: ogim@oslomet.no

Vibeke Bjarnø er instituttleder og førstelektor i IKT og læring ved OsloMet – storbyuniversitetet. Hun har erfaring som lærerutdanner og fra arbeid med etter- og videreutdanning av lærere. Hun deltar i forsknings- og utviklingsarbeid retta mot flerfagsdidaktikk, utdanningsledelse og digital kompetanse. Hun har publisert artikler og skrevet bøker, deriblant en lærebok i IKT-didaktikk.
Institusjonstilknytning: Institutt for grunnskole- og faglærerutdanning, OsloMet – storbyuniversitetet, Postboks 4, St. Olavs plass, 0130 Oslo, Norge.
E-post: vibekeb@oslomet.no

Ove Edvard Hatlevik er professor i pedagogikk ved OsloMet – storbyuniversitetet. Han er forsker og lærerutdanner. Han har vært involvert i studier om skolens og lærerutdanningas digitale tilstand og kartlegging av digitale ferdigheter. Hans interesser er blant annet retta mot motivasjon og læringsstrategier. Han har publisert en rekke artikler og bøker innafor feltet IKT og læring.
Institusjonstilknytning: Institutt for grunnskole- og faglærerutdanning, OsloMet – storbyuniversitetet, Postboks 4, St. Olavs plass, 0130 Oslo, Norge.
E-post: ovedha@oslomet.no

Referanser

Auld, W. (1987). *Fenomenet Esperanto*. Oslo: Esperantoforlaget.

Bjarnø, V., Hatlevik, O. E. & Standal, Ø. F. (2018). *ILD – Integrerte læringsarenaer gjennom digitalisering*. Oslo: OsloMet – storbyuniversitetet. https://www.uv.uio.no/proted/publikasjoner/final-report-oslo-met.pdf

Brekke, G. (2002). *Kartlegging av matematikkforståelse. Introduksjon til diagnostisk undervisning i matematkk*. Utdanningsdirektoratet. https://www.matematikksenteret.no/sites/default/files/attachments/Elever%20som%20presterer%20lavt/P3_M3Brekke-G-Diagnostisk-undervisning_Utdrag.pdf

Burton, L. (1999). Why Is Intuiton so Important to Mathematicians but Missing from Mathematics Education? *For the Learning of Mathematics, 19*(3), 27–32. https://www.jstor.org/stable/40248307

Bærland, T. & Gilje, Ø. (2017,). *Fremmedspråket programmering*. Kronikk, Morgenbladet, 02.06.2017. https://morgenbladet.no/ideer/2017/06/fremmedspraket-programmering

Cheong, K. H., Koh, J. M., Yeo, D. J., Tan, Z. X., Boo, B. O. E. & Lee, Y. L. (2019). Paradoxical Simulations to Enhance Education in Matematics. *IEEE Access, 7*, 17941–17950. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8611076

Daroczy, G., Wolska, M., Meurers, W. D. & Nuerk, H.-C. (2015). Word problems: a review of linguistic and numerical factors contributing to their difficulty. *Frontiers in psychology, 6* Art. 348. https://www.frontiersin.org/articles/10.3389/fpsyg.2015.00348/full

Derriennic, Y. (2003). Pascal et les problèmes du chevalier de Méré. De l'origine du calcul des probabilités aux mathématiques financières d'aujourd'hui. *SMF – Gazette, 97*, 45–71. https://perso.univ-rennes1.fr/guy.casale/enseignement/cours/PRB/ChevalierdeMere.pdf

Eckhardt, R. (1987). Stan Ulam, John von Neumann, and the Monte Carlo Method. *Los Alamos Science, 15*(Special issue), 131–141. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9068

Fischbein, E. & Schnarch, D. (1997). The Evolution with Age of Probabilistic, Intuitively Based Misconceptions. *Journal for Research in Mathematics Education, 28*(1), 96–105. https://www.jstor.org/stable/749665

Gilje, Ø., Landfald, Ø. F. & Ludvigsen, S. (2018). Dybdelæring – historisk bakgrunn og teoretiske tilnærminger. *Bedre skole, 30*(4), 22–27.

Indriasari, T. D., Luxton-Reilly, A. & Denny, P. (2020). A Review of Peer Code Review in Higher Education. *ACM Transactions on Computing Education, 20*(3), Art. 22. https://doi.org/10.1145/3403935

KD (2015–2016). *Fag – Fordypning – Forståelse – En fornyelse av Kunnskapsløftet*. Meld. St. 28 (2015–2016). Oslo: Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/meld.-st.-28-20152016/id2483955/?ch=1

KD (2016a). *Forskrift om rammeplan for grunnskolelærerutdanning for trinn 1–7*. Oslo: Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/fbaf26939bbd40abacba73e34a95d2fc/forskrift-om-rammeplan-for-grunnskolelarerutdanning-for-trinn-1-7.pdf

KD (2016b). *Forskrift om rammeplan for grunnskolelærerutdanning for trinn 5–10*. Oslo: Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/fbaf26939bbd40abacba73e34a95d2fc/forskrift-om-rammeplan-for-grunnskolelarerutdanning-for-trinn-5-10.pdf

Kolikant, Y. B.-D. & Mussai, M. (2008). "So my program doesn't run!" Definition, origins, and practical expressions of students' (mis)conceptions of correctness. *Computer Science Education, 18*(2), 135–151. https://www.tandfonline.com/doi/full/10.1080/08993400802156400

KUF (1996). *Læreplanverket for den 10-årige grunnskolen* (L97). Oslo: Kirke-, utdannings- og forskningsdepartementet. https://www.nb.no/items/URN:NBN:no-nb_digibok_2008080100096?page=171

Lister, R., Simon, B., Thompson, E., Whalley, J. L. & Prasad, C. (2006). Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITICSE'06)*, 118–122. https://doi.org/10.1145/1140124.1140157

Lye, S. Y. & Koh, J. M. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computer in Human Behavior, 41*, 51–61.

Maxwell, M. (1999). A mathematical perspective on gambling. *MIT Undergraduate Journal of Mathematics, 1*, 123–132. http://educypedia.karadimov.info/library/MAXWEL~1.PDF

Metropolis, N. (1987). THE BEGINNING of the MONTE CARLO METHOD. *Los Alamos Science, 15*(Special issue), 125–130. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9067

Nabb, K. A. (2010). CAS as a restructing tool in mathematics education. *Proceedings of the 22nd International Conference on Technology in Collegiate Mathematics (Bd. 39)*, 247–259. http://archives.math.utk.edu/ICTCM/VOL22/R007/paper.pdf

Niss, M. & Jensen, T. H. (red.) (2002). *Kompetencer og matematiklæring: Ideer og inspiration til udvikling af matematikundervisning i Danmark.* Uddannelsesstyrelsens temahæfteserie nr. 18. https://www.researchgate.net/profile/Tomas-Hojgaard/publication/290429774_Kompetencer_og_matematiklaering_Ideer_og_inspiration_til_udvikling_af_matematikundervisning_i_Danmark/links/59e0cff4458515393d4cd87c/Kompetencer-og-matematiklaering-Ideer-og-inspiration-til-udvikling-af-matematikundervisning-i-Danmark.pdf?origin=publication_detail

Ore, O. (1960). Pascal and the invention of probability theory. *The American Mathematical Monthly, 67*(5), 409–419.

Poincaré, H. (1905). *La valeur de la science*. Paris: Ernest Flammarion. http://henripoincarepapers.univ-lorraine.fr/chp/hp-pdf/hp1919vs.pdf

ProTed (2016). *STIL – Studentaktiv læring i lærerutdanningene*. ProTed – Senter for fremragende lærerutdanning, Det utdanningsvitenskapelige fakultet, Universitetet i Oslo. https://www.uv.uio.no/proted/utviklingsomrader/stil.html

Qian, Y. & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education, 18*(1), 1–24. https://dl.acm.org/citation.cfm?id=3077618

Ragonis, N. & Ben-Ari, M. (2005). A Long-Term Investigaton of the Comprehension of OOP Concepts by Novices. *Computer Science Education, 15*(3), 203–221. https://cormack.uwaterloo.ca/papers/t4v336x210720l08.pdf

Regjeringen (2017). *Mer koding og teknologi inn i skolen*. Pressemelding. https://www.regjeringen.no/no/aktuelt/mer-koding-og-teknologi-inn-i-skolen/id2568375/

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM, 52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Rigby, P. C. & Bird, C. (2013). Convergent contemporary software peer review practices. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, August, 202–212.

Roller, M. R. (2019). A quality approach to qualitative content analysis: Similarities and differences compared to other qualitative methods. *Forum: Qualitative Sozialforschung/ Forum: Qualitative Social Research, 20*(3), Art. 31. https://www.qualitative-research.net/index.php/fqs/article/view/3385/4486

Sawyer, R. K. (2014). *Introduction: The New Science of Learning*. New York: Cambridge University Press.

Scherer, R., Siddiq, F. & Sánchez Viveros, B. (2019). The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects. *Journal of Educational Psychology, 111*(5), 764.

Selvin, S. (1975). Letters to the Editor. *The American Statistician, 29*(1), 67–71. https://doi.org/10.1080/00031305.1975.10479121

Shermer, M. (2009). The 3-Door Monty Hall Problem. *Scientific American, 300*, 2, 10–12. https://www.scientificamerican.com/article/the-3-door-monty-hall-problem/

Sigal, M. J. & Chalmers, R. P. (2016). Play It Again: Teaching Statistics With Monte Carlo Simulation. *Journal of Statistics Education, 24*(3), 136–156. https://doi.org/DOI:10.1080/10691898.2016.1246953

Sleeman, D., Putnam, R. T., Baxter, J. & Kuspa, L. (1986). Pascal and High School Students: A Study of Errors. *Journal of Educational Computing Research, 2*(1), 5–23. https://doi.org/10.2190/2XPP-LTYH-98NQ-BU77

Sprenger, J. (2010). Probability, rational single-case decisions and the Monty Hall Problem. *Synthese, 174*(3), 331–340.

Thagaard, T. (1993). *A Structural Approach to Qualitative Data Analysis.* Rapportserien ved sosiologi. Universitetet i Oslo.

Thagaard, T. (2013). *Systematikk og innlevelse: En innføring i kvalitativ metode* (4. utg.). Bergen: Fagbokforlaget.

Udir (2006). *Læreplan i matematikk fellesfag (MAT1-04).* Oslo: Utdanningsdirektoratet. https://www.udir.no/kl06/mat1-04

Udir (2017). *Rammeverk for grunnleggende ferdigheter*. Oslo: Utdanningsdirektoratet. https://www.udir.no/laring-og-trivsel/rammeverk/rammeverk-for-grunnleggende-ferdigheter/

Udir (2019a). *Læreplan i matematikk fellesfag 1.–10. trinn*. Oslo: Utdanningsdirektoratet. https://www.udir.no/lk20/mat01-05

Udir (2019b). *Læreplanverket – Fagfornyelsen*. Oslo: Utdanningsdirektoratet. https://www.udir.no/laring-og-trivsel/lareplanverket/

Udir (2019c). *Læring og trivsel – Dybdelæring*. Oslo: Utdanningsdirektoratet. https://www.udir.no/laring-og-trivsel/dybdelaring/

Vihavainen, A., Helminen, J. & Ihantola, P. (2014). How novices tackle their first lines of code in an IDE: Analysis of programming session traces. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (s. 109–116): ACM. https://dl.acm.org/citation.cfm?id=2674692

Watson, J. (2005). The Probabilistic Reasoning of Middle School Students. I G. A. Jones (red.), *Exploring Probability in School. Challenges for Teaching and Learning* (s. 145–169). Matematics Education Liberary: Springer.

Weltman, D. & Tokar, T. (2019). Using a Monte Carlo Simulation Exercise to Teach Principles of Distribution: An Enhanced Version of the Classic Transportation Problem. *INFORMS Transactions on Education, 19*(3), 111–120. https://doi.org/10.1287/ited.2018.0200

Wood, M. (2005). The Role of Simulation Approaches in Statistics. *Journal of Statistics Education, 13*(3), 1–11. https://www.tandfonline.com/doi/pdf/10.1080/10691898.2005.11910562

Vedlegg: De to oppgavene i arbeidskravet som omhandler programmering og er inkludert i studien

Oppgave 1

(a) Anta at du vedder på at du skal få minst en sekser på fire kast med en sekssidet terning. Vil du i lengden vinne eller tape på et slikt veddemål? Regn ut for hånd. Hint: Se på sjansen for ikke å få noen seksere først.
(b) Petter påstår at sjansen for å få minst en sekser i a)-oppgaven er $4 \cdot 1/6 = 2/3$. Hvordan har Petter tenkt? Hva er det han har glemt å ta hensyn til? Hvordan kan du hjelpe Petter til å få en bedre forståelse?
(c) Anta at du vedder på at du skal få minst en dobbeltsekser på 24 kast med to terninger. Vil du i lengden vinne eller tape på et slikt veddemål? (Derriennic, 2003, s. 45) Lag først en Monte Carlo-simulering i Excel som skal gi deg svaret på deloppgave a). Bruk 1000 forsøk. Sammenlikn med svaret du regnet ut. Gjør deretter en Monte Carlo-simulering i Excel som skal gi deg sannsynligheten for dobbeltsekseren. Bruk 1000 forsøk. Hva finner du? Kan du regne ut dette for hånd (frivillig å regne ut for hånd)?
(d) Hvor mange kast må du ha med to terninger for at sjansen for å få minst en dobbeltsekser er større enn 1/2? Bruk Monte Carlo-simulering. Velg selv passende antall forsøk.

Oppgave 2

I denne oppgaven skal vi se på Monty Hall-problemet (Shermer, 2009). Du er med i en konkurranse i et spilleprogram på fjernsynet som går ut på at du av verten Monty Hall blir vist tre dører. Du blir fortalt at bak en av dem er det en bil, og bak hver av de andre er det en geit. Du bestemmer deg for en av dørene og stiller deg foran den. Deretter åpner Monty Hall en av de andre dørene og viser at bak den døra var det en geit. Han gir deg så sjansen til å bytte dør.

(a) Når vi skal gjøre dette forsøket, hvilke forutsetninger kan det være fornuftig å legge til grunn?
(b) Hva er argumentene for å bytte dør? Hva er argumentene for å ikke bytte dør? (Gitt at du ønsker å vinne bilen.) (Besvar denne oppgaven før dere har gjort c).)
(c) La en på gruppen være Monty Hall og en annen konkurransedeltakeren. Monty Hall setter tre kopper opp ned på bordet og legger en tegning av en bil under en av dem og en tegning av en geit under de to andre uten at konkurransedeltakeren ser på. Så velger konkurransedeltakeren en kopp og Monty Hall tar opp en av de andre koppene og viser at det under denne var en geit. Konkurransedeltakeren bytter dør, og dere noterer om han/hun vant bilen eller ikke. Gjør dette forsøket 25 ganger. Gjør etterpå 25 forsøk der

konkurransedeltakeren ikke bytter dør. Hva kan dere si om sannsynligheten for å vinne bilen i de to tilfellene «bytte dør» og «ikke bytte dør»?

(d) Besvar b) på nytt.